So you think your startup is worth $10 million...

EuroPython 2016
Bilbao, Basque Country, Spain
Marc-André Lemburg
Speaker Introduction

Marc-André Lemburg

- Python since 1994
- Studied Mathematics
- eGenix.com GmbH
- Senior Software Architect
- Consultant / Trainer
- Python Core Developer
- EuroPython Society
- Python Software Foundation
- Based in Düsseldorf, Germany
Agenda

• Introduction
• Analysis
• Models
• Valuation
• Make of buy
• Conclusion
Buying Python Startups
Disclaimer

• These ideas were used in an actual valuation
  – We do not claim completeness
  – We do not claim scientific accuracy

• The results do make sense based on our experience in running projects
Value of an IT startup

• **Business value**
  - Market share = users / market size
  - Cost efficiency (HR, processes)
  - Innovation factors
  - **Risks** (affecting operations)
  - ...

• **IT value**
  - Quality of developers / managers
  - Application design quality (structure, flexibility)
  - Code quality (structure, metrics, tests)
  - **Risks** (affecting technical capabilities)
  - ...

(c) 2016 eGenix.com GmbH, info@egenix.com
Risks
Business risks

• Affecting the business operation
  – Loosing important employees
  – Financial / investment risks
  – Market changes
  – Competing against open source / freebies
  – Infringements (patent/trademark/regulations)
  – Downtime
  – Data security breaches
  – ...

IT risks

• Affecting technical capabilities
  – Problems in third party tools / extensions /services (dependencies)
  – Scalability problems (increase in load or storage requirements)
  – Flexibility problems (slow innovation)
  – Maintenance problems (fixing bugs takes too long)
  – Hardware issues (failing servers, disks, connectivity)
  – Environmental issues (fire, earthquake, storm)
  – ...

Value of an IT startup

• Business value
  – Market share = users / market size
  – Cost efficiency (HR, processes)
  – Innovation factors
  – Risks (affecting operations)
  – ...

• IT value
  – Quality of developers / managers
  – Application design quality (structure, flexibility)
  – Code quality (structure, metrics, tests)
  – Risks (affecting technical capabilities)
  – ...
IT valuation project approach

- Analyze IT approach, team, system and data
- Initial development valuation based on:
  - COCOMO model
  - Effort model
- Apply “Added Value” factors (including risk)
- Compare with reimplementation estimate
  → Make or buy
Agenda

• Introduction
• Analysis
• Models
• Valuation
• Make of buy
• Conclusion
IT valuation analysis factors

• Soft factors
  – Quality of developers
  – Architecture quality
  – Data model quality
  – Algorithmic quality
  – Extensibility
  – Risks

• Factors (partially) based on metrics
  – Code quality

• Known inaccuracies
  – Estimation risk buffer
IT valuation analysis factors

• Soft factors
  – Quality of developers
  – Architecture quality
  – Data model quality
  – Algorithmic quality
  – Extensibility
  – Risks

• Factors (partially) based on metrics
  – Code quality

• Known inaccuracies
  – Estimation risk buffer
IT valuation analysis factors

• Soft factors
  – Quality of developers
  – Architecture quality
  – Data model quality
  – Algorithmic quality
  – Extensibility
  – Risks

• Factors (partially) based on metrics
  – Code quality

• Known inaccuracies
  – Estimation risk buffer
Raw code metrics

• Source data analysis
  – Lines of code (LOC), Source lines (SLOC), Logical lines (LLOC)
  – Blank lines = better readability
  – LOC per module
  – Functions/methods/classes per module
    → Affect maintainability

• Python tool: Radon
  – https://pypi.python.org/pypi/radon
Raw code metrics

• Inline documentation
  – Comment lines (in relation to LOC)
  – Doc strings (in relation to LOC)
    → Affect readability and maintainability

• Python tool: Radon
  – https://pypi.python.org/pypi/radon
Code metrics

• **Cyclomatic Complexity (CC)**
  - more decision nodes = higher complexity
  - higher values = worse

• **Maintainability Index (MI)**
  - combination of complexity, density, SLOCs and comment lines
  - higher values = better

• Python tool: Radon
  - https://pypi.python.org/pypi/radon
Test coverage

• Check **unit test** code coverage of code base
  – should show high values
  – note: 100% coverage is often misleading

• Check for **end-to-end tests**
  – should provide good coverage as well

• Check for randomized tests
  – to avoid biased test cases / missing test cases

• Python tool: coverage.py
  – [https://coverage.readthedocs.io/](https://coverage.readthedocs.io/)
Agenda

• Introduction
• Analysis
• Models
• Valuation
• Make of buy
• Questions
Intermediate COCOMO Model

• COCOMO model is an industry standard for code valuation based on LOC
  – C/C++
  – Java

• Models:
  – Organic projects - small teams, senior/regular people, agile process
  – Semi-detached projects – medium sized teams, mixed skill set, semi-rigid requirements
  – Embedded projects – tight requirements, low level architectures, usually hardware based
Intermediate COCOMO Model

• Formulas:
  – **Effort Applied** \( E = a \times \text{kLOC}^b \times \text{EAF} \) (in person months)
  – **Dev Time** \( D = c \times E^d \) (in months)
  – People required \( P = \frac{E}{D} \) (in persons)

• Parameter selection (organic project category):
  – \( a=2.40, b=1.05, c=2.50, d=0.38 \)

• Adjustment factor \( \text{EAF} \) (lower = more efficient)
  – Normal: 0.9 – 1.4 (Java, C)
  – Python: 0.5

https://en.wikipedia.org/wiki/COCOMO
Intermediate COCOMO Model Value

- **Value** = Developer *costs* * Development *time*
  - Take into account different costs for senior and regular developers
  - Use market rates / apply startup discounts
  - Add employer labor costs
Effort model

- **Time it took the company** to build its system
  - Broken down by senior and regular developers used in the process

- **Value** = Developer costs * Development time
  - Take into account different costs for senior and regular developers
  - Use market rates / apply startup discounts
  - Add employer labor costs
Agenda

• Introduction
• Analysis
• Models
• Valuation
• Make of buy
• Conclusion
Added Value
Added value

• Apply +/- Factor % in the following categories:
  – Quality of developers
  – Architecture quality
  – Data model quality
  – Algorithmic quality
  – Code quality
  – Extensibility
  – Risks

  – Estimation risk buffer
Code valuation

• Pragmatic approach: Average from applied models
  – COCOMO model
  – Effort model

• Apply added value factor

• Final estimate
Data valuation (if applicable)

• Average from applied models
  – COCOMO model
  – Effort model

• Apply added value factor

• Final estimate
Agenda

• Introduction
• Analysis
• Models
• Valuation
• Make or buy
• Questions
Make or buy

- Costs of replicating the company, including:
  - products / data
  - expertise
  - reaching market share
Make or buy

- Business side
  - Setting up company
  - Recruiting
  - Marketing costs
  - ...

- IT systems
  - Costs of acquiring needed expert knowledge
  - Costs of reimplementing all systems
  - (Costs of recreating data)
  - Development time
Make or buy

- Business side
  - Setting up company
  - Recruiting
  - Marketing costs
  - ...

- IT systems
  - Costs of acquiring needed expert knowledge
  - Costs of reimplementing all systems
  - (Costs of recreating data)
  - Development time
Make or buy

• For IT systems:
  – use existing system as specification
  – estimate effort needed to recreate systems
  – (estimate effort needed to recreate data)
  – since timing is important: use senior developers only

• Result: Offer for rebuilding the system
Agenda

• Introduction
• Analysis
• Models
• Valuation
• Make or buy
• Conclusion
Add Value to your Startup
How to increase the IT value of a Python startup

• Pay attention to code complexity / structure/ quality

• Design in a flexible and easily extensible way

• Pay attention to code test coverage and documentation

• Invest into good data(base) structures

• Invest into good algorithms

• Reduce risks added via 3rd party dependencies
>>> raise Question()
Photo References

CC BY / CC BY-SA licensed photos:
https://www.flickr.com/photos/132084522@N05/17086570218/
https://www.flickr.com/photos/chiropractic/6844437142/
https://www.flickr.com/photos/armydre2008/16366377932/
https://www.flickr.com/photos/nasaearthobservatory/6405553723/

Public domain clip arts:
https://openclipart.org/detail/8879/rubber-duck
https://openclipart.org/detail/215201/evolution

All other photos and images are used by permission.
Thank you for your attention

Beautiful is better than ugly.
Contact

eGenix.com Software, Skills and Services GmbH
Marc-André Lemburg
Pastor-Löh-Str. 48
D-40764 Langenfeld
Germany

eMail: mal@egenix.com
Phone: +49 211 9304112
Fax: +49 211 3005250
Web: http://www.egenix.com/