# Presentation slides — notes export

29.9.2018

# 1 Pursuit of a systematic interconnection

*tell: As you might have already gathered from the abstract, I'd like to motivate a certain interconnection of these three, at first sight rather loosely related topics:*

*tell: — while ‚translating' them to practical demonstrations of:*

*tell: Now to give a preliminary clue:*

- *org-mode is ...*
- *Docker is ...*
- *reproducible research refers to ...*

## 1.1 Literate Programming

- »

### 1.1.1 `org-mode`

**an ever more popular `major-mode` of the notorious GNU EMACS**

## 1.2 Containerisation

- »

### 1.2.1 `Docker`

**the industry-leading `container platform`**

## 1.3 Digital Humanities

- »

### 1.3.1 `reproducible research`

**a set of yet to come `methodological requirements` already established in the natural sciences**

1. possibly to be modified

   this relates to the future of DH

# 2 Theoretical approach

*tell: More important, still, than rushing through some always tentative examples, is a sound theoretical approach that shall provide a common ground for the sought interconnection.*

*tell: Remark: Here, I am especially happy that this particular workshop focuses on programming languages — and notations!*

*tell: Since with reference to notation the approach taken up in the following shall **literally** set out from the very cradle of the overall project [PROGRAMme] — asking „What a programme (etymologically and by extension ontologically) is.“*

## 2.1 Context: Programming Languages and *Notations*

### 2.1.1 Mark off the period

- when writing becomes autonomous—or, less drastic, `self`-contained.
- when the subject of a PROGRAMme turns recursive,
  - when it turns back *into* it`self`.

*tell: Here, I can only hint how this period might be delineated.*

**Historically,**

we are back a the antagonism between

- Tuning's and Zuse's tape machines

and

- the female programmer's diagrams written before to patch the ENIAC

**Discursively,**

there are many forerunners—clearly—before a rather late

- computer-awareness in the `self`-conception of the Humanities
- attached to them`selves` the prefix `digital`.

1. One of those,

   — using `EMACS`, long before `org-mode` came along —

   appears worth to be cited here:

> *tell: The quote is taken from the preface of the article collection „Draculas Vermächtnis. Technische Schriften.“ (Dracula's legacy. Technical writings).*
> *That is, from the collection containing the likewise renowned as provocative articles: „There is no software“ and „Protected Mode“*

> *tell: These articles are now available in Kittler, Friedrich A. & Gumbrecht, Hans Ulrich, The truth of the technological world: essays on the genealogy of presence, Stanford, California, Stanford University Press, 2013.*

   Kittler, Friedrich, Draculas Vermächtnis: technische Schriften, 1. Aufl., Leipzig, Reclam, (Reclam-Bibliothek, Bd. 1476), 1993.

   my translation:

   > „Who, then, at the same day writes ‚technical writings‘ and nevertheless optimises two, three assembler routines on his graphics machine, cannot avoid to ascertain that meanwhile the differences between writing and programming are virtually zero. It is the same machine, on which texts and virtual images are rendered, the same hunting for redundancies [...]“

   the original:

   > Schon wer selbigen Tags Technische Schriften schreibt und trotzdem noch zwei, drei Assemblerroutinen seiner Graphikmaschine optimiert, kann nicht umhin festzustellen, daß die Unterschiede zwischen Schreiben und Programmieren mittlerweile gegen Null gehen. Es ist dieselbe Maschine, auf der Texte und virtuelle Bilder entstehen, dieselbe Jagd nach Redundanzen, [...]

   [?, ]5]Kittler 1993.

> *tell: With this insight, I shall break free the minute word and commence or rather continue with some demonstrations.*

## 2.2 Literally setting out from the very word PROGRAM*me*

*tell: Obviously, it is a combination of the prefix PRO- and GRAMME.*
*Therefore, and still in English ...*

### 2.2.1 Combination of the prefix PRO- and GRAMME

**Still in English,**

- the prefix PRO- synonymously translates to pre-
- and GRAMME to something written or inscribed.

Σ: Literally referring to something **'pre(in)scribed'.** Most simply put: a **thing written before.**

*tell: Thus metaphorically, it can mean...*

Metaphorically,

- the title of a prescription
- the order of the day, that is, an agenda
- a public proclamation, e.g. of a law

—set out in front of yourself
All these meanings are documented already for Ancient Greece:

- Henry George Liddell, Robert Scott, A Greek-English Lexicon, πρόγραμμα

### 2.2.2 Rooted in the Greek word: ΠΡΟ-ΓΡΑΜΜΑ

**Etymologically,**

—to be sure—
the verb γράφειν means

1. originally
    - to scratch, to scrape, to gaze. most generally put:
        - to draw
2. and only secondarily
    - to write tokens or characters

**However,**

— as soon as —
the prefix ΠΡΟ- is added in front that **thing written before**,

- the reference is always something distinct or discrete.

| *tell: That is to say, that then ...* |
|---|

Then, the word ΓΡΑΜΜΑ **always and only** involves the secondary meaning of γράφειν:

- `writing down letters`
  - which—in the Greek context—can also be `digits` or even musical notes.

**What gets ‚blurred away' in English**

— thus —
for a PRO-GRAMME as something **pre(in)scribed** or as a **thing written before**
*is*
that the prefix ΠΡΟ- unerringly confines the notated thing to the distinctness and discreteness of the Greek writing system: `the alphabet`
In this way, any programme hits on the well known Greek breach
between

- written `characters` or `digits`

and

- drawn `lines`.

by extension, between

- numerical mathematics or number theory

and

- geometry

1. Understanding the `'digital'` *dia*-grammatically

| *tell: To cut things short, one may point to the philosopher Nelson Goodman...* |
|---|

| *tell: ... who's „Languages of Art"*<br>*—with its basic differentiation between autographs and allographs—*<br>*has tried to draw from the same breach in order to define „notational languages".* |
|---|

*tell: His conception of the ‚digital' might be helpful to understand diagrams as programmes.*

Basic differentiation between

- autographs

and

- allographs

Goodman, Nelson, Languages of Art. An Approach to a Theory of Symbols, 2. Aufl., Indianapolis, Hackett Publishing Company, 1976, p. 163ff.:

> Many topology diagrams, for example, only need to have the correct number of points or joints connected by lines according to the correct scheme. Here, the points and lines act as characters in a `notational language`. These diagrams, as with electrical circuits, are purely `digital`.

Yet, after all, his theory of symbols boils down to a theory of authenticity or forgery in the arts—simply because:

If a notation `(read: a pro-gramme)` can be specified,

- a piece of art
- as something **pre(in)scribed**

remains identical and thus cannot be forged.

2. More important though

— in the original context of notation —

is what can be gathered from the opposite,

- from the NEGATION of γράφειν in the sense of `PRO-GRAMME`,
- from that which is , incapable of being written:

Henry George Liddell, Robert Scott, An Intermediate Greek-English Lexicon, ἀγράμματος

- without learning

Henry George Liddell, Robert Scott, A Greek-English Lexicon, ἀγράμμ-α^τος

- I. the illiterate
- III.
  a) of animals, unable to utter articulate sounds (Arist.HA488a33)
  b) incapable of being written, Porph.Abst.3.3, cf. Eustr.in

c) of sounds, inarticulate (Id.Int.16a29, D.L.3.107) a song without words (Phld.Po.2Fr.47.22.)

With all these references to sound, one thing sticks out unmistakably:

- this Greek thing that we may recognise as the epistemological basis of the alphabet,
- that is, its unique feature to be
  - an elementary analysis
  - distinct and discrete
  - a phonetic `code`.

In other words,

- the Greek writing system is the literate PROGRAM*me* to educate the *self*;
- it is the literate cradle to grow that, what later shall be called the Humanities.

**That is to state one basic thing:**

**Not any** grammatology or semiotics will do as an approch to tell „what a `PRO-GRAMME` is".

*tell: ... and certainly not the one best known in France!*

*tell: You may call this approach Kittlerian and, basically, I would agree.*

*tell: However, there are modifications to be made — as we shall see later on.*
*Anyway ...*

A programme *is not*

- a system of abstract signs,
- nor accessible by a theory of symbols.

Finally, realising that
— at least etymologically —

1. a PROGRAMme essentially involves a literate (en-)coding system
   - to be physically written down

   and
   - to be read or deciphered

2. by some ‚self‘
3. before it can cause any pro-grammed effect,

also allows the jump back form this **common ground** of ancient Greece in order to
— ontologically —
mark off what a `PROGRAMme`

- in the *technical* context of computation.

is.

- Context: Programming Languages and Notations

# 3 `org-mode` as a writing and notation system

*tell: Because the some thing that struck Kittler holds true for `org-mode` — only — in potency!*

*tell: And because — as you have realised since long —*
*all that what I have been doing until now was already demonstrating `org-mode`*
*as built on top of stock EMACS' `outline-mode`.*

## 3.1 The historical core: `outline-magic.el`

*tell: In fact, as the inventor, Carsten Dominik, tells us:*

The Org Manual: History and Acknowledgments

> Org was born in 2003, out of frustration over the user interface of the Emacs Outline mode [...]
>
> Visibility cycling and structure editing were originally implemented in the package 'outline-magic.el', but quickly moved to the more general 'org.el'. As this environment became comfortable for project planning, the next step was adding TODO entries, basic timestamps, and table support. These areas highlighted the two main goals that Org still has today: to be a new, outline-based, plain text mode with innovative and intuitive editing features, and to incorporate project planning functionality directly into a notes file.

## 3.2 PROGRAM*me* your*self*!

— the `org` agenda dispatcher: ~~C-c a~~

## 3.3 Starting from a major mode for orga & notes taking

`org-mode` has been ever more growing by

- a huge community
- of open-source enthusiast.

*tell: That much, that one might even say: by today ...*

### 3.3.1 `org-mode` has become the main attractor

- to delve into
- to cope with

and, finally,

- to master EMACS

#### Now, EMACS is not just an editor, nor an application

— that just doesn't like the mouse —
EMACS *is*

- a full fetched `lisp machine`
  - an interpreter
    - with byte-compiled libraries
  - a programming environment
    - with total reflection.

1. Since everything is plain text

   `org-mode`'s mark-up (just like EMACS' `font-lock` for syntax coloring) heavily empoys

   - regular expression searches
     - just in time.

```
(save-excursion
  (while (re-search-backward "EMACS" nil t)
    (sit-for 2)))
```

2. Yet, the mark-up of `org-mode` structures the text
   into far more detailed elements than hierarchical outlines.
   It has
   - an efficient add-hoc parser

   and, since lately,
   - a lightning fast cache.

   For example:
   Get the smallest element at the current cursor position:

```
(org-element-at-point)
```

   Or, get the local element in context:

```
(save-excursion
  (org-back-to-heading)
  (forward-char 15)
  (org-element-context))
```

## 3.4 support for inline math snippets creation

If $a^2 = b$ and $b = 2$, then the solution must be either

$$a = +\sqrt{2}$$

or

$$a = -\sqrt{2}$$

.

$$F(f) = \int_{-\infty}^{\infty} \left( f(t) \cos\left(ft\right) + f(t) \sin\left(ft\right) \right) \mathrm{d}t$$

$$\phi(t,\tau) = \lim_{\tau \to \infty} \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) f(t+\tau) \, \mathrm{d}t$$

## 3.5 multi-format export of documents

Call the export dispatcher: ~~C-c C-e~~

# 4 org-mode as *the* environment for LP

## 4.1 Classic Literate Programming

*tell: Just like TEX and METAFONT, still at the core of LATEX today,*
*the original concept of Literate Programming stems*
*from an outstanding „American computer scientist“: Donald Ervin Knuth*

*tell: The WIKI just knows him as:*

Donald Knuth - PROGRAMme „American computer scientist, father of Analysis of Algorithms.“

### 4.1.1 The simple but telling slogon of WEB

*tell: strikes ...*

Don't comment, write a book!

Knuth, D. E., Literate Programming, in: The Computer Journal, 27 (1984), S. 97–111.

#### Possibly in need and now the backbone of TAOCP

The Art of Computer Programming — since 1968 — still incomplete.

- Knuth, Donald E., The Art of Computer Programming.
  - The Art of Computer Programming
- [?, ]1]Knuth, Donald Ervin, The art of computer programming, Upper Saddle River, NJ, Addison-Wesley, 2005.

### 4.1.2 The concepts rests on two procedures

#### 'tangling'

automated extraction of code from source blocks

**'weaving'**

macro-controlled interweaving of code from source blocks

### 4.1.3 A simple example

**for tangling (and weaving)**

```
1 echo "About executing the findmnt command:\n"
2 findmnt -u
3 echo "\ndone!"
```

1. let's see

```
1 cd ~/Git.gitSCM/Arbeit/Lit_DH.gitSCM/demo
```

a) A true relief

> *tell: not only for book authors on the Art of Programming*
> *but also for humble system administrators.*

for system administration: tangle `/etc/fstab`

## 4.2 the `org-babel` framework

> *tell: The `org-babel` framework of `org-mode` not only implements,*
> *but extends the two procedures in many ways, and in such ways practically impossible back in the 80ies.*

> *tell: Among the most important features rank:*

### 4.2.1 `tangling` of code from *source-blocks*

### 4.2.2 ‚live-execution' of many interpreted languages

- Babel: Languages

### 4.2.3 parametrised `weaving` of *source-blocks* ‚on the fly'

### 4.2.4 cross-connecting results of *source-blocks*

make the literal

- output of any one language
- the input of any other language

# 5 Containerisation

*tell: Now, that we are stranded with system administration,*
*it is high time to take a look at containers.*

*tell: As said, this will be done by example of `Docker`,*
*the by now industry-leading container platform*
*that has caused nothing less but a severe revolution*
*of how to deal with machines and software! by their famous promise:*

## 5.1 Build, share and run any application, anywhere

- Container Platform
- Modern Application Architecture for Enterprise - Jan 2016.pdf — **CaaS**
- Build_Ship_Run

### 5.1.1 `Docker 'Engine'` — container runtime

Container Runtime

> Docker Engine enables containerized applications to run anywhere consistently ~~on any infrastructure~~, solving "dependency hell" for developers and operations teams, and eliminating the "it works on my laptop!" problem.

- Docker_Engine

### 5.1.2 `Docker Desktop` — virtual machine running `containerd`

A strategic product

- wrapping the `Linux` kernel into virtual machine
  - with a GUI app for `macOS` and `Windoze`

Docker Desktop for Mac and Windows

### 5.1.3 `Docker Hub` — container image library

Docker Hub - Container Image Library

## 5.2 Two simple, already running examples

### 5.2.1 A mini `Docker Swarm` cluster

Consisting of

- the `whoami` service
- running as a container on 3 machines at the Ionian University at Corfu

whoami - Docker Hub
They are accessed through the load-balancing frontend `traefik` traefik - Docker Hub

- `https://traefik.larigot.linkpc.net/dashboard/`
- `https://larigot.linkpc.net/whoami`

fallback

- `https://traefik.antidoron.linkpc.net/dashboard/`
- `https://antidoron.linkpc.net/whoami`

### 5.2.2 Milkypostman's Emacs Lisp Package Archive

Docker Hub is not the only container provider:

- melpa/melpa: Recipes and build machinery for the biggest Emacs package repo
- MELPA @home

## 5.3 But what are containers?

### 5.3.1 In essence: the next step in virtualisation technology

- Virtual Machines

versus

- Docker Containers

## 5.4 Demystifying `dockerd`

— the master daemon process managing the `container runtime` —

### 5.4.1 by the core technology

The strategy of virtualisation is nothing new at all.

#### chroot

Neither is the isolation of filesystems or precesses to achive it.

#### `LXC` vs. `dockerd`

If an isolated filesystem gets added to the `kernel` such an aggregation is called

1. either a `system container`
   Each instance of a `Linux container` (LXC)
   - runs a full and fully virtualised operating system on top of the `kernel`.
2. or an `application container`
   Whereas a container spawned by `dockerd`
   - is just an ordinary user process with seperated `namespaces`
     - pid
     - mnt
     - net

     and resource limitations controlled by `cgroups`
     - CPU drain
     - memory usage
     - block I/O restrictions

### 5.4.2 by history

— Now —

1. `namespaces` for root processes date back to the `kernel 2.4.19` from 2002.
2. `cgroups` reached the mainline `kernel 2.6.24` by 2008.
3. But `application container` virtualisation relies on `user namespaces`
   - only finished by the release `3.8` in February 2013.

**In fact**

Linux Container Technology (LCT) was ready

- one month before Docker
  - was introduced by Solomon Hykes at his French company ~~dotCloud.~~

Setting out as an high-level feature-wrapper around LXC

- an API for user application insulation was soon achieved by libcontainer,
  - the core library of dockerd.

1. That's the docker story
   — technically!

## 5.5 What containerisation means

Containerisation sings the old liberation song of software louder then ever.

Containerisation is *the* reverse-provocation of the provocative statement: There is no software.

Carfully avoiding any reference to

- ‚bare matel' or hardware requirements,

but instead,

- replacing it by „any infrastructure",

containerisation articulates the ultimate deprivation of the machine.

To get rid off dependencies, containerisation is the very opposite of redundency eradication.

Nonetheless, with the isolation of containerisation and the ‚independency' it provides, comes a freedom of choice that supports and enforces open-source projects through competition.

### 5.5.1 Finally, one may say:

If there is not even software anymore,

- but only services (CaaS),
  - also the pro-gramme is gone away.

Sofware agnostic orchestration of services eliminates the PROGRAMme.

# 6 Digital Humanities

Well, in the beginning I already said

- that the Humanities depend on the media technology of the alphabet.
  - That might appear trivial.

That `Digital` Humanities are widely understood as

- the digitisation of our cultural heritage
  - is trivial as well.

## 6.1 No `Digital Physics` or `Digital Mathematics`?

*tell: Did you ever ask the question why there is no such thing as `Digital Physics` or `Digital Mathematics`?*

There might be computational physics or discrete mathematics.
Pretty sensible and telling terms.
But what about the `'digital'` of the Humanities?
Is there something like a concept of the digital?

## 6.2 An umbrella term for selling books and do fund raising

*tell: It is pretty devastating to learn*
*that the term — not only practically — but also historically*
*is nothing but a suitable signifier for an effective fund raising.*

### 6.2.1 „A Companion to Digitized Humanities.“

Marketing enforcement for a wider range of readers at Blackwell Publishing.

- Schreibman, Susan, Siemens, Ray & Unsworth, John, Companion to Digital Humanities (Blackwell Companions to Literature and Culture), Hardcover., Oxford, Blackwell Publishing Professional, Blackwell Companions to Literature and Culture, 2004.
  - A Companion to Digital Humanities

For the time around **November 2001**, one of the editors reports:

> „[...] the editorial and marketing folks at Blackwell wanted "Companion to Digitized Humanities." I suggested "Companion to Digital Humanities" to shift the emphasis away from simple digitization."
>
> [John Unsworth, 5. April, 2010 E-mail to Matthew Kirschenbaum]

[?, ]3]Kirschenbaum, Matthew G, What Is Digital Humanities and What's It Doing in English Departments?, in

#### Blackwell Publishing

- Wiley-Blackwell - Wikipedia Wiley-Blackwell is the international scientific, technical, medical, and scholarly publishing business of John Wiley & Sons. It was formed by the merger of John Wiley's Global Scientific, Technical, and Medical business with Blackwell Publishing, after Wiley took over the latter in 2007

### 6.2.2 „economy of scale" — or — „creating an umbrella entity for themselves"

> „As anyone who has ever tried to run a scholarly organization will know, economies of scale are difficult to come by with only a few hundred members and so the thought was to consolidate and share infrastructure and services."

[?, ]3]Kirschenbaum, Matthew G, What Is Digital Humanities and What's It Doing in English Departments?, in

## 6.3 The turn away from

*tell: But, actually, it should have been a lable to turn away from pure*

### 6.3.1 „Humanities Computing“

**pioneering work reaches way back into the 1930s and 1940s**

> The field's background in humanities computing typically, but far from exclusively, results in projects that focus on computing methods applicable to textual materials.

[?, ]2]Fitzpatrick, Kathleen, The Humanities, Done Digitally, in

1. reaches way back
    - A Companion to Digital Humanities
        - Schreibman, Susan, Siemens, Ray & Unsworth, John, Companion to Digital Humanities (Blackwell Companions to Literature and Culture), Hardcover., Oxford, Blackwell Publishing Professional, Blackwell Companions to Literature and Culture, 2004.

**first specialised journals**

- 1966: Computers and the Humanities
- 1977: Association for Literary and Linguistic Computing (ALLC)
- 1978: Association for Computers and the Humanities (ACH)

### 6.3.2 „changes that digital technologies are producing“

> However, when many of us hear the term „digital humanities“ today, we take the referent to be not the specific subfield that grew out of humanities computing but rather the changes that digital technologies are producing across the many fields of humanist inquiry.

[?, ]2]Fitzpatrick, Kathleen, The Humanities, Done Digitally, in

## 6.4 Creating a self-awareness of ‚digital culture‘

### 6.4.1 by means of computational methods

- `http://voyant-tools.org/`
    - Voyant Tools PROGRAMme
        * GitHub - sgsinclair/Voyant

*tell: Is copy and pasting text for some analysis enough?*

*Is this sustainable?*
*Doesn't this ask for some proper reproducible research?*

#### Distant Reading

- Moretti, Franco, Distant Reading., 2016. Aufl., Konstanz, Konstanz University Press, 2016.
  - Production, bücher de IT and, Distant Reading (eBook, ePUB)

#### What Does Data Want?

- Manovich, Lev, What Does Data Want?

### 6.4.2 but also as an archaeology

of how a technologically ever more outdated humanism

- transforms into an ever more post-human and globalised world.

#### fights between ‚analysis' and ‚reflection'

> It's clear that there's an overlap between these fields [digital-media studies] and that which has been called „digital humanities" — between scholars who use digital technologies in studying traditional humanities objects and those who use the methods of the contemporary humanities in studying digital objects — but clear differences lie between them. Those differences often produce significant tension, particularly between those who suggest that digital humanities should always be about making (whether making archives, tools, or new digital methods) and those who argue that it must expand to include interpreting.

[?, ]3]Fitzpatrick, Kathleen, The Humanities, Done Digitally, in

## 6.5 Critical code studies

*tell: Most interesting for our present focus on „Languages and Notations" of today,*
*surely are the recent sub-disciplines*
*calling themselves software studies or critical code studies.*

### 6.5.1 Looking at the forerunners of `Digital Humanities`

*tell: And who do they discover among those of the highest rank?*

> Within the broad field of intersections between Computer Sciences and Humanities comprised by the Digital Humanities today, Kittler's teachings may well grant him the title of a forefather.

DHQ: Digital Humanities Quarterly: Friedrich Kittler's Digital Legacy – PART II - Friedrich Kittler and the Digital Humanities: Forerunner, Godfather, Object of Research. An Indexer Model Research

#### There is Kittler's software!

*tell: So, one last demo that, I hope, will bring it all together.*

But how shall one study his highly optimised assembler code,

- if we don't have his old machines
- nor that we can install the by now outdated libraries that he used?

# 7 Conclusions

## 7.1 The re-invocation of `Literate Programming` methods

— as seen by `org-mode` today —
turns
**meta-instructions**

- of what a certain programme is all about
- or why it was written just this and not another way

into a
**meta-programming system**

- of handling code in general
- for an on-line orchestration of services
- and—last but not least—of yourself

> *tell: As such a meta-programming system, I suppose,*
> *it becomes a hot candidate for the topic*
> *„Notations not just as formal languages“*
> *as proposed on the WIKI*

## 7.2 As a meta-programming system

it becomes a hot candidate for the topic

- Notations not just as formal languages - PROGRAMme

## 7.3 In the spirit of Knuth

### 7.3.1 we may conclude

**that a `PRO-GRAMME`**

1. as opposed to an application or service
    a) is there
        i. where things become literate — again.