Freie Universität Berlin

Softwareprojekt Übersetzerbau 2010

Entwicklerdokumentation – Abschlussbericht

René Kijewski

7. August 2010

Das Softwareprojekt Übersetzerbau 2010 setzte sich zum Ziel, einen XML- nach Java-Source-Code-Übersetzter zu schreiben. Dieses Dokument erklärt im Speziellen die Komponenten XML-Lexer, Statement-Lexer und die Symboltabelle. Es richtet sich an Entwickler und enthält deshalb gegebenenfalls Jargon und versucht auch nicht, Sachverhalte zu vereinfachen. Die entsprechende Kundenpräsentation, die sich nicht an Programmierer richtet, steht unter https://dev.spline.de/svn/ss10-swp-uebersetzerbau/trunk/dokumentation/rk-kundenpraesentation/rk-kundenpraesentation.pdf zur Verfügung.

Inhaltsverzeichnis

1 Einleitung

Das Softwareprojekt Übersetzerbau 2010 machte es sich zur Aufgabe, eine selbst-erdachte Programmiersprache zu spezifizieren und einen Compiler für diese zu Implementieren.

Die Projektgruppe, bestehend aus – in alphabetischer Ordnung – Alexander Rau, Ansgar Schneider, Igor Merkulow, Markus Rudolph, René Kijewski und Stefan Meißner, entschied sich, eine XML-artige Eingabesprache zu benutzen, die Java-SE-6-kompatiblen[1] Sourcecode ausgibt. Dementsprechend wurde auch Java 1.6 als Programmiersprache gewählt. Als einheitliche Programmierungebung wurde Eclipse Galileo[2] verwandt. Apache Subversion[3], gehostet bei Spline[4], wurde als Versionskontrollsystem benutzt. Der Quelltext kann von https://dev.spline.de/svn/ss10-swp-uebersetzerbau/ heruntergeladen werden.

Ich übernahm die Komponenten XML- und Statement-Lexer, sowie die Implementierung einer Symboltabelle.

Bei den von mir implementierten Teilprojekten achtete ich auf hohe Wiederverwendbarkeit und versuchte, eine Benutzung besonders einfach zu gestalten. So legte ich größeren Wert darauf, dass Weiternutzer die Komponenten ohne lange Einlesezeit benutzen können als auf besonders große Performanz. Dieser Ansatz musste leider später aufgewicht werden. In den Abschnitten Arbeitsverlauf und Abschlussbemerkungen finden Sie weitere Informationen dazu.

Um den Einstieg in die Nutzung zu Vereinfachen, verwandte ich ausgiebig die verbreitete Entwurfsmuster, vorrangig das *Iterator-Pattern*[5] und die *Factory-Methode*[6]. Beide Pattern werden im Buch *Design Patterns: Elements of Reusable Object-Oriented Software*[7] beschrieben und sind vielen Entwicklern von objektorientiertem Code geläufig.

Die einzelnen Komponenten trennen so weit wie möglich zwischen Interface und Implementierung. Dies sollte es bei einer späteren Weiterarbeit am Projekt vereinfachen, die Komponenten auszutauschen. Bei der Spezifikationsphase überlegten wir uns unter anderem, dass auch eine Kompilierung in Java-Bytecode nativ möglich sein sollte.[8] Zudem sollte auch die Ausgabesprache auswechselbar sein, so dass nur das Kernstück, der Syntaxbaum, sowie das Interface der weiteren Komponenten, also Symboltabelle und Builder, unveränderlich wäre.

2 Teilprojekte

Die einzelnen Teilprojekte sind für sich allein verwendbar – die Verzahnung mit anderen Komponenten ist minimal –, da sie jeweils am Anfang der Bearbeitungskette stehen.

Die Lexer-Komponenten benutzen jeweils einen Zeichenstrom als Eingabe. Die Rückgabedaten speichern die Stelle im Eingabestrom an der sie standen, so dass spätere Komponenten, wenn sie einen Fehler in der Eingabe feststellen, eine geeignete Meldung samt Position ausgeben können. Die Symboltabelle verwendet dasselbe Interface, um die Eingabeposition zu speichern, jedoch ist die Bedeutung an dieser Stelle wahrscheinlich geringer als bei den Lexern.

2.1 XML-Lexer

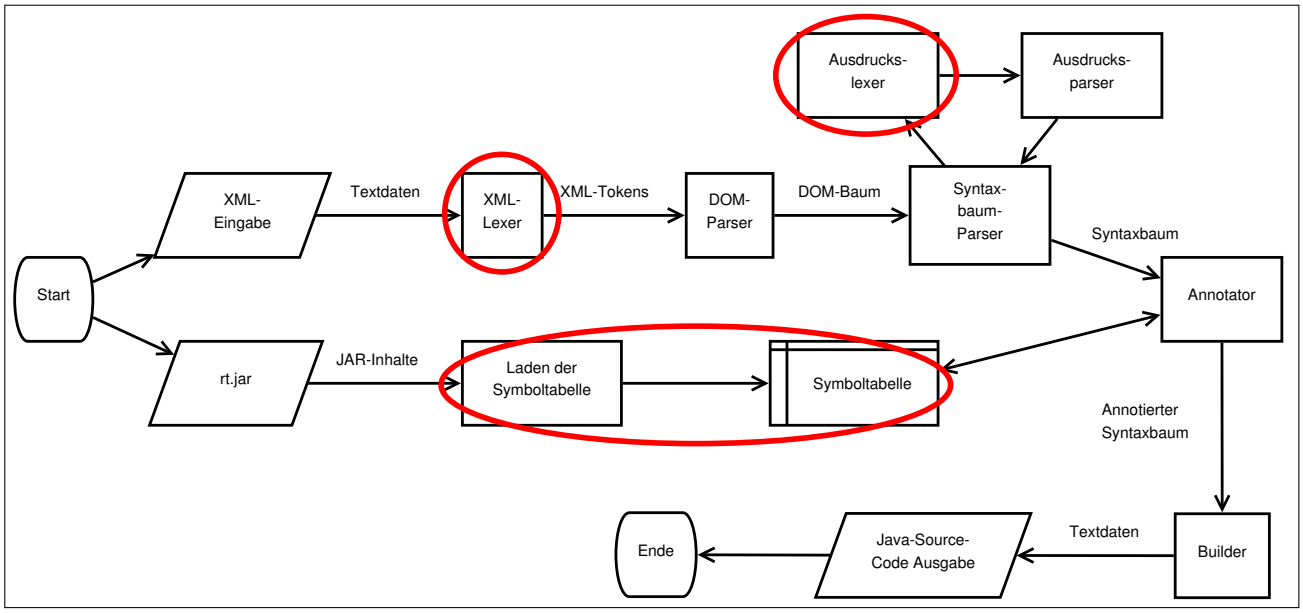
Der XML-Lexer *xmlNodeStream* vereinfacht die Arbeit des DOM-Parsers, indem er die syntaktischen Komponenten des Eingabestromes auftrennt, das heißt *tokenized*. Die erkannten Komponenten sind Textknoten, Kommentare, Tags, Endtags, Attribute und Processing instructions.

Als Eingabeformat wird XML 1.0 verwendet.[9] Es wurden bewusst Eineinfachungen jedenüber dem W3-Standard vorgenommen. Unter anderem sind nicht wohlgeformte Ausdrücke nach Art „`<tag attr="a < b"/>`“ erlaubt. Dieses Beispiel wäre aufgrund des Kleiner-Als-Zeichens im Attributewert nach der Spezifikation verboten.[10]

Die API des *xmlNodeStreams* ist an *StAX*, der Streaming API for XML[11], angelehnt. StAX hat sich seit der Spezifikation im JSR 173[12] aus dem Jahr 2006 als Standard für einfaches Parsen von XML-Eingaben etabliert. xmlNodeStream besitzt nicht dieselbe Mächtigkeit wie StAX und erfüllt in Folge dessen nicht die komplette Spezifikation des W3-Consortiums, da der XML-Prolog[13] nicht als solcher verstanden wird:

[1] http://java.sun.com/javase/6/
[2] http://www.eclipse.org/galileo/
[3] http://subversion.apache.org/
[4] http://dev.spline.de/
[5] http://c2.com/cgi/wiki?IteratorPattern
[6] http://c2.com/cgi/wiki?FactoryMethodPattern
[7] ISBN 0-201-63361-2
[8] Das heißt, dass wir nicht von `javac` abhängig sein wollten.
[9] Extensible Markup Language (XML) 1.0 (Fifth Edition)
[10] Spezifikation zu `AttValue`.
[11] http://stax.codehaus.org/
[12] http://jcp.org/en/jsr/detail?id=173
[13] Spezifikation zu `prolog`.

Abbildung 1: Verlaufdiagramm (in diesem Dokument beschriebene Komponenten hervorgehoben)



- Die XML-Deklaration[14] wird als Processing instruction[15] verstanden.
- Die Doctype-Deklaration[16] wird nicht unterstützt.

Konkretere Informationen über den Informationsfluss stehen in der API-Referenz.

Der XML-Lexer überprüft die Eingabe nicht auf Wohlgeformtheit, das heißt, auch „`<a></b>`“ würde nicht nicht als Fehler erkannt. Dies ist eine bewusste Designentscheidung, die einerseits den Overhead durch eine solche Überprüfung verhindern soll – der DOM-Parser führt eh über eine Überprüfung durch – und andererseits die Anwendungsmöglichkeiten der Implementierung verbreitern soll: Gewöhnliche Webseiten sind selten wohlgeformtes XML. Dadurch, dass *xmlNodeStream* agnostisch gegenüber Strukturfehlern ist, wäre er auch zum Lexen von Internetseiten geeignet. Fehler wie fehlende abschließende Anführungszeichen werden jedoch nicht übergangen, da sie eine aktive Fehlerkorrektur erforderten.

xmlNodeStream implementiert das Iterator-Interface Javas, wodurch der Einstieg in die Benutzung vereinfacht werden soll.

Beispiel

Quelltext 1: Beispiel.xml

```
<?xml version="1.0"?>
<module name="Test">
        Inhalt
</module>
```

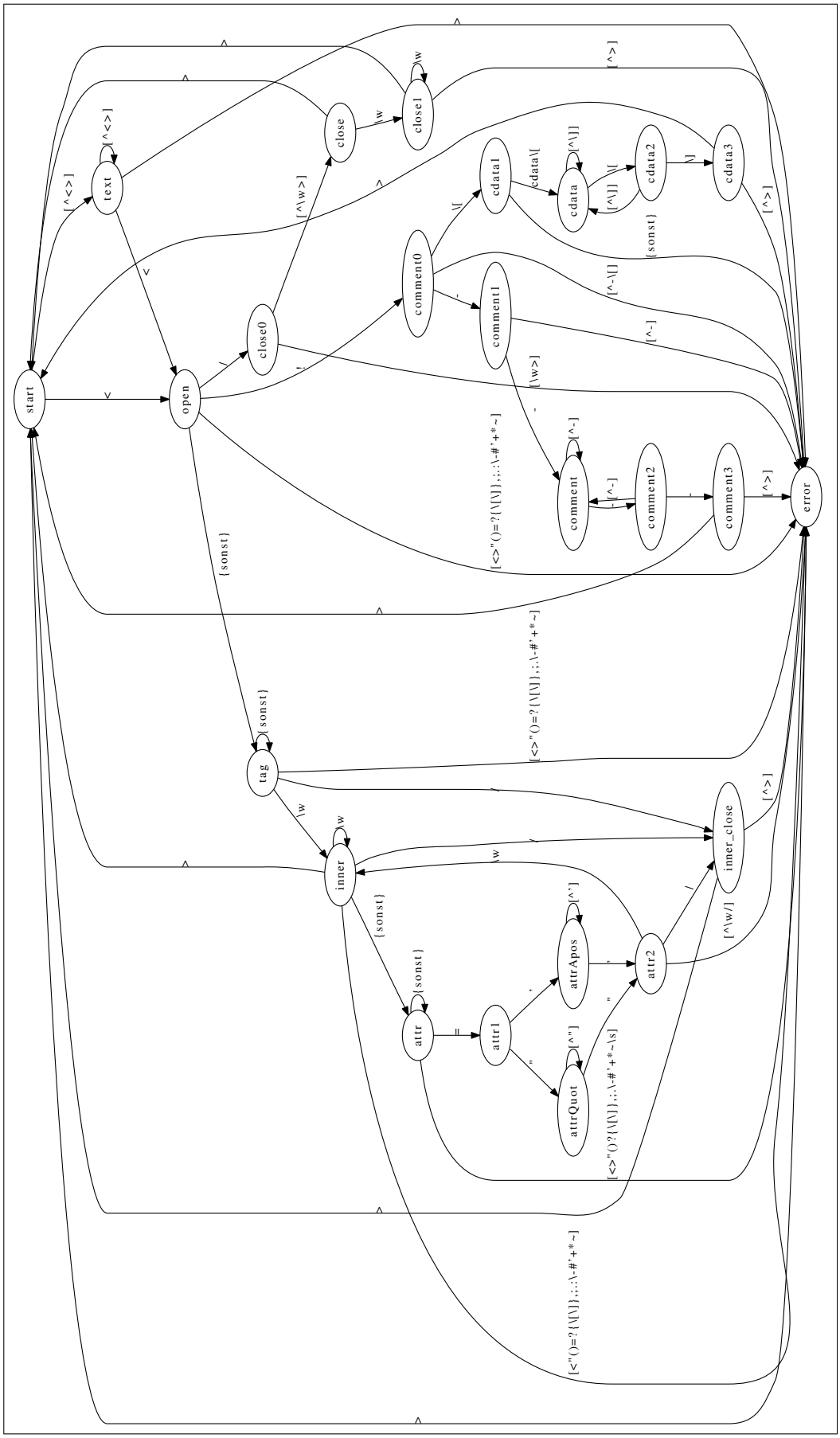
Die Eingabedatei obenstehende XML-Datei würde folgendermaßen erkannt:

- Processing instruction mit dem Namen „`xml`“ und Wert „`version="1.0"`“.
- Öffendes Tag mit dem Namen „`module`“.
- Attribute mit dem Namen „`name`“ mit dem Wert „`Test`“.
- Textknoten mit dem Wert „`Inhalt`“.
- Schließendes Tag mit dem Namen „`module`“.

An diesem Beispiel Erkennt man, dass das jedes eingelesene Token ein Tripel aus Typ, Name und Wert ist, wobei Name und/oder Wert auch leer beziehungsweise `null` sein können.

[14] Spezifikation zu `XMLDecl`.
[15] Spezifikation zu `PI`.
[16] Spezifikation zu `doctypedecl`.

Abbildung 2: Übergangstabelle der Eingabedaten



2.2 Statement-Lexer

Der Statement-Lexer hat eine ähnliche Aufgabe wie der XML-Lexer: Einen Eingabestrom in seine Komponenten auftrennen. Das Lexen der Statements ist die Vorstufe zum Parsen.

Der Statement-Lexer verwendet das Iterator-Interface, wobei die Rückgabedaten Tupel aus dem Typen des Datums und eventuell einem zugehörigen Wert, zum Beispiel bei Strings. In der derzeitigen Implementierung gibt es 33 Arten von erkannten Tokens, sowie zwei Statuscodes für das Eingabeende beziehungsweise bei lexikalisch fehlerhaften Eingaben. Der Umfang der erkannten Eingabefehler ist gering; die Fehlererkennung geschieht im *Statement-Parser*. Als Fehler würde unter anderem „`1a`“ erkannt; „`a b c`“ hingegen nicht.

Die Syntax der Statements lehnt sich an Java SE 6 an. Da Javas Operatoren nur ein oder zwei Zeichen lang sind[17] und es keine Zustände wie beim Lexen von XML-Daten gibt, wird keine Übergangstabelle benutzt, sondern ein `switch`- und `if`-Konstrukt. Dies vereinfacht im Allgemeinen sowohl das Lesen und Schreiben des Lexers, erschwert jedoch Spezialfälle. So wird zum Beispiel zum Erkennen der Tokens `new` und `null` auf Tricks wie Push-Back-Buffer zurückgegriffen. Ein entscheidender Vorteil ist jedoch, dass das Einlesen der Daten wesentlich schneller als beim XML-Lexer geschieht.

2.3 Symboltabelle

Die Symboltabelle ist die zentrale Speicherstelle der in der Eingabe vorkommenden Symbole. Symbole sind hierbei unter anderem Packages, Klassen und Methoden, jedoch auch Scopes und und `void`. Hiermit steht sie parallel zum Syntaxbaum. Der Annotator stellt die Schnittstelle zwischen beiden Komponenten dar.

Ebenso wichtig wie den Einblick in die Symbole des Eingabecodes zu haben, ist die Schnittstelle zum Zielsystem. So „kennt“ die Symboltabelle unter anderem die nativen Typen des Zielsystems und „weiß“, was verbotene Bezeichner sind beziehungsweise kann Bezeichner für das Zielsystem dekorieren. Die Symboltabelle „merkt sich“ Referenzierungen von Symbolen, die an der Stelle, in der sie im Quelltext auftraten, noch nicht definiert waren. Diese Symbole bezeichnet man als *unqualified*. Dem gegenüber sind bekannte Symbole *qualified*. Unqualified Symbols können unter anderem dann auftreten, wenn eine Funktion `a()` eine Funktion `b()` aufruft und `b()` wiederum `a()`.

Die Symboltabelle arbeitet in drei Phasen:

1. Aufbau des Grundsystems, das heißt, dass der Sichtbarkeitsraum mit den primitiven Datentypen gefüllt wird — bei Java also unter anderem `int` und `float`. Im nächsten Schritt werden die Symbole des Zielsystems eingelesen, was im Falle von Java unter anderem `java.lang.String`. Nach dem Einlesen der Symbole des Zielsystems wird der Kontrollfluss an den Builder abgegeben.
2. Im nächsten Schritt wird die Symboltabelle mit den Symbolen aus der Eingabedatei befüllt. Die Symboltabelle erkennt an dieser Stelle die Verwendung von ungültigen Bezeichnern und weist sie ab. Nachdem der Struktur der Eingabe in der Symboltabelle wiedergegeben wurde, wird versucht sämtliche Symbole zu qualifizieren.
3. Die letzte Phase stellt die Anwendung der Symboltabelle dar, in der der Builder sie unter anderem Benutzt, um die Datentypen von Symbolen zu erkennen, da Operatoren wie + je nach Datentyp eine andere Bedeutung haben. Sollte in der letzten Phase noch ein unqualified Symbol auftreten so muss dem Benutzer durch den Builder eine geeignete Fehlermeldung ausgegeben werden.

3 Arbeitsverlauf

Die Programmierung des XML-Lexers wurde bereits am 30. April begonnen; die Komponente ist deshalb wohl am ausführlichsten getestet. Die Aufgabe des Statement-Lexers wiederum wurde am 19. Mai begonnen und die Arbeit an der Symboltabelle wurde am 28. Mai aufgenommen, wodurch gerade für die letzte Aufgabe ein erheblicher Zeitdruck bestand.

[17] Der Operator >>> wird nicht verstanden.

3.1 XML- und Statement-Lexer

Die Arbeit am XML-Lexer verlief verhältnismäßig zielgerichtet. In der Vorlesung Übersetzerbau wurde ausgiebig erklärt, wie ein Parser geschrieben wird, so dass sich die tatsächliche Umsetzung als einfach erwies. Probleme ergaben sich nur, wenn ich Übergänge im Zustandsdiagramm (siehe Abbildung 2) übersah, weshalb ich sehr schnell dazu überging, das Diagramm in der Sprache DOT[18] festzuhalten. Zeichnungen auf dem Papier erwiesen sich nicht als sachdienlich.

Der Statement-Lexer wurde in kurzer Zeit geschrieben, da die Aufgabe wesentlich einfacher als beim XML-Lexer war. Wo beim XML-Lexer ein Zustand mitgeführt werden muss, der zum Beispiel vermerkert, ob man sich derzeit in der Mitte zwischen einem „<“ und „>“ befindet, existieren solche Schwierigkeiten beim Statement-Lexer nicht. Die zu implementierenden Operatoren habe ich unserem Notizblatt im Spline-Pad[19] entnommen.

3.2 Symboltabelle

Die Arbeit an der Symboltabelle erwies sind als kompliziert. Da die Aufgabe leider sehr spät erst vergeben wurde und aufgrund der Anforderung, dass sie Typisierung beherrschen sollte, gehörte die Symboltabelle planerisch zu den größten Komponenten.

Die Symboltabelle sollte zunächst für sämtlichen Zielsprachen gleich implementiert sein. Nachdem ich mich jedoch weiter in die Thematik einlas, erwies sich das nicht als möglich, da die verschiedenen Zielsysteme zu unterschiedlich sind. So haben verschiedene Programmiersprachen, selbst wenn man sich auf objektorientierte Programmiersprachen beschränkt, wenig Gemeinsamkeiten. Als Beispiele gegenüber Java sei die prototypen-basiere Programmiersprache ECMAScript[20] und ihre unterschiedlichen Implementierungen wie Javascript angeführt, deren Klassenkonzept keine Gemeinsamkeiten mit Java aufweist, jedoch sinnvolle Zielsprachen wären.

Ich beschränkte mich auf die Umsetzung einer Symboltabelle, die Java 1.6 sowohl für die Ausgabe nach Quelltext als auch Bytecode beherrscht. Das Interface wiederum sollte so generisch gehalten werden, dass später weitere Zielsysteme eingeführt werden könnten, wobei sich ein zu generischen Interface als zu kompliziert – und damit als unbenutzbar – erwies. Herr Rudolf gab mir des Öfteren sinnvolle hinweise für Verbesserungsmöglichkeiten am Interface, die mir zuvor nicht einfielen. Wiederum war das Interface dadurch während der gesamten Entwicklungszeit einem ständigen Wandel unterlegen, wobei ich mir eigentlich eine schnelle Festlegung wünschte, gegen die ich die konkrete Implementierung entwickeln könnte. Retrospektiv kann ich sagen, dass ich mich anfangs zu wenig mit meinen anderen Teammitgliedern zusammengesetzt habe und deshalb nicht die Fülle der Anforderungen kannte.

Bei der Programmierung an der Symboltabelle habe ich einige wertvolle Kenntnisse über die Interna Javas gewonnen. Gerade bei der Reflexion-API, die einem Programm zur Laufzeit Informationen über das Klassen- beziehungsweise Typsystem geben soll, habe ich erhebliche Wissenssprünge gemacht, die ich auch für andere Kurse nutzen konnte.[21] Leider erwies sich die sehr einfach zu benutzende Reflexions-API als Haupt-Flaschenhals des gesamten Projektes, weshalb ich einige „halbherzige“ Behelfsmaßnahmen eingebaut habe – *halbherzig* deshalb, weil ich das Problem nur versteckt und nicht beseitigt habe. Die Reflexions-API ist leider äußerst langsam und verbraucht große Mengen an RAM-Speicher, da die Inspizierten Klassen, sowie sämtliche (transitiven) Abhängigkeiten in den Speicher geladen werden, wobei eigentlich nur die Struktur der Klasse bedeutsam für die Erfüllung der Aufgabe wäre.

Als erste Behelfsmaßnahme werden nur noch Klassen in die Symboltabelle gelesen, die auch von außen sichtbar sind, das heißt entweder `public` oder `protected` sind. Default-Private und `private`-Klassen werden nicht mehr eingelesen. Jedoch offenbart diese Beschreibung bereits das größte Problem dieser Umsetzung: Nur weil Klassen nicht adressierbar *wäre*, gilt das nicht für zum Beispiel Rückgabewerte. Als weitere Maßnahme zum Beschleunigen habe ich das Laden parallelisiert, wodurch sich – gefühlt – ein fast linearer Anstieg der Geschwindigkeit mit der Zahl der Benutzen Rechenkernen ergab.

4 Offene Baustellen

Leider blieben aufgrund der geringen Projektdauer noch einige Komponenten fehlerhaft oder unvollständig. Jene Verbesserungsvorschläge, die mir einfielen, hielt ich in `Offene Baustellen.txt`[22] beziehungsweise als `TODO`s an der Entsprechenden Stelle im Quelltext fest. Die einzelnen Teilprojekte sind noch nicht ausreichend durch Unittests überprüft worden. Bei einer Weiterarbeit an dem Projekt täten folgende Programmierer gut daran, zuerst Blackbox-Text der einzelnen Komponenten durchzuführen, um eventuelle Verstöße gegen das Interface festzustellen.

[18] http://www.graphviz.org/doc/info/lang.html
[19] http://pad.spline.de/ep/pad/view/ro.v9mc/rev.9252
[20] http://www.ecma-international.org/publications/standards/Ecma-262.htm
[21] Siehe zum Beispiel `Containers.java` des XML-Technologien-Projekts
[22] https://dev.spline.de/svn/ss10-swp-uebersetzerbau/trunk/dokumentation/Offene%20Baustellen.txt

4.1 XML-Lexer

Der XML-Lexer steht an erster Stelle in der des Arbeitsablaufes und kann deshalb, trotz des Fehlens formeller Unittests als überprüft betrachtet werden. Eventuelle Fehler hätten sich beim Gebrauch zeigen sollen.

Offen ist noch die komplette Implementierung des XML-1.0-Standards, wie bereits oben erwähnt, wobei meines Erachtens der Mehrwert durch die vollständige Implementierung des Standars nicht der Aufwand rechtfertigen würde.

4.2 Statement-Lexer

Der Statement-Lexer enthält noch einen dokumentierten Fehler, dessen Ursache noch nicht ausgemacht werden konnte. Der Bezeichner „`nul`“ ist nicht erlaubt, was mit der Implementierung der Erkennung des Keywords `null` zusammenhängen wird. Das Erkennen der Schlüsselwörter ist derzeit suboptimal implementiert, da mir leider erst bei bei der Arbeit an der Symboltabelle ein besserer Algorithmus einfiel.

Derzeit werden wird die Eingabe mit einem Pushback-Buffer eingelesen. Sollten Zeichen gelesen werden, die nicht zu dem derzeitigen Token gehören, werden sie wieder in den Eingabestrom zurückgeschrieben. Sollte nur ein Zeichen zurückgeschrieben werden müssen, arbeitet diese Methode performant und ist zudem einfach zu implementieren. Sollten jedoch mehr Zeichen zurück zu schreiben sein, so kann sich diese Methode als kompliziert zu warten erwiesen. Zuerst schrieb ich einen Pushback-Buffer mit einem Zeichen als Pushback und schrieb ihn im weiteren Arbeitsverlauf zu einem allgemeinen Puffer um, in dem man beliebig viele Zeichen zurückschreiben kann. Vermutlich ist mit dabei ein Fehler unterlaufen, der in den Beobachtung resultiert, dass der Bezeichner „`nul`“ nicht erlaubt ist.

Eine sinnvollere Implementierung als die gleichzeitige Überprüfung, ob in der Eingabe ein Identifier oder ein Schlüsselwort steht, wäre, dass zuerst nur Identifier gelesen werden und, nachdem ein vollständiges Token gelesen wurde, überprüft wird, ob dieses zu einem Schlüsselwort wie `null` gehört.

4.3 Symboltabelle

Im weiteren Programmier- und Testverlauf erwies sich, dass die Symboltabelle optimiert werden muss, um nutzbar zu sein, da die Ladezeit auf Windows-Laptops mehrere Minuten betrug. So machte es zum Beispiel – und zwar in einem Maße, das mich selbst erstaunt hat – einen Unterschied, ob eine Stringeingabe gegen 10 oder $\log 10$ Strings verglichen wird, letzteres durch die Verwendung eines `Set<String>`. Häufig ist der iterative Vergleich von von kleinen Mengen von Werten, hier zirka zehn, schneller als ein optimierter, der in $\mathrm{O}(\log n)$ arbeitet, jedoch scheint der Vergleich von Strings in Java um mehrere Zehnerpotenzen langsamer als bei C zu sein.

Um das Problem der hohen Ladezeit zu umgehen und den Speicherbedarf minimal zu halten wären zwei Ansätze, idealer Weise kombiniert, gangbar.

- Die Reflexions-API ist zu mächtig und schwerfällig. Ein manuelles Auslesen der klassenstruktur auf Objekt-Code-Ebene ließe es zu, dass man unnötige Informationen nicht im Speicher hält beziehungsweise gar nicht erst einliest.
- Ein Nadelöhr stellt die Benutzung von `TreeMap`s dar, die ich an vielen Stellen verwende, da sie im Gegensatz zu `HashMap`s nicht die Gefahr von Kollisionen in sich bergen. Vermutlich ist die Datenmenge klein genug, so dass es zu keinen Kollisionen der Streuwerte käme. Eine Überprüfung dieser Annahme wäre jedoch erforderlich.

5 Abschlussbemerkungen

Eine Lektion des Projektes war, dass Besprechungen notwendig sind. Noch bevor man eine Idee umsetzt, sollte man sich mit den betroffenen Teammitgliedern zusammen setzten und um gemeinsam die Implikationen abzuschätzen oder Verbesserungsvorschläge einholen zu können. Ebenso lassen sich Beschreibungen wesentlich besser vis-à-vis vermitteln als das mit Texten möglich wäre. Als Programmierer ist man häufig zu tief in der Materie drin, als dass man sinnvolle Fragen abschätzen könnte, die in eine Dokumentation zu schreiben wären, wodurch zudem viel Zeit mit dem Schreiben unnötiger Kommentare verloren geht. Ich denke, dass Pair-Programming das Mittel der Wahl ist, wenn es sich zeitlich organisieren lässt. Die Kommunikation über Chat oder Telefon kann leider nicht dasselbe leisten.

Die wöchentlichen Teambesprechungen habe ich als reine Zeitverschwendung empfunden, da wöchentliche Berichte gerade in der Anfangsphase eines Projektes – über die das Projekt aufgrund des Zeitdrucks auch nicht heraus kam – viel

zu selten sind und man zu viele für einen selbst unrelevante Informationen anhören muss, so dass das Aufmerksamkeitsfenster bei relevanten Dingen bereits wieder geschlossen sein kann. Dass die Teambesprechungen in unserem Fall um acht Uhr morgens stattfanden, war der Aufmerksamkeit ebenso nicht zuträglich.

Nichtsdestoweniger hat das Projekt Spaß gemacht, wobei leider einige – meines Erachtens eklatante – Fehler in der Planung gemacht wurden. So haben wir viel zu viel Zeit damit aufgewendet, Lexer und Parser zu schreiben, wobei es zu diesem Zwecke bereits viele gute Open-Source Tools gäbe, die wir hätten stattdessen verwenden sollen. Tatsächlich habe ich für die Verwendung einer XML-ähnlichen Sprache optiert, damit wir einen vorhandene DOM-Parser hätten benutzen können.

Ich für meinen Teil hatte mir erhofft, mehr über die Optimierung von Zwischencode zu lernen; andere Projektmitglieder wiederum wollten eine Programmiersprache mit neuartigen Konzepten schaffen, wobei ich gerade Herrn Rudolfs Konzept einer Zustandsmaschine interessant fand. Ebenso hörte sich das Konzept einer einsteigerfreundlichen Programmiersprache gut an.[23] Retrospektiv muss ich sagen, dass wir uns als Gruppe zu wenig Zeit genommmen haben, uns über die Möglichkeiten klar zu werden. Leider haben wir die Konzeptphase zu schnell abschließen wollen, wobei ich jedoch hoffe, so einen Fehler in Zukunft nicht mehr zu machen.

Im Verlauf des Projektes wurde leider klar, was wir die Planung des Projektes überstürzt hatten und viele wichtige Komponenten nicht formell aufgeschrieben oder schlichtweg übersehen haben. Teilweise kam es zu unnötigen Leerlaufphasen, da es scheinbar zu jedem Zeitpunkt keine Aufgabe gab. Erst in weiteren Gesprächen wurde klar, welche Dinge noch zu tun wäre. So wurde unter anderem die Aufgabe der Symboltabelle bei der Aufgabenverteilung erkannt, jedoch nicht vergeben,[24] so dass sie später vergessen wurde. Dass später seitens der Teamleiter angemerkt wurde, dass der generierte Code typensicher sein solle und die Symboltabelle dies sicherstellen soll, war dann ein umso größerer „Schock“ für mich.

Zur Eskalation am Ende des Projektes hat meinerseits gerade geführt, dass das Projekt nicht die Dinge umfasste, die wir uns am Anfang vorgestellt haben. Ich wollte Probleme offen ansprechen, um sie eventuell besser angehen zu können — Probleme, die nicht angesprochen werden, werden sich nur vergrößern. Leider ist mein Ton wohl zu aggressiv herübergekommen,[25] wodurch ich jene Problem wohl nur noch verstärkt habe. Ferner jedoch hat die strikte Trennung zwischen „Management“ und den Programmieren zur Eskalation beigetragen oder sie erst möglich gemacht. Es bestand ein großes Misstrauen zwischen uns und dem „Management“, unter anderem geschuldet dadurch, dass uns nicht mitgeteilt wurde, was in den Wochenbesprechungen zwischen den Teamleitern und Frau Fehr beredet wurde,[26] sowie dass uns Programmierern die Dokumentationen und Berichte nicht ausgehängigt wurde,[27] als wir danach fragten. Die strikte Weigerung der Teamleiter uns ihren Bearbeitungsstand der Dokumentation mitzuteilen erweckte bei uns Entwicklern den Eindruck, das Herrn Schneider und Merkulow keine Aufgaben im Projekt übernähmen. Die Antwort, dass sie sich jede Woche nach dem Gruppengespräch für eine Stunde mit Frau Fehr träfen, wirkte für mich regelrecht höhnisch. Meines Erachtens ist an dieser Stelle wesentlich mehr Transparenz nötig.

Ich denke, dass es für weitere Softwareprojekte wichtig ist, die Teamleitung dazu anzuhalten, sich auch in die Programmierung einzubringen, da ansonsten die Stimmung *unweigerlich* kippen *muss*. Dennoch denke ich, dass ich durch dieses Softwareprojekt eine Erfahrung fürs Leben gemacht habe — mehr als es ein Besuch der Vorlesung Softwaretechnik[28] je könnte.

6 API-Referenz

6.1 XML-Lexer

Der XML-Lexer trennt zwischen Interface und Implementierung. Das Package des Interfaces heißt `de.fu_berlin.compilerbau.xmlNodeStream`; die Implementierung befindet sich unter `de.fu_berlin.compilerbau.xmlNodeStream.impl`.

Interface

[23] Ich muss gestehen, dass ich den Namen des Diplomstudierenden vergessen habe.
[24] Siehe Stand der Besprechung vom 7. Mai im Spline-Pad: http://pad.spline.de/ep/pad/view/ro.v9mc/rev.7990
[25] Vermutlich war er aggressiv, was jedoch nie meine Intention war.
[26] Meine Haupsorge an dieser Stelle war, dass unsere Bearbeitungsstand als besser als er war weitergegeben wurde, so dass ein mäßiges Ergebnis beim Projektende umso schlechter auf uns zurückfalle würde.
[27] Das Spline-Pad war nie als normativ angedacht, weshalb sich die Teamleiter bereit erklärten sich, eine formelle Definition zu schreiben. Es war deshalb unverständlich für mich, warum uns diese Arbeitsgrundlage vorenthalten wurde.
[28] https://www.inf.fu-berlin.de/w/SE/VorlesungSoftwaretechnik2008

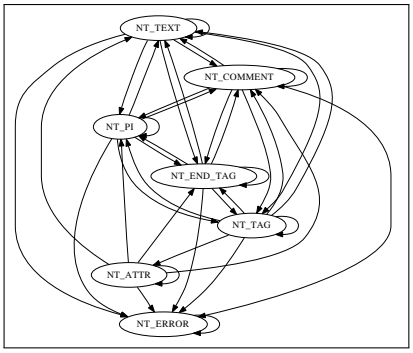
Das Interface definiert in `NodeType.java` die Arten der zu erkennenden Knoten. Dies sind derzeit `NT_TEXT`, `NT_COMMENT`, `NT_TAG`, `NT_END_TAG`, `NT_ATTR`, `NT_PI`, sowie für fehlerhaften Eingaben `NT_ERROR`.

Ein einzelnes gelesenes Token wird in einem `XmlNode` zurückgegeben, welches die Methoden `NodeType getType()`, `PositionString getKey()` und `PositionString getValue()` besitzt. Nicht für alle Tokenarten sind sowohl Key also auch Value gesetzt. Konsultieren dafür bitte das entsprechende JavaDoc für `XmlNode`. Ein `XmlNode` implementiert das Interface `StreamPosition`.

Die letzte Komponente des Interfaces des XML-Lexers stellt `XmlNodeStream` dar, welches angibt, dass die XmlNodeStreams das Interface `Iterable<XmlNode>`, `StreamPosition` und `Closeable` erfüllen müssen.

Wie in Abbildung 3 zu sehen, kann der XmlNodeStream nicht an jeder Stelle ein beliebiges Token zurückgeben. So kann unter anderem `NT_ATTR` nur auf `NT_ATTR` oder `NT_TAG` folgen. Bei der Benutzung von `XmlNodeStreamFactory` darf man sich darauf verlassen, dass dieser Contract nicht verletzt wird.

Abbildung 3: Mögliche Rückgabewerte des Statement-Lexers nach Zustand



Implementierung

Die Implementierung verfügt nur über eine öffentlich sichtbare Klasse, nämlich `XmlNodeStreamFactory`. Die Klasse verfügt über drei statische Factory-Methoden

- `static XmlNodeStream createNewInstance(PositionString str);`
- `static XmlNodeStream createNewInstance(Reader reader);`
- `static XmlNodeStream createNewInstance(Reader reader, StreamPosition pos);`

Wobei die ersten beiden Methoden Wrapper für die letzte Methode darstellen.

6.2 Statement-Lexer

Der Statement-Lexer trennt zwischen Interface und Implementierung. Das Package des Interfaces heißt `de.fu_berlin.compilerbau.statementLexer`; die Implementierung befindet sich unter `de.fu_berlin.compilerbau.statementLexer.impl`.

Das Interface definiert in `TokenType.java` fünfunddreißig verschiedene Arten von gelesenen Tokens, wobei `EOF` und `ERROR` Statusmeldungen darstellen. Die Tokenarten orientieren sich an der Grammatik Javas und enthalten auch keine eigenen Elemente. Jedes gelesene Token wird in einem `StatementNode` zurückgegeben, wobei ein Node ein Tupel aus dem Typen und optional einem Wert darstellt. Tokens tragen nur dann einen Wert, wenn sie vom Typ `ID`, `STRING`, `INT` oder `REAL` sind. Zum Auslesen der Werte gibt es die Methoden

- `Object getValue() throws IllegalAccessException;`
- `Number getNumber() throws IllegalAccessException;`
- `CharSequence getString() throws IllegalAccessException;`

Eine `IllegalAccessException` wird geworfen, wenn das Token keinen (entsprechenden) Wert enthält. `getNumber()` enthält nur für `INT` und `REAL` einen Wert; `getString()` wiederum nur für `ID` und `STRING`.

Das Tokenisieren geschieht über die Factory-Methode `StatementLexer.tokenize(PositionCharacterStream)`, wobei der Rückgabewert ein `Iterable<StatementNode>` ist und somit über die Methoden `next()` und `hasNext()` angesprochen werden kann.

6.3 Symboltabelle

Das Interface der Symboltabelle liegt im Package `de.fu_berlin.compilerbau.symbolTable`. Die Implementierung ist – wie der eng verbundene Annotator – abhängig vom Zielsystem.[29] Meine Implementierung der Symboltabelle hat als Zielsystem Java 1.6. Die konkrete Implementierung liegt in `de.fu_berlin.compilerbau.symbolTable.java`. Die Ausnahmebehandlungsklassen liegen in `de.fu_berlin.compilerbau.symbolTable.exceptions`. Die Exceptions sind ihrerseits unabhängig vom Zielsystem.

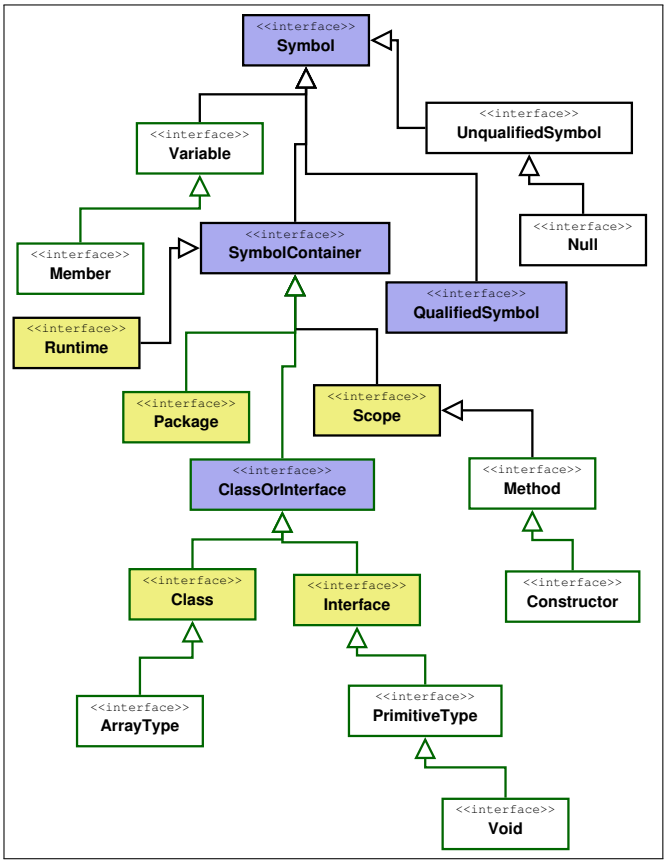
Die Aufbau der Symboltabelle ist verhältnismäßig komplex. Um die Benutzung zu vereinfachen, habe ich auf ein Factory-Pattern gesetzt. Aus dem Paket der Implementierung ist nur die Klasse von `RuntimeFactory` von außen sichtbar. Sämtliche anderen Klassen sind package-private. Diese Klasse wiederum hat nur eine public Methode:

- `static Runtime newRuntime(URL[] classpath, URL rtJar)`

Mit dem Parameter `classpath` kann eine Liste mit JARs übergeben werden, deren Interface bei der Kompilierung verfügbar sein soll. Die Liste entspricht dem Parameter `-classpath` von `javac`. Die `rt.jar` wiederum enthält das Basissystem Javas, vornehmlich die Klassen in `java.lang`. Aufgrund der Benutzung der Reflexions-API (siehe Abschnitt unter „Offene Baustellen“) muss die übergebene `rt.jar` dieselbe sein, die die JVM des Compilers verwendet. Dieses Problem durch die Abkehr von der Reflexions-API behoben werden.

Jedes Symbol hat einen Symboltypen. Die Symboltypen sind in der Klasse `SymbolType` definiert. Die vier bläulich hinterlegten Interfaces in der Abbildung 4 haben keinen äquivalenten Symboltyp – sie sind reine Metaklassen, die nur ein gemeinsames Interface der erbenden Klassen zusammenfassen, wobei es jedoch keine tatsächlichen Instanzen dieser Interfaces geben kann. Alle in der Abbildung grün umrandeten Interfaces erben von `QualifiedSymbol`.

Abbildung 4: Vererbungshierarchie der Symbole:
Factories gelblich hinterlegt
Metaklassen bläulich hinterlegt
qualified Symbols grün umrandet



Das Interface der Symboltabelle ist streng hierarchisch aufgebaut. Es gibt die eine Basisklasse, von der alle anderen Interfaces (transitiv) erben. Diese strenge Hierarchie ist jedoch nur im Interface garantiert. In tatsächlichen Implementierungen könnten verschiedene Klassen zusammengefasst werden, um die Programmierung zu vereinfachen, weshalb eine Abfrage, ob ein Objekt Instanz einer bestimmten Symbolart nicht über `instanceof` erfolgen darf. Aus diesem Grund habe ich die Abfragemethode `Boolean hasType(SymbolType leftType)` eingeführt. Diese gibt das Ergebnis in drei-wertiger Logik zurück, wobei der Rückgabewert `null` gleichbedeutend mit unbekannt ist. Das Ergebnis `null` wird einerseits zurückgegeben, wenn das abgefragte Objekt unqualified ist oder `leftType` den Wert `UNQUALIFIED` trägt.

Die Bedeutung des Parameters `leftType` ist hierbei so zu verstehen: Das abgefragte Symbol ist der `rightType`. Das Methode `hasType` wiederum entspricht der Implikation. Es wird also `true` zurückgegeben, falls $leftType \Rightarrow rightType$. Dies ist der Fall, wenn `leftType` in der Verbindungshierarchie über `rightType` steht, beziehungsweise, wenn beide Typen gleich sind.

Die drei gelblich hinterlegten Interfaces sind Factories und können weitere Elemente erstellen, die sie dann enthalten. Außer durch die Factories können keine Objekte instanziiert werden. Instanzen können nicht an andere Container abgegeben werden. Dies soll versehentlichen Anwendungsfehlern vorbeugen.

[29] Angemerkt sei hierbei, dass die Implementierungen des Annotators und der Symboltabelle nicht von der konkreten anderen Implementierung abhängig sind.

- `Runtime`s können `Package`s erzeugen.
- `Package`s können `Class`es und `Interface`s erzeugen.
- `Class`es können `Member`s erzeugen.
- `Class`es und `Interface`s können `Method`s erzeugen, wobei jede Methode einen äußersten `Scope` enthält, abrufbar mit `Scope getScope()`.
- `Scope`s können `Variable`s erzeugen.

Sämtliche anderen Instanzen werden durch die `Runtime` erzeugt.

Abfragen nach Objekten geschehen in `SymbolContainer`n und ihren Unterinterfaces. Zu diesem Zweck stellt das Interface die Methode

- `Symbol tryGetQualifiedSymbol(PositionString name)`

bereit. Das zurückgegebene Symbol könnte qualified sein, falls das referenzierte Symbol bereits bekannt ist. Sollte das Symbol zum Zeitpunkt des Aufrufes noch nicht bekannt sein, so wird ein `UnqualifiedSymbol` zurückgegeben. Zu einem späteren Zeitpunkt kann ein unqualified Symbol mittels `QualifiedSymbol qualify()` qualifiziert werden. Sollte das Symbol dann immer noch nicht bekannt sein, so wird `null` zurückgegeben. Der Aufrufer muss in diesem Fall geeignet reagieren.

7 Anhang

7.1 Abbildungs- und Quelltextverzeichnis

Abbildungsverzeichnis

Quelltextverzeichnis

7.2 Lizenzierung

Teile des Quelltextes des Softwareprojektes Übersetzerbau 2010 stehen unter der Freien Lizenz GNU AGPL in der Version 3.0. Die Dokumentation selbst, sowie die enthaltenen Graphiken, werden unter den Bedingungen der GNU FDL in der Version 1.3 weitergegeben. Bindend sind nur die englischen Originaltext, die im nächsten Absatz angehängt sind.

Sollten Sie den Quelltext zu anderen Konditionen weiterverwenden wollen, so setzen Sie sich bitte mit mir in Verbindung: rene.kijewski@fu-berlin.de.

7.2.1 Quelltext

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.>

Copyright (C) <textyear> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a “Source” link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see `http://www.gnu.org/licenses/`.

7.2.2 Dieses Dokument

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.