Software development

for startup entrepreneurs

Prof. Dr. Ralf Lämmel
University of Koblenz-Landau
Faculty of Computer Science
Software Languages Team

Challenges of software development for startups

https://www.entrepreneurs-journey.com/10226/the-challenge-of-developing-a-software-startup/

The Challenge Of Develop

Ralf

https://www.entrepreneurs-journey.com/10226/the-challenge-of-developing-a-software-startup/

HOME | START GUIDE | REPORTS | EMAIL COURSES | COACHING | SUCCESS STORIES

The Challenge Of Developing A Software Startup

By Yaro Starak
24 Comments

Facebook
Twitter
Google+
LinkedIn

13
SHARES

I've been working on CrankyAds.com now for over a year, although there have been a few start-stop sessions, so it hasn't been exactly all-steam ahead during that time.

Developing software is a lot of fun, when you have the people who can make your vision a reality. Thinking of how you want something to work in your head and then seeing it work in real life is very gratifying.

On the flipside though, it's just as frustrating to realize how many features you are NOT including because you simply don't have the funds or the manpower to complete them.

When I first began development of CrankyAds I figured a *quick and dirty* version could be done in a couple of months to offer the core features. The development company I hired seemed to agree, and quoted a timeframe that slightly exceeded my expectations (although I always add one or two extra months on top of my

http://www.socalcto.com/2009/12/startup-software-development-do-your.html

Startup Software Develop ×

Ralf

www.socalcto.com/2009/12/startup-software-development-do-your.html

SOCAL CTO

Startup CTO :: Startup Development :: Matching :: Los Angeles Startup Community

WEDNESDAY, DECEMBER 9, 2009

Startup Software Development – Do Your Homework Before You Develop Anything

I just had an all-too common conversation with the founder of a startup who had spent more than a year working with a software development company who had produced a mess. The mess really comes from a developer who was willing to get started on a product that was not fully thought out.

I always take a very different approach in early conversations. If I'm being asked to do startup software development, I'm going to push fairly hard on key questions that the startup needs to have answered before they develop anything. Some founders are taken aback. They are calling me to go build what they tell me to build. Why would I question whether they've done their homework? And most often I only know enough about their business to be dangerous. So why ask all these questions if I might lose a potential client?

If I don't ask the questions and do a little bit of homework, then likely we will end up with a mess.

So what are the questions I'm likely going to ask?

ABOUT ME

TONY KARRER

Follow 2.9k

Dr. Tony Karrer works as a part-time CTO for startups and midsize software companies - helping them get product out the door and turn around technology issues. He is considered one of the top technologists in eLearning and is known for working with numerous startups including being the original CTO for eHarmony for its first four years. Dr. Karrer taught Computer Science for eleven years. He has also worked on projects for many Fortune 500 companies including Credit Suisse, Royal Bank of Canada, Citibank, Lexus, Microsoft, Nissan, Universal, IBM, Hewlett-Packard, Sun Microsystems, Fidelity Investments, Symbol Technologies and SHL Systemhouse. Dr. Karrer was valedictorian at Loyola Marymount University,

http://www.softwarebyrob.com/2011/08/24/lessons-learned-from-a-software-developers-first-attempt-at-launching-a-startup/

Software by Rob

Lessons Learned by a Serial Startup Founder

home about press micropreneurs archives

← Why Making Something Customers Want Isn't Enough

Ten Highly Successful Bootstrapped Startups →

Lessons Learned from a Software Developer's First Attempt at Launching a Startup

Twitter LinkedIn Google+ Facebook

If you're trying to grow your startup you've come to the right place. Get my 170-page ebook on how to grow a startup and join thousands of self-funded entrepreneurs by subscribing to my newsletter at right.

This is a guest article by Karl Falconer. Karl is a software engineer with more than 10 years of experience who specializes in agile web development and web services integrations. He authors a software development blog at http://www.falconerdevelopment.com/.

http://www.infoq.com/articles/what-do-we-know-about-software-development-in-startups

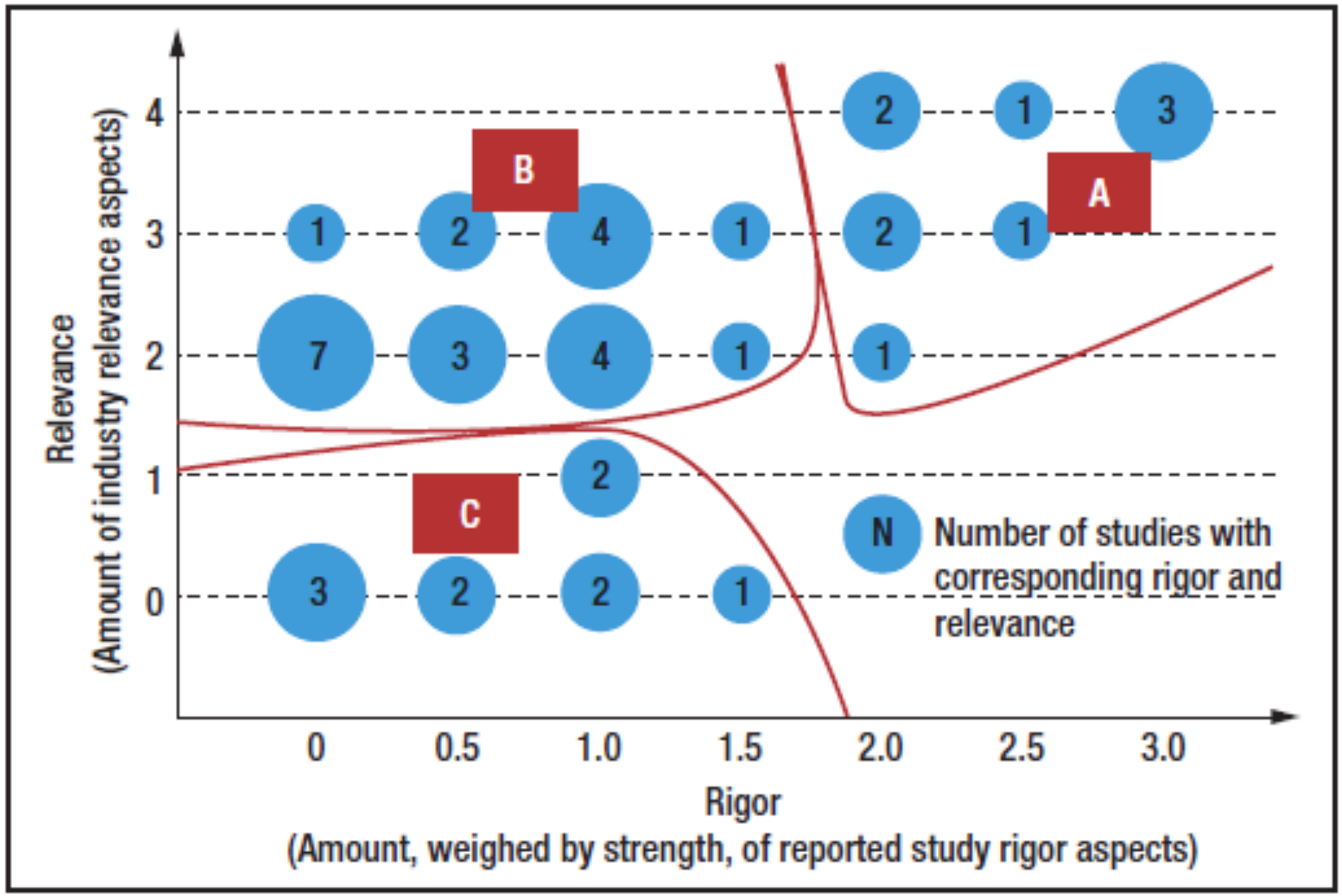
Startups are newly created companies with little or no history of facing high volatility in technologies and markets. In the US alone, 476,000 new businesses are established each month,1 accounting for nearly 20 percent of job creation.2 As such, startups are an important factor in the economy. However, the environment of startups is dynamic, unpredictable, and even chaotic, forcing entrepreneurs to act quickly, fail fast, and learn faster to fi nd a market niche and acquire a sustainable income. Sixty percent of startups don’t survive the fi rst fi ve years, and 75 percent of venture capital funded startups fail.3 Most of this is due to the high risk of startups, missed market windows, and other business reasons. To what extend engineering practices impact this high failure rate is still unknown given the premature state of research. We present a detailed investigation and collection of all known empirical software engineering sources related to startups and their engineering practices, as well as an analysis of how accurate and reliable this available evidence is.4 We see this as a first critical step into a largely unknown area—the world of software engineering practices in startups.

http://www.infoq.com/articles/what-do-we-know-about-software-development-in-startups

Theme	Description
Lack of resources	Economical, human, and physical resources are extremely limited.
Highly reactive	Startups are able to quickly react to changes in the underlying market, technologies, and product (compared to more established companies).
Innovation	Given the highly competitive ecosystem, startups need to focus on and explore highly innovative segments of the market.
Uncertainty	Startups deal with a highly uncertain ecosystem under different perspectives: market, product features, competition, people, and finance.
Rapidly evolving	Successful startups aim to grow and scale rapidly.
Time pressure	The environment often forces startups to release fast and to work under constant pressure (terms sheets, demo days, investors' requests).
Third-party dependency	Due to lack of resources, startups heavily rely on external solutions to build their product: external APIs, open source software, outsourcing, COTS, and so on.
Small team	Startups start with a small number of individuals.
One product	Company activities gravitate around one product/service only.
Low-experienced team	A good part of the development team is formed by people with less than five years of experience and often recently graduated students.
New company	The company has been recently created.
Full organization	Startups are usually founder-centric, and everyone in the company has big responsibilities, with no need for upper management.
Highly risky	The failure rate of startups is extremely high.
Not self-sustained	Especially in the early stage, startups need external funding to sustain their activities (venture capitalist, angel investments, personal funds, and so on).
Little working experience	The basis of an organizational culture isn't present initially.

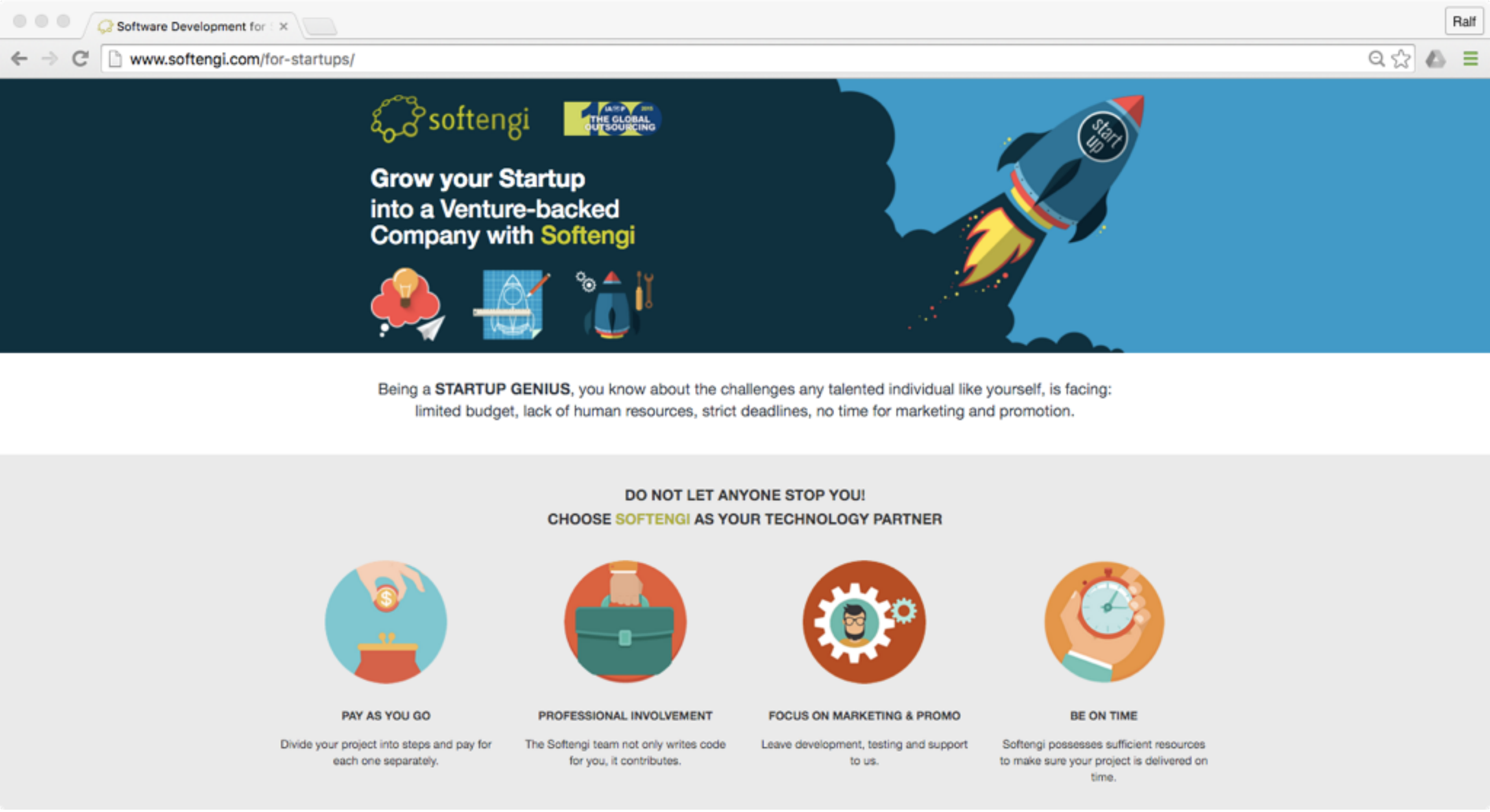
Does research understand how software development in startups? (Rather not.)

http://www.infoq.com/articles/what-do-we-know-about-software-development-in-startups

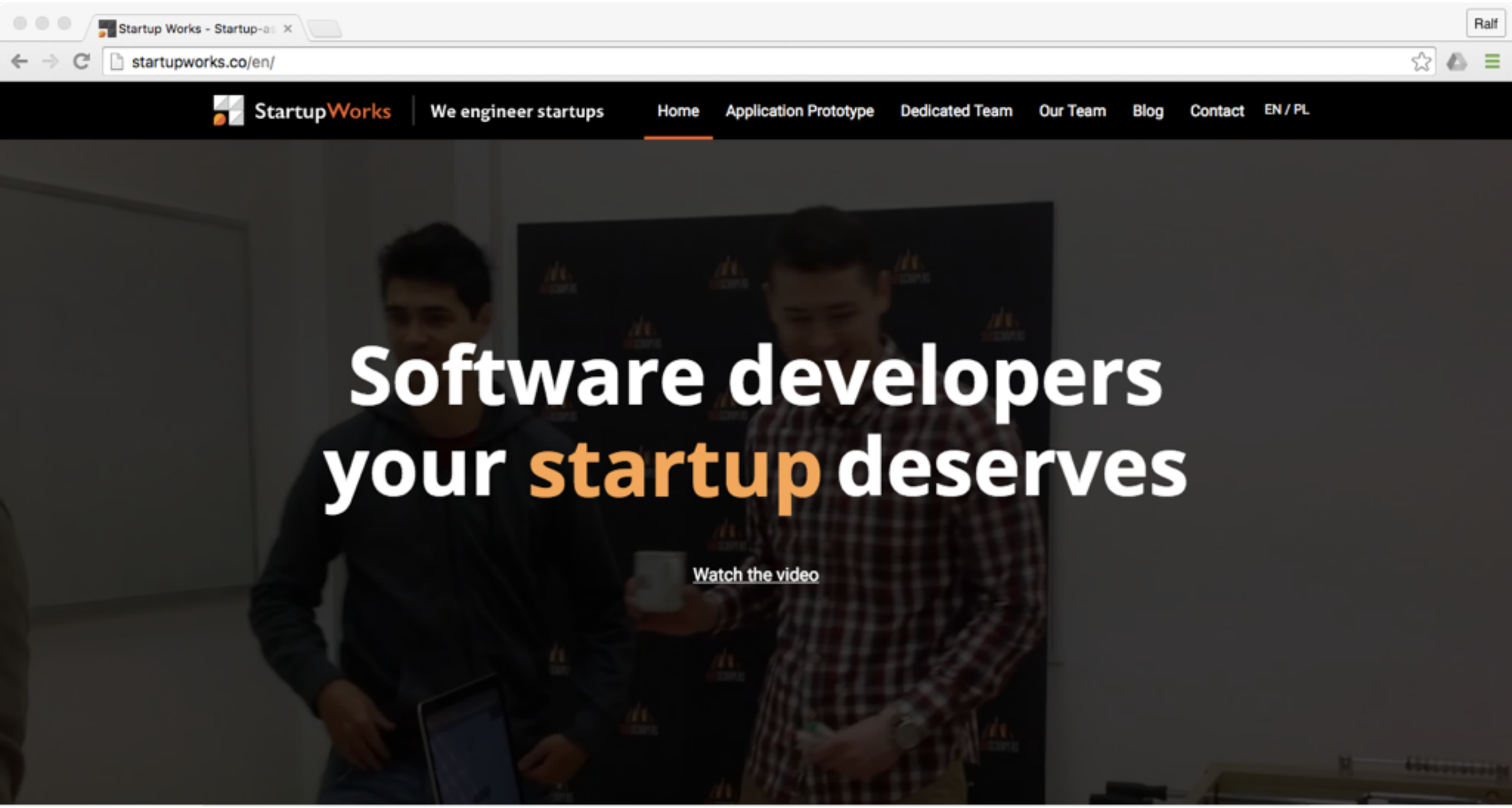


You could even depend on other companies to develop your software.

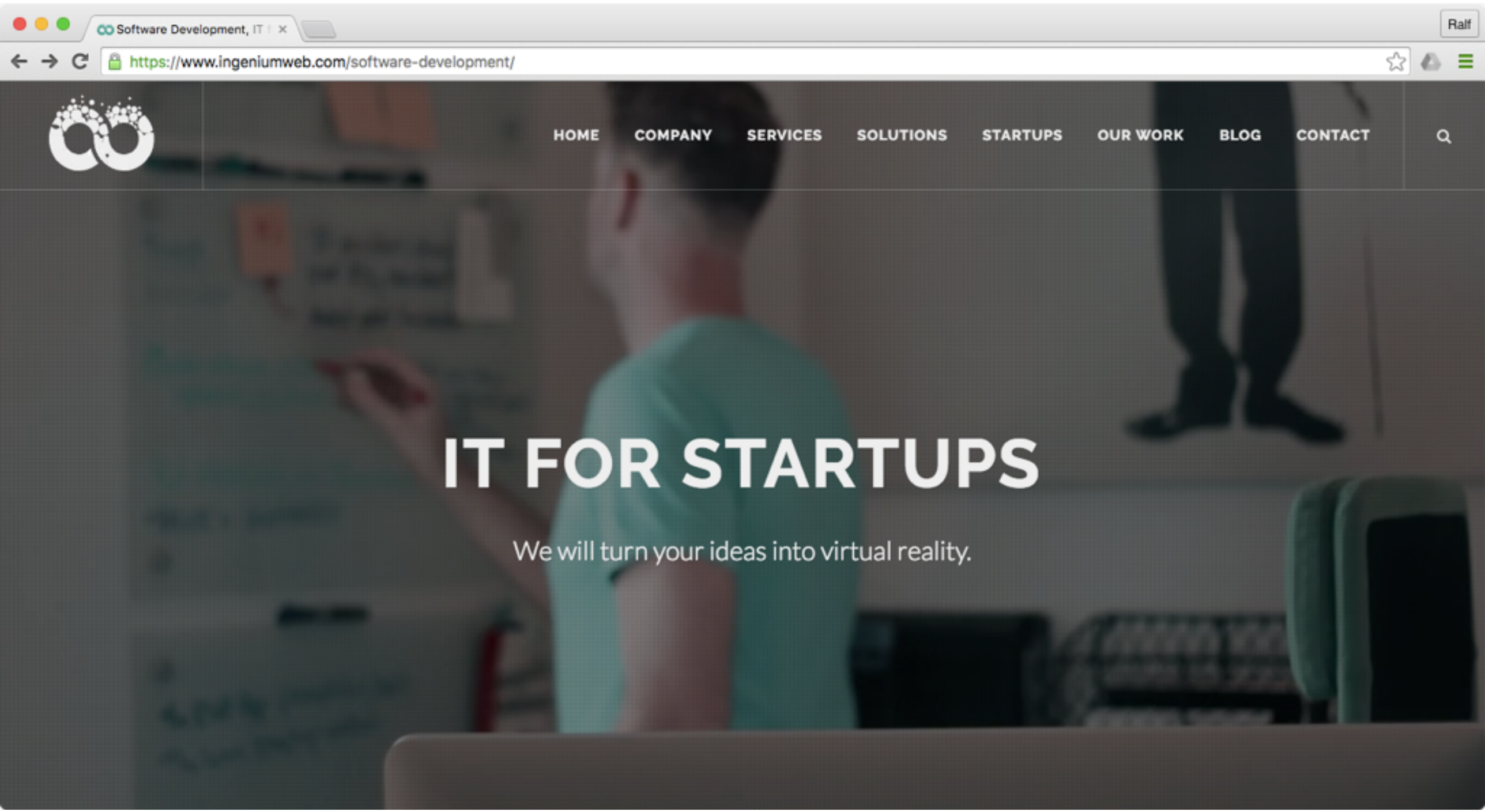
http://www.softengi.com/for-startups/



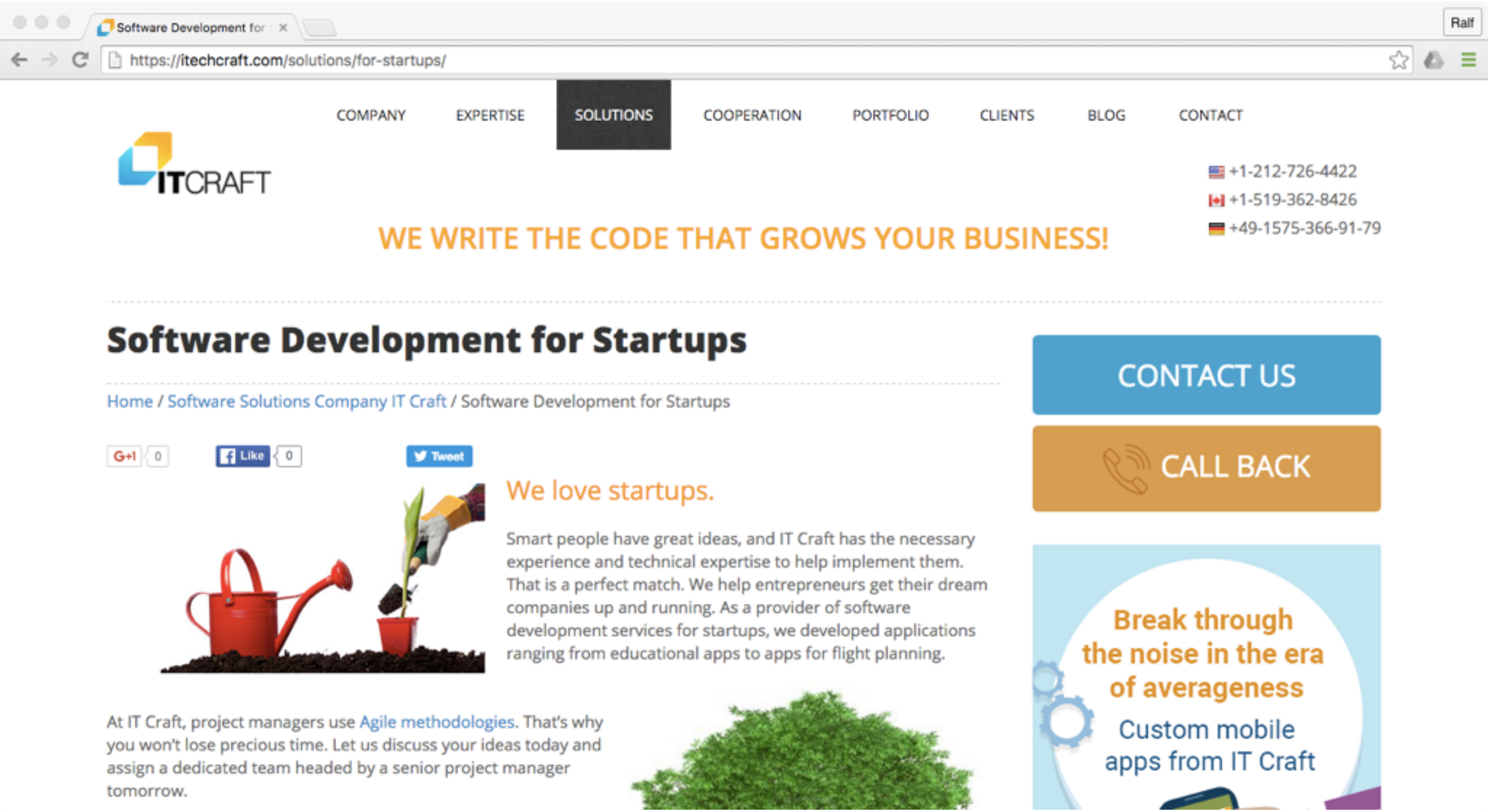
http://startupworks.co/en/



https://www.ingeniumweb.com/software-development/



https://itechcraft.com/solutions/for-startups/



Key concepts

- App stores
- **File hosting services**
- **Revision control systems**
- **Source code hosting facilities**
- **Software package platform**
- **Software configuration management**
- **Continuous integration**
- **Web API**
- **Social coding**
- **Lean software methodology**
- **Rapid application development**
- **Agile software development**
- **Extreme programming**
- Software design
- Software testing
- Software deployment
- Problem reporting and tracking
- Open source

File hosting service

https://en.wikipedia.org/wiki/File_hosting_service

A **file hosting service**, **cloud storage service**, **online file storage provider**, or **cyberlocker** is an Internet hosting service specifically designed to host user files. It allows users to upload files that could then be accessed over the internet from a different computer, tablet, smart phone or other networked device, by the same user or possibly by other users, after a password or other authentication is provided. Typically, the services allow HTTP access, and sometimes FTP access. Related services are content-displaying hosting services (i.e. video and image), virtual storage, and remote backup.

See also **file sync and sharing services**.

https://en.wikipedia.org/wiki/Comparison_of_file_hosting_services

Web host	Storage size	Max. file size	Direct access[1 1]	Traffic or bandwidth limit	File expiration[1 2]	Remote uploading?	Developer API?
Amazon Cloud Drive[2]	5 GB 3mo. free trial, unlimited GB paid	2 GB upload via Web, 10 GB download	Yes	Amazon S3 limits	90days after subscription expires data is deleted.	No	Yes
Amazon S3[3]	5 GB 12-month free trial with credit-card, paid bandwidth, Unlimited paid	5 GB per file, unlimited files per bucket	Yes	Amazon S3 limits	None, pay for storage each month	No	REST, SOAP
Asus WebStorage[4]	5 GB free, Up to 5 TB paid	500 MB free, 1–2 GB paid	No[5]	500 MB per 30 min free, 1 to 8 TB per month paid[6][7]	?	No	No
Baidu Cloud	2 TB (6 GB) free[8]	20 GB Windows paid, 4 GB Mac & free	Yes	None	None	Yes	Yes
Bitcasa	5-20 GB free, 1TB and 10TB Paid Plans	None through desktop client.[9]	Yes	None	None	Yes	Yes

. . .

Revision control systems

https://en.wikipedia.org/wiki/Revision_Control_System

The **Revision Control System** (**RCS**) is a software implementation of revision control that automates the storing, retrieval, logging, identification, and merging of revisions. RCS is useful for text that is revised frequently, for example programs, documentation, procedural graphics, papers, and form letters. RCS is also capable of handling binary files, though with reduced efficiency. Revisions are stored with the aid of the diff utility.

Comparison of version control software

https://en.wikipedia.org/wiki/Comparison_of_version_control_software

- CVS
- SVN
- GIT
- …

Source code hosting facility

https://en.wikipedia.org/wiki/Comparison_of_source_code_hosting_facilities

A **source code repository** is a file archive and web hosting facility where large amounts of source code for software, but also for web pages are kept, either publicly or privately. They are often used by open-source projects and other multi-developer projects to handle various versions. They help developers submit patches of code in an organized fashion. Often these web sites support version control, bug tracking, release management, mailing lists, and wiki-based documentation.

Source code hosting facility

https://en.wikipedia.org/wiki/Comparison_of_source_code_hosting_facilities

Name	Manager	Established	Server side: all free software	Client side: all-free JS code	Developed and/or used CDE	Require free software on registration	Ad-free	Cost	Notes
Alioth	Debian Project	2003	Yes	Yes	FusionForge	Unknown	Yes	Free	Preference for Debian related projects
Assembla	Assembla, Inc	2005	No	Unknown	Unknown	No	Yes	7-day free trial. Commercial projects with customizable sets of tools and features.	
Betavine	Vodafone	2007	No	Unknown	Unknown	No	No	Free	
Bitbucket	Atlassian	2008	No	No	Unknown	No	Yes	Free private repositories are limited to 5 users	

Software package platform

https://en.wikipedia.org/wiki/Software_repository

Careful: Software repository is a very ambiguous term.

Overview [edit]

Many software publishers and other organizations maintain servers on the Internet for this purpose, either free of charge or for a subscription fee. Repositories may be solely for particular programs, such as CPAN for the Perl programming language, or for an entire operating system. Operators of such repositories typically provide a package management system, tools intended to search for, install and otherwise manipulate software packages from the repositories. For example, many Linux distributions use Advanced Packaging Tool (APT), commonly found in Debian based distributions, or yum found in Red Hat based distributions. There are also multiple independent package management systems, such as pacman, used in Arch Linux and equo, found in Sabayon Linux.

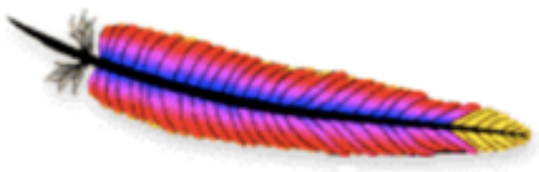
As software repositories are designed to include useful packages, major repositories are designed to be malware free. If a computer is configured to use a digitally signed repository from a reputable vendor, and is coupled with an appropriate permissions system, this significantly reduces the threat of malware to these systems. As a side effect, many systems that have these capabilities do not require anti-malware software such as anti-virus software.[1]

Most major Linux distributions have many repositories around the world that mirror the main repository.

https://en.wikipedia.org/wiki/Software_repository

Language / purpose	Package Development Process	Repository	How to install	Collaborative development platform
C++		Boost		
Haskell	Common Architecture for Building Applications and Libraries (CABAL)	Hackage	[1]	
Java		Maven	[2]	
.NET	NuGet	NuGet	[3]	
Node.js		NPM	[4]	
Perl		CPAN	PPM[5]	
PHP	PEAR	PECL		
Python	Setuptools	PyPI	pip, EasyInstall, PyPM	
R	R CMD check process[2][3]	CRAN	install.packages	R-Forge
		Bioconductor	BiocLite.R	
Ruby	RubyGems	Ruby Application Archive		RubyForge
TeX, LaTeX		CTAN		

https://commons.apache.org/



Apache Commons™
http://commons.apache.org/

Apache Commons ™ Last Published: 06 April 2016 | Version: 15

Components Sandbox Dormant ApacheCon Apache

COMMONS
- Home
- License
- Mailing Lists
- PMC
- Components
- Sandbox
- Dormant

RELEASES

Welcome to Apache Commons

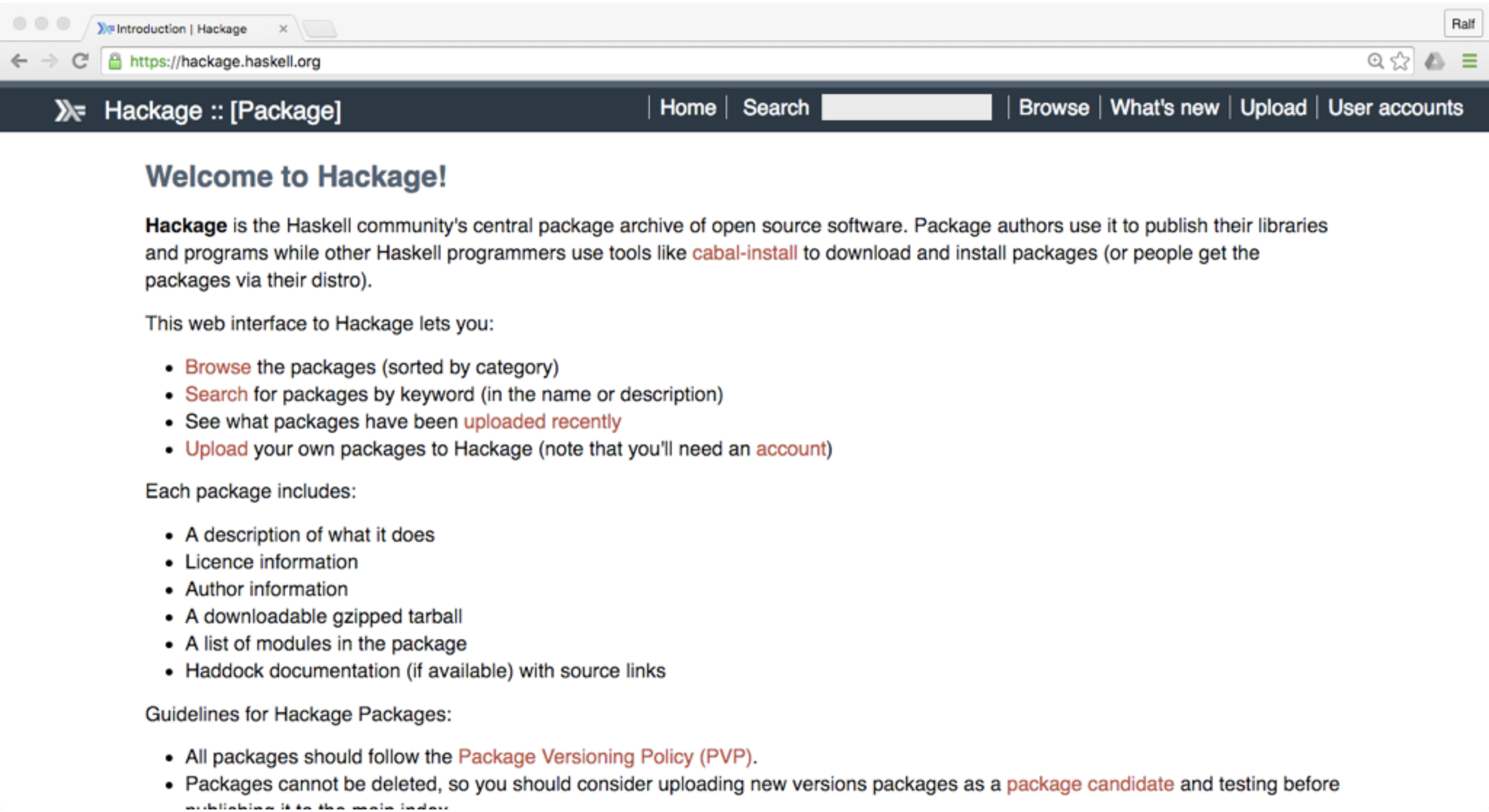
Apache Commons is an Apache project focused on all aspects of reusable Java components.

The Apache Commons project is composed of three parts:

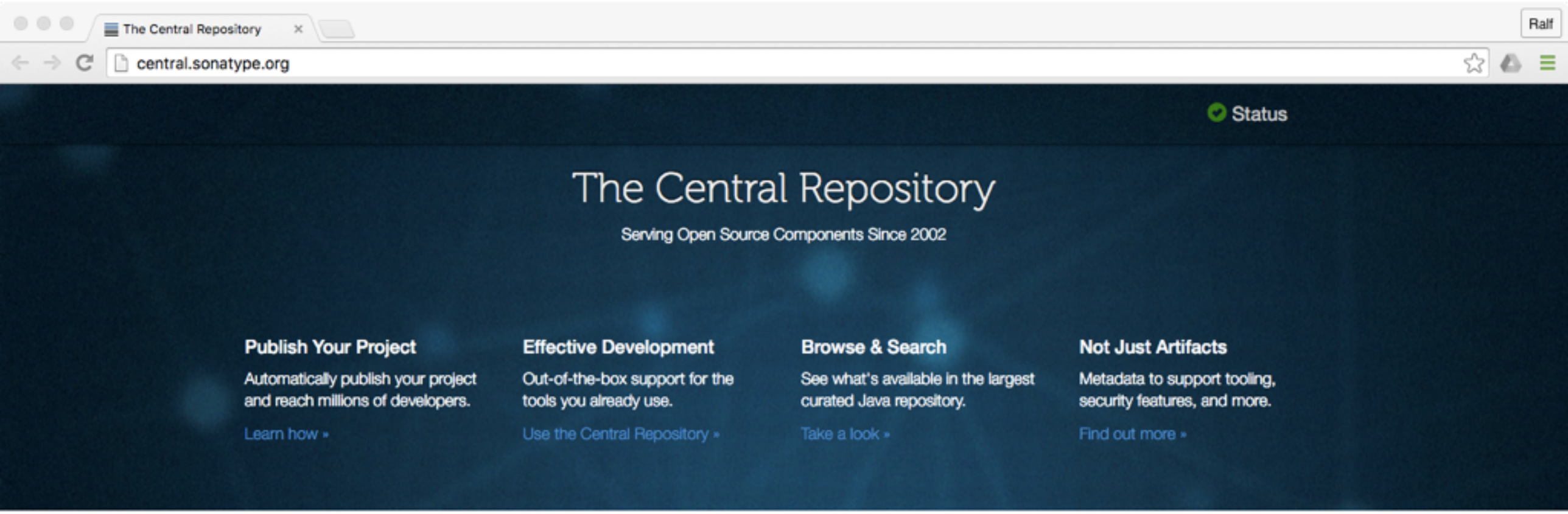
- The Commons Proper - A repository of reusable Java components.
- The Commons Sandbox - A workspace for Java component development.
- The Commons Dormant - A repository of components that are currently inactive.

You may also read our charter, which spells out the goals of the project in greater detail.

https://hackage.haskell.org/



http://search.maven.org/



Quick Links

FAQ

Support

Search

News & Updates

Free Video Series - Easy Publishing to the Central Repository

Tue 02 February 2016

Why Use Central?

Ease of Use

The Central Repository is the the default repository for Apache Maven, SBT and other build systems and can be easily used from Apache Ant/Ivy, Gradle and many other tools.

Ubiquitous

Open source organizations such as the Apache Software Foundation, the Eclipse

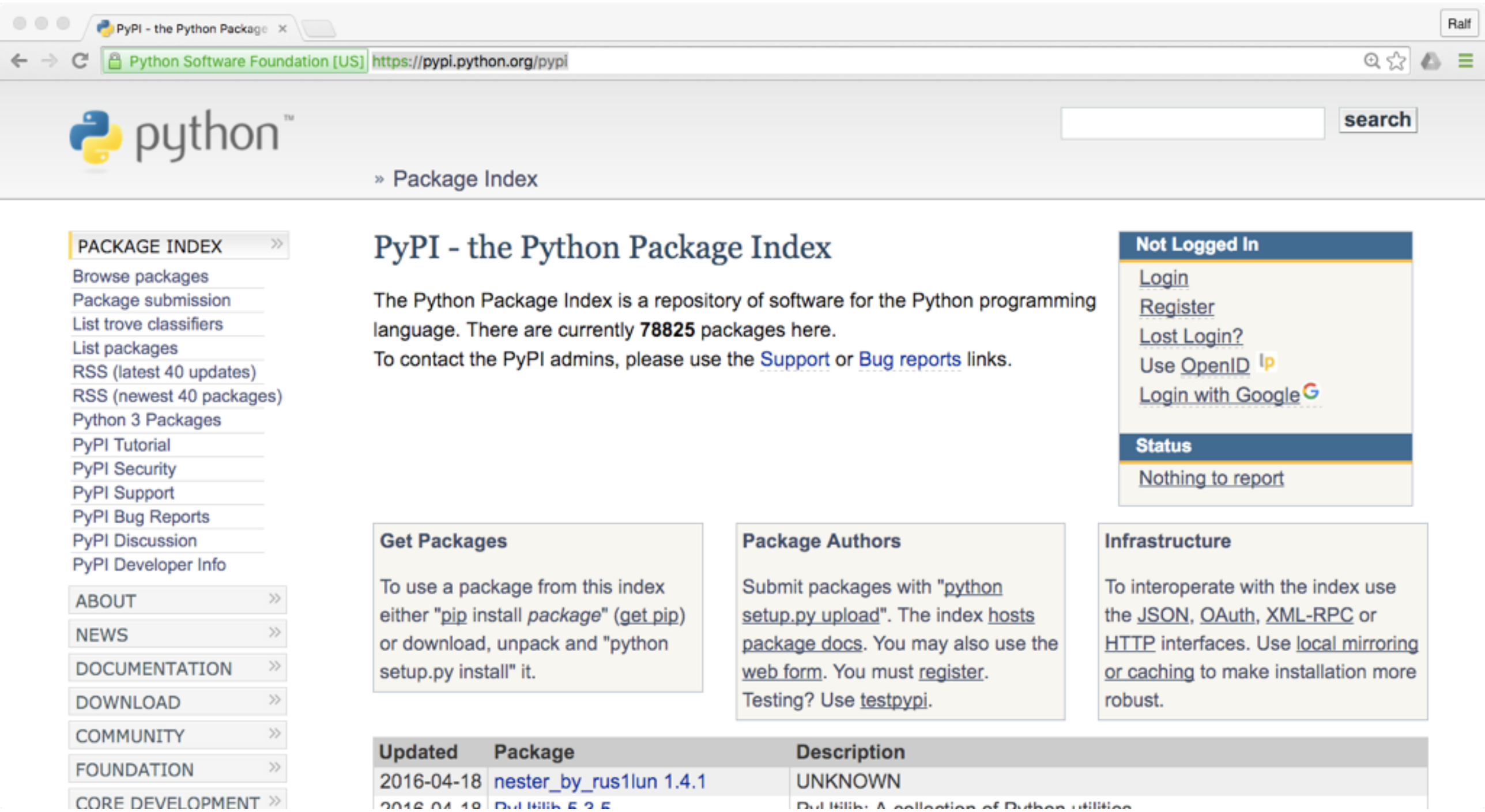
And Publishing is *Easy*!

Requirements

There are some minimal requirements for publishing your component to the Central Repository. These are things that the community has insisted upon: in short, the basic elements of quality *metadata* that developers rely on.

Register Your Project

https://pypi.python.org/pypi



Software configuration management

https://en.wikipedia.org/wiki/Software_configuration_management

In software engineering, **software configuration management** (**SCM** or **S/W CM**)[1] is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management.[2] SCM practices include revision control and the establishment of baselines. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

Comparison of open-source configuration management software

https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software

	Language	License	Mutual auth	Encrypts	Verify mode	Agent-less	Have a GUI
Ansible	Python	GPL	Yes[1]	Yes[2]	Yes	Yes	Yes[3] (Free 30-day Trial)
Bcfg2	Python	BSD[5]	Yes[6]	Yes[7]	Yes[8]	No	Yes[9]
BundleWrap	Python	GPL	Yes[1]	Yes[2]	Yes	Yes	No
Capistrano	Ruby	MIT License		Yes[2]			
cdist	Python	GPL	Yes[1]	Yes[2]		Yes	
Chef	Ruby, Erlang	Apache	Yes[12]	Yes[13]	Yes[14][15]	No	Yes

Continuous integration

https://en.wikipedia.org/wiki/Continuous_integration

Continuous integration (**CI**) is the practice, in software engineering, of merging all developer working copies to a shared mainline several times a day. It was first named and proposed by Grady Booch in his 1991 method,[1] although Booch did not advocate integrating several times a day. It was adopted as part of extreme programming (XP), which did advocate integrating more than once per day, perhaps as many as tens of times per day.

Comparison of continuous integration software

https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software

Name	Platform	License	Windows builders	Java builders	Other builders	Notification	IDE Integration	Other Integration
AnthillPro	Cross-platform	Proprietary	MSBuild, NAnt, Visual Studio	Ant, Maven 1, 2, & 3	Shell script, Batch script, Cross-platform command-line, Groovy, Make, RTC Jazz, TFS Build, Custom Script Interpreter	E-mail, XMPP/Jabber, RSS, Systray	Eclipse, Visual Studio	(many)
Apache Continuum	JDK, Servlet Container	Apache 2.0	Unknown	Maven 1 & 2 & 3	Shell script[1]	Mail, Jabber and Google Talk, MSN, IRC, report deployment with wagon	Unknown	Unknown
Apache Gump	Python	Apache 2.0	Unknown	Ant, Maven 1	Unknown	E-mail	Unknown	Unknown
AppVeyor CI	Hosted	Proprietary	Visual Studio, MSBuild, Psake	No	Custom Script, PowerShell	E-mail, HipChat, Slack, Catlight	No	GitHub, Bitbucket, Kiln, Windows Azure

...

Web API

https://en.wikipedia.org/wiki/Web_API

A **web API** is an application programming interface (API) for either a web server or a web browser. It is a web development concept, usually limited to a web application's client-side (including any web frameworks being used), and thus usually does not include web server or browser implementation details such as SAPIs or web browser engine APIs unless publicly accessible by a remote web application.

Popular Web APIs

http://www.programmableweb.com/news/most-popular-apis-least-one-will-surprise-you/2014/01/23

Most Popular APIs: At Least One Will Surprise You

News, News Services, API, Directories, Mapping, Mashups, Social, Travel, Video, Weather

Jan. 23 2014 By Adam DuVander

There are a number of ways to discuss API popularity. One of the common methods *ProgrammableWeb* has used is by mashups, the number of completed apps. However, there may be a leading indicator before developers have even started writing code. The "track" functionality on *ProgrammableWeb* lets developers declare an interest in receiving updates on particular APIs. By diving into this data we can see many things. For example, recently developers have loved travel. Overall, social and visual APIs rule.

Related: Top 10 Weather APIs

Top 10 tracked APIs of all time

1. Facebook

Social coding

GitHub's CEO, Tom Preston-Werner

(Source: https://vsanywhere.com/web/what-is-social-coding/)

We like the ideas of social networking. We think that developers work more effectively when they work together. So let's take the ideas of a social network and add on top of that code hosting, and let's create a site that makes it easy to share and collaborate on code.

Margaret Rouse's definition:

(Source: http://whatis.techtarget.com/definition/social-coding)

DEFINITION

social coding

Part of the Agile, Scrum, XP glossary:

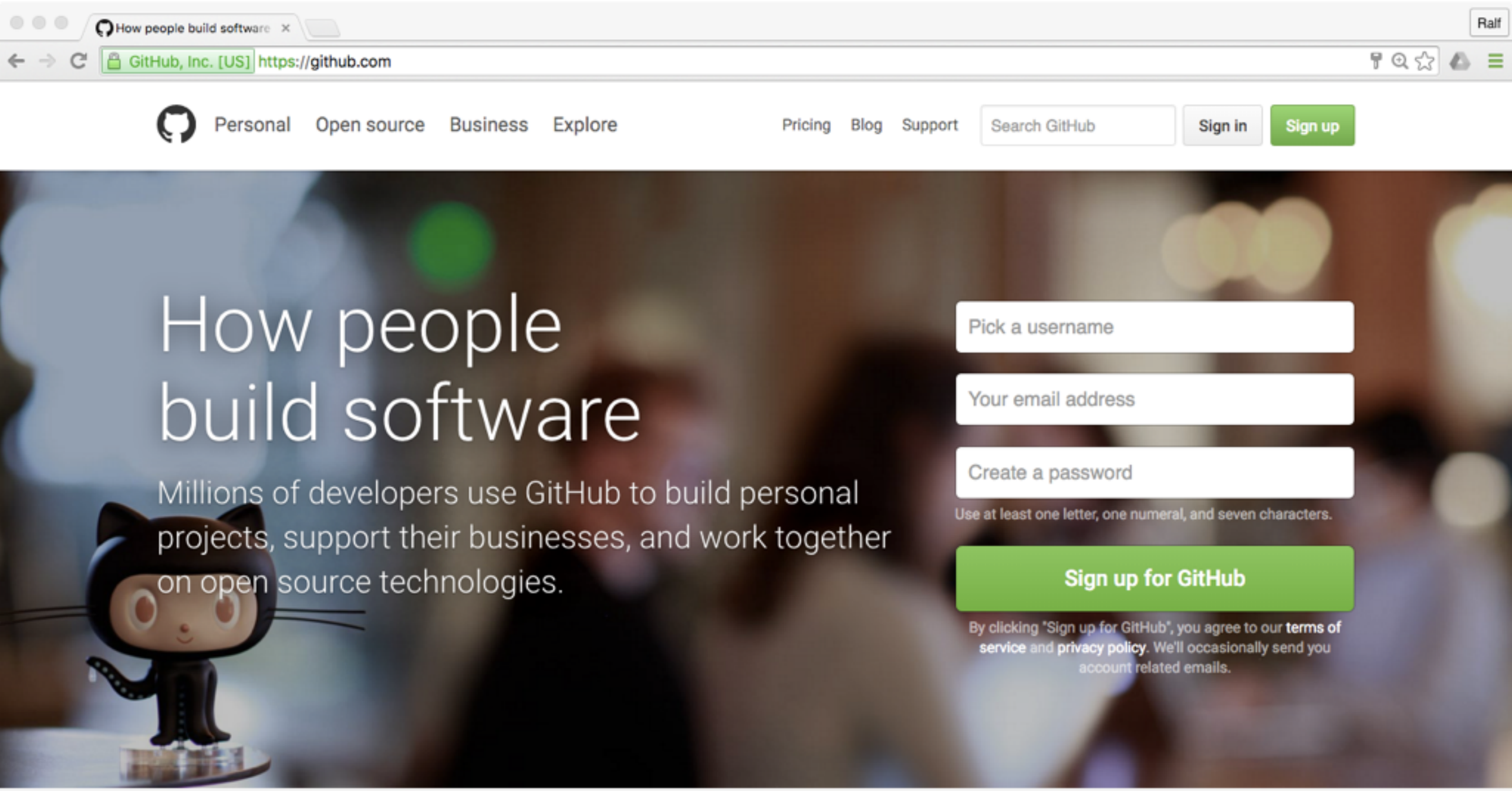
Social coding is an approach to software development that places an emphasis on formal and informal collaboration.

Although the term is often associated with social coding websites such as GitHub, BitBucket, CodePlex and Google Code, the term can be used to describe any development environment that encourages discussion and sharing.

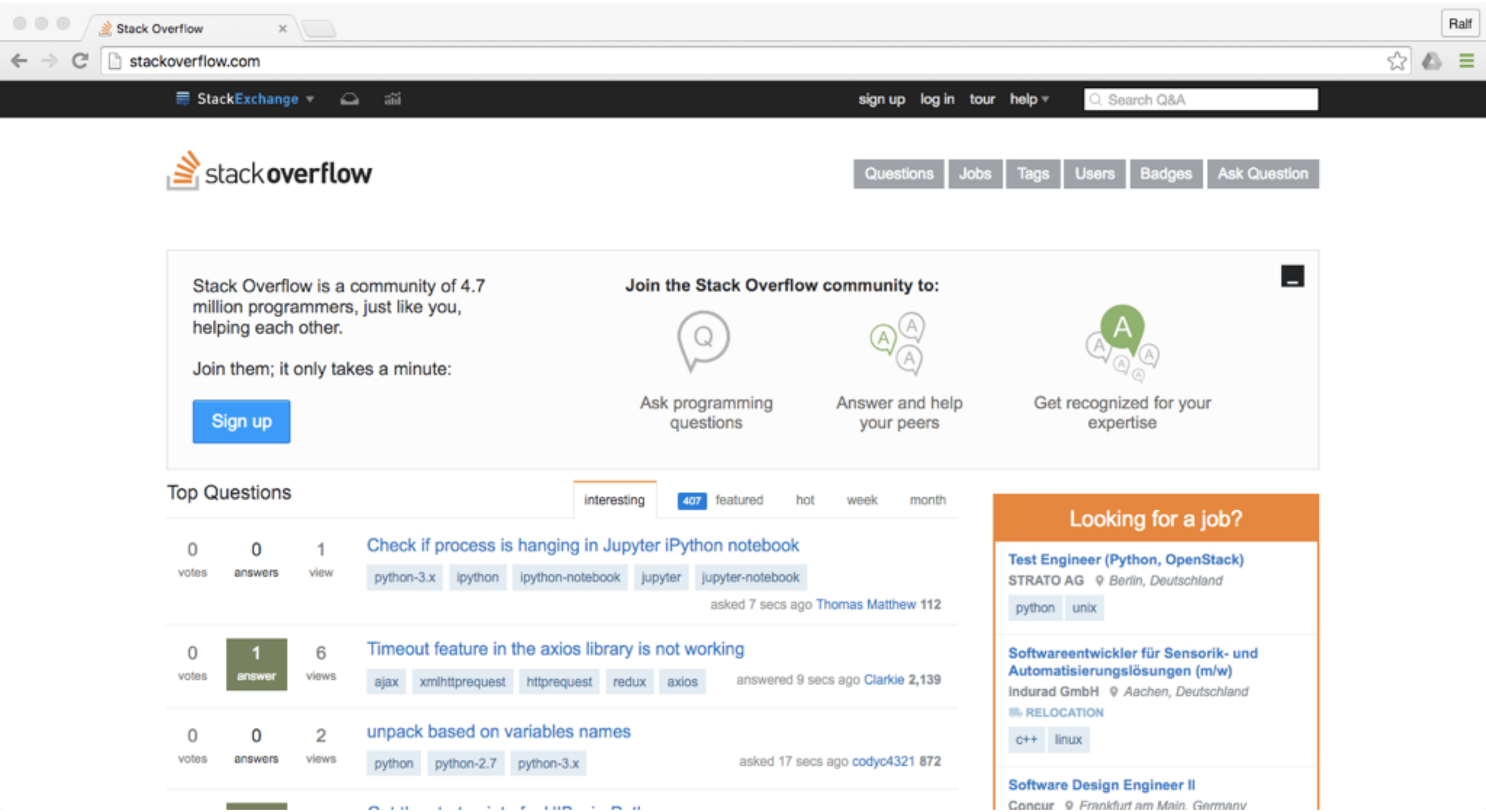
This was last updated in June 2012

Posted by: Margaret Rouse

https://github.com/

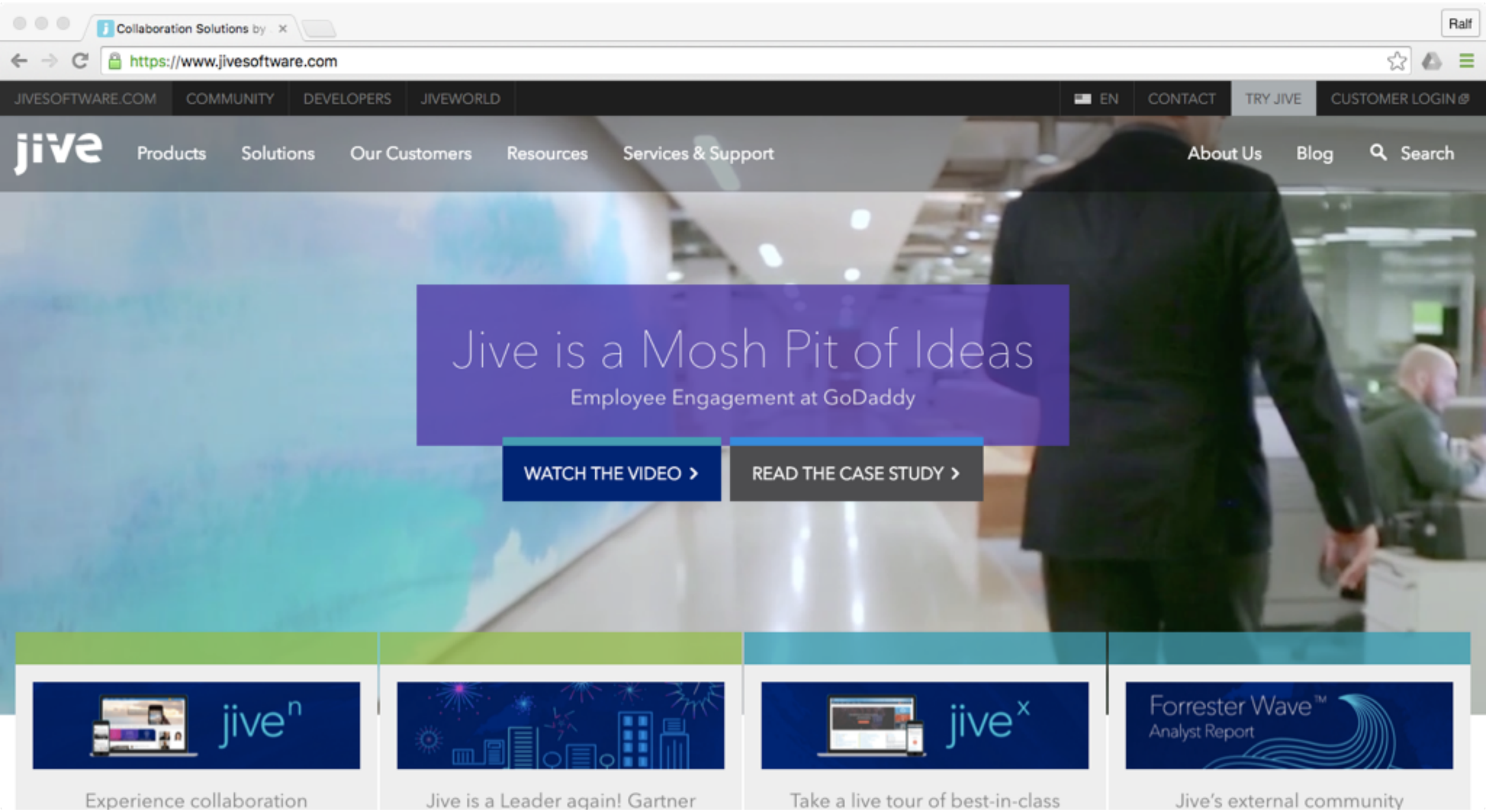


http://stackoverflow.com/



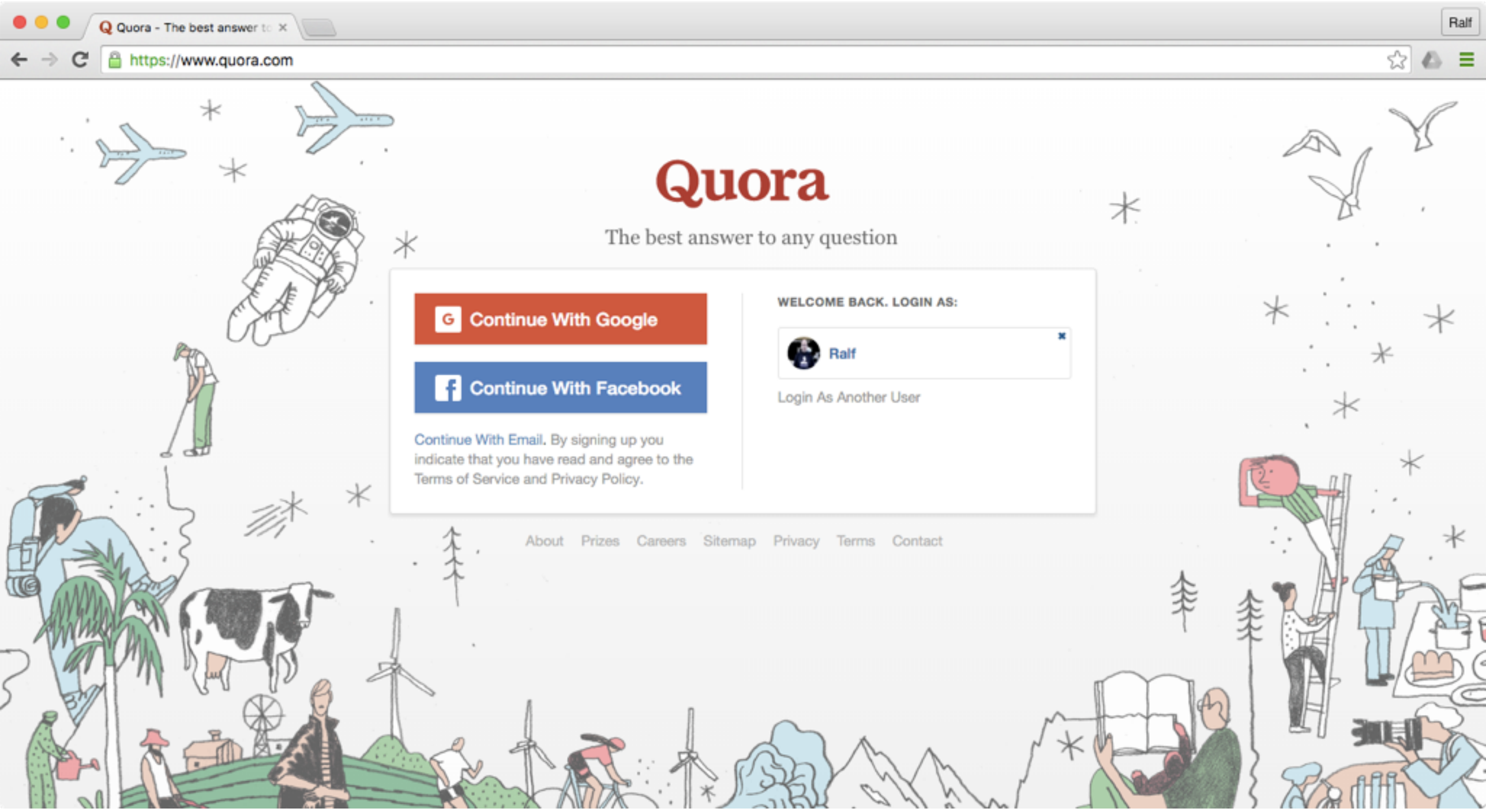
Jive

https://www.jivesoftware.com/



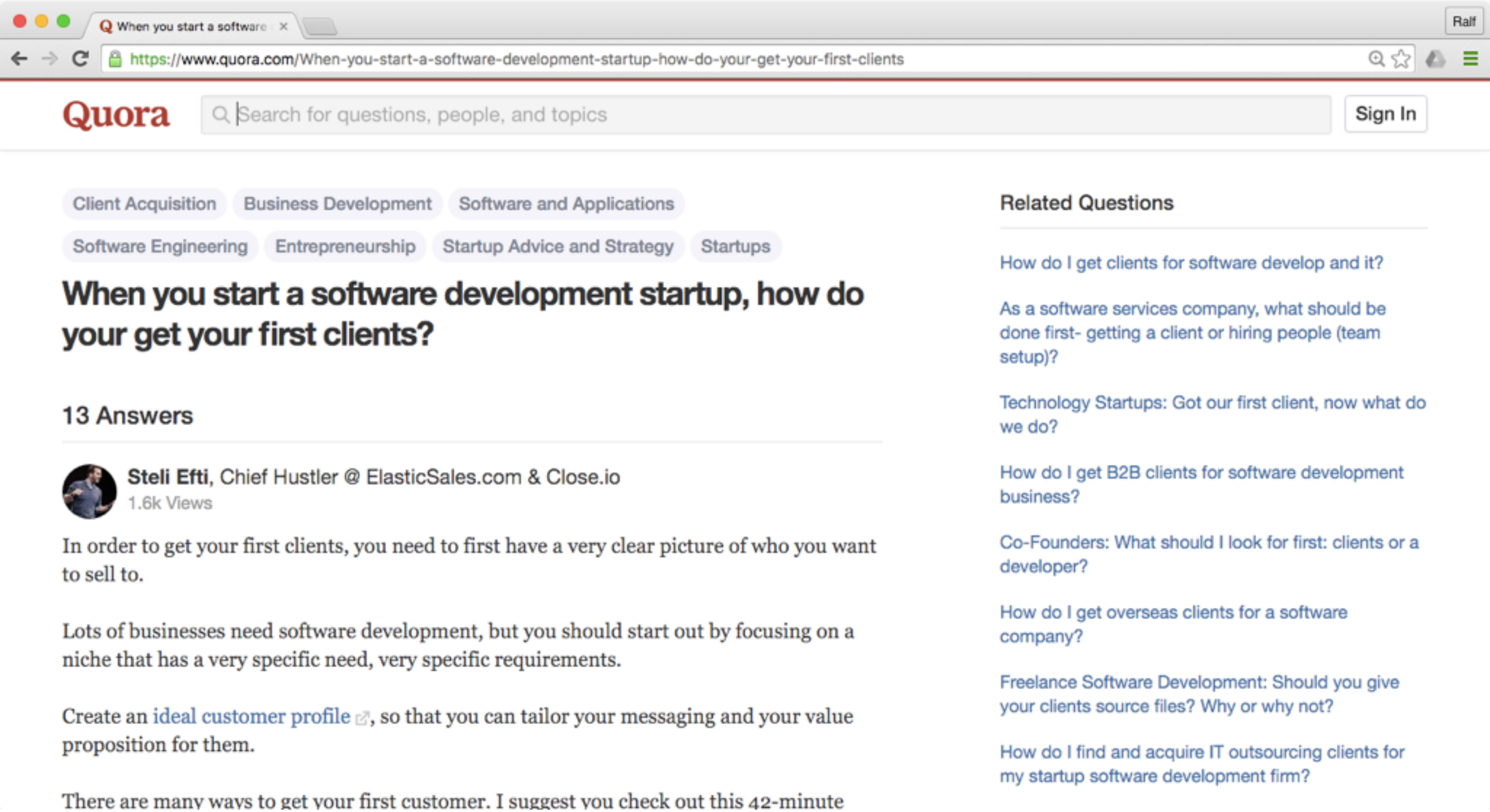
Other knowledge resources

https://www.quora.com/



For instance:

https://www.quora.com/When-you-start-a-software-development-startup-how-do-your-get-your-first-clients



Wikipedia

For instance: https://en.wikipedia.org/wiki/Java_(programming_language)

Java (programming language)

From Wikipedia, the free encyclopedia

"Java language" redirects here. For the natural language from the Indonesian island of Java, see Javanese language.
This article is about a programming language. For the software package downloaded from java.com, see Java SE.
Not to be confused with JavaScript.

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented,[13] and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA),[14] meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.[15] Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use,[16][17][18][19] particularly for client-server web applications, with a reported 9 million developers.[*citation needed*] Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

Java

Dbpedia

For instance: http://dbpedia.org/page/Java_(programming_language)

DBpedia Browse using Formats Faceted Browser Sparql Endpoint

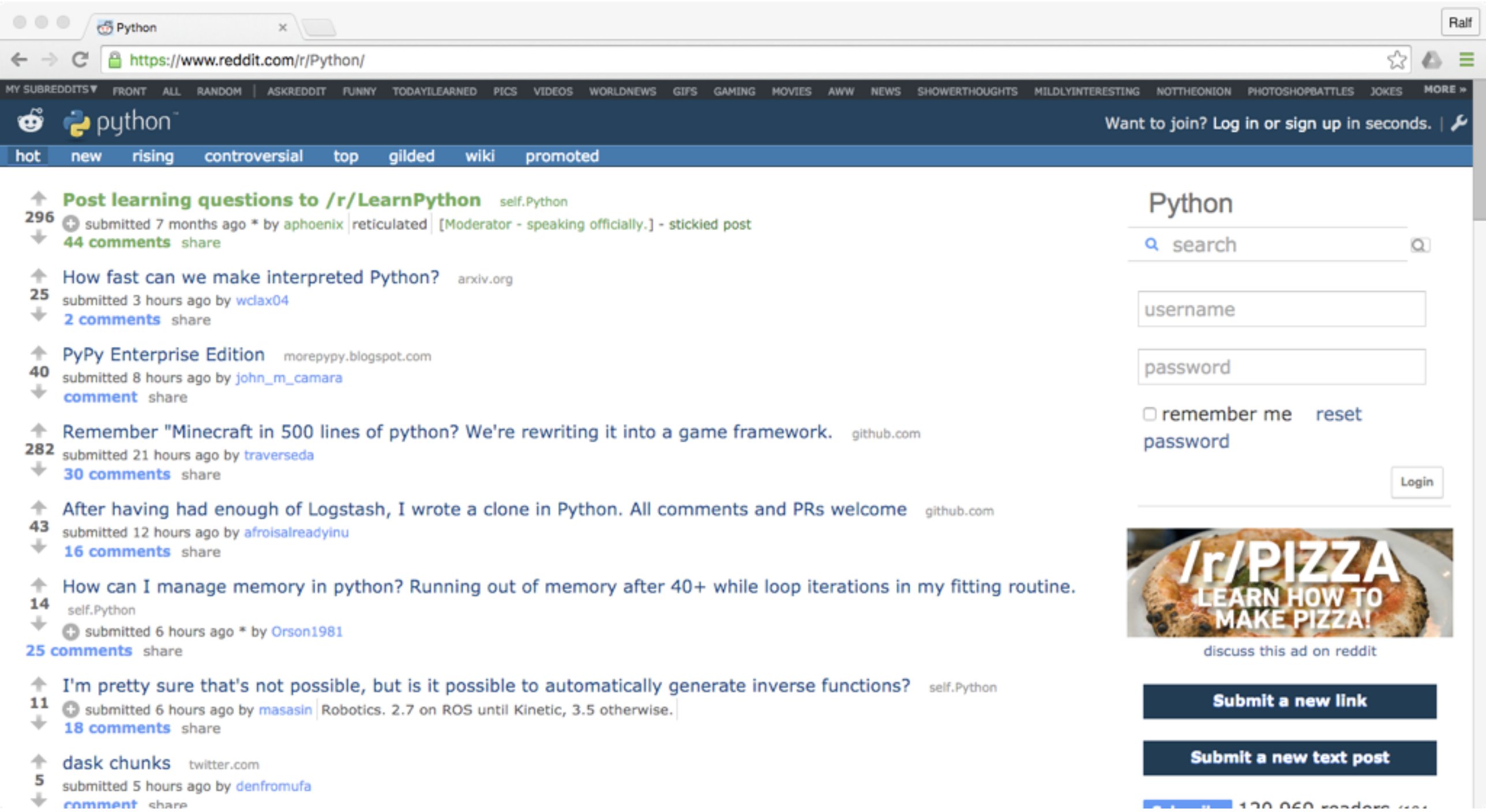
About: Java (programming language)

An Entity of Type : software, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

reddit

https://www.reddit.com/



Social coding science

Source: http://arxiv.org/abs/1408.6012

Computer Science > Social and Information Networks

Collaboration on Social Media: Analyzing Successful Projects on Social Coding

Yuya Yoshikawa, Tomoharu Iwata, Hiroshi Sawada

(Submitted on 26 Aug 2014 (v1), last revised 4 Sep 2014 (this version, v2))

Social Coding Sites (SCSs) are social media services for sharing software development projects on the Web, and many open source projects are currently being developed on SCSs. One of the characteristics of SCSs is that they provide a platform on social networks that encourages collaboration between developers with the same interests and purpose. For example, external developers can easily report bugs and improvements to the project members. In this paper, we investigate keys to the success of projects on SCSs based on large data consisting of more than three hundred thousand projects. We focus on the following three perspectives: 1) the team structure, 2) social activity with external developers, and 3) content developed by the project. To evaluate the success quantitatively, we define activity, popularity and sociality as success indexes. A summary of the findings we obtained by using the techniques of correlation analysis, social network analysis and topic extraction is as follows: the number of project members and the connectivity between the members are positively correlated with success indexes. Second, projects that faithfully tackle change requests from external developers are more likely to be successful. Third, the success indexes differ between topics of softwares developed by projects. Our analysis suggests how to be successful in various projects, not limited to social coding.

Eirini Kalliamvakou, Daniela E. Damian, Kelly Blincoe, Leif Singer, Daniel M. Germán: **Open Source-Style Collaborative Development Practices in Commercial Projects Using GitHub**. ICSE (1) 2015: 574-585

Müller Miranda, Renato Ferreira, Cleidson R. B. de Souza, Fernando Marques Figueira Filho, Leif Singer: **An exploratory study of the adoption of mobile development platforms by software engineers**. MOBILESoft 2014: 50-53

Leif Singer, Fernando Marques Figueira Filho, Margaret-Anne D. Storey: **Software engineering at the speed of light: how developers stay current using twitter**. ICSE 2014: 211-221

Andrea Capiluppi, Alexander Serebrenik, Leif Singer: **Assessing Technical Candidates on the Social Web**. IEEE Software 30(1): 45-51 (2013)

Brendan Cleary, Carlos Gómez, Margaret-Anne D. Storey, Leif Singer, Christoph Treude: **Analyzing the friendliness of exchanges in an online software developer community**. CHASE@ICSE 2013: 159-160

Bin Lin, Alexey Zagalsky, Margaret-Anne D. Storey, Alexander Serebrenik: **Why Developers Are Slacking Off: Understanding How Software Teams Use Slack**. CSCW Companion 2016: 333-336

Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G. J. van den Brand, Alexander Serebrenik, Premkumar T. Devanbu, Vladimir Filkov: **Gender and Tenure Diversity in GitHub Teams**. CHI 2015: 3789-3798

Bogdan Vasilescu, Alexander Serebrenik, Premkumar T. Devanbu, Vladimir Filkov: **How social Q&A sites are changing knowledge sharing in open source software communities**. CSCW 2014: 342-354

Bogdan Vasilescu, Alexander Serebrenik, Mark G. J. van den Brand: **The Babel of Software Development: Linguistic Diversity in Open Source**. SocInfo 2013: 391-404

Lean software development

https://en.wikipedia.org/wiki/Lean_software_development

Principles

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. See the whole

Rapid application development

https://en.wikipedia.org/wiki/Rapid_application_development

Rapid application development (**RAD**) is both a general term used to refer to alternatives to the conventional waterfall model of software development as well as the name for James Martin's approach to rapid development. In general, RAD approaches to software development put less emphasis on planning tasks and more emphasis on development. In contrast to the waterfall model, which emphasizes rigorous specification and planning, RAD approaches emphasize the necessity of adjusting requirements in reaction to knowledge gained as the project progresses. This causes RAD to use prototypes in addition to or even sometimes in place of design specifications. RAD approaches also emphasize a flexible process that can adapt as the project evolves rather than rigorously defining specifications and plans correctly from the start. In addition to James Martin's RAD method, other approaches to rapid development include Agile methods and the spiral model. RAD is especially well suited (although not limited to) developing software that is driven by user interface requirements. Graphical user interface builders are often called rapid application development tools.

Agile software development

https://en.wikipedia.org/wiki/Agile_software_development

The Agile Manifesto

In February 2001, 17 software developers[12] met at the Snowbird resort in Utah to discuss lightweight development methods. They published the *Manifesto for Agile Software Development*,[3] in which they said that by "uncovering better ways of developing software by doing it and helping others do it," they have come to value *Individuals and interactions* over Processes and tools, *Working software* over Comprehensive documentation, *Customer collaboration* over Contract negotiation, and *Responding to change* over Following a plan.

- *Individuals and interactions*: self-organization and motivation are important, as are interactions like co-location and pair programming.
- *Working software*: working software is more useful and welcome than just presenting documents to clients in meetings.
- *Customer collaboration*: requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- *Responding to change*: agile methods are focused on quick responses to change and continuous development.[13]

Extreme programming

https://en.wikipedia.org/wiki/Extreme_programming

Extreme programming practices

https://en.wikipedia.org/wiki/Extreme_programming_practices

The End