
SOFTWARE REQUIREMENTS SPECIFICATION

für Codegenerator GetOptGen

**Programmentwurf C/C++ 2022
TIK/TIM/TIS/TIT21**

Draft 1.0a

Version 1.0a not yet approved

Erstellt durch Th. Staudacher und A. Maus

Inhaltsverzeichnis

1	Einführung	4
1.1	Gültigkeit	4
1.2	Gruppengröße	4
1.3	Abgabe der Prüfungsleistung	4
1.4	Aufgabenstellung	4
2	Allgemeine Beschreibung	6
2.1	Funktionale Anforderungen	6
2.1.1	Genereller Ablauf	6
2.1.2	XML-Format	6
2.1.3	Optionsparameter	8
2.1.4	Optionen	9
2.1.5	Codegenerierung	12
2.1.6	Allgemeine Dateigenerierungs Regeln	12
2.1.7	Implementierung in eine Zielapplikation	13
2.1.8	Demonstration durch ein Beispielprojekt	14
2.2	Nichtfunktionale Anforderungen	16
2.2.1	Tools und Build Umgebungen	16
2.3	Optionale Anforderungen	19
2.3.1	Tools und Build Umgebungen	19
2.4	Programmierung	20
2.5	Programmbeispiele	21
2.5.1	Formatierung unter C++	21
2.5.2	SAX mit Xerces-C	21
2.5.3	XML, utf-16, utf-8 (ASCII)	23
2.5.4	Muster XML	24

Revision History

- Version 0.9: Diese Version ist noch nicht final. Sie ist noch nicht mit den Kursen abgeglichen und einzelne Requirements können sich noch ändern, bzw hinzugefügt werden.
- Version 1.0a: Diese Version ist die Alphaversion der ersten verbindlichen Version. Sie ist noch nicht mit den Kursen abgeglichen und einzelne Requirements können sich noch ändern. Änderungen zu Version 0.9 gab es viele. Größere Änderungen in Kapitel 1.3, 1.4, 2.1.2, 2.1.3, 2.1.4, 2.1.5. Folgende Kapitel sind komplett neu: 2.1.7, Identifikation der Gruppe.

1 Einführung

1.1 Gültigkeit

Diese Version ist Version 1.0a . Sie ist die Alphaversion der ersten verbindlichen Version. Sie ist noch nicht mit den Kursen abgeglichen und einzelne Requirements können sich noch ändern.

1.2 Gruppengröße

Die empfohlene Gruppengröße für die Bearbeitung dieses Projekts sind 4 Teilnehmer. Die maximale Gruppengröße liegt bei 5 Teilnehmern. Der Prüfungsumfang ändert sich nicht durch kleinere Gruppengrößen.

1.3 Abgabe der Prüfungsleistung

Jeder Student und Studentin gibt eigenständig sein Programmentwurf bei Moodle ab, spätestens bis zum vorgesehenen Abgabezeitpunkt der unter Moodle vermerkt ist.

Anmerkung: Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg (DHBW) (Studien- und Prüfungsordnung DHBW Technik – StuPrO DHBW Technik) § 11 Versäumnis, Rücktritt, Täuschung, Ordnungsverstoß / 1

1.4 Aufgabenstellung

Erstellen Sie einen Codegenerator der eine Klasse zum Auswerten von Übergabeparametern erzeugt. Die Optionen sollen hierbei durch eine XML-Datei (XML-Konfigurationsdatei) an das Programm übergeben werden. Des weiteren sind in der XML-Datei auch Angaben zum Autor, Beispiele, Beschreibungen und Ablageorte hinterlegt. Je nach den Vorgaben aus der XML-Datei soll die mit dem Codegenerator erzeugte Klasse entweder direkt instanziiierbar oder abstrakt sein. Die Formatierung der erzeugten Header- und Source Datei ist nicht Umfang des Programmentwurfs, diese kann unter Zuhilfenahme des Kommandozeilenprogramms `astyle` erfolgen. Der Aufruf von `astyle` findet ebenfalls nicht im Programmentwurf statt. Die Main Funktion ist ebenfalls nicht Bestandteil der Generierung. Um das Auswerten der Optionen durchzuführen muss eine Instanz der generierten Klasse in der Applikation erzeugt werden und eine Funktion zum Auslesen der

Optionen aufgerufen werden. Falls sich ein abstraktes Element in der generierten Klasse befindet muss diese dann nochmals in der Zielapplikation abgeleitet werden und die abstrakte Methode implementiert werden.

Nähere Information zur Umsetzung sind in den Requirements der folgenden Kapitel detailliert aufgeführt.

2 Allgemeine Beschreibung

2.1 Funktionale Anforderungen

Funktionale Anforderungen beschreiben was das Programm ausführen soll. Im Gegensatz zu nichtfunktionalen Anforderungen sind sie direkt in der Software zu finden, d.h. es wird kein Tooling beschrieben sondern was im Programmablauf zu beachten ist.

2.1.1 Genereller Ablauf

Dem Codegenerator soll per Kommandozeilenparameter eine XML-Konfigurationsdatei übergeben werden. Der Codegenerator soll diese XML-Konfigurationsdatei einlesen und basierend auf den darin enthaltenen Einstellungen Code-Dateien generieren. Die auf diese Weise generierten Dateien sollen das Grundgerüst für eine neue Applikation bilden.

2.1.2 XML-Format

Das Format der XML-Konfigurationsdatei ist wie folgt vorgegeben (siehe Example.xml). Zur Reduktion der Komplexität wird es sich anbieten, Sonderzeichen im XML-Text bereits fertig für C/C++ zu „escapen“.

GetOptSetup

Das Attribut SignPerLine stellt die Anzahl der Buchstaben pro Zeile ein. Typisch für die Kommandozeile ist der Defaultwert 79.

Author

Die Attribute Name, Phone und Mail beinhalten Informationen über den Autor, die beim Hilfetext mit ausgegeben werden sollen.

HeaderFileName

HeaderFileName hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Dateiname der anzulegenden Header-Datei.

SourceFileName

SourceFileName hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Dateiname der anzulegenden Quellcode-Datei.

NameSpace

NameSpace hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Name des anzulegenden Namespace in der sich die generierte Klasse befindet.

ClassName

ClassName hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Name der anzulegenden Klasse.

OverAllDescription

OverAllDescription hat keine Attribute. Zwischen Start- und End-Tag stehen 1 oder mehrere Block-Tags. Block hat keine Attribute. Zwischen Start- und End-Tag steht ein Textblock, der das Programm beschreibt.

SampleUsage

SampleUsage hat keine Attribute. Zwischen Start- und End-Tag stehen 1 oder mehrere Sample-Tags. Sample hat keine Attribute. Zwischen Start- und End-Tag steht ein möglicher Programmaufruf.

Options

Options hat keine Attribute. Zwischen Start- und End-Tag stehen 1 oder mehrere Option-Tags. Option hat die Attribute Ref (Optional), ShortOpt, LongOpt (minimum 1 von beiden muss belegt sein), Exclusion, Interface, ConvertTo, DefaultValue, ConnectToInternalMethod, ConnectToExternalMethod, HasArguments und Description.

- Ref ist ein Integerwert und dient dazu die einzelnen Attribute referenzieren zu können, zum Beispiel wenn sie sich gegenseitig ausschließen sollen.
- ShortOpt definiert die Parameter die nur aus einem einzelnen Buchstaben bestehen können.
- LongOpt definiert die Parameter die aus ganzen Worten bestehen können.
- Interface ist die Schnittstelle wo nach Finden der Option der Wert hinterlegt wird
- Exclusion listet die Referenzen der Optionen mit der diese Option nicht gemeinsam aufgerufen werden darf
- ConvertTo gibt eine Umwandlung vor da ein Command-Line Parameter initial immer Text ist.
- DefaultValue gibt bei Optionalen Parametern den Default-Wert vor, der angenommen werden soll, wenn der Parameter nicht gesetzt wird.

- ConnectToInternalMethod definiert die Methode die bei der Generierung dieser Klasse angelegt werden muss und beim Auswerten dieses Optionsparameters aufgerufen wird (z.B. den Hilfetext).
- ConnectToExternalMethod definiert die Methode die beim Auswerten dieses Optionsparameters aufgerufen wird. Diese ist abstrakt und die Klasse muss einmal abgeleitet werden.
- HasArguments kann die Werte optional und required annehmen. Wenn die Option keine Argumente nutzt kann HasArguments weggelassen werden.
- Description ist der eigentliche Hilfetext der zu der Option angezeigt werden soll.

2.1.3 Optionsparameter

Darstellung im Hilfe Menü

Um kein aufwendiges sortieren von Hand in der XML Konfiguration durchführen zu müssen soll der Parser die Optionen nach Alphabet sortieren. Prior sind die Buchstaben der Kurzoptionen, dann deren der Langoptionen (ohne führende -/- zu beachten).

{ReqFunc1} Nach Parsen aller Optionen sollen die Optionen nach Alphabet aufsteigend sortiert werden [A-Z]

Es soll eine Hilfetextausgabe realisiert werden. Diese soll als Methode in der Klasse hinterlegt werden. Der Hilfetext soll aus den Optionen generiert werden. Dabei ist die Kurz-, die Langoption, die Beschreibung sowie die Angabe zum Autor und Benutzung unterzubringen. Die Textbreite soll durch die XML-Einstellungen vorgegeben werden.

{ReqFunc2} Es ist eine virtuelle protected Methode printHelp zu erstellen die den Hilfetext auf der Konsole ausgibt.

{ReqFunc3} Der Hilfetext soll die Inhalte Blockformatieren.

{ReqFunc4} Der Hilfetext ist so einzufassen, dass er die Angabe aus dem Attribute SignPerLine nicht überschreitet (Defaultwert 79 Zeichen/Linie).

{ReqFunc5} Der Hilfetext soll am Ende eine Zeile mit den Angaben aus dem XML Tag Author enthalten.

{ReqFunc6} Der Hilfetext soll die Beschreibungen aus dem XML Tag OverAllDescription enthalten.

{ReqFunc7} Der Hilfetext soll die Beispielbenutzung aus dem XML Tag SampleUsage enthalten.

2.1.4 Optionen

Generelle nicht konfigurierbare Funktionalität

Die Optionen die nicht geparkt werden können, sollen mit einer Fehlermeldung ausgegeben werden und dann das Programm beendet werden.

- {ReqFunc8} Ist eine Option nicht vorhanden, dann soll das Programm eine Fehlermeldung mit Angabe welche Kombination in diesem Fall nicht erlaubt war auf stderr ausgegeben werden
- {ReqFunc9} Ist eine Option fehlerhaft ausgeführt, dann soll das Programm eine Fehlermeldung mit Angabe welche Kombination in diesem Fall nicht erlaubt war auf stderr ausgegeben werden
- {ReqFunc10} Wenn eine Option nicht vorhanden oder fehlerhaft ausgeführt wird, dann soll das Programm abgebrochen werden

Referenzen und Exclusion

Da nicht alle Optionsparameter sinnvoll gleichzeitig verwendet werden können, soll es Einschränkungen geben. Diese Einschränkungen sollen durch Referenzen in den Einschränkungen eingetragen werden können. Tritt nun so eine Kombination auf, dann soll das Parsen der Optionen im generierten Programm beendet werden und auf dem Bildschirm eine Fehlermeldung ausgegeben werden.

- {ReqFunc11} Referenzen dürfen Integer Werte von 1 bis 63 aufnehmen
- {ReqFunc12} Referenz 0 wird nicht in den Ausschluss mit ein bezogen
- {ReqFunc13} Referenzen werden in Exclusion komma-separiert eingetragen
- {ReqFunc14} Wenn eine Option in Kombination mit einer Exclusion vorkommt, dann soll das Programm eine Fehlermeldung mit Angabe welche Kombination in diesem Fall nicht erlaubt war auf der stderr ausgegeben
- {ReqFunc15} Wenn eine Option in Kombination mit einer Exclusion vorkommt, dann soll das Programm abgebrochen werden

Interfaces

In der generierten Klasse werden, für die Optionen wo ein Interface in der Konfiguration angegeben wurde, entsprechende Getter Methoden in der generierten Klasse angelegt. Wird diese Option später dann gefunden so ist einzutragen dass sie aufgerufen wurde, ggf. sind auch Werte nach aussen zu übertragen.

- {ReqFunc16} Ist die Option Interface angewählt, dann soll eine Getter Funktion in der generierten Datei angelegt werden damit ausgelesen werden kann ob die Option vorgekommen ist. Die Schnittstellensyntax ist `bool isSet+Optionsname` (bevorzugt `LongOptName`, ggf. `ShortOptname`)

- {ReqFunc17} Ist die Option Interface angewählt und es gibt einen optionalen oder required Übergabeparameter, dann soll eine Getter Funktion zum Auslesen der Übergabe angelegt werden. Die Schnittstellensyntax ist T getValueOf+Optionsname (bevorzugt LongOptName, ggf. ShortOptname, T kann std:string, bool oder int sein)
- {ReqFunc18} Ist die Option Interface angewählt und es gibt einen optionalen Übergabeparameter, dann soll eine Getter Funktion den Defaultwert übernehmen falls kein Übergabewert bei dem Aufruf der Option übergeben wurde
- {ReqFunc19} Ist die Option Interface angewählt und es gibt einen optionalen oder required Übergabeparameter, dann soll eine Getter Funktion den Wert in das Zielformat wandeln. Vorgesehen sind hier Grundformat string, Wandlungsformat bool und integer
- {ReqFunc20} Ist keine der drei Option Interface, ConnectToInternalMethod, ConnectToExternalMethod ausgewählt, dann soll eine Getter Funktion in der generierten Datei angelegt werden damit ausgelesen werden kann ob die Option vorgekommen ist. Die Schnittstellensyntax ist bool isSet+Optionsname (bevorzugt LongOptName, ggf. ShortOptname)

ConnectToInternalMethod zum Aufruf klasseninterner Methoden

Wird ConnectToInternalMethod mit einem Namen belegt so soll die interne Methode aufgerufen werden. Dieses kann z.B. die vordefinierte Hilfsfunktion helpText sein.

- {ReqFunc21} Ist ConnectToInternalMethod für eine Option ausgewählt, dann soll diese Methode aufgerufen werden sobald die Option erkannt wird
- {ReqFunc22} Ist ConnectToInternalMethod und ein required Übergabeparameter für eine Option ausgewählt, dann soll diese Methode mit dem passenden Übergabeparameter aufgerufen werden sobald die Option erkannt wird
- {ReqFunc23} Ist ConnectToInternalMethod für eine Option ausgewählt und es gibt einen optionalen Übergabeparameter, dann soll die Methode so generiert werden, dass der Defaultwert gesetzt wird falls kein Übergabewert bei dem Aufruf der Option übergeben wurde
- {ReqFunc24} Ist ConnectToInternalMethod für eine Option ausgewählt und es gibt einen optionalen oder required Übergabeparameter und eine Formatwandlung ConvertTo ist mit konfiguriert, dann soll die Übergabe im Zielformat stattfinden. Vorgesehen sind hier Grundformat string, Wandlungsformat bool und integer

ConnectToExternalMethod zum Aufruf klassenexterner Methoden

Wird ConnectToExternalMethod mit einem Namen belegt so soll die Methode der Ableitung dieser Klasse aufgerufen werden. Dazu ist eine abstrakter Methodenrumpf zu generieren.

- {ReqFunc25} Ist ConnectToExternalMethod für eine Option ausgewählt, dann soll ein abstrakter Methodenrumpf unter Verwendung des Namens in ConnectToExternalMethod erstellt werden
- {ReqFunc26} Ist ConnectToExternalMethod für eine Option ausgewählt, dann soll diese aufgerufen werden sobald die Option erkannt wird
- {ReqFunc27} Ist ConnectToExternalMethod und ConvertTo für eine Option ausgewählt, dann soll diese mit dem passenden Übergabeparameter aufgerufen werden sobald die Option erkannt wird
- {ReqFunc28} Ist ConnectToExternalMethod für eine Option ausgewählt und es gibt einen optionalen Übergabeparameter, dann soll der Defaultwert übernehmen falls kein Übergabewert bei dem Aufruf der Option übergeben wurde
- {ReqFunc29} Ist ConnectToExternalMethod für eine Option ausgewählt und es gibt einen optionalen oder required Übergabeparameter und eine Formatwandlung ConvertTo ist mitkonfiguriert, dann soll die Übergabe im Zielformat stattfinden. Vorgesehen sind hier Grundformat string, Wandlungsformat bool und integer

HasArguments zur Steuerung für Übergabeparameter

- {ReqFunc30} Ist in HasArguments Required für den Übergabeparameter eingetragen und der Nutzer übergibt keinen Übergabeparameter, dann soll dann unter Ausgabe einer Fehlermeldung auf stderr abgebrochen werden

ConvertTo zur Steuerung der Datenwandlung für Übergabeparameter

Soll ein Übergabeparameter in ein anderes Format gewandelt werden so ist die Möglichkeit gleich das Zielformat mit anzugeben. Default werden alle Texte in std::string utf-8 Übertragen. Weitere Möglichkeiten sind bool und int.

- {ReqFunc31} Ist in ConvertTo Integer für den Übergabeparameter eingetragen, dann soll der Übergabeparameter zur Weiterverarbeitung in das Zielformat int überführt werden.
- {ReqFunc32} Ist in ConvertTo Bool für den Übergabeparameter eingetragen, dann soll der Übergabeparameter zur Weiterverarbeitung in das Zielformat bool überführt werden. Erlaubt sind 0, 1, true, false in der Übergabe
- {ReqFunc33} Ist die Wandlung mit ConvertTo auf Grund der Wertart und Bereiche die da übergeben werden nicht möglich, dann soll unter Ausgabe einer Fehlermeldung auf stderr abgebrochen werden

2.1.5 Codegenerierung

Dateinamen

Die Dateinamen der Header- und der Source Datei sollen aus der XML-Konfigurationsdatei entnommen werden. Es gibt dafür keine Vorgaben. Sind diese Tags leer so kann die jeweilige Datei nicht generiert werden. Auf eine komplette Pfadextraktion wird verzichtet da unter Windows die Pfadauflösung anders geregelt ist wie unter Unix. Es handelt sich hierbei nur um reine Dateinamen. Der Namen der Header- und Source-Datei ist separat aufgeführt, da falls dieses Applikation einmal produktiv eingesetzt wird sich die Endungen der Header- und Source Datei unterscheiden wird.

UNIX-V .h/.cc

Visual Studio .hpp/.cpp

Linux .h/.cpp

macOS .h/.cpp

{ReqFunc34} Ist in der XML-Datei der XML Tag HeaderFileName befüllt, dann soll eine Header-Datei mit diesem Namen erstellt werden.

{ReqFunc35} Ist in der XML-Datei der XML Tag SourceFileName befüllt, dann soll eine Source-Datei mit diesem Namen erstellt werden.

Header Datei

Um die Header Datei mehrfach includieren zu können ist eine Mehrfacheinbindung vorzusehen. Diese soll in die erzeugte Datei geschrieben werden dass sie sowohl unter Windows MinGW wie auch unter UNIX GNU-C Compiler funktioniert. Das Kommando `#pragma once` ist nicht erwünscht.

{ReqFunc36} Die erzeugte Header Datei soll Mehrfacheinbindung unterstützen.

{ReqFunc37} Die Mehrfacheinbindung der erzeugten Header Datei soll kompatibel zum Compiler Windows MinGW wie auch unter UNIX zu GNU-C sein.

2.1.6 Allgemeine Dateigenerierungs Regeln

Namespaces

Ist in der XML-Datei ein Namespace benannt, dann soll dieser mit in die Erzeugung der Zieldateien geschrieben werden.

{ReqFunc38} Ist in der XML-Datei der XML Tag NameSpace befüllt, dann soll in der Header-Datei dieser Namespace eingefügt werden.

{ReqFunc39} Ist in der XML-Datei der XML Tag NameSpace befüllt, dann soll in der Source-Datei dieser Namespace eingefügt werden.

2.1.7 Implementierung in eine Zielapplikation

Die Main-Funktion ist nicht Bestandteil der Generierung. Um das Auswerten der Optionen durchführen zu können muss eine Instanz der generierten Klasse in der Applikation erzeugt werden und eine Funktion zum Auslesen der Optionen aufgerufen werden.

Aufruf der Auswertung der Optionen

{ReqFunc40} In der Generierung soll eine Methode implementiert werden mit der die Optionen geparkt werden. Die Parameter dieser Methode sind **void parseOptions(int argc, char **argv)**

Anwendungsbeispiel

Interface Bei einem Interface (Attribut Interface) liegen die Daten im Objekt und können durch Getter Funktionen abgerufen werden.

```
//Fusst auf der im Anhang beigefügten Muster XML-Konfiguration.
2 #include <options.h>
  //...
4
  //Aufruf programm --astyle--path=/usr/bin
6 int main(int argc, char argv) {
8     GenGetCreator::COptionParser options;
10    options.parseOptions(argc, argv);
12    if(option.isSetAstylePath()) {
14        //std::string ... = option.getValueOfAstylePath();
        //...
16 }
```

Interne Methode Bei einer internen Methode (Attribut ConnectToInternalMethod) wird die im internen Objekt befindliche Methode aufgerufen.

```
//Fusst auf der im Anhang beigefügten Muster XML-Konfiguration.
2 #include <options.h>
  //...
4
  //Aufruf programm -h
6 int main(int argc, char **argv) {
8     GenGetCreator::COptionParser options;
10    options.parseOptions(argc, argv);
12    //Durch Aufruf von Help wird der Hilfetext ausgegeben
```

```

14 } //und das Programm beendet (geschieht bereits in Parse)

```

Externe Methode Bei einer externen Methode (Attribut `ConnectToExternalMethod`) wird die im internen Objekt befindliche Methode aufgerufen.

```

2 //Fusst auf der im Anhang beigefügten Muster XML-Konfiguration.
3 #include <options.h>
4 //...
5
6 class MyOptionParser : public GenGetCreator::COptionParser {
7 public:
8     //Anlegen von Ctor und Destruktor
9 protected:
10     virtual void Version() {
11         //Befüllen mit dem Versionstext
12         exit(EXIT.SUCCESS);
13     }
14 }
15
16 //Aufruf programm -v
17 int main(int argc, char **argv) {
18     MyOptionParser options;
19
20     options.parseOptions(argc, argv);
21
22     //Durch Aufruf von Version wird der Versionstext ausgegeben
23     //und das Programm beendet (geschieht bereits in Parse)
24 }

```

2.1.8 Demonstration durch ein Beispielprojekt

Inhalte

Es soll eine Beispielapplikation erstellt werden zur der eine XML-Konfiguration erstellt wird. In der XML-Konfiguration soll die gesamte Funktionalität vorgeführt werden. Was das Programm macht ist zweitrangig, die Testimplementierung der erzeugten Klasse steht im Vordergrund.

- {ReqFunc41} In der Musterimplementierung soll eine Autorengabe der Gruppenteilnehmer enthalten sein.
- {ReqFunc42} In der Autorengabe der Gruppenteilnehmer muss keine Telefonnummer enthalten sein.

- {ReqFunc43} In der Autorenangabe der Gruppenteilnehmer muss mindestens eine Kontakt-Mail Adresse enthalten sein um Rückfrage seitens Dozent stellen zu können.
- {ReqFunc44} In der Musterimplementierung soll ein Namespace enthalten sein.
- {ReqFunc45} In der Musterimplementierung soll im Hilfetext die Beschreibung des Programms (OverAllDescription) enthalten sein.
- {ReqFunc46} In der Musterimplementierung soll im Hilfetext die Musterbenutzung (Sample Usage) enthalten sein.
- {ReqFunc47} Die erstellte Klasse soll eine beliebige Zahl von Argumenten parsen können die nur von getopt(...) und getopt_long(...) limitiert sind.
- {ReqFunc48} Die erzeugte Parser Klasse soll Character vom Type ASCII verarbeiten können.
- {ReqFunc49} Die generierte Klasse soll Ihren Speicher selbständig verwalten. Nach Entfernen der Instanz soll der ganze Speicher wieder freigegeben werden.
- {ReqFunc50} Das Generierungsprogramm soll den Speicher selbständig verwalten und auch wieder freigeben.
- {ReqFunc51} In der Musterimplementierung soll die Option Hilfe mit dem ShortOpt -h und LongOpt -help enthalten und zur internen Methode printHelp (ConnectToInternalMethod) verbunden werden.
- {ReqFunc52} In der Musterimplementierung soll die Option Version mit dem ShortOpt -v und LongOpt -version enthalten (Inhalte der Version bitte selbst bestimmen) und zu einer externen Routine verbunden werden (Interface).
- {ReqFunc53} In der Musterimplementierung soll mindestens ein Ausschluss (Ref/Exclusion) enthalten sein.
- {ReqFunc54} In der Musterimplementierung soll mindestens eine Option mit einem required Parameter mit Typwandlung bool enthalten sein.
- {ReqFunc55} In der Musterimplementierung soll mindestens eine Option mit einem optional Parameter mit Typwandlung Integer enthalten sein, wobei der Integerwert aus DefaultValue ausgelesen wird.

Erstellungsprozess der Musterimplementierung

Es soll eine CMake Datei erstellt werden die die Musterimplementierung sowohl mit MinGw (Windows) als auch mit GNU-C/C++ (UNIX) erstellen kann.

- {ReqFunc56} Es soll eine CMake Datei erstellt werden die die Musterimplementierung sowohl unter MinGw (Windows) und GNU-C/C++ (UNIX) erstellen kann.

Dokumentation der Musterimplementierung

Die Musterimplementierung soll dokumentiert werden. Zusammen mit der erzeugten Header- und Source-Datei können die Dozenten die Funktion der einzelnen Teilfunktionen aus der Dokumentation verstehen.

- {ReqFunc57} Die Dokumentation im Beispielpunkt soll im doxygenFormat sein.
- {ReqFunc58} Die Kommentare der Methoden sollen doxygen-konform sein.
- {ReqFunc59} Eine erzeugte Doxygen-Dokumentation soll der Abgabe beiliegen.

2.2 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen sind umzusetzen. Im Gegensatz zu funktionalen Anforderungen, sind nichtfunktionale Anforderungen nicht direkt an Features des Programms gebunden sondern beschreiben die Umgebung und das Tooling.

2.2.1 Tools und Build Umgebungen

Unterstützte Plattformen und Compiler

Um die Software auf den Systemen Windows 7/10/11 und Linux Ubuntu lauffähig zu bekommen ist zwingend erforderlich einen Compiler zu verwenden, der die gleichen Kommandozeilenparameter versteht und frei von Lizenzen ist. Das ist aktuell beim MSVC2017 Compiler nicht gegeben.

- {ReqNonFunc1} Die Software soll auf **MinGW64** für Windows compilierbar sein
- {ReqNonFunc2} Die Software soll mit **GNU-C/C++** für UNIX compilierbar sein

Erstellung der ausführbaren Datei

Um die Software frei von graphischen Entwicklungsumgebungen erstellen zu können wird ein Makefile benötigt. Dieses kann dann unter mingw32-make (Windows) oder make (UNIX) erstellt werden. Da der Umfang dieses von Hand zu erstellen den Umfang des Programmentwurfs überschreiten würde, wird hier auf CMake als Erstellungstool für das Makefile verwiesen.

- {ReqNonFunc3} Das Erstellen der Software soll mit **CMake** erfolgen

Einlesen von XML-Dateien

Eine XML-Datei kann mit einer schier unüberschaubaren Anzahl von Libraries in C/C++ eingelesen werden. Die bekannteste hier ist XERCES von Apache. Apache ist ein kostenloses Framework zum hosten von XHTML Seiten. Die Library beinhaltet die Varianten SAX/SAX2 und DOM. SAX durchläuft das Dokument linear und DOM baut eine Baumstruktur auf. Beides ist geeignet zum auslesen einer XML-Datei. Mit DOM kann auch

eine XML-Datei erstellt werden.

Hinweis: Ob die XML-Datei als SAX oder DOM ausgelesen wird bleibt der Gruppe die die Implementierung erstellt überlassen.

{ReqNonFunc4} Zum einlesen der XML Datei muss die Softwarelibrary **XERCES-C** verwendet werden

Dokumentation des Programmentwurf

Um den Dozenten Th. Staudacher und A. Maus die Chance zu geben, die Struktur und Funktionalität des Programmentwurf zu Überblicken ist es dringend gegeben, dass die Dokumentation des Codes im doxygen-Format erfolgt. Durch gute Dokumentation werden Rückfragen im Fehlerfall minimiert und es können ggf. auch besser Teilpunkte vergeben werden.

{ReqNonFunc5} Die Dokumentation des Codes muss mit dem Syntax von **doxygen** erfolgen

Einlesen der Optionen

Um eine einheitliche Basis zu schaffen und das Programm portabel zu halten wird zum Parsen der Optionen die getopt(_long) Implementierung verwendet. Das implementieren anderer Libraries oder eigengeschriebene Anwendungen ist nicht gestattet.

{ReqNonFunc6} Das Parsen der Optionen soll über die LibC Funktionalität getopt(...) und getopt_long(...) aus getopt.h erfolgen

Dateien und Verzeichnisstruktur

Um eine saubere Trennung zwischen Implementierung und Deklaration zu bekommen ist eine Auftrennung in zwei Verzeichnisse nötig. Unterverzeichnisse in dieser Auftrennung sind ebenfalls möglich.

{ReqNonFunc7} Header- und Source Dateien sollen sich in verschiedenen Unterverzeichnissen befinden

{ReqNonFunc8} Header Dateien sollen relativ und nicht absolut inkludiert werden

Identifikation der Gruppe

Um die Bewertung der Gruppe durch zu führen ist die angefügte Datei students.xml mit den Teilnehmer der Gruppe aus zu füllen. Die Angabe einer Mailadresse ist optional (das Attribut kann auch weggelassen werden). Hilfreich ist aber wenn die Dozenten mit der Gruppe für Nachfragen in Kontakt treten können.

{ReqNonFunc9} Die beigestellte Datei students.xml soll von der Gruppe vollständig befüllt sein und in jeder der Abgabe enthalten sein (Format utf-8)

{ReqNonFunc10} Die beigestellte Datei students.xml soll im Hauptverzeichnis der Abgabe liegen

{ReqNonFunc11} Die beigestellte Datei students.xml soll die abgebende Person kennzeichnen (Attribute deliver ist auf true zu setzen bei der abgebenden Person). Für die anderen Studenten bleibt dieses Attribut nicht belegt (oder mit false belegen)

Beispiel einer students.xml richtig befüllt Diese students.xml ist mit einer Gruppengröße 4 Studenten befüllt. Diese Abgabe ist Herr Max Mustermann zu zu ordnen.

```
2 <?xml version="1.0" encoding="UTF-8" ?>
  <students>
    <student deliver="true" surname="Max" name="Mustermann" mail="optional"
      course="TIT21" />
4   <student surname="Erika" name="Mustermann" mail="optional" course="TIK21"
    />
    <student surname="John" name="Doe" mail="optional" course="TIM21" />
6   <student surname="Jane" name="Doe" mail="optional" course="TIS21" />
  </students>
```

Programmierung

In dieser Sektion sind die Codierrichtlinien enthalten die dafür sorgen dass der Code der geforderten Qualität entspricht. Diese Codierrichtlinien sind unbedingt einzuhalten. Muss aus irgendeinem Grund davon abgewichen werden so ist dies in einem Kommentar zu begründen.

- {ReqNonFunc12} Daten und Methoden die nur einer internen Berechnung dienen (nicht zum Export) sollen gegen externen Aufruf oder auslesen geschützt werden
- {ReqNonFunc13} Zeiger sollen immer initialisiert werden, Ausnahmen nur gestattet bei entsprechender Kommentierung
- {ReqNonFunc14} Der Gültigkeitsbereich einer Variablen soll beachtet werden
- {ReqNonFunc15} Werden logische Verknüpfungen gemacht so sind logische Operatoren zu verwenden
- {ReqNonFunc16} Berechnungen in logischen Verknüpfungen sind nicht gestattet
- {ReqNonFunc17} Werte die im weiteren Kontext nicht mehr verändert werden sollen, sind konstant zu halten
- {ReqNonFunc18} Zeiger die im weiteren Kontext nicht mehr verändert werden sollen, sind konstant zu halten
- {ReqNonFunc19} Das Wegcasten von const ist nicht gestattet
- {ReqNonFunc20} Die Gültigkeit von Variablen soll an Übergabestellen überprüft werden
- {ReqNonFunc21} Jede Header- und Source Datei soll am Anfang ein Hinweis auf den Bearbeiter enthalten

- {ReqNonFunc22} Das Kommentieren des Codes soll in deutscher Sprache erfolgen, optional in Englisch
- {ReqNonFunc23} Werden Variablen verwendet, die eine fixe Bitlänge benötigen, so soll die Definition aus `cstdint` verwendet werden
- {ReqNonFunc24} Berechnungen mit Fließkommazahlen sind nicht gestattet
- {ReqNonFunc25} Das Verwenden von `goto` ist nicht gestattet
- {ReqNonFunc26} Das Verwenden von `#pragma` ist nur gestattet, solange dadurch keine Compilerabhängigkeit entsteht

2.3 Optionale Anforderungen

Optimale Requirements fließen nicht in die Bewertung ein. Es hat sich aber in den letzten Jahren gezeigt, dass eine saubere Entwicklungsumgebung viele Fehler vermeidet. Es ist angeraten, einige der optionalen Requirements mit um zu setzen.

2.3.1 Tools und Build Umgebungen

Libraries für C++

Es kann der ganze Umfang der mitgelieferten Headerdateien (soweit unter Windows und UNIX gleich einsetzbar) verwendet werden. Alternativ muss per Präprozessor jeweils eine Umschaltung erfolgen. Vordefinierte Macros sind hierbei `__WIN32__`, `__UNIX__` etc. Dies sind dann ggf. in der Dokumentation des Compilers nachzuschlagen. Um die volle Funktionalität einer C++ Implementierung auspielen zu können, kann sowohl die STL wie auch Boost als Library verwendet werden.

Anmerkung: Die STL ist Bestandteil des g++ Compilers, Boost hingegen nicht. Diese kann leicht unter UNIX installiert werden (boost-dev). Bei MinGW64 wird geraten, gleich ein fertiges Paket aus **MinGW64+Boost** zu installieren.

- {ReqOptFunc1} Die Library **STL** kann vollumfänglich verwendet werden
- {ReqOptFunc2} Die Library **Boost** kann vollumfänglich verwendet werden

Logging von Ereignissen

Um den Bildschirm nicht mit Meldungen zu überfrachten, hat es sich eingebürgert, ein Logging zu verwenden. Das ursprünglich für Java entwickelte Log4J wurde in fast alle Programmiersprachen überführt. Im Unterricht wurden zumindestens eines der beiden Logging-Libraries Easylogging++ oder Boost Logging besprochen. Es ist dringend angeraten, dieses auch im Programmentwurf ein zu setzen. Es kann auch helfen, im Fehlerfall im Dialog mit den Studenten eine lauffähige Version nach Abgabe zu erzeugen, falls diese fehlschlägt.

{ReqOptFunc3} Das Logging der Zustände während des Programmablauf kann mit der Library **Easylogging++** erfolgen

{ReqOptFunc4} Das Logging der Zustände während des Programmablauf kann mit der Library **Boost Logging** erfolgen

Formatieren der Ausgabe

Die Formatierung der Ausgabe ist kein Bestandteil des Programmentwurfs. Daher sollte keine Zeit in die Formatierung des Ergebnisses verwendet werden. Um aber eine bessere Übersicht im Arbeitsergebnis zu erhalten wäre eine Formatierung der Header- und Source Datei vorteilhaft. Hierbei bietet sich **astyle** an.

{ReqOptFunc5} Das Formatieren des erzeugten Codes kann mit **Artistic Style** erfolgen

Testen von Teilfunktionalitäten

Um die einzelnen Programmteile auf Funktionalität zu untersuchen ist es angeraten einen Unittest für die einzelnen Programmteile zu erstellen. Es hat sich aber in der Vergangenheit gezeigt, dass sowohl die Studenten wie auch die Dozenten aus den Unittest nachvollziehen konnten wie das Programm funktioniert und häufig auch Fehler gefunden wurden. Zwei Libraries sind hier von Bedeutung, einmal die einfach zu implementierende **Catch2** die nur als reine Header Datei inkludiert werden kann sowie das etwas aufwendigere Framework von **Boost Test**. Der Vorteil von **Catch2** ist die leichtere Implementierbarkeit, der Vorteil von **Boost Test** der bessere Funktionsumfang und die bessere Integrierbarkeit von C++.

{ReqOptFunc6} Das Testen der Module kann mit der Library **Catch2** erfolgen

{ReqOptFunc7} Das Testen der Module kann mit der Library **Boost Test** erfolgen

2.4 Programmierung

{ReqOptFunc8} Das verwenden des ? Operators ist gestattet

{ReqOptFunc9} Bei einer Kombination aus if/else if das kein else hat kann das else mit einem leeren Codeblock mit entsprechender Kommentierung hinzugefügt werden

{ReqOptFunc10} Das verwenden von mehrdimensionalen Arrays soll vermieden werden

{ReqOptFunc11} Soweit technisch möglich sollten die Header- und Source Dateien in utf-8 Zeichencodierung erstellt werden

2.5 Programmbeispiele

2.5.1 Formatierung unter C++

Sehr präzise kann Text unter C formatiert werden. Im Prinzip gibt es diese Optionen auch unter C++. Ein einfaches Mittel die C Formatierung von printf auch unter C++ zu übernehmen ist die Template Klasse von Boost Format. Der Syntax ist identisch zu printf und es gibt auch Erweiterungen.

```
//https://www.boost.org/doc/libs/1_75_0/libs/format/doc/format.html
2 #include <iostream>
   using namespace std;
4
   #include <boost/format.hpp>
6   using namespace boost;

8   int main() {
       cout << format("%-20s %s") % "Hello" % "World" << endl;
10   return 0;
   }
```

2.5.2 SAX mit Xerces-C

Die Xerces-C Library beinhaltet alles was ein guter XML Parser kann, SAX, Dome, X-Path etc. Hier ein Beispiel mit SAX das ein gültiges XML Dokument parsen kann.

```
//g++ -o sax.exe sax.cpp -IC:\~PathToLib~\xerces-c\include -LC:\~PathToLib
~\xerces-c\lib -lxerces-c.dll -std=c++20
2 //Aufruf: sax datei.xml
   #include <xercesc/util/XMLString.hpp>
4   #include <xercesc/parsers/SAXParser.hpp>
   #include <xercesc/sax/HandlerBase.hpp>
6   #include <xercesc/util/OutOfMemoryException.hpp>
   XERCES_CPP_NAMESPACE_USE

8
   #include <locale>
10  #include <codecvt>
   #include <string>
12  #include <iostream>
   using namespace std;
14
   class SimpleSAXParser : public HandlerBase {
16   public:
       virtual void startDocument() {
18       cout << "Dokument beginnt!" << endl;
       }

20       virtual void endDocument() {
22       cout << "Dokument ist zu Ende!" << endl;
       }
```

```

24     virtual void startElement(const XMLCh* const name, AttributeList&
attributes) {
26         cout << "Start-Element: " << converter.to_bytes(name) << endl;

28         for (XMLSize_t i = 0; i < attributes.getLength(); ++i)
            cout << converter.to_bytes(attributes.getName(i)) << '=' <<
converter.to_bytes(attributes.getValue(i)) << endl;
30     }

32     virtual void endElement(const XMLCh* const name) {
        cout << "Element ist zu Ende: " << converter.to_bytes(name) << endl
;
34     }

36     virtual void characters(const XMLCh* const chars, const XMLSize_t
length) {
        cout << "Buchstaben (" << length << "):" << converter.to_bytes(
chars, chars + length) << endl;
38     }

private:
40     wstring_convert<codecvt_utf8_utf16<char16_t>,char16_t> converter;
};

42
43 int main(int argc, char* argv[]) {
44     try {
45         XMLPlatformUtils::Initialize();
46     }
47     catch (const XMLException& toCatch) {
48         char* message = XMLString::transcode(toCatch.getMessage());
49         cerr << "Error during initialization! :\n"
50             << message << "\n";
51         XMLString::release(&message);
52         return 1;
53     }

54
55     SAXParser* parser = {nullptr};
56     parser = new SAXParser;

57
58     int errorCount = {0};

59
60     try
61     {
62         //Das eigentliche Parsen der Datei
        SimpleSAXParser handler;
63         parser->setDocumentHandler(&handler);
64         parser->parse(argv[1]);
65         errorCount = parser->getErrorCount();
66     }
67     catch (const OutOfMemoryException&)
68     {
69         XERCES_STD_QUALIFIER cerr << "OutOfMemoryException" <<
XERCES_STD_QUALIFIER endl;
70     }

```

```

72     catch (const XMLException& toCatch)
73     {
74         char* message = XMLString::transcode(toCatch.getMessage());
75
76         //XMLString::release(message);
77         /*
78         XERCES_STD_QUALIFIER cerr << "\nAn error occurred\n  Error: "
79                                     << StrX(toCatch.getMessage())
80                                     << "\n" << XERCES_STD_QUALIFIER endl;
81         */
82         cerr << "XMLException: " << message << endl;
83     }
84     catch (...) {
85         cerr << "Unbekannter Fehler" << endl;
86     }
87
88     cout << "Anzahl Fehler: " << errorCount << endl;
89
90     //Parser sauber beenden
91     delete parser;
92     //Terminate muss immer am Schluss stehen
93     XMLPlatformUtils::Terminate();
94
95     return 0;
96 }

```

2.5.3 XML, utf-16, utf-8 (ASCII)

XML Parser arbeiten mit utf-16. Die Weiterverarbeitung in utf-16 ist sperrig. Eine Wandlung in utf-8 bietet sich hier an. Da im Programmentwurf nur mit ASCII gearbeitet wird ist utf-8 das richtige Format, denn ASCII sind die ersten 127 Zeichen von utf-8. Anbei ein paar Beispiele wie sich die utf-16 in utf-8 wandeln lässt.

```

#include <xercesc/util/XMLString.hpp>
2  const XMLCh * const UTF16TEXT = {u''Ich bin ein Text im utf-16 Format''};
3
4  //Eine Wandlung in utf-8 geht mit der statischen Klasse XMLString::
5      transcode
6  const char *utf-8 = XMLString::transcode(UTF16TEXT);
7  cout << utf-8 << endl;
8  XMLString::release(&utf-8);
9
10 //Aufwendig ist das freigeben des Speichers unter transcode.
11 //Die STL bietet hier eine einfachere Methode
12 wstring_convert<codecvt_utf8_utf16<char16_t>, char16_t> convUTF16UTF8;
13
14 //Falls sich die strlen mit der Methode XMLString::stringLen ermittel
15     laesst
16 string tagvalue = convUTF16UTF8->to_bytes(chars);
17 //Falls eine Laengenangabe mit verarbeitet werden muss

```

```
16 string tagvalue = convUTF16UTF8->to_bytes(chars, chars + length);
```

2.5.4 Muster XML

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <GetOptSetup SignPerLine="79">
  <Author Name="Thomas Staudacher, Andreas Maus" Phone="07541-77-961003"
    Mail="thomas.staudacher@zf.com, andreas.maus@zf.com" />
4  <HeaderFileName>options.h</HeaderFileName>
  <SourceFileName>options.cpp</SourceFileName>
6  <NameSpace>DHBW</NameSpace>
  <ClassName>COptionParser</ClassName>
8  <OverAllDescription>
    <Block>Erstellt einen Rumpf zum einlesen von Argumente aus der
      Kommandozeile.</Block>
10   <Block>Es kann sowohl mit innenliegenden Container wie externer
      Klassenanbindung eine Datenhaltung erfolgen.</Block>
    <Block>Sobald ein Methodenaufruf abstrakt ist, wird die Basisklasse
      abstrakt.</Block>
12   <Block>Fuer die Formatierung der generierten Dateien wird astyle
      verwendet.</Block>
  </OverAllDescription>
14  <SampleUsage>
    <Sample>getoptgen [options] ... QUELLE</Sample>
16   <Sample>getoptgen [--out-path] ... QUELLE</Sample>
  </SampleUsage>
18  <Options>
    <!-- Option help greift auf die interne Klasseninterne Methode
      printHelp zu. Ein gleichzeitiger Aufruf mit version und parse-only ist
      nicht erlaubt. -->
20   <Option Ref="1" ShortOpt="h" LongOpt="help" Exclusion="2,3"
      ConnectToInternalMethod="printHelp" Description="Diese Hilfe ausgeben
      und beenden" />
    <!-- Option version greift auf die interne Klassenexterne (abstrakte)
      Methode printVersion zu. Ein gleichzeitiger Aufruf mit help und parse-
      only ist nicht erlaubt. -->
22   <Option Ref="2" ShortOpt="v" Interface="Version" Exclusion="1,3"
      ConnectToInternalMethod="printVersion" Description="Gibt die Version
      des Programms aus und beendet" />
    <!-- Option out-path braucht ein zusatzliches Argument und schreibt auf
      einen Klassenintern generierten String das Argument. Die Option help,
      version und parse-only darf nicht angegeben sein -->
24   <Option LongOpt="out-path" HasArguments="Required" Exclusion="1,2,3"
      Interface="OutputPath" Description="Der Pfad wo das Ergebnis
      hingenriert werden soll (sonst ins aktuelle Verzeichnis)" />
    <Option LongOpt="astyle-path" HasArguments="Required" Interface="
      AstylePath" Exclusion="1,2,3" Description="Der Pfad wo die Astyle
      executable gefunden werden kann" />
26   <!-- Option sign-per-line kann ein zusatzliches Argument uebergeben
      werden und schreibt auf einen Klassenintern generierten Integer

```



```

SignPerLine. Wenn kein Parameter übergeben wird ist der Defaultwert 79.
Die Option help, version und parse-only darf nicht angegeben sein —>
28 <Option LongOpt="sign-per-line" HasArguments="optional" Interface="
SignPerLine" Exclusion="1,2,3" ConvertTo="Integer" DefaultValue="79"
Description="Die Anzahl der Zeichen pro Linie für den Helptext. Ohne
Argument wird der Standartwert genommen." />
28 <Option ShortOpt="n" LongOpt="only-if-newer" Interface="OnlyIfNewer"
Exclusion="1,2,3" Description="Generiert nur wenn die Eingangsdatei
neuer ist wie die bereits generierte" />
30 <Option LongOpt="no-format" Interface="NoFormat" Exclusion="1,2,3"
Description="Erzeugte Datei wird nicht formatiert" />
30 <!-- Option parse-only liest ein zusätzliches Argument ein und ruft
die interne Klassenexterne (abstrakte) Methode ParseXML auf. Die Option
help, version darf nicht angegeben sein —>
32 <Option Ref="3" LongOpt="parse-only" HasArguments="Required"
ConnectToInternalMethod="ParseXML" Exclusion="1,2" Description="Parst
die Datei einmal und beendet das Programm" />
32 </Options>
</GetOptSetup>

```