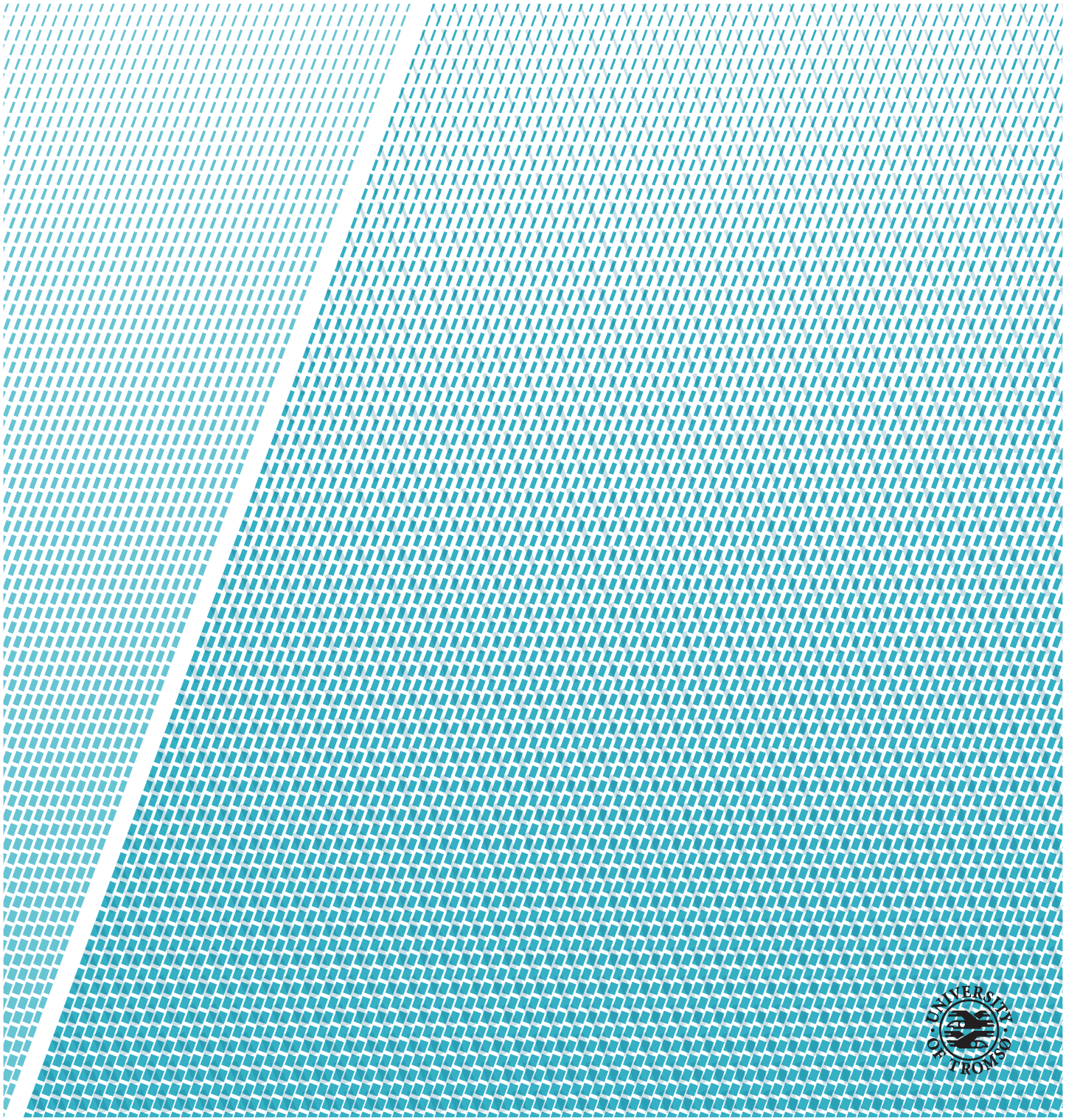
UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Computer Science

Metrix: Real-time Analysis of Physical Performance Parameters in Elite Soccer

Kim Hartvedt Andreassen

INF-3981 Master's Thesis in Computer Science, June 2018



"I have not failed. I've just found 10,000 ways that won't work."
–Thomas A. Edison

"I'll create a GUI interface using Visual Basic to see if I can track an IP address."
–CSI: New York

Abstract

In recent years, technology has had a vast impact on the sports industry, particularly in soccer. Elite soccer teams utilize digital information systems to quantify performance metrics, in order to assess their strengths and weaknesses. Applied methods mostly rely on post-game analytics, allowing coaches to review games in retrospect and implement corrections to their team thereafter. However, this method is inadequate in its ability to provide immediate feedback for coach intervention *during* match or training.

This thesis presents Metrix, a computerized toolkit for coaches to monitor their players *during* match or practice. The system performs real-time analysis on captured sensor data, used to quantify specific movement patterns of players during games. Performance parameters are instantly available to coaches through the Metrix client, accessible on the field through their mobile devices. Metrix provides coaches with a toolkit to individualize training load to different playing positions on the field, or to the player himself.

Metrix is developed for, and in close collaboration with, the coaches in the elite soccer club Tromsø Idrettslag (TIL). The functionalities our system provides are implemented based on their specified requirements, and further customized to their needs.

This thesis describe the requirement, design and implementation of Metrix. We evaluate the performance of our system, as well as its usefulness to our end-users. Our results show that Metrix is able to quantify player performance parameters and propagate it to coaches, in real-time, during match or practice. The coaches express that this is a valuable asset in day-to-day work.

Acknowledgements

First, and foremost, I would like to thank my supervisor Professor Dag Johansen. Your advice, guidance and continuous motivation throughout this year is invaluable. Your passion is truly inspiring!

Further, I wish to thank my co-advisor Professor Pål Halvorsen for his assistance and general guidance regarding ForzaSys technology.

A big thanks to Ivan Baptista and Svein-Arne Pettersen for a great collaboration throughout this thesis. Your help with everything regarding TIL, sports science and general soccer related research have been very helpful to me during my work.

To the people of the Corpore Sano lab: Thank you for your input and advice, and for providing an excellent academic environment to work in.

I would also like to thank my fellow classmates, especially Helge Hoff, Jon Foss Mikalsen and Christoffer Hansen. Thank you for your general presence, and for using your precious time to help me, even though you were busy with your own thesis. A special thanks goes to Marius Andreassen, for your friendship and collaboration through the first three years of this degree. I would not have gotten this far without you!

And finally, I would like to thank to my girlfriend Martine Guttormsen for her endless love, support and motivation throughout my studies.

Contents

List of Figures

List of Code Listings

List of Abbreviations

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CET Central European Time

CQL Cypher Query Language

CRUD Create, Read, Update, Delete

GDPR General Data Protection Regulations

GUI Graphical User Interface

HCI Human-Computer Interaction

HIR High Intensity Run

HLS HTTP Live Streaming

HTML5 Version 5 of the HyperText Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

MVC Model-view-controller

NFF Norwegian Football Association

OS Operating System

PTW Pre-Training Wellness

RPE Rating of Perceived Exertion

SGX Software Guard Extensions

TCP Transmission Control Protocol

TIL Tromsø Idrettslag

UiT University of Tromsø

URL Uniform Resource Locator

UX User Experience

VM Virtual Machine

1

Introduction

Advancements in digital technology are changing the face of the sports industry. Digital information systems are widely adopted in the field of sports science, being used to solve sport-related problems in both research and applied aspects, effectively bridging the gap between theory and practice.

Elite sports teams are constantly in search for comparative differentiators; new techniques, strategies or methods that give them a competitive advantage to their opponents. Popular sports such as football, baseball and basketball are examples of team sports driven by innovation and change, leveraging technology for pinpointing the recipe for success. Soccer, the most popular sport in the world, is no different. Research shows an increased demand of elite soccer players physical performance in the last decade [1], thus indicating that the bar is continuously being raised as the sport evolves. Through technological advances in the use of quantified data and associated analytics, teams obtain valuable insight into performance metrics, serving as a foundation for evidence-based decisions regarding team improvements. The volume and immediate availability of such data allows coaches and sports scientists to make more informed decisions about current and future needs, thus increasing the teams' potential to perform.

Due to the non-linear flow of a soccer match, automated analysis is inherently difficult. There are just too many parameters to consider, including anything between pre-game nutrition to post-game recovery. A most common approach is player-centric analysis, where teams collect large volumes of performance

metrics regarding each individual player. Larger sports organizations are known to, in worst case, employ one analyst per player on the field, in order to collect and study individual performances to provide personal feedback. This is not an option for smaller teams, due to insufficient resources or funds. Teams create extensive profiles on their players, holding information the coaches deem relevant for maintaining and increasing player performance. As data volume and complexity grow, efficient tools for automated high-precision retrieval become essential.

Data quantification methods mostly relies on post-game analytics, using automated or semi-automated tools to study performance metrics. This is often achieved through video based analysis tools or data captured by sensor devices [2]. Posterior evaluation is great for hindsight notation, allowing coaches to apply corrections thereafter. Its weakness, however, is the lack of immediate feedback *during* match or practice, in situations that might require swift action from the coaches.

This thesis presents Metrix, a system providing live monitoring of player performance metrics on the soccer field. Parameters considered imperative by coaches is captured by our system, and immediately made accessible through mobile devices or laptops operated on the field during match or practice.

1.1 Problem Definition

Currently applied sports analysis tools are largely focused on providing coaches with player performance statistics summaries, reviewed in retrospect after match or trainings. We wish to investigate a solution for providing instant feedback on physical exercises as they unfold, allowing coaches to continuously follow up on their players during physical activity. Hence, we formalize the following definition:

"*This thesis shall design, implement and evaluate a system supporting live monitoring of player performance metrics in soccer, facilitating real-time analysis during match or practice. Our goal is to provide coaches with a computerized toolkit to quantify specific movement patterns of players, in relation to individual training goals and physical demands of different playing positions on the field.*"

1.2 Scope and Limitations

This thesis shall specify requirements, design, implement and evaluate a prototype application for performing real-time data analysis of elite soccer players in Tromsø Idrettslag (TIL). The system shall capture and analyze physical performance parameters deemed as highly relevant by the coaching staff.

The system shall provide a vertical software supporting lower-level sensor processing and higher-level data visualization. This involves procedures that enable data transformation and processing, communication primitives for handling data flow, and visualization of the analysis results.

Our main focus is to provide a robust processing backend, featuring both efficient and accurate detection of on-field physical events. Secondly, our focus will concern presentation of the data output with regards to User Experience (UX). While this might be considered more important to our users, the provision of exotic user interfaces will not be our main concern.

In addition, we will investigate possibilities for providing real-time video playback of the captured field-events. This involves integration with currently deployed video systems at Alfheim, the home arena of TIL's.

We require our system to support a small amount of users, namely the coaches in TIL. However, we will still investigate how our system scales with regards to a larger amount of users and players on the field.

The system will not implement a strong analysis tool for historical data, such as graphs and reports of long-term performance parameters. The focus remains on short-term analysis and tracking of *current* physical performance.

While security and data privacy are important subjects when working with player health data, we will not address these issues in this thesis. We defer these problems to future work.

1.3 Methodology

In their final report [3], the ACM Task Force presents a scientific framework for describing the discipline of computer science and computer engineering. The report defines the discipline through three major paradigms, forming the basis of scientific work within computing:

Theory is rooted in mathematics, where a valid theory is described through

defining the objects, hypothesize possible relationships among them, proving said relationships, and finally interpreting the results.

Abstraction is rooted in the experimental scientific method, focusing on the investigation of phenomenons. Its method involves designing experiments from collected data and performing analysis, based on an initial hypotheses.

Design is rooted in engineering, applicable to the construction of a system or device. The steps involve stating requirements and specifications of the system, designing and implementing it, and finally testing it.

This thesis is rooted in the area of *systems engineering*, consequently using methods that closely relates to the design paradigm. We will start by specifying our requirements, with regards to both the end-users and to system features. Then, we will design and implement our solution through an iterative process, based on our preset requirements. Finally, through experiments and evaluations, we will demonstrate our system's capabilities and usefulness.

1.4 Context

This project is written as a part of the Corpore Sano Centre[1], a centre for sport and health technology at University of Tromsø (UiT). The centre focuses on innovations within computer science, sport science and medicine, namely holistic systems and solutions targeting topics like security, fault-tolerance and privacy in a human-centric context.

Corpore Sano has a vertical focus, ranging from low-level infrastructure software such as Operating Systems (OSs) and Virtual Machines (VMs), to higher level cloud services and video streaming. Our work with OS architectures include Vortex [4], which instantiates the omni-kernel architecture, providing fine-grained scheduler control of resource allocation.

Further, we have done extensive work within the area of distributed data processing [5, 6] and mobile agents [7, 8]. Early work includes the TACOMA project [9], which concerns itself with implementing OS support for *agents*, processes that migrate through a network. More recent work include Cogset [10, 11], providing an efficient and generic engine for reliable storage and parallel processing of data, enabled by the MapReduce-model.

1. http://www.corporesano.no

In the area of security and fault-tolerance, our work includes Fireflies [12], an intrusion tolerant overlay network protocol, providing a secure and scalable gossip and membership service. From this, we built Firepatch [13], a secure software dissemination system which limits adversaries attack window while issuing security patches. Our work in data security [14] includes research addressing the performance characteristics of trusted computing in cloud infrastructures, using Intel's new Software Guard Extensions (SGX) platform.

Research more related to this thesis is Davvi [15], which presents a novel end-to-end prototype of an Internet based video system in the sports domain. Davvi supports keyword-based search on large video archives, allowing users to find and select specific soccer events, and further combine them into a single logically composed video. The collection and aggregation of video content effectively implements personalized video streams, played back to the user in soft real-time.

Further work in the soccer domain includes Muithu [16, 17], a non-invasive notational analysis system, and Bagadus [18, 19, 20], a real-time sports analysis system. We talk more about these systems in Section 2.3.

The privacy of the soccer players are preserved through our work with Code capabilities [21], a mechanism for embedding executable code fragments in cryptographically protected capabilities, implemented in user-space. This enables flexible discretionary access control in cloud-like computing infrastructures.

Other partners in this thesis involve ForzaSys[2], a company spawned from the Simula Research Laboratory[3] in collaboration with UiT. ForzaSys (through Simula) have a wide competence with large-scale multimedia systems, now focusing on delivering next generation multimedia platforms to enhance the relationship between sport clubs and their followers.

1.5 Outline

The remainder of this thesis is structured as follows.

Chapter 2 contains relevant background information regarding technologies used in soccer, and how it endorse methods for quantifying the sport. Additionally, we explain some of the existing technologies in use by TIL.

2. http://www.forzasys.com
3. http://www.simula.no

Chapter 3 outlines the coaches motivation behind the implementation of Metrix, and their requirements to application features. Based on their input, we define a set of functional and non-functional properties required by our system.

Chapter 4 describes the overall architecture of our system, outlining the major components and how they relate to each other.

Chapter 5 describes the design and implementation of Metrix, giving details of all components in the system. We discuss different approaches to our design and the reasoning behind our choices.

Chapter 6 evaluates the applicability and performance of Metrix through experiments and a user survey.

Chapter 7 concludes our thesis and outlines future work.

2

Background

2.1 Quantified Soccer

Match analysis in soccer generally refers to the objective measurements and analysis of discrete events during training or competition [22]. It can be divided into three separate subcomponents; technical, tactical and physical. The technical components involve quality of skills executed by players in a match. The tactical component refers to overall strategy and style of play. The physical component incorporates measured movement patterns and efforts made by players on the field. A match analysis may range from individual focus regarding specific players, to composite analysis of the interplay between all the players in the team.

2.1.1 Methods

Motion analysis is the most common approach for player analysis, focusing on the measurement of individual activity and movement patterns during a session. Typical parameters include total distance covered, number of turns, and number of efforts performed in varying movement categories (i.e. jogging, running, sprinting) [23, 24]. This information is used to develop extensive player activity profiles [25], outlining average physical demands of each player and their playing position on the field. The activity profiles aid coaches, sport scientists and other practitioners to monitor changes to physical performance parameters over time, enabling them to quantify player training load, or compare players

with similar attributes (i.e. teammate or opponent).

Structured match analysis dates back to the 1970's [26], where coaches used *notational analysis* to capture field events. This involved observers on the sideline documenting player or team activity on the field using pen and paper. Not only does this involve tedious manual labor, but requires extensive human resources. The method is also limited in its ability to provide data regarding velocity or speed metrics.

An improvement to the classic notational analysis is *video-based time-motion* analysis, involving players to be filmed during match or training [27]. Footage is analyzed post-game, allowing observers to pause, review and slow down the videos for a closer look. We typically separate between two classes of video analysis; individual and collective. With individual recordings, one camera follows a single player. Its disadvantages include the requirement of a camera operator, as well as missing contextual information with regards to the team as a whole. The collective approach solves this by using one or more cameras providing an overview of the field, able to capture videos of all the players at once. The downside with this is the difficulty of accurately measuring individual movement patterns.

With the advancements of digital technology, more semi-automated systems have replaced the manual approach of collecting player data. The most renown system is ProZone [28] (now called STATS), who in the early 2000's introduced a semi-automated video tracking solution using multiple cameras placed in fixed positions at the stadium, covering the entire field. Through video image analysis, player trajectory is determined through x and y coordinates, measured in meters from the center circle on the pitch at a sampling rate of 10 Hz [29]. This allows for calculation of distance covered and average speed between samples, effectively quantifying players speed and distance metrics. Even today, semi-automated video tracking is still a popular approach for assessing player movement patterns. However, the systems are limited by their requirement of an observer for cleaning, structuring and verifying the captured data, which is a time-consuming process. Consumers of such analysis must often expect a 24 hour delay before it is available to them. Another downside is the lack of portable system components, limiting game analysis to specific training facilities that employs the static camera setup.

Another common approach for quantifying player movement patterns is the use of *radio-based sensor systems* [30]. Here, each player wear a data chip with sensors during physical activity. The sensors send signals to radio receivers, typically mounted on antennas located around the perimeter of the pitch. Using Radio Frequency-based technology, observers can track players with high precision. Some systems also support integration with supplementary

equipment, such as heart-rate monitors or accelerometers, providing additional data of player performance metrics. The most renown system to employ radio-based systems is ZXY Sports Tracking [31], which we talk about in detail in Section 2.2. Similar to that of semi-automated video systems, most radio-based systems are deployed as fixed installations, which limits their usage to specific training facilities. Though its downsides include complexity of deployment, radio-based systems are considered accurate and non-invasive to the players.

In later years, commercially available GPS units designed for sports tracking have become increasingly popular for quantifying player performance metrics [32, 33]. The most renown systems using this technology includes ZXY Sports Tracking [31], GPSports [34], CatapultSports [35] and StatSports [36]. With advancements in GPS technology, the sensor components have decreased dramatically in size, now considered non-invasive for players to wear underneath their clothing during physical activity. With high-frequency sampling rates, sensor units capture physical performance parameters at high accuracy, providing information such as speed, acceleration and distance covered, as well as positional data to track players on the field. Opposite to the more semi-automated solutions, GPS units are portable, allowing them to be used at both home and away matches. Sensor data is automatically captured and stored, allowing for automated analysis software to further process its content through data aggregation and filtering.

Through notable advancements in technology, the evolution of motion analysis has progressed from data initially collected with pen and paper to fully automated computer systems capable of capturing fine-grained movement patterns on the field. These advancements have increased efficiency, reliability and accuracy of quantifying soccer players during match or practice.

2.1.2 Physical Demands

While technology accommodates different means of acquiring accurate measurements, it is up to the sports scientists to decide which parameters they deem most relevant for evaluating player performance. Through match analysis, physical performance metrics are quantified in order to determine the load players experience in a competitive setting.

The movement demands of players have been much researched over the years [37, 38, 39], using a variety of match analysis methods. In general, the total distance covered by professional male soccer players is approximately 11 000 meters, ranging between 9000-12 000 meters. Typically, 75 to 85% of the total distance is covered at speeds classified as low intensity, such as walking

and jogging. This leads researchers to conclude that movement demands in soccer are generally aerobic in nature. However, the remaining movements are classified as anaerobic, consisting of high-intensity efforts such as accelerations, decelerations, sprints, etc. These high-intensity movements are, for many, considered critical for the outcome of success, for both team and individual player performance [40, 41, 42]. For this reason, lots of research have focused on quantifying these movements, in order to establish physical demands of their players. Through such measurements, coaches may adjust and plan their trainings accordingly, to best simulate a competitive setting.

Further, research show a statistical difference in physical demands between different playing positions on the field [43, 40]. This information gives coaches an incentive to *individualize* training programs to the demands required by the specific roles on the field. Hence, coaches require a method for monitoring individual players in real-time *during* practice, allowing them to do personalized intervention to each player's training load.

2.2 ZXY Sports Tracking

The ZXY Sports Tracking system [31] is a product of ChyronHego, providing advanced technology solutions for sports teams to monitor their athletes during sports events. Their product targets the soccer community in particular, providing an assortment of commodities, like tracking players and balls on the field, monitor physical performance of players, and providing statistical data for post-session analytics.

ZXY Arena uses Radio Frequency-based technology and is designed for fixed installations, such as dedicated training facilities or competition arenas. This system is currently deployed at Alfheim stadium, and consists of eleven stationary radio receivers mounted around the stadium area, as depicted in Figure 2.1. Each receiver has approximately 90 degrees field-of-view, forming overlapping zones of the soccer field to provide high immunity to signal blocking and occlusion.

Figure 2.1: ZXY radio receiver mounted on antennas around Alfheim stadium

The current installation is based on the 2.45 GHz ISM band for radio communications and signal transmissions. Each radio receiver computes the position data based on the radio signal from the sensor belts worn by the players on the field. The belts are issued with accelerometers, a gyro, a heart-rate monitor and a compass. Combined, they provide valuable performance metrics and positional data of active players on the field.

ZXY provides high precision tracking data with sampling rates of up to 100 Hz. At Alfheim, the sampling rate is currently set to 20 Hz, transmitting data records in real-time to a relational database for storage.

2.3 Bagadus

Bagadus [18, 19, 20] is a real-time sports analysis system deployed and in use by TIL at Alfheim stadium. The system supports automatic processing and retrieval of events in the sports arena, with their novel integration of three sub-systems; a video system, a sensor system (ZXY) and a soccer analytics annotations system.

The video sub-system consists of a camera array which covers the entire field. Videos are further processed by stitching the camera output together in real-time, encoding it and persisting it to storage.

Using ZXY technology, the sensor system collects data from sensor belts used by players on the field. The sensor data enables Bagadus to perform statistics on players, like total length run, number of sprints, foot frequency, etc., in addition to tracking players on the field.

Figure 2.2: Illustration of the Bagadus system. The camera array captures video from the field. Video events can be extracted by performing queries on the ZXY database containing captured sensor data.

The annotation sub-system is a subset of Muithu [16, 17], a light-weight, non-invasive notational analysis system deployed at Alfheim stadium some years ago. Muithu is based on the concept of *hindsight recording*. Multiple cameras are installed on the field, recording full matches or practices. Whenever the coach have witnessed an event worth capturing, he uses the phone to mark *the end* of that particular event. The system will later use this marking to find the beginning of the event, typically configured to 15 seconds prior to the end notation (but may be adjusted). Finally, all tagged events are extracted from the raw video footage and persisted to a database.

Bagadus uses Muithu technology for event-based tagging. The video data is, however, captured from Bagadus' own cameras and processed entirely by their own video sub-system.

Bagadus presents a holistic platform where videos are captured and processed, then further coupled with sensor data of the involving athletes. Annotated video sequences are automatically pushed to cloud storage within seconds and made publicly available online[1] or privately to the annotator.

1. http://til.forzasys.com

2.4 Summary

In this chapter, we have described several different methods for quantifying physical performance parameters in soccer, illustrating their strengths and weaknesses with regards to accuracy and efficiency. We also explained how the physical demand of players is often measured by quantifying their high-intensity efforts on the field. Further, we have given an overview of the ZXY Sensor System and Bagadus, technologies currently in use by TIL today.

3

Requirement Specification

This chapter describes the motivation behind the development of Metrix and establishes a series of requirements, both system-wise and with regards to the coaches in TIL, the end users. Section 3.1 explains the coaches' motives for the implementation of Metrix, influenced by their latest research and observations in the field. Further, Section 3.2 encapsulates a set of application requirements set by the coaching staff, followed by a detailed explanation of the physical parameters they wish to monitor. Lastly, Section 3.3 defines the functional and non-functional requirements of Metrix, based on the established specifications set by the coaches.

3.1 TIL: A Casy Study

Metrix is built in close collaboration with the intended users, namely the coaching staff in TIL. The application has gone through several stages of implementation, where different solutions have been reviewed and discussed in an open dialogue throughout the process. The most frequent correspondence have been with Ivan Andre Matias Do Vale Baptista, the assistant coach responsible for player development, and effectively appointed as TIL's "numbers guy". Between matches and training, Baptista spends numerous hours breaking down team statistics regarding anything from performance metrics to player wellness, looking for correlating patterns between the different classes of data. Through analytics, Baptista aims to improve team performance by identifying

both short-term goals concerning upcoming matches, and long-term goals with regards to player development.

Through his work and studies, Baptista discovered significant differences in physical parameters across playing positions during match play [44, 45, 46]. In the study, 18 players from TIL were tracked in 23 home matches using the ZXY sensor system. The captured parameters illustrated that, for example, the wide mid-fielders had higher accounts of accelerations than others, and that full-backs performed more accounts of long distance HIR's than central backs. These observations, among many others, provide the foundation for developing a coach toolkit to customize individual training load to playing positions.

TIL's collection of sensor data spans several seasons, consequently reflecting evolvement or changes in the team. The most apparent one is perhaps the appointment of a new head coach, which subsequently induced a new style of play using different tactics and formations. This further affects the position specific demands, where the players' role on the field might require a more passive or aggressive oriented play-style, requiring a different set of qualities than before. There is a consensus in sports sciences that the most effective training for preparing athletes for competition is that which most closely replicates competitive performance conditions. Hence, the coaches are responsible for implementing training program cycles that are most relevant to match-play. With the observed differences of physical requirements across playing positions, Baptista underlines the importance of individualizing these training programs. The first step is role-specific individualization, but the ultimate goal is to be able to further adjust trainings to the player himself, recognizing his physical limitations and form.

Naturally, trainings are scheduled intermediate of official matches. For TIL, this involves one day of restitution, followed by daily sessions of physical workout. Trainings are carefully planned and executed with regards to physical load and intensity. It is the coaches responsibility to find the balance between obtaining the desired training goals, and wearing out the players before an official match. Coaches have expressed the need for a computerized toolkit to define periodic training goals for individual players, and further monitor their progression throughout multiple sessions. E.g. a coach may require the central mid-fielders to achieve 70% of match-load over a period of four days. By quantifying specific load-intensive performance metrics, coaches can better monitor their players on a granular level. Players who are pushing themselves too close to the limit can be rested from specific drills, while those who underperform can receive additional physical load.

3.2 User Specification

Through continuous discussions and consultations with the coaches in TIL, we have pinpointed a set of functionalities Metrix should provide. With their expert knowledge within the field of sports, we regard their specified requirements as most important when designing Metrix. Following is a list of user requirements, interpreted from coach input:

Progress The application should provide a method for displaying progression of physical activity performed during training weeks. The progress should be relevant to an input training goal for each individual, set by coaches beforehand. Progress should be measured over a finite timespan, namely the periods intermediate of, and including official matches.

Physical load The weekly training load (or goal) for an individual player should be based on the player's all-time best performance in official match play. For each measured physical parameter, coaches should be able to specify a percentage of the best performance value, effectively setting the player's specific goal for the week. For example, if a player's highest account of sprints in an official match is 50, and coaches expects him to perform 50% of that during the week, his goal will be to achieve at least 25 sprints. The initial best-performance values are gathered from historical match data, provided by ZXY.

Simplicity The user interface should provide an intuitive visualization of player progress, and should be easy to use for non-technical personnel. The application should not be bloated with extensive statistical data, but only contain simple illustrations of captured data.

Event Captured physical parameters should adhere to a set of predetermined conditions. These are further explained in detail in the following subsection.

Instant The captured physical parameters from the the field should be instantly propagated to coach-operated devices, enabling them to monitor their players in real-time.

Accessible Coaches must be able to access the application from the field. This implies that Metrix should be available for use on mobile devices, such as pads or cellular phones.

3.2.1 Event Type Definition

There are three different physical parameters TIL's coaches wish to monitor with Metrix; HIR, Sprints and Accelerations. For the remainder of this thesis we will refer to these as *events*. For each event type we classify two sub-categories; the number of completed events and distance covered during them. This makes a total of six physical parameters observable by coaches during match or practice.

The next two subsections will explain more detailed definitions of each event type.

Runs

Run-events can be either HIRs or Sprints. Figure 3.1 show an example graph of a typical run-event captured by ZXY data. A run is defined by six event markers, described as follows:

A: The speed increases above the run speed limit of 4.0 m/s. This marks the start of the run event. Distance metrics is being calculated from this point.

B: The speed increases above the HIR speed limit of 5.5 m/s. This is required for the run to be counted as a valid run event. Also, the player must be in this zone for at least 1 second ($E - B > 1$).

C: The speed increases above the sprint speed limit of 7.0 m/s. Crossing this threshold conforms the run event into a valid Sprint event.

D: The speed decreases below the sprint speed limit.

E: The speed decreases below the HIR speed limit.

F: The speed decreases below the run-speed limit. This marks the end of the run event.

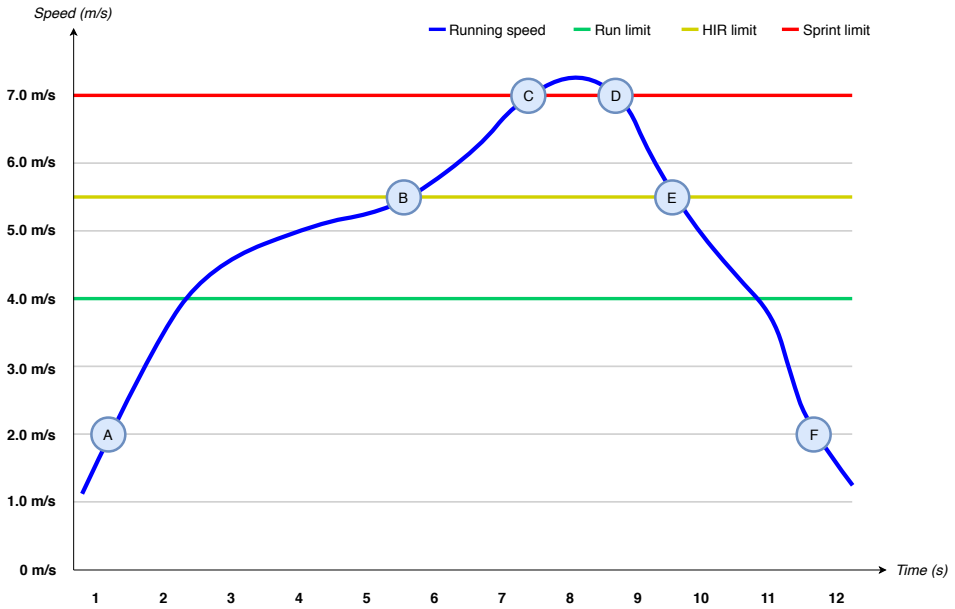


Figure 3.1: Speed-graph illustrating the definition of HIR and Sprint events.

During event processing, timestamps from event markers A, B, E and F are captured. The time from A to F defines the duration of the event, while the time from B to E asserts a valid run.

The following data is captured and stored at event completion:

- Timestamp of event start (A)
- Timestamp of event end (F)
- Duration of the event (A to F)
- Distance covered during event (A to F)
- Maximum achieved speed during event

Accelerations

An acceleration event is similar to the run event, but is derived from different sensor parameters. The event is defined by the following four event markers:

A: The acceleration reaches the minimum acceleration limit of 1.0 m/s^2. This marks the start of the acceleration event.

B: The acceleration reaches the acceleration limit of 2.0 m/s². The value must exceed this limit to be counted as a valid acceleration event.

C: Acceleration value decreases below the acceleration limit. It is required to remain above the acceleration limit for at least half a second ($C-B > 0.5$).

D: The acceleration value falls below the minimum acceleration limit. This marks the end of the event.

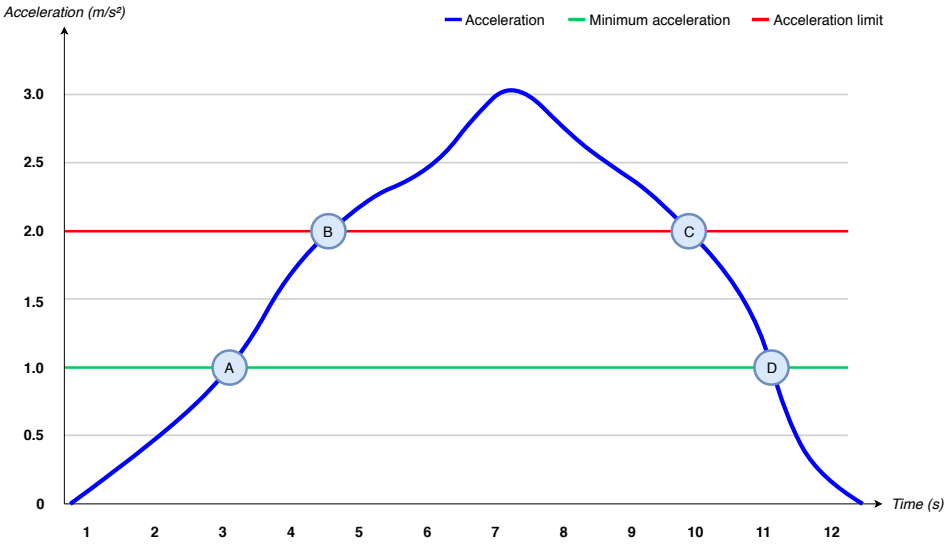


Figure 3.2: Acceleration-graph illustrating the definition of an Acceleration event.

The following data is stored in the ZXY Session after a completed acceleration event:

- Timestamp of event start (A)
- Timestamp of event end (D)
- Duration of the event (A to D)
- Distance covered during event (A to D)
- Initial speed on acceleration event start (A)
- Speed on acceleration event end (D)

3.3 System Specification

From the established user specifications we derive a set of requirements to Metrix, in order to support the functionalities imperative to TIL. We divide the system specification into two categories; functional and non-functional requirements.

3.3.1 Functional Requirements

The following list describes the functional requirements of our design, regarding behavior on input and output data.

Live view Player performance metrics should be processed and displayed in real-time to coaches on the field, through a visual interface accessible by mobile devices.

Historical view Coaches should be able to examine live player data in context of previous trainings. An overview of physical parameters from previously executed trainings must be accessible, in order to give further detail to the current progression.

Training goals The system should support external input for administering periodic training goals for each individual player. In effect, this should provide short-term goals the player should achieve in a variable amount of days. The goals should be visible in the live view, defining a target for players to progress towards.

Video playback The application should provide video playback of captured field-events. Videos should be accessible in soft real-time, provided at coaches request.

Persist data Performance metrics captured during a session should be stored in a database for subsequent aggregation. This is vital for tracking physical performance data that spans multiple training sessions.

3.3.2 Non-functional Requirements

Non-functional requirements specifies a set of qualities demanded by our system, relative to how it is perceived by the users.

Applicability The system should be applicable for the coaching staff to use on a daily basis, during trainings and matches.

User-friendly The application should be intuitive and easy to use for the coaching staff. The design should be simplistic, displaying only relevant information to coaches.

Scalability The system should be able to scale relative to a finite amount of players on the field, and a limited amount of coaches monitoring them.

Real-time The output data from Metrix should keep up with the flow of input ZXY sensor data, providing analytics in real-time. This means optimizing the data processing and communication, so that field-events may be propagated to the view instantly. The application output must not fall farther and farther behind the input sensor data stream as the session progresses.

3.4 Summary

This chapter has described the coaches motivation behind the implementation of Metrix. We outline their requirements regarding features that Metrix should support, which we further use to establish our system's functional and non-functional requirements.

4

Architecture

This chapter outlines the overall architecture of Metrix. We define a three-tiered architecture, composed of a Backend, a Frontend and Clients. Using a three-tiered architecture provides a conceptual segregation between the services our system must support, as well as a logical seperation between their subcomponents. Implementing each service as independent modules makes the application easier to develop, extend and maintain, as changes to one tier does not have to affect the other.

The following sections will give a general description of each tier in the Metrix architecture, outlining their major components and how they interact with each other.

Figure 4.1 illustrates an overview of the architectural composition. Data flows through the system in a bottom-up fashion. Raw sensor data (1) is received and processed by the Backend (2), who pushes aggregated data to the Frontend (3), further responsible for updating the Clients (4). We use this figure as a visual reference point for the following sections, which gives further details to each tier.

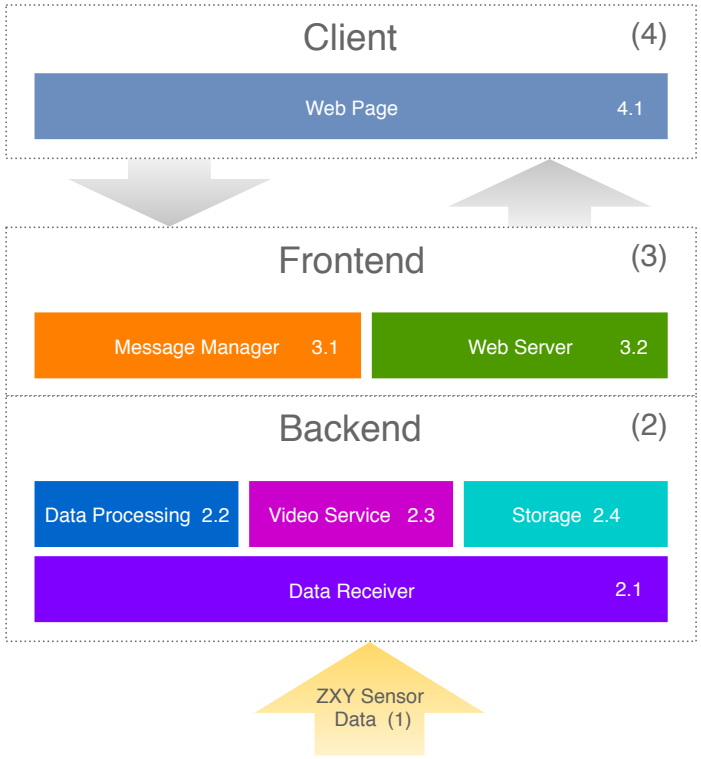


Figure 4.1: Metrix architecture

4.1 ZXY Data Input

The ZXY sensor data input (1) consists of measured parameters regarding player activity on the soccer field. Prior to a match or practice, coaches distribute sensor belts among the players and activates them through a designated ZXY subsystem. The output data records contain measurements from exactly one ZXY sensor belt. Belts are uniquely identified by a tag id, and each player wears exactly one belt.

The current ZXY Sensor System setup at Alfheim uses a sampling rate of 20 Hz. In effect, this means that Metrix receives 20 data records per player, per second. For a 90-minute soccer match with eleven players on the field, this equals to 220 data records per second, a total of roughly 1.2 million records for the entire match.

A data record is comprised of an array of sixteen unique data fields, measured by the sensor technology. The fields include positioning, direction, speed, etc. In our system, we need only concern ourselves with a subset of the data. Only

those values relevant for measuring the events described in Subsection 3.2.1 are required by our system. The relevant values from the sensor data are defined as follows:

tag_id (int) - ID of the ZXY sensor belt.

timestamp (string) - Local unix Central European Time (CET) encoded as ISO-8601 format.

speed (float) - Current speed of the player. Measured in meters per second (m/s).

acceleration (float) - Current acceleration of the player. Measured in meters per second per second (m/s^2)

total_distance (float) - Cumulative distance the player have traveled so far.

During the development of Metrix we have used a substitute sensor data input source. We have created a simple server that outputs real ZXY sensor data, acquired from a published dataset [47] that contains sensor data captured from official matches in 2013. In production, this component will be substituted by the real ZXY output stream, with little to no effort.

4.2 Backend

The system backend constitutes the core service of our application. The Backend (2) is comprised of a series of data resources, processing units and a single storage device. We divide the backend into four main components; a *data receiver* (2.1), a *processing service* (2.2), a *video service* (2.3) and a *storage service* (2.4). Together, these components constitute a *ZXY Session*, initiated by coaches during soccer match or practice.

The data receiver (2.1) is the initial entry point for the incoming ZXY data stream. It is responsible for parsing sequentially received data records, and transform its content into logical application structures.

The processing service (2.2) performs live, per-record, data analysis in real-time. Transformed data records are distributed amongst a series of processing units, whose main functionality is to detect on-field events executed by players during match or practice. Observed events are pushed to the Frontend message manager (3.1), who subsequently updates clients with the new data.

The video service (2.3) provides instant video playback of registered field events. Clients may request videos of individual events for visual feedback of player performance. The service is responsible for seeking through raw video recordings from Alfheim, assemble the specified content, transcode the video data into a HTML5-compatible format, and finally produce a coherent video sequence of the specific event.

Players' performance metrics are aggregated through captured events and persisted to a database maintained by the storage service (2.4). The stored data is subsequently used for accumulating weekly performance statistics for individual players.

4.3 Frontend

The Frontend (3) layer sits between the Client application and the Backend services. It implements communication primitives for distributing captured field data from the underlying ZXY Session to connected clients.

We divide the Frontend into two components; a *message manager* (3.1) that handles data transmission and communication between the Backend and the Clients, and a *web server* (3.2) implementing a framework for displaying the data in a web page.

The message manager (3.1) supervises clients that subscribe to an on-going ZXY Session, and is responsible for serving them live updates from on-field events. When a new event is propagated from the Backend, the message manager serializes its content into pre-defined message structures and forwards it to connected clients. The message service is also responsible for handling client-side requests on further details regarding events or weekly performance metrics.

The web server (3.2) implements web page specific logic and functionality. It is responsible for processing user input and serve static page content accordingly. Further, the web server handles user sessions and authentication, limiting the application to coaches only.

4.4 Client

The Client (4) is a web application providing a Graphical User Interface (GUI) for users to interact with Metrix. Users connect to the client through standard

web browsers by visiting a URL to the application web page (4.1). The Client view presents a visual representation of data served by the Frontend, through graphical and textual elements. Users may interact with the system through input from a keyboard, mouse or touch interface.

Users must log in by submitting their user name and password. When authenticated, a coach may navigate the main page content, such as administer training periods, set player-specific training goals and start or stop live ZXY sessions.

The live training session interface relies heavily on client-side scripting for updating the page dynamically when new data is received. The Client does not perform any processing, nor does it cache previous data records. It relies solely on frequently received updates from the Frontend, and response from user-triggered requests.

4.5 Summary

This chapter has described the general architecture of Metrix. We defined a three-tiered architecture, composed of a Backend, Frontend and Client layer. We further explained each tier in closer detail, outlining their subcomponents and how they interact with each other. We also illustrated the data flow in Metrix, giving an overview of how the input ZXY data is processed and further distributed to connected clients.

5

Design and Implementation

In this chapter, we present the design and implementation of Metrix. With our established three-tier architecture, we will now dive deeper into each system component, giving further details of how they are implemented and our reasoning behind our methods.

The following sections will present Metrix in a bottom-up fashion. We illustrate each tier in detail through highlighted figures as we go along. First, we will describe the Backend services, involving data transformation, processing and management. Next, we take a closer look at the Frontend services, responsible for processing user input and distribute data to clients. Finally, we will explain the Client properties and its methods for visualizing live, on-field events to its users.

5.1 Backend

The Backend lies at the core of Metrix, responsible for parsing incoming data and perform analysis on its content. We divide the Backend into several main components that are further comprised of a series of smaller subcomponents, as illustrated in Figure 5.1. Through this section we will explain each of them

in turn, giving reason to our chosen design.

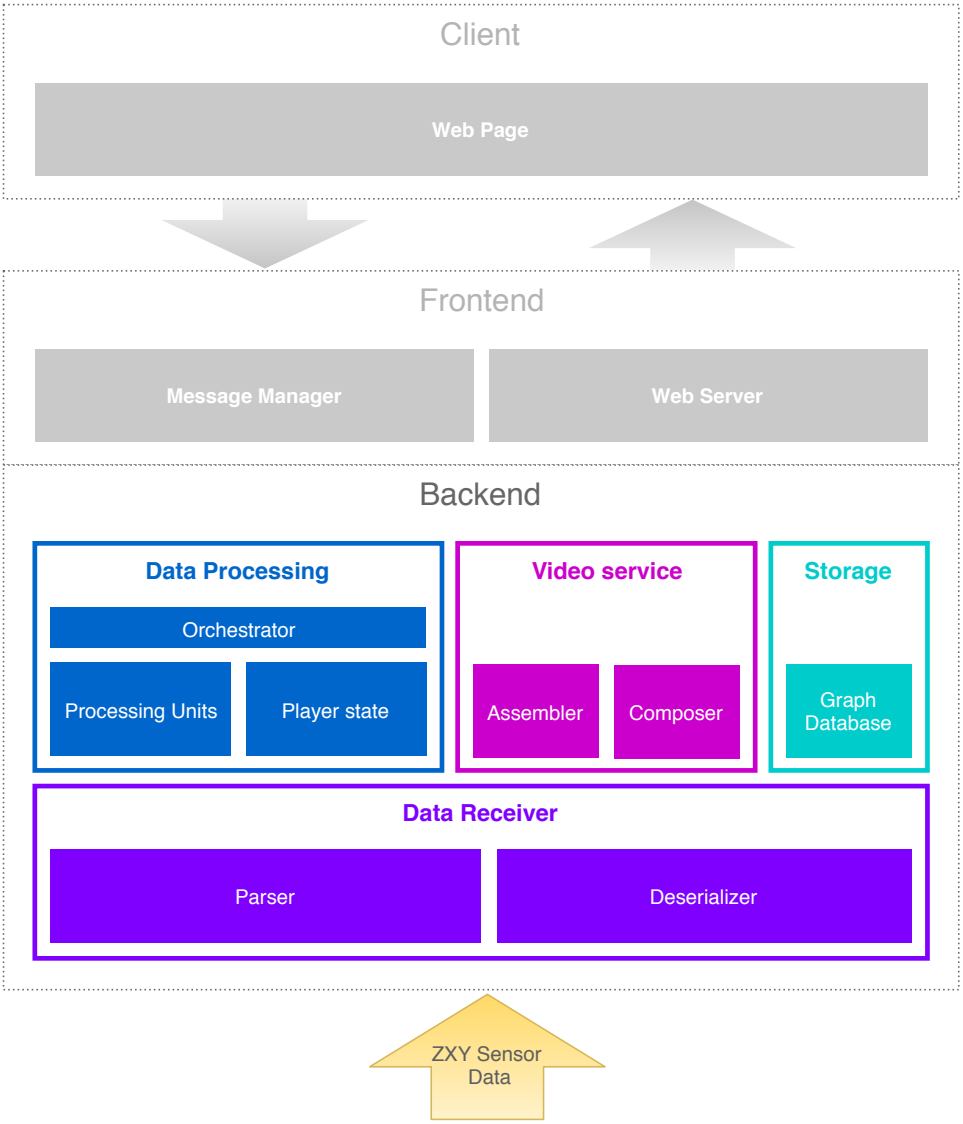


Figure 5.1: Backend components and subcomponents

5.1.1 Data Receiver

The *data receiver* is the entry point for the input ZXY Sensor Data. When a coach starts a new training session, the data receiver will connect to the ZXY Sensor system server and listen for incoming data, received through a TCP connection. Raw sensor data is received as a stream of single data records, formatted as string segments with space-separated sensor data points (as described in

Section 4.1).

The data receiver has two main tasks. First, it parses data records by splitting the string into independent data points. This involves type-casting data point values into their respective formats (i.e. integers, floats, time, etc.). Then, the parsed values are deserialized into internal, session-specific data structures, and indexed for further processing.

Parsing and deserializing timestamps is particularly complex, due to the strict syntax of built-in time conversion libraries. Since timestamps are encoded as ISO-8601 formatted strings, which has microsecond granularity, the parser must handle all the special cases. For example, if the input microsecond unit is represented with five digits instead of six (56384 ms vs. 563840 ms), or when the hour/minutes string 09:34 is parsed as 9:34.

The data receiver component is hand-tailored to the format of the dataset described in Section 4.1. In production, we expect some changes to the incoming data format, which leads to minor changes to the current data parser.

5.1.2 Data Processing

Preprocessed sensor data is further managed by the *data processing* component, responsible for analyzing the data in order to detect on-field events. Similar to the classic Master-Slave model, the data processing component is comprised of a single session *orchestrator* (master) and multiple *processing units* (slaves). From this point forward, we will refer to the processing units as *workers*.

Orchestrator

The orchestrator is a single-threaded routine administering an active ZXY Session. Its responsibilities involve setup and tear-down of the session, including initialization of workers and distribution of data records they need to process.

The orchestrator handles automatic detection and initialization of active players on the field. When a data record with an unidentified belt id is received, the orchestrator performs a lookup in the database, in order to match the belt with the specific player assigned to it. Retrieved player data consists of the following:

Personalia Attributes identifying the player, such as name, player id and his position on the field.

Week summary Aggregated training data from the current training week. These are accumulated at initialization, summarizing the players weekly progression, defining the starting point from which they can progress.

Weekly goals The training goals for the player in the current period, defined and set by the coaching staff through Metrix.

Best performance The player's all-time best performance in official match-play.

Player data is fetched and stored in memory for the duration of the on-going session, indexed through a map. Once a player has been identified, the orchestrator initiates a new worker assigned to the specific player, further elaborated in the following subsection.

When a shutdown signal is received, the session orchestrator will close the connection to the ZXY data stream, shut down the workers and persist the captured performance data to storage.

Worker Pool

The analytical components consist of a series of worker-routines, responsible for processing the data records to detect and analyze on-field events.

We assign one worker-routine per active ZXY belt, meaning each worker handles all data records regarding one particular player. As a result, the number of concurrently processing worker-routines is equal to the amount of active players on the field.

Our design is similar to the classic *producer-consumer* paradigm. Each worker is assigned a message-channel, in which they receive new records to process, as shown in Figure 5.2. The session orchestrator distributes sensor data records to its designated worker by pushing the records into the channel. Job-channels are non-blocking for the producer, so the orchestrator does not have to wait for workers to finish previously distributed tasks. Each worker will continuously fetch new data records from the message channel and process its content. If no new tasks are available, the worker will block and wait for more tasks.

Figure 5.2: Illustration of how the orchestrator distributes ZXY data records to concurrent worker-routines through a message channel

Using per-player worker threads leverage concurrency, facilitating our requirement of serving player feedback in real-time. Assuming the server uses multiple cores, we achieve parallelism on a critical path in the data pipeline, ensuring low processing duration of field events.

Assigning distinct worker routines to specific players in the session provides a logical separation between system components, both conceptually (code-wise) and concretely. Such a design lessens the necessity for frequent use of data synchronization primitives between worker-threads, as each worker only concerns itself with a unique subset of the data. Hence, the design allows us to programmatically avoid data race conditions, as each concurrent worker only modifies values within their own separate part of the address space.

Processing Units

Opposite to post-match analysis on static datasets, detecting player events from a live data stream is more complex. In example, there is no way to measure the duration of a run-event before it is fully completed. There are several approaches to achieve live data analysis, some of which we contemplated at early stages of implementation. One is to store all the received data records in memory and continuously poll the dataset for specific events. The downside with such a design is the excessive use of memory, as well as complexity of searching through the dataset as it grows larger. A second solution, based on the *sliding-window protocol*, is to cache data records for a limited timespan, and search for events within that particular time-frame. Processing will be more efficient as the data subset is small, but more complex to maintain due to adjustments of the window size. If the window is too big we get higher search complexity, if it is too small we are prone to miss critical parts of the event. E.g. if the window is ten seconds, but a player performs a sprint for fifteen seconds, we have already discarded the first five seconds of the event.

Our current solution implements each worker routine as a *finite state machine* [48]. A state machine is described as any device storing the status of something

at a given time. The status changes based on inputs, providing the resulting output for the implemented changes. In Metrix, each worker routine maintains the current state of the player on the field, monitoring changes to his state as it progresses. Figure 5.3 illustrates our implemented state machine.

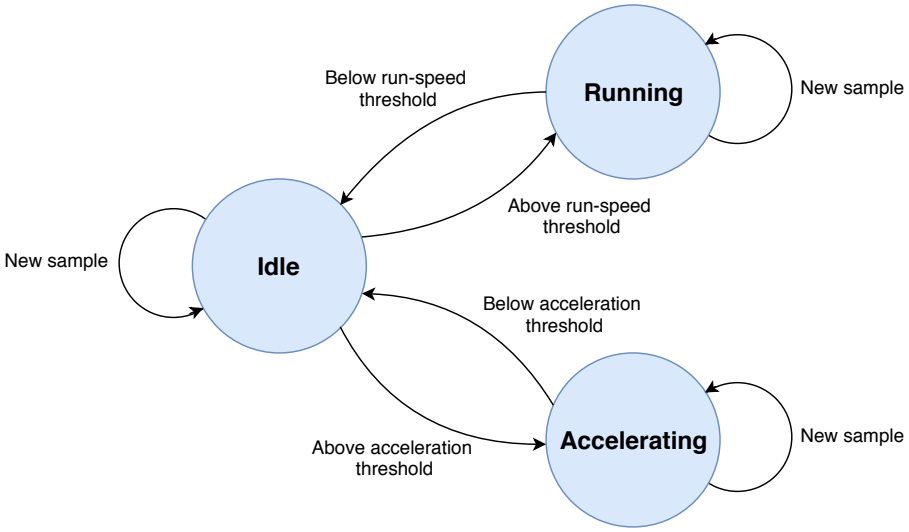


Figure 5.3: Per-player state machine

Each player has three different states; *idle*, *running* and *accelerating*. Idle is the default state, denoting that the player has not exceeded the speed or acceleration thresholds defined in Subsection 3.2.1. The running state and acceleration state are mutually inclusive, meaning the player can be in both states at once, or only one of them. This results in two separate state classifiers, monitoring each state independently during processing.

Code Listing 5.1 shows a pseudo-code of how a single sensor data record is processed by Metrix. When a player exceeds the run-speed threshold, we change the players state to running, and mark the start of the event by storing the timestamp and the current distance covered. When the player speed decreases to below the threshold, it marks the end of the event, computes relevant event-values and pushes the update to the web-client. Other metrics, such as top speed and HIR-duration are continuously monitored from start to end, ensuring events fulfill the requirements specified in Subsection 3.2.1.

Monitoring players in real-time with a state machine is possible due to the high data sampling rate. The close intervals between measurements results in small changes to values from one reading to the next. A significantly lower sampling rate may cause mis-readings, as the implemented algorithms will

Code Listing 5.1: Pseudo-code of how Metrix use states to detect run-events

```
// Player has started running
if player.state != RUNNING && speed >= runThreshold {
    player.state = RUNNING
    event = startNewEvent()

// Player is still running
} else if player.state == RUNNING && record.speed >= hirThreshold {
    // Run qualified as a High Intensity Run
    event.isHir = true

// Player has stopped running
} else if player.state == RUNNING && record.speed < runThreshold {
    // Duration of run not long enough to be valid
    if event.hirDuration < time.Second {
        player.state = IDLE
        return
    }

    storeHirEvent(event)
    sendToClient(event)
    player.state = IDLE
}
```

have problems distinguishing one event from another. E.g. if a player performs subsequent run-events in a short timespan, the system will count it as one long run-event.

5.1.3 Video Service

The video service allows coaches to request video playback of player events during an on-going session. As of today, the video component is conceptual, demonstrating that it is possible for real-time video playback of transpired events during trainings or matches. The service is based on the Bagadus architecture, that records and stores video data on a daily basis. We infer that a closer integration with the Bagadus video system will provide our system with the video backend required to serve videos from on-going sessions. This will involve routing our video handler to a currently non-existing, Bagadus end-point, that can serve video requests based on given timestamps, similar to their methods in [49].

Our implemented video service is based on the dataset, previously mentioned in Section 4.1, where sensor data is coupled with Bagadus video recordings from a soccer match at Alfheim. The video repository setup is similar to the Bagadus system, consisting of 3-second, H.264-encoded video segments, stored locally at the server. Video file names have a monotonically increasing segment number as its prefix, followed by an ISO-8601 formatted timestamp of the recording.

The video service includes two modules; an *assembler* and a *composer*. Clients use a captured field-event's start and stop timestamp to issue a video request to the service. The assembler uses these timestamps to search the video repository for the specific video segments within the given time frame. The received time frame is adjusted by setting an offset of 3 seconds both prior and subsequent to the event. This provides some context to the event in the video, as well as ensuring inclusion of one additional 3-second segment at both start and end of the depicted event. The assembler identifies the matching video segments by the timestamp in their file name, and further collects their name in a text document for further processing by the composer.

The composer use FFmpeg[1] to concatenate and transcode video segments into the HTML5 supported mp4-video format. FFmpeg is a free software library used for handling multimedia data, built on top of the libav[2] video processing toolkit. The text document output by the composer is a required input to the FFmpeg command-line tool, which further concatenates and transcodes the listed segments into a full video sequence. At completion, the composer returns the path to the composed video to the HTTP-handler, which serves the video back to the client.

Concatenating and transcoding video segments on the fly works adequately, presuming field-events are short in duration. However, the service scales poorly with regards to video length, unsuitable for longer video sequences. This results in noticeable latency for end-users, worst case scenario being the HTTP-request times out and does not serve anything at all. A suggested improvement is to use HTTP Live Streaming (HLS), a protocol that complements the segmented video structure by arranging subsequent video clips in a playlist, and further stream them as one continuous video.

5.1.4 Storage

For storing player data we have chosen Neo4j [50] as our underlying database. Neo4j is a state-of-the-art transactional graph database that offers high performance and scalability, while having the same ACID properties as a traditional database. It is structured according to the property graph model: entities (*edges* or *nodes*) hold attributes and represent roles, and those entities are directionally connected by one or several relationships (*vertices*). By leveraging index-free adjacency [51], Neo4j is able to query a subset of the graph data by traversing relationships, opposed to searching through the entire graph dataset.

1. https://www.ffmpeg.org/
2. https://www.libav.org/

We chose Neo4j due to its expressive structure and inherent properties. It provides a so-called "whiteboard-friendly" data model, meaning that the implemented model maps closely to how one would draw it on a whiteboard. In our experience, this simplifies the development process. Also, the Cypher Query Language (CQL) is intuitive and easy to use, making complex queries seem simple. While other database solutions are equally capable of modeling our class of data, they may introduce more complex methods of querying collections of data. For example, relational databases like MySQL would suffer from extensive join tables which would be inconvenient both performance-wise and during implementation.

Data Model

The implemented data models accommodate basic Create, Read, Update, Delete (CRUD) operations (or some subset of it). The graph structure is comprised of several, per-player subgraphs, containing collections of player related data. Each sub-graph includes the following nodes:

Player A single player node, storing player-related attributes such as name, id, position, etc.

Week Multiple week nodes, containing attributes delimiting week duration, as well as the players training goals for the week.

Session Summary Variable amount of session summary nodes, holding accumulated performance data from sessions the player has attended.

Figure 5.4 illustrates a sub-graph for a single player. The Player node is connected to a series of Week nodes by an IN relationship, which is further connected to multiple Session Summaries nodes in that particular week by a PART_OF relationship.

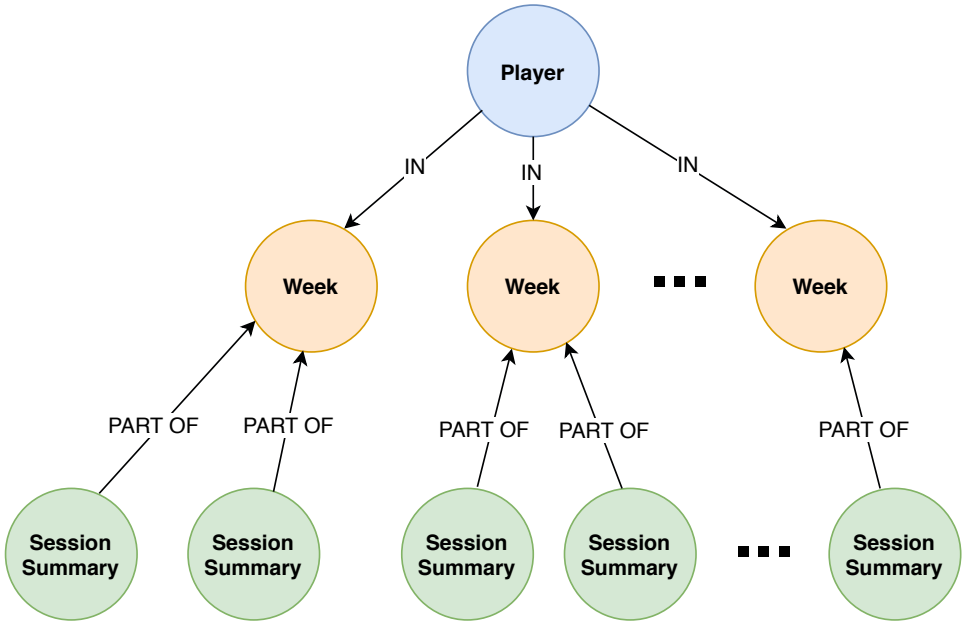


Figure 5.4: Illustration of a sub-graph storing a single player's data

The Player node attributes specifies a players identity and role, as well as his all-time best performance for each event type. Week nodes hold a unique week identifier, timestamps defining the span of the training period, and the player's goals for that specific period. As its name suggest, Session Summary nodes store summaries of measured performance metrics from a single training session that the player has participated in.

Code Listing 5.2: Example query for listing a player's week progression

```
MATCH (p:Player {pid: 62})-[:IN]->
    (w:Week {week_id: 3})<-[:PART_OF]-(s:SessionSummary)
RETURN s
ORDER BY s.date
```

With no distinct relationship between players, the model essentially implements players as first-class citizens. Since most queries are executed from a player-specific context in the ZXY Session, this design makes queries both simple and intuitive. Code Listing 5.2 shows an example of a query for collecting a players weekly progression.

5.2 Frontend

The Frontend is the connecting point between interacting clients and the Backend services. It is composted of a variety of external communication primitives, as well as an independent web server implementing the client interfaces.

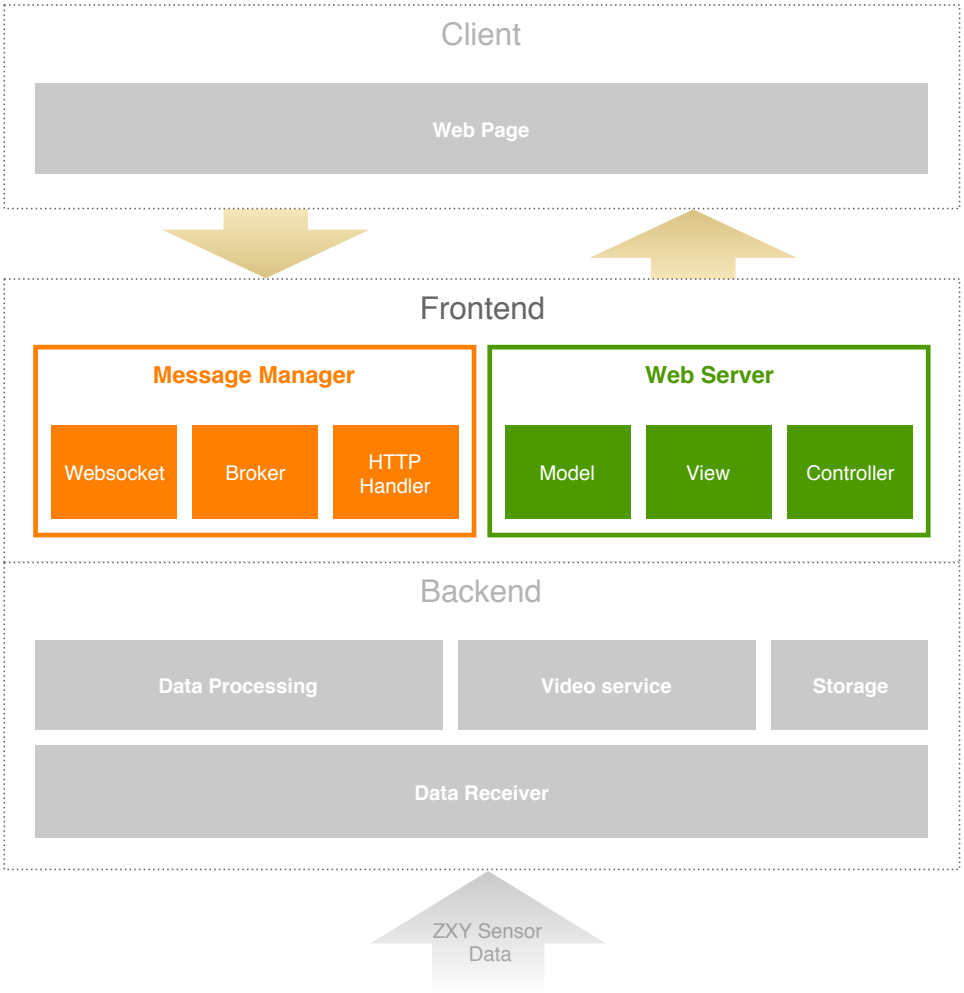


Figure 5.5: Frontend components and subcomponents

5.2.1 Message Manager

The message manager administers the communication primitives between Backend processing components and connected clients. We divide it into two parts; a *push-based* service using WebSockets to send data updates, and a

pull-based service handling HTTP-requests from clients.

Our design suits the specifications defined by coaches in TIL. Their requirement for instant propagation of player performance parameters, combined with their request for keeping the data visualization simplistic, dictates our communication model design. In our solution, vital data points are pushed to users at all times, while less significant data is request-based. For example, the occurrence of a sprint-event is pushed at once, while details regarding the sprint (duration, top speed, etc.) can be further requested by coaches when needed.

Messages are encoded in JavaScript Object Notation (JSON), a widely adopted data-interchange format that provides seamless and effective data object communication between our detached services.

Websocket

The WebSocket protocol [52] provides full-duplex communication channels over a single TCP connection, enabling interaction between a web client (such as a browser) and a web server, with low overhead. It provides a standardized way for the server and client to exchange messages without requiring the data recipient to perform requests on the content beforehand. Essentially, it provides a push-based message service, allowing messages to be passed back and forth while keeping the channel open.

The message manager utilizes the WebSocket protocol to achieve real-time data transfer of live updates from the field. It is employed as a unidirectional message channel from Backend processing units to connected clients, providing efficient dissemination of session-specific events and performance metrics. Client management is handled by a *broker*. Newly connected clients subscribe to the broker, who keeps track of whom to push updates to. This allows multiple coaches to receive updates on player performance parameters to their personal device on the field, simultaneously.

Initialization-messages are sent from the data processing orchestrator the moment a new ZXY belt is detected at the field. Update messages regarding on-field events and general performance metrics are generated and forwarded from the data processing units to the message manager, who further dispatches messages to connected clients. Updates are not batched, but sent individually.

We use six distinct message types to differentiate between the various message data content. They are uniquely identified by an attribute in the message body. The different message types are as follows:

Init Initialization of a new player in the session. Contains the player's personalia, training week progress, weekly training goals and best performance metrics.

Init State Initialization of the current state of the player within the on-going training session. Typically sent subsequent of the *Init* message.

Update Per-player update message. Contains the player's total distance covered and top speed so far during the session. Since coaches deem these parameters as less urgent, we only push these updates every five seconds. We infer that this interval is frequent enough for being perceived as live, while avoiding spamming clients with too many updates.

HIR-event Event-triggered message. Sends the new number of HIRs for a specific player.

Sprint-event Same as HIR-event, but with sprints.

Acceleration-event Same as HIR-event, but with accelerations.

Using distinct message types for separating between various data content is a scaling technique based on the principle of upstream evaluation [53]. By providing expressive data filtering as close as possible to the data source, we avoid redundant data transfer between the server and the clients. A simpler solution is pushing all player-relevant data at frequent intervals, but this will result in larger data transfers, as well as additional client-side complexity of constantly updating the view, often on data that has not even changed. Code Listing 5.3 shows an example of the init-message structure. As the example illustrates, a large portion of player data are static, never changing during an active session. This data needs only be sent once per connected client, in order to setup the current view of the session. By separating between live and static data we minimize the amount of updates sent, as well as reducing the size of messages. This allows for better horizontal scaling with regards to numerous connected clients.

There are two separate circumstances that prompt the initialization of players, causing the need for two distinct init-messages; one is *server-side initialization* and the other is *client-side initialization*. In server-side initialization, the Back-end detects a new player on the field and notifies the clients to update their view accordingly. Here, a single init-message is sufficient, as the player has not yet been involved in any new events this session. In client-side initialization, a new client connects to an already active session, where new field events have already occurred. Hence, the new client requires initial player data to append them to the view, as well as a snapshot of measured parameters from

Code Listing 5.3: Simplified example of the init-message structure.

```
{
    "message_type": "init",
    "belt_id": 6,
    "player_name": "John Doe",
    "best_performance": {
        "best_num_hir": 210,
        "best_hir_dist": 3405,
        "best_num_sprints": 150,
        ...
        },
    "week_progress": [
            {
                "day": "monday",
                "date": "2017-05-14",
                "session_type": "training",
                "num_hir": 35,
                "num_sprints": 20,
                ...
            },
            {
                "day": "tuesday",
                "date": "2017-05-15",
                ...
            }
    ]
    "aggregated_week_progress": {
        "agg_num_hir": 150,
        "agg_num_sprints": 120,
        ...
    },
    "week_goals": {
        "num_hir_goal": 75,
        "num_sprints_goal": 67,
        ...
    },
}
```

the currently progressing session. Because current session data only resides in memory, such read operations require some form of data synchronization to avoid race conditions on continuously updated session variables. Simply using synchronization primitives (i.e. locks) will avoid races, but decrease processing efficiency through frequent thread suspension and lock contention. To preserve processing efficiency, we utilize message channels for data synchronization. When a new client connects, the message manager signals each processing unit through individual channels, telling them to push a snapshot of their data to the new client. This way no locks is required and clients are eventually brought up to speed through lazy propagation.

Request-based API

The pull-based communication model is supported through a HTTP server in the message manager. It implements a RESTful interface used by clients to request additional information regarding training week progression, or details about completed events from an on-going session.

As explained earlier, event data from an on-going session (such as HIRs or sprints) resides in memory, requiring synchronization primitives to prevent race conditions on read operations. Unlike the push based WebSocket, the HTTP server handles requests synchronously, contrary to lazily propagating response data to the end user. As a consequence, we choose to handle data synchronization through locks instead of message channels on user requested data. To minimize lock contention, the server acquires the lock only to retrieve a copy of the required data, further processing the request through a memory-safe duplicate.

A downside of using locks for synchronization is the exposure to user-generated contention. A substantial amount of concurrent user requests (generated by web page interactions) can force frequent suspensions of processing units, causing the threads to wait for the lock rather than processing sensor data. While this is not a genuine concern considering the small amount of coaches in TIL using the application simultaneously, we still recognize it as a vulnerability in the data path.

In hindsight, we suggest an alternate solution through client-side caching. This can be achieved by pushing extended event data along with update-messages, store the data at the client and retrieve it locally at user request. Such a scheme will offload request handling from the Frontend to the Client, as well as avoid potential user-generated contention. Its downsides include a more complex client implementation, as well as increased update-message sizes, resulting in larger communication overhead. Also, if coaches do not require explicit event details, data is cached redundantly.

5.2.2 Web Server

The web server is a simple HTML5 web-application. It is implemented using Blueprint[3], a Model-view-controller (MVC) framework for Go. Further, the GUI is designed using Bootstrap 4[4] to support smaller devices such as mobile phones or tablets, primarily used by coaches when they are on the field.

3. https://blue-jay.github.io/
4. https://www.getbootstrap.com/

The framework consist of the following:

The model is represented by two databases. One is a MySQL database that holds user credentials for client authentication. The other is the graph database that resides in the Backend, used for populating view variables.

The view displays static web page content to the users.

The controller makes API calls to our backend service, as well as handling user I/O on interactive components (i.e. buttons, forms).

Using the MVC design pattern simplifies implementation due to the strong coupling between the Frontend and the Backend layers. By connecting the controller directly to the Backend ZXY Session we can interact with session-specific functionality without the need to call procedures remotely. Though this tight coupling makes our system more error prone and complex, it serves well for a first version of a proof-of-concept application.

5.3 Client

The Client is the main point of interaction for users of Metrix. Users access the web page served by the Frontend through standard web browsers. Authenticated users are presented with several interfaces allowing them to interact with Metrix. Through this section we will explain each interface and the functionality they provide.

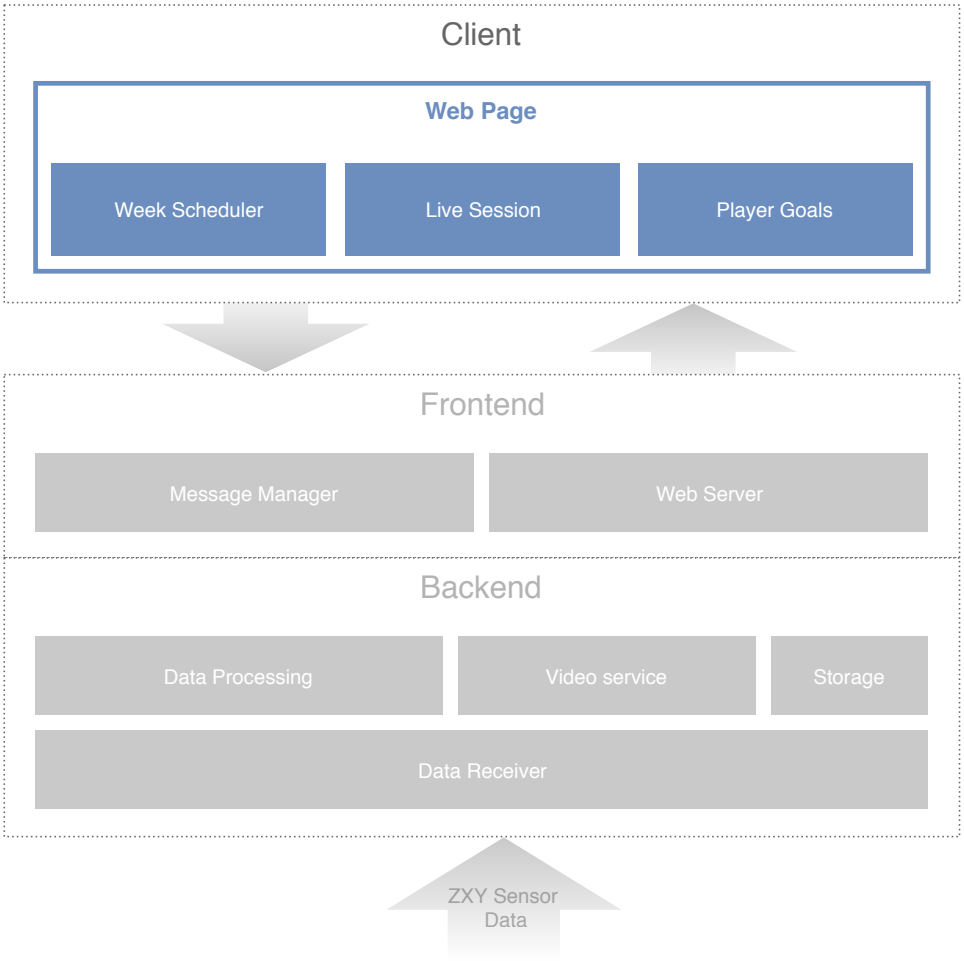


Figure 5.6: Client components and subcomponents

5.3.1 Layout

The web application interfaces are designed according to our non-functional requirements of being user-friendly, with the user requirement regarding simplicity in mind. This is achieved through the principles from the field of Human-Computer Interaction (HCI) [54]. We use a light color theme for page bodies, while the header is red corresponding to TIL's team colors. We use the same color palette across the different application interfaces to create a uniform design across pages.

Elements within the page use the Bootstrap grid system for placement and alignment. The grid is dynamic and adjusts accordingly with the supported screen resolution of the user's device. The use of grids and white space in the

canvas creates the perception of a simple and user-friendly interface, directing the focus to important elements in the view. The layout of the player-grid from a live ZXY Session is shown in Figure 5.7.

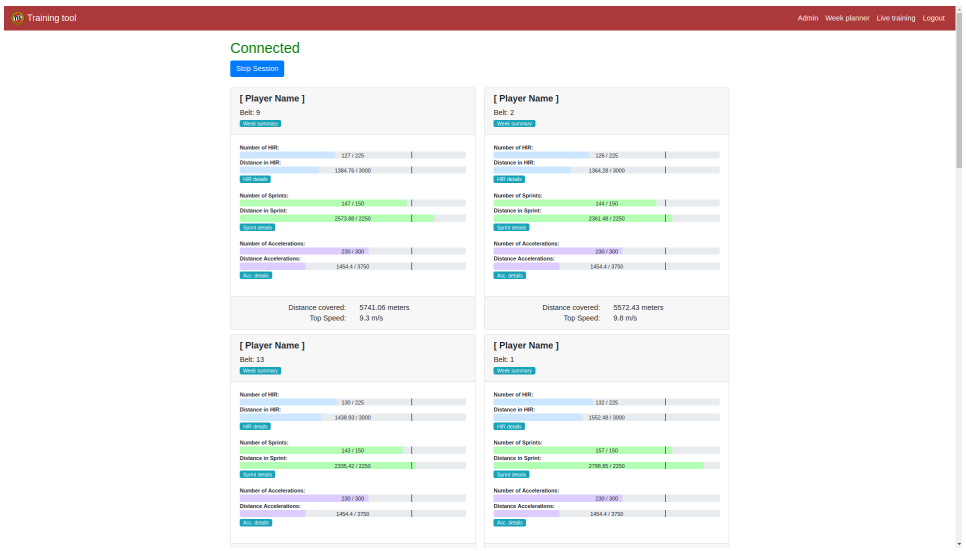


Figure 5.7: Grid of active players

5.3.2 Training Schedule

The match schedule of elite soccer teams is frequently altered throughout the season. As trainings are scheduled intermediate of official matches, the training schedule is naturally affected by this. For Metrix to accurately track player progression throughout the training week, our system is dependent on knowing the dates of when the weeks start and end. To relieve coaches from the tedious task of manually updating the Metrix training schedule every time a match date changes, we have implemented an automated solution.

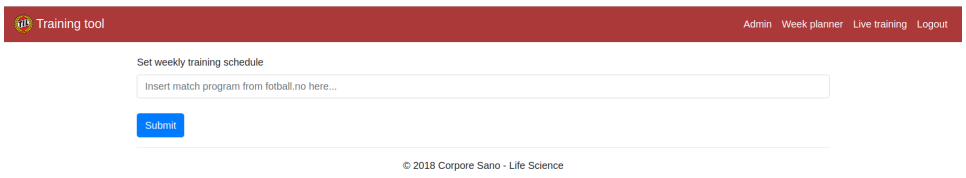


Figure 5.8: Interface for scheduling training periods.

Our implemented Training Schedule interface allows coaches to input the URL to the official season match schedule. The schedule is a spreadsheet issued by

Norwegian Football Association (NFF), and is publicly available through their web page.[5] The input schedule is parsed and stored in the database, updating the start and stop points of training weeks automatically. If the schedule changes during the season, the trainer must manually re-post the new schedule.

5.3.3 Week Planner

Adhering to our functional requirement, the Week Planner interface provides coaches with a method to set player-specific training goals within the current training period. The page displays a table of all the players in the team. Each player has adjustable sliders ranging from 0-100 percent. Coaches may adjust these individually to the player, specifying the exercise load in each of the measured physical performance parameters, further illustrated in Figure 5.9.

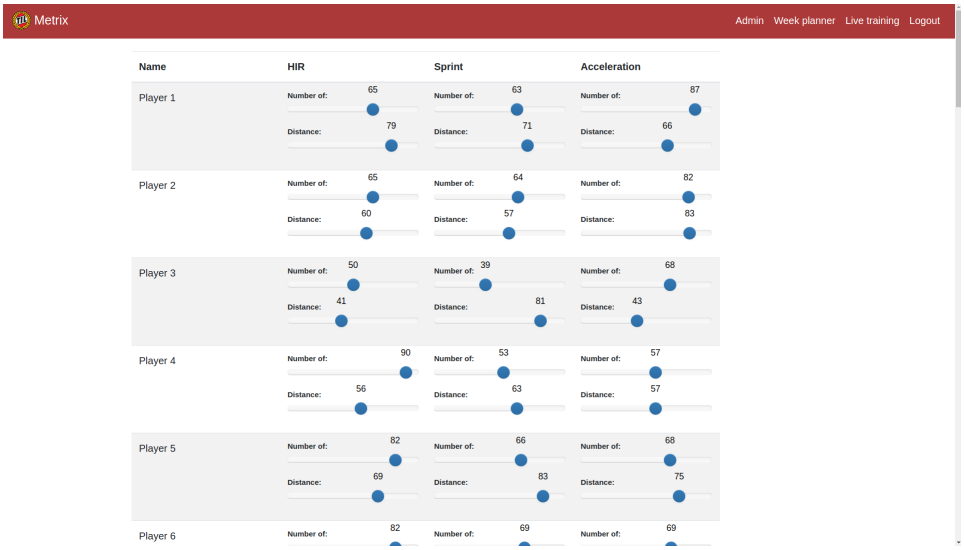


Figure 5.9: Interface for planning weekly training load for individual players

Coherent with our specified user requirement, the percentage is calculated based on each player's all-time best performance. The default setting is provisional, set to 75% for all event parameters. These may further be adjusted manually by the coaches throughout the training period. Submitted goals associated with the current training period are stored in the database. These values are further used to portray the players goal on the progress bar during a live ZXY session, as we later illustrate in 5.3.4. When a new training period begins, the player goals is reset to its default setting.

5. https://www.fotball.no

5.3.4 Live Session

The Live Session interface displays live data of players participating in an on-going match or training session. The interface satisfies our functional requirement of providing coaches with a live view of their players while operating on the field. When a user navigates to the session page he is met with the option to start a new session by clicking a button. If a session is already started, the coach is immediately connected to the on-going one. During an active session the start button is replaced by a stop button, allowing coaches to end it. Pressing the button will send a stop signal to the Backend, who subsequently persists session data and resets.

Unlike the rest of the web application, the session interface is dynamically rendered using client-side scripting. Native JavaScript[6] and jQuery[7] is used to handle dynamic creation and update of page elements, while HTTP requests to the Frontend is performed by Asynchronous JavaScript and XML (AJAX)[8]. The view is constantly changing in response to received updates through the WebSocket, displaying player performance metrics to the end-user.

The Live Session page implements our most complex user interface. We are challenged by our requirement for displaying the player performance metrics in a orderly fashion. On one hand we want to present player data through graphic elements (i.e. progress bars), which are easy for users to comprehend. On the other hand, we see the need for displaying the explicit data values through plain numbers. In an attempt to do both, we experience how easy the view becomes unorganized and cluttered. Hence, our current design focuses on displaying the data through graphic elements, and only display numbers where it is most needed.

6. https://www.javascript.com/
7. http://jquery.com/
8. http://api.jquery.com/jQuery.ajax/

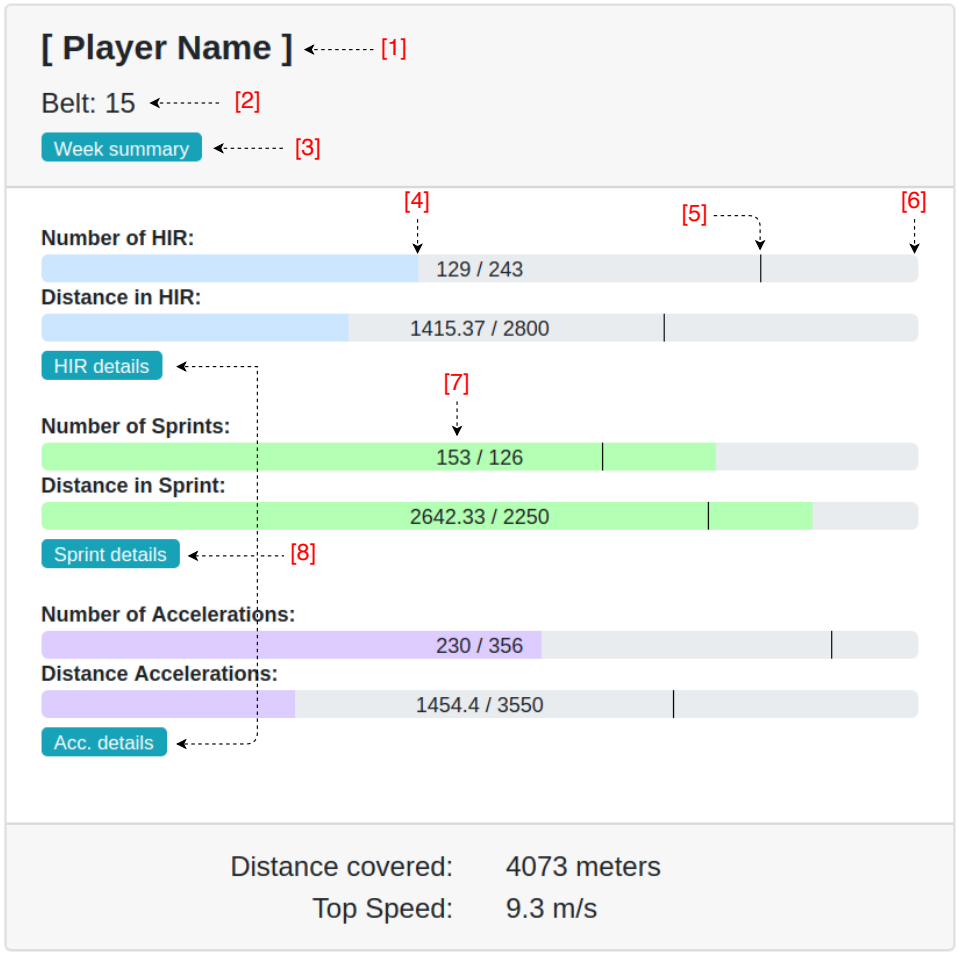


Figure 5.10: Detailed view of a single player card

The players are displayed as an array of cards, as depicted in Figure 5.7. Each card is a small square container showing live, player-specific data. Figure 5.10 illustrates a single player card. The header of the card contains the player's name (1), belt id (2) and a button (3) for listing detailed performance data from previous training sessions in the current week. The detailed info is displayed in a popover, only visible through user interaction. This feature responds to our functional requirement of providing a historical view of previously executed trainings. The card body consists of six progress bars, visualizing number of conducted HIR (4), sprint and acceleration events, as well as distance covered during them. Progress bars display accumulated performance metrics from the entirety of the training week. A small marker on the bar (5) indicates the preset goal that coaches have set for the player for the current training period. The end of the progress bar (6) is defined by the player's all-time best

performance. Taking into account that the player may exceed this limit we also show the values explicitly with a label (7) in the center of the bar. The label show *accomplished value* out of *weekly goal* (e.g. 126 / 225 HIRs in the figure).

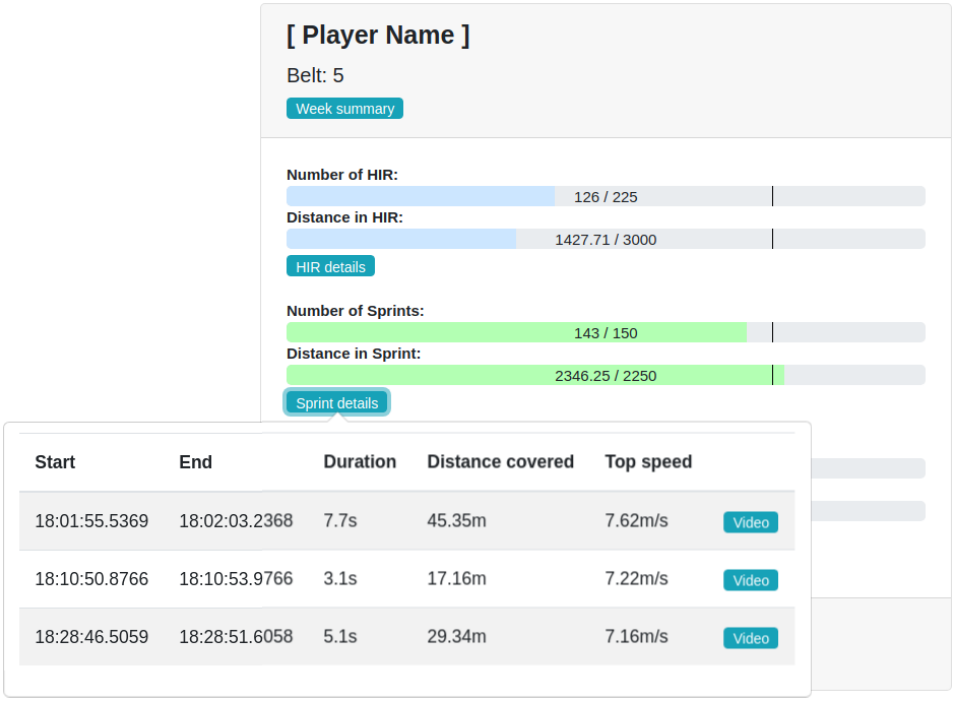


Figure 5.11: Detailed view of session events

Users may request a detailed view of completed events by the click of a button (8). Detailed data is comprised of single events, arranged in a table, containing additional information on each of them. Event details are displayed in a popover, shown and hidden by user interactions. Figure 5.11 shows an example of a detailed view on completed sprints for a specific player. Each event in the detailed view is coupled with a button for playing a video of the performed event. When pressed, a video player will pop up and display the requested content.

5.4 Summary

This chapter has given a detailed description of the design and implementation of Metrix. We have explained how the Backend processes the input from the

ZXY sensor data stream by utilizing a worker pool of concurrent processing units, administered by an orchestrator. We further described a conceptual video service that allows coaches to request video playback of captured field-events. We have described the Frontend layer, explaining how data is served to the clients through both push and pull-based services. Lastly, we described our Client and its supported user interfaces, giving details to how performance parameters are displayed to its users.

6 Evaluation

In this chapter, we evaluate the performance and usefulness of Metrix. We are particularly interested in evaluating our system with regards to our non-functional requirements concerning scalability, real-time responsiveness, applicability and user-friendliness. The evaluation includes a conducted user survey, as well as performance experiments.

6.1 Performance

The performance evaluation of Metrix concerns the systems capability of processing physical performance parameters and delivering the results, in real-time, to connected clients. We want to investigate if the system output keeps up with the continuous input stream of ZXY sensor data, without falling farther and farther behind as the match or training session progresses.

The data input volume scales linearly with the number of players on the field, consequently affecting system behavior. Our experiments cover two realistic scenarios; an official match and a training session. The difference between the two are the number of active players on the field, differing from 11 to 25, respectively. While there are never more than 11 players on the field during a match, the number of players attending trainings will vary from session to session. We set this number to 25, which covers full attendance from the entire team. Additionally, we are interested in how our system scales with regards

to an increasing amount of coaches using Metrix simultaneously. Even though TIL only have a limited amount of coaches, we are still curious of how Metrix performs when serving a substantial amount of clients.

6.1.1 Experimental Setup

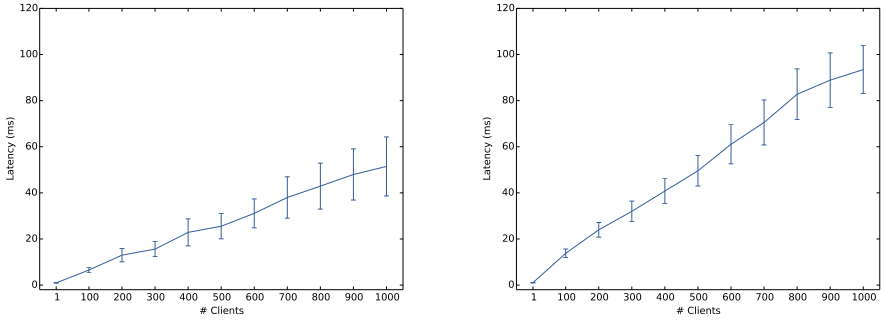
In order to test real-time viability we measure end-to-end latency, from a player executes an event on the field until it is received by the clients. Our test setup involves a simulated ZXY server with real ZXY sensor data, feeding raw data records into Metrix. Also, we use a separate program for simulating numerous clients receiving the processed data from Metrix. Both are hosted on the same PC to synchronize time readings. This allows us to measure the time span between the data record marking the end of an event is sent, until its content is received by the clients after it is processed by Metrix. As each client will receive events consecutively, we only measure the longest latency across all clients for each event.

The simulated ZXY server transmits data records at 20 Hz, similar to the setup installed at Alfheim. Our ZXY test data is one half (45 minutes) of a real soccer match from 2013. Hence, each experiment is equally long as the period. For the 11-player experiment, there is a total of 221 events captured by Metrix, distributed among the players. In the 25-player experiment we have duplicated some of the player data, resulting in a total of 525 captured events.

Metrix is deployed on a desktop computer with an Intel Core i7-2600 processor, 32GB DDR3 RAM, connected to a 1 Gb/s Ethernet network. The databases are deployed locally at the server, using a Samsung 850 EVO30 SSD hard drive as storage. Our ZXY data server and client simulations runs on a workstation with an Intel Core i5-4200M, 2,50 GHz CPU connected by a 1 Gb/s Ethernet network. All units use the same network, consequently resulting in close to zero wide-area latency.

6.1.2 Results

Figure 6.1 shows the results of the end-to-end latency on captured events from the match. The graphs show that the latency scales linearly with the increasing amount of clients. Average latency during the 45 minute session is below 100 milliseconds, with both 11 (a) and 25 (b) players on the field, and up to 1000 clients using Metrix. Further, the graphs show how end-to-end latency increase with the added number of players on the field. We observe that the average latency approximately doubles when increasing from 11 to 25 players.

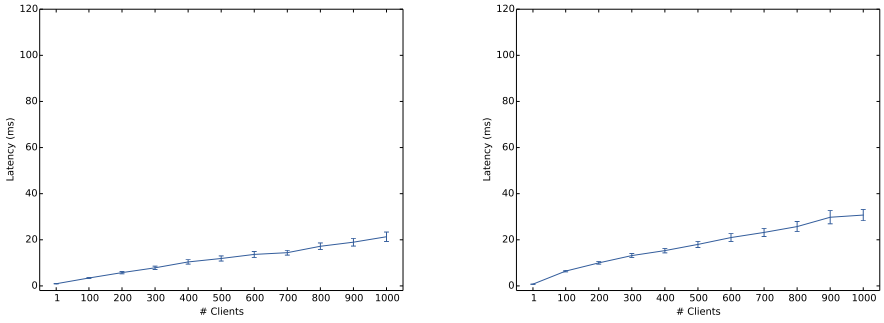


(a) 11 players, 221 events **(b)** 25 players, 525 events

Figure 6.1: End-to-end latency with 11 and 25 active players. Error bars show the 95th-percentile confidence interval

The error bars in the graphs show a significant deviation in latency between each received event. When taking a closer look at the numbers we observe mostly stable latencies, but also sudden spikes occurring at regular intervals. These findings led us to perform a follow-up experiment in order to investigate the variance more closely.

As we mentioned in subsubsection 5.2.1, we periodically push an update-message to all clients at 5-second intervals. Observing that the occurrence of latency spikes were so consistent, being almost predictable, we suspected it to be correlated with these continuous updates. Figure 6.2 shows the results of performing the same experiment again, but without Metrix pushing the periodic update-messages to the clients.



(a) 11 players, 221 events **(b)** 25 players, 525 events

Figure 6.2: End-to-end latency with 11 and 25 active players, without periodic client updates every five seconds. Error bars show the 95th-percentile confidence interval

The graphs show a significant improvement to average latency, as well as a major decrease in variation between measurements. From our second experiment, we gather that the message manager is a bottleneck in Metrix. When multiple updates are pushed from the Backend simultaneously, they are put in a message queue at the Frontend, further dispatched sequentially by order of first come, first served. Hence, the variation we observe in our first experiment is due to the shifting queue time. This affects the end-to-end latency of all push-based client updates that should occur simultaneously, including initialization of newly connected clients and events captured from the field. A potential solution is pushing updates concurrently, which induce alterations to the message manager and its current data synchronization primitives. Another solution is to batch updates, involving changes to our message format and communication model.

As our end-to-end latency is measured between devices operating on the same network, wide-area latency is not properly assessed through our experiments. In a real-world deployment, we expect the general latency to increase, depending on factors such as clients bandwidth or their proximity to the server.

From our experiments we conclude that Metrix satisfies our requirement of performing player analysis in real-time. The latency from an event occurs on the field until it is propagated to Metrix clients are low and does not fall behind as the session progresses. We have shown that our system can handle up to at least 1000 clients, without compromising user experience. With the limited amount of coaches in TIL, we regard Metrix performance as more than proficient, satisfying our requirement regarding scalability. The high variance in latency is negligible, as the differences are sub one-hundred milliseconds. We conjecture that this is not perceived by the end user.

6.2 User Evaluation

We have previously described how everything from the requirements to system implementations have been guided by continuous feedback from the coaches in TIL. Since Metrix has been developed for, and in collaboration with, its end-users, we infer that the most suitable way to evaluate the system is to perform user surveys targeting these end-users.

Since Metrix is not yet in operational use, we are unable to evaluate first-hand user experience of the application. Instead, we base our evaluation on a user-oriented presentation, involving an extensive demonstration of Metrix and its implemented features. The demo was conducted at Alfheim stadium, where the assessors were given a realistic demo based on simulated ZXY data

input. Mock-up profiles were made for each simulated player, illustrating how Metrix aggregates performance parameters from previous trainings with the live data. The demo was followed by a questionnaire, evaluating Metrix by three main categories; functionality, design and overall interest in using Metrix. The questions focuses particularly on attitudes regarding the availability of player performance parameters in real-time during match or practice, opposite to only having them post-game.

The user survey consists of statements regarding Metrix, each rated using a balanced 5-point Likert scale,[1] with the following mapping of each rating.

- 1 Strongly Disagree
- 2 Disagree
- 3 Neutral
- 4 Agree
- 5 Strongly Agree

The plotted charts show the average rating, with error bars indicating the lowest and highest registered ratings. The full questionnaire is available in Appendix A.

6.2.1 Assessors

The assessors participating in the evaluation includes three coaches from TIL's elite soccer team, and one former coach of the Norwegian national soccer team, now a sports scientist working with quantification of player data. All participants are considered experts in their respective field, with first-hand experience in player development, soccer analysis and physical training disciplines.

6.2.2 Results

The first category of our survey examines the functionality of Metrix. The two first questions concern the general usefulness of having a computerized toolkit for monitoring players live *during* match or practice. The assessors were in agreement that having access to data instantly on the field is a valuable asset for monitoring physical load on individual players. Further, they indicate that such a toolkit would be useful for doing personalized intervention, such as resting players, or increase individual load through alternative drills.

1. https://en.wikipedia.org/wiki/Likert_scale

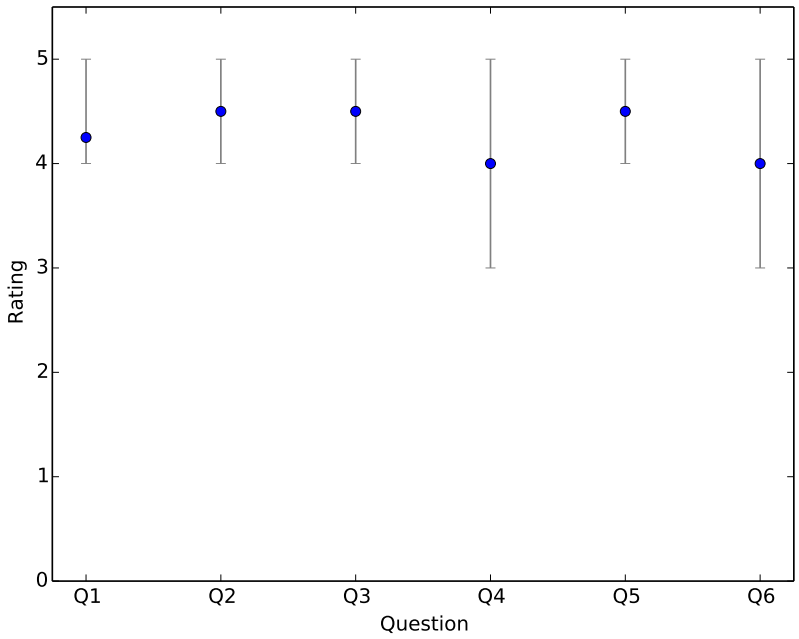


Figure 6.3: Survey of Metrix **functionality**.

Questions three through six examines how the assessors consider Metrix as a viable toolkit for real-time monitoring of players. The results indicates that the assessors believes Metrix will improve objective monitoring of player load, and can be very useful for identifying suitable training drills in order to accomplish weekly training goals. The survey also indicates that the assessors were diverged on our question about Metrix enhancing the individualization of training programs *during* trainings. Some assessors Strongly Agreed, while others were Neutral. We speculate that this variance might be rooted in how coaches prepare the trainings in advance. Making alteration to a planned training program midway, specialized to each individual at that, might seem less practical for some.

Our question regarding the usefulness of video playback of events also showed a diversity between assessors opinion. Some Strongly Agreed that such a feature would be useful, while others were Neutral. From this we conjecture that access to video snippets is more of a nice-to-have than a need-to-have feature. However, we believe that such a feature must be experienced in practice in order for the users to rate its applicability in day-to-day use.

The next category in our survey evaluates the Metrix design. The questions

make inquiries regarding user experience of the Metrix interface, as well as how the player data is presented. We were particularly interested in evaluating if the users understood the player performance data through the implemented progress bars and its supportive elements.Figure 6.4 shows the gathered results from the design category.

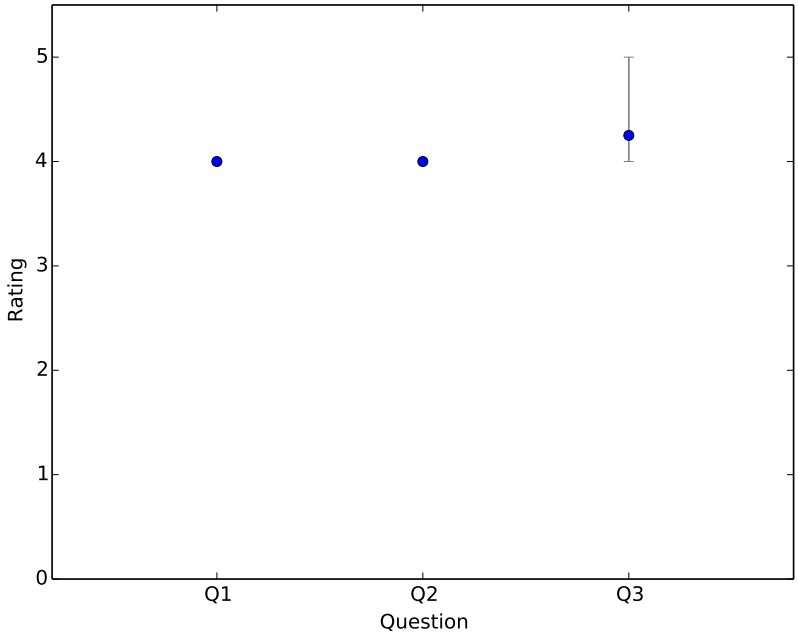


Figure 6.4: Survey of Metrix **design**.

The results show an overall agreement between the assessors, conveying that Metrix provides a user friendly interface, with neatly presented data. All of the assessors also answered that the progress bars made player performance data easy to comprehend. From our gathered results we conclude that Metrix satisfies our non-functional requirement of being user-friendly. Additionally, we fulfill the user requirement regarding simplicity, by visualizing player data in an intuitive way.

Our last category summarizes the assessors overall attitude towards Metrix as a coaching toolkit. Figure 6.5 shows the results of the inquiry.

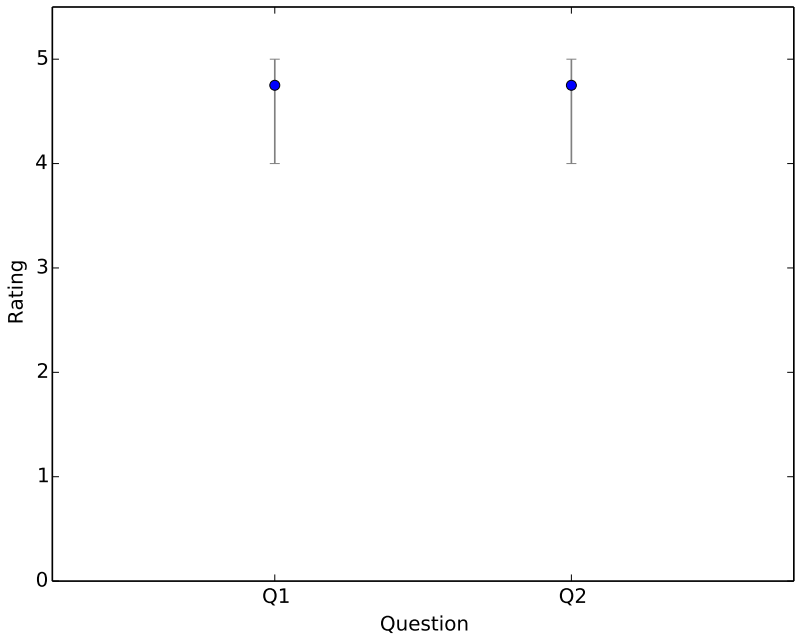


Figure 6.5: Survey of **overall attitude** towards Metrix.

From our results, we gather that all the assessors believe Metrix will have great impact on individual training load monitoring, and that it enhances coach intervention during match or trainings. All the assessors state that they would use Metrix on a daily basis if provided. These results emphasize the necessity for putting Metrix in operational use.

6.3 Summary

This chapter has evaluated Metrix with regards to our non-functional requirements concerning scalability, real-time responsiveness, applicability and user-friendliness. Through performance experiments, we have shown that Metrix is able to detect and propagate on-field events to clients with low end-to-end latency, and that it scales to a high amount of users. The results from a conducted user survey show that end-users are happy with Metrix functionality, and find it easy to use. Assessors state that they would use Metrix on a daily basis, if provided.

/7 Conclusion

This thesis have presented the development of Metrix, a system supporting real-time monitoring of elite soccer players' performance parameters during match or practice. Metrix provides coaches with a toolkit to quantify specific movement patterns of individual players, enabling them to analyze their training load in relation to preset training goals. Additionally, we have provided a method for coupling captured events with video recordings, allowing coaches to view replays of player-performed events.

Metrix has been developed in close collaboration with its end-users; the coaches in TIL. Through their expert knowledge we were able to establish user requirements of Metrix, which further implicated our chosen system design, graphical interface and implemented features.

Through a performance evaluation we have shown that Metrix is able to efficiently perform *real-time* analysis on player performance metrics, provided by the ZXY sensor system. Metrix is able to detect, process and propagate captured field-events with minimal end-to-end latency. Further, a conducted user evaluation shows that coaches find Metrix highly useful for monitoring physical performance parameters, and has great impact on the individualization of physical training load. Users express they would use Metrix on a daily basis, if provided.

7.1 Future Work

Through the development of Metrix we have discovered several areas of improvement, as well as additional features that could prove useful. This section will address the potential future work for making Metrix better.

Deployment In the future, we plan on putting Metrix in operational use at Alfheim stadium. The current prototype is fully functional, but requires minor integrations with the actual ZXY data stream. We also believe that giving coaches first-hand experience with Metrix will reveal further tweaks and improvements to the system.

Privacy As we mentioned in Section 1.2, the privacy of players is not addressed in this thesis. Future work will involve extensive improvements to data privacy, adhering to the new General Data Protection Regulations (GDPR) [55]. This will involve more secure data processing, communication and storage.

Video Integration While Metrix have exemplified how using video replays of events is possible in real-time, we currently have no access to the recorded content stored at Alfheim. Future work will involve a closer integration with the Bagadus video system, allowing us to request videos from their service, in real-time on the field.

pmSys Through pmSys [56], a player monitoring system used by TIL, players register daily Rating of Perceived Exertion (RPE) and Pre-Training Wellness (PTW) reports. Coaches have expressed the usefulness of having these reports displayed in the Metrix live tool. These reports, coupled with Metrix performance parameters, would further portray the players physical form and exertion.

Bibliography

[1] C. Barnes, D. Archer, M. Bush, R. Hogg, and P. Bradley, "The evolution of physical and technical performance parameters in the english premier league," *International Journal of Sports Medicine*, vol. 35, pp. 1–6, 2014.

[2] H. D. Johansen, S. A. Pettersen, P. Halvorsen, and D. Johansen, "Combining video and player telemetry for evidence-based decisions in soccer," *Proc. of icSPORTS*, 2013.

[3] D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, and P. R. Young, "Computing as a discipline," *Commun. ACM*, vol. 32, pp. 9–23, Jan. 1989.

[4] Å. Kvalnes, D. Johansen, R. van Renesse, F. B. Schneider, and S. V. Valvag, "Omni-kernel: An operating system architecture for pervasive monitoring and scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2849–2862, 2015.

[5] S. V. Valvåg and D. Johansen, "Oivos: Simple and efficient distributed data processing," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pp. 113–122, IEEE, 2008.

[6] S. V. Valvag and D. Johansen, "Update maps–a new abstraction for high-throughput batch processing," in *Networking, Architecture, and Storage, 2009. NAS 2009. IEEE International Conference on*, pp. 431–438, IEEE, 2009.

[7] D. Johansen, K. Marzullo, and K. Lauvset, "An approach towards an agent computing environment," in *Electronic Commerce and Web-based Applications/Middleware, 1999. Proceedings. 19th IEEE International Conference on Distributed Computing Systems Workshops on*, pp. 78–83, IEEE, 1999.

[8] D. Johansen, H. Johansen, and R. van Renesse, "Environment mobility: moving the desktop around," in *Proceedings of the 2nd workshop on Mid-*

dleware for pervasive and ad-hoc computing, pp. 150–154, ACM, 2004.

[9] D. Johansen, R. Van Renesse, and F. B. Schneider, "Operating system support for mobile agents," in *Hot Topics in Operating Systems, 1995. (HotOS-V), Proceedings., Fifth Workshop on*, pp. 42–45, IEEE, 1995.

[10] S. V. Valvag and D. Johansen, "Cogset: A unified engine for reliable storage and parallel processing," in *Network and Parallel Computing, 2009. NPC'09. Sixth IFIP International Conference on*, pp. 174–181, IEEE, 2009.

[11] S. V. Valvåg, D. Johansen, and Å. Kvalnes, "Cogset: a high performance mapreduce engine," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 1, pp. 2–23, 2013.

[12] H. D. Johansen, R. V. Renesse, Y. Vigfusson, and D. Johansen, "Fireflies: A secure and scalable membership and gossip service," *ACM Transactions on Computer Systems (TOCS)*, vol. 33, no. 2, p. 5, 2015.

[13] H. Johansen, D. Johansen, and R. van Renesse, "Firepatch: Secure and time-critical dissemination of software patches," in *IFIP International Information Security Conference*, pp. 373–384, Springer, 2007.

[14] A. T. Gjerdrum, R. Pettersen, H. D. Johansen, and D. Johansen, "Performance of trusted computing in cloud infrastructures with intel sgx," in *Proceedings of the 7th International Conference on Cloud Computing and Services Science. Porto, Portugal: SCITEPRESS*, pp. 696–703, 2017.

[15] D. Johansen, P. Halvorsen, H. Johansen, H. Riiser, C. Gurrin, B. Olstad, C. Griwodz, Å. Kvalnes, J. Hurley, and T. Kupka, "Search-based composition, streaming and playback of video archive content," *Multimedia Tools and Applications*, vol. 61, no. 2, pp. 419–445, 2012.

[16] D. Johansen, M. Stenhaug, R. B. Hansen, A. Christensen, and P.-M. Høgmo, "Muithu: Smaller footprint, potentially larger imprint," in *Digital Information Management (ICDIM), 2012 Seventh International Conference on*, pp. 205–214, IEEE, 2012.

[17] M. Stenhaug, Y. Yang, C. Gurrin, and D. Johansen, "Muithu: A touch-based annotation interface for activity logging in the norwegian premier league," in *International Conference on Multimedia Modeling*, pp. 365–368, Springer, 2014.

[18] H. K. Stensland, V. R. Gaddam, M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljødal, Ø. Landsverk, *et al.*, "Bagadus:

An integrated real-time system for soccer analytics," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 10, no. 1s, p. 14, 2014.

[19] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz, *et al.*, "Bagadus: an integrated system for arena sports analytics: a soccer case study," in *Proceedings of the 4th ACM Multimedia Systems Conference*, pp. 48–59, ACM, 2013.

[20] S. Sægrov, A. Eichhorn, J. Emerslund, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen, "Bagadus an integrated system for soccer analysis," in *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, pp. 1–2, IEEE, 2012.

[21] R. van Renesse, H. Johansen, N. Naigaonkar, and D. Johansen, "Secure abstraction with code capabilities," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pp. 542–546, IEEE, 2013.

[22] C. Carling, T. Reilly, and A. M. Williams, *Handbook of soccer match analysis: A systematic approach to improving performance*. Routledge, 2007.

[23] W. R. Data, "The role of motion analysis in elite soccer," *Sports Med*, vol. 38, no. 10, pp. 839–862, 2008.

[24] D. B. Dwyer and T. J. Gabbett, "Global positioning system data analysis: Velocity ranges and a new definition of sprinting for field sport athletes," *The Journal of Strength & Conditioning Research*, vol. 26, no. 3, pp. 818–824, 2012.

[25] P. S. Bradley, M. Di Mascio, D. Peart, P. Olsen, and B. Sheldon, "High-intensity activity profiles of elite soccer players at different performance levels," *The Journal of Strength & Conditioning Research*, vol. 24, no. 9, pp. 2343–2351, 2010.

[26] T. Reilly and V. Thomas, "A motion analysis of work rate in different positional roles in professional football match play," vol. 2, pp. 87–97, 01 1976.

[27] J. Bangsbo, L. Nørregaard, and F. Thorsoe, "Activity profile of competition soccer.," *Canadian journal of sport sciences= Journal canadien des sciences du sport*, vol. 16, no. 2, pp. 110–116, 1991.

[28] "Sports data company | sports technology, data feeds, content | stats." https://www.stats.com. [Online].

[29] D. S. Valter, C. Adam, M. Barry, and C. Marco, "Validation of prozone®: A new video-based performance analysis system," *International Journal of Performance Analysis in Sport*, vol. 6, no. 1, pp. 108–119, 2006.

[30] S. Wang and G. Zhou, "A review on radio based activity recognition," *Digital Communications and Networks*, vol. 1, no. 1, pp. 20–29, 2015.

[31] "Sports tracking - chyronhego." https://chyronhego.com/products/sports-tracking. [Online].

[32] B. Wisbey, P. G. Montgomery, D. B. Pyne, and B. Rattray, "Quantifying movement demands of afl football using gps tracking," *Journal of science and Medicine in Sport*, vol. 13, no. 5, pp. 531–536, 2010.

[33] M. Pentz, "Sounders among pioneers in mls with analytics, training," August 2015. [Online; posted 03-August-2015].

[34] "Gps tracking systems for elite sports | globally positioning sport since 2001." http://www.gpsports.com. [Online].

[35] "World leader in sport technology for elite sports | catapult sports." http://www.catapultsports.com. [Online].

[36] "Statsports | GPS player tracking and performance analysis." http://www.statsports.com. [Online].

[37] J. Bangsbo, M. Mohr, and P. Krustrup, "Physical and metabolic demands of training and match-play in the elite football player," *Journal of sports sciences*, vol. 24, no. 07, pp. 665–674, 2006.

[38] B. Drust, T. Reilly, and E. Rienzi, "Analysis of work rate in soccer," vol. 4, pp. 151–155, 11 1998.

[39] T. Reilly and D. Gilbourne, "Science and football: a review of applied research in the football codes," *Journal of sports sciences*, vol. 21, no. 9, pp. 693–705, 2003.

[40] V. Di Salvo, R. Baron, H. Tschan, F. C. Montero, N. Bachl, and F. Pigozzi, "Performance characteristics according to playing position in elite soccer," *International journal of sports medicine*, vol. 28, no. 03, pp. 222–227, 2007.

[41] P. S. Bradley, W. Sheldon, B. Wooster, P. Olsen, P. Boanas, and P. Krustrup, "High-intensity running in english fa premier league soccer matches," *Journal of sports sciences*, vol. 27, no. 2, pp. 159–168, 2009.

[42] P. S. Bradley, C. Carling, D. Archer, J. Roberts, A. Dodds, M. Di Mascio, D. Paul, A. Gomez Diaz, D. Peart, and P. Krustrup, "The effect of playing formation on high-intensity running and technical profiles in english fa premier league soccer matches," *Journal of sports sciences*, vol. 29, no. 8, pp. 821–830, 2011.

[43] C. Lago-Peñas, E. Rey, J. Lago-Ballesteros, L. Casais, and E. Domínguez, "Analysis of work-rate in soccer according to playing positions," *International Journal of Performance Analysis in Sport*, vol. 9, no. 2, pp. 218–227, 2009.

[44] I. Baptista, D. Johansen, A. Seabra, and S. A. Pettersen, "Position specific player load during match-play in a professional football club," *PLOS ONE*, vol. 13, pp. 1–10, 05 2018.

[45] I. Baptista, A. Seabra, D. Johansen, and S. A. Pettersen, "Position specific player load during match in a professional football club," in *The city of Rennes (France) is immensely honored to host the Vth World Congress in Science and Soccer, after Portland (2014) and Copenhagen (2015) and before Melbourne (2019).*, p. 65.

[46] I. Baptista, A. Seabra, D. Johansen, and S. A. Pettersen, "A comparison of weekly training load and match performance in selected physical variables in elite soccer players," in *The city of Rennes (France) is immensely honored to host the Vth World Congress in Science and Soccer, after Portland (2014) and Copenhagen (2015) and before Melbourne (2019).*, p. 359.

[47] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland, and P. Halvorsen, "Soccer video and player position dataset," in *Proceedings of the 5th ACM Multimedia Systems Conference*, pp. 18–23, ACM, 2014.

[48] F. Wagner, R. Schmuki, T. Wagner, and P. Wolstenholme, *Modeling software with finite state machines: a practical approach*. CRC Press, 2006.

[49] A. Mortensen, V. R. Gaddam, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen, "Automatic event extraction and video summaries from soccer games," in *Proceedings of the 5th ACM Multimedia Systems Conference*, pp. 176–179, ACM, 2014.

[50] "The neo4j graph platform - the #1 platform for connected data." https://www.neo4j.com. [Online].

[51] D. Montag, "Understanding neo4j scalability," *White Paper, Neotechnology*, 2013.

[52] I. Fette and A. Melnikov, "The websocket protocol," RFC 6455, RFC Editor, December 2011. `http://www.rfc-editor.org/rfc/rfc6455.txt`.

[53] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems (TOCS)*, vol. 19, no. 3, pp. 332–383, 2001.

[54] C. D. Wickens, J. Lee, Y. D. Liu, and S. Gordon-Becker, *Introduction to Human Factors Engineering (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2003.

[55] G. D. P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46," *Official Journal of the European Union (OJ)*, vol. 59, pp. 1–88, 2016.

[56] T. T. Hoang, "pmsys: Implementation of a digital player monitoring system," Master's thesis, 2015.

Appendices

A

User Survey

METRIX QUESTIONNAIRE

Please indicate to what extent you agree or disagree with each statement:

	Functionality:	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1.	Generally: Having access to objective player performance parameters live *during* match or practice is valuable for physical load monitoring (as opposed to afterwards).	1	2	3	4	5
2.	Generally: A computerized toolkit for individual athlete performance monitoring and training load planning *during* practice can be useful for personalized intervention (e.g. resting an athlete from specific drills or exercise at the end of a practice; adding extra physical load to an athlete that has underperformed).	1	2	3	4	5
3.	Metrix potentially improves the objective monitoring of players' physical load *during* match or practice.	1	2	3	4	5
4.	Metrix potentially enhances the individualization of training programs *during* practice (e.g. to the player himself or playing position on the field).	1	2	3	4	5
5.	Metrix is useful for evaluating and identifying suitable training drills *during* a training session.	1	2	3	4	5
6.	Access to video playback of events (coupled to specific load monitoring events) is useful *during* match or practice (e.g. show the video of player X when sprinting over 20 meters).	1	2	3	4	5
	Design:					
1.	The application has a user friendly interface.	1	2	3	4	5
2.	The physical parameters of players are neatly presented.	1	2	3	4	5
3.	The progress bars are easy to comprehend.	1	2	3	4	5
	Overall:					
1.	This toolkit potentially has impact on individual training load monitoring and intervention.	1	2	3	4	5
2.	We will use this toolkit on a daily basis if provided.	1	2	3	4	5