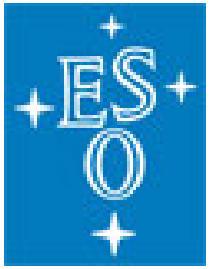


Model Based Document Generation

Michele Zamparelli

Robert Karban

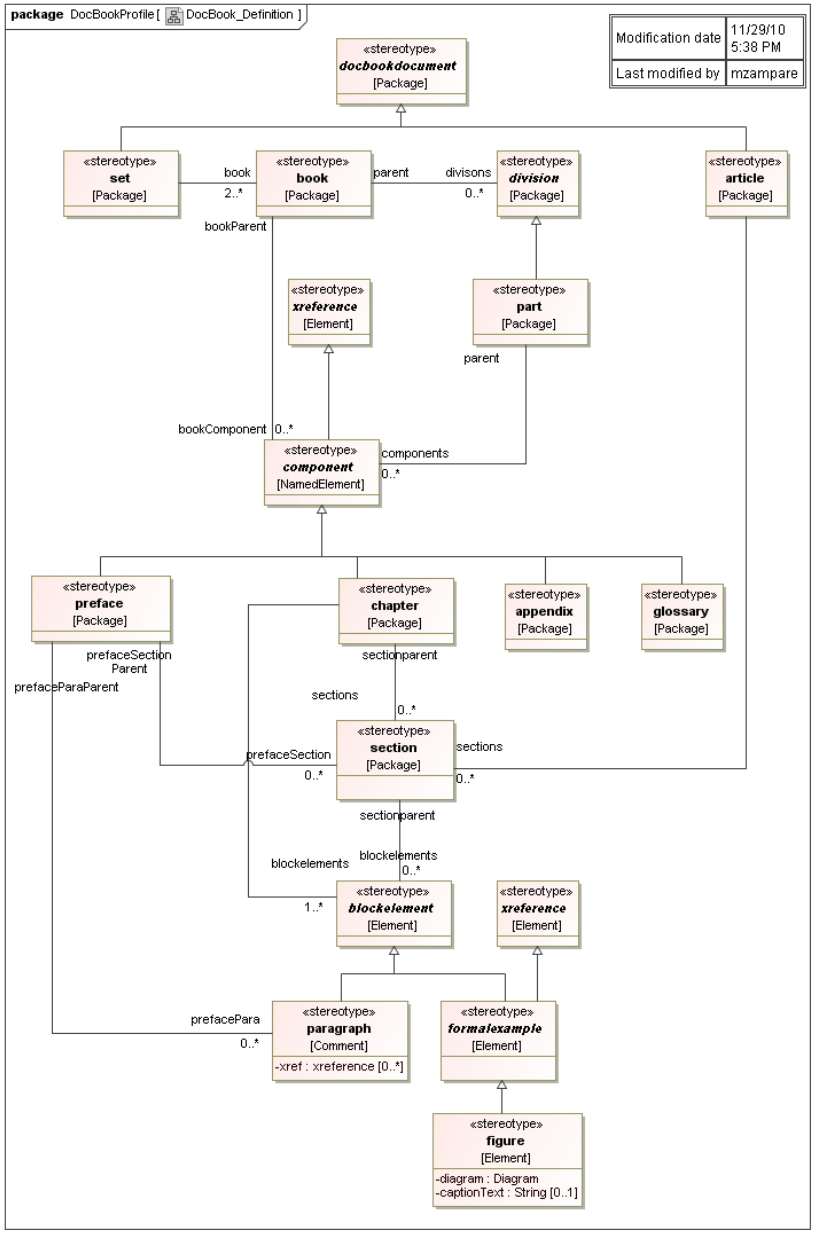
The Problem

- Document is culturally the primary artifact
- Models of system live a separate life
- Diagrams are copy/pasted into documents
- Description of model elements is partially kept separate from model (description added to diagrams in document)
- Two sources of information (Model and Document) have to be maintained – get out of sync
- Document generation engines (velocity template) have typically hard-coded references to model elements and diagrams

Suggested solution

- Create a model of the document
- Reference existing system model elements by unique identifiers
- Document model and system model may be aggregated in same project model
- A subset of the DocBook ontology is mapped to model elements using a dedicated SysML profile

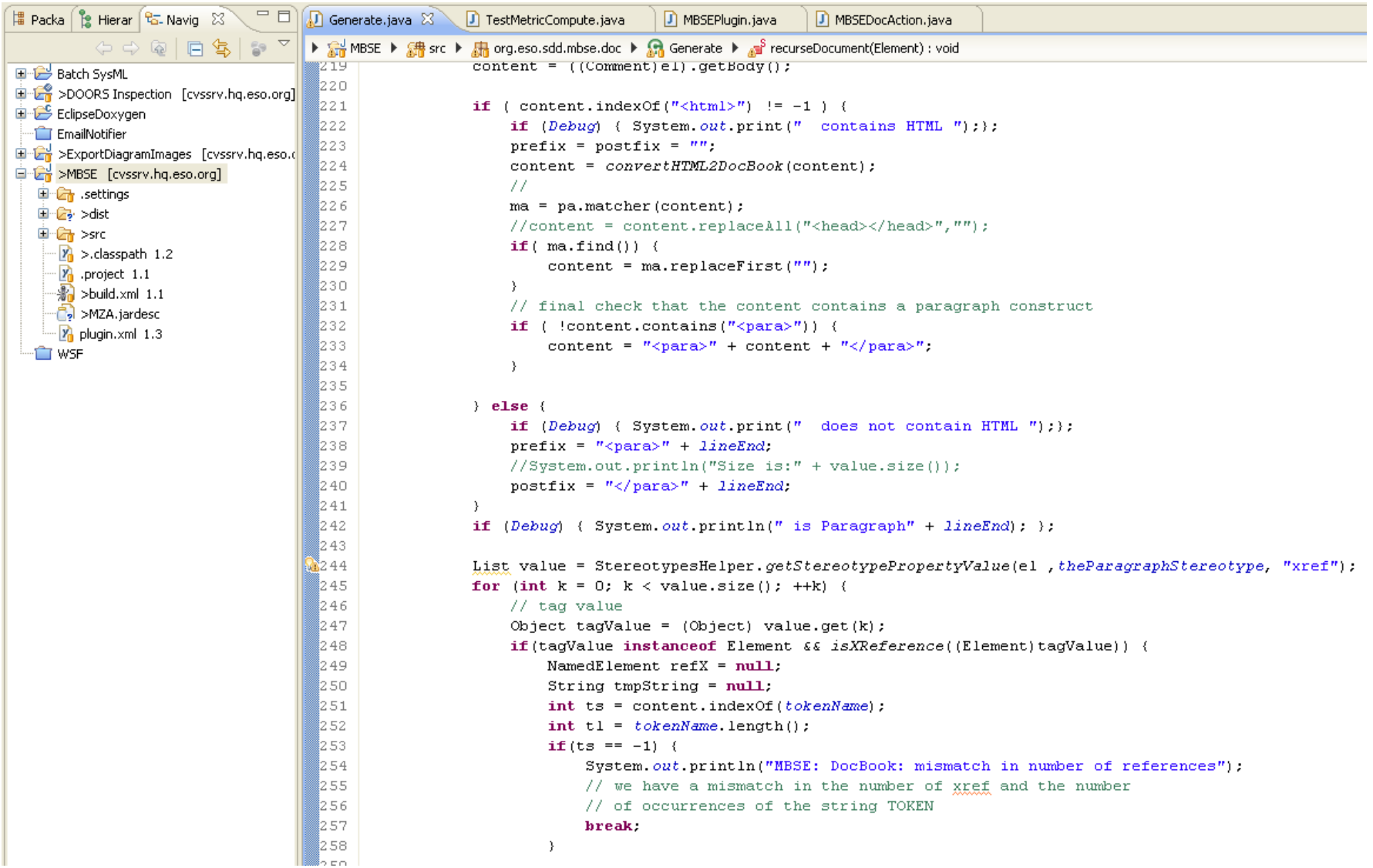
SysML Profile



Implementation

- Use standard documentation format:
  - DocBook, maintained by OASIS, http://www.oasis-open.org/home/index.php
- Use SysML profiles to introduce DocBook elements in SysML models, in a tool independent fashion
- Use customization facilities provided by SysML Tool vendor to write code for export

Software


Packa
Hierar
Navig
Batch SysML
>DOORS Inspection [cvssrv.hq.eso.org]
EclipseDoxygen
EmailNotifier
>ExportDiagramImages [cvssrv.hq.eso.
>MBSE [cvssrv.hq.eso.org]
.settings
>dist
>src
>.classpath 1.2
.project 1.1
>build.xml 1.1
>MZA.jardesc
plugin.xml 1.3
WSF
Generate.java
TestMetricCompute.java
MBSEPlugin.java
MBSEDocAction.java
MBSE
src
org.eso.sdd.mbse.doc
Generate
recurseDocument(Element) : void
content = ((Comment)el).getBody();
if (content.indexOf("<html>") != -1) {
if (Debug) { System.out.print(" contains HTML ");};
prefix = postfix = "";
content = convertHTML2DocBook(content);
//
ma = pa.matcher(content);
//content = content.replaceAll("<head></head>","");
if(ma.find()) {
content = ma.replaceFirst("");
}
// final check that the content contains a paragraph construct
if (!content.contains("<para>")) {
content = "<para>" + content + "</para>";
}
} else {
if (Debug) { System.out.print(" does not contain HTML ");};
prefix = "<para>" + lineEnd;
//System.out.println("Size is:" + value.size());
postfix = "</para>" + lineEnd;
}
if (Debug) { System.out.println(" is Paragraph" + lineEnd); };
List value = StereotypesHelper.getStereotypePropertyValue(el ,theParagraphStereotype, "xref");
for (int k = 0; k < value.size(); ++k) {
// tag value
Object tagValue = (Object) value.get(k);
if(tagValue instanceof Element && isXReference((Element)tagValue)) {
NamedElement refX = null;
String tmpString = null;
int ts = content.indexOf(tokenName);
int tl = tokenName.length();
if(ts == -1) {
System.out.println("MBSE: DocBook: mismatch in number of references");
// we have a mismatch in the number of xref and the number
// of occurrences of the string TOKEN
break;
}

Usage

- Stereotype tags are used to reference to system element models
- Text is entered in comments (which can be visualized in diagrams)
- References to model element or diagrams are selected using the tool navigation facilities

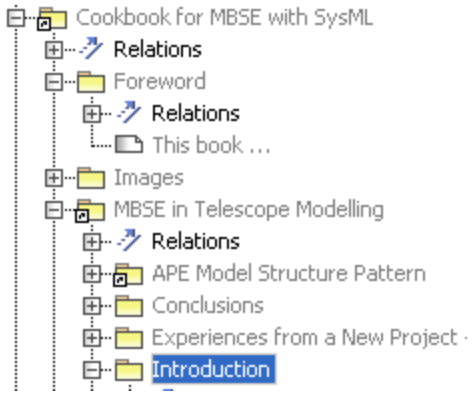
Usage Example: adding a chapter

Adding a chapter B to a book A

1. Create a package B for the chapter, give it a name
2. Stereotype package B by «chapter»
3. Modify properties of «book» element A, by adding a reference to B, .e.g. modifying the tagged value to include B in its list

Internal References

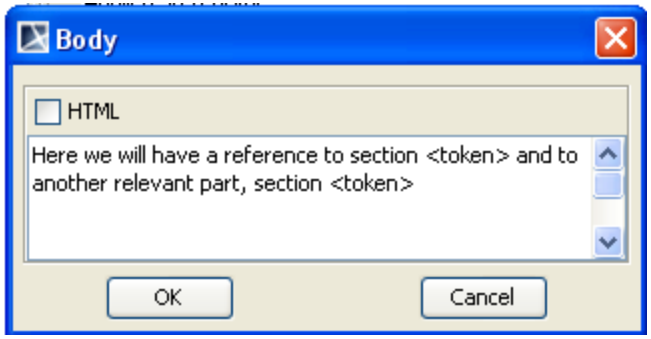
- Unique elements ID are generated following containment tree hierarchy

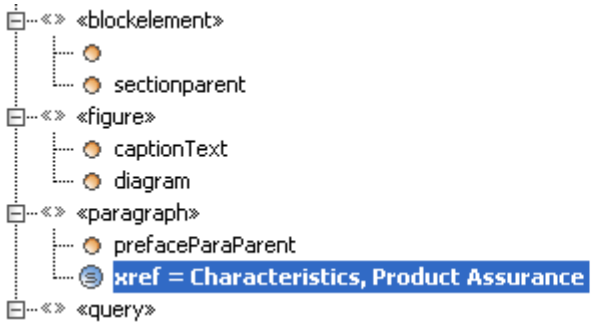


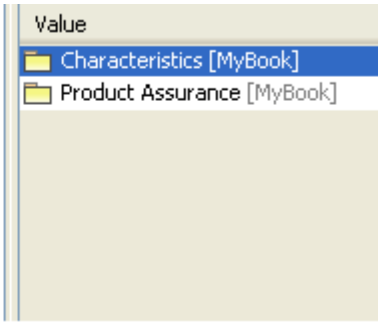
<chapter
xml:id="APE_Project_Views__Cookbook__CookbookforMBSE withSysML__MBSEinTelescopeModelling__Introduction" >

Usage: adding internal reference

Paragraphs may contain references to book elements







Shortcomings

- Missing validation rules (but could be added)
- Difficult deployment
- Dependency on API version (potentially)
- The insertion of document hierarchies is tedious without a WYSIWIG editor or similar

Shortcomings 2

- which subset of DocBook should be supported?
- creating cross-references (also internal ones) is possible but a bit cumbersome
- comments are elements without a name and therefore a bit difficult to find in the model browser
- MagicDraw's text editor supports only HTML which is not compliant with DocBook and the text has to be transformed (which is done automatically when generating)

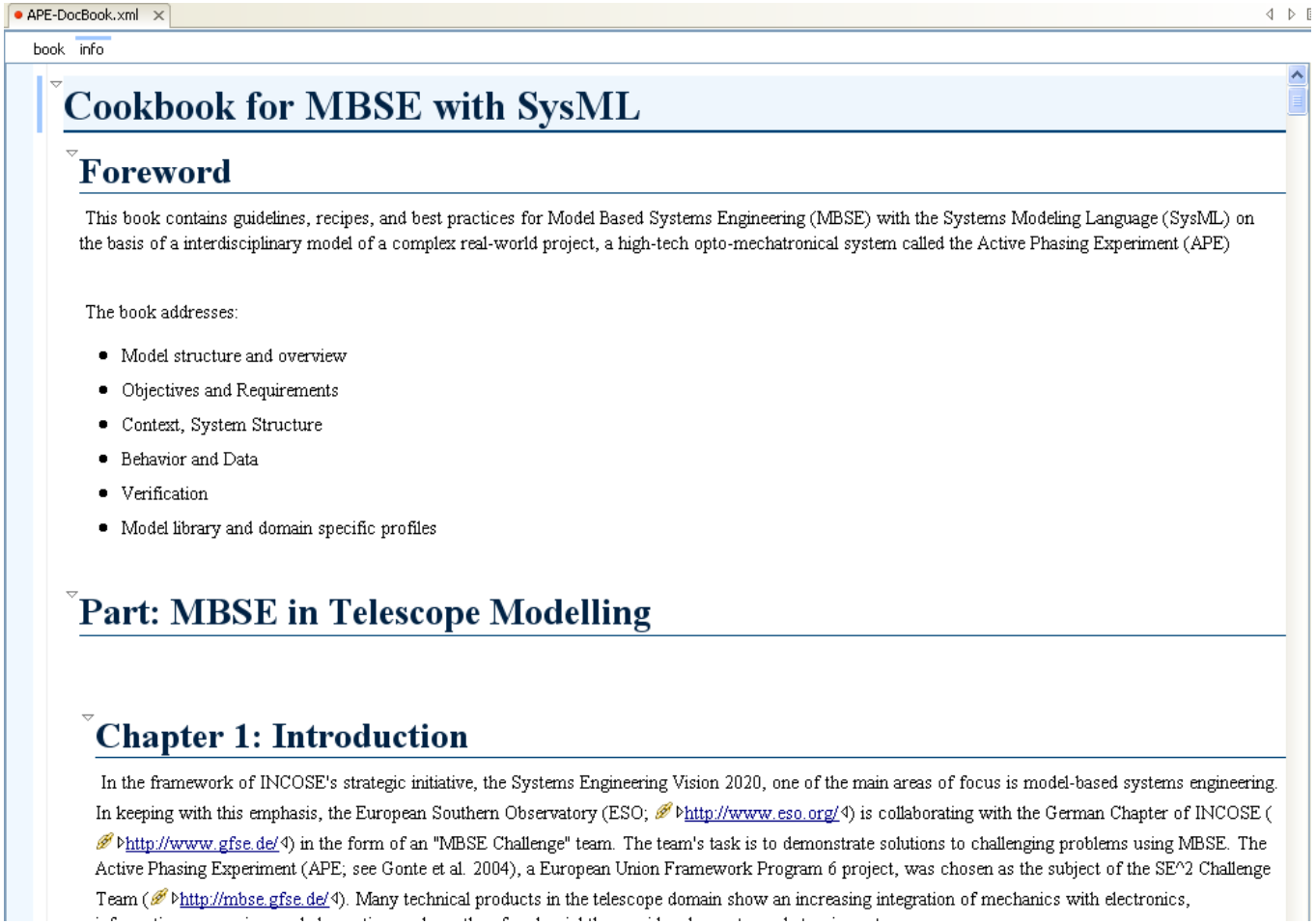
Advantages

- It adds advantages for configuration control since different document parts may be modified concurrently by different authors (assumes a version control system for underlying model from vendor's side)

Using comments

- Pros and Cons of using the *comment* element:
  1. Comments can be visualized directly in a SysML diagram
  2. Comments do not have a name, which makes it difficult to identify them.

Document Example



Advantages

- consistent integration of system model and system documentation
- direct linking to model elements (also diagrams) from the document
- changes of diagram names is automatically reflected in the document
- using proper definition of the stereotype associations only compatible elements can be selected to compose the document, e.g.
- a figure references diagrams, a chapter references paragraphs.
- the documentation is at the same time navigable in the model and printable
- documents are modeled in a tool independent way. Only a small plugin is needed to generate DocBook XML.