MASTER

Visual analysis of multi-agent systems
a case study

Reijrink, J.

Award date:
2016

Link to publication
Visual Analysis of Multi-Agent Systems - A case study

Master Thesis

J. Reijrink

Supervisors:
MSc. B. Postma
dr.ir. H.M.M. van de Wetering

Eindhoven, July 2016
Abstract

Multi-agent systems generate large amounts of logging data that, when properly understood, could be used to understand and subsequently improve their behaviour. A multi-agent system is a computerized system that consists of multiple interacting intelligent agents in an environment. Typically, the agents in the multi-agent system keep each other informed of their actions through some kind of communication. This communication ensures that the agents are aware of each other’s decisions. Because this occurs at a high frequency, a large amount of data is generated.

We developed a visual tool that provides insight into a large amount of parallel data generated by a multi-agent system that consists of a small number of agents. The tool supports the user in analysing the behaviour of the multi-agent system and the behaviour of each individual agent.

As a case study, we designed & implemented a visual tool for TechUnited. TechUnited is a multidisciplinary team of (former) students, PhD’s and employees of the Technical University of Eindhoven engaged in the development of robots. More specific, the development of football robots.

Each year there is a world championship for football robots called RoboCup [25]. A robot football team consists of multiple autonomous robots that communicate at a high frequency during the football match. The developed tool should give them a better understanding of the actual behaviour of the agents during a football match. This way, they can detect anomalies earlier and improve their team more rapidly.

To find a visual solution we start with a comprehensive problem analysis. In the problem analysis we defined the data structure of the software events and presented five tasks that the tool should support. In addition, we defined tasks and requirements for our case study, TechUnited.

Next, we searched the literature for existing visualisations of multi-agent systems. Existing visualisations of multi-agent systems are often based on a large amount of agents, in our case we do not have many agents. Therefore, we also looked at visual analytics solutions and sport visualizations in the literature.

Based on the data structure, tasks and literature we made a selection of three visual designs that could offer a solution; small multiples, parallel coordinate plots (PCP) and linked charts. After investigating each of them, we chose one of the three designs. Small multiples did not support enough of the presented tasks from the problem analysis. Both the PCP and linked charts design offered a good solution. The temporal order of the data was an important characteristic that the visual solution should support. The PCP can not properly deal with this aspect of the data, therefore the linked chart design was chosen.

The shortcoming of the linked charts design was comparing attribute values between agents. To improve this shortcoming, we introduced two additional charts; the categorical- and agent-chart.

Based on the linked chart design and the additional charts we implemented a fully functional prototype. The prototype contains extra features to support the additional requirements that were desired by TechUnited. The prototype also supports the derivation of new attributes from the existing data using simple data processing.
Finally, the prototype has been verified using two different methods; use case testing and a small user evaluation.

In the first method we presented a use case for each task of the problem analysis. Each use case had to be solved using the prototype. This would show that the prototype could be used for all types of multi-agent systems.

The second method to evaluate the prototype was a small user evaluation. Four TechUnited team members had to solve a number of assignments using the prototype.

The results of the two methods verified that the prototype offers a good solution. Unfortunately the two additional charts did not completely solve the shortcoming of linked charts. It is still difficult to compare attribute values between agents. A possible solution can be to support zooming on the x-axis, showing the data more detailed.

During the project we mainly focussed on TechUnited, but the tool can be used for all kinds of multi-agent systems. The only requirement is that the data satisfies the structure described in section 2.3. We believe that this tool can provide great insight into all kinds of multi-agent systems.
Acknowledgements

I would like to express my gratitude to my supervisors dr.ir. H.M.M. van de Wetering and MSc. B. Postma for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Robin Soetens, Koen Meessen and the other team members of TechUnited for making this project possible and participating in the user evaluation. Thanks are also due to the Algorithms and Visualization group of the Technical University Eindhoven for providing this project and guidance. I would also like to thank MSc. D.J.W.G.M. Dingen for providing daily help and testing the user evaluation. Finally, I would like to acknowledge the support provided by my girlfriend and family during the preparation of this master project.
# Contents

Contents \hspace{2cm} v

List of Figures \hspace{1cm} vii

List of Tables \hspace{1cm} ix

1 Introduction \hspace{1cm} 1

1.1 Results \hspace{2cm} 2

2 Problem analysis \hspace{1cm} 3

2.1 General problem definition \hspace{2cm} 4

2.2 Approach \hspace{2cm} 5

2.3 Data abstraction \hspace{2cm} 6

2.4 Task abstraction \hspace{2cm} 7

2.5 Case study: TechUnited \hspace{2cm} 8

3 Related Work \hspace{1cm} 10

4 Visual design \hspace{1cm} 16

4.1 Three alternatives \hspace{2cm} 17

4.2 Conclusion \hspace{2cm} 20

4.3 Additional enhancements \hspace{2cm} 21

5 Realization \hspace{1cm} 23

5.1 Layout \hspace{2cm} 24

5.2 Data configuration \hspace{2cm} 25

5.3 Derived attributes \hspace{2cm} 25

5.4 Charts \hspace{2cm} 27

5.5 Field visualisation \hspace{2cm} 28

5.6 Performance \hspace{2cm} 31

6 Results \hspace{1cm} 33

6.1 Use case 1: understanding behaviour over time \hspace{2cm} 34

6.2 Use case 2: find patterns in the software events \hspace{2cm} 35

6.3 Use case 3: find occurrences of a specific attribute value \hspace{2cm} 35

6.4 Use case 4: compare attributes of agents \hspace{2cm} 37

6.5 Summary \hspace{2cm} 40

7 Evaluation \hspace{1cm} 41

7.1 Design \hspace{2cm} 41

7.2 Set-up \hspace{2cm} 42

7.3 Results \hspace{2cm} 42
List of Figures

2.1 Conceptual system overview with three moving agents that communicate with each other. A base station captures and stores all communicated software events to create a visualization. ................................................................. 4

2.2 The visual analysis process consists of interaction between the system, data, visualization and the expert. The expert gains knowledge which is used to improve the system. ................................................................. 5

2.3 Three-part analysis framework[15] that helps with design choices for a visual analytics tool in a systematic way. ............................................................. 5

2.4 (l) Front view of a turtle; (r) Example of a match played by TechUnited. ................. 8

3.1 Bluetooth scatternet of multiple agents. ................................................................. 10

3.2 Visualization of real-world telecom data using three different scaling algorithms, some customers are coloured and labelled so they can be traced through the different visualizations. ................................................................. 11

3.3 Visual solution from Visualization Analysis and Design[15]: (l) Streamgraph[2]; (m) Parallel Coordinate Plot (PCP)[7]; (r) Exploratory Data Visualizer[26]. ................. 11

3.4 Best visual solutions according to categorisation scheme [1]: (l) ThemeRiver[9]; (m) TimeWheel[22]; (r) SimVis[5]. ................................................................. 12

3.5 (l) Classification of visualization techniques [12]; (r) Polaris [21] visualisation. ........ 13

3.6 Visualisation for exploring and understanding spatio-temporal and multivariate patterns. ................................................................. 13

3.7 SoccerStories user interface: (1) game timeline, (2) display of selected thumbnail on field, (3) details of the phase (4) thumbnail overview (5) generated text of phase. 14

3.8 Visual analytics to help soccer analysts find the most important and interesting events in a match. ................................................................. 14

4.1 Conceptual design of small multiples. The x-axis shows the time and the y-axis shows an arbitrary attribute. Using mouse selection a subset of the data is selected in a chart, the data-points with the same time-frame are highlighted in the other charts. ................................................................. 17

4.2 Conceptual design of a parallel coordinate plot. Using mouse selection a filter can be created on an axis. The data outside the filter is faded to make the data inside the filter more visible. ................................................................. 18

4.3 Conceptual design of linked charts. Using mouse selection a subset of the data is selected in a chart, the data-points with the same time-frame are highlighted in the other charts. In the bottom chart only three agents are selected to be plotted. ....... 20

4.4 Conceptual design of categorical chart. ................................................................. 21

4.5 Conceptual design of agent chart. ................................................................. 22

5.1 Overview of the FoRoVis tool. ................................................................. 23

5.2 Docking a floating element into the main layout. ................................................................. 24
5.3 Available controls of a chart. ......................................................... 24
5.4 Example calculation of derived attribute $\alpha$ based on the $P(t)$, $F(t)$ and $S(t)$ boolean conditions. Based on these boolean conditions three ranges are identified. .......... 26
5.5 (l) Plot without range; (r) Plot with range activated. ......................... 27
5.6 Data filtered using the $y$-axes of charts. Only the role Attacker main and Defender main are preserved for turtle 2 and 6, the other data is filtered. ..................... 28
5.7 (l) Overview of the field visualization. Showing five turtles in the top, four opponents in the bottom and the ball in the center; (r) Perspective of turtle 6. The lines originating from the turtle show of which opponents the turtle is aware. .......... 29
5.8 (l) Show the turtles that see the ball at this location; (r) History lines of turtle 6 moving towards the enemy goal. .............................................. 30
5.9 Three proposals of a merged opponent. ......................................... 30
5.10 Example of an execution of the data-points reduction algorithm. ........ 32
6.1 (l) Derived attribute that shows successful and failed shots at goal; (r) Plot of the derive goal attribute. The plot shows that turtles two and five scored at the same time. ................................................................. 33
6.2 (l) Improved version of the derived goal attribute; (r) Plot of the improved derived goal attribute. Only turtle five scored a goal at time-frame 7,800. ....................... 34
6.3 Correlation between CPU-load (top chart) and ball possession (bottom chart). The right figure shows the same charts with a filter applied when the team has ball possession, this shows the CPU-load peaks in the top chart more clearly. ............. 35
6.4 Categorical-chart that shows the referee decisions over time, where 11 throw-ins for the magenta team. ...................................................... 36
6.5 A filter is applied on the throw-in of magenta (left chart). On the right chart only refbox 1 remained by applying a filter. The turtle that has refbox 1 is going to execute the throw-in. ................................................................. 36
6.6 (l) Derived attribute for throw-ins; (r) Resulting data of the throw-in attribute. ... 37
6.7 The scatter-charts shown the CPU-load of all turtles. Turtle 5 has a constant high CPU-load in comparison with the other turtles. ................................. 37
6.8 (l) Derived attribute that shows when an illegal move occurred. An illegal move takes place when a turtle performs a move with the ball instead of a dribble; (r) Resulting data of the illegal move attribute. ......................................................... 38
6.9 Both charts show the RoleID attribute. The bottom chart has a filter applied on the main attacker. As a result the top chart also only shows the main attacker. It seems there was a main attacker at all times. ................................. 39
6.10 Shows the moments each turtle was the main attacker during the match. .... 39
7.1 Two different concepts to represent an opponent on the field visualisation. .... 42
7.2 It is difficult to see if there is a gap in the Attacker Main role (Role Id = 1). ... 43
## List of Tables

4.1 The compatibility of the solutions for the presented problem definitions. $SM =$ Small Multiples, $PCP =$ Parallel Coordinate Plot, $LC =$ Linked Charts .......................... 20

5.1 Derived attribute calculation of shots on goal. $P(t)$ holds when the turtle has the ball and shoots at the goal. $F(t)$ holds if the referee gives a stop signal. $S(t)$ holds if the referee gives a goal signal. ......................................................... 26

6.1 Summary of the performance of the tool for each task. ........................................ 40

7.1 Answers to the first five questions of the questionnaire. ........................................ 43

A.1 Data structure of the data send by turtle. ......................................................... 50
Chapter 1

Introduction

Multi-agent systems generate large amounts of logging data that, when properly understood, could be used to understand and subsequently improve their behaviour. A multi-agent system is a computerized system that consists of multiple interacting intelligent agents in an environment. In our case, these agents are autonomous robotic systems. Multi-agent systems are used to solve problems that can be difficult or impossible to solve by a single agent. Sometimes, a multi-agent system is referred to as a parallel system.

Typically, the agents in the multi-agent system keep each other informed of their actions through some kind of communication. This communication ensures that the agents are aware of each other’s decisions and status. Often, this communication data is saved into a log and used for analysis of the agents afterwards. Because the communication occurs at a high frequency, a large amount of data is generated.

We develop a visual tool that provides insight into a large amount of parallel data generated by a multi-agent system that consists of a small number of agents. The tool supports the user in analysing the behaviour of the multi-agent system and the behaviour of each individual agent.

We use Visual analytics to provide this support. The European website of visual analytics[3] presents the definition of visual analytics as: "Visual Analytics methods allow decision makers to combine their human flexibility, creativity, and background knowledge with the enormous storage and processing capacities of today's computers to gain insight into complex problems".

As a case study, we design & implement a visual tool for TechUnited. TechUnited is a multidisciplinary team of (former) students, PhD’s and employees of the Technical University of Eindhoven engaged in the development of robots. More specific, the development of football robots.

Each year there is a world championship for football robots called RoboCup [25]. The purpose of this international robots competition is to promote robotics and artificial intelligence research. A robot football team consists of multiple autonomous robots that communicate at a high frequency during the football match.

All communication data is stored and currently only used to replay the football match. The developed tool should give them a better understanding of the actual behaviour of the agents during a football match. This way, they can detect anomalies earlier and improve their team more rapidly.

In this report we start with a comprehensive problem analysis. Followed by exploring the related work. Based on these findings we present three visual designs that could offer a solution. Next, one of the designs is chosen and turned into a fully working prototype. Finally, the prototype is evaluated using various use cases and through a user evaluation, carried out by team members of TechUnited.
1.1 Results

We have developed a visual analytics tool for TechUnited. They supported us during the development process and are very happy with the result. The tool allows the TechUnited team to gain new insights into their multi-agent system that were previously hidden. Now they can better analyse matches during a tournament and make quick improvements to their agents. We expect that the design of this tool could also help other experts to gain new insight into their multi-agent system.

TechUnited has already used the tool during the world championship of 2016 in Leipzig. They mentioned that the tool helped finding an error in the calibration of the Kinect and Omnivision. Because there was an error in the calibration the ball was not recognised correctly. They were able to identify this problem using the tool. TechUnited won the championship of 2016 and we are very pleased to hear that the tool was able to support them.
Chapter 2

Problem analysis

It can be tricky to get a good grip on systems that run in parallel. Typically, these systems keep each other informed of their actions through some kind of communication. The communication can be seen as a stream of software events, where each single system creates events over a period of time. In the context of this project a single system is called an agent, where multiple agents together form a multi-agent system. Some examples of multi-agent systems are:

- Flying drones that explore an area together. Each drone is an agent, together they form a multi-agent system. The drones have to communicate in order to search the area efficiently, where the communication is a stream of software events.

- Autonomous vehicles that transport people safely and quickly, where each vehicle is an agent and together they form a multi-agent system. A vehicle has to be aware of the decisions, issues and trajectories of other vehicles in the area. This is achieved because each vehicle continuously sends software events to all vehicles within a specific range. With this information decisions can be taken with respect to safety and possible alternative routes.

Software events can be used to get more insight into the behaviour of an agent and of the multi-agent system as a whole. We distinguish two types of attributes in software events: functional and diagnostic attributes.

The functional attributes are required for the functionality of the multi-agent system, without it the system cannot function properly. An example of a functional attribute is the location of a drone, if drones are not aware of each other’s location they are unable to explore an area together.

The diagnostic attributes are not required for the functionality of the system but are mainly used for analysis of an agent and the system as a whole. An example of a diagnostic attribute is the CPU load of a drone, without this attribute the system would still function properly.

Software events often contain information about the state of an agent and calculated values of shared attributes between agents. For example, flying drones keep each other informed of their current location, status and the regions that have been examined.

A conceptual overview of a multi-agent system is given in figure 2.1. The figure shows an example of how a multi-agent system might work, there are many possibilities of how a multi-agent system could work. In the figure multiple agents that are moving and communicating with each other are shown. The software events are captured by a separate system that stores all received data into a database. Finally, the data from the database is used to create a visualization to get more insight into the behaviour of the system. The set of software events in the database are also referred to as the log. Multiple logs can be used to compare different runs of the multi-agent system. The motivation for the use of a visualisation is presented in the next section.
2.1 General problem definition

The agents have to achieve common goals. In order to do so agents communicate by sending software events. These software events ensure that the agents are aware of each other’s decisions, this in turn has an impact on the decisions of an agent. Because the agents have to be informed of decisions by other agents they communicate at a high frequency, this results in a large amount of data.

By analysing the software event data, patterns and anomalies can be found. Also unexpected behaviour could be explained and knowledge of the system is gathered. As a result, the multi-agent system can be improved.

It is very difficult to analyse large amounts of data without the support of a tool. Automated analysis tools are often insufficient because they are unable to find patterns that are unknown. In order to find these unknown patterns in the data the support of a domain expert is required. This expert has domain knowledge that can be used to find patterns that would not have been discovered by automated analysis tools. Also, the domain expert knows which anomalies are interesting and which are not. Unfortunately, a domain expert alone is not enough. He needs to be supported by a tool that provides insight into the large amount of data. This tool should enable the domain expert to search for patterns and anomalies in the data. **Visual analytics** is an excellent solution to provide this support.

The general problem we want to solve is **Which visual analytics solution is most useful to support the domain expert with the analyses of the behaviour of agents and the multi-agent system as a whole?**
CHAPTER 2. PROBLEM ANALYSIS

2.2 Approach

In chapter 2.2 of the book *Solving problems with visual analytics*\(^{[13]}\) an abstract overview of the different stages and their transitions in the visual analytics process is presented. The overview is characterised through interaction between data, visualisations, models about the data, and the expert.

We use a derived version of the presented process, which is shown in figure 2.2. It shows a system that generates data, which represents the multi-agent system. This data is used to present an interactive visualization to the domain expert. This expert interprets the visualization and interacts where desired by browsing, zooming, selecting, etcetera. The expert can configure new attributes, which are derived based on the existing data. Finally, the knowledge gathered from the visualization can be used to improve the system. The improvement line is dotted because the improvements on the system are not necessarily executed immediately during the process.

The book *Visualization Analysis and Design*\(^{[15]}\) presents an analysis framework that helps with design choices for a visual analytics tool in a systematic way. Figure 2.3 shows a simple overview of their framework, which consists of three questions; what data does the user see, why does the user intend to use the visualization tool, and how to visualize the data & interact with it? By answering the first question, the semantics of the data are identified. The second question is designed to determine the tasks that the visual solution should support. The final question transforms the answers of the first two questions into a visual design.

We will use this framework to find a visual solution for the presented problem. In the next two sections we will abstract the semantics of the data and the tasks that the tool should support. Later, in chapter 4, we use these abstractions to find a visual solution.

Figure 2.2: The visual analysis process consists of interaction between the system, data, visualization and the expert. The expert gains knowledge which is used to improve the system.

Figure 2.3: Three-part analysis framework\(^{[15]}\) that helps with design choices for a visual analytics tool in a systematic way.
2.3 Data abstraction

In this section we will answer the question: what data does the user see? Chapter 2 of [15] defines several types of datasets and attributes of data that can be visualized. We use this chapter to identify the structure of software events in the scope of this project.

Software events consist of three components: time, source and attributes. The time component describes the creation time of the software event. Using this time component software events can be sequentially ordered. The source identifies the agent that created the software event. The combination of time and source are unique in the complete dataset. This is because an agent can only create one software event at a time. As a result, each software event of an agent has a unique time value. Finally, a software event contains attributes. As described before, attributes come in two flavours; functional and diagnostic attributes. An attribute can be of various types:

- Geographical; an attribute value with geographical meaning.
- Categorical; an attribute value in some known finite set.
- Ordinal, just like categorical; an attribute value in some known finite set. The values have a well-defined ordering and can therefore be ordered and compared.
- Quantitative, this represents numerical values that can be ordered and compared.

The ordering direction of the data is sequential per agent, sorted on the time component of each software event. The following notation is used to represent a list of all software events of an agent:

\[ s(a) = (e_{t_0} | e_{t_1} | ... | e_{t_n}) \]

Where \( a \) is an agent and \( e_{t_i} \) represents the software event of agent \( a \) at time \( t_i \) with \( t_i < t_j \) if \( i < j \). The notation to represent a single software event of an agent at a given time:

\[ s(a,t) = e_t \]

We assume that the dataset contains a small number of agents, at most ten agents. There is no limit to the number of attributes, but generally it will remain around a few dozen. Because the agents communicate at a high frequency there will be a lot of software events, which will result in a large dataset.

We focus on visualizing the captured data afterwards (offline), but of course it is also interesting to present a real-time visualization (online) that directly shows the behaviour of the agents and the multi-agent system as a whole.
2.4 Task abstraction

In this section we will answer the question: why does the user intend to use the visualization tool? Chapter 3 of [15] defines several tasks for which a visualisation can be used, for instance; discover or compare. We use that chapter to identify tasks that the tool should support based on the general problem.

The goal of this project is to create a visual analytics tool that supports the domain expert with the analysis of the behaviour of agents and the multi-agent system as a whole. Chapter 3 of [15] identifies three task groups; analyse, search and query. For each group multiple tasks are stated. We use these groups to identify the tasks that our solution must support.

Based on the analyse group the solution should support the discover and derive tasks. Based on the search group the solution should support the explore and locate tasks. Finally, based on the query group the solution should support the compare task. So, the visualization should allow the expert to discover, explore, locate, compare and derive.

In this way, the expert can determine anomalies like invalid behaviour, trends or outliers from one agent or between multiple agents. If, for example with the drones that explore an area together, there must always be exactly one leader, using the visualization it must be possible to see whether this holds during the complete runtime. The new insight gathered from the visualization gives the ability to improve the agents and make a more efficient multi-agent system.

The presented tasks (discover, explore, locate, compare and derive) are defined too generic. To make the tasks more concrete the tool should, in addition to basic functionalities of any visual analytics tool such as plotting an attribute, support the expert to:

T1. understand behaviour of software events over time (discover)
T2. find patterns in the software events (explore)
T3. find software events that have a specific attribute value (locate)
T4. compare attributes of agents with each other: (compare)
   T4-1. invalid attribute values
   T4-2. missing attribute values
   T4-3. duplicate attribute values
T5. derive new attributes using simple data processing (derive)
CHAPTER 2. PROBLEM ANALYSIS

2.5 Case study: TechUnited

As a case study of the visualisation we use a dataset of TechUnited. The football team of TechUnited has five autonomous robots that communicate with each other at a high frequency, this can be seen as a multi-agent system. These agents work together to achieve a goal and, eventually, win the football match.

A robot football match has a large overlap with a human football match. The match is played with two teams against each other, each team consists of five robots. Additional robots can sit on the bench in case one of the robots on the field breaks or is not functioning properly.

Each agent (also referred to as turtle by TechUnited) has its own camera to identify opponents and the ball. This results in deviations between turtles in the estimated location of an opponent or the ball. The location of an opponent can also be communicated by team mates. Based on the location of team mates, opponents and the ball a turtle chooses its action.

A turtle of TechUnited is shown on the left of figure 2.4. The right side of figure 2.4 gives an impression of a robot football match. A team always plays with one of two colors; magenta or cyan. The other team plays with the other color. The team color is visible on each turtle, as shown in figure 2.4.

There is a referee on the sideline. This is a computer controlled by a human that transmits game commands to the robots in the field. Examples of these commands are kick-off, throw-in or goal scored. There can also be a human referee on the field which communicates with the human behind the referee computer.

TechUnited has defined various roles for their turtles such as goalkeeper and main attacker. Each agent in the field has a role, a role can only be fulfilled by one agent at a time. All this information, and more, is communicated between the agents at a high frequency.

During the match all communication data is collected at approximately 10 Hz, at the end of the match it is stored into a large matrix. The current data that is communicated between agents is shown in appendix A. The appendix shows the attributes of a software event at any given time sent by the agents. The software events sent by the agents of TechUnited have the same structure as described in section 2.3. The software events of TechUnited contains attributes such as; the current role of the turtle (RoleID), does the turtle have ball possession (CPB) or which opponent has the ball (Opponent with ball).

Figure 2.4: (l) Front view of a turtle; (r) Example of a match played by TechUnited.
At the moment TechUnited has no clear insight into the decisions and status of the agents during a match. They would like to have a tool that gives a quick understanding what went wrong in a match. It is very important that the tool shows the errors of a match fast without too much configuration and searching. This is necessary to them because the software on the agents is often adapted between the matches in a tournament, so the tool should show problems with agents at a glance or allow quick identification of a problem using analysis.

Together with TechUnited we have defined additional tasks where they would like to have additional support. These tasks are listed below with some motivating examples.

TU1. Understanding events over time
   *Shot on goal; Free kick; Substitution of an agent.*

TU2. Find patterns between event/agents
   *A long pass is more often followed by a shot on goal than after a dribble action.*

TU3. Find interesting events/situations
   *Absence of the keeper; The target location of the agent is at an impossible location.*

TU4. Compare agents
   *What agent has substantially more loss of ball possession?*

TU5. Understanding events/situations in space
   *Poor communication in the north-west corner of the field.*

TU6. Benchmarking
   *How well does the world view of the agents reflect the reality?*

TU7. Tuning
   *Does the agent take the right decisions?*

TU8. Find patterns over multiple matches
   *Compare ball handling between several matches.*

In this project we will find a visual solution for the task TU1 up to TU5. Tasks TU6 up to TU8 fall outside the scope of this project because it is not feasible within the set time frame. Also, the first five tasks are more in line with multi-agent systems in general. Hopefully, this allows the presented solution to be transferable to various multi-agent systems.

Besides the tasks we also present some additional requirements for the implementation of the design. These requirements are based on the practical needs of TechUnited and will make the tool more convenient for them to use.

R1. Rapid analysis; typically between matches in a tournament.

R2. Handling of special/invalid attribute values.

R3. Flexibility; support for future expansions of the dataset and changes of metrics such as the field size or number of turtles.
Chapter 3

Related Work

We have analysed the structure of the data (what) and defined tasks that the visual solution should support (why). The next step is answering the how, so finding a visual solution. Before we can do this, we must know what visual solutions already exist. In this chapter we look at existing visual solutions and methods that help to find them. In the next chapter we will, based on the what, why and findings of this chapter, answer the how question.

A lot of visualizations relating multi-agent systems exist. In contrast to robot football, these visualizations feature a large number of agents. This results in a completely different visualization than the solution we are looking for. The visualisation often gives an overview of the relations or similarities between the agents.

For example, in the paper AgentViz: A Visualization System for Mobile Agents\(^4\) a visualization tool is presented that supports developers of mobile agent-based systems to get insight into the parallel behaviour of the agents that operate over large network topologies. The resulting visualization is shown in figure 3.1. The figure shows which agents are connected with each other over Bluetooth. Each agent is shown as a node, where red nodes identify the ‘master’ nodes.

Another visualization of a multi-agent system is presented in Multi-agent visualisation based on multivariate data\(^{19}\). In this paper the number of dimensions of the multivariate data is reduced. The agent similarities are shown on a lower-dimensional space through multi-dimensional scaling. The visualisation is examined using real-world telecom data of 90,000 calls by 100 customers.

The visualization is shown in figure 3.2. The figure shows the data being scaled by three different algorithms, some customers are coloured and labelled so they can be traced through the different visualizations. They conclude that each algorithm has its own strength to answer particular questions.

Figure 3.1: Bluetooth scatternet of multiple agents.
These existing visualizations are powerful to get insight into the overall behaviour of a large group of agents. In our case we do not have a large number of agents and we want to be able to investigate agents more comprehensively. In order to do this we make use of visual analytics.

We used [15] to identify the structure of the data and define tasks the tool should support. It also provides a lot of visualisations and describes, for each visualisation, what data structure and tasks it supports. Using the data and task abstractions of the previous chapter we found three visualisations that match our data and tasks, the visualisations are shown in figure 3.3.

The first visualisation in the figure shows a streamgraph that shows music that was listened to over time. A streamgraph is a type of stacked graph which is displaced around a central axis. The streamgraph supports multidimensional data that contains a key attribute (time) which represents the central axis. A streamgraph can be used to find trends and distributions over time.

The second visualisation in the figure shows a parallel Coordinate Plot (PCP). A PCP is used to visualise many quantitative attributes at once using multiple axes that are placed in parallel. A PCP can be used to find trends, outliers, extremes and correlation.

The final visualisation in the figure shows the Exploratory Data Visualizer. The visualisation is based on linked highlighting between views and can be used for multivariate data. The views can consist of various types of charts, for example, bar-charts, scatter-plots or histograms. The exploratory data visualizer can be used for analysis and exploration of the dataset.

Figure 3.3: Visual solution from Visualization Analysis and Design [15]. (l) Streamgraph; (m) Parallel Coordinate Plot (PCP); (r) Exploratory Data Visualizer.
The paper *Visualizing time-oriented data - A systematic view* \(^\text{[1]}\) presents a systematic view on the various methods for visualizing time-oriented data. They also present a categorisation scheme for visualizing time-oriented data. The scheme considers three main elements; time, data and representation. This categorisation scheme is connected with nine different visualisations for time-oriented data. The scheme shows which visualizations fit best with the presented data. For our data the visualisations in figure 3.4 are the best solutions according to the scheme, this is based on the structure and representation of our data.

The first visualization is called *ThemeRiver* \(^\text{[9]}\) and shows a static representation of changes in document collections. It is the same type of visualisation as the streamgraph mentioned before.

The second visualization is called *TimeWheel* \(^\text{[22]}\) and is an axes-based visualization of multivariate data where the center axis is based on the temporal aspect of the data. This visualisation is based on the parallel coordinate plot visualisation.

The last visualization is called *SimVis* \(^\text{[5]}\) and shows multiple views to facilitate flow visualization. This visualisation has a lot of overlap with the linked highlighting between views of the exploratory data visualizer.

The outcome of the categorisation scheme is very similar to the visual solutions of \(^\text{[15]}\).

*Information Visualization and Visual Data Mining* \(^\text{[12]}\) proposes yet another classification to find a visualisation. The classification is based on three elements; data type, visualization technique and the interaction and distortion technique. They present a classification of information visualization techniques, shown on the left of figure 3.5. This classification can be used to gather information about the data, visualization technique and the interaction and distortion technique. The classification helps with finding the right visualization to a given problem and dataset.

Based on multidimensional data, interactive linking and brushing, interactive filtering and the use of 2D displays, such as bar charts and x-y plots, the *Polaris* \(^\text{[21]}\) visualisation seems to be the best fit for our problem and data structure using the classification. The Polaris visualisation is shown on the right of figure 3.5. The main part of the Polaris tool consists of a *small multiples* visualization. A small multiples visualization consists of a series of similar charts where each chart has the same scale and axes, this makes them easy to compare. The difference between the charts is the data presented, each chart shows a different partition of the dataset.
Finally, a visualisation for exploring and understanding spatio-temporal and multivariate patterns is presented in *A Visualization System for Space-Time and Multivariate Patterns* [8]. [8] present a tool that helps analysts investigate complex patterns across multivariate, spatial and temporal dimensions via clustering, sorting and visualising.

The tool is shown in figure 3.6 and consists of a layout containing various types of charts that are linked with each other through data filtering. The type of charts involve a self-organizing map, parallel coordinate plot, reorderable matrices, geographic small multiple display and a two-dimensional cartographic color design method.

**Figure 3.6:** Visualisation for exploring and understanding spatio-temporal and multivariate patterns.
CHAPTER 3. RELATED WORK

Figure 3.7: SoccerStories user interface: (1) game timeline, (2) display of selected thumbnail on field, (3) details of the phase (4) thumbnail overview (5) generated text of phase.

Because the case study involves football, we also investigated sport visualisations used for analysis. Sport visualisations use the same kind of events as describes in chapter 2.3, making these visualizations a potential solution for our problem.

In SoccerStories: A Kick-off for Visual Soccer Analysis\textsuperscript{[16]} visual analytics is used to create stories from a football match. These stories are based on the different football phases of the match and the locations of the ball and players. The phase is selected using thumbnails, each thumbnail represents a single phase in the match. Next a selected phase is displayed on the field. Based on the details of this phase a descriptive text is generated. The tool is shown in figure 3.7.

Feature-Driven Visual Analytics of Soccer Data\textsuperscript{[18]} presents a visual analytics tool to help analysts find the most important and interesting events of a match. The tool is shown in figure 3.8. It is separated into multiple dynamic views. It supports the analysis of a single player and multiple players. Just like the previous football visualisation, it supports multiple phases that can be selected in the top right panel. Besides the main field visualisation it also supports other charts like a parallel coordinate plot, line chart and horizon-graph (shown in the right bottom). Using all these charts comparisons between players and attributes features can be made.

Figure 3.8: Visual analytics to help soccer analysts find the most important and interesting events in a match.
In sports visualisation, it is noticeable that the visual analysis tools often use multiple linked views. Where the field is the main view. Other views contain charts such as parallel coordinate plot or small multiples.

In this chapter we started by examining the literature for existing visualisations of multi-agent systems. In contrast to robot football, these visualizations feature a large number of agents. This results in a completely different visualization than the solution we are looking for. The visualisation often gives an overview of the relations or similarities between the agents.

In our case we do not have a large number of agents and we want to be able to investigate agents more comprehensively. In order to do so we make use of visual analytics. Therefore, we investigated several classification schemes that help finding a visual analytics solution. Using the schemes, data structure and defined tasks we found four types of designs that could offer a possible solution: streamgraph, parallel coordinate plot, small multiples and linked charts. There were more types of visualizations, but these were mainly based on these four designs.

Finally, because the case study involves football, we examined sports visualisation. Also, sport visualisations use the same kind of events as describes in chapter 2.3. The sport visualisations often contain multiple linked views. Where the field is the main view and other views contain charts such as parallel coordinate plot or small multiples. We have seen this type of visualisation also during the examination of the visual analysis tools, such as the Polaris [21] or the tool presented in [8].
Chapter 4

Visual design

In this chapter we will answer the question; how to visualize the data & interact with it? Meaning, the data and task abstractions are transformed into a visual solution. We have already defined the data structure and determined which tasks the tool should support. We will use the data structure and tasks to find the visual solution. In Chapter 3 we investigated existing visualizations in the literature. We will use these existing visualisations as the starting point for finding the solution.

In this design phase, we will ignore the geographical data-type because the visual solution for this data-type will be different for each multi-agent system. However, we will present a potential visual solution in chapter 5.5.

We only use the visual tasks defined in chapter 2.4, so only tasks $T_1$ up to $T_4$. Task $T_5$ describes the extension of the dataset based on existing attributes and has nothing to do with the choice for a visual design. The implementation of task $T_5$ will be discussed in chapter 5.3.

Based on the literature we found four types of designs that could offer a possible solution; streamgraph, parallel coordinate plot, small multiples and linked charts. There were more types of visualizations, but these were mainly based on these four designs. We want to focus on the basic designs and continue from there.

The streamgraph does not provide a good enough solution because the attribute values can only be compared relatively. There is no visible y-axis, therefore the actual value of an attribute is not known and can only be compared against other values by its relative size. That is why we will only continue with three designs; parallel coordinate plot, small multiples and linked charts.

Each design will be discussed separately in the next sections. For each design a concept is developed that gives an impression of the visualization. For each design the pros and cons are determined with respect to the presented tasks and data structure. Then methods that could solve some of the cons from the design are presented, if any. Based on these findings one design is chosen that will be implemented into a working prototype.
4.1 Three alternatives

4.1.1 Small multiples

The first potential solution is a small multiple visualization. The small multiple visualization consists of a series of similar charts. Each chart has the same scale and axes, this makes them easy to compare. The difference between the charts is the data presented, each chart shows a different partition of the dataset.

In the paper Small Multiples, Large Singles: A New Approach for Visual Data Exploration [23] a very intuitive tool using small multiples is presented. The data is presented to the user through an overview of small multiples. After a small multiple is selected, the chart is enlarged in order for the user to analyse. From this state, a new overview of small multiples is presented to the user. This enables the user to easily explore the data using various charts and partitions.

A concept design of small multiples is shown in figure 4.1. We decided to show the time on the x-axis because it is a fundamental aspect of the presented data. An arbitrary attribute is shown in the y-axis. The data is partitioned over agents; each chart shows the dataset of one agent. This allows the agents to be easily compared.

The mouse selection used on the small multiples in figure 4.1 is referred to as linking and brushing. A subset of the data is selected in one chart using the mouse. The data-points with the same time-frame as the selection are highlighted in the other charts. This way, data-points that have the same time-frame can easily be compared between agents.

The advantage of small multiples is that they are relatively easy to understand and are very organized. The attribute values of agents can be compared at a glance.

The main downside of the concept is that only one attribute can be compared at a time. Two attributes can be compared if the time axis is replaced with an attribute, but this is still very sparse. A small multiples visualisation does not necessarily consists of scatter-plots, it could show any type of visual representation. But a scatter-plot shows the pattern of an attribute very clearly over time, therefore we decided to use scatter-plots.

Small multiples can visualise a lot of agents simultaneously. If the dataset becomes too big the charts are no longer clearly readable because each plot is small.

![Figure 4.1: Conceptual design of small multiples. The x-axis shows the time and the y-axis shows an arbitrary attribute. Using mouse selection a subset of the data is selected in a chart, the data-points with the same time-frame are highlighted in the other charts.](image-url)
CHAPTER 4. VISUAL DESIGN

Most of the tasks can be solved using small multiples. Understanding behaviour over time (T1), finding patterns (T2) and specific attribute values (T3) is not a problem with small multiples. However, finding missing (T4-2) or duplicate (T4-3) attribute values between the agents is quite difficult.

4.1.2 Parallel coordinate plot

The second proposal for a solution is a Parallel Coordinate Plot (PCP). PCPs are used to visualise multidimensional data. Each dimension of the data corresponds to a vertical axis. Each data element is displayed as a series of connected points along the vertical axes.

The book *Parallel coordinates* \(^{[10]}\) presents a more elaborate explanation of parallel coordinate plots.

A concept design is shown in figure 4.2. The four vertical axes each show a dimension of the data. Each software event is shown as a series of connected points along the four axes, where the color of the series depends on the corresponding agent of the software event.

Finally, the data can be filtered using mouse selection. By creating a filter on an axis the filtered data becomes more visible by fading out the other data.

PCPs are known to be difficult to understand. In *Interacting with parallel coordinates* \(^{[20]}\) it is argued that this is not necessarily true, but that many parallel coordinate implementations lack essential features. By implementing parallel coordinate browsers with better interaction tools the PCPs are easier to understand. Their study shows that interacting with PCPs is not so difficult as generally believed.

The power of PCPs is that all dimensions of the data can be compared at a glance. This makes it easy to compare all kinds of attribute values between agents. The drawback of all these dimensions is that the plot gets cluttered quickly, making it hard to see patterns.

There are many papers that solve the cluttering of a PCP. For example, *Cluster Visualization in Parallel Coordinates Using Curve Bundles* \(^{[14]}\) introduces curve bundles in PCPs. The curve bundle visually separates data clusters, combining data items of a cluster into a curve. This way the data items in a clusters are more combined, showing the general data structure of the dataset more clearly.

Figure 4.2: Conceptual design of a parallel coordinate plot. Using mouse selection a filter can be created on an axis. The data outside the filter is faded to make the data inside the filter more visible.
Orientation-Enhanced Parallel Coordinate Plots \cite{17} proposes a technique to improve the cluttering of PCPs by using the overall data structure of the dataset. They visually enhance the parts of each PCP line with respect to its slope, this way patterns and outliers are more distinguishable.

Although a PCP includes the complete dataset, it only contains a subset of connections between all data items. As a result, an important pattern between two dimensions may remain unnoticed if the axes are not configured in the right order.

A PCP cannot properly deal with the temporal aspect of the dataset, hence, the sequential order of data items is not visible. The time attribute of the data can be used as an axis on the PCP but this is still not sufficient because the order of the axes have to be set just right to be able to see anything relevant.

As a result, it is not possible to show the behaviour over time (T1). However, the visualization is good at finding patterns (T2) and specific attribute values (T3). Comparing invalid (T4-1) and missing (T4-3) attribute values between different agents is easy using a PCP. Because the PCP cannot properly deal with the temporal aspect it is impossible to find duplicate (T4-3) attribute values at the same time-frame.

4.1.3 Linked charts

The final possible solution is linked charts. Linked charts look like small multiples in the sense that it also contains multiple charts. The difference is that the scale and axes do not have to be the same for each chart. Each chart also doesn’t necessarily show a partition of the data, it could basically show the complete dataset. The charts that can be used for linked charts can be any type of chart. This can be a regular scatter plot, PCP or even small multiples.

For example, the linked charts of the prototype developed in GeoAnalytics Visual Inquiry and Filtering Tools in Parallel Coordinates Plots \cite{11} contains a scatter plot, PCP and geographical map. This way, the user can easily compare different types of data.

A concept design is shown in figure 4.3. The axis of a chart can be any attribute in the dataset or time. A selection of agents that are plotted on a chart can be made, for example, in the lower chart of figure 4.3 the software events of agent 2 are not plotted.

Just like small multiples, all charts are linked with each other through mouse selection. The selected data-points are highlighted in all charts.

Linked charts allow the user to easily compare various attribute values between multiple agents. It also supports the use of all kind of charts. The downside of all these charts is that it takes some time to configure them all. The design can show a lot of data and patterns at one time, but if each plot shows all data it becomes cluttered.

Most of the general problem definitions can be solved with linked charts. However, finding duplicate (T4-3) attribute values between the agents is quite difficult.
4.2 Conclusion

Table 4.1 shows an overview of each design with respect to the defined tasks of chapter 2.4. Besides the support of the tasks it is also important that the characteristics of the design fits the data requirements. For example, a large dataset or the three types of data.

The fact that small multiples cannot handle as much data and does not perform very well on table 4.1 makes it clear that this is not the solution to the presented problem.

PCP and linked charts (LC) are fundamentally different and both offer a great solution in their own distinctive way. But as mentioned before, the temporal order of the data is an important characteristic. A PCP can not properly deal with the temporal aspect of the dataset. Therefore we choose linked charts as the best fit. The PCP could still be used as one of the charts for the linked charts design.

Linked charts still offers no complete solution, therefore in the next section two additional charts are presented to solve the problem with comparing agents for duplicate values.

<table>
<thead>
<tr>
<th>Task</th>
<th>SM</th>
<th>PCP</th>
<th>LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1. Understanding behaviour of software events over time</td>
<td>Good</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>T2. Find patterns in the software events</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>T3. Find software events that have a specific attribute value</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>T4. Compare different attributes of an agent</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>T4-1. Compare attributes of agents with each other (invalid)</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>T4-2. Compare attributes of agents with each other (missing)</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>T4-3. Compare attributes of agents with each other (duplicate)</td>
<td>Poor</td>
<td>Fair</td>
<td>Poor</td>
</tr>
<tr>
<td>Show time order of attribute(s)</td>
<td>Good</td>
<td>Poor</td>
<td>Good</td>
</tr>
</tbody>
</table>

Table 4.1: The compatibility of the solutions for the presented problem definitions.

SM = Small Multiples, PCP = Parallel Coordinate Plot, LC = Linked Charts
4.3 Additional enhancements

As mentioned in the previous conclusion, linked charts (with just regular scatter-plots) are not good at finding duplicate attribute values between different agents (T4-3). We could use a PCP as a chart for the linked charts, but a PCP is also not very good at solving this problem. Therefore we developed two new charts that will be used in the linked chart visualisation, each will be discussed in its own subsection.

4.3.1 Categorical chart

The first chart we developed is called the categorical-chart, the concept is shown in figure 4.4. This chart is developed to give a quick overview of overlapping attribute values between different agents at the same time-frame.

The categorical-chart can only support categorical and ordinal data, so it does not support quantitative data. The reason we allowed this limitation is because finding duplicate attribute values is mainly used on categorical and ordinal data. Because of the assumption of the data-type we can show each category on the y-axis and separate the chart area horizontally into a piece for each category. Each area of a category is again separated, again horizontally, by the number of agents. The data-items of each agents are shown in their own area within each category.

To find duplicates, simply look at the desired category and search for overlapping areas of different agents. For example in figure 4.4, agent 3 and 4 both have a duplicate attribute value (Cat. 1) for some time.

The advantage of this chart is that overlap in attribute values can be seen at a glance. A disadvantage of the chart is that it cannot handle a large number of agents together with a lot of different category values. However, in the presented data we do not have a large number of agents.

Figure 4.4: Conceptual design of categorical chart.
4.3.2 Agent chart

We also developed another chart called the *agent-chart*, the concept is shown in figure 4.5. This chart is developed to give a quick overview of the attribute values for each agent separately.

The agent-chart in the concept also shows categorical data. But by using a gradient scale color map it could also support quantitative data. In this chart the agents separate the y-axis. For each agent its own area is filled with its attribute values over time.

The agent-chart has a lot of overlap with the categorical-chart; instead of the attribute the agent is shown on the y-axis. Also the full height is used to represent a value because a turtle can only have one attribute value at a time.

As mentioned, the main purpose of the agent-chart is to give a quick overview of the attribute value development of a specific agent. But it can also be used to show duplicate attribute values between different agents. For example in figure 4.5, agent 3 and 4 both have a duplicate attribute value (Cat. 1) for some time.

We developed this second chart because using the categorical-chart it can be difficult to see the attribute value development over time for each agent separately. To show the attribute value development for each agent, a scatter-plot could be used. But as mentioned, a scatter-plot does not show overlap of attribute values. The advantage of this chart is that it shows the attribute value development over time at a glance for each individual agent.

We have developed two additional charts for finding duplicate attribute values between different agents. These additional charts will be part of the prototype, and evaluated as well.

![Agent Chart Concept](image)

**Figure 4.5:** Conceptual design of agent chart.
Chapter 5

Realization

We have developed a prototype for TechUnited, based on the linked charts design of previous chapter. This Football Robot Visualisation or FoRoVis in short is shown in figure 5.1. It contains the charts described in the previous chapter: scatter-, category- and agent-chart. We also added a line-chart because in some cases a line-chart could provide more information than a scatter-chart. In the figure the agent-chart is called the turtle-chart, this is because TechUnited refers to agents as turtles. The FoRoVis prototype also contains a visualisation of the football field.

Each chart is linked through horizontal mouse selection. The mouse selection rectangle has a bold line either on the utmost left or right side, this depends on the drag direction of the selection. The field visualisation shows the situation of the match at the time-frame of the bold line. This way, the situation on the field can be shown at a specific time-frame. The field visualisation also provides the ability to visualize the data of the entire mouse selection, this feature is described later in this chapter. A detailed description of the features of the field visualisation is described in section 5.5.

Figure 5.1: Overview of the FoRoVis tool.
Besides the basic functionalities of linked charts described in the previous chapter, the FoRoVis tool has a lot of extra features. The development of these features are mainly based on the additional requirements of TechUnited, presented in section 2.5. As mentioned the tool must be able to quickly show errors of a match without too much configuration and searching. Other functionalities were added because of obstacles we encountered during the development of the tool. Finally, certain features have been added to make the tool more convenient to use. In the next sections we will describe the most important functionalities that were added to the tool.

5.1 Layout

The tool has a dynamic layout of its charts that can be arranged as desired. The layout is based on a docking framework where each element (i.e. chart or field visualisation) can be docked into the layout or shown as a separate floating window. The size of each element can also be modified using the split-pane between two docked elements.

In figure 5.2 a floating turtle-chart is being docked below the scatter-chart. The height of the scatter-chart will be reduced to make space for the new turtle-chart.

A layout design can be saved for later use. This allows the user the create a pre-set of layouts that can easily be loaded at any time, saving a lot of configuration time. The feature is developed to speed up the analyses of turtles, this is requirement $R1$ of chapter 2.5.

Each element has five controls in the top right corner, these are shown in figure 5.3.

The first control can be used to enlarge the chart. If a chart is docked into the tool it can become small if there are many charts. By activating the enlarge functionality the element will be shown full screen, this can be convenient when a closer look is required.

The second control opens the element specific settings. For a chart element the plotted attribute or shown turtles can be configured.

The last three controls are used to minimize, undock or remove the element from the docking layout.
5.2 Data configuration

TechUnited uses Matlab as their main development tool, therefore the logging data of a match is
saved into a .mat file. This file contains a large matrix consisting out of doubles. Due to this data
structure, it is impossible to identify the attributes. Therefore we introduced a data-mapper to
identify each data attribute.

In the mapping each attribute is given a name, type and unit-type. If the type of the attribute
is categorical then the categories must be specified, where each category represents a unique
number. The range of the attribute can also be specified. The use of this range will be discussed
in section 5.4.

The mapping is saved to a JSON file, this way the tool can easily be distributed between team
members of TechUnited. If the structure of the matrix is changed the mapping can easily be
modified, no coding and recompiling required.

Because the mapping is dynamic the tool does not know what attributes must be used to draw
the components on the field visualisation, such as the positions of the turtles. To solve this we
have developed a configuration screen where the required attributes can be set. Also, the metrics
must be specified in the configuration screen, such as the size of the field or the size of the goals.
The required attributes in the configuration screen consists of field metrics and turtle, ball and
opponent positions.

The data-mapper and configuration screen are built to support the requirement $R3$; the expansions
of the dataset and changes of metrics.

5.3 Derived attributes

The dataset of TechUnited contains many, often straightforward, data attributes. For example
the current role (goalkeeper, defender, etc.) or CPU-load of a turtle. However, more complex
data may be desired to get the right insight; this could often be achieved by combining multiple
attributes. For example, to check whether a shot on the goal has succeeded, you have to combine
the shot type of the turtle and the game situation after the shot. So you have to check whether
the referee says a goal has been made immediately after the shot of a turtle, if this is the case
a goal was successful. This type of querying is possible with the tool, but it takes a lot of time,
and relevant details can easily be overlooked. To support the user with this type of querying we
introduced derived attributes.

A derived attribute $\alpha$ can have one of three values at time $t$:

$$\alpha(t) \in \{\text{None, Failed, Success}\}.$$

The resulting value of $\alpha$ for each $t$ depends on three boolean conditions, which are defined
by the user. The first boolean condition $P(t)$ describes the precondition. The other two boolean
conditions are postconditions; $F(t)$ describes the failed postcondition and $S(t)$ describes the success
postcondition. The boolean conditions are defined using simple boolean expressions over the
existing attributes in the dataset.

Using the boolean conditions we define non-overlapping ranges $[s_i, t_i]$ from left to right, see
figure 5.4. At the start $s_i$ of a range the precondition must hold. At the end $t_i$ of the range at
least one of the postconditions must hold and the precondition must not hold. So, for a range
$[s_i, t_i]$ the following holds:

$$\overline{P(s_i) \land (S(t_i) \lor F(t_i)) \land \neg P(t_i)}$$
CHAPTER 5. REALIZATION

P(t) F(t) S(t) t α

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failed</td>
<td>None</td>
<td>Success</td>
<td>None</td>
<td>Success</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Failed Success Success

0 1 2 3 4 5 6 7 8 9 10 11 12 13

Figure 5.4: Example calculation of derived attribute α based on the P(t), F(t) and S(t) boolean conditions. Based on these boolean conditions three ranges are identified.

To make the set of ranges unique, ranges start as soon as possible and last as short as possible. In figure 5.4 three ranges ([0, 3], [5, 8] and [11, 13]) are identified based on the boolean conditions. If range [5, 8] started later, for example [6, 8], it would still be a valid range. Therefore a range is required to start as soon as possible. Range [0, 3] could be extended to [0, 4] and still be a valid range. To prevent this we required that a range should be as short a possible. The introduction of these two additional requirements ensured that only a single set of ranges is valid for the derived attribute.

Based on the ranges the value of α(t) is set:

$$\alpha(t) = \begin{cases} 
\text{Success} & t \in [s_i, t_i] \land S(t_i) \\
\text{Failed} & t \in [s_i, t_i] \land F(t_i) \land \neg S(t_i) \\
\text{None} & \text{Otherwise}
\end{cases}$$

If t is an element of any range then α(t) is either Success or Failed, based on the postconditions. If t is not an element of any range then α(t) is set to None.

Figure 5.4 shows that α has value Failed during the first range because F_t_3 is true. During the second and third ranges α has value Success because the S_t_8 and S_t_13 are true. At the places were no ranges are present α has value None.

To clarify the calculation of a derived attribute we will present a simplified example. We want to derive a new attribute that shows whether a shot at the goal was successful. The precondition P(t) state that the turtle must have the ball and shoots at the goal. When the precondition is satisfied the value of the derived attribute should be successful if the ball is inside the goal, this information is given by the referee. So S(t) holds if the referee gives a goal signal. If the referee gives a stop signal, the attempt on the goal has failed. So F(t) holds if the referee gives a stop signal. The calculation of the derived attribute is shown in table 5.1. The turtle tries to score a goal twice, the first attempt is successful, the latter is not.

Table 5.1: Derived attribute calculation of shots on goal. P(t) holds when the turtle has the ball and shoots at the goal. F(t) holds if the referee gives a stop signal. S(t) holds if the referee gives a goal signal.

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has ball</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Shot at goal</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Referee decision</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>No</td>
<td>Goal</td>
<td>None</td>
<td>None</td>
<td>Stop</td>
</tr>
<tr>
<td>P(t)</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>F(t)</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>S(t)</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Goal (derived)</td>
<td>None</td>
<td>None</td>
<td>Success</td>
<td>Success</td>
<td>Success</td>
<td>None</td>
<td>Failed</td>
<td>Failed</td>
</tr>
</tbody>
</table>
With the derived attributes many situations can be analysed at a glance, saving a lot of time. This feature is based on task T5 of chapter 2.4: derive new attributes using simple data processing.

## 5.4 Charts

The specific functionality of each chart has been described in chapter 4. Besides the default functionalities the charts support some extra features which we describe in this section.

Based on requirement R2 we introduced ranges on data attributes. The scatter-plot uses the range of an attribute to set boundaries on the y-axis, which is shown in figure 5.5.

The left chart shows the data of the attribute without ranges. Because of the outliers at the bottom of the chart the remaining data becomes compressed and there is a lot of white-space.

The right chart shows the data with the range activated. Data that falls outside of the given range is displayed with red rectangles at the boundaries of the chart. Now the data inside the range is shown in more detail.

Based on task T3 all charts support filtering of the plotted data. The filtering helps the user to find and compare specific attribute values. By dragging the mouse vertically over the y-axis a filter is created. The data within the range of the filter is preserved, all other data is hidden. The data on all charts is updated based on the filters. A chart can have multiple filters. If a chart has multiple filters they will act like an OR filter. Filters between different charts act like AND filters.

For example in figure 5.6, there are two filters on the top left categorical-chart. This chart shows the RoleID attribute, one filter is on Defender main and the other on Attacker main. The software event of each turtle where the RoleID is neither Defender main nor Attacker main will be filtered out. On the top right turtle-chart there is a filter on turtles 2 and 6, so only their software events are shown. The final outcome is that the top left categorical-chart only shows when turtles 2 or 6 where an Attacker main or a Defender main. On the turtle-chart, the data-points of turtles 2 and 6 are only shown if they are an Attacker main or a Defender main. Finally, the scatter-chart in the bottom shows the CPU-load of turtles 2 and 6 when they were either Attacker main or Defender main.

![Figure 5.5: (l) Plot without range; (r) Plot with range activated.](image)
CHAPTER 5. REALIZATION

Figure 5.6: Data filtered using the y-axes of charts. Only the role Attacker main and Defender main are preserved for turtle 2 and 6, the other data is filtered.

5.5 Field visualisation

Besides a number of charts the tool also supports a field visualisation. The field visualisation consists of four main components.

The first component is the football field. Which is based on the actual metrics used during the match. These are set by using the configuration screen, which is discussed in section 5.2.

The second component of the field visualization are the glyphs of the turtles. In both the charts and the field visualisation each turtle has its own specific color, making them easily distinguishable from each other. Sometimes the colors are a bit darker in order to increase contrast. On the field visualisation, each turtle glyph has the color of the corresponding turtle color. The glyph is arrow-shaped with the corresponding turtle number in the center, the glyphs are shown in figure 5.7. The direction of the arrow depends on the orientation of the turtle. The turtle glyph is similar with the actual turtle shape, see figure 2.4 of chapter 2.5.

By hovering over a turtle using the mouse, the perspective of the turtle is shown. On the right side of figure 5.7 the perspective of turtle 6 is shown. The perspective of a turtle is indicated with lines and shows of which opponents the turtle is aware. The opponents of which the turtle is aware can come from his own view using camera recognition, but it can also originate from team-mates. The figure shows that turtle 6 does not know there is a goalkeeper at that time-frame.

The third component of the field visualization is the ball, which is shown as an orange circle on the field visualisation.

Each turtle tries to localize the ball. The ball has an orange or yellow color because, with image recognition, it is easier to recognise. When the turtle sees the ball itself it uses this location, else it uses the ball position communicated by its team-mates.

Because each turtle tries to identify opponents and the ball by itself using image recognition there can be many deviations in the localization. This means that the position of the ball and opponents can vary a lot. There can also be false-positives, which means that the identified ball or an opponent is not actually there.
To make the field visualisation more clear we try to merge the ball positions localized by the turtles.
If the ball positions have a lot of overlap, the average position is used to draw one ball on the field visualisation. In many cases, this ensures that there is only one ball on the field visualisation. If the distance between two ball positions differ too much they will not be merged. In that case multiple balls appear in the field visualisation.

By hovering over a ball using the mouse, the turtles that localised that ball at that position are shown. This is shown on the left of figure 5.8, only turtle 5 does not identify the ball at that location.

The last major component of the field visualisation is the drawing of the opponents. Just like ball positions the perceived position of an opponent can also differ a lot.

However, there are typically five opponents, and they are not distinguishable by a unique number. Sometimes the turtles can identify more than five opponents, this extra opponent is often the human referee on the field.

Because the identified locations of the opponents cannot be distinguished, the tool has to decide whether two locations express the same opponent or not. This is based on the same principle as the merging of the ball. If two positions have a lot of overlap they are merged together.

However, locations that originate from the same turtle are not merged. So, if one turtle sees two opponents very close to each other they are not merged.
CHAPTER 5. REALIZATION

Because there are multiple opponents drawn on the field visualisation we wanted a way to quickly identify how many turtles see the opponent at the position. If the position of the opponent is seen by all turtles, then it is almost certain that there is an opponent at that position. And exactly the opposite, if only one turtle sees the opponent at the position, then it is probably noise.

We developed two visualizations to represent an opponent, shown in the first two glyphs of figure 5.9. Both glyphs show which turtles see the opponent based on the fixed turtle colouring. The difference is that the first glyph has a fixed height to show the turtles that see the opponent. The second glyph separates the rectangle based on the amount of turtles that see the opponent.

We decided to use the first glyph of figure 5.9 because it is better to read. This is because you become accustomed to the fixed positions of each turtle in the glyph.

Finally, we proposed another version of the merged opponent glyph, shown on the right of figure 5.9. This glyph shows the inverse of the first glyph. We propose this glyph because it is calmer. In general, the turtles agree on the positions of the opponents. Therefore the first glyph will often show an abundance of rainbow colors. We will test both versions of the glyphs during the user evaluation.

Finally, the tool supports what we call history lines. This feature is based on task TU5; understanding events/situations in space. The history lines are based on the mouse selection. As mentioned earlier, the field visualisation only shows the situation on the field on the time-frame of the bold line. The history lines shows the path of the turtle over the entire mouse selection. Only data-items that are not removed using filters are shown. The history lines can be used to find geographical patterns. For example, at which position was the main attacker most during the match? An example of history lines is shown on the right of figure 5.8. At this example the history lines of turtle 6 are shown when moving towards the enemy goal.

Figure 5.8: (l) Show the turtles that see the ball at this location; (r) History lines of turtle 6 moving towards the enemy goal.

Figure 5.9: Three proposals of a merged opponent.
5.6 Performance

Plotting an attribute showing all turtles can easily result in 100,000 data-points on one single chart. We used the charts from the JavaFX library. The charts in this library cannot handle this amount of data-points.

We implemented a mouse selection system as described in chapter 4.1, but updating the selected data-points was very slow. Requirement $R1$ states that the tool has to be fast. Therefore the mouse selection has to be improved.

The selection is very slow because with each mouse selection on a chart the selected software events have to be found. Next, the selected software events must be highlighted on all charts. Because two arbitrary attributes can be plotted against each other the software events are not shown in an order. Therefore, all plotted software events have to be replotted every time a selection is made, which can take some time if this has to be done for each chart.

To improve the mouse selection we decided to set a fixed attribute on the x-axis, the time-frame attribute. This way, two arbitrary attributes cannot be plotted against each other any more. Since the time aspect plays an important role the time will primarily be put on the x-axis.

By using a fixed x-axis the mouse selection between the charts can be done without a hitch. This is because software events are always plotted in the same order. Thus, the mouse selection always selects a range of software events in a sequential order. In this way, we can draw a rectangle in each chart on top of the data, and do not need to re-plot the data on the charts. The rectangle starts at the first software event of the selected range and ends at the last selected software event. Because of the logical order, we know that we have always selected the software in between the rectangle. With this method, software events can be selected very fast over a period of time. Where without the fixed x-axis, the user would have to wait every-time a selection is made.

Another performance improvement is the reduction of data-points that are being plotted. The reduction is based on the size of a chart. A small chart has to show less data-points than a full-screen chart. This is because a small chart has less space to plot the data.

As mentioned, plotting an attribute showing all turtles can easily result in 100,000 data-points on one single chart. If a chart is small, a lot of data-points will be drawn on top of each other. This is a waste of time because you can not see most of the data-points but it takes a lot of time to plot. Therefore we introduced an algorithm that reduces data-points that are close to each other. We used the following x- and y-tolerance to determine the degree of reduction:

\[
X_{\text{tolerance}} = \left(\frac{\# \text{ time-frames}}{\text{chart width}}\right) \times C
\]

\[
Y_{\text{tolerance}} = \left(\frac{\text{value}_{\text{max}} - \text{value}_{\text{min}}}{\text{chart height}}\right) \times C
\]

Where $C$ is a constant which can be used to adjust the precision/speed ratio. A small $C$ gives a more precise chart but makes the plotting take longer, and vice versa. In our case, we used $C = 2$. $value_{\text{max}}$ and $value_{\text{min}}$ are the maximum and minimum value of the plotted attribute.
Algorithm 1 Reduce the amount of data-points; data-points contains the data from each software event of the selected turtles. Note that it only contains the data of selected turtles that we want to plot on the chart. The $x$-value of data-points is the time attribute, the $y$-value contains the selected attribute value.

1: sorted ← data-points sorted on time attribute (x-axis)
2: result ← data-points
3: for all $P \in$ sorted do
4:   if $P \in$ result then
5:     $N \leftarrow$ next point after $P$ in sorted
6:     while do
7:       if then $\text{distance}_x(P,N) < X_{\text{tolerance}}$
8:         remove $N$ from result
9:       end if
10:     end while
11:   end if
12: end for
13: return result

Algorithm 1 shows how the data-points of an attribute are reduced. The algorithm is based on the Douglas-Peucker algorithm \cite{6}. The data-points of an attribute of the selected turtles are merged together. Note that it only contains the data of selected turtles that we want to plot on the chart. Then all data-points are sorted on the x-axis, i.e. sorted on the time attribute. Then for each point their neighbours on the right are visited until the $X_{\text{tolerance}}$ between them is reached. For each neighbour is checked whether the y-distance between them is within the $Y_{\text{tolerance}}$, if so, the neighbour is removed from the dataset.

And example of the execution of the algorithm is shown in figure 5.10.

The data-points reduction algorithm increased the performance a lot, but it can still take some time to plot the data-points. Therefore we decided not to re-plot the data-points every time a chart is resized or shown in full-screen. When a chart is resized the new set of data-points is calculated using the $X_{\text{tolerance}}$ and $Y_{\text{tolerance}}$ based on the new width and height of the chart. Only if the number of data-points in- or decrease with more than 20% then the new data-points are plotted.

![Figure 5.10: Example of an execution of the data-points reduction algorithm.](image-url)
Chapter 6

Results

In this chapter we will evaluate the presented tasks of chapter 2.4. We defined the following tasks:

T1. understand behaviour of software events over time
T2. find patterns in the software events
T3. find software events that have a specific attribute value
T4. compare attributes of agents with each other:
   T4-1. invalid attribute values
   T4-2. missing attribute values
   T4-3. duplicate attribute values
T5. derive new attributes using simple data processing

The defined additional tasks of TechUnited described in section 2.5 are evaluated in the user evaluation of chapter 7.

We use the general tasks to demonstrate the applicability of our solution for multi-agent systems in general. We present a use case for each task stated above. For each use case we start with presenting a concrete problem that has to be solved and then work our way through the tool to solve it. The concrete problem will be based on the task, but of course, it will include a robot football problem because this is the only available dataset for the tool at the moment.

After all use cases are completed we will give a brief summary of the findings and performance of the tool. Task T5 will be omitted as a separate use case because the derived attributes are used within the other use cases.

Figure 6.1: (l) Derived attribute that shows successful and failed shots at goal; (r) Plot of the derive goal attribute. The plot shows that turtles two and five scored at the same time.
6.1 Use case 1: understanding behaviour over time

Task $T1$ states that the tool should support the expert to understand the behaviour of one or more agents over time. This can vary from the plotting of a simple attribute over time up to the comparison of a complex combination of attributes over time.

In this use case we want to know how many shots there were on the goal, and how many of these shots were successful. In the dataset there is no attribute that describes whether a shot at the goal was successful. It can however, be solved by deriving an attribute.

The $\text{refbox-command}$ attribute describes all the referee decisions like throw-in, free-kick and goals scored. So the goal $\text{refbox-command}$ will be used for the success-postcondition. As a fail-postcondition we used the inverse, so any other game situation, for example if the ball missed the goal.

Finally, we need to decide when an attempt for a goal is made. If the turtle has the ball it has various options, for example pass the ball to another turtle, dribble or take a shot at the goal. These decisions are maintained in the $\text{SkillID}$ attribute. So the preconditions consist of the $\text{SkillID}$ and ball possession of the turtle. Ball possession is maintained in the $\text{CPB}$ attribute, which is short for $\text{Check Possess Ball}$.

The derivation of the attribute, based on the described conditions, is shown on the left of figure 6.1, the resulting data with the given dataset is shown on the right of figure 6.1.

The right side of figure 6.1 shows that turtles two and five scored at the same time (around time-frame 7,800), this is impossible. After some investigation it seemed that turtle five actually failed and the ball ended up at turtle two. This is possible because turtle five could have hit the keeper or the pole.

To solved this we added another fail-postcondition. The extra fail-postcondition stated that the ball does not end up at a team-mate, this is shown on the left of figure 6.2.

The resulting data is shown on the right of figure 6.2, this now shows that turtle 5 failed. The final derived attribute shows that there are seven attempts for a goal, of which four succeeded. It also shows that different three turtles made an attempt on the goal. Both shots of turtle two were successful, turtle four and five only scored once.

![Figure 6.2: (l) Improved version of the derived goal attribute; (r) Plot of the improved derived goal attribute. Only turtle five scored a goal at time-frame 7,800.](image)
CHAPTER 6. RESULTS

6.2 Use case 2: find patterns in the software events

Task $T2$ states that the solution should support the expert with finding patterns in the software events. In this case we do not present a specific problem to be solved for the use case. Instead we used the tool to find a pattern in the dataset. To do so we analysed various attributes until interesting behaviour was found that required further investigation.

We found that the CPU-load had very high spikes at some moments. To find out if there was a correlation with another attribute we added another scatter-chart and started looking for attributes that could possibly have a link with the CPU-load. The result is shown in figure 6.3. This shows that each time a turtle has the ball the CPU-load peaks. The pattern is presented to TechUnited and they confirmed this. The peak occurs because the turtle has to process much more when possessing the ball.

6.3 Use case 3: find occurrences of a specific attribute value

Task $T3$ states that the tool should support the expert with finding occurrences of a specific attribute value. In this use case we want to know when we got a throw-in, who executed the throw-in and whether it was successful. To find out when we got a throw-in and who executed it we have to find a specific attribute value.

As mentioned before, the $refbox-command$ attribute shows whether there was a throw-in, this attribute is shown with a categorical-chart in figure 6.4. There are two attributes for throw-in, one for each team. So we need to know what team we are, using the $team-color$ attribute we know that we are the magenta team. This means that we got eleven throw-ins during the match.
CHAPTER 6. RESULTS

Figure 6.4: Categorical-chart that shows the referee decisions over time, there where 11 throw-ins for the magenta team.

Next, we need to know what turtle is going to execute the throw-in. This is maintained in the refbox-role attribute, according to the procedure of TechUnited the turtle with role one is going to execute it.

In figure 6.5 the turtles that execute the throw-in are shown. It shows that there is some swapping of roles before the execution of the throw-in. This is shown in the immediate role changes right after a throw-in command is given by the referee. The actual throw-ins are only done by turtles three and four.

Now we want to know if the throw-in was successful. We could do this by investigating each throw-in using the visualization of the playing field, and verify whether the ball successfully arrives at a team-mate. But a better alternative would be to create a derived attribute.

We already know the preconditions, refbox-role has to be equal to one and refbox-command has to be throw-in magenta. The throw-in succeeded if the ball arrives at a team-mate from the turtle that executes the throw-in. It has failed if the opponent has the ball or the game is stopped by the referee.

The resulting derived attribute is shown on the left of figure 6.6. The resulting data is shown on the right of figure 6.6. It shows that most throw-ins result in failure, only four of them were successful, all of them were executed by turtle four. Using this derived attribute we know when we got a throw-in, who executed the throw-in and whether it was successful.

Figure 6.5: A filter is applied on the throw-in of magenta (left chart). On the right chart only refbox 1 remained by applying a filter. The turtle that has refbox 1 is going to execute the throw-in.
6.4 Use case 4: compare attributes of agents

Task $T_f$ states that the tool should support the expert with comparing attribute values between agents. For example, which agent has the highest CPU-load. This can easily be determined with the help of a scatter-chart, as shown in figure 6.7.

But besides the regular comparison of attribute values, the task also describes three other types of attribute value comparisons.

The first is finding an invalid combination of two attribute values, this can be for one agent, or between multiple agents.

The second is finding missing attribute values. Sometimes a specific value must always be present between the agents. For example, there must always be a goalkeeper.

The last type is finding duplicate attribute values between agents at the same time-frame.

In the next subsections, for each sub-task a use case is presented to determine whether the tool can handle them.
6.4.1 Compare invalid attribute values

Sub-task $T4-1$ states that the tool should support the expert with finding invalid combinations between two attribute values, this can be for one agent, or between multiple agents.

A common problem with the turtles of TechUnited is a so-called, illegal move. An illegal move occurs when a turtle with ball possession performs a move when it should actually perform a dribble. We will derive an attribute that shows when an illegal move occurs in the dataset.

The derived attribute is shown in figure 6.8. The precondition states that the turtle must have the ball and perform a move. The derived attribute has no postconditions, as a result a test is directly set as successful.

The result of the derived attribute is shown on the right of figure 6.8, it shows that turtle 5 performed an illegal move over a long period of time.

6.4.2 Compare missing attribute values

Sub-task $T4-2$ states that the tool should support the expert with finding a missing attribute value, for example, there should always be a specific attribute value present.

In this use case we want to know if there was always a main attacker present during the match. This information is given in the $RoleID$ attribute, which is plotted in figure 6.9. Both the top scatter-chart and the bottom categorical-chart show the $RoleID$ attribute. We applied a filter on the main attacker on the bottom chart, now the scatter-chart also only shows the main attacker.

Next, we analyse the top chart for gaps, if there is a gap it means there was no main attacker at that time-frame. It appears there is no gap, so the main attacker was always present during the match.

Using this method, we can not guarantee there was always a main attacker present during the match. Because of the data-point reduction the plotted data only shows subset of the dataset. Also, it is very easy to oversee a small gap in the scatter-chart.

Figure 6.8: (l) Derived attribute that shows when an illegal move occurred. An illegal move takes place when a turtle performs a move with the ball instead of a dribble; (r) Resulting data of the illegal move attribute.
CHAPTER 6. RESULTS

Figure 6.9: Both charts show the *RoleID* attribute. The bottom chart has a filter applied on the main attacker. As a result the top chart also only shows the main attacker. It seems there was a main attacker at all times.

### 6.4.3 Compare duplicate attribute values

Finally, sub-task *T4-3* states that the tool should support the expert with finding duplicate attribute values between agents at the same time-frame. We want to know if there were two main attackers at the same time during the match.

Using the data-mapper we created a *RoleID* attribute that only shows the main attacker as a categorical value. The result is shown in figure 6.10. The figure shows every moment a turtle was the main attacker.

Now all we have to do is find overlap between two turtles. It seems that there is no overlap, however it is quite difficult to see using this chart. The chart does not clearly show small overlaps, but we do know that there was no duplicate role for a long period of time.

Figure 6.10: Shows the moments each turtle was the main attacker during the match.
6.5 Summary

Overall the tool performed very well, the results are summarized in table 6.1.

In the first use case, understanding behaviour over time, we used a derived attribute to show the shots at the goal. The tool can handle this type of querying very well. The charts show the behaviour of an attribute over time clearly. Using derived attributes saved a lot of configuration and attribute comparison time.

In the second use case we had to find a pattern. Finding a pattern can take quite some time, this is because you do not know what you are looking for. But once a pattern is found, it is easy to confirm the pattern using the tool.

The third use case, finding occurrences of specific attribute values, is handled fine by the tool. Sometimes, the data-points can become really small and hard to read. For example, when there are lot of categories to show in the categorical-chart. This can be solved by creating a extra instance of the attribute mapping, only configuring subset of the category values. This will result in the categorical-chart showing less categories.

The last use case is separated into three sub-cases. These sub-cases are handled less well by the tool. Comparing regular attributes using the scatter-chart is handled excellent by the tool. But it is difficult to find missing or overlapping attribute values between agents. If the gap or overlap of attribute values is large enough, the tool shows it very clearly. But small gaps or overlaps are not clearly visible. Also, the reduction of the data-points can cause gaps or overlaps to become hidden.

Based on these findings, it seems the additional charts presented in section 4.3 are not sufficient for the task. A possible solution can be to support zooming on the x-axis, showing the data in more detail. This would make gaps and overlaps more visible. Possibly, a parallel coordinate plot could provide a solution as well. Also, the tool could show some additional visual indication on the charts at the occurrences of gaps or overlaps. Finally, the derived attributes could also be extended to show the overlaps and gaps.

The current implementation of the derived attributes already offer a good addition to the tool, it provides new insight and saves a lot of time.

<table>
<thead>
<tr>
<th>Task</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1. Understanding behaviour of software events over time</td>
<td>Good</td>
</tr>
<tr>
<td>T2. Find patterns in the software events</td>
<td>Good</td>
</tr>
<tr>
<td>T3. Find software events that have a specific attribute value</td>
<td>Good</td>
</tr>
<tr>
<td>T4. Compare different attributes of an agent</td>
<td>Good</td>
</tr>
<tr>
<td>T4-1. Compare attributes of agents with each other (invalid)</td>
<td>Good</td>
</tr>
<tr>
<td>T4-2. Compare attributes of agents with each other (missing)</td>
<td>Fair</td>
</tr>
<tr>
<td>T4-3. Compare attributes of agents with each other (duplicate)</td>
<td>Fair</td>
</tr>
<tr>
<td>T5. Derive new attributes using simple data processing</td>
<td>Good</td>
</tr>
</tbody>
</table>

Table 6.1: Summary of the performance of the tool for each task.
Chapter 7

Evaluation

To assess whether the prototype is successful we conduct a user evaluation. In this chapter we will start with describing the design and set-up of the evaluation. Next, we present the results of the evaluation.

7.1 Design

The evaluation consists of two parts. The first part contains a number of assignments that have to be solved using the prototype. The assignments are based on the task TU1 up to TU5 defined in section 2.5. We have composed two assignments for each task. During the execution of the assignments, we use the think-aloud method. This means that we will ask the participant to say out loud what he or she is thinking during the entire evaluation. This way we get a better understanding whether the user interface of the tool has a logical structure.

The second part of the evaluation consists of a questionnaire that has to be answered right after the completion of the assignments. The first five questions use a 5-point Likert rating scale to answer them. By making use of this scale the answers can easily be compared and it requires little effort to answer. These questions mainly focus on rating the visualization and its separate elements.

After a test run of the evaluation it became clear that the evaluation took too long, a little over an hour for one participant. This can be partially explained because the test subject had no domain knowledge and did not know the attributes contained in the data, for the actual evaluation group this will not be the case. Therefore we have decided not to reduce the number of assignments. Also, we would like each task to have at least two assignments. We did however add some hints to the first few questions so that they would be easier to solve.

The final form used for the evaluation can be found in appendix B.
CHAPTER 7. EVALUATION

As mentioned in section 5.5, we will examine which visual representation of an opponent on the field visualisation is best.

The first concept of an opponent shows which turtles see the opponent at the given position on the field visualisation, see figure 7.1a. The other concept shows the turtles that do not see the opponent at the position on the field visualisation, see figure 7.1b. At the left visualisation of both concepts in figure 7.1 the first five turtles think there is an opponent at the position. At the right visualizations of both concepts only the sixth (yellow) turtle thinks there is an opponent at the position.

During the evaluation some of the participants will work with the first concept of the visualisation, other participants will work with the second concept. After the evaluation we will ask them whether they liked the representation of the opponent on the field visualisation.

7.2 Set-up

The participants that evaluated the prototype are team members of TechUnited, they have the necessary domain knowledge to asses the usefulness of the tool. The evaluation is performed by four members of the team. Before the evaluation takes place a demonstration of the tool is presented. This gives the participants a general impression of the tool.

7.3 Results

The tool has a steep learning curve. It generally takes a while before the participant exactly knows how the tool works and with what approach an assignment could be solved. Once this curve has been overcome the participant gets excited and sees the capabilities of the tool.

Each evaluation took about 45 minutes per participant, this was longer then expected. It probably took longer because they used the tool for the first time, once they have used the tool a few times it will go faster. On the other hand, all the questions were answered correctly.

Using the think-aloud method, it showed that it sometimes took a while for the participant to pick the right chart to solve the assignment. At other times they were able to solve the assignment on their own ingenious way.
CHAPTER 7. EVALUATION

The answers to the first five questions of the questionnaire are shown in table 7.1. The general impression of the tool got a 4 1/4 on average on the 5-point Likert scale. The tool almost completely met their expectations, it also got a 4 1/4 on the scale. The field visualisation got a 4 on the scale. One participant gave it a score of 3, so its rating of the field visualisation was neutral. This means that there are still some improvements to be made on the field visualisation. Unfortunately he did not write any feedback about the field visualisation in the comments / improvements section on the questionnaire. The categorical-chart got a 4 3/4 on the scale, everyone was very positive about this chart. Finally the turtle-chart got a 4 1/2 on the scale. These scores show that the participants are very satisfied with the tool.

<table>
<thead>
<tr>
<th>Question</th>
<th>(bad)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 (good)</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is your general impression of the tool?</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>4 1/4</td>
<td></td>
</tr>
<tr>
<td>Does the tool meet your expectations?</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>4 1/4</td>
<td></td>
</tr>
<tr>
<td>What do you think of the field visualization?</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>What do you think of the categorical-chart?</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>4 3/4</td>
<td></td>
</tr>
<tr>
<td>What do you think of the turtle-chart?</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>4 1/2</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1: Answers to the first five questions of the questionnaire.

We also asked the participants to write down two assignments that they found most difficult. The most difficult assignment was to determine whether the match suffered of poor localization, three participants wrote this assignment. The intention was to let the participant scroll through time and see if there is a lot of noise between the observed location of opponents. Unfortunately, this question was interpreted completely different by the participants. Looking back, the formulation of the assignment was incorrect. We think that if the question was phrased better the assignment would not be considered so difficult.

Another assignment that was deemed difficult was finding out whether there was always a main attacker, half of the participants found this difficult. The configuration of the graphs and data was done quickly by everyone. The charts that could be used to solve the assignment are shown in figure 7.2. It is obvious that it is difficult to find a gap using the charts like this. Also, the data-points on the scatter-chart can overlap because they are multiple pixels wide, this is not convenient when searching for a gap. Because of this, each participant gave the feedback that it would be useful to zoom on the x-axis.

Figure 7.2: It is difficult to see if there is a gap in the Attacker Main role (Role Id = 1).
Everyone has answered that they will use the tool in the future. The main feedback of the features that the tool lacked was the zooming on the time-axis, everybody mentioned this as a lacking feature.

Additionally there were small individual comments. Currently their data is being mapped in the tool. Their football match data is continuously expanding, new attributes are added. It would be convenient if the tool can load different (historical) mappings, so old data-sets can easily be loaded.

The entire robot football league is planning to use a new standard for storing match data using JSON. This is currently only used to a small extent. It would be nice if the tool would support this future standard.

The derived attributes were very much appreciated. They would like to have the derived attributes feature extended with more functionalities. For example to solve equations like distance or simple calculations. It would also be nice if there is a derived attribute that shows whether there was a duplicate role during a match. At the moment finding a duplicate role is quite difficult with the tool.

Furthermore, there was also a request to zoom in on the y-axis, just like the time on the x-axis. Sometimes the tool froze because it had to load a lot of data-points on the scatter-chart, it would be nice if this execution could be stopped prematurely.

Finally, the time on x-axis is based on the index of the column in the matrix, this is not correct. The .mat datafile used by TechUnited contains a separate list that provides a time-stamp for each column of the matrix, these timestamps should be used.

We asked the participants whether they liked the representation of the opponent on the field visualisation. Based on the answers it seems that the participants find it more convenient to work with the visual representation of the opponent where the turtles that see the opponent at the given position are shown. This is the opponent representation of figure 7.1a. The representation was mostly used to see which turtles see the opponent at the position. The representation of figure 7.1a shows this very quickly, the other representation can be confusing.

Finally, there was only positive feedback in the comments/improvements area. Almost everyone wrote down that they found it a very nice tool that they will definitely use in the future.
Chapter 8

Discussion & Conclusion

In the problem analysis we defined the general problem we want to solve: Which visual analytics solution is most useful to support the domain expert with the analyses of the behaviour of agents and the multi-agent system as a whole?

To answer this question we defined the data structure and determined tasks $T_1$ up to $T_5$ that the tool should support. We defined additional tasks $TU_1$ up to $TU_5$ and requirements $R_1$ up to $R_3$ for our case study, TechUnited.

Based on the data structure, tasks and literature we made a selection of three visual designs that could offer a solution; small multiples, parallel coordinate plots (PCP) and linked charts. After investigating each of them, we chose one of the three designs. Small multiples did not support enough of the presented tasks from the problem analysis. Both the PCP and linked charts design offered a good solution. The temporal order of the data was an important characteristic that the visual solution should support. The PCP can not properly deal with this aspect of the data, therefore the linked chart design was chosen.

The main shortcoming of linked charts was comparing attribute values between agents. To improve this shortcoming, we introduced two additional charts; the categorical- and agent-chart. Based on the findings of the visual design we implemented a fully functional prototype.

The prototype contains extra features to support the additional requirements that were desired by TechUnited. The most important features were the dynamic docking layout that could be saved and re-opened at a later time, a dynamic mapping of the dataset, the derivation of attributes, filtering on the y-axis of each chart, range boundaries on the scatter-chart and improvements of the performance due to the large datasets.

The presented solution has been verified using two different methods; use case testing and a user evaluation.

In the first method we presented a use case for tasks $T_1$ up to $T_4$ from the problem analysis. Each use case had to be solved using the prototype. This showed that the prototype could be used for all types of multi-agent systems. From the use cases it followed that the tool could handle each task well except for the comparison of attribute values between agents ($T_4$). It seemed that the presented additional solutions from the visual design process are not sufficient. It is difficult to find missing ($T_4\cdot2$) or overlapping ($T_4\cdot3$) attribute values between agents if the gap or overlap is small.
Four potential solutions for finding missing or overlapping attribute values between agents were presented. The first solution can be to support zooming on the x-axis, showing the data more detailed. This would make gaps and overlaps more visible. Secondly, using a parallel coordinate plot as a chart could provide a solution. Also, the tool could show some additional visual indication on the charts at the occurrences of gaps or overlaps. Finally, the derived attributes could also be extended to show the overlaps and gaps.

The derived attribute feature that has been developed for task T5 were used within each use case. From the use cases it followed that the derived attributes offer a good addition to the tool, they provide new insight and save a lot of time.

The second method to evaluate the prototype was a small user evaluation. Four TechUnited team members had to solve a number of assignments using the prototype. The tool has a steep learning curve, but once this curve has been overcome the user sees the capabilities of the tool. The biggest problem is that it is difficult to see small overlaps or gaps between agents. Therefore the feedback was that zooming on the x-axis is required. But besides this lacking feature they were very impressed with the tool.

There were also some additional comments, but these were mainly nice to have extensions. These extensions contained; loading different mappings, supports for the new JSON standard, extended functionality of derived attributes, zooming on the y-axis, prematurely stop the loading of chart and use of the correct time attribute. A detailed description of each of these extensions can be found in chapter 7.3.

Each team member gave the tool and its elements a high rating. Though, based on the ratings, the field visualisation could still be improved. Unfortunately there was no feedback given about what could be improved in the field visualisation.

The results of the two methods verified that the prototype offers a good visual solution but can still be improved. We mainly focussed on TechUnited, but the tool can be used for all kinds of multi-agent systems. The only requirement is that the data satisfies the structure described in section 2.3. We believe that this tool can provide great insight into all kinds of multi-agent systems.

8.1 Future work

In the problem analysis we set some restrictions because of the limited time available. As a result there are still unvisited items that can be examined.

The first item is processing online data, so streaming directly, for example during a match, instead of analysis afterwards.

Also, only five of the eight additional tasks of TechUnited are considered. These unvisited tasks are; benchmarking the world view of agents with reality (TU6), tuning the decision making of an agent (TU7) and finding patterns over multiple runs of the multi-agent system (TU8).

Besides these omitted items there are also improvements based on the use case testing and user evaluation.

The main problem that has to be solved is that it is difficult to see small overlaps or gaps between agents. Some potential solutions to solve this problem has been presented; zooming on the x-axis, additional visual indication on the charts, parallel coordinate plot or an extension of the attribute derivation.

Besides the main problem there are also nice to have extensions which could be implemented to improve the tool.
Bibliography


Appendix A

TechUnited data structure
### APPENDIX A. TECHUNITED DATA STRUCTURE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turtle</td>
<td>Geographical</td>
<td>Position of the turtle</td>
</tr>
<tr>
<td>Ball</td>
<td>Geographical</td>
<td>Position of the ball</td>
</tr>
<tr>
<td>Ball WM</td>
<td>Geographical</td>
<td>Ball position in world-model</td>
</tr>
<tr>
<td>Obstacles</td>
<td>Geographical</td>
<td>Obstacles in the field seen by the turtle</td>
</tr>
<tr>
<td>Opponent</td>
<td>Geographical</td>
<td>Position of opponents</td>
</tr>
<tr>
<td>Turtle WM</td>
<td>Geographical</td>
<td>obsolete</td>
</tr>
<tr>
<td>Target</td>
<td>Geographical</td>
<td>Position of the turtle’s current target</td>
</tr>
<tr>
<td>Ball movement</td>
<td>Geographical</td>
<td>Position and velocity of the ball</td>
</tr>
<tr>
<td>Sub-target</td>
<td>Geographical</td>
<td>Sub-target position to reach target</td>
</tr>
<tr>
<td>Line position recognition</td>
<td>Geographical</td>
<td>Estimated progression of the ball</td>
</tr>
<tr>
<td>Sub-sub-target</td>
<td>Geographical</td>
<td>Sub-target position to reach sub-target</td>
</tr>
<tr>
<td>Opponent velocities</td>
<td>Geographical</td>
<td>Velocity of each opponent</td>
</tr>
<tr>
<td>Way-points</td>
<td>Geographical</td>
<td>obsolete</td>
</tr>
<tr>
<td>CPB poi xyo</td>
<td>Geographical</td>
<td>Position where the ball is intercepted</td>
</tr>
<tr>
<td>Skill ID</td>
<td>Categorical</td>
<td>current skill and shottype (move, dribble, aim, kick, shield, intercept)</td>
</tr>
<tr>
<td>CPB</td>
<td>Categorical</td>
<td>Check Possession Ball (does the robot have the ball)</td>
</tr>
<tr>
<td>Team color</td>
<td>Categorical</td>
<td>Color of the team</td>
</tr>
<tr>
<td>Blue is home-goal</td>
<td>Categorical</td>
<td>Boolean defining what the home goal is</td>
</tr>
<tr>
<td>Emergency status</td>
<td>Categorical</td>
<td>Emergency status of the turtle</td>
</tr>
<tr>
<td>Role ID</td>
<td>Categorical</td>
<td>Role of the turtle (current, previous)</td>
</tr>
<tr>
<td>Ref-box role</td>
<td>Categorical</td>
<td>role during a refbox situation (free kick, corner, etc)</td>
</tr>
<tr>
<td>Default role</td>
<td>Categorical</td>
<td>Assigned role ID</td>
</tr>
<tr>
<td>CPB team</td>
<td>Categorical</td>
<td>Check Possession Ball of team (does the team have the ball)</td>
</tr>
<tr>
<td>Robot in field</td>
<td>Categorical</td>
<td>Is the robot active / participating in the game</td>
</tr>
<tr>
<td>Ball found</td>
<td>Categorical</td>
<td>does the robot know where the ball is</td>
</tr>
<tr>
<td>Camera status</td>
<td>Categorical</td>
<td>Status of the camera</td>
</tr>
<tr>
<td>Ball confidence</td>
<td>Categorical</td>
<td>confidence of seen ball</td>
</tr>
<tr>
<td>Motion status</td>
<td>Categorical</td>
<td>Status of the motion</td>
</tr>
<tr>
<td>Vision status</td>
<td>Categorical</td>
<td>Status of the vision</td>
</tr>
<tr>
<td>World-model status</td>
<td>Categorical</td>
<td>Status of the world-model</td>
</tr>
<tr>
<td>Active</td>
<td>Categorical</td>
<td>Status of the turtle</td>
</tr>
<tr>
<td>Ref-box command</td>
<td>Categorical</td>
<td>command which is send by the rebox (referee computer)</td>
</tr>
<tr>
<td>Used ball turtle ID</td>
<td>Categorical</td>
<td>Turtle ID indicating the source of the ball which is sent along</td>
</tr>
<tr>
<td>Merged ball source</td>
<td>Categorical</td>
<td>What sensor is providing a ball, kinect or omnivision</td>
</tr>
<tr>
<td>Opponent label number</td>
<td>Categorical</td>
<td>Labels of each opponent</td>
</tr>
<tr>
<td>Opponent with ball</td>
<td>Categorical</td>
<td>Number of the opponent with the ball</td>
</tr>
<tr>
<td>Battery voltage</td>
<td>Quantitative</td>
<td>Voltage of the battery</td>
</tr>
<tr>
<td>Cpu0 load</td>
<td>Quantitative</td>
<td>CPU load of first processor</td>
</tr>
<tr>
<td>Cpu1 load</td>
<td>Quantitative</td>
<td>CPU load of second processor</td>
</tr>
<tr>
<td>Restart count motion</td>
<td>Quantitative</td>
<td>number of crashes and deadlocks</td>
</tr>
<tr>
<td>Restart count vision</td>
<td>Quantitative</td>
<td>number of crashes and deadlocks</td>
</tr>
<tr>
<td>Restart count world-model</td>
<td>Quantitative</td>
<td>number of crashes and deadlocks</td>
</tr>
<tr>
<td>Path length</td>
<td>Quantitative</td>
<td>obsolete</td>
</tr>
<tr>
<td>Motor temperature</td>
<td>Quantitative</td>
<td>Temperature of each motor</td>
</tr>
<tr>
<td>Packet Loss turtles</td>
<td>Quantitative</td>
<td>Packet loss with the other turtles</td>
</tr>
<tr>
<td>Packet loss coach</td>
<td>Quantitative</td>
<td>Packet loss with the coach</td>
</tr>
<tr>
<td>Mu-field Nr</td>
<td>Quantitative</td>
<td>Mu-fields indicate the best position for the turtle on the field</td>
</tr>
</tbody>
</table>

Table A.1: Data structure of the data send by turtle.
Appendix B

User evaluation
User-evaluation Prototype

Assignments (Think aloud)

1. When was there a free kick for the team?
   *Tip: use the categorical-chart*

2. How many shots on goal were successful?
   *Tip: Use a derived attribute*

3. Was there always a main attacker (roleId = 1)?
   *Tip: combine categorical- and scatterchart with a filter*

4. Was there a (significant) duplicate role main defender during the game (roleId = 3)?

5. From which position do we often score?
   *Tip: Use the field with history-lines*

6. Did the game suffer from poor localization (noise)?
   *Tip: Use the opponents on the field*

7. Was poor communication-coverage (> 25%) with the keeper mainly on the own side of the field or at the opponents’ side?
   *Tip: Use the filter and history-lines on the field*

8. What caused the noise in the battery voltage?

9. Which turtle shoot at the goal most often?

BONUS
10. Is there a relation to the negative battery voltage of turtle 4 in the data?
Questions

What is your general impression of the tool? (bad) 1 2 3 4 5 (good)

Does the tool meet your expectations? (not at all) 1 2 3 4 5 (completely)

What do you think of the field visualization? (bad) 1 2 3 4 5 (good)

What do you think of the categorical-chart? (bad) 1 2 3 4 5 (good)

What do you think of the turtle-chart? (bad) 1 2 3 4 5 (good)

Which two assignments did you found most difficult?

Are you planning to use the tool? Yes / No

What features does the tool lack?

Comments / Improvements