**Message Logger**

**System-Spezifikation**

[1. Systemübersicht 2](#_Toc362016966)

[2. Architektur und Designentscheide 3](#_Toc362016967)

[2.1. Modell(e) und Sichten 3](#_Toc362016968)

[2.2. Daten (Mengengerüst & Strukturen) 3](#_Toc362016969)

[2.3. Entwurfsentscheide 3](#_Toc362016970)

[3. Schnittstellen 4](#_Toc362016971)

[3.1. Externe Schnittstellen 4](#_Toc362016972)

[3.2. wichtige interne Schnittstellen 4](#_Toc362016973)

[3.3. Benutzerschnittstelle(n) 4](#_Toc362016974)

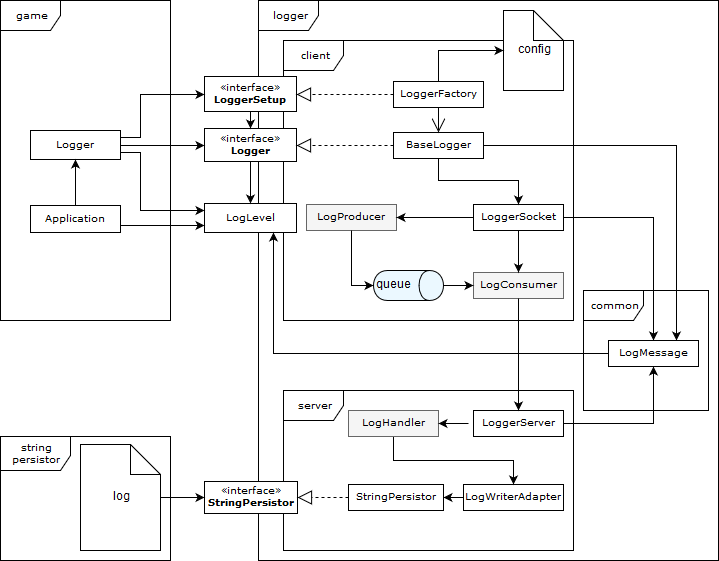
[4. Environment-Anforderungen 5](#_Toc362016975)

Versionen:

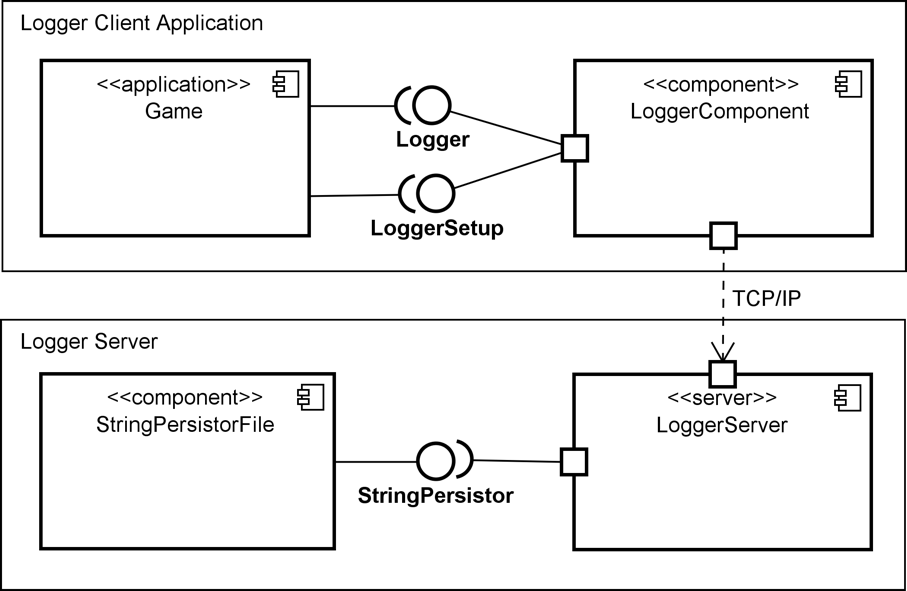
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev. | Datum | Autor | Bemerkungen | Status |
| 0.1 | 02.11.2017 | Valentin Bürgler | 1. Entwurf | in Arbeit |
| 0.1 | 03.11.2017 | Christopher Christensen | 2. Erweiterung der Systemübersicht  3. Architektur und Designentscheide erweitert | in Arbeit |
| 0.2 | 03.11.2017 | Valentin Bürgler | Kapitel 2 überarbeitet und erweitert  Kapitel 4 erzeugt | in Arbeit |
| 0.3 | 04.11.2017 | Valentin Bürgler | Diagramme hinzugefügt  Kapitel 3 überarbietet | in Arbeit |

1. Systemübersicht

Das folgende UML soll eine Übersicht über das implementierte System schaffen.



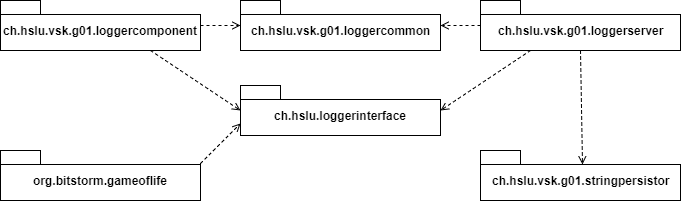
Das UML ist aufbauend auf dem aus dem Input erhaltenem UML, welches nur eine grobe Darstellung der Hauptkomponenten ist.



1. Architektur und Designentscheide

## Modell(e) und Sichten

### Paketdiagramm



## Klassendiagramme

## Entwurfsentscheide

Wir haben generell über das Projekt hinweg versucht uns an den Clean-Code-Prinzipien zu orientieren. Wir versuchten Vererbung zu vermeiden und das «Favour Composition over Inheritance»-Prinzip zu verfolgen. Dazu strebten wir an die Wiederverwendbarkeit zu erhöhen indem wir das DRY-Prinzip vor Augen hielten und die einzelnen Komponenten so zu gestalten, dass sie nur jeweils eine Aufgabe erfüllen (Seperation of Concerns).

### Strategie-Pattern, Singleton-Pattern und Fabrikmethode

Die Verbindung zwischen dem Spiel und der Logger-Komponente wurde nach dem Strategie-Pattern umgesetzt. Der Klient ist das Spiel. Den Kontext bildet die zusätzlich eingefügte Klasse Logger im Spiel-Package. Die Strategie wird von der Instanz-Variabel «instance» im Kontext in Form eines Singletons vom Interface-Typ Logger gehalten. Beim Ausführen der start-Methode wird die Strategie mit einer Fabrikmethode instanziiert.

### Adapter-Pattern

Für die Übertragung der LogMessage vom LogHandler zum StringPersistor, welcher danach die LogMessage in ein File schreibt, verwenden wir das Adapter-Modell. Damit verletzen wir die Wiederverwendbarkeit des StringPersistor nicht und können eine angepasste Implementation für den LogHandler erstellen. Damit erhalten wir die effektiv gewünschte Zielschnittstelle.

1. Schnittstellen

## Externe Schnittstellen

Folgende Schnittstellen wurden vorgeschrieben:

* Logger
* LoggerSetup
* LogLevel
* LogMessage
* StringPersistor

## Wichtige interne Schnittstellen

Folgende Schnittstellen wurden von uns vorgeschrieben:

* LogWriterAdapter
* config.cfg

Die Schnittstellen werden alle im beiliegenden Dokument **DokumentationMessageLogger.md** ausführlich beschrieben.

### *3.x.1. Schnittstelle A*

#### *3.x.1.1. Steckbrief*

*Genauer Name der Schnittstelle, Kurzbeschreibung der Funktionalität, ggf. Autoren und Besitzer (zwischen wem wurde die Schnittstelle ausgehandelt?), ggf. Version*

#### *3.x.1.2. Interaktionen*

*Je nach Schnittstellenart Operationen (z.B. Funktionen, Methoden) oder Datenaustausch (z.B. Nachrichten).*

*Einschränkungen und Voraussetzungen, Berechtigungen, zeitliche Einschränkungen, parallele Benutzung, Voraussetzungen zur Nutzung*

*Generelles zur Fehlerbehandlung, mögliche Fehlersituationen als auch deren Behandlung.*

*Je Interaktion:*

*• Beschreibung der Semantik (Fachlichkeit)  
Diagramm und/oder Beschreibung der fachlichen Abläufe  
Fachliche Bedeutung der Daten  
Nebenwirkungen, Konsequenzen*

*• Beschreibung der Technik  
Methoden/Funktionen, Daten und Datenformate, Gültigkeits- und Plausibilitätsregeln, Codierung, Zeichensätze*

*• Fehlerbehandlung*

#### *3.x.1.3. Einstellungen*

*Kann das Verhalten der Schnittstelle oder der Ressourcen verändert oder konfiguriert werden?*

*Mögliche Konfigurationsparameter*

#### *3.x.1.4. Qualitätsmerkmale*

*Aussagen über Qualitätsmerkmale, an die Implementierer gebunden sind und auf die sich Nutzer verlassen können.*

*Welche Qualitätseigenschaften wie Verfügbarkeit, Performance, Sicherheit gelten für diese Schnittstelle? Neudeutsch heisst dieser Teil der Schnittstellen­beschreibung Quality-of-Service (QoS) Requirements.*

*Mengengerüste Laufzeit Durchsatz / Datenvolumen Verfügbarkeit*

#### *3.x.1.5. Entwurfsentscheidungen*

*Fragestellungen, Einflüsse, Annahmen, Alternativen und Begründungen für Entwurfsentscheidungen im Zusammenhang mit der Schnittstelle, falls angebracht*

*Welche Gründe haben zum Entwurf dieser Schnittstelle geführt? Welche Alternativen gibt es, und warum wurden diese verworfen?*

#### *3.x.1.6. Beispielverwendung*

*Pseudocode oder Quelltext bei Operationen, Beispieldaten bei Datenformaten*

*Hinweise und / oder Beispiele zur Benutzung dieser Schnittstelle*

1. Environment-Anforderungen

* Die Logger-Komponente ist mit Java 1.8.0 realisiert. Es gelten die entsprechenden System-Anforderungen für Java 1.8.0.
* Der Fully-Qualified Class Name der LoggerFactory, die IP Adresse und die Portnummer des Servers müssen in einer Konfigurationsdatei «config.cfg» vorliegen, um eine beliebige Logger-Komponente eines anderen Teams ohne Anpassungen im Code an das Spiel zu koppeln.
* Eine Internetverbindung wird benötigt, um die Nachrichten an den Server zu senden.