THE ANDREW W. MELLON FOUNDATION

TEI Simple

Power, economy, and a processing model for encoders and developers

NORTHWESTERN UNIVERSITY

TEI

DiXiT Digital Scholarly Editions Initial Training Network

UNIVERSITY OF OXFORD

University of Nebraska Lincoln

TEI Simpletons

- **Project Investigators:**
  - Martin Mueller, Northwestern University
  - Brian Pytlik Zillig, University of Nebraska-Lincoln
  - Sebastian Rahtz, University of Oxford
- **Additional Project Staff:**
  - Magdalena Turska, DiXiT Project / University of Oxford
  - James Cummings, DiXiT Project (Oxford PI) / University of Oxford
- **Advisory Committee:**
  - Pip Willcox, Bodleian Library, Oxford
  - Suzanne Haaf, Deutsches Textarchiv, Berlin
  - Matthias Goebel, University of Gottingen
  - James Cummings, University of Oxford
- And other members of the TEI-Simple mailing list and TEI Community

About TEI Simple

- Joint project between TEI Consortium, Mellon Foundation, Northwestern University, University of Nebraska-Lincoln, and the University of Oxford
- TEI Simple aims to 'define a new highly-constrained and prescriptive subset of the Text Encoding Initiative (TEI) Guidelines suited to the representation of early modern and modern books, and a formally-defined set of processing rules which permit modern web applications to easily present and analyze the encoded texts'
- All outputs are open source, all working is in the open on github https://github.com/TEIC/TEI-Simple/
- All discussion is on a open mailing list: https://web.maillist.ox.ac.uk/ox/info/teisimple

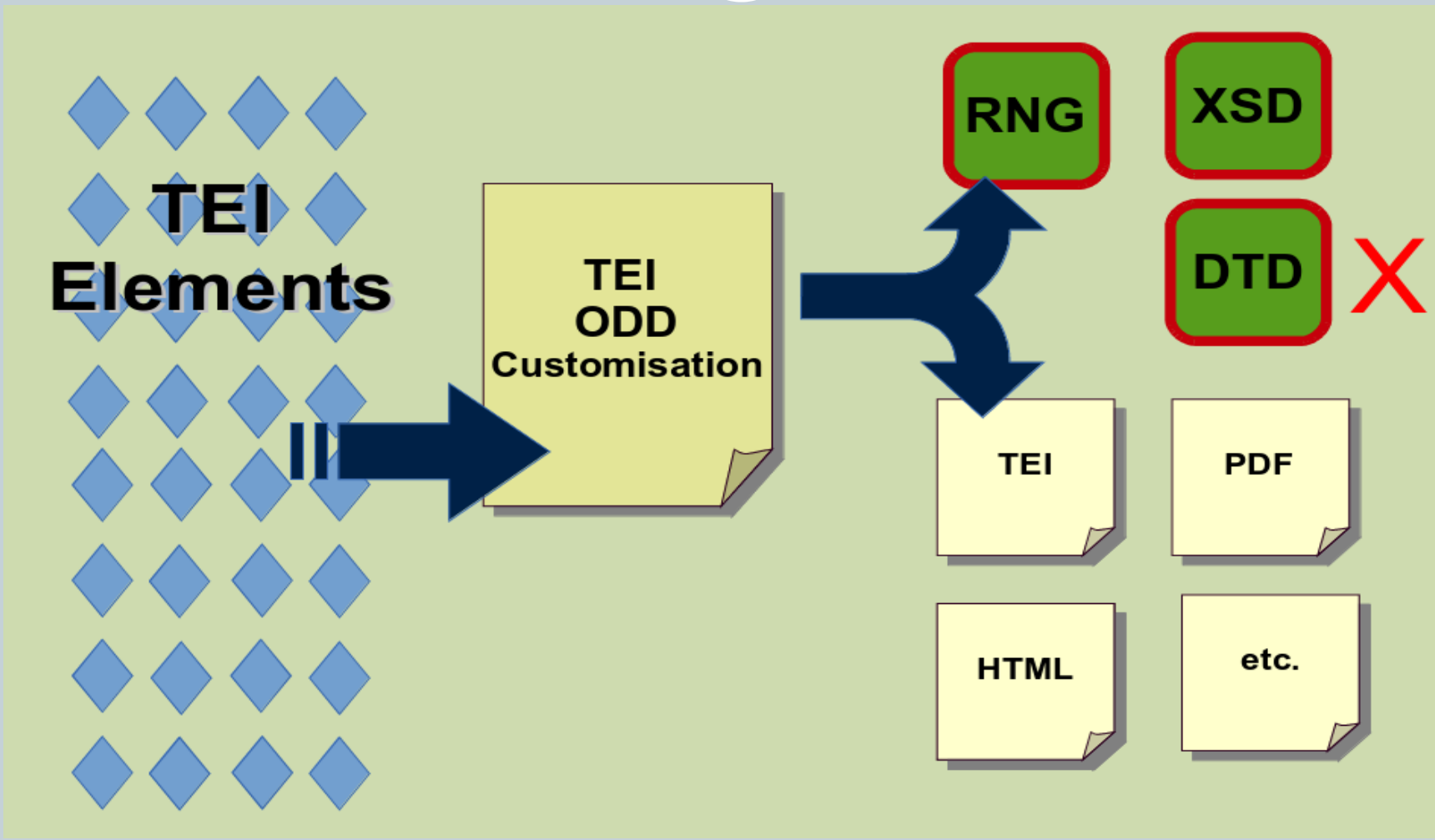
TEI Simple: High-Level Goals

- Definition of a new highly constrained and prescriptive subset of the TEI Guidelines suited to the representation of early modern and modern books. The degree of detail supported will be sufficient to encompass, at a minimum, the current practices of the TCP's EEBO, ECCO, and Evans collections plus those of other major European initiatives such as Text Grid or the German Text Archive (DTA) and the Consortium Cahiers in France.
- Developing and implementing processing model documentation for TEI Simple.
- Formal mapping of the elements used by TEI Simple to the Conceptual Reference Model of the International Council of Museums (CIDOC CRM).
- Definition and implementation of machine-readable descriptions of the encoding status and richness of TEI texts, providing 'TEI Performance Indicators', which help to document expectations for the encoded text.
- Full integration of TEI Simple into the TEI Guidelines and infrastructure with ongoing maintenance by the TEI Technical Council.

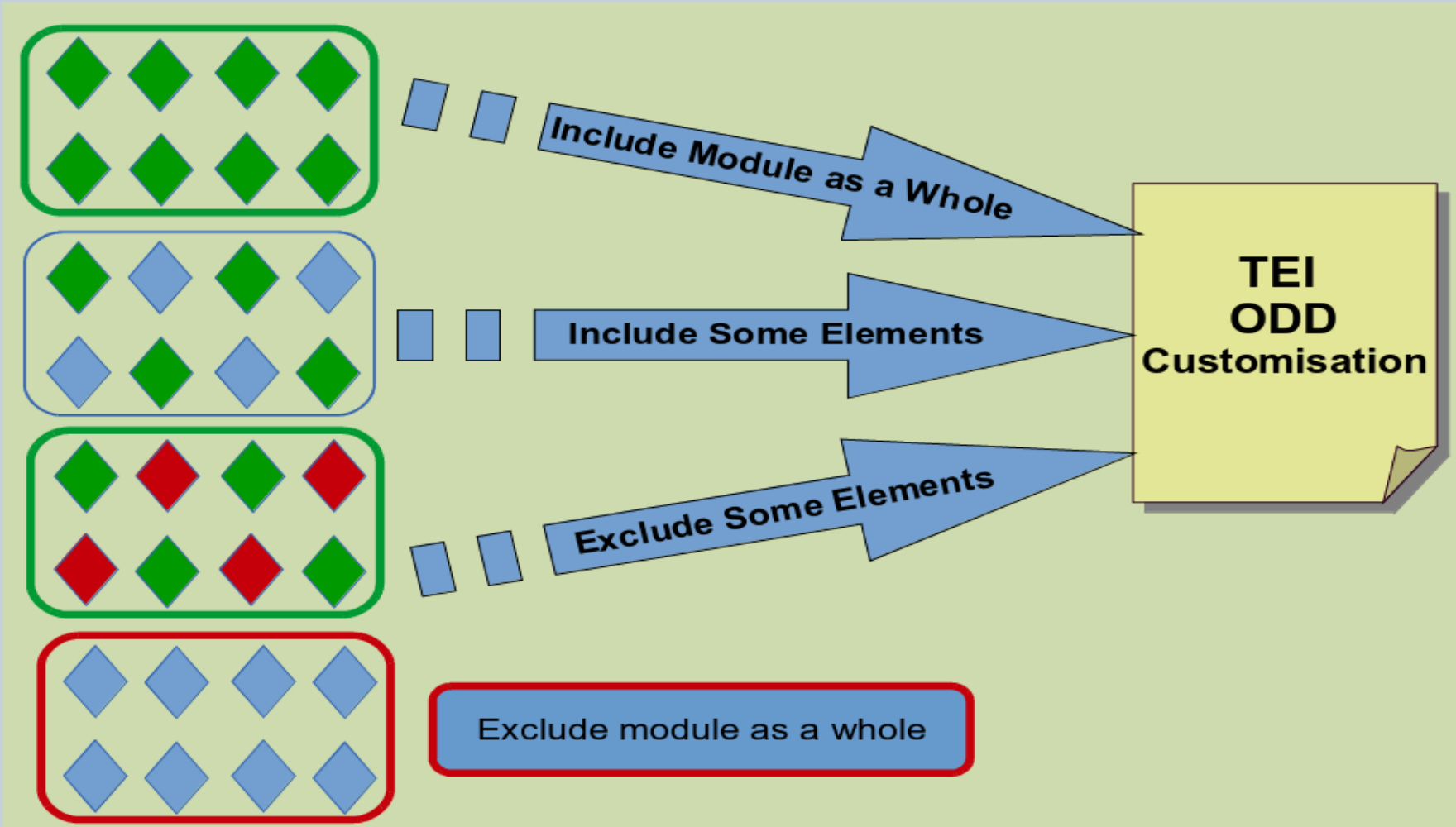
TEI vs TEI Simple

TEI	TEI Simple
• concentrates on data modelling aspects and ideological agnosticism • prefers freedom of projects to customize over rigid standardization • avoids any recommendation for output processing & publishing • TEI stylesheets often too complicated to customize • Set of existing customizations which allow maximum flexibility • usually only concerned with TEI vocabularies	• concentrates on eliminating the ambiguity for the encoder • aims at increased interoperability • provides out-of-the-box mechanism for default processing • allows for [relatively] easy customization and extension • Constrained customization with lack of flexiblity • TEI Simple processing model can be applied to other vocabularies

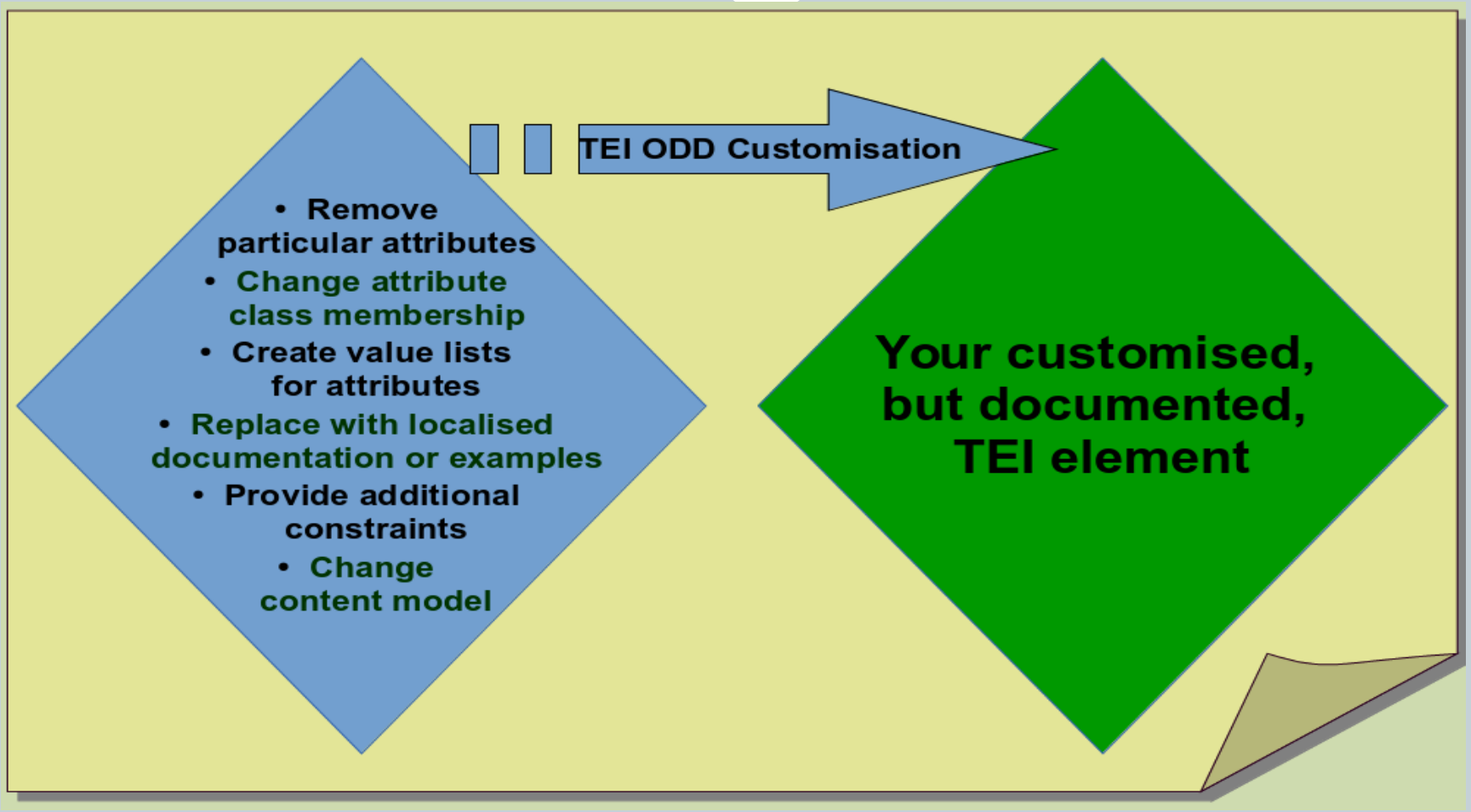
TEI Customization: A Quick Reminder



TEI Customization: A Quick Reminder



TEI Customization: A Quick Reminder



All conformant use of the TEI relies on a TEI ODD Customisation.

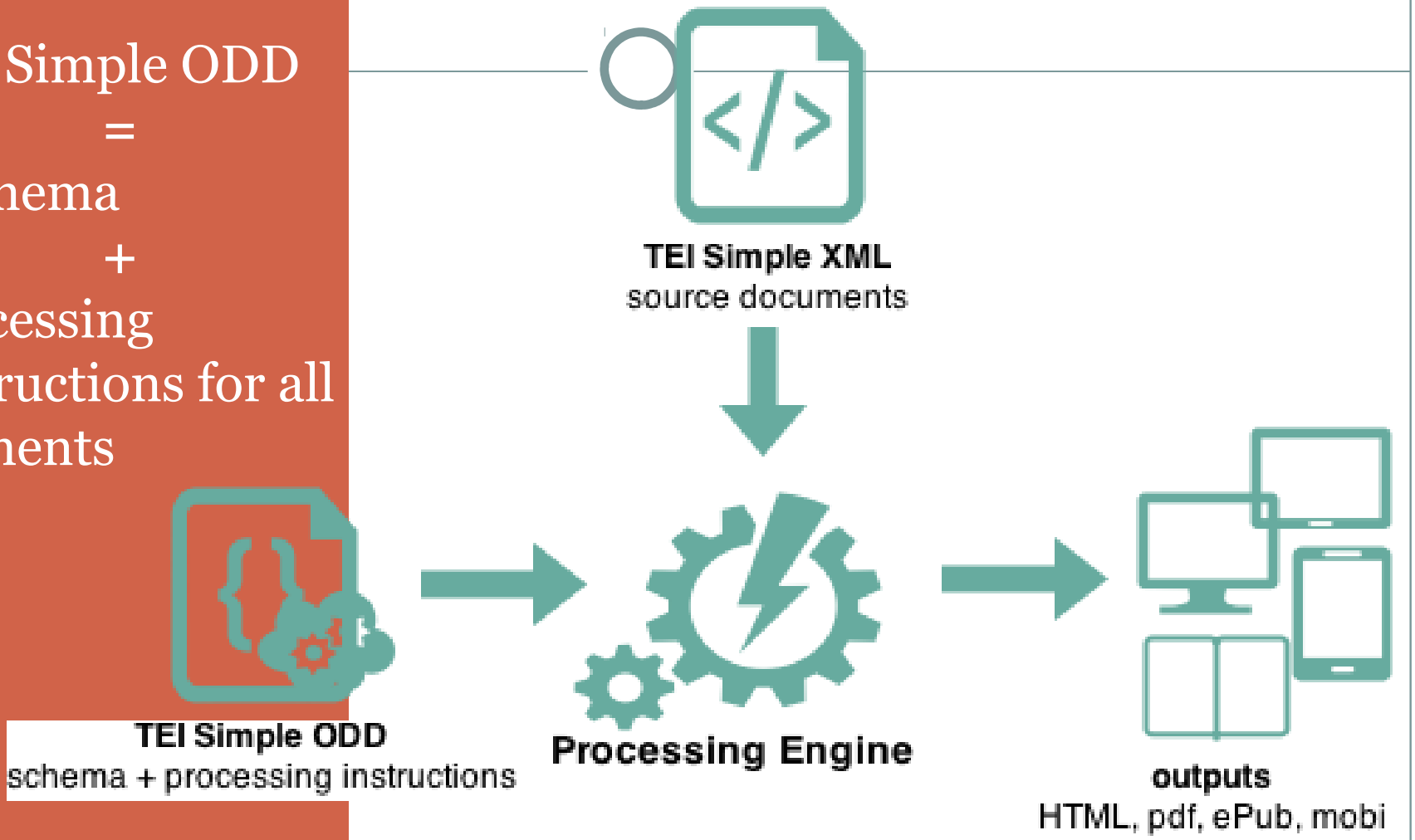
TEI Simple Subset

- This subset of TEI Simple attempts to remove ambiguity for both encoders and developers.
- 'Simple' does not necessarily mean 'small', nor does it mean 'simplistic'
- Selection of elements on TEI texts from a set of large archives or text collections:
  - Text Creation Partnership: Evans, ECCO, EEBO (including the unreleased phase 2).
  - Oxford Text Archive: All TEI P5 files.
  - Deutsches Textarchiv.
  - Documenting the American South.
  - CESR.
  - OBVIL: corpus critique.
- Concentrates on constraining the contents of the <text> element, not metadata or facsimiles.
- Subset of elements reduced the 546 (as of TEI P5 version 2.7.0) elements could be limited to 104 (at time of writing) not including <teiHeader> elements.

TEI Simple Constraints

- Closed attribute value lists, where TEI has open ones, e.g. name/@type
- Schematron constraints, e.g. for checking local URI fragment references
- Uses Documented Private URI Syntax for standardized @rendition values, e.g.:
  - simple:allcaps, simple:blackletter, simple:bold, simple:bottombraced, simple:boxed, simple:centre, simple:cursive, simple:display, simple:doublestrikethrough, simple:doubleunderline, simple:dropcap, simple:float, simple:hyphen, simple:inline, simple:justify, simple:italic, simple:larger, simple:left, simple:leftbraced, simple:letterspace, simple:literal, simple:normalstyle, simple:normalweight , simple:right, simple:rightbraced, simple:rotateleft, simple:rotateright, simple:smallcaps, simple:smaller, simple:strikethrough, simple:subscript, simple:superscript, simple:topbraced, simple:typewriter, simple:underline, simple:wavyunderline
- And of course many elements and attribute (classes) removed

TEI Simple ODD
=
a schema
+
processing instructions for all elements

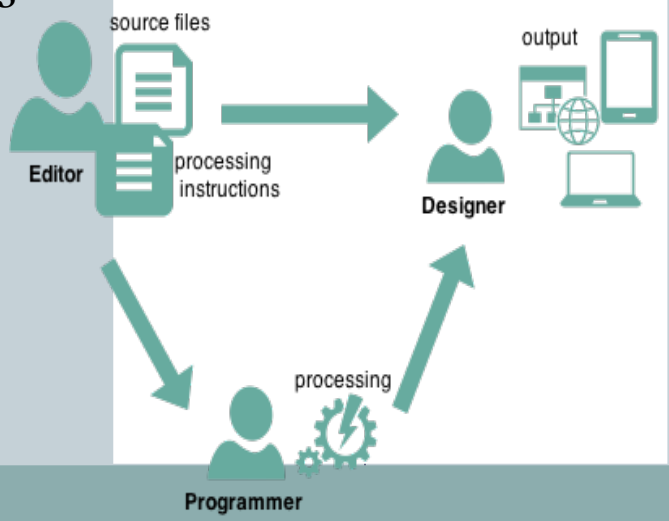


TEI Simple processing model

Rahtz Rationale

workflow with at least three distinct roles (not necessarily people)

Editor	Programmer	Designer
manages the text integrity, makes the high-level output decisions: ○ **structural** descriptions '*should the original or corrected version be displayed by default*', or '*is this a block level or inline component*' ○ indications of **appearance** '*titles should be in italics*'	takes the editor's specification, and the TEI text(s), and creates the input for the designer to make the final output	creates the output envelope (for example book layout using InDesign, or a web site using Drupal), making decisions about the appearance in conjunction with the editor *'use Garamond font throughout'* *'every page must show the departmental logo'*



simple scenarios

default	customized
• editorial decisions recommended by teiSimple fit project's needs perfectly: just use teisimple.odd	• project requires customization: import and override teisimple.odd with custom processing and rendition instructions

Editor’s (Or TEI Assistant's) Simple Tasks

- identify elements that require individual treatment
- if treatment differs depending on context, identify all possible situations via XPath expressions
  (eg. <head> elements inside should be treated differently than all other <head> elements)
- decide which behaviour is required under given circumstances, specify parameters
  (eg. If it exists use only <lem> child as 'visible by default' in critical apparatus <app> entries, provide other <rdg> as alternates)
- if treatment differs depending on output type create additional models with *@output* as necessary
- specify rendition using simple:* URI syntax

Editor’s (Or TEI Assistant's) Simple Skills

- Familiarity with source files and XML encoding used
- Ability to identify different use scenarios for their output
- Extremely basic XPath fluency to specify model parameters (or an assistant)
- Basic CSS knowledge for specifying output rendition (or an assitant)

In many cases the editor will work with a TEI/Xpath/CSS fluent assistant, but in other cases the editor already knows these.

Turska Tenet

maximum expressivity to the editor

The ODD should store as much information as possible

For each element there are potentially numerous <model> instructions that specify intended processing and rendering for different outputs and in various contexts. This gives the editor, or their TEI assistant, maximum expressivity and control over output without implementation details

```
<elementSpec mode="change" ident="date">
    <model output="print" predicate="text()" behaviour="inline"/>
    <model output="print" predicate="@when and not(text())"
        behaviour="inline">
        <param name="content">@when</param>
    </model>
    <model predicate="@when" output="web" behaviour="alternate">
        <param name="default">.</param>
        <param name="alternate">@when</param>
    </model>
</elementSpec>
```

Simple Processing Model for <choice>

```
<elementSpec mode="change" ident="choice">
    <model predicate="sic and corr" behaviour="alternate">
        <param name="default">corr</param>
        <param name="alternate">sic</param>
    </model>
    <model predicate="abbr and expan" behaviour="alternate">
        <param name="default">expan</param>
        <param name="alternate">abbr</param>
    </model>
    <model predicate="orig and reg" behaviour="alternate">
        <param name="default">reg</param>
        <param name="alternate">orig</param>
    </model>
</elementSpec>
```

<model> Processing Model Documentation

- *@output* specifies output in which model applies
- *@predicate* specifies context in which model applies
- *@behaviour* specifies behaviour from *teiSimple behaviour library* to apply
- *@useSourceRendition* indication to preserve *@rendition* value from the source
- *<param>* elements specify parameters for *behaviour*
- *<rendition>* elements specify CSS instructions to indicate outline appearance

TEI Simple *behaviours* library

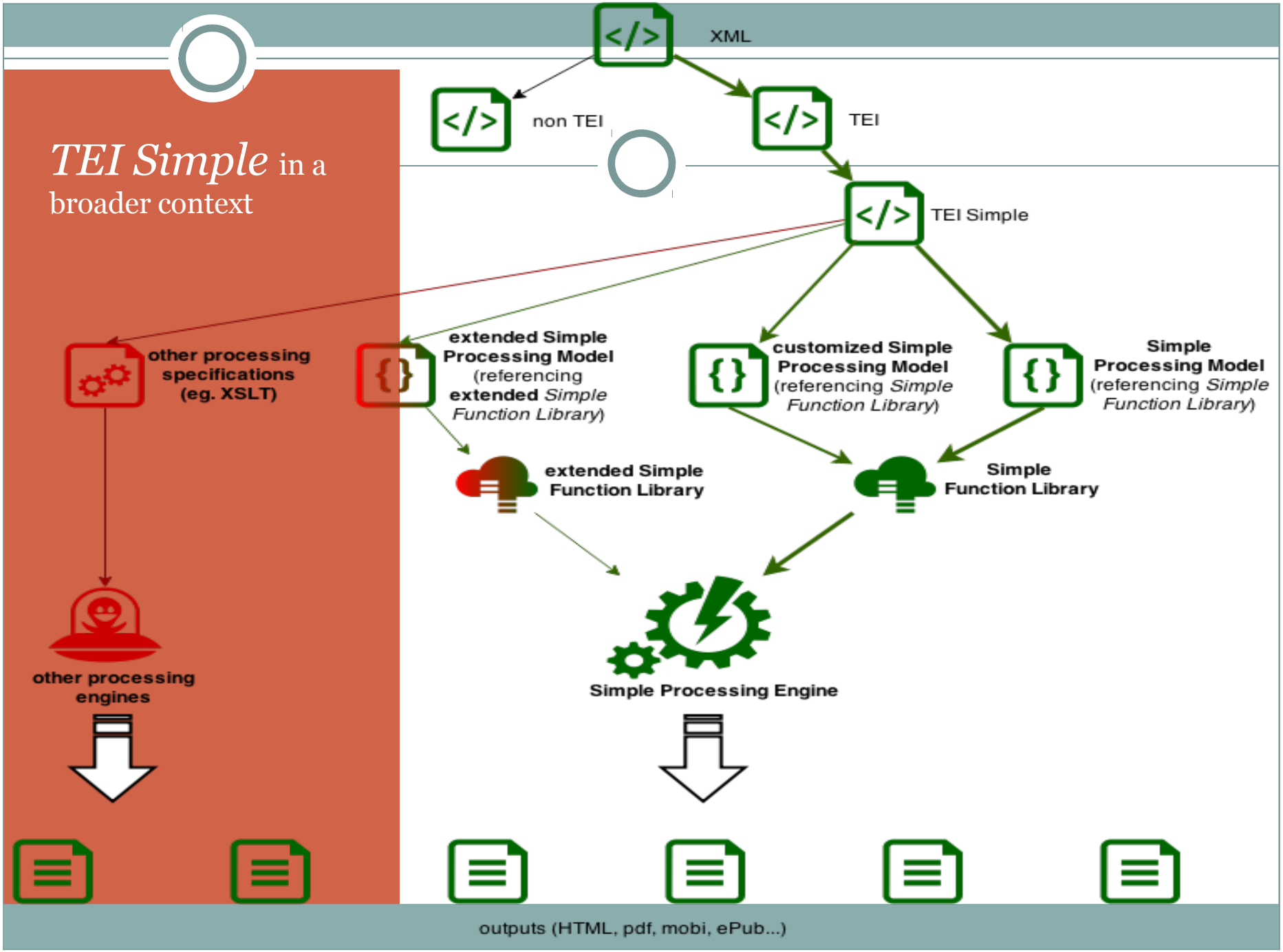
handful of behaviours that achieve > 80% tasks

behaviours based on commonly used terms

if no content parameter specified, all behaviours use current element as default content

alternate (default ,alternate)	create a specialized display of alternating elements for displaying the preferred version, both at once or toggling between the two.
anchor (id)	create anchor with ID
block (content)	create a block out of the content parameter.
body (content)	create the body of a document.
break (type,label)	make a line, column, or page break according to type
cell (content)	create a table cell
cit (content,source)	show the content, with an indication of the source
document (content)	start a new output document
figure (title)	make a figure with the title as caption
glyph (content)	show a character by looking up reference
graphic (url)	if url is present, uses it to display graphic, else display a placeholder image.
heading (content)	creates a heading.
index (type)	generate list according to type
inline (content,label)	creates inline element out of content if there's something in rendition, use that formatting otherwise just show text of selected content.
link (content,target)	create hyperlink
list (content)	create a list by following content
listItem (content)	create list item
metadata (content)	create metadata section
note (content,place, marker)	create a note, according to value of place; could be margin, footnote, endnote, inline
omit	do nothing, do not process children
paragraph (content)	create a paragraph out of content.
row (content)	make table row
section (content)	create a new section of the output document
table (content)	make table
text (content)	literal text
title (content)	make document title

TEI Simple in a broader context



Implementations of Simple Processing Model documentation

As a test of development methodologies based on the processing model documentation we had developers implement it in different languages:

- **XSLT:** implemented by Sebastian Rahtz and Magdalena Turska (University of Oxford), with significant TEI and TEI-Simple knowledge
- **Java:** implemented by Matthew Buckett (University of Oxford), with little or no TEI knowledge
- **XQuery and eXist-db**: implemented by Wolfgang Meier (eXist Solutions) with only minor TEI knowledge
- See for example: http://showcases.exist-db.org/exist/apps/tei-simple/
- **Does it work?**

Matthew had no problems, Wolfgang says "*This saved a few thousand lines of code compared to the original app I used as a template*"

What Now?

- **Documentation:** While there is some (developer level) documentation, what TEI Simple needs is detailed front-facing worked examples and explanatory documentation
- **Integration:** The TEI Technical Council has suggested some minor modifications, and these need to be incorporated before TEI Simple can become a supported customization of the TEI, and the Simple Processing Model can be integrated into the TEI Guidelines as a whole.
- **Expansion:** TEI Simple has a relatively tight remit for works it covers. We'd be interested in people using it as base to expand for their projects (not just works by dead, white, guys!). They'll only have to extend for those bits we don't cope with.
- **Github:** https://github.com/TEIC/TEI-Simple/
- **Mailing List:** https://web.maillist.ox.ac.uk/ox/info/teisimple