What is textual variance in the eyes of a software technologist?

Manfred Thaller, Brühl, Germany

March 15th, 2016

Köln: Digital Editions

Two puzzles

(1) Virtually all software is able today to ignore the differences between upper and lower case; problems with national characters like “ä” are still quite frequent.

Why?

(2) Most children at elementary schools have extensive experience using XML.

Huh?

Upper and lower case

a	097	A	65
b	098	B	66
c	099	C	67
d	100	D	68
e	101	E	69
f	102	F	70
g	103	G	71
h	104	H	72
i	105	I	73
j	106	J	74
k	107	K	75
…	…	…	…

Upper and lower case

a	01100001	A	01000001
b	01100010	B	01000010
c	01100011	C	01000011
d	01100100	D	01000100
e	01100101	E	01000101
f	01100110	F	01000110
g	01100111	G	01000111
h	01101000	H	01001000
i	01101001	I	01001001
j	01101010	J	01001010
k	01101011	K	01001011
…	…	…	…

Upper and lower case

a	01100001	A	01000001
b	01100010	B	01000010
c	01100011	C	01000011
d	01100100	D	01000100
e	01100101	E	01000101
f	01100110	F	01000110
g	01100111	G	01000111
h	01101000	H	01001000
i	01101001	I	01001001
j	01101010	J	01001010
k	01101011	K	01001011
…	…	…	…

Upper and lower case

All of this is completely arbitrary, however:

lower(x) = upper(x) | '00100000'

== fastest existing operation on any chip

- therefore supported by almost all programming languages
- therefore almost no cost in supporting in any program.

Observation 1

If people designing a basic technical tool are aware of a problem, all applications derived from that basic solution will find it easy to solve that problem.

If the don’t, they won’t.

XML …

Most children at elementary schools have extensive experience using XML.

When they type

What my family did last Sunday.

into Microsoft Word …

XML …

… it's stored like this (abbreviated):

<w:body><w:p w:rsidR="003E7AE5" w:rsidRPr="003E7AE5" w:rsidRDefault="003E7AE5" w:rsidP="003E7AE5"><w:pPr><w:pStyle w:val="StandardWeb"/><w:spacing w:before="200" w:beforeAutospacing="0" w:after="0" w:afterAutospacing="0" w:line="216" w:lineRule="auto"/><w:rPr><w:lang w:val="en-GB"/></w:rPr></w:pPr><w:r><w:rPr><w:rFonts w:ascii="Freestyle Script" w:eastAsia="+mn-ea" w:hAnsi="Freestyle Script"/><w:color w:val="000000"/><w:kern w:val="24"/><w:sz w:val="80"/><w:szCs w:val="80"/><w:lang w:val="en-GB"/></w:rPr><w:t xml:space="preserve">What my family did last Sunday. </w:t></w:r></w:p><w:p w:rsidR="00EB3D3C" w:rsidRPr="003E7AE5" w:rsidRDefault="003E7AE5"><w:pPr><w:rPr><w:lang w:val="en-GB"/></w:rPr></w:pPr><w:bookmarkStart w:id="0" w:name="_GoBack"/><w:bookmarkEnd w:id="0"/></w:p><w:sectPr w:rsidR="00EB3D3C" w:rsidRPr="003E7AE5"><w:pgSz w:w="11906" w:h="16838"/><w:pgMar w:top="1417" w:right="1417" w:bottom="1134" w:left="1417" w:header="708" w:footer="708" w:gutter="0"/><w:cols

XML …

… it’s stored like this (abbreviated):

<w:body><w:p w:rsidR="003E7AE5" w:rsidRPr="003E7AE5" w:rsidRDefault="003E7AE5" w:rsidP="003E7AE5"><w:pPr><w:pStyle w:val="StandardWeb"/><w:spacing w:before="200" w:beforeAutospacing="0" w:after="0" w:afterAutospacing="0" w:line="216" w:lineRule="auto"/><w:rPr><w:lang w:val="en-GB"/></w:rPr></w:pPr><w:r><w:rPr><w:rFonts w:ascii="Freestyle Script" w:eastAsia="+mn-ea" w:hAnsi="Freestyle Script"/><w:color w:val="000000"/><w:kern w:val="24"/><w:sz w:val="80"/><w:szCs w:val="80"/><w:lang w:val="en-GB"/></w:rPr><w:t xml:space="preserve">**What my family did last Sunday.**</w:t></w:r></w:p><w:p w:rsidR="00EB3D3C" w:rsidRPr="003E7AE5" w:rsidRDefault="003E7AE5"><w:pPr><w:rPr><w:lang w:val="en-GB"/></w:rPr></w:pPr><w:bookmarkStart w:id="0" w:name="_GoBack"/><w:bookmarkEnd w:id="0"/></w:p><w:sectPr w:rsidR="00EB3D3C" w:rsidRPr="003E7AE5"><w:pgSz w:w="11906" w:h="16838"/><w:pgMar w:top="1417" w:right="1417" w:bottom="1134" w:left="1417" w:header="708" w:footer="708" w:gutter="0"/><w:cols

Observation 2

If people designing a mark up scheme have a clear conceptual model of what they are marking up, all applications operating on that markup scheme, will find it easy to implement all operations implied by that model.

If they don’t, they won’t.

Rumblings of a coder …

contains(QString *string*, Qt::CaseSensitivity *cs* = Qt::CaseSensitive)

myString = “Lorem ipsum **episcopus** sit amet, …“

if (myString.contains(”Episcopus”,false) == true) …

➔ “true”

if (myString.contains(”Episcopus”,true) == true) …

➔ “false”

Rumblings of a coder …

contains(QString *string*, Qt::CaseSensitivity *cs* = Qt::CaseSensitive)

myString = “Lorem ipsum **episcopus** sit amet, …“

Does “myString” contain ”Episcopus” ? / Ignore lower / upper case

➔ “true”

Does “myString” contain ”Episcopus” ? / Consider lower / upper case

➔ “false”

Musings of a coder …

contains(QString *string*, DH::Levenshtein *maxLev* = int)

myString = “Lorem ipsum **episkopus** sit amet, …“

if (myString.contains(”episcopus”,1) == true) …

➔ “true”

if (myString.contains(”episcopus”,0) == true) …

➔ “false”

Musings of a coder …

contains(QString *string*, DH::Levenshtein *maxLev* = int)

myString = “Lorem ipsum **episkopus** sit amet, …“

Does “myString” contain ”episcopus” ? / Levenshtein <= 1

➔ “true”

Does “myString” contain ”episcopus” ? / Levenshtein <= 0

➔ “false”

Dreams of a coder …

How about:

Check whether the text in chapter 25 contains episcopus in at least 50 % of all witnesses.

if (myText.chapter25contains(”episcopus”,0.50)) == true) …

Please write 500 times:

No, *I* am **not** supposed to be able to write a program containing

```
if  (myText.chapter25contains(”episcopus”,0.50)) == true) …
```

But if programmers are able to do so, all the programs they write will easily understand about *witnesses* in the editorial sense.

If they don’t, they won’t.

But why?

Immanuel Norman: *Digitale Editionen als Web-Services,* in: Konferenzabstracts DHd2016, http://dhd2016.de/boa-large.pdf (March 11th, 2016), 209.

Idea: Permanent references between “texts”.

➔ How about references between other services – e.g. data bases - and texts?

This text contains variations …

Lorem ipsum dolor sit amet. Henricus **episcopus** de aliquo fecit aliquis, ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Lorem ipsum dolor sit amet. Henricus **episkopus** de aliquo fecit aliquis, ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Lorem ipsum dolor sit amet. Henricus **comes** de aliquo fecit aliquis, ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

… or:

{ Lorem ipsum dolor sit amet. } { Henricus { **episcopus | episkopus | comes** } de aliquo fecit aliquis } {, ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.}

… derive a database of persons, containing …

Nomen ::= { Henricus }

Appelatio ::= { episcopus | episkopus | comes }

Origo ::= { aliquo }

Actio ::= { fecit aliquis }

… to support research questions like …

Nomen ::= { Henricus }

Appelatio ::= { episcopus | episkopus | comes }

Origo ::= { aliquo }

Actio ::= { fecit aliquis }

Give me all occurrences of *Henricus*

who is an *episcopus* { insisting on a specific spelling || disregarding the spelling }.

… or…

Nomen ::= { Henricus }

Appelatio ::= { episcopus | episkopus | comes }

Origo ::= { aliquo }

Actio ::= { fecit aliquis }

Give me all occurrences of *Henricus*

who is an *episcopus* in the majority of the textual tradition.

… or…

Nomen ::= { Henricus }

Appelatio ::= { episcopus | episkopus | comes }

Origo ::= { aliquo }

Actio ::= { fecit aliquis }

Give me all occurrences where the textual tradition arrives at a common profile, replacing a clerical rank with a secular, however.

… that is (technical version):

Administer in the database, which contains *factoids,* a description of the *textual tradition* which carries the information out of which these factoids have been constructed.

Or:

Keeping the uncertainty of the textual tradition alive in the analysis of derived information.

… that is (intuitive version):

Copy and paste a text including a functional representation of textual variation from a digital edition directly into the "name" field of a data base, preserving textual variation within the data base field.

Technical challenges:

(1) Let a concrete database – e.g. “Spurious people of the 13th century”, http://sp10c.someuniversity.terra – provide that service.

(2) Make it easy to create such databases with a specific data base system, e.g. Neo5J.

(3) Make it easy to create such data base systems, implemented with a specific programming language.

(4) Embed the necessary abstractions deeply enough in the software hierarchy, that many programming languages – or libraries provided for them - find it easy to provide access to these abstractions.

Basic challenge: Find a generalized low level abstraction.

Why?

Tagliacozzo, August 23rd, 1268
Conradin of Hohenstaufen v. Charles of Anjou

How?

Remember:

{ Lorem ipsum dolor sit amet. } { Henricus { **episcopus | episkopus | comes }** de aliquo fecit aliquis } {, ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.}

Simplified:

{ Henricus { episcopus | episkopus | comes } fecit. }

Easily convertible to “strings”, e.g. …

String A ::= {H}{e}{n}{r}{i}{c}{u}{s}

String B ::= { }

String C ::= {e}{p}{i}{s}{c}{o}{p}{u}{s}

String D ::= {e}{p}{i}{s}{k}{o}{p}{u}{s}

String E ::= {c}{o}{m}{e}{s}

String F ::= { }

String G ::= {f}{e}{c}{i}{t}{.}

… which are connected into a “graph”.

String A ::= {H}{e}{n}{r}{i}{c}{u}{s}

String B ::= { }

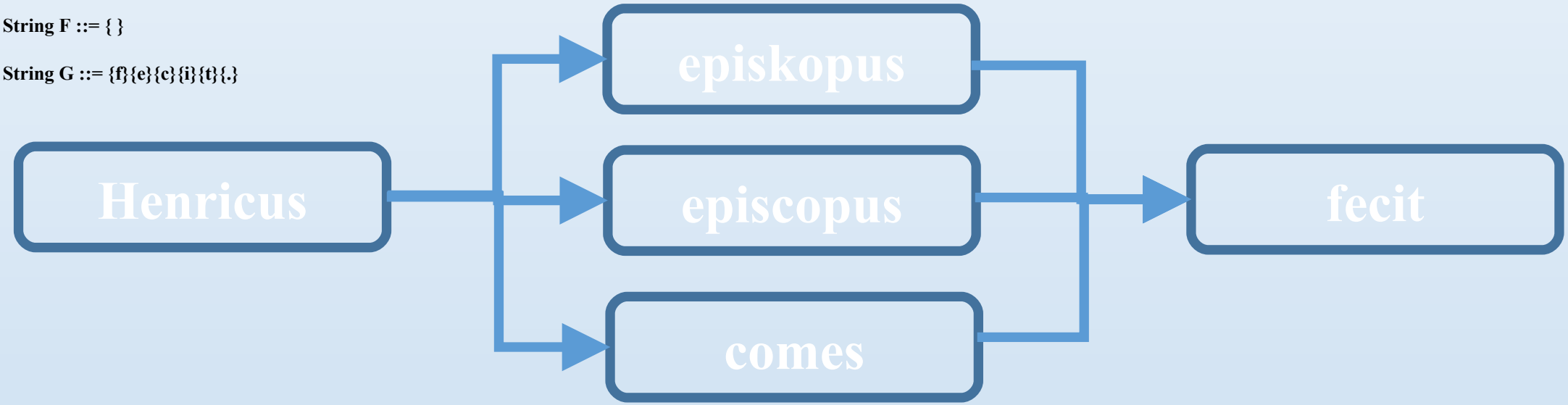
String C ::= {e}{p}{i}{s}{c}{o}{p}{u}{s}

String D ::= {e}{p}{i}{s}{k}{o}{p}{u}{s}

String E ::= {c}{o}{m}{e}{s}

String F ::= { }

String G ::= {f}{e}{c}{i}{t}{.}



Problem:

(1) Let a concrete database – e.g. “Spurious people of the 13th century”, http://sp10c.someuniversity.terra – provide that service.

(2) Make it easy to create such databases with a specific data base system, e.g. Neo5J.

(3) Make it easy to create such data base systems, implemented with a specific programming language.

(4) Embed the necessary abstractions deeply enough in the software hierarchy, that many programming languages – or libraries provided for them - find it easy to provide access to these abstractions.

Basic challenge: Find a generalized low level abstraction.

For strings such low level abstractions exist; for graphs they do not.

How about changing the notion of strings, than?

String A ::= {H}{e}{n}{r}{i}{c}{u}{s}

String B ::= { }

String C ::= {e}{p}{i}{s}{c}{o}{p}{u}{s}

String D ::= {e}{p}{i}{s}{k}{o}{p}{u}{s}

String E ::= {c}{o}{m}{e}{s}

String F ::= { }

String G ::= {f}{e}{c}{i}{t}{.}

How about changing the notion of strings, than?

String A ::= {H}{e}{n}{r}{i}{c}{u}{s}

String B ::= { }

String C ::= {e}{p}{i}{s}{c}{o}{p}{u}{s}

String D ::= {e}{p}{i}{s}{k}{o}{p}{u}{s}

➔

String’ C’ ::= {e}{p}{i}{s}{c|k}{o}{p}{u}{s}

String E ::= {c}{o}{m}{e}{s}

String F ::= { }

String G ::= {f}{e}{c}{i}{t}{.}

How about changing the notion of strings, than?

A string is almost always* implemented as an array that stores a sequence of characters in some *n*-byte oriented character encoding.

➔

{e}{p}{i}{s}{c|k}{o}{p}{u}{s} implies:

A string should be implemented as a list that stores a sequence of elements each of which is a set of alternative characters in some *n*-byte oriented character encoding.

* No personal nostalgia, but respect for LISP (year of birth: 1957).

How about changing the notion of strings, than?

String' A ::= {H}{e}{n}{r}{i}{c}{u}{s}

String' B ::= { }

String' C' ::= {e}{p}{i}{s}{c|k}{o}{p}{u}{s}

String' E ::= {c}{o}{m}{e}{s}

String' F ::= { }

String' G ::= {f}{e}{c}{i}{t}{.}

How about changing the notion of strings, than?

String' A ::= {H}{e}{n}{r}{i}{c}{u}{s}

String' B ::= { }

String' C' ::= {e}{p}{i}{s}{c|k}{o}{p}{u }{s}

String' E ::= {c}{o}{m}{e}{s}

➔

String'' C'' ::= {[b➔]{e}{p}{i}{s}{c|k}{o}{p}{u}{s}[|]{c}{o}{m}{e}{s}[➔b]}

String' F ::= { }

String' G ::= {f}{e}{c}{i}{t}{.}

How about changing the notion of strings, than?

A string should be implemented as a list that stores a sequence of elements each of which is a set of characters in some *n*-byte oriented character encoding.

➔

{[b➔]{e}{p}{i}{s}{c|k}{o}{p}{u}{s}[|]{c}{o}{m}{e}{s}[🡐b]}

implies:

A string should be implemented as a list that stores a sequence of elements each of which is a set of characters in some *n*-byte oriented character encoding, together with branches between such lists.

How about changing the notion of strings, than?

String' A ::= {H}{e}{n}{r}{i}{c}{u}{s}

String' B ::= { }

String'' C'' ::= {[b➔]{e}{p}{i}{s}{c|k}{o}{p}{u}{s}[|]{c}{o}{m}{e}{s} [➔b]}

String' F ::= { }

String' G ::= {f}{e}{c}{i}{t}{.}

➔

String''' C''' ::= {[b➔][c^]{e}{p}{i}{s}{c|k}{o}{p}{u}{s}[^c][|] [c^] {c}{o}{m}{e}{s} [c^][➔b]}

➔

What is semantic variance in the eyes of a software technologist?

Manfred Thaller, Brühl, Germany

Do not worry, not today

However …

didactic coordinates

publication coordinates

semantic coordinates

editorial coordinates

textual coordinates

symbol coordinates

base image [text]

r …

didactic coordinates

publication coordina

semantic coordinates

editorial coordinates

textual coordinates

symbol coordinates

base image [text]

Why (2)?

Observation 2

If people designing a mark up scheme have a clear conceptual model of what they are marking up, all applications operating on that markup scheme, will find it easy to implement all operations implied by that model.

If they don’t, they won’t.

Corollary

If software shall be able to answer questions arising in a specific knowledge domain, the tools used to implement it need a conceptual model of the requirements of that knowledge domain.

Summary

- A digital edition is not a static result, but a dynamic resource, which other digital systems can refer to.
- This requires, that a non-linear representation of texts is implemented in the technology stack of modern information technology at such a deep level, that it comes without cost for the implementer of an application.
- As soon as we start to see digital editions not as isolated monolithic work, but as nodes within a network of historical statements, we have to move towards standoff markup solutions.

Tagliacozzo, August 23rd, 1268
Conradin of Hohenstaufen v. Charles of Anjou

Questions

So, what *requirements* follow from *your* early battles?

What *conceptual* consequences do they have?

Which might lead to technological innovation?

Heretical opinion: Critical editions should be good for something, besides giving the editor tenure.

Ceterum Censeo

The game of the image …

(1) Clio‘s attributes are papyrus and stylus, not the violin.

(2) Well … some people in antiquity *did* show her with a Lyra; a Lyra still is not a violin, however.

(3) Somebody in fifteen century’s Einbeck *did* show her with a violin. This is wrong for a learned humanist; but it was his/her statement and inheritance.

(4) To understand the fifteenth century we have therefore to represent what is there, even if we know it to be wrong.

(5) Sources may be commented, but never “corrected”.

(6) We need systems to handle information, which is just as wrong, inconsistent, contradictory and vague, as the historical sources are.

The game of the image …

(1) Clio‘s attributes are papyrus and stylus, not the violin.

(2) Well … some people in antiquity *did* show her with a Lyra; a Lyra still is not a violin, however.

(3) Somebody in fifteen century’s Einbeck *did* show her with a violin. This is wrong for a learned humanist; but it was his/her statement and inheritance.

(4) To understand the fifteenth century we have therefore to represent what is there, even if we know it to be wrong.

(5) Sources may be commented, but never “corrected”.

(6) We need systems to handle information, which is just as wrong, inconsistent, contradictory and vague, as the Humanities’ information is.

Thank you for your attention!

manfred.thaller@uni-koeln.de