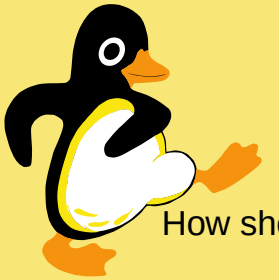


Collaborative design

How should collaboration platforms look like?

Dr. Ralf Schlatterbeck
Open Source Consulting

Email: office@runtux.com
Web: http://www.runtux.com
Tel. +43/650/621 40 17

Contents

Contents

Motivation

- I'm Open Source Software developer
- you make money by selling services – not licenses
- enough to live but not enough to kill
- example: Linux kernel ISDN-driver problem
  - customer had difficult problem where machine would die after some days or weeks – critical application
  - customer payed me to find and fix the bug (without a guarantee from my side)
  - bug was found and fixed, solution is available to *all* users of the kernel

Focus

- I’m focusing on reusable data

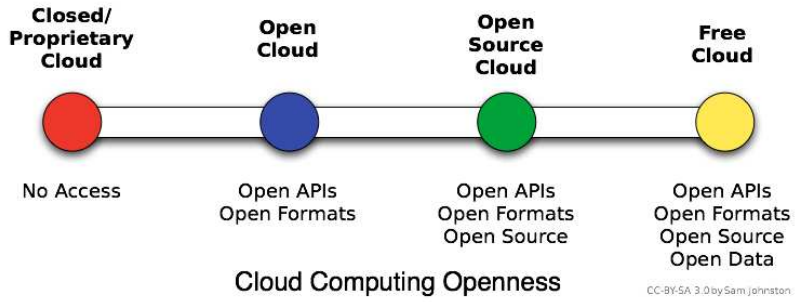
→ so I’m looking at the machine-machine interface, not the HMI

- This needs free open source licenses for content
- we need to get our content back – at least
- . . . but most collaboration platforms are closed – in the sense that you won’t get your data back

→ peer-to-peer social software: see my blog entry Cloud computing, Vendor Lock-In and the Future

- We’re concerned with API’s, Data formats, Source Code and Data

Degrees of Openness



Degrees of Openness from Open Cloud Initiative

Since “Cloud Computing” is just a new name for “Software as a Service” this is what applies to all the web applications out there

Meshups

- Interlinking content of several websites into a new application
- A new approach of distributed (P2P) applications
- Decentralisation: Let everybody do what they can do best
- Resilience: Due to machine interfaces we can easily mirror a site
- No dependency on big provider
- Needs Open Data or negotiations with each website owner

What’s Possible With Meshups

- thingiverse: developers, things (made by developers), tools (used by developers)
- might meshup with other sites focusing on the social side of people and use of things
- Andrius Kulikauskas organizes people around their values and looks at solutions that have worked for people
- A meshup could create more context: A tool has successfully been used in a certain context
- We really *need* documentation on what tools work in which environments!

Disclaimer: I haven’t looked at the license of Thingiverse

Invitation: List Your Web Applications

. . . and look at their Cloud Computing Openness

- Skype: Closed protocol, scary
- Facebook: People have been thrown off Facebook for getting their data
- Xing: Has a sync interface – for paying customers
- Ning: I haven't found data export, have you?
- Sourceforge: Data is open but no API
- Python Package Index (PyPI) no central storage like Sourceforge, API for up/download, install . . .
- Roundup Issue-Tracker: XMLRPC (Open API), Open Formats, Open Source individual issue tracker

Licensing

- To avoid Vendor Lock-In look at licenses
- GNU General Public License (GPL) not enough to keep software in a cloud open
- cloud provider could take the software, make own modifications (which you will depend upon) and not release the modified software to you as a customer.
- In GPL this counts as a private modification
- To prevent this, the GNU Affero General Public License has been designed that prevents closed-source modifications to hosted applications.

Avoid Vendor Lock-In

- Generally: use local tools where possible
- Design-Tools often need local performance (e.g. 3D modelling)
- don't move into the cloud if you can avoid it
- . . . at least have a way back out
  - Open Source, proper license
  - next best: open data format
- this avoids vendor lock-in
- . . . or loss of your data when a provider goes out of business

Vertical Design Languages

- application domains need their own design tools
- Open Source still has terrain to conquer currently occupied by *very* expensive vertical solutions.
- Examples are electronic circuit design, 3D CAD, FPGA and integrated circuit design
- . . . and tomorrow we'll build open source cars
- combine with automated production and prototyping formats (CNC)
- allow for product chain sustainability management (bridge to regional solidarity economies mapping and negotiation systems)

Design: User Feedback, User Codesign

- allow for virtual product testing
- build a language that is descriptional and educational at the same time, enabling users to codesign, give inputs and learn at the same time (active vs. passive competence model from linguistics)
- Learnable User Interface that allows users to become experts
- Allow for synchronous work (mesh whiteboarding and multiuser CAD with Textual, Symbolic, Video and Audio metachannels),

Don’t Get Trapped in a Data Silo

We as users (of web services and design tools) need to decide what applications we use:

“These problems cannot be solved by the companies themselves. Companies make silos. It’s as simple as that. Left to their own devices, that’s what they do. Over and over and over again.”

Doc Searls (Co-author of Cluetrain Manifesto)
in blog article “Silos End”

What we’re doing in Vienna

- Have: Mail, Wiki (ProWiki), Mailinglists, Jabber
- Have: Backup to the Cloud: Cumulus
- Next: Content-Mgmt for Journal-Style publishing
- Next: Asterisk with google talk and maybe Skype, Voice-conferencing, later Video
- Want: Social networking with *open* interface: “Friend of a Friend” (FOAF)
- Want: Easy solution to do meshups (needs APIs, e.g., REST, XMLRPC, SOAP)
- Maybe: Shop, Marketplace, our own Money (!)
- Maybe: Bug-Tracker (project specific)

User Programmable Content

- In search of an “Easy solution”
- Apple’s Hypercard was user programmable widget
- In 1^{st} year 1 million circulated apps
- Tilestack.com: Hypercard in a web-widget
- Easy programming language similar to Apple’s HyperTalk
- Translated to JavaScript that runs in the Browser
- The idea is right but it’s proprietary
- . . . and we don’t want to get into a new vendor lock-in