

MASTER

Interactive visualization of event logs of medical imaging systems

Lok, W.

Award date:
2015

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain



Department of Mathematics and Computer Science
Visualization Research Group

Interactive Visualization of Event Logs of Medical Imaging Systems

Master Thesis

W. Lok

Supervisors:

Prof. dr. ir. J.J. van Wijk
TU/e

Dr.ir. M. Barbieri
Dr.ir. J. Korst
Philips Research

Eindhoven, October 2015

Abstract

Philips Healthcare produces various types of medical imaging systems, such as X-ray, Computed Tomography (CT) and Magnetic Resonance (MR) Imaging. These systems are mainly used in the examination of patients for the diagnosis of diseases. It is therefore of high importance that these devices are available whenever they are required. When a problem occurs in a system, the problem needs to be analyzed and solved as quickly as possible. These systems are however very complex and expertise is required for maintaining and repairing them. For analyzing the behavior of medical imaging systems, the event logs that have been produced by the system need to be analyzed.

Philips has worked on researching several data analytics methods that support failure diagnostics for corrective maintenance and failure prediction for predictive maintenance. An area that has not been extensively researched yet is visual data exploration. This thesis explores the possibility of using interactive data visualization for the analysis of event logs for serviceability purposes. The designed and implemented prototype can be used for analyzing the utilization, behavior and errors of individual systems. The system uses easy to understand visualization methods for showing multiple overviews of the data. The user is enabled to request details on demand by interacting with the prototype.

Preface

With this thesis, I conclude my studies at Eindhoven University of Technology. I would therefore like to use this opportunity to thank all those who have guided and encouraged me during this time.

I would like to start by thanking my supervisor Prof. dr. ir. J.J. van Wijk at the Eindhoven University of Technology. He guided me through the whole project, and his comments and feedback on the design and implementation of the prototype as well as this thesis were of great value. In each meeting, he showed great inspiration which benefited me a lot.

Second, I would like to thank my supervisors Dr.ir. M. Barbieri and Dr.ir. J. Korst for giving me the possibility of performing my graduation project at Philips. At the start of this project, they were of great help of introducing me into the field of event logs of medical imaging systems and the processes for maintaining them. During the project, they continuously provided good feedback on my progress and were always available when I needed help.

I would also like to thank my intern colleagues for organizing activities during and outside of working hours. Especially the social drinks helped me relax and make it through the summer. I thank my parents for supporting me, socially and financially, during my studies and during this project. Finally, I would like to thank all my friends and fellow students for their support.

Contents

Contents	vii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Project Goal	2
1.2 Document Structure	2
2 Background	3
2.1 Raw Log Data	3
2.2 Maintenance Strategies	4
2.3 Healthcare Maintenance Services	6
2.3.1 Remote Service Engineers	7
2.3.2 Field Service Engineer	7
2.3.3 Diagnostic Designer	7
2.3.4 Research & Development	8
2.4 Maintenance Administration	8
2.5 Scoping	9
2.5.1 Imaging Systems Data Sources	9
2.5.2 Actors	12
2.5.3 Scope Decision	12
3 Problem Analysis	15
3.1 Related Work	15
3.1.1 Philips systems	15
3.1.2 Event Timeline Visualization	16
3.1.3 Event Data Mining and Visualization	20
3.2 Application of Related Work	23
3.2.1 Data comparison	23
3.2.2 Interactive Event Timeline Visualization	26
3.2.3 Event Log Data Mining and Visualization	27
3.2.4 Discussion	28
3.3 Requirements	29
4 MR Event Log Data	33
4.1 Magnetic Resonance Imaging Systems	33
4.2 MR Data Model	35
4.2.1 Event Extraction	35
4.2.2 Generalization	36
4.3 MR Model Data Statistics	37

5	Interactive MR Event Log Browser	39
5.1	System Overview	39
5.2	User Interface Design	42
5.3	Site View	43
5.4	Calendar View	44
5.5	Event Timeline	48
5.5.1	Event Grouping	48
5.5.2	Visualization	48
5.5.3	Overlapping Events	49
5.5.4	Event properties	51
5.5.5	Linked events	52
5.5.6	Zooming and panning	53
5.6	Additional Interactive Features	54
5.6.1	Timeline Configuration	54
5.6.2	Event Filtering	55
5.6.3	Configuration and System State Persistence	57
6	Evaluation	59
6.1	Evaluation of Requirements	59
6.1.1	Summary Prototype	59
6.1.2	Scalability	60
6.2	User Feedback	63
6.3	Use Case Analysis	64
6.3.1	Regular Event Logs	65
6.3.2	Analysis of Abnormal Event Logs	66
6.3.3	RF Amplifier Error Analysis	71
6.4	Future Work	73
7	Conclusion	75
	Bibliography	77
	Appendix	79
A	Event Types	79
B	Prototype Development	80
B.1	JavaFX	80
B.2	Environment	80
B.3	Architecture	82
B.4	Metrics	86

List of Figures

2.1	Impact of Maintenance Plans on system condition	6
3.1	Common analyzer tool	16
3.2	MEBEF Dashboard	17
3.3	SeeSoft: Snippet of one hour visualization	17
3.4	MieLog: A highly interactive visual log browser	18
3.5	ASML TWINSCAN event log visualizer overview	19
3.6	LifeFlow: Visualizing an overview of event sequences	20
3.7	Schematic approach of Röder et al.	21
3.8	Visualization of event types	23
3.9	Schematic comparison of several data aspects	25
4.1	Ingenia 3.0T MRI and schematic MRI components diagram	33
4.2	Coils of an MRI system.	34
4.3	MR hierarchical data model	36
4.4	Cumulative histogram of the number of event logs per site.	38
4.5	Average exams per site per day.	38
5.1	Visualization Pipeline	39
5.2	Schematic overview of the views of the prototype	41
5.3	MR Event Explorer and Visualizer overview	41
5.4	The application with the Site View hidden	42
5.5	Site View	43
5.6	Line-charts for event distribution visualization	45
5.7	Line-chart showing the trend in the number of scans performed over half a year.	46
5.8	Dense-Pixel Displays	46
5.9	Two examples of event calendar.	47
5.10	Abnormal event distribution in the calendar view	47
5.11	Trivial rendering of the event timeline	49
5.12	Timeline visualization with resolved overlapping	51
5.13	Timeline after merging coil events.	51
5.14	Four examples of the property pop-up.	52
5.15	Property values for the raw-hardware and the coil event	53
5.16	Examples of linked events	54
5.17	Timeline configuration	55
5.18	Calendar before and after applying filter	56
5.19	Filter Dialog examples	56
5.20	SQL syntax errors	57
5.21	Storing or loading a setting or a system state.	57
6.1	Timeline with high utilization and cluttering of events.	62
6.2	Timeline of event log with all events enabled.	63

LIST OF FIGURES

6.3	Timeline of an average sized log.	65
6.6	Calendar View irregular behavior	66
6.4	Zoomed in and annotated timeline of a normal event log.	67
6.5	A site filter for selecting a small subset of sytems.	68
6.7	Event log with irregular behavior.	69
6.8	Event log with a coil problem.	70
6.9	Coil error messages.	70
6.10	VSWR (reflected power) error.	72
6.11	Amplifier error and UI-Message correlation.	72
B.1	The directory structure of the packages.	81
B.2	The package structure of the architecture	82
B.3	Class overview of the controller package.	83
B.4	Class overview of the Scene package.	83
B.5	Class overview of the model package.	84
B.6	Class overview of the service package.	84
B.7	Class overview of the Util package.	85

List of Tables

2.1	MR raw event log	10
2.2	iXR Event log snippet	11
4.1	MR Event Properties Examples	37
A.1	Event types and event statistics	79
B.1	The development and testing environment	81
B.2	Code metrics	86

Chapter 1

Introduction

Human healthcare is a complex process in which diseases, illnesses and injuries are diagnosed, treated and prevented. To support this process, several medical imaging systems are used by doctors and other medical staff. Medical imaging systems are capable of creating a visual representations of the interior of the human body or other objects. Philips Healthcare produces various types of medical imaging systems, such as X-ray, Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and Ultrasound. Each of these systems is very distinct and each system contains complex software and hardware components.

Since all of these systems are very important in diagnosing and treating patient's diseases, a high reliability and availability of these devices is of utmost importance. They should rarely fail, but if they do, they should be repaired as fast as possible. For the latter, it is important that the root cause of a failure can be identified quickly, such that the system can be repaired effectively and the system can be used optimally again. The process of maintaining and repairing medical imaging systems is part of the healthcare maintenance services process. To diagnose the root cause of a failure, the system in question needs to be analyzed. But since these systems are so complex, this diagnosis cannot be done by just looking at the external behavior of the system.

The log data of an imaging system has to be analyzed for creating a diagnosis. The data recorded in the log files describe the performed internal actions of the system and the errors that have occurred. Since medical imaging systems are very complex, the diagnosis have to be performed by experts. Philips offers multiple levels of support in order to be able to maintain their systems as efficient as possible. The first line of support is performed by the remote service team. Whenever a client reports a fault or a failure, a remote service engineer (RSE) tries to analyze the log files of the corresponding system in order to find the root cause of the problem. Whenever this is not sufficient or when the system needs to be repaired, a field service engineer (FSE) is sent to the machine to perform further analysis or to perform maintenance actions to repair the machine. Another form of support is preventive maintenance. In this case, the service engineers try to detect faults in a system before they actually affects the system.

The main problem with maintaining the medical imaging system is that the log files are complex due to the sheer amount of information that is logged and that tasks can be executed in parallel, causing interleaving sets of records. The log files are currently mostly viewed in textual form, using a dedicated log analyzer and each system produces multiple log files containing thousands of log messages per day. Most logs are only loosely structured and contain relevant as well as irrelevant information. There exist guidelines that have been created by diagnostic designers. While these do make the diagnosis more simple, it is still difficult for a service engineer to analyze these logs to detect anomalies in a system or to find the root cause of a failure. A large part of discovering diagnostic patterns is to analyze how a systems is behaving normally, what the system was doing prior to a failure, and in which context errors occur that are related to failures.

1.1 Project Goal

To support service engineers and the diagnostics designers, Philips has already performed various kinds of research in discovering diagnostic patterns. Most of this research however only involve the field of data mining and machine learning. An approach that has not extensively been applied is data visualization. This project aims to provide visual means for displaying event logs for serviceability purposes. A powerful addition to using visualization is the use of interactive features that enable the user to change the scope or other aspects of the data that is shown. It can enable the user to zoom from an overview of the data into a more detailed view, allowing more careful analysis of the data.

In other words, in this thesis we aim to discover:

“How can event logs of medical imaging systems be visualized for serviceability purposes?”

To be able to do this, we answer the following questions:

1. What are event logs for medical imaging systems?
2. How are the medical imaging systems currently being maintained?
3. What information that is stored in the event logs is interesting to show?
4. How can we visualize this information?

1.2 Document Structure

The remainder of the thesis is divided into six chapter. In Chapter 2, we first answer the first two question by presenting a general overview of the contents of event logs and the difficulties faced in analyzing them, and we present how Philips maintains her medical imaging systems as part of the healthcare maintenance services process. Philips supplies many different kinds of imaging systems. The structure and type of the log files differ for each kind of system. Due to the time constraints of the project, we have limited the scope of the project to one type of machine and a small set of target users. Section 2.5 therefore discusses the differences between the event logs and target users and presents the choice made.

Next, Chapter 3 first presents what has already been done in the context of event log visualization. Based on our discussion of the related work in Section 3.2, we then present our final approach towards the design of an interactive visualization tool for event logs. Chapter 3 is closed by presenting the requirement in Section 3.3. The requirements describe what information must be visualized and how the user should be enabled to interact with the system.

The goal of the project is to develop a prototype that is able to visualize the event logs for maintenance purposes. Chapter 5 presents and discusses the design and implementation of the visualization methods for the prototype. The implemented prototype is then evaluated in Chapter 6 by reviewing the requirements, by presenting the feedback from target users and by presenting a number of use cases that have been analyzed using the prototype.

Finally, we conclude this thesis in Chapter 7 by presenting a summary of this thesis and by looking back at the questions that have been posed above.

Chapter 2

Background

Log files are an important source for maintaining medical imaging systems. These files contain time-stamped records to describe the history of such a system. The log files are called *Event logs*, since most of these records can be related to an event that occurred in the system or in the environment of the system, e.g., a decision or action performed by the system or something that happened inside or outside of the system. These log files are used to analyze and diagnose system faults and failures. A fault is a condition in the system that causes the system to run sub-optimal. A failure indicates a loss in functionality due to one or more broken components. Commonly, all system features are still available whenever there is a fault in comparison to a failure. If there is a failure in a core component of the system, the system will not be able to operate. For both, the root cause of the problem needs to be found in order to fix the issue.

In this chapter, we describe the background of the project. First, Section 2.1 describes the event logs from the medical imaging systems and discusses the problems associated with analyzing and interpreting the event logs. Section 2.2 presents the maintenance strategies that are commonly used in the industry for maintaining machines. Section 2.3 presents how these maintenance strategies are used by Philips to maintain medical imaging systems and describes the main actors that are involved the most. The data that is administrated while performing maintenance is presented in Section 2.4. Finally, the different medical imaging systems and actors are discussed in Section 2.5 in order to motivate the decision of the scope.

2.1 Raw Log Data

Log information of Philips medical imaging systems is automatically sent to a central server on a daily basis or can be accessed remotely in case of a failure. A log file typically contains thousands of log records per day. It is important to note that the records that are produced by the systems are not specifically targeted towards maintaining the system. The log files can better be observed as debug logs and were initially performed for development and system debugging purposes. Therefore, some log records are irrelevant, and can be ignored, and other log records are useful, but can be cryptic, since the message associated with the record can contain some software or system specific details. As a consequence, system knowledge is often required to be able to understand most of the log records, and the inspection of log files to identify root causes of failures or indications of upcoming failures is very complex.

In addition, Philips Healthcare produces various types of medical imaging systems, referred to as *modalities*, such as X-ray, CT and MRI scanners. In this project, there was the possibility using the event logs of either MRI or iXR systems. In Section 2.5 we shortly compare the event logs and present the data source that has been chosen. Also, Table 2.1 2.2 show samples of event logs of both modalities.

Since each of these modalities work very differently and the raw data is logged in different formats, analyzing and diagnosing problems is unique for each modality. Most of the data that is being logged by the different types of systems can however be grouped into three categories: raw events, raw parameters and configuration data. We speak of raw data and parameters, since the log data is produced straight from the executing system and no analysis or modifications are made on the data.

Raw Event Data

First, an important source of data for analyzing system failures is raw events. Raw events describe information of any action that is performed by a subsystem of a machine. It could for example be a simple record describing the successful start or end of an operation or a task, or the event could describe a warning or a major error that occurred. Within an event log, a distinction can be made between events, operations and tasks. An operation can produce multiple events, e.g., describing the start, end and result of the operation. A task is a set of operations that is performed by one or more subsystems. The subsystems of the imaging systems often perform these operations in parallel. The events that are produced by the subsystems are therefore also produced in parallel, resulting in an interleaved sequence of time-stamped events.

The occurrences and properties of a task are often not logged implicitly as a single event but can be derived from the set of operations. For example, most of the modalities perform a scan of the body part that is being examined before the final image can be reconstructed. The start and the end of this scan is depicted by two different events. Events that occur between these two events can depict measurements or results of the scan. These events can however also be related to a sub-task or to an unrelated task, since these tasks are performed and logged in parallel.

Raw Parameters

The second data source is raw parameters. Raw parameters describe values used during the execution of operations or they describe the state of some components. Values can be user inputs, e.g., type of exam or positioning of the patient table; or values that are obtained from sensors, e.g., temperature of the room, power supply usage or level of cooling liquids. Depending on the values of the parameters, tasks could be executed differently and result in different operations being performed. These parameters can either be logged in the event logs as a set of individual events when the parameters are related to certain actions or tasks, or the parameters are logged separately when the parameters are measured continuously in set intervals.

System Configurations

Lastly, the system configuration describes properties and state information of the system itself. Each of the subsystems can have its own configuration. The configuration includes for example the system type, software version, values of global input parameters and constraints.

Difficulties

The generalization of event logs into raw events, parameters and configurations do however help only a little in analyzing them. The major problem is that different modalities have different log structures, the machines all work differently and some system knowledge is required for understanding the event logs. Therefore, the process of analyzing event logs for finding root causes of problems is also different for each modality. Furthermore, while different models of the same modality often have the same log structure, but their internal behavior can differ a lot. This is because the system can change significantly over the years. In the end, the event logs of all the different systems have to be interpreted and analyzed in different ways. All these different aspects make it difficult to interpret or to automatically process the event logs into a more logical model such as a process or relational model.

2.2 Maintenance Strategies

The next question is how medical imaging systems are maintained, what problems and difficulties are faced by the service engineers, and how these could be solved or made simpler. First, in this section, the different maintenance plans that are commonly used are discussed, and second, Section 2.3 describes how Philips uses these maintenance plans and the different actors that are involved.

As discussed above, the main goal of service maintenance is to make sure that the system is able to run optimally and minimizing the downtime of the maintained systems, while keeping the maintenance costs at an acceptable level. Different maintenance strategies can be employed to achieve this goal. There are two common strategies: reactive (or corrective) maintenance and proactive maintenance, where proactive maintenance can be split further into Preventive and Predictive Maintenance. These maintenance strategies are defined as follows [12, 16]:

- **Reactive Corrective Maintenance:**

This is the simplest form of maintenance. In this strategy, maintenance is performed after the failure of a component. This failure results in downtime of the system when this component is critical for operating the system. The failed component therefore has to be repaired or replaced as soon as possible in order for the system to be brought back into a functioning state and to limit the downtime of the system.

While corrective maintenance often results in the minimal amount of maintenance performed, it is often not an optimal strategy. First, there are some risks involved in allowing systems to be used until a failure occurs, since many failures occur while the system is being actively used. In the case of medical imaging systems, patients are involved. So, if a system fails while performing an examination on a patient, the patient may be harmed. Second, it can be difficult to determine when and which components will fail. This may result in long downtimes for a system, due to difficult troubleshooting and the time required for the delivery of spare parts. This is highly undesirable, since imaging systems are heavily used in most hospitals, effectively decreasing the amount of examinations that can be performed. The drawbacks can result in very high maintenance costs, so it is recommended to minimize the amount of reactive corrective maintenance actions necessary by also performing other types of maintenance.

- **Proactive Maintenance:**

In the following maintenance plans, engineers act before a system failure occurs in order to try to postpone or to prevent system failures, therefore limiting the amount of corrective maintenance that needs to be performed.

- **Preventive Maintenance:**

Preventive maintenance is a planned maintenance strategy, in which the system is scheduled to be regularly serviced in order to determine the global state of the system by performing system checks. Maintenance activities can be scheduled at regular intervals or can be based on the operational time of the system. Maintenance actions may include equipment lubrication, parts replacement, cleaning, and adjustment. Parts that show signs of deterioration can be replaced before the end of their lifetime to prevent failure in the future.

Preventive maintenance should reduce the probability of a system breakdown and extend the general lifetime of the system. A benefit of planned maintenance is that the maintenance can be planned ahead of time and outside of operational hours, therefore limiting or preventing downtime of the system. The costs of preventive maintenance will initially be higher than corrective maintenance, due to the regular maintenance actions that need to be performed, but overall it should reduce maintenance costs by reducing the amount of parts that need to be replaced and repaired, and by limiting downtime, e.g., a larger productivity of a system. A small drawback of preventive maintenance is that maintenance might be performed while it is not necessary, which is sometimes caused by the lack of historical data of the system to determine an optimal preventive maintenance plan, hence wasting time and resources.

- **Predictive Maintenance:**

In the predictive maintenance strategy, or condition-based maintenance strategy, system failures are prevented by trying to predict when certain components will fail. In condition-based monitoring, condition-monitoring data plays a pivotal role. This data can be obtained from diagnostic equipment, e.g., sensors or software programs that generate condition-based data from log data of the machine and describe the state of the components of the system. By linking condition-based data and the current condition of a system, one might be able to predict a failure of the system and intervene before the failure actually occurs, allowing optimal maintenance actions to be taken accordingly [9].

In comparison to preventive maintenance, actions are taken only when the system requires it, prolonging the usage time to be more closer to the expected life-time or even extend the life-time of components due to being well maintained. The longer the system is in use, the more condition-based data can be gathered, which can be used to further improve the condition-based monitoring equipment and to predict the remaining life-time of components more precisely.

Figure 2.1 shows the impact of the general condition of the system over time due to the usage of the different maintenance plans.

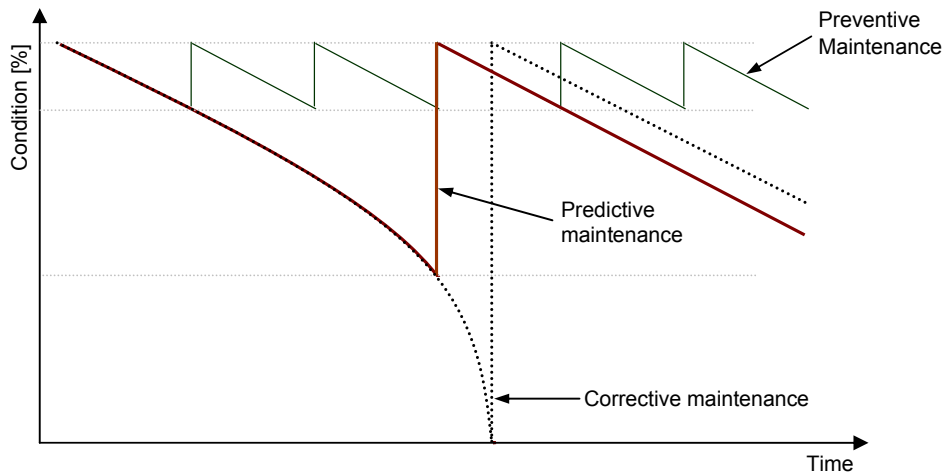


Figure 2.1: Comparison of the impact that maintenance plans (Corrective Maintenance, Preventive Maintenance) have on a system over time [12].

In practice, the described maintenance strategies are not or cannot be used in isolation. Optimally, only pro-active maintenance is employed, such that all failures can be prevented. This is not possible however, since there will always be system failures that cannot be prevented or predicted and it is difficult to create highly reliable systems at acceptable cost levels. Reactive corrective maintenance is therefore always required. On the other hand, a company will also not likely only offer corrective maintenance, since system failures are too unpredictable and it is often more costly in the long run. The time it takes to fix problems can take long, due to the logistics of maintenance engineers and the ordering and delivery of spare parts, causing long downtimes. Hence, most maintenance strategies use a combination of the different types of maintenance strategies to limit the maintenance costs. A combination of corrective and preventive maintenance is most often used, since the usage of condition-based monitoring is relatively new and is also not always possible.

2.3 Healthcare Maintenance Services

Now we present the field and processes that are used for designing and performing maintenance strategies for imaging systems to make sure that the system can be optimally used. We refer to it as healthcare maintenance services (HMS), but note that this name is not used officially, but it is generally used as a term to depict the area that is being targeted by the research.

Each machine that is produced and retailed by Philips has the option of employing a Service Contract. A service contract can offer several services, but the most important services offered include the maintenance strategies with a guaranteed uptime of the machine. The top contracts guarantee a 98% or 99% uptime [11].

This means that, to uphold the requirements that are stated by the contract, a high reliability is of utmost importance for the systems. They should rarely fail and if so, they should be repaired fast. The contracts often include a combination of corrective and preventive maintenance, and when possible and desired, also predictive maintenance is offered. Predictive maintenance is still relatively new to Philips, therefore the current condition-based monitoring is only functional for key system parameters.

In other words, the goal of maintenance services is to employ an optimal strategy for the imaging systems such that the customers can rely on their systems to generate diagnostic quality images, to perform on demand, and to help deliver quality care. In this process, a lot of different positions are employed in order to be able to deliver the best services that can be offered. In this masters research project however, three major actors are involved: the remote service engineers, the field service engineers and the diagnostic designers. Each of these actors is described below.

2.3.1 Remote Service Engineers

Whenever a customer detects a fault or a failure of the system, the customer can report the problem and any additional details at the support team. The first line of support, i.e., remote corrective maintenance, is offered by the *remote service engineers* (RSEs).

Since the log data of the imaging system is transferred on a daily basis to a central server of Philips and since systems can be remotely accessed, the remote service engineers are able to analyze the history of the machine remotely in order to find the cause of the problem. Whenever a simple solution is possible an RSE can, sometimes in cooperation with the customer, make some changes to the system. For example, most imaging systems cannot work optimally whenever the device is not calibrated correctly. Calibration of the system is often performed during planned preventive maintenance, but sometimes systems need to be re-calibrated or some configurations need to be changed due to changes in the environment or minor changes in hardware components.

The remote service team is also responsible for the proactive preventive remote support. Based on the continuous monitoring of key system parameters, one of the remote service engineers can be notified of any anomalies and act on it before any impact is made to the performance of the system or before a failure occurs in the system which might result in broken components.

Since an RSE is not a product expert, most of the RSEs are not able to systematically analyze system logs blindly, nor is there time to analyze the event logs in detail. Therefore, there exist systematic procedures for the RSE to follow in order to find the problem that has caused the reported failure. These procedures are described later in Section 2.3.3.

2.3.2 Field Service Engineer

Finding the root cause of a problem is one of the most difficult tasks while maintaining an imaging system, since, as stated, these machines are very complex and analyzing the system logs is also no trivial task. Whenever the Remote Service Engineer is unable to exactly determine what the cause of the problem is, a *Field Service Engineer* (FSE) is employed. The FSE physically goes to the site of the machine and is able to analyze the system more closely. The FSE is able to execute system tests hands-on and observe the behavior of the system. Depending on the expertise of the FSE, the system analysis might be performed in cooperation with an RSE, since an RSE might have more knowledge about the system than the FSE.

Other cases in which the FSE is employed, is when the problem cannot be resolved remotely, for instance when it is determined that a component of the system is defect or a hardware component requires maintenance. The FSE has knowledge on maintaining the system and replacing hardware components. This way, the imaging system can be repaired on-site without moving the device. This is very beneficial, since most imaging system are quite large and it can thus be very costly to transport the system. This moreover limits the downtime of the system because it can be repaired relatively fast and be brought back into service more quickly.

Finally, regular planned maintenance jobs are scheduled for the imaging systems. During this process the FSE performs system checks, calibrates the system and lubricates moving parts to make sure that the system can operate optimally. By regularly performing maintenance on the system, unexpected downtime of the system can be prevented and it can extend the lifetime of the machine.

2.3.3 Diagnostic Designer

The final actor that is greatly involved in HMS is the *diagnostic designer*. The diagnostic designer is responsible for creating and improving procedures and regulations concerning the maintenance of an imaging system by analyzing event logs and previous performed maintenance jobs. Most diagnostic designers are product experts and are able to analyze systems by using their knowledge of the internal workings of the system. Therefore some diagnostic designers are also responsible for providing third line service support whenever a problem cannot be solved by the Remote and Field Service Engineering team. Most faults and failures that occur within an imaging system are complex and the root cause of the problem often cannot be derived directly. The procedures therefore support the service engineers in finding the root cause of a

problem. Most procedures can be thought of as decision trees or flow charts. Based on the symptoms of the problem, a likely cause and possible solution can be derived.

But since the imaging systems are so complex, not all root causes of all the system faults and failures are known. Sometimes, particular symptoms can indicate a failure of several parts. In these cases, the root cause cannot be determined. The engineer then often has to choose to replace all parts or choose which part to replace first. In most cases, the possibly broken parts are replaced one after the other, until the issue is resolved. The order in which parts are replaced depends on the price, the time to replace and the highest likelihood of resolving the issue. This is a difficult choice, since it might result in more downtime when not the right part is replaced. Based on this information, the diagnostic designer can try to analyze the system's event logs in more detail, such that he might be able to discover the actual root cause and update the procedures, which might help the service engineers in the future when they are faced with the same or a similar problem.

Next to analyzing event logs to determine root causes of system failures, it is also important to know the context of system faults and failures. Whenever an error or an anomaly within the system is reported, it is important to take note of what the system was doing at that moment or what the system has done during the last few minutes, hours or even days. If it is known which operations can cause certain problems for the system, then this problem could possibly be prevented by adapting the configuration of these operations. Furthermore, this information can be useful input for the design of new components and systems. In this way, one is able to create systems that are more reliable.

2.3.4 Research & Development

Another method for reducing the potential downtime of an imaging system is to make the systems more reliable, such that less failures should occur. The reliability could be increased by using more expensive components, adding redundancy, performing more extensive tests for example. However, as noted in Section 2.2 it is difficult to create highly reliable systems at acceptable cost levels.

To be able to improve on the current system, the system designers and developers need to know which aspects are lacking or which component break most often, and what is causing it. This input is often retrieved from the data and reports that are created as part of performing maintenance (Section 2.4) and performing system tests. The event logs were originally designed for debugging systems, and were thus made by the system developers. However, no person can possibly know everything about the system and its implementation, due to the complexity of the imaging systems. Moreover, the imaging systems are designed and improved over many years, therefore many changes have been made from the earlier systems to the newest system.

For the development of a system, the system designers are interested in the utilization, behavior and errors that occurred in the system. Not only on a global basis, but individual cases also need to be looked at. They therefore also need to be able to efficiently analyze the event logs that are produced by the systems.

2.4 Maintenance Administration

The last source of data that is related to HMS is the administrative data that is generated from performing the maintenance services.

A single system located at some location is referred to as a *site*. A site has an associated customer and the type of system that is sold to the customer. Making contact with the service center by a customer is referred to as a *call*, or a *Service Work order*. Commonly, a call starts at the time the customer makes contact and ends when the problem has been resolved. Any service activities performed by an RSE or an FSE is related to the initial call and is referred to as a *job*. Since many problems cannot be solved on first contact, multiple jobs can be registered to a call. Furthermore, whenever a part is replaced by an FSE, this is also registered and related to the respective job and call.

Calls are also created for creating a schedule for Planned Preventive Maintenance. Since the usage of the system is often tightly scheduled, it can sometimes be difficult to set a date for performing the maintenance. The call is often made on first contact and only closed after the maintenance has been performed, hence these kind of calls can remain open for several months.

Optimally, a single problem of a customer results in only a single call within the system. It is however common for new calls to be registered whenever the customer later contacts the support center again for the same problem. This is sometimes caused by calls that remain open for a period, even though a resolution was made during a job, to observe the system for a limited time in order to determine whether the performed actions actually resolved the issue.

2.5 Scoping

As presented above, many actors and different systems are involved within HMS. In order to efficiently research and design solutions for event log analysis, we need to limit the scope of the project. In this section, we first discuss the different systems and actors and then argue the final scoping of the project in terms of targeted actors and system type. In the next chapter, an analysis of the current problems is made by discussing related work, and the aim of the project is defined.

2.5.1 Imaging Systems Data Sources

Let us start with discussing the different systems for which mainly data was available. Of all the different imaging systems that are produced by Philips, the research department currently has access to event log data originating from Magnetic Resonance Imaging (MRI) and interventional X-Ray (iXR) machines. These types of systems use a consistent, but different, log format.

Pieces of both event logs are shown in Table 2.1 and 2.2 for MR and iXR respectively. When looking at both logs, with some basic knowledge on the logs, we can easily make a comparison of both log types:

Formatting:

The MR log is formatted nicely in comparison to the iXR log. The MR log uses tabs to make sure that each section is nicely aligned. The iXR uses the letter *a* to delimit the different categories. Due to this formatting, the iXR logs are much more difficult to read if the logs are only presented in raw format. The tools developed by Philips however at least parse the logs and present the text in a more readable format.

Content:

Both logs contain similar information. Basic event properties as time-stamp, severity, associated process and event message are present. The difference can be seen in the information about where the event was produced. For each event, the iXR log contains references to software module, file and line number of the event origin, where the MR log only contains some technical references to the process. From this, we can conclude that both logs originally were produced as debug logs and are thus not created with maintenance in mind.

Parallelism:

In both event logs it can be seen how events from different processes are interleaving. As explained before, this is caused by the fact that these processes are running in parallel and thus also log events in parallel. This is often cumbersome, when manually reading the logs, the flow of events is often disrupted by events from another process. The interleaving of the events might also need to be taken into account when parsing these logs.

Size:

This cannot be seen from the snippets, but the average size of MR logs is about 10 times as large as iXR logs. MR logs commonly contain hundreds of thousands of events, while iXR contain only tens of thousands of events each day. This is mostly caused by the fact that MR machines are more complex and MR machines also log more information on parameters and results. Furthermore, these parameters and results often span over multiple lines, i.e., multiple event records in order to create a nicely formatted log. While this can make parsing the logs more difficult, it is more suitable for analyzing the event logs in textual form, without the need for a special parser or formatter.

Table 2.1: A selection of MR event log records. The events of MR logs are nicely formatted into multiple records such that event logs are more easy to read. Some parts of the text have been shortened such that the table is able to fit on a single page.

Date	Time	Severity	ProcessName	ProcessId	ThreadId	FacilityField	Message
16-6-2014	02:27:40.28	I	RECON	2178	2174	SingleSc	[128] Internal reconstruction parameters:
16-6-2014	02:27:40.28	I	RECON	2178	2174		[128]
16-6-2014	02:27:40.28	I	RECON	2178	2174		[128] Reconstruction parameters independent of mix, echo and location:
16-6-2014	02:27:40.28	I	RECON	2178	2174		[128]
16-6-2014	02:27:40.28	I	RECON	2178	2174		[128] 01 00 00: noise.arr: 0.1143 0.1125 0.1378 0.1387
16-6-2014	02:27:40.28	I	RECON	2178	2174		[128] 03 00 00: noise.arr: 0.2137 0.2160 0.1550 0.1549
16-6-2014	02:27:40.28	I	RECON	2178	2174		[128] 05 00 00: noise.arr: 0.1281 0.1280 0.1829 0.1827
16-6-2014	02:27:40.31	S	oc-scan-engine	1a3c	2884	SingleSc	SingleScan ChangesState from Measuring to Aborted.
16-6-2014	02:27:40.31	E	oc-scan-engine	1a3c	2884		Aborting ScanSet ID: 227
16-6-2014	02:27:40.31	I	oc-scan-engine	1a3c	2884	ScanEngi	ScanEngineAccess ChangesState from Running to ReadyForScanSet.
16-6-2014	02:27:40.31	I	oc-scan-engine	1a3c	2884		Critical state reference count decreased to 0
16-6-2014	02:27:40.31	I	oc-scan-engine	1a3c	2884		Magnet(Control): B0 heater not switched on with timestamp 01-JAN-1900 00:01
16-6-2014	02:27:51.87	E	ProcessingService	17d8	1810		Old = 30010;Eid = 16; Tmm = GarbageCollectorThread; Utilities_GC.LOG: GarbageCollector.run() error
16-6-2014	02:27:51.87	E	ProcessingService	17d8	1810		Exception: [NullPointerException] :Object reference not set to an instance of an object
16-6-2014	02:28:21.93	E	ProcessingService	17d8	1810		Old = 30010;Eid = 16; Tmm = GarbageCollectorThread; Utilities_GC.LOG: GarbageCollector.run() error
16-6-2014	02:28:21.93	E	ProcessingService	17d8	1810		Exception: [NullPointerException] :Object reference not set to an instance of an object
16-6-2014	13:35:03.63	S	Examcards	0d74	2918	USERLOG	ExamCardsUI::ECExecutionList(UserEvent): (Action) OpenSingleScan: [SingleScan #12359,ReadyToRun,T2* LIVER ...
16-6-2014	13:35:03.63	S	Examcards	0d74	2918		ExamCardsUI::ECExecutionList(UserEvent):
16-6-2014	13:35:03.63	I	Examcards	0d74	2918		!ST:AWPLAN_SMARTPLAN_TYPE_NONE, AO30, B0S False; B1S False; B0 False; B1 FalsepatientReferenceID 0]
16-6-2014	13:35:03.63	I	Examcards	0d74	2918		ScanSet(SingleScan-call): -?.EndView
16-6-2014	13:35:03.63	I	Examcards	0d74	2918		PdfInterface(DefinePdf): IPlanScanPdf::EndView() #12359, T2* LIVER AX
16-6-2014	13:35:03.63	I	Examcards	0d74	2918		ExamCardsUI(TabPageManager): TabPageManager OnTabControlSelectedIndexChanged()
16-6-2014	13:35:03.63	I	Examcards	0d74	2918		ExamCardsUI(TabPageManager): TabPageManager OnTabControlSelectedIndexChanged()
16-6-2014	13:48:43.34	S	patsup	142c	155c	H.PASS.S	Tabletop position update: 1400.20 mm. PC-position: 1400.20 mm. PC-state: Travelling.
16-6-2014	13:48:43.51	S	patsup	142c	155c	H.PASS.S	Tabletop position update: 1370.10 mm. PC-position: 1370.10 mm. PC-state: Travelling.
16-6-2014	13:48:43.70	S	patsup	142c	155c	H.PASS.S	Tabletop position update: 1335.40 mm. PC-position: 1335.40 mm. PC-state: Travelling.

Table 2.2: A number of records that have semi-randomly been taken from an iXR system. Some properties, like software referrals, have been omitted for confidentiality reasons. Also, some of the descriptions have been shortened to fit the table on a single page.

Unit	Date	Time	Severity	Event ID	Initial Description	ThreadID
Patient and Beam pos.	14-1-2014	17:40:08	Information	580009921	Description: Command: RecallPosition.StartRecall	ThreadName: 6888
Field Service	14-1-2014	17:40:07	Information	620000001	Description: Base.Job.Performer-performer: Starting (PM- STransferPerformer@315fc07) ExportNetwork with InParameters: .ExportNetworkInParam...	ThreadName: EventThread-1
Field Service	14-1-2014	17:40:07	Information	620000001	Description: Base.Job.Queue.job: ExportJobQueue@218f99c-?AutoPushArchive.28722504....Cardiac@e5fbc: Start at 20140114174007.590000	ThreadName: EventThread-1
Acquisition	14-1-2014	17:40:07	Information	510020540	Description: Application name = Cardiac Application ID = 12300 Procedure name = CAG 15 fps Procedure ID = 10030 Total number of images = 72	ThreadName: 6772
Acquisition	14-1-2014	17:40:07	Information	510020541	Description: X-Ray duration Exposure: 4.800	ThreadName: 6772
PC-Infrastructure	14-1-2014	17:40:06	Information	730069921	Description: Command: Lowerpriority	ThreadName: 9592
PC-Infrastructure	14-1-2014	17:40:06	Information	730069921	Description: Command: Minimize	ThreadName: 9592
Patient Administration	14-1-2014	15:29:16	Information	540019921	Description: Command: SelectRevExam	ThreadName: 10544
Acquisition	14-1-2014	15:29:16	Information	510028688	Description: Applied tube protection	ThreadName: 2432
Beam Limitation	14-1-2014	15:29:16	Information	590010401	Description: User output: Autowedge follow command via EPX procedure selection	ThreadName: 5364
Patient Administration	14-1-2014	15:29:16	Information	540000004	Description: Delete examination.	ThreadName: 10800
Field Service	14-1-2014	13:04:52	Error	620000005	Description: Services.SystemController...: Creation of SystemMonitor failed. Check configuration.	ThreadName: Unknown
Field Service	14-1-2014	13:04:52	Error	620000005	Description: Services.SystemController...: Exception has been thrown by the target of an invocation. .Exception: [Tar...	ThreadName: Unknown
Geometry	14-1-2014	13:03:06	Information	060009930	Description: POST AD7 Passed	ThreadName: 5412
Geometry	14-1-2014	13:03:06	Information	060009930	Description: POST IPC Geometry Passed	ThreadName: 5412
Geometry	14-1-2014	13:03:06	Information	063000236	Description: GSC startup time is 0 seconds	ThreadName: Not Applicable

For both log types, there exist processing scripts that processes the event log into another format and possibly retrieves some additional information from the event records. After processing the event log, the information is stored in a common data format (CDF), such that tools can be made that are able to analyze event logs of different modalities. These processed event logs can be used more easily for creating diagnostic applications. For iXR, only a processing script is available, that extracts most events, excluding the software references, and some other information, like system parameters and configuration, performed system tests, and performed scans.

For MR, there additionally exists more advanced processing scripts, that is able to extract a daily MR model containing performed tasks by the system, different types of error events and other types of events, where each task and type of event has its own properties and possibly parameters. The events in this MR model describe more clearly what the machine has done and which errors have occurred, effectively filtering all the noise from the event log. The MR model is presented in more detail in Chapter 4. Since processing the event log is not trivial, some important aspects might be left out. Most records in the MR model therefore contain a reference to the origin of the event within the raw event log. By creating a link between the processed event and the log, the raw event log might be used for a more detailed analysis.

The data that is administrated by performing maintenance is stored in a global database. Each call or job can have an associated free text record that may contain additional information on the problem that was reported or what the service engineer has done during the job and whether he was able to solve the problem. Any parts replaced during a job can also be retrieved. This information can possibly be used in our approach to locate periods in which problems occurred in the system. Also the replaced parts can be used to analyze the event logs with respect to the parts that were replaced.

2.5.2 Actors

We shortly discuss the different actors that could be involved in the project in order to conclude which user might benefit most of a new method regarding event log visualization. First, we have the RSE and FSE. Both engineers have similar responsibilities, i.e., tasks related to performing maintenance on systems. Since the majority of these engineers are no product experts, they need methods to efficiently analyze problems in a system. The procedures that they use, sometimes in a strict manner and at other times as a guideline, have been designed by the diagnostic designers.

If we were to design tooling for either the RSE or the FSE, we have to gain insight into their way of working and discuss the problems and difficulties that they face. Moreover, new methods should optimally be compatible with the procedures, since the procedures are very important for performing maintenance. This is however not an easy task. Performing system maintenance is an important aspect of the healthcare business of Philips. The procedures and systems used for performing maintenance must therefore be protected and are considered a trade secret. It is therefore very difficult to retrieve access to this information.

Second, we have the diagnostic designers. Analyzing event logs is much more difficult and tedious for them, since the designers do not have procedures for finding new root causes for existing problems. The designers have to rely on their expertise of the system to systematically analyze event logs. Therefore, many product experts create their own scripts or use several different tools to make the data analysis easier. Since Philips Research works much closer with the diagnostic teams than the maintenance team, more information is available about the way that the designers and product experts perform their job. The same rules apply to the system designers and developers. They know a lot of the systems and are able to analyze event logs in detail, but also regularly create their own scripts and tool to perform this analysis.

2.5.3 Scope Decision

Now that the event logs from different imaging systems and the several actors involved with analyzing event logs have been discussed and compared, we present and conclude the decision of the scoping on this matter.

For the imaging systems, we decided to focus on the MR event logs. We came to this conclusion based on the following:

1. The raw event logs are more detailed. Even though the logs might be more difficult to parse, a subset of events might be used to research methods for analyzing event logs.
2. The processed event logs are nicely structured and give a good overview of the what the machine has done. If necessary, the raw event log can be used in addition to the processed log by using the event log references contained within an event.
3. The processed MR logs give a wider possibility on designing visualization methods. This is caused by the wider range of data and data types that is available for MR event logs.

Due to timing constraints, we later decided to currently only use the event logs that have been processed into the MR model for the proposed solution of the problem.

For target users, we decided to mainly focus on the diagnostic designers and developers of imaging systems. We can call them product experts. And when possible, some features of the proposed method might be beneficial for the Service Engineers. This was concluded based on:

1. There is too little information available on the systems used by and the way of working of the Service Engineers. It would be too difficult to try and come to an arrangement and it would take too much time to study the work of the engineers such that the problem and goal of the project could be well defined. There is much more information available on the diagnostic designers and they are also more approachable.
2. The work of Service Engineers is well defined and structured. Even if we were to have access to the procedures it might be very difficult to create a method which can be well integrated into their workflow. The product experts are just more fit as a target for researching methods for improving log analysis.
3. The product experts might be a better fit for designing a system that uses an interactive visualization. Commonly, they have time to carefully analyze event logs. The Service Engineers are more focused towards performing certain steps, which are defined by the procedures, such that the root cause can be found efficiently.

Chapter 3

Problem Analysis

Having discussed the main workflow of the healthcare service maintenance process, the actors involved, and the problems and difficulties faced by them related to the log event data, we next present related work involving (event) log data and discuss if and how the related work can be applied to the MR log data. By comparing the feasibility and benefits of each approach, we argue why we have chosen for the approach made in this project.

3.1 Related Work

As discussed, the current problem of the event logs is that there is little tooling available to visually analyze the system logs and most often this means that the user has to analyze the event logs in textual form. To make this process easier, we wanted to develop a tool that uses visual data exploration and interactive visualizations. Visual data exploration aims at applying the perceptual abilities of humans to the data exploration process of large data sets [7]. The basic idea is to present the data in a visual form, enabling humans to gain insight into the data by drawing conclusions from observing and interacting with the data. In comparison to the textual form of the data, where only a few hundred items can be displayed in a plain way, a visual representation can show large amounts of data in different ways, allowing the user to see patterns or interesting data points in a single view.

During the start of the project, neither the exact user target nor the problem that we wanted solve was well defined. To further limit the scope of the project, a literature study has been performed. In this study, we looked at methods to improve the current diagnostic tooling from Philips, as well as new approaches. During this study, the more advanced processing scripts, that process the MR raw event logs into the MR model, as presented in Section 2.5.1, also became available. In the related work, we therefore looked for methods that might be suitable for either the raw event logs or for the MR model.

In the following subsections, research performed and tooling developed by Philips is presented first, after which we discuss related work that propose approaches that could be used to improve the work that has been performed by Philips. We present two categories of related work: event timeline visualizations (Section 3.1.2) and event data mining visualizations (Section 3.1.3).

3.1.1 Philips systems

In this section, we present some of the research and development that has already been performed by Philips regarding the analysis of event logs. The methods developed by Philips mostly focus on analyzing the event logs using advanced algorithms and most methods do not rely on any visualization methods.

First we present the Common Analyzer Tool (CAT) that has been developed by Philips and is commonly used by field service engineers for diagnosing a problem in medical imaging systems. The analyzer tool is highly dependent on a log pattern knowledge base. This knowledge base has been build up by the field service engineers amongst others. A log pattern is a set of rules involving a set of log events types. The event types are defined by regular expressions where variables are allowed. The rules specify a set

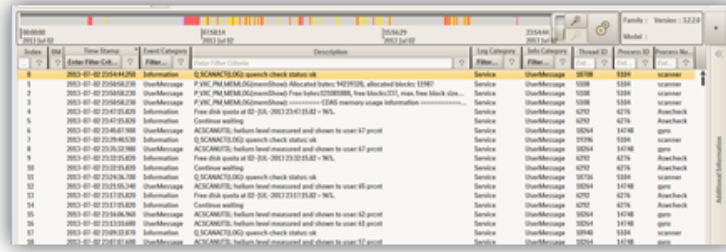


Figure 3.1: A screenshot of the Common Analyzer tool.

of conditions that must hold within a particular time frame. For example, let’s say we have event types A, B, C and D . Then a rule can be the occurrence of event A , followed by event B within 10 seconds which is not followed by event C or D , again within 10 seconds. Or in a formula (without time constraints): $A \wedge B \wedge \neg(C \vee D)$. Whenever a pattern matches against the events in an actual event log, then it is likely that the problem that is described by the pattern has occurred in the system. For solving the issue, one or several possible solutions are associated with the log patterns.

CAT first parses an event log, and then performs an analysis on the event log using the pattern knowledge base, and finally presents the results. The results are presented by showing a pattern timeline and the actual event log. The pattern timeline highlights the periods in which log patterns have been found. The event log is presented in textual form and the records that are involved in a pattern are also highlighted. Figure 3.1 shows an example of CAT.

It can be very tedious to manually define the log patterns. The correctness and precision is dependent on the knowledge of the field service engineer. Therefore research was performed to find reactive log patterns in an automated way. In this research, the events that occur when something was wrong with the system are most important. The goal is to find event log patterns that correlate well with the replacement of parts. These log patterns contain the sets of events that occur with high probability whenever a part is broken and likely do not occur otherwise. These patterns can then be used by CAT for example for the diagnosis of problems. Other research areas include the development of anomaly and machine learning models. These models can also be used for reactive maintenance, but more important, some of these models can be used in predictive maintenance in order to predict customer calls or problems in the systems.

Another tool developed by Philips is the MEBEF (Mean Exams Between Exam Failure) dashboard. Unlike the methods above, MEBEF is not designed for diagnostic analysis. MEBEF is a reliability feedback framework that provides insight in product utilization and product behavior by visualizing metrics and aggregated data. It was created for the monitoring of product quality. Figure 3.2 shows examples of MEBEF. First, utilization metrics are visualized that have been calculated based on the event logs of a few months. Examples of the calculated metrics include the type and number of procedures performed, and the average duration of these procedures. Second, the dashboard shows which errors occurred, on which days they occurred, and the impact these errors have on the MEBEF score. The metrics are mostly visualized by simple charts, but there is also a timeline visualizing the MEBEF score and the amount of occurred errors for each error level on each day. The tool however shows very little details for individual event logs.

3.1.2 Event Timeline Visualization

The first approach that comes to mind is to visualize the events on a timeline, since events are nothing more than timestamped, instant or interval, messages with some properties. A simple method is to take a time-axis and then draw the events at a position relative to their timestamp and duration. Different properties as color, size and shape can be used to show different properties of the events.

There have been many examples of using timelines, both primitive and more advanced. A standard timeline can for example be used for project planning, where icons or shapes are used to depict a moment of choice or a particular action and where lines or bars are used to depict the start and duration of the task. These simple timelines are also often used to easily show time sensitive relations between events,

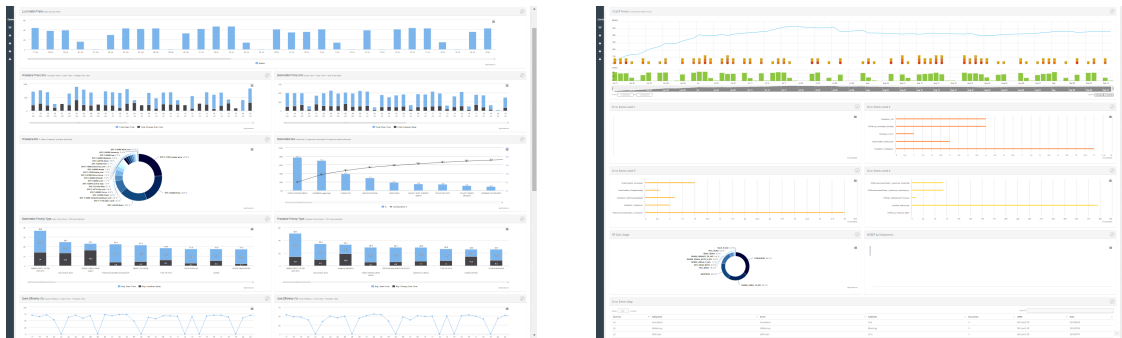


Figure 3.2: Parts of the MEBEF dashboard. **Left:** Visualization of utilization metrics. **Right:** Visualization of errors and error aggregations.

e.g., event A happens before, after or during event B [2]. In this section we however discuss some more advanced timeline visualizations.

Eick et al. [4] and Takada and Koike [17] both propose a timeline visualization technique, implemented in the applications SeeLog and MieLog respectively, to graphically display large log files and offer interactive features in order for administrators to analyze the log files in a more efficient manner.

Eick et al. proposed a method in which the log files are presented in a photo-reduced manner such that the log can be displayed in a single figure. They use blocks of one hour intervals and each line of text in the log is visualized as a one pixel thick line, where the horizontal position and the length of the line correspond to the indentation and length of the text in the log respectively. Each line is assigned a color depending on the statistic or category of the record. Since each record spans multiple lines and each line of text has a different length, some kind of shapes are formed. Figure 3.3 shows a one hour interval as it is visualized in SeeSoft.

They argue that the expert is able to identify key records based on the shape of the record. The application offers some interactive features to filter the data set and changes the statistic used to determine the color of the events. Also by hovering the mouse or creating time-windowed selections, the actual events are shown using the same colors as the visualization.

Takada and Koike improved on this idea. Their main concern with the approach of Eick et al. and other approaches was that the user needs a lot of expertise on the event data, since the user has to define keywords that can be used for filtering and color assignments to extract unusual log messages. To solve this, Takada and Koike also visualize statistical information on the log data. First, they compute the amount of log records in each unit of time for a periodical time span, i.e., daily and hourly, as well as for the entire period of the log. Second, they compute the frequency of individual tags, each series of two words and custom defined keywords, where tags are defined by the log format.

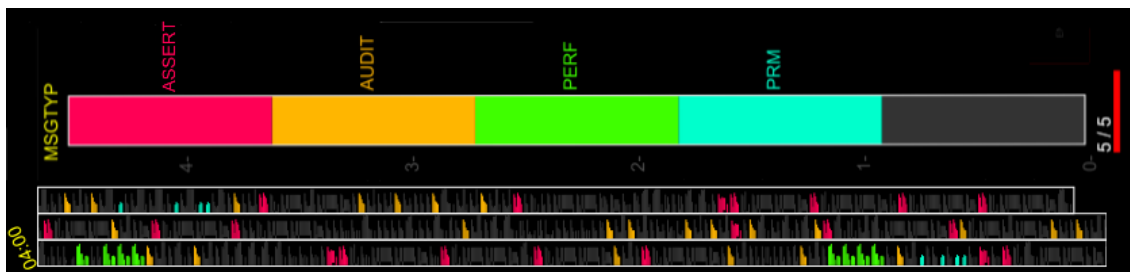


Figure 3.3: Two snippets stitched together (rotated) from the SeeSoft application [4]. The visualized shapes are a photo-reduced version of the actual event record in a computer event log. A color filter has been set as shown by the legend, indicating different different event types with different colors. The snippet shows the events that occurred within a single hour, i.e., from 04:00 to 05:00. The visualization wraps the sequential events into neighboring columns to be able to show all events in a single view.

These statistics are then used for several visualizations, as can be seen in Figure 3.4. In the first few areas, colors are used to denote quantities of frequencies. The tag area shows an ordered list of tag and keyword frequencies. The time area shows the amount of events for each day of the week, the amount of events in each hour of the day and finally a frequency histogram of the whole period. The outline area uses the same photo-reduced technique as Eick et al., but the color of a line is determined by the frequency of the associated tag of the log record. Lastly, the actual log is shown, but tags or words that have a low frequency value and keywords are highlighted. The user is then able to analyze the log data by observing patterns and anomalies in the statistics view and by filtering log messages by interacting with the statistics.

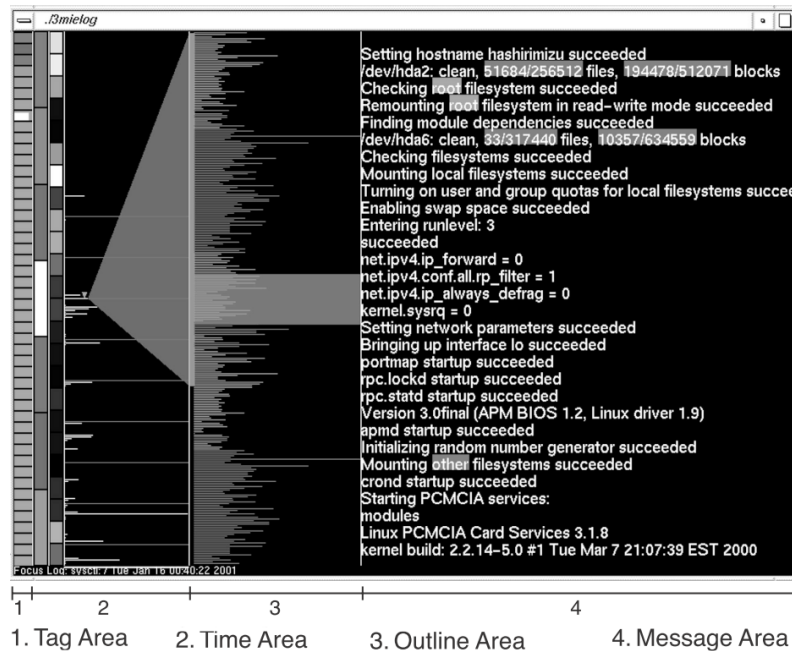


Figure 3.4: MieLog: a highly interactive visual log browser. Uses a combination of log statistics visualization, photo-reduced log visualization, keyword highlighting and interactive filtering to analyze event logs for unusual log messages [17]. (Note that the visualization uses colors, but no version with color was available)

A method that is more close to the idea of visualizing individual events on a timeline is proposed by Van Venrooij [18]. In his master's thesis he describes the design of an interactive tool to visualize the event logs of wafer scanners that have been produced by ASML. An overview of this tool is shown in Figure 3.5.

The tool aims to show what and how well a wafer scanner has been doing. The design of the tool is driven by Shneiderman's mantra [15]: *Overview first, zoom and filter, then details-on-demand*. The first two views show an overview of the system by visualizing the productivity and activities of the system. These are determined by aggregating all of the event data in the current timespan. The productivity view also marks periods of low productivity. The global cause of low productivity can be seen in the activity view. When more detail is required, more detailed views show performed tasks and the actual events that have occurred on a smaller timescale.

For describing the system, Van Venrooij shows the visualization of an event log that contains some simple cases. The case is shown with annotations in Figure 3.5. Over the timespan of a day, the system had multiple periods with low productivity (A1-A3 among others). The activity view reveals that the wafer scanner performed many maintenance activities during the day (B1 and B3). In another period (B2), a set of wafers was rejected after production. The wafers were not scanned correctly, since the fluids under the lens were not stabilized properly. The task view shows a part of regular behavior and a small part of one of the periods with low productivity. The event view clearly shows the pattern of events occurring during a normal lot task (C and D).

The tool succeeds to accurately show what a wafer scanner has been doing, but performs no analysis on the data. Just like in our case, several diagnostic methods are already offered by other tools, so there

was no focus on this while designing the tool. The productivity and activity views could however be used in addition to the diagnostic tooling, since these views provide unique features that are not offered by the diagnostic tooling. Van Venrooij therefore suggests to implement these parts, as a sort of index, into the diagnostic tooling.

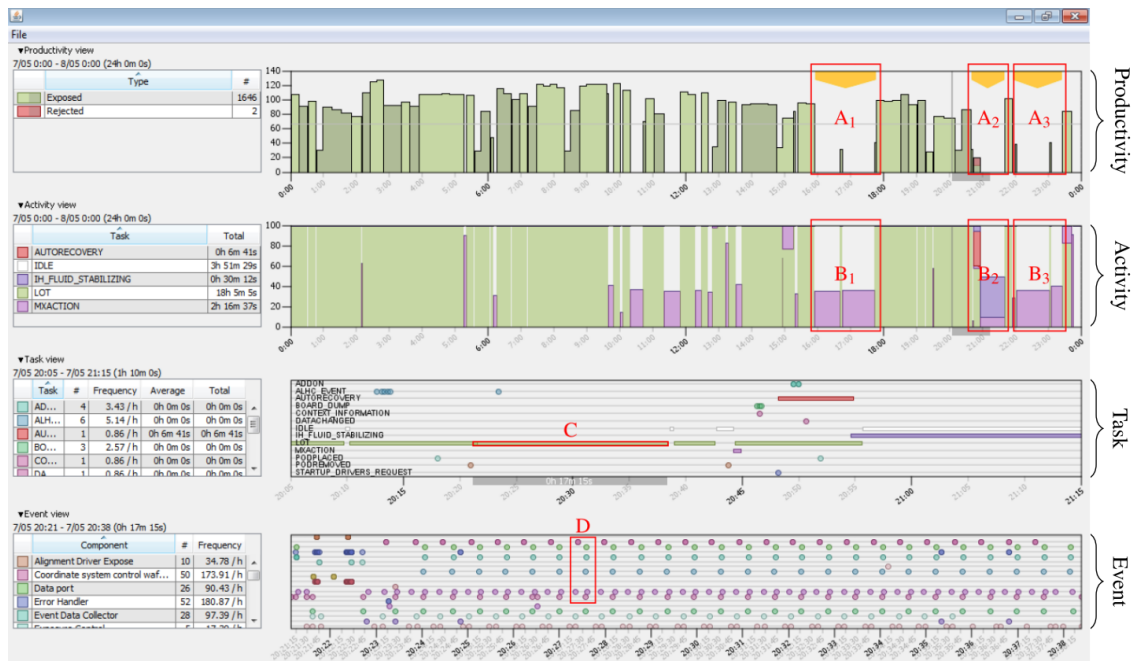


Figure 3.5: The ASML TWINSKAN event log visualizer. The tool is divided into four different views. The top two views give a global overview in terms of productivity and activities using the same scale. When desired the user can zoom in on an interval to view more detail. The third view shows the executed task of the system on a smaller scale. The last view shows the events of the system on a smaller scale. Legends and statistics of each view are shown on the left side. Note that the annotations are not part of the tool.

When using a timeline, the most straightforward method, like used by the techniques discussed above, is to use an absolute continuous time scale. When comparing event logs originating from different sources however, the absolute occurrence of events in different systems might not be interesting. In the case of our event data, while the absolute timing between events that occur during a short timespan (hours to days) in a single system are important, events occurring in different entities might have no relation in absolute timing, e.g., the same failure can occur in different systems at different times. A different method that is more suitable for comparing events related to several sources is to observe the events as sequences, where only the relative time between events matter.

Wongsuphasawat et al. [21] propose a method for visualizing these temporal event sequences. They base their method on the LifeLines2 application that has been proposed by Wang et al. [19]. Figure 3.6 shows both approaches in an application called LifeFlow, which has been developed by Wongsuphasawat et al. The visualization on the right side presents the relative timeline of multiple patient records as proposed by Wang et al. The events are aligned by the first event (arrival), but events in the timeline can be aligned by any of the available event types. In this way, the user is able to analyze the event logs to find precursor, co-occurring, and aftereffect events. Wang et al. later improve on their work by also visualizing the distribution of the events [20].

Since the event logs are only presented as individual events or by the distribution of single events, it is difficult to analyze patterns in event sequences. Therefore, Wongsuphasawat et al. introduce a scalable visualization that provides an overview of all the event sequences, as seen on the left side of Figure 3.6. In order to create this visualization, all event sequences are aggregated into a hierarchical tree. This hierarchical tree is then visualized like an Icicle Tree, which also shows the mean length between events in

the sequence. Just like in the timeline visualization, the user is enabled to select on which event type the tree will be aligned. According to their user studies, LifeFlow can be used to answer questions about the prevalence of interesting sequences, find anomalies, and gain significant insight from the data.

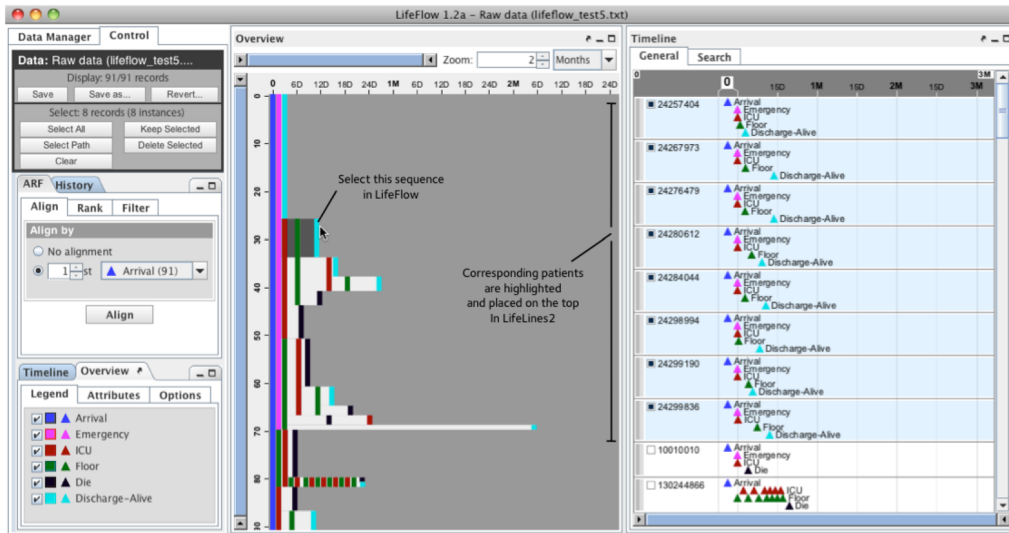


Figure 3.6: LifeFlow: The applications visualizes an overview of event sequences as well as sets of individual event sequences, allowing easy and detailed analysis of regular and unusual event sequences. [21]

3.1.3 Event Data Mining and Visualization

Most of the approaches discussed in the previous section all imply that all the event types and sometimes the properties of events are known. Even if the event types were known, these event types need to be manually defined in some form. A common approach is to define the event types using regular expressions or other similar approaches that uses templates. This can be very challenging and time consuming when the amount of events and event types grow considerably. Another approach is to automate the discovery process of event types. By automating the event type discovery process, the analyst only needs to verify the accuracy of the output and adjust the discovery algorithm accordingly. Depending on the approach it might also be possible to distinguish between semantically relevant and non-relevant (sets of) records.

An other aspect that is not discussed by the previous work is that complex systems can produce events in parallel, where some of these parallel processes are related. Due to these different processes, events of several processes are interleaving. The approach used by Wang et al. [19] and Wongsuphasawat et al. [21] for example, can therefore not be applied in a trivial manner. The challenge is to design a pattern finder that can process the log files and group several event types into a concise representation of processes or sets of events that are often related.

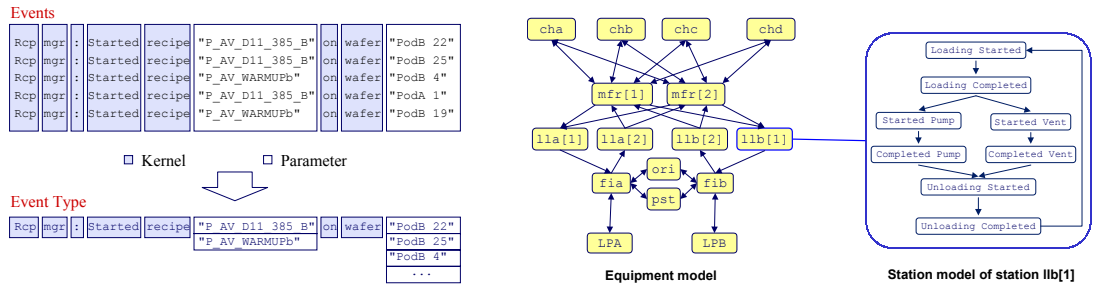
The aspects of a large number of (unknown) event types, and the production of events in parallel both hold in the case of the medical imaging systems of Philips. Though many event types and patterns have already been defined for usage in the CAT-tool. In this section, we discuss other related work that use data mining techniques as well as visualize the event logs. Note that these methods are not state of the art in process or data mining of event logs, but we present these methods because they use the mining techniques as a preparation for visualizing the event logs.

The methods discussed below come close to the field of process mining. Process mining is a set of techniques that can be used to discover models describing processes, organizations, and products based on event logs. While the output of process mining can be visualized, we did not research any methods involving process mining, since we want to stay close to visualizing the event logs.

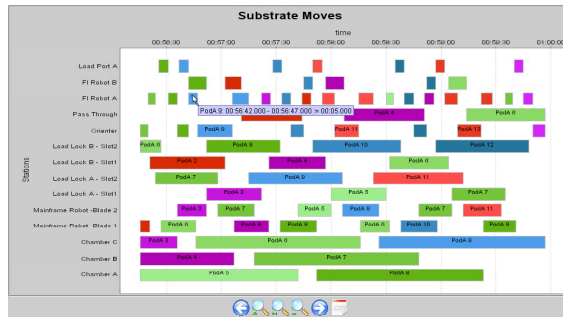
Assisted Data Mining

Röder et al. [13] address the need for evaluating equipment productivity and performing diagnosis using state-transition models of the equipment. These models are however not provided by equipment manufacturers. Manually constructing such models from log files is a time-consuming process considering the unknown structure of events in the logs. In their case, the analysis is made on event logs from semiconductor equipment. They therefore propose a method to automate the constructions of the state-transition models. This method consists of two global steps. First, text analysis is performed to aggregate events to event types for each individual equipment module. Second, finite state machines are generated from the sequence of events. The approach is shown in Figure 3.7.

In the first step, before text analysis is performed, the name of modules and substrates are identified manually. The identification is done manually, because the names of the modules are not used consistently in the event logs. Note that this only needs to be performed once for a type of log. Then, the log is filtered and text analysis is performed for each individual module. They use a combination of *prefix trees* and a modified version of the Levenshtein distance to group events into event types.



(a) Reconstruction of the event type from five events. (b) Hierarchical equipment model and station model constructed from an event log



(c) Gantt diagram for station and substrate model

Figure 3.7: Approach of Röder et al. [13] to automate the construction of state-transition models. (a) Based on the similarity of the events, the text analysis algorithm is able to distinguish the core words (the kernel) from the variables (parameters) using a modified version of the Levenshtein distance. (b) After retrieving sequences of move events and sequences of event types for each module, an equipment model and, per station, a station model is constructed and (c) the movement and processing times of substrates is visualized.

After discovering the event types for each module and extracting move events, the equipment model and the station models can be constructed. Both models are constructed based on the temporal sequence of events. The equipment model is constructed by the sequence of move events of a single substrate between stations. The station model is constructed by the sequence of event types of the corresponding station. Since the event logs are filtered for each module, each station model only consists of states and transitions of the corresponding model.

Next to visualizing the state machines, Röder et al. also use the meta-data of events, e.g., timestamps, involved station and substrate, to visualize a timeline showing the movement and processing times of the substrates in several stations, and they show statistical charts to show the equipment productivity. The

different visualizations can be used by an analyst to evaluate the model accuracy, detection of erroneous states and productivity of stations within a much smaller timespan than the manual evaluation, i.e., 30-40 min and 6-8 hours respectively [13]. A compromise was made between accuracy, development support and corrections by an analyst, where the accuracy is mostly influenced by the quality of the event log, the accuracy of the manual assignments, and the ratio between the size of the kernel and the amount of parameters.

Full automated Data Mining

Aharon et al. [1] propose two methods that deal with the challenges of creating an event dictionary, and finding event patterns or sequences in event logs with interleaving events.

The first method, the *online dictionary creation algorithm*, first clusters events into event types. Just like the previous method, they rely on the fact that events of the same type are produced by the same message template and thus have many words in common. Instead of using the Levenshtein distance, a cosine similarity is used to compare the messages from two events. An essential difference with the previous method is that clusters are further split based on the entropy of the cluster. For example the events "The network is up" and "The network is down" are very similar, but it is important that they belong to a different event type. Clusters are split when a word position in the cluster has a low entropy, i.e., there is a variation of words and some words occur often at that position.

The second method proposed by Aharon et al., the *Principal Atoms Recognition In Sets* (PARIS) algorithm, is designed for identifying sets of events, referred to as *atoms*, that belong to one process or failure. They assume multiple processes produce events in parallel and the events that tend to occur together are not always reported in the same order. By converting the input events into sets event types, the PARIS algorithm is essentially able to build another, much smaller, representation of the original event log.

The proposed methods are validated using artificial and real data sets. For analyzing the event logs, they visualize the events on a timeline where the event types are ordered vertically by the event id. An example of a use case is displayed in Figure 3.8. For the PARIS algorithm they show that the algorithm is able to accurately create atoms that clearly make a distinction between processes, which may differ in only one event type. Furthermore, the algorithm was able to find a failure pattern that was being produced by an old script that was still being run regularly. Finally, they note that the dictionary can be used as an index, by storing only event types and varying words of each event type, which can save up to 90% space compared to only storing distinct messages as an index [1].

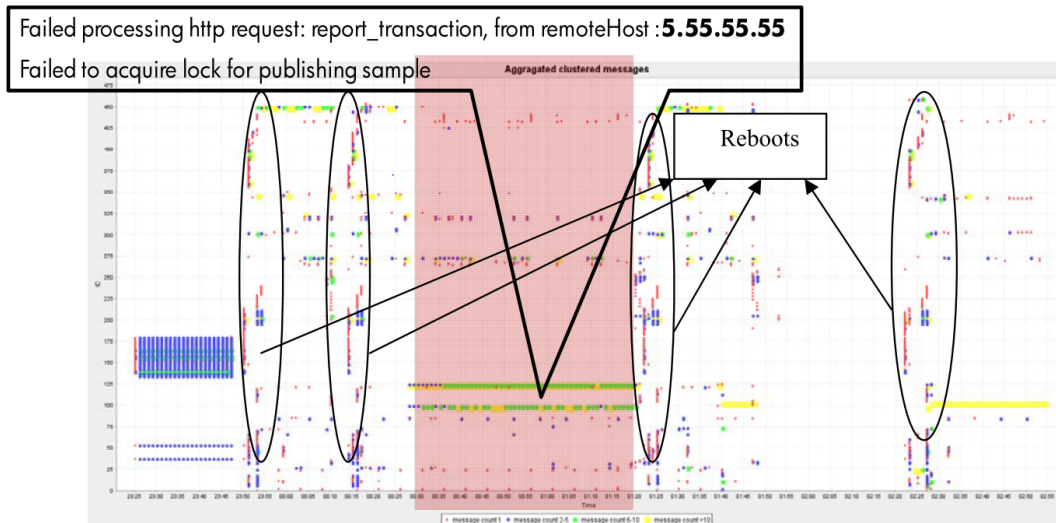


Figure 3.8: Visualization of event types on a timeline [1]. The vertical axis depicts the several different event types found in the log and the horizontal axis is the time. By analyzing the visualization, root causes of problems and patterns can be found. The red region shows the time in which a problem occurred in the system. The marked event clusters were deemed to describe the root cause of the problem. Furthermore, the oval regions mark a pattern of events that occurred when the system was performing a reboot.

3.2 Application of Related Work

As discussed in Section 3.1.1 Philips has already performed some work on analyzing event logs, but most of these methods do not use the idea of interactive visual data exploration, and the one that does, is not targeted towards analyzing event logs for root cause discovery or for system performance analysis. In this section, we discuss if and how the related work can be used to design an interactive visualization system that is targeted towards the diagnostic designer and use either the raw or processed MR event logs.

This section is structured as follows: First we compare the scaling and properties of the MR data with the data used in the related work. Second, Section 3.2.2 discusses the applicability of the interactive visualization methods to the MR data to analyze events on a timeline. Section 3.2.3 discusses the applicability of the methods that use a combination of data mining and visualization to process and present event logs. Finally, the approach that was chosen for the master project is presented in Section 3.2.4.

3.2.1 Data comparison

Before we discuss the applicability of the methods that have been proposed in the discussed literature, we compare the data that has been used in these methods with the MR event data. We compare the data on the following aspects:

Range of Data:

The timespan that is present in the event log or the timespan that is used for analysis.

Amount of Data:

The frequency of event production and the total amount of events used during the analysis.

Event Diversity:

The number of different event types that is present in the log or is used during the analysis.

Event Severity:

The proportion of regular behavior events to events that indicate errors or anomalies.

First the data aspects are presented for each method separately and then a short comparison is made discussing the similarities and differences. The data aspects are also summarized in Figure 3.9.

Philips MR event logs (Raw and Processed)

First, we present these aspects for the MR log data. Since the processing of MR logs into the MR log model results in a reduced version of the raw log, we present the details for both of them. The MR event logs contain events from a single system and, most often, a new log is created at midnight. The maximal timespan of a single log is therefore 24 hours. The analysis of the event logs could however cover a longer period. An analysis could for example be made over all the available data of a system, which spans up to two years (2013-2015). Of those two years, up to about one and a half years have been converted into the MR model.

The MR imaging systems log events at a high rate. The actual frequency is however difficult to define, since the event logs are cluttered with debug messages, reporting the values of parameters and variables. To keep the log nicely formatted, often several records are used for this. Most records use a single line, but on rare occasions multiple lines are used to print a long message. In approximation, the rate of relevant events is larger than 1Hz whenever the machine is actively being used. The processed logs only contain a small subset of the reported events. The total number of original records ranges from 200,000 to 2,000,000 depending on the usage and version of the machine. In comparison, on average, events in the processed log occur on a scale of tens of seconds. The MR model contains about 5,000 up to 20,000 events per day, when the system is used normally to perform a few to tens of examinations per day respectively. There are some systems that are heavily used. For these event logs, the number of events can grow up to about 70,000 events. But the event rate is irregular, since more events are logged while the machine is actively being used. Furthermore, multiple event records are used to create a single event in the processed log.

For the same reasons as above, it is also difficult to estimate how many different event types are present in the raw event logs, but it is likely that there are hundreds of relevant event types. For the processed logs, currently about 21 core event types can be distinguished, but some event types could be split further into more detailed event types in a way that is similar to Aharon et al. [1], where event types are split further based on the entropy of words in the event.

Finally, the distribution of events is skewed towards regular events in both the raw and processed event logs. Error events can give an indication that something went wrong, but the error itself often does not give the cause of the problem. The cause has to be analyzed based on the behavior of the system and status of regular events and parameters.

SeeLog

Eick et al. [4] use their photo-reduced log visualization method to analyze logs that are produced by the software of telecommunications switches during lab testing. Unlike the MR event logs, their log format is not regulated, and events are unstructured and span multiple lines. For extended testing sessions, the log may contain over 50,000 lines. In their examples, the log contains just over 30,000 lines for a total duration of eight hours, which approximately results in one log line per second. Log messages are categorized into main- and sub-categories and also into event types. The log contains five main categories, that are split into a maximum of 24 sub-categories, and there are a total of 253 event types. Even though the log is somewhat cluttered with noise, most of the log messages are relevant for analyzing problems in the system.

MieLog

Takada and Koike [17] rely on structured logs to provide visualization and statistical analysis. Their method is able to visualize data from a single or multiple systems. They do not specifically report the kind and amount of log data used. The data aspects are therefore based on the examples which are used to discuss their method. The examples show the logs of a single system spanning one week, where each day consists of hundreds of events. The system however scales well in the amount of events. Their method therefore also supports the combination of multiple event logs from the same period. Events can be logged either consistently or less periodically, so the total size of the logs is difficult to derive. The event diversity and severity cannot be derived from their example, but their method is more or less suitable for any kind of log data that has a consistent log format such that tags and user-defined keywords can be extracted.

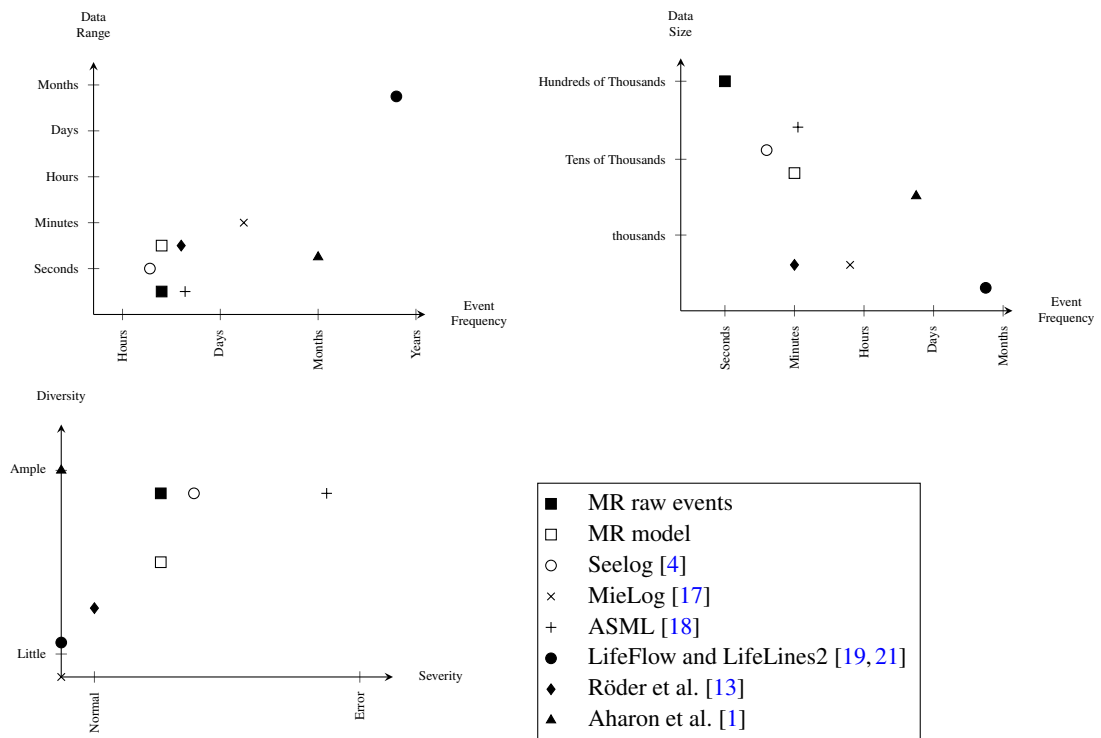


Figure 3.9: Schematic comparison of our data and data from related work.

ASML TWINSCAN Event Log Visualization

Van Venrooij [18] uses the event logs of a wafer scanner to visualize the productivity and history of a system. For most systems, at least a half year of data is available, but the tool only shows a few days of data. An above average machine logs about 60,000 events a day and individual events are produced at a rate of about 1Hz. The actual rate of events is much larger, but the event logs are filtered such that more or less only relevant events remain. A custom parser is used to parse the event logs which contain up to 250 distinct events. Furthermore, events can be mapped to one or more of the 155 different tasks performed by the system. Since wafer scanners are continuously producing wafers, the system reports events at very regular intervals.

LifeFlow and LifeLines2

Wongsuphasawat et al. [19] use data from electronic health records to show the history of patients in a hospital. While more information was available, they focused on the transfer sequences of patients between medical rooms. They started with a set of just over 7,000 patients, which was filtered to eliminate uninteresting cases. The events of each patient span a period of a few days up to three months. While the system scales well in the amount of data, it does not scale well in the amount of different event types, otherwise the amount of different sequences will clutter the visualization. Therefore only a few event types are used. Their approach is not targeted towards analyzing root causes of problems, so no distinction is made between the severity of the events.

The data used by Wang et al. [19] is also originating from electronic health records. The main differences are that they focus on the sequence of diagnosed diseases and the range of their smaller set of patients spans multiple years. The event diversity and severity is the same.

Log-based State Machine Construction

The text analysis and state machine construction method proposed by Röder et al. [13] use event logs of semiconductor equipment. In the analysis they use the logs of up to several days to analyze the productivity and behavior of the equipment. A small log contains approximately 2000 events and 22 event types for a single equipment type, where a factory can contain up to a hundred types of equipment. Most events in the log describe the regular behavior of the system.

Text clustering and Pattern Discovery

Aharon et al. [1] have tested their text analysis and pattern finder algorithms on several different log sources. The range of data varies from a few minutes (a printer press) up to three years (windows events), which contained over 66,000 and 480,000 events respectively. Their largest source contained over four million events that have been gathered over the course of eight months. They applied their event clustering algorithm on all sources, discovering up to 4100 event types. Depending on the source the distribution of the severity of events varies, but they shown that the PARIS algorithm is able to detect event patterns that indicate common failures that occurred in the system.

Comparison

On immediate comparison can be made on the type of data used by the related work. Excluding the patient history used in LifeFlow [19], the event logs that are used, originate from software that is run on some kind of system. Moreover, aside from filtering the event log beforehand, these methods also seem to use a form of raw event data.

The aspects that influence the approach and visualization the most are the data size and the event diversity. When comparing these aspects with a typical raw MR event log, i.e. large amount of events and event types in a single day, then the event logs that are most comparable, are the ASML TWINSCAN event logs [18] and one of the logs used by Aharon et al. [1], namely the printer press log. For the processed MR logs, when looking at the number and diversity of tasks, also the ASML TWINSCAN event logs are most comparable.

3.2.2 Interactive Event Timeline Visualization

As discussed in Section 3.1.2, relatively simple interactive visualization methods can be used to visualize and analyze event data. The CAT-tool only visualizes the outline of the log patterns that are present in an event log and MEBEF only aggregates data and shows a summary of the usage and the occurred errors in an MRI system. An approach that uses a time-line to visualize individual events might be able offer event log analysis in a different way. In this section, we discuss if and how the proposed methods can be applied to the MR event logs.

Eick et al. [4] and Takada and Koike [17] both visualize a photo-reduced version of the event logs. In the case of MR raw event logs, little insight can be gained from analyzing the outline of a log, since the visual structure is highly irregular and dependent on the usage of the system, so the outline of each log is likely very different. We are more interested in analyzing the system behavior when problems occur. It is very unlikely that the outline can contribute to finding periods with errors or contribute to showing the utilization of a machine. Furthermore, the statistical analysis and highlighting of Takada and Koike has little benefit, since CAT already shows interesting regions in the event logs. These methods cannot be applied to the MR log model, since the processed log has no visual structure.

Wang et al. [19] and Wongsuphasawat et al. [21] use the medical history of patients to analyze event sequences for the diagnosis of deceases and transfer between medical rooms respectively. They exploit the small number of event types and the well defined sequence of events. Medical events do not interleave and most events in the sequence are related to each other. Due to the limited knowledge of the structure and the content of MR event logs, we do not know for sure how important event sequences are. Furthermore, events of different processes are interleaved. Some interleaved events might be related while others are not. Hence, we cannot say whether processing events separately for each individual process might result in correct sequences or not. Finally, the amount of traces for the raw event would be much larger and their method does not seem to scale well in the amount of traces. The importance of event sequences might be more easily defined for events from the MR log model, since fewer event types exist.

The log data used by and the goal of Van Venrooij [18] are very similar to our case and his method of overview first, then zoom and filter on demand can be very well applied to the MR logs. This is especially true for the MR log model, since event types and properties are nicely defined. Ultimately, we would also like to provide some options for failure diagnosis, but this is not something that is offered by the tooling of Van Venrooij. But being able to show what a system has been doing during a day, and being able to show which events occur during the time that the system is behaving unexpectedly, is also a large interest in this project.

The productivity of wafer scanners is a very important aspects of these systems. Therefore, the tool designed by Van Venrooij is driven by showing the productivity and showing more details when the productivity is low. For the MR systems an aspect of productivity could be defined in some way. For example, some people analyze the log data to see how efficient an MRI system is used by an hospital, e.g., by looking at the average time between scanning two patients. But this is a complete research topic by itself, and is therefore not suitable for us. Furthermore, it might be difficult to define an aspect for MR logs that measures the quality of the system at any given moment. We therefore have to find a different method for finding interesting data points for a system.

3.2.3 Event Log Data Mining and Visualization

There already exists a custom parser for MR event logs and, as discussed in Section 3.1.1, Philips has manually created expressions for event types to discover event patterns. However, we still could research and apply more generalized or independent data mining techniques, like the techniques discussed in Section 3.1.3, to a subset of the events and in addition use visualization techniques for diagnostics. However, next to analyzing the event logs using data mining, the methods discussed also visualize the results of the analysis, which might also be applicable. Below, we discuss the applicability of the mining and visualization methods that have been presented in Section 3.1.3.

Data Mining

To resolve the issue of interleaving and irrelevant events, Röder et al. [13] use manual assignment of words to stations and objects to filter each log for the corresponding station. While this increases the accuracy of the text analysis and state machine construction, this process may be a too large of an overhead for MR logs, due to the larger size of the MR logs and the irregular usage of the machines. On the other hand, manual assignment might not be necessary for raw MR logs, since the MR logs are likely more consistent and each record contains a process name that logged the record.

Aharon et al. [1] mine event types independently of the processes that produces them, they improve the accuracy of the discovery by taking the entropy of words into account and their PARIS algorithm does not rely on a fixed ordering of events. In comparison with the approach of Röder et al., the PARIS algorithm is not designed to retrieve a full set of events that defines the transitions of a complete process, but retrieves small sets of events, the atoms, that tend to occur together. A method like the PARIS algorithm could be used to research a more general approach towards parsing event logs. On the other hand, Philips already has an existing event type and event pattern database that can be used to visualize the events on a timeline.

Visualization

Röder et al. [13] use a combination of finite state diagrams and a timeline to assess the productivity and behavior of equipment. If we can reliably retrieve sequences of related event types, we can use a similar method to visualize the state transitions. Röder et al. show basic finite state transitions with relative few states. Since the MR logs have likely more states and branches, we would have to find a efficient way to show the transitions. More information like transition count could be used to enhance the diagram with qualitative information such that anomalies in event transitions can be found.

The question is whether generating and visualizing state machines is an efficient method for analyzing failure diagnostics. We have very limited knowledge of the internal workings of a MR system and the states and transitions in terms of events might be highly dependent on the usage of the machine, making it difficult to analyze the applicability of the state diagrams. Furthermore, for MR systems, there is no productivity involved and the performance is not as important as the behavior and failure of components.

Aharon et al. [1] visualize the events on a timeline, where the event-types are vertically ordered. Due to the structured visualization, event patterns and anomalies can be found visually. The results of the PARIS algorithm are however not visualized. A simple improvement can be to cluster the event types on the vertical axis of the timeline and assign a color and/or shape to depict to which cluster the event belongs. A problem is however that an event type can belong to multiple atoms and typically there are many atoms. Due to the presence of many event clusters, the visualization can become cluttered. They do not describe if they offer any interactive features in their tool, but some interactive features, e.g., filtering, zooming and highlighting of atoms, can easily improve the effectiveness of the visualization and atom analysis.

3.2.4 Discussion

We conclude this section by discussing the approach we have chosen. From Section 3.1.1, it is clear that Philips is actively researching methods to improve the serviceability of medical imaging systems. And as discussed in Section 2.3, there are several areas in which we could improve the serviceability, i.e., by improving corrective, preventive and predictive maintenance strategies and procedures. By using data mining techniques, the current focus of Philips is to find event patterns and construct predictive models that can either confirm the root cause of a problem or predict the occurrence of failures. The data mining methods discussed in Section 3.1.3 can offer a more generalized way of discovering event types and event patterns, but these methods seem to offer limited benefit over the current approaches. So we can conclude that there is already plenty of focus in diagnostic research for reactive and predictive maintenance.

While the diagnostic research compares individual event logs using data mining, there is little focus on analyzing individual event logs or showing the structure of individual event logs. The event logs are aggregated for utilization and behavior for MEBEF, but it does not show these aspects on a daily basis. Overall, currently missing is the use of interactive visualization and visual data exploration of individual event logs for analyzing the utilization, behavior and occurred errors.

In the previous sections, we discussed several visualization methods that visualize one or several event logs. The methods that visualize the utilization and behavior are:

1. **ASML TWINSCAN**, visualizing event log history by Van Venrooij [18];
2. **LifeFlow**, visualizing event sequences by Wongsuphasawat et al. [21]; and
3. **State Machine** and substrate transition visualization by Aharon et al. [1].

Based on the discussion above and the applicability of these methods in Sections 3.2.2 and 3.2.3, we have chosen to use an approach that is similar to Van Venrooij [18], since an approach that is similar to his seems most suitable for our interest in visualizing system utilization, behavior, and errors. We moreover chose to use the event logs that have been processed into the MR Model as primary data source, since tasks, event types and properties are already well defined in that model. The MR model is presented in detail in Chapter 4.

Our approach, for designing an interactive visualization tool that visualizes the MR model event logs, is as follows:

Shneiderman's Matra

The concept of 'Overview first, zoom and filter, then details-on-demand' is very suitable for our large scale of data. There are thousands of systems that are currently being maintained by Philips. The available data of each system spans up to two years and each day can contain thousands of events. Even for a single system, we cannot arbitrarily show all the data.

Event focussed

We focus on visualizing and discovering individual events. A large part of performing failure diagnostics is to analyze the behavior of the system in the period before the failure occurred. We therefore want to carefully visualize the events in a clear way and we want to offer a method for finding periods in which particular events occurred.

Standard Visualizations

To this day, much research has been performed in the field of data visualizations and various methods are available. But the prototype is targeted towards engineers that require methods to make the log analysis more efficient and simple. The visualization techniques that are used therefore need to be simple and easy to interpret, allowing the user to focus on answering difficult questions.

3.3 Requirements

Having discussed the data we want to use, the users we want to target, the goal we want to achieve and the general approach we want to take based on the related work, we describe the requirements and approach of the project in more detail. These requirements have been iteratively determined while designing and implementing the prototype, which is presented in Chapter 5.

First, the base of the prototype was implemented, containing some basic features that have been determined in consultation with the supervisors of this project, i.e., Dr.ir. M. Barbieri, Dr.ir. J. Korst and Prof. dr. ir. J.J. van Wijk. After that, more specific requirements and features have been proposed by the service and product experts. Due to time constraints, we have not been able to implement all additional features and requirements proposed by the target users. These additional features and requirements are discussed for future work in Section 6.4. The requirements are listed below:

Functional Requirements: The functional requirements define what the system must be able to show or what features must be supported:

- **What:** It should be clear that a lot of data is available. The following requirements shortly describe which information is important to be shown. When visualizing or showing the data, no modifications should or have to be made, i.e., the data is visualized as is.

FW1. Available sites

A few thousand MRI systems are currently being maintained by Philips. There should be an overview of the sites for which data is available such that the user can easily switch between the event logs of several systems and a global comparison can be made of the event logs of different systems.

FW2. Systems properties

The MRI systems are continuously being developed to improve reliability and to add extra features or better components. The type of system is important for analyzing the event logs, since newer versions can behave differently or different aspects might be more important. Therefore, the properties of the system, like type and version, need to be displayed.

FW3. The distributions of the event types

Depending on the usage and status of a system, the number of events for each event type can vary greatly per day. A day on which a failure occurred has likely more error events than regular days for example. Also, for some days, no data is available. To give an indication on the amount data that is available, the distribution of certain event types over the history of the system needs to be shown.

FW4. Tasks and events

When inspecting a machine, it is desirable to show detailed information of what the machine has done and which events occurred over a specific interval. Most interval events in the model describe the performed tasks of a system, and instant events describe other operations or errors that have occurred. The tasks and events should therefore be clearly visualized in an intuitive manner.

FW5. Event properties

Even more important than the actual events are the properties of the events. The properties actually describe what the system has done or which errors have occurred. When visualizing the events, the values of the most important properties or all properties need to be available for inspection.

- **Interactivity:** It is not possible to simply show all the available data for a single or multiple systems. It is very easy to lose perspective when an overwhelming amount of data is shown. Interactive features can be used to limit the amount of data that is shown at any give moment. The following requirements define how the user should be enabled to interact with the system.

FI1. Data selection

By Shneiderman’s mantra, we first want to give an overview of the data, and let the user decide for which systems and for which period the data needs to be shown. So the user needs to be able to select and change the scope of the data that is shown.

FI2. Data filtering

Thousands of machines are available and each log can contain thousands of events. Even by limiting the scope of the data, still too much data could be remaining. Moreover, the user might want to focus on a particular system type or period in which certain event properties are available. The user therefore needs to be able to filter the data that is shown to further limit the scope of the data. Sites can be filtered based on attributes like the location of the site and the type of system that is stationed at the site. The events can be filtered based on the value of the event properties, like whether a scan is completed correctly or on the code of some error.

FI3. Store and load context

Whenever the user has found an interesting period within the data using a particular scope selection and filter, then the user might want to use the same period, scope or filter again at a later moment. The user therefore needs to be able to save and load any selective decisions or filters such that they can retrieve it again at a later moment.

Non-Functional requirements and constraints: Due to time constraints, some decisions regarding the performance and scope have been made. The following requirements define these performance characteristics and the constraints of the prototype:

C1. Visualizations for Single System

While it could be useful to be able to make a direct comparison of two or several systems, it highly affects the design and possibly makes the analysis more difficult if it is not designed well. Dealing with the large amount of data available for a single system is already challenging. We therefore decided to limit the visualizations of events to a single system.

C2. Real-Time Interaction

An important aspect of the goal of our approach is that the tool needs to offer an easy and efficient method of visually analyzing event logs. The interactions that are offered to the user therefore need to be available at any moment and in real-time. This means that the results of any interaction should be available within a few seconds. The largest bottleneck in the performance is the time it requires to retrieve the data. So we should try to create smart queries such that data can be retrieved as fast as possible. Another aspect is that any interaction with the visualization should be responsive. Therefore the system should be able to respond fast to any interactions performed.

C3. Uncluttered

Even though we limited the scope of the event visualizations to a single system and enable the user to filter the data that is shown, a single event log can still contain thousands of events. A display with too much information is difficult to process. The distributions and the actual events should be clearly visible on the screen, such that the value for a single day can be easily seen and that the properties for a individual event can be requested easily.

C4. Confidentiality

The data used by the prototype is property of Philips and should therefore only be accessible from within the network of Philips. The tool is designed for diagnostic designers and product experts that work within Philips. At the moment, there is no need to anonymize or limit the access to certain subsets of the data for the current intended users.

C5. Desktop Application

The application is not targeted towards remote or field service engineering. The prototype is therefore designed to be run locally on a laptop or PC with remote access to the data. Although Microsoft Windows is used by most people in Philips, we use Java to allow easy porting to various operating systems. This will hardly have an impact on the potential performance of the tool, since the data should be mostly visualized as is, and therefore should not require any heavy calculations.

Chapter 4

MR Event Log Data

As stated in Chapter 2 and 3, we have chosen to use the processed MR logs as our main data source. Before we present the details of our approach in Chapter 5, we first describe how the raw MR logs are processed and how the MR model is structured. To give some more insight in the data, Section 4.1 first shortly presents the internal workings of an MRI system, such that the data contained in the model can be presented more clearly. Second, Section 4.2 describes the global structure of the MR model and how it is constructed from the raw logs. In Section 3.2.1, we already shortly presented a few aspects of the data contained in the MR model, Section 4.3 presents some more statistics on the data.

4.1 Magnetic Resonance Imaging Systems

A Magnetic Resonance Imaging system (MRI) is a complex system with several sophisticated designed components. In this section, we shortly present the principles of an MRI system to support the description of the data in this chapter. This information is moreover beneficial for Chapter 6, where we evaluate the proposed method.

The strength of MRI is its ability to provide cross-sectional images of anatomical regions in any arbitrarily plane and display excellent contrast of different soft-tissues. First, a large magnet creates a strong magnetic field around the area of the object that is being examined. Then radio frequency photons are sent into the atoms that are contained within the tissue of the object. Due to differences in the strength of the magnetic field, tissue at one position resonates at a different frequency than tissue at another position, hence the term *resonance* imaging. The nuclear energy states of certain atoms, mostly hydrogen atoms, then interact with these photons. When the transmission of radio frequency photons is stopped, the energy state of the atoms drops and radio frequency waves are also emitted by the atoms. The radio frequency that is emitted is different for different kinds of tissue, which is exploited to generate images.

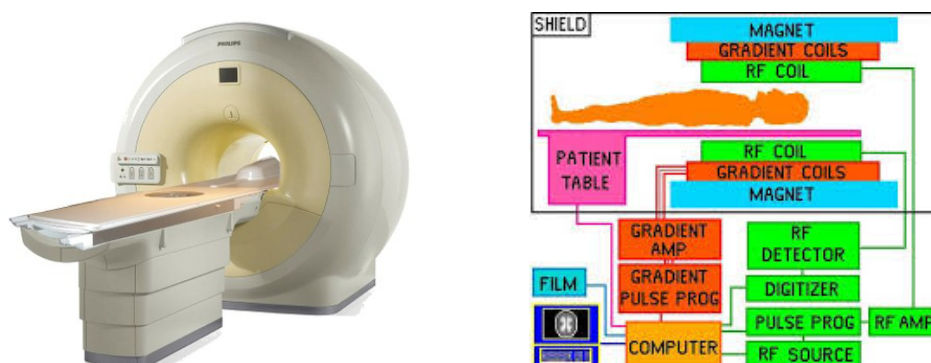


Figure 4.1: A picture of an Ingenia 3.0T MRI system [10] and a schematic diagram of the components of an MRI system [6].

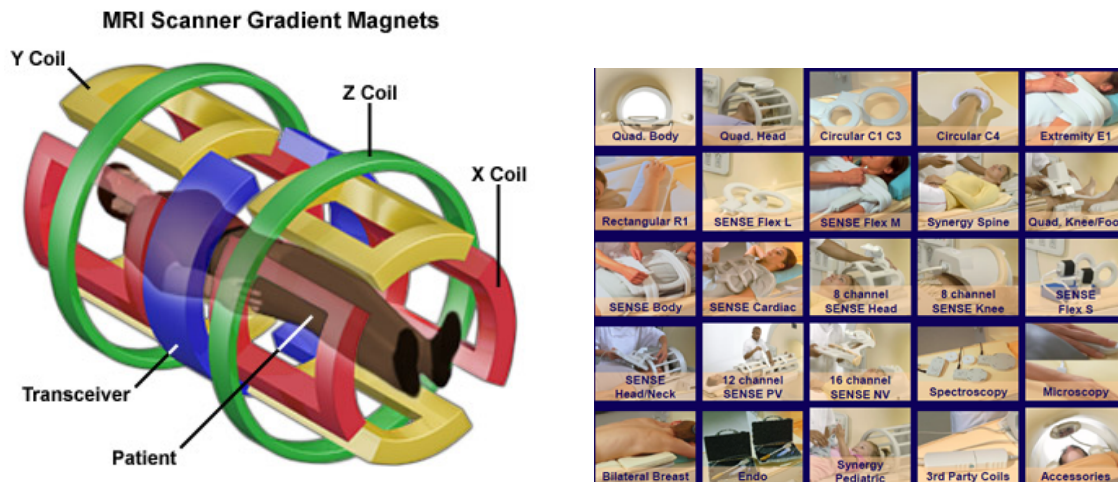


Figure 4.2: Different coils for an MRI system. **Left:** The composition of the three gradient coils [8]. **Right:** Overview of several RF coils [9].

As displayed in Figure 4.1, an MRI has several components that work together to be able to produce high quality images. The most important components are the *magnet*, the *gradient coils* and the *RF coils*:

Magnet

The core of the MRI system is formed by the large magnet. Most MRI systems use a superconducting magnet, since they are able to create a high-powered and a stable magnetic field. The conducting material of the magnet is submerged in liquid helium at 269.1 degrees Celsius below zero (about 4 Kelvin) to reduce the resistance of the wires to almost zero. The helium and the wires are contained within a vacuum container.

Gradient Coil

The main magnet creates a uniform magnetic field. To be able to create images of a body or a body part, gradient coils (or gradient magnets) need to create gradients in three orthogonal directions (the X, Y and Z axis) to distort the whole magnetic field in a certain direction. During the imaging process, the three gradient coils (as seen in Figure 4.2) are quickly turned on and off many times to create slices in the magnetic field. Each slice causes the tissue at that area to resonate at a particular frequency.

RF Coil

The Radio Frequency (RF) coils are used to create and transmit the RF pulses that are absorbed by the atoms which in turn emit RF signals that are again received by the RF coils and used to reconstruct images. For each slice created by the gradient coils, a different frequency is used. The RF coils are placed close to the area that needs to be examined and, dependent on the region, different kinds of RF coils are used. Figure 4.2 shows several kinds of RF coils.

Other Components

The rest of the components are used to regulate the process and link all the components to each other. The main computer is mainly used for this. The main computer is also responsible for reconstructing images from the RF signals that are received by the RF coils for each slice. Other components include the cooling cabinets, the patient support and amplifiers. Cooling cabinets are used to regulate the temperature of several components. The patient support can precisely adjust the position of the table on which the patient lies. Finally, the amplifiers are used to create the right signals for the coils.

4.2 MR Data Model

As discussed in Section 2.5, the logs of medical imaging systems were initially created for debug purposes and the logging process has not been adjusted for maintenance purposes. The event logs contain technical references and lots of data that is not directly relevant for maintenance purposes. To be able to more easily analyze the event logs, Philips has written scripts that extract the most important events and relevant information regarding the behavior of the system. This information is used to build an MR data model.

After constructing the data model from an event log, the model is stored in a machine friendly format. In this section, we first describe the extraction process (Section 4.2.1) and then we present how the event types are abstracted even further, such that the data structures might also be suitable for events logs of other imaging systems (Section (4.2.2)).

4.2.1 Event Extraction

First, based on sets of regular expressions and a set of rules, the scripts extract the following information from the logs:

Performed tasks and intervals:

Many events in the event log are related to some task that is executed by the system. In the MR model, multiple events are combined and reduced to a single event with certain properties, like the duration, results and parameters. Some tasks are part of some other task. For example, during an exam that is performed on a patient, a number of images are reconstructed. But before an image can be created, the system first has to perform some preparations after which a scan is made. The preparations and the scan can be seen as sub-tasks of the image reconstruction, where multiple image reconstructions can be performed during a single examination.

Events:

Other events that are not directly related to one of the extracted tasks, or individual event types that are denoted as important, like errors and warnings, are also extracted. Examples are the *user-interface* events, that occur whenever a message or a signal is sent to the user-interface. Occasionally, a user-interface event might give some more information on an error event that has occurred. Some event types have been generalized into a more general event type. Examples are the *Hardware* and *Condition* events. The hardware event depicts an error or warning that occurred for a hardware component. The condition event often denotes an important state change or an error or warning concerning the state of the system.

Properties and parameters:

Each task and event also has a number of associated properties and possible parameters. As stated in Section 3.2.1, events in the MR model are constructed from multiple events or records in the raw event log. The number of properties of an event varies heavily, e.g., the scan event has 179 properties, while the coil event only has 9 properties. However, some properties do not always have a value, which is dependent on context in which the task or event occurred. For the scan event for example, different sets of properties are used for different kinds of scan types.

Each performed task, interval or event is described by an event type. An event type is defined by a name and a set of properties. As presented in Section 3.2.1, currently there are 21 core event types in total. There is a current focus on improving and extending the processing scripts. It is therefore likely that the number of event types will increase in the foreseeable future.

Next, the events that are related to each other are linked by building a hierarchical model of the collected events. For example, the preparation, scan and reconstruction events are linked to the exam event during which these events occurred. The actual MR model is shown in Figure 4.3. Finally, the events, their properties and links are stored in an XML-formatted file.

The processing scripts only convert the log into another structure and each log day is still stored in an individual file. What was missing, was an efficient way to analyze certain aspects of several logs. Due to the absence of an index to individual events, all the logs from the required period need to be completely parsed

in order to aggregate some parts of the data. To improve the access to single events and their properties, Philips converted the structured MR model into an entity-relationship (ER) model, such that the processed event logs could be stored in a database, where the tables are the event types and where the columns are the event properties. This way the events are indexed by using a combination of a unique event identifier, the log date, and the system identifier.

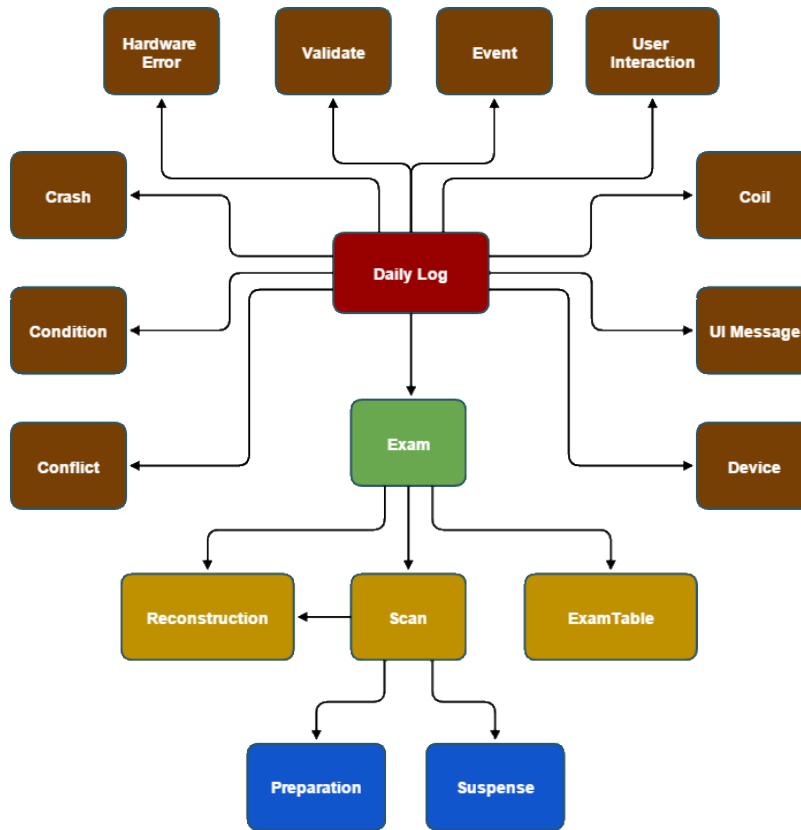


Figure 4.3: MR hierarchical data model

4.2.2 Generalization

The approach made in this thesis project exclusively uses the MR logs that have been stored in the database. Our goal however, is to create a tool that also could be used for event logs of other medical imaging systems, like the event logs of iXR systems. We therefore decided to use a generalized data structure in our implementation, instead of hard-coding the event types and their properties. As described above, the MR model mainly contains events, where each event type has a set of properties and a distinction can be made between interval events (or tasks), and normal (instant) events.

We denote an event as a tuple

$$e_i = \begin{cases} (T_i, t_i, P_i) & \text{if } e_i \text{ is an instant event} \\ (T_i, t_i, e_i, P_i) & \text{if } e_i \text{ is an interval event} \end{cases} \quad (4.1)$$

where T_i is the event type, t_i is the time at which the event occurred or started, e_i is the time at which the interval event ended and P_i is a set of valued properties. Table 4.1 shows some examples of event properties. A full list of event types is included in Appendix A.

Event	Property	Description	Examples
Exam	Exam Name	Name of the examination procedure that has been manually selected or created by the doctor	'BREAST 13-CH', 'L-SPINE LOWER SAR', 'Spalla Destra arto'
Exam	Sex	Sex of the patient	'M', 'F'
Scan	Name	Name of the scan	CoilSurveyScan, B1_calibration, AX T1 TSE
Scan	Anatomic Region	The region of the body that is scanned	SNM3.T-D1100.Head, SRT.T-D9200.Knee, SRT.T-D00F9.Lumbosacral spine
Scan	Completed	Was the scan completed successfully, aborted by the user or failed	Completed, Aborted, Failed
Device	DeviceName	Name of the device	Anterior, Posterior, NVC_HEAD_NECK, SENCE_HR_KNEE_ACI
Device	12NC	The actual reference number of the component	12 digits
Condition	Type	Type of condition, related to the severity	NetworkChange, ConflictWithPC, Malfunction
Condition	Failed Originator	The device this related to the error	Name device followed by numbers containing 12NC amongst others.
Coil	Name	Name of the Coil	NVC_BASE, Anterior, Posterior
Coil	Element	The element number of the coil that is used	E01 - E16

Table 4.1: Some examples of some properties of MR events.

4.3 MR Model Data Statistics

To give an impression of the total data size and the average utilization of a system, we present some basic statistics on the data of the event logs that are stored in the database. Appendix A shows, next to a list of all the event types, also some statistics on each of the event types, such as number of properties, the total number of event records and the average number of occurrences per day.

Figure 4.4 shows a cumulative histogram for the number of daily event logs for all the systems. Most systems have between 350 and 450 event logs. The sequence of event logs is not complete continuous however. It regularly occurs that event logs for certain days are not available. This can be caused by various reasons such as the system being broken or offline. Between the first and the last event log of a system, on average 73% percent of the days contain an event log. Moreover, it also can happen that the system is online, but that the system is not being actively used. On average, for 62% of the days an event log is available in which at least one exam has been recorded. Figure 4.5 shows the cumulative average number of exams per system per day. It shows that about 60% of the systems perform up to 15 exams a day on average and up to 30 exams are performed on average per day by 30% of the systems.

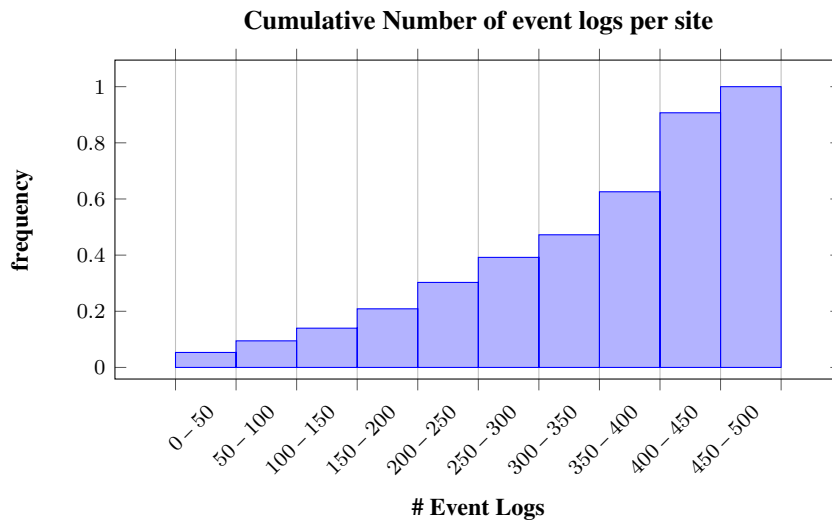


Figure 4.4: Cumulative histogram of the number of event logs per site.

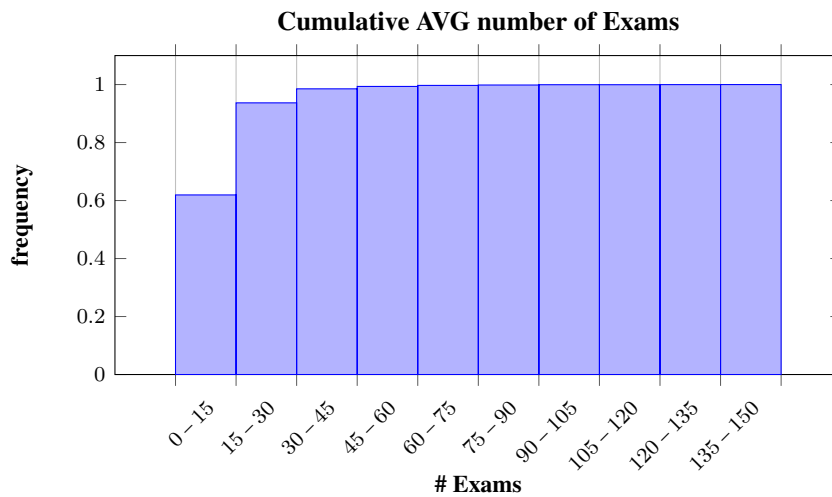


Figure 4.5: Cumulative histogram of the number of average exams per site per day.

Chapter 5

Interactive MR Event Log Browser

In the previous chapters, we presented the background on performing maintenance services, we studied and discussed related work that has inspired our approach, and we explained the main data source that is used in this project. Based on this information, we now present our approach and the implementation of the prototype in more detail.

This section is structured as follows: First the general design and overview of the system is presented in Section 5.1, after which Section 5.2 discusses how the interface has been designed based on the implemented views. Then, the data views are presented individually in Sections 5.3 to 5.5. Finally, this chapter is closed by presenting additional interactive and configuration features in Section 5.6.

5.1 System Overview

We first present a global overview of the system in terms of the back-end, or the visualization pipeline, and the front-end, or data and visualization views. The visualization pipeline of the prototype is similar to the traditional visualization pipeline [14] as shown in Figure 5.1. Each of the views in the prototype essentially has its own pipeline and depending on the goal and content of the view, the user is able to perform one or more of the specified interactions. Some views are linked with one another. Whenever the user performs some action in one view, it can have an effect in another view.

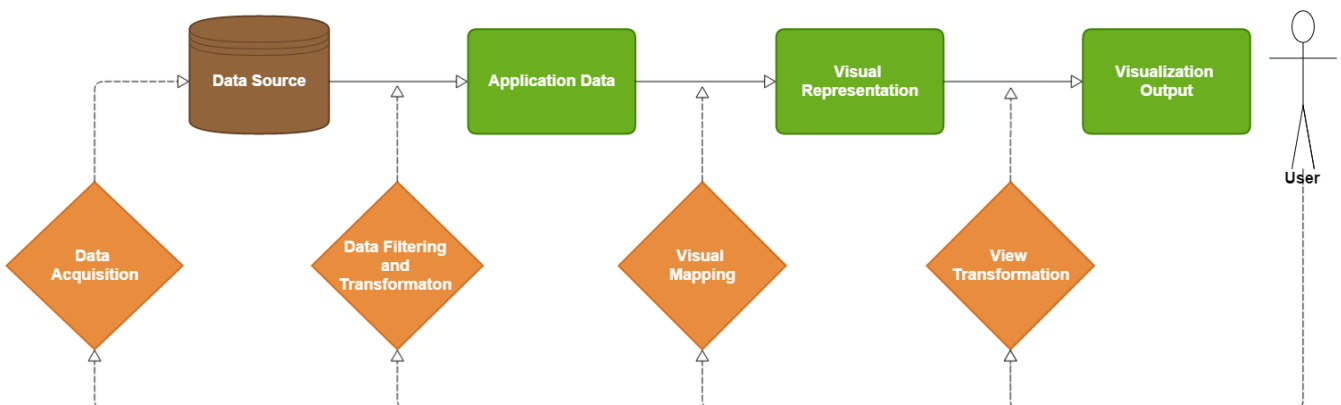


Figure 5.1: The traditional visualization pipeline. First, the raw data of a given data source is transformed to the application data. Based on the application data, a visual representation of the data is created that is finally rendered into the visualization. Each of these steps can be influenced or initiated by interactions performed by the user. The user can request new or additional data, specify data filters, configure the rendering representation, and interact with the visualization.

The visualization pipeline is implemented in the prototype as follows:

Data Source

As noted in Section 4.2, the processed MR logs that have been stored in a database are the main data source for the prototype. We take advantage of the query interface and indexes of the database for retrieving the data. It allows us to efficiently retrieve or aggregate data for any system for any arbitrary period without having to spend time and resources on parsing and processing individual event logs. For some views, the data that is loaded is determined by the user interaction in a different view. The user interactions and data flow between views is discussed below in Section 5.2.

Application Data

Most of the data structures of the application data are generalized or are universally applicable for their goal, such that the prototype can easily be adapted to other data sources. The tasks and events of the MR logs that are retrieved from the database use the generalized data structure as given by Definition 4.1. Sometimes, too much data is shown or taken into account. The user is therefore able to filter the data that is less relevant for the current analysis. The filtering of data is discussed in Section 5.6.2.

Visual Representation

The visual representation of data is generally determined, sometimes based on the data, by the application, sometimes based on the data, without direct influence of the user. An exception to this is the component that visualizes the tasks and events (Section 5.5).

Visualization Output

The actual views and the use of visualizations are presented and discussed in Sections 5.3-5.5. Some of the views do not use a specific visualization technique, but just present the data in textual form. Depending on the view, different view transformations can be performed. A view transformation can change the scope of the data or can change the actual data that is shown at any given moment.

As stated previously, similar to Van Venrooij [18], we have used Shneiderman's visual information-seeking mantra for the design and implementation of the prototype:

Overview first, zoom and filter, then details on demand

This mantra is applied by using several views and by offering interactive features. The requirements FW1-FW5 (Section 3.3) describe the data and the aspects of the data that needs to be shown. Due to the differences in the data and the scale of the data, we have created multiple views, where each view focuses on a certain aspect of the data instead of showing as much data as possible in a single view.

It is important that the user is enabled to easily retrieve the data from any system for any day. The first two views, the site and calendar views, therefore give an overview of the data. These views provide an index for the individual event logs of an imaging system. The site view shows for which systems and for how many days data is available (Requirement FW1). The calendar view shows the distribution of several event types of a single system (Requirement FW3). These views are discussed in Sections 5.3 and 5.4 respectively.

Based on the data in the calendar view or based on information retrieved from another application, the user can choose a single day of interest. The events of this day are then shown in the event timeline view (FW4). The timeline should by default not show all data, since, first, this would clutter the timeline with too many events even for an average day, and second, not all event types are initially important. The user can therefore change the event types that are shown on the timeline. Whenever the user requires additional information on some data point, it is possible to request more information of a single system or an event (FW2, FW5).

Figure 5.3 shows an overview of the prototype. The system behaved normally for the most part of the day, except for two periods. Figure 5.4 shows a part of the timeline in more detail. The scans are presented in the timeline by yellow bars that occur after the little blue bars (preparation events). Commonly, scan events have a duration of several minutes. The difference in the duration of the scan events, and the occurrence of errors, give an indication that something went wrong during some of the scan events.

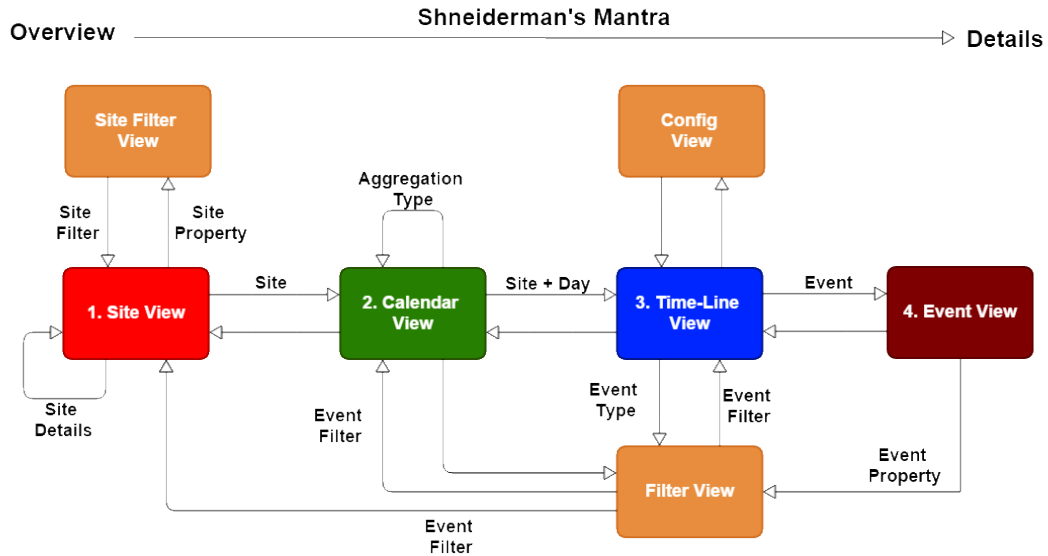


Figure 5.2: User Interface and Data Flow diagram. It shows the global flow between the views of the application and the data that is shared among these views. The numbered blocks refer to the data and visualization views. The orange blocks are related to views that support the interactive features of the system and are integrated into the data views. The labels on the transitions denote the data that is communicated between the views.

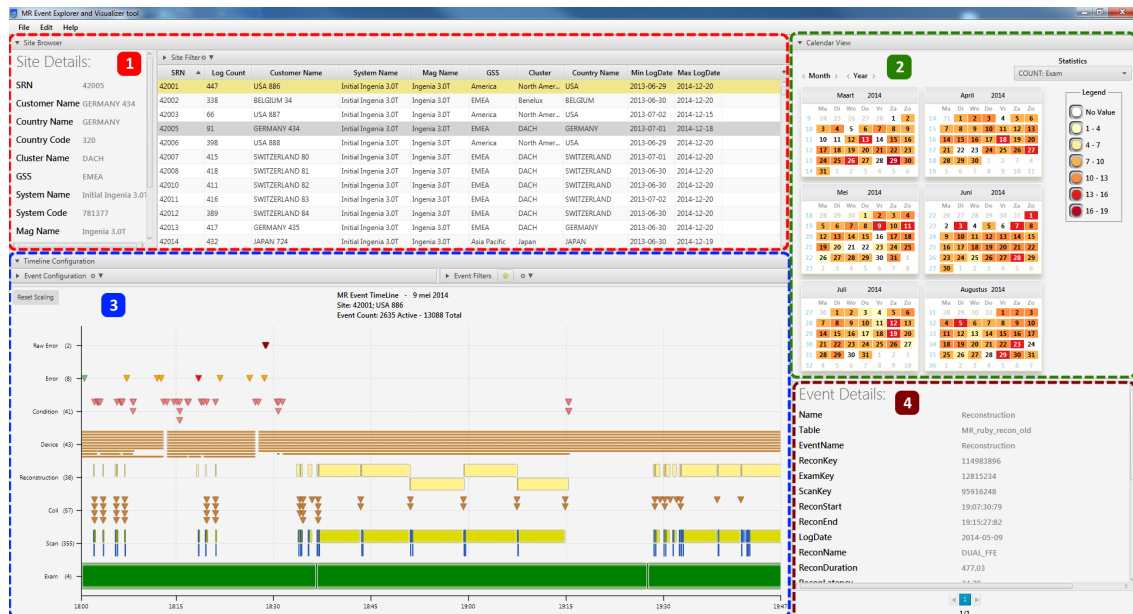


Figure 5.3: Global overview of the MR Event Explorer tool. **1.** List with all systems. For each system the amount of logs available and some properties are shown. **2.** Calendar View showing the daily statistics on the events data. **3.** Event timeline, visualizes a subset of all of the events that occurred on this day. **4.** Event Pane, shows the full details of the event.

5.2 User Interface Design

One of the most important aspects of the visual information-seeking mantra and visual data mining techniques is the user interaction. The user needs to be able to request additional information, filter the current data set or needs to be in control of the amount of detail that is shown. For designing the views and the composition of these views in the application, it is important to think about how the user would navigate between the views. In Figure 5.2, a diagram of the general flow through the several views of the application is shown.

As stated in the previous section, each view in the application is targeted towards a different aspect of the available data and the actual data that is shown can be determined through another view. Whenever the user selects some data in one view, this data is communicated to the next view such that this view can retrieve and show the requested data. It is expected that the focus of the user would also transfer to this view, since the data in this view also changes.

The data set for the application can effectively be split into two types. First, there is the set of MR systems, and second, each system has a set of event logs. The Site view mainly uses the administrative data set that describe the sites, while the other views show subsets of the log data. Due to the differences in the kind of data that is shown, it first seemed a good idea to design two separate screens where only one screen is visible at any moment. The first screen was related to the systems and contained a list of systems, the details of the highlighted system and the system filter. After choosing a system in the list, the second screen was loaded which showed an overview of the available event logs and the events of the selected day were also visualized in the same screen. Hence, enough space was available for each of the individual views and it was possible to easily switch between log days. It however felt like it was not as easy to switch to another system to make a comparison or analyze a similar case in another system. Furthermore, in some cases it felt like it would be useful to have the log calendar and the event timeline visible on the screen while considering selecting another system.

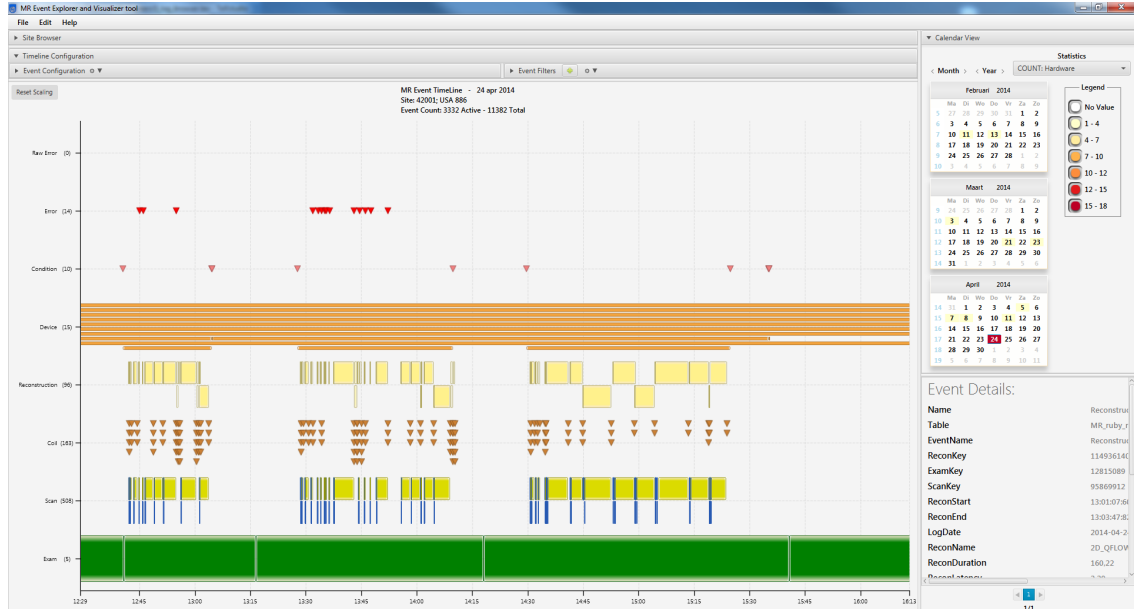


Figure 5.4: The application with the Site View hidden. The system shows the same data as shown in Figure 5.3. Comparing both screens, it can be seen that more space is available for the timeline. Smaller interval events can now be seen more easily while the same period is shown. The extra vertical space could also be used to add additional event types, which are currently disabled.

We therefore chose to compose all the views into a single screen instead. The size required for a view can depend on the situation, thus the views have been placed within a layout that allows the user to change the size of the views or hide them. In this way, the user can reduce the size of less important views and give

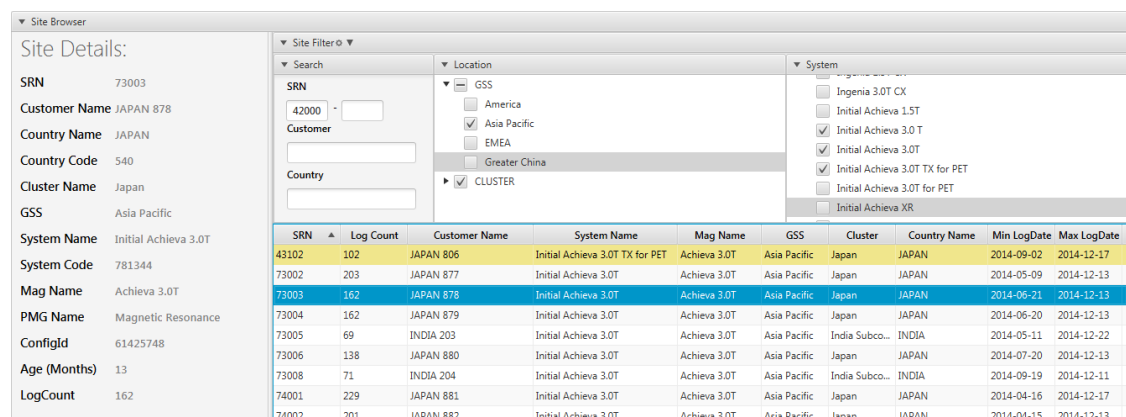
more space to views that are more important for the current task. For example, after selecting a site and an event log to inspect, the site view is not as important while analyzing the current log. But the calendar view might still be important, since the user might want to inspect several event logs for the current system. So the calendar view remains visible in a smaller size, while the site list and site details are hidden, making a large amount of space available for the event timeline. After analyzing one or several event logs of a single system, the site view can be restored easily, while the other views remain visible. An example of this composition of views is shown in Figure 5.4.

5.3 Site View

As presented in Section 2.4, a system that is located at some location for some customer is referred to as a *site*. The Site View is the top level view and provides an overview of all the available sites in the database. It functions as an index into the event logs of the individual systems (Requirement FW1). For each site, a set of properties and the amount of logs available for the system of the site is shown (Requirement FW2). Figure 5.5 shows an example of the site view.

The dataset of sites goes back longer than the period for which event logs have been processed. The set of sites therefore also contains many old sites for which no longer event logs are retrieved and hence also no MR log models are created for these sites. The set of sites is therefore filtered such that it only contains sites for which at least one MR log model is present.

After eliminating the non-relevant sites, still thousands of sites are available for analysis. To make the list more manageable, a simple manual filter is offered. By using this filter, the list of sites can be easily reduced to a smaller set. The filter offers enough options to easily create a subset of systems which could be used in the analysis. The user might want to compare the usage or the behavior of a set of systems of similar type of location. For some locations and system type properties, a list of all the distinct values is shown and each value can be either enabled or disabled. The site identifier can also give an indication on the type of the system, e.g., 2xxxx or 4xxxx series. Therefore, a range filter has been included to either make a subset of sites within a specific range or one particular site can be found easily by using its identifier.



The screenshot shows the 'Site Browser' interface. On the left, 'Site Details' for SRN 73003 are displayed, including Customer Name (JAPAN 878), Country Name (JAPAN), Country Code (540), Cluster Name (Japan), GSS (Asia Pacific), System Name (Initial Achieva 3.0T), System Code (781344), Mag Name (Achieva 3.0T), PMG Name (Magnetic Resonance), ConfigId (61425748), Age (Months) (13), and LogCount (162). The main area shows a 'Site Filter' with a search bar and a tree view of filters for Location (GSS, America, Asia Pacific, EMEA, Greater China, CLUSTER) and System (Ingenia 3.0T CX, Initial Achieva 1.5T, Initial Achieva 3.0T, Initial Achieva 3.0T T, Initial Achieva 3.0T TX for PET, Initial Achieva 3.0T for PET, Initial Achieva XR). Below the filter is a table of sites:

SRN	Log Count	Customer Name	System Name	Mag Name	GSS	Cluster	Country Name	Min LogDate	Max LogDate
43102	102	JAPAN 806	Initial Achieva 3.0T TX for PET	Achieva 3.0T	Asia Pacific	Japan	JAPAN	2014-09-02	2014-12-17
73002	203	JAPAN 877	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	Japan	JAPAN	2014-05-09	2014-12-13
73003	162	JAPAN 878	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	Japan	JAPAN	2014-06-21	2014-12-13
73004	162	JAPAN 879	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	Japan	JAPAN	2014-06-20	2014-12-13
73005	69	INDIA 203	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	India Subco...	INDIA	2014-05-11	2014-12-22
73006	138	JAPAN 880	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	Japan	JAPAN	2014-07-20	2014-12-13
73008	71	INDIA 204	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	India Subco...	INDIA	2014-09-19	2014-12-11
74001	229	JAPAN 881	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	Japan	JAPAN	2014-04-16	2014-12-17
74002	201	JAPAN 882	Initial Achieva 3.0T	Achieva 3.0T	Asia Pacific	Japan	JAPAN	2014-04-15	2014-12-13

Figure 5.5: Site View showing a subset of the sites. The filter has been used to limit the site list to a few systems of a certain type that are stationed in the Asia Pacific region.

We considered to offer a more sophisticated filter, where sites can be filtered based on sub-components or configuration settings of the system. By doing so, it would be possible to create a very specialized subset of systems that are very similar in terms of hardware and configuration. But we decided against it for two main reasons:

1. It is not clear how to actually retrieve the current set of components, or even how to model the hierarchy of components. MRI systems are complex and the time it would take to create a system that is able to retrieve and store this information did outweigh the actual benefit this would have for the prototype.

2. The main focus of the prototype is on the visualization of the events. An advanced filter feature for sites is not a priority. The prototype will likely not be used extensively in a manner where the event logs of a set of systems are compared based on the exact hardware configuration. Furthermore, most of the views in the prototype target only a single system and no extensive features are available for comparing the event logs of multiple systems.

5.4 Calendar View

The site view only shows how many event logs are available for each system and in which time span. It gives no indication on how often or how regularly a system has been used. To be able to analyze the global usage of a system over time and to find interesting log dates, it can be useful to show, for several event types, the distribution of the number of events over the entire history of the system (Requirement FW3). By showing the number of events that occurred for each day over a longer period of time, regular patterns and abnormal log dates can be discovered.

First, we need to know for which event types we want to show the distribution. The event types that are to be included in the overview need to be meaningful. It should give an indication on how often the system is used, show some global system behavior over time, or show that something was or went wrong with the system. Some event types are not important, give little insight in how often the machine is used, or give little extra insight in combination with other event types, e.g., user-interface messages, user-interactions and movement of the examination table. Therefore not all event types have to be included in the overview. The following event types were considered to be the most important to be shown:

- **Exams:** The number of exams can clearly show how often the machine has been used on a day. The most common usage is on patients, but MRI machines can also be used to scan other objects.
- **Scans:** While the number of scans performed is linear in the amount of performed exams, the scan event type contains some important properties that can be useful for analyzing the usage of the system. Showing the distribution without taking these properties into account gives little extra information about the exam count, but when filtering the distribution on certain scan types or scan properties (Section 5.6.2), the distribution of the scan events can become more useful.
- **Error and Warnings:** A large number of errors or warnings on a given day does not ensure that a system failure occurred. But it can give a good indication of which event logs might be interesting to analyze, since these days have some anomaly. Just like the scan events, filtering the errors on a particular error type can direct the user to event logs in which something went wrong that is related to that specific error.
- **Calls and Jobs:** The administrative maintenance data gives an indication when something went wrong in the system or when a service engineer has been performing maintenance on the system.

Next, we need to determine how large the period that is showed, needs to be. Optimally, the entire history of the system is shown in a clear manner. By showing the data over a longer period, it can be easier to discover patterns over a certain period, or some differences might be discovered between several smaller periods.

For some event types, e.g., some error types, calls and jobs, the number of occurrences is not as large. Some of these event types may not occur at all for most days. We therefore have to make sure that the individual value for each day is clearly visible, since we want to be able to see anomalies in the data.

Finally, whenever an interesting data point is found, it is important that we are able to quickly tell which date corresponds to that point, such that this day can be selected for inspection. It can also happen that the user already has a date in mind and is currently not interested in the values of other dates. So it has to be possible to quickly select a particular date.

Two options come to mind for determining the range of the period that is shown:

1. Use Shneiderman's mantra. First provide an overview that shows the data over the entire history and enable the user to interactively zoom into and filter the data.

- Use one level of detail where the range of the data can be adjusted interactively and the amount of data that is shown is based on the available space.

Both of these options can fulfill the requirements, but one of both might not be suitable for certain visualization techniques. So, before we decide which approach to take, let us first look at the visualization methods that we could use to visualize the distribution of the number of events for several event types.

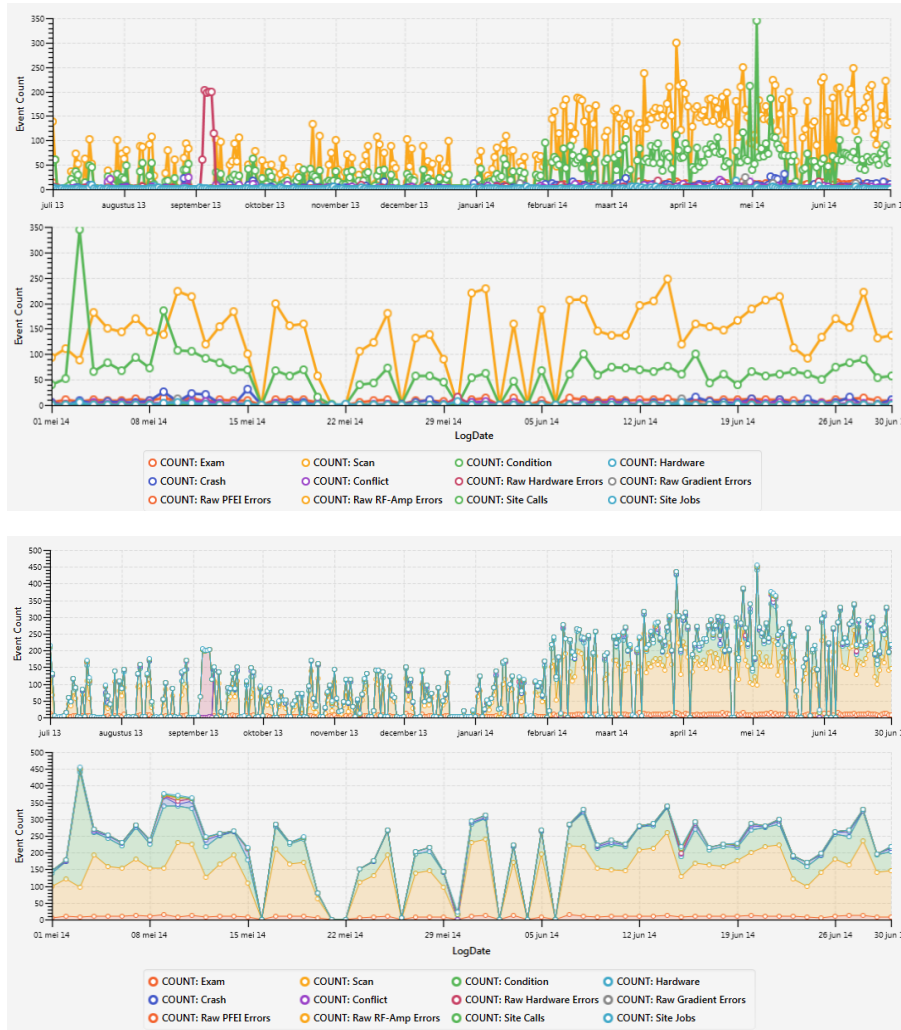


Figure 5.6: Two line-charts and two area-charts showing the distribution of several event types. The first chart spans a period of a year, while the second chart spans a period of two months. The trend of the occurring events that occur most often, can be seen, but the more uncommon event types are cluttered together.

The first technique that comes to mind is to use a standard kind of chart, e.g., a line-chart, bar-chart or area-chart. Both of the options for determining the period shown can be applied, e.g. by allowing zooming or panning of the chart. A different color is used for each event type to make them distinct. Figure 5.6 shows line-charts and area-charts of an average machine for a period of a year and for a period of two months. It shows that using this kind of charts does not work well when showing the daily number of events for each event type. First, some event types have a similar number of events and some events have a large deviation. Therefore lines in the line-chart are overlapping and lines are crossing each other. Second, the longer the shown period, the more cluttered the overview becomes, if the bins size for each displayed value is not adjusted. But by adjusting bin size, and for example showing the average value in each bin, might give a distorted view on the data, since data is regularly missing for most systems. Hence, showing

a long period seems to be not so useful. The user would always have to zoom into the data to get a better impression of the trend in the number of events.

While the area-chart improves on the overlapping of lines by stacking the values, it is not able to solve a second problem of using charts. Some event types have a few hundred instances a day, while most of the event types only have a few to tens of event instances each day. Due to the large difference in the number of events, the height of the area of less common events is too small for reading the height. For some event types, the area is event too small to be seen.

Instead of showing all event types at the same time, we could also show only a single event type and allow the user to change between the event types, as seen in Figure 5.7. But for a long period, the data points are still too clustered, and a relative large effort has to be made to get an impression on the number of events for individual days.

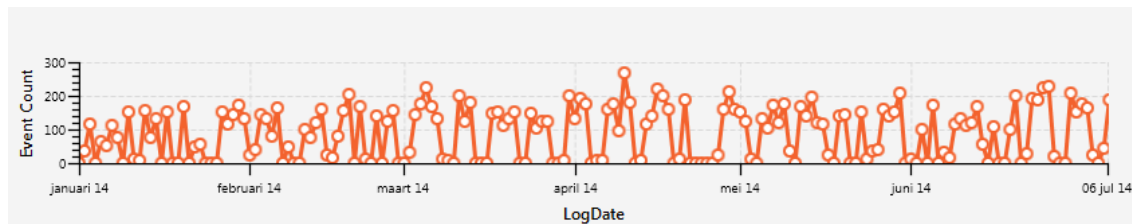


Figure 5.7: Line-chart showing the trend in the number of scans performed over half a year.

So instead of using spacing to show the number of events, another technique has to be applied. A solution can be to use colors to denote the number of events. Examples of approaches that use colors for visualizing time-series are Dense-Pixel Displays or Heat-maps. In these techniques, several time series are laid out in a specific pattern and each data point is assigned a color depending on its value. Figure 5.8 shows two different layouts for dense pixel displays. The colored displays are able to clearly visualize the trend of values over time. A downside however is that only a small space is available for each data-point when the size of the time series grows, making it difficult to determine the value for a particular day, and also the limited number of colors that can be perceived reduces accuracy.

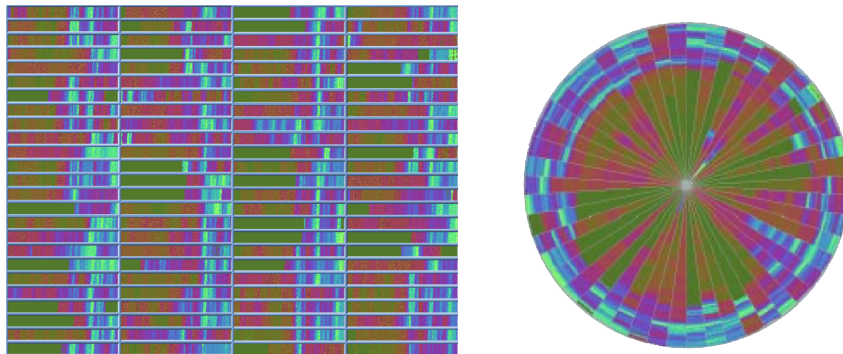


Figure 5.8: Two types of Dense-Pixel Displays. The first display uses a simple rectangular shaped sub-windows to layout 100 time series. The second display uses a circular layout to show 50 time series. Both displays visualize twenty years (January 1974 - April 1995) of daily prices of stocks contained in the Frankfurt Stock Index (FAZ). High values correspond to bright colors [7].

From both techniques we can conclude that using colors for visualizing time series, using a smaller period, and only showing a single event at one time can be a good approach. It enables both the trend and the individual values to be visualized clearly. A simple, well known, and powerful method for showing a sequence of consecutive dates is a calendar. We therefore have chosen to combine calendars with color displays to create a calendar heat-map. The calendar clearly shows individual dates and a color is assigned to each date to visualize the number of events for a single event type. The user is able to manually switch between event types to explore the trend and anomalies of different event types. The current layout allows

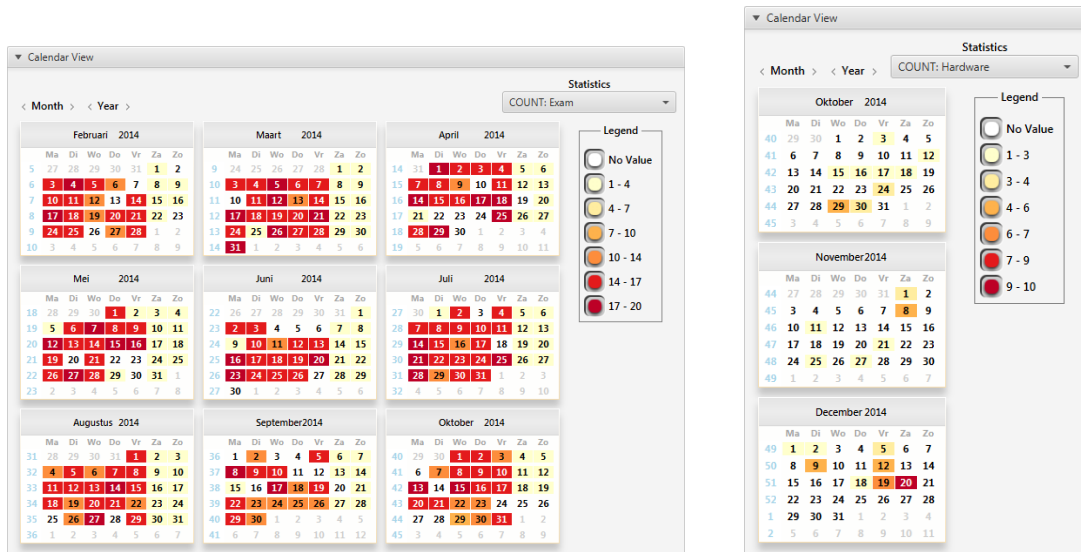


Figure 5.9: Event calendar of two different systems and two different event types. For the exam calendar, it can be clearly seen that the system is used very regular except for the weekends. For the hardware (error) event type, days for which problems might have occurred can be clearly seen, since hardware events do not occur that often. But for some days the number of hardware errors has spiked.

the user to view the distribution of events up to a span of one year. Now, we only need to choose a suitable coloring for showing the values. Instead of using a continuous color scale, a decision was made to divide the numbers from zero to the maximum number into evenly spaced bins. Otherwise the user still has to guess the number that is corresponding to color. Each bin is assigned a color from a continuous color spectrum, creating enough distinction in the different colors. Figure 5.9 shows the event calendar for two different event types for different sized periods.

A downside of using equally sized bins is that when strong outliers occur, then the lower numbered values will all get assigned the same color and only the few dates with the outliers will stand out. This happens occasionally for the hardware error type. In these cases, a few particular errors are reported repeatedly over a few hours, resulting in hundreds of error events. Currently this does not occur for many machines and is limited to two event types. Therefore there was no immediate need to change the determination of the bins. A relative simple solution can be to determine the size of each bin dependent on the amount of values in each bin. Quartile binning can for example be used to better distribute the values over several bins. An example of outliers is shown in Figure 5.10.

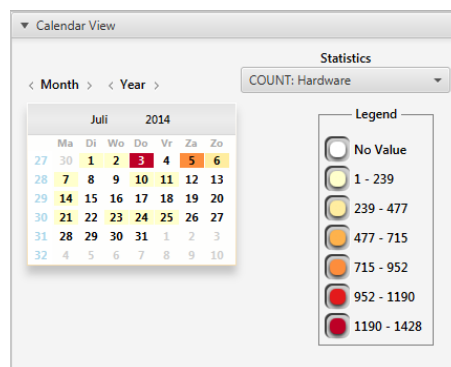


Figure 5.10: An abnormal event distribution in the calendar view for the hardware error types. On July 3rd, something was wrong with the patient support, causing errors to be produced constantly. The large amount of errors for two days causes the value for other dates to be difficult to determine.

5.5 Event Timeline

The usage, behavior and state of a system is described in detail by the actual tasks and events that have occurred in the system. To enable the analysis of the daily usage and behavior of the system, these tasks and events need to be visualized in some manner (Requirement FW4). Most of the related work discussed in Section 3.1 use glyph-based visualization, where the glyphs are presented on a timeline to denote when the event has occurred. Therefore it makes sense to use a similar approach. We first determine how to group the events. Second, the options for visualizing the events and their properties are discussed. Then, we present a problem that occurred for the initial approach and how it was solved. Finally, we discuss how we use Shneiderman's mantra for zooming, filtering and showing details on demand.

5.5.1 Event Grouping

The timeline approaches used by Van Venrooij [18], Röder et al. [13], and Aharon et al. [1] all group the events in their system by either the component that produced them or by the type of the event. If we were to group the events vertically by event type, a clear separation of event types can be made. However, due to a relative large amount of event types, a large vertical space is required, or the glyphs need to be visualized very small. The events contained in the MR model are not linked to a dedicated component, so we cannot group the events in that way. In order to have enough vertical space available for each line in the timeline, we can specify a number of categories and partition the event types into these groups. Event types that are closely related or similar can be put into a single category. On the other hand, event types are not necessarily equally important. So some event types do not have to be shown by default. Hence, we can save space by limiting the amount of event types shown at any given moment.

We have chosen to use a combination of these two options. We predefine the categories, the partitioning of event types into these categories, and which events to show. The predefined settings might not be suitable for each user or certain analysis approaches could do with a different configuration. It is therefore possible for the user to modify the configuration. This system is described in Section 5.6.1.

5.5.2 Visualization

Now that it has been decided that glyphs are used to present the events, and that events are grouped by categories in the timeline, we need to determine what glyphs to use. Each event has many properties, but there are not many visual options to show all of them. The most important distinction that needs to be made, is the kind of event (i.e., instant or interval) and the event type. Common techniques to show attributes are to use the following graphical encodings:

- **Size:** The size of the glyph could vary in vertical and in horizontal size. To keep the timeline nicely ordered and keep a consistent size for each group, we cannot easily use the vertical size to show an attribute. Moreover, it can be difficult to determine the actual property value if the size is not distinct enough.
- **Shape:** Different shapes can be used to denote the value of a certain attribute. To avoid confusion, only a few shapes can be used. There are however no common attributes for events aside from being an instant or interval event, and the event type. Since there are many event types, it seems logical to use the shape to denote whether the event is an instant event or an interval event.
- **Color:** Different colors can be used, but only a limited number of colors (10-12) can be perceived easily when used for small glyphs. Currently, 21 event types are used from the MR model. This would be too many if we were to show all event types at the same time and solely visualize the event type with color. But in most cases, it is not interesting to show all the event types and also event types are divided into categories. Hence, some event types can have a similar color when they are contained in different categories or not shown at the same time.

For interval events, we want to be able to show the duration of the event. This is often displayed by the width of the glyph, where the glyph is aligned at the time that the interval started and ended. A shape that



Figure 5.11: Rendering of the events using a timeline by trivially aligning the events at the time that they are produced. By looking at the events and the number of events that are supposed to be shown, it is clear that the glyphs are overlapping. The coil category should show 50 glyphs, but only 6 are shown. The events in the device and the coil category seem to suffer the most from this.

is commonly used for intervals, which also scales well in the width, are bars. The shape for instant events need to be different from the intervals. Any simple shape like a circle, triangle or diamond could work. We decided to use downward pointing triangles. This way the bottom point of the triangle can be exactly aligned with the time at which the event occurred.

Regularly, sequences of events are recorded within a short period of the event log, which can be of a single event type or of multiple event types. This implies that two events follow each other very closely, or that the start time for one event is the end time of the previous event. To prevent the glyphs of several events to blend with one another, the glyphs are visualized with a thin border.

We have chosen to group events into custom categories. Each category gets its own vertical space on the timeline. To make clear which line belongs to which category, we draw labels next to the vertical axis. Since events are ordered horizontally in time, it is obvious to also show labels on the horizontal axis to denote the time at any point in the timeline. Each category contains several event types and each event type can have a different number of event instances. A category can contain hundreds of events on a single day. To give a better indication of how many events are present in a single category, we also draw this number next to the category. Note that we only show the total number of events in the category and not the number of events for each event type. Due to the different number of event types in each group, it would create an inconsistent labeling.

5.5.3 Overlapping Events

Now that we have determined how we visualize the timeline and the events in the timeline, we could expect that we are now able to easily show the events that occurred in the system. However, when looking closer at the instances of events, it can be seen that many events of the same event type are produced at the same time and that interval events overlap with each other. Figure 5.11 shows an example of the rendering of a timeline with overlapping events.

We need a technique to remedy the overlapping of events that allows us to see the individual events. Van Venrooij [18] had similar problems. He showed the overlapping of events by drawing all glyphs in a transparent manner. However, this only partly solves the problem. When more than two interval events are overlapping, it is difficult to see where each interval starts and ends, and it makes it difficult to select one particular event.

In our case, some instant events occur exactly at the same time and for some event types, like the device event and coil event, it is almost always the case that multiple events overlap with each other. Using transparency only shows that there are multiple events, but it does not allow us to easily select a particular event. Hence, another method is required that physically separates the events from each other.

Instead of trying to visualize overlapping events at the same position, the available vertical space within a line can be used to create multiple event lines for a single category. While this means that the event types might have to be displayed smaller, it should display the individual events more clearly.

We therefore require an algorithm that detects the overlapping of events and is able to optimally lay out the events, such that the minimum number of lines is used. This problem is similar to the channel routing problem for integrated circuits that have no vertical constraints. In their case, the circuits have to be laid out on a chip, but the circuits may not overlap horizontally on an individual track. To solve this problem, Hashimoto and Stevens [5] proposed the left edge algorithm. The algorithm can be used to optimally lay out the circuits using a minimum number of tracks. The algorithm, which was adapted to our problem, is displayed in Algorithm 1.

Algorithm 1 Left Edge Algorithm

```

1: procedure LEFTEDGE(I)                                ▷ I set of events  $E_1 \dots E_n$  ordered by increasing  $s_j$ 
2:    $i \leftarrow 0$                                        ▷ Event  $E_j = [s_j, e_j]$  with start time  $s_j$  and end time  $e_j$ 
3:   while  $I \neq \emptyset$  do
4:      $i \leftarrow i + 1$ 
5:      $x \leftarrow 0$ 
6:     while There exists  $E_j \in I$  s.t.  $s_j \geq x$  do
7:       Pick the first event  $E_j \in I$  with  $s_j \geq x$ 
8:        $track[j] \leftarrow i$                                ▷ Assign  $E_j$  to track  $i$ 
9:        $x \leftarrow e_j$ 
10:       $I \leftarrow I \setminus \{E_j\}$ 
11:    end while
12:  end while
13:  return  $track$ 
14: end procedure

```

The algorithm is applied individually for the events in each category. Computing the tracks for the events takes $O(n^2)$ time, where n is the number of events that should be displayed in the category. The maximum running time however only occurs when all events in the set overlap. In practice the algorithm runs in linear time, since it only has to go over the set of events as often as the maximum number of overlapping events. The result of the algorithm can be seen in Figure 5.12.

For consistency, each category in the timeline has equal height. To be able to fit multiple tracks into a single category, the height of tracks in which the glyphs are placed is based on the number of tracks and based on the total available vertical space for the category. The height of a track Δy can be computed as $\Delta y = \frac{h_{cat}}{n_{tracks}}$, where h_{cat} is the height of the category and n_{tracks} is the number of tracks required for the corresponding category.

The height of an interval bar is therefore adjusted based on the track it is placed on. It is often also the case that multiple instant events occur within a very small timespan, i.e., less than a second. We therefore need to prevent the triangles to overlap horizontally as little as possible. First, in the algorithm, we assume that instant events have a very small timespan of 500 milliseconds. Second, the triangles for the instant events have an acceptable fixed size. This way, the instant events have minimal overlap horizontally when visualizing a larger span of events and the chosen size allows multiple triangles to be placed on top of each other.

As stated before, interval events are aligned on the timeline according to the start and end time of the interval. The interval events therefore scale horizontally based on the period shown. The width of intervals however is forced to be at least two pixels. This is to prevent intervals, that have a very small period, to disappear due to being too small.

As seen in Figure 5.12, the layout determined by the left-edge algorithm prevents the event to overlap, while the events remain at an acceptable size. However, by looking closely at the events in the coil category, it can be seen that, in some cases, there are too many overlapping events. The rest of the events cannot be seen, since the region of the category is clipped vertically. Therefore, even more events are present than actually shown. Other events types have this problem only in very rare occasions.

In Section 4.1, it is explained that an MRI system uses several coils for distorting the magnetic field and sending RF pulses, but the real situation is more complex. Coils can contain multiple elements and each element can be used or not, depending on the region of the body that is being scanned. The RF body coil

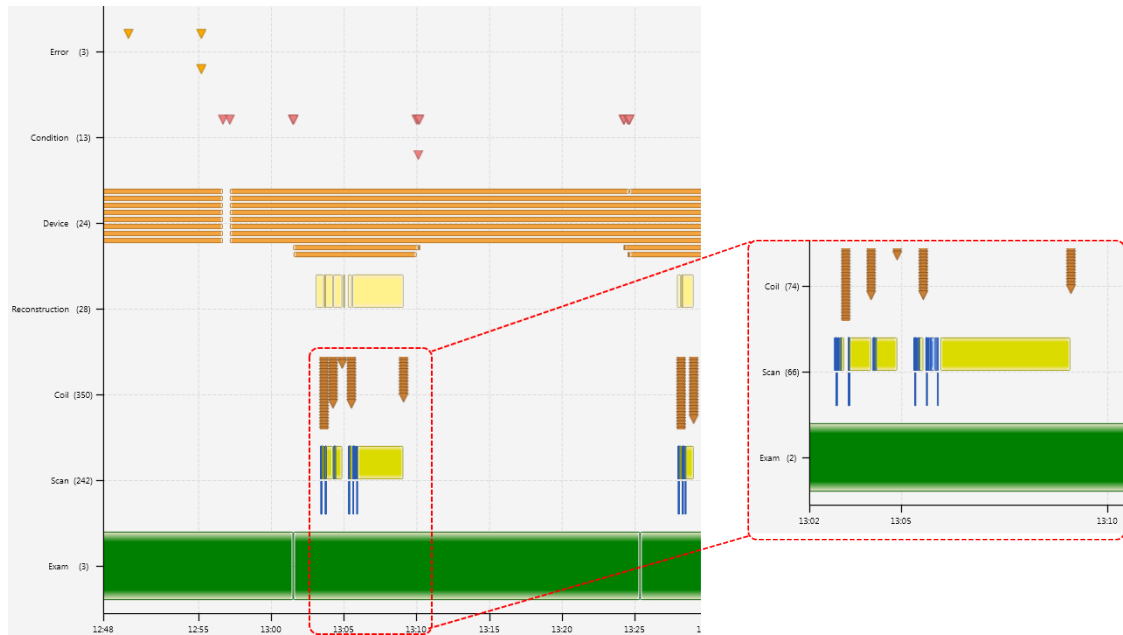


Figure 5.12: The overlapping of events has been resolved by placing the overlapping events below each other. There is plenty of space available for the event types with multiple overlapping events, e.g., the device events. In this screenshot, the Timeline View was assigned the maximum amount of available vertical space on a high resolution screen. In many cases however, the coil event type has too many overlapping events and not enough vertical space is available to show all the events.

for example, can be used to scan the whole body. But in other cases, only the lungs or the hart of the patient needs to be examined. In the latter cases, only a subset of the coil elements is used. For each element used during a scan, a single event is created in the MR model. So when multiple coils and elements are used, many events are produced at approximately the same moment. It was therefore decided to combine all the related coil events into a single event, which highly reduces the amount of coil events. Figure 5.13 shows the same timeline as shown in Figure 5.12, but with merged coil events.

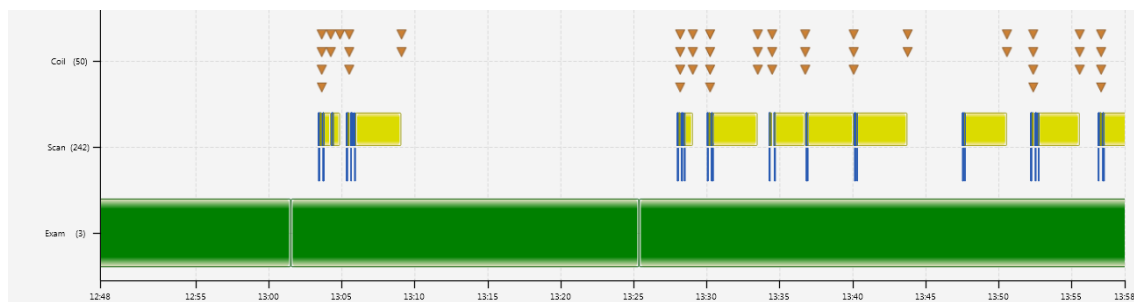


Figure 5.13: Visualization of the timeline after the related coil events are merged into a single event. The log and period shown is the same as shown in Figure 5.12.

5.5.4 Event properties

The events are now nicely visualized onto a timeline with minimal overlap of glyphs, and shows which events occurred when. However, as stated in Section 4.2, events have two more attributes, namely properties and a set of linked events. Like described by Requirement FW5, event properties need to be shown for analyzing what the system has actually been doing during a single day. What is important to note, is that the properties of an event are not all equally important. Most properties are only required for the further

analysis of a particular case. Again in the spirit of Shneiderman’s mantra, we initially only need to show the properties that are useful for discovering the global usage and behavior of a system. Only when further information is required, it should be possible to retrieve the rest of the details on demand.

Based on the values of the properties and in discussion with some product experts, properties that seemed most useful for the global analysis of events have been selected. For each event type several properties have been chosen to be displayed together with the event glyphs. There is no simple visual method for clearly visualizing multiple values in addition to the current glyphs using one or more graphical encodings. We therefore decided to display the values of the properties for an event in a pop-up window whenever the mouse cursor hovers over the glyph of the event. For this pop-up, a semi-transparent dark color is used for the background and a light color is used for the text, such that the pop-up window can be clearly seen on top of the timeline while the events behind the pop-up can still be seen. The pop-up is automatically shown when hovering the mouse cursor over the glyph of an event and is hidden again when the cursor is moved away from the glyph. A few examples of the pop-up are displayed in Figure 5.14.

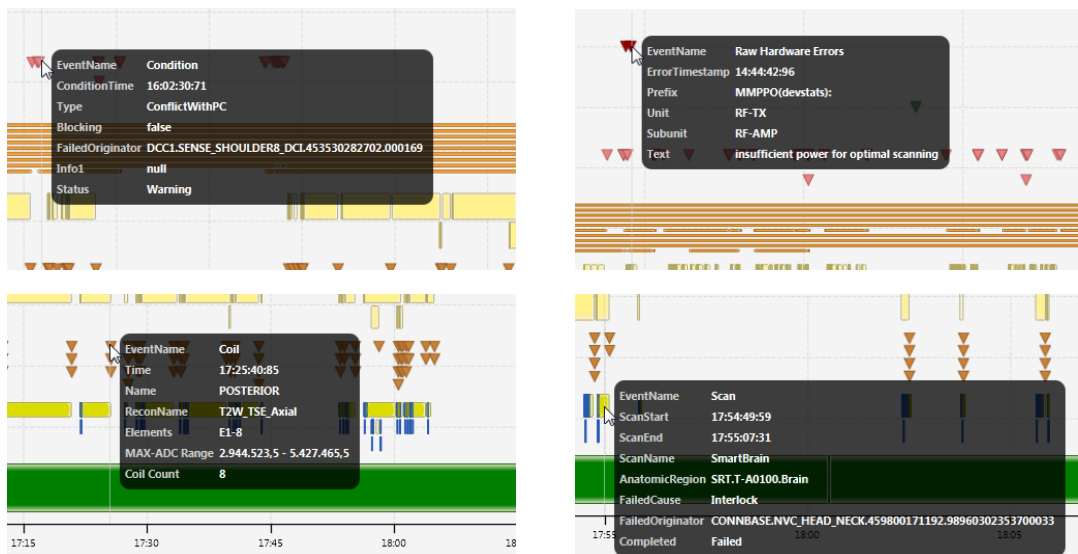


Figure 5.14: Four examples of the property pop-up for the Condition (top-left), Raw Hardware Error (top-right), Coil (bottom-left) and Scan (bottom-right) event types. These have been captured semi-randomly from different event logs, so there is no relation between the examples.

As described above, the pop-up for events only displays the values for a subset of properties, but it should be possible to retrieve the values for all of the properties for an event such that they can be analyzed in more detail. We enable this by selecting the event and displaying all available properties in the Event View.

To optimize performance, the information required for the pop-ups is retrieved from the database when loading the events for the timeline, such that the information for the pop-up is available as soon as it is required. Whenever an event in the timeline is selected for the Event View, detailed information is loaded from the database. By minimizing the amount of information that is retrieved, we are able to minimize the required memory as well as retrieve events in less time.

5.5.5 Linked events

As described in Section 4.2, the events in the MR model are linked together by a hierarchical structure. This structure is a directed graph, where the events are nodes and the edges are the hierarchical links between events. A single event type can be linked to zero or more event types. For example, the exam event type has several directly related event types, including preparation, scan and reconstruction. Currently, the relation of events is defined directly by the relational model of the MR model in the database. In practice, more relations can be defined. The coil events could for example be related to the scan that was performed, since

Event Details:	
Name	Raw Hardware Errors
Table	MR_raw_logstat_hw_errors
EventName	Raw Hardware Errors
Key	625554909
ErrorTimestamp	11:29:32:98
Prefix	H_PASS_SVR_IMT(devstats):
Unit	PATSUP
Subunit	PLC
Text	Error 1.12 out-end switch becomes active
SRN	42001
Logfile	/home/2012-0151_hms/MR/42001/logsta
LogDateTime	16:34:24:00
LineNumber	6.728
1/1	

Event Details:	
Name	Coil
Table	MR_ruby_coil_old
EventName	Coil
CoilKey	372689057
CoilTime	21:21:51:09
LogDate	2014-03-06
Name	POSTERIOR
ReconName	CoilSurveyScan
Element	E07
MaxADC	12.677.301
SRN	42001
DailyLogKey	3023185
3/12	

Figure 5.15: Property values for the raw-hardware and the coil event. Note how for the coil event type multiple events are present since the original events were composed into a single event in the timeline.

a set of coil events is always preceded by a single scan event. We decided not to implement extra relations, since it was not clear how we could define these relations in a structured manner.

There are several options for visualizing the links between events. The first option that comes to mind is to visually draw the edges between events like a graph. This is however not that simple to integrate into the current timeline. First, the event nodes do not have a fixed location and size, neither vertically nor horizontally. Second, we have visualized overlapping events on top of each other. Finally, related events can follow each other immediately on the same line of the Timeline View. Therefore, edges that are drawn between glyphs would cross each other and other glyphs. This would likely lead to a cluttered visualization which would only confuse the user.

Looking more closely at how the relations are defined, it might not be important to visualize the actual links between individual glyphs. The most important aspect is to be able to show which events are directly or indirectly linked in the hierarchy, since we are interested in analyzing the context in which events occur and how long the period of related events is. Therefore, instead of drawing the links, we only need to highlight the related events. Whenever an event is selected, we first find all direct and indirect related events. In other words, we find all 'ancestors' and 'descendants'. In a directed graph, we do this by performing a depth first search in both directions. After finding all related events, we can visualize the relations. We do this by applying a blue color to the border of the event that has been selected, applying a red color to the border of all related events and by making unrelated events semi-transparent. As a result, all the related events become clearly visible. Figure 5.16 shows the set of related events for an exam and a scan event.

5.5.6 Zooming and panning

When the data for an event log is first loaded into the system, the timeline shows all events for the enabled event types during the corresponding day. The overview of events for a whole day should give a good indication of how much the system has been used or if some problems have occurred during the day, and there might be some interesting periods that can be seen by the visual pattern of events. When more detail is required, the user can zoom into the data by using one of the following actions:

1. Scrolling with the mouse wheel. By scrolling forward, the timeline will zoom into the data, and by scrolling backwards, the timeline will zoom out again.

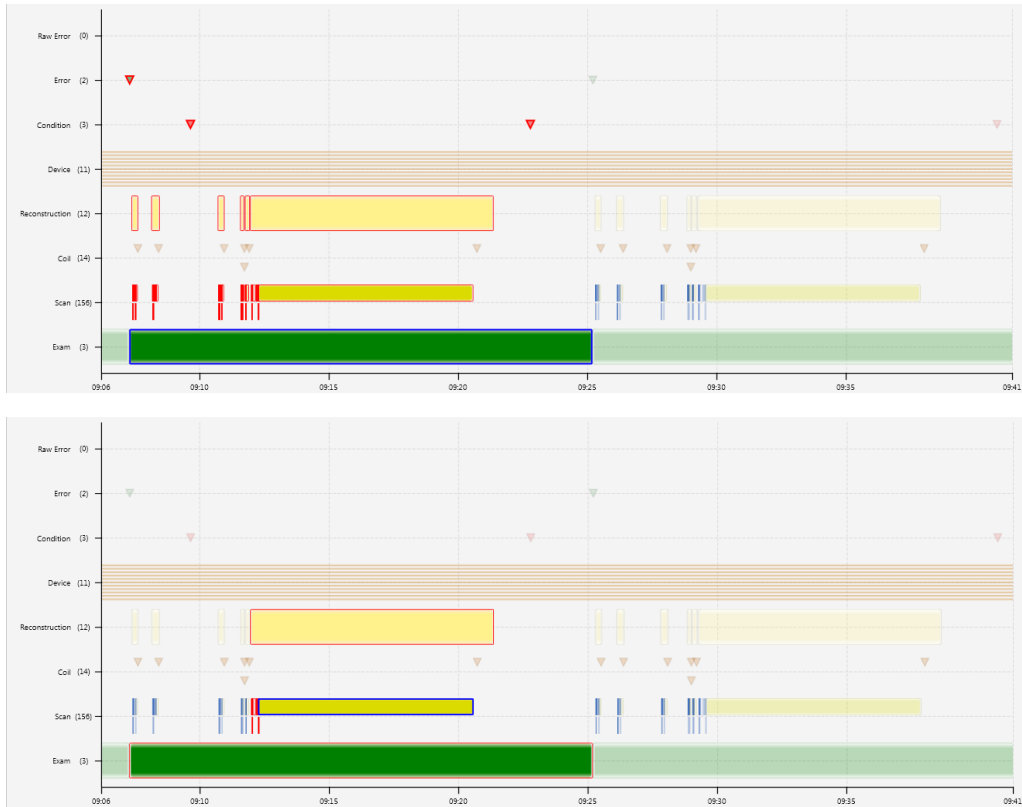


Figure 5.16: Examples of linked events. On the top, all related events to the selected exam are shown. After selecting one of the scans that is related to this exam, the number of related events is reduced. This is due to the fact that this scan event is inside a branch of the hierarchy that starts with the exam event.

2. Selecting a subregion by dragging the mouse while holding the right mouse button. The timeline will zoom into the time of the selected region.
3. Double clicking an interval event. The timeline will zoom into the time period of the selected interval.

After zooming in, the timeline can be panned in time by dragging the mouse while holding the left mouse button.

5.6 Additional Interactive Features

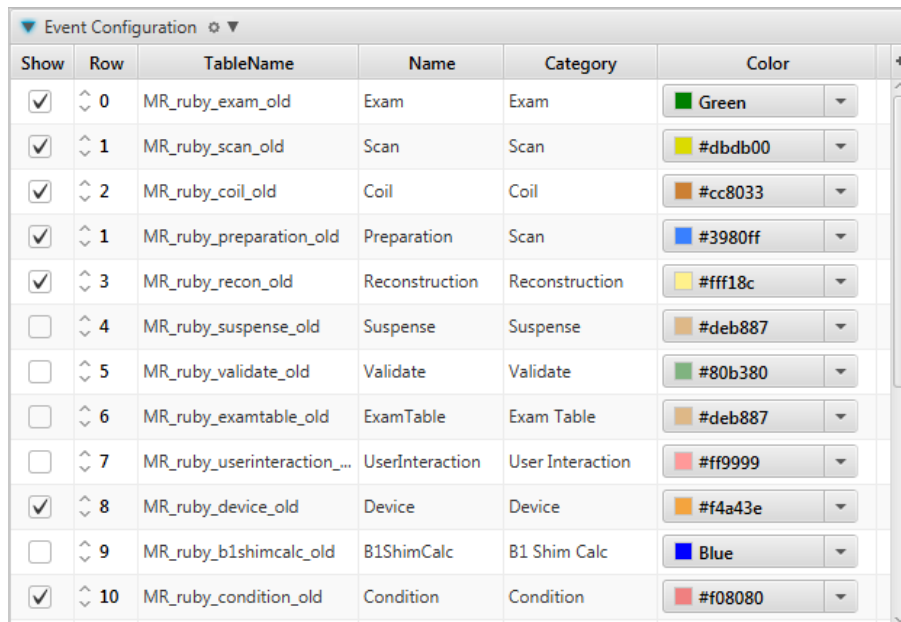
In the previous sections, we discussed how we designed and implemented the features regarding showing or visualizing the data of the prototype (Requirements FW1-5) and how data is communicated between views for data selection (Requirement FI1). In this section, we discuss the design and implementation of the remainder of the interactive features. First, the configuration options for the timeline are presented. Second, we discuss the design of the filter feature for the events data (Requirement FI2). Finally, we present how the user is enabled to store and load timeline configuration, event filters, and the system state.

5.6.1 Timeline Configuration

The timeline configuration determines the look and the content of the Timeline View. As discussed in Section 5.5, the timeline comes with a preconfigured visual representation for the events and by default, it does not show the events for all event types. In the Configuration View, the following attributes can be configured for each event type:

1. **Show:** Should the event-type be shown in the timeline or not.
2. **Category:** On which category (line) of the timeline should the event type be aligned. Multiple event types can use the same category.
3. **Row:** This property determines the order of the categories in the timeline. Changing the row for one event type changes the row for other event types as well, such that all event types in the same category are updated to the same row and that no two categories use the same row.
4. **Color:** The color that the event type should use for the visual representation.

Figure 5.17 below shows a part of the default configuration for the prototype. As discussed before, by default, we only show the event types that were determined to be initially important in a way that enables the user to get a good impression on the usage and state of the system during the day. An example of an event type that can contribute towards the analysis of a problem, but does add little information to the overview of events, is the *user-interface message* event type. As is shown during the evaluation of the prototype in Chapter 6, the UI messages can give additional information on errors that have occurred.



Show	Row	TableName	Name	Category	Color
<input checked="" type="checkbox"/>	0	MR_ruby_exam_old	Exam	Exam	Green
<input checked="" type="checkbox"/>	1	MR_ruby_scan_old	Scan	Scan	#dbdb00
<input checked="" type="checkbox"/>	2	MR_ruby_coil_old	Coil	Coil	#cc8033
<input checked="" type="checkbox"/>	1	MR_ruby_preparation_old	Preparation	Scan	#3980ff
<input checked="" type="checkbox"/>	3	MR_ruby_recon_old	Reconstruction	Reconstruction	#fff18c
<input type="checkbox"/>	4	MR_ruby_suspense_old	Suspense	Suspense	#deb887
<input type="checkbox"/>	5	MR_ruby_validate_old	Validate	Validate	#80b380
<input type="checkbox"/>	6	MR_ruby_examtable_old	ExamTable	Exam Table	#deb887
<input type="checkbox"/>	7	MR_ruby_userinteraction_...	UserInteraction	User Interaction	#ff9999
<input checked="" type="checkbox"/>	8	MR_ruby_device_old	Device	Device	#f4a43e
<input type="checkbox"/>	9	MR_ruby_b1shimcalc_old	B1ShimCalc	B1 Shim Calc	Blue
<input checked="" type="checkbox"/>	10	MR_ruby_condition_old	Condition	Condition	#f08080

Figure 5.17: Timeline configuration. Several properties can be set for each event type to suit the users needs.

5.6.2 Event Filtering

The calendar view gives a good overview of the number of events for several event types. Event types, like exams and scans, give an impression of how much the system has been used, while other event types, like errors and warnings, can give an indication of the days in which the system showed abnormal behavior. But we like to take this a step further. We want to be able to find event logs in which certain events occurred. The aggregated data therefore needs to be able to take the properties of events into account such that only events that fit the supplied description are counted. The same idea should be applied to the event timeline such that events that fit the description can be found easily among all other events. Figure 5.18 shows an example of the filter being applied to the Calendar View.

For this, we introduce a filter feature into the system that is simultaneously applied to the Calendar View as well as the Timeline View. It should be possible to describe the desirable events by using any of the properties of the events. Optimally, the filter feature should be easy to use as well as give a wide variety of options for describing the required event properties. In the first design, we considered to build a kind of interactive dialog or wizard in which the user is able to create an event specification using a guided system.

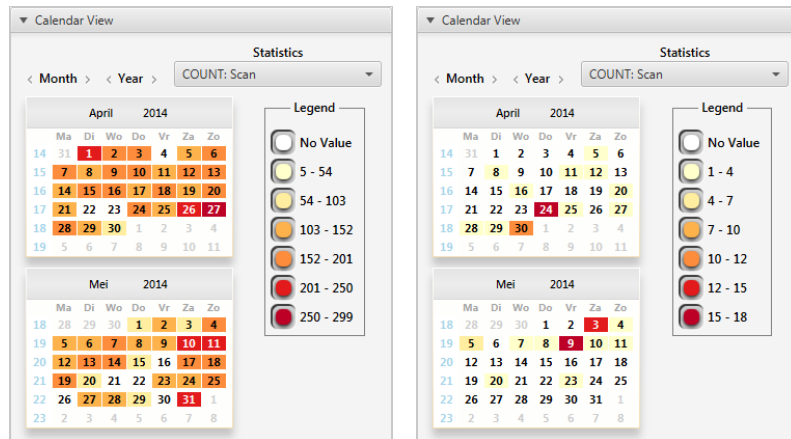


Figure 5.18: The Calendar View before and after applying a scan filter that only retrieves scans that have failed. Days that contained less data before have become more prominent, while other log dates have become less interesting.

Due to the generalized event data structure and a variety in data types for property values, designing and implementing a simple and complete system would take a lot of time.

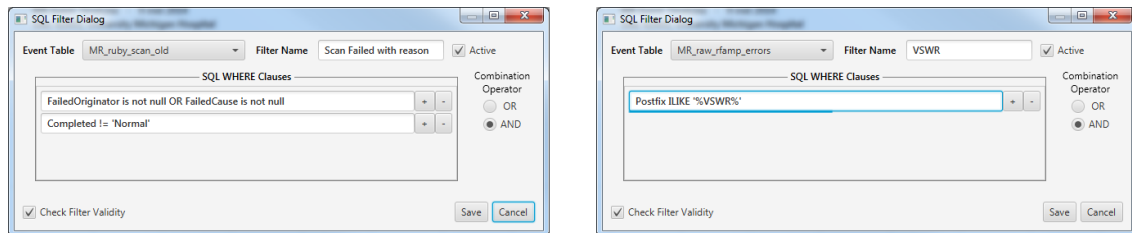


Figure 5.19: The dialog for specifying a data filter. The first filter is targeted to retrieve all scans that are not completed as normal and as well have a reason for it. The second filter retrieves all RF-amplifier errors where the postfix property contains the text *VSWR*

We therefore made a compromise and designed a filter system that is more difficult to use, but enables the user to describe events in full detail. The data is stored in a database and the prototype uses SQL to request data from the database. We therefore decided to exploit this for the filter feature. For each event type, the user is enabled to specify part of the *WHERE Clause* for the SQL-statements that are used to aggregate and retrieve events. A downside of this system is that the user must know some basics of SQL to be able to specify the filters. The prototype is however targeted towards engineers working in Diagnostic Design or R&D. We therefore assume that most of them are familiar with SQL or are able to easily learn the basic syntax that is required.

Two examples of filters are shown in Figure 5.19. We added two features to give some support to the user. First, a kind of auto-completion feature is implemented. When the user starts typing, a list of the available property names based on the current text is shown. The user can then easily select the property he wants to use and the word is completed with the selected property. Second, the entered event specification is checked for syntax by performing a test request. The SQL engine is exploited to create an SQL statement with a condition that is always false, returning an empty result immediately, unless there is a syntax error in the specification or if one of the specified columns does not exist. As shown in Figure 5.20, if there is an error in the specification, a error message is shown with the error given by the SQL engine, which can be used to find and correct the mistake that was made.

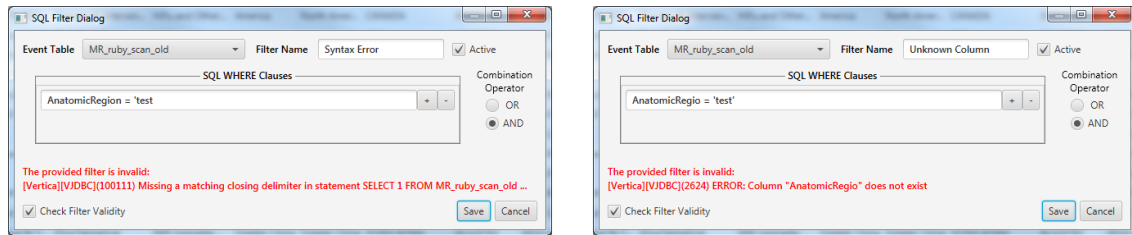


Figure 5.20: Two examples that support the user in specifying a SQL filter that has the correct syntax and to make sure that the specified columns exist.

5.6.3 Configuration and System State Persistence

Whenever a configuration or filter is created that proved to be useful, it would be nice to reuse this for future sessions. We therefore enable the user to store the created configurations and filters locally. If the user starts up the tool next time, the stored configurations can be loaded into the system very easily. The system always provides a default configuration, but the user is also able to set any of the stored configurations as default, such that the corresponding settings are automatically loaded when the tool is started.

The user can store or load a configuration by clicking the gear icon at the top of the corresponding view as seen in Figure 5.21. In this menu, the user can override the current settings file with the current configuration, save the current configuration into a new settings file, load an existing settings file, remove the active settings file, or set the current settings file as default.

Another persistence option is to store the current system state. Whenever an interesting data set is found, it might be useful to retrieve the current situation again at a later moment. By persisting the current system state, it can be loaded into the system again immediately without having to remember and manually recreate the situation. In the top bar of the application, the 'Edit -> System State' menu can be used to store the system state into a new file or load a previously stored state. The system state includes the following:

1. Current active Site in the Calendar View;
2. Current active day in the Timeline View;
3. The current state of the Site Filter;
4. The current state of the Timeline Configuration; and
5. The current state of the Event Filter.

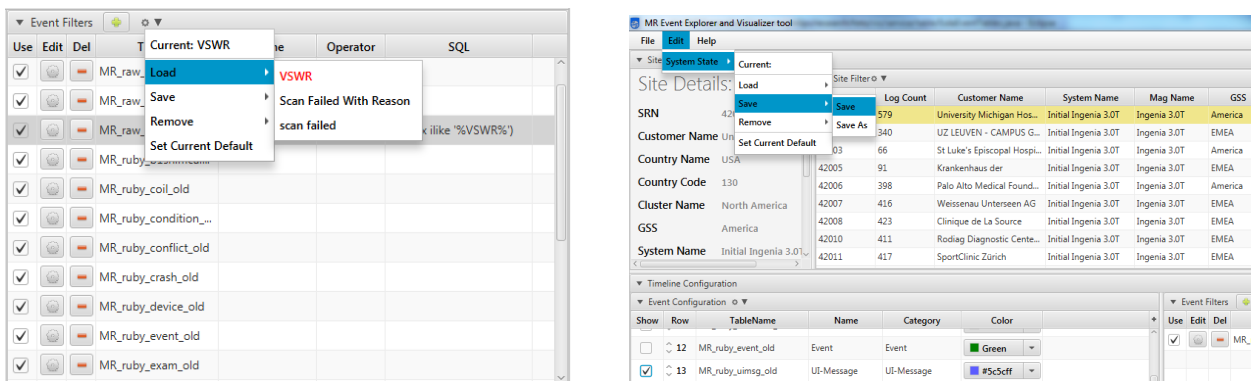


Figure 5.21: Storing or loading a setting or a system state.

Chapter 6

Evaluation

Now that the design and implementation of the prototype has been described in detail, we discuss whether the tool is able to fulfill the requirements that have been proposed. We will do this by first discussing the requirements and their relation with the developed views in Section 6.1. Second, in Section 6.2, we present the global feedback that we have gathered about the design and implementation of the prototype. Next, Section 3.2 presents and discusses a few use cases that have been executed using the prototype. Finally, we present possible future work to further improve our proposed method.

For the discussion below, we have primarily taken a subset of the systems into account, namely the *Initial Ingenia* and newer systems. The systems can easily be identified by having an identifier larger than 41000. For these systems, the processing scripts are able to retrieve the most information and are more accurate. For earlier releases, some of the event types cannot be constructed with the current scripts.

6.1 Evaluation of Requirements

We evaluate our approach (Section 3.2.4) and the requirements (Section 3.3) by summarizing the features of the tool and discussing the design of the tool based on the scalability of the data. Our approach was led by Shneiderman's mantra to create an interactive tool that was event focused and uses standard, easy to understand, visualization techniques.

6.1.1 Summary Prototype

We begin our evaluation by presenting a summary of the features implemented in the prototype and their relation with the requirements.

Data Functional Requirements

First, we have the *what* functional requirements (Requirements FW1 to FW5). These requirements state what aspects of the MR logs should be visualized. Based on these requirements, we have created four views that are shown simultaneously in the tool. The *Site View* shows a list of the available sites and their available properties (Requirement FW1-FW2). The filter feature enables the creation of a subset of systems of interest based on site properties such as location and system type. The *Calendar View* is the first visualization view. It uses an enhanced calendar system to visualize the distribution of events over a period up to a year (Requirement FW3). The *Timeline View* provides an overview of tasks and events (Requirement FW4) by visualizing glyphs on a timeline. The timing and occurrences of events can be seen more clearly by allowing zooming and panning of the visualized region. Furthermore, the most important event properties are shown immediately when hovering over an event glyph and the values for all properties of an event can be requested on demand by selecting an event in the timeline (Requirement FW5).

Interactive Functional Requirements

Second, we defined the *interactive* functional requirements (Requirements FI1 to FI3). We enable the user to analyze any arbitrary system, any event log, and any event by linking the several views together and decide the scope of the data shown based on the selections of the user (Requirement FI1). By creating an event filter that selects events based on the desired event properties, event logs of interest can be found in the Calendar View, or events in the Timeline View can be filtered (Requirement FI2). The last interactive feature enables the user to locally store the configurations, the filters and the system state that are proven to be useful, such that they can be retrieved again during another session (Requirement FI3).

Non-functional Requirements

Finally, we have the non-functional requirements and constraints that we defined for the design and implementation of the prototype (Requirements C1 to C5). Limiting the scope of events to a single system (Requirement C1), the confidentiality (Requirement C4) and the implementation as a desktop application (Requirement C5) are satisfied by construction. While the application have not yet been tested on another operating systems than Microsoft Windows, the prototype was implemented with Java and does not use any operating system specific features. It should therefore work, perhaps with a few adjustments, on other operating systems.

6.1.2 Scalability

The usability requirements state that the tool should react to user interactions in real-time and that the visualization views should be as uncluttered as possible (Requirements C2 and C3 respectively). The evaluation of these requirements requires more elaboration, since it greatly depends on the data shown. We evaluate these requirements by discussing how the scalability of the data affects these requirements.

Data Retrieval

We first shortly present some implementation details that are required for the discussion of how the scalability of the data affects the retrieval of the data. As described in Section 4.2, the data source used in the prototype is stored in the database, there are 21 event types available, and each event type involves several properties. Each event type is stored in an individual table inside the database. Hence, in order to retrieve information on multiple event types, we need to perform a single query for each event type that is involved. An average size event log contains about 12,000 to 17,000 events for a single day. The actual number of events is however about 10-20% larger, since we combined several coil events into a single event, which reduced the number of coil events by 80-90%.

Next to the retrieval of all sites, which only needs to be done once when launching the tool, there are two different types of queries: The Calendar query and the Events query. The Calendar query retrieves the information required for the Calendar View for a single event type and the Events query retrieves a set of events for a single day of a single event type for the Timeline View.

For the Calendar View, we have chosen to only show the distributions for a few event types. We therefore only have to perform a few Calendar queries whenever a new site is selected in the Site View. For the Timeline View however, we decided that, even though we do not shown all the event types by default, we want all the events for the current event log to be available as soon as they are requested by the user. As a consequence however, we need to perform 21 queries when events for the timeline have to be retrieved. Also the values of the properties, which are shown in the pop-up window when hovering over a event glyph, need to be available immediately on demand. To be able to perform queries as fast as possible, we only retrieve the information that is necessary, instead of retrieving all information for all required events.

Scalability Performance

If the number of events that are contained within a single event log grows significantly, the time required to perform queries and retrieve the data will increase. However, because the data seems to be indexed on log date and system identifier, events for one specific event log can be retrieved relative quickly. The time for initiating a query and locating the potential records requires most time of performing a query. After this, the time required to retrieve the actual events grows linear.

After the events are retrieved from the database, we need to process them and create the glyphs for the currently shown events. The computing time of this process increases linearly in the number of events. For an average sized event log, it requires about six seconds to execute all 21 Events queries and less than one second to process and display the events in the Timeline View. This is acceptable, since it can be expected that the user takes some time to analyze an event log before requesting a new one. The query time is however dependent on the current load of the database. The database is life, i.e., each day, new events logs are processed and stored into the database and other projects also use the database. Based on testing and using the tool, data retrieval can increase to about 15 seconds for average sized event logs. Even if the number of events grows significantly, the events can still be retrieved within an acceptable time frame. The same holds for performing the Calendar queries, which commonly can be retrieved in less than three seconds.

When thousands of events are shown in the timeline, it has a large effect on the time it requires to draw the events, therefore the initial zooming and panning may not be as responsive. For about 20,000 events it can take up to a second to redraw all the events. However, most of the analysis is targeted at shorter periods. When zooming in on the data, less events are shown and the interaction becomes more responsive.

As described in Section 4.2, new processing scripts are being developed. It is therefore likely that the number of event types will increase in the foreseeable future. As presented above, the largest impact of retrieving events is caused by performing individual queries for each event type. Hence, if the number of event types increases, then the query time also becomes significantly larger. If the query time becomes too large, then there are some options for improving it. We do not show all event types by default and some of the event types are not very relevant in most cases. To improve the query time, we can ignore those events that are not necessary and only load them on demand.

The data size also has an impact on the amount of memory that is used by the implemented prototype. The default shown event types cause about a quarter of the events for a daily event log to be shown. In the case of an averaged sized event log, about 4000 events are shown and the system uses about 80 - 100MB of memory in total. If all event types are shown at the same time, then the amount of memory used increases to about 160-180 MB. The largest proportion of the memory is used to store the visual representation that is used for rendering the events. Approximately, the memory used grows linearly in the amount of events.

The large increase in memory is mostly caused by the interface library that is used to create and draw all the interface elements. Some details on the implementation and the library is presented in Appendix B. We have chosen to use the library to decrease the time to implement all the features.

Again, since not all event types are as important, the memory used is manageable. An example of an unimportant event, that has many instances in each event log, is the general *event* event type. This event type reports the start and end of most of the interval events and can therefore be considered to be redundant. There are more event types that are expected to be used rarely. On the other hand, the *User Interface* event type, which can be very informative when errors have occurred, is not shown by default. These events however account for about 20 to 25% of the total number of events. The majority of the UI message events (80-90%) however do not contain any message, and a simple filter can be created to ignore them, greatly reducing the amount of memory used.

In conclusion, there should be enough memory available to allow an increase in data size. For improving performance, one of the changes that can be made is to further limit the amount of data that is being retrieved from the database. The event types that are not currently active do not have to be loaded from the database until they are enabled for example. Another option is to manage the rendering of the events without using an extensive library.

Scalability Visualization

Next we discuss the impact of the size of the data with respect to the cluttering of events. The Calendar View is hardly influenced, since no extra elements are added by an increasing data size. If the maximum number of events for an event type increases, then the bin size for each color also expands. This can affect the precision of the visualization. This can however be easily solved by adding extra bins or by using a dynamic bin size that is dependent on the available data.

In Section 6.3, a use case is discussed describing regular event logs and Figure 6.3 shows the timeline of such a typical event log that has approximately an average size. For these logs, some of the rows in the timeline are already somewhat cluttered when the events for a whole day are shown. This is mostly

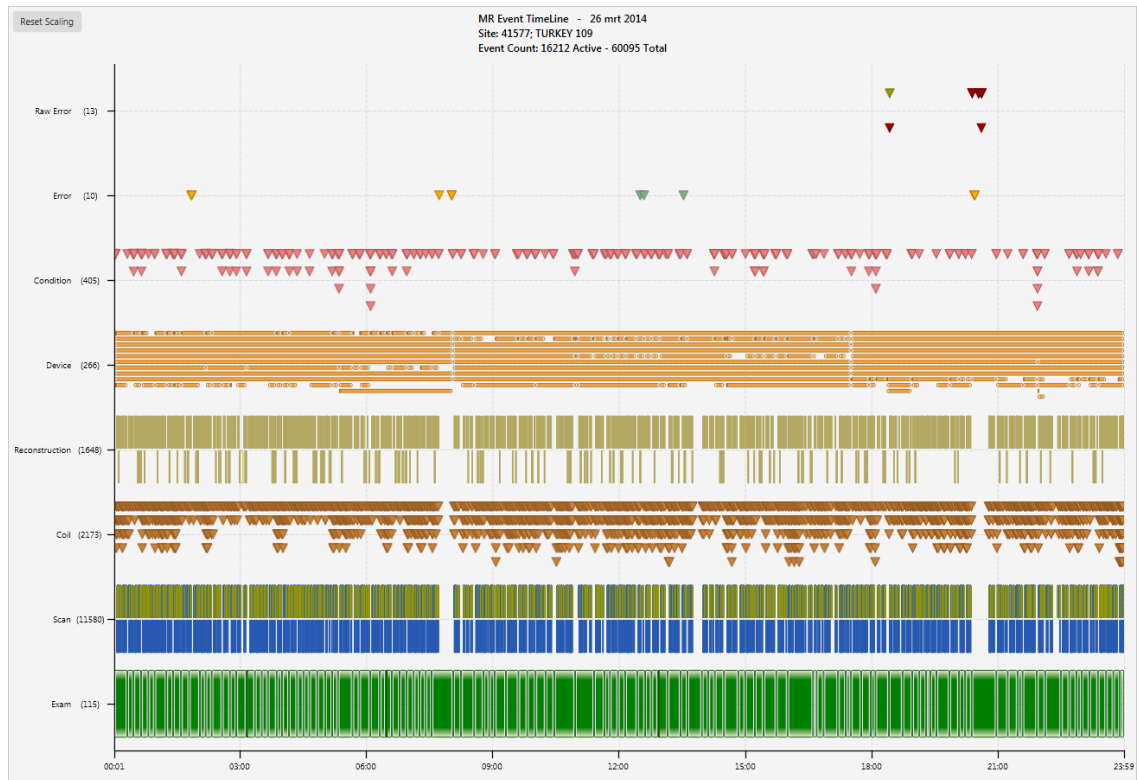


Figure 6.1: Timeline with high utilization and cluttering of events.

caused by the fact that events are produced most when the system is being actively used. Therefore large groups of events occur within a small period of time. Visually, the events that are produced with a high frequency are affected the most. For instant events, the glyphs overlap for a large part horizontally. For the interval event, the short bars of the relative short events are visualized consecutively, which might give an impression that these events are not intervals. Both these aspects can be confusing if the user is not yet used to the visualization. The cluttering does however give a good impression of the amount of utilization and whether the system was able to behave as normal. Knowing this, also periods of abnormal behavior can be seen, which can help the user determine which periods might be interesting to look at.

Figure 6.1 shows an event log with a significant larger number of events. It shows the impact of an increasing data size well. The system is utilized for a longer period and the average examination time is much shorter than the event log shown in Figure 6.3. In conclusion, an increase of data for the currently shown event types does increase the amount of cluttering, which can be confusing for users that have little experience with the visualization. However, most of the time, not the entire event log is displayed, since the user has to zoom into the data to be able to analyze the events in more detail. But the timeline could be improved, such that the overview of all events is shown more clear and less cluttered. We can for example combine events on a larger scale to reduce the number of glyphs shown. It should however be clearly visualized, such that the user does not get the wrong impression.

Enabling other event types for visualization does also increase the number of events and thus also increases the amount of clutter. As can be seen in Figure 6.3, the vertical size of most interval bars is still quite large, so it is possible to create space for additional event types, but this amount is limited. In Figure 6.2, the same event log as in Figure 6.3 is shown, but with all event types enabled. The Site View needed to be closed, otherwise the height of each category would be too small. Currently, a total of 21 event types are available that are configured into 15 categories. It is likely that this number of rows is the limit, when the overlapping of events is taken into account. Therefore, if more event types need to be shown by default, it might be necessary to place more event types into a single category.

In conclusion, adding additional event types does make the timeline more cluttered. However, like



Figure 6.2: The visualization of a normal average sized event log, but with all event types enabled.

stated before, it is highly unlikely that all the event types need to be displayed at the same time. It is however advised to adapt the event retrieval process, since it can significantly increase the time required for retrieving all events.

6.2 User Feedback

During the design and implementation of the prototype, we have discussed the features with some target users. Furthermore, after the prototype was more or less finished, we have shown the system to several people, including a diagnostic designer, an experienced service engineer, a systems engineer, and other researchers that work with the same data source. In this section, the general feedback that has been gathered is presented. The quotes presented below are a combination of literal quotes that have been obtained from email communication, and summarized vocal feedback that has been received as part of demoing the prototype.

First, the global impression of most people was that the prototype can quickly provide an overview of the utilization and behavior of an MR system:

“If you have a certain system and event log in mind, then with a few clicks it is possible to view the utilization and the errors that occurred in an MR system.”

“The prototype enables the user to quickly dive into the data and get a nice overview of the events that have occurred within the system.”

“The prototype works fine. The selection based on statistics and the color coding is nice. The filtering is cool. I could derive the usage of the features and get information from playing around, which is good because then the tool is intuitive.”

It was also noted that the current prototype shows potential to be used in several areas such as diagnostic design, service support and research and development, but currently a lot of domain knowledge and effort is required to effectively being able to analyze event logs.

“The tool has potential for several areas, including remote support. However much domain knowledge is required for effectively analyzing the events. People working in maintenance services need to be able to quickly see and analyze relevant parts of the event log. Analyzing the whole set of events requires too much effort to quickly see why something went wrong.”

“The event timeline of the prototype could be used to monitor the utilization and behavior of new systems or new system components. By showing the various events it is possible to see how the new system is utilized and when errors occur.”

The filter feature does give the user the ability to find event logs in which particular events occurred for a single system. But there is however also the need to be able to compare particular use cases for several systems. The prototype does not give the ability to find systems which showed similar behavior at any moment.

“I like the overview of events and it is nice to see what happened when errors occurred in a system. But what I am missing is the ability to see for which systems similar problems have occurred.”

“If I am looking at a site with a specific error or problem, then I want to know how often and in which systems the same error occurred, and does the problem for example occur for very specific scan types.”

Finally, Krelis Blom, a systems engineer for MR systems, who is also involved in the MEBEF project, has shown his interest in the tool and he sees some potential of integrating some parts of the tool into the MEBEF dashboard.

“I would like to have the functionality embedded in the MEBEF tool, which will be very helpful for people working in development and experts working at service. It will position perfectly between the product (reliability) and serviceability branch of the MR-Eye vision”

Extended Field Testing

We have shown the current prototype to several people, each of whom is working in different areas that are related to imaging systems. We were able to receive some general feedback on the current implementation of the prototype. We have also asked several people to try out the tool themselves and check whether some insight can be gained based on known use cases. However due to time constraints and other reasons, most were not able to do this. Therefore, very little feedback on practical usage of the tool was gathered. The prototype therefore needs more extended field testing to determine to what extent the tool can be used in the context of serviceability, diagnostics and R&D.

6.3 Use Case Analysis

In the evaluation of the requirements and the user feedback we have shown that the tool can effectively show the daily utilization and behavior of a system. Something that we not have discussed yet is the question whether it is possible to perform any manual diagnostics by using the tool. In other words, are we able to diagnose when and why errors are produced by components of MR systems or are we able to find the root cause of some failure?

As noted in the previous section, we have regrettably not been able to perform an extended field test nor were we able to get a good impression of how a product expert or diagnostic designer would use the tool to analyze event logs. For these reasons, we cannot present advanced use cases that have been analyzed by someone with detailed knowledge of MR systems and the corresponding event logs.

Instead, we present an analysis of three uses cases by utilizing the prototype. We take on the role of a data analyst who has some knowledge of the event logs and the MR model. First, based on the knowledge that has been gained during the development of the tool, we describe event logs that show regular behavior.

Second, we inspect some event logs that show abnormal behavior. Finally, we further analyze the use case that has been presented by Ingmar Graesslin, a research scientist, working for Philips in Hamburg, Germany. He described a use case he was interested in, which involves the occurrence of a very specific error.

6.3.1 Regular Event Logs

Before we analyze errors or irregular behavior of systems, we first present what a regular event log looks like. Figures 6.1, 6.2, and 6.3 show the events of an event log for a full day, in which the system was used continuously and almost no errors occurred. We expect that most of the time MR systems behave and function as normal. These event logs should therefore give a good impression of what a regular event log looks like. Figure 6.4 shows a subset of events in more detail by zooming in into the timeline and showing the properties for some events. Based on the visual patterns of the event glyphs and the information obtained from the properties pop-up, we can follow the steps that the MR system performs and we can see when particular events are produced.

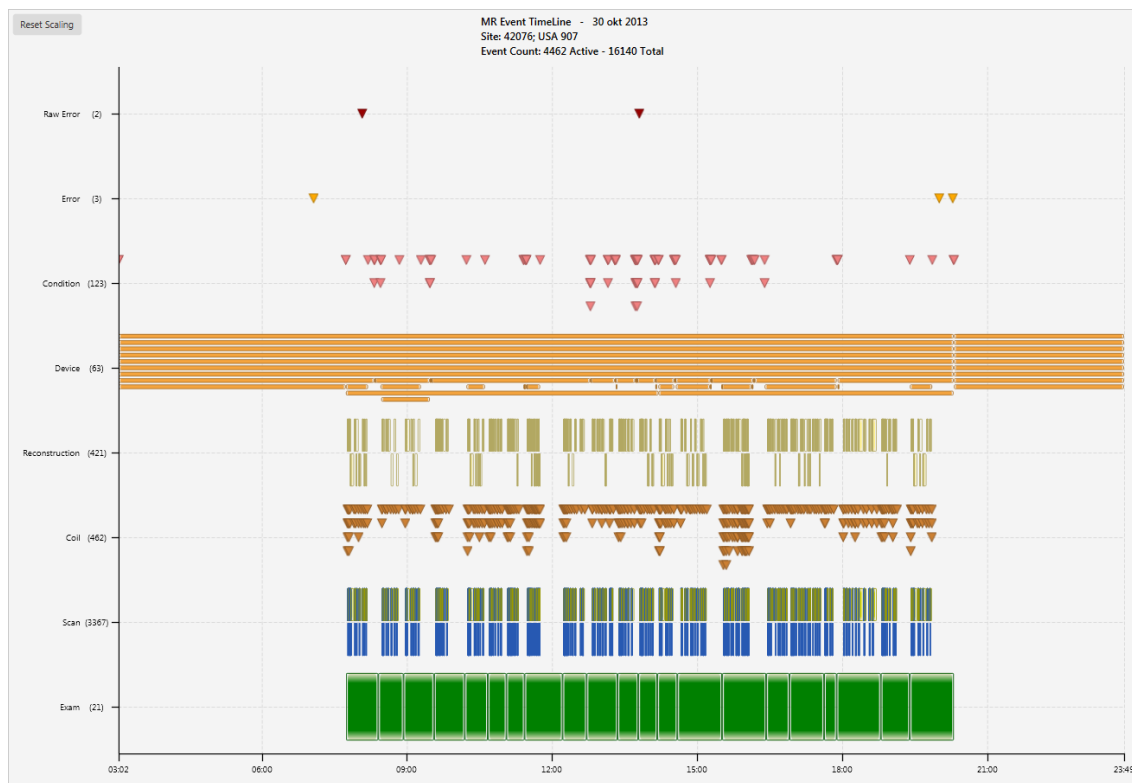


Figure 6.3: Timeline rendering of a typical average sized event log.

On regular days, a MR system can continuously be used for the examination of patients. Most often, the starting time for the next exam is used for closing the current exam, but there are however also other conditions that can start or end an exam. During an exam, multiple scans are made and each scan is preceded by several preparation steps, which can occur in parallel. Preparation and scan events are visualized by the blue and yellow bars in the scan category respectively. Each scan performed has a different name, this is because each scan scans the anatomic region in a different way such that different kinds of images can be reconstructed. After a scan is completed, the related coil events are produced and a new scanning phase can occur. Before a coil can be used, it has to be connected to the system. The device events clearly show that the correct coils are connected to the system (see annotations in Figure 6.4). Coils can stay connected for multiple exams if they are required for following exams. Some other components and coils are fixed in

the machine, causing them to be always connected. Moreover, most of the condition events also indicate when a device is being connected or disconnected from the system.

6.3.2 Analysis of Abnormal Event Logs

In this use case, we analyze the data for event logs in which a system showed abnormal behavior. Furthermore, we look if there is any relation with the data and the calls that have been made reporting the problem that is shown in the event log.

Let us start with selecting a set of suitable systems to analyze. There are many MR systems available, but we do not want to browse the data of just any random system. As stated at the start of this chapter, the data of the MR model is most accurate for the newer set of systems. We therefore decided to use the site filter feature to limit the set of systems to the Initial Ingenia 3.0T systems that have a system identifier ranging from 41000 to 50000. To further filter the list, we limit the list to systems that are placed within the Benelux. A subset of the resulting list can be seen in Figure 6.5.

We are interested in analyzing the context in which errors occur and how these errors affect the behavior of a system. For finding event logs with abnormal behavior, we inspect the *hardware error* events and the *calls* in the Calendar View. The hardware error event is the most common amongst the error events that are part of the MR model. If the client made a call, it can be the case that the client has reported a problem at customer support.

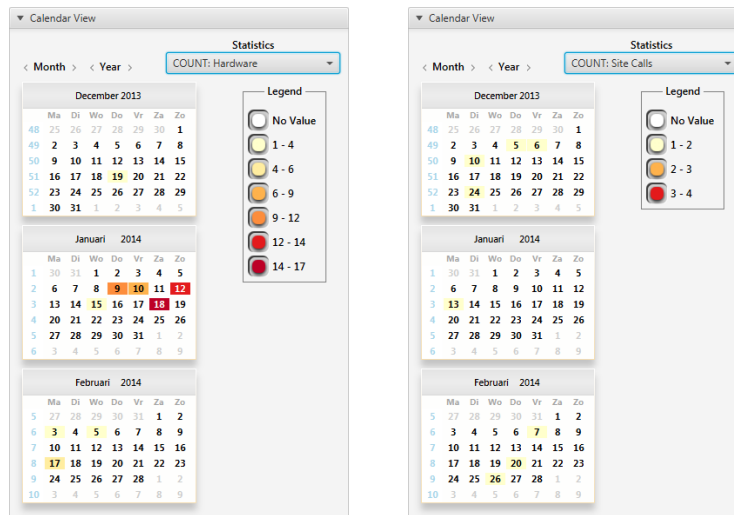


Figure 6.6: Calendar View for one of the systems in the selected subset of system. In January there are several event logs that contain a relative large number of hardware errors. However, only a call was made for January 13th. It is therefore likely that on January 12, a problem occurred that had an impact on the system.

For several systems, we inspect the Calendar View for the hardware and call events. We then select an event log that had relatively many hardware errors and for which a call was made for the same or next day as shown in Figure 6.6. The corresponding event log is shown in Figure 6.7. From the overview, the period that was influenced by the hardware errors is shown clearly. The hardware events only occur during a few examinations. Each of these examinations are related to examining a knee. Since no other examination types are affected, it is likely that there is a problem with a component that is responsible for scanning knees.

To get a better impression of what went wrong, we need to zoom into the data. Figure 6.8 shows a part of the period that was affected by the problem in more detail. By inspecting the pattern of events and the properties of several events, it becomes clear that the problem is related to the knee coils. The default set of event types however gives little information of what is wrong with these components. To get some more information, we enable various hidden event types and inspect whether any other events are related to the problem and whether more information can be retrieved. After inspecting several *UI-Message* events,

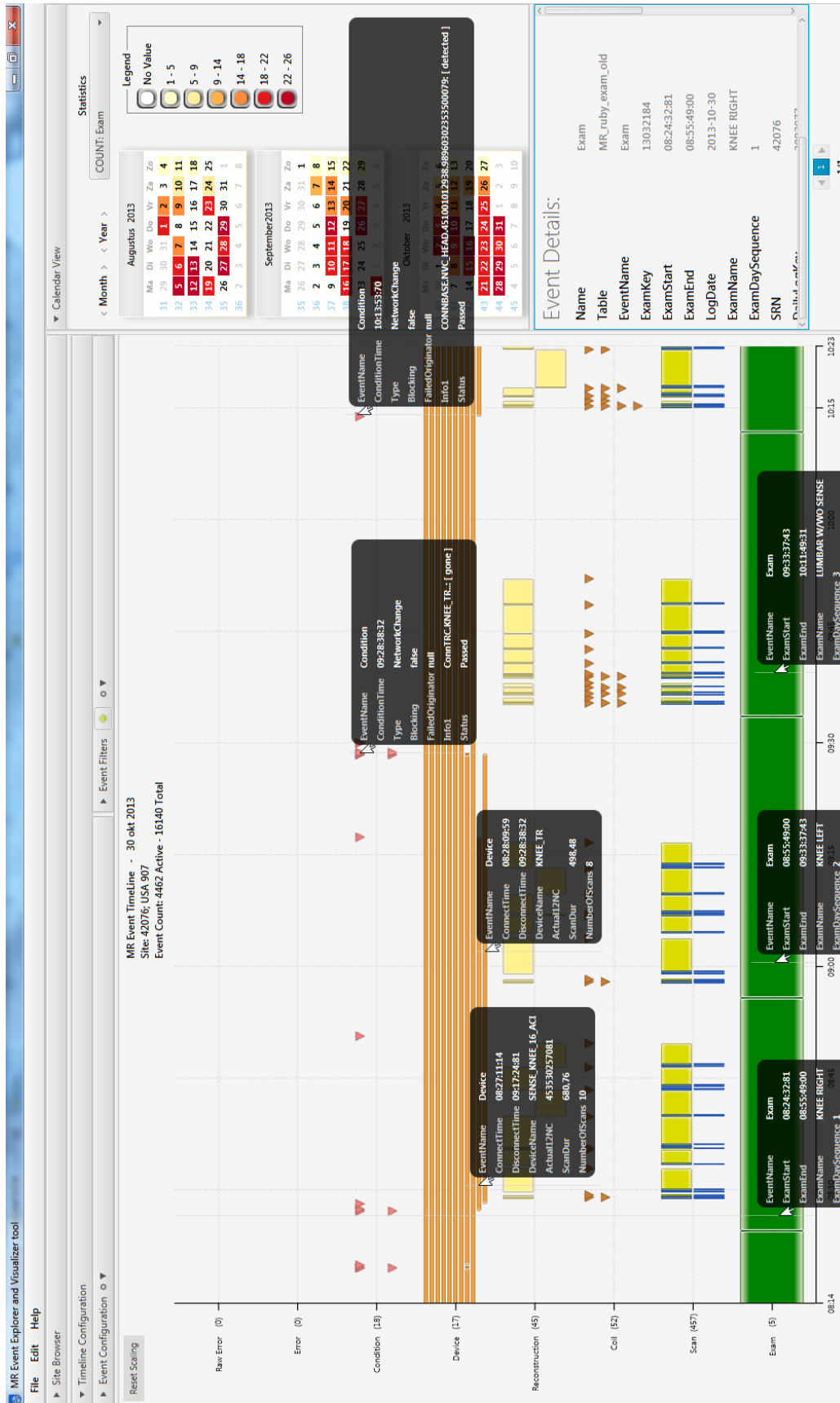


Figure 6.4: Zoomed in and annotated timeline of a normal event log.

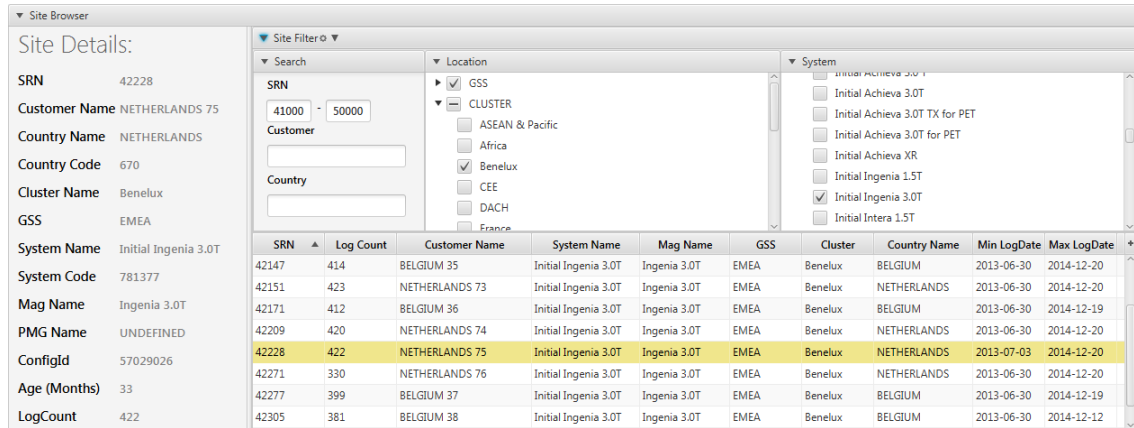


Figure 6.5: A site filter for selecting a small subset of systems.

which have been filtered such that only UI-Message events that actually contain a message are shown, it is clear that relevant information has been reported to the system operator. There are several messages giving various possible reasons for the problem. Each of these messages are repeated several times during the period of the problem. There are two messages that most likely explain the problem. It seems that something is wrong with the connectors causing the coils to be unable to connect properly to the system. These messages are displayed in Figure 6.9.

Our assumption is confirmed by the administrative data that is related to the call that reported the original problem. The field service engineer has reported that a connector was unplugged from the transmitter coil. He has reconnected the connector and the problem was resolved. If we look through the history of calls and performed jobs, it shows that this kind of problem is not uncommon. A product expert explained to us that the high frequency vibrations due to gradient coils, can lead to disconnected or broken connectors or parts.

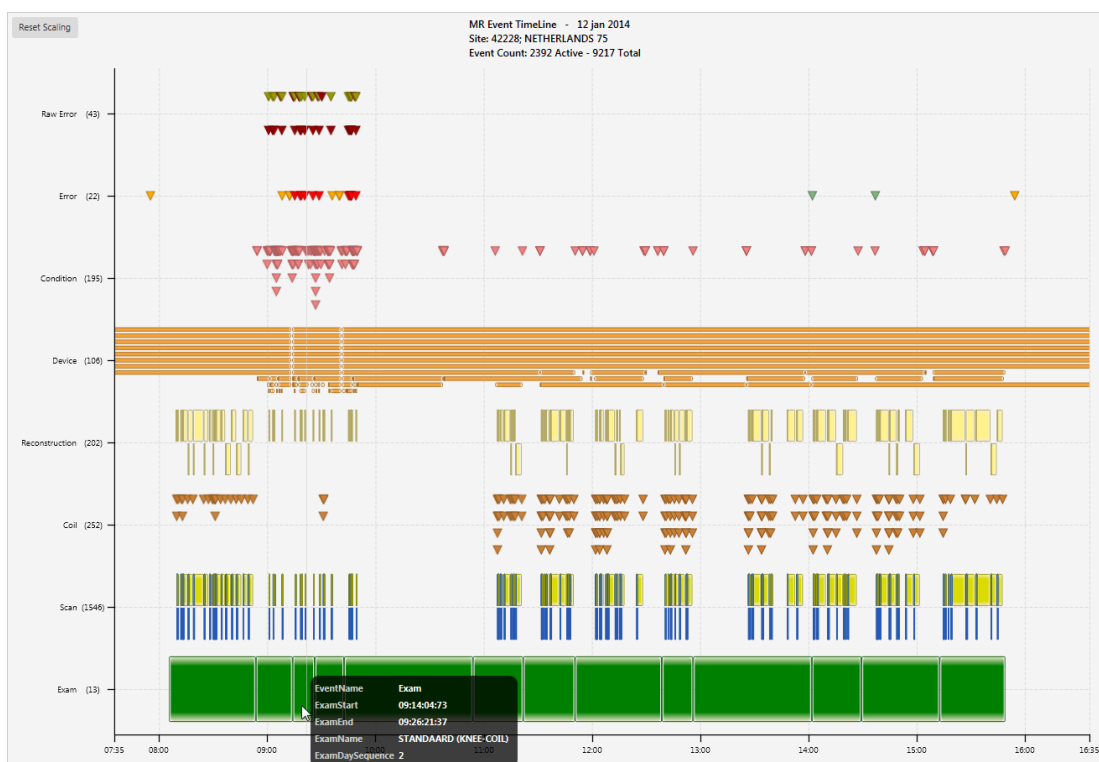


Figure 6.7: An event log that contained a relatively large number of hardware errors. The hardware events (the red triangles in the error category) are concentrated into a single period during a few exams and they are also accompanied by several other errors. For the rest of the day, the system shows normal behavior.

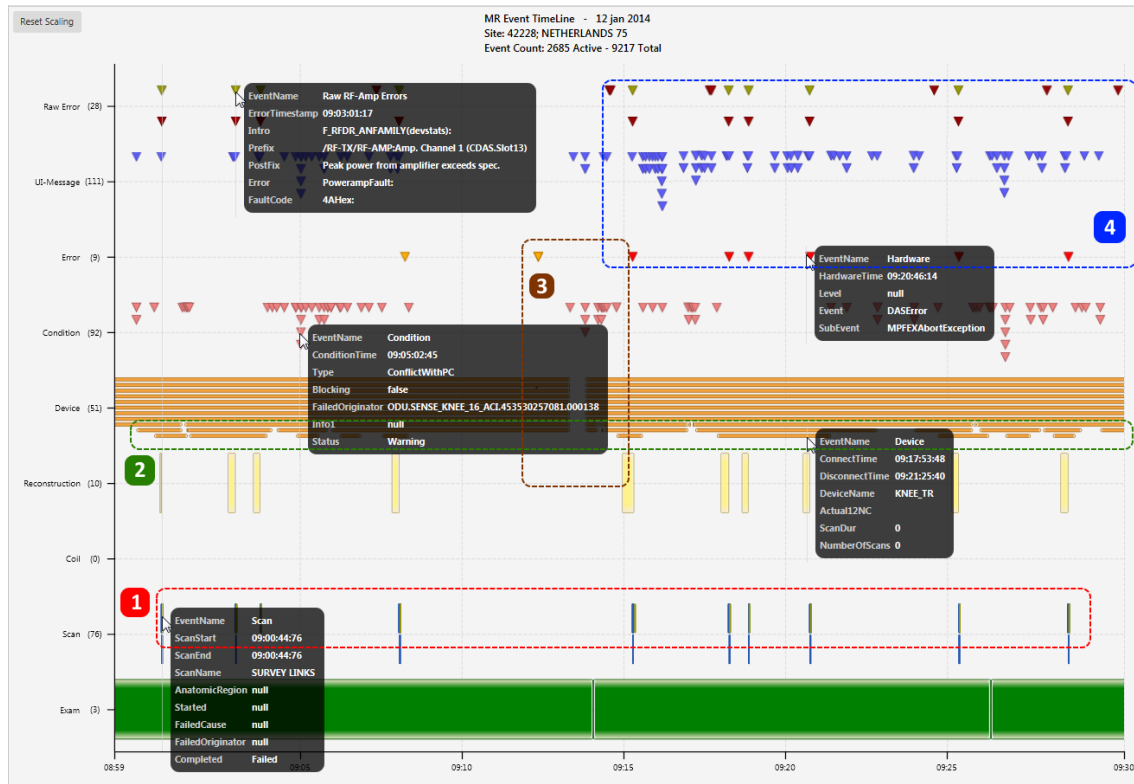


Figure 6.8: A period of an event log for which many errors occurred that is related to the failure of a knee coil. Several patterns can be discovered by inspecting several events. 1. First, almost each time an error is reported, it causes the current scan process to fail and to be aborted. The value of the *Completed* property for each of these scans is *Failed*. 2. Second, several times per examination, the system tries to connect two different knee coils, but it stays connected. 3. After the first knee examination was unsuccessful, there are two *Crash* events (yellow triangle under error) stating the stop and start of a foreground application, and all devices are disconnected and connected again. Therefore, it seems like either a system or a software restart was performed in an attempt to solve the problem. 4. After the reboot, the same examination is attempted, but this time even more errors are produced.

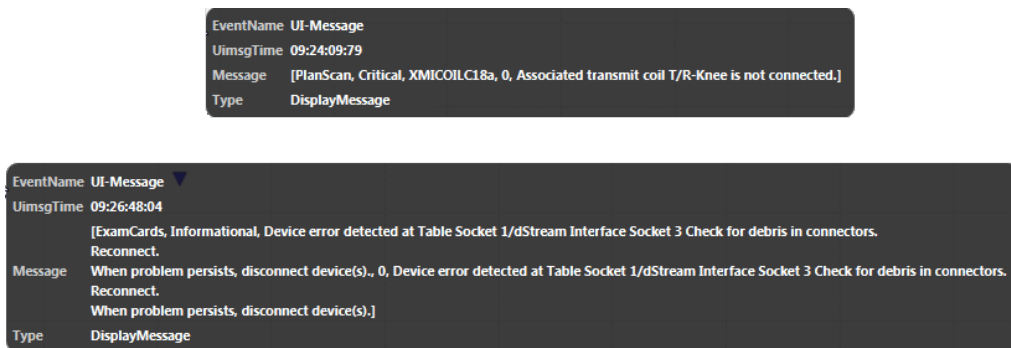


Figure 6.9: Two error messages that give a good indication where the problem of the event log displayed in Figures 6.7 and 6.8 is originating from.

6.3.3 RF Amplifier Error Analysis

In the final use case, we further explore the use case that was presented by Ingmar Graesslin. He has shown interest in a very specific RF-amplifier error. The error message states the following: *Reflected power or VSWR high*. VSWR is an abbreviation for Voltage Standing Wave Ratio and it measures the ratio between the power that is output by the RF pulse signal source and the power that is converted by the RF amplifier [3]. Optimally most power is utilized by the amplifier and almost no power is being reflected. If the amplifier is unable to convert enough power, then the system is unable to accurately scan the patient, causing noisy images with poor contrast. Moreover, a large amount of reflected power can cause damages to the related components.

Due to time constraints, Graesslin was able to use the tool for only a brief period. He was interested in what context the error occurs most often and whether it is related to any other events. He noted that the error most often only occurred for a single examination. It therefore gives no direct indication of a broken component.

Given the information from Graesslin, we take a closer look to the RF amplifier error. We first queried the database to check how often this error occurred and how many systems have experienced it. The error has been reported at least once by 71 systems with an identifier larger than 40000. For 89% of these systems, the error occurred in less than ten event logs. For only four systems, there are more than 20 event logs containing the error. Hence, we can conclude that this particular error occurs quite rarely.

By using the data obtained directly from the database and the event filter feature, we are able to easily find the log dates that contain the error. A part of an event log in which the error occurred is shown in Figure 6.10. After inspecting several event logs, some patterns can be seen:

1. Whenever the error occurs, it causes a scan to fail. In some occasions the error occurs several time in a row, effectively terminating the examination. In other cases however, the error occurs only once and the system is able to recover and complete the examination. However, there is no information available about the quality of the images that have been reconstructed. Finally, occasionally the error is produced multiple times a day. However, no direct connection can be found between the different exams or scans in which the error was involved.
2. The error is always repeated by the *Raw Hardware Error*. From experience, this happens most often for any RF-amplifier error and is likely caused by a duplicate definition of errors in the processing scripts.
3. In many occasions, a set of particular UI-message events is produced before this RF-amplifier error. The messages report on the time it will take to power-up the RF-amplifier. The timer often starts at around one minute. It can be an indication that the amount of power received by the amplifier is too low and thus it takes a long time for the amplifier to charge up. By creating a filter for the user interface messages, it can be seen that the power-up messages correlate well with the occurrence of the VSWR RF-amplifier error. The correlation between the two events can be seen well in Figure 6.11.

The problem is often solved by simply rebooting the system. The administrative data further reveals that in other occasions cables were a little loose or broken.

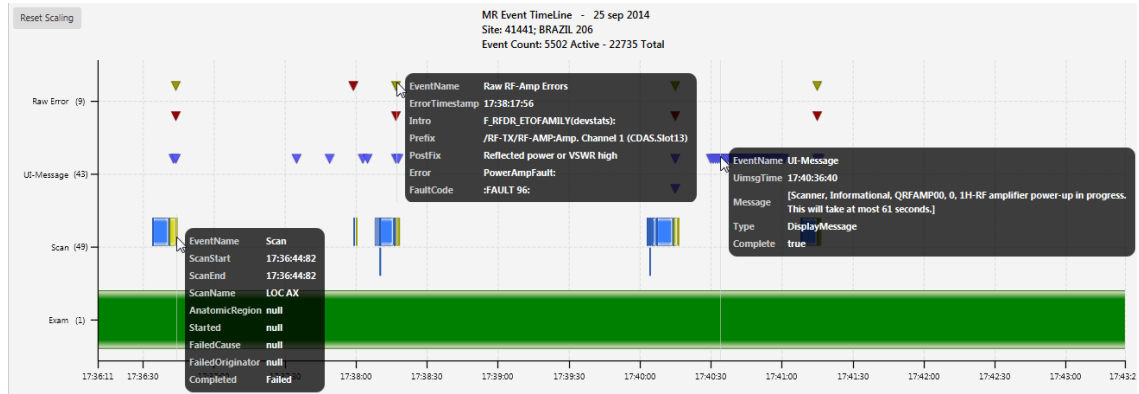


Figure 6.10: VSWR (reflected power) error. This error is relative rare. When this error occurs, it is often only for a single exam and it commonly does not affect any future exams.

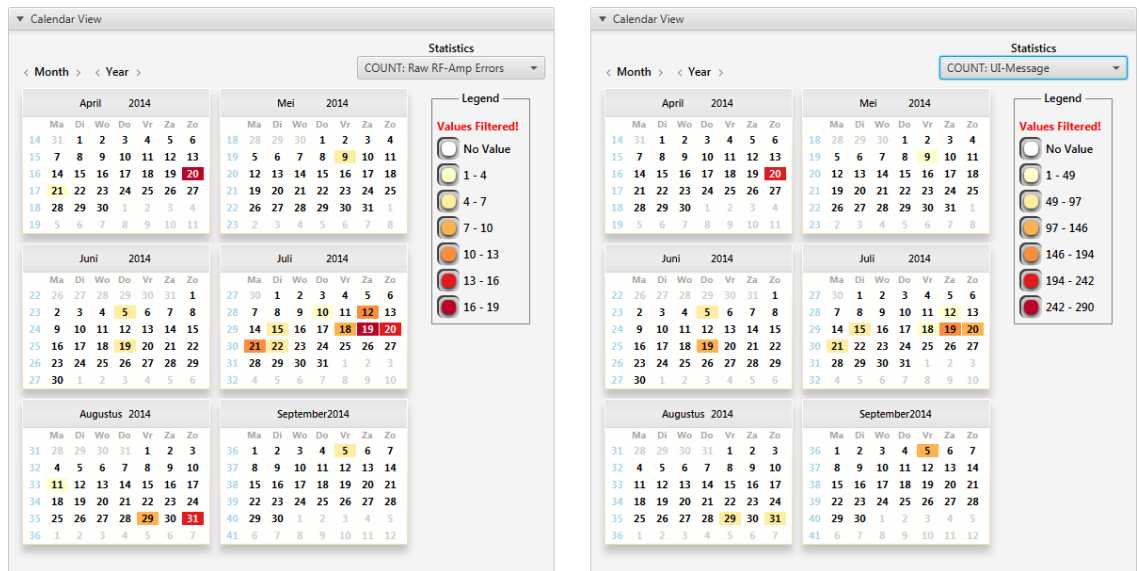


Figure 6.11: The power-up user messages correlate well with the occurrence of the VSWR RF-amplifier error. For each event log that contains the power-up message also has a number amplifier errors. This indicates that this message is only produced when the amplifier has problems charging up.

6.4 Future Work

The proposed method and the prototype was not solely designed by ourself. During the project, René Bergmans, who has worked in the area of MR remote service support has been our primary external contact for the design of the application. He and other people to whom we have shown the prototype have suggested features that might be useful for the tool. However, due to time constraints or due to the scope of the project, some of their ideas have not been implemented. In this section, we present several of them.

Visualize Event Distinction

Currently, the events of the same event type are all visualized using the same color and the same glyph. This makes it difficult to quickly see a distinction between the events of the same event type. For example, for the scan events we would like to immediately see which type of scan event was performed or whether the scan was failed, aborted or completed. Currently, the user has to hover over each of the scan events individually and look at values displayed in the property popup window. This is tedious when there are many events.

Two options were proposed for solving this. The first option involved the use of labeling. For each event, a label is shown along with the glyph. For each event type, it would be possible to configure which property value should be displayed. For interval events, the label could be placed within the interval bar whenever there is enough space for it. For instant events, this is more complicated, since no space is available within the glyph.

The second option that was proposed could be a better option. It allows the conditional formatting of events glyphs. The feature can be compared with the conditional formatting within Excel. Based on conditional formulas involving one or more values, the current cell can be formatted in several ways. The same can be applied to the glyphs of events. For each event type, one or several glyph visualization configurations can be created. If the event satisfies the conditions specified in the configuration, the look of the glyph can be adjusted. The glyph could use a particular color for the glyph itself or for the border of the glyph. Another possibility is to show another glyph shape. However, just like discussed for the filter feature, these configurations have to be carefully designed, such that the user can create these in a simple manner. Furthermore, the formatting can also be in conflict with the current coloring of events.

More Statistics for the Calendar View

The Calendar View currently only shows the number of events for a set of selected event types. The calendar could however be used to show other values, which might be of larger interest for selecting an event log. A simple addition can be to show the sum of multiple event types or the ratio of one event type to another, e.g., total number of error events, the average number of scans per exam or the average number of errors per scan event. Also, event properties could be involved. It might be interesting to see the average duration of interval events or the average or maximum value for a particular scan parameter. The visualization of the calendar does not have to be adjusted for this, since the values shown can still be divided into bins.

Visualization and Display of Additional Data

The prototype currently only visualizes the events and optionally the values of their properties. For the imaging system, there is a lot more data available:

- Some events contain parameter values, and also, continuous values from sensors are available.
- The administrative data concerning the maintenance of the systems. Currently, only the Calendar View shows when calls have been made or when maintenance jobs have been performed.
- System configuration and hardware components of individual systems.

For a larger context, it can be interesting to also show these aspects. When extending the different kinds of data that are shown, it might be better to integrate our event visualization methods along with the new aspects into an existing or a whole new tool.

Show Similar Systems

The tool can be used to analyze the utility and behavior of MR systems. If the user is able to gain some insight on some aspect, it can be very useful to compare the findings to the event logs of other systems that showed similar behavior or in which the same error occurred. By using the filter feature and the Calendar View, the user is enabled to find other event logs with similar events, but only for the same system. The user needs to manually switch between several systems, and possibly multiple event logs, to see whether a similar situation has occurred in other systems. If the case that is being analyzed is not very common, it can take a long time to find another suitable system without resorting to other tools or methods, like manually performing queries in the database. Hence, automated support for this would be welcome.

Cooperation with Development Processing Scripts and Target Users

The processing of event logs into the MR model is currently live. This means that the raw event logs that are gathered from the systems are processed into the MR model daily. However, in order for the current system to be used for monitoring systems that are being tested, there needs to be a cooperation between the system developers and the developers of the processing scripts, to make sure that new types of event records can be parsed by the processing scripts. This process can also increase the precision of the processing scripts for the live system.

Moreover, in order to determine whether the proposed method can actually be useful in practice and for determining the actual further development of the tool, there needs to be more contact and cooperation with the target users.

Chapter 7

Conclusion

In this thesis, we explored the possibilities for visualizing event logs of medical imaging systems for serviceability purposes, since the exploration and analysis of large textual event logs of medical imaging systems is a difficult problem. These event logs contain various types of information and many of the records are irrelevant for maintaining medical imaging systems.

To support failure diagnostics for corrective maintenance and failure prediction for predictive maintenance, Philips has performed research to automate the discovery of event patterns and predictive models. However, besides the MEBEF reliability dashboard, which shows some utilization metrics and an overview of occurred errors for an MR system, Philips has invested little in the research of visualization methods for the analysis of event logs. Visual data exploration of event logs has been shown to be valuable for several applications, such as electronic health records, web server logs and event logs of wafer scanners.

Based on the related work and in discussion with target users, we have designed and implemented an interactive visualization prototype that visualizes the event logs of MRI systems. The prototype was built for the analysis of MR event logs that are described by the MR model. The prototype enables the user to quickly view and analyze the utilization and behavior of an MR system.

The large amount of data poses a problem for visualizing event logs. Even for a single MR system, the total number of event logs and the number of individual events per event log is too large to be shown at the same time. In our approach we therefore apply Shneiderman's mantra. The data has been split into multiple levels, where each level has a different scope of the data. First, the Site View presents an overview of the available sites. Second, the Calendar View provides an overview of the distribution of events over the entire history of the system. Next, the Timeline View presents an overview of the events for a single day. And finally, the Event View shows the details for a single event.

To further reduce the amount of data shown, the user can filter or zoom into the data. The Site View offers a filter that can reduce the number of sites based on some basic properties like location and system type. In the Timeline View, semantic zooming can be used to determine the period that is shown, which can effectively reduce or increase the number of events that is shown. Finally, the event filter, which can filter events based on event properties, enables the user to find event logs for a single system that contain specific events.

The large amount of data also has an effect on performance. The data is stored in a database and the time required to retrieve the data is largely affected by the number of event types. The time currently required to retrieve the essential information for all the events of a single event log is acceptable. However, if the amount of data or the number of event types grows, it is advised to limit the amount of data that needs to be retrieved at any moment. Furthermore, the interface library that is used for the development of the prototype is not optimal in terms of memory usage or rendering performance. If the current prototype is used as a basis for the further development of the tool, then it is advised to use a different technique for rendering the data.

A final concern for the amount of data is the cluttering of events in the Timeline View. First, the overview of all events for a single day gives a good indication of the amount of utilization and the general behavior of the system. Second, the interface of the prototype allows the user to enlarge or shrink individual views. This allows enough space to be reserved for the Timeline View when many event types need to be

shown. The large number of events can however be confusing if the user is not yet familiar with the visualization. The visualization of events could for example be improved by introducing several levels of detail.

The prototype has been shown to several people, including a diagnostic designer, an experienced service engineer, a systems engineer, and other researchers that work with the same data source. The general feedback from them is positive. They see potential in the prototype, but they note that some additional features could be very useful, such as the automated discovery of similar systems and visualizing the distinction between events.

We have performed an analysis of some use cases to show the usability of the prototype. First, we have shown what regular event logs look like. Second, we have used the tool to look for event logs that show irregular behavior and we were able to derive the cause of a problem that had occurred in a system. In the final use case analysis, we performed a data analysis on a use case that has been presented by a researcher from Philips. Finally, we presented some potential future work for the expansion of the current prototype.

In conclusion, the results show that we have built a prototype that is able to visualize the event logs and that can be used to effectively analyze the utilization and behavior of an MR system. However, we were unable to extensively test the prototype. It is therefore not yet clear how the tool can be used by product experts in the context of serviceability, diagnostics, or research and development.

Bibliography

- [1] Michal Aharon, Gilad Barash, Ira Cohen, and Eli Mordechai. One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs. In *Machine Learning and Knowledge Discovery in Databases*, pages 227–243. Springer, 2009. 22, 23, 24, 25, 26, 27, 28, 48
- [2] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of time-oriented data*. Springer Science & Business Media, 2011. 17
- [3] CPC Amps. Introduction to nmr/mri amplifiers. <http://www.cpcamps.com/introduction-to-nmr-mri-amplifiers.html>, October 2015. 71
- [4] Stephen G Eick, Michael C Nelson, and Jerry D Schmidt. Graphical analysis of computer log files. *Communications of the ACM*, 37(12):50–56, 1994. 17, 24, 25, 26
- [5] Akihiro Hashimoto and James Stevens. Wire routing by optimizing channel assignment within large apertures. In *Proceedings of the 8th Design Automation Workshop*, pages 155–169. ACM, 1971. 50
- [6] Joseph P Hornak. *Basics of nmr*, 1997. 33
- [7] Daniel A Keim, Christian Panse, Jörn Schneidewind, Mike Sips, Ming C Hao, and Umeshwar Dayal. *Pushing the limit in visual data exploration: Techniques and applications*. Springer, 2003. 15, 46
- [8] MagLab. Mri: A guided tour. <https://nationalmaglab.org/education/magnet-academy/learn-the-basics/stories/mri-a-guided-tour>, September 2015. 34
- [9] Yi Sum Irene Man. Condition-based maintenance of magnetic resonance imaging systems. Master’s thesis, Eindhoven : Technische Universiteit Eindhoven, 2015. 5, 34
- [10] Philips. Philips research - intranet webpages. <https://pww.research.philips.com>, September 2015. 33
- [11] Philips. Rightfit service agreements imaging systems - philips. http://www.healthcare.philips.com/ca_fr/support/customer_services/rightfit_service_agreements/imaging_systems.wpd, September 2015. 6
- [12] Johan Ribrant. Reliability performance and maintenance survey of failures in wind power systems. *Unpublished doctoral dissertation, XR-EE-EEK*, 2006. 4, 6
- [13] A Röder, V Vasyutynskyy, K Kabitzsch, Th Zarbock, and G Luhn. Log-based state machine construction for analyzing internal logistics of semiconductor equipment. In *Proc. International Conference on Modeling and Analysis of Semiconductor Manufacturing*, pages 54–60, 2005. 21, 22, 25, 27, 48
- [14] Will J Schroeder, Bill Lorensen, and Ken Martin. *The visualization toolkit*. Kitware, 2004. 39
- [15] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996. 18
- [16] Laura Swanson. Linking maintenance strategies to performance. *International journal of production economics*, 70(3):237–244, 2001. 4

- [17] Tetsuji Takada and Hideki Koike. Mielog: A highly interactive visual log browser using information visualization and statistical analysis. In *LISA*, volume 2, pages 133–144, 2002. [17](#), [18](#), [24](#), [25](#), [26](#)
- [18] Han Van Venrooij. Visualizing asml twinscan event log history. Master’s thesis, Eindhoven : Technische Universiteit Eindhoven, 2014. [18](#), [25](#), [26](#), [28](#), [40](#), [48](#), [49](#)
- [19] Taowei David Wang, Catherine Plaisant, Alexander J Quinn, Roman Stanchak, Shawn Murphy, and Ben Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 457–466. ACM, 2008. [19](#), [20](#), [25](#), [26](#)
- [20] Taowei David Wang, Catherine Plaisant, Ben Shneiderman, Neil Spring, David Roseman, Greg Marchand, Vikramjit Mukherjee, and Mark Smith. Temporal summaries: supporting temporal categorical searching, aggregation and comparison. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1049–1056, 2009. [19](#)
- [21] Krist Wongsuphasawat, John Alexis Guerra Gómez, Catherine Plaisant, Taowei David Wang, Meirav Taieb-Maimon, and Ben Shneiderman. Lifeflow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1747–1756. ACM, 2011. [19](#), [20](#), [25](#), [26](#), [28](#)

Appendix A

Event Types

This chapter lists all the event types and other tables that are used in the prototype.

	Property Count	Total Records	AVG event/day	AVG Duration
B1shimcalc	8	1,454,745	7	n/a
Coil	9	612,549,693	2922	n/a
Condition	13	17,457,740	61	n/a
Conflict	12	3,970,058	5	n/a
Crash	9	5,865,245	4	n/a
Dailylog	15	1,900,241	n/a	n/a
Device	26	12,591,102	44	04:02:33
Event	7	1,194,642,225	640	n/a
Exam	25	22,964,259	14	00:33:11
Examtable	15	126,090,914	94	00:00:08
Hardware	9	6,582,333	58	n/a
Preparation	105	420,800,574	1916	00:00:02
Recon	22	232,551,365	156	00:02:00
Scan	166	201,719,856	139	00:01:55
Suspense	6	4,335,014	7	00:00:20
Uimsg	8	1,434,666,698	1063	n/a
Userinteraction	9	315,576,142	1413	00:00:24
Validate	8	1,025,116,753	755	00:00:01
raw_logstat_hw_errors	10	317,489,806	332	n/a
raw_gradient_errors	10	250,311	8	n/a
raw_pfei_errors	10	161,969	2	n/a
raw_rfamp_errors	14	238,702	2	n/a
Grand Total	516	5,958,975,745	9642	n/a

Table A.1: A list containing all the event types that are used in the prototype. For each event type the following statistics are shown: the number of properties, the total number of records in the database, the average number of occurrences per day per site and the average duration for interval events. The average amount of events for each day is somewhat lowered due to incomplete events logs. The events with *raw* prefix are not officially contained within the MR model, but are extracted using separate scripts. These events are however stored in the same database as the MR model events. For this reason we are able to use them alongside the MR model event.

Appendix B

Prototype Development

In this appendix, some implementation details and statistics on the developed prototype. First, the Java Software Platform that is mainly used for the development of the interface and visualizations is shortly described in Appendix [B.1](#). Second, Appendix [B.2](#) describes the main development and testing environment. A brief overview of the designed architecture is shown in Appendix [B.3](#). And finally, some metrics that give an indication of the complexity of the developed software are presented in Appendix [B.4](#).

B.1 JavaFX

As stated by Requirement C4, we implemented the prototype in Java. More specifically, we have decided to use the relative new user-interface Framework JavaFX. JavaFX is actively being developed by Oracle and is intended to replace Swing or AWT as the standard GUI library. JavaFX is part of the official JRE/JDK from Java 8, therefore no external library is required to make it run. We have mainly chosen to use this Interface library for the ease of development and its active development.

Unlike Swing, JavaFX uses a scene graph that takes care of managing and rendering the controls, which can be modified by applying transforms, effects and CSS styling. Furthermore, data binding is natively offered and simplify connecting different views together. In this project, we have used some controls that are offered by JavaFX and customized them to our needs. The most prominent customizations have been performed on the Calendars and the Chart controls which are used the Calendar View and the Timeline View respectively.

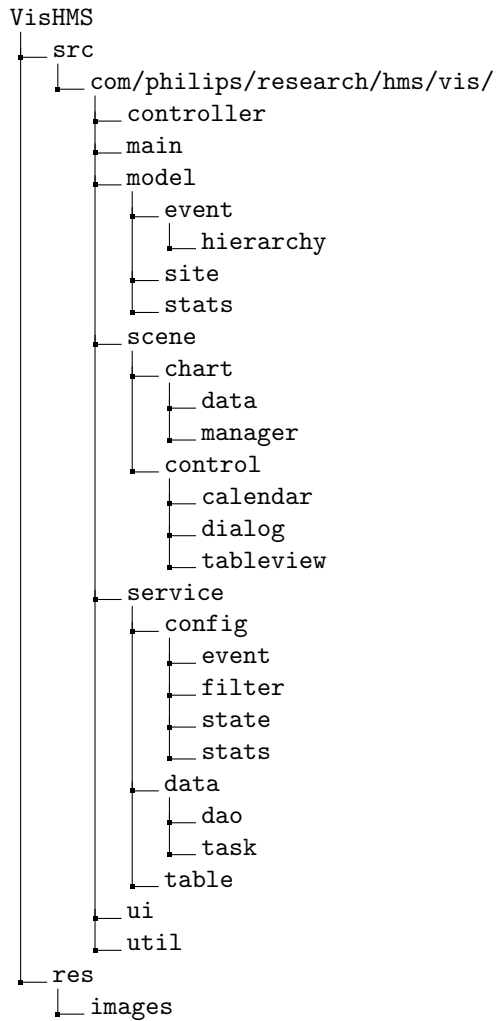
However, if it is decided to further expand on the prototype that has been built, it should be taken into consideration to rebuild some of the views due to performance reasons. JavaFX does give good performance in general, but if too many nodes (thousands) are added to the scene graph, and if interaction is required for these nodes, then the performance can quickly decline.

B.2 Environment

This section presents the environment in which the prototype was developed and tested. Table [B.1](#) shows the properties of the development and testing environment and Figure [B.1](#) shows the directory structure for the build and the available packages.

Table B.1: The development and testing environment

Java Verion	Java 1.8.0_31, 32bit
Libraries	Vertica-jdbc-7.1.1-0 for connecting with the database and perform queries
IDE	Eclipse with e(fixture)clipse plugin
Build System	Ant which compiles the code and generates two builds. 1. An executable jar with webstart launcher. 2. A stand-alone Windows build.
OS	Windows 7 64-bit

**Figure B.1:** The directory structure of the packages.

B.3 Architecture

In this appendix the architectural structure of the implemented prototype is shown. Figure B.2 shows the package structure and Figures B.3 to B.7 show the classes contained within the packages as well as the dependencies and other connections between classes.

When the application launches, the user interface is constructed and each View is assigned a controller. The controllers control the interface elements, manage the data that stored shown in a view and manages the interaction performed by the user. The several controllers are linked together by using data binding. Therefore, there is no direct dependency between most of the controller classes. The interface uses custom nodes that have been defined in the Scene package. The most important control node is the RestorableTitlePane. This node allows views to be packaged into a collapsible layout which can be used to hide a view and create space for other views. The most important node in the Chart package is the TimeLineChart. The TimeLineChart is a customized JavaFX XYChart that is able to manage and render multiple sets of events. The classes contained in the Manager sub-package enable the interactive features that are supported by the Timeline View.

Next, the model package describes the models in which the data is stored. The Event sub-package contains the generalized event structure. Furthermore, there are models for storing sites and site details, and models for storing the distribution of the several event types. The Service package contains all the classes that are responsible for retrieving the data, managing configurations and describing the available events. First, a set of services is defined in the data package, each of which is responsible for retrieving different kinds of data. Each service uses the corresponding task to actually execute the data retrieval in parallel. The Table package contains the classes that are used to describe how the event types are defined in the database. The Config package is responsible for describing and managing the event configurations, filters and system states. Finally, the Util package contains some utilization classes that are useful for visualizing or managing some data.

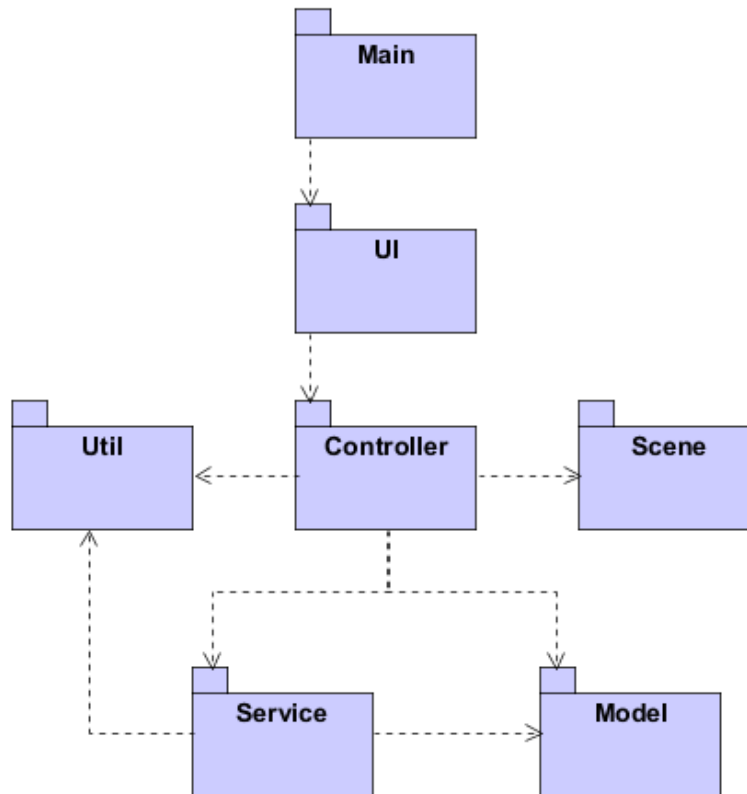


Figure B.2: The package structure of the architecture

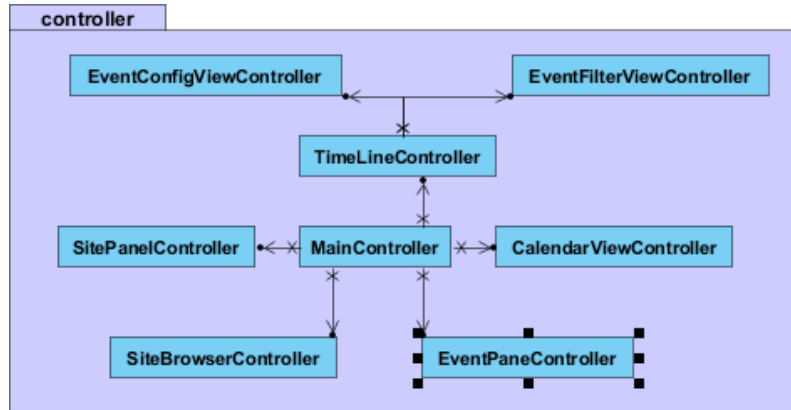


Figure B.3: Class overview of the controller package.

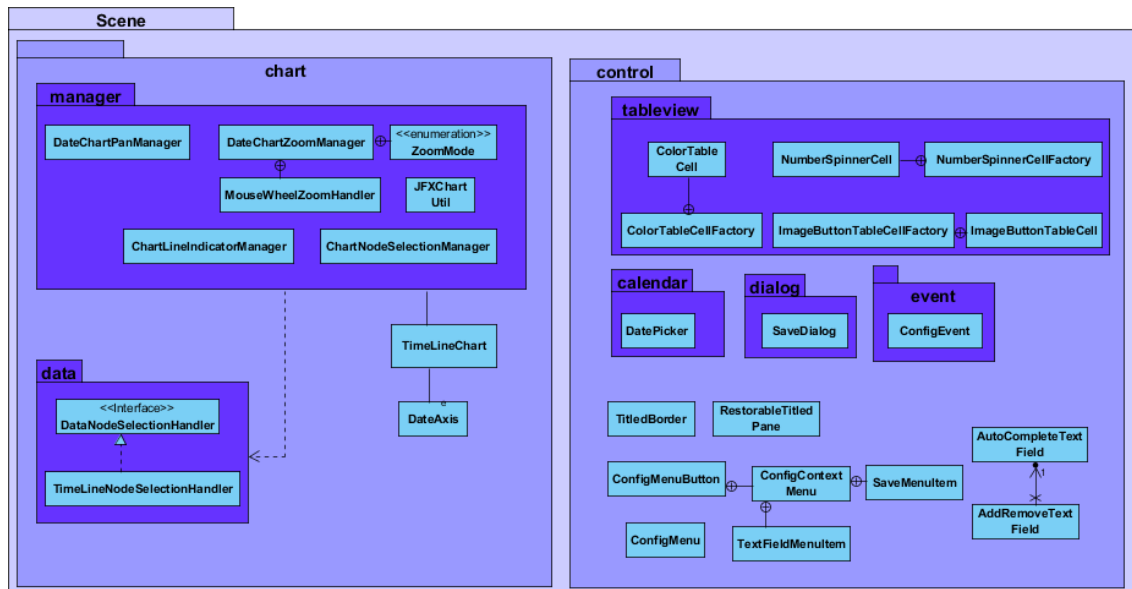


Figure B.4: Class overview of the Scene package.

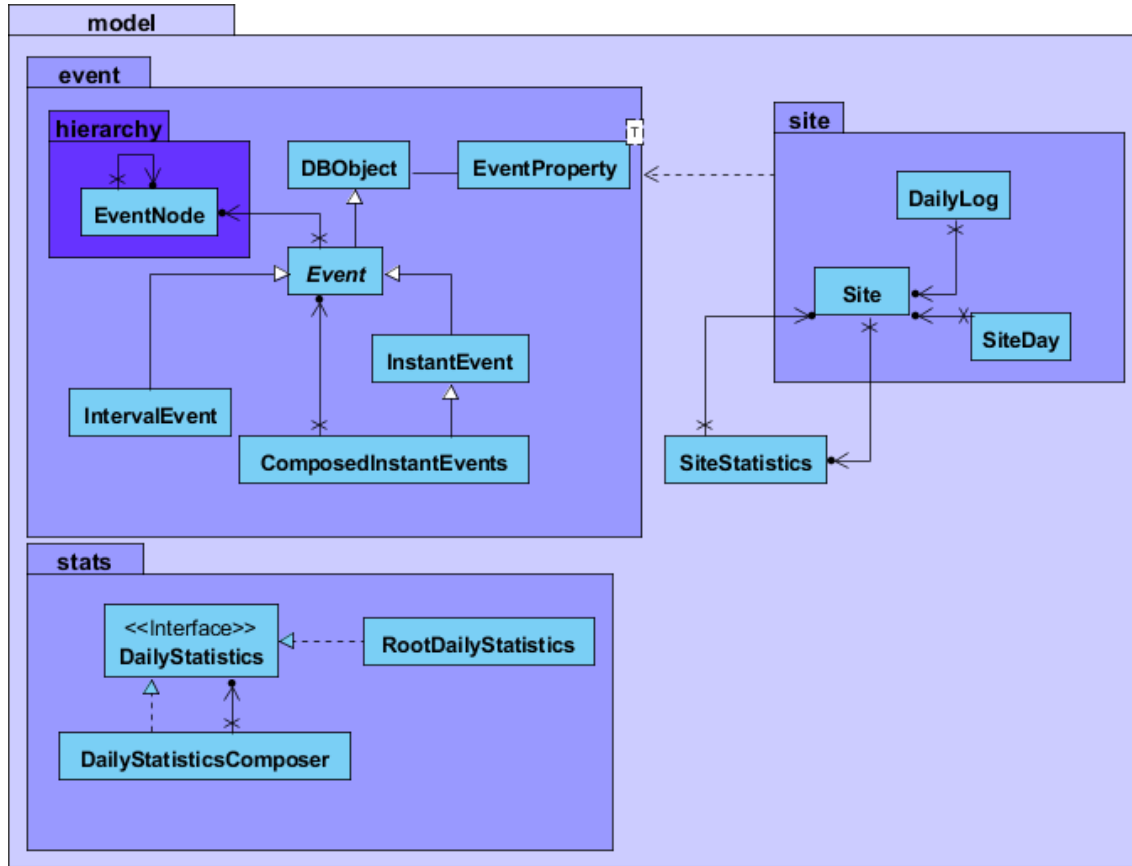


Figure B.5: Class overview of the model package.

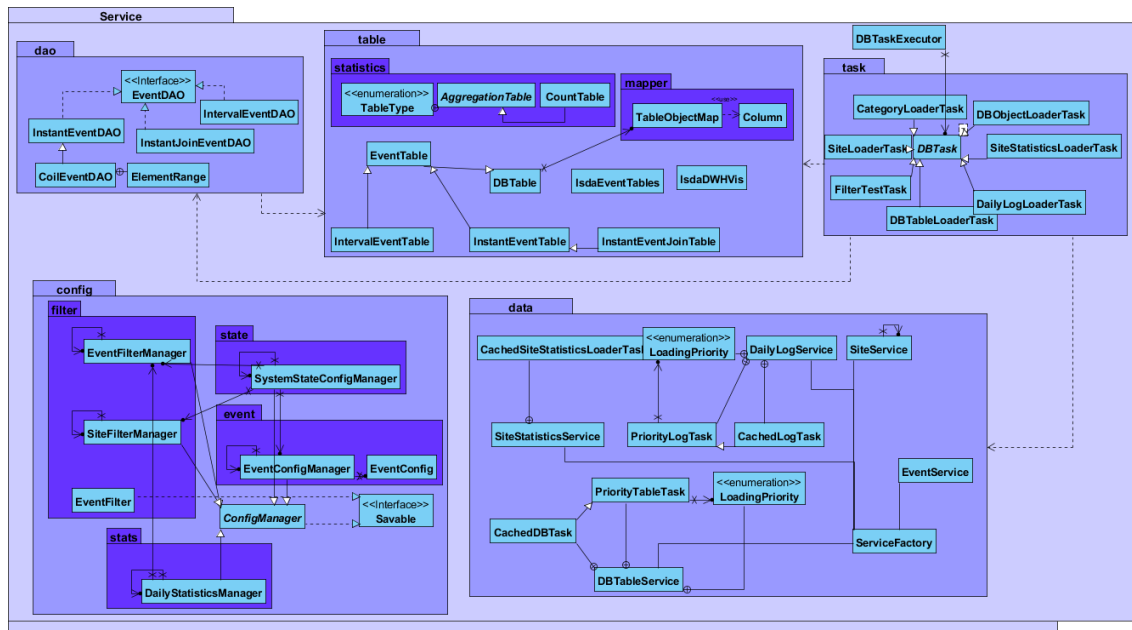


Figure B.6: Class overview of the service package.

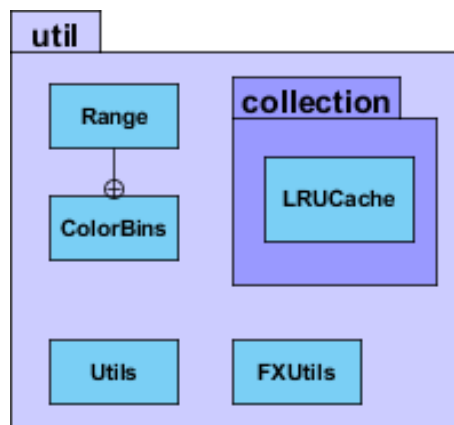


Figure B.7: Class overview of the Util package.

B.4 Metrics

For each package of the developed software, some basic code metrics is shown in Table B.2. The metrics give some indication of the complexity and maintainability of the implemented prototype. Classes with many lines of code, many methods or a large Cyclomatic Complexity are assumed to be complex and difficult to maintained. The number of lines of code do not include any comments or java-doc that are used to describe and reasoning behind the code. The Cyclomatic Complexity measures the number of paths that can be taken through a method. The score of the metric is largely influenced by the number of branching statements like *if*, *for* and *while* statements. Methods that have a Cyclomatic Complexity greater than ten are considered to be highly complex.

		Lines Of Code	Number of Classes	Number of Methods	AVG Cyclomatic Complexity
Controller		2332	20	163	1.73
Main		49	1	2	1.33
Model	Event	294	7	48	1.80
	Site	226	3	51	1.16
	Stats	74	4	12	1.25
		799	16	152	1.35
Scene	Chart	2351	27	186	2.34
	Control	2085	38	188	1.61
		4436	65	374	1.97
Service	Config	1309	15	148	1.73
	DAO	516	6	26	3.27
	Data	517	13	61	1.77
	Table	1196	12	70	1.69
	Task	465	8	22	3.5
		4358	54	327	1.97
Util		258	9	31	1.66
All		12232	165	906	2.13

Table B.2: Some code metrics per package. Some of the reside in the Model, Scene and Service, therefore the metrics of the main packages do not directly follow from the resulting metrics of the sub-packages.