

# Referat 09: TEI anpassen mit Roma

**DARIAH-DE Tutorial *Digitale Textkodierung mit TEI***

Redaktion: Christof Schöch (Univ. Würzburg)

Version 1.0, 02/2014

Grundlage der Folien: DH@Oxford 2012

Lizenz: [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) (BY-SA)



# 1. Warum und wie ein Schema definieren?

# Wann, warum, wie TEI anpassen?

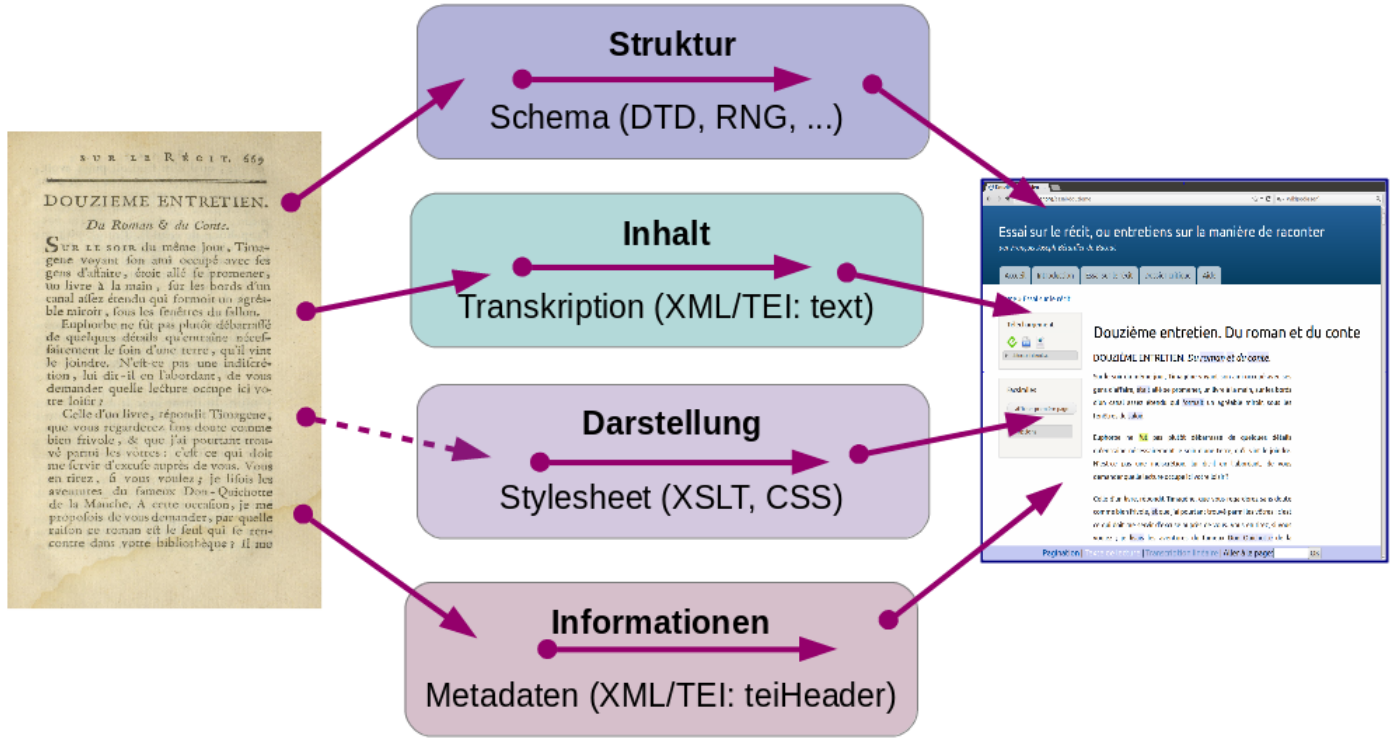
Zunächst: man passt TEI immer irgendwie an

- TEI ist umfangreich, flexibel, modular
- Konsistenz der Kodierung erhöhen
- Dokument-Modell formal ausdrücken und dokumentieren

Daher behandeln wir hier:

- Wie ist TEI aufgebaut?
- Wie definiert man ein TEI Schema?
- Wie generiert man die eigene Dokumentation?

# Zu Erinnerung: Prozessmodell



# Etwas Terminologie

## Allgemein

- TEI besteht aus Modulen; siehe Input „Guidelines 1-23“
- Jedes Modul enthält „element specifications“

## Jede Element-Spezifikation enthält:

- einen Element-Namen („canonical name“ / „generic identifier“; <gi>);
- eine kanonische Beschreibung
- Angabe zur Klassenzugehörigkeit
- Angabe zum Kontext
- Definition der Attribute
- Definition des „content model“
- Beispiele und Anmerkungen
- <title>

# Schema-Spezifikation

Ein eigenes Schema definieren, heißt:

- Bestimme Module auswählen
- In Modulen bestimmte Elemente auswählen
- Für Elemente deren Attribute bestimmen

## Schema-Spezifikation

- Ein TEI-Dokument, das eine `<schemaSpec>` enthält, wird ODD genannt (One Document Does it all)
- Hieraus können Schema-Dateien (RelaxNG, W3C Schema, etc.) und eine Dokumentation generiert werden

# Wie kann man eine Auswahl treffen?

## Möglichkeiten

- Alle Module und Elemente verwenden (keine gute Idee)
- Vordefinierte Teilmengen verwenden (TEI Lite, etc.)
- Selbst Module und Elemente auswählen

## Konkrete Umsetzung

- Entweder selbst ein Schema schreiben (Experten)
- „Roma“ verwenden: Web-Frontend für Schema-Generator:  
<http://www.tei-c.org/Roma/>

# Roma (1): Basis auswählen

TEI Roma: generating customizations for the TEI

TEI Roma is a tool for working with TEI customizations. A TEI customization is a document from which you can generate a schema defining which elements and attributes from the TEI system you want to use, along with customized HTML or PDF documentation of it. The schema generated can be expressed in any of DTD, RELAXNG W3C Schema or Schematron languages.

You can make or modify your TEI customization in several different ways:

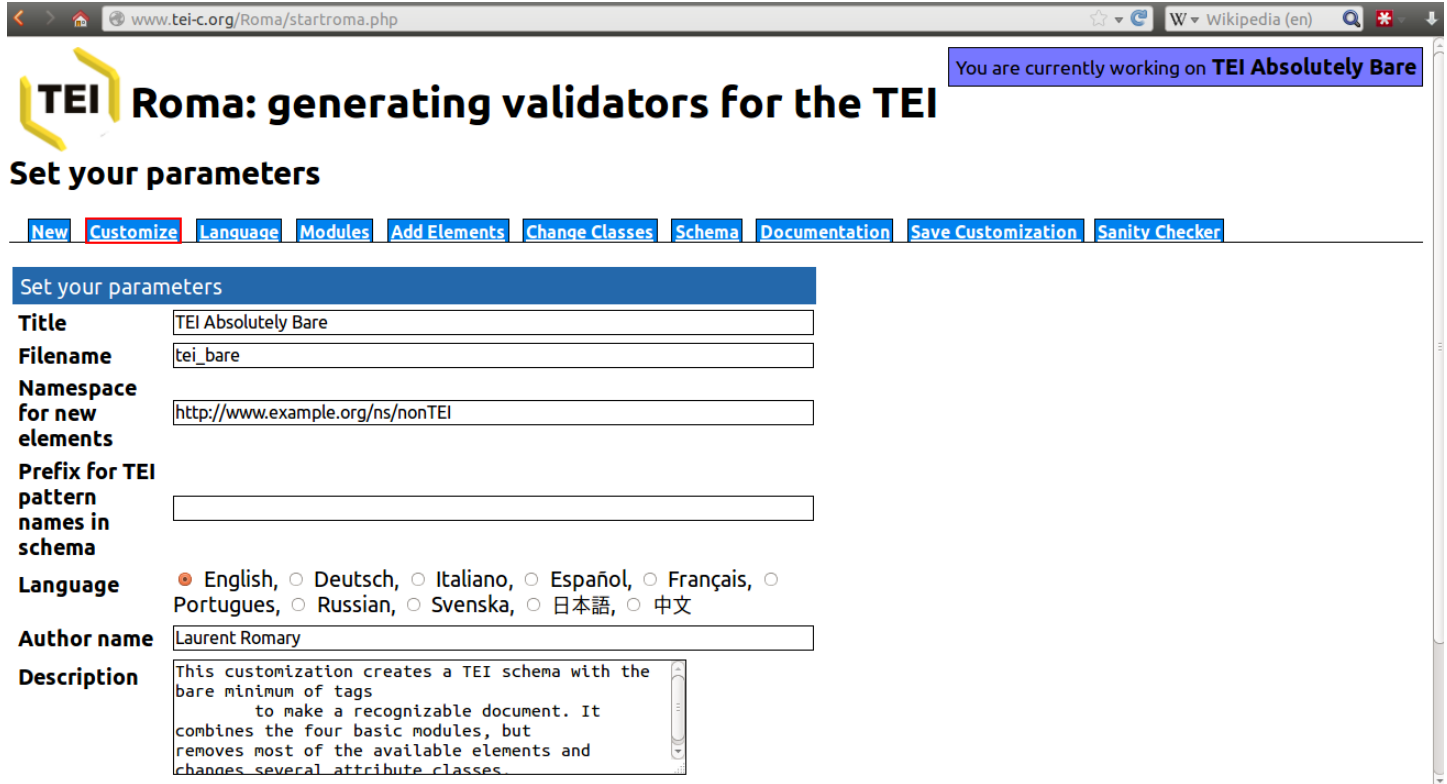
- Build up: create a new customization by adding elements and modules to the smallest recommended schema
- Reduce: create a new customization by removing elements and modules from the largest possible schema
- Create a new customization starting from a template
- Use or modify an existing TEI-defined customization
- Upload a customization  No file selected.

Community-maintained customizations can be downloaded from [the TEI website](#)

**Start**



# Roma (2): Metadaten eingeben



The screenshot shows a web browser window with the URL `www.tei-c.org/Roma/startroma.php`. The page title is "Roma: generating validators for the TEI". A blue notification bar at the top right states "You are currently working on TEI Absolutely Bare". Below the title is a navigation menu with buttons for "New", "Customize", "Language", "Modules", "Add Elements", "Change Classes", "Schema", "Documentation", "Save Customization", and "Sanity Checker". The "Customize" button is highlighted in red. The main content area is titled "Set your parameters" and contains a form with the following fields:

- Title**:
- Filename**:
- Namespace for new elements**:
- Prefix for TEI pattern names in schema**:
- Language**:  English,  Deutsch,  Italiano,  Español,  Français,  Portugues,  Russian,  Svenska,  日本語,  中文
- Author name**:
- Description**:

# Roma (3): Module und Elemente

www.tei-c.org/Roma/startroma.php?mode=main

Wikipedia (en)



## Roma: generating validators for the TEI

You are currently working on **TEI Absolutely Bare**

### Modules

[New](#) [Customize](#) [Language](#) [Modules](#) [Add Elements](#) [Change Classes](#) [Schema](#) [Documentation](#) [Save Customization](#) [Sanity Checker](#)

#### List of TEI Modules

	Module name	A short description	Changes
<a href="#">add</a>	<a href="#">analysis</a>	Simple analytic mechanisms	
<a href="#">add</a>	<a href="#">certainty</a>	Certainty and uncertainty	
<a href="#">add</a>	<a href="#">core</a>	Elements common to all TEI documents	changed
<a href="#">add</a>	<a href="#">corpus</a>	Corpus texts	
<a href="#">add</a>	<a href="#">dictionaries</a>	Dictionaries	
<a href="#">add</a>	<a href="#">drama</a>	Performance texts	
<a href="#">add</a>	<a href="#">figures</a>	Tables, formulæ, notated music, and figures	
<a href="#">add</a>	<a href="#">gaiji</a>	Character and glyph documentation	
<a href="#">add</a>	<a href="#">header</a>	The TEI Header	changed
<a href="#">add</a>	<a href="#">iso-fs</a>	Feature structures	
<a href="#">add</a>	<a href="#">linking</a>	Linking, segmentation and alignment	
<a href="#">add</a>	<a href="#">msdescription</a>	Manuscript Description	

#### List of selected Modules

[remove](#) [core](#)  
[tei](#)  
[remove](#) [header](#)  
[remove](#) [textstructure](#)

# Roma (4): Schema generieren

www.tei-c.org/Roma/startroma.php?mode=createSchema

Wikipedia (en)



## Roma: generating validators for the TEI

You are currently working on TEI Absolutely Bare

### Time to give you a schema

- New
- Customize
- Language
- Modules
- Add Elements
- Change Classes
- Schema
- Documentation
- Save Customization
- Sanity Checker

#### Creating a schema

Which format do you prefer?

- RELAX NG schema (compact syntax)
- RELAX NG schema (compact syntax)
- RELAX NG schema (XML syntax)
- ISO Schematron
- Schematron
- W3C schema (in ZIP archive)
- DTD

Generate

We use RELAX NG schema (XML syntax) [http://www.w3.org/wiki/XML\\_Schema\\_Language\\_comparison](http://www.w3.org/wiki/XML_Schema_Language_comparison).

Roma was written by Arno Mittelbach and is maintained by Sebastian Rahtz. Sanity check written by Ioan Bernevig. Documentation language en. Please direct queries to the [TEI@Oxford project](mailto:TEI@oxfordproject.org).

This is Roma version 4.15, last updated 2013-01-11. Using TEI P5 version 2.5.0

# Roma (5): Dokumentation speichern

www.tei-c.org/Roma/startroma.php?mode=createDocumentation

Wikipedia (en)



## Roma: generating validators for the TEI

You are currently working on **TEI Absolutely Bare**

### Documentation?

- [New](#)
- [Customize](#)
- [Language](#)
- [Modules](#)
- [Add Elements](#)
- [Change Classes](#)
- [Schema](#)
- [Documentation](#)
- [Save Customization](#)
- [Sanity Checker](#)

Getting some nice documentation

Which output would you prefer?

- HTML web page
- HTML web page
- PDF**
- TEI Lite
- TEI ODD

[Generate](#)

## 2. Etwas mehr Informationen zum Klassen-System von TEI

# Zugrundeliegende <schemaSpec>

```
...
<schemaSpec ident="tei_bare" start="TEI">
  <moduleRef key="core"/>
  <moduleRef key="tei"/>
  <moduleRef key="header"/>
  <moduleRef key="textstructure"/>
  <elementSpec ident="abbr" mode="delete" module="core"/>
  <elementSpec ident="add" mode="delete" module="core"/>
  <!-- ... -->
  <elementSpec ident="trailer" mode="delete"
module="textstructure"/>
  <elementSpec ident="title" mode="change" module="core">
    <attList>
      <attDef ident="type" mode="delete"/>
    </attList>
  </elementSpec>
  <!-- ... -->
</schemaSpec>
...
```

# Das System der TEI-Klassen („classes“)

## Warum ein Klassen-System?

- TEI hat über 500 Elemente
- Klassen schaffen Übersicht, Modularität, machen Änderungen einfacher

## Welche Klassen gibt es?

- Attribut-Klassen: alle Elemente in der Klasse haben die gleichen Attribute
- Modell-Klassen: die Elemente haben den gleichen Kontext

## Außerdem

- Klassen können andere Klassen beinhalten
- Ein Element kann verschiedenen Klassen angehören
- Die Modul-Zugehörigkeit ist davon unabhängig

# Attribut-Klassen

## Allgemeines

- Attribut-Klassen haben Namen, die mit „att.“ beginnen, bspw.: „att.named“
- Jede Attribut-Klasse beinhaltet ein Set von Attributen
  - „att.named“ => @key und @ref;
  - „att-typed“ => @type und @subtype.

## Also

- Soll ein Element das Attribut @type haben, kann man es der Klasse „att.typed“ hinzufügen



# Eine wichtige Attribut-Klasse: „att.global”

Alle Elemente sind Mitglieder der Klasse „att.global”

Diese Klasse enthält:

- @xml:id - „unique identifier”
- @xml:lang – die Sprache des Inhalts des Elements
- @n – eine Nummer oder ein Kürzel für ein Element
- @rend – wie der Inhalt des Elements im Quelldokument aussah

# Modell-Klassen

## Modell-Klassen definieren Kontexte

- Sie enthalten Elemente, die im gleichen Kontext erlaubt sind
- Die Namen von Klassen beruhen auf einem „like“ oder „part“-Suffix
- „model.biblLike.class“ enthält Element, die sich „like“ <bibl> verhalten

## Beispiel

- Ein Element das immer dort erlaubt sein soll, wo auch <bibl> erlaubt ist, kann der Klasse „model.biblLike.class“ hinzugefügt werden
- Alle Elemente, die in <bibl> vorkommen können, gehören der Klasse „model.biblPart.class“ an.

# Modell-Klassen: Basics

Es gibt drei „base model classes“

- `divisions` – Grobstruktur von Texten; Bsp.: `<div>`
- `chunks` – Elemente, die in „divisions“ vorkommen; Bsp.: `<p>`
- „phrase-level elements“ - Elemente in „chunks“; Bsp.: `<hi>`

Außerdem:

- `inter-level elements` – Innerhalb oder zwischen „chunks“: Bsp.: `<list>`
- `components` – Kleinere Elemente, die direkt in „divisions“ vorkommen
- „`model.global`“: Elemente, die überall vorkommen können