A guide to techno-galactic software observation

I am less interested in the critical practice of reflection, of showing once-again that the emperor has no clothes, than in finding a way to diffract critical inquiry in order to make difference patterns in a more worldly way. [1]

The techno-galactic software survival guide that you are holding right now was collectively produced as an outcome of the Techno-Galactic Software Observatory. This guide proposes several ways to achieve critical distance from the seemingly endless software systems that surround us. It offers practical and fantastical tools for the tactical (mis)use of software, empowering/enabling users to resist embedded paradigms and assumptions. It is a collection of methods for approaching software, experiencing its myths and realities, its risks and benefits.

With the rise of online services, software use has been increasingly knitted into production, while suggesting that these roles constitute separate realms. This has an effect on the way software is used and produced, and radically alters its operative role in society. The shifts ripple across galaxies, through social structures, working conditions and personal relations, resulting in a profusion of apparatuses aspiring to be seamless while optimizing and monetizing individual and collective flows of information in line with the interests of a handful of actors. The diffusion of software services affects the personal, in the form of intensified identity shaping and self-management. It also affects the public, as more and more libraries, universities and public infrastructures as well as the management of public life rely on “solutions” provided by private companies. Centralizing data flows in the clouds, services blur the last traces of the thin line that separates bio- from necro-politics.

---

[1] Haraway, Donna: “Modest Witness: Feminist Diffractions in Science Studies”. In: Galison, Peter/Stump, David J. (eds.): *The Disunity of Science: Boundaries, Contexts, and Power*. 1996, 428–442.

Given how fast these changes resonate and reproduce, there is a growing urgency to engage in a critique of software that goes beyond taking a distance, and that deals with the fact that we are inevitably already entangled. How can we interact, intervene, respond and think with software? What approaches can allow us to recognize the agency of different actors, their ways of functioning and their politics? What methods of observation enable critical inquiry and affirmative discord? What techniques can we apply to resurface software where it has melted into the infrastructure and into the everyday? How can we remember that software is always at work, especially where it is designed to disappear into the background?

We took on the term of observation for a number of reasons. We regard observation as a way to approach software, as one way to organize engagement with its implications. Observation, and the enabling of observation through intensive data-centric feedback mechanisms, is part of the cybernetic principles that underpin present day software production. Our aim was to scrutinize this methodology in its many manifestations, including in “observatories” – high cost infrastructures [testing infrastructures?] of observation troubled by colonial, imperial traditions and their problematic divisions of nature and culture – with the hope of opening up questions about who gets to observe software (and how) and who is being observed by software (and with what impact)? It is a question of power, one that we answer, at least in part, with critical play.

We adopted the term techno-galactic to match the advertised capability of “scaling up to the universe” that comes in contemporary paradigms of computation, and to address different scales of software communities and related political economies that involve and require observation.

Drawing on theories of software and computation developed in academia and elsewhere, we grounded our methods in hands-on exercises and experiments that you now can try at home. This Guide to Techno-Galactic Software Observation offers methods developed in and inspired by the context of software production, hacker culture, software studies, computer science research, Free Software communities, privacy activism, and artistic practice. It invites you to experiment with ways to stay with the trouble of software.

Figure 0.1: Techno-Galactic Software Observation team, WTC Brussels, June 2017

The Techno-Galactic Software Observatory

In the summer of 2017, around thirty people gathered in Brussels to explore practices of proximate critique with and of software in the context of a work-session entitled “Techno-Galactic Software Observatory”[REF:worksessions]. The worksession called for software-curious people of all kinds to ask questions about software. The intuition behind such a call was that different types of engagement requires a heterogeneous group of participants with different levels of expertise, skill and background. During three sessions of two days, participants collectively inspected the space-time of computation and probed the universe of hardware-software separations through excursions, exercises and conversations. They tried out various perspectives and methods to look at the larger picture of software as a concept, as a practice, and as a set of techniques.

The first two days of The Techno-Galactic Software Observatory included visits to the Muse de l’Informatique Pionnire en Belgique [REF:NAM-IP] in Namur and the Computermuseum KULeuven. In the surrounding of these collections of historical ’numerical artefacts’, we started viewing software in a long-term context. It offered us the occasion to reflect on the conditions of its appearance, and allowed us to take on current-day questions from a genealogical perspective. What is software? How did it appear as a concept, in what industrial and governmental circumstances? What happens to the material conditions of its production (minerals, factory labor, hardware) when it evaporates into a cloud?

The second two days we focused on the space-time dimension of IT development. The way computer programs and operating systems are manufactured changed tremendously through time, and so did its production times and places. From military labs via the mega-corporation cubicles to the open-space freelancer utopia, what ruptures and continuities can be traced in the production, deployment, maintenance and destruction of software? From time-sharing to user-space partitions and containerization, what separations were and are at work? Where and when is software made today?

The Walk-in Clinic

The last two days at the Techno-galactic software observatory were dedicated to observation and its consequences. The development of software encompasses a series of practices whose evocative names are increasingly familiar: feedback, report, probe, audit, inspect, scan, diagnose, explore, test ... What are the systems of knowledge and power within which these activities take place, and what other types of observation are possible? As a practical set for our investigations, we set up a walk-in clinic on the 25th floor of the World Trade Center, where users and developers could arrive with software-questions of all kinds.

Figure 0.2: Documenting the path of a Software Curious Person going through the Walk-in Clinic

Do you suffer from the disappearance of your software into the cloud, feel oppressed by unequal user privilege, or experience the torment of software-ransom of any sort? Bring your devices and interfaces to the World Trade Center! With the help of a clear and in-depth session, at the Techno-Galactic Walk-In Clinic we guarantee immediate results. The Walk-In Clinic provides free hands-on observations to software curious people of all kinds. A wide range of professional and amateur practitioners will provide you with Software-as-a-Critique-as-a-Service on the spot. Available services range from immediate interface critique, collaborative code inspection, data dowsing, various forms of network analyses, unusability testing, identification of unknown viruses, risk assessment, opening of black-boxes and more. Free software observations provided. Last intake at 16:45. (invitation to the Walk-In Clinic, June 2017)

On the following pages: Software as a Critique as a Service (SaaCaaS) Directory and intake forms for Software Curious People (SCP).

Technogalactic Software Observation Essentials

WARNING

The survival techniques described in the following guide are to be used at your own risk in case of emergency regarding software curiosity. The publisher will not accept any responsability in case of damages caused by misuse, misundestanding of instruction or lack of curiosity. By trying the action exposed in the guide, you accept the responsability of loosing data or altering hardware, including hard disks, usb key, cloud storage, screens by throwing them on the floor, or even when falling on the floor with your laptop by tangling your feet in an entanglement of cables. No arms have been done to human, animal, computers or plants while creating the guide. No firearms or any kind of weapon is needed in order to survive software.
Just a little bit of patience.

Software observation survival stresses

Physical fitness plays a great part of software observation. Be fit or CTRL-Quit.

When trying to observe software you might experience stresses as such :

- Anxiety
- Sleep deprivation
- Forgetting about eating
- Loss of time tracking

Can you cope with software ? You have to.

our methods for observation, like mapping, come with their luggage.

See also: http://observatory.constantvzw.org/etherdump/toc.md.diff.html

Close encounters

Method: Encounter several collections of historical hardware back-to-back

Remember:

What:

How:

This can be done by identifying one or more computer museums and visit them with little time in-between. Visiting a friend with a large basement and lots of left-over computer equipment can help. Seeing and possibly touching hardware from different contexts (state-administration, business, research, . . .), periods of time, cultural contexts (California, Germany, French-speaking Belgium) and price ranges allows you to sense the interactions between hardware and software development.

When:

Urgency:

Note: It's a perfect way to hear people speak about the objects and their contexts, how they worked or not and how objects are linked one with another. It's also showing economic and cultural aspects of softwares.

Warning: **DO NOT FOLD, SPINDLE OR MUTILATE**

At one point during the demonstration of a Bull computer, the guide revealed the system's "software" – a suitcase sized module with dozens of patch cords. She made the comment that the term "spaghetti code" (a derogatory expression about early code usign many "GOTO" statments) had its origin in this physical arrangement of code as patchings.

Preserving old hardware in order to observe physical manifestation of software. See software here : we did experienced the incredible possibility of actually touching software.

Playing with the binary. Bull cards. Happy operator! Punch card plays. "The highlight of the collection is to revive a real punch card workshop of the 1960s."

Collection de la Maison des critures d'Informatique & Bible, Maredsous
The particularity of the collection lies in the fact that it's the conservation of multiple stages of life of a software since its initial computerization until today. The idea of introducing informatics into the work of working with/on the Bible (versions in Hebrew, Greek, Latin, and French) dates back to 1971, via punch card recordings and their memorization on magnetic tape. Then came the step of analyzing texts using computers.

See also: http://pad.constantvzw.org/p/observatory.guide.jean.heuns

Source: http://www.histoireinform.com/Histoire/+Infos/jmclcadr.htm

Method: Interview people about their histories with software

Remember:

What: Observe personnal narratives around software history. Retrace the path of relation to software, how it changed during the years and what are the human access memories that surrounds it. To look at software through personal relations and emotions. **How:** Interviews are a good way to do it. Informal conversations also. **When:**

Urgency:

Note:

Warning:

Example:

Jean Heuns has been collecting servers, calculators, softwares, magnetic tapes hard disks for xxx years. Found an agreement for them to be displayed in the department hallways. Department of Computer sciences - Kul Leuven.

[interview transcription goes here]

See also:

Source:

Method: Ask several people from different fields and age-groups the same question: "$*Whatissoftware?*$"

Remember: The answers to this question will vary depending on who is asking it to who.

What: By paying close attention to the answers, and possibly logging them, observations on the ambiguous place and nature of software can be made.
How:

When:

Urgency:

Note:

Warning:

Example:

Examples:

Jean Huens (system administrator at the department of Computer Science, KULeuven): “It is difficult to answer the question ‘what is software’, but I know what is good software”

Thomas Cnudde (hardware designer at ESAT - COSIC, Computer Security and Industrial Cryptography, KULeuven): “*Software is a list of sequential instructions! Hardware for me is made of silicon, software a sequence of bits in a file. But naturally I am biased: I’m a hardware designer so I like to consider it as unique and special*”.

Amal Mahious (Director of NAM-IP, Namur): “*This, you have to ask the specialists.*”

*what is software? –the unix filesystem says: it’s a file —-what is a file? —-in the filesystem, if you ask xxd: —— it’s a set of hexadecimal bytes ——-what is hexadecimal bytes? —— -b it’s a set of binary 01s —-if you ask objdump ——-it’s a set of instructions –side channel researching also says: —-it’s a set of instructions –the computer glossary says: —-it’s a computer’s programs, plus the procedure for their use http://etherbox.local/home/pi/video/A_Computer_Glossary.webm#t=02:26 —— a computer’s programs is a set of instrutions for performing computer operations Source Pad on fileflowchart.raw.html [2]

To answer the question “what is software” depends on the situation, goal, time, and other contextual influences. Source: Pad on Multiple Software Axes [3]

---

[2] http://observatory.constantvzw.org/etherdump/fileflowchart.raw.html
[3] http://observatory.constantvzw.org/etherdump/multiple-software-axes.md.raw.html

See also: http://pad.constantvzw.org/p/observatory.guide.everyonescp

Source:

Method: FMEM and /DEV/MEM

What: Different ways of exploring your memory (RAM). Because in unix everything is a file, you can access your memory as if it were a file. **When:**

Urgency: To try and observe the operational level of software, getting closer to the workings, the instruction-being of an executable/executing file, the way it is when it is loaded into memory rather than when it sits in the harddisk **Remember:** In Unix-like operating systems, a device file or special file is an interface for a device driver that appears in a file system as if it were an ordinary file. In the early days you could fully access your memory via the memory device (/dev/mem) but over time the access was more and more restricted in order to avoid malicious processes to directly access the kernel memory. The kernel option CONFIG$_{S}TRICT_{D}EVMEMwasintroducedinkernelversion2.6andupper(2.6.362.6.39, 3.03.8, HEAD).Soyou'llneedtousetheLinuxkernelmodulefmem : thismodulecreates/dev/fmemdevice, thatc$

/dev/mem tools to explore processes stored in the memory

ps ax | grep process cd /proc/numberoftheprocess cat maps –> check what it is using

The proc filesystem is a pseudo-filesystem which provides an interface to kernel data structures. It is commonly mounted at /proc. Most of it is read-only, but some files allow kernel variables to be changed.

dump to a file–>change something in the file–>dump new to a file–>diff oldfile newfile

“where am i?”

to find read/write memory addresses of a certain process awk -F “-|” ‘$3 ~ /rw/ { print $1 ” ” $2}’ /proc/PID/maps

take the range and drop it to hexdump sudo dd if=/dev/mem bs=1 skip=$((16b7526000-1)) count =$((16#b7528000 - 16#7b7526000 + 1)) | hexdump -C

Besides opening the memory dump with an hex editor you can also try and explore it with other tools or devices. You can open it as a raw image, you can play it as a sound or perhaps send it directly to your frame-buffer device (/dev/fb0). **Warning:** Although your memory may look like/sound like/read like

gibberish, it may contain sensitive information about you and your computer!

Example:

See also: http://pad.constantvzw.org/p/observatory.guide.monopsychism **Source:** http://observatory.constantvzw.org/etherdump/files.html

Method: Pan/Monopsychism

What: Reading and writing sectors of memory from/to different computers **How:** Shell commands and fmem kernel module **Urgency:** Memory, even when it is volatile, is a trace of the processes happening in your computer in the form of saved information, and is therefore more similar to a file than to a process. Challenging the file/process divide, sharing memory with others will allow a more intimate relation with your and other's computers. **See also:** http://pad.constantvzw.org/p/observatory.guide.devmem **Note:** The parallel allocation and observation of the same memory sector in two different computers is in a sense the opposite process of machine virtualization, where the localization of multiple virtual machines in one physical comptuers can only happen by rigidly separating the memory sectors dedicated to the different virtual machines.

Warning: THIS METHOD HAS NOT BEEN TESTED, IT CAN PROBABLY DAMAGE YOUR RAM MEMORY AND/OR COMPUTER

Example:

First start the fmem kernel module in both computers:

```
sudo sh fmem/run.sh
```

Then load part of your computer memory into the other computer via dd and ssh:

```
dd if=/dev/fmem bs=1 skip=1000000 count=1000 - ssh user@othercomputer dd of=/dev/f
```

Or viceversa, load part of another computer's memory into yours:

```
ssh user@othercomputer dd if=/dev/fmem bs=1 skip=1000000 count=1000 -  dd of=/dev/
```

Or even, exchange memory between two other computers:

```
ssh user@firstcomputer dd if=/dev/fmem bs=1 skip=1000000 count=1000 - ssh user@sec
```

pan/monopsychism: (aquinas famously opposed averroes..who's philosophy
can be interpreted as monopsychist)

shared memory

copying the same memory to different computers

https://en.wikipedia.org/wiki/Reflection_%28computer_programming%29

it could cut through the memory like a worm

or it could go through the memory of different computers one after the other
and take and leave something there

Source: etherpad snippets, code speculation

Temporality

Method: Fountain refreshment

Remember:

What: Augmenting a piece of standardised office equipment designed to dispense water to perform a decorative function. **How:** Rearranging space as conditioning observations (WTC vs. Museum vs. University vs. Startup Office vs. Shifting Walls that became Water Fountains) **When:**

Urgency: EU-OSHA (European Agency for Safety and Health at Work) Directive 2003/10/EC noise places describes the minimum health and safety requirements regarding the exposure of workers to the risks arising from physical agents (noise). However no current European guidelines exist on the potential benefitial uses of tactially designed additive noise systems. **Note:**

Warning:

Example:

The Technogalactic Software Observatory – Comfortable silence, one way mirrors

A drinking fountain and screens of one-way mirrors as part of the work session The Technogalactic Software Observatory organised by Constant.

For the past 100 years the western ideal of a corporate landscape has been has been moving like a pendulum, oscillating between grids of cubicles and organic, open landscapes, in a near to perfect 25-year rhythm. These days the changes in office organisation is supplemented by sound design, in corporate settings mostly to create comfortable silence. Increase the sound and the space becomes more intimate, the person on the table next to you can not immediately hear what you are saying. It seems that actual silence in public and corporate spaces has not been sought after since the start of the 20th century. Actual silence is not at the moment considered comfortable. One of the visible symptoms of our desire to take the edge off the silence is to be observed through the appearance of fountains in public space. The fountains purpose being to give off neutral sound, like white noise without the negative connotations. However as a sound engineer's definition of noise is unwanted sound that all depends on ones personal relation to the sound of dripping water.

This means that there needs to be a consistent inoffensiveness to create comfortable silence.

In corporate architecture the arrival of glass buildings were originally seen as a symbol of transparency, especially loved by governmental buildings. Yet the reflectiveness of this shiny surface once combined with strong light – known as the treason of the glass – was only completely embraced at the invention of one-way-mirror foil. And it was the corporate business-world that would come to be known for their reflective glass skyscrapers. As the foil reacts to light, it appears transparent to someone standing in the dark, while leaving the side with the most light with an opaque surface. Using this foil as room dividers in a room with a changing light, what is hidden or visible will vary throughout the day. So will the need for comfortable silence.

Disclaimer :
Similar to the last 100 years of western office organisation,
this fountain only has two modes:
on or off

If it is on it also offers two options
cold water and hot water

This fountain has been tampered with and has not in any way been approved by a proffesional fountain cleaner. I do urge you to consider this before you take the decision to drink from the fountain.

Should you chose to drink from the fountain, then I urge you to write your name on your cup, in the designated area, for a customised experience of my care for you.

I do want you to be comfortable.

http://observatory.constantvzw.org/documents/mia/mia6.gif http://observatory.constantvzw.org/documents/mia
http://observatory.constantvzw.org/documents/mia/IMG_5695.JPG http://observatory.constantvzw.org/docume

See also:

Source: Mia Melvaer

Method: Create ”nannyware”: Software that observes and addresses the user

Remember:

What: Nannyware is software meant to protect users while limiting their space of activity. It is software that passive-aggressively suggests or enforces some kind of discipline. In other words, create a form of parental control extended to adults by means of user experience / user interfaces.

Nannyware is a form of Content-control software: software designed to restrict or control the content a reader is authorised to access, especially when utilised to restrict material delivered over the Internet via the Web, e-mail, or other means. Content-control software determines what content will be available or be blocked.

How: [...Restrictions] can be applied at various levels: a government can attempt to apply them nationwide (see Internet censorship), or they can, for example, be applied by an ISP to its clients, by an employer to its personnel, by a school to its students, by a library to its visitors, by a parent to a child's computer, or by an individual user to his or her own computer. (source: https://en.wikipedia.org/wiki/Content-control$_{s}oftware$)

When:

Urgency: As with all new lifestyle technologies that come along, in the beginning there is also some chaos until their impact can be assessed and rules put in place to bring order and respect to their implementation and use in society. When the automobile first came into being there was much confusion regarding who had the right of way, the horse or the car. There were no paved roads, speed limits, stop signs, or any other traffic rules. Many lives were lost and much property was destroyed as a result. Over time, government and society developed written and unwritten rules as to the proper use of the car. (source: https://www.netnanny.com/products/netnanny/protecting-your-family/)

Note:

Warning: Disadvantages of explicit proxy deployment include a user's ability to alter an individual client configuration and bypass the proxy. To counter this, you can configure the firewall to allow client traffic to proceed only through the proxy. Note that this type of firewall blocking may result in some applications not working properly. (from: https://web.archive.org/web/20120418150020/http://www.websense.c

Example: The main problem here is that the settings that are required are different from person to person. For example, I use workrave with a 25 second micropause every two and a half minute, and a 10 minute restbreak every 20 minutes. I need these frequent breaks, because I'm recovering from RSI. And as I recover, I change the settings to fewer breaks. If you have

never had any problem at all (using the computer, that is), then you may want much fewer breaks, say 10 seconds micropause every 10 minutes, and a 5 minute restbreak every hour. It is very hard to give proper guidelines here. My best advice is to play around and see what works for you. Which settings ”feel right”. Basically, that’s how Workrave’s defaults evolve. (from http://www.workrave.org/documentation/faq/)

Figure 0.3: Content-control software

Figure 0.4: A “nudge” from your music player

Facebook is working on an app to stop you from drunk-posting “Yann LeCun, who overseas the lab, told Wired magazine that the program would be like someone asking you, ‘Uh, this is being posted publicly. Are you sure you want your boss and your mother to see this?’ ”

Figure 0.5: This Terminal Dashboard Reminds You to Take a Break When You’re Lost Deep Inside the Command Line

See also:

Figure 0.6:

Figure 0.7:

Source: Silvio Lorusso

Method: Useless scroll against productivity

Remember:

What:

How:

When:

Urgency:

Note:

Warning:

Example:

See also:

Source:

Method: Investigating how humans and machines negotiate the experience of time

Remember:

What: http://observatory.constantvzw.org/images/Screenshot$_{f}rom_{2}017-06-10_{1}72547.png$**How:**$pythonscript$**When:**

Urgency:

Note:

Warning:

Example:

<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.w3.org/TR/xhtml1/DTD/xhtr
transitional.dtd”> <html xmlns=“http://www.w3.org/1999/xhtml”> <head> <meta
http-equiv=“Content-Type” content=“text/html; charset=utf-8” /> <meta http-
equiv=“Content-Style-Type” content=“text/css” /> <meta name=“generator”
content=“pandoc” /> <title></title> <style type=“text/css”>code{white-space:
pre;}</style> <link rel=“stylesheet” href=“/home/pi/include/markdownpad.css”
type=“text/css” /> </head> <body> <h1 id=“ends-of-time”>ends of time</h1>
https://en.wikipedia.org/wiki/Year_2038_problem Exact moment
of the epoch: 03:14:07 UTC on 19 January 2038 <h2 id=“commands”>commands</h2>
local UNIX time of this machine date +%s UNIX time + 1 echo
$((<code>date +%s</code> +1)) <h2 id=“goodbye-unix-time”>goodbye
unix time</h2> <pre><code>while : do sleep 1 figlet $((2147483647 - 1))
done</code></pre> <h1 id=“sundial-time-protocol-group-tweaks”>Sundial
Time Protocol Group tweaks</h1> <pre><code>printf ‘Current Time in
Millennium Unix Time:’ printf $((2147483647 - 1)) echo sleep 2 echo $((1
+ 2)) > ends-of-times/idletime idletime=1 echo figlet “Thank you for having
donated 2 seconds to our ${idletime} seconds of collective SSH pause” echo
echo</code></pre> </body> </html>

See also:

Source:

Languaging

Method: Quine **What:** A program whose function consists of displaying its own code. Also known as ”self-replicating program” **Why:** Quines show the tension between ”software as language” and ”software as operation”. **How:** By running a quine you will get your code back. You may do a step forward and wonder about functionality and aesthetics, uselessness and performativity, data and code.

Example:

Example of a quine (Python). When executed it outputs the same text as the source:

[] s = ’s = print(s

Example of a oneline unibash/etherpad quine, created during relearn 2017: ¡pre class=”quaverbatim”¿ wget -qO- http://192.168.73.188:9001/p/quine/export/txt — curl -F ”file=@-;type=text/plain” http://192.168.73.188:9001/p/quine/import ¡/pre¿

Warning:

The encounter with quines may deeply affect you. You may want to write one and get lost in trying to make an ever shorter and more elegant one. You may also take quines as point of departure or limit-ideas for exploring software dualisms.

“A quine is without why. It prints because it prints. It pays no attention to itself, nor does it asks whether anyone sees it.” “Aquine is aquine is aquine.” Aquine is not a quine This is not aquine

Remember: Although seemingly absolutely useless, quines can be used as exploits.

Exploring boundaries/tensions

databases treat their content as data (database punctualization) some exploits manage to include operations in a database

See also: http://pad.constantvzw.org/p/observatory.guide.monopsychism **Source:**

Part of Aquine, a discussion of and research into dualism in software http://observatory.constan

Method: Glossaries as an exercise

Remember:

What:

How:

When:

Urgency:

Note:

Warning:

Example:

See also:

Source:

Method: Adding qualifiers (secure, bad, bourgeois, queer...)

Remember:

What: Bringing a moral, ethical, or otherwise evaluative/adjectival/validating lens: ”This morning, Jan had difficulties to answer the question ”what is software”, but he said that he could answer the question ”what is good software”. What is good software?

How: The more adjectives, the easier the answering? **When:**

Urgency:

Note: A qualifier like ”good”, ”bad”, ”spy”, ”queer”, ”proletarian”, ”bourgeisoie” can help narrow down definitions.” **Warning:**

Example:

See also:

Source: http://observatory.constantvzw.org/etherdump/multiple-software-axes.html

Method: Searching ”software” through software **What:** A quick way to sense the ambiguity of the term 'software', is to go through the manual files on your hard drive and observe in which cases is the term used. **How:** command-line oneliner **Why:** : Software is a polymorphic term that take different meanings and comes with different assumptions for the different agents involved in its production, usage and all other forms of encounter and subjection. From the situated point of view of the software present on your machine, when and why does software call itself as such?

Example:

so software exists only outside your computer? only in general terms? checking for the word software in all man pages:

```
grep -nr software /usr/local/man
!!!!
```

software appears only in terms of license:

```
This program is free software
This software is copyright (c)
```

we don't run software..we still run programs
nevertheless software is everywhere

Source: [Day1], etherpad snippet, line 574-589 http://observatory.constantvzw.org/etherdump/

See also: http://pad.constantvzw.org/p/observatory.guide.samequestion

Method: Persist in calling everyone a Software Curious Person

Remember:

What: Persistance in naming is a method for changing a person's relationship to software by (sometimes forcibly) call everyone a Software Curious Person. **How:** Insisting on curiosity as a relation, rather than for example 'fear' or 'admiration' might help cut down the barriers between different types of expertise and allows multiple stakeholders feel entitled to ask questions, to engage, to investigate and to observe. **When:**

Urgency: Software is too important to not be curious about. Observations could benefit from recognising different forms of knowledge. It seems important to engage with software through multiple interests, not only by means of technical expertise. **Note:**

Warning:

Example:

This method was used to address each of the visitors at the Technogalactic Walk-in Clinic.

See also:

Source:

Healing

Method: Setup a Relational software observatory consultancy (RSOC)

Remember:

- Collectivise research around hacking to save time.
- Self-articulate software needs as your own Operating (system) perspective.
- Change the lens by looking to software through a time perspective.

What: By paying a visit to our ethnomethodology interview practice youll learn to observe software from different angles / perspectives. Our practionners passion is to make the ”what is the relation to software” discussion into a service.
How: Reading the signs. Considering the everchanging nature of software development and use and its vast impact on globalized societies, it is necessary to recognize some of the issues of how software is (often) either passively-perceived or actively-observed, without an articulation of the relations. We offer a method to read the signs of the relational aspect of software observance. It's a crucial aspect of our guide. It will give you another view on software that will shape your ability to survive any kind of software disaster.

Strategy: Case study
Tool: Qualitative interview

Reading the signs. From: John “Lofty” Wiseman, SAS Survival Handbook: The Ultimate Guide to Surviving Anywhere [4]

When:

Urgency:

Note:

Warning:

Example:

What follows is an example of a possible diagnostic questionnaire.

---

[4] http://gallery.constantvzw.org/index.php/Techno-Galactic-Software-Observatory/IMAG1319

Sample Questionnaire

What to expect You will obtain a cartography of software users profiles. It will help you to shape your own relation to software. You will be able to construct your own taxonomy and classifcation of software users that is needed in order to find a means of rescue in case of a software catastrophy.

- SKILLS

- What kind of user would you say that you are?
- What is your most frequently used type of software?
- How often do you install/experiment/learn new software?

- History
- What is your first recollection of software use?
- How often do / when did you last purchase software or pay for a software service?

- Ethics
- What is the software feature you care about the most?
- Do you use any free software?
  - if yes than
  - do you remember your first attempt at using this software service? Do you still use it? If not why?

- Do you pay for media distribution/streaming services?
- Do you remember your first attempt at using free software and how did that make you feel?
- Have you used any of these software services : facebook, dating app (grindr, tinder, etc.), twitter, instagram or equivalent.

- Can you talk about your favorite apps or webtools that you use regularly?
- What is most popular software your friends use?

- SKILL
- Would you say that you are a specilised user?

- Have you ever used the command line?
- Do you know about scripting?
- Have you ever edited an HTML page? A CSS file? A PHP file? A configuration file?

- Can you talk about your most technical encounter with your computer / telephone?

- ECONOMY

- How do you pay for your software use?
  - Please elaborate (for example, do you buy the software? / contribute in kind / deliver services or support)
  - What is the last software that you paid for using?
  - What online services are you currently paying for?
  - Is someone paying for your use of service?

- Personal
- What stories do you have concerning contracts and administration in relation to your software, Internet or computer?
- How does software help you shape your relations with other people?
- From which countries does your softwares come from / reside? How do you feel about that?
- Have you ever read a terms of software service, what about one that is not targeting the American market?

Sample questionnaire results

Possible/anticipated user profiles

. . . meAsHardwareOwnerSoftwareUSER:

“I did not own a computer personally until very very late as I did not enjoy gaming as a kid or had interest in spending much time behind PC beyond work (and work computer). My first was hence I think in 2005 and it was a SGI workstation that was the computer of the year 2000 (cost 10.000USD) and I got it for around 300USD. Proprietary drivers for unified graphics+RAM were never released, so it remained a software dead-end in gorgeous blue curved chassis http://www.sgidepot.co.uk/sgidepot/pics/vwdocs.jpg”

. . . meAsSoftwareCONSUMER:

“I payed/purchased software only twice in my life (totalling less then 25eur), as I could access most commercial software as widely pirated in Balkans and later had more passion for FLOSS anyway, this made me relate to software as material to exchange and work it, rather than commodity goods I could or not afford.”

. . . meAsSoftwareINVESTOR:

“I did it as both of those apps were niche products in early beta (one was Jeeper Elvis, real-time-non-linear-video-editor for BeOS) that failed to reach market, but I think I would likely do it again and only in that mode (supporting the bleeding edge and off-stream work), but maybe with more than 25eur.”

. . . meAsSoftwareUserOfOS:

“I would spend most of 80s ignoring computers, 90ties figuring out software from high-end to low-end, starting with OSF/DecAlpha and SunOS, than IRIX and MacOS, finally Win 95/98 SE, that permanently pushed me into niches (of montly LINUX distro install fests, or even QNX/Solaris experiments and finally BeOS use).”

. . . meAsSoftwareWEBSURFER:

“I got used to websurfing in more than 15 windows on UNIX systems and never got used to less than that ever since, furthermore with addition of more browser options this number only multiplied (always wondered if my first system was Windows 3.11 - would I be a more focused person and how would that form my relations to browser windows>tabs).”

. . . meAsSoftwareUserOfPropertarySoftware:

”I signed one NDA contract in person on the paper and with ink on a rainy day while stopping of at trainstaion in north Germany for the software that was later to be pulled out of market due to problematic licencing agreement (intuitivly I knew it was wrong) - it had too much unprofessional pixeleted edges in its graphics.

. . . meAsSoftwareUserOfDatingWebsites:

“I got one feature request implemented by a prominent dating website (to search profiles by language they speak), however I was never publicly acknowledged (though I tried to make use of it few times), that made our relations feel a bit exploitative and underappreciated.”

. . . meAsSoftwareUserTryingToGoPRO:

“my only two attempts to get into the software company failed as they insisted on full time commitments. Later I found out ones were intimidated in interview and other gave it to a person that negotiated to work part time with friend! My relation to professionalism is likely equally complex and pervert as one to the software.”

Case study : W. W.

. . . ww.AsExperiencedAdventerousUSER - experiments with software every two days as she uses FLOSS and Gnu/Linux, cares the most for maliabity of the software - as a result she has big expectations of flexibility even in software category which is quite conventional and stability focused like file-hosting.

. . . ww.AsAnInevstorInSoftware - paid compiled version of FLOSS audio software 5 years ago as she is supportive of economy and work around production, maintainance and support, but she also used closed hardware/software where she had to agree on licences she finds unfair, but then she was hacking it in order to use it as an expert - when she had time.

. . . ww.AsCommunicationSoftwareUSER - she is not using commercial social networks, so she is very concious of information transfers and time relations, but has no strong media/format/design focus.

Q: What is your first recollection of software use?
A: ms dos in 1990 at school _ i was 15 or 16. oh no 12. Basic in 1986.

Q: What are the emotions related to this use?
A: fun. i'm good at this. empowering

Q: How often do / when did you last purchase software or pay for a software service?
A: I paid for ardour five years ago. I paid the developper directly. For the compiled version. I paid for the service. I pay for my website and email service at domaine public.

Q: What kind of user would you say you are?
A: An experienced user drawing out the line. I don't behave.

Q: Is there a link between this and your issue?
A: Even if it's been F/LOSS there is a lot of decision power in my package.

Q: What is your most frequently used type of software?
A: Web browser. email. firefox & thunderbird

Q: How often do you install/experiment/learn new software?
A: Every two days. I reinstall all the time. my old lts system died. stop being supported last april. It was linux mint something.

Q: Do you know about scripting?
A: I do automating scripts for any operation i have to doi several times like format conversion.

Q: Can you talk about your most technical encounter with your computer / telephone?
A: I've tried to root it. but i didn't succeed.

Q: How much time do you wish to spend on such activities like hacking, rooting your device?
A: hours. you should take your time

Q: Did you ever sign licence agreement you were not agree with? How does that affect you?
A: This is the first thing your when you have a phone. it's obey or die.

Q: What is the software feature you care for the most?
A: malleability. different ways to approach a problem, a challenge, an issue.

Q: Do you use any free software?
A: yes. there maybe are some proprietary drivers.

Q: Do you remember your first attempt at using free software and how did that make you feel?
A: Yes i installed my dual boot in . . . 10 years ago. scared and powerful.

Q: Do you use one of this software service: facebook, dating app (grindr of sort), twitter, instagram or equivalent?
A: Google, gmail that's it

Q: Can you talk about your favorite apps or webtools that you use regularly?
A: Music player. vanilla music and f-droid. browser. I pay attention to clearing my history, no cookies. I also have iceweasel. Https by default. Even though i have nothing to hide.

Q: What stories around contracts and administration in relation to your software internet or computer?
A: Nothing comes to my mind. i'm not allowed to do, to install on phone. When it's an old phone, there is nothing left that is working you have to do it.

Q: How does software help you shape your relations with other people?
A: It's a hard question. if it's communication software of course it's it's nature to be related to other people.there is an expectency of immediate reply, of information transfer. . . It's troubling your relation with people in certain situations.

Q: From which countries does your softwares live / is coming from? How do you feel about that?
A: i think i chose the netherlands as a miror. you are hoping to reflect well in this miror.

Q: Have you ever read a terms of software service; one that is not targeting the American market?
A: i have read them. no.

See also:

Source:

http://observatory.constantvzw.org/etherdump/SCP.see.html http://observatory.constantvzw.org/etherdump/SC
http://observatory.constantvzw.org/etherdump/SCP.wendy.html This method was developed by The RSOC Group.

Method: Agile Sun Salutation

Remember:

Agile software development describes a set of values and principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change. These principles support the definition and continuing evolution of many software development methods. Source: https://en.wikipedia.org/wiki/Agile_software_development

What: You will be observing yourself **How:** Scrum is a framework for managing software development. It is designed for teams of three to nine developers who break their work into actions that can be completed within fixed duration cycles (called ”sprints”), track progress and re-plan in daily 15-minute stand-up meetings, and collaborate to deliver workable software every sprint. Approaches to coordinating the work of multiple scrum teams in larger organizations include Large-Scale Scrum, Scaled Agile Framework (SAFe) and Scrum of Scrums, among others. (from: https://en.wikipedia.org/wiki/Scrum$_(software_development))$**When:**$Anywherewhereit$

Warning:

The agile movement is in some ways a bit like a teenager: very self-conscious, checking constantly its appearance in a mirror, accepting few criticisms, only interested in being with its peers, rejecting en bloc all wisdom from the past, just because it is from the past, adopting fads and new jargon, at times cocky and arrogant. But I have no doubts that it will mature further, become more open to the outside world, more reflective, and also therefore more effective. —?Kruchten, Philippe (2011-06-20). “Agile’s Teenage Crisis?”. InfoQ. (from https://en.wikipedia.org/wiki/Agile_software_development#Criticism)

Example:

Hello and welcome to the presentation of the agile yoga methodology. I am Allegra, and today I’m going to be your personal guide to YOGA, an acronym for why organize? Go agile! I’ll be part of your team today and we’ll do a few exercises together as an introduction to a new path into your professional and personal life towards creativity, focus and health.

A few months ago, I was stressed, overwhelmed with my work, feeling alone, inadequate, but since I started practicing agile yoga, I feel more productive. I have many clients as an agile yoga coach, and I’ve seen new creative business opportunities coming to me as a software developer.

For this first experience with the agile yoga method and before we do physical exercises together, I would like to invite you to close your eyes. Make yourself comfortable, lying on the floor, or sitting with your back on the wall. Close your eyes, relax. Get comfortable. Feel the weight of your body on the floor or on the wall. Relax.

Leave your troubles at the door. Right now, you are not procrastinating, you are having a meeting at the <SAY THE NAME OF YOUR LOCATION HERE>, a professional building dedicated to business, you are meeting yourself, you are your own business partner, you are one. You are building your future.

You are in a room standing with your team, a group of lean programmers. You are watching a white board together. You are starting your day, a very productive day as you are preparing to run a sprint together. Now you turn towards each other, making a scrum with your team, you breathe together, slowly, inhaling and exhaling together, slowly, feeling the air in and out of your body. Now you all turn towards the sun to prepare to do your ASSanas, the agile Sun Salutations or ASS with the team dedicated ASS Master. She’s guiding you. You start with Namaskar, the Salute. your palms joined together, in prayer pose. you all reflect on the first principle of the agile manifesto. your highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Next pose, is Ardha Chandrasana or (Half Moon Pose). With a deep inhalation, you raise both arms above your head and tilt slightly backward arching your back. you welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. then you all do Padangusthasana (Hand to Foot Pose). With a deep exhalation, you bend forward and touch the mat, both palms in line with your feet, forehead touching your knees. you deliver working software frequently.

Surya Darshan (Sun Sight Pose). With a deep inhalation, you take your right leg away from your body, in a big backward step. Both your hands are firmly planted on your mat, your left foot between your hands. you work daily throughout the project, business people and developers together. now, you're flowing into Purvottanasana (Inclined Plane) with a deep inhalation by taking your right leg away from your body, in a big backward step. Both your hands are firmly planted on your mat, your left foot between your hands. you build projects around motivated individuals. you give them the environment and support they need, and you trust them to get the job done.

You're in Adho Mukha Svanasana (Downward Facing Dog Pose). With a deep exhalation, you shove your hips and butt up towards the ceiling, forming an upward arch. Your arms are straight and aligned with your head. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Then, Sashtang Dandawat (Forehead, Chest, Knee to Floor Pose). With a deep exhalation, you lower your body down till your forehead, chest, knees, hands and feet are touching the mat, your butt tilted up. Working software is the primary measure of progress.

Next is Bhujangasana (Cobra Pose). With a deep inhalation, you slowly snake forward till your head is up, your back arched concave, as much as possible. Agile processes promote sustainable development. You are all maintaining a constant pace indefinitely, sponsors, developers, and users together.

Now back into Adho Mukha Svanasana (Downward Facing Dog Pose). Continuous attention to technical excellence and good design enhances agility.

And then again to Surya Darshan (Sun Sight Pose). Simplicity–the art of maximizing the amount of work not done–is essential. Then to Padangusthasana (Hand to Foot Pose). The best architectures, requirements, and designs emerge from self-organizing teams.

You all do again Ardha Chandrasana (Half Moon Pose). At regular intervals, you as the team reflect on how to become more effective, then tune and adjust your behavior accordingly. you end our ASSanas session with a salute to honor your agile yoga practices. you have just had a productive scrum meeting. now i invite you to open your eyes, move your body around a bit, from the feet up to the head and back again.

Stand up on your feet and let's do a scrum together if you're ok being touched on the arms by someone else. if not, you can do it on your own. so put your hands on the shoulder of the SCP around you. now we're joined together, let's look at the screen together as we inhale and exhale. syncing our body together to the rythms of our own internal software, modulating our oxygen level intake requirements to the oxygen availability of our service facilities.

Now, let's do together a couple of exercise to protect and strengthen our wrists. as programmers, as internauts, as entrepreneurs, they are a very crucial parts of the body to protect. in order to be able to type, to swipe, to shake hands vigourously, we need them in good health. So bring to hands towards each other in a prayer pose, around a book, a brick. You can do it without but I'm using my extreme programming book - embrace change - for that. So press the palms together firmly, press the pad of your fingers together. do that while breathing in and out twice.

Now let's expand our arms towards us, in the air, face and fingers facing down. like we're typing. make your shoulders round. let's breath while visualizing in our heads the first agile mantra : Individuals and interactions over processes and tools.

Now let's bring back the arms next to the body and raise them again. And let's move our hands towards the ceiling this time. Strenghtening our back. In our head, the second mantra. Working software over comprehensive documentation. now let's bring back the hands in the standing position. Then again the first movement while visualizing the third mantra : Customer collaboration over contract negotiation and then the second movement thinking about the fourth and last mantra : Responding to change over following a plan and of course we continue breathing. Now to finish this session, let's do a sprint together in the corridor !

5 6

---

[5] http://observatory.constantvzw.org/guide/agileyoga/8-Poses-Yoga-Your-Desk.contours.png

[6] http://observatory.constantvzw.org/guide/agileyoga/gayolab-office-chair-for-yoga.contours.png

See also:

Source: Developed by: Anne Laforet, performed by: Allegra

Method: Hand reading **How:** Visit the Future Blobservation Booth to have your fortunes read and derive life insight from the wisdom of software. **What:** put your hand in the reading booth and get your line read. **Why:** The hand which holds your mouse everyday hides many secrets.

Example:

```
 sample reading timeline:
15:00 a test user, all tests clear and systems are online a user who said  goodby
 15:35 another nice user
 15:40 another nice user
 15:47 happy user (laughing)
15:51 user complaining about her fortune, saying it's not true. Found  the readi
15:59 another nice user:  http://etherbox.local:9001/p/SCP.sedyst.md
 16:06 a polite user
 16:08 a  friendly playful user (stephanie)
 16:12 a very giggly user (wendy)
 16:14  a playful user - found the reading process erotic - DEFRAGMENTING? NO!  T
 16:19 a  curious user
16:27 a friendly user but oh no, we had a glitch and  computer crashed. But we st
16:40 a nice user, the printer jammed but it was sorted out  quickly
 16:42 another nice user
 16:50 nice user (joak)
 16:52 yet another  nice user (jogi)
 16:55 happy user! (peter w)
 16:57 more happy user  (pierre h)
 16:58 another happy user
 17:00 super happy user (peggy)
 17:02  more happy user
```

Example:

Software time is not the same as human time. Computers will run for AS LONG AS THEY WILL BE ABLE TO, provided sufficient power is available. You, as a human, don't have the luxury of being always connected to the power grid and this have to rely on your INTERNAL BATTERY. Be aware of your power cycles and set yourself to POWER-SAVING MODE whenever possible."

Source:

Method: Bug reporting for sharing observations

Remember:

What: Etherpad had stopped working but it was unclear why. Where does etherpad 'live'? **How:** Started by looking around the pi's filesystem by reading /var/log/syslog in /opt/etherpad and in a subdirectory named var/ there was dirty.db, and dirty it was. **When:** Monday morning **Urgency:** Software (etherpad) not working and the Walk-in Clinic was about to start. **Note:** http://pad.constantvzw.org/p/c

Warning:

Example:

from jogi@mur.at to [Observatory] When dirty.db get's dirty

Dear all,

as promised yesterday, here my little report regarding the broken etherpad.

<md> ### When dirty.db get's dirty

When I got to WTC on Monday morning the etherpad on etherbox.local was disfunct. Later someone said that in fact etherpad had stopped working the evening before, but it was unclear why. So I started looking around the pi's filesystem to find out what was wrong. Took me a while to find the relevant lines in /var/log/syslog but it became clear that there was a problem with the database. Which database? Where does etherpad 'live'? I found it in /opt/etherpad and in a subdirectory named var/ there it was: dirty.db, and dirty it was.

A first look at the file revealed no apparent problem. The last lines looked like this:

```
–"key":"sessionstorage:Ddy0gw7okwbkv5BzkR1DuSLCV˙IA5˙jQ","val":–"cookie
":–"path":"/","˙expires":null,"originalMaxAge":null,"httpOnly":true,"secure":fal
–"key":"sessionstorage:AU1cffgcTf˙q6BV9aIdAvES2YyXM7Gm1","val":–"cookie
":–"path":"/","˙expires":null,"originalMaxAge":null,"httpOnly":true,"secure":fal
–"key":"sessionstorage:˙H5SdUlDvQ3XCuPaZEXQ5lx0K6aAEJ9m","val":–"cookie
":–"path":"/","˙expires":null,"originalMaxAge":null,"httpOnly":true,"se
cure":false˝˝˝
```

What I did not see at the time was that there were some (AFAIR something around 150) binary zeroes at the end of the file. I used tail for the first look and that tool silently ignored the zeroes at the end of the file. It was Martino who suggested using different tools (xxd in that case) and that showed the cause of the problem. The file looked something like this:

```
00013730: 6f6b 6965 223a 7b22 7061 7468 223a 222f  okie":–"path":"/
00013740: 222c 225f 6578 7069 7265 7322 3a6e 756c  ","˙expires":nul
00013750: 6c2c 226f 7269 6769 6e61 6c4d 6178 4167  l,"originalMaxAg
00013760: 6522 3a6e 756c 6c2c 2268 7474 704f 6e6c  e":null,"httpOnl
00013770: 7922 3a74 7275 652c 2273 6563 7572 6522  y":true,"secure"
00013780: 3a66 616c 7365 7d7d 7d0a 0000 0000 0000  :false˝˝˝.......
00013790: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

So Anita, Martino and I stuck our heads together to come up with a solution. Our first attempt to fix the problem went something like this:

```
dd if=dirty.db of=dirty.db.clean bs=1 count=793080162
```

which means: write the first 793080162 blocks of size 1 byte to a new file. After half an hour or so I checked on the size of the new file and saw that some 10% of the copying had been done. No way this would get done in time for the walk-in-clinic. Back to the drawing board.

Using a text editor was no real option btw since even vim has a hard time with binary zeroes and the file was really big. But there was hexedit! Martino installed it and copied dirty.db onto his computer. After some getting used to the various commands to navigate in hexedit the unwanted zeroes were gone in an instant. The end of the file looked like this now:

```
00013730: 6f6b 6965 223a 7b22 7061 7468 223a 222f  okie":-"path":"/
00013740: 222c 225f 6578 7069 7265 7322 3a6e 756c  ","˙expires":nul
00013750: 6c2c 226f 7269 6769 6e61 6c4d 6178 4167  l,"originalMaxAg
00013760: 6522 3a6e 756c 6c2c 2268 7474 704f 6e6c  e":null,"httpOnl
00013770: 7922 3a74 7275 652c 2273 6563 7572 6522  y":true,"secure"
00013780: 3a66 616c 7365 7d7d 7d0a                 :false˝˝˝.
```

Martino asked about the trailing '.' character and I checked a different copy of the file. No '.' there, so that had to go too. My biggest mistake in a long time! The '.' we were seeing in Martino's copy of the file was in fact a '' (0a)! We did not realize that, copied the file back to etherbox.local and waited for etherpad to resume it's work. But no luck there, for obvious reasons.

We ended up making backups of dirty.db in various stages of deformation and Martino started a brandnew pad so we could use pads for the walk- in-clinic. The processing tool chain has been disabled btw. We did not want to mess up any of the already generated .pdf, .html and .md files.

We still don't know why exactly etherpad stopped working sometime Sunday evening or how the zeroes got into the file dirty.db. Anita thought that she caused the error when she adjusted time on etherbox.local, but the logfile does not reflect that. The last clean entry in /var/log/syslog regarding nodejs/etherpad is recorded with a timestamp of something along the line of 'Jun 10 10:17'. Some minutes later, around 'Jun 10 10:27' the first error appears. These timestamps reflect the etherbox's understanding of time btw, not 'real time'.

It might be that the file just got too big for etherpad to handle it. The size of the repaired dirty.db file was already 757MB. That could btw explain why etherpad was working somewhat slugishly after some days. There is still a chance that the time adjustment had an unwanted side effect, but so far there is no obvious reason for what had happened. </md> – J.Hofmller

```
http://thesix.mur.at/
```

See also:

Source: jogi, http://pad.constantvzw.org/p/observatory.inventory.jogi

Method: Interface Dtournement

Remember:

What:

How:

When:

Urgency:

Note:

Warning:

Example:

See also:

Source:

Embodiment / body techniques

Method: Comportment of software (occupational hazards)

Remember: Find a quote from Addicted by Design

What: Observing ways software produces bodies **How:**

When:

Urgency:

Note: Our bodies, our selves and our software...the collective body

Warning: Software may harm your physical and emotional health both by design and by accident

Example:

See also: Agile Sun Salutation, Natashcha Schull's Addicted by Design

Source: Carlin

Flow-regulation, logistics, seamlessness

Method: Continuous integration

What: Continuous integration is a sophisticated form of responsibility management: it is the fascia of services. Continous integration picks up after all other services and identifies what needs to happen so that they can work in concert. Continuous integration is a way of observing the evolution of (micro)services through cybernetic (micro)management. **How:** Continuous integration keeps track of changes to all services and allows everyone to observe if they still can work together after all the moving parts are fitted together. **When:** Continuous integration comes to prominence in a world of distributed systems where there are many parts being organized simultaneously. Continuous integration is a form of observation that helps (micro)services maintain a false sense of independence and decentralization while constantly subjecting them to centralized feedback. **Urgency:** Continuous integration reconfigures divisions of labor in the shadows of automation. How can we surface and question its doings and undoings? **Warning:** When each service does one thing well, the service makers tend to assume everybody else is doing the things they do not want to do.

Example:

At TGSO continuous integration was introduced as a service that responds to integration hell when putting together a number of TGSO services for a walk-in software clinic. Due to demand, the continuous integration service was extended to do “service discovery” and “load balancing” once the walk-in clinic was in operation.

Continuous integration worked by visiting the different services of the walk-in clinic to check for updates, test the functionality and think through implications of integration with other services. If the pieces didn’t fit, continuous integration delivered error messages and solution options.

When we noticed that software curious persons visiting the walk-in clinic may have troubles finding the different services, and that some services may be overloaded with software curious persons, continuous integration was extended. We automated service registration using colored tape and provided a lookup registry for software curious persons. http://gallery.constantvzw.org/index.php/Techno-Galactic-Software-Observatory/IMAG1404

Load balancing meant that software curious persons were forwarded to services that had capacity. If all other services were full, the load balancer defaulted to sending the software curious person to the Agile Sun Salutation [7] service.

Warning: At TGSO the bundling of different functionalities into the continuous integration service broke the ”do one thing well” principle, but saved the day (we register this as technical debt for the next iteration of the walk-in clinic).

See also:

Source: While continous integration held the day together, we are sorry to report that her work is only documented in images and in the choreography of the day but not in any of our writings.

Remember: Continous integration may be the string that holds your current software galaxy together.

“More technically, I am interested in how things bounce around in computer systems. I am not sure if these two things are relted, but I hope continuous integration will help me.”

Method: make make do

Remember:

What: Makefile as a method for quick/collective assemblages + observing amalgamates/pipelines **How:**

When:

Urgency:

Note: Note: http://observatory.constantvzw.org/etherdump/makefile.raw.html etherpad->md->pdf->anything pipeline. makefile as a method for quick/collective assemblages + observing amalgamates/pipelines CHRISTOPH

Warning:

Example:

See also:

Source:

---

[7] http://pad.constantvzw.org/p/observatory.guide.agile.yoga

Beingontheside/inthemiddle/behind

Method: Something in the Middle Maybe (SitMM)

Remember:

What: The network traffic gets observed. There are different sniffing software out there which differ in granularity and how far the user can taylor the different functionality. SitMM builds on one of these tools called [scapy](http://www.secdev.org/projects/scapy/).
How: SitMM takes a closer look at the network traffic coming from/going to a software curious person's device. The software curious person using SitMM may ask to filter the traffic based on application or device of interest. **When:**

The software curious person gets to observe their own traffic. Ideally, observing ones own network traffic should be available to anyone, but using such software can be deemed illegal under different jurisdictions.
For example, in the US wiretap law limit packet-sniffing to parties owning the network that is being sniffed or the availability of consent from one of the communicating parties. Section 18 U.S. Code 2511 (2) (a) (i) says: 1 See here for a paper [8] on the topic. Google went on a big legal spree to defend their right to capture unencrypted wireless traffic with google street view cars. The courts were concerned about wiretapping and infringements on the privacy of users, and not with the leveraging of private and public WiFi infrastructure for the gain of a for profit company. The case raises hard questions about the state, ownership claims and material reality of WiFi signals. So, while WiFi sniffing is common and the tools like SitMM are widely available, it is not always possible for software curious persons to use them legally or to neatly filter out "their traffic" from that of "others".

- When: SitMM can be used any time a software curious person feels the weight of the (invisible) networks.
- Why: SitMM is intended to be a tool that gives artists, designers and educators an easy to use custom WiFi router to work with networks and explore the aspects of our daily communications that are exposed when we use WiFi. The goal is to use the output to encourage open discussions about how we use our devices online.

  Urgency:

---

[8] http://spot.colorado.edu/ sicker/publications/issues.pdf

Note:

Warning:

Example:

Snippets of a Something In The Middle, Maybe - Report

```
UDP 192.168.42.32:53649 -¿ 8.8.8.8:53
TCP 192.168.42.32:49250 -¿ 17.253.53.208:80
TCP 192.168.42.32:49250 -¿ 17.253.53.208:80
TCP/HTTP 17.253.53.208:80 GET http://captive.apple.com/mDQArB9orEii/X
TCP 192.168.42.32:49250 -¿ 17.253.53.208:80
TCP 192.168.42.32:49250 -¿ 17.253.53.208:80
TCP 192.168.42.32:49250 -¿ 17.253.53.208:80
UDP 192.168.42.32:63872 -¿ 8.8.8.8:53
UDP 192.168.42.32:61346 -¿ 8.8.8.8:53
...
TCP 192.168.42.32:49260 -¿ 17.134.127.97:443
TCP 192.168.42.32:49260 -¿ 17.134.127.97:443
TCP 192.168.42.32:49260 -¿ 17.134.127.97:443
TCP 192.168.42.32:49260 -¿ 17.134.127.97:443
TCP 192.168.42.32:49260 -¿ 17.134.127.97:443
TCP 192.168.42.32:49260 -¿ 17.134.127.97:443
TCP 192.168.42.32:49260 -¿ 17.134.127.97:443

################################################
Destination Address: 17.253.53.208
Destination Name:    nlams2-vip-bx-008.aaplimg.com

Port: Connection Count
  80:    6

################################################
Destination Address: 17.134.127.79
Destination Name:    unknown

Port: Connection Count
 443:    2
```

```
################################################
Destination Address: 17.248.145.76
Destination Name:    unknown

Port: Connection Count
 443:   16
```

See also:

Source: *Sitm* emerges from the collective practice of the Alternative Learning Tank and is heavily inspired by projects such as [Dowse](http://dowse.equipment/), [alt.exit](http://alternativelearningtank.net/) and the [NetAidKit](https://netaidkit.net/). http://observatory.constantvzw.org/SomethingInTheMiddle/

Method: What is it like to be AN ELEVATOR?

Remember:

What: Understanding software systems by becoming them (TODO: extend this text .... how to observe software in the world around you. How to observe an everyday software experience and translate this into a flowchart) **How:** Creating a flowchart to incarnate a software system you use everyday **When:**

Urgency:

Note:

Warning: Uninformed members of the public may panic when confronted with a software performance in a closed space.

Example: What is it like to be an elevator?

For example:

what
is
it
like
to
be
an
elevator?

“from 25th floor to 1st floor”

light on button light of 25th floor

check current floor

if current floor is 25th floor

no

if current floor is . . .

go one floor up

. . . smaller than 25th floor

go one floor down

. . . bigger than 25th floor

stop elevator

turn button light off of 25th floor

turn door light on

open door of elevator

play sound opening sequence

yes

start

user pressed button of 25th floor

close door of elevator

if door is closed

user pressed 1st floor button

start timer for door closing

if timer is running more than three seconds

yes

yes

light on button

go one floor down

no

if current floor is 1st floor

update floor indicator

check current floor

stop elevator

no

yes

light off button

turn door light on

open door of elevator

play sound opening sequence

end

update floor indicator

See also:

Source: Developed by Joseph Knierzinger, Michaela Lakova + Other Members of the SSOGY Group

Method: Side Channel Analysis

Remember:

What:

How:

When:

Urgency: Side Channel attacks are possible by disregarding the abstraction of software into pure logic: the physical effects of the running of the software become backdoors to observe its functioning, both threatening the control of processes and the re-affirming the materiality of software. **Note:**

Warning: ** engineers are good guys! **

Example:

https://www.tek.com/sites/default/files/media/image/119-4146-00%20Near%20Field%20Probe

See also:

Source:

Collections / collecting

Method: Compiling a bestiary of software logos

Remember:

What: Since the early days of GNU-linux and cemented through the ubiquitous O'Reilly publications, the visual culture of software relies heavily on animal representations. But what kinds of animals, and to what effect? **How:**

Compile a collection of logos and note the metaphors for observation: - stethoscope - magnifying glass - long neck (giraffe)

When:

Urgency:

Note:

Warning:

Example:

[check Testing the testbed pads for examples] [something on bestiaries]

See also:

Source:

Method: Encounter several collections of historical hardware back-to-back

Remember:

What:

How:

This can be done by identifying one or more computer museums and visit them with little time in-between. Visiting a friend with a large basement and lots of left-over computer equipment can help. Seeing and possibly touching hardware from different contexts (state-administration, business, research, . . .), periods of time, cultural contexts (California, Germany, French-speaking Belgium) and price ranges allows you to sense the interactions between hardware and software development.

When:

Urgency:

Note: It's a perfect way to hear people speak about the objects and their contexts, how they worked or not and how objects are linked one with another. It's also showing economic and cultural aspects of softwares.

Warning: **DO NOT FOLD, SPINDLE OR MUTILATE**

At one point during the demonstration of a Bull computer, the guide revealed the system's "software" – a suitcase sized module with dozens of patch cords. She made the comment that the term "spaghetti code" (a derogatory expression about early code usign many "GOTO" statments) had its origin in this physical arrangement of code as patchings.

Preserving old hardware in order to observe physical manifestation of software. See software here : we did experienced the incredible possibility of actually touching software.

Playing with the binary. Bull cards. Happy operator! Punch card plays. "The highlight of the collection is to revive a real punch card workshop of the 1960s."

Collection de la Maison des critures d'Informatique & Bible, Maredsous
The particularity of the collection lies in the fact that it's the conservation of multiple stages of life of a software since its initial computerization until today. The idea of introducing informatics into the work of working with/on the Bible (versions in Hebrew, Greek, Latin, and French) dates back to 1971, via punch card recordings and their memorization on magnetic tape. Then came the step of analyzing texts using computers.

See also: http://pad.constantvzw.org/p/observatory.guide.jean.heuns

Source: http://www.histoireinform.com/Histoire/+Infos/jmclcadr.htm

Method: Testing the testbed: testing software with observatory ambitions (SWOA) **Warning:** this method may make more sense if you first take a look at the [Something in the Middle Maybe (SitMM)](http://pad.constantvzw.org/p/observatory.guide.s which is an instance of a SWOA **How:** The interwebs hosts many projects that aim to produce software for observing software, (from now on Software With Observatory Ambitions (SWOA)). A comparative methodology can be produced by testing different SWOA to observe software of interest. Example:

use different sniffing software to observe wireless networks, e.g., wireshark vs tcpdump vs SitMM. Comparing SWOA reveals what is seen as worthy of observation (e.g., what protocols, what space, which devices), the granularity of the observation (e.g., how is the observation captured, in what detail), the logo and conceptual framework of choice etc. This type of observation may be turned into a service (See also: Something in the Middle Maybe (SitMM)). **When:** Ideally, SWOA can be used everywhere and in every situation. In reality, institutions, laws and administrators like to limit the use of SWOA on infrastructures to people who are also administering these networks. Hence, we are presented with the situation that the use of SWOA is condoned when it is down by researchers and pen testers (e.g., they were hired) and shunned when done by others (often subject to name calling as hackers or attackers). **What:** Deep philosophical moment: most software has a recursive observatory ambition (it wants to be observed in its execution, output etc.). Debuggers, logs, dashboards are all instances of software with observatory ambitions and can not be separated from software itself. Continuous integration is the act of folding the whole software development process into one big feedback loop. So, what separates SWOA from software itself? Is it the intention of observing software with a critical, agonistic or adversarial perspective vs one focused on productivity and efficiency that distinguishes SWOA from software? What makes SWOA a critical practice over other forms of sotware observation. If our methodology is testing SWOA, then is it a meta critique of critique? **Urgency:** If observation is a form of critical engagement in that it surfaces the workings of software that are invisible to many, it follows that people would develop software to observe (SWOAs). Testing SWOAs puts this form of critical observation to test with the desire to understand how what is made transparent through each SWOA also makes things invisible and reconfigures power. **Note:** Good SWOA software usually uses an animal as a logo.:D

Warning: Many of the SWOA projects we looked at are promises more than running software/available code. Much of it is likely to turn into obsolete gradware, making testing difficult. **Remember:**

Source: By/history/source/origin/process: the ”original testbed” was proposed by researchers/collaborators at Princeton University. Testing this testbed at a local Constant workshop led to the meta-project on testing software meant for testing software.

See also: making a bestiary of visual cultures/logos around it **See also:** http://pad.constantvzw

See also: http://pad.constantvzw.org/p/observatory.guide.sitmm

Method: Prepare a reader to think theory with software

Remember:

What: Compile a collection of texts about software. **How:** Choose texts from different realms. Software observations are mostly done in the realm of the technological and the pragmatic. Also the ecology of texts around software includes first and foremost manuals, technical documentation and academic papers by software engineers and these all 'live' in different realms. More recently, the field of software studies opened up additional perspectives fuelled by cultural studies and sometimes filosophy. By compiling a reader ... ways of speaking/writing about. Proximity. **When:**

Urgency:

Note:

Warning:

Example:

Pull some quotes from the reader, for example from the chapter: Observation and its consequences

Lilly Irani, Hackathons and the Making of Entrepreneurial Citizenship, 2015 http://sci-hub.bz/10.1177/0162243915578486

Kara Pernice (Nielsen Norman Group), Talking with Participants During a Usability Test, January 26, 2014, https://www.nngroup.com/articles/talking-to-users/

Matthew G. Kirschenbaum, Extreme Inscription: Towards a Grammatology of the Hard Drive. 2004 http://texttechnology.mcmaster.ca/pdf/vol13_2_06.pdf

Alexander R. Galloway, The Poverty of Philosophy: Realism and Post-Fordism, Critical Inquiry. 2013, http://cultureandcommunication.org/galloway/pdf/Galloway,%20Poverty%

Edward Alcosser, James P. Phillips, Allen M. Wolk, How to Build a Working Digital Computer. Hayden Book Company, 1968. https://archive.org/details/howtobuildaworking

Matthew Fuller, “It looks like you’re writing a letter: Microsoft Word”, Nettime, 5 Sep 2000. https://library.memoryoftheworld.org/b/xpDrXE_VQeeuDDpc5RrywyTJwbzD8eatYGHKmyT2A_HnIH

Barbara P. Aichinger, DDR Memory Errors Caused by Row Hammer. 2015 www.memcon.com/pdfs/proceedings2015/SAT104_FuturePlus.pdf

Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, Ruby B. Lee. Last-Level Cache Side-Channel Attacks are Practical. 2015 http://palms.ee.princeton.edu/system/files/SP_vfinal.pdf

See also: http://pad.constantvzw.org/p/observatory.guide.samequestion

Source: http://pad.constantvzw.org/p/observatory.reader

==========

Colophon

The Guide to technogalactic software observing was compiled by Carlin Wing, Martino Morandi, Peggy Pierrot, Anita, Christoph Haag, Michael Murtaugh, Femke Snelting

License: Free Art License

Support:

Sources:

Constant, February 2018