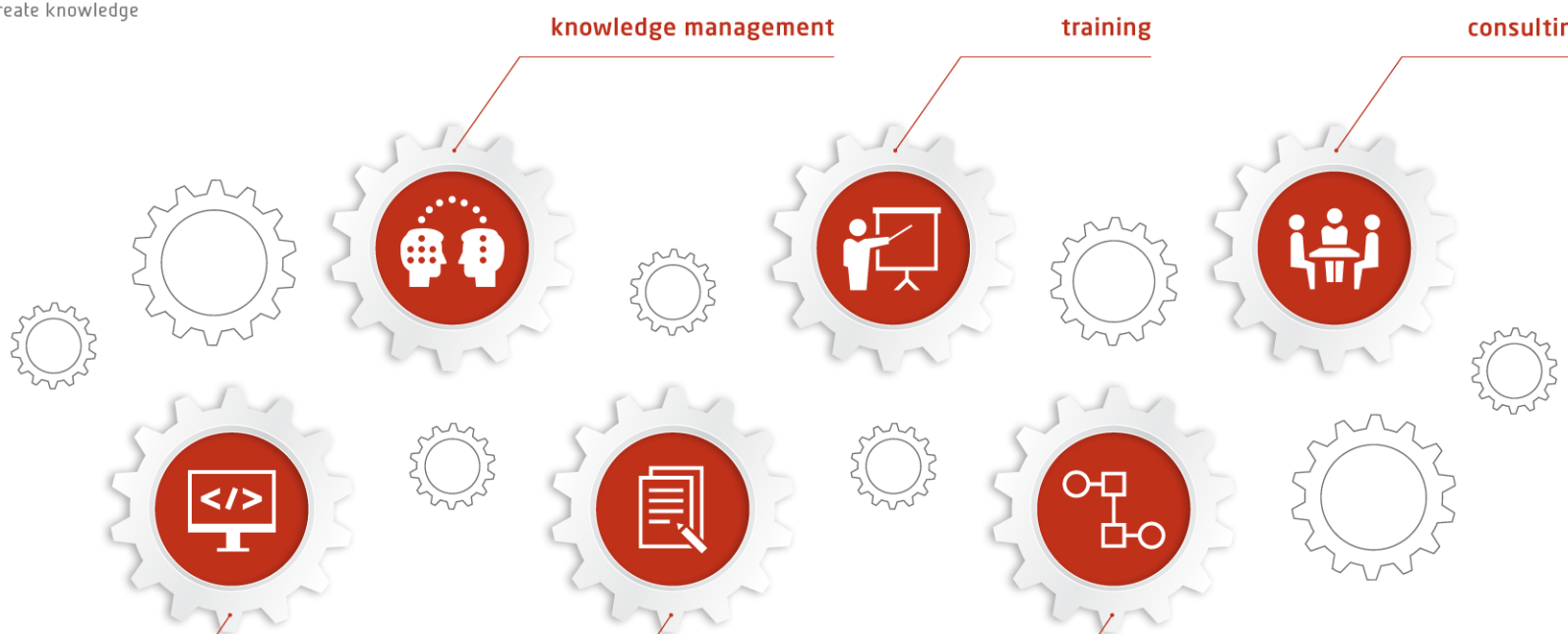
parson

Relax, Just Do It – DITA Customization with RELAX NG

Frank Ralf

parson
we create knowledge



www.parson-europe.com

parson

Fundamentals

DITA Architecture

Introduction to Customization

Configuration: Custom Concept Shell

Constraint: Custom Topic Body and Domain Constraint

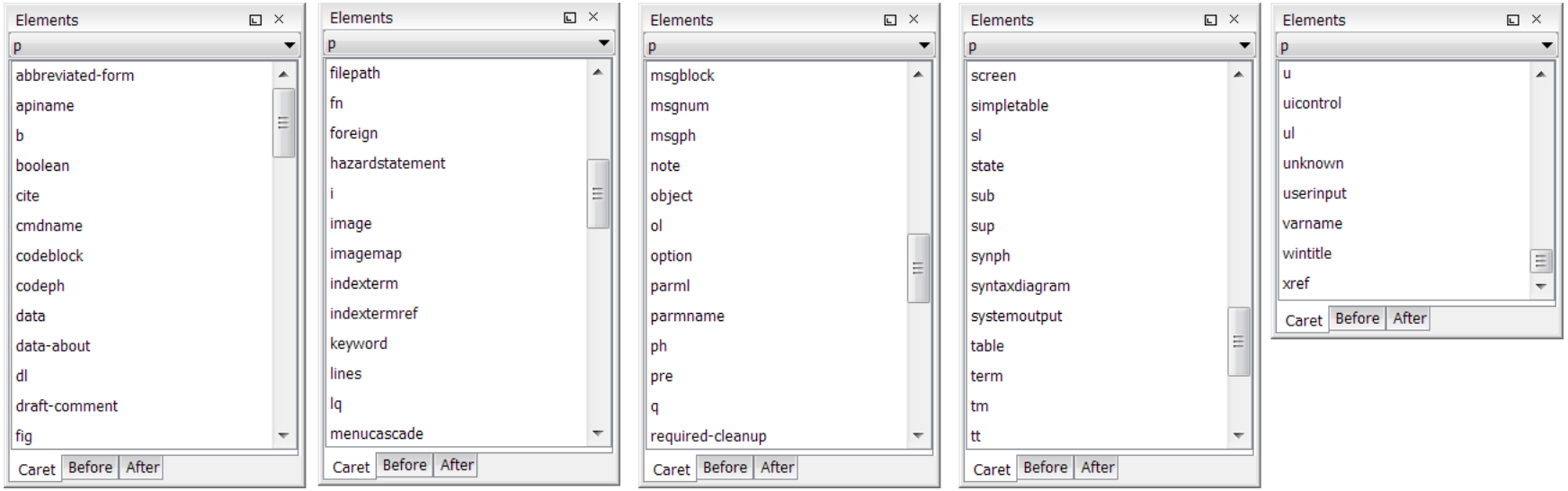
Specialization: New Element

Conclusion

parson

Why Customize DITA?

- Huge number of elements/attributes, unnecessary complexity, authors confused, leads to inconsistencies
- Customization is part of information architecture process
- Default content of the <p> element:



parson

DTD, XSD, or RNG?

- DTD = Document Type Definition
- XSD = XML Schema Definition
- Relax NG (RNG) = Regular Language Description for XML New Generation

- Up to DITA 1.2 > DTD
- XSD not recommended for DITA: “Generating constraints with XSD is essentially not possible in the general case.” (Eliot Kimber)
- RNG = normative grammar as of DITA 1.3

parson

Advantages of RNG

- Self-integration
- Easily combine multiple definitions of an element
- Valid XML
- Integration of foreign grammars > embed Schematron rules
- Easier to learn
- Data typing > not compatible with DTD conversion

parson

RNG – Basic Syntax

```
<element name="p">
  <oneOrMore>
    <choice>
      <text/>
      <element name="b"/>
      <element name="i"/>
    </choice>
  </oneOrMore>
  <zeroOrMore>
    <element name="fig"/>
  </zeroOrMore>
  <attribute name="importance">
    <choice>
      <value>low</value>
      <value>high</value>
    </choice>
  </attribute>
</element>
```

```
<grammar>
  <start>
    <ref name = "concept.element"/>
  </start>
  <define name = "concept.element">
    <element name= "concept"/>
      <ref name="title"/>
      <optional>
        <ref name="shortdesc"/>
      </optional>
      <ref name="conbody"/>
      <ref name="related-links"/>
  </define>
</grammar>
```

parson

Fundamentals

DITA Architecture

Customization - Introduction

Configuration: Custom Concept Shell

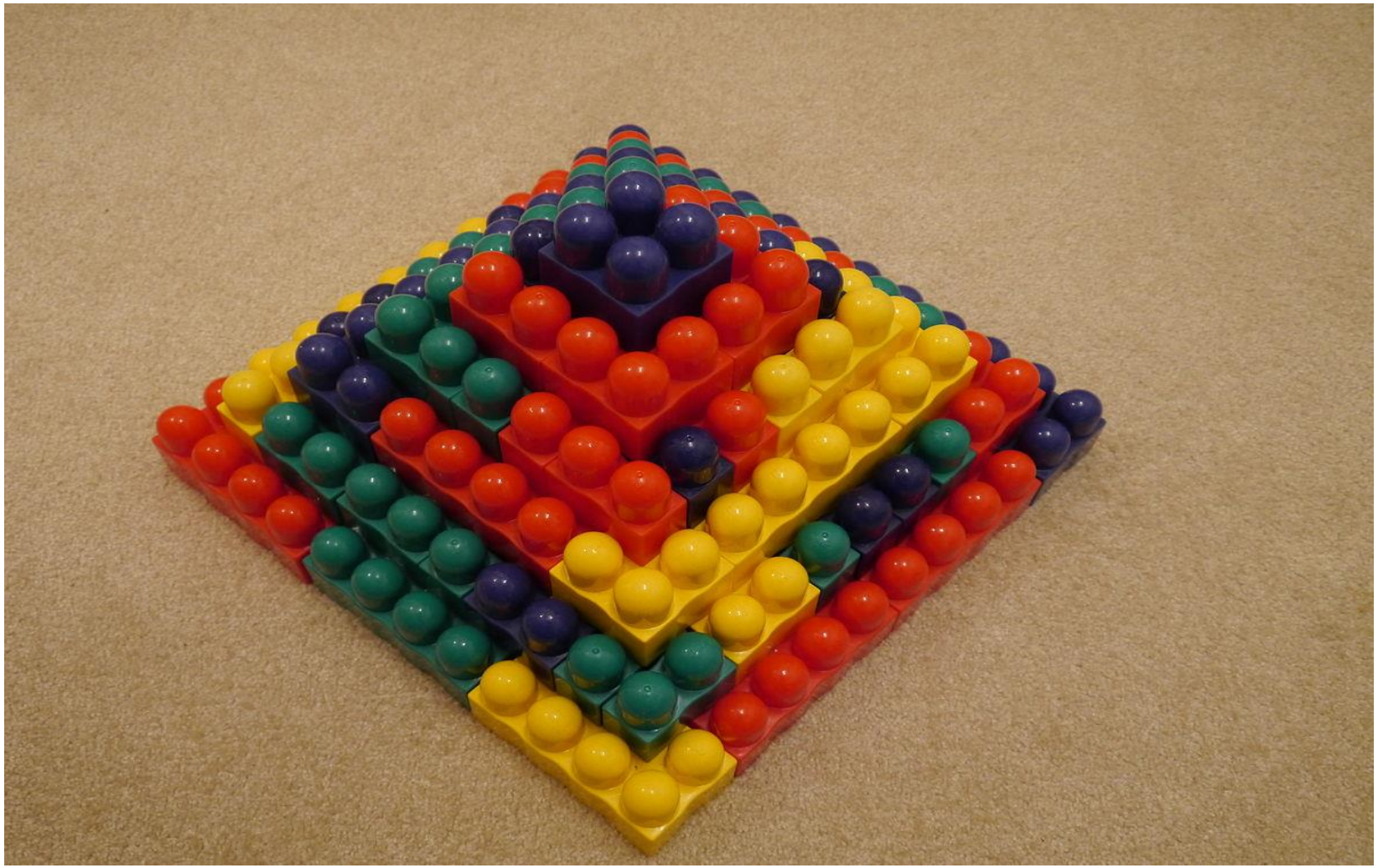
Constraint: Custom Topic Body and Domain Constraint

Specialization: New Element

Conclusion

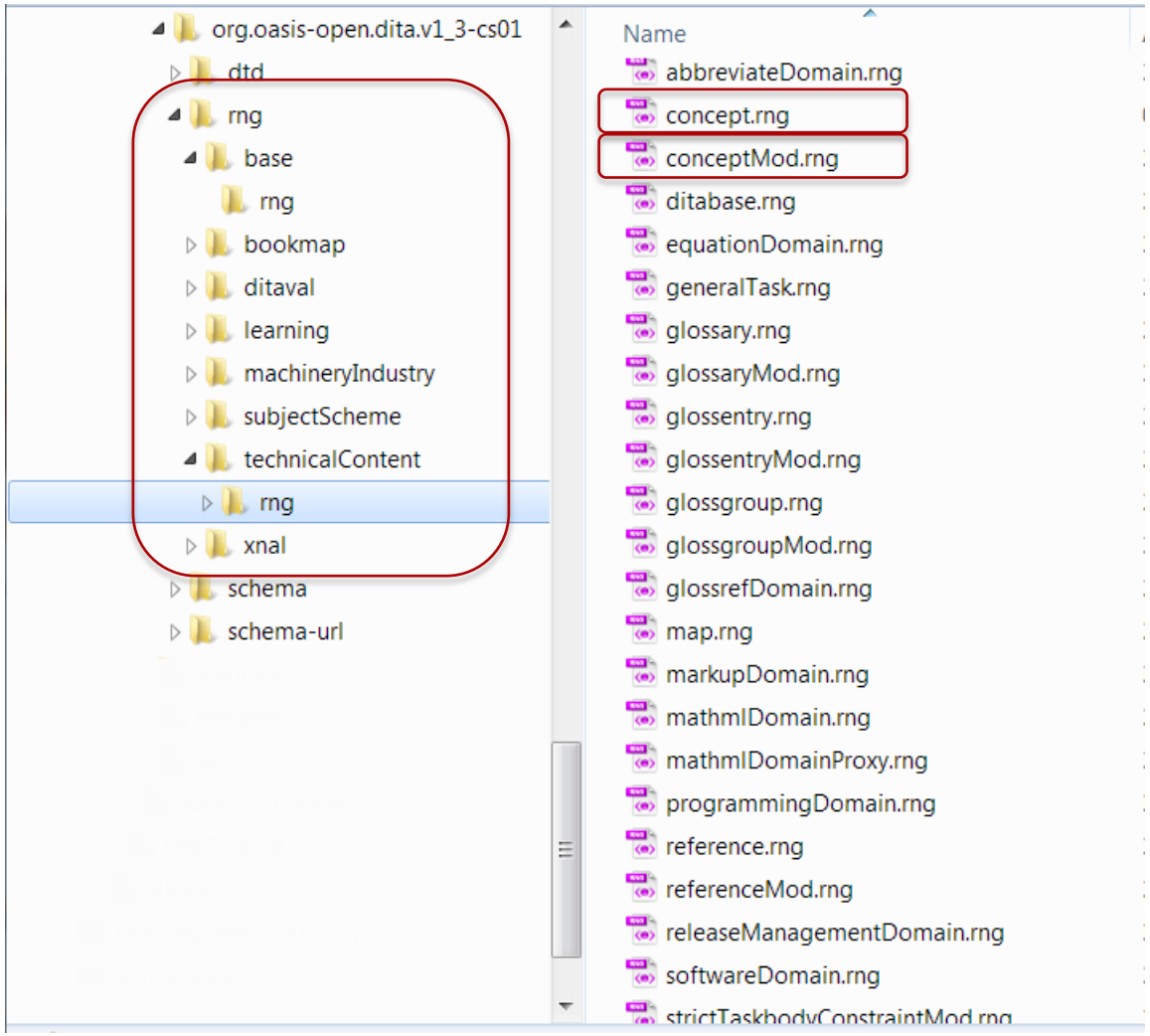
parson

DITA Architecture Basics



parson

Modularization



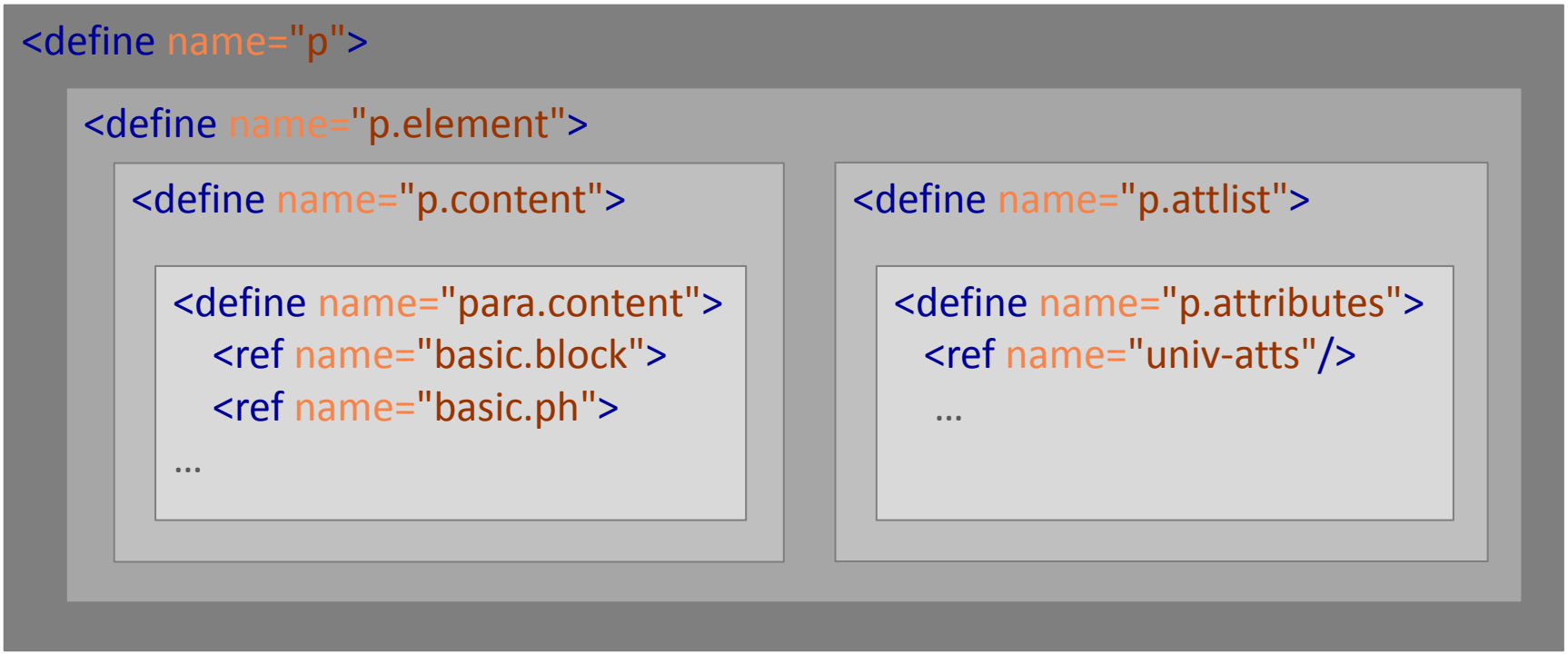
Each document type has an associated RNG file called shell.

Element and attribute definitions are outsourced to Mod files.

parson

Cascading Definitions

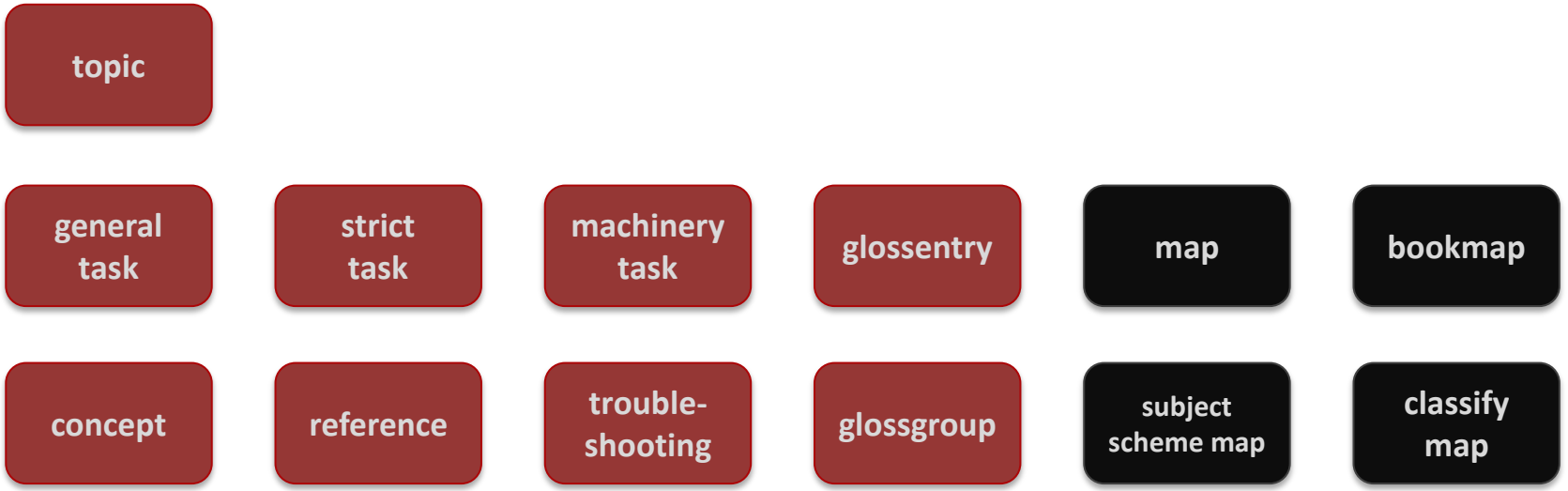
- Content models of elements DITA RNG files use defines within defines.



parson

Document Type Shells

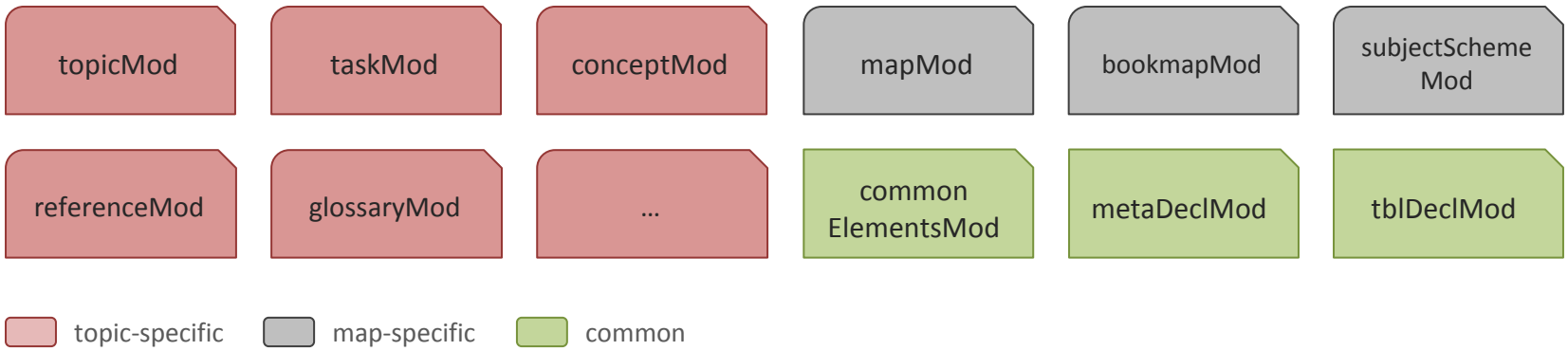
- XML grammar files that define which elements and attributes are allowed in a DITA topic.
- Do not directly declare element or attribute types.
- Integrate structural modules, domain modules, and constraint modules.



parson

Structural Modules and Common Elements

- Structural modules define map or topic types.
- Contain actual topic or map content.
- Common elements, metadata definitions and table elements are outsourced to separate files.



parson

Examples of Common Element Sets

- Common element sets are reused in many element definitions.

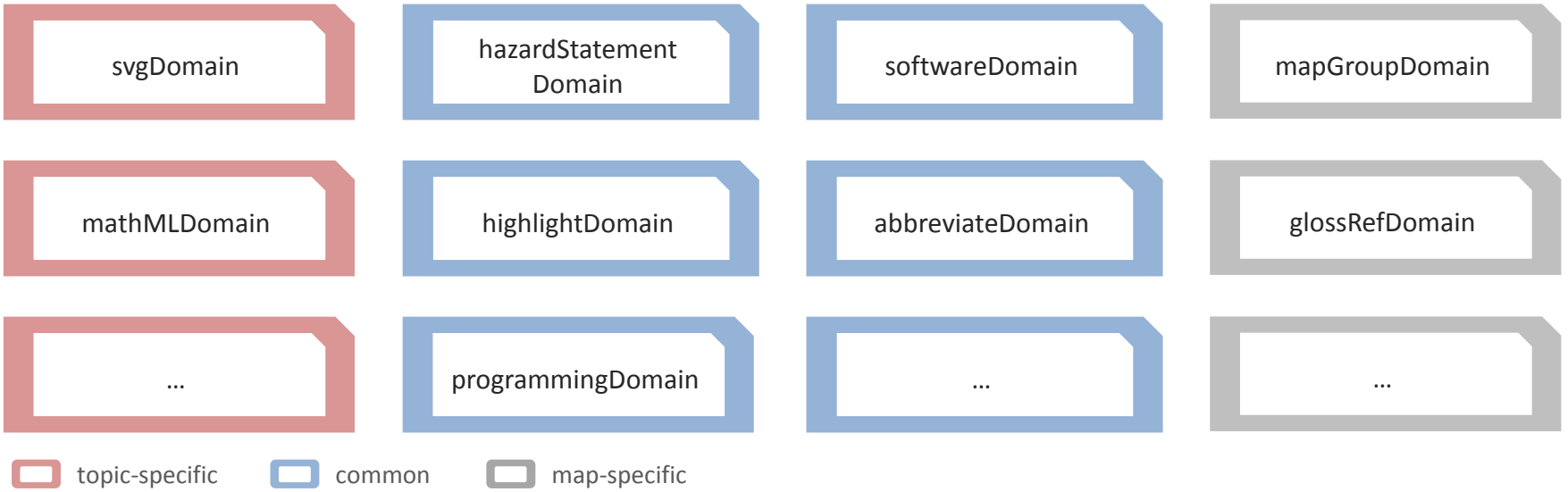
```
<define name="basic.ph">
      <choice>
        <ref name="boolean"/>
        <ref name="cite"/>
        <ref name="keyword"/>
        <ref name="ph"/>
        <ref name="q"/>
        <ref name="term"/>
        <ref name="text" dita:since="1.3"/>
        <ref name="tm"/>
        <ref name="xref"/>
        <ref name="state"/>
      </choice>
    </define>
```

```
<define name="basic.block">
     <choice>
       <ref name="dl"/>
       <ref name="div"/>
       <ref name="fig"/>
       <ref name="image"/>
       <ref name="lines"/>
       <ref name="lq"/>
       <ref name="note"/>
       <ref name="object"/>
       <ref name="ol"/>
       <ref name="p"/>
       <ref name="pre"/>
       <ref name="simpletable"/>
       <ref name="sl"/>
       <ref name="table"/>
       <ref name="ul"/>
      </choice>
    </define>
```

parson

Domains

- Provide semantic categories to group elements and attributes across document types.
- Useful to remove or add whole groups of elements from or to document or map types.



parson

Reuse & Redefinition

- Cascading defines allow reuse and redefinition in multiple places.

commonElementsMod.rng

```
<define name="ph">
    <ref name="ph.element"/>
  </define>
```

➢ adds ph

highlightDomain.rng

```
<define name="ph" combine="choice">
    <ref name="hi-d-ph"/>
  </define>
```

➢ adds b, i, line-through, overline, sup, sub, tt, u

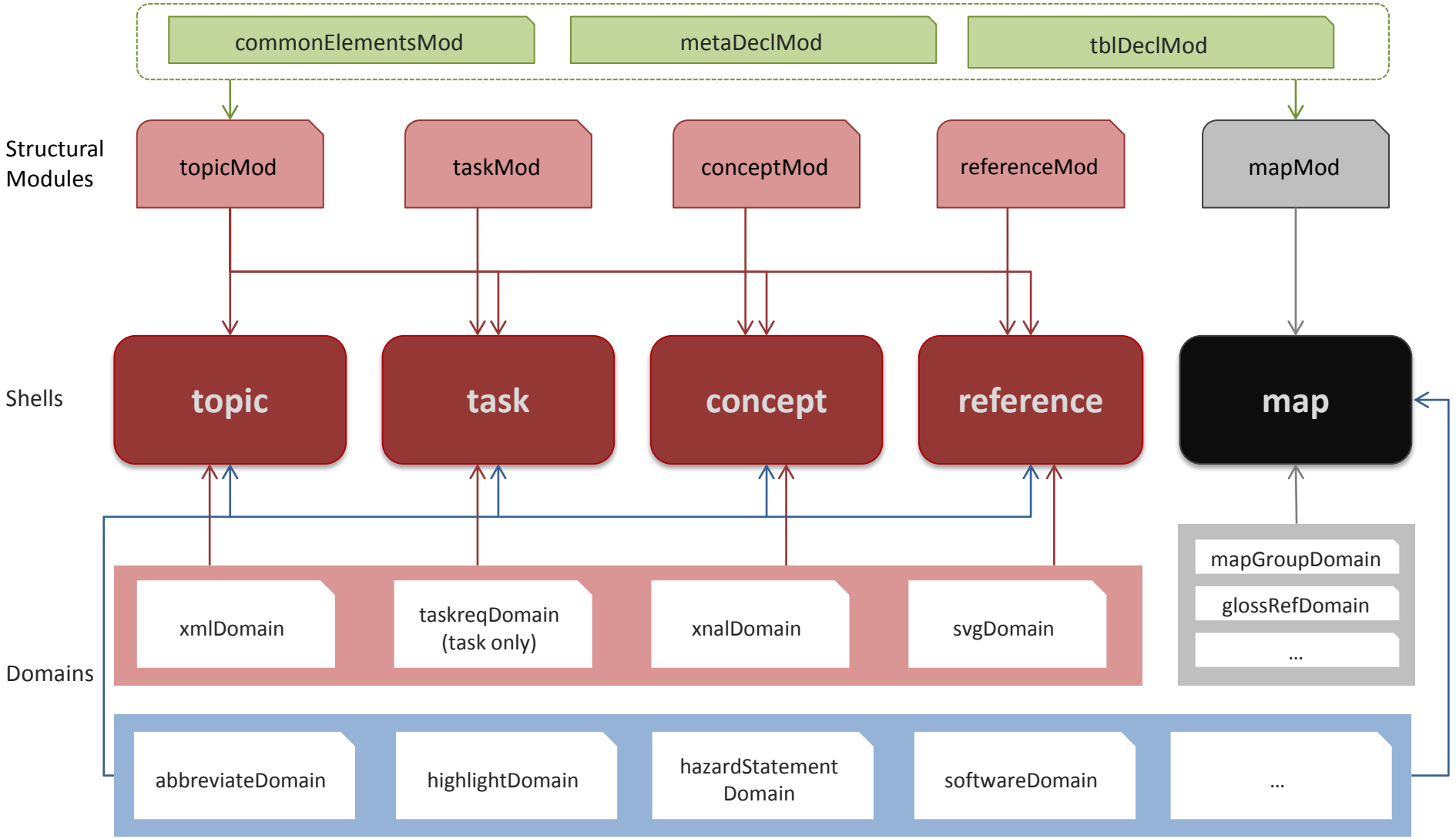
programmingDomain.rng

```
<define name="ph" combine="choice">
    <ref name="pr-d-ph"/>
  </define>
```

➢ adds codeph, synph

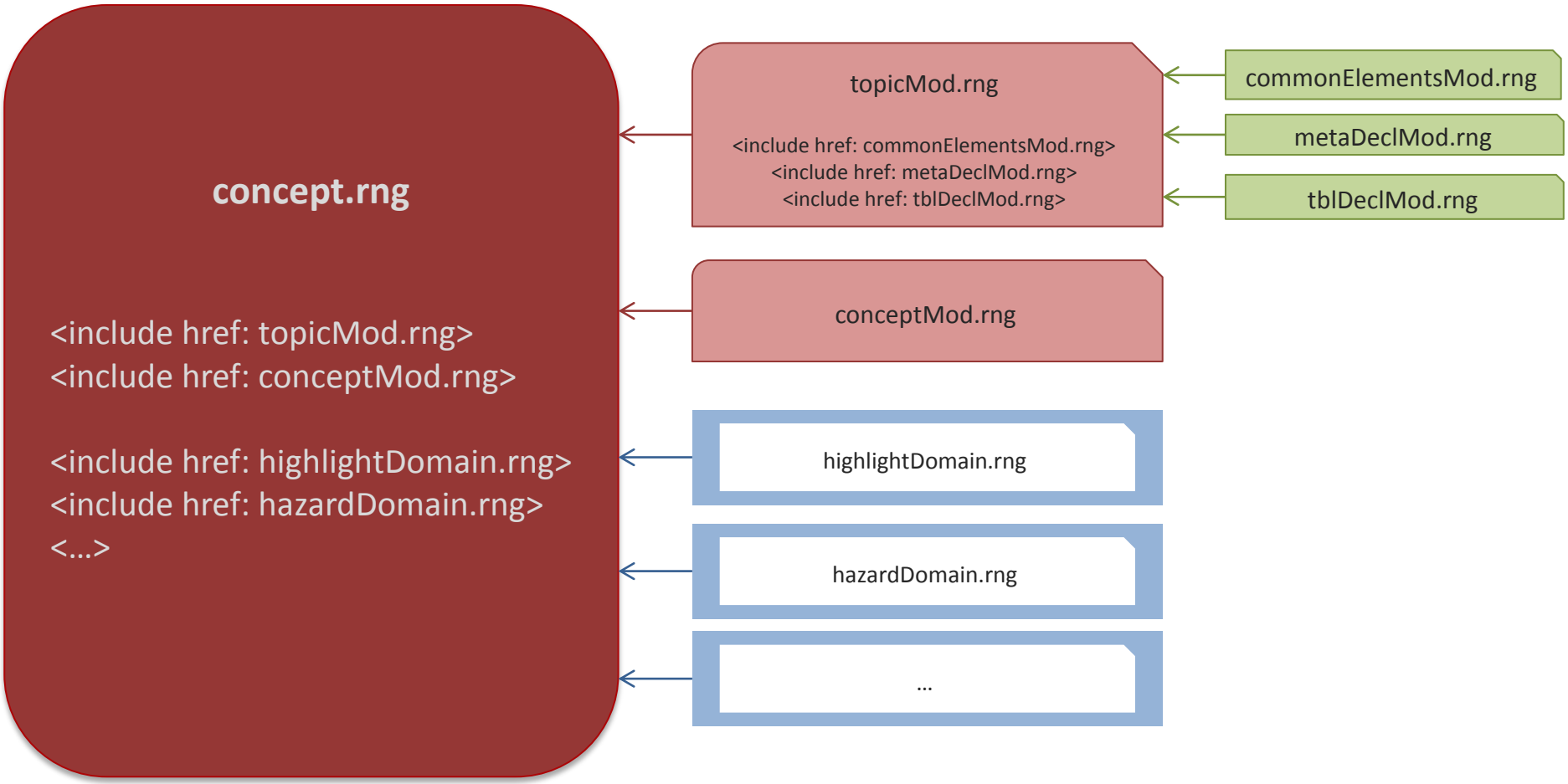
➢ All specialized elements are automatically allowed where the base element is allowed.

parson


commonElementsMod
metaDeclMod
tblDeclMod
Structural Modules
topicMod
taskMod
conceptMod
referenceMod
mapMod
Shells
topic
task
concept
reference
map
Domains
xmlDomain
taskreqDomain (task only)
xnalDomain
svgDomain
mapGroupDomain
glossRefDomain
...
abbreviateDomain
highlightDomain
hazardStatement Domain
softwareDomain
...

parson

Includes and Chaining



parson

XML Catalog Files

- Map DITA topics to RNG files via URNs.
- Include RNG files via URNs.
- Separate catalog for customization files to resolve new identifiers.
- Combine custom catalog with default DITA catalog > integrate both as DITA-OT plug-ins.
- Observe DITA naming conventions for URNs.

catalog.xml

```
<uri name="urn:oasis:names:tc:dita:rng:concept.rng" uri="rng/concept.rng"/>
```

concept topic

```
<?xml-model href="urn:oasis:names:tc:dita:rng:concept.rng"
schematypens="http://relaxng.org/ns/structure/1.0"?>
```

parson

Domains Attribute

- RNG-based shells directly specify values for the @domains attribute.
- The attribute values correspond to the integrated domains and structural types.
- Each domain and constraint module *MUST* provide a value for use by the @domains attribute.

```
<define name="domains-att" combine="interleave"><optional>
    <attribute name="domains"
     a:defaultValue="(topic abbrev-d)
                (topic concept) (topic equation-d)
                (topic hazard-d) (topic hi-d)
                (topic indexing-d) (topic markup-d xml-d)
                (topic markup-d) (topic mathml-d) (topic pr-d)
                (topic relmgmt-d) (topic svg-d) (topic sw-d)
                (topic ui-d) (topic ut-d) a(props deliveryTarget)
                (topic concept napsor-concept-c)"/>
  </optional></define>
```

parson

Fundamentals

DITA Architecture

Customization - Introduction

Configuration: Custom Concept Shell

Constraint: Custom Topic Body and Domain Constraint

Specialization: New Element

Conclusion

parson

DITA Customization

Modularization allows to use original parts and altered parts.

Configuration

- Configure document type shells.

Constraints

- Restrict content models and attribute lists.

Specialization

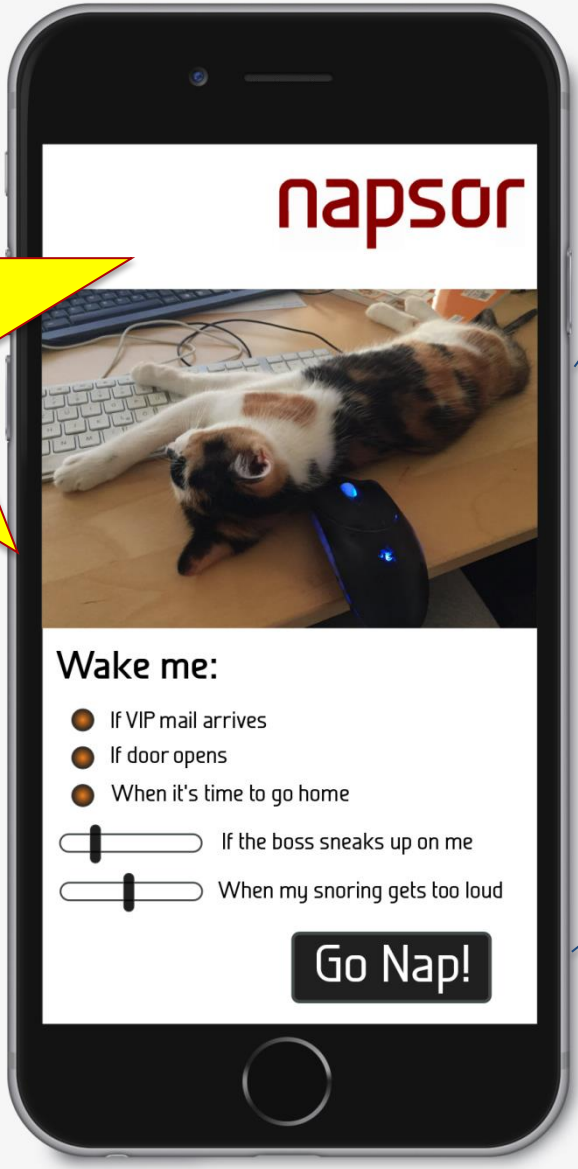
- Create new elements and attributes.

Generalization

- Reversing specializations > not covered today.

parson

Brand-new app

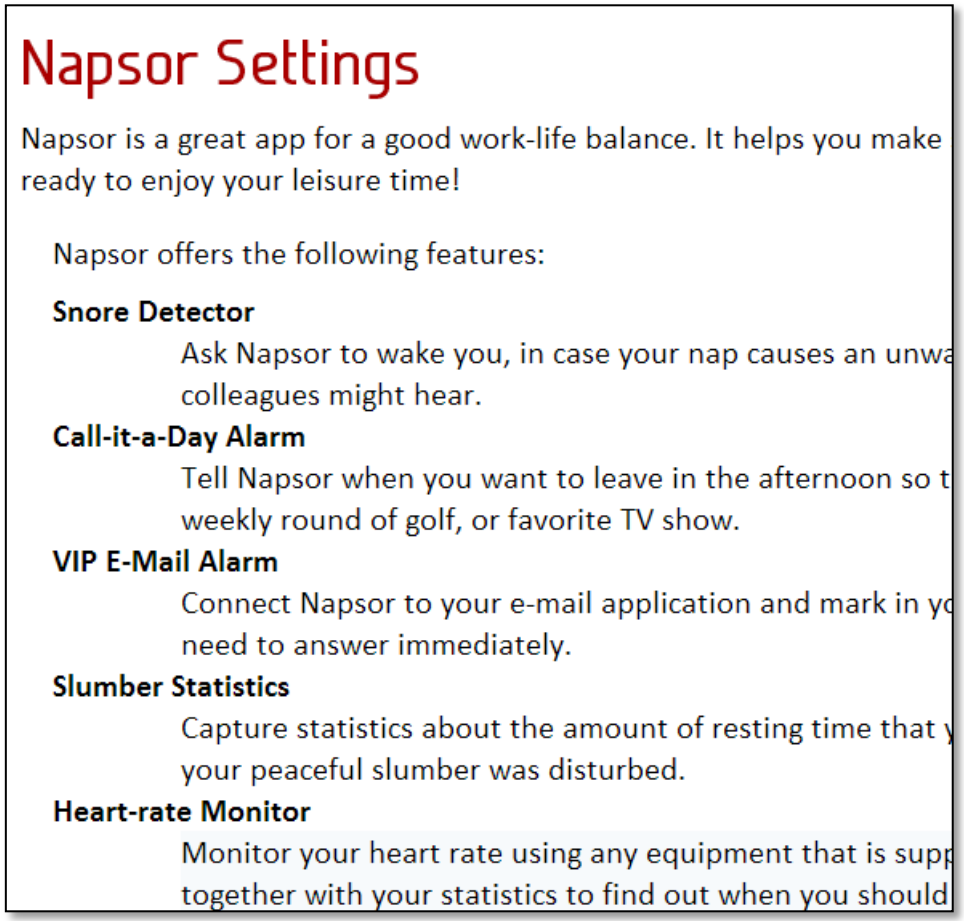


Do as your cat would: Go Nap!

Never again tired after work. Go Nap!

parson

Napsor Documentation



Napsor Settings

Napsor is a great app for a good work-life balance. It helps you make ready to enjoy your leisure time!

Napsor offers the following features:

Snore Detector

Ask Napsor to wake you, in case your nap causes an unwa colleagues might hear.

Call-it-a-Day Alarm

Tell Napsor when you want to leave in the afternoon so t weekly round of golf, or favorite TV show.

VIP E-Mail Alarm

Connect Napsor to your e-mail application and mark in yo need to answer immediately.

Slumber Statistics

Capture statistics about the amount of resting time that y your peaceful slumber was disturbed.

Heart-rate Monitor

Monitor your heart rate using any equipment that is supp together with your statistics to find out when you should

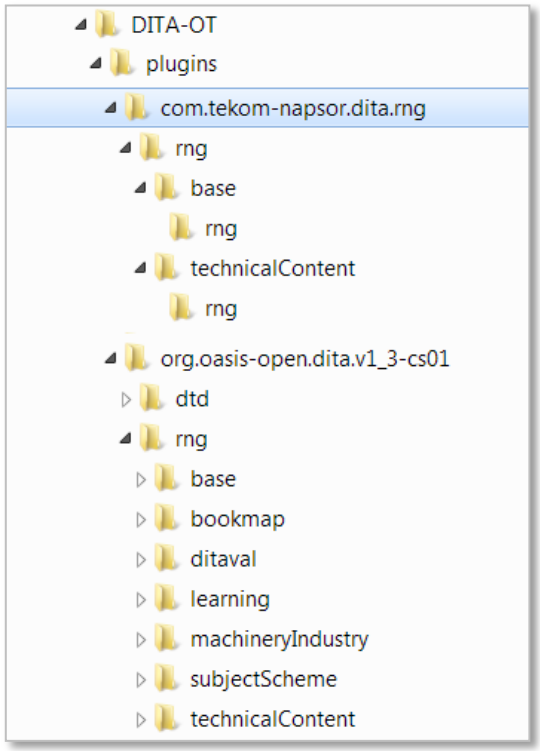
- Mobile help
  > short topics, no frills
- Software only
  > no machinery stuff
- Only semantic highlighting
  > no bold or italics
- Special request from Marketing
  > thankyou element
- Fun!

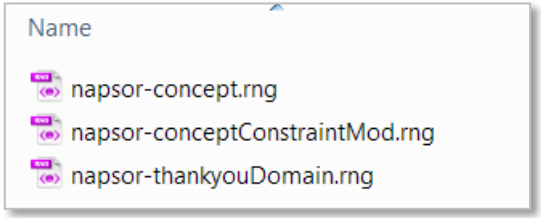
parson

Preparations

- Current version of DITA-OT, e.g. 2.0.1
- DITA 1.3 CS01 RNG grammar
- ➢ DITA-OT plug-in created and integrated

- New plug-in = com.tekom-napsor.dita.rng
- ➢ Already set up and integrated.
- ➢ Same folder structure as DITA 1.3 plug-in.
- ➢ Catalog files prepared for all exercises.
- ➢ Shell and Mod files prepared.
- ➢ All relative paths in @href attributes exchanged with URNs.






```
<!-- System ID (URL) catalog entries -->
<system systemId="urn:napsor:dita:rng:napsor-concept.rng"
      uri="rng/napsor-concept.rng"/>
<system systemId="urn:napsor:dita:rng:napsor-conceptConstraintMod.rng"
      uri="rng/napsor-conceptConstraintMod.rng"/>
<system systemId="urn:napsor:dita:rng:napsor-thankyouDomain.rng"
      uri="rng/napsor-thankyouDomain.rng"/>
```

parson

Fundamentals

DITA Architecture

Customization - Introduction

Configuration: Custom Concept Shell

Constraint: Custom Topic Body and Domain Constraint

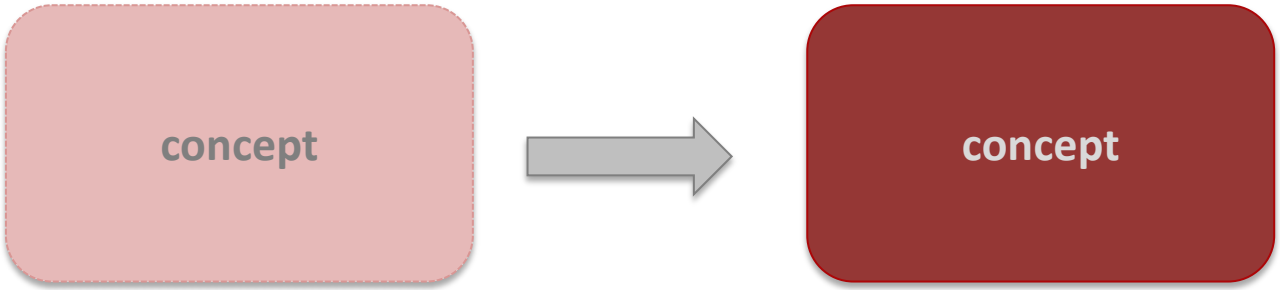
Specialization: New Element

Conclusion

parson

A) Configuration – Custom Document Type

- Create a shell file (RNG) for a new document type = *napsor-concept.rng*
- Customization of default DITA concept
- Preparation for modifying the actual content model of the document type
- New shell = structural specialization



Default DITA

Napsor DITA

parson

A) Create Napsor Concept Topic

1. Copy *concept.rng* to the new plug-in folder and rename as *napsor-concept.rng.*
2. Modify *napsor-concept.rng*:
   1. Set <moduleShortName> to napsor-concept and update metadata in header.
   2. Set <rngShell> to urn:napsor:dita:rng:napsor-concept.rng.
3. Open *napsor-documentation.dita* and change the xml-model processing instruction to:

   ```
   <?xml-model href="urn:napsor:dita:rng:napsor-concept.rng"
   schematypens="http://relaxng.org/ns/structure/1.0"?>
   ```

4. Validate the topic.

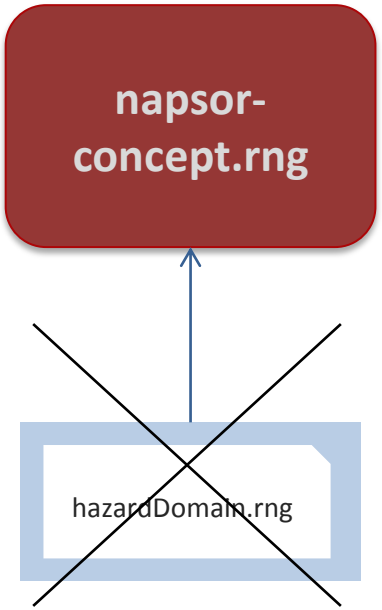
✓ Result:

- Base topic shell is ready. Topic validates against new shell.

parson

B) Configuration – Domains

- Remove domain from document type by commenting out the <include>.
- Make hazard domain unavailable, not needed for app documentation.



parson

B) Remove the Hazard Domain

1. Open *napsor-concept.rng*.
2. Search for MODULE INCLUSIONS.
3. Comment out <include href="urn:oasis:names:tc:dita:rng:**hazardDomain**.rng"/>.
4. Open *napsor-documentation.dita* and validate the topic.
5. Replace <hazardstatement> element with <note type="tip">.
6. Remove the hazard domain from the domains attribute:

```
<attribute name="domains" a:defaultValue="(topic abbrev-d)
                    (topic concept)
                    (topic equation-d)
                    (topic hazard-d)
```

✓ Result:

- Elements from the hazard domain are unavailable in editor and do not validate.

parson

Fundamentals

DITA Architecture

Customization - Introduction

Configuration: Custom Concept Shell

Constraint: Custom Topic Body and Domain Constraint

Specialization: New Element

Conclusion

parson

Constraint Modules

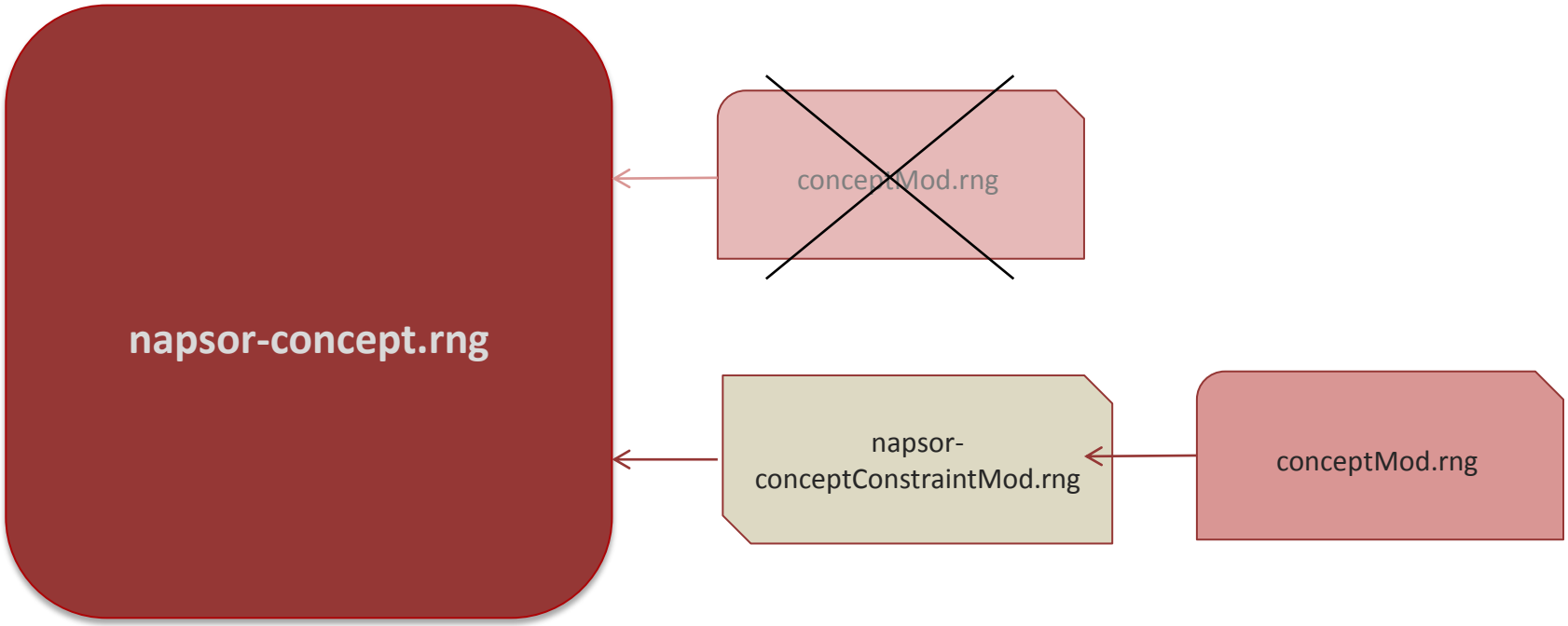
- Used to modify content models of elements and attributes:
  - **Remove** optional elements or attributes.
  - Make optional elements or attributes **required**.
  - Enforce a specific **sequence** of elements.
  - Limit available attribute values or define **specific attribute values**.

- Example: Disallow unsemantic highlighting elements, such as <b> and <i>.

parson

C) Constraining the Topic Content

- Create separate files for constraint modules.
- Insert constraint modules in include chain.
- Add values to domain contribution of document or map type.



parson

C) Simplify the Concept Body (1)

1. Copy *conceptMod.rng* to the new plug-in folder and rename as *napsor-conceptConstraintMod.rng.*
2. Open *napsor-conceptConstraintMod.rng.*
3. Set <moduleShortName> to napsor-concept-c and update metadata in header.
4. Set <rngMod> to urn:napsor:dita:rng:napsor-conceptConstraintMod.rng.
5. Set <domainsContribution> to (topic concept napsor-concept-c).
6. Remove all defines from the file.

parson

C) Simplify the Concept Body (2)

7. Add a div for the constraints with an include of the original concept mod:

```
<div><a:documentation>CONTENT MODEL OVERRIDES</a:documentation>
<include href="urn:oasis:names:tc:dita:rng:conceptMod.rng"> </include>
</div>
```

8. Add the following define for the <conbody> element **in** the include:

```
<define name="conbody.content">
      <zeroOrMore>
       <choice>
        <ref name="basic.block"/>
        <ref name="draft-comment"/>
       </choice>
      </zeroOrMore>
 </define>
```

   - You can further restrict the conbody by resolving basic.block.

9. Save the file.

parson

C) Prepare the Shell to Use the Constraint

1. Open *napsor-concept.rng*.
2. Comment out the include of the concept mod:

```
<include href="urn:oasis:names:tc:dita:rng:conceptMod.rng"><define name="concept-info-types"><ref name="concept.element"/></define></include>
```

3. Add a div for constraint modules with an include for the Napsor concept Mod constraint:

```
<div><a:documentation>CONSTRAINT MODULES</a:documentation>
    <include href="urn:napsor:dita:rng:napsor-conceptConstraintMod.rng"/>
</div>
```

4. Add the module's short name to the domain contribution:

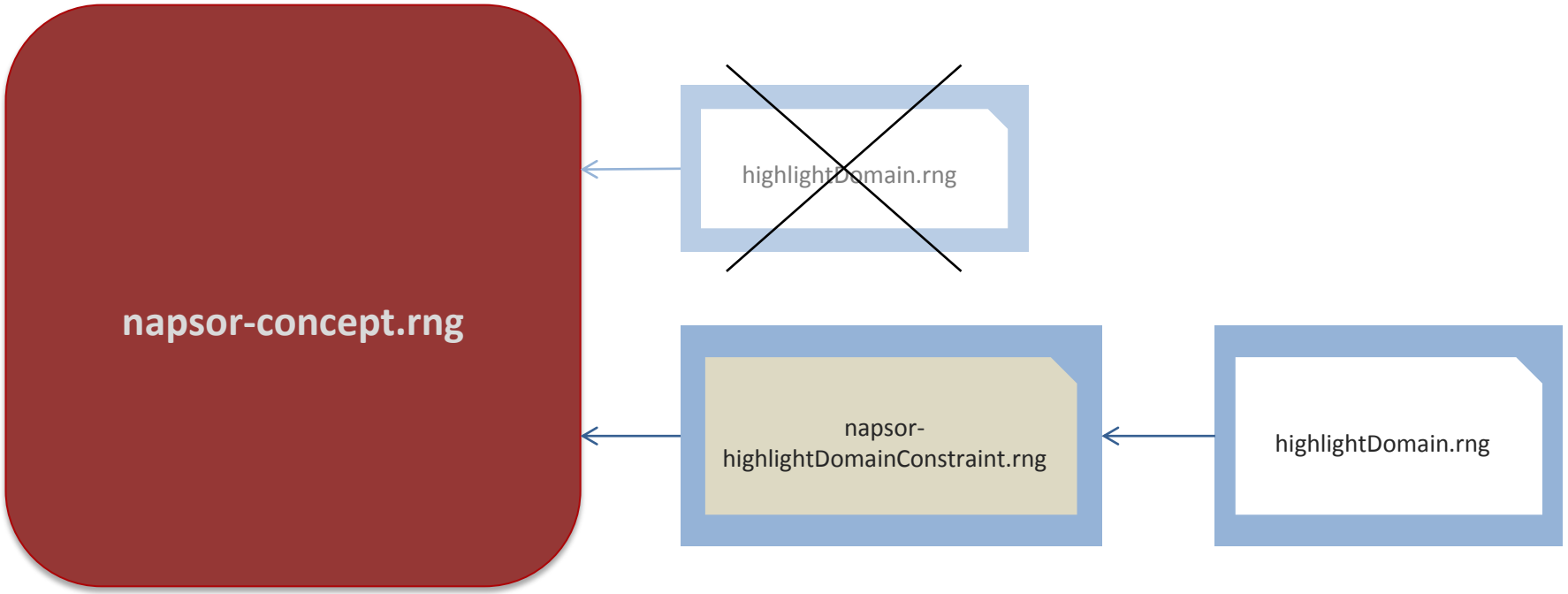
```
<attribute name="domains"
a:defaultValue="(topic abbrev-d) …  (topic concept napsor-concept-c)"
```

5. Open *napsor-documentation.dita* and validate the topic.
6. Convert the invalid <section> to something valid.

parson

D) Constraining a Domain

- Create separate files for constraint modules.
- Insert constraint modules in include chain.
- Add values to domain contribution of document or map type.



parson

D) Constrain the Highlight Domain (1)

1. Copy *highlightDomain.rng* to the new plug-in folder and rename as *napsor-highlightDomainConstraint.rng.*
2. Open *napsor-highlightDomainConstraint.rng.*
3. Set <moduleShortName> to napsor-highlightDomain-c and update metadata in header.
4. Set <rngMod> to urn:napsor:dita:rng:napsor-highlightDomainConstraint.rng.
5. Set <domainsContribution> to (topic hi-d napsor-highlightDomain-c).
6. Remove all defines from the file.

parson

D) Constrain the Highlight Domain (2)

7. Add a div for the constraints with an include of the original highlight domain:

```
<div><a:documentation>CONTENT MODEL OVERRIDES</a:documentation>
  <include href="urn:oasis:names:tc:dita:rng:highlightDomain.rng"/>
</div>
```

8. Add defines for the following elements:

   <b.element>, <i.element>, <overline.element>, <line-through.element>, <tt.element>, <u.element>

9. Set the content model of the elements to <notAllowed/>.

   Example: <define name="b.element"><notAllowed/></define>

10. Save the file.

- Alternative: Remove elements from domain extension pattern hi-d-ph for better DTD compatibility.

parson

D) Prepare the Shell to Use the Constraint

1. Open *napsor-concept.rng*.
2. Comment out the include of the highlight domain:

```
<include href="urn:oasis:names:tc:dita:rng:highlightDomain.rng"/>
```

3. In the div for constraint modules, add an include for the highlight domain constraint:

```
<div><a:documentation>CONSTRAINT MODULES</a:documentation>
<include href="urn:napsor:dita:rng:napsor-highlightDomainConstraint.rng"/>
```

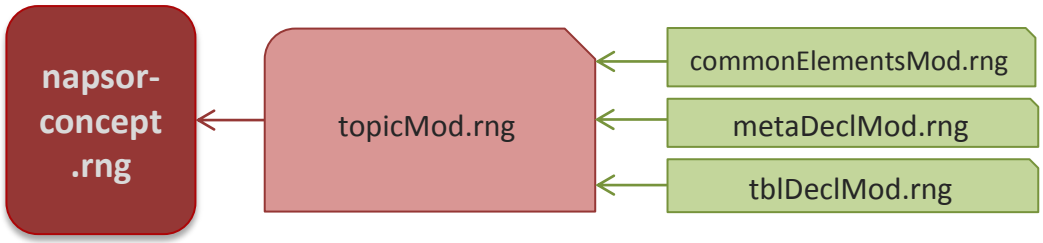
4. Add the module's short name to the domain contribution:

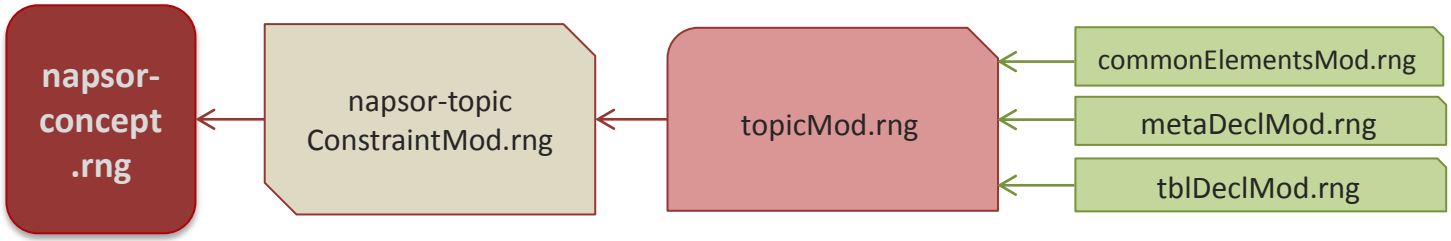
```
<attribute name="domains"
a:defaultValue="(topic abbrev-d) …  (topic hi-d napsor-highlightDomain-c)"
```

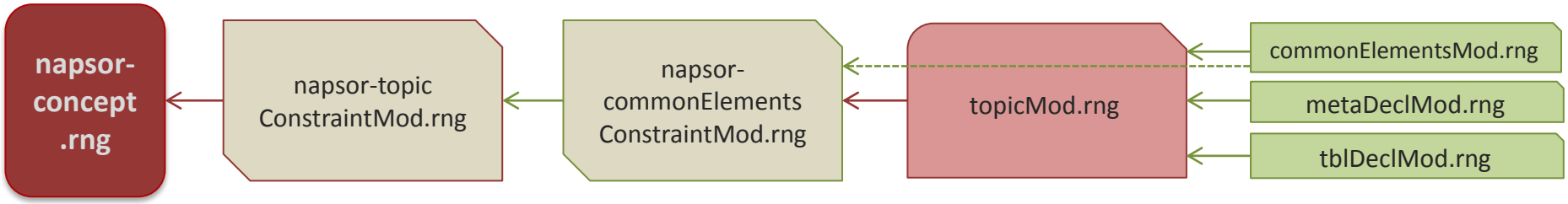
5. Open *napsor-documentation.dita* and validate the topic.
6. Remove the invalid <b> and <i> elements.

parson

Constraining Common Elements







parson

- Fundamentals
- DITA Architecture
- Customization - Introduction
- Configuration: Custom Concept Shell
- Constraint: Custom Topic Body and Domain Constraint
- Specialization: New Element
- Conclusion

parson

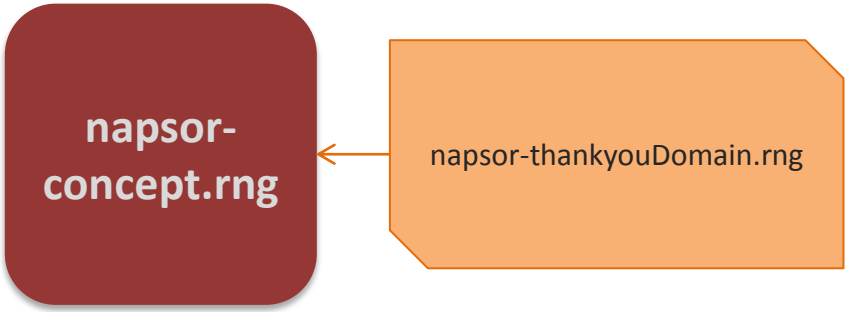
Specialization

- Enables creation of new elements and attributes that are derived from existing base types.
- Can be processed by DITA applications.
- Implemented as vocabulary modules that are integrated into shells.

parson

E) Element Specialization

- Set up a container file.
- Include the container file in the shell.
- Define a new element.
- Integrate the element into base element using @combine.



parson

E) Create Thankyou Element (1)

1. In the technicalContent folder, create a new RNG file called *napsor-thankyouDomain.rng*.
2. Add a header similar to the highlight domain file, including the following:
   - <moduleShortName>napsor-thankyou-d</moduleShortName>
   - <rngMod>urn:napsor:dita:rng:napsor-thankyou.rng</rngMod>
   - <domainsContribution>(topic napsor-thankyou-d)</domainsContribution>
3. Add a new domain extension pattern that includes the thankyou element:

```
<define name="thankyou-d-ph">
  <ref name="thankyou.element"/>
</define>
```

4. Add a define for <ph> that integrates the new definition with the base type:
5.

```
<define name="ph" combine="choice">
 <ref name="thankyou-d-ph"/> </define>
```

parson

E) Create Thankyou Element (2)

6. Add the following defines for the <thankyou> element to the element type name patterns:

```
<define name="thankyou">
  <ref name="thankyou.element"/>
</define>

<define name="thankyou.element">
  <element name="thankyou" dita:longName="Thank you">
    <ref name="thankyou.attlist"/>
    <ref name="thankyou.content"/>
  </element>
</define>

<define name="thankyou.content">
  <text/>
</define>
```

parson

Create Thankyou Element (3)

7. Add the following defines for the thankyou attributes:

```
<define name="thankyou.attlist" combine="interleave">
  <ref name="thankyou.attributes"/>
</define>

<define name="thankyou.attributes">
  <ref name="univ-atts"/>
  <optional>
    <attribute name="outputclass"/>
  </optional>
</define>
```

8. Add the following define to declare the @class attribute to the specialization attribute declarations:

```
<define name="thankyou.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="+ topic/ph thankyou-d/thankyou "/>
    </optional>
 </define>
```

parson

E) Prepare the Shell for the Specialization

1. Open *napsor-concept.rng*.
2. In the MODULE INCLUSIONS div, create a new div with an include for the thankyou element specialization:

```
<div>
    <a:documentation>SPECIALIZATION DOMAIN MODULES</a:documentation>
    <include href="napsor-thankyouDomain.rng"/>
</div>
```

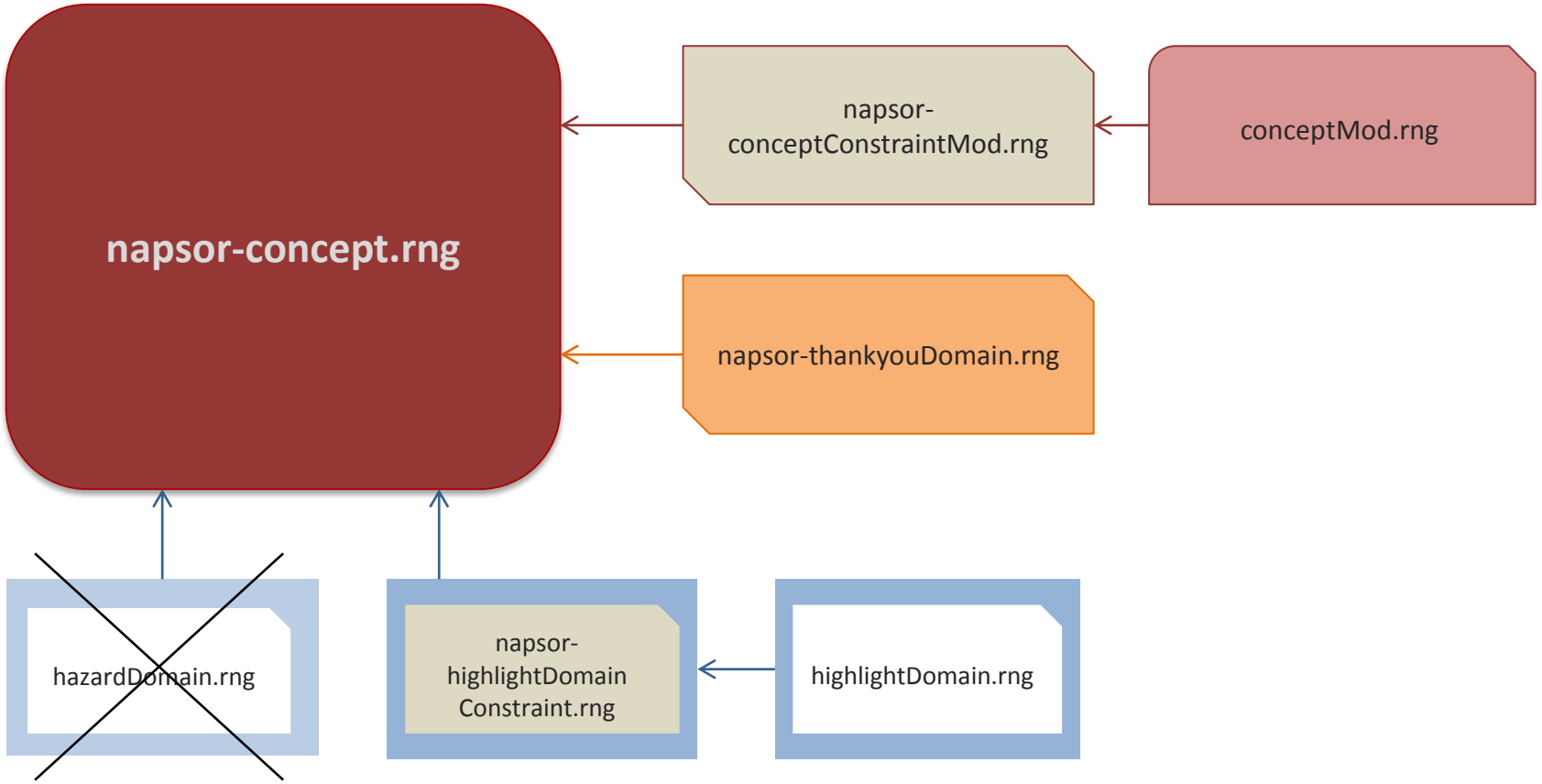
3. Add the module's short name to the domain contribution:

```
<attribute name="domains"
a:defaultValue="(topic abbrev-d) … (topic napsor-thankyou-d)"
```

4. Open *napsor-documentation.dita* and replace the thank you comment with a <thankyou> element.

parson

Napsor Customization – Result



parson

- Fundamentals
- DITA Architecture
- Customization - Introduction
- Configuration: Custom Concept Shell
- Constraint: Custom Topic Body and Domain Constraint
- Specialization: New Element
- Conclusion

parson

Tips & Tricks

- Do not touch any of the original files.
- Create a separate DITA-OT plug-in for your customization with new RNG files (shells) for your document types.
- Stick to the rules set by the coding requirements for RNG grammar modules.
- If in doubt, leave it out > start with fewer elements and expand if necessary.
- Get rid of mixed content.
- Use <notAllowed> to make elements or attributes generally unavailable.
- To make a base type unavailable, but keep a specialized version, use <notAllowed> on the base type element.element define, e.g. <define name="ph.element">
- Beware of side effects > always check where a definition might be reused. Use Search & Replace in all DITA-RNG folders.

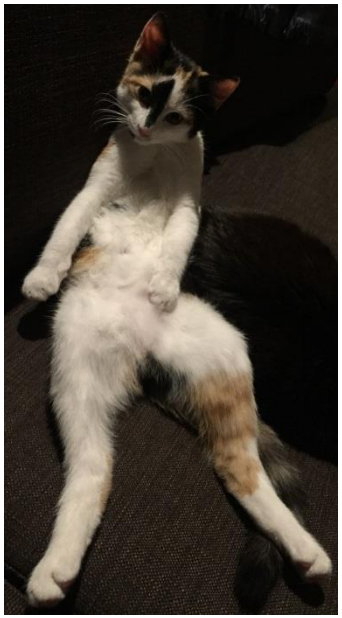
parson

References

- Official Relax NG website: http://relaxng.org
- Eric van der Vlist: *Relax NG* – http://books.xmlschemata.org/relaxng/page2.html
- David Mertz: "XML Matters: Kicking back with RELAX NG" (3 parts)
  - www.ibm.com/developerworks/xml/library/x-matters25/index.html
- DTD, XSD, or RELAX NG?
  - https://groups.yahoo.com/neo/groups/dita-users/conversations/topics/36666
- Using DITA-OT with RNG files?
  - https://groups.google.com/forum/#!topic/dita-ot-users/0XZAloq2cOY
  - https://code.google.com/p/dita-ng/source/browse/trunk/doc/DITA-NG.pdf
- DITA 1.3 Specification:
  - Configuration, specialization, generalization, and constraints
  - RELAX NG coding requirements

parson

Questions or Comments?



Frank Ralf

parson AG
Chrysanderstr. 69A
21029 Hamburg

+49 (0)40 7200 500-0
contact@parson-europe.com
www.parson-europe.com