Workshop *TAG and Alexandria*

Advancing digital editing: how to work with graph models, multiple perspectives, and flexible workflows

DH Benelux 2019 workshop
September 11, 2019

R&D group - KNAW Humanities Cluster

TIME	TYPE	TOPIC
10:00 - 10:30	Talk 1	Text encoding challenges; introduction TAG
10:30 - 11:00	Talk 2	Introduction TAGML
11:00 - 11:45	Hands-on 1	Transcribing in TAGML
11:45 - 12:00	Break	The quality of the coffee
12:00 - 12:45	Plenary 1	Taking stock; reflection
12:45 - 14:30	Lunch	The quality of the food
14:30 - 15:00	Talk 3	Workflow of *Alexandria*; distributed architecture, diffing and merging views
15:00 - 15:45	Hands-on 2	Working with *Alexandria*
15:45 - 16:00	Break	The quality of the tea
16:00 - 16:45	Hands-on 3	Publication workflow; TAGML - XML - HTML
16:45 - 17:15	Plenary 2	Taking stock, reflections

Talk 1
Text encoding and the TAG model

(Elli)

Talk 1. Text encoding - some truisms

Every transcription and every encoding is an *interpretation*

We use it to

- express our understanding of text(s)
- process, analyse, and/or publish text(s)

We hope it can be used to

- be reused by others

Talk 1. Text encoding - markup functionalities

What markup (allow us to) do:

- Formal description of interpretation
- Reification of our conceptions and assumptions
- Make logical inferences about our assumptions

Talk 1. Text encoding - some challenges

"Most texts are made for a special use in a specific context for a limited group of people"

(Hillesund 2005)

"A transcription at its most basic is mediated or subject to mediation"

(Shillingsburg 2014, 167)

Talk 1. Text encoding - more challenges

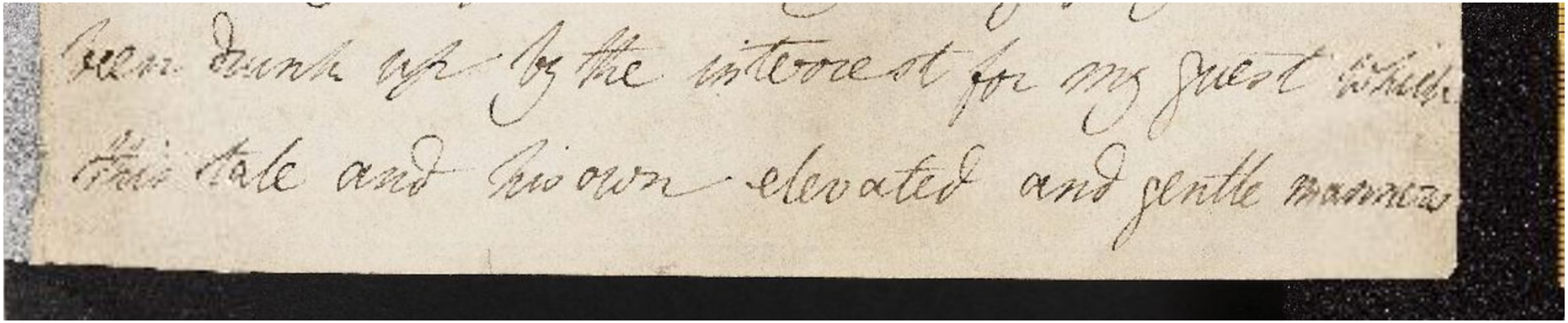
XML/TEI = the *de facto* standard

Some phenomena that do not fit naturally into the XML/TEI model

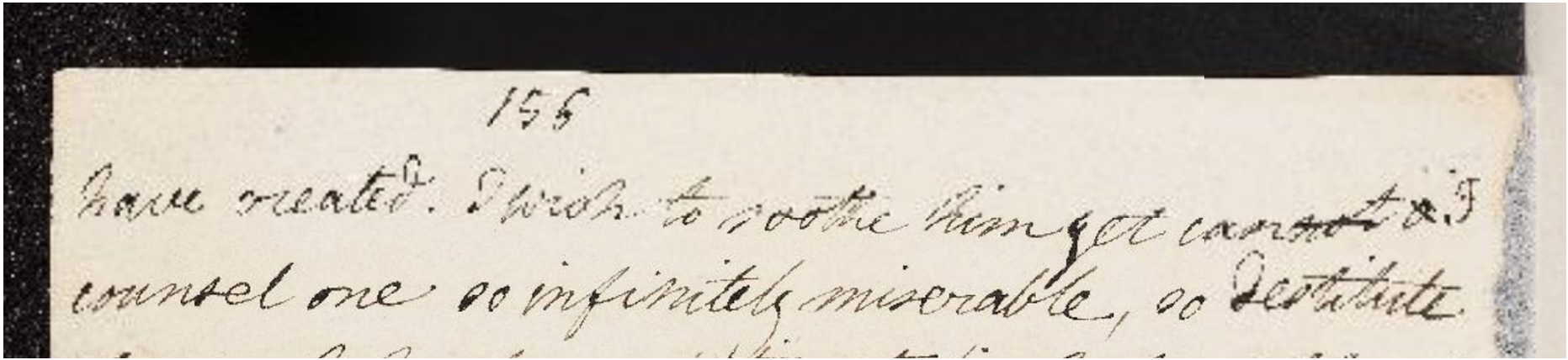
- Overlap / self-overlap
- Open variants
- Discontinuity
- Simultaneity and non-linearity

“Textual structures predate digitally representable information collections” (Vitali)

In short: literary texts are complex objects



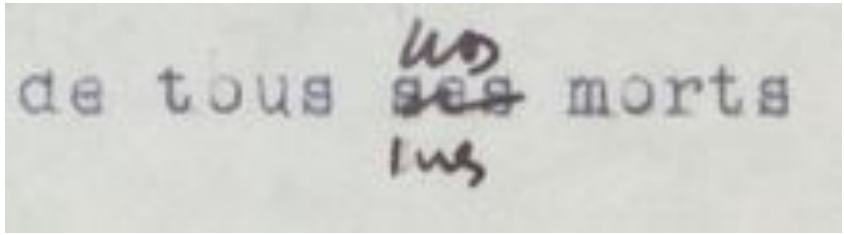
seen drunk up by the interest for my guest with
this tale and his own elevated and gentle manners



156

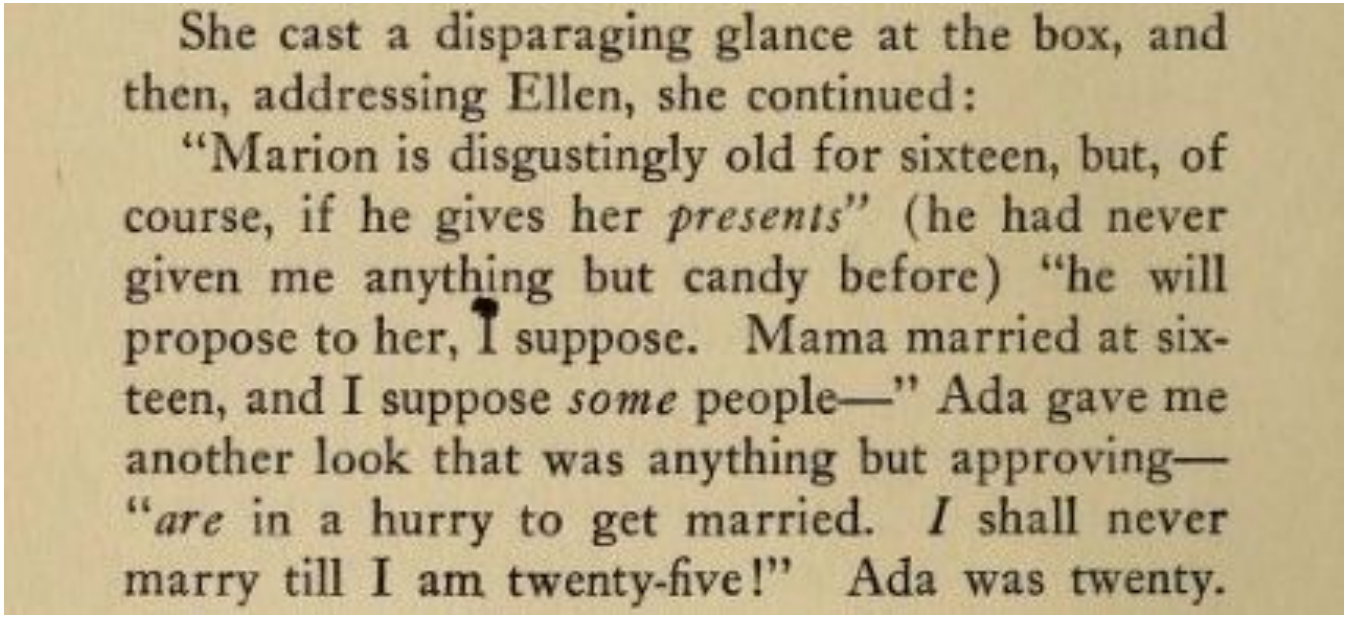
have created. I wish to soothe him yet cannot
counsel one so infinitely miserable, so destitute

Source: Shelley, M. W. "Frankenstein, MS. Abinger C. 58", in The Shelley-Godwin Archive, MS. Abinger c. 58, 22r



Source: Gustave Roud's *Requiem*. GR MS 1H/16d, f. 1r

Source: the BDMP encoding manual



She cast a disparaging glance at the box, and then, addressing Ellen, she continued:

"Marion is disgustingly old for sixteen, but, of course, if he gives her *presents*" (he had never given me anything but candy before) "he will propose to her, I suppose. Mama married at sixteen, and I suppose *some* people—" Ada gave me another look that was anything but approving—"*are* in a hurry to get married. *I* shall never marry till I am twenty-five!" Ada was twenty.

Source: Onoto Watanna, *Marion: the story of an artist's model* (1916)
https://archive.org/details/marionstoryofart00wata/page/n207

Talk 1. Text encoding - workarounds

with handovers & workarounds	Data	Text	Hierarchies	Presentation	Validation	References	Annotations	Overlapping
CSV						■		
JSON	■		■					
RDF	■				■	■	■	
Markdown		■						
HTML		■	■	■				
HTML+RDFa	■	■	■	■		■	■	
XML	■	■	■	■	■	■	■	
Overlapping fmts	■	■	■			■	■	■

Source: Vitali 2016 (https://bit.ly/2jWm96t)

Talk 1. Text encoding - workarounds

with handovers & workarounds ***& some coding***	Data	Text	Hierarchies	Presentation	Validation	References	Annotations	Overlapping
CSV								
JSON								
RDF								
Markdown								
HTML								
HTML+RDFa								
XML								
Overlapping fomats								

Source: Vitali 2016 (https://bit.ly/2jWm96t)

Talk 1. Text encoding - the primary goal?

“The primary goal of text encoding in the humanities should not be to conform to standards ... Rather, we encode texts and represent them digitally in order to present, examine, study, and reflect on the rich heritage of knowledge and expression presented to us in our cultural legacy”

Wendell Piez, 2014

Talk 1. Text encoding

The use of workarounds is ingrained in our text encoding practice. But why *wouldn't* we want to use them?

- Workarounds are not part of a standard and "in-house solutions" hinder interoperability, reuse, and analysis
- The limits and potential of a data model influence how we look at text

Talk 1. Text encoding

"TEI konzentriert sich for allen auf den Text als Werk(-Struktur), als sprachliche Äußerung und als kanonisierte, definierte oder auch variante Fassung. Der Text als intentionale Mitteilung, als semantischer Inhalt, aber auch der Text als physisches Object, als Dokument, wird nur am Rande unterstützt. Der Text als komplexes Zeichen, als semiotic Entität, spielt bei der TEI keine Rolle."

Patrick Sahle, 2013

Talk 1. Text encoding - TAG data model

In the Text-As-Graph (TAG) data model, we understand text to be

“a multilayered, nonlinear object containing information that is at times ordered, partially ordered and unordered.”

Talk 1. Text encoding - TAG’s layers

- A layer is a set of markup nodes
- Layers can overlap (= they can share textual content)
- Layers can be discontinuous
- Layers can share markup nodes
- Layers are hierarchically structured: there is one hierarchy within a layer
- Layers are not global, can be created/started anywhere

Talk1. Text encoding - orders of information

Different types of ordered data:

- fully ordered (e.g., a string of text)
- unordered (e.g., a record database of employees)
- partially ordered data (e.g., textual variation in a literary manuscript)

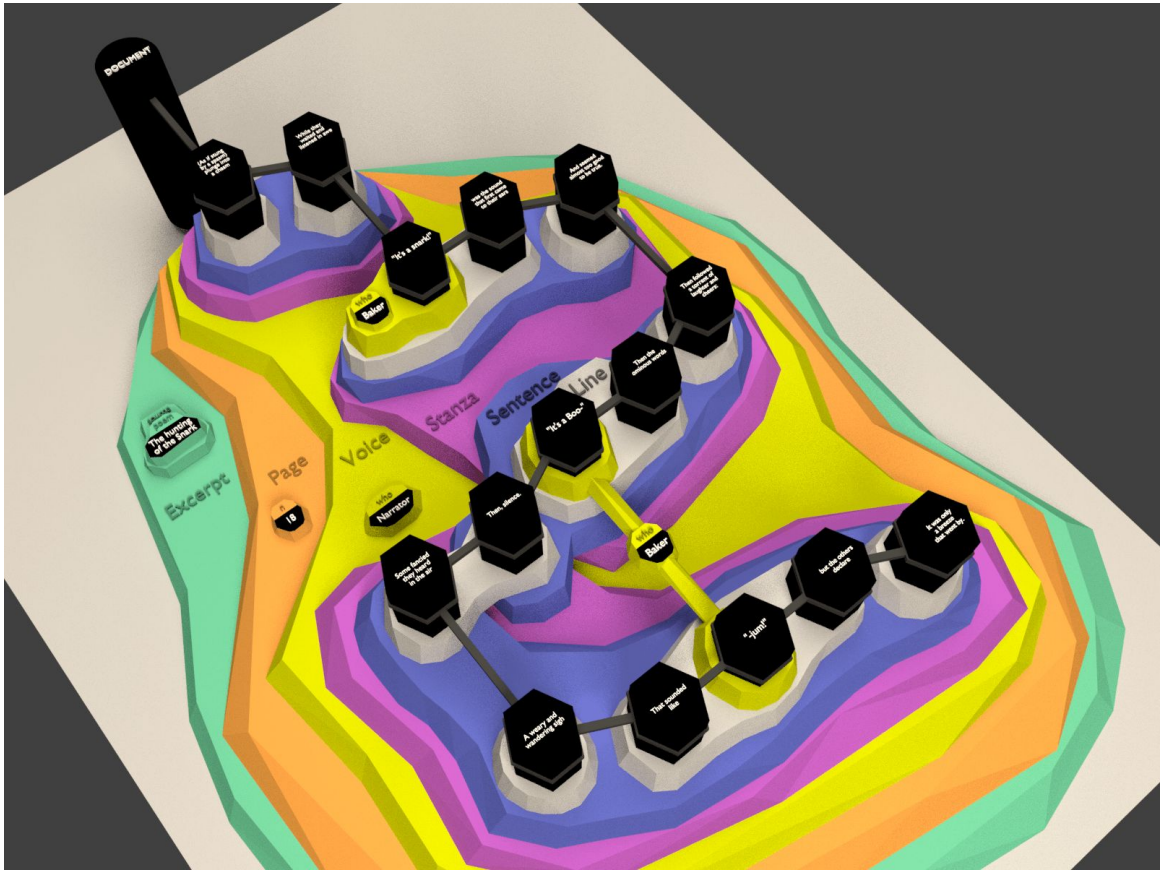
Talk 1. Text encoding - TAG data model

TAG uses a **hypergraph model for text**

An intuitive model for text encoding?

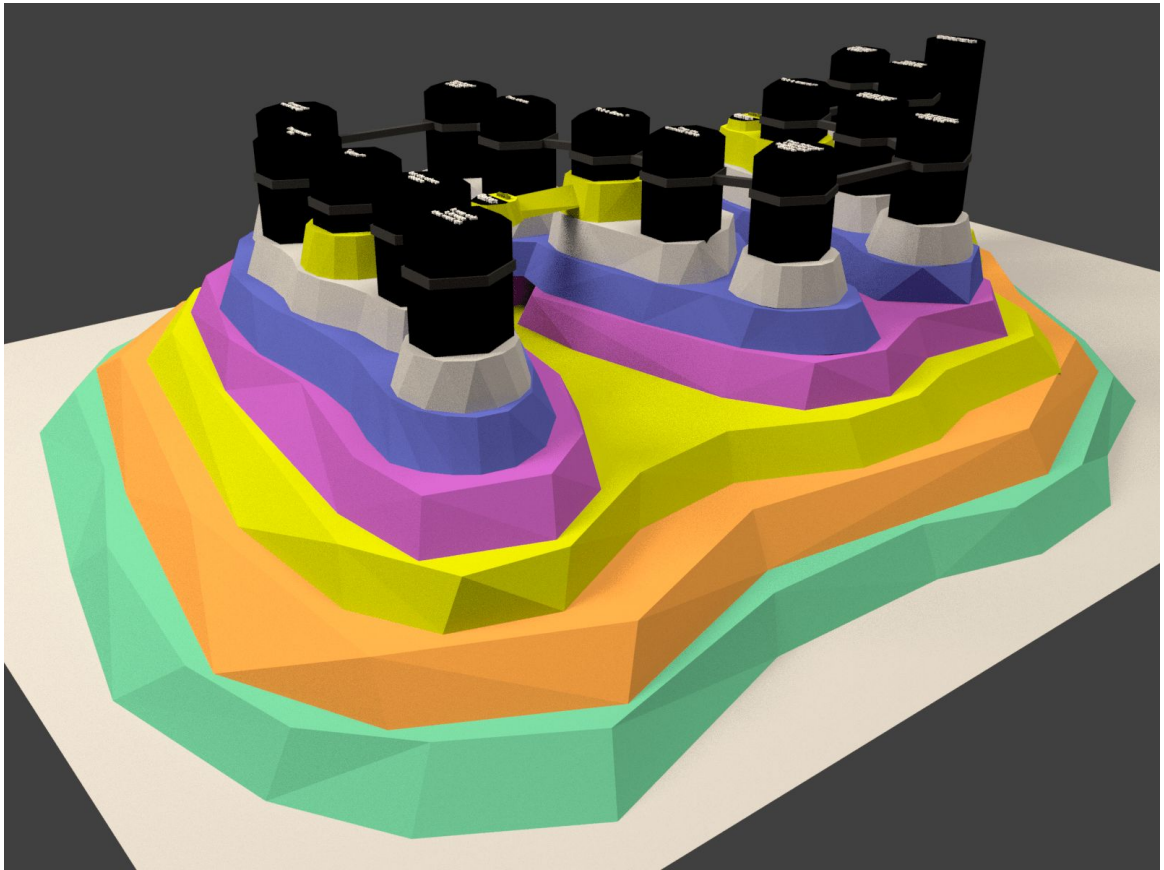
- Text
- Annotations on that text, grouped in layers
- Expressive and rich (strings, boolean, numbers, lists, nested annotations)

Talk 1. Text encoding - TAG data model



Source: Dekker *et al*. 2018 (images by Gijsjan Brouwer)

Talk 1. Text encoding - TAG data model



Source: Dekker *et al*. 2018 (images by Gijsjan Brouwer)

Talk 1. Text encoding - TAG data model

Source: Dekker *et al*. 2018 (images by Gijsjan Brouwer)

Talk 1. Text encoding - modeling in TAG

Modeling complex textual phenomena in TAG is easier, conceptually (once you get used to it)

Modeling literary texts in TAG allows you to express your understanding of text easily and in great detail

Talk 2
The TAG markup language

(Ronald)

Talk 2. What is a markup language?

A markup language is a formal language that contains text as well as annotations in the same file.

TAGML is a markup language that allows an editor to model texts as graphs.

Can be edited with any text editor.

Talk 2. What does it look like? Structure of a tagml file

```
[tagml creator='humanities cluster'>
    [p>
        Content of the first paragraph
    <p]
    [p>
        Content of second paragraph
    <p]
<tagml]
```

Talk 2. What markup looks like

An embedded annotation on text is called markup.

The text that we want to supply with extra information is preceded by an open tag and followed by a close tag.

Open tag: **`[tag>`**

Close tag: **`<tag]`**

where **tag** is flexible, but has to be the same in the open and close tag.

Format: **[tag >** … text content … **< tag]**

Talk 2. Example of annotated text

[p>This is a paragraph**<p]**

or

[page>This is a page**<page]**

Talk 2. Adding comments

```
[transcription>

[! This is what a comment looks like !]

[page>The red fox jumped over the brown dog<page]

<transcription]
```

Format: **[!** .... **!]**

Talk 2. Annotations on markup: adding a number

```
[tagml>

[page number=5>The red fox jumped over the brown dog<page]

<tagml]
```

Format: `[ tag name_of_annotation = value_of_annotation >`

Talk 2. Annotations on markup: String

```
[tagml>

The red fox jumped over the brown [note comment='the dog should have been chained'>dog<note]

<tagml]
```

Talk 2. Adding multiple annotations on markup

[transcription creator=’humanities cluster’ **date=**’September 2019’**>**

[page number=5**>**The red fox jumped over the brown dog**<page]**

<transcription]

Talk 2. Adding multiple annotations on markup: lists

```
[transcription authors=[“beatrijs”,”jan”, “piet”, “klaas”]>

The red fox jumped over the brown fox.

<transcription]
```

Format:

```
[ tag annotation_name = [ list_item1, list_item2, list_item3 ] >
```

Talk 2. Layers

Markup can be placed in layers by using layer identifiers.

Format: `[ tag | layer_identifier >`

When using a layer for the first time there needs to be a **+** sign in front of the layer identifier.

Talk 2. Example layer

```
[transcription|+L>

[p|L>

[s|L>The red fox jumped over the brown dog.<s]

<p]

<transcription]
```

Talk 2. Layers can overlap

[transcription|+L,+PA>

[page|PA n=1**>**

[p|L>

[s|L>The red fox jumped**<page][page|PA n=**2**>** over the brown dog**<s]**

<p]<page]

<transcription]

Talk 2. Advanced features: discontinuous markup

Markup can be paused, and then later on resumed.

```
[transcription>

[q>and what is the use of a book,<-q] thought Alice
[+q>without pictures or conversation?<q]

<transcription]
```

Format: `[ tag > text < - tag ] text content [ + tag > text. < tag ]`

Talk 2. Advanced features: nested annotations

```
[transcription creator={
    forename=”jan”
    surname=”klaassen”
 }>
The red fox jumped over the brown dog.
<transcription]
```

Talk 2. Advanced features: nonlinearity

Textual content does not have to be linear:

[transcription>

Text before **<|[sic>**slaugter**<sic]|[corr>**slaughter**<corr]|>**

<transcription]

<| open branch **|** next item in branch **|>** close branch

Format: **<| [tag> … <tag] | [other_tag> … <other_tag] |>**

Hands-on 1
Transcribing in TAGML

https://github.com/HuygensING/TAG/tree/master/TAGML

Hands-on 1. TAGML syntax

Basic TAGML syntax

```
[s>Cookie monster likes cookies<s]
```

Layers

```
[s|+L>Cookie monster likes cookies<s]
```

Overlap (using layers)

```
[s|+L>[a|+D>Cookie Monster likes<s] cookies.<a]
```

Discontinuity

```
[s>[q>"Hi,"<-q] she said [+q>"how are you?"<q]<s]
```

Talk 3

Layers in TAGML

Functionalities and use

Talk 3. A little more on layers

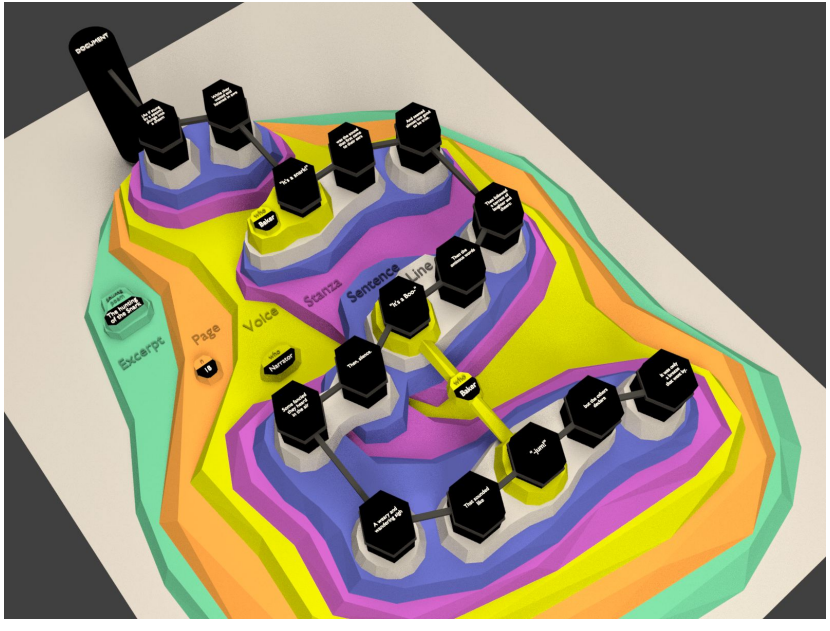
Recap:

- A layer is a set of markup nodes
- Layers can share textual content
- Layers can be discontinuous: `[q|L>` text `<-q]` text `[+q|L>` text `<q]`
- Layers can share markup nodes `[q|L,S>` text `<-q]`
- Layers are hierarchically structured: there is one hierarchy within a layer
- Layers are not global, can be created/started anywhere

Talk 3. A little more on layers

Layers are in line with how we conceptually think about text:

A multi-ordered construct with one or more layers of markup.



Talk 3. A little more on layers

Layers are *also* TAG's solution to easily express complex textual information

- Overlapping structures
- Readability of an information-rich file

Discussion 1
Reflections on encoding

(Elli)

Discussion 1.

Recap:

- TAG considers text to have zero or more layers of markup
- A layer of markup is hierarchically ordered
- A view consists of one or more layers

Q: What are the implications of using layers and views?

- Abandoning one perspective in favor of multiple, co-existing perspectives
- Documentation: communicate what markup layer constitutes what view
- Increased readability
- Facilitates conversion from TAGML to XML
- ...

Discussion 1.

Statement:

Workarounds hinder interoperability and accessibility

How can we best deal with them?

- Document your encoding practice: what workarounds do you use and why
- Adhere to standards as much as possible
- Provide a (way to) clean TEI/XML encoding for repurposing

Discussion 1.

Statement:

Models influence the way we think about text

In TAG, a text is:

- Multilayered
- At times ordered, unordered and partially ordered.

Q: How would the TAG hypergraph model for text influence how you think about your text?

Coffee/tea break

(15 min)

Next up: Working with TAGML files in *Alexandria*

Talk 4
Alexandria's workflow

(Ronald)

Talk 4. Alexandria

Alexandria is a text repository and the reference implementation of TAGML.

Alexandria works like a version control system. This means that it keeps track of the changes an editor makes to a file.

Talk 4. Alexandria workflow

Aims:

- to be able to share changes with others.
- to be editor - and platform independent.
- to be able to edit layers in isolation.

Talk 4. To reach our aims

- We need to keep track of the changes that are made to the documents.
- Have the ability to create views to select layers to edit.
- Being able to merge the changes back into the existing layers.

Talk 4. Definitions

Workspace

Location on the local file system where files are stored.

Repository

System that parses the markup files and stores the resulting graph(s).

Talk 4. Alexandria init

```
$ alexandria init
```

Prepares the workspace and repository. Has to be run only once.

Run the init command in the directory that you want to be your workspace.

It creates three subdirectories in the workspace: tagml, sparql, views and creates a repository in the background.

Talk 4. Alexandria general workflow

Create a new file using your favorite editor and store on the file system.

```
$ alexandria add <file_name>
```

```
$ alexandria commit -a
```

Tells alexandria to adds the new transcription to the repository and monitor it (track changes).

Talk 4. Alexandria general workflow

Make some changes to the file in your favorite editor

```
$ alexandria status
```

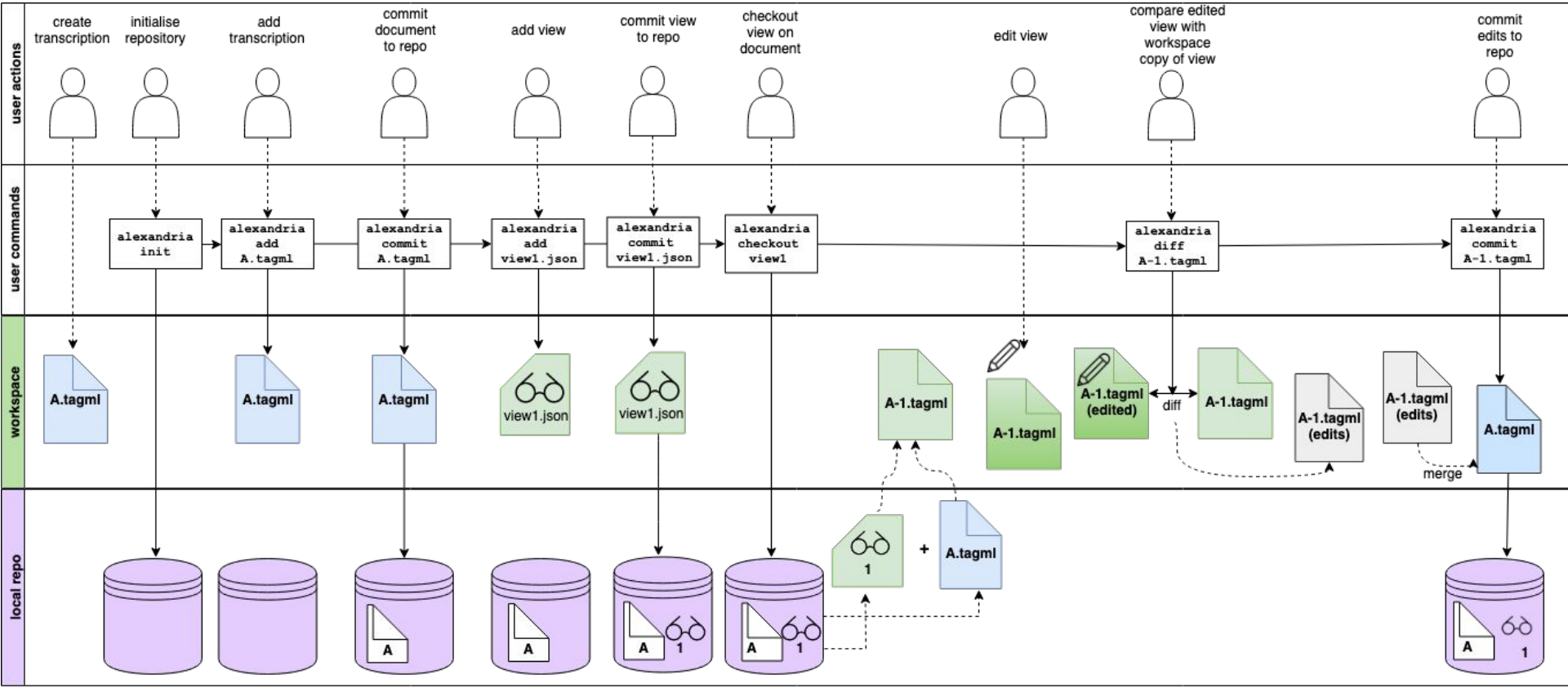
Shows the files currently added to the repository and whether they are changed.

```
$ alexandria commit -a
```

Commits the changes made to the files in the workspace of the repository

Make some more changes in your favorite editor

etc.


user actions
user commands
workspace
local repo
create transcription
initialise repository
add transcription
commit document to repo
add view
commit view to repo
checkout view on document
edit view
compare edited view with workspace copy of view
commit edits to repo
alexandria init
alexandria add A.tagml
alexandria commit A.tagml
alexandria add view1.json
alexandria commit view1.json
alexandria checkout view1
alexandria diff A-1.tagml
alexandria commit A-1.tagml
A.tagml
view1.json
A-1.tagml
A-1.tagml (edited)
diff
A-1.tagml (edits)
merge
1
+
A
1

Talk 4. Workflow with views

After creating a tagml file and adding it to the repository.

Create a view definition in your favorite editor.

For example: `{"includeLayers":["L"]}`

```
$ alexandria add <file_name_of_the_view>
```

```
$ alexandria commit -a
```

Now the view can be used to focus on a single layer.

Talk 4. Workflow with views; checkout a view

```
$ alexandria checkout <name_of_view>
```

The view is applied to all the files in the repository.

Allows the user to look at a single layer (or a selection) applied to all the files.

All the other layers are still present in the repository.

To return to the default view with all the layers:

```
$ alexandria checkout -
```

Talk 4. Future: editable views

Editing in a view

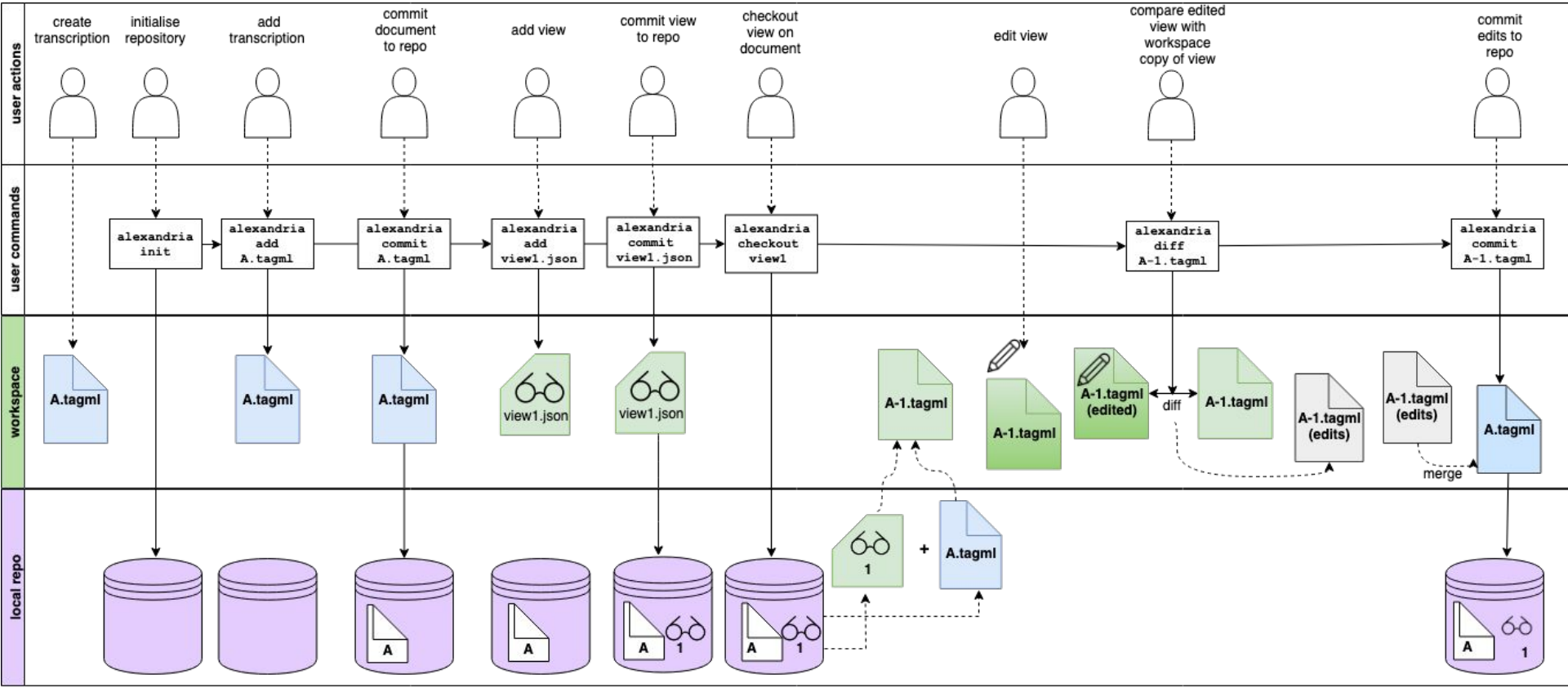
while retaining the information from the other layers.

Means that Alexandria has to diff and merge between layers.

Talk 4. Future: remote repositories

Remote repositories

- Sync your local repository with a remote repository
- And exchange layers of markup.


user actions
user commands
workspace
local repo
create transcription
initialise repository
add transcription
commit document to repo
add view
commit view to repo
checkout view on document
edit view
compare edited view with workspace copy of view
commit edits to repo
alexandria init
alexandria add A.tagml
alexandria commit A.tagml
alexandria add view1.json
alexandria commit view1.json
alexandria checkout view1
alexandria diff A-1.tagml
alexandria commit A-1.tagml
A.tagml
view1.json
A-1.tagml
A-1.tagml (edited)
diff
A-1.tagml (edits)
merge
1
+
A
1

Hands-on 2
Working with views in Alexandria

https://dhbenelux2019-alexandria.github.io/workshop/notebooks/handson2.html

Coffee/tea break

(15 min)

Next up: transformation and publication with XSLT

Hands-on 3
Transformation and publication

https://dhbenelux2019-alexandria.github.io/workshop/notebooks/handson3.html

Hands-on 3. Exporting TAGML

Why?

- tools for analysis
- visualisation
- publication frameworks

What?

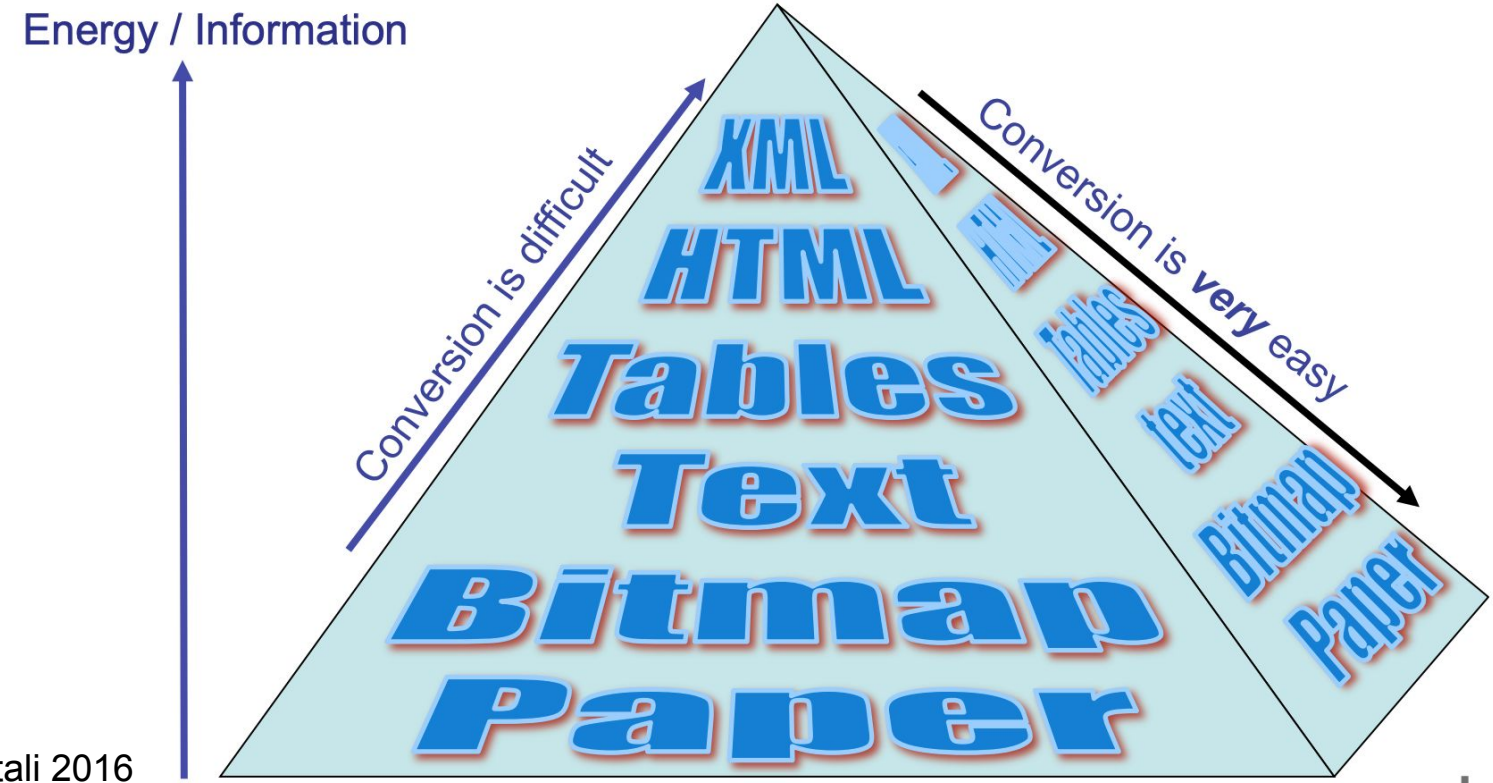
- SVG
- XML
- DOT
- PNG

Hands-on 3. Before you start...

Keep in mind:

- Envision the workflow as dynamic and fluid:
  - TAGML masterfile contains the source transcription with *all* layers of markup
  - Do you want to export that *as is*? Do you?
  - The more information, the more visually challenging the export
- Reduction of information:
  - What markup do you want to export?
  - What is the purpose of the export?

Hands-on 3. The pyramid of information



Source: Vitali 2016
https://bit.ly/2jWm96t

Hands-on 3. Exporting TAGML to XML

Export command:

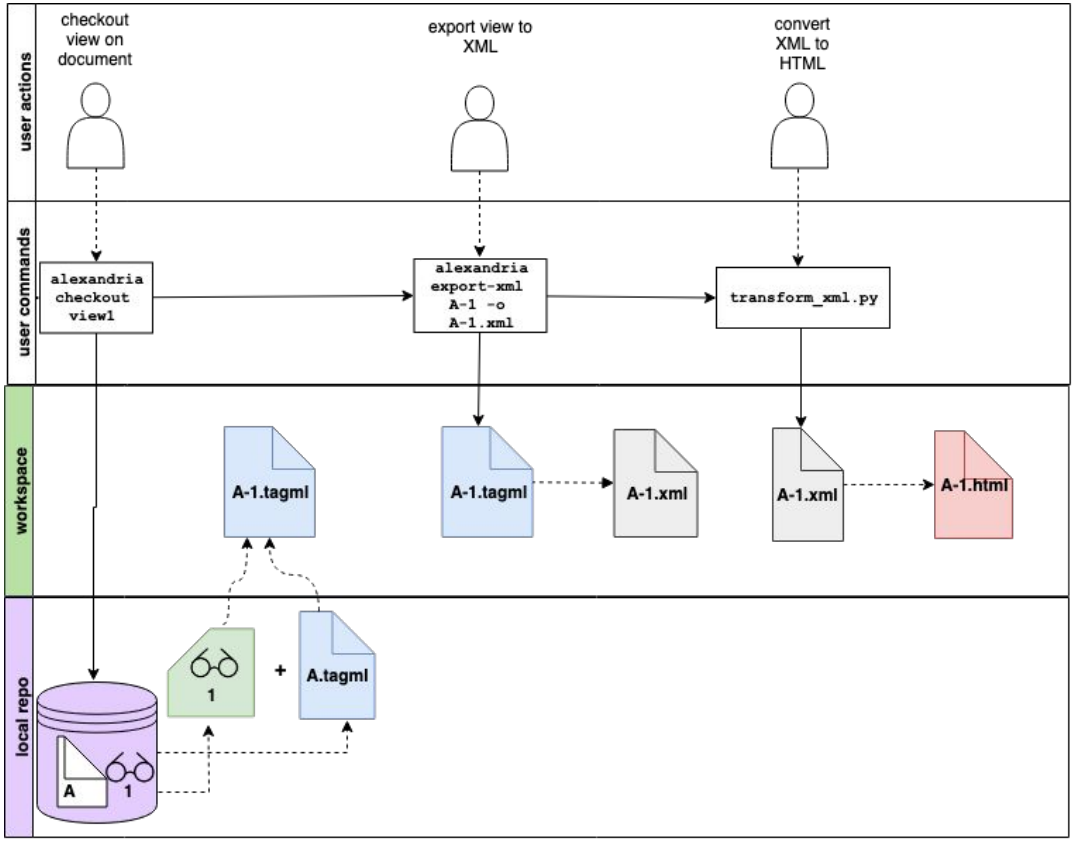
```
alexandria export-[format] [name document] -o [output-file]
```

Example:

```
alexandria export-xml woolf1 -o woolf1.xml
```

The file woolf1.xml will be in the folder where you run *Alexandria*

Hands-on 3. Transforming XML to HTML with XSLT



Hands-on 3. Exporting TAGML to XML

Implications of information reduction

- TAGML can handle (self)overlap; XML cannot
- Overlap in TAGML is handled with layers
- A TAGML file with multiple layers will be exported to XML Trojan Horse

Discussion 2
Reflections on workflow

(all; but perhaps Ronald starting with distributed workflow)

Discussion 2.

The combination of TAG and *Alexandria* provides us with a powerful modelling tool:

- multiple coexisting views

- inclusive data model

- modular, open source software

How would these features influence how we model, think about, process, and analyse text?

Discussion 2.

Recap

- With *Alexandria* you can add one or more layers of markup on a source text
- You can use *Alexandria* with the editor and on the platform of your choice
- You can implement *Alexandria* into an existing XML-based workflow with the export functionality
- *Alexandria* takes care of version control, which means
- *Alexandria* compares TAGML files on the level of the text as well as the markup

Discussion 2.

"… both the analysis and the representation of literary texts are enabled in new ways by being able to overlap freely in the model…"

Wendell Piez on *LMNL* (2014)

Discussion 2.

Both the analysis and representation of, and the collaboration on literary texts are enabled in new ways by being able to natively express complex textual features in the model.

Discussion 2.

Question 1:

How should we handle the merge of TAG files? Do we consider changes in markup as additions or replacements?

`[line>` becomes `[l>`

- `[line>` is replaced by `[l>`
- `[l>` is added to the `[line>` markup

Discussion 2.

Question 2

Is the source text part of a perspective or not? In other words, is a perspective only the markup or also the source text?

View material:

<|[sic>slaugter**<sic]|[corr>**slaughter**<corr]|>**

View poetic:

[rhyme>slaughter**<rhyme]**