Spezialthema Instant Messaging in python

Lukas Prokop

9. Juni 2009

Inhaltsverzeichnis

Kapitel 1

Instant Messaging

1.1 Der Begriff

Unter Instant Messaging bezeichnet man allgemein die Kommunikation unter einer Gruppe von Menschen über das Internet (oder kleinere Netzwerke) in Echtzeit. Ein Benutzer sendet dabei Nachrichten gezielt an eine Person, eine Benutzergruppe oder alle Benutzer. Die Nachrichten werden in Echtzeit übertragen, was bedeutet, dass übertragene Nachrichten in weniger als 1 Sekunde beim Kommunikationspartner erscheinen. Wie Daten übertragen werden, wird dabei in Protokollen spezifiziert (folgend werden einige vorgestellt). Den Prozess des Versenden und Empfangen von Nachrichten erledigen dabei eigene Programme (Clients), die bei manchen Betriebsystemen nicht mitgeliefert werden. Die Ansprüche an die Protokolle sind unterschiedlich. In manchen ist es möglich Verschlüsselung, Datentransfers und Video-Streams zu nutzen. Die Ansprüche an Clients sind beispielweise Smileys, Accountverwaltung, Aufzeichnung von Gesprächen und Einbinden von Erweiterungen mithilfe eines Plugin-Systems.

Sogenannte Multiprotokoll-Clients sind fähig mit unterschiedlichen Protokollen umzugehen. Sie verfügen meist über ein umfassenderes Account-Management und man erspart es sich für jedes Protokoll einen neuen Client installieren zu müssen. Jedoch ist die Verwendung von alternativen Clients oft nicht gestattet (siehe hierzu Sektion "OSCAR"). Ein Beispiel für einen Multiprotokoll-Client ist Pidgin. 2007 wurde Pigdins Logo (ein lila Vogel) zum hässlichsten Maskottchen gewählt, aber die Funktionalität und die Plugins machen dieses Manko wieder wett.

Kapitel 2

Verbreitung

2.1 OSCAR

OSCAR (Open System for Communication in Realtime) ist das IM-Protokoll von ICQ ("i seek you") und AIM ("AOL Instant Messenger"). Wie der Name nicht verrät, ist das Protokoll geschlossen und war bis 2008 ein Geheimnis, welches durch Reversed Engineering entschlüsselt wurde. Besondere Kritik empfangen die Protokolle durch ihre AGB, welche die Verwendung von firmenfremden Chatprogrammen verbietet. Allerdings stehen die firmeneigenen Programme auch unter erheblicher Kritik. Viele Programmierfehler wurden zu langsam behoben, es fehlt an Transparenz (die Benutzer werden nicht in den Entwicklungsprozess des Protokolls eingebunden) und eine Verschlüsselungsmöglichkeit ist nicht vorgesehen. Die ICQ-Server können somit die privaten Nachrichten aufzeichnen und die AGB sieht vor, dass die Urheberrechte beim Anmelden abgetreten werden.

Die OpenSource-Community ist nicht bereit den Forderungen von AOL anzukommen und es existiert eine Vielzahl an freien Clients, die eine Kommunikation über ICQ/AIM erlauben. Die ICQ-Server können damit nicht zwischen alternativen und "echten" Clients unterscheiden, da beim Login nur ein Clientidentifier übertragen wird, der den Client identifizieren soll. Da aber alle fremden Clients gesperrt werden, melden sich fremde Instant Messengern mit dem originalen Identifier an.

ICQ ändert unregelmäßig diesen Identifier. Dabei sind nicht nur die fremden Clients von Problemen betroffen, sondern auch die eigenen User. Statt des Loginprozesses sendet der Server dann eine Nachricht "Please upgrade to the newer version" aus. "Echte" ICQ-User werden dann gezwungen die aktuelle Version herunterzuladen und "fremde" ICQ-User müssen auf die Identifier-Adaption der Entwickler warten. Die OpenSource-Gemeinde reagiert auf solche Änderungen sehr schnell. Allerdings müssen sie die Login-Daten immer reverse engineeren. Humorvoll ist dabei zu sehen, dass oft der Neuinstallationsprozess vom originalen ICQ aufwändiger ist als bei einem fremden Client. Gerade unter Linux werden die Updates nach 2-3 Tagen verfügbar und werden meist direkt über einen

Paketmanager in Sekunden hinaufgespielt. Wer sich hingegen unter icq.com die aktuelle Version runterlädt, muss wegen der hohen Zugriffsraten zu solchen Zeiten mit Downloadproblemen rechnen. Eine ICQ-Version für Linux wird außerdem gar nicht von AOL angeboten.

Die Identifizierung der einzelnen User erfolgt über eine UIN (Unique Identification Number). In den Clients selbst wird aber ein Username eingetragen, den der User frei wählen und ändern kann. Da ICQ zeitweise eine große Popularität besaß, entwickelte sich ein Markt um einfach zu merkende UINs. Auch heute gibt es noch ICQ-Nummern bei ebay zu kaufen.

Ein großer Vorteil vom ICQ ist das Offline-Versenden von Nachrichten. Ist ein User offline, kann ihm eine Nachricht gesendet werden und beim nächsten Login erscheint ihm die Nachricht. Ein paar Protokolle verschlucken solche Nachrichten.

2.2 IRC

IRC ist eine Abkürzung von Internet Relay Chat und heutzutage eine der wichtigsten Anlaufstellen für Programmierer. IRC wurde in den 90ern entwickelt und hat sich seither etabliert. IRC organisiert sich in themenbasierenden Channel, wo jeder mitdiskutieren kann. Teilnehmer sind der IRC-Server selbst, Bots (zeichnen Logs auf), Bouncer (reservieren Usernamen) und die Clients (normalen Benutzer). Die Anzahl der Clients ist unbegrenzt und wird nur durch die Belastbarkeit des Servers definiert. Bekannte (große) Server sind freenode, quakenet und ircnet.

Um teilzunehmen, benötigt man ein IRC-fähiges Programm, wobei hier auch Plugins für Webbrowser[1] existieren. Hat man ein solches Programm gestartet, kann man meist ein Profil angeben und verschiedene IRC-Server auswählen. Die bekanntesten sind meist gelistet. Steht man in Verbindung mit dem Server, kann man sich von ihm die Liste mit den Channels (deutsch: Räume) geben lassen. Jetzt kann ein Channel selektiert und betreten werden.

Natürlich gibt es viele graphische Frontends, aber im Prinzip ist IRC vollständig textbasierend. Sämtliche Anweisungen an den IRC-Server werden mit einem Slash eingeleitet. Das Anzeigen der Räume (beginnend mit py) erfolgte beispielweise mit */list py**. Die Befehle selbst werden jedoch an den Server gesendet und dort abhängig. Da nicht überall die selbe Serversoftware läuft, kann es Unterschiede in den Befehlen geben.

IRC ist grundlegend offen konzipiert. Das heißt IRC-Server sollten es den Usern selbst erlauben neue Channels zu eröffnen und zu verwalten. Jedoch ist auch dies serverabhängig. freenode wird einen brg-viktring Channel gestatten; mozilla.org jedoch nicht. IRC ist nicht für Verschlüsselung konzipiert. Es ist möglich abseits des Channels (mit /query) einen einzelnen User anzusprechen. Channels werde immer mit einer beginnenden Raute benannt (zB #python.de).

[1] https://addons.mozilla.org/en-US/firefox/addon/16

Ein Grundproblem des IRCs sind Zeichensätze. Mit den lateinischen Buchstaben kommt man natürlich überall zurecht, doch aufgrund der unterschiedlichen Client- und Serverzeichensätzen gibt es meist zumindest ein paar Nutzer, die sich nicht an den Standard (meist UTF-8) halten und Umlaute werden in Folge falsch vom eigenen Client interpretiert.

Sehr interessant sind Nutzerstatistiken (zB die Anzahl an Nachrichten im Jahr 2009 im Channel #python sind aufschlussreich darüber, wie stark die Programmiersprache im Vergleich genutzt wird) und eigene Webseiten sammeln IRC-Quotes[2].

2.3 XMPP & Jabber

Das Extensible Messaging und Presence Protocol schließt an den freien Gedanken an und wollte Systemen wie ICQ und Skype entgegenströmen, die von Firmen verwaltet werden. XMPP ist somit total frei, quelloffen und wird von einer Community gut dokumentiert. XMPP setzt auf XML als Übertragungsformat auf. Das wichtigste Einsatzgebiet von XMPP ist Instant Messaging und daraus hervorgegangen ist das Protokoll Jabber. Eine Alternative wäre Google Talk.

Jabber-Clients bekommen eine Jabber-ID verliehen, die syntaktisch an eine E-mailadresse erinnern. Sehr gefragt ist Jabber bei Personen, die wert auf Verschlüsselung legen. Dabei wird sowohl Client-to-Server wie auch Server-to-Server-Verschlüsselung angewandt. Zum Einsatz kommt bevorzugt OpenPGP (asymmetrische Verschlüsselung; IDEA und RSA).

Bekannte Server sind beispielweise jabber.org und der CCC[3] betreibt auch einen. Die Server sind total abgeschottet organisiert. Das heißt die Nachrichten für einen Benutzer können über 20 verschiedene Server laufen, weil sich die Server nicht direkt kennen. Jeder Client könnte auch problemlos einen Server betreiben. Jabber ist somit dezentralisiert organisiert.

Ein Feature von Jabber sind transports. Bei transports handelt es sich um Server, die eine Anbindung an andere Protokollen besitzen. Besitzt man einen Jabber-Account und kennt verfügbare transports, können Nachrichten an den verfügbaren Server geschickt werden und dieser translatiert die Nachricht in ein anderes Protokoll. Über dieses Protokoll wird dann die Nachricht an den gewünschten Benutzer weitergeleitet. Da die meisten Protokolle Logins bei Servern verlangen, müssen sich die Clients bei transports grundsätzlich anmelden und dort ihre Login-Daten für das andere Protokoll hinterlassen, was als riskant anzusehen ist. Letztendlich vertraut man auf transports von Organisationen, die sich für Informationsfreiheit verschrieben haben.

[2] zB ibash.de, bash.org, german-bash.org
[3] Chaos Computer Club – deutscher Hackerverein

Kapitel 3

Werkzeuge

3.1 python

python ist eine Programmiersprache, die mehrere Programmierparadigmen erlaubt, doch vorwiegend ist sie auf funktionale Ausdrücke und objektorientierte Konzepte optimiert. Sie wird seit 1990 vom Monty Python-Fan Guido van Rassum entwickelt. Implementierungen von Python sind CPython (am weitesten verbreitet; in C entwickelt), Jython (Java), Stackless Python (experimentell), IronPython (experimentell, C#, Anbindung an .NET) und PyPy (experimentell, python in python, noch zu langsam). Alle Implementierungen erzeugen einen Bytecode, der dann vom Interpreter ausgeführt wird. Dies sieht man auch, wenn man ein Programm auf der Konsole mit "python monty.py" startet. CPython legt dann eine Datei "monty.pyc" mit Bytecode an. Python wurde von Anfang an so konzipiert, dass die Sprache leicht lesbar ist. Dies wird erreicht durch wenig Schlüsselwörter im Vergleich zu anderen Sprachen und die Syntax wurde auf ein Minimum reduziert.

3.1.1 Paradigmen, Geschichte und python.org

Das wichtigste Kriterium ist die Philosophie "Ein Befehl pro Zeile". Zwar lassen sich die Befehle durch Kommata (wie in C) trennen, doch wird dieser Stil nicht gepflegt. Durch List Comprehensions und andere funktionale Ausdrücke werden zwar mehrere Anweisungen in einer Zeile angegeben, aber es wird trotzdem eine Aktion pro Zeile ausgeführt.

```
primes = [x for x in xrange(1,1000) if not False in [ggt(y, x) == 1 for y in
xrange(1,x-1)]]
```

Als Resultat legen Python-Programmierer großen Wert auf einen guten Programmierstil. Guido van Rossum veröffentlicht auf python.org sogenannte Python Enhancement

Proposals, wo sich der Autor an die Community richtet und Richtlinien für offene Fragen (vor allem Stil und Module oder Releases betreffend) festlegt. Ganz bekannt ist jedem Entwickler die PEP008 ("Style Guide for Python Code"), wo Guido van Rossum beispielweise festlegt, wie Whitespace verwendet werden dürfen. Wem das Durchlesen zu schwer erscheint, kann ein kleines Werkzeug verwenden: pylint parst den Quelltext und versucht Unsauberkeiten zu finden. Wie in allen Programmiersprachen sollte einen Zeile nicht mehr als 75-80 Zeichen enthalten; pylint notiert die Zeilennummer, falls man einen Übertritt begeht. Am Ende wird der Quelltext bewertet. pylint ist allerdings sehr streng und teilweise enthält es noch Fehler, weshalb auch meine Quelltexte nicht vollständig pylint-valid sind. Für einen erfahrenen Python-Entwickler ist pylint trotzdem ein guter Wegweiser. Durch die schöne Syntax ist kein Wettbewerb wie "Obfuscated Perl Contest" bei Perl möglich.

3.1.2 Datentypen

Python geht dynamisch mit Datentypen um. Das bedeutet, dass es keine Typendeklaration wie *int i = 10;* bei Python gibt. *int* wäre dann nämlich auch wieder ein Schlüsselwort mehr. Python verfügt über eine automatische Speicherbereinigung (garbage collection). Bei einer Variablendefinition wie *a = 1* deutet der Buchstabe *a* auf eine Speicheradresse mit dem Inhalt *1*. Die Speicheradresse kann auch mit dem Kommando *id()* abgerufen werden. Wird *a* nach diesem Kommando wieder umdefiniert (*a = 2*), so wird der Zeiger einfach auf eine andere Speicheradresse umgebogen. Der Python Interpreter selbst bereinigt den Speicher ständig und ein Element (auf das kein Zeiger deutet) wird gelöscht. Die Allokation eines einer Variable mit demjenigen Wert findet also nicht statt (der Wert wird im Hauptspeicher nicht direkt an die Variable gebunden). Die Hauptspeicher-Adresse eines Wertes kann mit dem Befehl *id()* abgerufen werden.

Python besitzt verschiedene Datentypen. Integer, Float, Strings sind grundlegende Datentypen, aber durch die saubere OOP ist es auch möglich Klassen wie *int* zu vererben. Dann kann man eigene Datentypen entwerfen. Ein Beispiel ist cmath, welches die Verwendung von Komplexen Zahlen in Python erlaubt. Durch die Speicherverwaltung werden Objekte vom Typ String bei einer Modifikation nicht angepasst, sondern werden komplett neu definiert.

Die drei wichtigsten Containertypen sind Dictionaries (dict), Tuples (tuple) und Listen (lists). Mengen (set) ordnen ihre Elemente selbstständig und können Operationen wie Vereinigung (union) und Differenze (difference) ausführen. Listen sind Container, die veränderbar sind (mutable). Tuples besitzen das selbe Verhalten wie Listen, sind jedoch unveränderbar und damit speicheroptimierter (immutable). Dictionaries erlauben eine Zuordnung von Schlüsseln zu Werten. Alle Containerelemente schreiben keine Vorschriften bzw. der Datentypen vor, da es sich bei allem in python um Objekte handelt. Wie implementiert man Stacks? Für python-Interpreter von stackbasierenden Sprachen (zB beatnik) verwendet man Listen.

3.1.3 Einrücken, Iteration und lambda

Sprachen wie C verwenden Sonderzeichen, um Codeblöcke zusammenzufassen (zB geschwungene Klammer {} in C). python setzt hier bloß auf Einrückung. Dies sieht in etwa so aus:

```
while True:
    send_server_answer()
```

Wichtig ist bei der Einrückung die einheitliche Verwendung. Es sind wohl Tabs als auch mehrfache Leerzeichen möglich. Die Community empfiehlt es den Editor so zu konfigurieren, dass ein Tabzeichen durch 4 Leerzeichen ersetzt wird. Im vim kann man sich folgende Befehle setzen (in /.vimrc):

```
" python support
autocmd FileType python setlocal expandtab shiftwidth=4 tabstop=8 softtabstop=
let python_highlight_all = 1
let python_highlight_exceptions = 0
let python_highlight_builtins = 0
```

Iteration nennt man das Durchwandern von Containerelementen (schrittweiser Zugriff auf eine Folge). Bei Dictionaries wird das Schlüsselelement als Iterator geladen und sonst der Inhalt der aktuellen Variable. Wer gerne auch die Indizes von Tuples oder Listen bereit gestellt haben möchte, sollte enumerate() ausprobieren

```
for (index, value) in enumerate(liste):
    print "liste[%d] = '%s'" % (index, str(value))
```

Der Lambda-Kalkül ist eine formale Sprache zum Auswerten und Nutzen von Parametern. Python-Anfänger haben meist Probleme damit umzugehen, aber programmiererfahrene Geeks können es sich anonyme Funktion (zumindest ist es so in python implementiert) vorstellen, die Ausdrücke aber keine Anweisungen enthählt. Anweisungen unterscheiden sich von Ausdrücken dadurch, dass sie keinen Rückgabewert definieren.

```
>>> a = lambda x: x**2
>>> a(5)
25
>>>
```

Hier wird eine Lambda-Funktion mit dem Parameter x definiert. Erreicht die Variable das call-Signal wird der übergebene Parameter quadriert und zurückgegeben.

3.1.4 Exception

Ganz wesentlicher Teil von python sind Exceptions; Ausnahmenbehandlungen wenn Fehler auftreten. Sogar SyntaxErrors können abgefangen werden. Hierfür definiert man try-Blöcke in denen der Fehler auftreten kann und tritt ein Fehler beim Abarbeiten dieses

Blocks auf, sucht er nach dem entsprechenden except-Block. Findet er keinen, führt er (falls gegeben) den finally-Block aus.

```
try:
    a = b / 0
except ZeroDivisionError:
    a = 1
```

Die gesamte Liste an Builtin-Exceptions kann unter python.org[1] eingesehen werden. Möchte man eigene Klassen definieren, so müssen sie bloß als Subklassen von Exceptions definiert werden.

3.2 vim

Beim Vim (**V**isual **IM**proved) handelt es sich um einen freien Texteditor. Vim basiert auf dem Texteditor vi und vi basiert wiederum auf ex, welcher auf ed basiert. All diese Pakete sich heute standardmäßig auf UNIX-Systemen enthalten. Ein Editor war das wichtigste Programm eines modernen Computers, weil durch ihn Programme geschrieben werden konnten. Und ed/ex/vi/Vim waren/sind die Editoren der UNIX-Systeme. Eine gute OpenSource-Alternative zum vim bildet für Entwickler der GNU Emacs, welcher eine Anbindung an die Programmiersprache Lisp enthält (GNU Emacs wird somit nur durch die Programmiersprache Emacs Lisp konfiguriert). GNU Emacs selbst ist auch in Lisp geschrieben. Folglich besteht immer ein parodischer Kampf zwischen Entwicklern. Nicht zuletzt zitiert ein Komik, wo ein Sohn den Vater fragt und dieser wegläuft: "Why are we hiding from the police, daddy?" "Because we use vi son, they use emacs." Der Emacs ist auch Teil des Geekcodes.

Doch der Vim steht dem Emacs um nichts nach. Das erste Pro-Argument für den Vim ergibt sich darin, dass der Vim auf jedem UNIX-System vorinstalliert ist. Dies ist für Systemadministratoren besonders wichtig, da sie plattformübergreifend arbeiten müssen. Der Vor- und Nachteil des Vim ist seine Bedienung. Er stammt aus Zeiten ohne X und Maus und ist deshalb ohne Maus und graphischer Oberfläche zu bedienen. Das erwies sich bei mir als großer Vorteil als ubuntu Probleme hatte, die Tasten ihren Bedeutungen zuzuordnen (hatte Apple-Tastatur neu angesteckt). Im Vim editierte ich dann die .Xmodmap. Für graphische Oberfläche benötigt man standardmäßig die Cursortasten und Kombinationen wie Strg+C. Doch der Vim reserviert hierfür lateinische Buchstaben und zB die Cursortasten (die nicht funktionierten) konnte ich durch h, j, k und l ersetzen (die funktionierten). Der Vim brachte so mein System wieder zum Laufen.

Aber wer nicht auf graphische Elemente verzichten möchte kann auf Alternativen wie GVim und KVim setzen. Letztendlich hat der vi auch andere Distributionen als nur vim hervorgebracht (zB evlis, nvi, vile). Die Auswahl von Editoren bleibt also weiterhin groß, aber wir möchten an dieser Stelle kurz seine Funktionalität beleuchten.

[1] http://docs.python.org/library/exceptions.html#bltin-exceptions

Um den Vim bedienen zu können, müssen wir 2 verschiedene Modi unterscheiden; den Befehl- und Einfügemodus. Der Befehlsmodus nimmt Befehle entgegen, um diese an ex weiterzuleiten oder ruft selbst Funktionen aus vims Kern auf. Der Einfügemodus erlaubt uns die Modifikation und Navigation im Textbereich. Im Befehlsmodus kennt man die klassischen Befehle wie (:w), Datei neu laden (:e), Editor verlassen (:q), aber Vim denkt daran, dass ein Datenverlust entstehen könnte, wenn eine Datei nicht gespeichert wird. Wer also mit Datenverlust den Vim beenden möchte, muss den Quit erzwingen (:q!). Wie wir gesehen haben, beginnen die Befehle alle mit einem Doppelpunkt. Der Doppelpunkt leitet die Befehle an ex weiter. Befehle, die nicht von ex gesteuert werden, sind zum Beispiel Suchbefehle. Mit einem einleitenden Slash und danach dem Suchwort, wird die ganze Datei nach dem Suchwort durchforstet. Mit dem Buchstaben n wird zum nächsten Suchtreffer navigiert. Stichwort Navigieren... h, j, k und l habe ich bereits angesprochen. Mit 4l kann man den Befehl kombinieren und 4 Schritte nach l (rechts) gehen. $ steht generell für das Zeilenende und die 0 für den Zeilenanfang. Mit ist es des Öfteren bei der Entwicklung von PyIM passiert, dass mir Variablennamen nicht gefallen haben und ich sie abschnittsweise ausbessern musste. Oft verwendet man den selben Variablennamen in einem Dokument in unterschiedlichen Namespaces (zB Funktionen) und hier wollte ich nur erreichen, dass der Name nur im Bereich einer Funktion ersetzt wird. Bei graphischen Editoren habe ich bisher keine Möglichkeit gefunden, doch Vim bietet hierfür das mächtige Kommando

```
:145,192s/etcpasswd/etcshadow/g
```

In den Zeilen 145 bis 192 werden /etc/passwd durch /etc/shadow global (also nicht nur einmal) ersetzt. Wollte ich die gesamte Datei durchsuchen, müsste ich den Befehl mit :%s... beginnen.

Ein anderes Feature von Vim ist die Arbeit mit Puffern. Oft möchte man Zeilen austauschen und verschieben. Bei jedem Löschkommando (zB dd für die aktuelle Zeile) wird der gelöschte Text in den Hauptpuffer geschrieben und kann jederzeit wieder mit p (put) eingefügt werden. Wer mehrere Puffer braucht kann sie mit verschiedenen Zeichen ansprechen. Zum Beispiel "2p lädt die Daten aus Puffer 2 und fügt sie dann ein (put).

Wenn man Escape drückt landet man immer im Befehlsmodus. In den Eingabemodus gelangt man durch Drücken von i (insert), aber ebenso landet Vim automatisch im Einfügemodus, wenn ein Kommando getätigt wir, wo man meist mit dem Insert-Mode fortfahren möchte (zB o für newline).

Hat man einen Fehler gemacht, so kann ihn mit u (undo) rückgängig machen. Eine Fehlerkorrektur behebt man mit :redo und wer einen Befehl wiederholen möchte, kann dies mit einem Punkt erreichen. Wer zwischendurch auf der Konsole was erledigen muss, schreibt kurz :! in die Befehlszeile und landet dann temporär auf der Konsole.

Natürlich sind das alles recht nette Bedienungsmechanismen, aber was ist jetzt für Systemadministratoren wichtig? Syntaxvorhebung existiert für mehr als 500 Sprachen (python natürlich standardmäßig) und Vim kann Kommandos, Befehle bzw. Stichwörter

einer Programmiersprache vervollständigen. Welche Syntax Vim erkennt ist mit :set einsehbar und mit *:set syntax* konfigurierbar. Wer regelmäßige Arbeit automatisieren möchte, kann ein Vim-Makro programmieren und dabei eine der Anbindungen an Python, Perl, Ruby oder Tcl nutzen. Der Vim ist durch seine Modi und seine Befehle etwas gewöhnungsbedürftig und ist etwas für richtige Puristen, doch seine Bedienung resultiert in einer höheren Arbeitsgeschwindigkeit (zur Maus zu greifen bedeutet immer ein Zeitverlust).

Wer zum ersten Mal mit dem Vim arbeiten möchte, kann den Vim starten und F1 drücken oder in der Konsole vimtutor starten. Beides sind Hilfen für den Umgang mit dem Vim. Doch aufgrund der großen Community findet man auch viele beantwortete Fragen im Internet.

3.3 pylint

Bei pylint handelt es sich um einen kleinen Python Syntax Checker. Er zählt beispielweise die Länge einer Zeile und ab 80 Zeichen wirft pylint einen Fehler mit der Fehlernummer

3.4 mercurial

Bei mercurial handelt es sich um ein SCM (Source Code Management), welches plattformunabhängig funktioniert und in Python programmiert wurde. Der Name Mercurial ist angelehnt an die englische Bezeichnung von Quecksilber (Mercury), welches dem Atomsymbol hg zugeordnet ist. hg ist der verwendete Name auf der Konsole. Mercurial ist zur gleichen Zeit wie Git entstanden (2005), welches der große Konkurrent von Mercurial ist und von Linus Torvalds konzipiert wurde. Unter Linux-Entwickler hat sich Git durchgesetzt, aber unter anderen OpenSource-Entwicklern hat sicht mercurial genauso etabliert.

Die grundlegende Aufgabenstellung von mercurial ist die Verwaltung von Quelltexten in einem Großprojekt, wo die einzelnen Entwickler Patches (kleine Aus-/Verbesserungen von Software) zu Software liefern. Es ist also das Bestreben Entwicklungszweige zu erstellen (branch) und zusammenzuführen (merge). Es beginnt damit, dass man sich einen Klon von einer Software holt.

```
hg clone http://bitbucket.org/meisterluk/pyim/
```

Dadurch wird im lokalen Verzeichnis ein Ordner pyim angelegt, wo die aktuelle Version der Software verfügbar ist. Versteckt ist ein Ordner .hg, der erzeugt wurde und sämtliche Informationen über die Dateien enthält. Wird eine Datei verändert, passiert noch gar nichts. Führt man jedoch *hg commit* aus, so untersucht mercurial alle Modifikationszeiten und überprüft die Dateien auf ihre neueste Version. Werden Modifikationen

festgestellt, so werden die Unterschiede aufgezeichnet und als "Changeset" festgehalten. Beim Ausführen von *hg commit* öffnet sich der Editor $EDITOR und es kann eine Nachricht hinterlassen werden, die die Modifikationen beschreiben. Mit *hg status* kann überprüft werden, welche Dateien mercurial verfolgt, weil manche mit der Datei .hgignore ausgeschlossen werden können. *hg log* liefert eine Übersicht über alle Changesets und *hg tip* gibt nur den letzten Changeset aus.

Wer ein Projekt von ganz Anfang an starten möchte, kann *hg init* ausführen, wodurch jenes Verzeichnis .hg erzeugt wird. Wird eine neue Datei im Projektverzeichnis angelegt, so kann man dies mercurial mit dem Befehl *hg add FILENAME* mitzuteilen. Mercurial "trackt" daraufhin diese Datei und notiert ihre Modifikationen. Vorsicht, beim Befehl *hg rm FILENAME*, weil die Datei nicht nur aus dem repository gelöscht wird, sondern auch wirklich im Projektordner. Hat man einen Changeset erfolgreich abgeschlossen, kann man ihn über SSH oder FTP auf einen Server hochladen, welcher die Veränderungen auf einer Webseite präsentiert und auch das Klonieren des Projekts für andere User wieder erlaubt.

Was ist ein Repository (kurz repo)? Im Prinzip handelt es sich um ein Projektverzeichnis mit allen Dateien, jedoch arbeiten alle SCMs speicheroptimiert und so geht man bloß von einer ursprünglichen Version aus und beschreibt dann alle Veränderungen. Benötigt jemand wirklich einmal die aktuelle Version des Projekts (wie zB für clone), so wird das Projekt errechnet und das Paket meist als tar.gz (oder eben also repo wie bei clone) angeboten.

Mercurial wird beispielweise von Gajim (freier Instant Messenger), QuakeNet IRC Networks (IRC-Server), bitbucket (die Plattform selbst), Mozilla (sämtliche Software wie zB der Browser Firefox), OpenSolaris (UNIX-System), OpenLibrary (sammelt Dokumente aus aller Welt), Pocoo.org (Websoftware).

Meine Skripte waren die meiste Entwicklungszeit im Internet verfügbar. Dabei lud ich die changesets immer auf einen mercurial-Server; bitbucket.org hat sich unter python-Entwicklern durchgesetzt. Im März 2009 gab die Mailingliste von python.org bekannt auch auf hg umzusteigen.

3.5 sockets

Der Verbindungsaufbau des PyIMs erfolgt über Sockets.

Ein Socket ist der einfachste Kommunikationsweg (bidirektional), um zwischen zwei UNIX-Systemen kommunizieren zu können. Es handelt sich um eine API (Application Programming Interface), wodurch Netzwerk- und Interprozesskommunikation ermöglicht wird. Ein sichtbares Beispiel für jeden Computerbenutzer ist ein Webbrowser, der beim Aufrufen einer Webseite eine Verbindung herstellen muss.

Sockets schließen an die "Alles ist eine Datei"-Philosophie von UNIX an. Das bedeutet sämtliche (virtuell oder nicht) Laufwerke, Verbindungen und Kernel-Daten sind

über die hierarchische Baumstruktur zugänglich. Sie lassen sich Schreiben, Lesen und Schließen. Mitgliefert (von den Paketen in Programmiersprachen) werden auch Netzwerkfunktionen (zB gethostbyname) und Funktionen zur Konfiguration der Verbindung (zB set_blocking).

Sockets sind ein Standard, der auch von Microsoft Produkten eingehalten bzw. unterstützt wird.

3.5.1 SOCK_STREAM und SOCK_DGRAM

Man kann zwei Varianten der Kommunikation unterscheiden: den Stream Socket (Daten werden als Strom übertragen) und Datagram Socket (Daten werden als kleine Pakete versandt). Stream Socket arbeitet mit dem verbindungsorientierten TCP, während Datagram Sockets UDP verwenden (die Pakete werden einfach rausgesandt, ohne Garantie der Ankunft).

3.5.2 Server vs. Client

Server und Client haben unterschiedliche Aufgaben. Ein Client baut eine Verbindung auf und sendet Daten. Er wartet auf die Antwort und beendet die Verbindung wieder. Ein Server hingegen muss auf Anfragen von Clients warten (also ständig laufen). Dies wird durch eine Endlosschleife erreicht.

3.5.3 Blocking and non-blocking

3.6 JSON

JSON ist das voreingestellte Austauschformat für eine Kommunikation mit PyIM.

JSON ist ein Datenaustauschformat, welches ursprünglich für das Web entwickelt wurde, wobei die Syntax auf Javascript basiert. Die Idee war es ein Format zu kreiieren, welches gut zu parsen ist und sich zum Tauschen von Daten eignet. In der Informatik muss man hierbei zwischen verschiedenen Datentypen unterscheiden. Zahlen werden als Integer oder float gespeichert. Texte werden als Strings bezeichnet.

Hier soll ein kleiner Überblick über die JSON-Datentypen gegeben werden.

```
{
  'JSON' : "1.2\n",
       0 : -0.1239875e+5,
 'array' : [1, 2, 3, 4],
  'data' : true
}
```

Mit JSON wird zuerst einmal ein Objekt übertragen. Ein Objekt ist ein Datentyp, der durch geschwungene Klammern umfasst wird. Die einzelnen Elemente des Objekts werden durch ein Komma getrennt. Die Elemente erzeugen jeweils eine Zuordnung von Schlüssel zu Wert. Das erste Element ordnet den Schlüssel *JSON* dem Wert *1.2\n* zu. Bei *JSON* handelt es sich um einen String, der unter einfachen Anführungszeichen steht. Die Anführungszeichen indizieren, dass es sich um einen ungeparsten String handelt; der Inhalt wird so behandelt, wie er eingetragen ist. Das Gegenteil bildet der String mit doppelten Anführungszeichen (geparst). Das $\backslash n$ im String wird als ein newline interpretiert und bei der Ausgabe dieses Strings wird ein Zeilenumbruch erzeugt. Das nächste Element besteht aus zwei Zahlen. Auf der linken Seite steht für den Schlüssel ein Integer; eine natürliche Zahl. Auf der anderen Seite haben wir ein Flusskommadarstellung. Der Wert ist negativ und mathematisch würde man ihn als $-0.1239875 \cdot 10^5$ notieren. Der nächste Wert zeigt ein Array; ein Container von Werte. Das Array umfasst die Werte mit eckigen Klammern. Der letzte Wert zeigt einen bool'schen Wert. Boolsche Werte umfassen nur zwei Zustände (*true* oder *false*). Nicht in diesem Beispiel enthalten: der Datentyp *null*, der quasi für undefiniert steht.

Parser für JSON sind in nahezu allen verbreiteten Sprachen verfügbar. Als Beispiel sei genannt: C, ActionScript, C#, C++, Delphi, Haskell, Java, Lisp/Scheme, Perl, Objective-C, PHP, Python, Prolog und Ruby.

Die Syntax von JSON ist weitgehend mit der von python kompatibel. Einen schönen visuellen Überblick über die JSON Syntax bietet die Webseite json.org[2]

3.7 SVG

Die PyIM Logos und Wallpapers wurden alle ursprünglich als SVG erzeugt. Verwendet wurde dafür die Software Inkscape[3].

Bei SVG handelt es sich um ein XML-Format, welches schon bald ins Web Einzug erhalten soll. Das Problem welches sich für Webentwickler stellt ist die Vergrößerung bzw. Anpassung von Graphiken an Bildschirmbereite bzw. Geräte. Werden Graphiken vergrößert erscheinen sie als pixelig; besonders bei starker JPEG-Komprimierung. JPEG hat sich als Standardformat im Web etabliert.

Als Reaktion auf diese Entwicklung entstand die Nachfrage nach einem Format, in dem Flächen als Objekte zusammengefasst werden, die Größen dynamisch anpassfähig sind und der Renderer somit in Echtzeit die Graphik neu berechnen kann und sie nicht aus dem alten Bild anpassen muss. Dies war jetzt eine Umschreibung für Vektorgraphiken und daher rührt auch der Name SVG ("Scalable Vector Graphics").

Im Gegensatz zu PNG, JPEG oder GIF Dateien handelt es sich um ein Textformat; der Quelltext wird also mit Sonderzeichen, Zahlen und Buchstaben geschrieben. In Zukunft

[2] http://json.org/
[3] http://inkscape.org/

soll es so möglich sein SVG Graphiken direkt in HTML Seiten einzubetten oder wie bisher als eine externe Ressource beziehen. Die Syntax basiert (wie erwähnt) auf XML. Im Folgenden sei ein kleines Beispiel genannt. Der Quelltext beschreibt die Augen des ACID2-Smileys[4].

```
<!-- eyes -->
<circle cx="80" cy="60" r="10"
style="fill:white; stroke:black; stroke-width:1" />
<circle cx="80" cy="60" fill="green" />
<path d="M67,50 C62,55 62,65 67,70"
style="fill:none; stroke:#A82; stroke-width:5" />
<circle cx="120" cy="60" r="10"
style="fill:white; stroke:black; stroke-width:1" />
<circle cx="120" cy="60" fill="green" />
<path d="M132,50 C138,55 138,65 132,70"
style="fill:none; stroke:#A82; stroke-width:5" />
```

Es war eines meiner Projekte den ACID2-Smiley ins SVG-Format zu übertragen[5]. In der ersten Zeile befindet sich ein Kommentar, welches durch *<!* (Einleitung für einen XML-Befehl) und -- (klassisches Zeichen für ein Kommentar / einen Anhang) begonnen und durch --> beendet wird. Die XML-Syntax ist grundlegend wie folgt aufgebaut:

```
<tagname eigenschaften="wert">inhalt</tagname>
```

Sehr markant ist die Eigenschaft *style*, die bereits aus CSS bekannt ist. Tatsächlich enthält der SVG-Standard sogar die Klassen- und Identitätskonzepte von CSS.

.1 Autor

Das Projekt "PyIM" wurde ursprünglich für meine Matura am BRG Viktring initiiert. Ich wählte es, weil ich so meine (fehlenden) Kenntnisse im Bereich OOP, Netzwerkaufgaben und python-libraries erweitern konnte. Leider war alles etwas zeitlich knapp, aber ich konnte einige Erfahrungen sammeln. Angefangen bei mercurial (SCM). Ich denke es ist eine gute Plattform, damit ich selbst noch einiges in Zukunft testen kann und wer weiß... vielleicht habe ich einmal viel Zeit frei und dann wird sicher daran weitergearbeitet.

[4] http://webstandards.org/action/acid2
[5] http://lukas-prokop.at/blog/2008/03/svg-acid2-smiley/

.2 Dank

Prof. Schmidhofer (Informatik) vom BRG Viktring
written with LaTeX 2$_\varepsilon$

Literaturverzeichnis

[1] de.wikipedia.org als Hauptquelle