DEUTSCHE BÖRSE
GROUP

# Python in the Financial Industry
# The universal tool for end-to-end development

Keynote - PyData London 22.02.2014

# Abstract

## Python in the financial industry: The universal tool for end-to-end development.

- In the context of a rapidly evolving financial industry, managing increasing amounts of data and coping with regulatory requirements, time-to market of services and cost efficiency along the value chain are key success drivers for any financial institution.

- Especially the shift from monolithic architectures (e.g. Open VMS/Cobol) to heterogeneous technology stacks and systems (e.g. Linux/Java/SQL) creates additional challenges for IT. In addition, the “technology empowerment of the business analysts” adds complexity to the implementation of IT systems if not managed properly.

- After the introduction of Python at Deutsche Börse Group several years ago, the presentation today is a reflection about experiences in real world applications, the potential of Python as a universal tool for end-to-end development and an outlook to the future of this language framework in the financial industry.

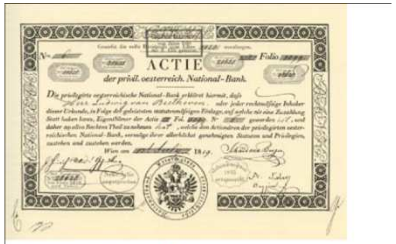
# Agenda

- Deutsche Börse Group Overview
- History of Python within DBG
- Python - State of The Union
- Potential of Python in the Software Factory
- Conclusion

DEUTSCHE BÖRSE
GROUP

# Deutsche Börse Group Overview

# Evolution of an Exchange - History of Deutsche Börse Frankfurt



№ Folio

ACTIE

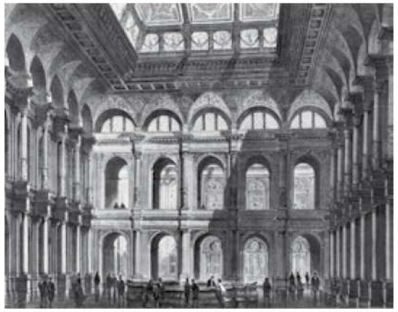
der privil. oesterreich. National-Bank.

Die privilegirte oesterreichische National-Bank erkläret hiermit, daß Herr Ludwig van Beethoven oder jeder rechtmäßige Inhaber dieser Urkunde, in Folge der geleisteten statutenmäßigen Einlage, auf welche nie eine Zurückzahlung Statt haben kann, Eigenthümer der Actie Fol. № geworden ist, und daher an allen Rechten Theil zu nehmen hat, welche den Actionären der privilegirten oesterreichischen National-Bank, vermöge ihrer allerhöchst genehmigten Statuten und Privilegien, zustehen und zustehen werden.

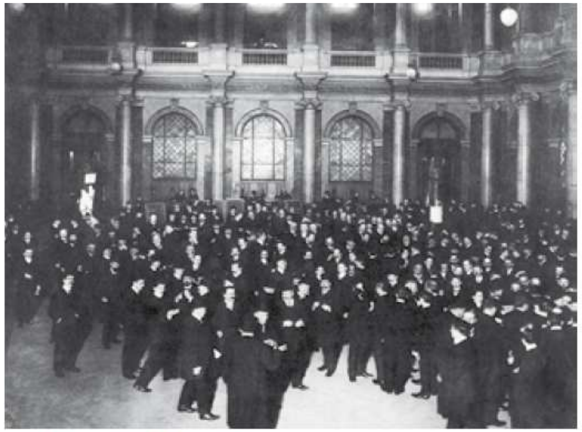
Wien am [illegible] 1819.

DIE ERSTE IN FRANKFURT GEHANDELTE AKTIE

1820



1879



BÖRSENVERSAMMLUNG UM 1914

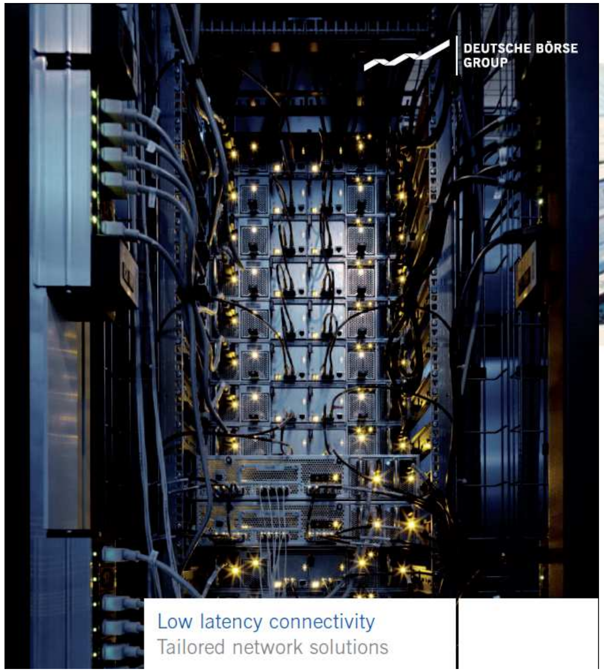
Fotos: Berliner Handels- und Frankfurter Bank KGaA, Frankfurter Wertpapierbörse

SITZUNG DER DEVISENBÖRSE MITTE DER 1960ER JAHRE

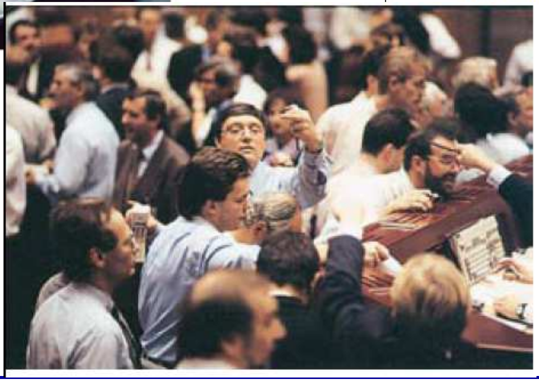
1969

1997

2014



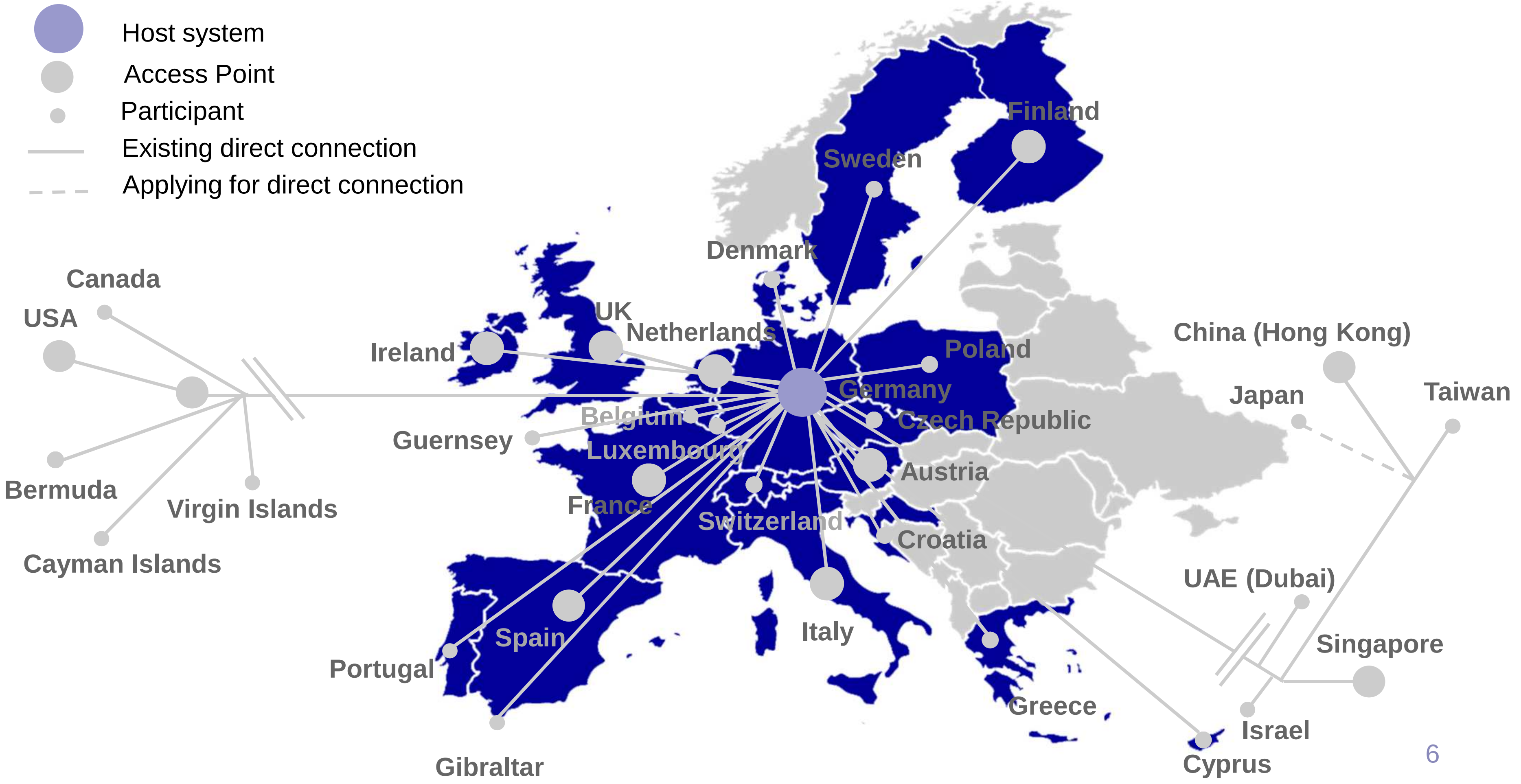
1988



# The Integrated Business Model Of Deutsche Börse Group Is Unique And Serves As The Global Role Model

| | | DEUTSCHE BÖRSE GROUP | CME Group A CME/Chicago Board of Trade Company | ice | London Stock Exchange | NASDAQ OMX | NYSE EURONEXT |
|---|---|---|---|---|---|---|---|
| **Cash** | **Eurex/ Xetra** | ● | ○ | ○ | ● | ● | ● |
| **Derivatives** | | ● | ● | ● | ◔ | ◑ | ● |
| **Clearing** | | ● | ● | ● | ◑ | ○ | ○ |
| **Settlement** | **Clearstream** | ● | ○ | ○ | ◑ | ○ | ○ |
| **Custody** | | ● | ○ | ○ | ◑ | ○ | ○ |
| **Collateral management** | | ● | ○ | ○ | ○ | ○ | ○ |
| **Market data** | **MD+S** | ● | ● | ● | ● | ● | ● |
| **Indices** | | ● | ◔ | ○ | ● | ○ | ○ |
| **Technology** | | ● | ○ | ○ | ● | ● | ● |

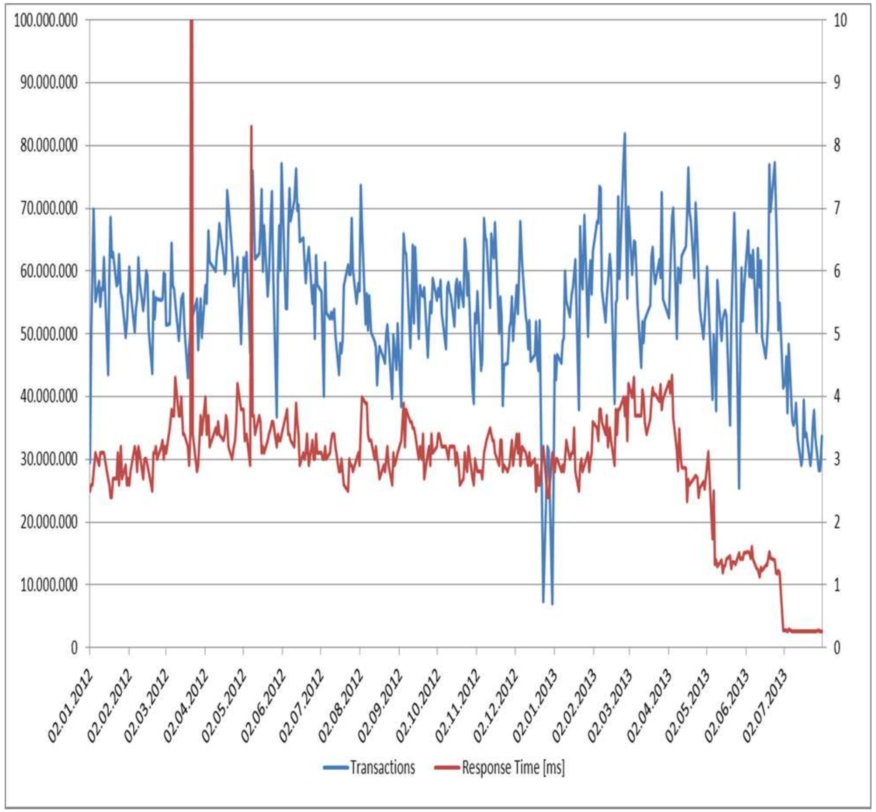
# The Trading Network Behind The Systems:
# > 420 Participants, > 8000 Traders, > 30 Markets



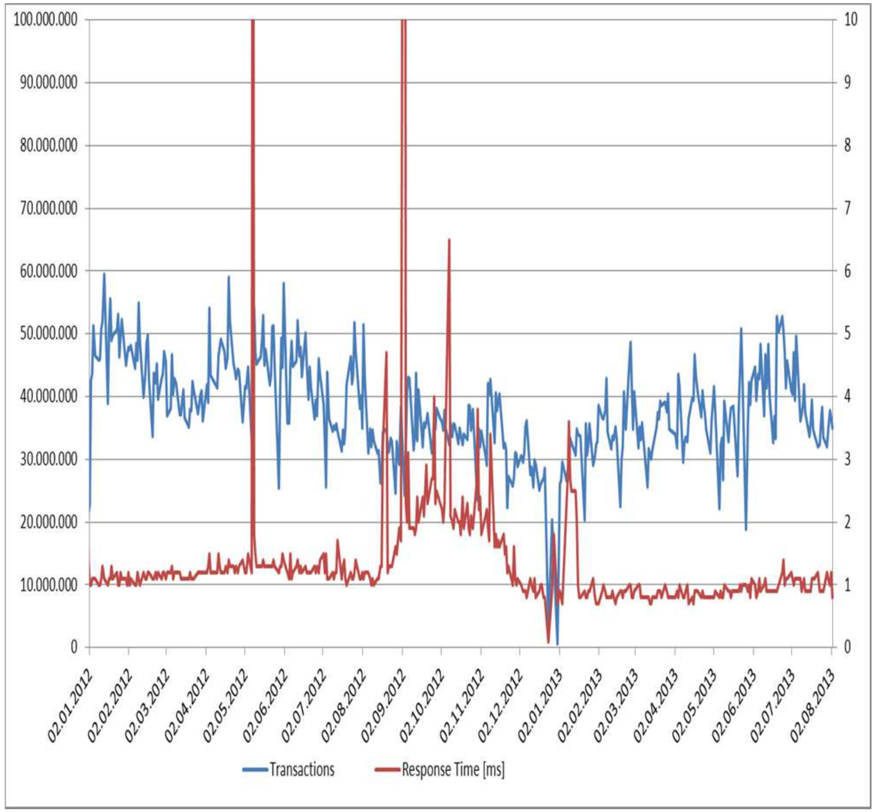
# Performance trends in 2012 and 2013

The significant drop in Eurex processing times in 2013 is due to the launch of Eurex Exchange's T7.

## Eurex®



## Xetra®

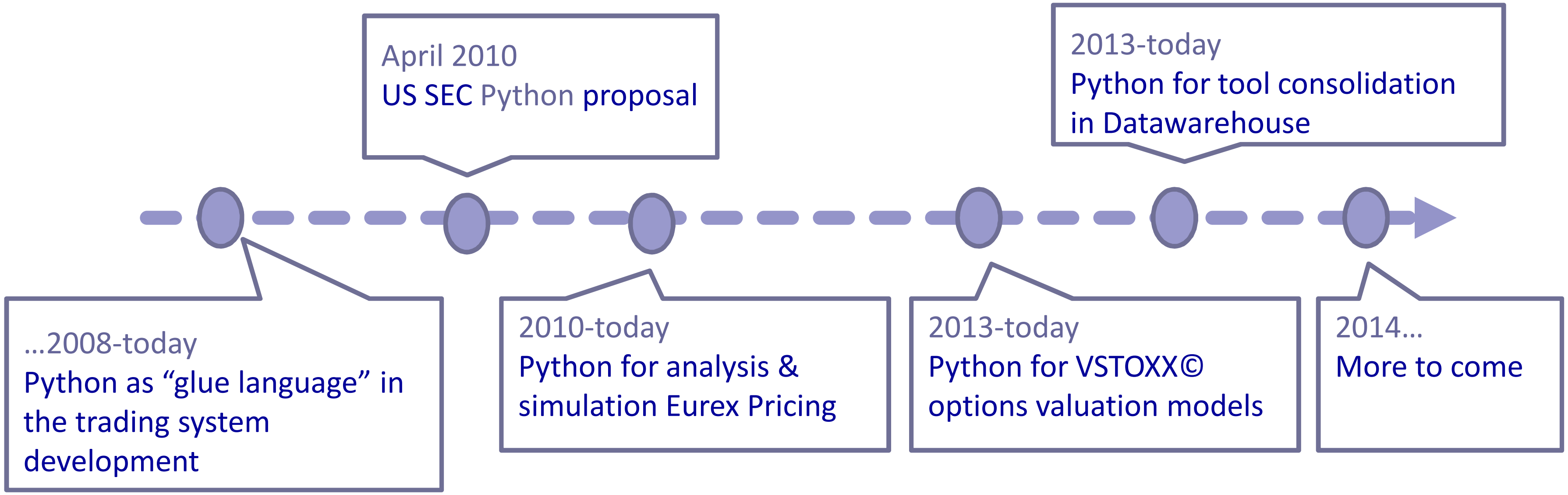


DEUTSCHE BÖRSE
GROUP

# History of Python within DBG

# History of Python within DBG

**SEC proposal for issuers of Asset Backed Securities:**
"…In addition, we are proposing to require, along with the prospectus filing, the filing of a computer program of the contractual cash flow provisions expressed as downloadable source code in **Python**, a commonly used open source interpretive programming language…"



April 2010
US SEC Python proposal

2013-today
Python for tool consolidation in Datawarehouse

…2008-today
Python as "glue language" in the trading system development

2010-today
Python for analysis & simulation Eurex Pricing

2013-today
Python for VSTOXX© options valuation models

2014…
More to come

# Open-Source in the DBG Technology Stack

The technology stack for our latest trading infrastructure for derivatives (T7) makes extensive usage of open-source technologies:

- **RedHat Enterprise Linux with MRG kernel**
- **Boost (C++ libraries)**
- IBM WLLM
- **Python (plus many modules)**
- **MySQL**
- **JBOSS**
- **Apache ActiveMQ**

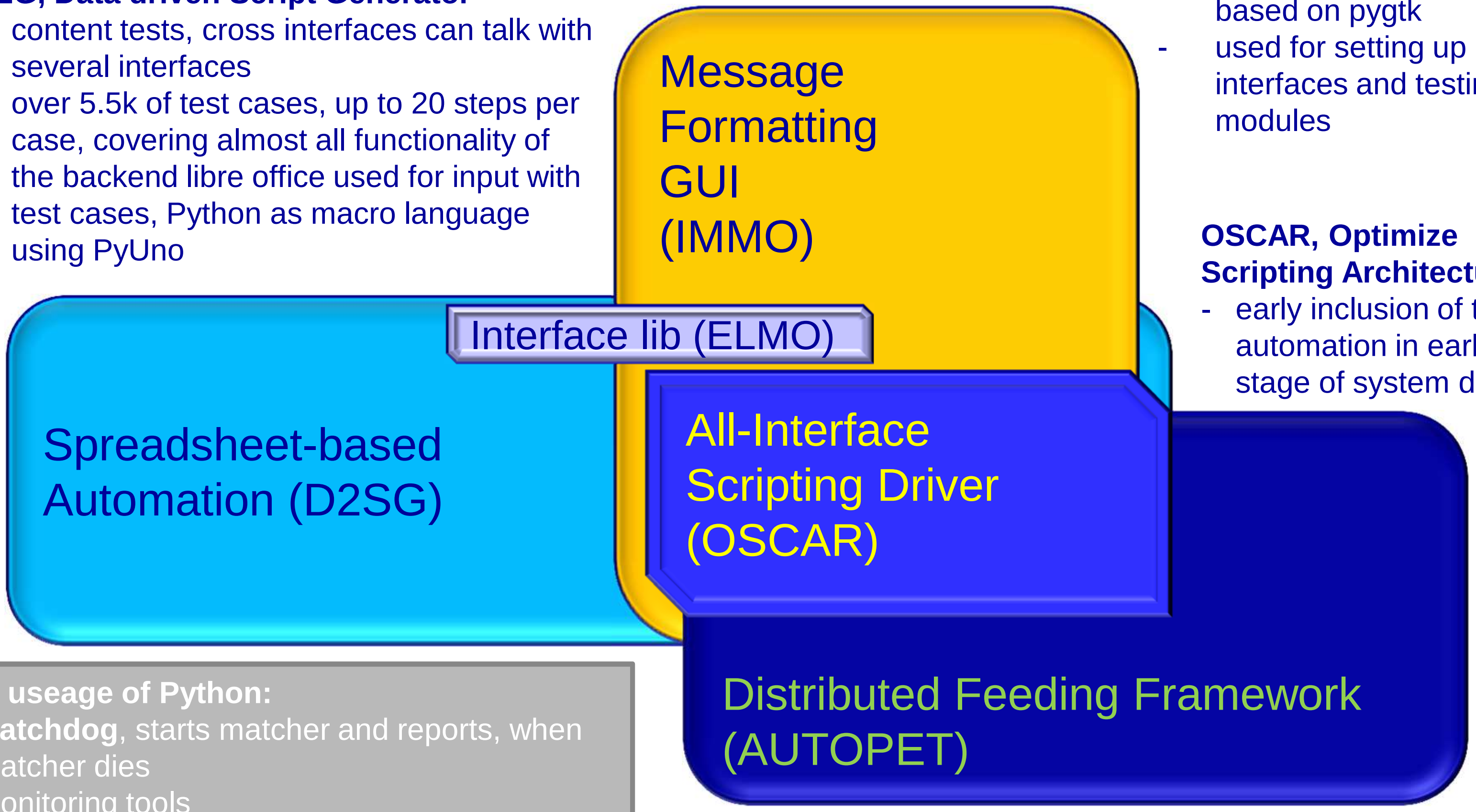
# Optimise test tool framework, heavily Python based…

**DS2G, Data driven Script Generator**

- content tests, cross interfaces can talk with several interfaces
- over 5.5k of test cases, up to 20 steps per case, covering almost all functionality of the backend libre office used for input with test cases, Python as macro language using PyUno

**IMMO**

- GUI, based on oscar, based on pygtk
- used for setting up user interfaces and testing modules



Message Formatting GUI (IMMO)

Interface lib (ELMO)

Spreadsheet-based Automation (D2SG)

All-Interface Scripting Driver (OSCAR)

Distributed Feeding Framework (AUTOPET)

**OSCAR, Optimize Scripting Architecture**

- early inclusion of test automation in early stage of system design

**Other useage of Python:**

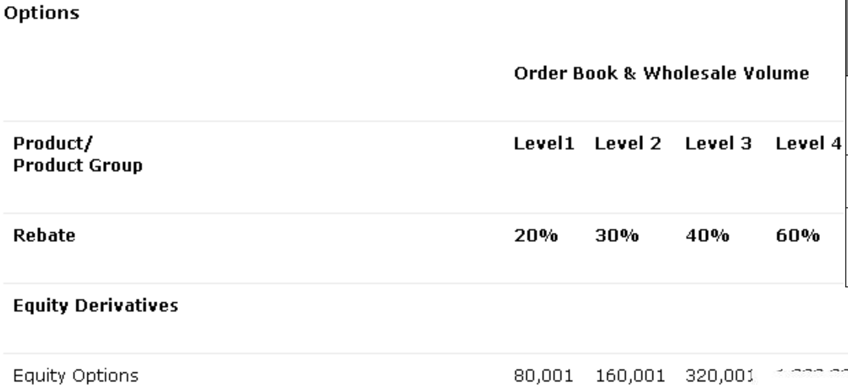
- **watchdog**, starts matcher and reports, when matcher dies
- monitoring tools
- glue language overall
- OPCON, major process console, uses mainly oscar

**AUTOPET, Automated Performance Tester**

- Dedicated Perfomance Feeder
- starting many clients that generate load cases, measuring output latency and distribution

# Eurex Pricing Engineering – Something not really suitable for a spreadsheet…

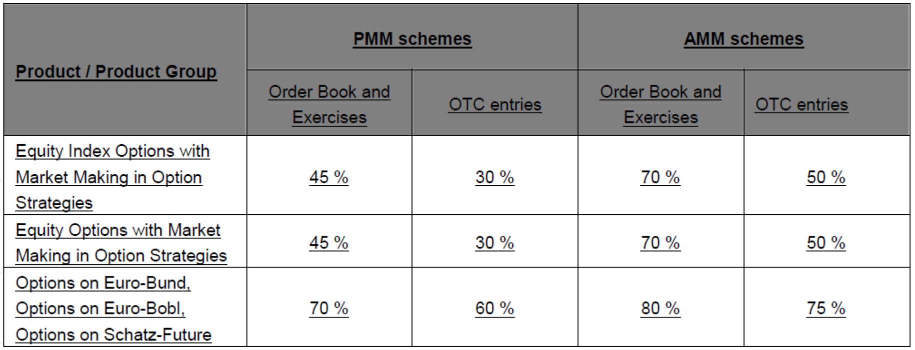
## Volume Rebates



Options

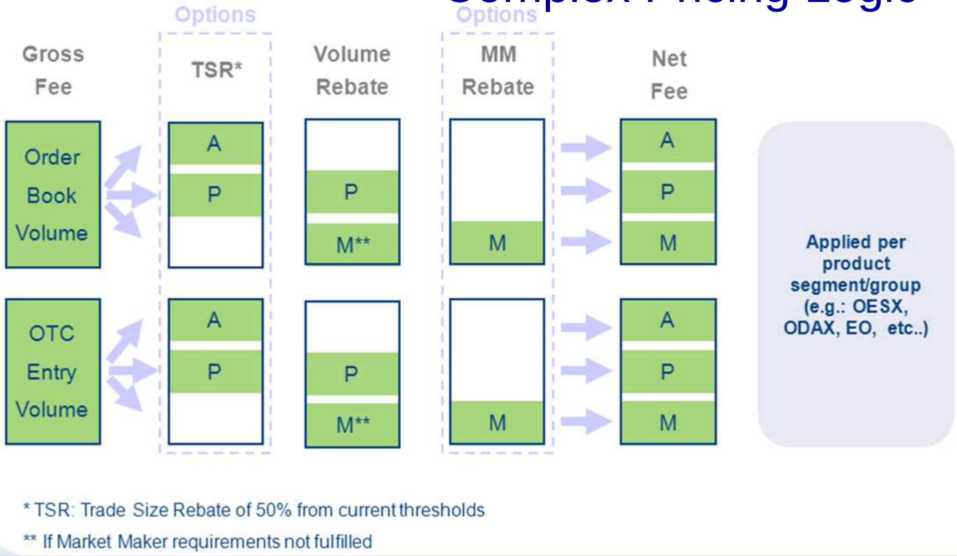
| Product/ Product Group | Order Book & Wholesale Volume<br>Level1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| Rebate | 20% | 30% | 40% | 60% |
| **Equity Derivatives** | | | | |
| Equity Options | 80,001 | 160,001 | 320,001 | [illegible] |
| **Equity Index Derivatives** | | | | |
| EURO STOXX 50® Index Options (incl. Weekly Options) | 80,001 | 160,001 | 320,001 | |

## Market Maker Rebates



| Product / Product Group | PMM schemes<br>Order Book and Exercises | PMM schemes<br>OTC entries | AMM schemes<br>Order Book and Exercises | AMM schemes<br>OTC entries |
|---|---|---|---|---|
| Equity Index Options with Market Making in Option Strategies | 45 % | 30 % | 70 % | 50 % |
| Equity Options with Market Making in Option Strategies | 45 % | 30 % | 70 % | 50 % |
| Options on Euro-Bund, Options on Euro-Bobl, Options on Schatz-Future | 70 % | 60 % | 80 % | 75 % |

## Complex Pricing Logic



* TSR: Trade Size Rebate of 50% from current thresholds

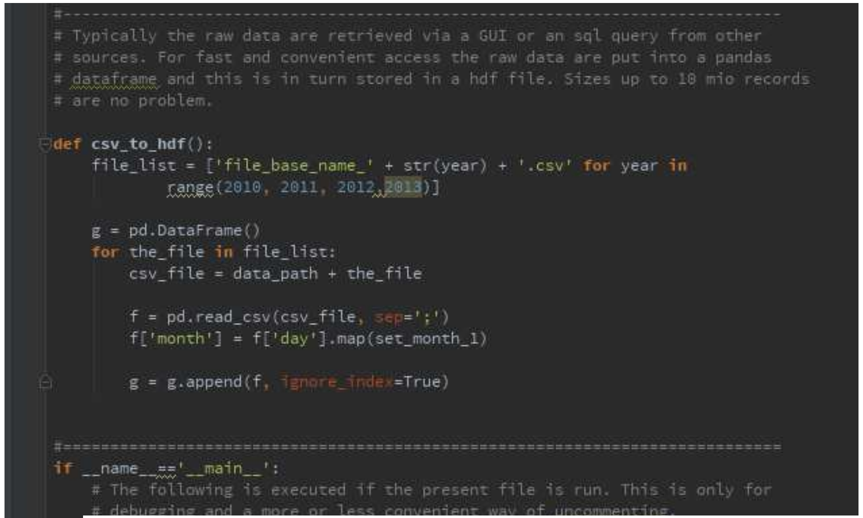
** If Market Maker requirements not fulfilled

Eurex:
- Over 2,000 Products
- Current avg. 5,6m contracts/day
- Daily MM reports/fulfilment

Xetra, Clearing, Connectivity

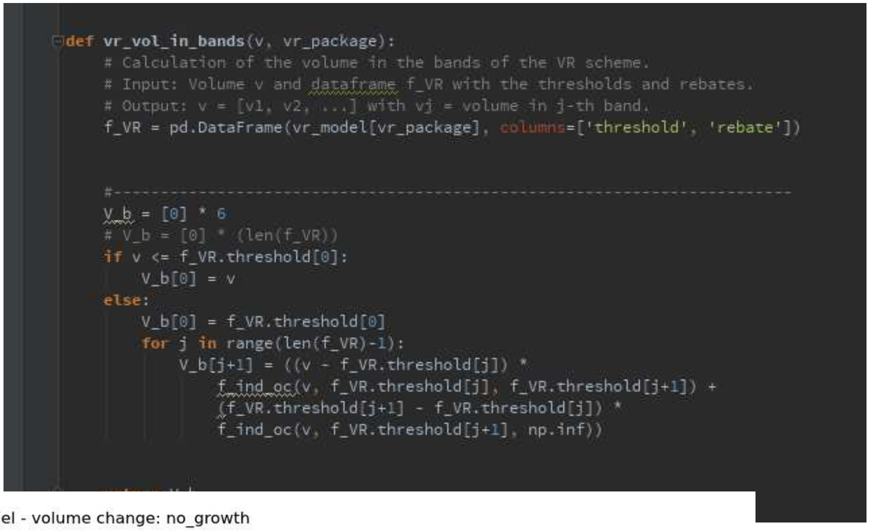
# Python in Pricing Engineering – Some Snapshots

## Extensive usage of hdf



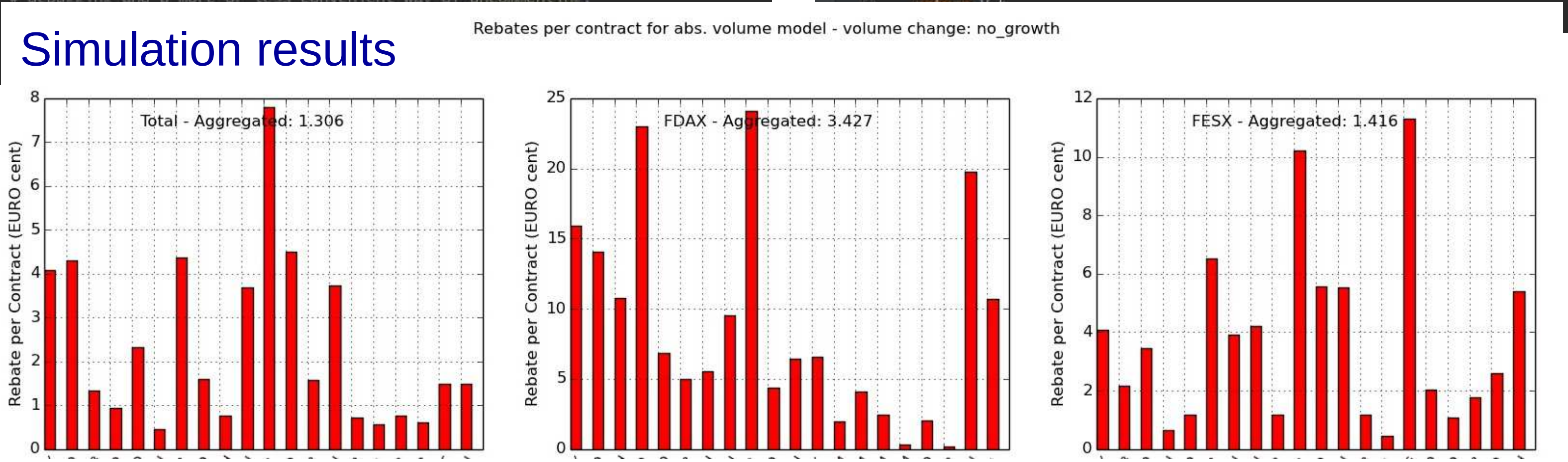
```
#---------------------------------------------------------------------------
# Typically the raw data are retrieved via a GUI or an sql query from other
# sources. For fast and convenient access the raw data are put into a pandas
# dataframe and this is in turn stored in a hdf file. Sizes up to 10 mio records
# are no problem.

def csv_to_hdf():
    file_list = ['file_base_name_' + str(year) + '.csv' for year in
            range(2010, 2011, 2012, 2013)]

    g = pd.DataFrame()
    for the_file in file_list:
        csv_file = data_path + the_file

        f = pd.read_csv(csv_file, sep=';')
        f['month'] = f['day'].map(set_month_1)

        g = g.append(f, ignore_index=True)


#===========================================================================
if __name__=='__main__':
    # The following is executed if the present file is run. This is only for
    # debugging and a more or less convenient way of uncommenting.
```

## Even complex calculations can be programmed very concise



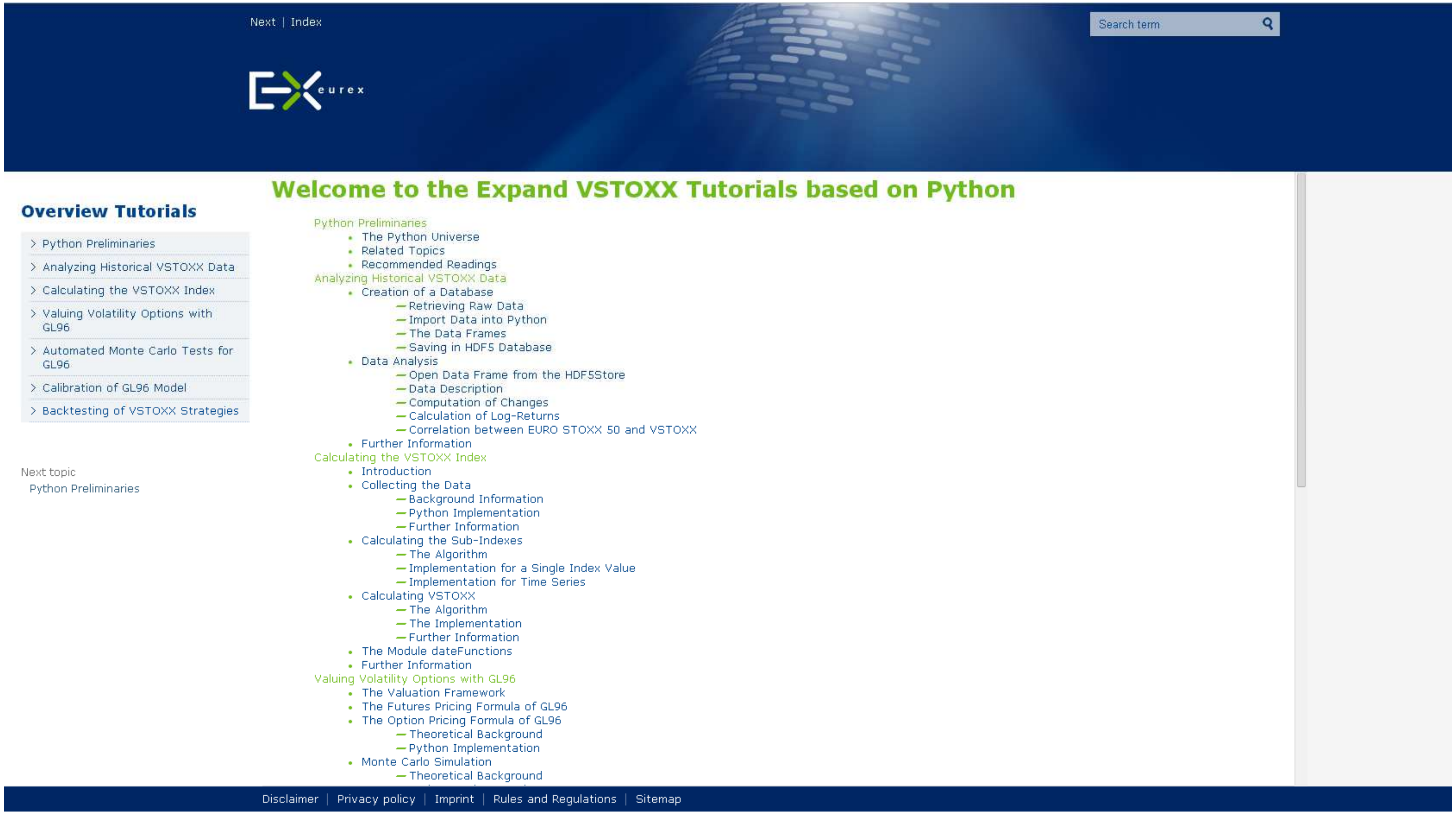
```
def vr_vol_in_bands(v, vr_package):
    # Calculation of the volume in the bands of the VR scheme.
    # Input: Volume v and dataframe f_VR with the thresholds and rebates.
    # Output: v = [v1, v2, ...] with vj = volume in j-th band.
    f_VR = pd.DataFrame(vr_model[vr_package], columns=['threshold', 'rebate'])


    #------------------------------------------------------------------------
    V_b = [0] * 6
    # V_b = [0] * (len(f_VR))
    if v <= f_VR.threshold[0]:
        V_b[0] = v
    else:
        V_b[0] = f_VR.threshold[0]
        for j in range(len(f_VR)-1):
            V_b[j+1] = ((v - f_VR.threshold[j]) *
                f_ind_oc(v, f_VR.threshold[j], f_VR.threshold[j+1]) +
                (f_VR.threshold[j+1] - f_VR.threshold[j]) *
                f_ind_oc(v, f_VR.threshold[j+1], np.inf))
```

## Simulation results



# Status quo – VSTOXX Advanced Services

http://www.eurexchange.com/vstoxx/



## Welcome to the Expand VSTOXX Tutorials based on Python

### Overview Tutorials

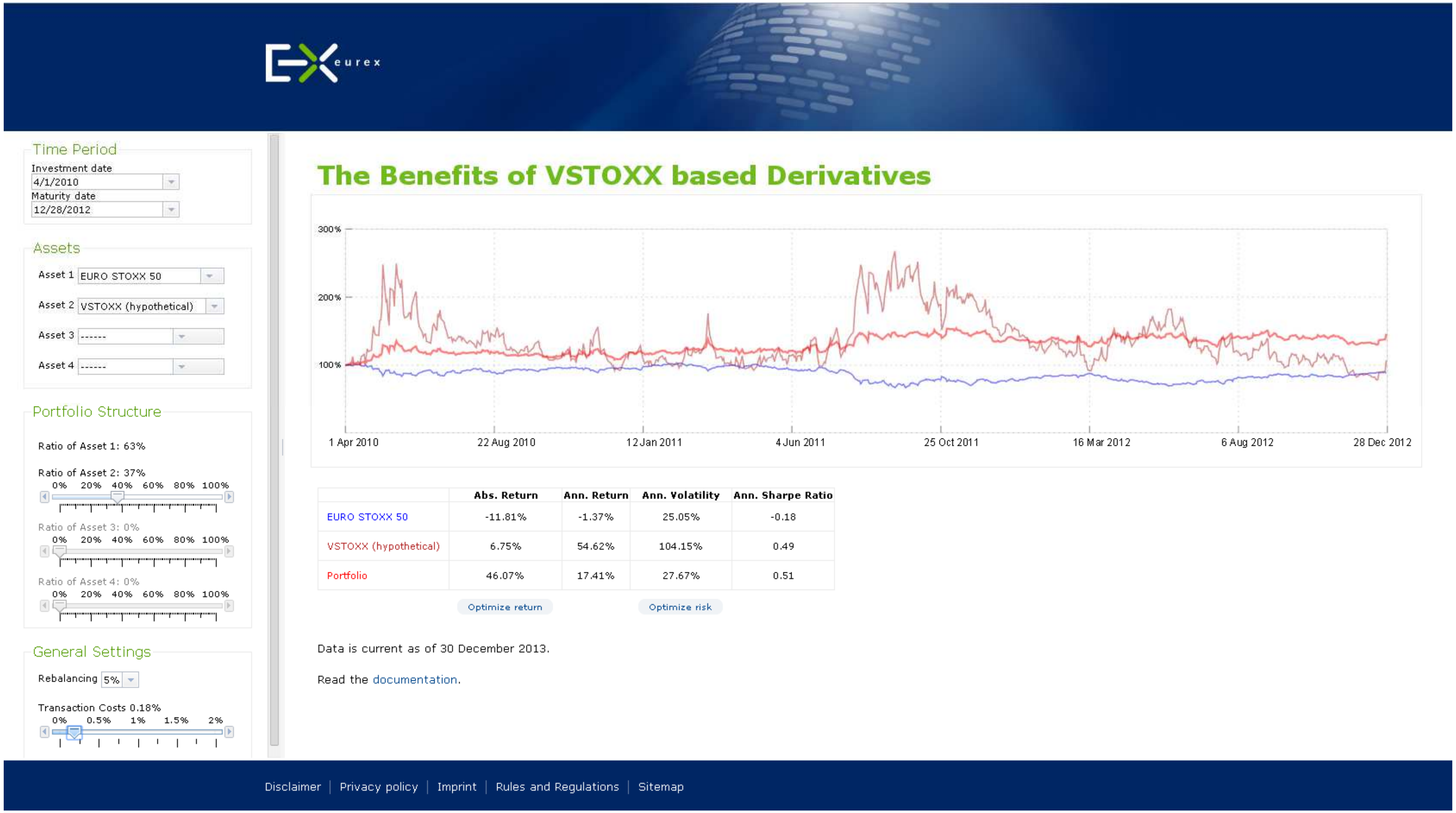
- Python Preliminaries
- Analyzing Historical VSTOXX Data
- Calculating the VSTOXX Index
- Valuing Volatility Options with GL96
- Automated Monte Carlo Tests for GL96
- Calibration of GL96 Model
- Backtesting of VSTOXX Strategies

Next topic
Python Preliminaries

Disclaimer | Privacy policy | Imprint | Rules and Regulations | Sitemap

# Status quo – Backtesting Applications

Source: www.eurexchange.com/vstoxx/app2/



Time Period

Investment date 4/1/2010

Maturity date 12/28/2012

Assets

Asset 1 EURO STOXX 50

Asset 2 VSTOXX (hypothetical)

Asset 3 ------

Asset 4 ------

Portfolio Structure

Ratio of Asset 1: 63%

Ratio of Asset 2: 37%

Ratio of Asset 3: 0%

Ratio of Asset 4: 0%

General Settings

Rebalancing 5%

Transaction Costs 0.18%

## The Benefits of VSTOXX based Derivatives

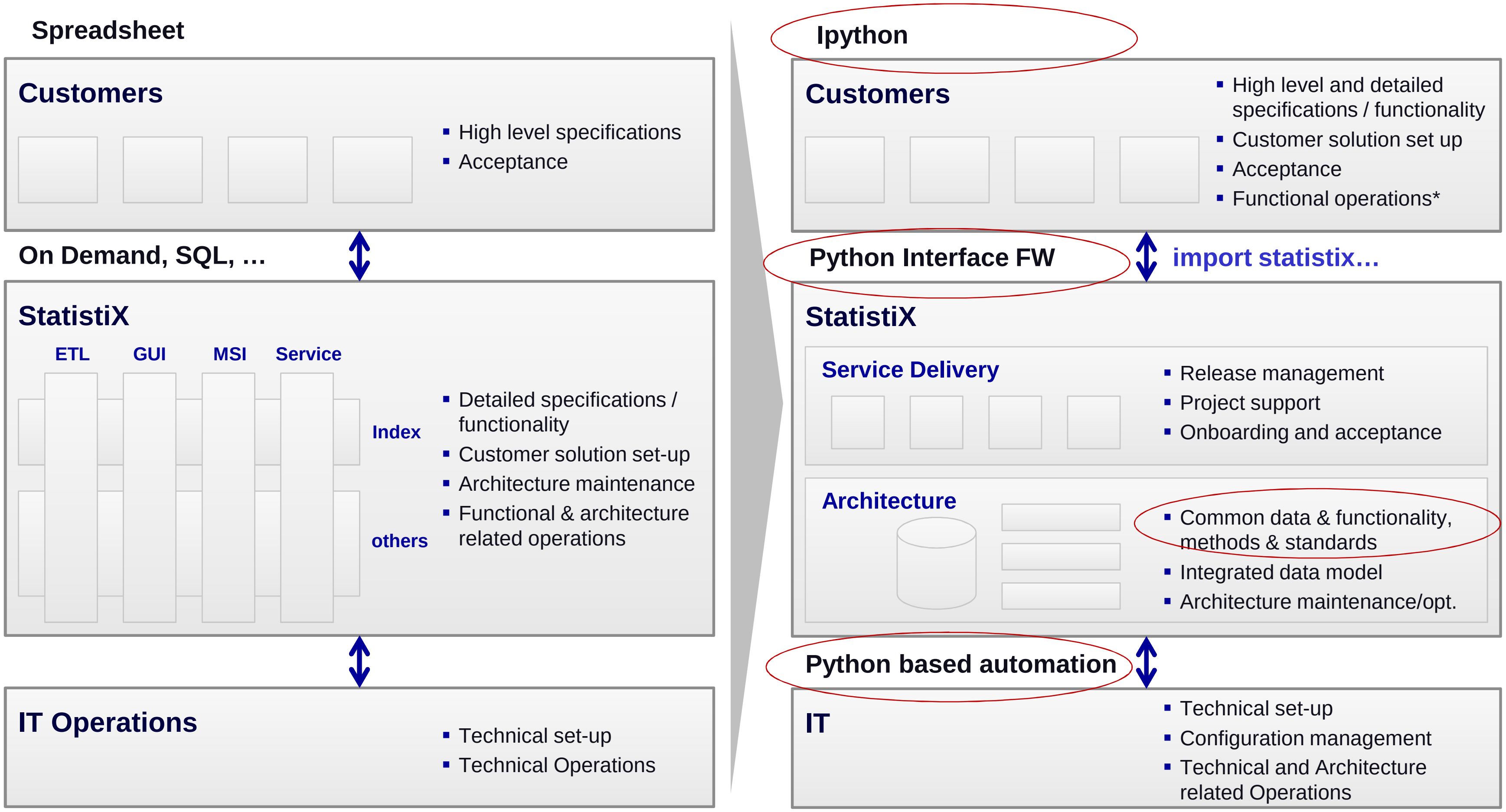
| | Abs. Return | Ann. Return | Ann. Volatility | Ann. Sharpe Ratio |
|---|---|---|---|---|
| EURO STOXX 50 | -11.81% | -1.37% | 25.05% | -0.18 |
| VSTOXX (hypothetical) | 6.75% | 54.62% | 104.15% | 0.49 |
| Portfolio | 46.07% | 17.41% | 27.67% | 0.51 |

Data is current as of 30 December 2013.

Read the documentation.

# Python within StatistiX©…Work in Progress

StatistiX® is the Data Warehouse of Deutsche Börse Group. StatistiX® provides external and internal users worldwide with statistical information on financial markets such as trading and price data, orders and quotes as well as clearing and settlement data. Furthermore StatistiX® offers Data Store Services and acts as Business Intelligence and Risk Analytics platform of Deutsche Börse Group.



**Spreadsheet**

**Customers**

- High level specifications
- Acceptance

**On Demand, SQL, …**

**StatistiX**

ETL GUI MSI Service

Index

others

- Detailed specifications / functionality
- Customer solution set-up
- Architecture maintenance
- Functional & architecture related operations

**IT Operations**

- Technical set-up
- Technical Operations

**Ipython**

**Customers**

- High level and detailed specifications / functionality
- Customer solution set up
- Acceptance
- Functional operations*

**Python Interface FW**

**import statistix…**

**StatistiX**

**Service Delivery**

- Release management
- Project support
- Onboarding and acceptance

**Architecture**

- Common data & functionality, methods & standards
- Integrated data model
- Architecture maintenance/opt.

**Python based automation**

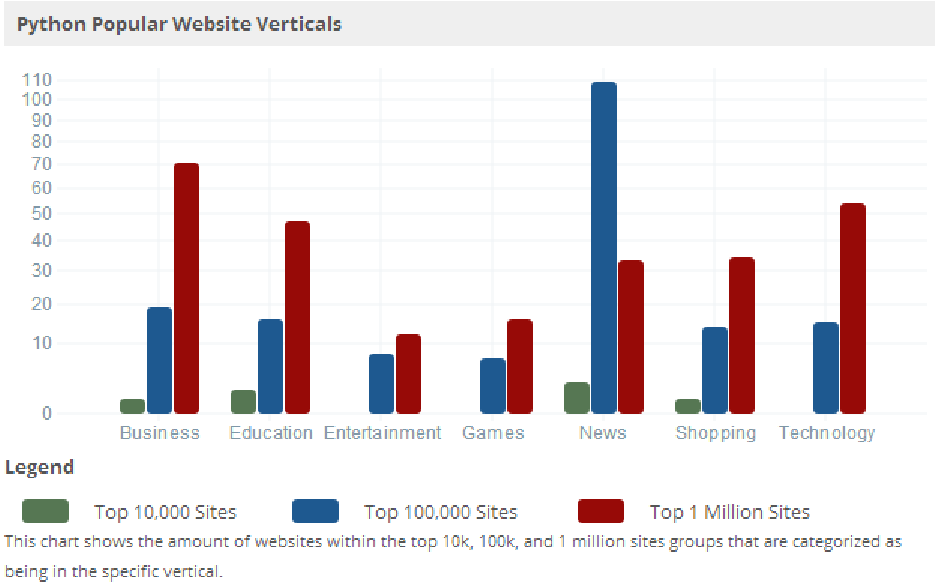
**IT**

- Technical set-up
- Configuration management
- Technical and Architecture related Operations

DEUTSCHE BÖRSE GROUP

# Python - State of The Union

# Websites using Python for specific verticals



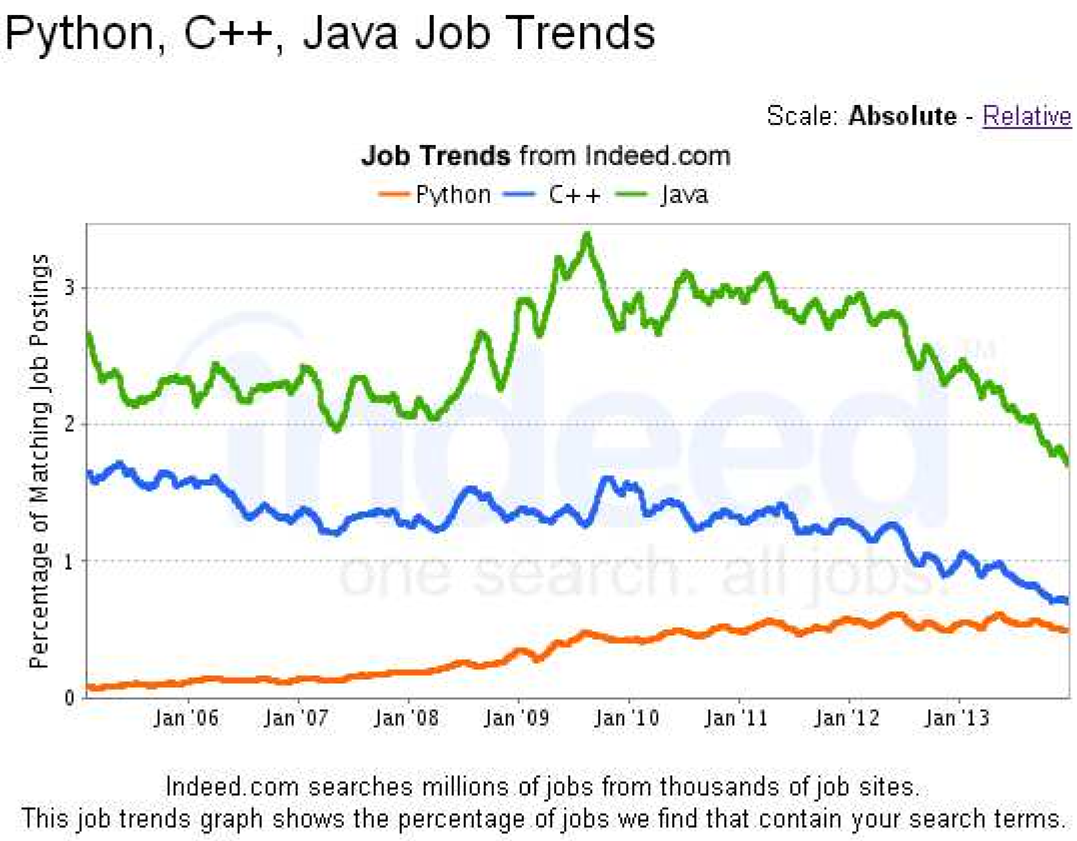
This chart shows the amount of websites within the top 10k, 100k, and 1 million sites groups that are categorized as being in the specific vertical.

http://trends.builtwith.com/framework/Python

**Some Python based technologies used to build the website:**

- **Django**
  - “Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.”. https://www.djangoproject.com/
- **Flask**
  - “The “micro” in microframework means Flask aims to keep the core simple but extensible…Flask will continue to provide a very simple glue layer to the best that Python has to offer.”. http://flask.pocoo.org/docs/
- **Pylons (Pyramid)**
  - “Pyramid is a very general open source Python web framework.”. http://www.pylonsproject.org/projects/pyramid/about
- **Tornado Server**
  - “Tornado s a Python web framework and asynchronous networking library…”. http://www.tornadoweb.org/en/stable/

# Indeed job trends for Python compared to C++ and Java



Python, C++, Java Job Trends

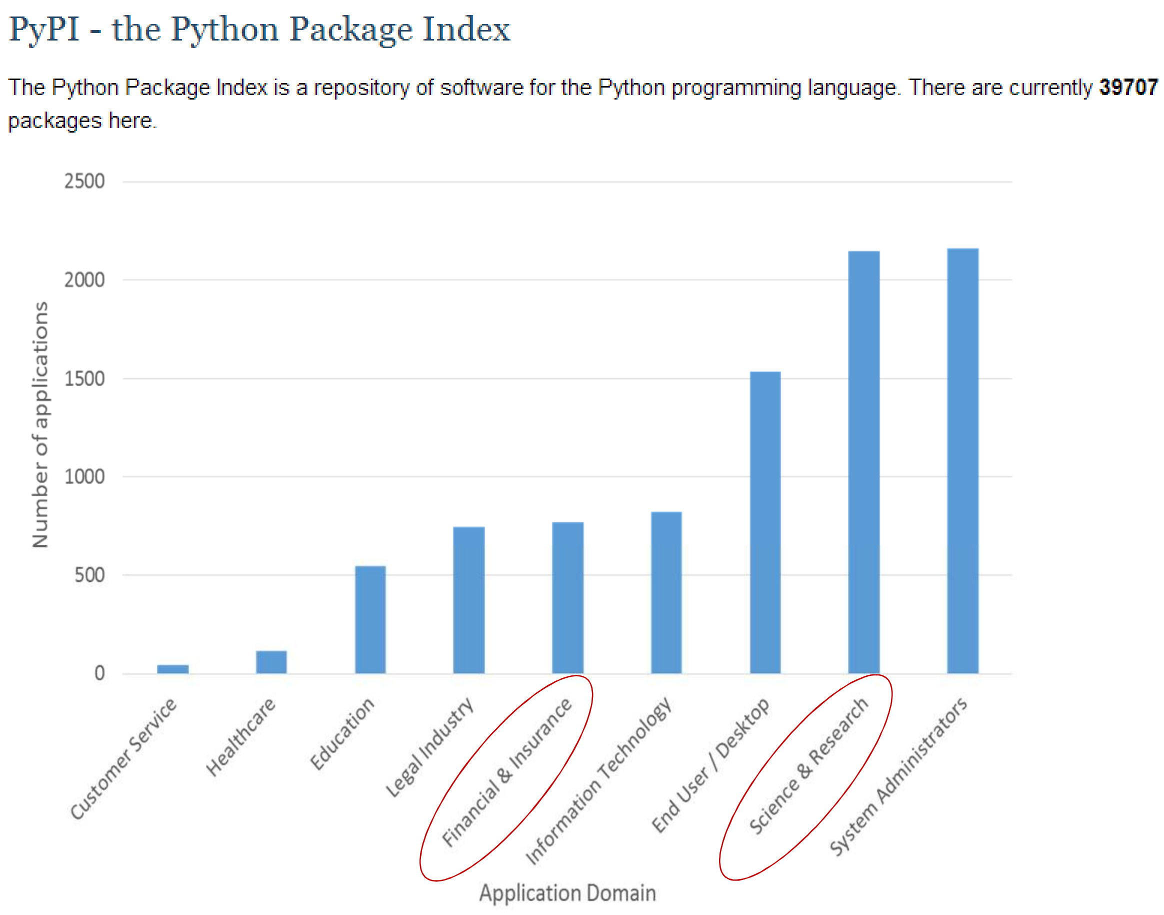
Scale: **Absolute** - Relative

**Job Trends** from Indeed.com

Indeed.com searches millions of jobs from thousands of job sites.
This job trends graph shows the percentage of jobs we find that contain your search terms.

http://www.indeed.com/jobtrends?q=Python%2C+C%2B%2B%2C+Java&l=&relative=1

# Python Repository statistics based on indented audience

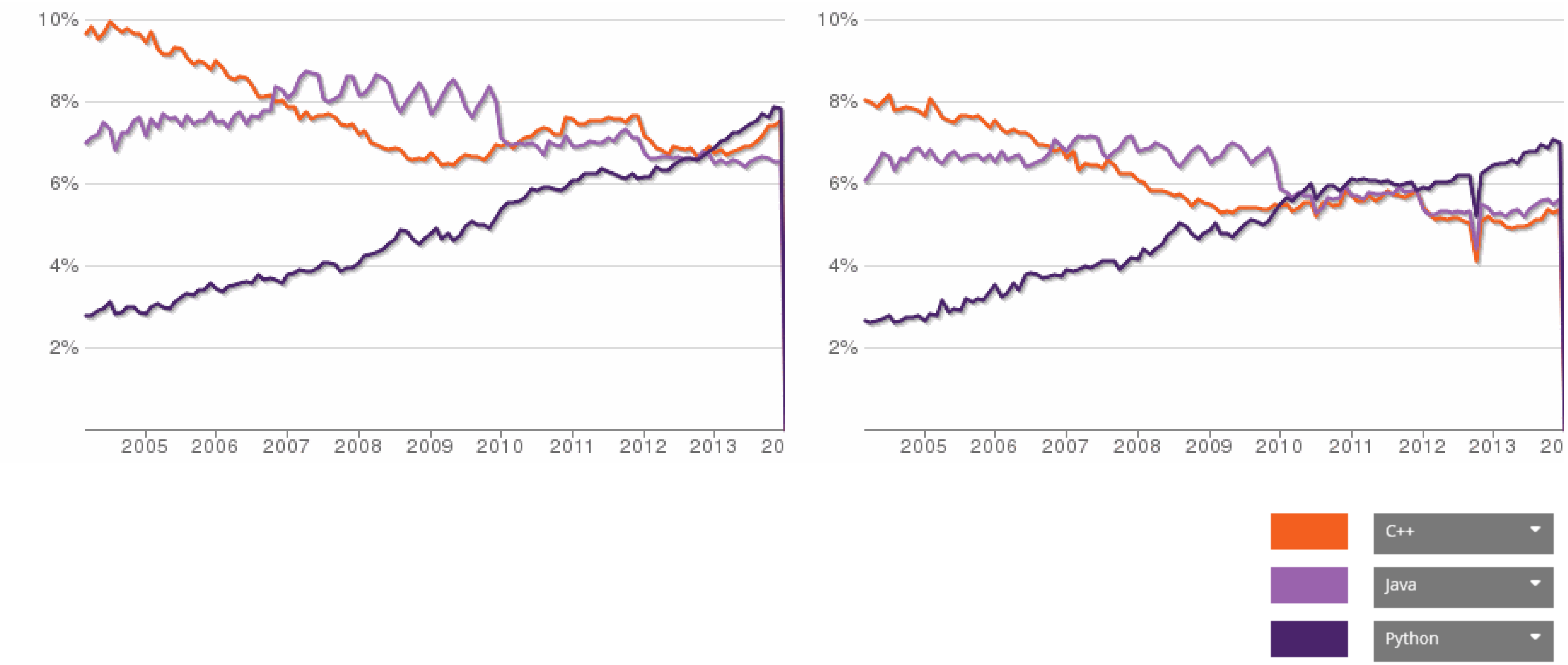


PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **39707** packages here.

https://pypi.python.org/pypi?%3Aaction=browse

# Oloh statistics for open source projects providing project statistics (activity, codebase, contributors) project language statistics (commits, changed lines of code, total number of new projects)



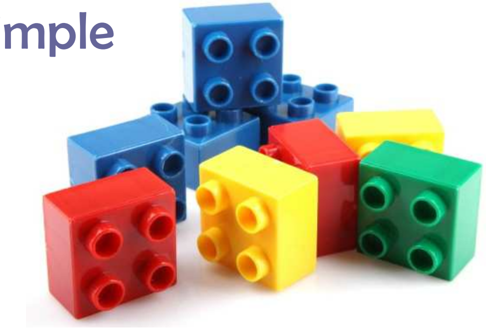
Monthly Contributors (Percent of Total)

Monthly Projects (Percent of Total)

http://www.ohloh.net/languages/compare?commit=Update&l0=cpp&l1=python&l2=java&l4=-1&measure=projects&percent=true

# Tools to build simple to complex financial models & applications

From Simple



to complex

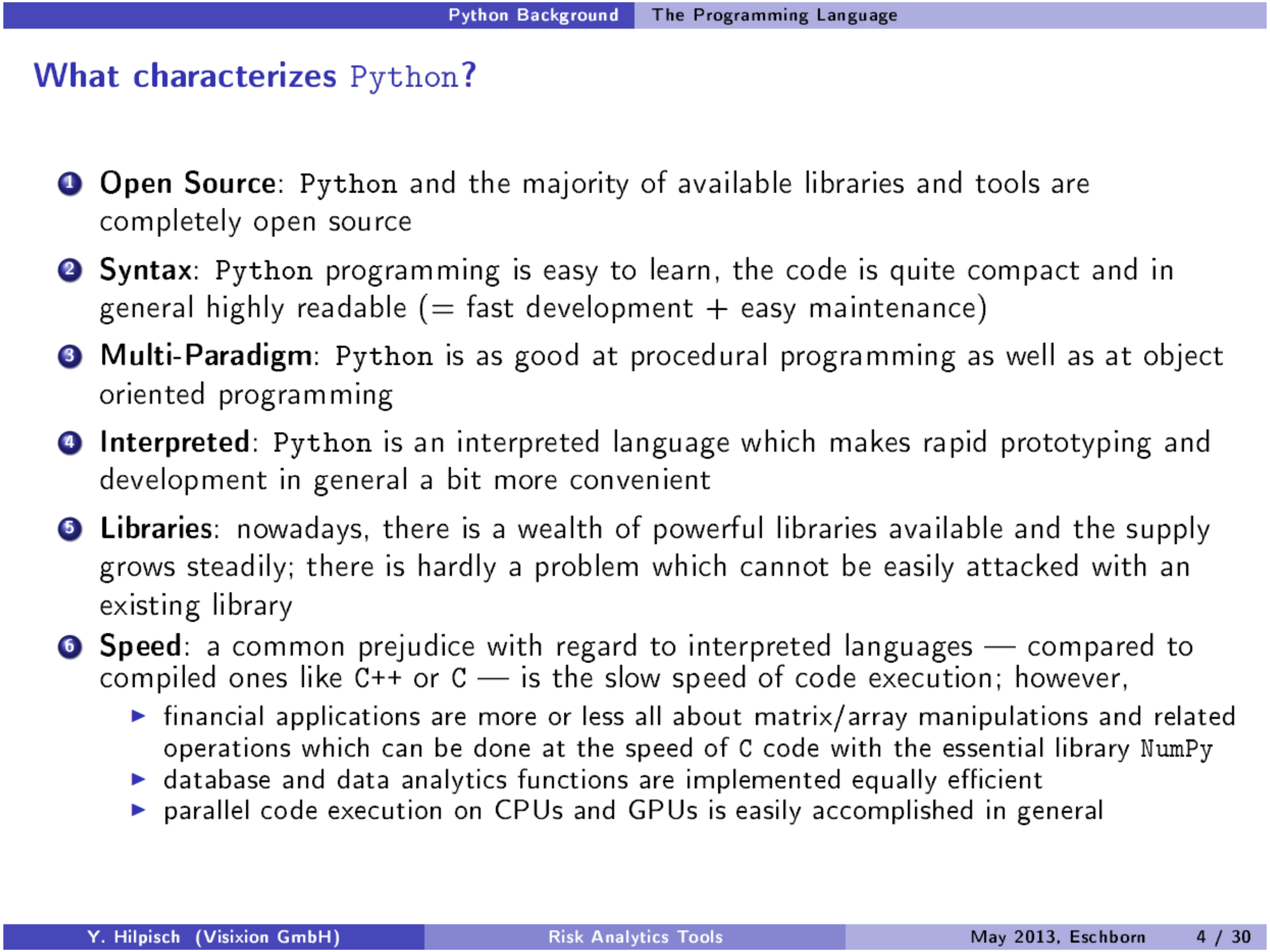
financial applications

- iPython : interactive development environment
- NumPy : high performance multi-dimensional arrays processing, linear algebra, random number generation, efficient binary I/O
- SciPy & Scikits : matrix manipulation, probability distributions, data minig, machine learning
- Pandas : high performance data structures and tools for Python
- Zipline : algorithmic trading simulator, includes common transforms and common risk calculations
- Statsmodels : large array of statistical models and statistical tests
- QuantLib-Python, PyQL
- Matplotlib : data visualization
- PyCUDA : GPU computing
- RPy2 : R to Python wrapper
- Cython : default compiler
- Boost.Python : C++ library allowing seamless interoperability between C++ and Python
- Jython : Java to Python
- Numba : LLVM Compiler
- …

# Potential of Python in the Software Factory

# Why bother about a programming language from a software factory perspective?

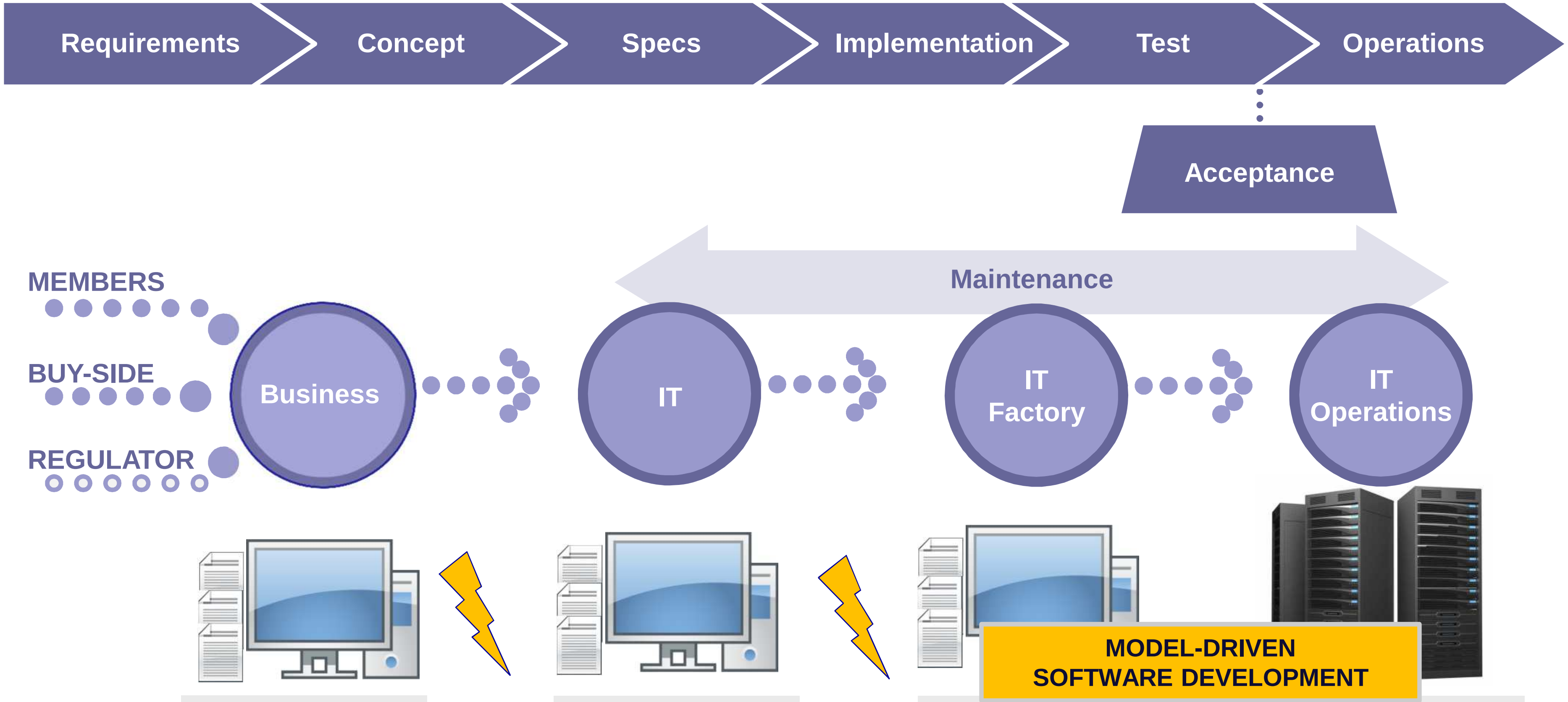
- **Increasing productivity/efficiency is a permanent challenge for almost any IT in any organization**
  - One multi-purpose language that allows solving most of the challenges reduces complexity and creates synergies
  - Test and fail fast
  - Multiplatform & End-to-End Capability (Research to Production)
- **Time-to-market**
  - Shortening of learning curves helps teams to quickly focus on tasks
  - Iterative approach in contrast to compilation cycles
  - Rich frameworks let teams focus on developing content and not fighting with the tool
- **Maintain degrees of freedom**
  - Vendor “Lock-In” makes only sense if the vendor is always ahead of my own requirements., which comes with a high “insurance policy”…
  - Open-Source paradigm “Invest in People and not in Licenses”



## What characterizes Python?

1. **Open Source**: Python and the majority of available libraries and tools are completely open source
2. **Syntax**: Python programming is easy to learn, the code is quite compact and in general highly readable (= fast development + easy maintenance)
3. **Multi-Paradigm**: Python is as good at procedural programming as well as at object oriented programming
4. **Interpreted**: Python is an interpreted language which makes rapid prototyping and development in general a bit more convenient
5. **Libraries**: nowadays, there is a wealth of powerful libraries available and the supply grows steadily; there is hardly a problem which cannot be easily attacked with an existing library
6. **Speed**: a common prejudice with regard to interpreted languages — compared to compiled ones like C++ or C — is the slow speed of code execution; however,
   - financial applications are more or less all about matrix/array manipulations and related operations which can be done at the speed of C code with the essential library NumPy
   - database and data analytics functions are implemented equally efficient
   - parallel code execution on CPUs and GPUs is easily accomplished in general

# Traditional IT Value Chain



Requirements → Concept → Specs → Implementation → Test → Operations

Acceptance

Maintenance

MEMBERS

BUY-SIDE

REGULATOR

Business

IT

IT Factory

IT Operations

MODEL-DRIVEN SOFTWARE DEVELOPMENT

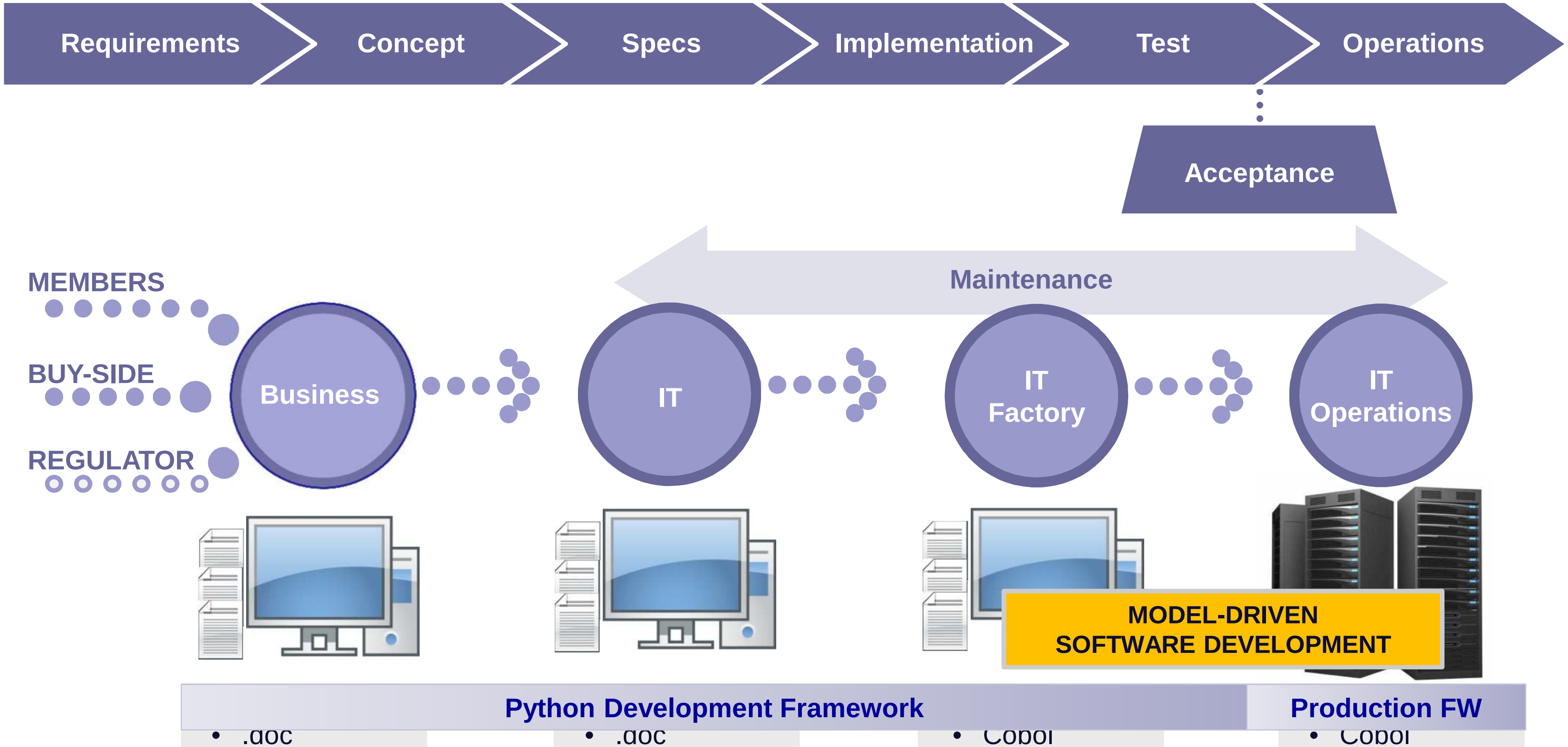
- .doc
- .xls
- R
- Matlab®
- Mathematica
- Python

- .doc
- .UML
- Testcases

- Cobol
- Java
- C++
- SQL
- HTML

- Cobol
- Java
- C++
- SQL
- HTML

# Alternative IT Value Chain (I) ???



- .doc
- .xls
- R
- Matlab®
- Mathematica
- Python

- .doc
- .UML
- Testcases

- Cobol
- Java
- C++
- SQL
- HTML

- Cobol
- Java
- C++
- SQL
- HTML

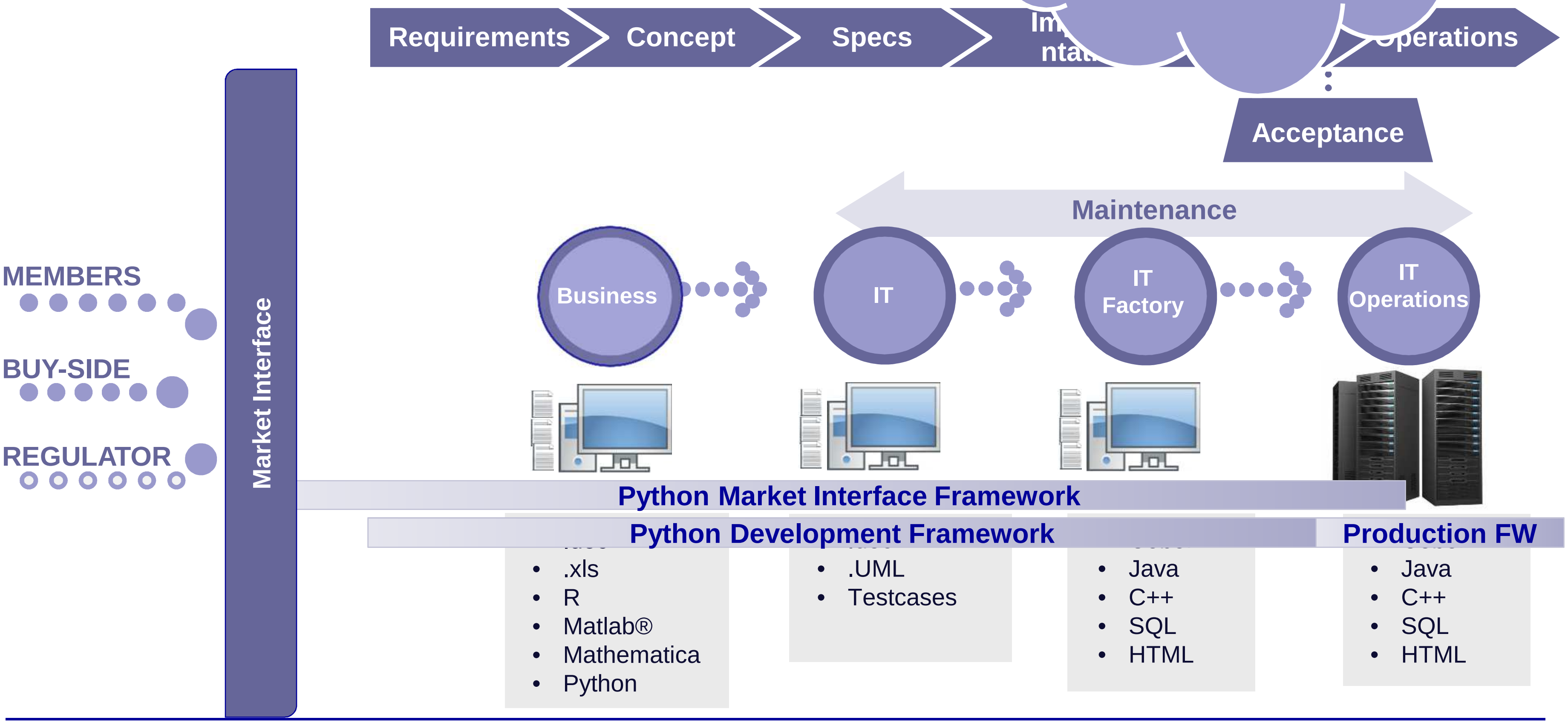
# Alternative IT Value Chain (II) ???

**pop·u·lar·ize**

1. Cause (something) to become generally liked
2. Make (something technical, scientific, or academic) accessible or interesting to the general public by presenting it in a readily understandable form

*Source: www.google.com*

Quants, Financial Engineers, Traders, Asset Managers, Risk Managers, Model Validators, Analytics Providers, Developers, Analysts, Students, Lecturers, Researchers, Auditors ...



- .xls
- R
- Matlab®
- Mathematica
- Python

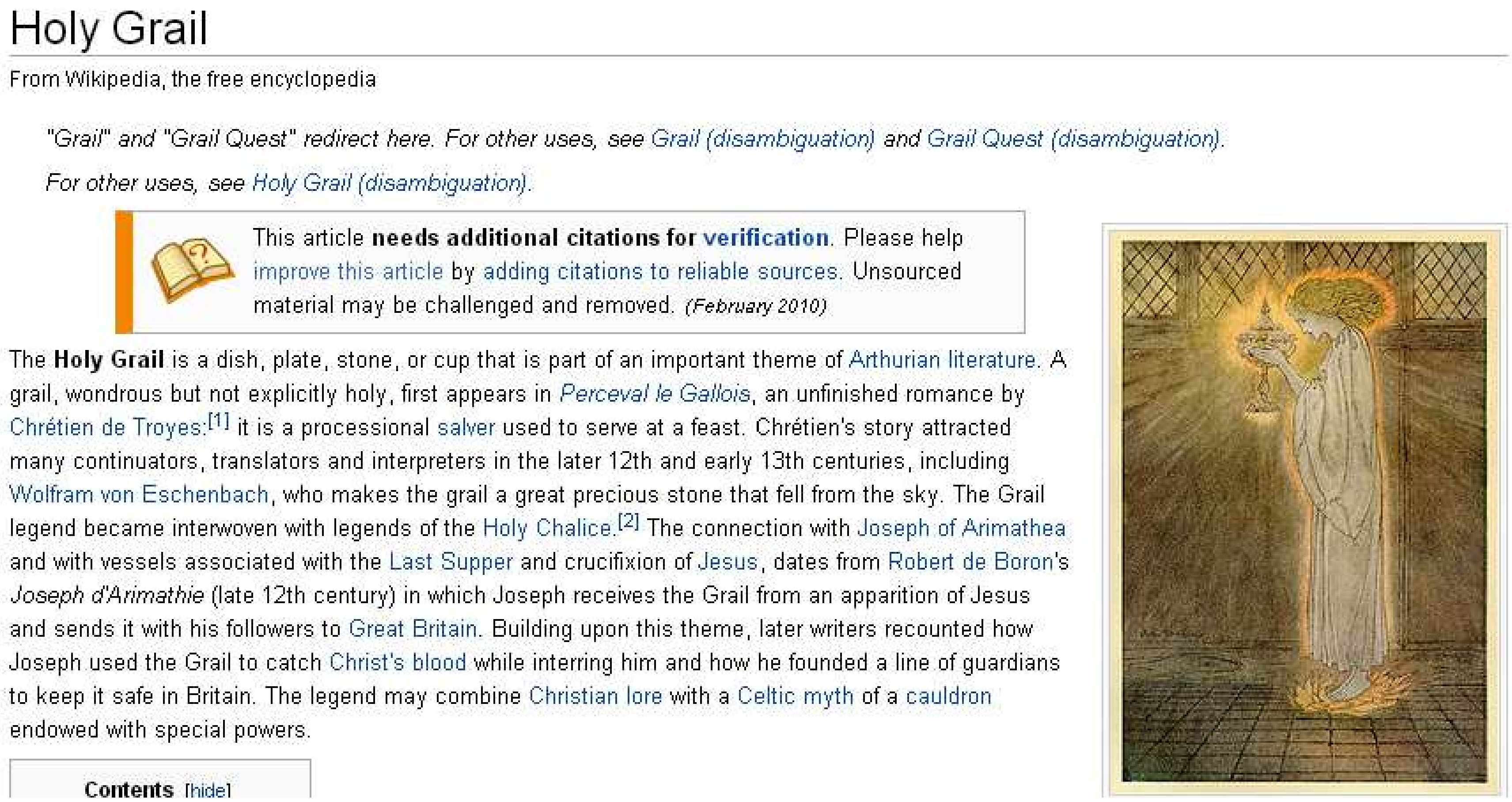
- .UML
- Testcases

- Java
- C++
- SQL
- HTML

- Java
- C++
- SQL
- HTML

# Conclusion

# The endless search for the…



# Holy Grail

From Wikipedia, the free encyclopedia

*"Grail" and "Grail Quest" redirect here. For other uses, see Grail (disambiguation) and Grail Quest (disambiguation).*

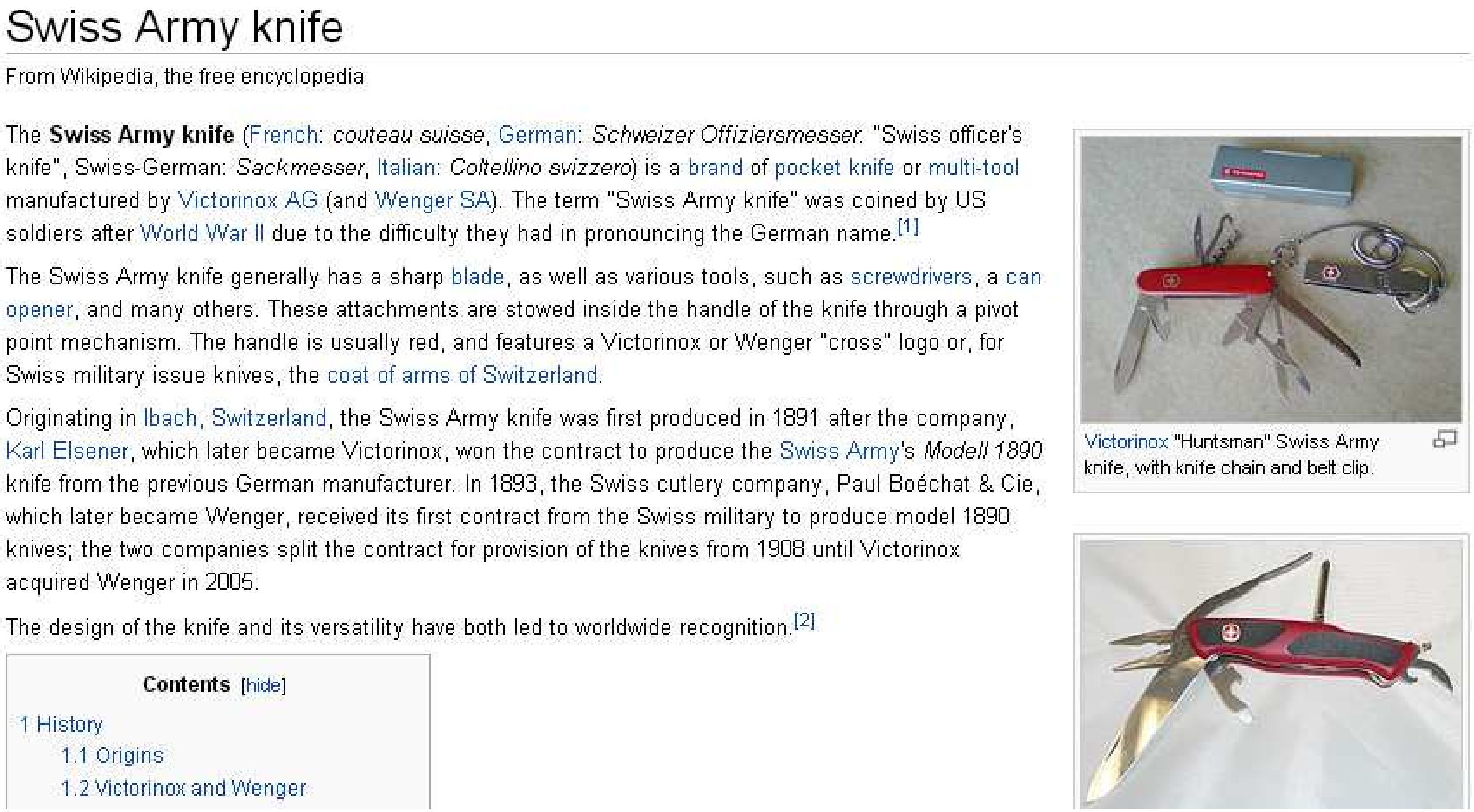
*For other uses, see Holy Grail (disambiguation).*

This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(February 2010)*

The **Holy Grail** is a dish, plate, stone, or cup that is part of an important theme of Arthurian literature. A grail, wondrous but not explicitly holy, first appears in *Perceval le Gallois*, an unfinished romance by Chrétien de Troyes:[1] it is a processional salver used to serve at a feast. Chrétien's story attracted many continuators, translators and interpreters in the later 12th and early 13th centuries, including Wolfram von Eschenbach, who makes the grail a great precious stone that fell from the sky. The Grail legend became interwoven with legends of the Holy Chalice.[2] The connection with Joseph of Arimathea and with vessels associated with the Last Supper and crucifixion of Jesus, dates from Robert de Boron's *Joseph d'Arimathie* (late 12th century) in which Joseph receives the Grail from an apparition of Jesus and sends it with his followers to Great Britain. Building upon this theme, later writers recounted how Joseph used the Grail to catch Christ's blood while interring him and how he founded a line of guardians to keep it safe in Britain. The legend may combine Christian lore with a Celtic myth of a cauldron endowed with special powers.

http://en.wikipedia.org/wiki/Holy_Grail

# …instead, go for a more robust approach.



## Swiss Army knife

From Wikipedia, the free encyclopedia

The **Swiss Army knife** (French: *couteau suisse*, German: *Schweizer Offiziersmesser.* "Swiss officer's knife", Swiss-German: *Sackmesser*, Italian: *Coltellino svizzero*) is a brand of pocket knife or multi-tool manufactured by Victorinox AG (and Wenger SA). The term "Swiss Army knife" was coined by US soldiers after World War II due to the difficulty they had in pronouncing the German name.[1]

The Swiss Army knife generally has a sharp blade, as well as various tools, such as screwdrivers, a can opener, and many others. These attachments are stowed inside the handle of the knife through a pivot point mechanism. The handle is usually red, and features a Victorinox or Wenger "cross" logo or, for Swiss military issue knives, the coat of arms of Switzerland.

Originating in Ibach, Switzerland, the Swiss Army knife was first produced in 1891 after the company, Karl Elsener, which later became Victorinox, won the contract to produce the Swiss Army's *Modell 1890* knife from the previous German manufacturer. In 1893, the Swiss cutlery company, Paul Boéchat & Cie, which later became Wenger, received its first contract from the Swiss military to produce model 1890 knives; the two companies split the contract for provision of the knives from 1908 until Victorinox acquired Wenger in 2005.

The design of the knife and its versatility have both led to worldwide recognition.[2]

Victorinox "Huntsman" Swiss Army knife, with knife chain and belt clip.

**Contents** [hide]

http://en.wikipedia.org/wiki/Swiss_Army_knife

# Python in the Financial Industry
# The universal tool for end-to-end development

1. Python is a robust basis for solving many of the daily software technology challenges that we face in the financial industry (but not limited to…)

2. The multi-purpose paradigm (and not working too bad in all these areas) creates inherent synergies across application domains

3. It’s readability fits the brain, it is fun to work with and has a vibrant community

4. The speed & momentum of frameworks development is impressive

5. Addressing many user groups allow for an end-to-end usage from Quants to system operators and all the way back into a broad user community

Deutsche Börse IT has included Python in it’s technology stack and will continue to use it as an integrative technology…end-to-end.