

RAD Enhanced Reporting Tool

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2017

Autor: Zafer Dogan
Betreuer: Thomas Corbat, Felix Morgner
Projektpartner: Rheinmetall Air Defence AG, Zürich Oerlikon

Inhaltsverzeichnis

1 Aufgabenstellung	6
1.1 Einführung	6
1.2 Betreuer	6
1.3 Ziele der Arbeit	6
2 Abstract	8
3 Management Summary	9
3.1 Ausgangslage	9
3.2 Vorgehen und Technologie	10
3.2.1 Entwicklungsetappen	10
3.2.2 Verwendete Technologien.....	10
3.3 Ergebnisse	11
3.3.1 Featureübersicht	11
3.3.2 UI Design.....	12
3.4 Ausblick	14
3.4.1 Offene Features	14
3.4.2 Optimierungsmöglichkeiten.....	14
4 Technischer Bericht	15
4.1 Ausgangslage	15
4.2 Analyse	17
4.2.1 Anforderungsspezifikation	17
4.2.2 Domainanalyse.....	20
4.2.3 Ablauf Verbindungsaufbau von Clients	22
4.2.4 RApplication/Client Konzept.....	23
4.3 Architektur	25
4.3.1 Logische Abhängigkeiten	25
4.3.2 Klassen Abhängigkeiten	26
4.3.3 Window	27
4.3.4 Domain	31
4.3.5 Modell	36
4.3.6 Worker	38
4.3.7 Common.....	41
4.3.8 CustomWidgets.....	44
4.4 Design.....	45
4.4.1 MVC	45

4.4.2 Leistung und Multithreading	47
4.4.3 Logfile Parsing.....	53
4.4.4 Persistenz	54
4.4.5 GUI.....	56
4.5 Diskussion	61
4.5.1 Leistung	61
4.5.2 UI Design.....	61
4.6 Ausblick	62
5 Glossar	63
6 Anhang A	64
6.1 Formulare	64
6.1.1 Eigenständigkeitserklärung	64
6.1.2 Nutzungsrechte	65
6.2 Persönlicher Bericht.....	66
6.3 Projektplan	67
6.3.1 Projekt Übersicht.....	67
6.3.2 Projektorganisation	68
6.3.3 Management Abläufe	69
6.3.4 Phasen / Iterationen	70
6.3.5 Meilensteine.....	71
6.3.6 Meetings	72
6.3.7 Risikomanagement.....	73
6.3.8 Arbeitspakete.....	75
6.3.9 Infrastruktur.....	75
6.3.10 Qualitätsmassnahmen	76
6.3.11 Testen.....	77
6.4 Zeitabrechnung.....	78
6.6 Sprint Reports.....	80
6.6.1 RRT Sprint 1 - Elaboration 1	80
6.6.2 RRT Sprint 2 - Elaboration 2.....	81
6.6.3 RRT Sprint 3 - Construction 1.....	82
6.6.4 RRT Sprint 4 - Construction 2.....	83
6.6.5 RRT Sprint 5 - Construction 3.....	84
6.6.6 RRT Sprint 6 - Construction 4.....	85
6.7 Sitzungsprotokolle HSR	87

6.7.1 Anwesende (üblicherweise)	87
6.7.2 Protokollführer	87
6.7.3 Besprechung Projekt KW1	87
6.7.4 Besprechung Projekt KW2	88
6.7.5 Besprechung Projekt KW3	89
6.7.6 Besprechung Projekt KW4	90
6.7.7 Besprechung Projekt KW5	92
6.7.8 Besprechung Projekt KW6	92
6.7.9 Besprechung Projekt KW7	93
6.7.10 Besprechung Projekt KW9	94
6.7.11 Besprechung Projekt KW10	95
6.7.12 Besprechung Projekt KW11	95
6.7.13 Besprechung Projekt KW12	96
6.7.14 Besprechung Projekt KW13	97
6.7.15 Besprechung Projekt KW14	99
6.8 Sitzungsprotokolle RAD	101
6.8.1 Datum: 09.03.2017 / 09:00	101
6.8.2 Datum: 23.03.2017 / 09:00	102
6.8.3 Datum: 05.04.2017 / 09:00	104
7 Anhang B	105
7.1 Entwickleranleitung	105
7.1.1 Visual Studio and Qt Setup	105
7.1.2 Qt Testing Setup	106
7.2 Qualitätsmanagement	111
7.2.1 Code Reviews	111
7.2.2 Code Kommentare	111
7.2.3 Code Analyse	112
7.2.4 Testing	113
7.2.5 Unit Tests	113
7.2.6 Integration Tests	116
7.2.7 GUI Tests	116
7.2.8 GitFlow	118
7.2.9 Continuous Integration	118
7.2.10 Usability Tests	118
7.2.11 Protokolle	118

7.3 Usability Tests	119
7.4 Testlogs	120
7.4.1 Alphaversion - 10/4/2017	120
7.4.2 Betaversion - 8/5/2017	124
7.4.3 Releaseversion - 23/5/2017	128
7.4.4 Abgabeverversion - 1/6/2017	132
7.5 Use Cases (Essential Style)	136
7.5.1 UC 1 Logging.....	136
7.5.2 UC5 Logging suchen und filtern	137
8 Anhang C	138
8.1 Installationsanleitung	138
8.2 Benutzerhandbuch.....	139
8.2.1 Shortcuts	139
8.2.2 Client aktivieren und im Tab anzeigen	140
8.2.3 Client in den Fenstermodus wechseln lassen	140
8.2.4 Client Logs routen.....	141
8.2.5 Filter auf Clienttabelle anwenden	142
8.2.6 Globale Log Einstellungen vornehmen	143
8.2.7 View Elemente ein- und ausblenden	143
8.2.8 Client Logs exportieren	143
8.2.9 Client Logs importieren	143
8.2.10 Applikationseinstellungen ändern und persistieren	144
8.2.11 Applikationseinstellungen laden	144
8.2.12 API Dokumentation einsehen	144

1 Aufgabenstellung

1.1 Einführung

Das Projekt ReportingTool hat zum Ziel die bestehende Monitoring-Lösung von Rheinmetall Air Defence AG (RAD), welche zur Überwachung des Testbetriebs ihrer Systeme eingesetzt wird, neu zu entwickeln. Eine Library für den Empfang der Daten wird von RAD zur Verfügung gestellt und der Fokus der Arbeit liegt in der Realisierung einer benutzerfreundlichen grafischen Schnittstelle zur Visualisierung der Log-Daten

1.2 Betreuer

Diese Studienarbeit wird für die Rheinmetall Air Defence AG (RAD) durchgeführt und wird von Thomas Corbat, HSR, IFS, thomas.corbat@hsr.ch betreut.

1.3 Ziele der Arbeit

Das Ziel der Arbeit ist die Entwicklung einer plattformunabhängigen GUI-Applikation zur Überwachung des Dauertestbetriebs (24h/7Tage) der Systeme von RAD.

Die grosse Menge an Daten, welche übers Netzwerk empfangen werden, müssen dabei effizient verarbeitet und gefiltert werden. Es soll zudem ein sparsamer 24/7-Betrieb gewährleistet werden können, welcher die anderen Applikationen auf dem Rechner nicht beeinflusst. Die Informationen sollen dabei in einem benutzerfreundlichen GUI dargestellt werden, welches den Entwickler bei seiner Arbeit unterstützt.

Massgebend für das Design und die Realisierung der Applikation sind die Anforderungen von RAD. Zusätzliche Funktionen können je nach Zeitbudget und Vorankommen umgesetzt werden.

Erwartete Features:

- Loggen der Daten in verschiedenen Fenstern oder Dateien
- Filtern der Daten nach verschiedenen Kriterien wie zum Beispiel Gruppen/Clients oder Schweregrad
- Speichern der angezeigten Daten in Dateien
- Weitere Anforderungen sind mit RAD abzuklären

Es finden wöchentliche Besprechungen mit dem Betreuer statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen.

Alle Besprechungen, ausser der Kick-off Besprechung, sind von den Studierenden mit einer Traktandenliste vorzubereiten und zu leiten. Als erstes Traktandum soll immer der Stand des Projektes präsentiert werden (Was wurde gemacht? Was wurde erreicht? Wie viel Zeit wurde in was investiert?). Die Beschlüsse der Besprechungen sind durch die Studierenden zu protokollieren und an den Betreuer anschliessend zuzustellen.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen (oder auch Zwischenversionen) gemäss Projektplan sind einzelne Arbeitsergebnisse in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsergebnisse erhalten die Studierenden ein

vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Resultate.

Technische Einschränkungen des Industriepartners sind zu beachten. Für die Realisierung der grafischen Oberfläche ist das GUI-Framework Qt 5.6 zu verwenden. Für die Entwicklung der Applikation ist die Programmiersprache C++ zu verwenden. Die resultierende Applikation muss für Linux und Windows kompiliert werden können. Die Entwicklung sollte in Visual Studio 2015 erfolgen.

Die Clientseite kann nicht verändert werden, sprich die Daten sind so zu verarbeiten wie sie über die vorgegebene Library empfangen werden. Ein entsprechendes Testprogramm wird von RAD zur Verfügung gestellt werden.



Thomas Corbat



Zafer Dogan

2 Abstract

Currently, applications within the Rheinmetall Air Defence corporation use a reporting library for logging via network. Applications communicate their logs to a central workstation, where logs are viewed and analysed. The core target of the new reporting tool developed during the term project can be summarized as porting the old reporting tool to a modern, user-friendly application, with new features like tabbed views and filtering capabilities. By requirement of the industry partner, Qt 5.6 was used as the GUI framework. The programming language was defined as C++11.

The project was realized in two major phases: first, reimplementing all features of the old reporting tool and second, extending the application with the desired new features. During the first phase, the aspect of performance and memory management received special focus, as by requirement the tool needed to be capable of processing large amounts of logs, arriving in short intervals and remain stable and responsive 24/7. The capabilities of the Qt GUI framework in regards to those two aspects were researched thoroughly and the solutions designed accordingly.

The new reporting application offers a modern, approachable tool, making previously cumbersome tasks easier. It joins log handling under one parent application and offers more control in log analysis. Before, in order to initialize analysing, two separate projects needed to be started, one, the listening server application, two, the old reporting tool application, which served more as a configurations manipulator. Furthermore, logs were displayed in simple windows without any capabilities of searching or filtering. Limitations like these made log analysis in the previous reporting tool cumbersome and confusing. Now, the two separate initial steps have been integrated into one and thus made initializing log analysis more convenient. As with the old reporting tool, the new application supports external configuration for quick start-up or workspace arrangement. Gathering the parts under one roof application provides more oversight and makes log analysis more comfortable.

3 Management Summary

3.1 Ausgangslage

Das zurzeit bei der Rheinmetall Air Defence AG in Oerlikon ZH eingesetzte Reporting Tool fungiert als ein Werkzeug zur Manipulation der Client - Konfigurationen, sprich was sollen die Clients, die das Reporting Tool benutzen, loggen, wohin sollen sie es loggen und ob das Geloggte in die Gruppenansicht oder in das "Global" - Output umgeleitet werden sollen, oder nicht. Ausserdem bietet das alte Tool zusätzlich weitere Konfigurationsmöglichkeiten, wie z.B. das Umleiten der Logs direkt in eine Datei oder welche Arten von Logs Klienten überhaupt loggen sollen. Gemäss diesen Konfigurationen werden dann auf dem "Server" - PC, also dem Gerät, auf dem die Loganalyse per Reporting Tool ausgeführt wird, je nach Konfiguration Logfenster geöffnet und die entsprechenden Log-Informationen in diese Fenster umgeleitet (oder eben in eine Datei, falls so per Reporting Tool eingestellt). Zur Verwendung dieses Tools muss zusätzlich noch ein "Listener" - Programm gestartet werden, das sogenannte "Master", welches dann auf einem bestimmten Port auf die Lognachrichten hört und diese an die entsprechenden Fenster weiterleitet. Die Logfenster boten im alten Tool keinerlei Funktionen zum Filter der Informationen an, man musste also die Zeilen einzeln durchgehen und die Analyse per Observation der einzelnen Einträge durchführen, was bei einer grossen Anzahl an Logs schnell einmal unübersichtlich werden kann.

Die Konfigurationen werden im alten Reporting Tool anhand drei verschiedenen Konfigurationsdateien realisiert:

1. Clients - Konfiguration
Fasst alle vorhandenen Clients und deren Gruppen zusammen. Anhand dieser Datei weiss das Reporting Tool, welche Clients und Gruppen vorhanden sind.
2. Control - Konfiguration
Fasst alle Klient basierten Einstellung zusammen, sprich welche Klienten sind aktiv, welche sollen in eine Datei umleiten, ob z.B. Logs vom Typ "Info" geloggt werden sollen, oder nicht etc.
3. Config - Konfiguration
Fasst alle allgemeinen Informationen wie z.B. was das Log Buffer Limit ist, wo die Logdateien platziert werden, falls man diese in eine Datei umleitet etc.

Diese werden zwischen den entsprechenden Client - PCs und dem "Server"-PC, also dem Gerät, auf dem das Reporting Tool läuft, via Netzwerk in einem aktualisierten Zustand ausgetauscht. Die aktualisierten Konfigurationsdateien werden dann entsprechend gelesen und das Logverhalten angepasst. Entsprechend erfährt auch das Reporting Tool auf diese Weise, ob es nun einen weiteren Client gibt oder nicht.

Um die ganze Tätigkeit der Loganalyse zu vereinfachen, wurde im Rahmen dieser Studienarbeit ein neues Tool beantragt, das neu alle Loganalysetätigkeiten in einer zusammengefassten und modernen GUI Applikation ermöglichen soll. Als mögliche Inspirationsquelle wurde das Überwachungstool für Netzwerktätigkeiten Wireshark vorgeschlagen. So soll das neue Tool die Loganalyse vereinfachen und im allgemeinen die Analyse der Logs angenehmer gestalten. Ausserdem soll im neuen Tool auf den Austausch von Konfigurationsdateien ganz verzichtet werden und das ganze Logging direkt per TCP - Verbindung realisiert werden. Hierfür wurde die zugrundeliegende Architektur schon gelegt, so dass das neue Tool nur noch die entsprechend Schnittstellen verwenden soll, um die Logdateien aufzugreifen und die Einstellungen per TCP - Protokoll direkt zu übertragen.

Das neue Reporting Tool soll die bestehenden Funktionen des alten Tools auf die gleiche Art und Weise anbieten und einige weitere Features wie Logfilterung und Tab-Ansicht anbieten.

3.2 Vorgehen und Technologie

Vom Industriepartner wurde für die Entwicklung des neuen Reporting Tool das Qt GUI Framework, Version 5.6, ausgewählt. Das Qt GUI Framework ist ein C++ GUI Framework, das die plattformübergreifende Entwicklung von Applikationen in C++ erlaubt. So kann die entwickelte Applikation auf verschiedenen Betriebssystem einfach verfügbar gemacht werden.

3.2.1 Entwicklungsetappen

Die Umsetzung der Applikation wurde in zwei Etappen umgesetzt. In der ersten Etappe lag der Fokus auf der Wiederherstellung der Funktionalität des alten Tools. Hier zeichneten sich besonders die Aspekte Leistung und Speicherverwaltung als Herausforderung aus. In der zweiten Etappe wurden dann die zusätzlichen Features implementiert, wie z.B. das Filtern von Logs oder die Ansicht in Tab-Ansicht.

3.2.2 Verwendete Technologien

Die Haupttechnologie die zur Umsetzung des neuen GUI Tools verwendet wurde, ist das Qt GUI Framework. Nebst dem Qt Framework kamen keine weiteren speziellen Technologien zum Einsatz.

3.3 Ergebnisse

Das Tool LogShark wurde erfolgreich realisiert und bietet eine einheitliche Übersicht aller Clients und Groups. Zudem erlaubt das neue Tool das Filtern von Logs, das Laden von verschiedenen Konfigurationsdateien, mit denen die Applikation je nach Bedürfnis konfiguriert werden kann und bietet einen fließenden Übergang zwischen Tabs und Fenster - Ansichten. Ausserdem bietet das Tool die Möglichkeit verschiedene Clients in einer MergeView - Gruppe zusammengefasst anzusehen und Logs mehrere Clients in dieselbe Datei zu routen. Die resultierende Applikation sieht in Aktion in etwa wie in Bild 1 aus.

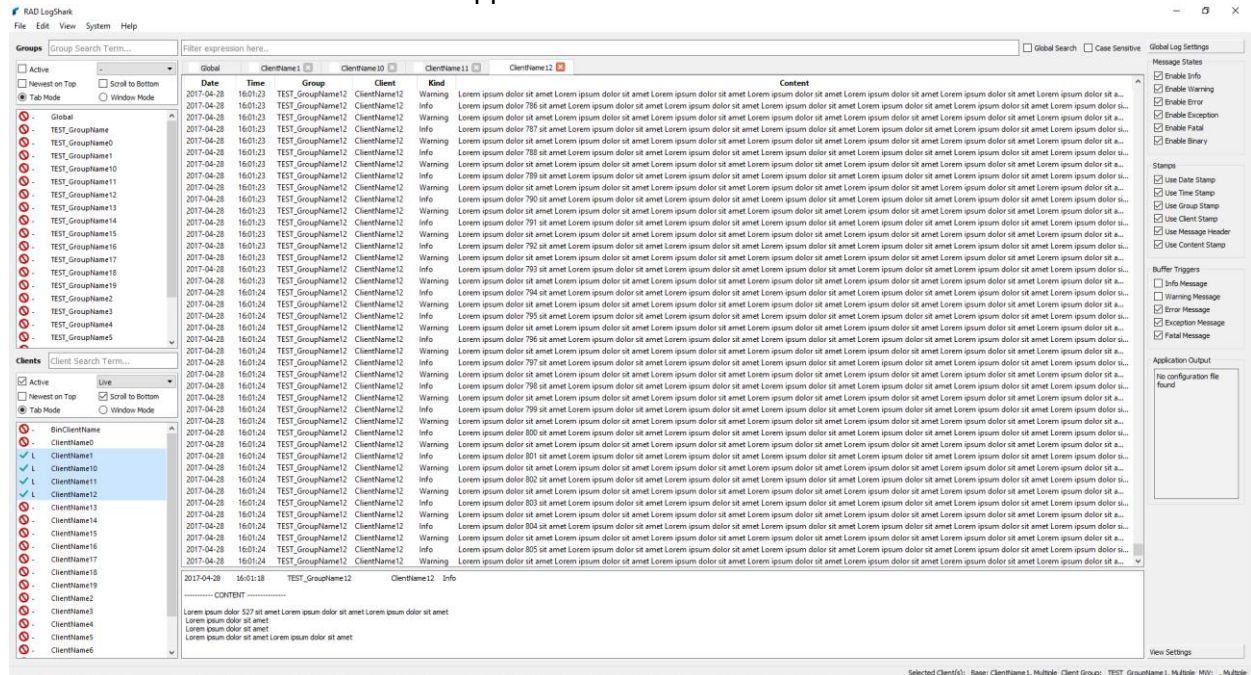


Bild 1: Hauptfenster RAD LogShark

Wobei man erkennen kann, das vier Clients aktiv sind und im Tab Modus loggen.

Im Folgenden sind die Applikationsergebnisse aufgeführt. Unter anderem, welche Features implementiert wurden und wie das UI designt wurde.

3.3.1 Featureübersicht

In der unten aufgeführten Tabelle ist eine Übersicht der neu implementierten Features für das neue Tool aufgelistet.

Feature	Beschreibung
Logfilterung	Logs können per Regexpfilter auf die Kliententabellen angewendet werden und somit die Anzahl angezeigter Logs so eingrenzen, so dass nur Logs angezeigt werden, die dem Filterausdruck entsprechen
Tab - Ansicht Logs	Die Logs können nebst in Fenster auch als Tabs im Hauptfenster angesehen werden
Tabmode / Windowmode	Die Klienten können zwischen den zwei Modi hin- und hergeschaltet werden

Drag'n'Dropable Tabs / Windows	Tab-Views können per Drag und Drop zu Fenstern umgeschaltet werden und vice versa.
Applikationskonfiguration	Das Tool kann mit einer Konfigurationsdatei gestartet werden, Applikationskonfigurationen können im Nachhinein geladen und angewendet werden und Applikationskonfigurationen können exportiert werden. Eine Applikationskonfiguration kann Informationen über Klienten speichern und Einstellungsinformationen zu bestimmten Voreinstellungen enthalten, sprich ob z.B. Logfilterbereich angezeigt werden soll oder nicht.
Merge Views	Es können MergeViews für mehrere Klienten erstellt werden, die dann als Gruppenklienten fungieren und die Logs der Klienten zusammenfassen.
To Shared File	Mehrere Klienten können Ihre Logs in die gleiche Datei umleiten.
Tabelleneinfärbung	Für die einfache Auseinanderhaltung von Kliententabellen können diese eingefärbt werden.

Tabelle 1: Featureübersicht

Nebst diesen Features sind auch alle Features aus dem alten Tool umgesetzt, wie z.B. welche Art von Logs die Klienten loggen sollen.

3.3.2 UI Design

Das UI wurde mit der Applikation Wireshark im Hinterkopf entwickelt. So hat man, wie in Bild 2 ersichtlich, z.B. oben einen Bereich zur Eingabe des Filterausdrucks, in der Mitte die Logs und unten die Logdetailanzeige.

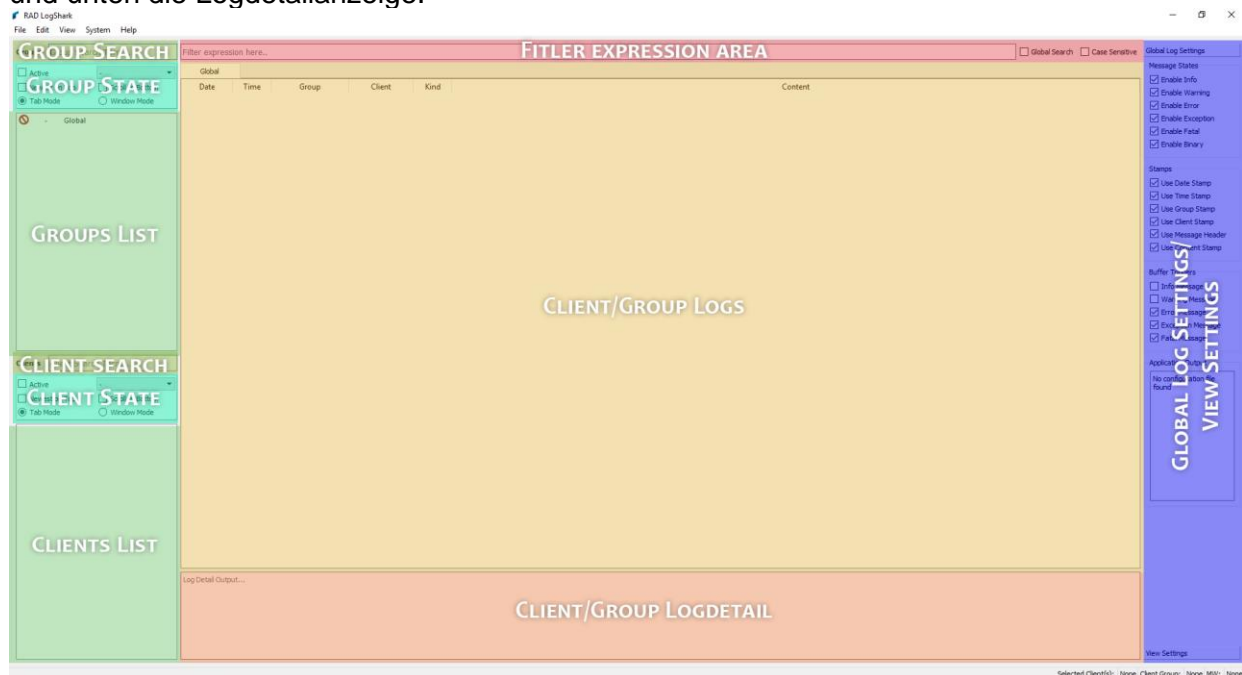


Bild 2: Layout Hauptfenster

Das Ziel dieser Annäherung ist es für den Benutzer der Applikation so viel Übersicht wie möglich zu gewähren. Die wichtigsten Einstellungen zur Loganalyse sind direkt ersichtlich und änderbar. Will man bestimmte Bereiche des UI ausblenden, so ist diese Funktion unter View Settings direkt verfügbar. Die Logfenster wurden in einem ähnlichen Layout designt, jedoch jeweils auf einen Klienten massgeschneidert. In einem Logfenster sind die Clients/Groups - Listen nicht vorhanden. Ausserdem ist der Filterausdruck in einem Logfenster nur auf den aktuellen Klienten anwendbar. Im Hauptfenster hingegen hat man die Optionen den Filterausdruck auf alle Klienten anzuwenden. Ein Logfenster sieht wie in Bild 3 aus.

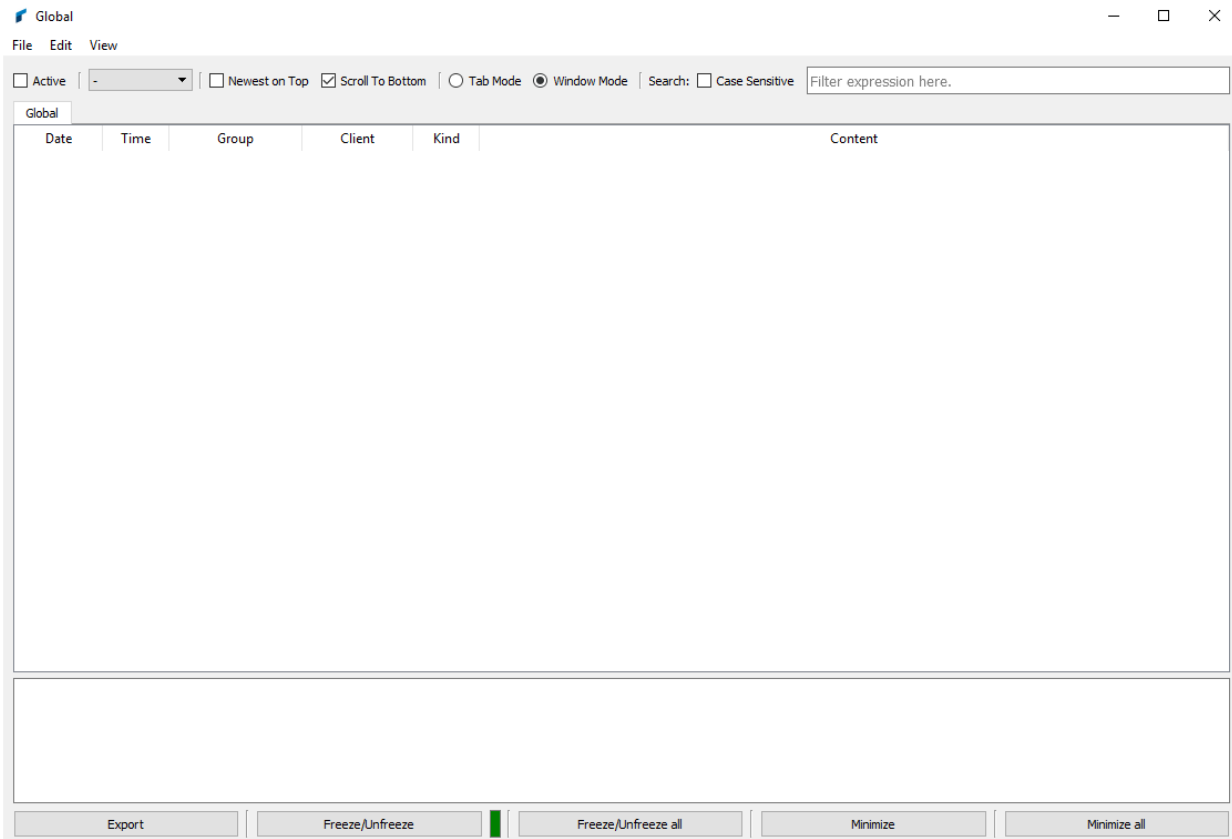


Bild 3: Layout Logfenster

3.4 Ausblick

Leider konnten nicht alle Features wie gewünscht umgesetzt werden, vor allem einige optionale Features, die interessante Ansätze bildeten, wurden aus Zeitmangel ausgelassen. Doch für die Weiterentwicklung der Applikationen sollen hier noch offene und interessante Features festgehalten werden und Optimierungsmöglichkeiten der bestehenden Applikation Erwähnung finden.

3.4.1 Offene Features

Parallel Scrolling	Paralleles scrollen von mehreren Logfenstern zur gleichzeitigen Analyse von Logs zu einer bestimmten Zeit
Tabs to MergeView	Direktes Zusammenführen von Tabs zu einem Tab
Windows to MergeView	Direktes Zusammenführen von Client Fenstern zu einem Fenster
Layouts	Einstellbare Layouts
Klienten basierte Vieweinstellungen	Einstellungen wie sichtbare Spalten, welche Logtypen ein Client loggen soll.
Filterausdruck Dialog	Ein Dialog zum Erstellen von Filterausdrücken ohne spezielle Filterregexausdrücke schreiben zu müssen

3.4.2 Optimierungsmöglichkeiten

Es bestehen viele Möglichkeiten zur Optimierung. Zum Beispiel das Zusammenführen von mehreren Clients in eine MergeView. Man kann zwar eine MergeView definieren und die Client Logs zu dieser umleiten, doch besteht momentan keine "Eigenschaften" - Ansicht auf den Clients, so dass man auf Client-Basis nachschauen könnte, zu welcher MergeView dieser umleitet. Zwar werden die Clients einer MergeView angezeigt, aber dies benötigt einen zusätzlichen Schritt und ist nicht unbedingt benutzerfreundlich.

Dasselbe Problem besteht zurzeit bei den "To Shared File" Dateiausgabe; nachdem man den Namen der "geteilten Datei" definiert hat, kann man diesen nicht nachträglich anzeigen lassen; hat man sich den Namen nicht gemerkt, müsste man, im Falle dass man andere Klienten zu dieser geteilten Datei umleiten möchte, im Ausgabedossier "/logs" nachschauen.

Hierbei könnte ein Kontextmenü, dass auf die Clients/Groups - Listen definiert ist, Aushilfe schaffen. Dadurch könnte man mehr Kontrolle über die Clients und Group Clients erhalten und die Loganalyse würde ein bisschen weiter vereinfacht werden.

Eine andere Optimierungsmöglichkeit bestünde im Einstellungen-Fenster. Zurzeit werden die Clients/Groups - Zustände aus den aktiven Einstellungen gelesen und gemäss diesen gespeichert. Es wäre besser, wenn im Einstellung - Fenster die Clients und Group Clients in einer Duallist zuweisbar wären, inkl. Zustandseinstellungen, ohne dass diese das Logging direkt beeinflussen würden, sprich setzte man einen Klienten auf aktiv im Einstellungen - Fenster, würde dieser nicht beginnen, zu loggen.

4 Technischer Bericht

Der technische Bericht bietet einen detaillierten Einblick in den Aufbau der Applikationen und die Entscheide während der Konstruktion. Es wird einleitend mit der Ausgangslage begonnen, mit der Architekturübersicht weitergefahren und zum Schluss werden noch die Designentscheide erörtert.

4.1 Ausgangslage

Das zurzeit bei der Rheinmetall Air Defence AG in Oerlikon ZH eingesetzte Reporting Tool fungiert als ein Werkzeug zur Manipulation der Klient - Konfigurationen, sprich was sollen die Clients, die das Reporting Tool benutzen, loggen, wohin sollen Sie es loggen und ob das Geloggte in die Gruppenansicht oder in das "Global" - Output umgeleitet werden soll oder nicht. Ausserdem bietet das alte Tool zusätzlich weitere Konfigurationsmöglichkeiten, wie z.B. das Umleiten der Logs direkt in eine Datei oder welche Arten von Logs Klienten überhaupt loggen sollen. Gemäss diesen Konfigurationen werden dann auf dem "Server" - PC, also dem Gerät, auf dem die Loganalyse per Reporting Tool ausgeführt wird, je nach Konfiguration Logfenster geöffnet und die entsprechenden Log-Informationen in diese Fenster umgeleitet (oder eben in eine Datei, falls so per Reporting Tool eingestellt). Zur Verwendung dieses Tools muss zusätzlich noch ein "Listener" - Programm gestartet werden, dass sogenannte "Master", welches dann auf einem bestimmten Port auf die Lognachrichten hört und diese an die entsprechenden Fenster weiterleitet. Die Logfenster boten im alten Tool keinerlei Funktionen zum Filter der Informationen an, man musste also die Zeilen einzeln durchgehen und die Analyse per Observation der einzelnen Einträge durchführen, was bei einer grossen Anzahl an Logs schnell einmal unübersichtlich werden kann.

Die Konfigurationen werden im alten Reporting Tool anhand drei verschiedenen Konfigurationsdateien realisiert:

1. Clients - Konfiguration
Fasst alle vorhandenen Clients und deren Gruppen zusammen. Anhand dieser Datei weiss das Reporting Tool, welche Clients und Gruppen vorhanden sind.
2. Control - Konfiguration
Fasst alle Client basierten Einstellung zusammen, sprich welche Clients sind aktiv, welche sollen in eine Datei umleiten, ob z.B. Logs vom Typ "Info" geloggt werden sollen, oder nicht etc.
3. Config - Konfiguration
Fasst alle allgemeinen Informationen wie z.B. was das Log Buffer Limit ist, wo die Logdateien platziert werden, falls man diese in eine Datei umleitet etc.

Diese werden zwischen den entsprechenden Client - PC's und dem "Server"-PC, also dem Gerät, auf dem das Reporting Tool läuft, via Netzwerk in einem aktualisierten Zustand ausgetauscht. Die aktualisierten Konfigurationsdateien werden dann entsprechend gelesen und das Logverhalten angepasst. Entsprechend erfährt auch das Reporting Tool auf diese Weise, ob es nun einen weiteren Client gibt oder nicht.

Um die ganze Tätigkeit der Loganalyse zur Vereinfachen, wurde im Rahmen dieser Studienarbeit ein neues Tool beantragt, der neu alle Loganalysetätigkeiten in einem zusammengefassten und modernen GUI Applikation ermöglichen soll. Als mögliche Inspirationsquelle wurde das Überwachungstool für Netzwerktätigkeiten Wireshark vorgeschlagen. So soll das neue Tool die Loganalyse vereinfachen und im allgemeinen die Analyse der Logs angenehmer gestalten. Ausserdem soll im neuen Tool auf den Austausch von Konfigurationsdateien ganz verzichtet werden und das ganze Logging direkt per TCP - Verbindung realisiert werden. Hierfür wurde die zugrunde liegende Architektur schon gelegt, so dass das neue Tool nur noch die entsprechend Schnittstellen verwenden soll, um die Logdateien aufzugreifen und die Einstellungen per TCP - Protokoll direkt zu übertragen.

Das neue Reporting Tool soll die bestehenden Funktionen des alten Tools auf die gleiche Art und Weise anbieten und einige weitere Features wie Logfilterung und Tab-Ansicht anbieten.

4.2 Analyse

In diesem Abschnitt wird die Analyse zum Projekt aufgelistet.

4.2.1 Anforderungsspezifikation

Im Folgenden werden die Anforderungsspezifikationen für das Reporting Tool Projekt aufgeführt. Im zweiten Meeting mit dem Industriepartner wurden die Anforderungen diskutiert und die Use Cases daraus abgeleitet.

4.2.1.1 Use Case Diagramm

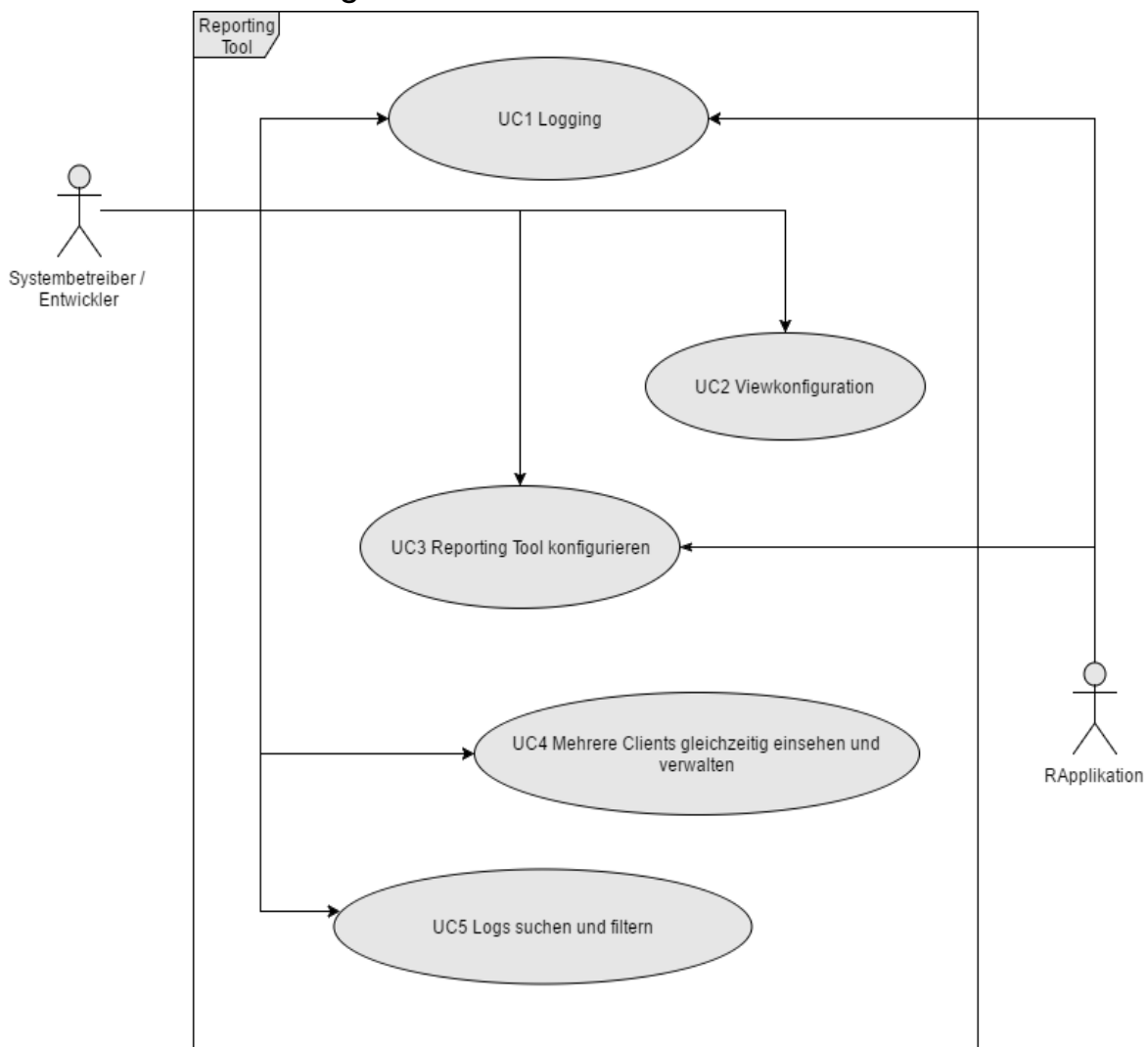


Bild 3: Use Case Diagramm

Der Hauptstakeholder hier sind die Systembetreiber und Entwickler von Software, die die Reporting Bibliothek zum Loggen verwenden. Hierbei sind auch an bestimmten Use Cases die RApplikationen interessiert, da Sie gemäss diesen Use Cases Ihr Verhalten anpassen. RApplikation sind RAD Applikationen, die die Reporting Bibliothek zum Loggen einsetzen.

4.2.1.2 Use Cases (Brief)

In diesem Abschnitt sind die Use Cases aufgeführt. Detailliertere Beschreibungen wichtiger Use Cases kann man im Anhang B unter Abschnitt Use Cases (Essential Style) finden.

4.2.1.2.1 UC1 Logging

Logs können eingesehen, gespeichert und geparkt werden. Werden die Log - Files in eine Datei geschrieben, so kann diese Datei geladen und angezeigt werden. Das Parsing für die Logdateien betrifft die binären Logs, welche vom binären Format in das HEX - Format geparkt werden sollen.

4.2.1.2.2 UC2 Viewkonfiguration

Der Benutzer kann die Views konfigurieren, wie z.B. von einer Anzeige im Tab - Format in freischwebende Fenstern umschalten. Ausserdem soll der Benutzer die Views zusammenführen, den Views eine bestimmte Hintergrundfarbe zuordnen, paralleles Scrolling aktivieren/deaktivieren können.

Optional: Die Views können in voreingestellten Layouts angezeigt werden, wie z.B. 2-Fenster, 4-Fenster, Fenstergruppierungen etc.

4.2.1.2.3 UC3 Reporting Tool konfigurieren

Der Benutzer kann System- und Reportingkonfigurationen vornehmen, wie z.B. auf welcher IP das Reporting Tool hören soll (integriertes Master), wo die Log - Dateien gespeichert werden, wo die Konfigurationsdateien sich befinden etc. Ausserdem sollen verschiedene Konfigurationen per verschiedenen Konfigurationsfiles geladen werden können (Profile), so dass z.B. direkt Clients A, B und C mit Enable Debug "reporten" und man das nicht jedesmal neu einstellen muss.

Bestimmte Client's sollen unterdrückt werden können, andere erst loggen, wenn der Buffer überläuft.

4.2.1.2.4 UC4 Mehrere Clienten gleichzeitig verwalten

Das Reporting Tool kann von mehreren Clients gleichzeitig kontaktiert werden (automatisches Update der Einträge in Group und Clientlist), angezeigt und eingestellt werden (Filterung).

4.2.1.2.5 UC5 Logs durchsuchen und filtern

Der Benutzer kann anhand Schlüsselwörter und speziellen Ausdrücken Logs durchsuchen oder auf Logs Filter anwenden. Hierbei soll eine History zur Anwendung kommen, die eine bestimmte Anzahl an Einstellungen sich merkt und dem Benutzer bei der nächsten Suche oder Anwendung eines Filters per Ausdrücke diese vorschlägt. Die Suche und das Filtern soll auf globaler wie auch auf applikationsspezifischer Basis möglich sein (*).

(*) Identifizierung der einzelnen Clients für Filter, die clientseitig "filtern" (wurde per Control - Files im alten Reporting Tool umgesetzt), muss hierfür möglich sein. Alle Clients, die nicht identifizierbar sind, sollen in einer zusammengeführten View angezeigt werden.

4.2.1.3 Nicht funktionale Anforderungen

4.2.1.3.1 Benutzbarkeit

- Das Reporting Tool soll alle vorherigen separaten Tasks in einer Applikation auf sammeln und somit eine einfache, angenehme Benutzung gewährleisten. Vor allem da die Anzahl Clients sehr hoch sein kann, sollte der Benutzer auf Anhieb eine gute Übersicht erhalten und wissen, wo, was, wie zu erledigen ist.

4.2.1.3.2 Zuverlässigkeit

- Für die Übertragung der Daten über das Netzwerk wird das TCP Protokoll benutzt und somit wird sichergestellt, dass die Logdaten auch ankommen. Das Tool soll alle erhaltenen Logs auch anzeigen.
- Das Reporting Tool muss 24/7 verfügbar sein.

4.2.1.3.3 Performanz

- Da das Reporting Tool in Echtzeitsystemen arbeitet, sollte es unter 24/7 Betrieb mehrere Clients gleichzeitig verarbeiten können.
- Buffers sollen nicht überlaufen: Daten sollen schnell genug auf die Platte geschrieben werden können
- Netzwerkperformanz: schnelles lesen vom Netzwerk (Alle ~20 ms 1 KB, pro Socket/Client)
- Das System soll bis zu 20 gleichzeitige Clients verwalten können, ohne Performanzeinbussen.
- Loghistory: 10'000 Zeilen pro Client

4.2.1.3.4 Wartbarkeit

- "Sollte für die nächsten 30 Jahre laufen"
- Jemand anders sollte den Source Code später weiterentwickeln können.

4.2.2 Domainanalyse

In diesem Abschnitt wird auf die Domainobjekte eingegangen. Im Klassendiagramm wurde die Auflistung von Getter/Setter Funktionen bewusst verzichtet und nur jene aufgelistet, die relevant sind.

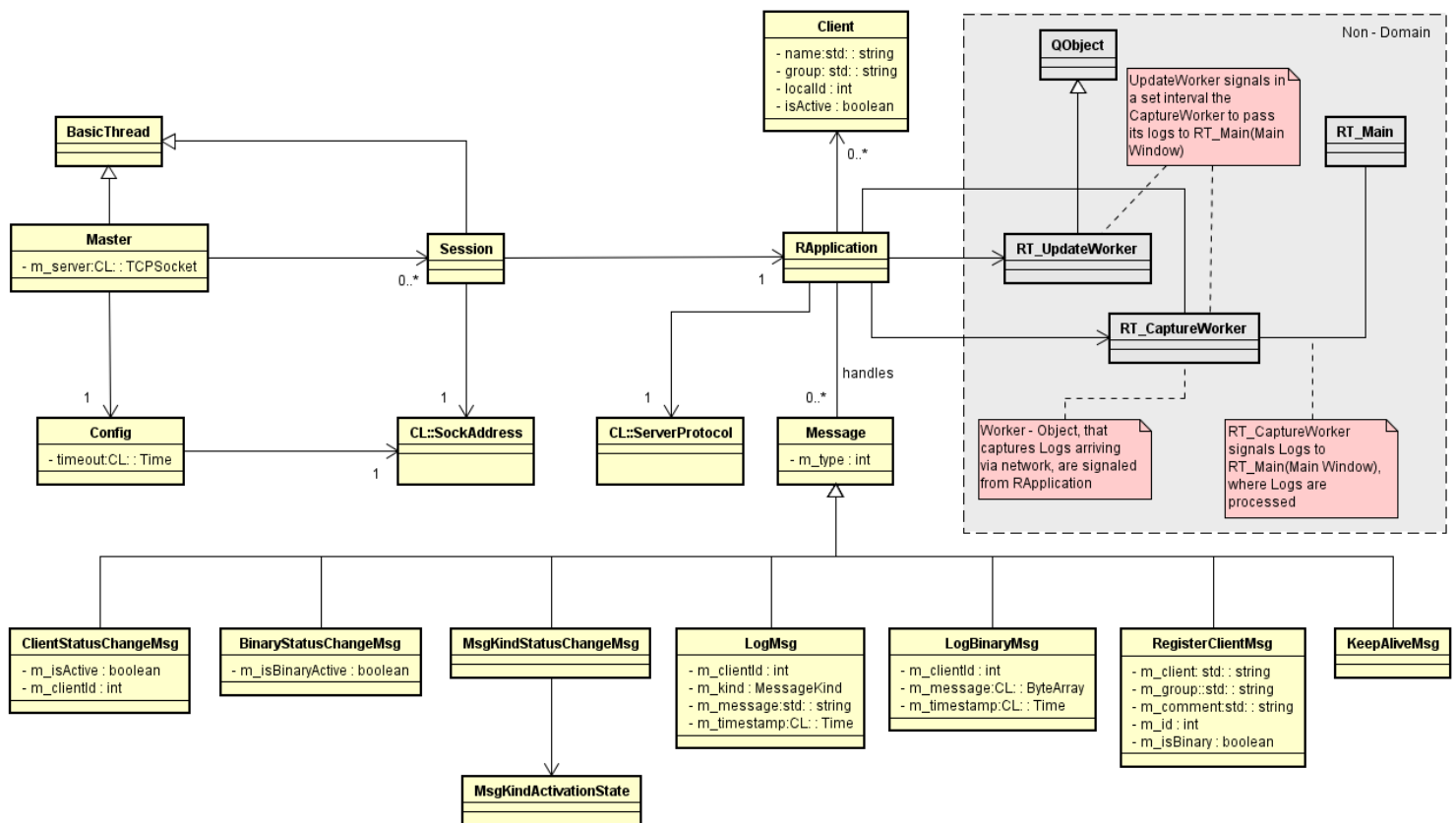


Bild 4: Klassendiagramm Domain

Die Klassen RT_CaptureWorker, RT_UpdateWorker und RT_Main werden hier nicht weiter erläutert und sind nur zur Erläuterung der Beziehung des Domains mit dem View - Layer aufgeführt.

Objekt	Beschreibung
BasicThread	Eine einfache Thread-Implementation der ReportingLibrary
Master	Ein Listener - Thread, der auf Applikationsverbindungen hört, die die Reporting Library einsetzen.
Session	Ein Session - Thread, in der verbundene Applikationen verwaltet werden. Session enthält eine Instanz RApplication, dass die Applikationen, die die Reporting Library benutzen, modelliert.
Client	Struktur, die die Channels in einer RApplication repräsentiert.
RApplication	Modelliert eine Applikation, die die Reporting Library einsetzt und verarbeitet Lognachrichten dieser Applikation, die übers Netzwerk ankommen. Die Klasse leitet von QObject ab und wird vom Qt Framework mit Metainformationen erweitert. Dies ist nötig, um die Logs per Signals and Slots Konzept des Qt Frameworks an die entsprechenden Worker - Klassen zu kommunizieren, die dann die

Config	<p>Logs an das Hauptfenster weiterleiten.</p> <p>Einfache Konfigurationsstruktur für den Master - Thread. Enthält Informationen wie z.B. Timeout für Verbindungen oder Listening Adresse.</p>
CL::SockAddress	Eine Repräsentation einer Socket Adresse aus Common Library aus dem Reporting Library
CL::ServerProtocol	ServerProtocol Instanz aus dem Common Library (in der Reporting Library enthalten). Anhand dieser Instanz werden die Lognachrichten für eine RApplication verarbeitet.
Message	Basisklasse für Lognachrichten. Enthält das Feld "type", dass den Typ der Lognachricht identifiziert. Anhand diesem "type" - Feld wird das Loghandling in der RApplication - Klasse abgewickelt.
ClientStatusChangeMsg	Message - Klasse um übers Netzwerk Client Statusänderungen zu kommuniziert. Mit diesem Message - Typ werden Clients clientseitig aktiviert/deaktiviert.
BinaryStatusChangeMsg	Message - Klasse um übers Netzwerk binäre Client Statusänderungen zu kommuniziert. Mit diesem Message - Typ werden binäre Clients clientseitig aktiviert/deaktiviert. Binäre Clients unterscheiden sich hauptsächlich in Ihrem Messageinhalt, ie. binäre Clients enthalten Ihre Lognachricht in Binärform (ByteArray).
MsgKindStatusChangeMsg	Message - Klasse um übers Netzwerk die zu loggenden Logtypen zu kontrollieren. Anhand dieser Message - Klasse wird allen Clients z.B. kommuniziert, ob Sie Logs vom Typ "Info" loggen sollen oder nicht.
MsgKindActivationState	Modelliert den Status für die verschieden Logtypen und ob diese aktiviert/deaktiviert sein soll.
LogMsg	<p>Message - Klasse, die eine normale Lognachricht modelliert. LogMsg enthalten Ihre Lognachricht in einfachem String-Format. Ausserdem enthalten die LogMsg's Informationen zu welchem Client sie gehören, um was für einen Logtyp es sich handelt und Zeitinformationen.</p>
LogBinaryMsg	Genau wie LogMsg, einfach dass der Loginhalt als ByteArray, also in Binärform enthalten ist. Dieser Message-Typ wird von Binärclients verwendet.
RegisterClientMsg	Message - Instanz, anhand derer Clients sich übers Netzwerk beim Reporting Tool registrieren können. Enthält Information wie Clientname, Gruppenname etc.
KeepAliveMsg	Eine einfache KeepAliveMsg, die in kurzen Intervallen an die ReportingGUI seitens der Applikation (die die Reporting Library benutzt) geschickt wird. In der Funktion, in der KeepAliveMsg verarbeitet werden, können z.B. Clientstatsänderungen an die Clients zurückgesendet werden.

4.2.3 Ablauf Verbindungsaufbau von Clients

Das Systemsequenzdiagramm in Bild 5 zeigt den wichtigsten Systemablauf auf, nämlich den Aufbau der Verbindung der Applikation mit den Clients und das Weiterleiten der Logs zum Master-Programm, der auf einem bestimmten Port auf Logs hört. Nachdem die Verbindung aufgebaut wird, werden die ankommenden Lognachrichten zuerst analysiert und der zugehörige Client determiniert. Mit diesen Informationen werden die Logs dann zum CaptureWorker weitergeleitet, wo Sie bis zum nächsten Update-Signal angestaut werden. Beim Update-Signal werden die Logs in Batches zum UI weitergesendet, wo Sie dann der entsprechenden Client Tabelle zugeordnet werden.

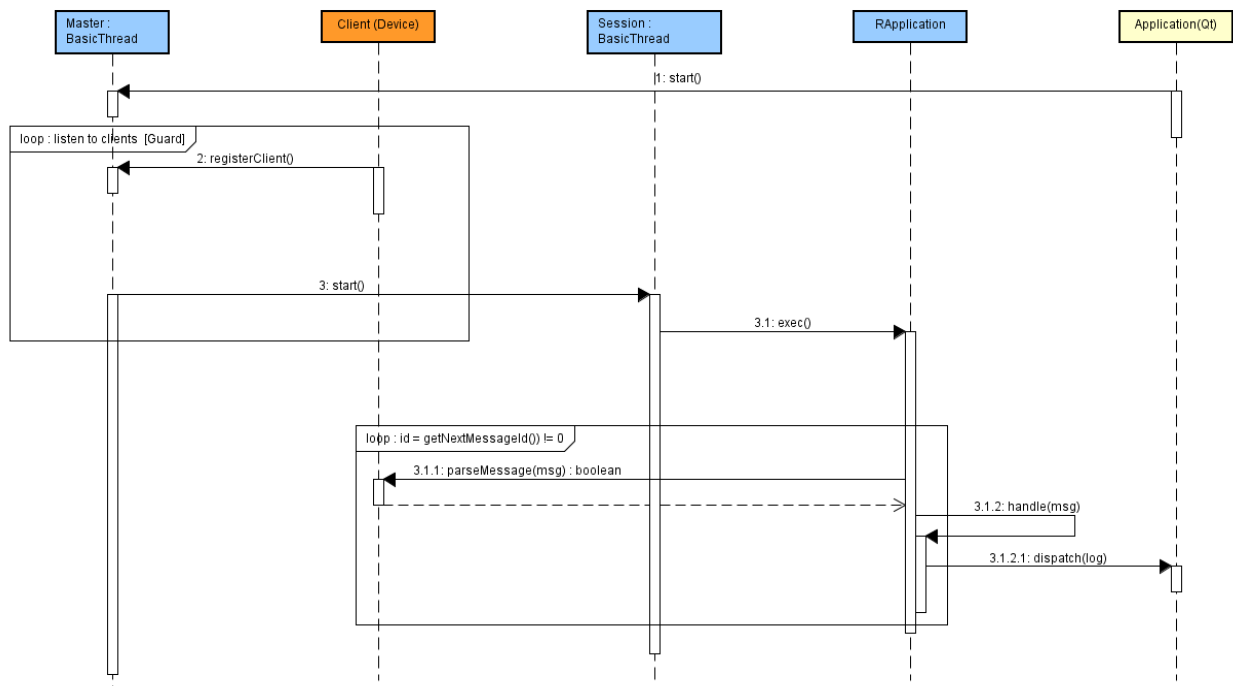


Bild 5: SSD Master-Thread in Verbindung mit der GUI Applikation

4.2.4 RApplication/Client Konzept

Die Reporting Bibliothek wird von verschiedenen Applikationen der Rheinmetall Air Defence AG verwendet. Dabei gibt es die Möglichkeit, die Applikationen für das Loggen zu gruppieren. Hierbei kommt das RApplication/Client Konzept ins Spiel. Eine RApplication stellt im Groben eine Applikation dar, die die Reporting Bibliothek einsetzt. Hierbei können folgende Einstellung, wie in Bild 6 angezeigt, für die RApplication vorgenommen werden.

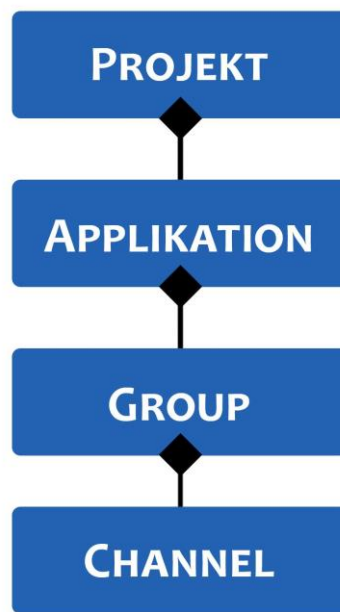


Bild 6: RApplication Gruppierungen

Level	Beschreibung (kurz)
Projekt	Es gibt verschiedene Projekte, die mehrere Applikationen haben.
Applikation	Jede Applikation macht eine eigene TCP - Verbindung zum ReportingGUI. Applikationen werden per Group - Prefix bestimmt und sind auf Applikationsbasis nicht unterscheidbar.
Group	Jede Gruppe gehört zu einer Applikation. Gruppennamen sind nicht eindeutig; was heissen soll, dass mehrere Applikationen den gleichen Gruppennamen haben können. Das ReportingGUI kann deswegen zurzeit nicht auf Basis Gruppennamen eindeutig zwischen Applikationen unterscheiden. Man muss bei der Vergabe einer Gruppennamen also darauf achten, dass dieser eindeutig ist, damit die Applikationen voneinander unterschieden werden.
Channel	Jeder Channel ist einer einzigen Gruppe zugeordnet. Channels stellen Subsektionen von Gruppen dar. Als Clients werden die Channels angezeigt.

Tabelle 1: Beziehungen zwischen verschiedenen Ebenen der Logging Konfigurationen

Um die Beziehung zwischen den verschiedenen Ebenen besser zu verdeutlichen, soll das Konzept anhand eines Projektbeispiels in Bild 7 verdeutlicht werden.

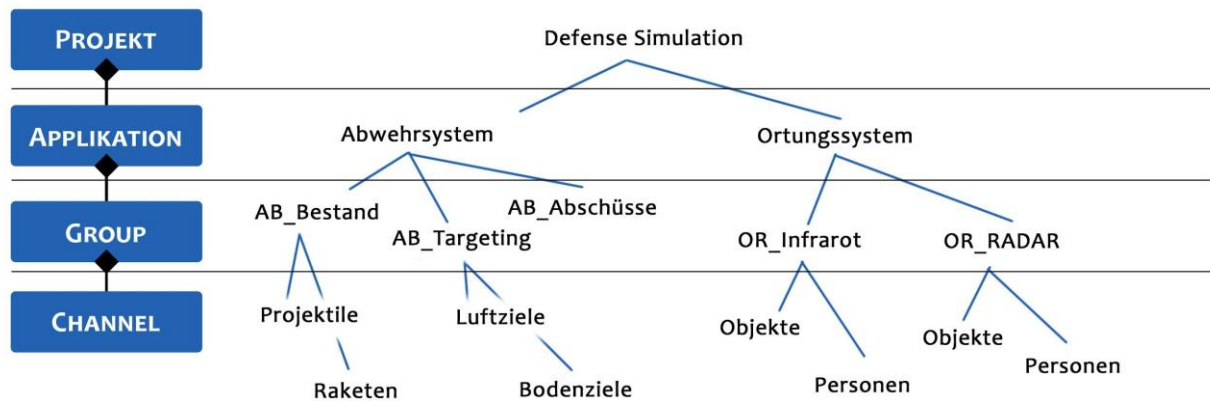


Bild 7: Ebenenbeziehung RApplication, Beispiel Projekt Defense Simulation

Da nur auf den Ebenen Group und Channel eindeutig unterschieden werden kann, werden im ReportingGUI jeweils Groups und Clients zum Verwalten angeboten. Die jeweiligen Gruppennamen werden, falls vorhanden, mit den zugehörigen Präfixe der oberen Levels vorangestellt, um wenigstens auf die Zugehörigkeit anzudeuten.

4.3 Architektur

Hier werden die wichtigsten Klassen der Applikation aufgeführt und beschrieben. Es wird auf relevante Felder und Funktionen eingegangen und die Rolle der Klasse beschrieben. Eine detaillierte Aufführung aller Felder und Funktionen kann in der API Dokumentation nachgelesen werden. Die API Dokumentation kann in der Applikation per F1 oder direkt unter dem doc - Folder, den man unter dem Applikationsfolder finden kann, aufgerufen werden. Ausserdem werden in diesem Abschnitt wichtige Abhängigkeiten zwischen Klassen und der logisch gruppierten Elemente aufgezeigt.

4.3.1 Logische Abhängigkeiten

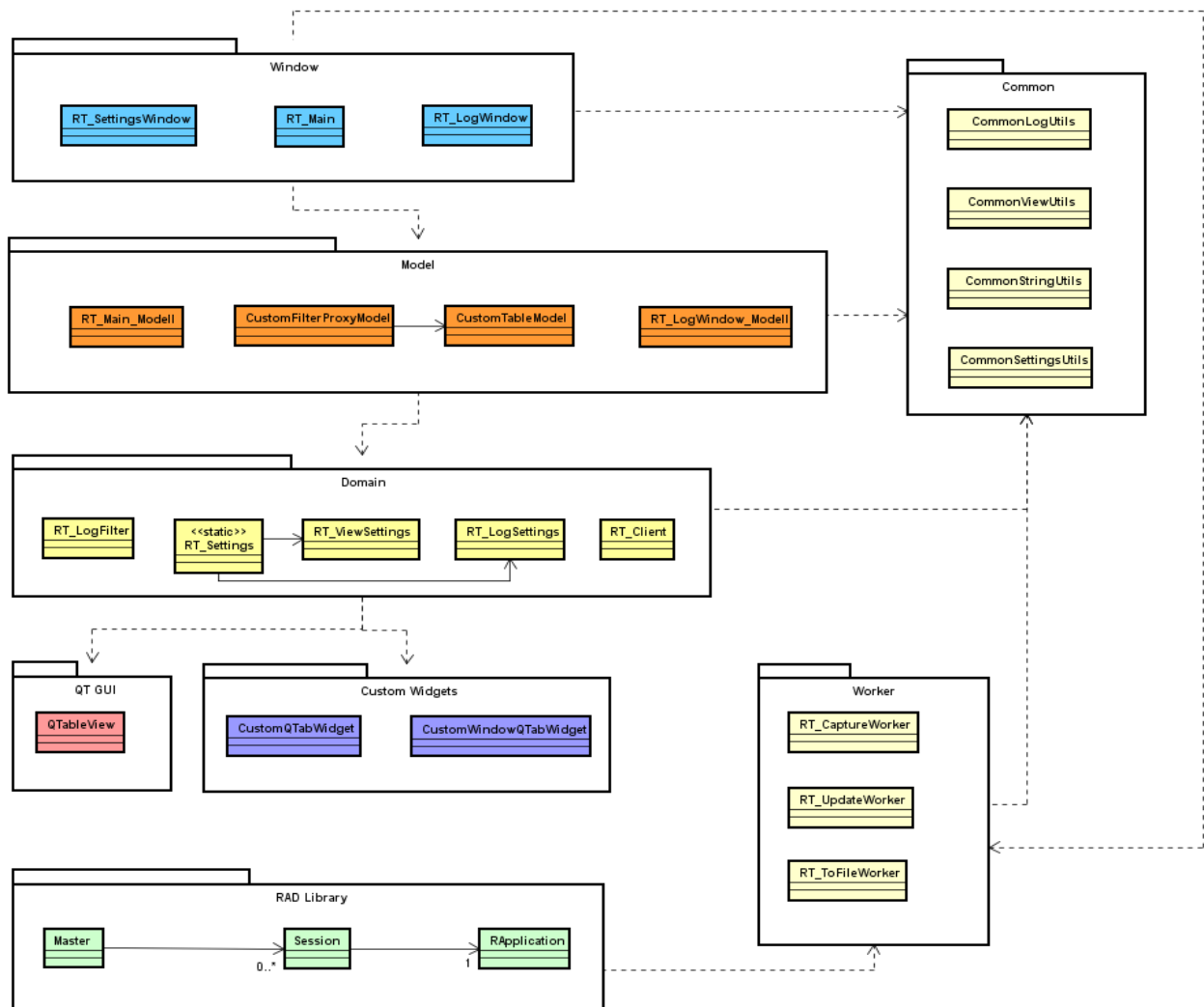


Bild 8: Diagramm logischer Abhängigkeiten

4.3.2 Klassen Abhängigkeiten

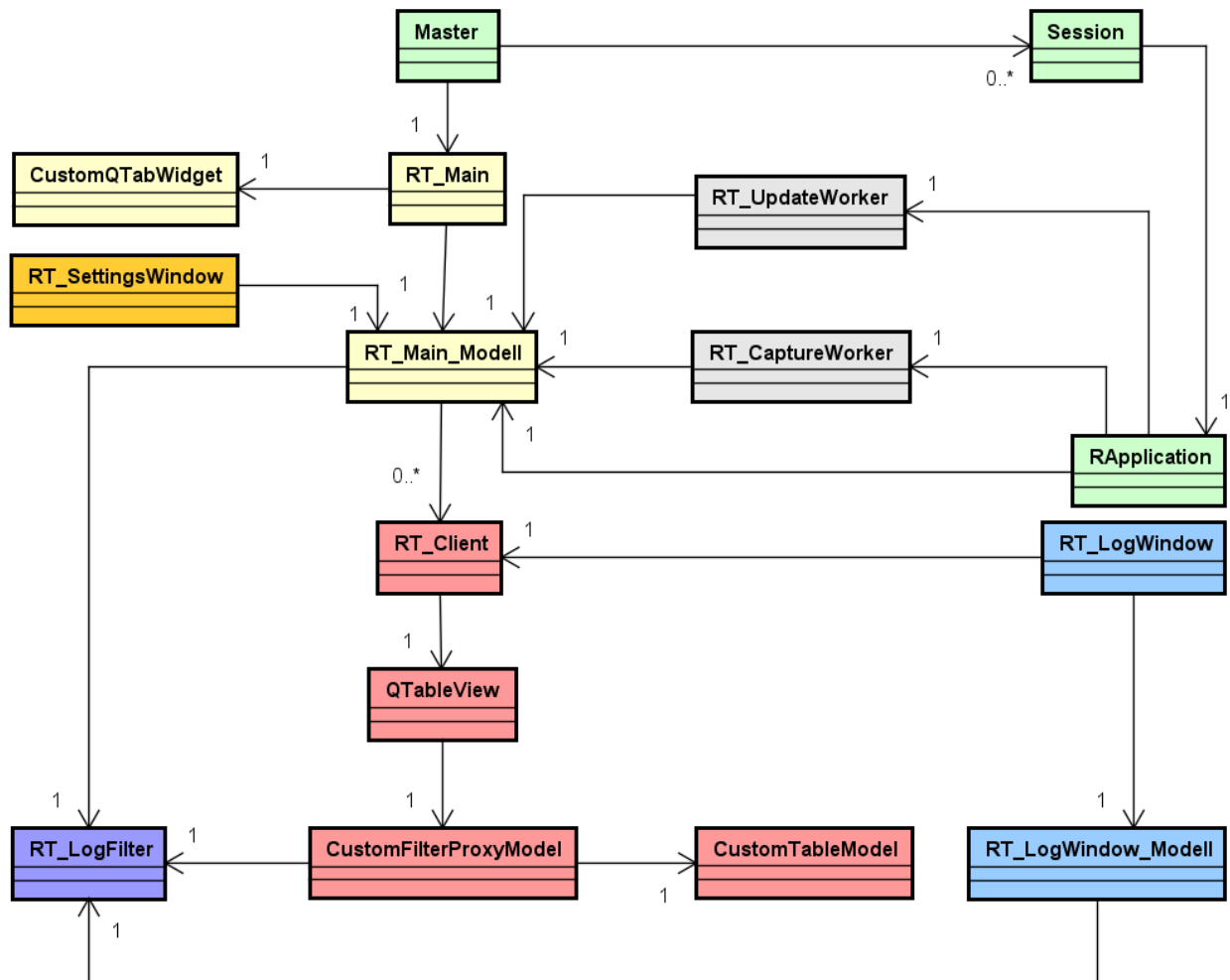


Bild 9: Klassendiagramm Abhängigkeiten

Hierbei sind Master, Session und RApplication Klassen aus der Reporting Library. Die Farben sollen hier zusammengehörige Elemente visualisieren.

4.3.3 Window

In diesem Abschnitt werden die Fensterklassen beschrieben und erläutert.

4.3.3.1 RT_Main

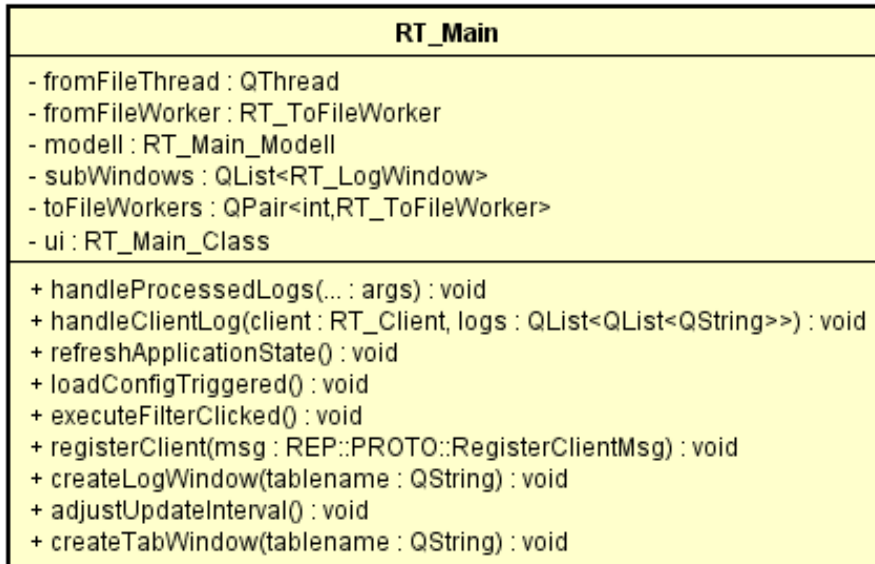


Bild 10: Klassendiagramm RT_Main

Die RT_Main Klasse stellt die Hauptfensterklasse dar und leitet von Klasse QMainWindow ab. QMainWindow ist eine Fensterklasse von Qt die eine Menüsektion und einen Statusbar anbietet. Diese Klasse stellt den "main entry point" in die Applikation dar.

Felder	Beschreibung (kurz)
fromFileThread	Ein eigener, separater Thread für Leseoperation gespeicherter Logs. Das fromFileThread Feld ist vom Typ QThread und wird von der Fensterklasse RT_Main verwaltet, sprich wird das Fenster zerstört, wird auch dieser Thread automatisch aufgeräumt.
fromFileWorker	Ist eine Instanz von RT_ToFileWorker, der speziell für Leseoperationen alleinstehend instanziiert wird. Der RT_ToFileWorker wird per moveToThread dem fromFileThread (QThread) zugewiesen.
modell	Ist die Modellklasse des Hauptfensters und verwaltet hauptsächlich Clients vom Typ RT_Client. Die RT_Main_Modell Klasse wird ausserdem an verschiedenen Stellen der Applikation referenziert und enthält wichtige Felder wie updateInterval etc., siehe Beschreibung RT_Main_Modell für mehr Informationen
subWindows	Ist ein Container vom Typ QMap<QString, RT_LogWindow> und verwaltet vom Hauptfenster verschiedene Fenster, z.B. wenn Logfenster erzeugt werden, werden diese in den subWindows Container eingetragen.
toFileWorkers	Ist ein Container vom Typ QPair<int, QList<RT_ToFileWorker>>. Das QPair enthält in seinem ersten Feld (toFileWorkers.first) die Nummer der aktiven Worker - Instanz, an den Logdateien als nächstes geschickt werden. Das zweite Feld enthält eine Liste von Pointers auf die RT_ToFileWorkers. Anhand dieses Containers werden die zirkuläre Zuweisung von toFile Operationen auf vier verschiedene RT_ToFileWorkers realisiert.

ui	Bildet die Referenz auf die generierte <code>Ui::RT_MainClass</code> dar, anhand dieser auf Elemente der UI des Hauptfensters zugegriffen werden kann.
-----------	--

Funktionen	Beschreibung (kurz)
handleProcessedLogs	Ist ein Slot des Hauptfensters, zu der die angehäuften Logs im <code>RT_CaptureWorker</code> gesandt werden. Hier wird der Client (<code>RT_Client</code>) aus dem <code>RT_Main_Modell</code> determiniert und an <code>handleClientLog</code> weitergeleitet.
handleClientLog	Stellt die Hauptfunktion des Loghandlings seitens der Hauptfensterklasse dar. <code>HandleClientLog</code> nimmt eine <code>RT_Client</code> Referenz und eine diesem Client zugehörige Liste <code>QList<QList<QString></code> von Logs entgegen und routet diese gemäss dem <code>RT_Client</code> Zustand, sprich falls der <code>RT_Client</code> auf <code>toFile</code> eingestellt ist, werden die entsprechenden Logs in eine Datei geschrieben.
refreshApplicationState	Die Funktion <code>refreshApplicationState</code> aktualisiert, wie der Name es erahnen lässt, den Applikationszustand. Dazu gehören die Groups und Clients, die globalen Logeinstellungen und Vieweinstellungen. Ausserdem initialisiert diese Funktion gegebenenfalls Tabs und Logfenster. Falls in der Konfiguration angegeben, werden bestimmte Clients auch aktiviert und fangen an zu loggen. Diese Funktion hängt eng mit den Applikationskonfigurationen zusammen und hat die primäre Aufgabe, den persistierten Zustand in der Applikationskonfiguration wiederherzustellen.
loadConfigTriggered	Initialisiert einen <code>FileDialog</code> , von der aus die gewünschte Konfiguration ausgewählt und geladen werden kann. Nach Auswahl des Files wird <code>refreshApplicationState</code> aufgerufen, um den Applikationszustand der Konfiguration auf die Applikation abzubilden.
executeFilterClicked	Wendet den eingegeben Filterausdruck je nach Einstellung auf die aktuelle Client Tabelle oder alle Client Tabellen (Global Checkbox).
registerClient	Ist ein Slot der Hauptfensterklasse, auf die <code>REP::PROTO::RegisterClientMsg's</code> , die übers Netzwerk ankommen, weitergeleitet werden. Die <code>registerClient</code> Funktion erstellt, falls noch nicht vorhanden, einen neuen Client, welcher im <code>RT_Main_Modell</code> hinterlegt wird.
createLogWindow	Erzeugt ein <code>RT_LogWindow</code> Fenster für einen Client. Der Funktion <code>createLogWindow</code> wird der normalerweise der Name des Clients übergeben, welcher dann als Fenstertitle angezeigt wird.
createTabWindow	Die gleiche Funktionalität wie <code>createLogWindow</code> , nur das diese Funktion eine Tab - Ansicht für einen Client erzeugt.
adjustUpdateInterval	Aktualisiert das Updateintervall des <code>RT_UpdateWorker's</code> . Diese Funktion wird mit der Erzeugung von neuen Logfenstern aufgerufen, wobei gilt: je mehr Fenster, desto grösser das Updateintervall.

4.3.3.2 RT_LogWindow

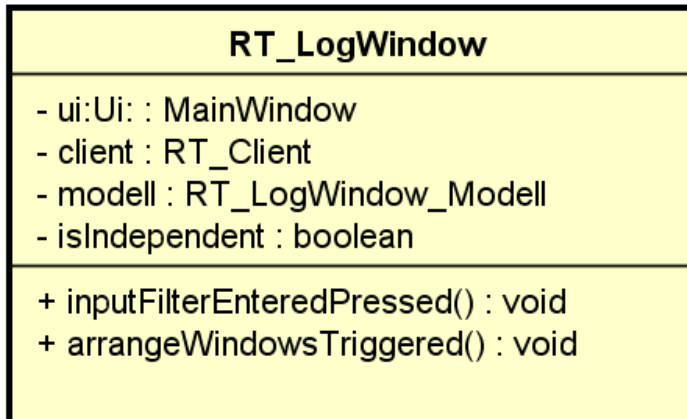


Bild 11: Klassendiagramm RT_LogWindow

Die RT_LogWindow Fensterklasse stellt ein Fenstertyp dar, dass Fenstermodi für Clients darstellt. Die RT_LogWindow Klasse enthält im Gegensatz zur Hauptfensterklasse eine sehr schmale Modelklasse, die RT_LogWindow_Modell Klasse, die Fensternamen und Filter festhält.

Felder	Beschreibung (kurz)
ui	Enthält Referenz auf die generierte Fensterklasse Ui::MainWindow, welche Zugang auf die UI Elemente anbietet.
client	RT_Client Referenz des Clients, für das das Logfenster erzeugt worden ist.
modell	Eine schmale Modelklasse vom Typ RT_LogWindow_Modell, dass Fensternamen und Filter festhält.
isIndependent	Boolean-Wert, dass den Typ des Logfensters festhält. Das Feld isIndependent ist true, falls das Fenster per Logimport generiert wurde. Das Logfenster wird entsprechend mit einem engeren Featureset initialisiert.
Funktionen	Beschreibung (kurz)
inputFilterEnteredPressed	Wendet den eingeben Filter auf den aktuellen Client an (RT_Client)
arrangeWindowsTriggered	Initialisiert die Prozedur für die Anordnung der Fenster am Monitor.

4.3.3.3 RT_SettingsWindow

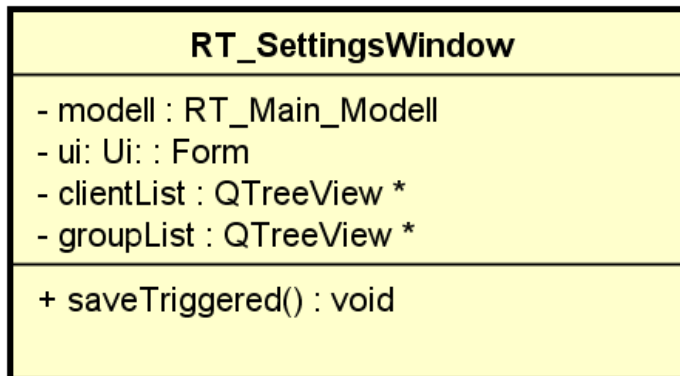


Bild 12: Klassendiagramm RT_SettingsWindow

Das RT_SettingsWindow stellt eine Fensterklasse für Applikationseinstellungen dar. RT_SettingsWindow leitet von QDialog ab. QDialog stellt eine Qt Fensterklasse dar, die keine Menüleiste weder einen Statusbar anbietet.

Felder	Beschreibung (kurz)
modell	Ist eine Referenz auf die RT_Main_Modell Klasse. Die RT_SettingsWindow benötigt diese Referenz, um z.B. Clients/Groups und Ihre Zustände zu persistieren.
ui	Enthält Referenz auf die generierte Fensterklasse Ui::Form, welches Zugriff auf die UI Elemente des RT_SettingsWindow erlaubt.
clientList	Pointer auf Clientliste der Hauptfensterklasse. Diese Referenz wird gebraucht, um die selektierten Clients ausfindig zu machen.
groupList	Pointer auf Groups - Liste der Hauptfensterklasse. Diese Referenz wird gebraucht, um die selektierten Groups ausfindig zu machen.

Funktionen	Beschreibung (kurz)
saveTriggered	Initialisiert den Speichervorgang. Falls persistSettingsCheckBox aktiviert ist, ruft diese Funktion zusätzlich die Funktion save der Klasse RT_Settings, welches die Einstellungen in einer externen Datei, einer Konfigurationsdatei, abspeichert.

4.3.4 Domain

In diesem Abschnitt werden die Domainobjekte näher betrachtet.

4.3.4.1 RT_LogFilter

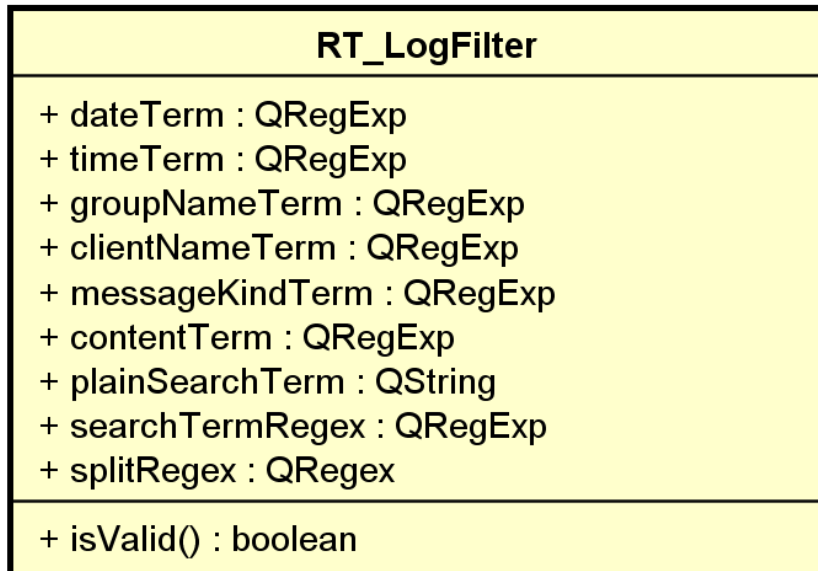


Bild 13: Klassendiagramm RT_LogFilter

Die Klasse `RT_LogFilter` modelliert die Filtereinstellungen für die Logfilterung. Im Konstruktor nimmt diese Klasse den Filterstring entgegen und zerlegt diesen in verschiedene Regexpfelder. Anhand dieser Felder findet die Filterung im `CustomFilterProxyModel` statt.

Felder	Beschreibung (kurz)
clientNameTerm	Feld vom Typ <code>QRegExp</code> . Enthält Filterausdruck für Clientnamen.
contentTerm	Feld vom Typ <code>QRegExp</code> . Enthält Filterausdruck für Loginhalt.
dateTerm	Feld vom Typ <code>QRegExp</code> . Enthält Filterausdruck für das Datumfeld.
groupNameTerm	Feld vom Typ <code>QRegExp</code> . Enthält Filterausdruck für Gruppennamen.
messageKindTerm	Feld vom Typ <code>QRegExp</code> . Enthält Filterausdruck für den Logtyp.
timeTerm	Feld vom Typ <code>QRegExp</code> . Enthält Filterausdruck für das Zeitfeld..
plainSearchTerm	Enthält den originalen Filterausdruck. Dieses Feld ist vom Typ <code>QString</code>
searchTermRegex	Feld vom Typ <code>QRegExp</code> . Anhand dieses Feldes wird der Filterausdruck auf seine Gültigkeit überprüft.
splitRegex	Feld vom Typ <code>QRegExp</code> . Anhand dieses Feldes wird der Filterausdruck gesplittet.
Funktionen	Beschreibung (kurz)
isValid	Funktionen, die Auskunft über die Gültigkeit des Filterausdrucks gibt.

4.3.4.2 RT_LogSettings

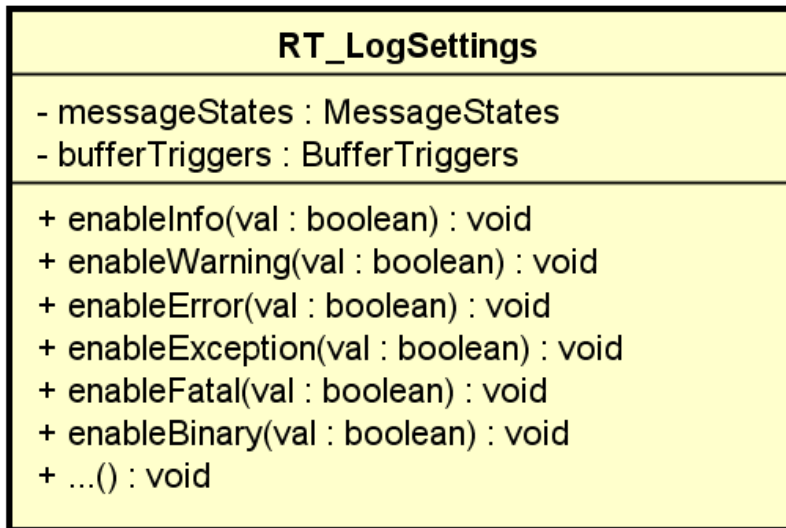


Bild 14: Klassendiagramm RT_LogSettings

Die Klasse RT_LogSettings modelliert die globalen Logeinstellungen. Der Zustand dieser Klasse wird in der Klasse RApplication gelesen und das Logverhalten angepasst.

Felder	Beschreibung (kurz)
messageStates	Feld vom Typ MessageStates. Die Klasse MessageStates ist eine einfache Klasse, die die Zustände festhält, welche Logtypen die Clients loggen sollen oder nicht.
bufferTriggers	Feld vom Typ BufferTriggers. Die Klasse BufferTriggers ist eine einfache Klasse, die die Zustände festhält, welche Logtypen buffert werden sollen, bevor Sie an die Hauptfensterklasse weitergeleitet werden.

Funktionen	Beschreibung (kurz)
enableInfo	Funktion, die den Zustand der Clients für den Logtyp Info wiedergibt. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
enableWarning	Funktion, die den Zustand der Clients für den Logtyp Warning wiedergibt. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
enableError	Funktion, die den Zustand der Clients für den Logtyp Error wiedergibt. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
enableException	Funktion, die den Zustand der Clients für den Logtyp Exception wiedergibt. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
enableFatal	Funktion, die den Zustand der Clients für den Logtyp Fatal wiedergibt. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
enableBinary	Funktion, die den Zustand der Clients für den Logtyp Binary wiedergibt. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
infoMessage	Funktion, die den Bufferzustand der Clients für den Logtyp Info festhält. Diese Funktion ist überladen und über diesen kann auch der

	Zustand gesetzt werden.
warningMessage	Funktion, die den Bufferzustand der Clients für den Logtyp Warning festhält. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
errorMessage	Funktion, die den Bufferzustand der Clients für den Logtyp Error festhält. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
exceptionMessage	Funktion, die den Bufferzustand der Clients für den Logtyp Exception festhält. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.
fatalMessage	Funktion, die den Bufferzustand der Clients für den Logtyp Fatal festhält. Diese Funktion ist überladen und über diesen kann auch der Zustand gesetzt werden.

4.3.4.3 RT_Client



Bild 15: Klassendiagramm RT_Client

Die Klasse RT_Client modelliert die Clients (Channels). In dieser Klasse werden die Clientzustände festgehalten. Nebst den Zuständen enthält die RT_Client Klasse auch die zugehörige Tabelle, der die zugehörigen Logs eingefügt werden.

Felder	Beschreibung (kurz)
--------	---------------------

isBinary	Boolean Zustand, ob der Client ein Binärclient ist.
isActive	Boolean Zustand, ob der Client aktiv ist. Wenn der Client aktiv ist, fangen die Clients an zu loggen.
isLive	Boolean Zustand, ob der Client Live ist. Die Logs eines live Clients werden im UI angezeigt (im Tab oder im Fenster).
isToFile	Boolean Zustand, ob der Client seine Logs in eine Datei schreiben soll.
isToRingFile	Boolean Zustand, ob der Client seine Logs in eine Ringdatei schreiben soll.
isToShareFile	Boolean Zustand, ob der Client seine Logs in eine geteilte Datei schreiben soll.
isToParent	Boolean Zustand, ob der Client seine Logs zu seiner Gruppe loggen soll.
isToGlobal	Boolean Zustand, ob der Client seine Logs zum Global Ausgabe loggen soll.
isToMergeView	Boolean Zustand, ob der Client seine Logs zum MergeView loggen soll.
isNewestOnTop	Boolean Zustand, ob die Logs des Clients zuoberst eingefügt werden sollen.
isScrollToBottom	Boolean Zustand, ob der Client seine View nach unten scrollen soll.
isWindowed	Boolean Zustand, ob der Client im Fenstermodus ist oder nicht.
clientId	Integer Id des Clients.
clientName	Name des Clients vom Typ QString
groupName	Name der Gruppe des Clients vom Typ QString
mergeViewName	Name des MergeViews, zu der der Client loggen soll. Arbeitet zusammen mit dem Flag isToMergeView. Ist vom Typ QString.
sharedFileName	Name der geteilten Datei, zu der der Client loggen soll. Arbeitet zusammen mit dem Flag isToShareFile. Ist vom Typ QString
Funktionen	
addLog	Beschreibung (kurz) Funktion, um der Client Tabelle Logs hinzuzufügen. Die Funktion addLog benutzt intern die Werte isNewestOnTop und isScrollToBottom um zu determinieren, wie die Logs zur Tabelle hinzugefügt werden sollen.

4.3.4.4 RT_Settings

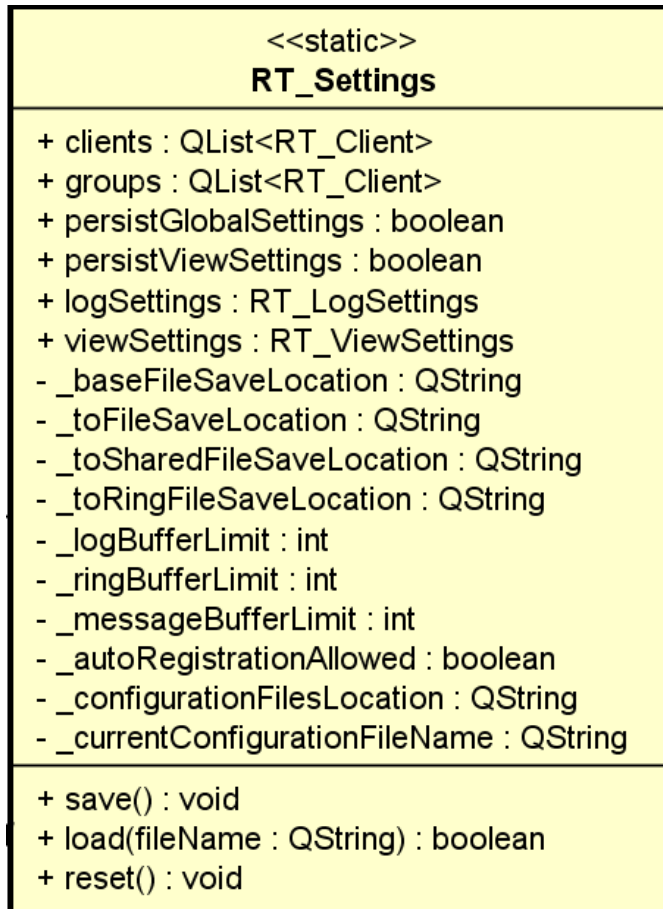


Bild 16: Klassendiagramm RT_Settings

Die Klasse RT_Settings modelliert die Applikationskonfiguration und stellt systemweit Applikationswerte bereit. RT_Settings ist eine statische Klasse.

Felder	Beschreibung (kurz)
persistGlobalSettings	Flag, anhand der die global Logeinstellungen persistiert werden oder nicht.
persistViewSettings	Flag, anhand der die Vieweinstellungen persistiert werden oder nicht.
logSettings	Feld vom Typ RT_LogSettings, dass Zustände über globale Logeinstellungen verwaltet.
viewSettings	Feld vom Typ RT_ViewSettings. RT_ViewSettings modelliert View bezogene Zustände der Applikation.
clients	Container vom Typ QList<RT_Client>, dass zu speichernde Clients festhält. Dieser Container wird je nach Speicherart angepasst.
groups	Container vom Typ QList<RT_Client>, dass zu speichernde Groupclients festhält. Dieser Container wird je nach Speicherart angepasst.
baseFileSaveLocation	Feld vom Typ QString, dass den Basispfad für zu speichernde Logs enthält.
toFileSaveLocation	Feld vom Typ QString, dass den Speicherpfad für Logs enthält,

	die in eine einfache Datei routen.
toSharedFileSaveLocation	Feld vom Typ QString, dass den Speicherpfad für Logs enthält, die in eine geteilte Datei routen.
toRingFileSaveLocation	Feld vom Typ QString, dass den Speicherpfad für Logs enthält, die in eine Ringdatei routen.
logBufferLimit	Ein Integer - Wert für die maximale Anzahl Logs in einer Tabelle.
ringBufferLimit	Ein Integer - Wert für die maximale Anzahl Logs in einer Ringdatei.
messageBufferLimit	Ein Integer - Wert für die maximale Anzahl Logs in einem Buffer.
autoRegistrationAllowed	Wenn das Feld autoRegistrationAllowed auf "true" gesetzt wird, können sich Clients per Netzwerkmessage REP::PROTO::RegisterClientMsg automatisch bei der Applikation anmelden.
configurationFilesLocation	Feld vom Typ QString, dass den Pfad für Konfigurationsdateien enthält. Dieser ist normalerweise im Applikationspfad, der Pfad /config
currentConfigurationFileName	Feld vom Typ QString, das den Namen der aktuellen Konfiguration festhält.

Funktionen	Beschreibung (kurz)
save	Speichert die Applikationskonfiguration in ein externes Dokument vom Typ JSON.
load	Ladet eine Konfiguration vom Typ JSON und initialisiert die RT_Settings Felder
reset	Stellt die Default-Werte für RT_Settings wieder her.

4.3.5 Modell

In diesem Abschnitt werden die Modell-Klassen betrachtet und Ihre wichtigsten Felder und Funktionen erläutert.

4.3.5.1 RT_LogWindow_Modell

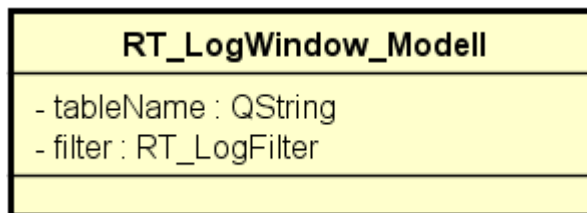


Bild 17: Klassendiagramm RT_LogWindow_Modell

Die Modell-Klasse für RT_LogWindow Logfenster ist sehr klein und enthält nur zwei Felder.

Felder	Beschreibung (kurz)
tableName	Name der Tabelle oder des Clients (Tabellenname normalerweise Clientname)
filter	Filterinstanz vom Typ RT_LogFilter, der zum Filtern der Sicht verwendet wird.

4.3.5.2 RT_Main_Modell

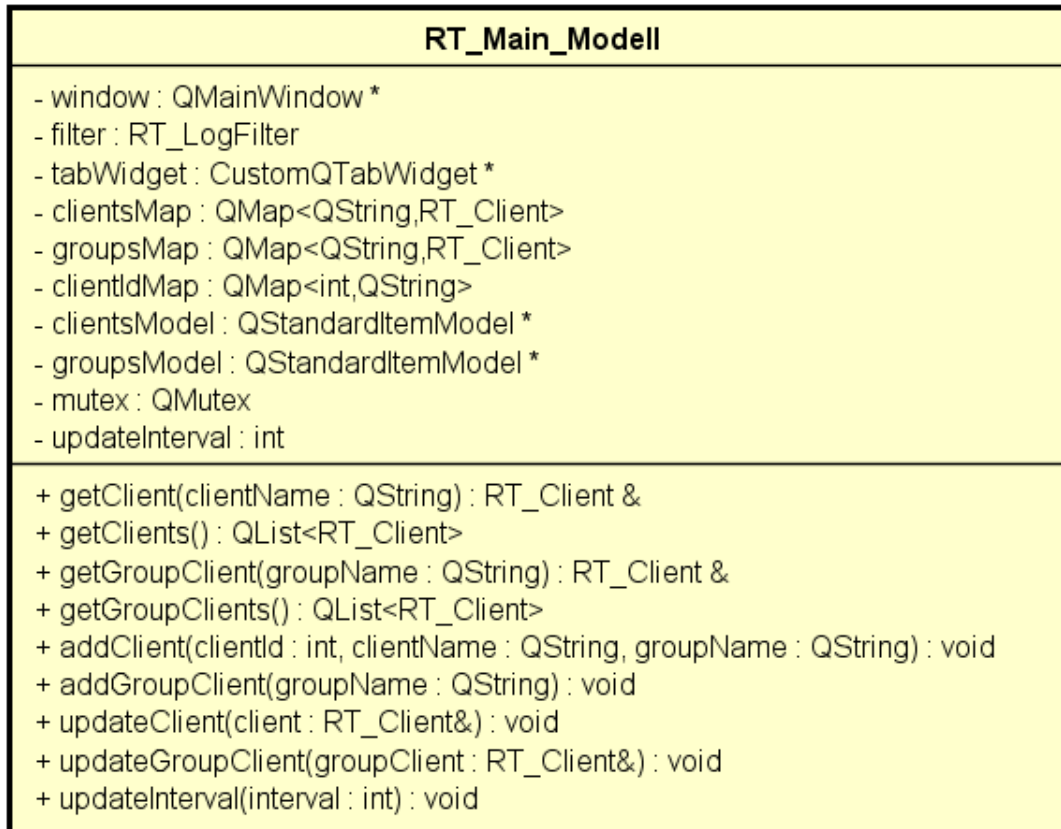


Bild 18: Klassendiagramm RT_Main_Modell

Die Modell-Klasse für RT_Main, der Hauptfensterklasse. Sie dient hauptsächlich zur Verwaltung von Clientobjekten.

Felder	Beschreibung (kurz)
window	Pointer auf die Fensterklasse RT_Main
filter	Filterinstanz vom Typ RT_LogFilter. Dieser Filter wird entweder zur Filterung der Sicht für die Clienttabellen benutzt.
mainContentGrid	Ist ein Qt Layoutelement und stellt den Bereich für die Logs dar, die in einem QTabWidget angezeigt werden.
tabWidget	Ein CustomQTabWidget, der nach Initialisierung dem mainContentGrid - Layout hinzugefügt werden. Clienttabs werden über diesen Pointer per tabWidget->addTab(...) hinzugefügt.
clientsMap	Ein Container vom Typ QMap<QString, RT_Client>, wobei der Schlüssel jeweils der Clientname ist.
groupsMap	Ein Container vom Typ QMap<QString, RT_Client> für Group Clients, wobei der Schlüssel jeweils der Group Clientname ist.
clientIdMap	Ein Container vom Typ QMap<int, QString>. Dieser Container enthält das Mapping von ClientId zu Clientname.
clientsModel	Ist ein Model vom Typ QStandardItemModel und stellt die Einträge für die Clientliste zur Verfügung.
groupsModel	Ist ein Model vom Typ QStandardItemModel und stellt die Einträge für die Group Client Liste zur Verfügung.
mutex	Ein Mutex Objekt vom Typ QMutex. Da bestimmte Felder des RT_Main_Modell von anderen Threads aus gelesen werden, müssen

	bestimmte Operation thread-safe gestaltet werden.
updateInterval	Updateinterval vom Typ int für die RT_UpdateWorker Klasse. Anfangswert ist üblicherweise 200 Millisekunden.
Funktionen	
getClient	Gibt eine RT_Client Referenz zurück, nimmt als Parameter den Clientnamen. Falls der Client nicht gefunden wird, wird die Referenz auf den globalen Client zurückgegeben.
getClients	Gibt eine List vom Typ QList<RT_Client> aller Clients zurück.
getGroupClient	Gibt eine RT_Client Referenz zurück, nimmt als Parameter den Group Clientnamen. Falls der Client nicht gefunden wird, wird die Referenz auf den globalen Client zurückgegeben.
getGroupClients	Gibt eine List vom Typ QList<RT_Client> aller Group Clients zurück.
addClient	Fügt einen RT_Client zur clientsMap hinzu. Diese Funktion initialisiert auch die Clienttabelle.
addGroupClient	Fügt einen RT_Client zur groupsMap hinzu. Diese Funktion initialisiert auch die Group Clienttabelle.
updateClient	Aktualisiert den angegebenen Client mit einem RT_Client Parameter.
updateGroupClient	Aktualisiert den angegebenen Group Client mit einem RT_Client Parameter.
updateInterval	Aktualisiert den updateInterval für den RT_UpdateWorker.

4.3.6 Worker

In diesem Abschnitt werden die Worker-Klassen erörtert.

4.3.6.1 RT_CaptureWorker

RT_CaptureWorker	
- modell : RT_Main_Modell	
- entries : QMap<QString, QList<QList<QString>>>	
+ handleLog(clientName : const QString &, groupName : const QString &, msg : const REP : :PROTO::LogMsg &) : void	
+ handleLog(clientName : const QString &, groupName : const QString &, msg : const REP : :PROTO::LogBinaryMsg &) : void	
+ signalLogsReady() : void	

Bild 19: Klassendiagramm RT_CaptureWorker

Der RT_CaptureWorker sammelt Logs die übers Netzwerk ankommen und sendet Sie beim Ankommen eines Updatesignals die angehäuften Logs

Felder	
modell	Referenz auf RT_Main_Modell, von dem der Wert updateInterval abgelesen wird. Dieser wird für die gestaffelten Updates gebraucht, um zu determinieren, wie lange zwischen Clientlogs pausiert werden muss.
entries	Angestaute Logs. Vom Typ QMap<QString, QList<QList<QString>>>. Hierbei wird als Schlüssel der Clientname und als Wert eine Liste von Logs aufbewahrt.
Funktionen	
handleLog	handleLog ist ein Slot, dass aufgerufen wird, wenn eine Lognachricht in RApplication ankommt und zum RT_Captureworker weitergeleitet wird. handleLog wird für Textlogs aufgerufen. handleLog bereitet die

	ankommenden Logs für die Clienttabellen auf und fügt Sie dem Buffer "entries" hinzu.
handleBinaryLog	handleBinaryLog ist ein Slot, dass aufgerufen wird, wenn eine Lognachricht in RApplication ankommt und zum RT_Captureworker weitergeleitet wird. handleLog wird für für binäre Logs (Logs, die als Binärinformationen festgehalten wird) aufgerufen. handleLog bereitet die ankommenden Logs für die Clienttabellen auf und fügt Sie dem Buffer "entries" hinzu.
signalLogsReady	Initialisiert gestaffelten Update Prozess, in dem die bisher angestauten Clientlogs im Buffer entries einzeln in einem bestimmten Abstand zum Hauptfenster weitergeleitet werden.

4.3.6.2 RT_UpdateWorker

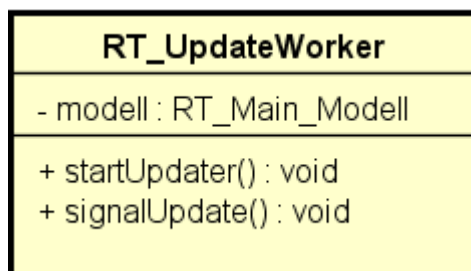


Bild 20: Klassendiagramm RT_UpdateWorker

Die Klasse RT_UpdateWorker ist ein einfacher Worker - Objekt, der in einer Schleife in einem bestimmten Intervall zum RT_CaptureWorker Signale zum Updaten sendet.

Felder	Beschreibung (kurz)
modell	Referenz auf RT_Main_Modell. Vom RT_Main_Modell wird der Wert updateInterval abgelesen.
Funktionen	Beschreibung (kurz)
startUpdater	Initialisiert Eventschleife für den UpdateWorker
signalUpdate	Signalisiert ein Update für RT_CaptureWorker.

4.3.6.3 RT_ToFileWorker

RT_ToFileWorker
- rx : QRegExp
+ adjustFile(file : QFile &, stream : QTextStream&) : void + fileLineCount(filename : const QString &) : int + readLogsFromFile(stream : QTextStream&) : QList<QString> + processLogToFile(clientName : const QString&, logs : const QList<QList<QString>>&, ... : boolean, sharedFileName : const QString&) : void + processLogFromFile(filename : const QString &) : void

Bild 21: Klassendiagramm RT_ToFileWorker

Der RT_ToFileWorker ist ein Worker - Objekt der Schreib- und Leseoperationen für Logs in Dateien ausführt.

Felder	Beschreibung (kurz)
rx	QRegExp mit Ausdruck «_D: _T: _G: _K: _M: _C:» zur Abfrage, ob eine Logentry zusätzliche Zeilen im File belegt.

Funktionen	Beschreibung (kurz)
adjustFile	Passt die Datei an für Ringfile Format. Der processLogToFile Funktion wird fügt die Loginformationen am Ende der Datei an, damit die Datei sich wie ein Ringfile verhält, werden bei ToRingFile Zustand eines Clients beim Überschritt des ringBufferLimit Wertes aus RT_Settings so lange Loginformationen vom Anfang der Datei entfernt, bis die Datei nur noch die Anzahl des ringBufferLimits an Loginformationen enthält.
fileLineCount	Zählt die Zeilen in einer Datei und gibt die Zahl als int zurück.
readLogsFromFile	Liest alle Einträge aus einer Logdatei und gibt diese als QList<QString> zurück.
processLogToFile	Schreibt Loginformationen in eine Datei. Der Funktion processLogToFile werden Client Zustandinformationen und Spaltensicherheitsinformationen überreicht, anhand derer processLogToFile seine Ausgabe anpasst. Diese Funktion nimmt als Parameter nebst clientName, logs und sharedFileName noch folgende Parameter: toFile (boolean), toRingFile(boolean) und toSharedFile(boolean)
processLogFromFile	Liest Einträge aus einer Logdatei und präpariert Sie für den Eintrag in eine Clienttabelle

4.3.7 Common

In diesem Abschnitt werden die der Gruppe der Common Utils zugehörigen freien Funktionen beschrieben. Die Gruppe der Common Utils enthält geteilte Funktionalität, die an verschiedenen Stellen der Applikation zum Einsatz kommen.

4.3.7.1 CommonLogUtils

<<namespace>> CommonLogUtils	
+ resolveMessageKind(kind : const REP::PROTO::MessageKind &) : QString	
+ parseLog(entry : const QString) : QList<QString>	
+ prepareLogEntry(clientName : const QString &, groupName : const QString &, msg : const REP::PROTO::LogMsg&) : QList<QString>	
+ prepareLogEntry(clientName : const QString &, groupName : const QString &, msg : const REP::PROTO::LogBinaryMsg&) : QList<QString>	

Bild 22: Klassendiagramm CommonLogUtils

Stellt freie Funktionen für Operation in Bezug auf Logs zur Verfügung.

Funktionen	Beschreibung (kurz)
resolveMessageKind	Der MessageKind einer REP::PROTO::LogMsg wird als ein int - Wert vermerkt. Diese Funktion mappt diesen Wert zu seiner String Repräsentation.
parseLog	Diese Funktion kommt beim Parsen einer Logdatei zu Logentries für eine Clienttable zum Einsatz und identifiziert die Loginformationen und mappt Sie zu Ihren zugehörigen Spalten.
prepareLogEntry	Diese Funktion konvertiert eine REP::PROTO::LogMsg oder REP::PROTO::LogBinaryMsg in eine für eine Clienttabelle entsprechende Repräsentation.

4.3.7.2 CommonSettingsUtils

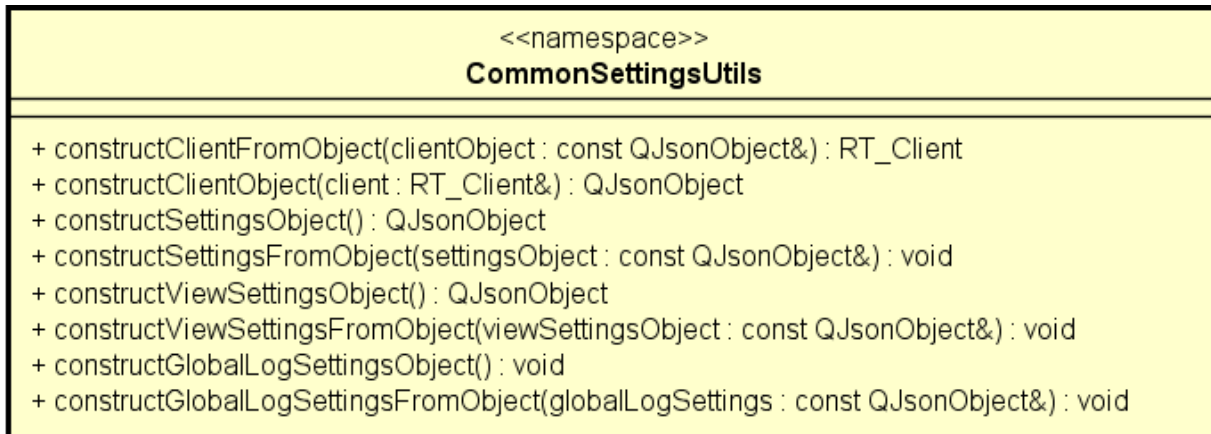


Bild 23: Klassendiagramm CommonSettingsUtils

Stellt freie Funktion für Operationen im Bezug auf die Settings (RT_Settings) zur Verfügung.

Funktionen	Beschreibung (kurz)
constructClientObject	Produziert ein QJsonObject aus einer RT_Client Referenz, die dann zum Schreiben in eine JSON - Datei verwendet wird.
constructClientFromObject	Generiert eine RT_Client Instanz von einem QJsonObject, der von einer JSON Konfigurationsdatei eingelesen wurde.
constructSettingsObject	Produziert ein QJsonObject aus der statischen RT_Settings Instanz, die dann zum Schreiben in eine JSON - Datei verwendet wird.
constructSettingsFromObject	Liest die Werte von einem QJsonObject und setzt die statischen Felder der statischen RT_Settings Instanz.
constructViewSettingsObject	Produziert ein QJsonObject aus der der statischen RT_Settings Instanz für das Feld viewSettings (RT_ViewSettings), der dann zum Schreiben in eine JSON - Datei verwendet wird.
constructViewSettingsFromObject	Liest die Werte von einem QJsonObject und setzt das Feld viewSettings (RT_ViewSettings) der statischen RT_Settings Instanz.
constructGlobalLogSettingsObject	Produziert ein QJsonObject aus der der statischen RT_Settings Instanz für das Feld logSettings (RT_LogSettings), der dann zum Schreiben in eine JSON - Datei verwendet wird.
constructGlobalLogSettingsFromObject	Liest die Werte von einem QJsonObject und setzt das Feld logSettings (RT_LogSettings) der statischen RT_Settings Instanz.

4.3.7.3 CommonViewUtils

<<namespace>> CommonViewUtils	
+ setupTable(tablename : boolean) : QTableView *	
+ initTabWidget(tabWidget : QTabWidget *) : QTabWidget *	
+ resetClient(client : RT_Client &) : void	
+ prepareRouteState(client : const RT_Client&) : QString	
+ prepareListEntry(client : const RT_Client&, tickImage : const QPixmap&, crossImage : const QPixmap&) : QList<QStandardItem*>	

Bild 24: Klassendiagramm CommonViewUtils

Stellt freie Funktion für Operation in Bezug auf Views zur Verfügung.

Funktionen	Beschreibung (kurz)
setupTable	Generiert eine Clienttabelle mit bestimmten Voreinstellungen.
initTabWidget	Initialisiert das QTabWidget, in der die Clienttabellen als Tabs angezeigt werden.
resetClient	Setzt den Client (RT_Client), der als Referenz übergeben wird, auf seine Default - Werte zurück.
prepareRouteState	Bereitet den Routestring für einen Client vor, der als Parameter übergeben wird. Dieser Routestring wird dann in der Clientliste/Groupliste zur Anzeige des Routezustands verwendet. Z.B. L für Live, P für Parent etc.
prepareListEntry	Bereitet den Listeintrag für einen Client/Group vor, der dann zur Anzeige in der Clients/Groups Liste verwendet wird. Die Funktion prepareListEntry gibt eine QList<QStandardItem*> zurück, wobei folgende "Augen" der Liste diese Bedeutungen innehalten: (1) Aktiv/Inaktiv - Icon, (2) Routestate, (3) Clientname

4.3.7.4 CommonStringUtils

<<namespace>> CommonStringUtils	
+ padString(inputString : const QString &, padAt : int, delimiter : const QString &) : QString	
+ adjustTimeString(value : int) : QString	
+ prepareTimeString(first : int, second : int, third : int, delimiter : const QString &) : QString	

Bild 25: Klassendiagramm CommonStringUtils

Stellt freie Funktion für Operation in Bezug auf Strings zur Verfügung.

Funktionen	Beschreibung (kurz)
padString	Diese Funktion padded einen String mit einem Delimiter Zeichen. z.B. "padme" kann anhand dieser Funktion mit dem Zeichen + erweitert werden zu "p+a+d+m+e"
adjustTimeString	Die Zeitwerte kommen als int - Werte in den Logs; diese Funktion past den int - Wert an und gibt einen QString zurück. Z.B. wird aus einer 1 der String "01", wobei der Wert 10, als String "10" bleibt.
prepareTimeString	Bereitet einen Zeitstring vor mit dem angegebenen Delimiter. z.B. die Werte 3, 3, 2017 (Tag, Monat, Jahr) mit Delimiter "-" gibt den String "3-3-2017" zurück.

4.3.8 CustomWidgets

In diesem Abschnitt werden auf die Custom Widgets eingegangen.

4.3.8.1 CustomTableModel

Die Klasse CustomTableModel wird als Model für die Clienttabelle, die eine Instanz von QTableView ist, verwendet. Der Grund für eine eigene Implementation ist Performanz. Mit dem CustomTableModel können mehrere Logs auf einmal hinzugefügt werden, ohne dass für jeden Eintrag ein eigenes Updatesignal generiert wird.

Felder	Beschreibung (kurz)
listOfEntries	Container zur Aufbewahrung der Logs in einer Clienttabelle. Dieses Feld ist vom Typ <code>QList<QList<QString>></code> .
Funktionen	Beschreibung (kurz)
appendRow	Fügt einen Logeintrag am Ende der Tabelle ein.
insertOnTop	Fügt einen Logeintrag am Anfang der Tabelle ein.
clearModel	Leert die Einträge einer Tabelle; listOfEntries wird geleert und so auch die Clienttabelleanzeige.

4.3.8.2 CutomFilterProxyModel

Ein Proxymodel, der zwischen der Clienttabelle und dem Model platziert wird, um Filteroperationen durchzuführen. So kann eine Clienttabelle einfach gefiltert werden, ohne das Einträge gelöscht und wieder neueingefügt werden müssen.

Felder	Beschreibung (kurz)
filter	Filterinstanz vom Typ <code>RT_LogFilter</code> , anhand der die Tabelleneinträge gefiltert werden.
Funktionen	Beschreibung (kurz)
filterAcceptsRow	Funktion zum Determinieren, welche Logs angezeigt werden sollen und welche nicht. Benutzt intern die Instanz filter vom Typ <code>RT_LogFilter</code> .
getProxyModelContent	Gibt die sichtbaren Einträge als <code>QList<QList<QString>></code> zurück. So kann man nur die gefilterten Logs exportieren und nicht die ganze Tabelle.

4.3.8.3 CustomQTabWidget

Eine sehr einfache, eigene QTabWidget Implementation, in der lediglich nur die für Drag'n'Drop relevanten Funktionen neu implementiert werden. Dadurch wird die Drag'n'Drop Funktionalität für QTabWidget bereitgestellt.

4.3.8.4 CustomWindowQTabWidget

Eine sehr einfache, eigene QTabWidget Implementation für Logfenster, in der lediglich nur die für Drag'n'Drop relevanten Funktionen neu implementiert werden. Dadurch wird die Drag'n'Drop Funktionalität für QTabWidget in Logfenster bereitgestellt.

4.4 Design

In diesem Abschnitt sollen die verschiedenen Designentscheidungen für die Applikation beschrieben werden. Von architekturellen Designentscheidungen zu GUI bezogenen Designentscheidungen; werden diese hier in Detail erläutert und erörtert.

4.4.1 MVC

Am Anfang des Projekts wurde das Konzept des Model-View-Controller Designpattern diskutiert und für die Applikation als passend determiniert. Hierzu wurde folgendes Schema in Bild 26 verwendet:

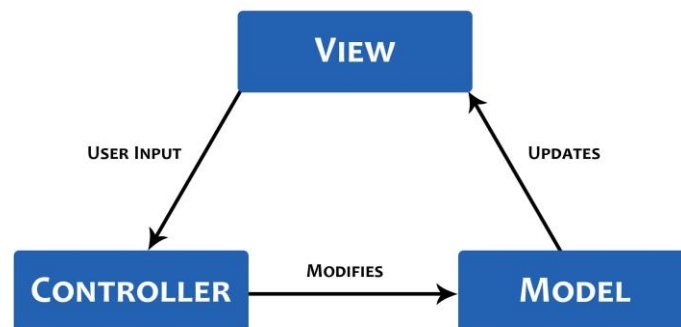


Bild 26: MVC Pattern

Dafür wurde pro Fensterklasse je ein Model und ein Controller definiert. Die eigene Implementierten des Model-View-Controller Patterns erwies sich anfänglich als beschwerlich und mühsam, weil bestimmte Aspekte des Qt Frameworks in Kombination zu Overhead führten. Im Laufe der Einarbeit in das Qt Framework wurde jedoch dann eine bessere Alternative gefunden. Das Qt Framework verwendet eine eigene Variante des MVC Pattern, dass unter Model/View Programming¹ in der Dokumentation zusammengefasst wird und die Umsetzung des MVC Patterns um Einiges vereinfacht, wie in Bild 27 zu sehen ist.

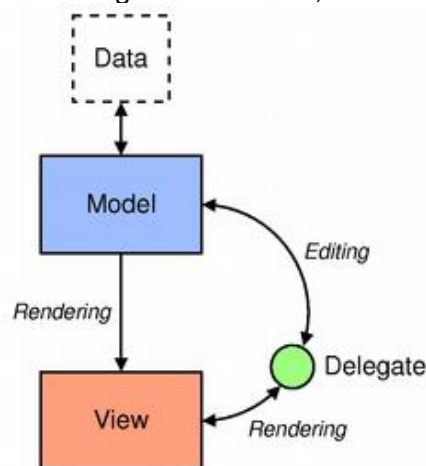


Bild 27: Model/View Konzept Qt Framework

Hierbei fungieren die Fensterklassen selbst als Controller. Enthält ein Fenster viel Funktionalität, so kann das die Fensterklasse schnell aufblähen und den Code

¹ <http://doc.qt.io/qt-5.6/model-view-programming.html>

unübersichtlich gestalten. Man könnte diesen Umstand mit bereichsspezifischen, eigenen Controllern umgehen, was jedoch aus Zeitmangel für dieses Projekt nicht umgesetzt werden konnte.

4.4.2 Leistung und Multithreading

Damit das UI auch unter Last responsive bleibt, wurden bestimmte Bereiche der Applikation für eine konstante Leistung mit Multithreading umgesetzt.

4.4.2.1 Client - Server Kommunikation

Der Master - Thread (Server) hört auf einem bestimmten Port, über den sich die Clients zum ReportingGUI verbinden, auf eingehende Verbindungen und initialisiert für jeden Client einen eigenen Session - Thread. Ein Session - Thread enthält eine RApplication - Instanz, der die Applikation abbildet, zu denen die Clients (Channel) gehören. In der Eventschleife der RApplication werden die Lognachrichten der Clients verarbeitet und weitergeleitet. Nach Aufbau des Session - Threads findet die Kommunikation direkt und ohne Umwege statt. Das heisst, die im Session - Thread erzeugte RApplication wird der Pointer `CL::TCPSocket*` überreicht, also Pointer des Sockets, über die die Verbindung hergestellt wurde. Die RApplication determiniert innerhalb seiner Schleife den Typ der ankommenden Messages und sendet entsprechend Änderungen an die Clients zurück, z.B. im Rahmen von "keep alive" Messages, werden `ClientStatusChangeMessage's` ausgetauscht, in der allfällige Client Änderungen kommuniziert werden. Die Architektur der beschriebenen Kommunikation wird in Bild 28 wiedergegeben.

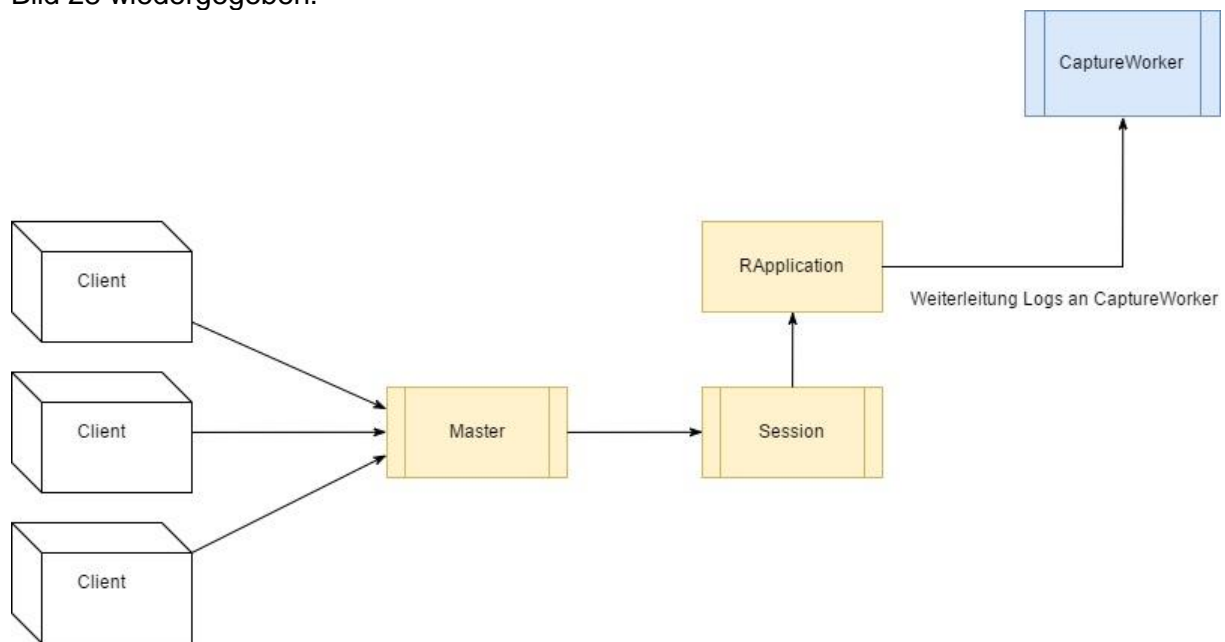


Bild 28: Verbindungsaufbau und Logverarbeitung Schema

Sind die Lognachrichten abgearbeitet und die Zugehörigkeit der Clients festgestellt, werden die Logs mit Clientname an den CaptureWorker weitergereicht, wo Sie bis zum Update Signal gesammelt werden.

4.4.2.2 View Update Signalisierung

Die untere Intervallgrenze, in der Logs gemäss Anforderung aufeinander eintreffen können, liegt bei 20 Millisekunden. Finden in diesem Intervall View Updates statt, sprich jedes ankommende Log verursacht einen View Update, und sind 10+ Client Tabellen gleichzeitig sichtbar - was bedeutet, es müssen 10+ Views aktualisiert werden - so kommt Qt mit den View Updatesignalen nicht nach und eine Performanzeinbusse wird spürbar. Das GUI wird unansprechbar, weil sich Updatesignale aufstauen, inkl. Signale des Benutzers wie Mausklick - Ereignisse. Dies hängt stark mit der Funktionsweise von Qt ab. So müssen für die Kommunikation zwischen Threads das Signals and Slots Konzept benutzt werden, wie Bild 29 illustrieren soll.

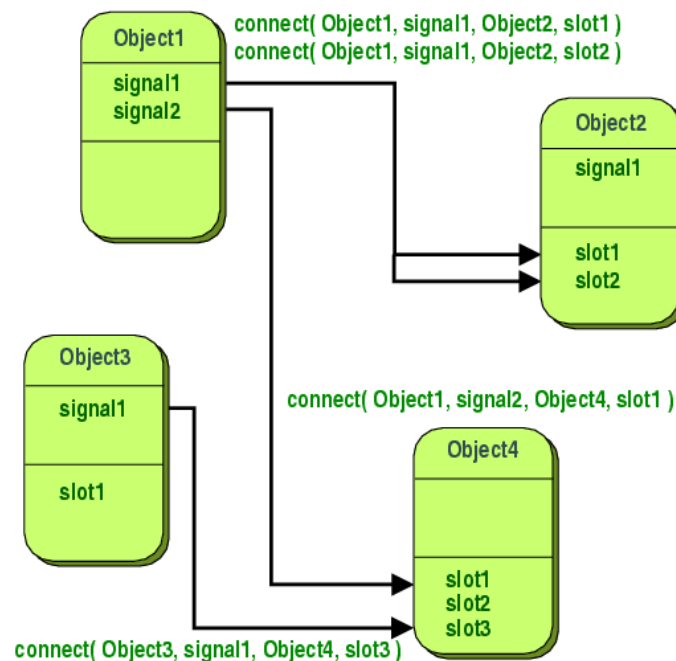


Bild 29: Qt Signals and Slots Konzept

Damit die Operationen auch wirklich in Ihren eigenen Threads laufen, muss man zusätzlich den Verbindungstyp der Signale zu den Slots als `Qt::QueuedConnection` definieren. Dieser Verbindungstyp platziert die Signale in einer Queue und sobald die Eventschleife "frei" wird, wird die Queue abgearbeitet. In Bild 30 soll das anhand eines Beispiels aus der Applikation dargestellt werden.

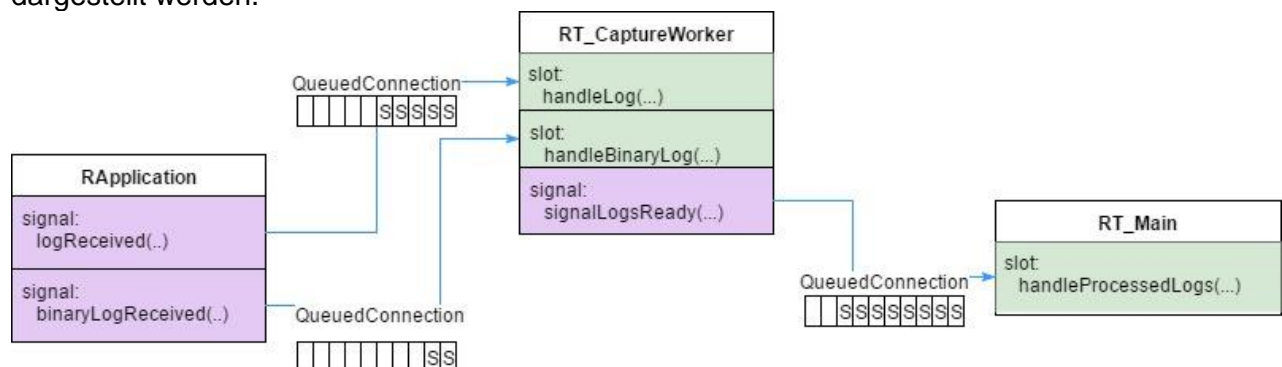


Bild 30: Qt Signals and Slots Konzept am Beispiel Loghandling

Damit sich die Updatesignale nicht aufstauen, wurde folgende Architektur in Bild 31 entworfen:

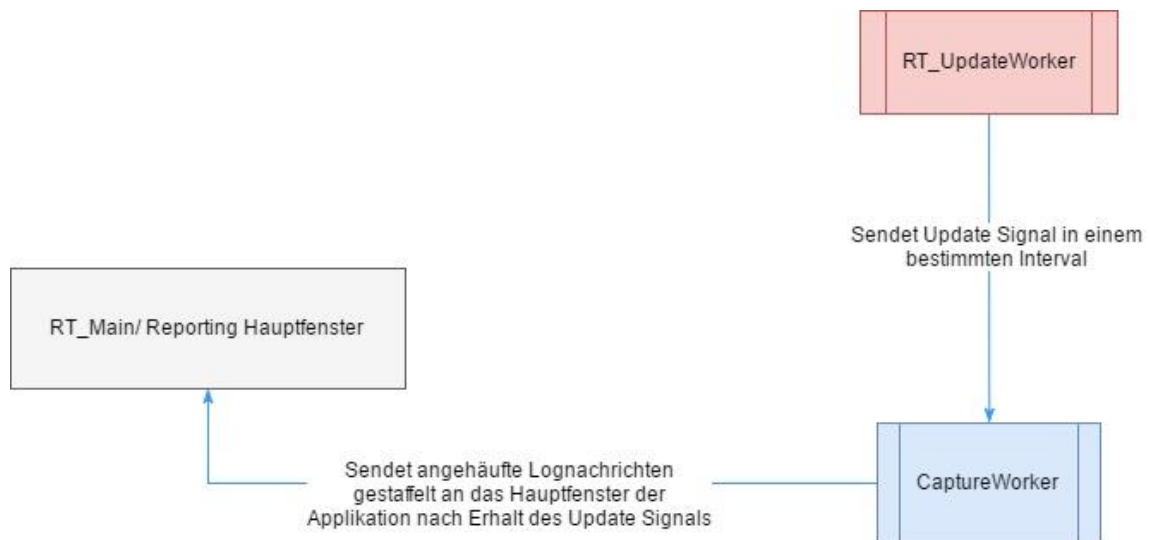


Bild 31: Gestaffelte View Updates Schema

Wobei man der Signalisierungsprozess zwischen UpdateWorker, CaptureWorker und RT_Main, der Hauptfensterklasse weitere Erklärung benötigt.

Der UpdateWorker ist in einem eigenen Thread mit einer einfachen while - Schleife, in der es in einem bestimmten Intervall (updateInterval) den CaptureWorker zum Senden seiner angehäuften Lognachrichten signalisiert. Dieses Intervall ist anpassbar und wird mit der Anzahl Fenster in 50ms Sprüngen erhöht. In Bild 32 wurde versucht, diesen Prozess als Sequenzdiagramm aufzuzeigen.

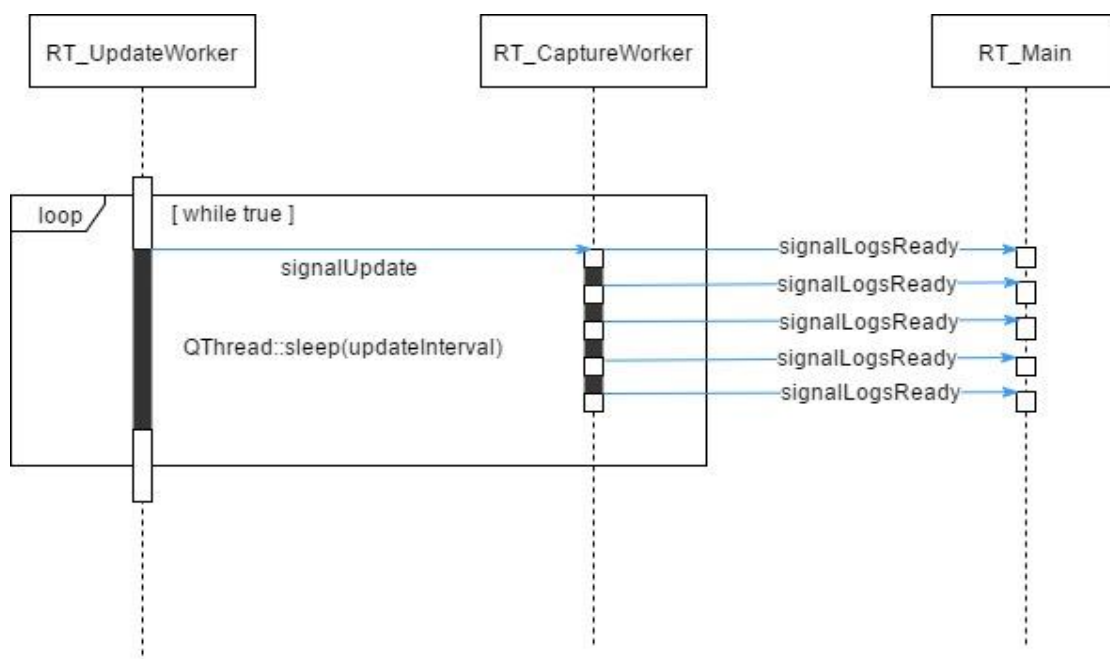


Bild 32: Sequenzdiagramm UpdateWorker Signalisierungsprozess

Der CaptureWorker befindet sich ebenfalls in einem separaten Thread und häuft bis zum Erhalt des Updatesignals die Logs in einer internen Map, mit dem Clientnamen als Schlüssel und als Wert eine Liste von Logs. Wird ein Update seitens des UpdateWorkers ausgelöst, so wird die ganze Map gestaffelt geleert. Dies will heissen, dass innerhalb des aktuellen Updateintervalls in einem gleichmässigen Abstand bis zum nächsten Intervall je ein Client mit seinen Logs zum UI zum Update geschickt werden.

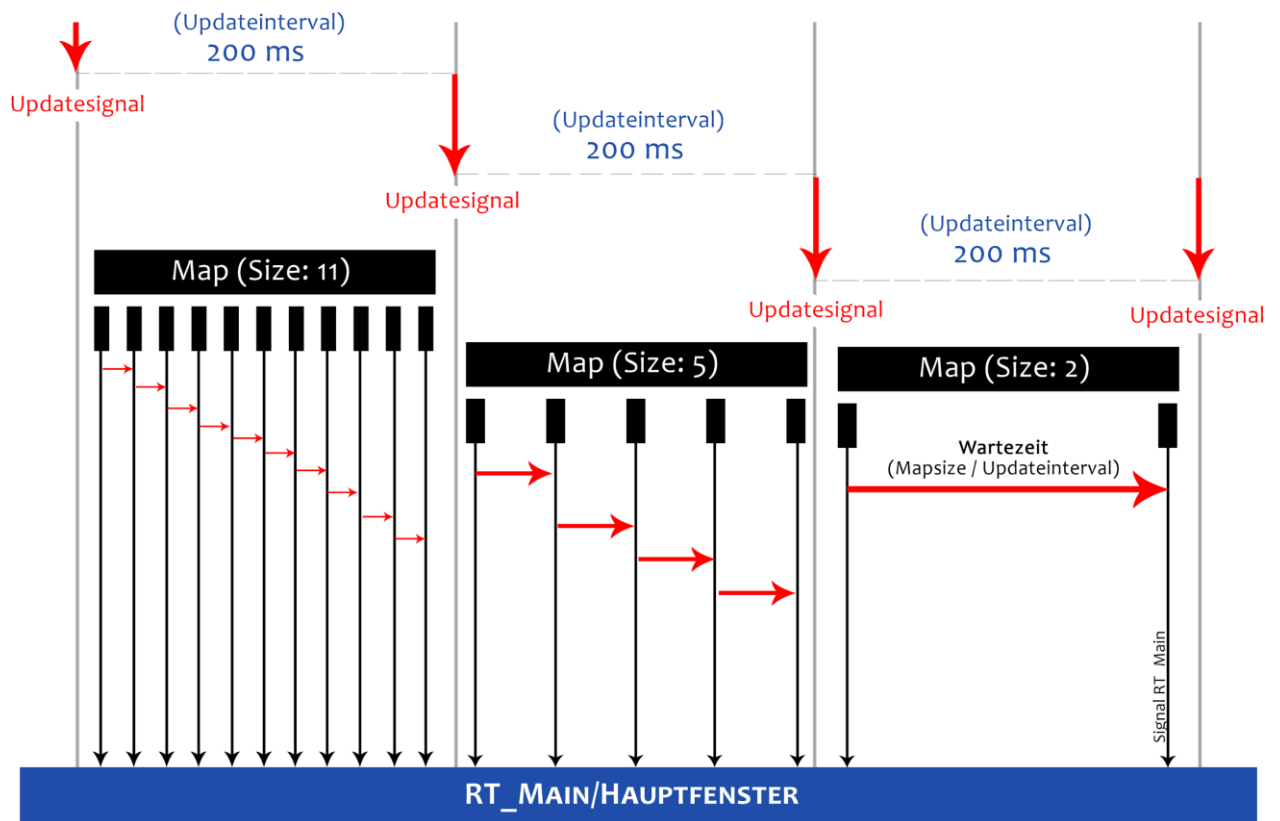


Bild 33: Diagramm gestaffelte Updates

Wie in Bild 33 ersichtlich, wird auf diese Weise sichergestellt, dass das UI nicht auf einmal von zu vielen Signalen blockiert wird. Damit wird ein responsives GUI gewährleistet.

Zur Vervollständigung soll in Bild 34 noch das komplette Diagramm aufgezeigt werden, von der Client - Server Verbindung zum Anhäufen der Logs, erhalten des Updatesignals und der Weiterreichung zum RT_Main/Hauptfenster.

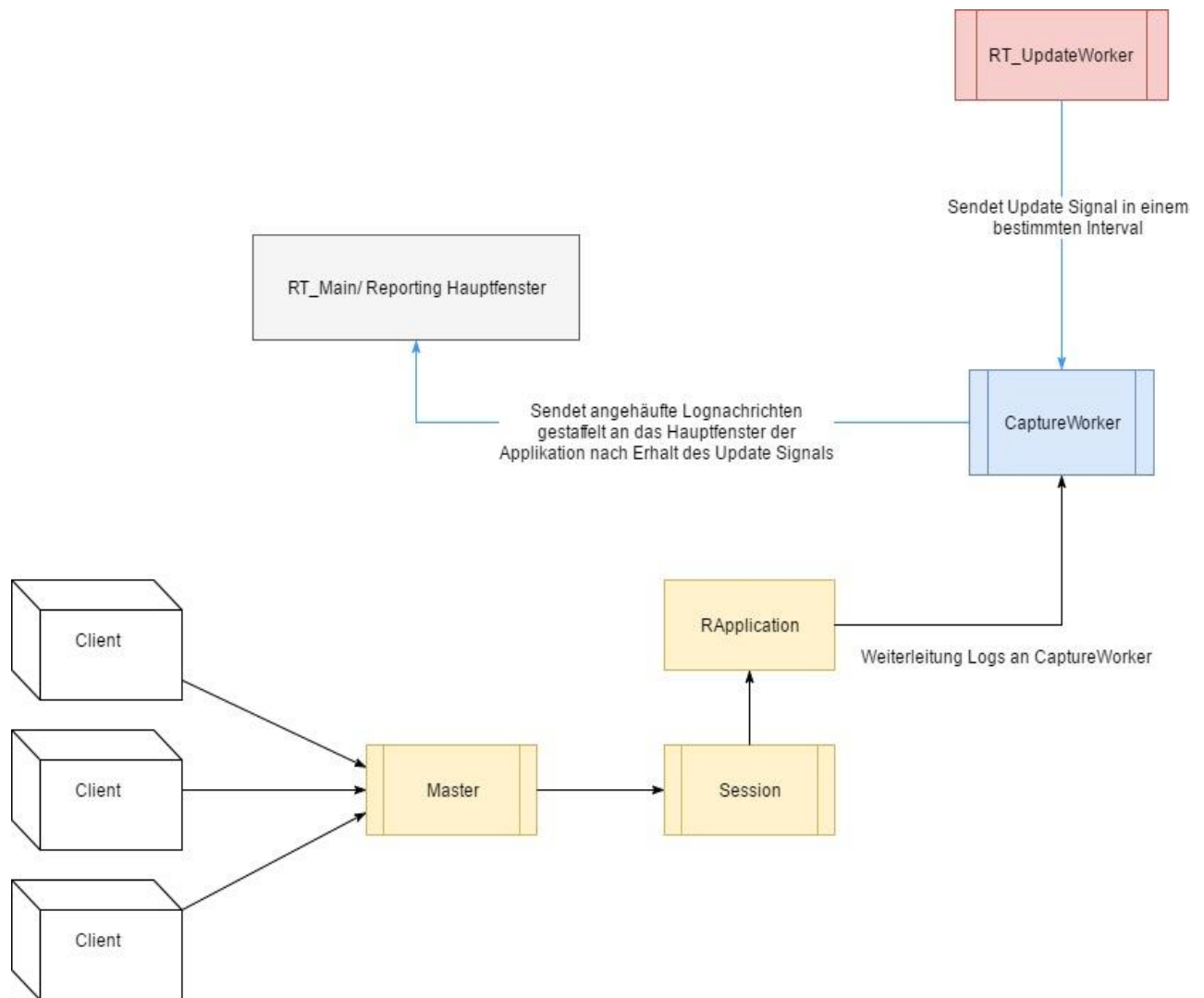


Bild 34: Gesamter Loghandling Zyklus

4.4.2.3 File Routing

Das File Routing in der Applikation stellt den Prozess der Ausgabe von Loginformationen in eine Datei dar. Dabei können insgesamt drei verschiedene Arten der Ausgabe zustande kommen.

1. *ToFile*
Die Loginformationen werden einfach in eine Datei geschrieben.
2. *ToRingFile*
Die Loginformationen werden in eine Ringdatei geschrieben. Wenn eine bestimmte Anzahl Logs erreicht wird, werden die ältesten Logs aus dem File kontinuierlich entfernt.
3. *ToSharedFile*
Die Loginformationen werden in eine geteilte Datei geschrieben.

Vor allem wenn die Anzahl gleichzeitiger Clients, die in einer Datei routen, eine bestimmte Anzahl erreicht, kann dies zum Einfrieren des GUI führen. Deswegen wurde das File Routing per Multithreading umgesetzt. Aber im Fall des File Routings genügt ein einzelner Thread nicht aus, vor allem wenn in mehrere Ringfiles oder viele Dateien gleichzeitig geroutet wird, stauen sich die Signale wegen der Bearbeitungsdauer auf und der Speicherverbrauch steigt schlagartig in die Höhe. Um dieses Problem zu umgehen, werden von Anfang an vier Threads auf Standby gestartet und die Signale zirkulär auf die Threads verteilt (siehe Bild 35). So bleibt das GUI responsiv und die Threads werden nicht gleichzeitig zu stark belastet.

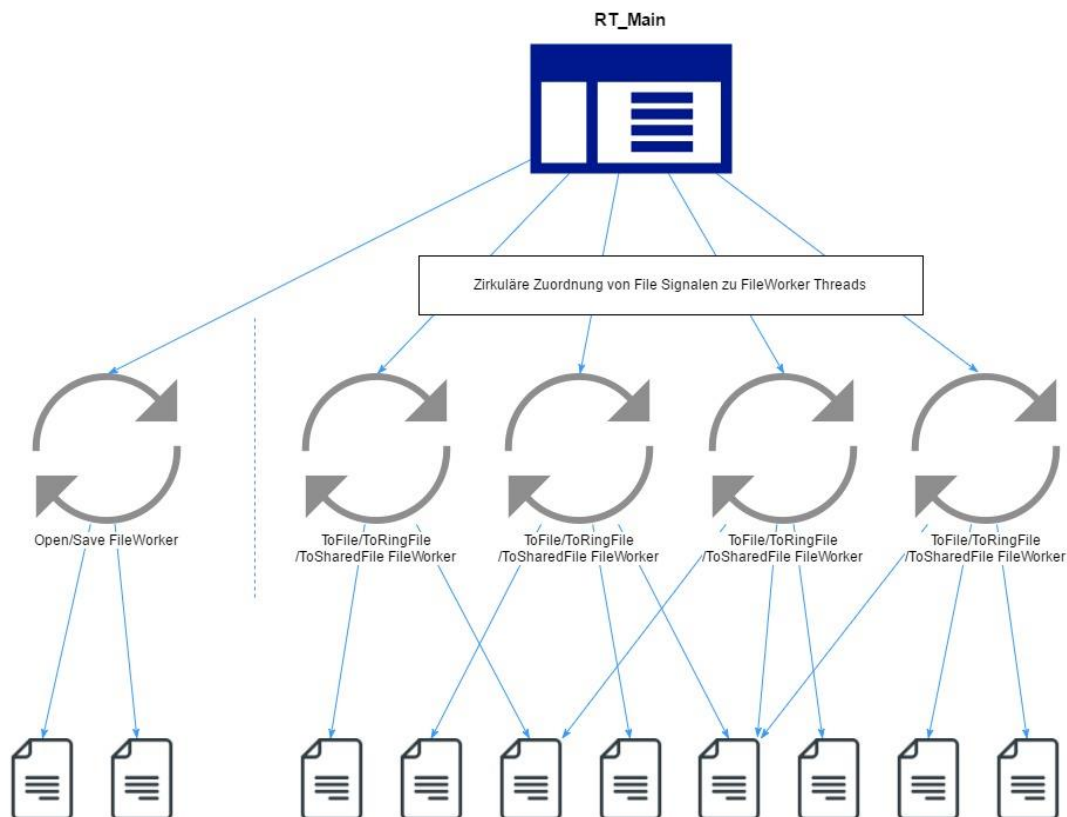


Bild 35: File Routing Multithreading Diagram

4.4.3 Logfile Parsing

Da beim Routen in Files die Logdaten gemäss den sichtbaren Spalten in die Dateien geschrieben werden, müssen die Loginformationen speziell aufbereitet werden, damit beim Parsing die Spalten eindeutig identifiziert werden können (siehe Bild 36). Ausserdem müssen für mehrzeilige Contenteinträge spezielle Vorkehrungen getroffen werden, damit die zusätzlichen Zeilen nicht als separate Logs interpretiert werden.

```
_D: 2017-04-30 _T: 16:43:20 _G: TEST_GroupName0 _K: ClientName0 _M: Warning _C: Lorem ipsum
dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem
ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet
_D: 2017-04-30 _T: 16:43:20 _G: TEST_GroupName0 _K: ClientName0 _M: Info _C: Lorem ipsum
dolor 184 sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet
_D: 2017-04-30 _T: 16:43:20 _G: TEST_GroupName0 _K: ClientName0 _M: Warning _C: Lorem ipsum
dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem
ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet
_D: 2017-04-30 _T: 16:43:20 _G: TEST_GroupName0 _K: ClientName0 _M: Info _C: Lorem ipsum
dolor 185 sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet
Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet
_D: 2017-04-30 _T: 16:43:20 _G: TEST_GroupName0 _K: ClientName0 _M: Warning _C: Lorem ipsum
dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem
ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet
```

Bild 36: Log Dateiansicht

Hierzu werden bei der Ausgabe die jeweiligen Spalteninformationen mit einem Identifier gekennzeichnet und anhand dieser beim Lesen der Datei die Spalten eindeutig identifiziert. Logeinträge, die mehrere Zeilen umfassen, werden beim Lesen zusammengefasst und der Contentspalte zugewiesen.

Auf diese Weise kann die Spaltenzahl im Laufe des Programms dynamisch angepasst werden ohne dass es das Log Parsing beeinflusst.

4.4.4 Persistenz

JSON wurde für seine Einfachheit und Lesbarkeit als Format für die Konfigurationsdatei gewählt. Zwar bestand eine eigene Implementation seitens der Reporting Library, Informationen in einer Konfigurationsdatei zu schreiben, jedoch konnte dieser nicht in kurzer so angepasst werden, dass dieser die Bedürfnisse der Applikation erfüllen konnte.

Es könnte allenfalls in der Zukunft eine Alternative zum JSON - Format bilden, falls die Reporting Library interne Struktur bevorzugt werden würde.

In seinem jetzigen Zustand erlaubt die Applikation das Persistieren von bestimmten Applikationseigenschaften, wie z.B. Clients und Groups mit Ihren Zuständen, globale Einstellungen wie File Routing Lokationen, globale Logeinstellungen, wie z.B. was für Logarten die Clients clientseitig Loggen sollen und bestimmte Viewkonfigurationen, wie z.B. ob der Logdetailinput sichtbar ist oder nicht. Ausserdem kann man die Autoregistration von Clients/Groups persistieren. Falls man nur mit einem bestimmte Subset von Clients und Groups arbeiten will, muss die Autoregistration ausgeschaltet sein. So werden dann RegistrationMessages ignoriert und die Clients werden nicht per Netzwerknachricht aktualisiert und man hat nur die Clients/Groups, die man per Logfile geladen hat.

```
"Clients": {
  "ClientName0": {
    "active": true,
    "binary": false,
    "clientId": 1,
    "clientName": "ClientName0",
    "color": "#ffffff",
    "groupName": "TEST_GroupName0",
    "live": false,
    "mergeViewName": "",
    "newestOnTop": false,
    "scrollToBottom": true,
    "sharedFileName": "",
    "toFile": true,
    "toGlobal": false,
    "toMergeView": false,
    "toParent": false,
    "toRingFile": false,
    "toSharedFile": false,
    "windowed": false
  },
},
"Groups": {
  "TEST_GroupName": {
    "active": false,
    "binary": false,
    "clientId": -2,
    "clientName": "TEST_GroupName",
    "color": "#ffffff",
    "groupName": "Global",
    "live": false,
    "mergeViewName": "",
    "newestOnTop": false,
    "scrollToBottom": true,
    "sharedFileName": "",
    "toFile": false,
    "toGlobal": false,
    "toMergeView": false,
    "toParent": false,
    "toRingFile": false,
    "toSharedFile": false,
    "windowed": false
  },
},
"Settings": {
  "autoRegistrationAllowed": true,
  "baseFileSaveLocation": "C:/Users/Zmote/Downloads/artifacts/bin/logs",
  "currentConfigurationFileName": "testConfig",
  "logBufferLimit": 10000,
  "messageBufferLimit": 200,
  "ringBufferLimit": 10000,
  "toFileSaveLocation": "C:/Users/Zmote/Downloads/artifacts/bin/logs",
  "toRingFileSaveLocation": "C:/Users/Zmote/Downloads/artifacts/bin/logs",
  "toSharedFileSaveLocation": "C:/Users/Zmote/Downloads/artifacts/bin/logs"
},
}
```

Bild 37: Ausschnitt Konfigurationsdatei

Wie man in Bild 37 leicht erkennt, sind bestimmte Konfigurationen gruppiert. JSON erlaubt das Verschachteln von JSON - Objekten, so können die Konfigurationen beliebig erweitert werden und man kann eine gute semantische Gruppierung erreichen. Zudem erlaubt das JSON Format das leichte Anpassen von Einstellungen, da die Einstellungen einfach zu lesen und editieren sind.

Wird eine Konfiguration gespeichert, so wird Sie im System hinterlegt und beim nächsten Start der Applikation, mit dieser Konfiguration geladen. Dies wird mit der QSettings - Klasse umgesetzt, die plattformunabhängiges persistieren von Einstellungen im System ermöglicht. Auf Windows z.B. wird die aktuelle Konfiguration, wie in Bild 38 dargestellt, in der Registry niedergeschrieben.

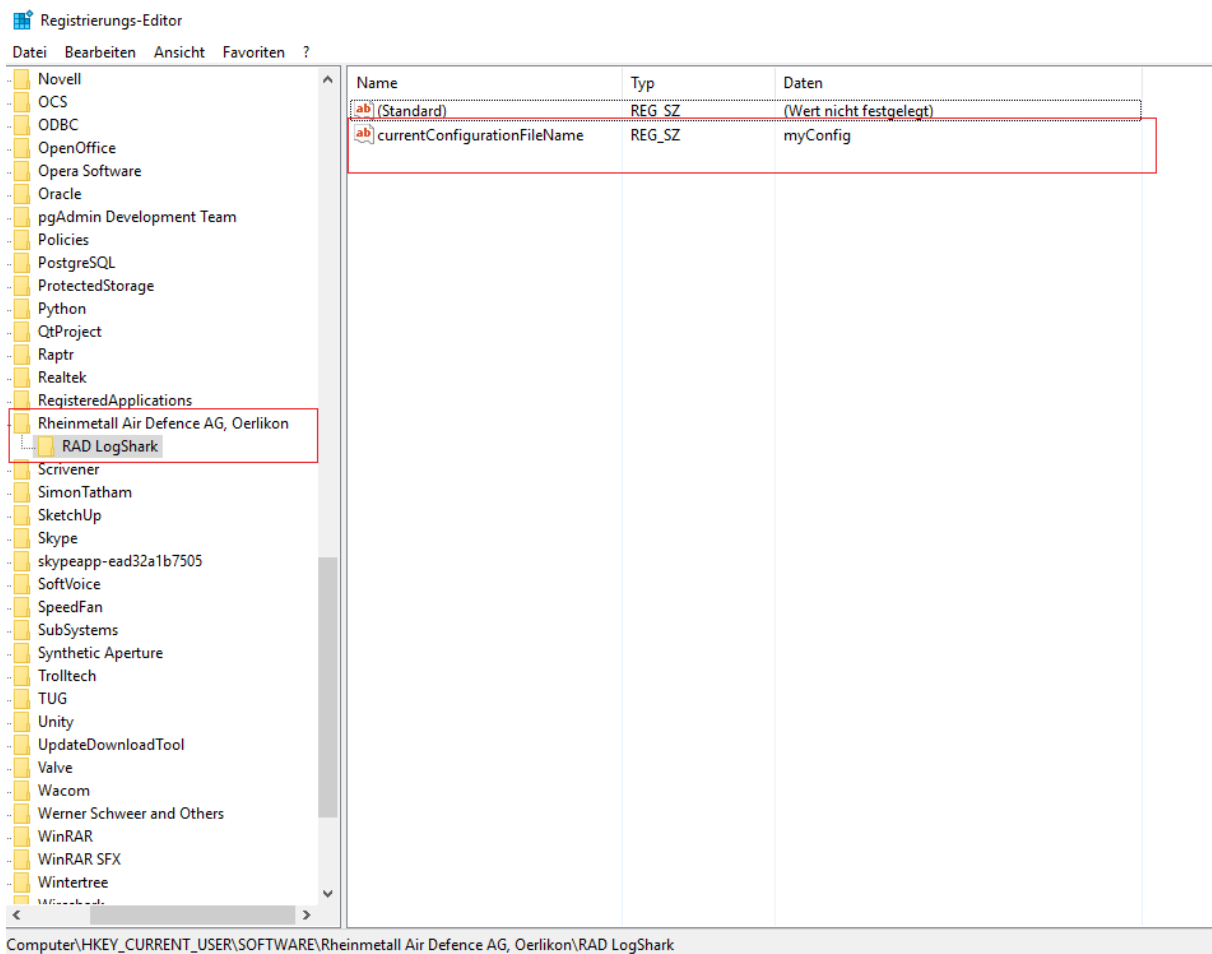
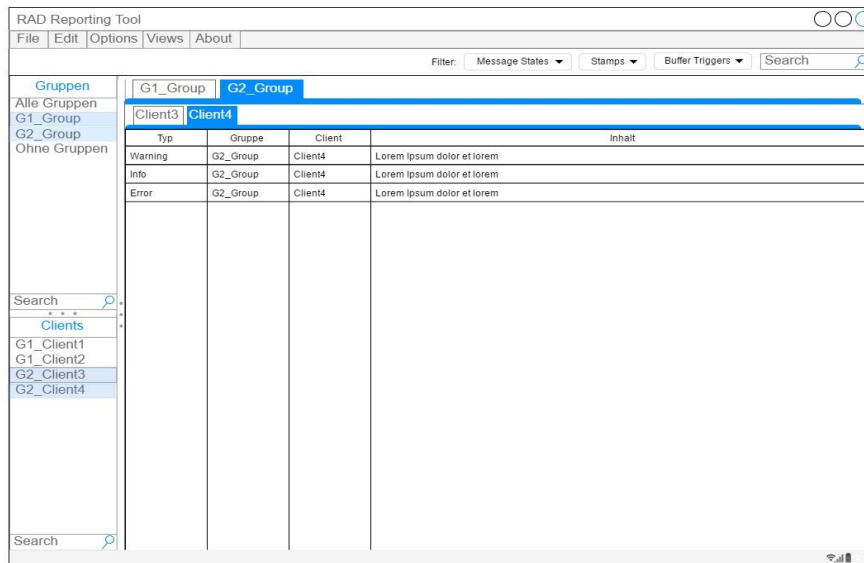


Bild 38: Ausschnitt Registry Windows

4.4.5 GUI

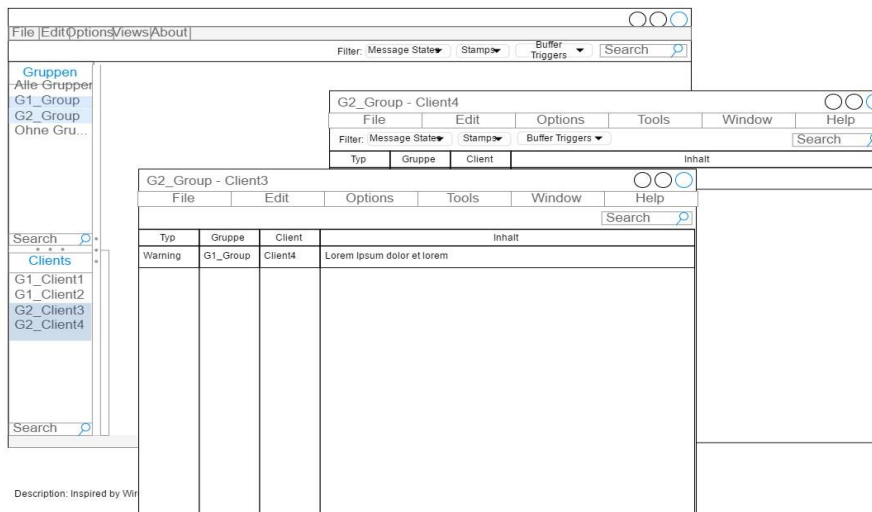
Das Design des User Interfaces durchging einige Iterationen bis zur finalen Version. Die anfängliche Herausforderung bei der Konzeption des UI bestand in der Frage: Wie gliedere ich die verschiedenen Elemente zur Benutzerinteraktion mit den Logs und biete gleichzeitig so viel Übersicht wie möglich? Glücklicherweise hatte der Industriepartner schon eine grobe Vorstellung davon, wie die Applikation ungefähr aussehen könnte und das Programm Wireshark wurde zur Hauptinspirationsquelle. Anhand dieser Inspiration entstanden die ersten Konzepte, wie im Bild 39 aufgezeigt.

Main Window - Client View Tab View



Description: Inspired by Wireshark

Main Window - Client View - Floating Windows



Description: Inspired by Wireshark

Bild 39: Erste Konzepte UI

Die Hauptidee bestand darin, eine einfache, klare Übersicht über die Groups und Clients zu präsentieren. Über den Toolbar sollte der Benutzer bequem die Clientzustände der selektierten Clients ändern können, wobei das Toolbar sich dynamisch an die selektierten Clients anpasste.

Dieses UI Konzept, oder mindestens die Idee dahinter, wurde fast bis zum Release befolgt und umgesetzt. In der Alphaversion sah das UI dann wie im Bild 40 aus.

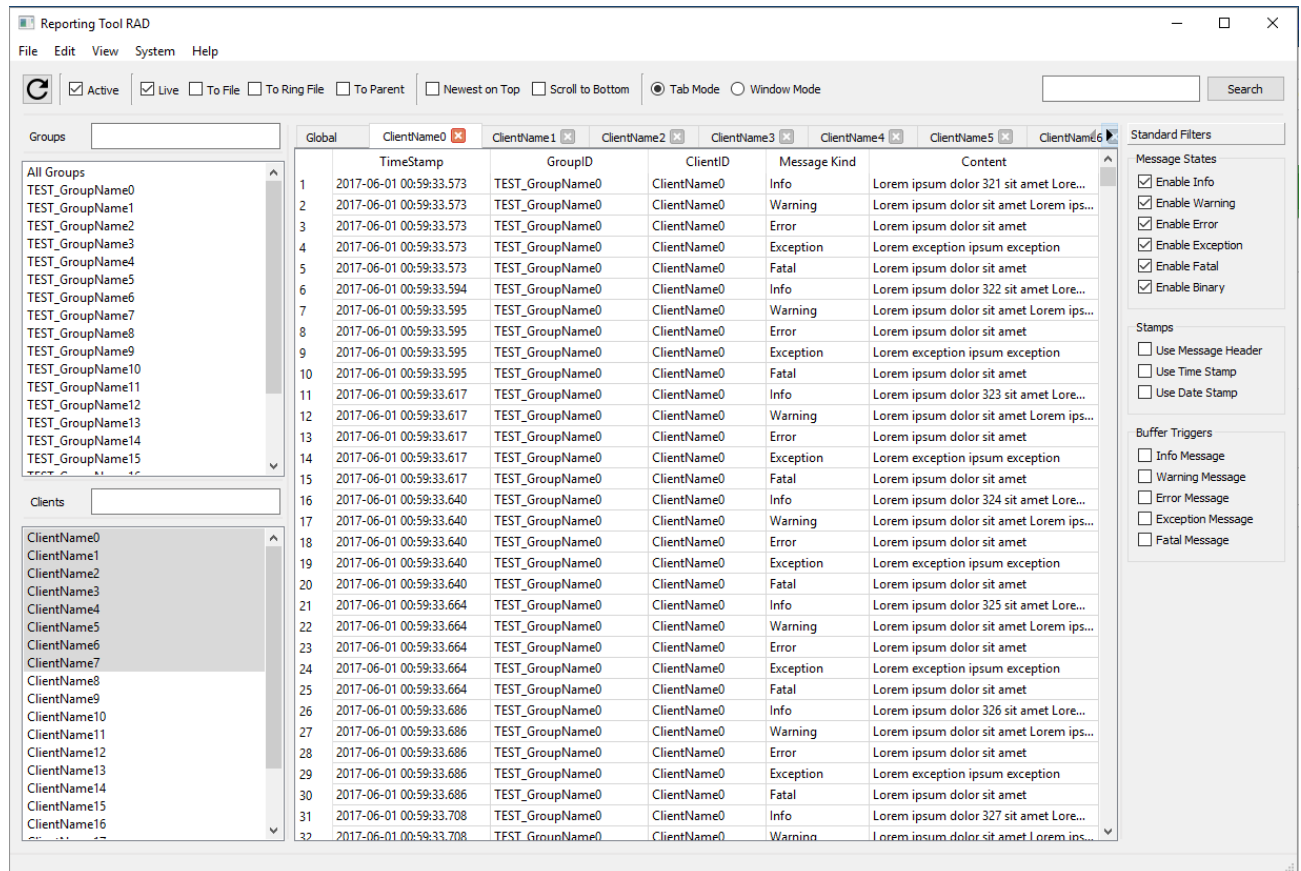


Bild 40: Alphaversion UI

In der Alphaversion waren die Client - Zustände noch nicht in der Listenansicht sichtbar. Zudem bestanden Probleme beim Update der Clients, da die Updates nur auf selektierte Clients angewandt wurde und die Clientliste den Fokus brauchte. Verlor die Clientliste den Fokus, wurden die Clients nicht aktualisiert. Jedoch brauchte es die Unterscheidung mit Fokus, da gemäss diesem der Toolbars aktualisiert wurde (z.B. welche Checkboxes angezeigt wurden oder nicht).

Diese Problematik wurde vorerst mit der Deaktivierung des Toolbars zu lösen versucht, wenn die Client- oder Groups Liste den Fokus verlor. Für eine Weile schien dieses Konzept zu funktionieren, und im Betaversion sah das UI dann wie im Bild 41 aus.

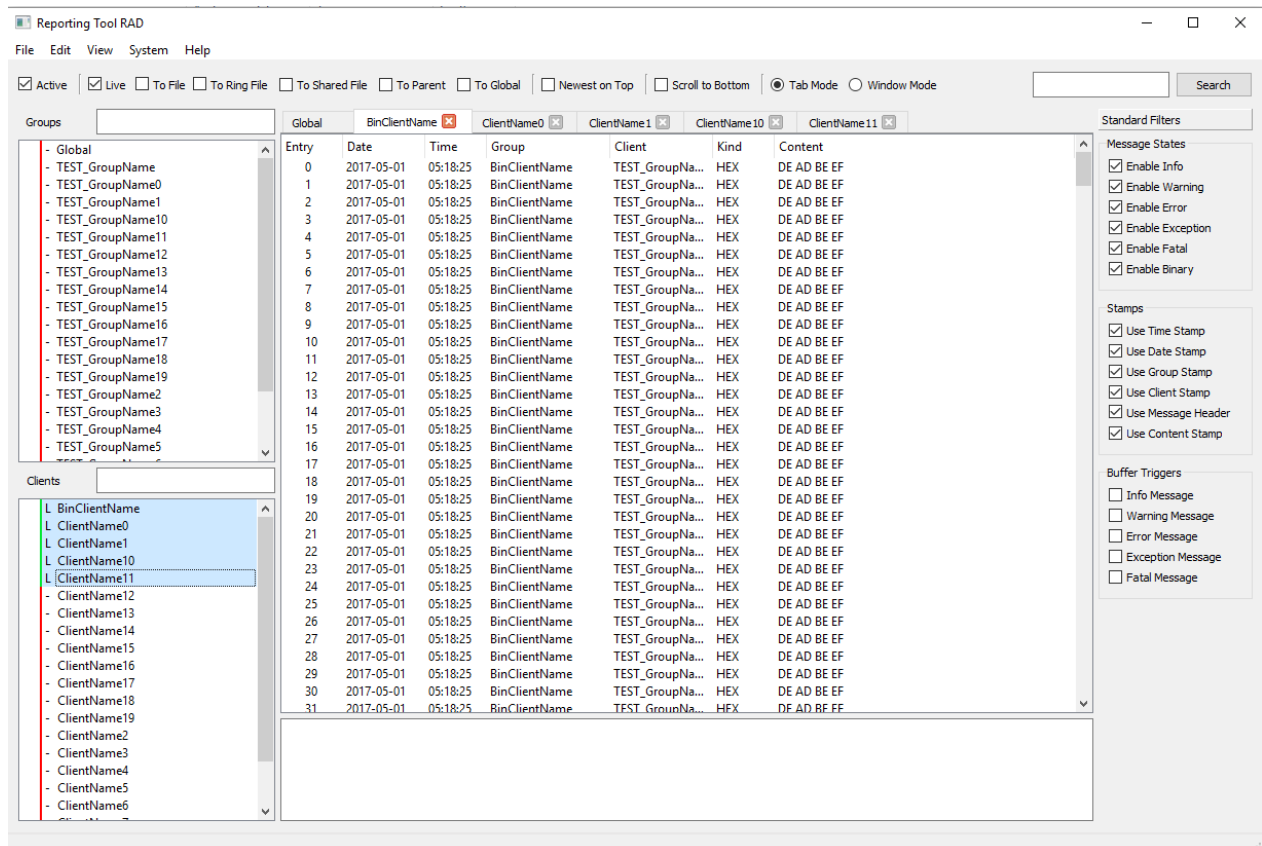


Bild 41: Betaversion UI

In der Betaversion wurden nach Feedback des Industriepartners auch die Clientzustände in die Listenanzeige aufgenommen. Ausserdem kamen neue Checkboxes hinzu, was ein neues Problem darstellte. Wegen der Anzahl Checkboxes konnte das UI bis zu einer bestimmte Breite verkleinert werden und nicht weiter. Ausserdem wurden die Clienteneinstellungen mit der steigenden Anzahl Checkboxe immer unübersichtlicher. Zudem wurde zu diesem Zeitpunkt der Logfilter implementiert, welches mit Filterausdrücken die Filterung von Logs erlaubte. Die Filterausdrücke konnten schnell einen grösseren String bilden und passten nicht in das kleine Logfilterinput - Feld, was es erschwerte, längere Filterausdrücke zu schreiben. Nach einem weiteren Feedback seitens des Industriepartners bei einem der Meetings, die alle zwei Wochen stattfanden, wurde die Deaktivierung des Toolbars beim verlorenen Client/Group - Listenfokus bemängelt. Ausserdem wurde die Spalte "Entry" als überflüssig bezeichnet.

An diesem Zeitpunkt wurde klar, dass das UI einer grösseren Anpassung untergehen musste, damit die Benutzerfreundlichkeit bewahrt wird. Deswegen wurde das Tool Wireshark noch einmal besucht und zusätzlich auch vom alten ReportingGUI Inspirationen geholt.

Das UI der Release Version sieht wie im Bild 42 aus.

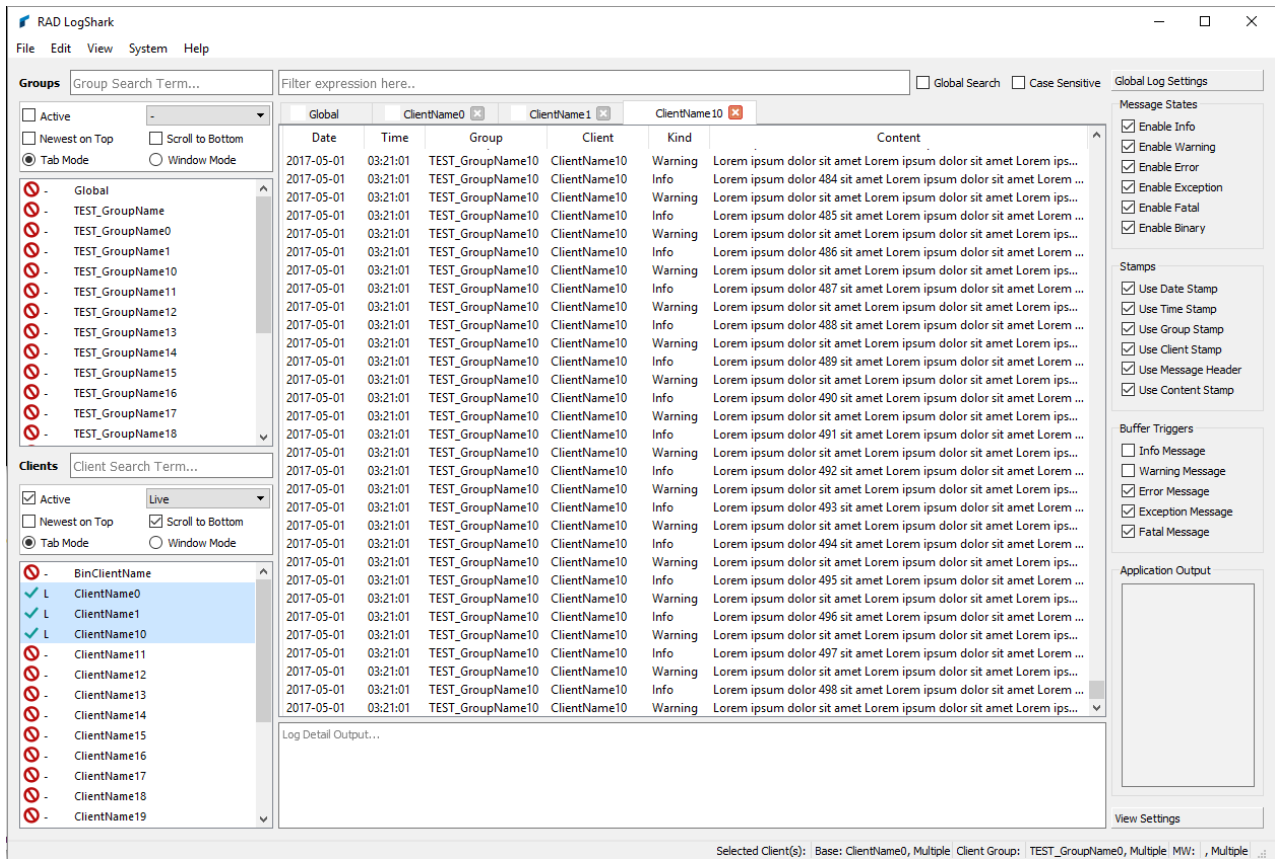


Bild 42: UI Releaseversion

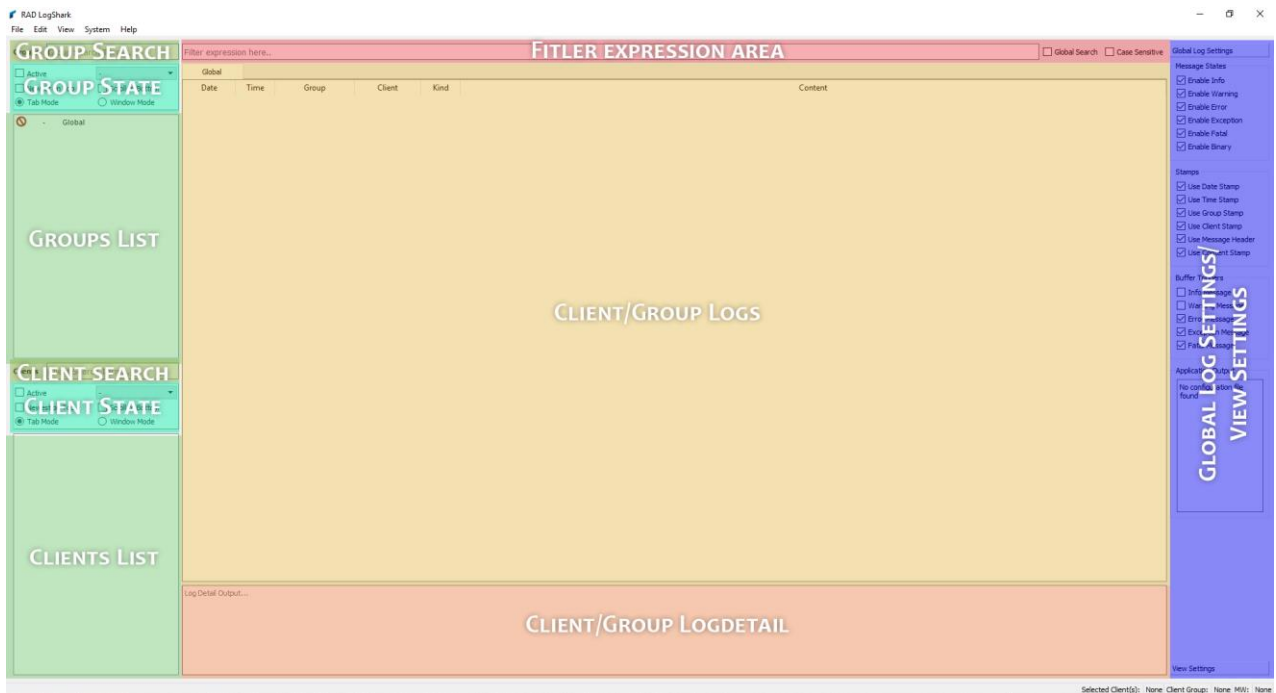


Bild 43: Layout UI Releaseversion

Diese UI Version bietet Ähnlichkeiten mit dem alten Tool, was mit dem Wiedererkennungswert Vereinfachung bringt. Die Einstellungen der Clients sind nun näher bei den zugehörigen Clients/Groups - Listen. Zudem sind die Clienteneinstellungen von einem Bereich zu zwei voneinander unabhängige Bereiche aufgeteilt. So muss kein UI Element deaktiviert werden und auch nicht nach Fokus gearbeitet werden, um die verfügbaren Funktionalitäten zu limitieren (diese können nun fix angegeben werden). Ausserdem wurde der Eingabebereich für Filterausdrücke mit dem Tab - Bereich gruppiert und bietet nun einen bereiteren Bereich zur Eingabe des Filterausdrucks. So können auch längere Filterausdrücke eingegeben werden ohne das Teile davon wegen der Breite ausgeblendet werden. Ferner wurden dem Statusbereich permanente Informationsfelder hinzugefügt, die Informationen zu den ausgewählten Clients/Groups geben, wie z.B. der Gruppenname eines Clients. Überdies wurde im Bereich der globalen Logeinstellungen eine zusätzliche Seite implementiert, in der Einstellungen zu den angezeigten UI Elementen vorgenommen werden können. Im Weiteren wurden die UI Elemente dynamischer gemacht; zuvor hatten UI Bereiche wie Logdetailinput eine fixe Höhe und konnten nur bis zu einer bestimmten Höhe skaliert werden; nun aber können die UI Bereich dynamisch unlimitiert skaliert werden.

So wurde in Folge der Feedbacks und der erneuten Betrachtung der Inspirationsquellen ein UI Design erreicht, dass benutzerfreundlich und übersichtlich zugleich ist.

4.5 Diskussion

Zum Abschluss des technischen Berichts sollen in diesem Abschnitt noch bestimmte Aspekte der Applikation in Rückblick kurz diskutiert werden.

4.5.1 Leistung

Da die Leistung der Applikation, nebst der Stabilität, einer der wichtigsten Merkmale der Applikation darstellte, wurde diesem Aspekt bei der Entwicklung die höchste Priorität gegeben. Viele Lösungswege wurden versucht, bis die oben unter dem Abschnitt Leistung und Multithreading beschriebene Lösung erreicht wurde. Es wurde versucht, die Viewupdates per Multithreading zu lösen, was jedoch zu Speicherzugriffverletzungen führte. In Qt müssen GUI Updates exklusiv aus dem GUI Thread erfolgen und können nicht von anderen Threads initiiert werden. Ferner wurde versucht durch Codeoptimierungen die Verarbeitungszeit im GUI Thread zu minimisieren, was jedoch auch nicht half. Nach langen Performanzanalysen mit dem Performzananalyse - Tool von Visual Studio stellte sich heraus, dass das Problem in zwei Libraries des Qt Frameworks lag (Qt5Gui.dll, Qt5Widget.dll). Nach zusätzlicher Recherche und Forumposts² im Qt Forum wurde schliesslich festgestellt, dass das Performanzproblem mit den angestauten View Updatesignalen zusammenhing.

Deswegen wurde die Lösung mit den gestaffelten View Updates gewählt, wobei je nach Fenster Anzahl das Intervall für die gestaffelten Updates dynamisch erweitert oder vermindert wird. Auch wenn diese Lösung für bis zu 20 Client Logfenster gut funktioniert, kann man dies für die Anzahl von 40+ Logfenster nicht einfach so weiter behaupten, da das Updateintervall bei 20+ Clients bei einer Sekunde liegt und bei 40+ Clients die zwei Sekunden Marke erreicht, kann man nicht mehr von einer Echtzeit-Aktualisierung der Logs in der UI sprechen; was jedoch von der Applikation erwartet wird.

Eventuell könnte man eine RT_UpdateWorker ähnliche Signalisierungsmechanismus direkt für den GUI Thread implementieren, so dass die UI Views in einem bestimmten, konstanten Intervall aktualisiert werden. So würden fortlaufend Lücken für Userinput geschaffen und die Views in kürzeren Intervallen konstant aktualisiert werden, was auch das Problem mit der 40+ Logfenster lösen könnte. Da diese Implementation aus Zeitmangel nicht ausprobiert werden konnte, ist dies jedoch lediglich eine ungetestete Hypothese.

Ausserdem wird zurzeit das Updateintervall mit der Anzahl Logfenstern angepasst. Das Updateintervall könnte auch aus dem Intervalldurchschnitt der ankommenden Logs zusammengestellt werden. So würde das Updateintervall abhängig von den ankommenden Logs bestimmt werden und würde eine bessere Intervallzeit ergeben, da in einem echten Szenario die Logs nie permanent in einem 20 Millisekunden Intervall ankommen.

4.5.2 UI Design

Leider konnten für das Design des UI keine formellen Usability Testsession geplant werden, da der Industriepartner dafür keine Zeit planen konnte. Die nötigen UI Anpassungen wurden gemäss Feedback in den wöchentlichen Meetings an der HSR, den zweiwöchigen Meetings beim Industriepartner und informellen Usability Tests im Freundeskreis ermittelt.

² <https://forum.qt.io/topic/78598/performance-issues-with-multiple-visible-windows/21>

Aus diesem Grund könnte das UI sicherlich noch weiter optimiert werden. Zum Beispiel könnte man das Ein- und Ausblenden von Logspalten mit einem Kontextmenü direkt bei der Clienttabelle umsetzen. So wäre die Operation dem Kontext entsprechend und wäre intuitiver. Auf die gleiche Art könnte das Ein- und Ausblenden von UI Elementen realisiert werden.

Ferner ist das UI Design noch grösstenteils im native Design gehalten. Mit weiteren Style - Anpassungen für die verschiedenen Bereiche der Applikation könnte man das UI besser gruppieren und den Zusammenhang der UI Elemente besser illustrieren, was für den Benutzer besseren visuellen Feedback geben würde und die UI Bereiche selbsterklärend sein würden.

4.6 Ausblick

Im zweiten Monat der Entwicklung wurde sehr viel Zeit mit der Performanzoptimierung verloren, wovon der Code sicherlich gelitten hat. An vielen Stellen besteht Bedarf für Refactoring, Vereinfachung und Codeoptimierung. Viele Routinen könnten verbessert und an vielen Stellen gemeinsamer Code extrahiert werden. Ausserdem besteht zurzeit, wie unter dem Abschnitt Abhängigkeiten ersichtlich ein Bedarf, die Abhängigkeiten zum RT_Main_Modell - Objekt zu minimieren. Die Applikation kann in seinem jetzigen Zustand zwar ohne Probleme schnell erweitert und neue Features auch ohne viel Effort in die bestehende Architektur integriert werden, doch wäre eine zusätzliche Runde Codeoptimierung anzuraten.

Viele der gewünschten Features konnten umgesetzt werden. Jedoch wurden diese Features in der einfachsten Form möglich realisiert, warum auch diese weiter optimiert werden könnten. Ein gutes Beispiel hierfür wären die MergeViews. Zurzeit wird für eine MergeView eine zusätzliche Gruppe erzeugt, in die mehrere Clients zusammen routen; es ist in gewissem Sinne das visuelle Pendant zum Ausgabemodi ToSharedFile. Dieses Feature könnte so angepasst werden, dass man direkt mehrere Fenster oder Logtabs selektieren könnten, um automatisch eine zusammengeführte View zu erzeugen. Ausserdem wurde ein bestimmter Aspekt der MergeViews noch nicht ganz umgesetzt. In einer MergeView sollten die Logs, die ungefähr zur gleichen Zeit ankommen, zusammengefasst angezeigt werden, was momentan nicht der Fall ist.

Eine Liste der ausstehenden Features kann in der folgenden Tabelle 2 gefunden werden.

Parallel Scrolling	Paralleles scrollen von mehreren Logfenstern zur gleichzeitigen Analyse von Logs zu einer bestimmten Zeit
Tabs to MergeView	Direktes Zusammenführen von Tabs zu einem Tab
Windows to MergeView	Direktes Zusammenführen von Clientfenstern zu einem Fenster
Layouts	Einstellbare Layouts
Clients basierte Vieweinstellungen	Einstellungen wie sichtbare Spalten, welche Logtypen ein Klient loggen soll.
Filterausdruck Dialog	Ein Dialog zum Erstellen von Filterausdrücken ohne spezielle Filterregexausdrücke schreiben zu müssen

Tabelle 2: Ausstehende Features

5 Glossar

In diesem Abschnitt werden wichtige Begriffe in der Dokumentation zusammengefasst wiedergegeben.

Begriff	Beschreibung
Client	Clients stellen in der Applikation loggende Komponenten einer RApplication dar und sind üblicherweise «Channels», also Kanäle in einer Applikation, die Logs generieren. In der Applikation LogShark sind Clients Abbildungen dieser «Kanäle», anhand derer die Logs an verschiedene Destinationen umgeleitet werden.
Groups	Groups sind eine Gruppierungsebene für Kanäle/Channels/Clients. Eine Applikation(RApplication) kann mehrere Gruppen/Groups haben und diese wiederum mehrere Clients/Channels/Kanäle besitzen.
MergeView	MergeView's sind spezielle Gruppen/Groups, die hauptsächlich zur Vereinigung verschiedener Clientlogs in der Applikation LogShark erzeugt werden. MergeViews existieren clientseitig nicht, wobei «Groups» als Ebene schon. MergeViews sind also spezielle Gruppenkonstrukte, die für die gemeinsame Visualisierung von Logs von beliebigen Clients erzeugt werden.
GitFlow	Ist eine Vorgehensart mit Git bei der Entwicklung von Code. Dabei beschreibt GitFlow, wie man seine Branches für Release, Development und Features aufteilt, um Codeänderungen besser zu verwalten.
SharedFile	SharedFile oder «geteilte Datei» ist ein Clientzustand, der Logs eines Clients in eine geteilte Datei routet. Also existiert einen File «shared_001.txt» in den Client X und Client Y hinein Ihre Logs schreiben.
RApplication	Eine Applikation, der die Reporting Library zum Loggen benutzt.
Master	Ein Listener – Thread, der auf eingehende RApplication Verbindungen hört und bei einer Verbindung eine neue Session für die Verbindung erzeugt.
Groupclient	Dieser Begriff ist gleichbedeutend mit Groups und deutet in seinem Namen als Modell, dass es eigentlich auch wie ein Client funktioniert. Clients und Groups werden mit demselben Domainobjekt, RT_Client, modelliert.
Worker	Spezielle «Arbeiter»-Objekte, die in einen Thread ausgelagert werden und da eine bestimmte Aufgabe erfüllen, wie z.B. das Schreiben von Logs in eine Datei.
Qt Framework	Ein C++ GUI Toolkit, mit dem plattformunabhängige Applikationen in C++ geschrieben werden können.
Log	Ein Log ist im Prinzip ein Informationsstring, der bestimmte Prozesse in einer Applikation wiedergibt, wie z.B., wenn etwas beim Laden schief läuft, wird dies geloggt, «notiert».
TCP	Transmission Control Protocol. Ist ein Netzwerkprotokoll auf Ebene Transport. TCP garantiert die Übertragung eines Paketes.

6 Anhang A

6.1 Formulare

6.1.1 Eigenständigkeitserklärung



Eigenständigkeitserklärung

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum:

8716 Schmerikon, 01.06.2017

Name, Unterschrift:

Zafer Dogan



6.1.2 Nutzungsrechte

Vereinbarung

1. Gegenstand der Vereinbarung

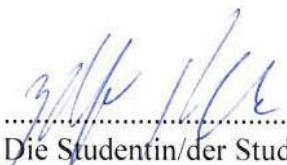
Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Projektarbeit «RAD Enhanced Reporting Tool» von Zafer Dogan unter der Betreuung von Thomas Corbat geregelt.


2. Urheberrecht

Die Urheberrechte stehen dem Studenten zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl vom Student, von der HSR wie von Rheinmetall Air Defence AG nach Abschluss der Arbeit verwendet und weiterentwickelt werden.

Rapperswil, den 29.05.2017 
Die Studentin/der Student

Rapperswil, den 29.5.2017 
Der Betreuer
Thomas Corbat

Rapperswil, den 31. Mai 2017 
Der Studiengangleiter / die Studiengangleiterin

6.2 Persönlicher Bericht

Die Studienarbeit, in der ich für die Rheinmetall Air Defence AG in Zürich eine neue GUI Applikation für Ihr veraltetes Tool entwickeln durfte, war, wie schon zuvor das Engineering – Projekt, eine für mich sehr lehrreiche Zeit.

Ich hatte schon zuvor Interesse an der Entwicklung von GUIs mit der C++ Sprache und hatte mich bis zur Studienarbeit mit dem wxWidgets – Framework und dem SFML - Framework auseinandergesetzt gehabt. Dank der Studienarbeit konnte ich mich in ein mir völlig unbekanntes GUI Toolkit einarbeiten und diese tiefgründig kennenlernen.

Bevor ich diese Studienarbeit begann, dachte ich, ich würde das Projekt mit einem Studienpartner durchführen und freute mich auf einen Arbeitspartner, der mich ab und zu entlasten könnte. Jedoch konnte mein Studienpartner dieses Semester die Studienarbeit nicht durchführen und mir blieb die Wahl entweder die Studienarbeit zu verschieben oder sie alleine durchzuführen. Ich wollte die Studienarbeit dieses Semester durchführen, vor allem weil mich das Thema der Arbeit sehr reizte. Zudem würde ich die Gelegenheit bekommen, mit einem Industriepartner zu arbeiten. Die Studienarbeit war aus meiner Sicht also sehr vielversprechend.

Das Projekt lief bis Anfang Construction 2 Phase eigentlich sehr geschmeidig; es gab zwar kleinere Probleme hier und da beim Setup des Qt Frameworks und dessen Testing Library, jedoch hatten diese keine unüberbrückbaren Ausmasse. Ab Construction 2 Phase wurde es dann plötzlich sehr interessant. Ich sah mich mit einem Leistungsproblem gegenübergestellt, dass sich mit den Qt internen Lösungen (wie z.B. eigene Modelle für QTableView schreiben etc.) nicht zu lösen schien. Das Intervall, in der die Logs ankamen, erzeugte so viele View Updatesignale, dass Qt mit dem Rendern der Views nicht mehr nachkam. Überall habe ich nach einer Lösung gesucht, in Online Präsentation zu Qt Performanzoptimierung bis zu Forum – Posts in Qt's archivierten Foren. Interessanterweise schien entweder niemand auf dasselbe Problem zugestossen zu sein oder es gab jene, die das gleiche Problem zu haben schienen, jedoch konnte niemand eine Lösung zum Problem liefern. Erst nach intensiven Analysen und Recherchen in das Thema wurde der Flaschenhals entdeckt und eine Strategie entsprechend definiert, um das Leistungsproblem umgehen zu können. Hierbei waren mir meine zwei Berater Thomas Corbat und Felix Morgner sehr grosse Unterstützung und gaben mir gute Ansätze, um das Leistungsproblem zu lösen.

Jedoch kostete mich diese Phase, in der ich eine Lösung für das Leistungsproblem suchte, nahezu drei Wochen und plötzlich stand ich unter enormen Stress. Ich hatte noch viele der Features umzusetzen, Tests zu schreiben und die Projektdokumentation fertigzustellen. Es wurde schlussendlich sehr knapp und ich musste mehrere Tage durcharbeiten. Diese Zeitknappheit führte zu unschönem Code und unschönem Codearchitektur.

Wenn ich Selbstkritik ausüben soll, dann wäre es wohl in diesem Punkt: Ich kann mich in eine Sache sehr schnell und lange hineinsteigern und verliere mich oft im Detail, was dazu führt, dass andere Aspekte des Projekts darunter leiden, wie die Projektdokumentation in diesem Fall. Zwar hat ein gewisse Menge Hartnäckigkeit seine Vorteile, aber ich muss lernen, eine bessere Balance zu erreichen.

6.3 Projektplan

6.3.1 Projekt Übersicht

Das "Reporting Tool", das für die Rheinmetall Air Defence AG entwickelt wird, soll ein Überwachungstool werden, dass beim Testbetrieb der Systeme zur Überwachung eingesetzt werden wird und das alte Reporting Tool ersetzen soll.

6.3.1.1 Zweck und Ziel

Das Ziel der Arbeit ist die Entwicklung einer plattformunabhängigen GUI-Applikation zur Überwachung des Dauertestbetriebs (24h/7Tage) der Systeme von RAD.

Die grosse Menge an Daten, welche übers Netzwerk empfangen werden, müssen dabei effizient verarbeitet und gefiltert werden. Es soll zudem ein sparsamer 24/7-Betrieb gewährleistet werden können, welcher die anderen Applikationen auf dem Rechner nicht beeinflusst. Die Informationen sollen dabei in einem benutzerfreundlichen GUI dargestellt werden, welches den Entwickler bei seiner Arbeit unterstützt.

Es soll im Gegensatz zum alten Reporting Tool eine zentrale Verwaltung in einer Applikation erlauben und erweiterte Verarbeitungsmöglichkeiten der Logdateien anbieten, wie z.B. Filterung, Parsen von Byte zu menschenlesbarem Format etc.

6.3.1.2 Lieferumfang

Das Projekt wird voraussichtlich als downloadbare Executable ausgeliefert. Die Executables (das Tool wird plattformübergreifend für Linux und Windowssysteme entwickelt, mit Fokus auf Windowssysteme) werden über eine gesicherte Plattform zum Download angeboten. Zusätzlich werden folgende Dokumente abgegeben: Projektplan, Domainmodell, UseCases, Nichtfunktionale Anforderungen, Sequenzdiagramme, Contracts, SAD, Testdefinitionen, Testprotokolle, Endversion der Applikation, Schlussbericht und die Präsentation.

6.3.1.3 Annahmen und Einschränkungen

Auch wenn die Zielplattform als Windows und Linux definiert sind, werden am Arbeitsplatz 95% Windowsrechner benutzt, weshalb die Linux-Entwicklung eher sekundär und "nice-to-have" ist. Trotzdem muss deswegen darauf geachtet werden, dass man plattformunabhängiges C++ entwickelt. Ausserdem soll bei Windows für 32bit Systeme im Vordergrund entwickelt werden.

6.3.2 Projektorganisation

Am Projekt ist nur eine Person beteiligt, der alle Rollen ausfüllt, wie Projektleiter oder Entwickler. Aufgaben werden in Arbeitspakete aufgeteilt und pro Iteration zugewiesen (agile Development). Die Sicherung des Informationsflusses und das Zeitmanagement obliegen der Verantwortung des Projektleiters, was in diesem Fall die gleiche Person wie den Entwickler und andere Rollen darstellt. Der Dozent überwacht den Projektablauf und ist für die Benotung zuständig. Das Projekt ist in die vier RUP-Phasen Inception, Elaboration, Construction und Transition aufgeteilt, sind jedoch als Grobgruppierungen zu verstehen, da agil entwickelt wird.

6.3.2.1 Organisationsstruktur

Dogan Zafer Projektleiter / Entwickler, zafer.dogan@hsr.ch

6.3.2.2 Externe Schnittstellen

Betreuer: Thomas Corbat, thomas.corbat@hsr.ch ,
 Felix Morgner, felix.morgner@hsr.ch
Kundenkontakt: Rico Steffen RAD, rico.steffen@rheinmetall.com

6.3.3 Management Abläufe

6.3.3.1 Kostenvoranschlag

Für die Umsetzung des Projektes stehen insgesamt 14 Wochen zur Verfügung. In dieser Zeit stehen pro Student 240 Stunden zur Verfügung (8 Credits). Dies ergibt ein Zeitbudget von 240 Stunden, da nur eine Person beteiligt ist. Diese Projektarbeit dauert vom 20.2.2017 bis zum 2.6.2017.

6.3.3.2 Zeitliche Planung

Die detaillierte Planung und Verwaltung der Arbeitspakete erfolgt in JIRA. Die Planung wird laufend aktualisiert und den gegebenen Umständen angepasst. Im untenstehenden Bild 44 ist die grobe Planung der Iterationen aufgeführt.

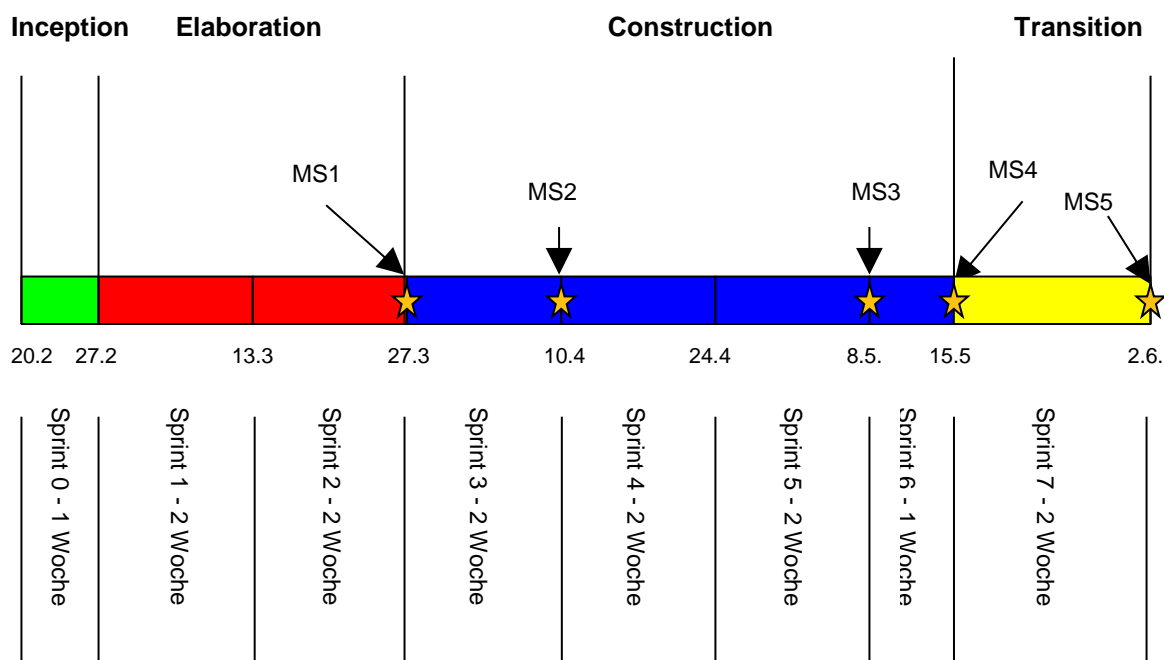


Bild 44: Iterationsplanung

6.3.4 Phasen / Iterationen

Die Entwicklung wird in die Phasen Inception, Elaboration, Construction und Transition aufgeteilt. Die einzelnen Iterationen tragen die Namen der Phasen, wobei die Iterationen innerhalb einer Phase durchnummeriert werden. Die Iterationen werden in JIRA verwaltet. Die Phasen sind eher als Gruppierung zu verstehen, da das Projekt agil entwickelt wird.

Startdatum	Enddatum	Iteration	Details
20.02.2017	27.02.2017	I1	Einrichtung Arbeitsplatz, Dokumentenstruktur, Einarbeitung Qt, Manuals zu SA
27.02.2017	13.03.2017	E1	Anforderungen vom Industriepartner entgegengenommen. Der Projektplan ist erstellt. Die UseCases sind definiert. Eine Domainanalyse wurde durchgeführt und das Domainmodell erstellt. Erste Wireframes vorhanden. JIRA und Bamboo konfiguriert. UI Prototyp steht
13.03.2017	27.03.2017	E2	Finalversionen von dem Domainmodell, Anforderungsspezifikationen, Contracts, SSDs und NFA vorhanden. SAD fertiggestellt. Wireframes sind fertiggestellt. UseCases sind alle im Brief Format vorhanden und die wichtigsten auch im Essential-Style. Integration Tool mit RAD Library umgesetzt. Beim Industriepartner vorgeführt.
27.03.2017	10.04.2017	C1	GUI Prototyp fertiggestellt, Grundfunktionalität altes Reporting Tool umgesetzt. Beim Industriepartner vorgeführt.
10.04.2017	24.04.2017	C2	Ausarbeitung GUI, Sicherung des Tools gegen Ausfall (24/7 Betrieb), neues GUI Funktionsprinzip umgesetzt (alles an einem Ort), Anfang neuer Features.
24.04.2017	08.05.2017	C3	Alle besprochenen Features umgesetzt, Usability Tests durchgeführt und UI an die finale Version angepasst. Beim Industriepartner vorgeführt.
08.05.2017	15.05.2017	C4	Bugfixing und Systemtesting Vorort.
15.05.2017	02.06.2017	T1	Behandlung Unvorhergesehenes, Schlussbericht und Präsentation erstellen und Fertigstellung aller Dokumente

6.3.5 Meilensteine

Datum	Meilenstein	Abgabe
27.03.2017	MS1 - Dokumentation	<ul style="list-style-type: none"> • Projektplan • Domainmodell • UseCases • Nichtfunktionale Anforderungen • Working UI Prototype, Integration RAD Library
10.04.2017	MS2 - Alphaversion	<ul style="list-style-type: none"> • Contracts • Sequenzdiagramme • SAD Prototyp • Working Prototype / Alphaversion • Testdefinitionen
08.05.2017	MS3 - Betaversion	<ul style="list-style-type: none"> • Betaversion / Release Candidate • Testprotokolle • SAD
15.05.2017	MS4 - Releaseversion	<ul style="list-style-type: none"> • Releaseversion • Testprotokolle
02.06.2017	MS5 - Abgabe	<ul style="list-style-type: none"> • Schlussbericht • Abgabe - CD • Präsentation

6.3.6 Meetings

6.3.6.1 Besprechungen

Mindestens einmal wöchentlich findet ein Treffen mit dem Projektbetreuer statt. Falls nichts Anderes bestimmt wird, finden diese Team-Meetings jeweils Montag ab 13.00 im Gebäude 8, Raum 8.261 (IFS) der HSR statt.

6.3.6.2 Reviews

Reviews mit dem Projektbetreuer finden am Montagmittag ab 13:00 statt.

Datum	Review	Details
20.02.2017	R0	Erstes Meeting, Besprechung SA Umsetzung
27.02.2017	R1	Statusbesprechung, Diskussion Projektplan und Setupprobleme
06.03.2017	R2	Statusbesprechung, Diskussion JIRA, Bamboo, Visual Studio und Qt Setup, Vorführung erster Testprogramme mit Qt, Diskussion grobe Use Cases und Domainmodell(grob), Rückblick auf Ziele der Woche
13.03.2017	R3	Statusbesprechung, Diskussion Anforderungsabnahme und weiteres Vorgehen, Diskussion umsetzbarer Ziele für SA, Diskussion Testing nach Absprache mit Industriepartner
27.03.2017	R4/MS1	Statusbesprechung, Diskussion Domainmodell, Anforderungsspezifikation und Dispatch, Fokussierung auf Implementation für Construction Phasen
03.04.2017	R5	Statusbesprechung, Diskussion Code
10.04.2017	R6/MS2	Statusbesprechung, Diskussion Code, Diskussion Stand Alphaversion
17.04.2017	Ostermontag	-
24.04.2017	R7	Statusbesprechung, Diskussion Performanzproblem mit Floating Windows
01.05.2017	R8/MS3	Statusbesprechung, Diskussion Lösung Performanzproblem, Feature Status
08.05.2017	R9	Statusbesprechung, Featurestatusbesprechung, Diskussion Projektdokumentation
15.05.2017	R10/MS4	Statusbesprechung, Diskussion Projektdokumentation
22.05.2017	R11	Statusbesprechung, Diskussion Projektdokumentation, Abgabe
29.05.2017	R12	Statusbesprechung, Diskussion Abgabe

6.3.7 Risikomanagement

6.3.7.1 Risiken

Eine detaillierte Auflistung aller Risiken befindet sich in der unteren Tabelle 3 unten.

Gewichteter
Schaden:

13.6

Nr	Titel	Beschreibung	max. Schaden [h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten		
R1	Setupverzögerungen	Vorallem in der Anfangsphase können Verzögerungen wegen Installationen, wie Setup von Redmine oder Jenkins sich in die Länge ziehen	8	90%	7.2	Nicht länger als 4 - 5 h an einem Setup sitzen	Hilfe vom Dozenten holen		
R2	Git - Repo Ausfall	Die Bitbucket Repository fällt aus	4	10%	0.4	Mit regelmässigen (wöchentlich) Github Backups vorbeugen	Git - Repo mit Backups neu aufspielen		
R3	Performanz Anforderung wird nicht erfüllt	Die Performanz des Qt Frameworks genügt den Anforderungen nicht	20	30%	6	Das Qt Framework in Bezug auf Performanz gut einstudieren	Mit Workarounds das Performanzproblem umgehen		
Summe			32		13.6				

Tabelle 3: Risikomanagement

6.3.7.2 Umgang mit Risiken

Das Management der Risiken wird von Anfang an überprüft und im Laufe des Projektes fortgehend sichergestellt. Dabei wird JIRA zum Tracking der Risiken eingesetzt.

Um Risiken entgegen zu wirken, werden nicht alle Iterationen komplett verplant und im Notfall kann das Engagement kurzzeitig erhöht werden.

6.3.7.3 Eingetroffene Risiken

Von den aufgelisteten Risiken ist R3 (Performanzanforderungen werden nicht erfüllt) aufgetreten. Zwar wurde das Qt Framework in Bezug auf die Performanz eingestudiert, doch da keine bestehenden Kenntnisse zum Qt Framework am Anfang des Projekts vorhanden waren, genügte das Einstudieren in die Thematik Performanz alleine nicht aus. Erst nach vielen manuellen Tests und Performanzanalysen mit Visual Studio wurde der Ursprung des Performanzproblems entdeckt und ein entsprechender Workaround implementiert. Die Lösung des Performanzproblems kann im technischen Bericht im Abschnitt Design, im Unterabschnitt Leistung und Multithreading nachgelesen werden.

6.3.8 Arbeitspakete

Arbeitspakete werden in JIRA verfolgt und verwaltet. Eine detaillierte Auflistung aller Tickets kann im Abschnitt Sprint Reports gefunden werden.

6.3.9 Infrastruktur

- Entwicklungssprache
 - **C++ 11**
- Entwicklungsumgebung
 - **Visual Studio 2015** Entwicklungsumgebung
- Bibliotheken
 - **Qt**: GUI Library
 - **External Lib**: Vom Industriepartner gelieferte Bibliotheken, die das Reporting umsetzen
 - **Boost**: Die Version 1.6 wird vom Industriepartner verwendet
 - **C++ 11 Standardbibliothek**
- Testing
 - **Qt Test**: Unit Tests für Qt
- Dokumentation
 - **Google Docs**: Projektdokumentation
 - **Qt Documentation**: Code Dokumentation
 - **Astah** für Diagramme
 - **draw.io** für Diagramme
- Management
 - **JIRA** (Projektmanagement)
 - **Bamboo** (Build Automation)
 - **Bitbucket** Code Repository
 - **SourceTree** Version Management Tool
 - **ReSharper C++** Code Quality Tool
- Geräte:
 - **Windows** (95% der Systeme) und **Linux** Systeme (>5%, mehr wegen älteren Systemen).

6.3.10 Qualitätsmassnahmen

6.3.10.1 Dokumentation

Die Dokumentation und deren Versionen werden in einem geteilten Google Drive Ordner verwaltet.

Den für die Einsicht der Dokumente relevanten Personen (Thomas Corbat, Felix Morgner) wurde eine Einladung zur Dokumenteinsicht verschickt.

<https://drive.google.com/drive/folders/0B5m8deNWbJhDR19PZ3ZZOU9YY28>

6.3.10.2 Projektmanagement

JIRA wird für die Umsetzung dieses Projektes verwendet (project tracking).

<http://sinv-56065.edu.hsr.ch:40004/secure/Dashboard.jspa>

Für den Zugang auf JIRA und Bamboo werden die für die Einsicht relevanten Benutzer Benutzerkonten erstellt.

6.3.10.3 Entwicklung

Für die Entwicklung wird das Versionsverwaltung Tool GIT eingesetzt. Der Code wird auf Bitbucket synchron für alle Beteiligten des Projektes gehalten. Die Betreuer Thomas Corbat und Felix Morgner wurden in das Team SA-HSR Zafer Dogan aufgenommen und die Repository RAD-Reporting Tool im Projekt

<https://bitbucket.org/account/user/sahsrzaferdogan/projects/RPH> für das Team zugänglich gemacht.

Direkter Link zur Repository:

<https://bitbucket.org/sahsrzaferdogan/rad-reportingtool>

Die Qualität wird sichergestellt durch Code Reviews. (Siehe Kapitel 8.3.4)

6.3.10.4 Vorgehen

Die Entwicklung des Projektes wird im iterativen, agilen Prozess voran gebracht. Jegliche Projektschritte werden im Voraus geplant und durch wöchentliche Meetings sichergestellt. Wird ersichtlich, dass die Planung einer Überarbeitung bedarf, wird die Überarbeitung im Projektplan nachgetragen. Die alte Version des Projektplans wird in den Backups gehalten.

6.3.10.5 Continous Delivery

Mit Bamboo wird CD gewährleistet und für die beiden Zielsysteme Windows und Linux Automation Builds bereitgestellt. Der Bamboo-Server ist unter folgender Adresse aufrufbar.

<http://sinv-56065.edu.hsr.ch:40005/allPlans.action>

6.3.10.6 Unit Testing

Weitere Angaben zu den Unit Tests im Kapitel 8.4 Testen.

6.3.10.7 Code Reviews

Code Reviews werden je nach Bedarf vom Studenten mit dem Betreuer geplant und

durchgeführt. Änderungspläne werden dann im JIRA festgehalten und als Arbeitspakete eingeplant.

6.3.10.8 Code Style Guidelines

Als Code Style Guideline wird das im Unterricht C++ Advanced in der Woche 13 behandelte Code Style Guideline Vorlage angewendet.

https://wiki.ifs.hsr.ch/CppAdvanced/files/lecture_13_core_guidelines.pdf

6.3.11 Testen

6.3.11.1 Unit Tests

Es sollen Unit Tests für alle wichtigen Teile des Reporting Tool geschrieben werden. Diese Unit Tests werden, wo möglich, mit dem Framework Qt Test erstellt, ansonsten anhand CUTE(einfache Unit Tests). UI Automation Tests wurden vom Industriepartner selbst nicht gefordert, aber wäre "nice to have", wenn möglich.

6.3.11.2 Integrations Tests

Im späteren Verlauf der Entwicklung werden automatisierte Integration Tests für das Testen der Zusammenarbeit der einzelnen Komponente erstellt.

6.3.11.3 Usability Tests

Es werden regelmässig Usability Tests für das Design und die Konnektivität des Tools mit dem System beim Industriepartner durchgeführt.

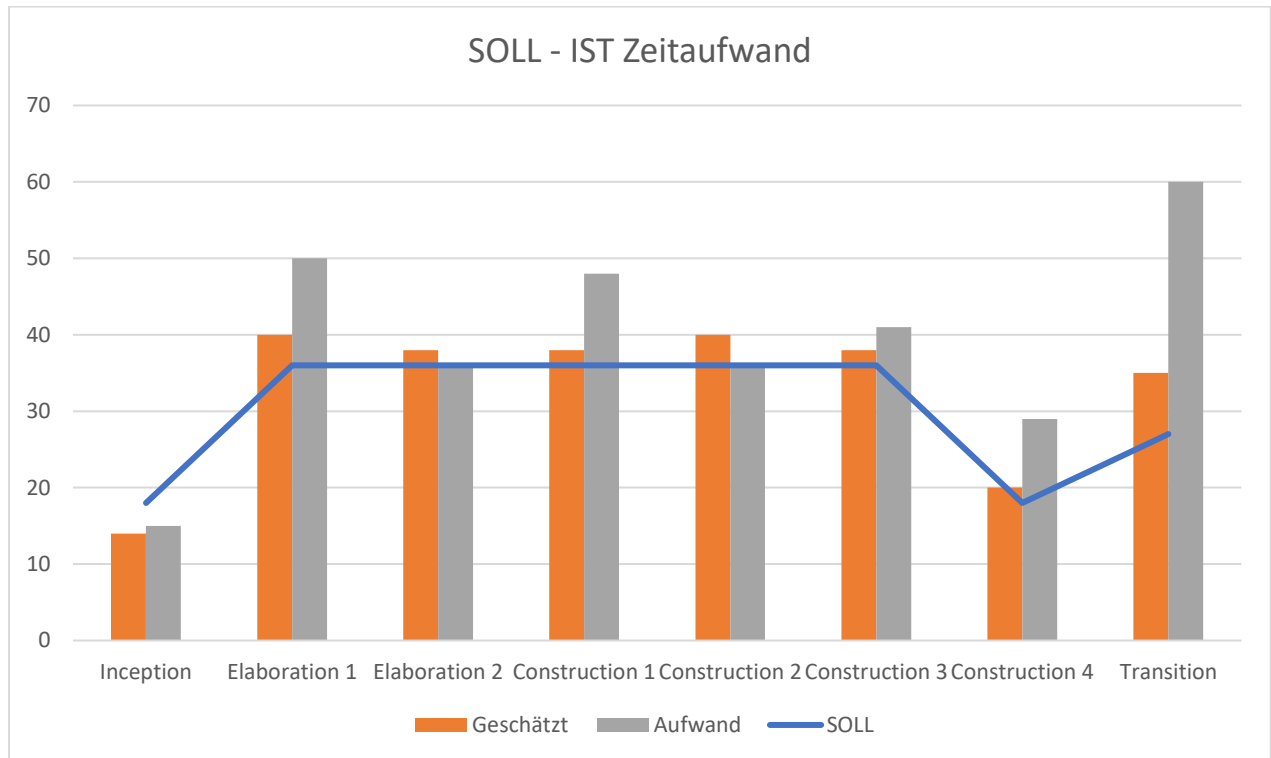
Diese Tests werden mit Szenarien und festen Zielen im Voraus definiert und von aussenstehenden Personen durchgeführt. Feedback wird im Google Drive nachgetragen.

6.3.11.4 System Tests

Ab der Iteration C1 wird angefangen Systemtests für die gesamte Applikation zu entwickeln. Diese werden mehrheitlich im Sinne von automatisierten Tests entstehen. Neben den automatisierten Tests können einzelne Teile jeweils auch mit Testprotokoll und vordefiniertem Test durchgeführt werden, falls ein automatisierter Test nicht sinnvoll erscheint.

6.4 Zeitabrechnung

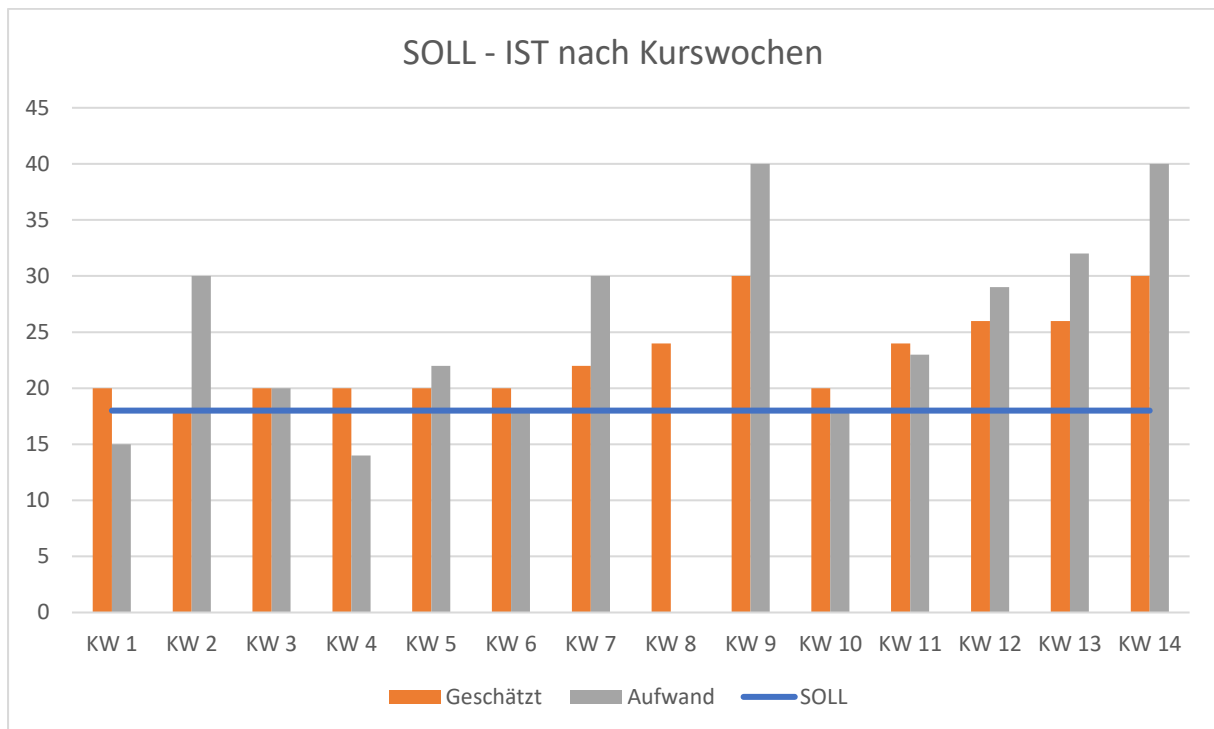
Die SOLL – IST Zeitabrechnung wurden aus den Protokollen und den JIRA Sprint Reports hergeleitet und zeigt nach Iteration, was der SOLL Wert ist, was geschätzt wurde und wie viel Aufwand in dieser Iteration geleistet wurde.



	Inception	Elaboration 1	Elaboration 2	Construction 1
SOLL	18	36	36	36
Geschätzt	14	40	38	38
Aufwand	15	50	36	48

	Construction 2	Construction 3	Construction 4	Transition
SOLL	36	36	18	27
Geschätzt	40	38	20	35
Aufwand	36	41	29	60

Und hier noch eine eine Auswertung der Zeit nach Kurswochen.



	KW 1	KW 2	KW 3	KW 4	KW 5	KW 6
SOLL	18	18	18	18	18	18
Geschätzt	20	18	20	20	20	20
Aufwand	15	30	20	14	22	18

	KW 7	KW 8	KW 9	KW 10	KW 11	KW 12
SOLL	18	18	18	18	18	18
Geschätzt	22	24	30	20	24	26
Aufwand	30		40	18	23	29

	KW 13	KW 14
SOLL	18	18
Geschätzt	26	30
Aufwand	32	40

6.6 Sprint Reports

6.6.1 RRT Sprint 1 - Elaboration 1

RRT Sprint 1 - Elaboration 1

[Reopen Sprint](#)

Closed Sprint, ended by info@z-motions.ch 27/Feb/17 11:00 PM - 14/Mar/17 1:07 PM [Linked pages](#)

[View RRT Sprint 1 - Elaboration 1 in Issue Navigator](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (- -> 1w 1d 2h 30m)
RRT-1 *	Ticket-Tracking System Setup	Projectmanagement	↑ Medium	DONE	- -> 1d
RRT-2 *	Qt Testing: Einarbeitung Unit und Automation Tests	Training	↑ Medium	DONE	- -> 2h
RRT-3 *	Einarbeitung JIRA	Training	↑ Medium	DONE	- -> 2h
RRT-4 *	Arbeitspakete für Epic MS1 definieren	Task	↑ Medium	DONE	- -> 2h
RRT-5 *	RAD Meeting: Requirements entgegen nehmen	Task	↑ Medium	DONE	- -> 1d
RRT-6 *	Setup Visual Studio und Qt	Task	↑ Medium	DONE	- -> 4h
RRT-7 *	Build Automation System Setup	Projectmanagement	↑ Medium	DONE	- -> 6h
RRT-11 *	Projektplan ausarbeiten	Projectdocumentation	↑ Medium	DONE	- -> 6h
RRT-12 *	Sitzungsprotokoll erfassen KW2	Projectdocumentation	↑ Medium	DONE	- -> 30m
RRT-13 *	Migration JIRA und Bamboo auf HSR Server	Projectmanagement	↑ Medium	DONE	- -> 5h
RRT-15 *	Datei importieren: Benutzer ruft File Browser auf	Story	↑ Medium	DONE	- -> 2h
RRT-16 *	Applikation beenden: Benutzer kann über File Menü Applikation beenden	Story	↑ Medium	DONE	- -> 2h
RRT-17 *	Datei importieren: Benutzer importiert Datei	Story	↑ Medium	DONE	- -> 2h
RRT-19 *	Sprint 0 / Inception Phase eintragen	Projectmanagement	↑ Medium	DONE	- -> 1h

Issues Removed From Sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (- -> 2d 1h)
RRT-8 *	UI: Wireframes: erste Prototypen	Projectdocumentation	↑ Medium	IN PROGRESS	- -> 4h
RRT-9 *	Domainanalyse erstellen	Projectdocumentation	↑ Medium	TO DO	- -> 6h
RRT-10 *	Use Cases / Stories definieren	Projectdocumentation	↑ Medium	IN PROGRESS	- -> 4h
RRT-14 *	JIRA/Bamboo/Visual Studio mit Qt etc. für Produktion konfigurieren	Projectmanagement	↑ Medium	TO DO	- -> 3h

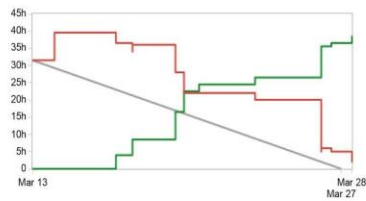
6.6.2 RRT Sprint 2 - Elaboration 2

RRT Sprint 2 - Elaboration 2

[Reopen Sprint](#)

Closed Sprint, ended by info@z-motions.ch 13/Mar/17 1:08 PM - 28/Mar/17 12:53 AM [Linked pages](#)

[View RRT Sprint 2 - Elaboration 2 in Issue Navigator](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (4d 7h 30m)
RRT-8 *	UI: Wireframes: erste Prototypen	Projectdocumentation	Medium	DONE	4h
RRT-9	Domainanalyse erstellen	Projectdocumentation	Medium	DONE	6h
RRT-10 *	Use Cases / Stories definieren	Projectdocumentation	Medium	DONE	4h
RRT-14	JIRA/Bamboo/Visual Studio mit Qt etc. für Produktion konfigurieren	Projectmanagement	Medium	DONE	3h
RRT-27	Einarbeit RAD Dokumente und Sourcecode	Training	Highest	DONE	6h
RRT-31	SAD Template erstellen	Projectdocumentation	Medium	DONE	4h → 1h
RRT-32	UC1 Logging: Benutzer kann Logs einsehen	Story	Medium	DONE	2h
RRT-36	GUI Entwurf: Grobe Umsetzung	Story	Medium	DONE	2h
RRT-37	Use Cases sind ausgearbeitet (Brief und Essential)	Projectdocumentation	Medium	DONE	2h
RRT-39	Nicht funktionale Anforderung sind erstellt	Projectdocumentation	Medium	DONE	1h
RRT-40	Termin mit RAD Partner vereinbaren	Task	Medium	DONE	30m
RRT-41	Vorführung GUI Working Prototyp RAD	Task	Medium	DONE	4h
RRT-44	Backups JIRA/Bamboo/Bitbucket erstellen	Task	Medium	DONE	1h
RRT-50 *	Setup Bamboo Qt Tests and Deployment (Fertigprodukt zum Downloaden)	Projectmanagement	Low	DONE	- → 2h
RRT-54 *	Sitzungsprotokoll erfassen KW5	Task	Medium	DONE	- → 1h

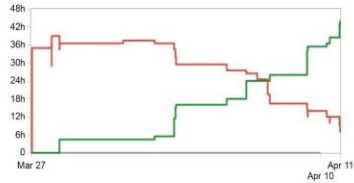
6.6.3 RRT Sprint 3 - Construction 1

RRT Sprint 3 - Construction 1

[Reopen Sprint](#)

Closed Sprint, ended by info@z-motions.ch 27/Mar/17 1:11 AM - 11/Apr/17 12:02 AM [Linked pages](#)

[View RRT Sprint 3 - Construction 1 in Issue Navigator](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (2d 5h → 4d 1h)
RRT-33 *	UC1 Logging: Benutzer kann Logs laden	Story	High	DONE	2h
RRT-34 *	UC1 Logging: Benutzer kann Logs speichern	Story	High	DONE	1h
RRT-35 *	Loghandling: GUI hat Reporting Library integriert (listening)	Story	High	DONE	4h
RRT-38 *	Sequenzdiagramme für wichtige Vorgänge sind erstellt	Projectdocumentation	Medium	DONE	2h
RRT-42 *	Loghandling: Benutzer kann über GUI Einstellungen vornehmen, welche Logs es anzeigt	Story	High	DONE	2h
RRT-43 *	Code Review	Task	Medium	DONE	2h
RRT-45 *	Anpassungen Prototyp nach Vorführung	Task	Medium	DONE	3h
RRT-46 *	Contracts erstellen	Projectdocumentation	Medium	DONE	2h
RRT-57 *	Vorführung RAD Projektstatus	Task	Medium	DONE	3h
RRT-58 *	UC1 Logging : Benutzer Logs werden als Tab's angezeigt	Story	High	DONE	- → 2h
RRT-59 *	UC1 Logging: Benutzer kann Logs importieren	Story	High	DONE	- → 2h
RRT-60 *	UC1 Logging: Binäre Logs werden geparkt angezeigt	Story	High	DONE	- → 2h
RRT-61 *	UC1 Logging: Benutzer kann Logs auf File umleiten	Story	High	DONE	- → 2h
RRT-62 *	UC1 Logging: Benutzer kann Logs unterdrücken	Story	High	DONE	- → 1h
RRT-64 *	Qt Dispatch/Observer Einarbeit	Training	High	DONE	- → 2h
RRT-65 *	Qualitätsmanagement: Erstelle Sektion Code Dokumentation	Projectdocumentation	Low	DONE	- → 1h

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (2h → 7h)
RRT-55 *	SAD erweitern	Projectdocumentation	Medium	IN PROGRESS	1h
RRT-56 *	Testprotokolle erstellen, Prototyp	Projectdocumentation	Medium	TO DO	1h
RRT-63 *	UC1 Logging: Benutzer kann Logs der selben Gruppe gemeinsam einsehen	Story	High	IN PROGRESS	- → 3h
RRT-67 *	UC1 Logging: Benutzer kann mehrere Logs ins gleiche File routen	Story	Medium	IN PROGRESS	- → 2h

Issues Removed From Sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1d 5h)
RRT-29 *	Basic Unit Testing für Projekt umsetzen	Task	Medium	TO DO	4h
RRT-51 *	UI Automation with Visual Studio : Try Setup	Task	Medium	TO DO	3h
RRT-52 *	Prepare Windows Installer for Easy Installation .msi	Task	Medium	TO DO	4h
RRT-53 *	Executables für Win10 signen	Task	Medium	TO DO	2h

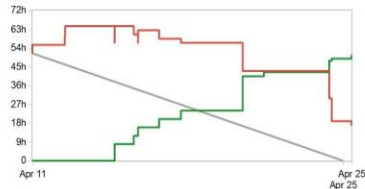
6.6.4 RRT Sprint 4 - Construction 2

RRT Sprint 4 - Construction 2

[Reopen Sprint](#)

Closed Sprint, ended by info@z-motions.ch 11/Apr/17 12:18 AM - 25/Apr/17 9:15 AM [Linked pages](#)

[View RRT Sprint 4 - Construction 2 in Issue Navigator](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3d 3h 30m → 4d 4h 30m)
RRT-56	Testprotokolle erstellen, Prototyp	Projectdocumentation	Medium	DONE	1h
RRT-68	UC1 Logging: Hierarchisches Logging	Story	Medium	DONE	4h
RRT-69	UC1 Logging: Capture startet beim Start der Applikation	Story	Medium	DONE	2h
RRT-70	UC2 Viewkonfiguration: Alle Tabs können auf einmal in Floating Windows umgewandelt werden	Story	Medium	DONE	4h
RRT-71	UC2 Viewkonfiguration: Tab-to-Window und Window-to-Tab per Drag & Drop	Story	Medium	DONE	2h
RRT-72	UC2 Viewkonfiguration: Tabs und Floating Windows können verschiedene Hintergrundfarben haben	Story	Medium	DONE	1h
RRT-83	UC4 Mehrere Clients gleichzeitig verwalten: Clients und Gruppen werden eindeutig in die client/group Liste eingetragen	Story	Medium	DONE	2h
RRT-88	UC1 Logging: Clients sollen gemäss Gruppenauswahl angezeigt werden	Story	Medium	DONE	1h
RRT-89	UC1 Logging: Client/Gruppen Map soll die Clients/Gruppe eindeutig identifizieren	Story	Medium	DONE	2h
RRT-91	UC1 Logging: Logging soll wie Tabs auch für Windows funktionieren	Story	Medium	DONE	4h
RRT-96	Code Optimierung, Refactoring	Task	Medium	DONE	4h
RRT-99	Termin RAD vereinbaren	Task	Medium	DONE	30m
RRT-100 *	UC1 Logging: Performanz und Stabilität optimieren	Story	High	DONE	- → 1d
RRT-101 *	Backups erstellen	Task	Medium	DONE	- → 1h

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1d 6h)
RRT-29	Basic Unit Testing für Projekt umsetzen	Task	Medium	IN PROGRESS	4h
RRT-55	SAD erweitern	Projectdocumentation	Medium	IN PROGRESS	1h
RRT-66	SSD, Contracts, SAD etc. ergänzen	Projectdocumentation	Medium	TO DO	3h
RRT-67	UC1 Logging: Benutzer kann mehrere Logs ins gleiche File routen	Story	Medium	IN PROGRESS	2h
RRT-84	UC1 Logging: Mehrere Clients können ins gleiche File geschrieben werden	Story	Medium	TO DO	2h
RRT-90	UC1 Logging: Tabs sollen beim Loggen eingefroren werden können (freeze/unfreeze)	Story	Medium	TO DO	2h

Issues Removed From Sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (1d 3h)
RRT-78	UC3 Reporting Tool konfigurieren: Konfigurationspersistenz	Story	Medium	TO DO	6h
RRT-79	UC3 Reporting Tool konfigurieren: Benutzer kann Konfiguration wie Workspaces laden	Story	Medium	TO DO	2h
RRT-92	UC1 Logging: Loganzeige auf Clientbasis bis zu einer bestimmten Anzahl Logs festlegen können	Story	Medium	IN PROGRESS	2h
RRT-94	UC1 Logging: Output von Logs mit mehreren Zeilen Logmessage optimieren	Story	Medium	TO DO	1h

6.6.5 RRT Sprint 5 - Construction 3

RRT Sprint 5 - Construction 3

Reopen Sprint

Closed Sprint, ended by info@z-motions.ch 24/Apr/17 10:16 AM - 09/May/17 10:39 AM Linked pages

View RRT Sprint 5 - Construction 3 in Issue Navigator



Status Report

* Issue added to sprint after start time

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (2d 2h → 3d)
RRT-63	UC1 Logging: Benutzer kann Logs der selben Gruppe gemeinsam einsehen	Story	High	DONE	3h
RRT-67 *	UC1 Logging: Benutzer kann mehrere Logs ins gleiche File routen	Story	Medium	DONE	2h
RRT-84 *	UC1 Logging: Mehrere Clients können ins gleiche File geschrieben werden	Story	Medium	DONE	2h
RRT-85	UC5 Logs durchsuchen und filtern: Benutzer kann Logfilter definieren	Story	Medium	DONE	2h
RRT-86	UC5 Logs durchsuchen und filtern: Benutzer kann Logfilter auf Clientbasis/Gruppenbasis anwenden	Story	Medium	DONE	4h
RRT-90 *	UC1 Logging: Tabs sollen beim Loggen eingefroren werden können (freeze/unfreeze)	Story	Medium	DONE	2h
RRT-92	UC1 Logging: Loganzeige auf Clientbasis bis zu einer bestimmten Anzahl Logs festlegen können	Story	Medium	DONE	2h
RRT-94	UC1 Logging: Output von Logs mit mehreren Zeilen Logmessage optimieren	Story	Medium	DONE	1h
RRT-102	UC1 Logging: Log Columns werden gemäss angezeigten Kolonnen gespeichert	Story	Medium	DONE	- → 1h
RRT-103 *	UC1 Logging: Client Status wird in der Clientliste angezeigt (ie. ob aktiv, inaktiv, zu parent, zu global etc.)	Story	Medium	DONE	- → 1h
RRT-104 *	UC1 Logging: Bugfixing und Anpassungen	Story	Medium	DONE	- → 4h

Issues Not Completed

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3d 30m)
RRT-29 *	Basic Unit Testing für Projekt umsetzen	Task	Medium	IN PROGRESS	4h
RRT-55 *	SAD erweitern	Projectdocumentation	Medium	IN PROGRESS	1h
RRT-66 *	SSD, Contracts, SAD etc. ergänzen	Projectdocumentation	Medium	TO DO	3h
RRT-73	UC2 Viewkonfiguration: Tabs können zusammengeführt werden (MergeViews)	Story	Medium	TO DO	2h
RRT-76	UC3 Reporting Tool konfiguration: Benutzer kann angeben, wohin die "to File" Dateien ausgegeben werden	Story	Medium	TO DO	1h
RRT-77	UC3 Reporting Tool konfiguration: Benutzer kann IP für Master festlegen	Story	Medium	TO DO	30m
RRT-78 *	UC3 Reporting Tool konfigurieren: Konfigurationspersistenz	Story	Medium	TO DO	6h
RRT-79	UC3 Reporting Tool konfigurieren: Benutzer kann Konfiguration wie Workspaces laden	Story	Medium	TO DO	2h
RRT-87	UC5 Logs durchsuchen und filter: Filter History	Story	Medium	TO DO	3h
RRT-93	UC3 Reporting konfigurieren: Loghistory Limite persistieren	Story	Medium	TO DO	2h

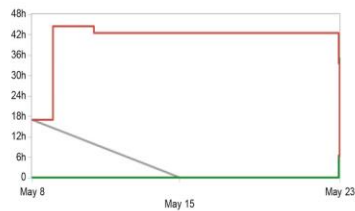
6.6.6 RRT Sprint 6 - Construction 4

RRT Sprint 6 - Construction 4

[Reopen Sprint](#)

Closed Sprint, ended by info@z-motions.ch 08/May/17 10:39 AM - 23/May/17 12:09 AM [Linked pages](#)

[View RRT Sprint 6 - Construction 4 in Issue Navigator](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (4d 7h 30m)
RRT-29 *	Basic Unit Testing für Projekt umsetzen	Task	Medium	DONE	4h
RRT-51	UI Automation with Visual Studio : Try Setup	Task	Medium	DONE	3h
RRT-55 *	SAD erweitern	Projectdocumentation	Medium	DONE	1h
RRT-66 *	SSD, Contracts, SAD etc. ergänzen	Projectdocumentation	Medium	DONE	3h
RRT-73 *	UC2 Viewkonfiguration: Tabs können zusammengeführt werden (MergeViews)	Story	Medium	DONE	2h
RRT-74	UC2 Viewkonfiguration: Benutzer kann mehrere Tabs/Windows parallel scrollen	Story	Low	DONE	2h
RRT-75	UC2 Viewkonfiguration: Floating Windows können angeordnet werden	Story	Lowest	DONE	2h
RRT-76 *	UC3 Reporting Tool konfiguration: Benutzer kann angeben, wohin die "to File" Dateien ausgegeben werden	Story	Medium	DONE	1h
RRT-77 *	UC3 Reporting Tool konfiguration: Benutzer kann IP für Master festlegen	Story	Medium	DONE	30m
RRT-78 *	UC3 Reporting Tool konfigurieren: Konfigurationspersistenz	Story	Medium	DONE	6h
RRT-79 *	UC3 Reporting Tool konfigurieren: Benutzer kann Konfiguration wie Workspaces laden	Story	Medium	DONE	2h
RRT-87 *	UC5 Logs durchsuchen und filter: Filter History	Story	Medium	DONE	3h
RRT-93 *	UC3 Reporting konfigurieren: Loghistory Limite persistieren	Story	Medium	DONE	2h
RRT-95	Code Optimierung, Refactoring	Task	Medium	DONE	4h
RRT-97	Bugfixing	Task	Medium	DONE	4h
RRT-53	Executables für Win10 signen	Task	Medium	TO DO	2h

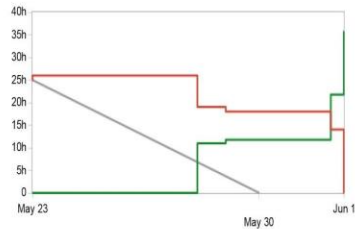
6.6.6.1 RRT Sprint 7 - Transition

RRT Sprint 7 - Transition

[Reopen Sprint](#)

Closed Sprint, ended by info@z-motions.ch 23/May/17 12:17 AM - 01/Jun/17 2:57 PM [Linked pages](#)

[View RRT Sprint 7 - Transition in Issue Navigator](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Original Time Estimate (3d 1h → 3d 2h)
RRT-106	Projektdokumentation	Projectdocumentation	Medium	DONE	1d
RRT-107	Bugfixing and Optimizations	Task	Medium	DONE	4h
RRT-108	SAD abschliessen	Projectdocumentation	Medium	DONE	2h
RRT-109	Diverse Projektdokumente erweitern (wie z.B. Usability Tests, Domainmodell Anpassungen etc.)	Projectdocumentation	Medium	DONE	6h
RRT-110	Code Dokumentation	Task	Medium	DONE	3h
RRT-111	Abgabe CD zusammenstellen	Task	Medium	DONE	2h
RRT-112 *	Backups erstellen	Task	Low	DONE	- → 1h

6.7 Sitzungsprotokolle HSR

6.7.1 Anwesende (üblicherweise)

- Zafer Dogan: zdogan@hsr.ch
- Thomas Corbat: thomas.corbat@hsr.ch
- Morgener Felix: felix.morgener@hsr.ch

6.7.2 Protokollführer

- Zafer Dogan

6.7.3 Besprechung Projekt KW1

Datum: 27.02.2017 / 13:00

6.7.3.1 Was wurde gemacht?

1. Kontaktaufnahme Industriepartner
 - a. E-Mail verschickt, warte auf Antwort
2. Inbetriebnahme (virtuelle) Serverinstanz / Setupversuch Redmine
 - a. Probleme mit VirtualHost einträge und Ruby on Rails Installation
3. Qt Einarbeitung / Installation Qt, Testen einfacher Beispiele in Qt IDE
4. Setup vorläufige Dokumentenraum auf Google Docs
 - a. Erstellung Vorlage und erstes Sitzungsprotokoll
5. Einarbeitung SA Dokumente (Server Manuals, Security etc.)

6.7.3.2 Wie viel Zeit wurde in was investiert?

1. Total investierte Zeit: 15h
 - a. Einarbeitung Qt : 4 - 5 h
 - b. Administrative Tätigkeiten(Doc. Setup, Manuals etc.): 2 - 3h
 - c. Inbetriebnahme S. und Setupversuch Redmine: 5 h
 - d. Projektplan etc. : 2 h

6.7.3.3 Entscheide der Besprechung

1. JIRA und BitBucket stehen frei zur Verwendung(BitBucket private Repo)
 - a. Backups vorsehen
2. Für Dokumentation allenfalls auch LaTeX
3. Qt Testing, eigene Frameworks von Qt verwenden (Lizenzen sollten open source sein)
4. Qt Lizenzen Rheinmetall fragen, Frameworks für Testing von Qt verwenden
 - a. Industriepartner abklären, wie vertraulich Code - Base sein muss (Laptop - Entwicklung)
 - b. Tsplus.hsr.ch → Server nicht meine Verantwortung
 - c. Clean Setup of virtual Server → müsste bei Support beantragt werden

- d. Christian Spielmann wegen Auflösung von sinv-56003.edu.hsr.ch anfragen (scheint eine Einstellungsproblem sein)
- e. Backups von virtuellen Server erstellen, regelmässig durchführen
 - i. Git-Repo in Redmine (für Backup eine Möglichkeit)
 - 1. Oder wie bei EP, normales Copy
- f. Docker-Installationen, allenfalls vorhanden, aber nicht per default
- g. Abgabe Dokumente für Sitzung jeweils bis 10:00 Montag
- h. Ruby on Rails: Rolf Bislin → E-Mail für Fragen bezüglich Ruby on Rails
- i. E-Mails zwischen Industriepartner, falls Projektrelevant, wie z.B. Anforderungen etc. → als Anhang mitliefern + CC zu Herr Corbat
- j. E-Mails zu Felix Morgener brauchen kein CC auf Herr Corbat

6.7.4 Besprechung Projekt KW2

Datum: 06.03.2017 / 13:00

6.7.4.1 Was wurde gemacht?

- 1. Setupversuche JIRA und Bamboo auf HSR Server
- 2. Setup Bamboo auf eigenem Server
 - a. <http://185.181.10.48:8085> (evaluation, 14 Tage)
- 3. Setup JIRA auf Cloud
 - a. <https://zdogan.atlassian.net/secure/Dashboard.jspa> (temporary)
- 4. Setup Repository auf Bitbucket
 - a. <https://bitbucket.org/Zmote89/rad-reportingtool> (definitiv)
- 5. JIRA und Bamboo mit Bitbucket verbunden
- 6. Visual Studio mit Qt: Filebrowser öffnen implementiert
- 7. Visual Studio mit Qt: QTest und Ranorex ausgetestet
 - a. Qt internete Test Framework mit Visual Studio nicht zum Laufen gebraucht → mit Visual Studio Tests umgesetzt (vorläufig)
- 8. Einarbeit JIRA und Bamboo
 - a. Erweiterte Konfigurationen und Buildpläne
- 9. Anleitungen Dokument erstellt, Visual Studio mit Qt Installation beschrieben
- 10. Arbeitspakete MS1 in JIRA definiert
- 11. Einarbeit LaTeX und erste Wireframe Prototypen erstellt

6.7.4.2 Wie viel Zeit wurde in was investiert?

Total investierte Zeit: **30 h**

- 1. JIRA/Bamboo/Bitbucket Setup: 14 h
 - a. Die Setupversuche auf dem HSR Server: 6 h
 - b. JIRA Cloud Setup: 2 h
 - c. Bamboo Setup (eigener Server): 4 h
 - d. Bitbucket und Verknüpfung mit JIRA und Bamboo: 2h
- 2. Visual Studio mit Qt Setup und Beispielprogramm: 9 h
 - a. Setup selber: 3 h

- b. Einarbeit (basic) in Qt: 4 h
 - c. Zum Laufen bringen: 2 h
 - i. Signals und Slots Mechanismus verstehen und umsetzen
 - d. Qt Testing: 3 h
 - i. Versuche ohne Erfolge
 - 3. Zusätzliche Einarbeitung: 3 h
 - a. JIRA: 1 h
 - b. Bamboo: 1 h
 - c. LaTeX: 1 h
 - 4. Wireframe Prototypen: 1 h
- Total Kosten: JIRA (10.- monatlich), Bamboo(10.- monatlich), Windowsserver (5.- monatlich) = **25.- monatlich** → für mich tragbar, falls OK

6.7.4.3 Entscheide der Besprechung

- a. JIRA und BitBucket stehen frei zur Verwendung(BitBucket private Repo)
 - i. Backups vorsehen
- b. Für Dokumentation allenfalls auch LaTeX
- c. Qt Testing, eigene Frameworks von Qt verwenden (Lizenzen sollten open source sein)
- d. Qt Lizenzen Rheinmetall fragen, Frameworks für Testing von Qt verwenden
- e. Industriepartner abklären, wie vertraulich Code - Base sein muss (Laptop - Entwicklung)
- f. Tsplus.hsr.ch → Server nicht meine Verantwortung
- g. Clean Setup of virtual Server → müsste bei Support beantragt werden
- h. Christian Spielmann wegen Auflösung von sinv-56003.edu.hsr.ch anfragen (scheint eine Einstellungsproblem sein)
- i. Backups von virtuellen Server erstellen, regelmässig durchführen
 - i. Git-Repo in Redmine (für Backup eine Möglichkeit)
 - 1. Oder wie bei EP, normales Copy
- j. Docker-Installationen, allenfalls vorhanden, aber nicht per default
- k. Abgabe Dokumente für Sitzung jeweils bis 10:00 Montag
- l. Ruby on Rails: Rolf Bislin → E-Mail für Fragen bezüglich Ruby on Rails
- m. E-Mails zwischen Industriepartner, falls Projektrelevant, wie z.B. Anforderungen etc. → als Anhang mitliefern + CC zu Herr Corbat
- n. E-Mails zu Felix Morgener brauchen kein CC auf Herr Corbat

6.7.5 Besprechung Projekt KW3

Datum: 13.03.2017 / 13:00

6.7.5.1 Was wurde gemacht?

- 1. Migration JIRA/Bamboo auf HSR - Server
- 2. Industriepartner getroffen, Anforderung abgenommen
- 3. Projektplan aktualisiert (Iterationsphasen)

- a. Gantt-Chart vorbereitet (Sprint 0 und 1)
- b. Gantt-Chart E2 Einteilung
4. Anforderungen, erster Prototyp
5. GUI Prototypen weiterentwickelt
6. Einarbeitung RAD altes Reporting Tool
7. Einarbeitung RAD Dokumente (Projektguide und grobe Anforderungen, Dokumente)
8. Bamboo Windowsbuild gefixt (hatte Problem mit dem Builden des Qt Projekts in Visual Studio)
9. Definition Inception-Phase Tickets
10. Sprint 2 - Elaboration 2 geplant

6.7.5.2 Wie viel Zeit wurde in was investiert?

1. Migration JIRA/Bamboo: 6 h
2. Meeting RAD Industriepartner: 4h
3. Einarbeitung RAD Dokumente + Vorbereitung Meeting: 3 h
4. Testen RAD Software: 1 h
5. Aktualisierung Projektplan: 1 h
6. Planung E2: 1 h
7. Anforderung (Prosa), User Stories: 2 h
8. GUI Prototypen: 1.5 h
9. Backups erstellen: 0.5 h

Total: ~ 20 h

6.7.5.3 Entscheide der Besprechung

1. Unit Tests mit CUTE OK, schauen was man GUI mässig umsetzen kann.
2. Allenfalls nebenbei auch Testprotokolle (manuelle UI Tests) formulieren

6.7.6 Besprechung Projekt KW4

6.7.6.1 Was wurde gemacht?

1. Termin für Vorführung Prototyp, GUI Entwürfe und Anforderung mit RAD Industriepartner vereinbart, 24.03.2017, 09.00 Uhr
2. GUI Entwürfe erstellt
3. Anforderungsspezifikation erstellt
4. Domainanalyse Prototyp erstellt
5. Grobe Umsetzung UI (Entwurf) in Code
6. Setup Testing mit Qt in Visual Studio + Bamboo Automation.
 - a. Inkl. Downloadbares Artefakt → ausführbares Programm .ZIP
7. Einarbeitung RAD Dokumente + Code
8. Recherche UI Automation Alternative

6.7.6.2 Wie viel Zeit wurde in was investiert?

1. GUI-Entwurf: 2 h
2. GUI-Umsetzung: 2 - 3 h
3. Bamboo Test Inklusion + downloadbares Artefakt (Software): 4 - 5 h
4. Domainanalyse Prototyp: 1 - 2 h
5. Anforderungsspezifikation: 2 h
6. Projektdokumentenaktualisierung: 1 h
7. Recherche UI Automation Alternative: 1 h

Total: ~ 14 h

6.7.6.3 Entscheide der Besprechung

1. UI Tests -> UI Projekt dünn halten, und Logik hinunter in Library migrieren und Modell + Aktionen so testen.
2. NFA → vom Industriepartner konkrete Zahlen abfragen

6.7.7 Besprechung Projekt KW5

Datum: 27.03.2017 / 13:00

6.7.7.1 Was wurde gemacht?

1. Use Cases definiert
2. Use Case Diagramm erstellt
3. Projektzustand beim Industriepartner vorgestellt, Anforderung diskutiert, NFA abgenommen
4. Weiteres Einarbeiten in RAD Dokumente und Source Code
5. Konzeptmodell fertiggestellt
6. Domainmodell (Klassenmodell) erstellt
7. Domainanalysedokument erstellt
 - a. Contracts und SSD fehlen noch
8. GUI Grober Entwurf erweitert
9. Qt Umbegung auf 32Bit umgestellt (vom Industriepartner so gewünscht)
10. User Story Logaufbereitung(neu: UC1 Logging): Benutzer kann Logs einsehen angefangen, Probleme mit RAD Partner diskutiert

Verschiedene Templates vorbereitet (SAD etc.)

6.7.7.2 Wie viel Zeit wurde in was investiert?

1. GUI-Umsetzung, Erweiterung: 3 h
2. Anforderungsspezifikation: 5 h
3. Domainanalyse: 5 h
4. Umstellung auf 32 Bit, Code + Bamboo: 4 h
5. Implementierung einfaches Modell-View-Controller im Code: 2 h
6. Vorführung RAD: 2 h
7. Backup Sprprint 1 - Elaboration 1: 30 min
8. Vorbereitung Templates (für Dokumentation, wie z.B. SAD): 30 min

Total: ~ 22 h

6.7.7.3 Entscheide der Besprechung

1. Persistierung per XML oder anderem Format OK? → Industriepartner fragen
2. Qt Dispatching/Observer etc. für Logdaten Management betrachten

6.7.8 Besprechung Projekt KW6

Datum: 03.04.2017 / 13:00

6.7.8.1 Was wurde gemacht?

1. Qt Dispatch/Observer Einarbeit
2. Ticket: UC1 Logging : Benutzer Logs werden als Tab's angezeigt fertiggestellt

3. Ticket: Loghandling: GUI hat Reporting Library integriert (listening) fertiggestellt
4. Ticket: UC1 Logging: Binäre Logs werden geparkt angezeigt angefangen
5. Ticket: Sequenzdiagramme für wichtige Vorgänge sind erstellt, erweitert
6. Qualitätsmanagement, Sektion Code Documentation ergänzt
7. Backups erstellt (Sprint, JIRA, Bamboo, BitBucket etc.)

6.7.8.2 Wie viel Zeit wurde in was investiert?

1. Umsetzung Code Implementationen: ~ 12 h
2. Code Refactorings: 1h
3. Einarbeit Qt Dispatch/Event System: 2 h
4. Dokumentationsarbeiten: 2 h
5. Backups: 1h

Total: ~ 18h

6.7.8.3 Entscheide der Besprechung

1. SetUniformRowHeight → verticalHeader.defaultSectionSize, das Gleiche?
Abfragen bei RAD Partner

6.7.9 Besprechung Projekt KW7

Datum: 10.04.2017 / 13:00

6.7.9.1 Was wurde gemacht?

1. Vorführung Projektstand Industriepartner
2. Benutzer kann Logs laden
3. Drag & Drop Feature für Tabs und Floating Windows (Tab-to-Window, Window-to-Tab)
4. Clientlistfilterung für Gruppen
5. Clients können im laufenden Betrieb angezeigt / deaktiviert werden (Capture raus)
 - a. Zurzeit per Click, später per Aktiv / Live Checkbox
6. Gespeicherte Logs können geladen werden
7. Unterliegende Struktur wurde abgeändert (auf Clientbasis: To File, To Parent etc. sind jetzt per Client einstellbar) → noch nicht ganz fertig umgesetzt
8. Diverse Refactorings (RT_Main_ControllerUtils -> CommonUtils -> CommonLogUtils | CommonStringUtils | CommonViewUtils), hauptsächlich wegen Anforderungen + include - Abhängigkeitsproblemen.
9. Separates Dialogwindow für Logs im Floating Modus bereitgestellt (RT_LogWindow)
10. CustomWindowTabWidget + CustomTabWidget für Drag & Drop Verhalten
11. Binary Client Messages werden zu HEX geparkt

12. Nächsten Sprint geplant, alle Use Cases als Tickets formuliert und auf Sprints verteilt
13. Implementation und Integration Domainklassen angefangen (RT_Client, RT_ClientFilter etc.)
14. → Message States per GUI werden zu Clienten kommuniziert

6.7.9.2 Wie viel Zeit wurde in was investiert?

1. Umsetzung Code Implementationen: ~ 20 h
2. Code Refactorings: 10h

Total: ~ 30h

6.7.9.3 Entscheide der Besprechung

1. RT_Main_Modell → Thread-Safe machen? Check Qt Doc für wie und was es braucht
2. Übersichtdiagramm erstellen: Welche Thread macht was, wo wird es gestartet, und zuschicken.

6.7.10 Besprechung Projekt KW9

Datum: 24.04.2017 / 13:00

6.7.10.1 Was wurde gemacht?

1. Vorführung Projektstand Industriepartner
2. UI States werden auch per Floating Window kommuniziert
3. Tab Performanz verbessert
4. Tabellen können eingefärbt werden
5. Headers ein/ausschaltbar (Kolonnen)
 - a. Muss auch noch auf File abgebildet werden
6. Hierarchisches Logging umgesetzt
7. Versuch, Performanzprobleme mit Floating Windows zu lösen
 - a. Konnte ich leider noch nicht lösen
 - i. Offene Ansätze
 1. Floating Windows als separate QApplications
 2. Floating Windows mit eigenem Event Loop
8. Refactoring: Move aller relevanten Elemente des Apps in die lib
 - a. Keine zyklischen Abhängigkeitsprobleme mehr für die Tests
 - b. Evtl. UI Automations mit Qt nun möglich
9. Testprotokoll angefangen
10. Stabilität der App verbessert
 - a. Hauptsächlich wegen Update per GUI Thread, kein Locking etc. mehr nötig

6.7.10.2 Wie viel Zeit wurde in was investiert?

1. Umsetzung Code Implementationen: ~ 10 h
2. Performanzoptimierungsversuche: ~ 20 h
3. Code Refactorings: 10h

Total: ~ 40h

6.7.10.3 Entscheide der Besprechung

keine

6.7.11 Besprechung Projekt KW10

Datum: 01.05.2017 / 13:00

6.7.11.1 Was wurde gemacht?

1. Umstellung Architektur gemäss Änderungen
 - a. Sammler-Thread
 - b. Updater-Thread
2. Code Optimierungen für Performanz
3. Client-State Anzeige in Clientliste

6.7.11.2 Wie viel Zeit wurde in was investiert?

1. Umsetzung Code Implementationen Architektur: ~ 6 h
2. Performanzoptimierungen: ~ 4 h
3. Implementation Clientstates: 4 h
4. Recherche für Fensterproblem, Forum Posts und Kommunikation: 2 h
5. Bugfixing: 2h

Total: ~ 18h

6.7.11.3 Entscheide der Besprechung

keine

6.7.12 Besprechung Projekt KW11

Datum: 08.05.2017 / 13:00

6.7.12.1 Was wurde gemacht?

1. Code Optimierungen
2. Filterung für Logs implementiert
3. Arrange Windows implementiert
4. Freeze/Unfreeze Window/Windows
5. Minimize all/Minimize

6. Show Binary Clients
7. Show Inactive Clients
8. Show Text Clients
9. Reset All/Selected
10. Reset Current Model / All Models
11. Anpassung Global/Gruppe/Client Hierarchie
12. Selektive Output to File
13. Output to Shared File
14. Icons for States
15. Actions Main Window implementiert

6.7.12.2 Wie viel Zeit wurde in was investiert?

1. Performanzoptimierungen: ~ 3 h
2. Implementation Features: 16 h
3. Bugfixing: 4h

Total: ~ 23h

6.7.12.3 Entscheide der Besprechung

- a. Performancevergleiche: Aktueller Platzverbrauch Logs vs SOLL
Platzverbrauch Logs (und derartige Vergleiche für Dokumentation)
- b. Projektdokumentation ist ein bisschen mehr als nur SAD etc., siehe Vorlagen
Skriptablage

6.7.13 Besprechung Projekt KW12

Datum: 15.05.2017 / 13:00

6.7.13.1 Was wurde gemacht?

1. Infrastruktur für Persistenz ausgebaut
 - a. Inkl. Persistenz Window -> RT_SettingsWindow
2. Alle Anbindungen implementiert (Actions, RT_LogWindow) etc.
3. Pointer Management improved (Parent-Child Hierarchy bei Pointeraufräumung Qt)
4. CheckBox UI für Live/To Parent etc. auf ComboBox umgestellt (Platzsparung + selective Routing)
5. Global Filter / Local Filter implemented
6. To Ring File Feature implemented
7. Memory Management Test durchgeführt
 - a. Bei 1-1.5h Laufzeit, nahezu 1 Million eingehenden Logs pro Client, 20 Clients, 20ms, Speicherverbrauch blieb bei 240MB (Fluktuationen zwischen 238MB und 242MB)
8. Diverse Bugfixes vorgenommen
9. 34 neue Unit Tests geschrieben

10. Client States Aktivierung nur wenn "active" Checkbox aktiv
 - a. Erlaubt nun die verschiedenen Clients einzustellen, bevor man Sie aktiviert
11. Diverse Fokusprobleme gefixt
12. Diverse Anpassungen bei RT_ToFileWorker vorgenommen
 - a. Wie das Outputformat der Logs, damit nachdem Speichern auch wieder sauber eingelesen werden kann.
13. Shortcuts implementiert
14. Abstract SA First Draft erstellt
15. Studium Abgabebericht
16. Grundstruktur Projektdokumentation angefangen
 - a. Notizen
17. Buffer Triggers implementiert (80%, Anbindung an Loghandling Prozedur fehlt)

6.7.13.2 Wie viel Zeit wurde in was investiert?

1. Implementation neuer Features : 10 h
2. Bugfixing: 6 h
3. Ergänzung bestehender Features LogWindow: 4h
4. Schreiben von Tests: 4h
5. Umstrukturierung UI: 2h
6. Pointer Management: 3h

Total: ~ 29h

6.7.13.3 Entscheide der Besprechung

1. 3-5 Seiten Management Summary
2. Technischer Bericht → SAD, Domainmodell, Tests etc.
3. Persönlicher Bericht 1 - 1,5 Seiten
4. To Ring File → oben raus, unten neue
5. Bis Mittwoch: Docu (Gesamt) Inhaltsverzeichnis(genau, nicht grob) schicken
6. Code Review bei Code Freeze
7. HSR keine Präsentation

6.7.14 Besprechung Projekt KW13

Datum: 23.05.2017 / 10:00

6.7.14.1 Was wurde gemacht?

1. UI Redesign gemäss Feedback
2. Implementation Persistenz
 - a. Mit JSON
 - i. Inkl. Client/Groups
3. Bugfixing

4. Code Dokumentation
 - a. Inkl. Einbindung des generierten Docs in die Applikation
5. MergeView's umgesetzt
6. Mehrere Konfigurationsdateien + ladbare Konfiguration
 - a. Inkl. letzte Konfiguration wird "erinnert"
 - b. Bei Windows in die Registry geschrieben
7. Qualitätsmassnahmen erweitert
8. SAD erweitert
9. Abgabearbeit erweitert, Inhaltsverzeichnis angepasst

6.7.14.2 Wie viel Zeit wurde in was investiert?

1. UI Redesign: 6h
2. Bugfixing: 6h
3. MergeViews: 4h
4. Persistenz: 8h
5. Projektdokumentation: 3h
6. Codedokumentation: 3h
7. Meeting RAD 2h

Total: ~ 32h

6.7.14.3 Entscheide der Besprechung

1. Term Project anstatt seminar project
2. Kurze Sätze verwenden, im Englischen angenehmer zum Lesen
3. Abstract letzte zwei Absätze streichen
 - a. Abgabe Abstract am 29. Mai kann ein bisschen mehr Informationen erhalten
 - b. Evtl Bilder zur Abgabe
 - c. Booklet Bachelorarbeiten für Beispiele Abgabe, kann man irgendwo downloaden
 - d. Software Dokumentations teil Anhang raus, und in technischen Bericht einarbeiten (UML etc. auch da rein), technischer Bericht ist SAD auf Anabolika, ie. Dokumentation, wieso man bestimmte Entscheide getroffen hat, wieso man bestimmte Designentscheidungen gefällt hat, wieso man bestimmte Module gemacht hat.
 - e. Literaturverzeichnis in Anhang, falls wenig, falls ½ - ¾ Seiten, eigenes Kapitel
 - f. Entwickleranleitung im Anhang B (anders als Benutzeranleitung), Benutzeranleitung in Anhang C
 - g. UI Screenshots da rein, wo gebraucht
 - h. Codeausschnitte embedded besser
 - i. API, Entwickleranleitung etc. Anhang B, Installationsanleitung, Benutzerhandbuch Anhang C

- j. API Dokumentation, Doxygen PDF
- k. Zeit SOLL IST, einfacher Graph, Soll ist Vergleich, jede Woche
- l. Code Metriken / Statistiken, interessant wäre Performanzmessungen
→ müssen aber auch beschrieben werden.
- m. Non-functional Anforderung sollten beantwortet werden (das Mindeste
sind die Anforderung, diese müssen beantwortet werden), z.B.
memory Consumption über einen Tag, über 1h, Code Metriken nur
falls Zeit
- n. Usability Tests → erwähnen, das Feedback direkt vom
Industriepartner, Meetings alle 2 Wochen
- o. Bei Testlogs, erwähnen was wie getestet wurde
- p. Falls nochmal Feedback verlangt, zuschicken Felix Morgner (per Mail)
- q. CodeReview → lieber nicht viel ändern

6.7.15 Besprechung Projekt KW14

Datum: 29.05.2017 / 13:00

6.7.15.1 Was wurde gemacht?

1. Letzte Code fixes
2. Erweiterung Tests
3. Management Summary
4. Anpassung Abstract, Abgabe Abstract
5. Dokument Qualitätsmassnahmen
6. Technischer Bericht, Erweiterung

6.7.15.2 Wie viel Zeit wurde in was investiert?

1. Code fixes: 4 h
2. Backups: 1 h
3. Test Erweiterung: 2 h
4. Abstract: 2 h
5. Management Summary: 3 h
6. Technischer Bericht: 3 h
7. Qualitätsmassnahmen: 2 h
8. Diverse Anpassungen Dokumente: 2 h

Total: ~ 19h

6.7.15.3 Entscheide der Besprechung

1. Farbig, Doppelseitig, Gebunden
2. Schwarz/Weiss einseitig, gebunden
3. API Doku von Inhaltsverzeichnis rausnehmen
4. Poster Studienarbeit, bis Dienstagabend Thomas abgeben, nicht zuviel Text
 - a. Check Vorlage Posters in Skripteserver

5. Nicht alles eindeutschen

6.8 Sitzungsprotokolle RAD

6.8.1 Datum: 09.03.2017 / 09:00

6.8.1.1 Anwesend

- Zafer Dogan: zdogan@hsr.ch
- Rico Steffen: rico.steffen@rheinmetall.com

6.8.1.2 Protokollführer

- Zafer Dogan

6.8.1.3 Fragen

- 1. Was für eine Testing Framework wird verwendet?**
 - a. Simple Unit Tests (CPP Unit Tests) / könnte je nachdem auch CUTE verwendet
- 2. GUI Automation Tests nicht vorhanden**
 - a. Auch wegen Zeitmangel, vielleicht vernachlässigbar
- 3. Ist auch eine UI Testing Automation Framework seitens RAD angeboten?**
 - a. Nein
- 4. Wie sieht die Integration davon in Visual Studio aus?**
 - a. -
- 5. Welche Version von Visual Studio wird verwendet?**
 - a. 2015, 32 bit (Version) (Build Target)
- 6. Welche Version von Qt soll verwendet werden?**
 - a. Open-Source, Lizenz hätte die Firma (nur für Tests relevant, libs open-source verwenden)
- 7. Wird die lizenzierte Version von Qt verwendet? Falls ja, wie erhalte ich die Lizenz?**
- 8. Welches Buildsystem verwendet RAD für dieses Projekt?**
 - a. Wäre frei wählbar
- 9. Wie wird das Reporting Tool bei RAD Multiplatform gebildet (Infrastruktur, Build Tool etc.)?**
 - a. Wäre gut wenn läuft (manuelle Tests auf Linux ausführen, Build), aber nicht sehr prioritär
- 10. Ist eine bestimmte Dokumentationsformat seitens RAD für dieses Projekt erwartet?**
 - a. HSR Format OK
- 11. Was wird für die Dokumentation inhaltlich erwartet?**
 - a. Nicht klare Definition, aber man sollte erkennen, wie das Tool funktioniert
- 12. Wie vertraulich ist der Code-Base? Ist das verwenden einer privaten online Repository wie BitBucket OK?**

- a. Kein Problem, hat nichts direkt mit Business selbst zu tun von RAD

6.8.1.4 Bemerkungen RAD

1. Bis nächsten Mittwoch melden, ob für nächsten Termin (Anforderung etc.)
2. Regelmässig Kontrollen mit RAD (ob Software in der Richtung wie gewünscht)
3. Evtl. Präsentation für RAD nach HSR Präsentation des Tools einplanen, evtl. mit Thomas Corbat besprechen
4. Bei Fragen zu den Anforderung etc. kann der Student sich bei Herr Steffen oder dem Entwickler des alten GUI Reporttools jederzeit melden (siehe CC im Mail).
5. Build für Linux
 - a. Manuelle Builds auf Linux Systemen ist schon genug; wichtig sei es, dass es auch auf Linux laufen können muss, aber als Auslieferung ist als Zielplattform Windows vorgesehen(95% der Systeme).

6.8.2 Datum: 23.03.2017 / 09:00

6.8.2.1 Anwesend

- Zafer Dogan: zdogan@hsr.ch
- Thomas Corbat: thomas.corbat@hsr.ch
- Rico Steffen: rico.steffen@rheinmetall.com
- Yves Strube: yves.strube@rheinmetall.com

6.8.2.2 Protokollführer

- Zafer Dogan

6.8.2.3 Fragen

1. **RegisterClient Messages werden empfangen, aber keine Client Logs, woran könnte das liegen?**
 - a. Muss per Master aktiv gesagt werden
 - b. Environment Variable muss dafür gesetzt werden
2. **Release Library des Master-Threads schliesst nicht bei terminate() oder ~Master() Aufruf, ist das ein Bug?**
 - a. Falls es weiter auftritt, nochmal melden.
 - b. Allgemein solche Probleme melden beim Industriepartner
3. **Wäre es möglich, denn Source-Code der Bibliotheksdateien einzusehen? (Verständnis wie Dateien z.B. über Socket's verschickt werden, wohin genau, welche Dateistruktur etc.)?**
 - a. Wird abgeklärt
 - b. Aber bei solchen, Bibliothek relevanten Fragen direkt an RAD wenden.
4. **Was für welche genauen NFA hat die RAD für das Reporting Tool?**
 - a. Siehe Anforderungsppez. Notes

6.8.2.4 Bemerkungen RAD

1. Master muss aktiv Klienten sagen, dass Sie loggen anfangen sollen
2. Start Capture / Stop Capture → master Thread sollte von Anfang an starten, (für Default Behaviour)
3. Ab und zu zum Testen die Bamboo Artifacts dem Industriepartner zuschicken.
4. Output, Design Doc, Source Code + App, Präsentation (Repetition)
5. Naming: ReportingServer / → ist nicht unbedingt ein Reporting Tool mehr, passenden Titel wählen (Projektname), RADShark?
6. Netzwerkkommunikationsperformanz → Buffers sollen nicht überlaufen, Daten sollen schnell genug auf die Platte geschrieben werden können + Netzwerkperformanz → schnelles Lesen vom Netzwerk. (Alle 20 ms 1 KByte, pro Socket/Client) → ~ 20 gleichzeitig aktivierte Clients, oberes Limit
7. Nicht z.B. 1 Byte nach dem anderen auf die Platte schreiben → die ganze Performanz sollte eigentlich schon vom Master etc. gehandelt werden, die GUI sollte von der Performanz her ungefähr parallel in der Zeit sein → setUniformRowHeight, TreeView etc. (Qt, für Performanz, solche Optimierungen vornehmen, in GUI in Echtzeit auch anzeigen können)
8. Alle Clients sollten gleiche Info, Warning etc. von einer Setting haben → man sollte auf Client-Basis aktivieren/deaktivieren
9. Wäre schön, dass GUI und Master getrennt sein sollte (shared Library)
10. Config files, multiple, für verschiedene Setups (ie. Track Client A and B, or track Client A, B, C und D (ie. z.B. das von Anfang an automatisch 10 Clients aktiv sind → benötigt: mehrere Config-Files, die gelesen, geladen werden können.
11. Falls Hauptspeicher überläuft → sollte Error geben, und Dialog, der warnt, aber man könne damit leben)
12. Zeitraum für Log-History, 10'000 Zeilen → sollte einstellbar sein
13. Loggen ist auf Seite Client teuer
14. unterdrücken auch als Feature ein
15. Log auf Buffer → z.B. es wird erst geloggt, wenn Buffer überschritten, hat auf GUI keinen direkten Einfluss → to File /to Live Feed, auf Gruppen und Client Basis → Client to File → wird angezeigt, wohin gespeichert wird → Umleitungen Clients etc., Idee: Mehrere Clients z.B. ins gleiche File schreiben etc.
16. C++ 11 Level ok, Standard, Flache Oberfläche ist gut

6.8.3 Datum: 05.04.2017 / 09:00

6.8.3.1 Anwesend

- Zafer Dogan: zdogan@hsr.ch
- Rico Steffen: rico.steffen@rheinmetall.com
- Yves Strube: yves.strube@rheinmetall.com

6.8.3.2 Protokollführer

- Zafer Dogan

6.8.3.3 Fragen

1. **Ist die Performanz OK?**
 - a. OK
2. **Ist das Erweitern der Master, RApplication, Session etc. OK? Oder müssen diese unabhängig bleiben?**
 - a. OK
3. **Windows MDI vs. Qt Windows?**
 - a. Fokus: unabhängig vom GUI, Qt Windows OK
4. **Mehrere Clienten - Output ins gleiche File → per MergeView OK?**
 - a. Grupperview Prinzip
 - b. RApplication ist nicht gleich Client → Clients sind Channels
5. **Signed EXE → welche Priorität hat dies für die RAD? (Warnung erscheint beim ersten Ausführen unter Windows 10)**
 - a. Nicht unbedingt priorität
6. **LocalID der Clients, Unique per Gerät oder auch im Netzwerk?**
 - a. Channel ID, Unique per Application
 - b. Registrierung sollte eigentlich per Namen geschehen, nicht ID's → ID's sind eindeutig innerhalb Socketraum.

6.8.3.4 Bemerkungen RAD

1. Log Messages mit mehreren Zeilen Loginhalt → Output und Anzeige optimieren
2. Clients werden gemäss Gruppenauswahl angezeigt (wenn Gruppenauswahl vorhanden)
3. ToFile / To Live → auf Client und Gruppen Basis, To Group (hierarchisches routen der Outputs)
4. Anpassungen Anforderung -> UDP -> TCP
5. Capture per Button weg → muss immer aktiv sein
6. Floating Windows → Docker Windows, per Drag & Drop der Tabs wäre gut
7. Config Auswahl wie Workspaces → Auswahl persistent

7 Anhang B

7.1 Entwickleranleitung

In diesem Abschnitt werden für Entwickler relevante Anleitung zusammengefasst, wie z.B. das Setup von Qt mit Visual Studio oder das Setup der Qt Tests.

7.1.1 Visual Studio and Qt Setup

7.1.1.1 Vorbereitung

- **7.1.1.1.1 Visual Studio 2015**
 - Community Version
- **7.1.1.1.2 Qt Addin (Qt5) <http://wiki.qt.io/QtAddin>**
 - Kann auch in Visual Studio intern installiert werden. Im Menu-Tab → Extras > Extensions und Updates → im Suchfeld Qt5 eintippen und die AddIns installieren. (Von Online Ablage)
- **7.1.1.1.3 Qt Version 5.6.0**
 - Die korrekte opensource Version für Visual Studio 2015 (64 bit)
 - http://download.qt.io/official_releases/qt/5.6/5.6.0/qt-opensource-windows-x86-msvc2015_64-5.6.0.exe
 - Die korrekte opensource Version für Visual Studio 2015 (32 bit)
 - http://download.qt.io/official_releases/qt/5.6/5.6.0/qt-opensource-windows-x86-msvc2015-5.6.0.exe

7.1.1.2 Installation

Wenn mit dem Installer für Visual Studio 2015 gearbeitet wird (Qt Framework), dann muss nachträglich noch der bin - Dossier (Dossier unter Qt Installation, die qmake.exe enthält) in die Umgebungsvariablen eingetragen werden. Der bin - Dossier muss auch im Qt5 - Tab, der nach der Installation von Qt Addins und Neustart von Visual Studio im Menu Tab erscheint, unter Qt Options gesetzt werden. Danach sollten unter Neue Projekte im Global Tab Qt Application Vorlagen erscheinen, mit denen man ein Qt Projekt aufsetzen kann. Ab diesem Punkt sollte das Projekt ohne Probleme kompilieren und Qt Programme entwickelt werden. Die .ui Dateien können im Qt Designer geöffnet werden (per default öffnet Visual Studio diese im Qt Designer) und die UI per Drag & Drop designed werden.

Je nachdem muss man unter Projekteigenschaften, C/C++ Tab die Includeverzeichnisse ergänzen (wie z.B. `${QTInstallDIR}/include/QtWidgets`), wenn man bestimmte zusätzliche Bibliotheken verwenden will.

7.1.2 Qt Testing Setup

Für die Verwendung der Qt Test Frameworks in Visual Studio Projekten, muss man ein neues Projekt in der Solution als eine Qt Console Application anlegen. Für die Qt Testklassen, die man erzeugen möchte, muss man unbedingt darauf achten, dass man die Qt - Klassen Template benutzt, wenn man ein Neues Item hinzufügt (anstatt Header oder Source-Datei, Qt Klasse wählen).

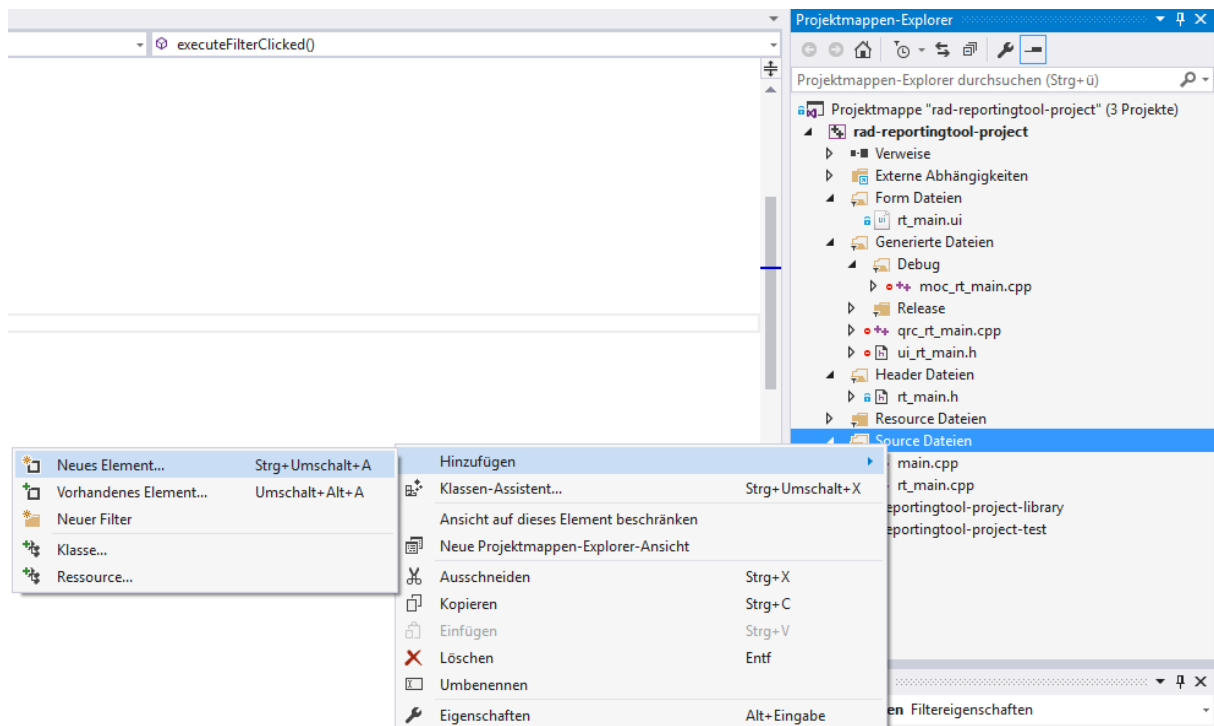


Bild 45: Testklassen hinzufügen

Damit im Wizard die Qt - Klasse als Auswahl verfügbar steht, müssen die Qt5 Tools installiert sein (im Setup Teil von Visual Studio mit Qt beschrieben). Sind die Tools installiert, muss man im Wizard folgende Klassentyp, wie in Bild 46 ersichtlich, wählen.

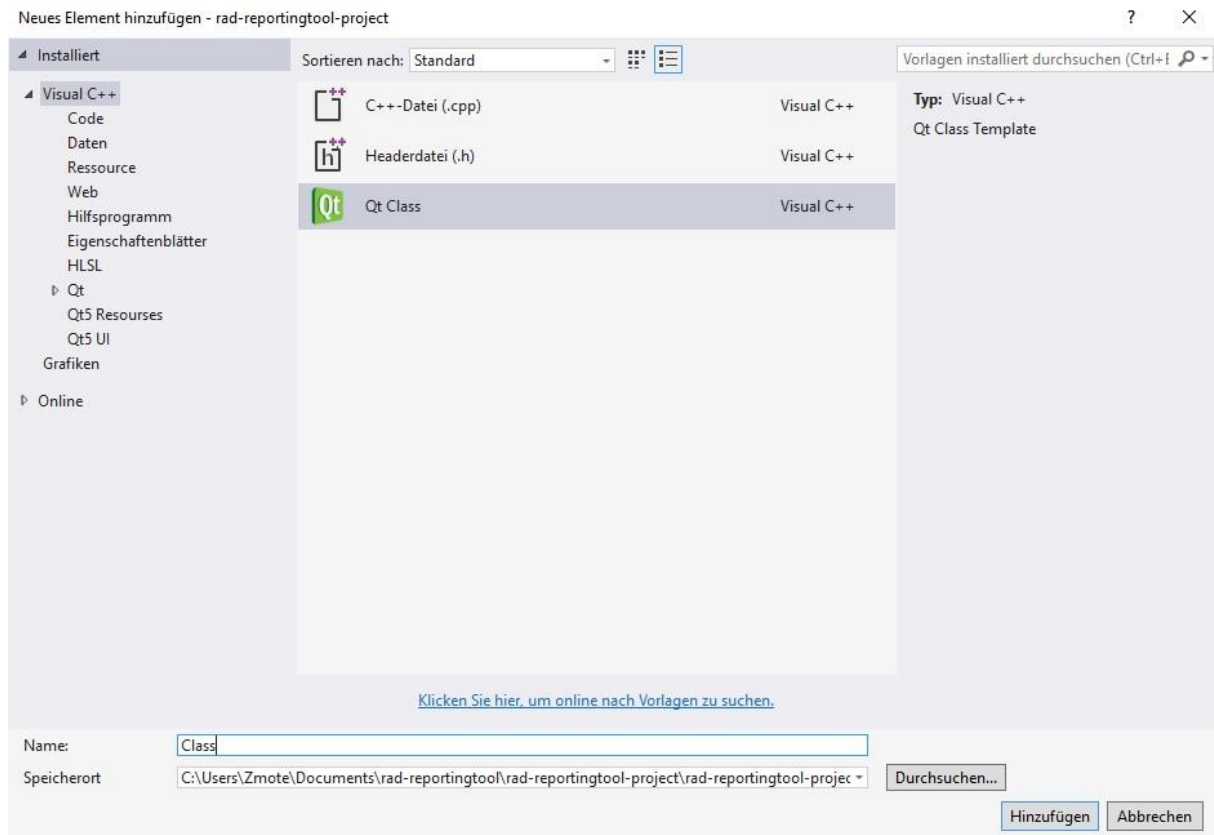


Bild 46: Wizard mit Qt - Klasse

Diese Klasse ist darum wichtig, da so die Klassen in die automatische Qt Build Prozedur aufgenommen werden (z.B. die benötigten moc-Dateien werden erzeugt).

Ein einfaches Testsetup kann wie folgt aussehen:

Main:

```
#include <QTest>
#include "test.hpp"
#include "rt_main_test.hpp"

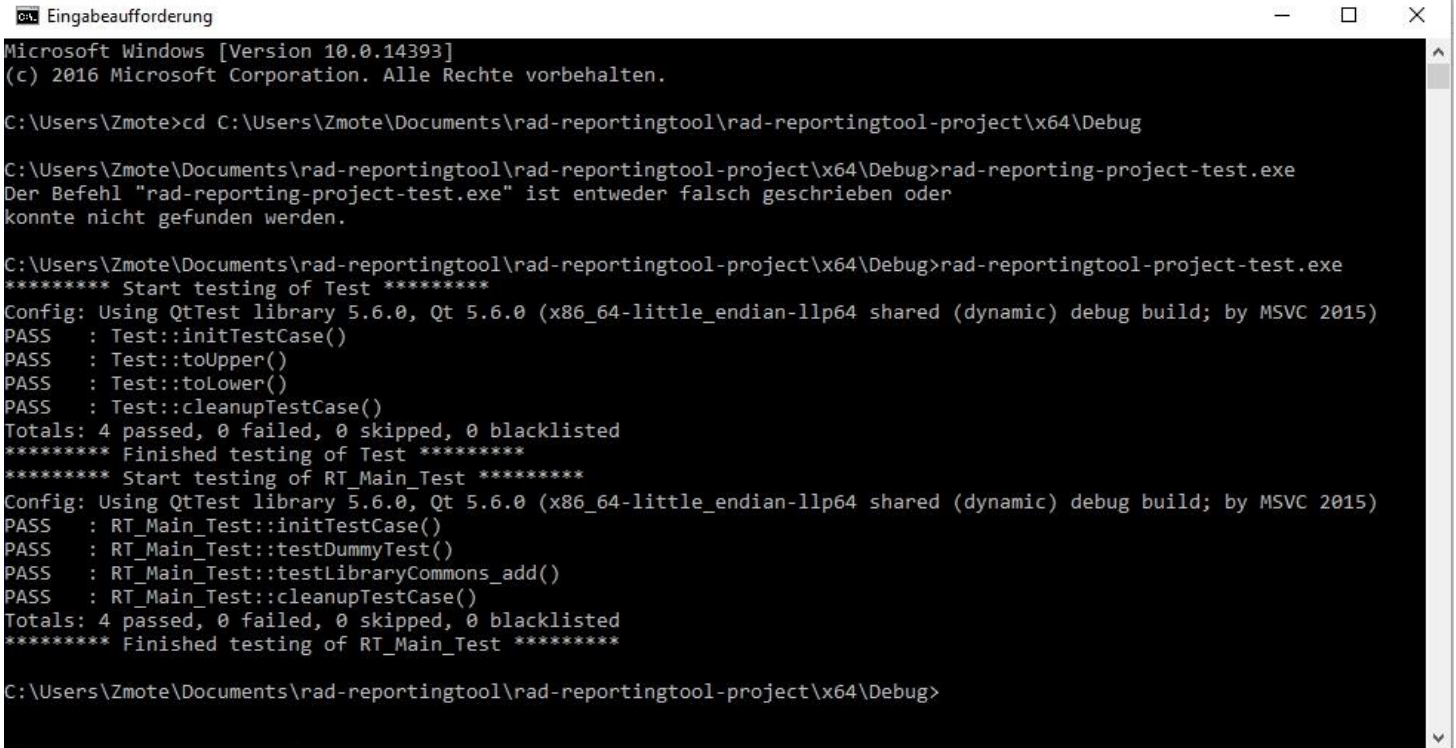
int main(int argc, char *argv[])
{
    int status{};
    Test test{};
    status |= QTest::qExec(&test, argc, argv);

    RT_Main_Test rt{};
    status |= QTest::qExec(&rt, argc, argv);

    return status;
}
```

Wobei "test.hpp" und "rt_main_test.hpp" von uns definierte Testklassen darstellen.

Per `QTest::qExec(..)` kann man eine Testklassen ausführen. Hat man zusätzlich zu den Qt5 Tools auch den Qt5 Test Adapter Extension installiert, so können die Tests auch über den Testexplorer ausgeführt werden. Jedoch ist das Tool buggy, im obigen Beispiel wird nur das erste `QTest::qExec(..)` angezeigt. Will man alle Ergebnisse der Tests sehen, so muss man das generierte Executable des Projekts (Projekt der Tests) ausführen und die Resultate werden auf die Command ausgegeben. Man könnte die Console Application auch direkt in Visual Studio laufen lassen und so die Testresultate einsehen, jedoch schliesst Visual Studio das Command sobald das Executable fertig ist, weswegen man nicht wirklich die Resultate betrachten kann. Darum rät es sich, ins Folder des Executable per Command - Tool zu navigieren und darin das Test-Executable laufen zu lassen. Dabei werden die Resultate wie folgt angezeigt:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Zmote>cd C:\Users\Zmote\Documents\rad-reportingtool\rad-reportingtool-project\x64\Debug

C:\Users\Zmote\Documents\rad-reportingtool\rad-reportingtool-project\x64\Debug>rad-reporting-project-test.exe
Der Befehl "rad-reporting-project-test.exe" ist entweder falsch geschrieben oder
konnte nicht gefunden werden.

C:\Users\Zmote\Documents\rad-reportingtool\rad-reportingtool-project\x64\Debug>rad-reportingtool-project-test.exe
***** Start testing of Test *****
Config: Using QtTest library 5.6.0, Qt 5.6.0 (x86_64-little_endian-llp64 shared (dynamic) debug build; by MSVC 2015)
PASS : Test::initTestCase()
PASS : Test::toUpper()
PASS : Test::toLower()
PASS : Test::cleanupTestCase()
Totals: 4 passed, 0 failed, 0 skipped, 0 blacklisted
***** Finished testing of Test *****
***** Start testing of RT_Main_Test *****
Config: Using QtTest library 5.6.0, Qt 5.6.0 (x86_64-little_endian-llp64 shared (dynamic) debug build; by MSVC 2015)
PASS : RT_Main_Test::initTestCase()
PASS : RT_Main_Test::testDummyTest()
PASS : RT_Main_Test::testLibraryCommons_add()
PASS : RT_Main_Test::cleanupTestCase()
Totals: 4 passed, 0 failed, 0 skipped, 0 blacklisted
***** Finished testing of RT_Main_Test *****

C:\Users\Zmote\Documents\rad-reportingtool\rad-reportingtool-project\x64\Debug>
```

Bild 47: Output des Testprojekts

Eine Testklasse kann wie folgt definiert sein:

Header (test.hpp):

```
#pragma once
#include <QObject>
#include <QTest>
```

```
class Test : public QObject {
    Q_OBJECT
```

public:

```
    Test(QObject * parent = Q_NULLPTR);
    ~Test();
```

private slots:

```
    void toUpper();
    void toLower();
```

```
};
```

Implementation (test.cpp):

```
#include "test.hpp"
#include "rt_main_test.hpp"
```

```
Test::Test(QObject * parent) : QObject(parent) {
```

```
}

Test::~Test() {

}

void Test::toLower()
{
    QString str = "HELLO";
    QVERIFY(str.toLower() == "hello");
}

void Test::toUpper() {
    QString str = "Hello";
    QVERIFY(str.toUpper() == "HELLO");
}
#include "moc_test.cpp"
```

Hier ist die letzte Zeile besonders wichtig. Für die interne Reflektion braucht Qt das moc-File der Testklasse, weshalb man es wie ob am Schluss der Impl. - Datei per #include anhängen muss.

Qt Testing Framework bietet vier zusätzliche, spezielle Methoden für jede Testklassen, die man überschreiben kann, um z.B. bestimmten Code vor Ausführung eines Tests auszuführen (Analogie @Before, @After etc. in JUnit, Java). Das sind die folgenden vier Klassen:

initTestCase() will be called before the first test function is executed.
cleanupTestCase() will be called after the last test function was executed.
init() will be called before each test function is executed.
cleanup() will be called after every test function.

Das Qt Testing Library erlaubt auch einfache GUI Tests, siehe dafür und mehr zu Qt Testing Library: <http://doc.qt.io/qt-5/qttest-index.html> (Qt Testing Tutorial page)

7.2 Qualitätsmanagement

7.2.1 Code Reviews

Code Reviews wurden nach bestimmten Meilensteinen informell im Freundeskreis und zu einem bestimmten Grad in den wöchentlichen Meetings an der HSR durchgeführt. Die Anpassungen im Code wurden dann als Tickets im Projektmanagementsystem festgehalten und in der laufenden Iteration umgesetzt. Da es zu keinen "formellen" Code Reviews gekommen ist, dienen die Protokolle als Code Review Dokumente, in denen Feedback zum Code teilweise enthalten sind.

7.2.2 Code Kommentare

Die Header - Dateien und im Code wichtige Stellen werden kommentiert. Für die Dokumentation wird Qt Documentation verwendet. Qt Dokumentationen sehen wie folgt aus (Doxygen - Format):

```
/*!
 * Resolves clientId to clientName for client log distribution
 * When the client can't be resolved, the function throws an invalid_argument
exception
 *
 * \param clientId Id of client a log belongs to
 * \returns the name of the client the log belongs to
 * \exception \ref std::invalid_argument
 */
QPair<QString, QString> resolveClient(const int clientId);
```

Dabei sind folgende Dokumentationstypen relevant:

<code>\param</code>	Für die Dokumentation der Parameter
<code>\returns</code>	Für die Dokumentation des Rückgabetyps
<code>\exception</code>	Für die Dokumentation der Exceptions, die eine Funktion werfen kann
<code>\ref</code>	Für die Referenz auf bestimmte Klassen innerhalb der Dokumentation
<code>\brief</code>	Für die Dokumentation einer kurzen Zusammenfassung
<code>\class</code>	Für die Dokumentation einer Klasse

7.2.3 Code Analyse

Für eine angemessene Code Qualität wurde das Tool ReSharper C++ von JetBrains verwendet, welche Codeoptimierungen vorschlägt und mögliche Probleme im Code aufzeigt. Ausserdem wird bei grösseren architekturellen Änderungen der Code mit informellen Codereviews überprüft und Anpassungen vorgenommen. Ein Vorschlag vom ReSharper C++ Tool kann wie in Bild 48 aussehen.

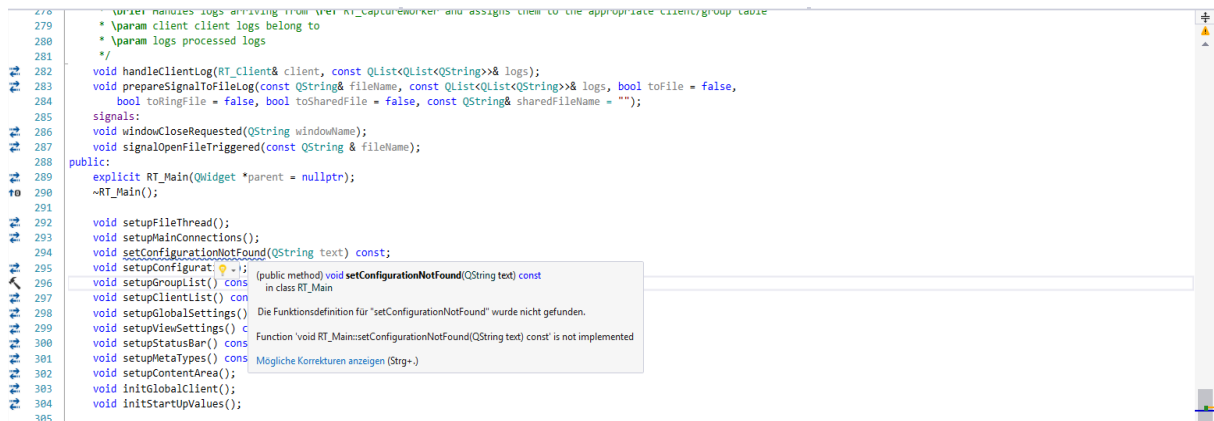


Bild 48: Optimierungsvorschlag ReSharper C++

7.2.4 Testing

Für das Reporting Tool wurden im Rahmen der Qualitätssicherung diverse Tests geschrieben. Dabei wurde schwerlastig mit dem Qt internen Testing Framework QTest gearbeitet. Das Testing Framework von Qt erlaubt unter anderem, die Testresultate als Xunit XML auszugeben, die man dann in einem Buildtool anzeigen und damit den Build kontrollieren kann (failing/passing Tests).

7.2.5 Unit Tests

Unit Tests in Qt sehen wie folgt aus:

```
1  #include "rt_main_test.hpp"
2  #include "Commons.h"
3
4  RT_Main_Test::RT_Main_Test(QObject * parent) : QObject(parent) {
5
6  }
7
8  RT_Main_Test::~~RT_Main_Test() {
9
10 }
11
12 void RT_Main_Test::testLibraryCommons_subtract()
13 {
14     Commons commons{};
15     QVERIFY(common.subtract(10, 2) == 8);
16 }
17
18 void RT_Main_Test::testLibraryCommons_add()
19 {
20     Commons commons{};
21     QVERIFY(common.add(1, 2) == 3);
22 }
23
24 void RT_Main_Test::testDummyTest()
25 {
26     QVERIFY(true);
27 }
```

Bild 49: RT_Maint_Test Source - Datei (rt_main_test.cpp) des rad-reportingtool-project-test Projekts

Wobei die Testmethoden im Header als private "Slots" definiert werden müssen:

```
1  #pragma once
2  #include <QObject>
3  #include <QTest>
4
5  class RT_Main_Test : public QObject {
6      Q_OBJECT
7
8  public:
9      RT_Main_Test(QObject * parent = Q_NULLPTR);
10     ~RT_Main_Test();
11
12 private slots:
13     static void testDummyTest();
14     static void testLibraryCommons_subtract();
15     static void testLibraryCommons_add();
16 };
```

Bild 50: RT_Main_Test Header - Datei (rt_main_test.hpp) des rad-reportingtool-project-test Projekts

Die Testobjekte in Qt müssen mit Qt's Metaobjekterweiterungen erweitert werden. Dies erreicht man, in dem man die Testklasse von der Klasse QObject ableitet und mit dem Makro Q_OBJECT kennzeichnet. Die Metadaten werden dann mit dem moc.exe (Meta-Objekt-Creator) generiert und an die bestehende Klasse angehängt. Dies ist wichtig, da das QTest Framework das "Signals and Slots" Konzept beim Ausführen der Teste anwendet und für diese die Metainformationen relevant sind.

Die Tests werden dann in einem separaten Test-Projekt (Qt Console Application) ausgeführt. Wobei das main wie folgt aussehen kann:

```
1 #include "test.hpp"
2 #include "rt_main_test.hpp"
3 #include "commonlogutils_test.hpp"
4 #include "commonviewutils_test.hpp"
5 #include "rt_client_test.hpp"
6 #include "rt_logfilter_test.hpp"
7
8 int main(int argc, char *argv[])
9 {
10     int status{};
11     Test test{};
12     status |= QTest::qExec(&test, argc, argv);
13
14     RT_Main_Test rt{};
15     status |= QTest::qExec(&rt, argc, argv);
16
17     CommonLogUtils_Test clu{};
18     status |= QTest::qExec(&clu, argc, argv);
19
20     CommonViewUtils_Test cvu{argc, argv};
21     status |= QTest::qExec(&cvu, argc, argv);
22
23     RT_Client_Test rct{argc, argv};
24     status |= QTest::qExec(&rct, argc, argv);
25
26     RT_LogFilter_Test rlft{ argc, argv };
27     status |= QTest::qExec(&rlft, argc, argv);
28
29     return status;
30 }
```

Bild 51: Main-Datei (main.cpp) des Testprojekts rad-reportingtool-project-test

Das QTest Framework erlaubt es, Tests in "Suite"s zu schreiben, sprich man kann die Tests gruppieren und in Gruppen zusammengefasst ausgeben. So kann man stets guten Überblick über die Anzahl und Typen von Tests behalten.

7.2.6 Integration Tests

Im Rahmen der Implementation der Tests für das ReportingTool Projekt wurden auch Integration Tests umgesetzt, in denen gewisse Teile der Applikation zusammenwirkend getestet wurden. Hierzu muss man zusätzlich jeweils eine QApplication initialisieren, die die Hauptereignisschleife einer Qt Applikation bildet. Ein Integrationstest kann dann wie in Bild 52 aussehen.

```
void RT_Client_Test::testAddLogToClientTable() const
{
    QApplication a{ argc, argv };
    RT_Client client{ 1, "Clientname", "Groupname" };
    const CL::ByteArray message{};
    const CL::Time timestamp{};
    REP::PROTO::LogBinaryMsg msg{ 1,message, timestamp };
    QList<QString> entry = CommonLogUtils::prepareLogEntry(client.getClientName(), client.getGroupName(), msg);
    QList<QList<QString>> logEntry{entry};
    client.addLog(logEntry);
    QCOMPARE(client.getTable()->model()->rowCount(), 1);
}
```

Bild 52: Integrationstest, füge Log einer Clienttabelle hinzu

Dieser Integrationstest ein einfaches Beispiel seiner Art. Hier wird beim Hinzufügen eines Logs in die Clienttabelle das Signals und Slots Mechanismus aktiviert, welches beim Hinzufügen von Daten ins Tabellenmodel Update Signale für die GUI produziert.

7.2.7 GUI Tests

GUI Tests werden implementiert, um das Verhalten der Benutzereingaben zu testen. So wird sichergestellt, dass bestimmte Benutzerinteraktion die erwarteten Resultate produzieren. Ein GUI Tests kann wie folgt aussehen:

```
void RT_Main_Test::testLogWindowCreation() const
{
    QApplication a(argc, argv);
    RT_Main w;
    REP::PROTO::RegisterClientMsg msg{ 2, "Clientname", "Groupname", "comment", false };
    w.registerClient(msg);
    w.createLogWindow("Clientname", QPoint(100, 100));
    QCOMPARE(w.subWindowExists("Clientname"), true);
}
```

Bild 53: GUI Test, erzeuge Logfenster

Qt erlaubt auch Mausklicks oder Tastatureingaben zu simulieren, diese werden dann per QTest::keyPress(...) oder QTest::mouseClick(...) simuliert. Ein GUI simulierter Test kann dann wie folgt aussehen:

```
void RT_Main_Test::testClientSearchTermEntered() const
{
    QApplication a(argc, argv);
    RT_Main w;
    QLineEdit * clientInput = w.findChild<QLineEdit*>("clientInput");
    QTest::keyClicks(clientInput, "client10");
    QCOMPARE(clientInput->text(), QString("client10"));
}
```

Bild 54: GUI simulierter Test

Auf diese Art und Weise können jegliche GUI Tests durchgeführt werden.

7.2.8 GitFlow

In der Entwicklung der Features wird nach dem GitFlow vorgegangen. Es wird nebst dem Master - Branch ein Develop - Branch verwendet, wo die umgesetzten Features zusammengeführt werden. Sind genug Features zusammengeführt, so dass man ein Release veröffentlichen kann, werden die Änderungen aufs Master gemergt. Features, Hotfixes wie auch Bugfixes werden in separaten Branches entwickelt und nach Beendigung der Umsetzung aufs Develop gemergt.

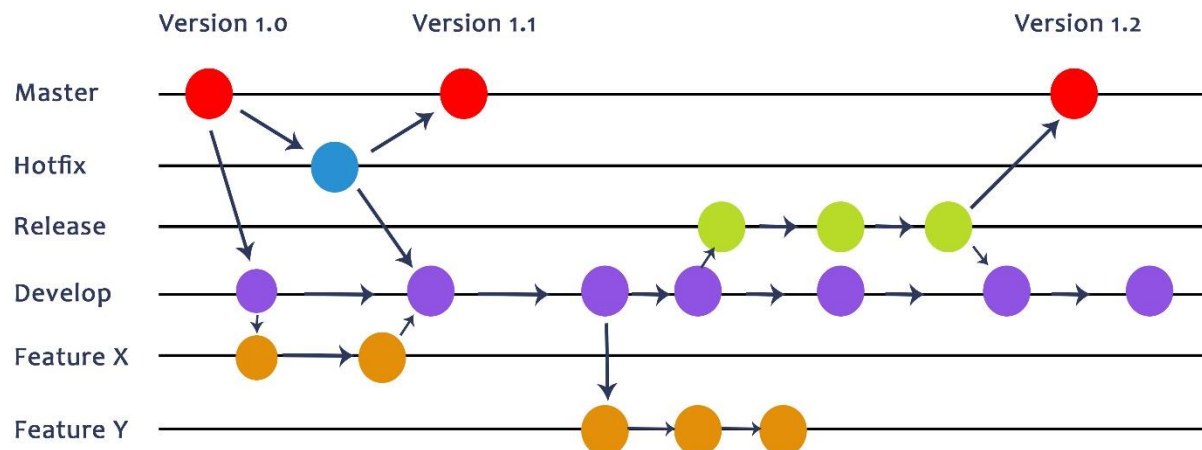


Bild 55: GitFlow Übersicht

7.2.9 Continuous Integration

Anhand Atlassian Bamboo wird Continuous Integration gewährleistet. Im Rahmen des GitFlow werden Features in eigenen Branches entwickelt und bei Fertigstellung mit dem develop Branch gemergt. Beim Merge auf den develop Branch produziert Bamboo die letzte, ausführbare Version der Applikation. So wird eine fortlaufende Integration neuer Features gewährleistet und stets ein ausführbares Executable geliefert.

7.2.10 Usability Tests

Da seitens des Industriepartners keine Termine für Usability Tests gesetzt werden konnten, da Sie selbst auch unter Zeitmangel leiden, wurden Usability Tests manuell umgesetzt und per Feedback an den Statusmeetings festgehalten. Diese wurden dann in JIRA als UI Anpassungen als Tickets festgehalten und im Laufe der Sprints umgesetzt.

7.2.11 Protokolle

Es werden zwei Arten von Protokolle festgehalten. Die wöchentlichen Protokolle der Statusmeetings an der HSR, an dem der Projektstand und der Stand der Applikation besprochen werden und die Statusmeetings beim Industriepartner, die alle zwei Wochen stattfinden. In diesen Protokollen werden jeweils der Status des Projekts und Probleme sowie Lösungen festgehalten. Ausserdem werden in diesen Protokollen Feedbacks festgehalten, welche bis zum Nächsten Meeting umgesetzt versucht werden.

7.3 Usability Tests

Es wurden aus Zeitmangel keine formellen Usability Tests durchgeführt. Der Industriepartner konnte jeweils alle zwei Wochen einen 1h Termin offerieren, in der der Status der Applikation besprochen wurde. In diesen Meetings wurde das UI diskutiert und Feedbacks entgegengenommen und für die nächste Iteration als "Anpassungen UI" als Ticket im Ticket Management System (JIRA) aufgenommen. Die verschiedenen Zustände des UI können im Abschnitt Design, im Unterabschnitt GUI nachgelesen werden.

Usability Tests wurden ferner informell im Freundeskreis durchgeführt und Feedback zum UI geholt. Diese wurden dann wie bei den Feedbacks des Industriepartners innerhalb der "UI Anpassungen" Tickets versucht, umgesetzt zu werden.

7.4 Testlogs

7.4.1 Alphaversion - 10/4/2017

ID	Title	Description	Date	Status	Bemerkung
0	Anzeige Clients	Clientsliste wird bei Start der Reporting Clients aktualisiert (Registration)	10/4/2017	OK	
1	Anzeige Gruppen	Gruppenliste wird bei Start der Reporting Clients aktualisiert	10/4/2017	OK	
2	Aktivierung Clients	Die Clients werden angewiesen, zu loggen, sobald das isActive flag per UI Checkbox "Active" gesetzt wird	10/4/2017	OK	
3	Live Routing Clients	Die Client Logs werden im UI visualisiert, so bald sie auf Active und Live gesetzt werden.	10/4/2017	OK	
4	To Parent Routing Clients	Clients routen ihre Logs zum Parent Objekt wenn Ihr Zustand auf ToParent gesetzt wird	10/4/2017	OK	
5	To File Routing Clients	Clients routen Ihre Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	10/4/2017	OK	
6	To Ring File Routing Clients	Clients routen Ihre Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	10/4/2017	OK	
7	To Global Routing Clients	Clients routen Ihre Logs zum global Output, wenn Ihr Zustand auf ToGlobal gesetzt wird	10/4/2017	OK	
8	Aktivierung von Groups	Groups verarbeiten zu Ihnen weitergereichte Logs, wenn Ihr Zustand auf Active gesetzt wird	10/4/2017	OK	
9	Live Routing Groups	Groups zeigen zu Ihnen weitergereichte Logs im UI an, wenn Ihr Zustand auf Live gesetzt wird	10/4/2017	OK	
10	To Parent Routing Groups	Groups routen Ihnen weitergereichte Logs zum Parent Objekt, was im Fall einer Gruppe der global Output ist	10/4/2017	OK	
11	To File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	10/4/2017	OK	

12	To Ring File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	10/4/2017	OK	
13	Select All Groups	Wird die Aktion (Menü) Select All Groups getriggert, werden all Groupclients selektiert	10/4/2017	OK	
14	Select All Clients	Wird die Aktion (Menü) select All Clients getriggert, werden all Clients selektiert	10/4/2017	OK	
15	Select All Active Clients	Wird die Aktion(Menü) select All Active Clients getriggert, werden alle aktiven Clients selektiert	10/4/2017	OK	
16	Show Inactive Clients	Wird die Aktion (Menü) select All Inactive Clients getriggert, werden alle inaktiven Clients selektiert	10/4/2017	OK	
17	Show Binary Clients	Wird die Aktion (Menü) show Binary Clients getriggert, werden alle Fenster von Binärclients angezeigt und die anderen minimiert	10/4/2017	OK	
18	Show Text Clients	Wird die Aktion (Menü) show Text Clients getriggert, werden alle Fenster von normalen Clients angezeigt und die anderen minimiert	10/4/2017	OK	
19	Reset Selected Clients	Wird die Aktion (Menü) reset Selected Clients getriggert, dann werden alle selektierten Clients auf Ihre Defaultwerte zurückgesetzt	10/4/2017	OK	
20	Reset All Clients	Wird die Aktion (Menü) reset All Clients getriggert, dann werden alle Clients auf Ihre Defaultwerte zurückgesetzt	10/4/2017	OK	
21	Arrange Windows	Wird die Aktion (Menü) Arrange Windows getriggert, dann werden die Logfenster am Monitor in 4er Gruppen angeordnet	10/4/2017	OK	
22	Assign Color	Wird die Aktion (Menü) assign Color getriggert, werden die Tabellen der selektierten Clients eingefärbt (Dialog erscheint zur Auswahl Farbe)	10/4/2017	OK	

23	Import	Wird die Aktion (Menü) Import triggered, so wird eine FileDialog geöffnet, von der aus man die Logdatei zum Laden auswählen kann. Der ausgewählte Log wird dann in einem unabhängigen Logfenster angezeigt	10/4/2017	OK	
24	Export	Wird die Aktion (Menü) export triggered, so wird ein FileDialog geöffnet, in dem man die Datei bestimmen kann, in die die Logs der aktuellen Tabelle (Tab) geschrieben werden.	10/4/2017	OK	
25	Enable/Disable Message Headers	Werden die bestimmte MessageHeader aktiviert/deaktiviert, so werden die entsprechend Spalten der Clienttabellen ein- oder ausgeblendet	10/4/2017	OK	
26	Enable/Disable Log Types	Werden die bestimmten Logtypes (enableInfo etc.) aktiviert/deaktiviert, so loggen die Clients (oder nicht) den ausgewählt Logtyp	10/4/2017	OK	
27	Drag'n'Drop Tab to Window	Drag'n'Drop einer Tab führt zu einem Logfenster	10/4/2017	OK	
28	Drag'n'Drop Window To Tab	Drag'n'Drop einer Tab in einem Logfenster in das QTabWidget des Hauptfenster führt zur Erzeugung eines neuen Tabs und der Logfenster wird geschlossen	10/4/2017	OK	
29	Enable Tab Mode	Wird das "Radio" auf Tabmode gesetzt, so geht ein Logfenster in die Tab-Ansicht über	10/4/2017	OK	
30	Enable Window Mode	Wird das "Radio" auf Window Mode gesetzt, so geht ein Clienttab in die Logfenster Ansicht über	10/4/2017	OK	
31	Minimize Window	Wird die Aktion (Menü) Minimize Window im Logfenster triggered, so wird dieses Fenster minimiert	10/4/2017	OK	
32	Minimize All	Wird die Aktion (Menü) Minimize All triggered(Hauptfenster/Logfenster), so werden alle Fenster ausser dem Hauptfenster minimiert	10/4/2017	OK	
33	Freeze Window	Wird die Operation Freeze Window aufgerufen, so werden Updates im aktiven Logfenster disabled	10/4/2017	OK	
34	Freeze All	Wird die Operation Freeze All aufgerufen, so werden für alle Clienttabellen Updates deaktiviert	10/4/2017	OK	

35	Test Performanz 20+ Windwos	Sind 20+ Client Logfenster offen, ist das UI immernoch responsive	10/4/2017	NOK	UI bei 10+ Clients unresponsive
36	Test Memory Management over 24h	Das Tool wird mit verschiedenen Routingoptionen für 24h laufen gelassen und der Speicherverbrauch überwacht. Bleibt der Speicherverbrauch ab einem Punkt stabil (Buffer Limite), so ist der Speicherverbrauch stabil	10/4/2017	NOK	Speicherverbrauch steigt kontinuierlich bei zu vielen ToRingFile Clients

7.4.2 Betaversion - 8/5/2017

ID	Title	Description	Date	Status	Bemerkung
0	Anzeige Clients	Clientsliste wird bei Start der Reporting Clients aktualisiert (Registration)	08/05/2017	OK	
1	Anzeige Gruppen	Gruppenliste wird bei Start der Reporting Clients aktualisiert	08/05/2017	OK	
2	Aktivierung Clients	Die Clients werden angewiesen, zu loggen, sobald das isActive flag per UI Checkbox "Active" gesetzt wird	08/05/2017	OK	
3	Live Routing Clients	Die Client Logs werden im UI visualisiert, so bald sie auf Active und Live gesetzt werden.	08/05/2017	OK	
4	To Parent Routing Clients	Clients routen ihre Logs zum Parent Objekt wenn Ihr Zustand auf ToParent gesetzt wird	08/05/2017	OK	
5	To File Routing Clients	Clients routen Ihre Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	08/05/2017	OK	
6	To Ring File Routing Clients	Clients routen Ihre Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	08/05/2017	OK	
7	To Global Routing Clients	Clients routen Ihre Logs zum global Output, wenn Ihr Zustand auf ToGlobal gesetzt wird	08/05/2017	OK	
8	Aktivierung von Groups	Groups verarbeiten zu Ihnen weitergereichte Logs, wenn Ihr Zustand auf Active gesetzt wird	08/05/2017	OK	
9	Live Routing Groups	Groups zeigen zu Ihnen weitergereichte Logs im UI an, wenn Ihr Zustand auf Live gesetzt wird	08/05/2017	OK	
10	To Parent Routing Groups	Groups routen Ihnen weitergereichte Logs zum Parent Objekt, was im Fall einer Gruppe der global Output ist	08/05/2017	OK	
11	To File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	08/05/2017	OK	

12	To Ring File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	08/05/2017	OK	
13	Select All Groups	Wird die Aktion (Menü) Select All Groups getriggert, werden all Groupclients selektiert	08/05/2017	OK	
14	Select All Clients	Wird die Aktion (Menü) select All Clients getriggert, werden all Clients selektiert	08/05/2017	OK	
15	Select All Active Clients	Wird die Aktion(Menü) select All Active Clients getriggert, werden alle aktiven Clients selektiert	08/05/2017	OK	
16	Show Inactive Clients	Wird die Aktion (Menü) select All Inactive Clients getriggert, werden alle inaktiven Clients selektiert	08/05/2017	OK	
17	Show Binary Clients	Wird die Aktion (Menü) show Binary Clients getriggert, werden alle Fenster von Binärclients angezeigt und die anderen minimiert	08/05/2017	OK	
18	Show Text Clients	Wird die Aktion (Menü) show Text Clients getriggert, werden alle Fenster von normalen Clients angezeigt und die anderen minimiert	08/05/2017	OK	
19	Reset Selected Clients	Wird die Aktion (Menü) reset Selected Clients getriggert, dann werden alle selektierten Clients auf Ihre Defaultwerte zurückgesetzt	08/05/2017	OK	
20	Reset All Clients	Wird die Aktion (Menü) reset All Clients getriggert, dann werden alle Clients auf Ihre Defaultwerte zurückgesetzt	08/05/2017	OK	
21	Arrange Windows	Wird die Aktion (Menü) Arrange Windows getriggert, dann werden die Logfenster am Monitor in 4er Gruppen angeordnet	08/05/2017	OK	
22	Assign Color	Wird die Aktion (Menü) assign Color getriggert, werden die Tabellen der selektierten Clients eingefärbt (Dialog erscheint zur Auswahl Farbe)	08/05/2017	OK	

23	Import	Wird die Aktion (Menü) Import triggered, so wird eine FileDialog geöffnet, von der aus man die Logdatei zum Laden auswählen kann. Der ausgewählte Log wird dann in einem unabhängigen Logfenster angezeigt	08/05/2017	OK	
24	Export	Wird die Aktion (Menü) export triggered, so wird ein FileDialog geöffnet, in dem man die Datei bestimmen kann, in die die Logs der aktuellen Tabelle (Tab) geschrieben werden.	08/05/2017	OK	
25	Enable/Disable Message Headers	Werden die bestimmte MessageHeader aktiviert/deaktiviert, so werden die entsprechend Spalten der Clienttabellen ein- oder ausgeblendet	08/05/2017	OK	
26	Enable/Disable Log Types	Werden die bestimmten Logtypes (enableInfo etc.) aktiviert/deaktiviert, so loggen die Clients (oder nicht) den ausgewählt Logtyp	08/05/2017	OK	
27	Drag'n'Drop Tab to Window	Drag'n'Drop einer Tab führt zu einem Logfenster	08/05/2017	OK	
28	Drag'n'Drop Window To Tab	Drag'n'Drop einer Tab in einem Logfenster in das QTabWidget des Hauptfenster führt zur Erzeugung eines neuen Tabs und der Logfenster wird geschlossen	08/05/2017	OK	
29	Enable Tab Mode	Wird das "Radio" auf Tabmode gesetzt, so geht ein Logfenster in die Tab-Ansicht über	08/05/2017	OK	
30	Enable Window Mode	Wird das "Radio" auf Window Mode gesetzt, so geht ein Clienttab in die Logfenster Ansicht über	08/05/2017	OK	
31	Minimize Window	Wird die Aktion (Menü) Minimize Window im Logfenster triggered, so wird dieses Fenster minimiert	08/05/2017	OK	
32	Minimize All	Wird die Aktion (Menü) Minimize All triggered(Hauptfenster/Logfenster), so werden alle Fenster ausser dem Hauptfenster minimiert	08/05/2017	OK	
33	Freeze Window	Wird die Operation Freeze Window aufgerufen, so werden Updates im aktiven Logfenster disabled	08/05/2017	OK	
34	Freeze All	Wird die Operation Freeze All aufgerufen, so werden für alle Clienttabellen Updates deaktiviert	08/05/2017	OK	

35	Test Performanz 20+ Windwos	Sind 20+ Client Logfenster offen, ist das UI immernoch responsive	08/05/2017	NOK	UI bei 10+ Clients unresponsive
36	Test Memory Management over 24h	Das Tool wird mit verschiedenen Routingoptionen für 24h laufen gelassen und der Speicherverbrauch überwacht. Bleibt der Speicherverbrauch ab einem Punkt stabil (Buffer Limite), so ist der Speicherverbrauch stabil	08/05/2017	NOK	Speicherverbrauch steigt kontinuierlich bei zu vielen ToRingFile Clients

7.4.3 Releaseversion - 23/5/2017

ID	Title	Description	Date	Status	Bemerkung
0	Anzeige Clients	Clientsliste wird bei Start der Reporting Clients aktualisiert (Registration)	23/5/2017	OK	
1	Anzeige Gruppen	Gruppenliste wird bei Start der Reporting Clients aktualisiert	23/5/2017	OK	
2	Aktivierung Clients	Die Clients werden angewiesen, zu loggen, sobald das isActive flag per UI Checkbox "Active" gesetzt wird	23/5/2017	OK	
3	Live Routing Clients	Die Client Logs werden im UI visualisiert, so bald sie auf Active und Live gesetzt werden.	23/5/2017	OK	
4	To Parent Routing Clients	Clients routen ihre Logs zum Parent Objekt wenn Ihr Zustand auf ToParent gesetzt wird	23/5/2017	OK	
5	To File Routing Clients	Clients routen Ihre Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	23/5/2017	OK	
6	To Ring File Routing Clients	Clients routen Ihre Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	23/5/2017	OK	
7	To Global Routing Clients	Clients routen Ihre Logs zum global Output, wenn Ihr Zustand auf ToGlobal gesetzt wird	23/5/2017	OK	
8	Aktivierung von Groups	Groups verarbeiten zu Ihnen weitergereichte Logs, wenn Ihr Zustand auf Active gesetzt wird	23/5/2017	OK	
9	Live Routing Groups	Groups zeigen zu Ihnen weitergereichte Logs im UI an, wenn Ihr Zustand auf Live gesetzt wird	23/5/2017	OK	
10	To Parent Routing Groups	Groups routen Ihnen weitergereichte Logs zum Parent Objekt, was im Fall einer Gruppe der global Output ist	23/5/2017	OK	
11	To File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	23/5/2017	OK	

12	To Ring File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	23/5/2017	OK	
13	Select All Groups	Wird die Aktion (Menü) Select All Groups getriggert, werden all Groupclients selektiert	23/5/2017	OK	
14	Select All Clients	Wird die Aktion (Menü) select All Clients getriggert, werden all Clients selektiert	23/5/2017	OK	
15	Select All Active Clients	Wird die Aktion(Menü) select All Active Clients getriggert, werden alle aktiven Clients selektiert	23/5/2017	OK	
16	Show Inactive Clients	Wird die Aktion (Menü) select All Inactive Clients getriggert, werden alle inaktiven Clients selektiert	23/5/2017	OK	
17	Show Binary Clients	Wird die Aktion (Menü) show Binary Clients getriggert, werden alle Fenster von Binärclients angezeigt und die anderen minimiert	23/5/2017	OK	
18	Show Text Clients	Wird die Aktion (Menü) show Text Clients getriggert, werden alle Fenster von normalen Clients angezeigt und die anderen minimiert	23/5/2017	OK	
19	Reset Selected Clients	Wird die Aktion (Menü) reset Selected Clients getriggert, dann werden alle selektierten Clients auf Ihre Defaultwerte zurückgesetzt	23/5/2017	OK	
20	Reset All Clients	Wird die Aktion (Menü) reset All Clients getriggert, dann werden alle Clients auf Ihre Defaultwerte zurückgesetzt	23/5/2017	OK	
21	Arrange Windows	Wird die Aktion (Menü) Arrange Windows getriggert, dann werden die Logfenster am Monitor in 4er Gruppen angeordnet	23/5/2017	OK	
22	Assign Color	Wird die Aktion (Menü) assign Color getriggert, werden die Tabellen der selektierten Clients eingefärbt (Dialog erscheint zur Auswahl Farbe)	23/5/2017	OK	

23	Import	Wird die Aktion (Menü) Import triggered, so wird eine FileDialog geöffnet, von der aus man die Logdatei zum Laden auswählen kann. Der ausgewählte Log wird dann in einem unabhängigen Logfenster angezeigt	23/5/2017	OK	
24	Export	Wird die Aktion (Menü) export triggered, so wird ein FileDialog geöffnet, in dem man die Datei bestimmen kann, in die die Logs der aktuellen Tabelle (Tab) geschrieben werden.	23/5/2017	OK	
25	Enable/Disable Message Headers	Werden die bestimmte MessageHeader aktiviert/deaktiviert, so werden die entsprechend Spalten der Clienttabellen ein- oder ausgeblendet	23/5/2017	OK	
26	Enable/Disable Log Types	Werden die bestimmten Logtypes (enableInfo etc.) aktiviert/deaktiviert, so loggen die Clients (oder nicht) den ausgewählt Logtyp	23/5/2017	OK	
27	Drag'n'Drop Tab to Window	Drag'n'Drop einer Tab führt zu einem Logfenster	23/5/2017	OK	
28	Drag'n'Drop Window To Tab	Drag'n'Drop einer Tab in einem Logfenster in das QTabWidget des Hauptfenster führt zur Erzeugung eines neuen Tabs und der Logfenster wird geschlossen	23/5/2017	OK	
29	Enable Tab Mode	Wird das "Radio" auf Tabmode gesetzt, so geht ein Logfenster in die Tab-Ansicht über	23/5/2017	OK	
30	Enable Window Mode	Wird das "Radio" auf Window Mode gesetzt, so geht ein Clienttab in die Logfenster Ansicht über	23/5/2017	OK	
31	Minimize Window	Wird die Aktion (Menü) Minimize Window im Logfenster triggered, so wird dieses Fenster minimiert	23/5/2017	OK	
32	Minimize All	Wird die Aktion (Menü) Minimize All triggered(Hauptfenster/Logfenster), so werden alle Fenster ausser dem Hauptfenster minimiert	23/5/2017	OK	
33	Freeze Window	Wird die Operation Freeze Window aufgerufen, so werden Updates im aktiven Logfenster disabled	23/5/2017	OK	
34	Freeze All	Wird die Operation Freeze All aufgerufen, so werden für alle Clienttabellen Updates deaktiviert	23/5/2017	OK	

35	Test Performanz 20+ Windows	Sind 20+ Client Logfenster offen, ist das UI immer noch responsive	23/5/2017	OK	
36	Test Memory Management over 24h	Das Tool wird mit verschiedenen Routingoptionen für 24h laufen gelassen und der Speicherverbrauch überwacht. Bleibt der Speicherverbrauch ab einem Punkt stabil (Buffer Limite), so ist der Speicherverbrauch stabil	23/5/2017	NOK	Speicherverbrauch steigt kontinuierlich bei zu vielen ToRingFile Clients
37	Clients ToMergeView	Clients mit Zustand ToMergeView routen Ihre Logs zur MergeView	23/5/2017	OK	
38	Clients/Groups ToSharedFile	Clients/Groups mit Zustand ToSharedFile routen Ihre Logs in eine geteilte Datei	23/5/2017	OK	
39	Persistenz	Applikationseinstellungen sind persistierbar und können geladen und angewendet werden	23/5/2017	OK	

7.4.4 Abgabeverision - 1/6/2017

ID	Title	Description	Date	Status	Bemerkung
0	Anzeige Clients	Clientsliste wird bei Start der Reporting Clients aktualisiert (Registration)	1/6/2017	OK	
1	Anzeige Gruppen	Gruppenliste wird bei Start der Reporting Clients aktualisiert	1/6/2017	OK	
2	Aktivierung Clients	Die Clients werden angewiesen, zu loggen, sobald das isActive flag per UI Checkbox "Active" gesetzt wird	1/6/2017	OK	
3	Live Routing Clients	Die Client Logs werden im UI visualisiert, so bald sie auf Active und Live gesetzt werden.	1/6/2017	OK	
4	To Parent Routing Clients	Clients routen ihre Logs zum Parent Objekt wenn Ihr Zustand auf ToParent gesetzt wird	1/6/2017	OK	
5	To File Routing Clients	Clients routen Ihre Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	1/6/2017	OK	
6	To Ring File Routing Clients	Clients routen Ihre Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	1/6/2017	OK	
7	To Global Routing Clients	Clients routen Ihre Logs zum global Output, wenn Ihr Zustand auf ToGlobal gesetzt wird	1/6/2017	OK	
8	Aktivierung von Groups	Groups verarbeiten zu Ihnen weitergereichte Logs, wenn Ihr Zustand auf Active gesetzt wird	1/6/2017	OK	
9	Live Routing Groups	Groups zeigen zu Ihnen weitergereichte Logs im UI an, wenn Ihr Zustand auf Live gesetzt wird	1/6/2017	OK	
10	To Parent Routing Groups	Groups routen Ihnen weitergereichte Logs zum Parent Objekt, was im Fall einer Gruppe der globale Output ist	1/6/2017	OK	
11	To File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine einfache Datei, wenn Ihr Zustand auf ToFile gesetzt wird	1/6/2017	OK	
12	To Ring File Routing Groups	Groups routen Ihnen weitergereichte Logs in eine Ringdatei, wenn Ihr Zustand auf ToRingFile gesetzt wird	1/6/2017	OK	
13	Select All Groups	Wird die Aktion (Menü) Select All Groups getriggert, werden all Groupclients selektiert	1/6/2017	OK	

14	Select All Clients	Wird die Aktion (Menü) select All Clients triggered, werden all Clients selektiert	1/6/2017	OK	
15	Select All Active Clients	Wird die Aktion(Menü) select All Active Clients triggered, werden alle aktiven Clients selektiert	1/6/2017	OK	
16	Show Inactive Clients	Wird die Aktion (Menü) select All Inactive Clients triggered, werden alle inaktiven Clients selektiert	1/6/2017	OK	
17	Show Binary Clients	Wird die Aktion (Menü) show Binary Clients triggered, werden alle Fenster von Binärclients angezeigt und die anderen minimiert	1/6/2017	OK	
18	Show Text Clients	Wird die Aktion (Menü) show Text Clients triggered, werden alle Fenster von normalen Clients angezeigt und die anderen minimiert	1/6/2017	OK	
19	Reset Selected Clients	Wird die Aktion (Menü) reset Selected Clients triggered, dann werden alle selektierten Clients auf Ihre Defaultwerte zurückgesetzt	1/6/2017	OK	
20	Reset All Clients	Wird die Aktion (Menü) reset All Clients triggered, dann werden alle Clients auf Ihre Defaultwerte zurückgesetzt	1/6/2017	OK	
21	Arrange Windows	Wird die Aktion (Menü) Arrange Windows triggered, dann werden die Logfenster am Monitor in 4er Gruppen angeordnet	1/6/2017	OK	
22	Assign Color	Wird die Aktion (Menü) assign Color triggered, werden die Tabellen der selektierten Clients eingefärbt (Dialog erscheint zur Auswahl Farbe)	1/6/2017	OK	
23	Import	Wird die Aktion (Menü) Import triggered, so wird ein OpenFileDialog geöffnet, von der aus man die Logdatei zum Laden auswählen kann. Das ausgewählte Log wird dann in einem unabhängigen Logfenster angezeigt	1/6/2017	OK	
24	Export	Wird die Aktion (Menü) export triggered, so wird ein OpenFileDialog geöffnet, in dem man die Datei bestimmten kann, in die die Logs der aktuellen Tabelle (Tab) geschrieben werden.	1/6/2017	OK	

25	Enable/Disable Message Headers	Werden die bestimmte MessageHeader aktiviert/deaktiviert, so werden die entsprechend Spalten der Clienttabellen ein- oder ausgeblendet	1/6/2017	OK	
26	Enable/Disable Log Types	Werden die bestimmten Logtypes (enableInfo etc.) aktiviert/deaktiviert, so loggen die Clients (oder nicht) den ausgewählte Logtyp	1/6/2017	OK	
27	Drag'n'Drop Tab to Window	Drag'n'Drop einer Tab führt zu einem Logfenster	1/6/2017	OK	
28	Drag'n'Drop Window To Tab	Drag'n'Drop einer Tab in einem Logfenster in das QTabWidget des Hauptfenster führt zur Erzeugung eines neuen Tabs und der Logfenster wird geschlossen	1/6/2017	OK	
29	Enable Tab Mode	Wird das "Radio" auf Tabmode gesetzt, so geht ein Logfenster in die Tab-Ansicht über	1/6/2017	OK	
30	Enable Window Mode	Wird das "Radio" auf Window Mode gesetzt, so geht ein Clienttab in die Logfenster Ansicht über	1/6/2017	OK	
31	Minimize Window	Wird die Aktion (Menü) Minimize Window im Logfenster getriggert, so wird dieses Fenster minimiert	1/6/2017	OK	
32	Minimize All	Wird die Aktion (Menü) Minimize All getriggert(Hauptfenster/Logfenster), so werden alle Fenster ausser dem Hauptfenster minimiert	1/6/2017	OK	
33	Freeze Window	Wird die Operation Freeze Window aufgerufen, so werden Updates im aktiven Logfenster disabled	1/6/2017	OK	
34	Freeze All	Wird die Operation Freeze All aufgerufen, so werden für alle Clienttabellen Updates deaktiviert	1/6/2017	OK	
35	Test Performanz 20+ Windwos	Sind 20+ Client Logfenster offen, ist das UI immernoch responsive	1/6/2017	OK	
36	Test Memory Management over 24h	Das Tool wird mit verschiedenen Routingoptionen für 24h laufen gelassen und der Speicherverbrauch überwacht. Bleibt der Speicherverbrauch ab einem Punkt stabil (Buffer Limite), so ist der Speicherverbrauch stabil	1/6/2017	OK	

37	Clients ToMergeView	Clients mit Zustand ToMergeView routen Ihre Logs zur MergeView	1/6/2017	OK	
38	Clients/Groups ToSharedFile	Clients/Groups mit Zustand ToSharedFile routen Ihre Logs in eine geteilte Datei	1/6/2017	OK	
39	Persistenz	Applikationseinstellungen sind persistierbar und können geladen und angewendet werden	1/6/2017	OK	

7.5 Use Cases (Essential Style)

Hier werden die wichtigsten der Use Cases in Essential Style Form aufgelistet.

7.5.1 UC 1 Logging

7.5.1.1 Preconditions

Master Instanz verarbeitet eingehende Lognachrichten.

7.5.1.2 Beschreibung

Logs können eingesehen, gespeichert und geparkt werden. Werden die Log - Files in eine Datei geschrieben, so kann diese Datei geladen und angezeigt werden. Das Parsing für die Logdateien betrifft die binären Logs, welche vom binären Format in das HEX - Format geparkt werden sollen.

7.5.1.3 Postconditions

Logs können im Reporting Tool eingesehen und bei Bedarf gespeichert werden.

7.5.1.4 Main Scenario

- 1 Reporting Tool starten
- 2 Live-Feed Modus wählen
- 3 Verfügbare Gruppen oder Klienten wählen (oder Kombination dieser zwei)
- 4 Capture starten (Anzeige in Fenstern)
- 5 Logs lesen.

7.5.1.5 Alternative

- 2a To File Modus wählen

7.5.2 UC5 Logging suchen und filtern

7.5.2.1 Preconditions

Logs werden empfangen und angezeigt

7.5.2.2 Beschreibung

Anhand Suchkriterien/Filter können Logs gefiltert werden (gefilterte Anzeigen)

7.5.2.3 Postconditions

Logs haben gefilterte Ansicht (auf einzelne oder mehrere Logs, je nach Anwendung)

7.5.2.4 Main Scenario

1 Suchkriterium im Suchfeld eingeben

7.5.2.5 Alternative

1a Suchkriterium aus History auswählen

1b Filter aus Filtersektion setzen (z.B. Enable Info Messages)

1c Filter per Ausdruck im Suchfeld eingeben (z.B. timestamp == 26.03.2017)

8 Anhang C

8.1 Installationsanleitung

Die Applikation wird als ein ausführbares Executable mit den nötigen Abhängigkeiten (Bibliotheken, Bilder etc.) in Form einer ZIP - Datei ausgeliefert. Für die "Installation" muss diese ZIP - Datei lediglich extrahiert werden. Per .bat - Datei, das ich in der ZIP - Datei befindet, kann die Applikation gestartet werden. Aber auch per rad-reportingtool-project.exe lässt sich die Applikation starten. Erstellt man eine Verknüpfung zum .EXE, kann die Applikation von beliebiger Stelle aus gestartet werden.

8.2 Benutzerhandbuch

In diesem Abschnitt werden die wichtigsten Funktionen der Applikation und Ihrer Bedienung beschrieben.

8.2.1 Shortcuts

Hauptfenster	
Ctrl + O	Import
Ctrl + S	Export
Ctrl + L	Load Configuration...
Ctrl + Q	Close (Application)
F1	Help.. (API Dokumentation)
F2	Select All Groups
F3	Select All Clients
F4	Select All (active)
F5	Reset Selected Clients
F6	Reset All Clients
F7	Clear All MergeViews
F9	Arrange Windows
Ctrl + F	Freeze All
Ctrl + U	Unfreeze All
Ctrl + E	Clear Current Table
Ctrl + R	Clear All Tables
Ctrl + I	Show
Ctrl + B	Show Binary Clients
Ctrl + T	Show Text Clients
Ctrl + Y	Assign Color To Selected...
Ctrl + N	Minimize All (Windows)
Ctrl + M	Maximize All (Windows)
Alt + W	Close All Tabs
Ctrl + W	Close All Windows
Ctrl + P	Settings..
Ctrl + Alt + S	Restore Default Settings
Ctrl + F1	About

Logfenster	
Ctrl + S	Export
Ctrl + Q	Close (Application)
Ctrl + E	Clear Table
Ctrl + N	Minimize
Ctrl + W	Minimize All
F9	Arrange Windows

8.2.2 Client aktivieren und im Tab anzeigen

Wählen Sie einen Client/Group aus der entsprechenden Liste aus (Selektion) und aktivieren Sie im Bereich der Client/Group Zustände die Checkbox «Active» und den Client/Group Status «Live», wie im Bild 56 vorgezeigt. Beachten Sie, dass Ihr ausgewählter Client im Tab Mode ist (Tab – Mode Radio).

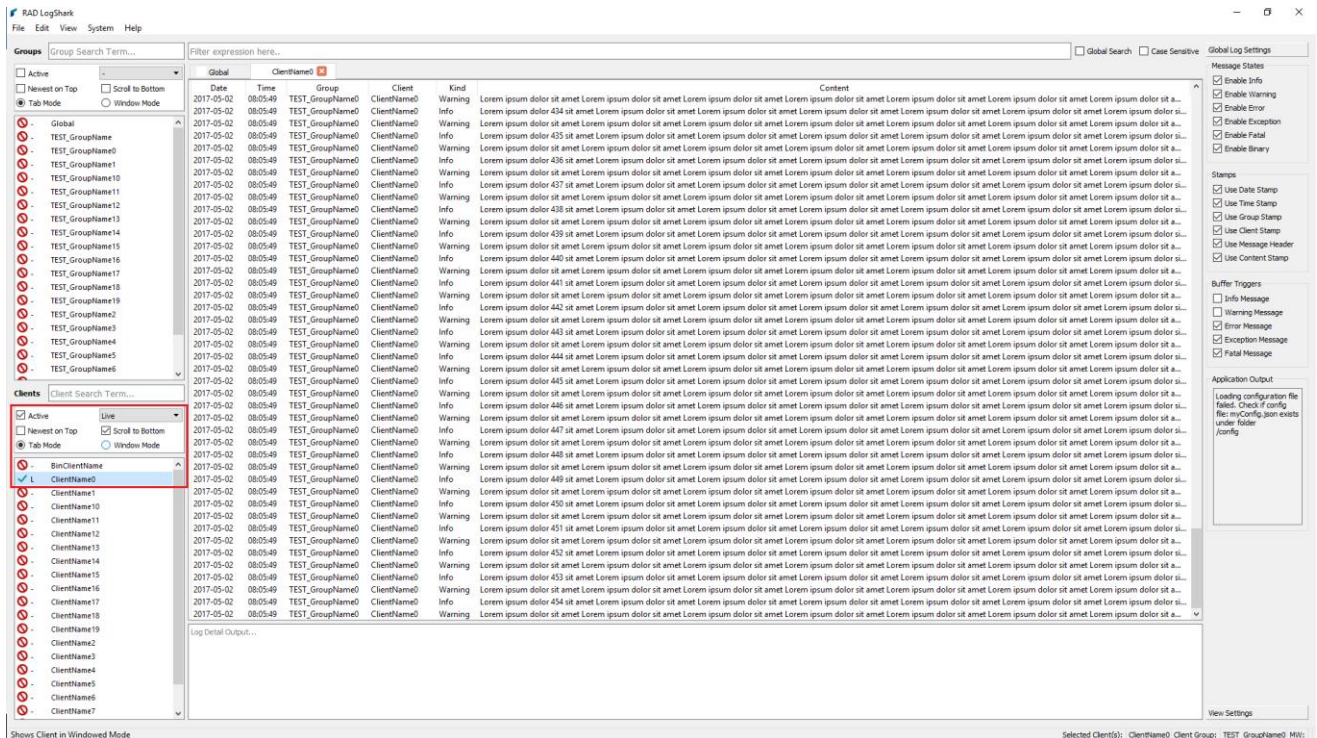


Bild 56: Client aktivieren und im Tab anzeigen

8.2.3 Client in den Fenstermodus wechseln lassen

Wählen Sie einen Client/Group aus der entsprechenden Liste und aktivieren Sie das Radio «Window Mode» wie im Bild 57. Die Clients/Groups wechseln in den Fenstermodus.

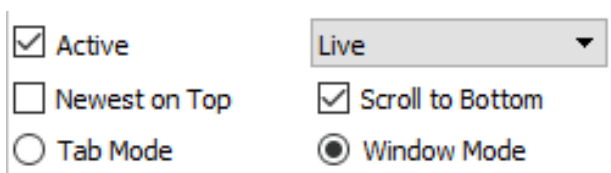


Bild 57: Window – Mode Radio im Zustandsbereich

8.2.4 Client Logs routen

In diesem Abschnitt werden verschiedene Routing Optionen im Bezug zu den Clients erklärt.

8.2.4.1 Live

Um einen Client «Live» zu routen, also im UI zu visualisieren, müssen Sie im Zuständebereich der Clients/Groups die ausgewählten Clients/Groups auf «Live» setzen. Achtung: Ein Client/Group rouet nur in die UI, wenn der Client/Group auch aktiv ist. Sie müssen also auch den Zustand «Active» setzen.

8.2.4.2 To File

Um einen Client «To File» zu routen, als die Lognachrichten in eine einfache Datei zu schreiben, müssen Sie im Zuständebereich der Clients/Groups die ausgewählten Clients/Groups auf «ToFile» setzen. Achtung: Ein Client/Group routet nur ins File, wenn der Client/Group auch aktiv ist. Sie müssen also auch den Zustand «Active» setzen.

8.2.4.3 To RingFile

Um einen Client «ToRingFile» zu routen, als die Lognachrichten in eine Ringdatei zu schreiben, müssen Sie im Zuständebereich der Clients/Groups die ausgewählten Clients/Groups auf «ToRingFile» setzen. Achtung: Ein Client/Group routet nur ins File, wenn der Client/Group auch aktiv ist. Sie müssen also auch den Zustand «Active» setzen.

8.2.4.4 To SharedFile

Um einen Client «ToSharedFile» zu routen, als die Lognachrichten in eine geteilte Datei zu schreiben, müssen Sie im Zuständebereich der Clients/Groups die ausgewählten Clients/Groups auf «ToSharedFile» setzen. Beim Setzen des Zustands auf «ToSharedFile» wird nach dem Namen der geteilten Datei gefragt; hier kann ein neuer Dateiname oder ein schon bereits bestehender Dateiname eingegeben werden. Achtung: Ein Client/Group routet nur ins File, wenn der Client/Group auch aktiv ist. Sie müssen also auch den Zustand «Active» setzen.

8.2.4.5 To Parent

Um einen Client «ToParent» zu routen, also zum Eltern – Objekt weiterreichen, müssen Sie im Zuständebereich der Clients/Groups die ausgewählten Clients/Groups auf «ToParent» setzen. Hierbei gilt zu beachten, dass bei Groups das Eltern – Objekt die globale Ausgabe ist. Achtung: Ein Client/Group routet nur zum Eltern - Objekt, wenn der Client/Group auch aktiv ist. Sie müssen also auch den Zustand «Active» setzen. Zudem muss das Eltern – Objekt ebenfalls auf «Active» und «Live» sein, möchte man die Logs im UI sehen.

8.2.4.6 To Global

Um einen Client «ToGlobal» zu routen, also um Lognachrichten zur globalen Ausgabe weiterzureichen, müssen Sie im Zuständebereich der Clients/Groups die ausgewählten Clients/Groups auf «ToGlobal» setzen. Achtung: Ein Client/Group routet nur in die globale Ausgabe, wenn der Client/Group auch aktiv ist. Sie müssen also auch den Zustand «Active» setzen. Zudem muss der Global – Client auch auf «Active» und «Live» gesetzt sein, damit die Logs im UI sichtbar werden.

8.2.4.7 To MergeView

Um einen Client «ToMergeView» zu routen, also um Lognachrichten zur globalen Ausgabe weiterzureichen, müssen Sie im Zuständebereich der Clients die ausgewählten Clients auf «ToMergeView» setzen. Beim Setzen des Zustands wird man nach dem Namen der MergeView gefragt; dieser kann ein neuer oder ein schon bestehender sein. Besteht die MergeView nicht, wird unter den Groups ein neuer Groupclient mit dem Präfix MERGEVIEW_ erzeugt. Die Lognachrichten werden dann zu diesem «Groups» - Objekt weitergereicht. Achtung: Ein Client routet nur zur MergeView, wenn der Client auch aktiv ist. Sie müssen also auch den Zustand «Active» setzen. Zudem muss der MergeView – Client auch auf «Active» und «Live» gesetzt sein, damit die Logs im UI sichtbar werden. MergeView ist nur auf Client – Ebene vorhanden und existiert für Groups nicht.

8.2.5 Filter auf Clienttabelle anwenden

Um einen Filter auf eine Clienttabelle anzuwenden, gibt man im Filterausdruckbereich einen Filterausdruck ein und bestätigt die Eingabe mit «Enter». Ist der Filterausdruck korrekt, färbt sich das Feld blau, ist der Filterausdruck falsch, erscheint eine MessageBox mit der Nachricht, den Filterausdruck zu korrigieren und das Feld bleibt rot gefärbt. Im Bild 58 ist ein Beispiel aufgeführt, dass eine erfolgreiche Filterung zeigt.

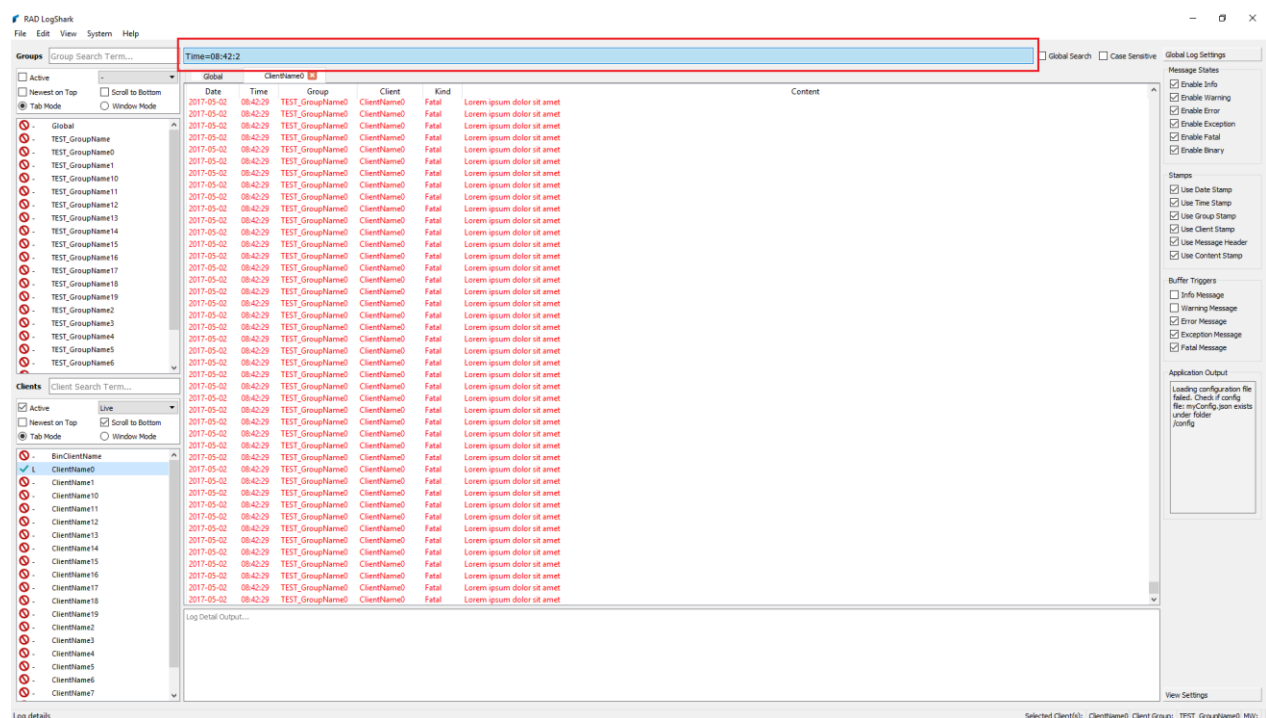


Bild 58: Anwendung eines einfachen Filters.

8.2.6 Globale Log Einstellungen vornehmen

Um globale Log Einstellungen vorzunehmen, müssen Sie im Global Log Settings Bereich der Toolbox zur Rechten die Checkboxes für Message States ändern. Bild 59 zeigt ein Beispiel für globale Log Einstellungen, wo Clients nur noch Info – Logs loggen.

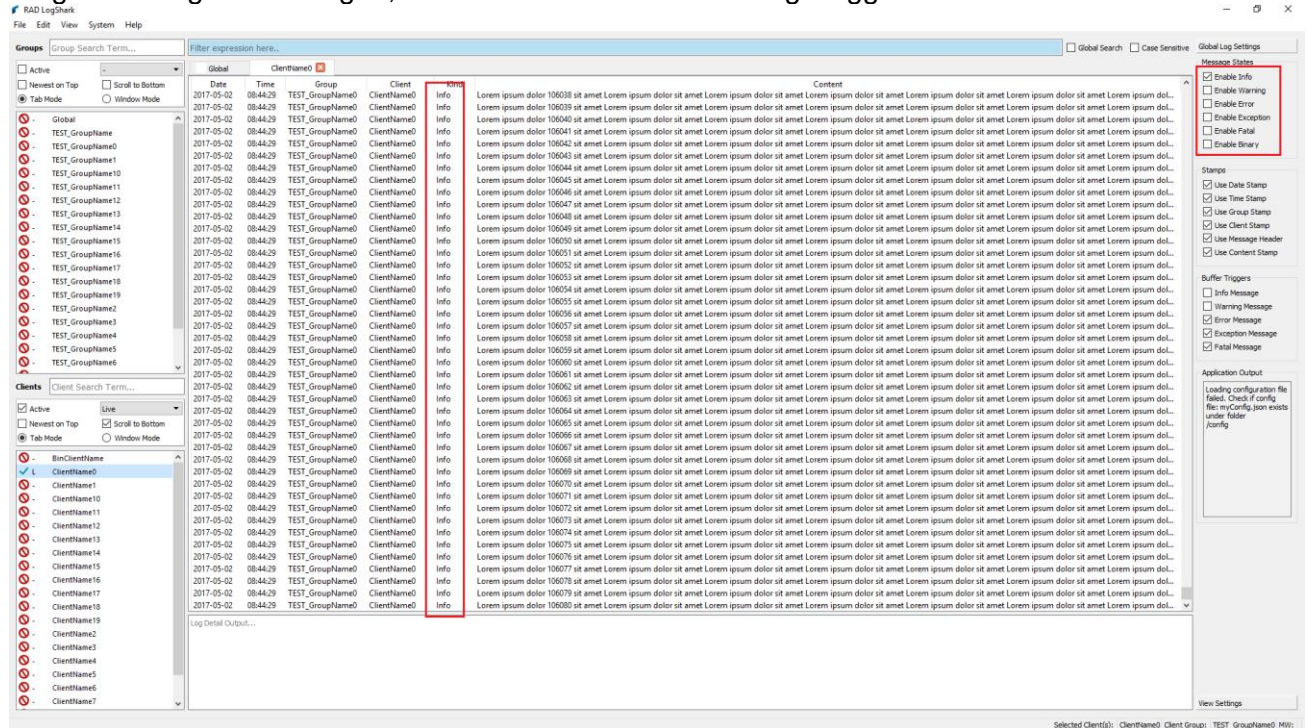


Bild 59: Globale Log Einstellungen auf «Nur Info Logs» gesetzt

8.2.7 View Elemente ein- und ausblenden

Wie bei den globalen Logeinstellungen, befinden sich die Steuerelemente für das Ein- und Ausblenden bestimmter Viewelemente in der Toolbox zur Rechten des Hauptfensters. Unter der Seite «View Settings» sind Checkboxes wie «Show Group Section» etc. vorhanden, anhand derer man Elemente im View ein- und ausblenden kann.

8.2.8 Client Logs exportieren

Um Client Logs im Hauptfenster zu exportieren, muss lediglich der Tab des Clients sichtbar sein, von dem man die Logs exportieren will. Danach klickt man auf das Menü «File» und wählt «Export». Ein FileDialog erscheint und fragt Sie nach der Ausgabedatei. Hierbei genügt es die Lokation und den Dateinamen anzugeben. Die sichtbaren Logs der Clienttabelle werden in diese Datei exportiert.

8.2.9 Client Logs importieren

Um Client Logs zu importieren, müssen Sie im Hauptfenster auf das Menü «File» und wählt «Import». Ein FileDialog erscheint, anhand derer man die Logdatei zum Import auswählt. Die Logdatei wird nach erfolgreicher Import-Operation in einem unabhängigen Logfenster angezeigt.

8.2.10 Applikationseinstellungen ändern und persistieren

Möchte man die Applikationseinstellungen ändern und persistieren, so muss man lediglich die Einstellungen in der Applikation vornehmen, wie z.B. die globalen Logeinstellungen setzen oder bestimmte Viewelemente ein- oder ausblenden. Globale Einstellungen wie Ausgabepfade werden vom Einstellungen - Fenster zur Änderung angeboten. Diesen startet man per Menü «System», «Settings...». Das Dialog, dass erscheint, dient zur Änderung von globalen Werten und erlaubt das Persistieren der Einstellungen, inkl. Globalen Logeinstellungen, Vieweinstellungen und Clients/Groups mit Client/Group Zuständen. Das Einstellungsfenster sieht wie in Bild 60 aus.

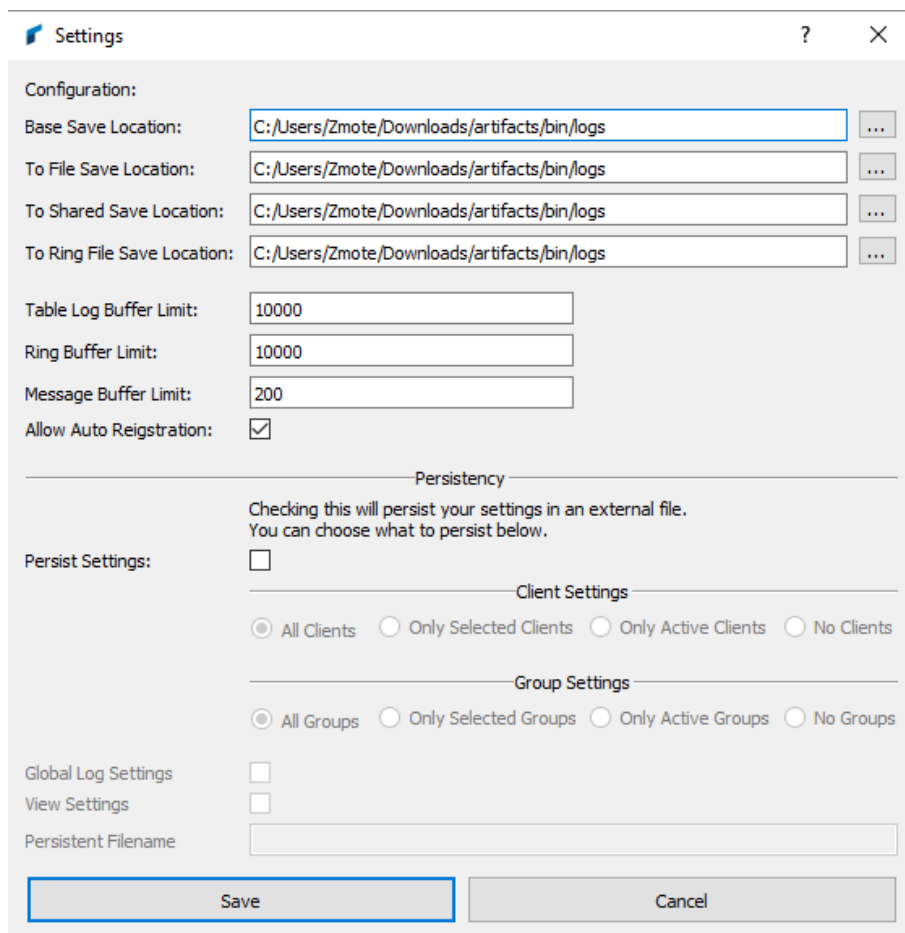


Bild 60: Einstellungen - Fenster

8.2.11 Applikationseinstellungen laden

Applikationseinstellungen können über das Menü «File» und «Load Configurations» Menüpunkt geladen und angewendet werden. Im FileDialog, das erscheint, wählt man die Konfigurationsdatei, die im JSON – Format sein muss, und die Applikation passt sich automatisch an die angegebenen Einstellungen an.

8.2.12 API Dokumentation einsehen

Klicken Sie auf F1 oder wählen Sie unter dem Menüpunkt «Help» den Eintrag «Help». Die API Dokumentation wird in einem Browser geöffnet.