Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Informatikdienste
Software Services

ETH Zürich
Benno Luthiger
STC E 13
Stampfenbachstrasse 67
8092 Zürich

Telefon +41 44 632 57 65
benno.luthiger@id.ethz.ch
www.foss.ethz.ch

Software Services (SWS)
Informatikdienste
ETH Zürich

Zürich, 30. August 2014

EuroPython 2014, Konferenz-Bericht

Die EuroPython 2014 fand in Berlin. An den fünf Tagen der Hauptkonferenz wurden in mehreren parallelen Tracks mehr als 100 Vorträge, Referate und Präsentationen angeboten.

Die in der folgenden Liste aufgeführten EuroPython-Präsentationen haben eine gewisse Relevanz für die Software-Entwicklung an der ETH Zürich:

Constanze Kurz: *One year of Snowden, what's next?*

C. Kurz ist Sprecherin des Chaos Computer Clubs Deutschland. Gemäss ihrer Einschätzung richten sich die Eingriffe des NSA in erster Linie gegen ökonomische Ziele. Die Eingriffe sind in diesem Bereich auch erfolgreicher als auf dem Gebiet der Terrorismus-Bekämpfung.
Mit der Aufdeckung von Snowden konnte der Umfang der NSA-Eingriffe erstmals dokumentiert werden. Dies führte zu einem starken Vertrauensverlust in amerikanische Firmen. Gemäss Kurz kann dies als Hebelarm gegen weitere NSA-Eingriffe gebraucht werden. Wenn die europäischen Firmen auf Grund von Sicherheitsüberlegungen abwandern, werden amerikanische Firmen Druck gegen NSA-Eingriffe machen.

Emily Bache: *Will I still be able to get a job in 2024 if I don't do TDD?*

Ein wesentliches Merkmal von Test Driven Development (TDD) ist, dass das API vor der Implementierung definiert wird. TDD bringt folgende Vorteile: schnelle Rückmeldung bzgl. Design-Entscheidungen, Selbst-Testender Code (als Voraussetzung für Refactoring), Gefühl der Freiheit und Produktivität. Damit TDD sich breit durchsetzt, muss es noch praktischer werden, ohne dass dabei diese Vorteile von TDD verloren gehen.

Weiterentwicklung von TDD: Double Loop TDD (vgl. Growing Object-Oriented Software, Guided by Tests (Freeman, S. and Pryce, N.), 2009):
Erster Zyklus: 1) einen Unit-Test schreiben, der fehlschlägt, 2) Code ändern, bis Test durchläuft, 3) Refactoring
Zweiter Zyklus: 1) einen Akzeptanz-Test schreiben, der fehlschlägt, 2) einen Unit-Test schreiben, der fehlschlägt, 3) Code ändern, bis Tests durchlaufen

Unter folgenden Umständen ist es schwierig, TDD durchzuführen: Unklare Anforderungen (es sind keine Akzeptanz-Tests möglich), Legacy-Code (ohne Unit-Tests). Unter solchen Bedingungen gibt es Alternativen zu TDD: Approval Testing (nur Outputs werden verglichen, ist geeignet für Legacy-Code), Spike&Stabilize.

TDD und gutes Programmieren allgemein kann geübt werden. Gut geeignet sind Code Katas (d.h. kleine, abgeschlossene Programmier-Übungen, die mehrmals hintereinander gelöst werden) und Coding Dojos. In einem Coding-Dojo finden mehrere Entwickler zusammen, um eine Code-Kata gemeinsam durchzuführen und so voneinander zu lernen (siehe http://cyber-dojo.org/ oder http://blog.mayflower.de/3970-So-wird-ein-Coding-Dojo-zu-einer-ernstzunehmenden-Lernveranstaltung.html)

Hynek Schlawack: *The Sorry State of SSL*

Alle Informationen auf https://hynek.me/talks/tls/

Robert Lehmann: *Teaching Python*

Wie soll Python gelernt werden? Referenzen sind keine Anleitungen (Tutorien)
Referenz: korrekt, komplett, idiomatisch
Anleitung: verständlich, kohärent, idiotisch

Python-turtle ist gut geeignet, um Python zu lernen (nicht nur für Kinder). Danach sollte Python an real-weltlichen-Problemen erlernt werden.

Maurizio Boscaini: *VPython goes to School*

VPython ist eine Bibliothek für 3D-Programmierung. Mit VPython können mit wenig Code mathematische und physikalische Abläufe visualisiert werden. Das macht VPython interessant für einen Einsatz an Mittel- und Hochschulen.

Pieter Hintjens: *Our decentralized future*

Pieter Hintjens ist Entwickler von ZeroMQ (0MQ), einer hochperformanten, asynchronen Messaging-Bibliothek für skalierbare, verteilte und nebenläufige Applikationen.
Hintjens hat seine Arbeit im Entwicklerteam von ZeroMQ als positives Beispiel erlebt, wie produktiv eine dezentralisierte Organisation ohne zentrales Planungskomitee arbeiten kann.
Der Vorteil solcher anarchischen Systeme ist die Vielfalt, welche damit gewonnen wird. Allerdings brauchen solche Systeme starke Autoritäten, um die Prozesse zu schützen.

Philip Brechler: *Don't fear our new robot overlords!*

Wie können die visuellen Aspekte von Smartphone-Applikationen getestet werden?
Der Präsentator zeigt, wie mit relativ wenig Aufwand ein Roboter zusammengestellt und ausgerüstet werden kann, welcher an Stelle einer Testperson die visuellen Aspekte einer mobilen Applikation testen kann.

Konzept: Tapsterbot (http://www.thingiverse.com/thing:123420), Sikuli (http://www.sikuli.org/)
Vorgehen: Bildschirmfoto erstellen, mit OpenCV Bild analysieren und Schalfläche finden, Schalfläche von Roboter anklicken.

Schlomo Schapiro: *DevOps Risk Mitigation: Test Driven Infrastructure*

Code-Qualität ist nicht nur für Entwickler und deren Applikationen wichtig, sondern auch für System-Operatoren und deren Skripte. Der Prozess der Software-Installation, welcher in der Verantwortung der System-Administratoren ist, sollte möglichst automatisiert werden. Dies erhöht die Produktivität der System-Operatoren. Die entsprechenden Skripte müssen allerdings getestet sein, damit die Prozesse stabil sind.
Notwendig sind sowohl Unit- wie auch System-Tests.
Für System-Tests müssen die richten Komponenten gemockt werden. Linux bietet viele Möglichkeiten für Mocking an.
(vgl. http://www.slideshare.net/schlomo/europython-2014-devops-risk-mitigation)

Mark Smith: *Writing Awesome Command-Line Programs in Python*

Der Vorteil von Kommandozeilen-Programmen ist, dass diese ohne Fenster, Widgets und visuellen Benützerschnittstellen (UI) auskommen. Kommandozeilen-Programme interagieren mit: Argumenten, Konfiguration, stdin/stdout/stderr, Signalen und Exit-Code.
Kommandozeilen-Programme könnten speziell für System-Administratoren interessant sein.

Hilfsmittel für Kommandozeilen-Programme:
Argumente parsen mit *argparse*.
Unterschied zwischen stdout und stderr: stdout kann gepipt werden, stderr kann geloggt werden.
Die einfachste Lösung zur Konfiguration von Kommandozeilen-Programmen sind INI-Files. Diese können mit ConfigParser.SafeConfigParser gelesen und interpretiert werden.
Signale an das Kommandozeilen-Programm können mit der Programmbibliothek *signal* verarbeitet werden.
Exit-Code: Standard-Verhalten ist 0 für fehlerfreies Programmende, 1 für ein Ende nach einer Exception. Mit *sys.exit(exit_code)* kann ein bestimmter Wert gesetzt werden.
Für das Paketieren ist *setuptools* besser geeignet als *distutils*.
(Beispiele auf https://github.com/judy2k/command-line-talk)

Travis Oliphant: *Python's Role in Big Data Analytics: Past, Present, and Future*

Python hat sich schon seit langer Zeit bewährt im Gebiet von Scientific Computing. Auf dieser Basis wird Python in der Forschung immer mehr in Kombination von Big Data und High Performance Computing verwendet. Die Python-Bibliotheken, welche diesen Erfolg möglich machten, sind NumPy bzw. der SciPy-Stack. NumPy ist eine Array-orientierte Sprache. Dies macht es möglich, die Verarbeitungsschritte zu parallelisieren, was wesentlich zur Leistungsstärke von NumPy beiträgt.
Aus der Finanzwirtschaft wurde die Pandas-Bibliothek (zur Analyse von Zeitreihen etc.) beigesteuert. Damit vergrösserte sich der Benützerkreis von Python weit über den naturwissenschaftlichen Rahmen hinaus.
Inzwischen wird deutlich, dass Python die Programmiersprache R als Werkzeug für Data-Science ablösen wird.

Welche Probleme gibt es beim Einsatz von Python in Big Data?
CPython ist veraltet, Array-orientiertes Programmieren wird nicht richtig verstanden.

Welche Probleme gibt es mit NumPy?
NumPy kommt einer In-Memory-Datenbank (ohne Index) nahe, es enthält etliche nicht optimierte Teile, das Dtype-System ist schwierig zu erweitern.

Die Zukunft von Python-Data wird heterogen sein.
Es gibt interessante neue Projekte (Biggus, DistArray, SciDB, Elemental, Spartan).
Interessante Integrationen von Python in andere Projekte: Spark, Impala, Disco.
Alternativen zu NumPy/SciPy in Python: Blaze, Numba, Conda.

Schlussbemerkungen / Anregungen:

- Die Kompetenz zu Python Big Data soll in den ID gepflegt werden. Diese Kompetenz sollte sowohl Beratung wie auch Schulung beinhalten.
- Innerhalb der Software-Entwicklung sollten neue Formen der Wissens- und Erfahrungs-Vermittlung wie Code Katas und Coding Dojos ausprobiert werden.
- Die System-Administratoren (in den BD und den ISGs) sollten auf die Bedeutung, welche der Code-Qualität ihrer Skripte zukommt, hingewiesen werden. Auch sollte das Wissen, wie gute Kommandozeilen-Programme aussehen, in diesem Personenkreis verbreitet werden.

Luthiger Benno (ID SWS)