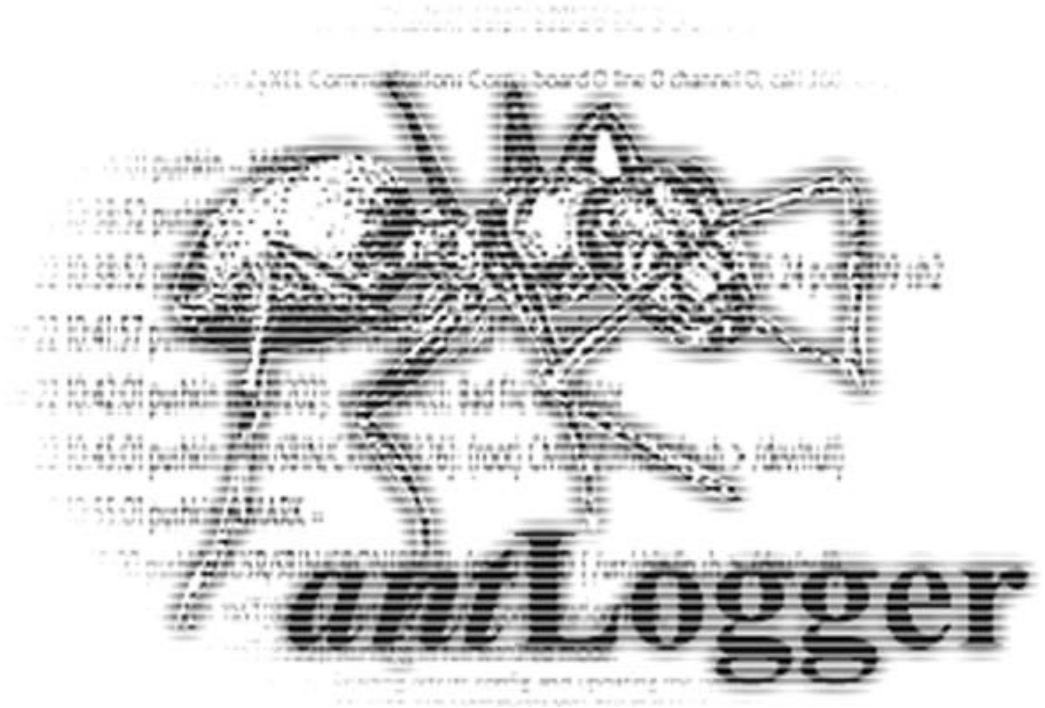
FHZ —> FACHHOCHSCHULE ZENTRALSCHWEIZ

HTA

HTA —> HOCHSCHULE FÜR TECHNIK+ARCHITEKTUR LUZERN

Abteilung Informatik->Software Engineering->Projekt AntLogger

Software Engineering Projekt



Software Requirements Specification
SRS

Version 1.0

Patrick Bründler, Pascal Mengelt, Andy Wyss, Fabian Heusser

Dozent: Jörg Hofstetter

HTA LUZERN
Technikumstrasse 21
CH—6048 Horw

T: 041—349—33—11
F: 041—349—39—60
W: www.hta.fhz.ch

History

Nr	Datum	Name	Beschreibung
1	27.3.02	men	Erster Entwurf
2	22.4.02	men	Version 1.0
3	22.4.02	bru	Review

Inhalt

1 Einführung

1.1 Absicht

Dieses Dokument beinhaltet alle Anforderungen für den *antLogger* Message Logger. Die Kapitel 1 und 2 sind besonders für den Kunden interessant (C-Requirements). Kapitel 3 ist hauptsächlich für den Software Ingenieur (D-Requirements).

1.2 Umfang

Dieses Dokument behandelt die Anforderungen für den Release 1.0 des *antLogger*, ein Message Logger für und in Java. Das SRS soll ein Hilfe für den Kunden, wie auch für den Software Entwickler sein während der Entwicklung des Systemes.

1.3 Definitionen, Begriffe und Abkürzungen

Begriff oder Term	Beschreibung
C-Requirement	Kunden- Anforderung (Customer): Erklärung einer Anforderung für die Anwendung, ausgedrückt in einer Form, welche dem Kunden klar ist.
D-Requirement	Entwickler-Anforderung (Developer): Erklärung einer Anforderung für die Anwendung, ausgedrückt in einer Form, welche genügend detailliert ist, um von den Entwicklern für das Design und die Implementation benutzt zu werden.
Message Logger	Software um Nachrichten entgegenzunehmen und diese zu speichern, sowie dem Benutzer zugänglich zu machen.
antLogger	Name des Message Logger.
Client	Ein Programm welches Nachrichten an den antLogger absetzen kann.
Log-Viewer	Ein Werkzeug, um die gesammelten Nachrichten anzeigen zu können.

1.4 Verweise

- Software Configuration Management Plan (SCMP)
- Software Project Management Plan (SPMP)
- Software Design Document (SDD)

1.5 Überblick

Das SRS ist gemäss dem IEEE Standart organisiert. (Siehe Inhaltsverzeichnis)

2 Allgemeine Beschreibung (C-Requirements)

Ein Message Logger sammelt und speichert verschiedenste Meldungen (Errors, Warnungen etc.) von verschiedensten Programmen (Clients). Diese Clients müssen den Logger ein- resp. ausschalten können sowie ihn konfigurieren können (welche Nachrichten loggen etc.).

Mit einem einfachen Werkzeug (View-Logger) können die Nachrichten eingesehen werden (Filtern, Suchen soll möglich sein). Die Clients können dieses Window aktivieren resp. deaktivieren.

2.1 Produkt Perspektive

Der *antLogger* ist eine Komponente, welche einfach von Java-Programmen implementiert werden kann. Diese Programme können über eine Schnittstelle (siehe System Schnittstellen) dem Message Logger Nachrichten übermitteln. Weiter sind sie in der Lage ein Fenster (View-Logger) zu öffnen (resp. zu schliessen), mit welchem die Nachrichten eingesehen, sortiert oder durchsucht werden können.

2.1.1 System Schnittstellen

MessageLogger:

Klasse

- um eine Nachricht an den *antLogger* abzusetzen.
- um den Message Logger zu konfigurieren.

  Zum Beispiel: Filtern der Nachrichten; zeigen, resp. verstecken des *Log-Viewer,* setzen des Programmnamen.

2.1.2 Benutzer Schnittstellen

Da ein Benutzer nicht direkt auf den *antLogger* zugreift (immer über ein Programm), werden für die Bedienung keine Schnittstellen benötigt.

Ein GUI braucht der *Log-Viewer* um einen einfachen Zugriff auf die gesammelten Nachrichten zu gewährleisten.

Per Interface kann ein Client ein Log-Viewer (SWING / AWT) aktivieren, welches „parallel“ und völlig unabhängig von der Client-Applikation arbeitet. Der Client hat dazu via Interface das Log-Viewer zu aktivieren. Nach dieser Aktivierung kann er normal weiterarbeiten, und das Log-Viewer bleibt bei Bedarf ebenfalls aktiv.

Ist dieses Log-Viewer aktiv, werden darin die aktuell auftretenden Messages des zugehörigen Clients in Listenform angezeigt und gleichzeitig dauerhaft gespeichert (Siehe MessageWindow von Together). Jeder Client hat sein eigenes Message-Window.

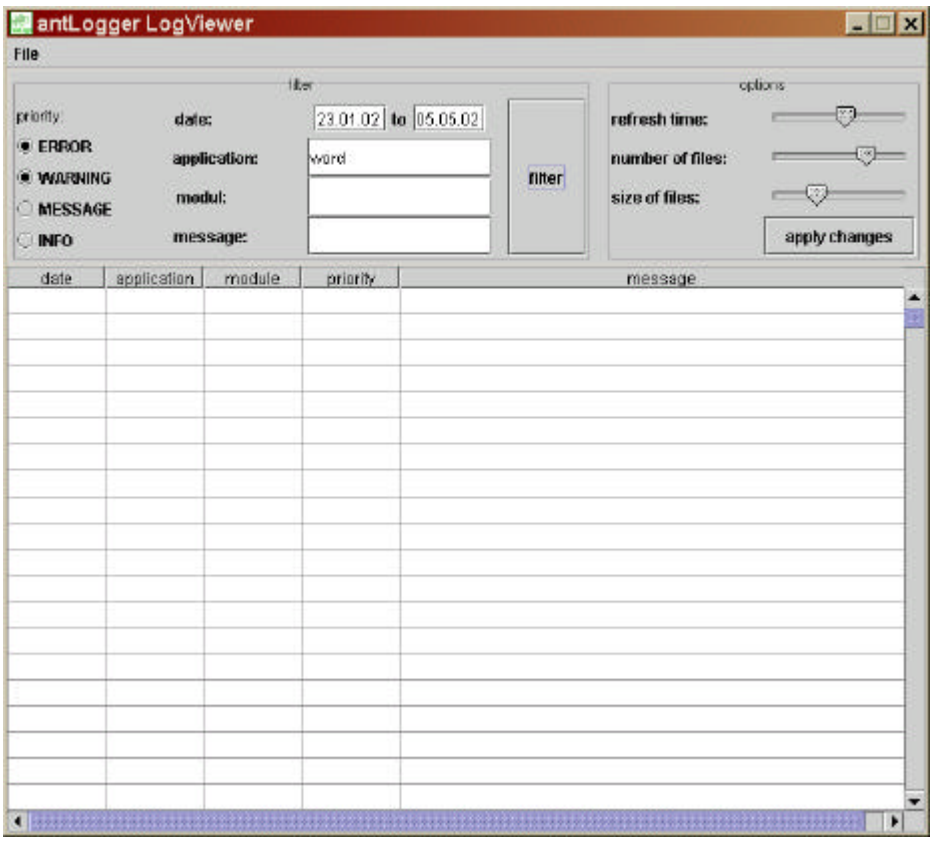
Die angezeigten Messages können unterschiedlich sortiert und gefiltert werden:

- Zeitbereich
- Nach Applikation
- Nach Modul
- Nach Priorität
- Wortsuche in Nachrichten

Es kann bei Wunsch auch auf ältere Meldungen zugegriffen werden (durch Laden alter Sicherungsdateien). Alte Meldungen können auch gelöscht werden (indem alte Sicherungsdateien gelöscht werden). -> Damit kann auf einfachste Art- und Weise eine Wartungs-Applikation für „alte“ auf Disk aufgezeichnete Messages realisiert werden.

Optional: Dem Betrachter kann angezeigt werden, welche Klasse einen Message-Eintrag erzeugt hat.

Der LogViewer wird etwa so aussehen:



2.1.3 Software Schnittstellen

Siehe 2.1.1

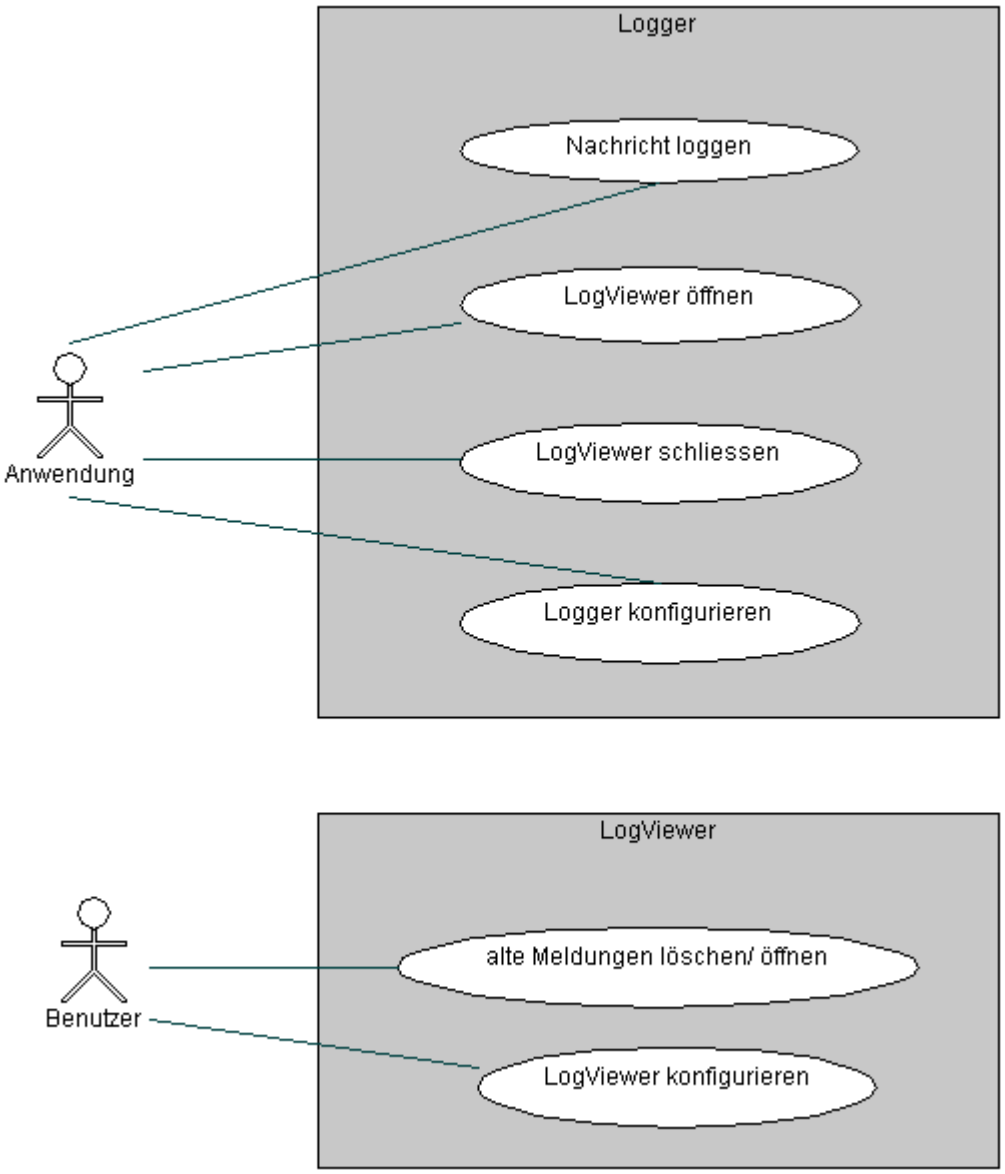
2.1.4 Voraussetzungen Speicher

Dauerhafte Speicherung der Messages auf Disk.

2.2 Produkte Funktionalität

Nr	Bezeichnung
1	Aufgezeichnete Messages müssen einem menschlichen Betrachter in „Klartext“ präsentiert werden können (keine kryptischen Zahlen-Codes). Es muss für den Betrachter zumindest ersichtlich sein, wann diese Message eingetragen wurde. Weitere wünschenswerte Informationen: Wo im Programmablauf wurde die Message abgesetzt? Warum wurde diese abgesetzt?
2	Die Speicherung sehr vieler Messages darf nicht zu einem Disk-Überlauf führen, d.h. die maximale Speichergrösse ist zu beschränken (MByte), „alte“ Messages werden bei Erreichen der maximalen Speichergrenze in eine Datei abgegelegt. Diese Speichergrenze ist konfigurierbar, wie auch die Anzahl der Sicherungsdateien.
3	Es gibt eine zentrale Datenhaltung, d.h. eine unabhängige Applikation hat Zugriff auf „alte“, abgespeicherte Messages anderer Applikationen. Es muss möglich sein, dass mehrere Applikationen „gleichzeitig“ Message-Logger nutzen können (d.h. jede Applikation beinhaltet eine eigene Instanz der Komponente, zentrale Datenspeicherung).
4	Komponenten-Architektur -> Message-Logger ist als Komponente zu realisieren, die einfach in eine Applikation integriert werden kann (d.h. einfaches Interface) und auch einfach austauschbar ist (neue Version). Bemerkung: Ein Austausch der Message-Logger Komponente (neue Version) muss möglichst einfach auszuführen sein, d.h. die entsprechende Applikation darf nur minimal davon betroffen sein. Vor einem Austausch aufgezeichnete Meldungen müssen mit der neuen Komponente nicht mehr sichtbar gemacht werden können, d.h. es ist kein normiertes Aufzeichnungs-Format nötig. Die Austauschbarkeit der Komponenten wird im Projekt geprüft.
5	Bei jedem Message-Eintrag hat die aufrufende Applikation einen **Message-Level** mitzugeben (Bsp: Fehler, wichtige Meldung, normale Meldung, unwichtige Meldung). Per Programm-Interface kann ein Level-Filter gesetzt werden. Damit kann definiert werden, welche Meldungen (mit welchem Level) tatsächlich aufgezeichnet werden. Dieser Level kann zur Laufzeit durch die Applikation jederzeit geändert werden, hat aber nur auf die danach eintreffenden Meldungen Einfluss. Idee: Man kann in einer Applikation zu Debug Zwecken sehr viele Meldungen einfügen, durch den Level-Filter aber erreichen, dass die Meldungen im Normalfall nicht aufgezeichnet werden (wegen Performance).
6	Der **Performance** ist besonderes Augenmerk zu widmen. Insbesondere ist zu beachten, dass Message-Einträge, welche vom Level-Filter nicht durchgelassen werden, die Applikation in ihrem Zeitverhalten nicht negativ beeinflussen.

Übersicht über die wichtigsten Anwendungsfälle (use cases), welche die Software erfüllt. Im Abschnitt 3 werden diese detaillierter beschrieben.



2.2.1 Use Case: Nachricht loggen

Benutzer	Anwendung
Vor-Bedingung	Logger läuft
Nach-Bedingung	
Schritte	1) Anwendung meldet Logger eine Nachricht zu loggen 2) Logger loggt die Nachricht
Variationen	2) Log Datei ist zu gross und einige Nachrichten müssen gelöscht werden

2.2.2 Use Case: LogViewer öffnen

Benutzer	Anwendung
Vor-Bedingung	Logger läuft
Nach-Bedingung	
Schritte	1) Anwendung meldet Logger den LogViewer zu öffnen 2) Logger öffnet den LogViewer
Variationen	2) Der LogViewer ist bereits geöffnet

2.2.3 Use Case: LogViewer schliessen

Benutzer	Anwendung
Vor-Bedingung	Logger läuft
Nach-Bedingung	
Schritte	1) Anwendung meldet Logger den LogViewer zu schliessen 2) Logger schliesst den LogViewer
Variationen	2) Der LogViewer ist bereits geschlossen

2.2.4 Use Case: Logger konfigurieren

Benutzer	Anwendung
Vor-Bedingung	Logger läuft
Nach-Bedingung	
Schritte	1) Anwendung konfiguriert den Logger
Variationen	1) Der Logger antwortet nicht

2.2.5 Use Case: alte Nachrichten löschen / öffnen

Benutzer	Anwender
Vor-Bedingung	Logger läuft
Nach-Bedingung	
Schritte	1) Benutzer löscht / öffnet alte Nachrichten (ganze Dateien)
Variationen	

2.2.6 Use Case: configure LogViewer

Benutzer	Anwender
Vor-Bedingung	Logger läuft
Nach-Bedingung	
Schritte	1) Benutzer konfiguriert den LogViewer
Variationen	

2.3 Benutzer Charakteristiken

Der *antLogger* sollte einfach zu bedienen sein. Einzige Voraussetzung ist die Bedienung eines *Clients*.

2.4 Vorraussetzungen

Keine.

2.5 Verteilung von Anforderungen

Sämtliche Anforderungen werden bereits in der Version 1.0 gefordert. Weitere Versionen sind nicht vorgesehen.

3 Konkrete Anforderungen (D-Requirements)

Auf die D-Requirements wird verzichtet. Das Projekt ist zu klein, um diesen Aufwand zu rechtfertigen. Die Anforderungen sind in den C-Requirements aufgelistet. Die Fragen der Architektur und des Designs werden im SDD geklärt.

4 Unterstützende Informationen

- Software Design Document (SDD) – (in Arbeit)
- Java Doc – (noch nicht erstellt)
- Betriebsanleitung – (noch nicht erstellt)