# DITA Perspectives

# Contents

**Chapter 1. Abstract** ..... **4**

**Chapter 2. Overview of DITA** ..... **5**

DITA shells ..... 5

DITA modules ..... 5

DITA elements ..... 7

bookmap ..... 11

learningSummary ..... 11

learningPlan ..... 12

learningOverview ..... 12

learningContent ..... 13

learningObjectMap ..... 13

learningBase ..... 13

learningGroupMap ..... 14

learningAssessment ..... 14

troubleshooting ..... 14

reference ..... 14

task ..... 15

glossentry ..... 16

glossgroup ..... 17

concept ..... 17

subjectScheme ..... 17

DITA domains ..... 18

DITA Learning Interaction Base 2 Domain ..... 18

DITA Learning Map Domain ..... 19

DITA Learning Interaction Base Domain ..... 19

DITA Learning Metadata Domain ..... 20

DITA Learning Domain ..... 21

DITA Learning 2 Domain.....22
DITA Abbreviated Form Domain.....22
DITA Markup Name Mention Domain.....23
DITA MathML Domain.....23
DITA SVG Domain.....23
DITA User Interface Domain.....23
DITA Equation Domain.....23
DITA Task Requirements Domain.....24
DITA Programming Domain.....25
DITA XML Construct Domain.....25
DITA Glossary Reference Domain.....25
DITA Software Domain.....26
DITA Release Management Domain.....26
DITA XNAL Domain.....27
DITAVAL Reference Domain.....27
DITA Delay Resolution Domain.....27
DITA Indexing Domain.....28
DITA Hazard Statement Domain.....28
DITA Highlight Domain.....28
DITA Map Group Domain.....28
DITA Utilities Domain.....29
DITA Subject Classification Domain.....29
**Chapter 3. Taking advantage of DITA elements hierarchy.....30**
Element selection during editing.....30
Automatic markup detection.....30
Soft generalization.....31
**Chapter 4. Exchanging DITA documents.....33**

# Chapter 1. Abstract

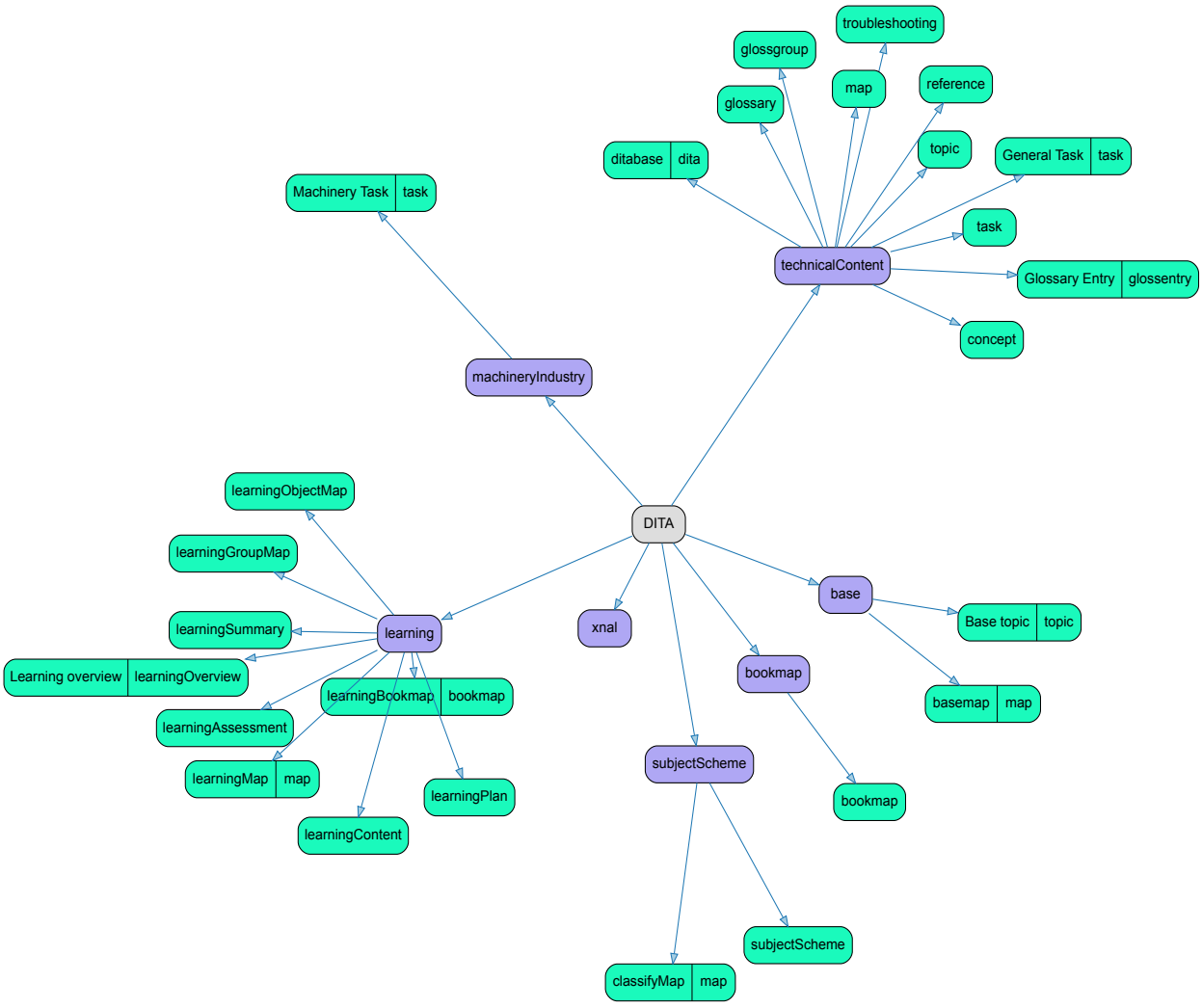
DITA is not defined as a flat list of elements, but each element is either a base element or it derives as a specialized version of another element. This hierarchy should actually decrease the cognitive complexity of a vocabulary because it allows you to find an element faster than working with a flat list.

In this presentation we want to show the hierarchy of elements in DITA and then explore how we may take advantage of this in understanding the DITA architecture, learning DITA, document authoring, etc.

# Chapter 2. Overview of DITA

## DITA shells

DITA 1.3 is defines multiple types of documents, along with the generic topic and map there are also many specialized topics types and maps types. Each type of document is defined by a schema that is marked as a `shell` schema, a schema that should be an entry point, the one that should be referred from an XML document. All the defined shells are presented in the following diagram, grouped by the folder they are defined in:




## DITA modules

The DITA specification mentions that each element has a `class` attribute that encodes information about the type of that element, for example:

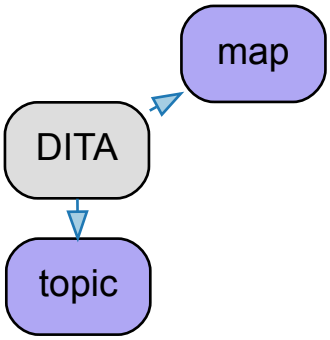
```
<step class="- topic/li task/step">
```

and this information includes:

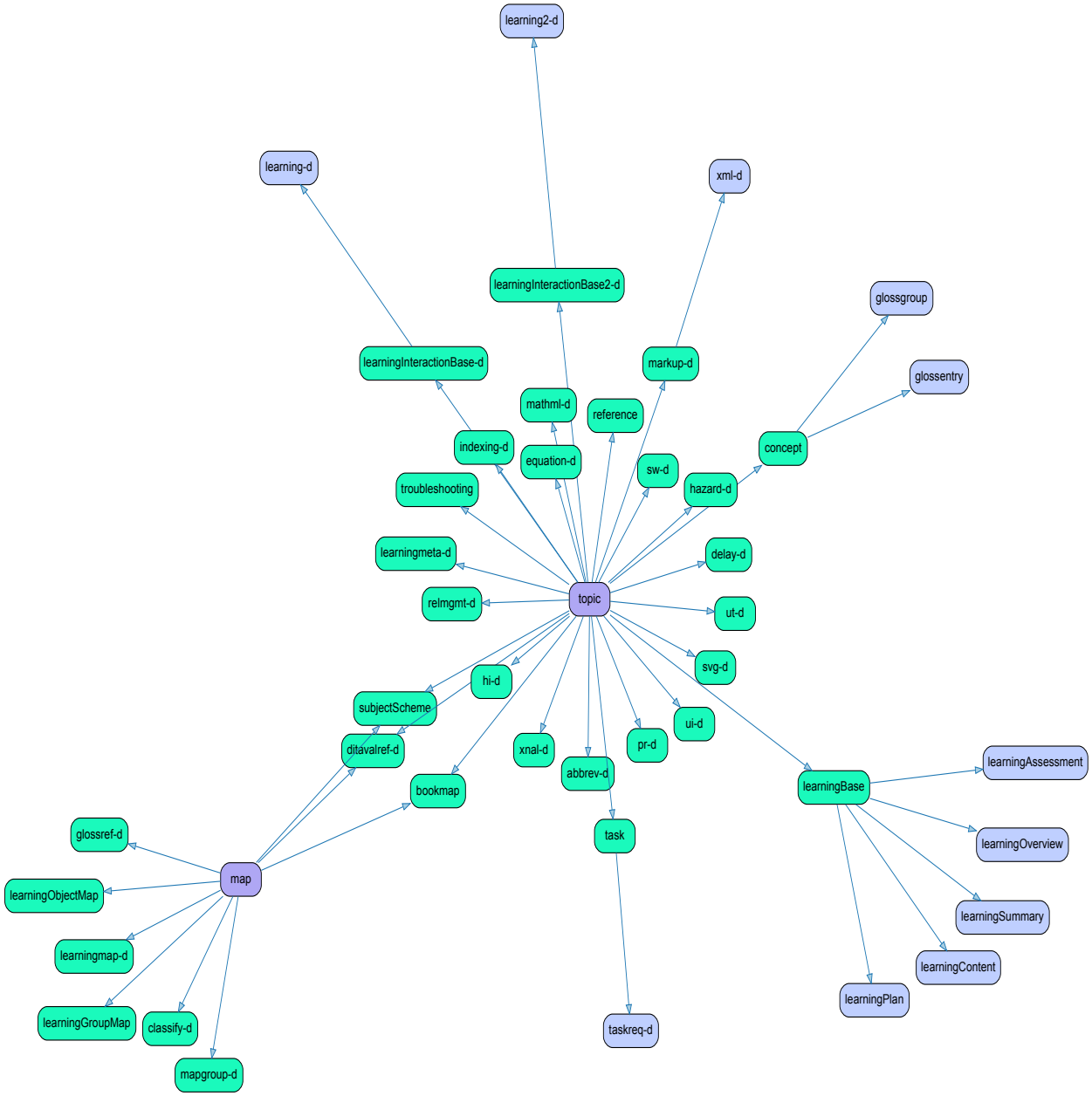
> "A sequence of one or more tokens of the form `"modulename/typename"`, with each token separated by one or more spaces, where `modulename` is the short name of the vocabulary module and `typename` is the element type name. Tokens are ordered left to right from most general to most specialized."

By analyzing all the class values from all the elements defined in the schemas we can identify the base modules (top level ones), in this case `topic` and a hierarchy implied by how the elements are defined, by this relation from more general to more specialized modules, thus identifying how modules are specialized from others, in this example `task` is specialized from `topic`.

The base DITA modules:

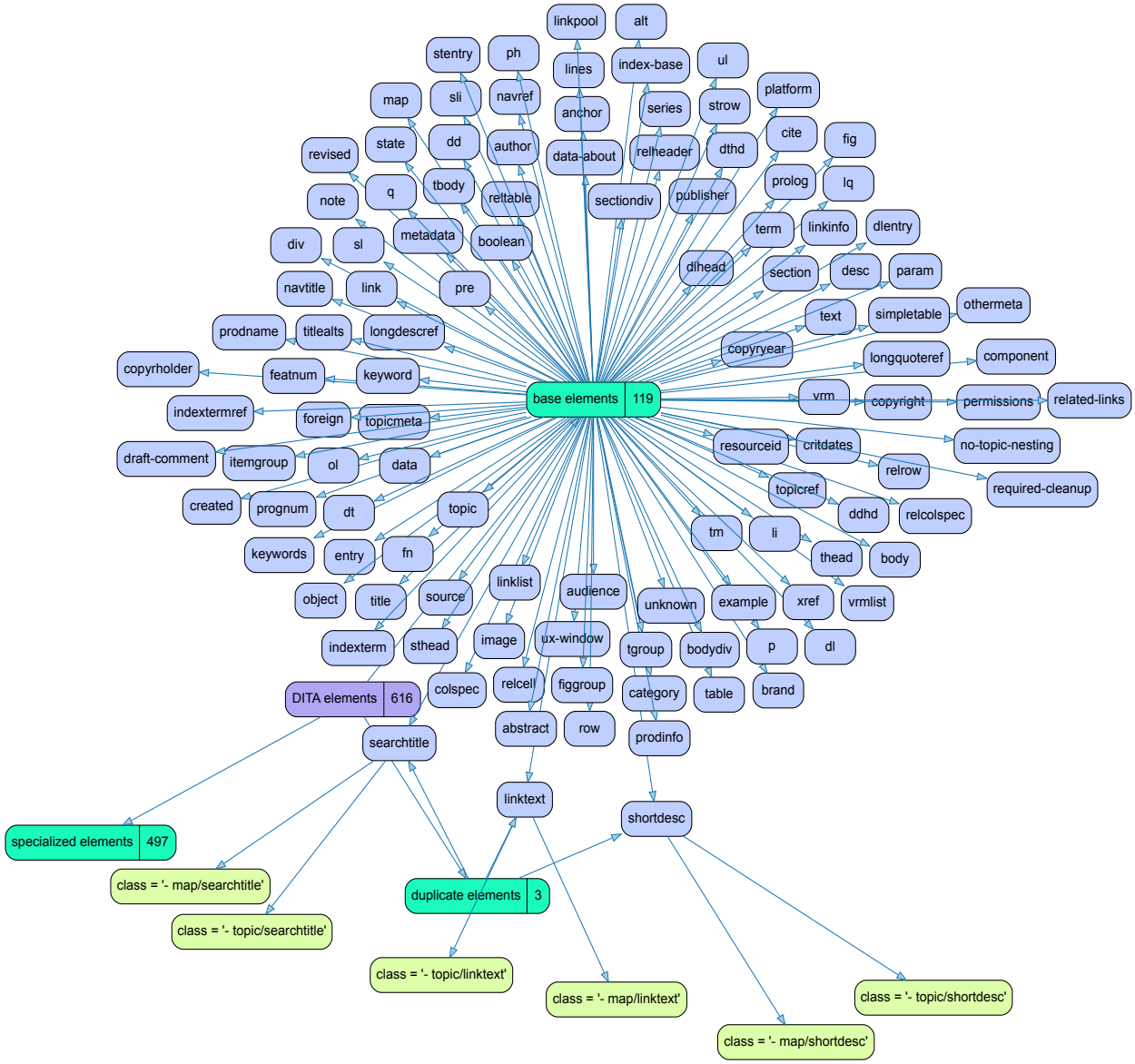



The modules specialization hierarchy:

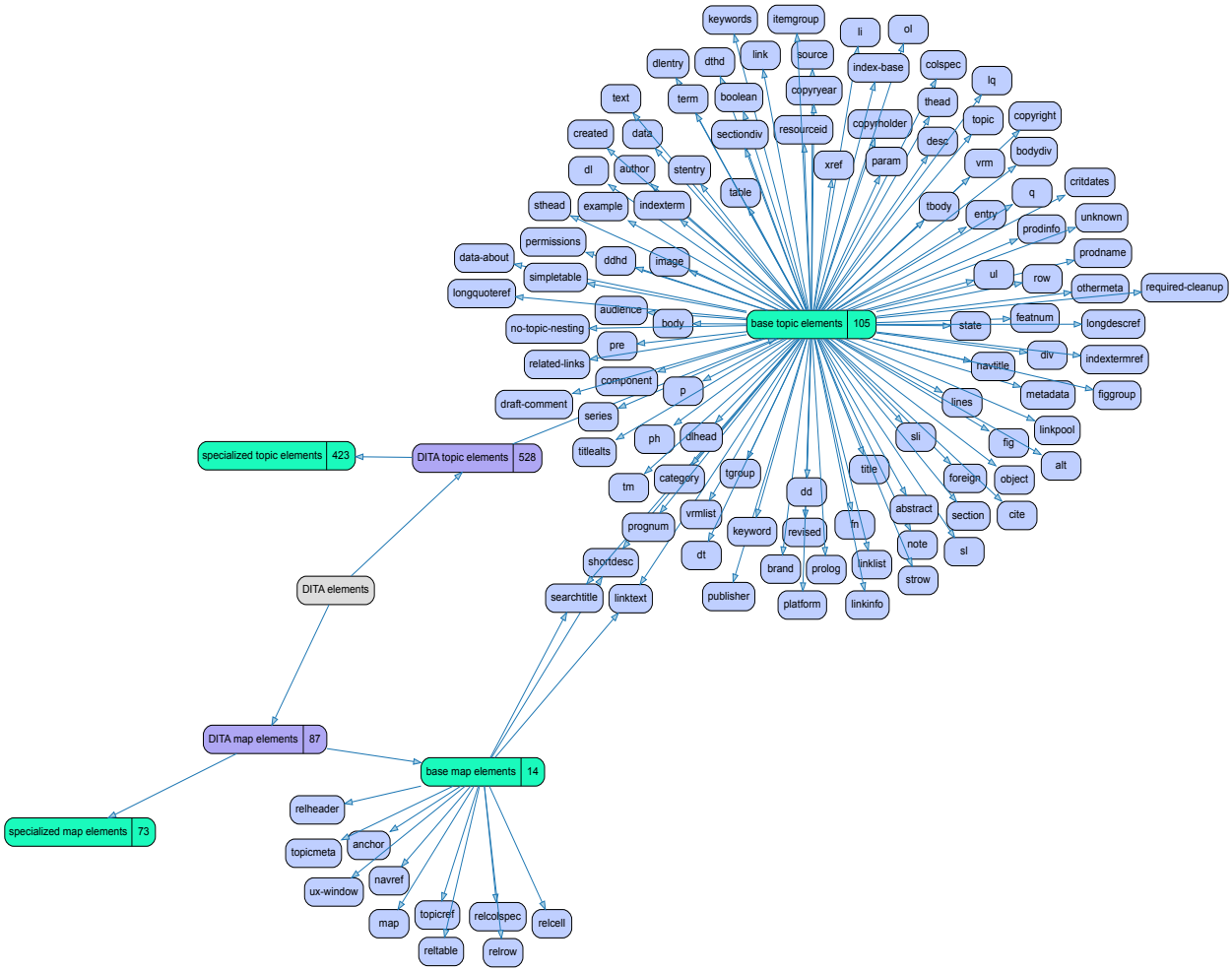



# DITA elements

The total number of DITA elements defined in the schemas and how they are split into base elements, specialized element and a highlight of the duplicate elements, having the same name but being defined on a different module can be observed here:

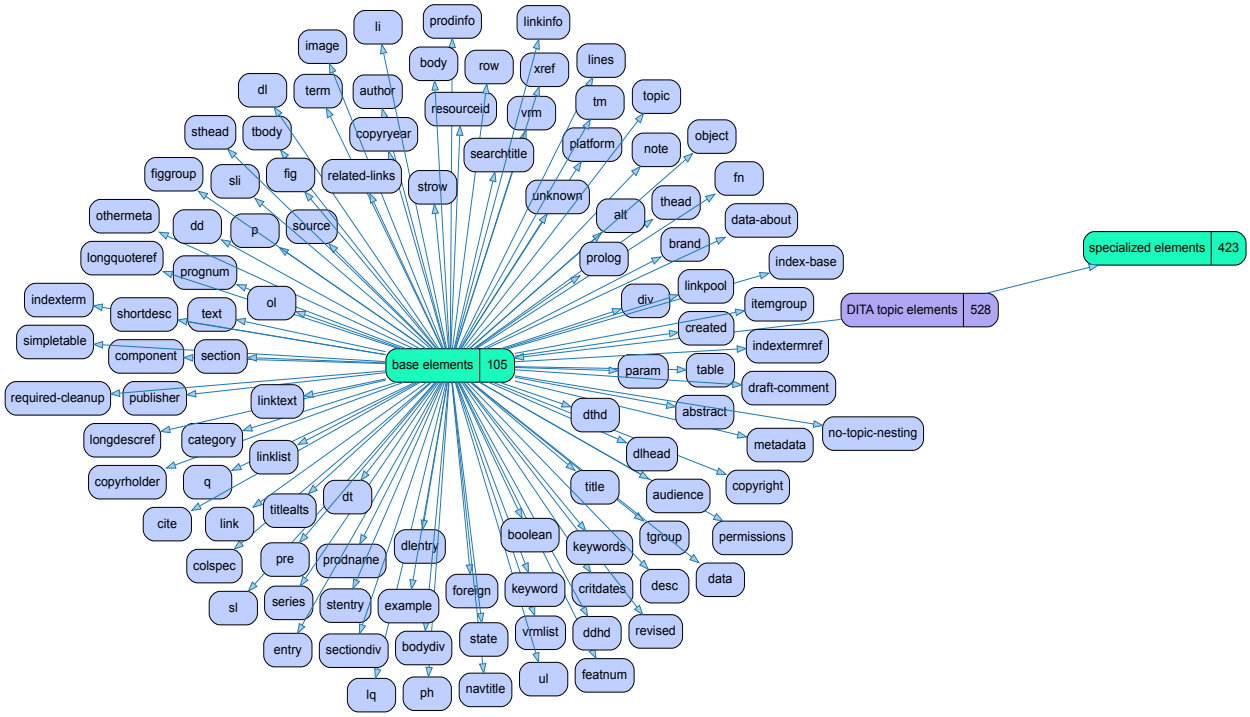

base elements 119
DITA elements 616
specialized elements 497
duplicate elements 3
class = '- map/searchtitle'
class = '- topic/searchtitle'
class = '- topic/linktext'
class = '- map/linktext'
class = '- map/shortdesc'
class = '- topic/shortdesc'
linkpool
alt
stentry
ph
lines
index-base
ul
map
sli
navref
anchor
series
strow
platform
state
dd
author
data-about
relheader
dthd
cite
fig
revised
tbody
reltable
sectiondiv
publisher
prolog
lq
note
q
metadata
boolean
term
linkinfo
dlentry
div
sl
dlhead
section
desc
param
navtitle
link
pre
text
simpletable
othermeta
prodname
titlealts
longdescref
copyryear
longquoteref
component
copyrholder
featnum
keyword
vrm
copyright
permissions
related-links
indextermref
foreign
topicmeta
resourceid
critdates
no-topic-nesting
draft-comment
itemgroup
ol
data
relrow
topicref
required-cleanup
created
prognum
dt
topic
ddhd
relcolspec
tm
li
keywords
entry
fn
thead
body
object
title
source
linklist
audience
unknown
example
xref
vrmlist
indexterm
sthead
image
ux-window
tgroup
bodydiv
p
dl
colspec
relcell
figgroup
category
table
brand
abstract
row
prodinfo
searchtitle
linktext
shortdesc

## Elements split by map and topic base



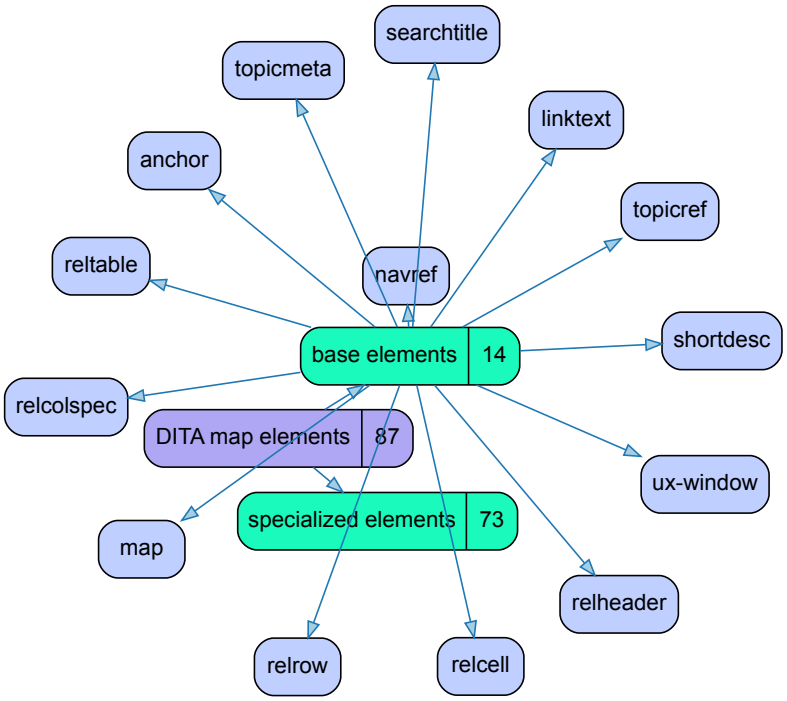

## Topic elements

We project the element information on the `topic` base module:



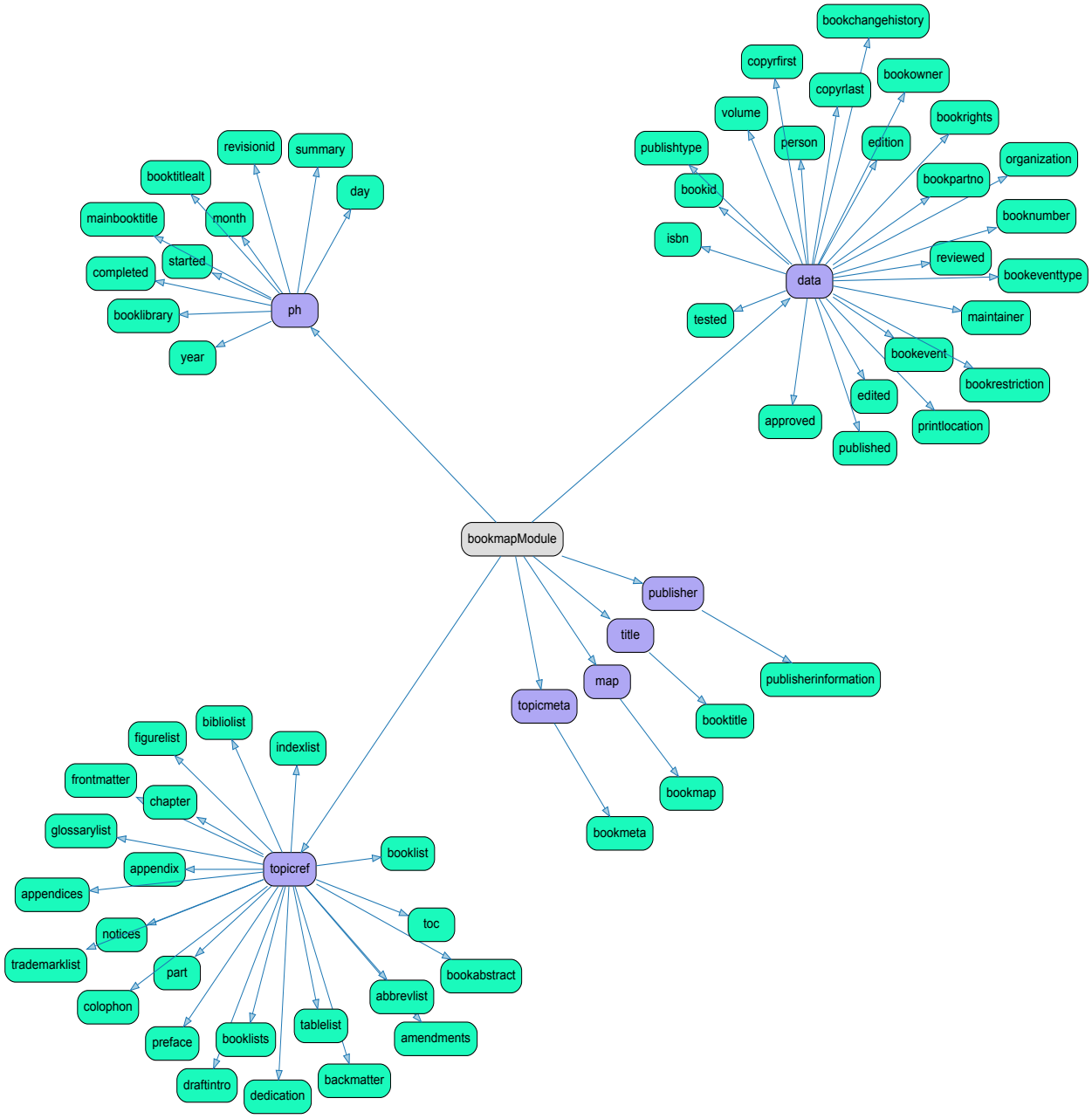

## Map elements

We project the element information on the `map` base module:



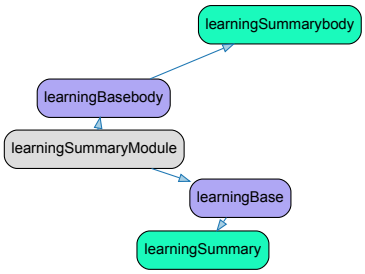
# bookmap

Defined in ../data/DITA1.3/rng/bookmap/rng/bookmapMod.rng



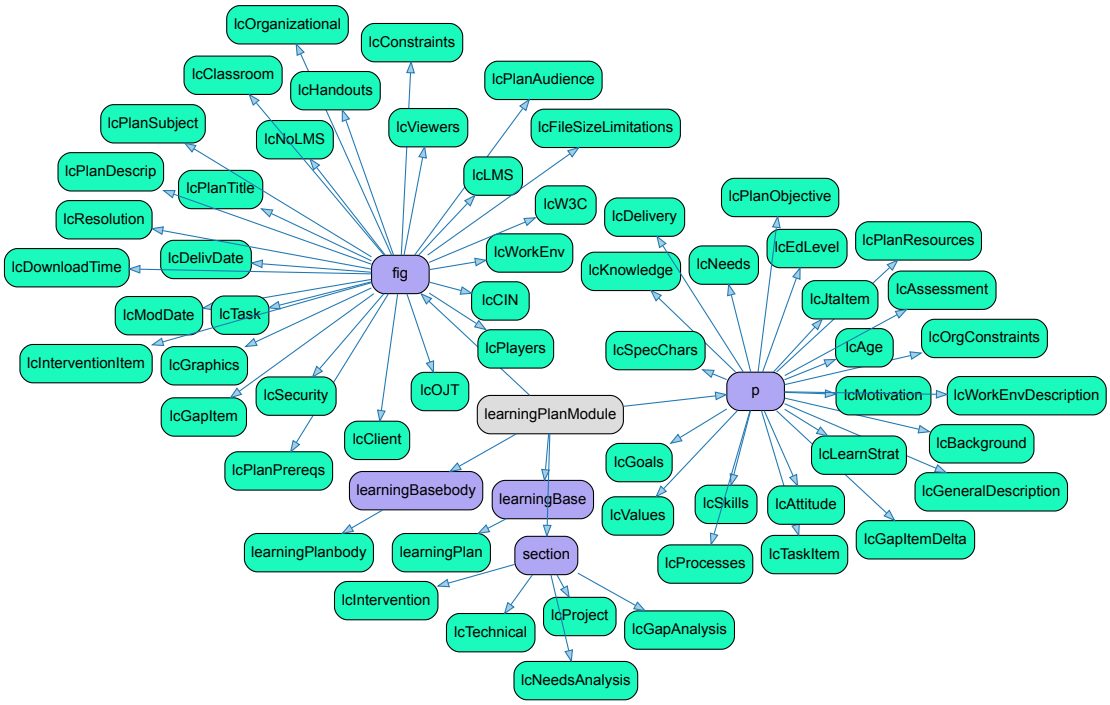

# learningSummary

Defined in ../data/DITA1.3/rng/learning/rng/learningSummaryMod.rng



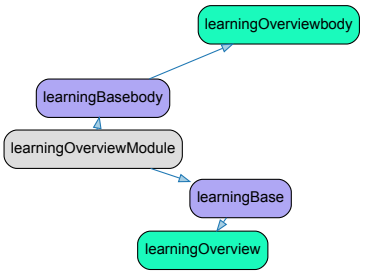

# learningPlan

Defined in ../data/DITA1.3/rng/learning/rng/learningPlanMod.rng



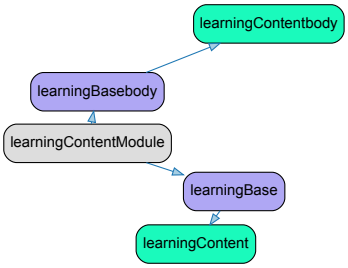

# learningOverview

Defined in ../data/DITA1.3/rng/learning/rng/learningOverviewMod.rng



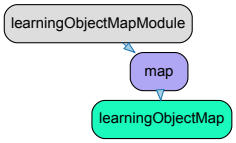
# learningContent

Defined in ../data/DITA1.3/rng/learning/rng/learningContentMod.rng



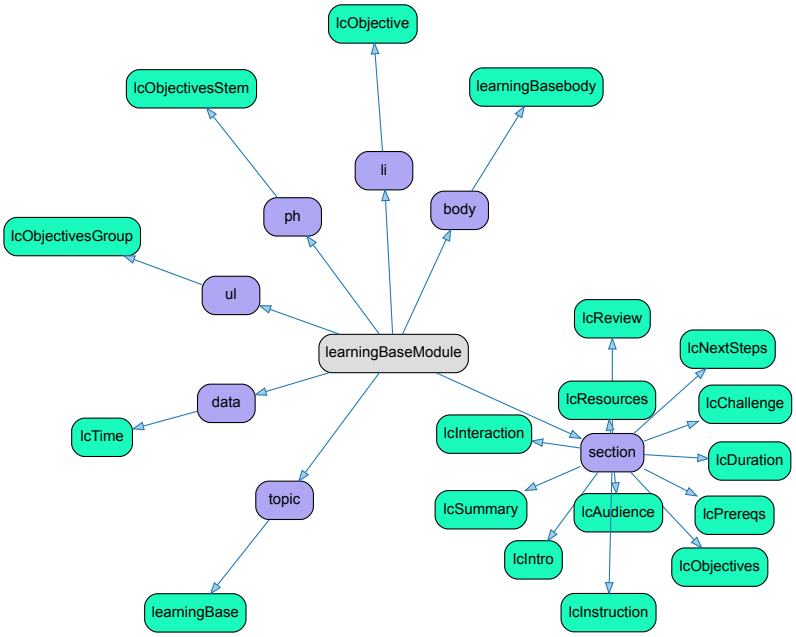

# learningObjectMap

Defined in ../data/DITA1.3/rng/learning/rng/learningObjectMapMod.rng



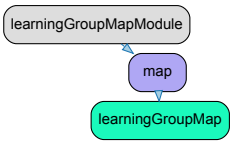

# learningBase

Defined in ../data/DITA1.3/rng/learning/rng/learningBaseMod.rng



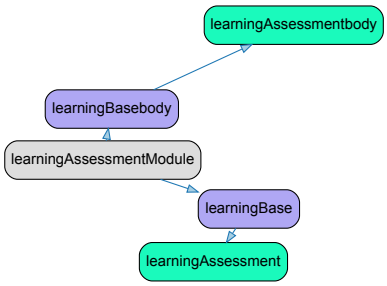
# learningGroupMap

Defined in ../data/DITA1.3/rng/learning/rng/learningGroupMapMod.rng



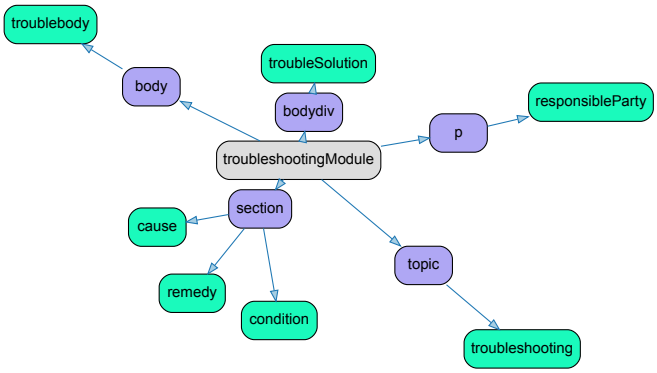

# learningAssessment

Defined in ../data/DITA1.3/rng/learning/rng/learningAssessmentMod.rng



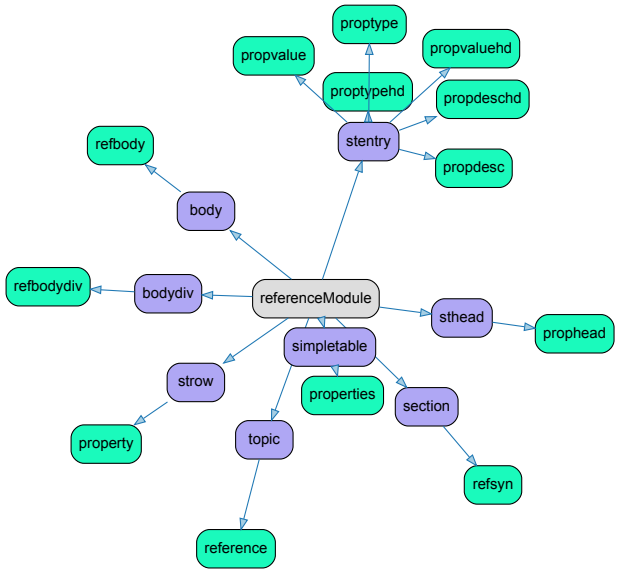

# troubleshooting

Defined in ../data/DITA1.3/rng/technicalContent/rng/troubleshootingMod.rng



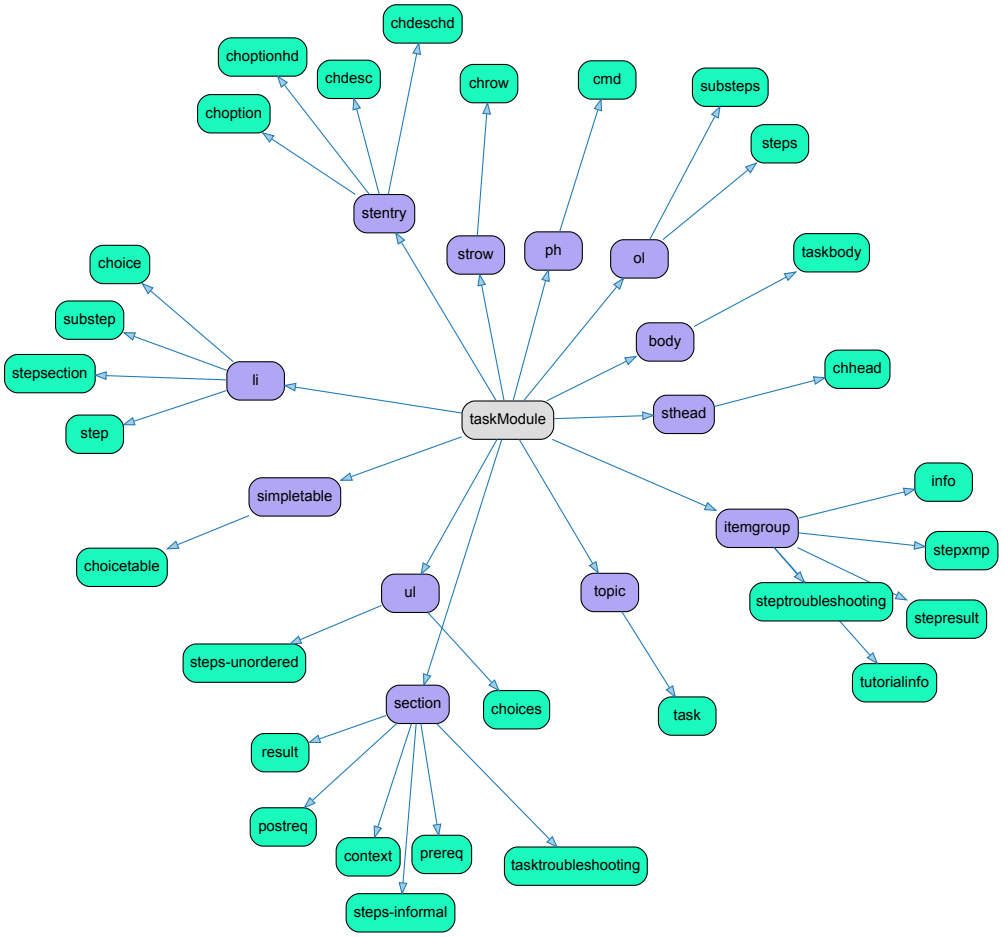

# reference

Defined in ../data/DITA1.3/rng/technicalContent/rng/referenceMod.rng



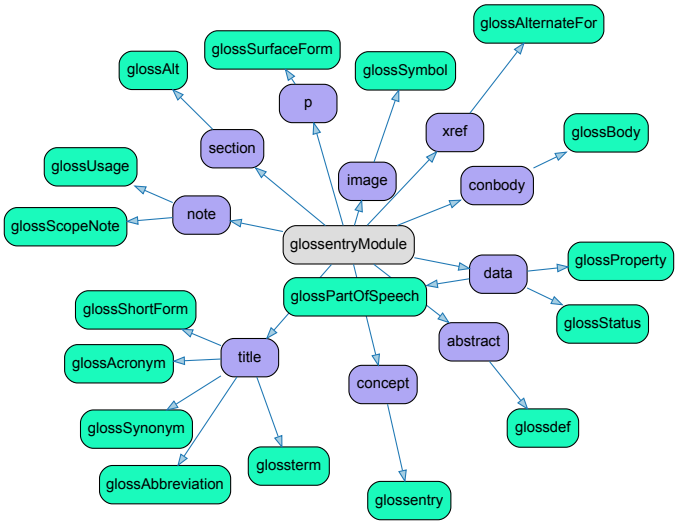

# task

Defined in ../data/DITA1.3/rng/technicalContent/rng/taskMod.rng



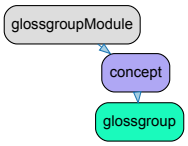

# glossentry

Defined in ../data/DITA1.3/rng/technicalContent/rng/glossentryMod.rng



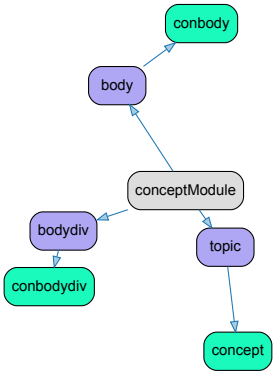
# glossgroup

Defined in ../data/DITA1.3/rng/technicalContent/rng/glossgroupMod.rng



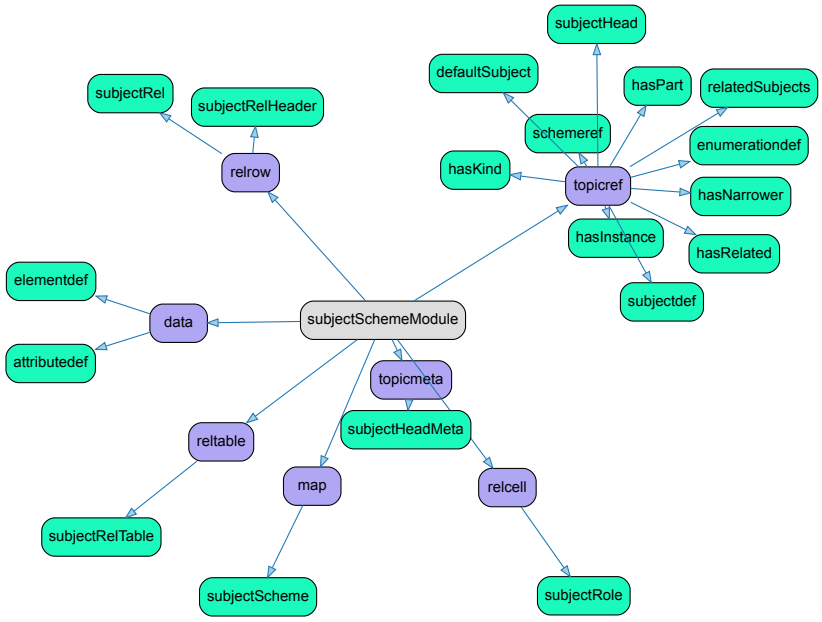

# concept

Defined in ../data/DITA1.3/rng/technicalContent/rng/conceptMod.rng




# subjectScheme

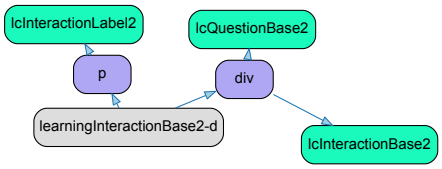
Defined in ../data/DITA1.3/rng/subjectScheme/rng/subjectSchemeMod.rng



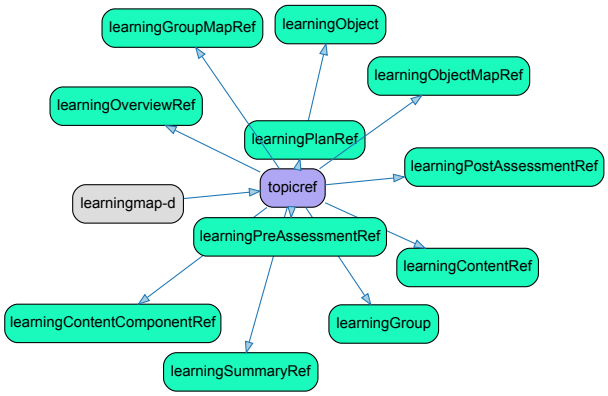

# DITA domains

Some DITA elements are defined to be part of a domain that can be added on any DITA document type as a pluggable component. A domain defines a number of semantic elements that are derived from other elements, providing a specialization of the base elements but reflecting the semantics of a domain like programming, software, etc.

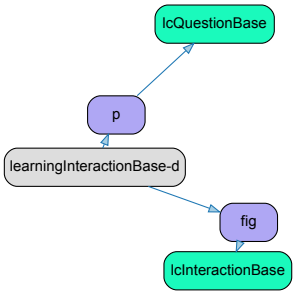
## DITA Learning Interaction Base 2 Domain



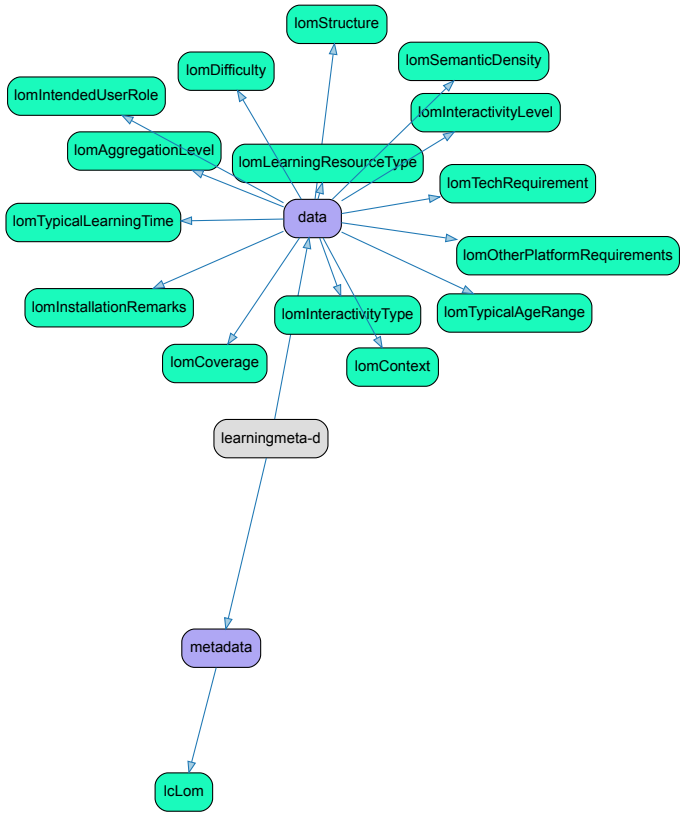
## DITA Learning Map Domain



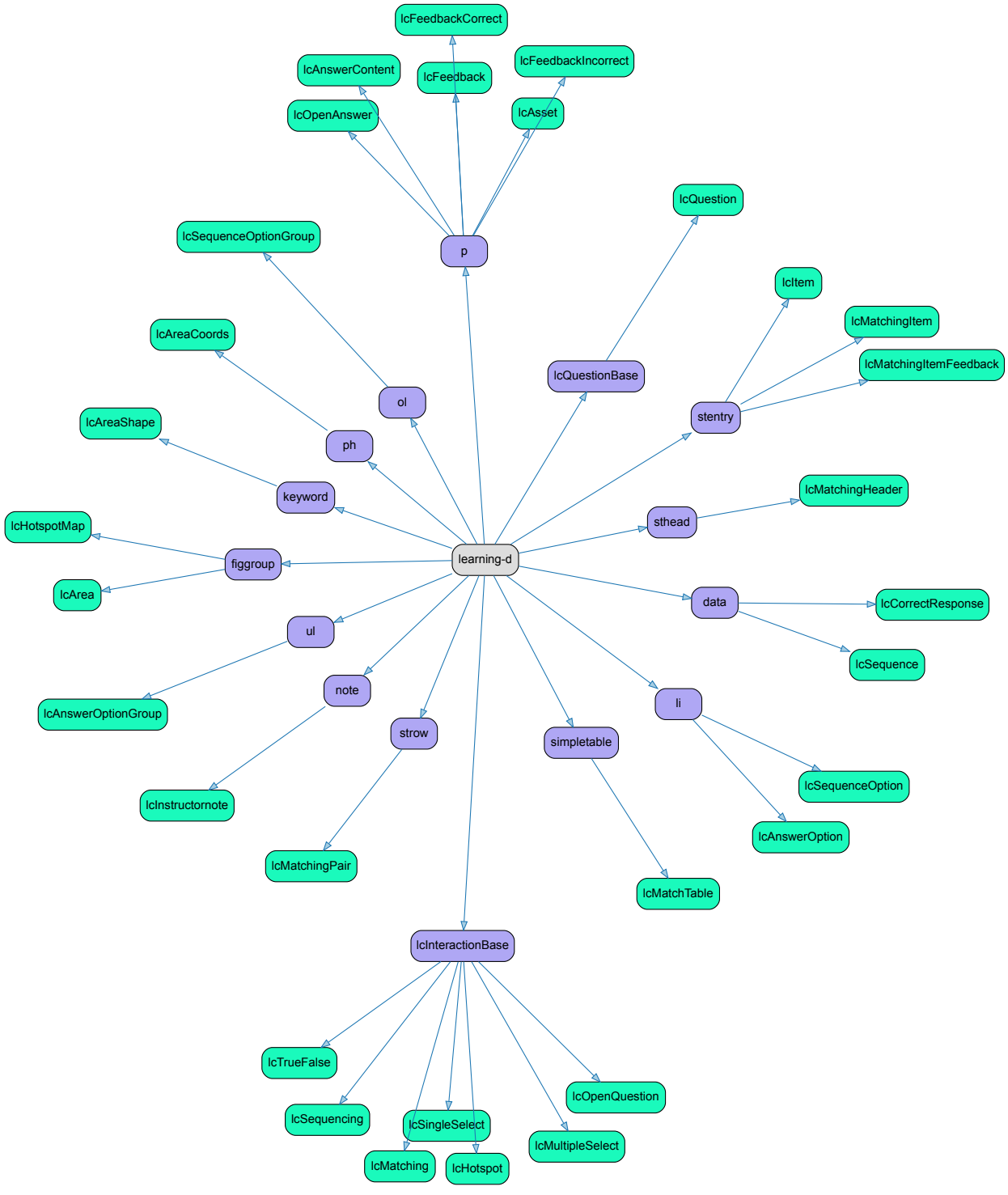

## DITA Learning Interaction Base Domain



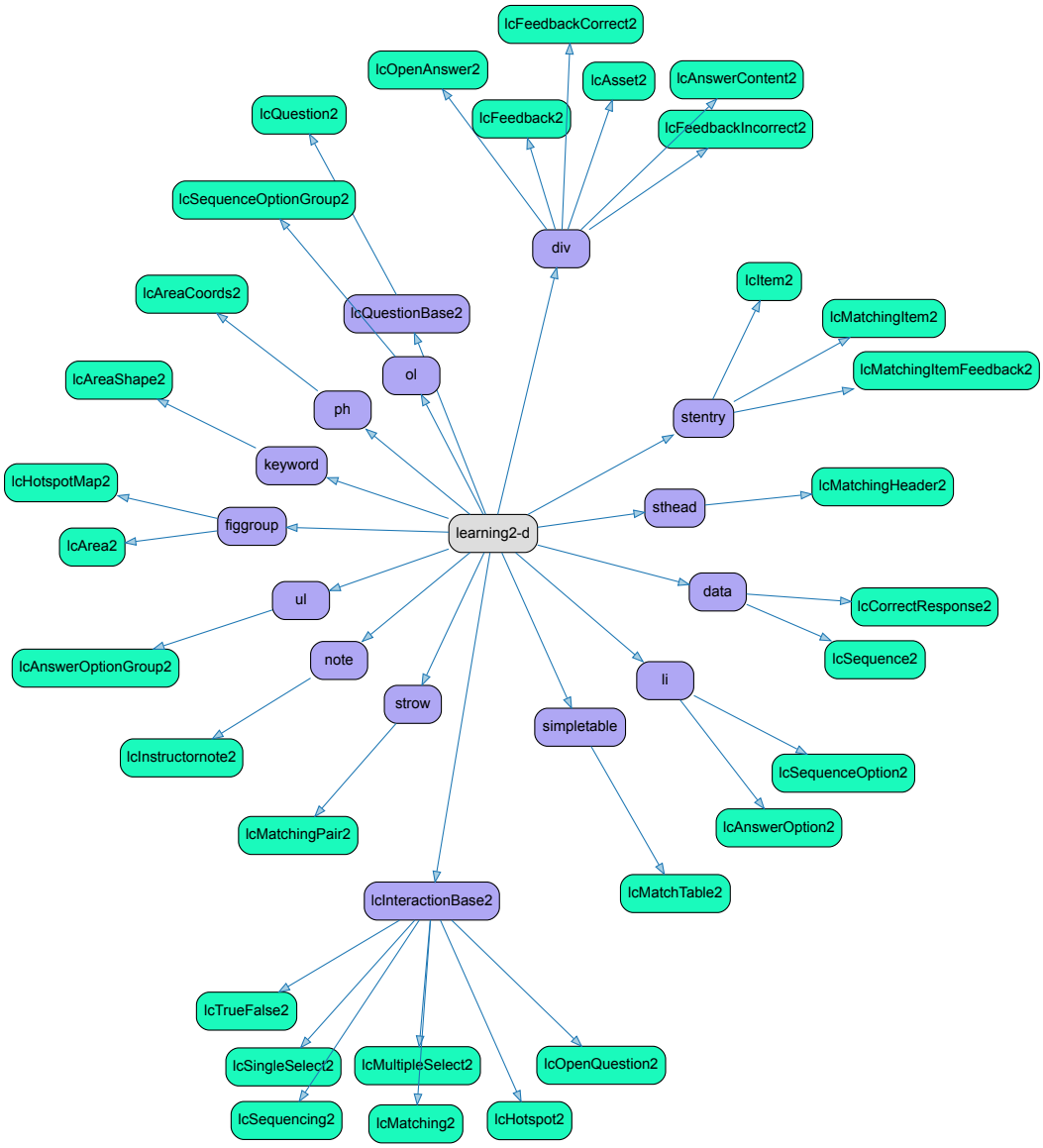
# DITA Learning Metadata Domain



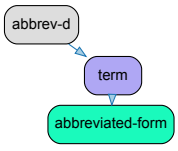
# DITA Learning Domain



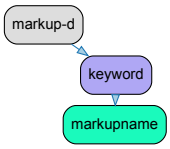
# DITA Learning 2 Domain



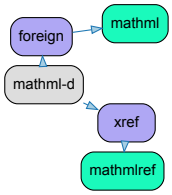

# DITA Abbreviated Form Domain



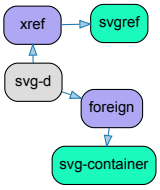
## DITA Markup Name Mention Domain



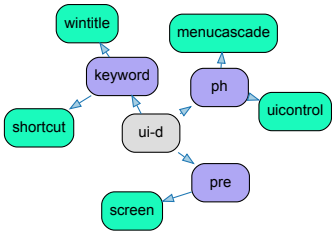

## DITA MathML Domain



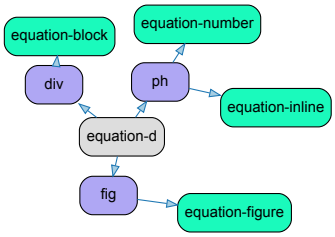

## DITA SVG Domain



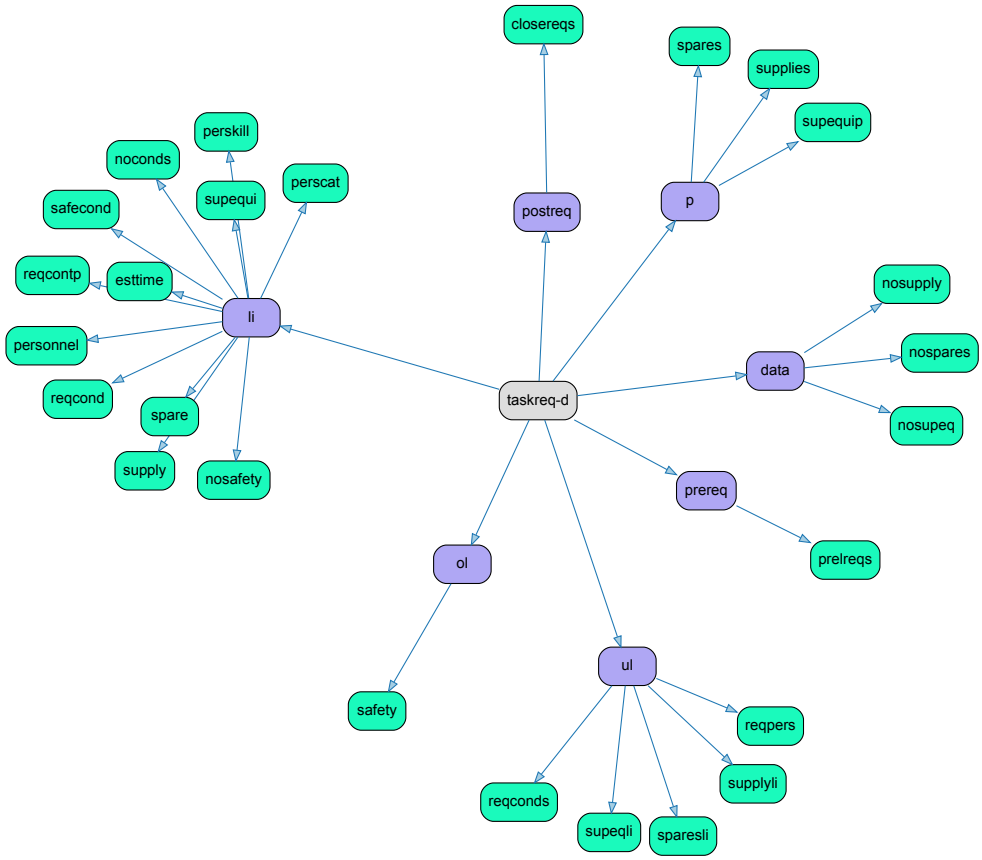

## DITA User Interface Domain



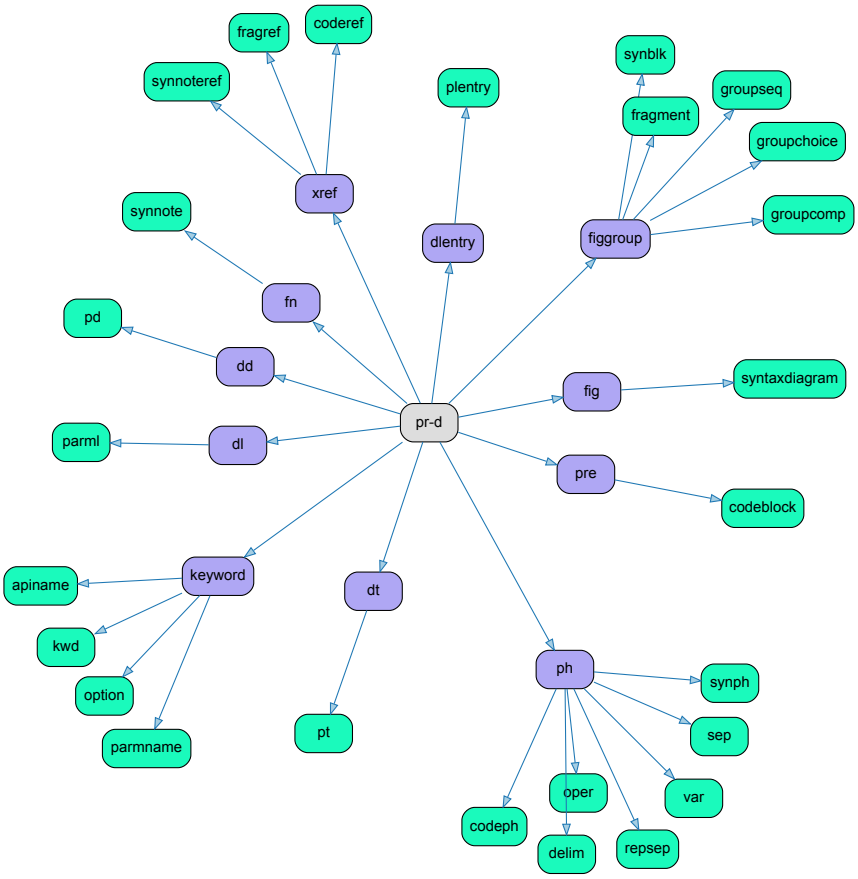

## DITA Equation Domain



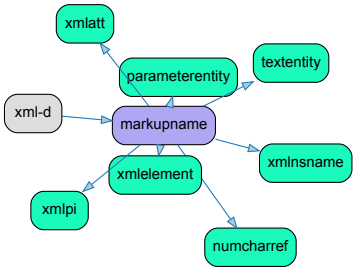
# DITA Task Requirements Domain



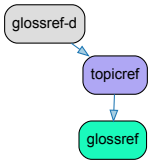
## DITA Programming Domain



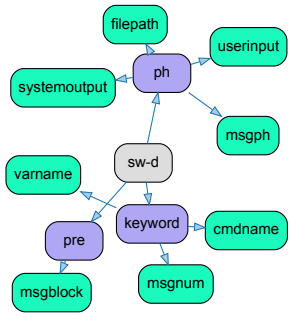

## DITA XML Construct Domain



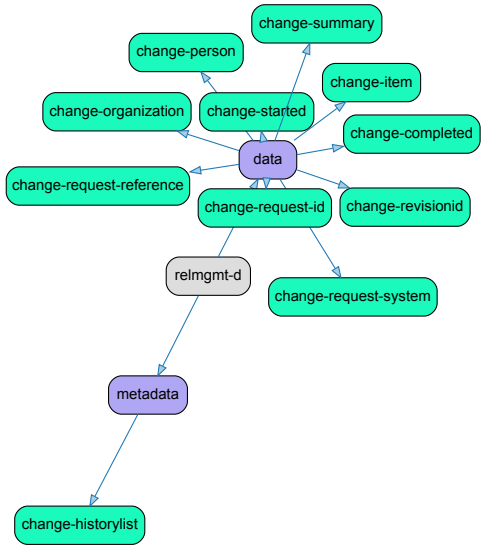

## DITA Glossary Reference Domain



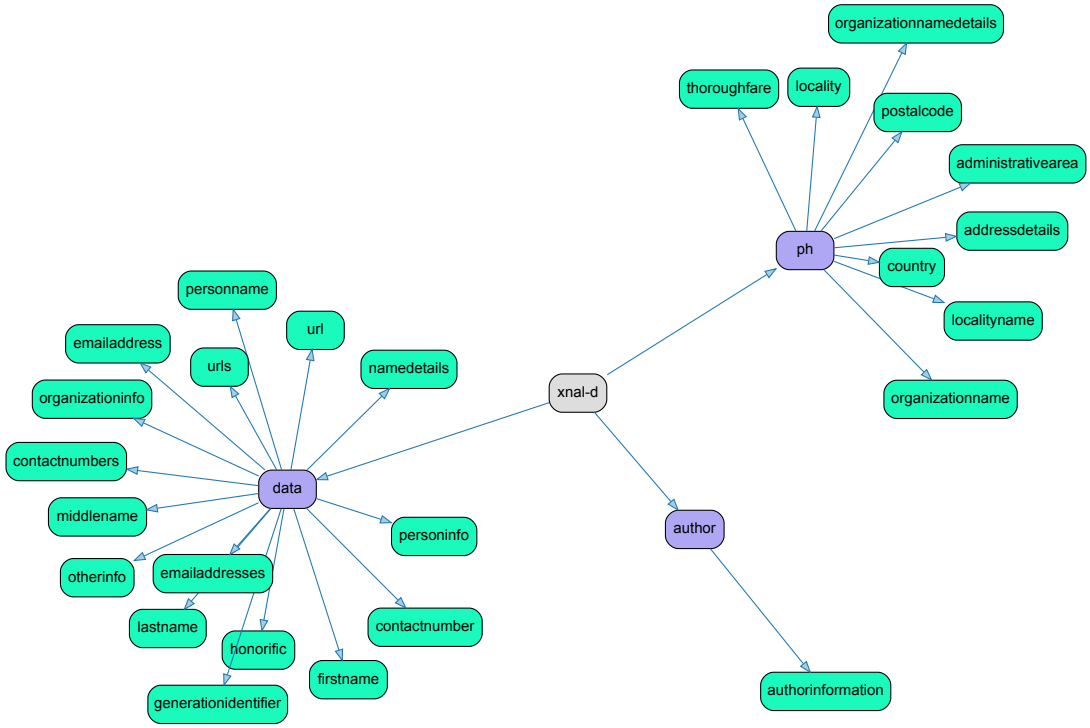
## DITA Software Domain



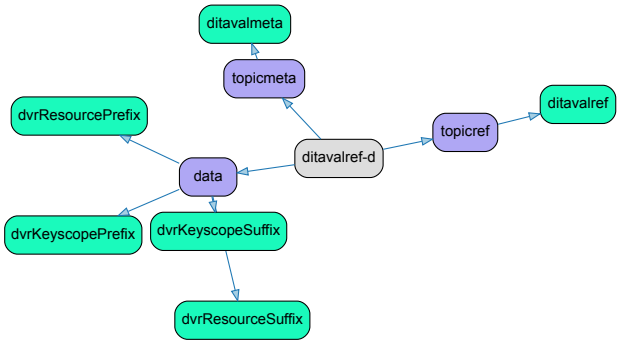

## DITA Release Management Domain



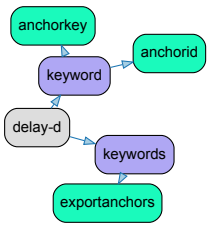
## DITA XNAL Domain



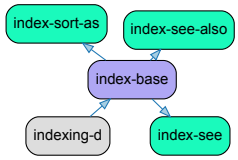

## DITAVAL Reference Domain



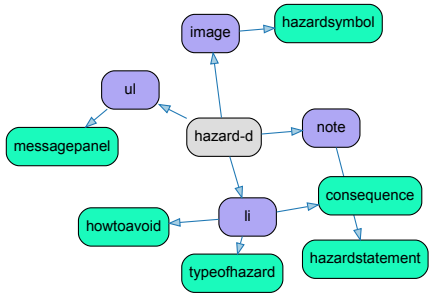

## DITA Delay Resolution Domain



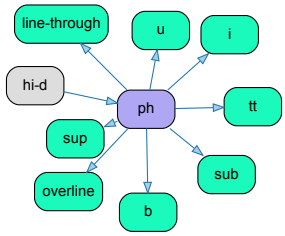
## DITA Indexing Domain



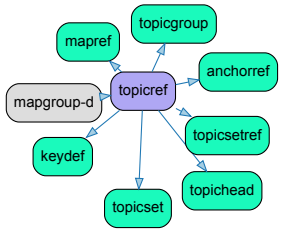

## DITA Hazard Statement Domain



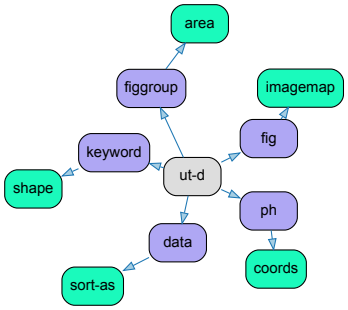

## DITA Highlight Domain



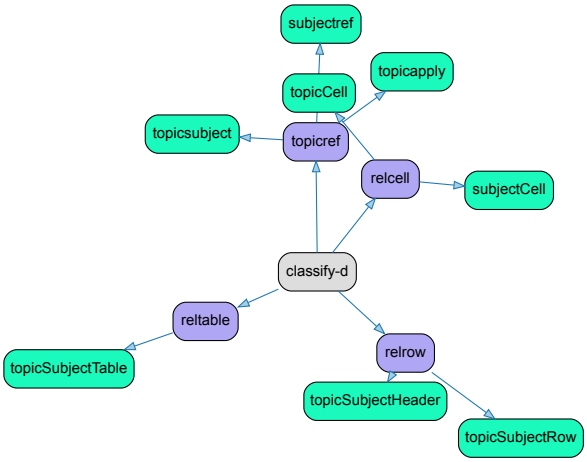

## DITA Map Group Domain



## DITA Utilities Domain




## DITA Subject Classification Domain



# Chapter 3. Taking advantage of DITA elements hierarchy

Elements in DITA are defined not as a flat list of elements, but instead DITA defines a hierarchy, similar to a type hierarchy, and elements are either base elements or they are derived from another element. If we look at the analogy with a type system, the base elements are similar to the primitive types and the other elements are equivalent to derived types.

This is realized using values specified in the **class** attribute. These values specify the category and the name of the current element and, if we talk about a derived element, also the name and category of its parent, as well as all the ancestors up to the base element.

For example the `@class` value for the `<lcPlanTitle>` element is:

```
class="- topic/fig learningBase/fig learningPlan/lcPlanTitle"
```

This means that the current element is `<lcPlanTitle>` from the `learningPlan` category and this is derived from the `<fig>` element from the `learningBase` category, which in turn is defined from the `<fig>` element from the `topic` category.

We may take this information into account in some situations.

## Element selection during editing

One possibility to take advantage of the fact that elements in DITA form a hierarchy is to follow this hierarchy when we present the user to choose an element to insert, so instead of presenting a flat list of choices we may organize the elements according to the hierarchy defined by the `@class` attribute values and thus if the user will select for example a `<ul>` to insert an unordered list we can present further all the elements specialized from `<ul>` that are valid in that position in the document. Of course, this makes sense if the user is browsing for an element to insert, if the user already knows the element and types its name then we can just filter on that name and eventually if there are elements specializing the one identified by the entered string then we can further show them.

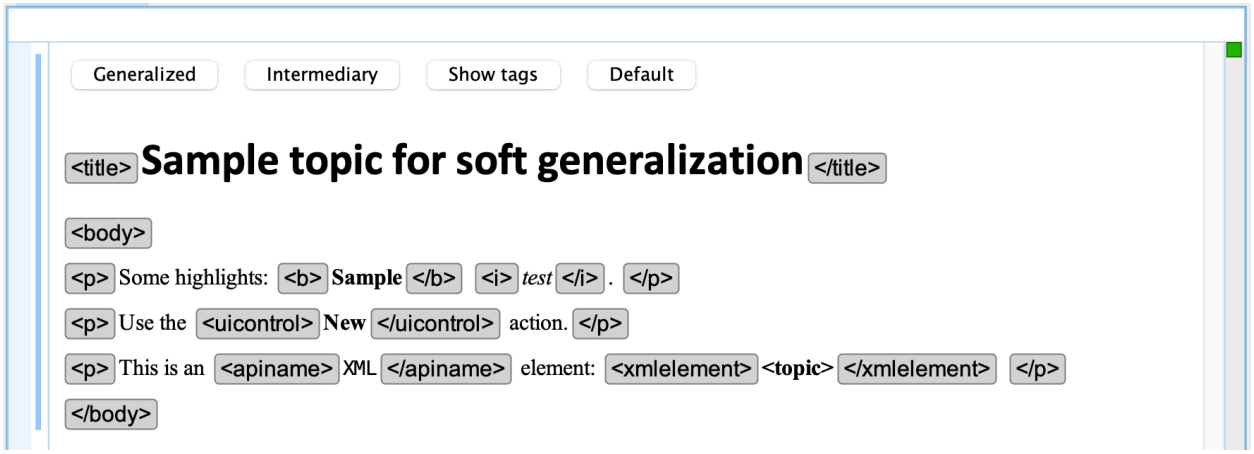
Another possibility is to present a drop down for an element that has specialized elements and those are valid at that location and allow the user to move to a more specialized element by selecting it from that drops down.

## Automatic markup detection

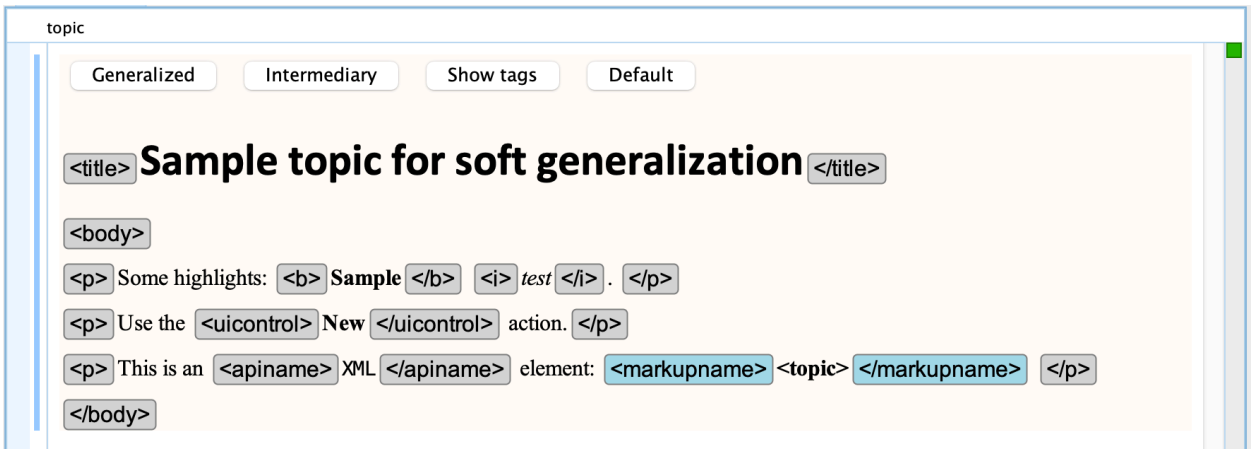
# Soft generalization

To visualize how a document will look like when it is generalized but without really generalizing it, that is without replacing the elements with more general ones, we implemented a CSS rendering to show the XML tags as if the document is generalized.

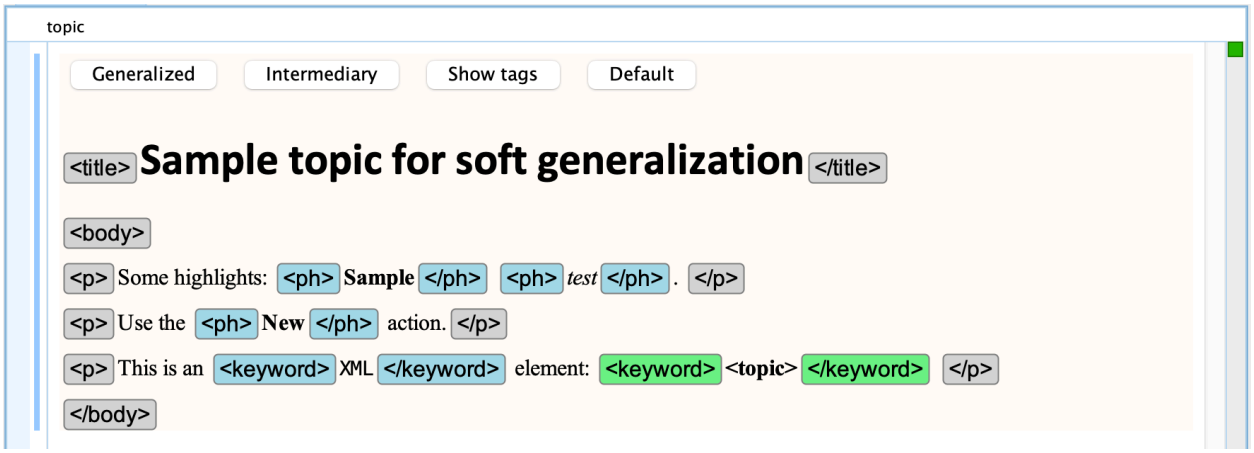
For example, an element on which we make the tags visible using CSS static content placed before and after the element looks like:




If we generalize on level, showing the first derived element type than in this sample `<xmlelement>` will turn into `<markupname>`:




If we generalize to show the base elements then many of the tag names will change, as highlighted in the following screen shot:


topic
Generalized
Intermediary
Show tags
Default
<title> Sample topic for soft generalization </title>
<body>
 Some highlights: <ph> Sample </ph> <ph> test </ph> . 
 Use the <ph> New </ph> action. 
 This is an <keyword> XML </keyword> element: <keyword> <topic> </keyword> 
</body>

# Chapter 4. Exchanging DITA documents

One advantage of using DITA is that we can tag at semantic level. For this to work, we need to create a specialization that reflects exactly the concepts the users of that specialization are familiar with. If one tries to layer its actual needs over an existing specialization - or one of the standard schemas - these may not match exactly the concepts of the audience and thus it will look like DITA is not the best fit.

But the problem with a specialization is how do we exchange it with other parties? We need to provide them also our specialization, and if they do not have our specialization, then what happens?

One approach will be to export the DITA content as generalized content. Then import such a generalized package into an existing DITA installation by specializing as much as possible taking into account what is available on the target DITA installation.

How can this be achieved?

From the class of the root element we can identify the possible specialized root elements and check if we have them defined in a schema in the target DITA installation. Once we choose a root element then we need to analyze the domain attributes of the root element and see what domains are defined in the target topic type and then specialize elements from those domains back, otherwise, if a domain is not present in the target schema we need to leave the base element.

We have support for generalization - and that is relatively easy to implement - and we can probably obtain also the export package in a generalized form. We are missing however a tool to take a generalized package, analyze a current DITA installation and specialize everything so that it matches what is available in that specific DITA installation.