1. ATL Transformation Example: Measure to XHTML

The Measure to XHTML example describes a transformation from a Measure model to a XHTML file representing the measurement data.

1.1. Transformation Overview

The aim of this transformation is to generate a XHTML file from the input data contained in a Measure model.

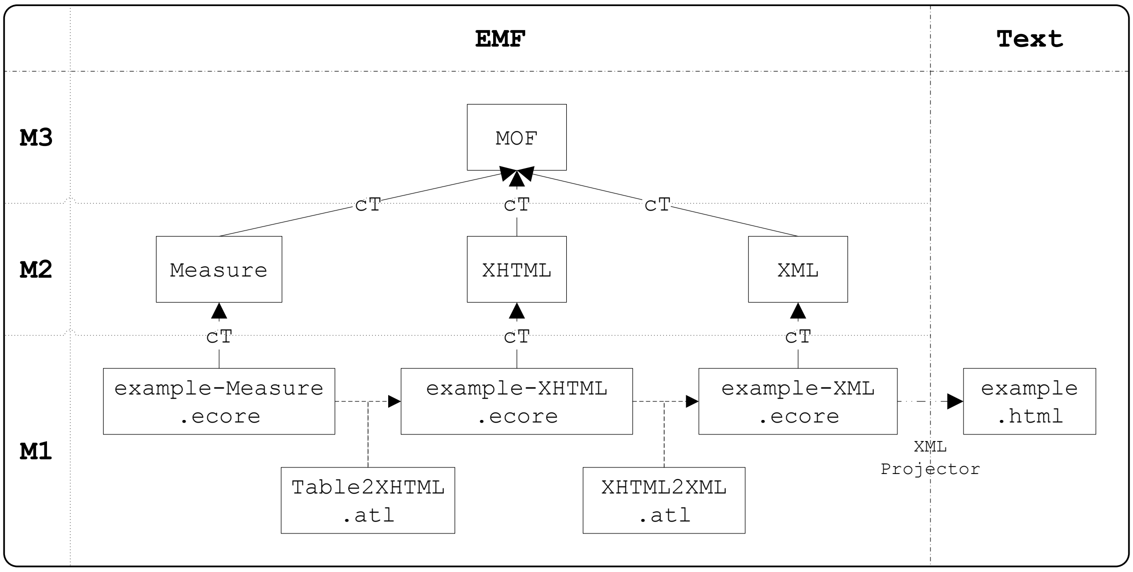
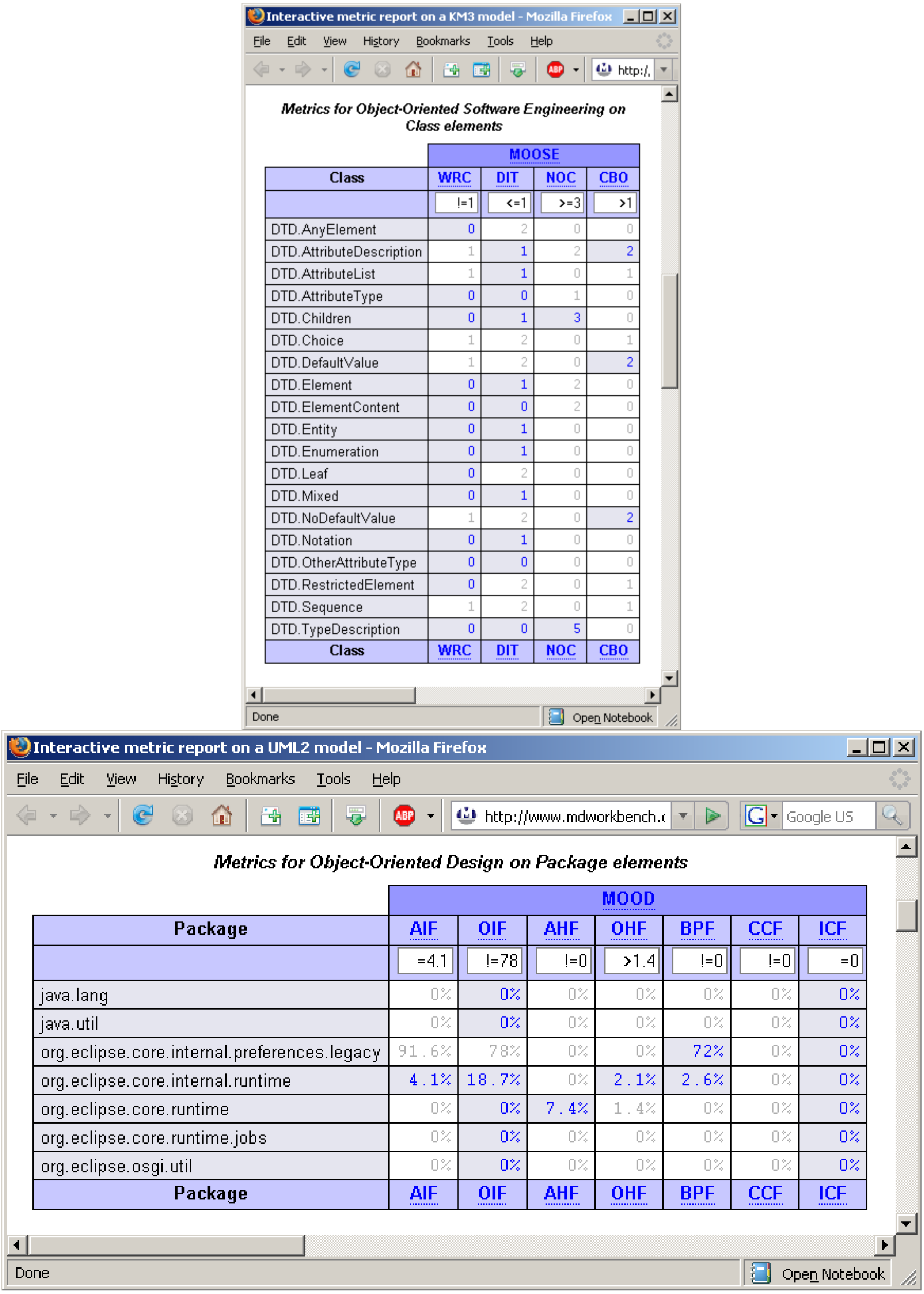


Figure 1: Overview of the transformation

The generation of the output XHTML file is realized by a first transformation from Measure to XHTML, followed by the usage of a projector. The projector consists in a transformation from XHTML to XML and the predefined XML extractor (XHTML is a XML-like language). The output .html file contains XHTML tables for each category of metrics and per kind of model element measured.


Interactive metric report on a KM3 model - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http:/,
Done
Open Notebook
Interactive metric report on a UML2 model - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mdworkbench.c
Google US
Done
Open Notebook

Metrics for Object-Oriented Software Engineering on Class elements

	MOOSE			
Class	**WRC**	**DIT**	**NOC**	**CBO**
	!=1	<=1	>=3	>1
DTD.AnyElement	0	2	0	0
DTD.AttributeDescription	1	1	2	2
DTD.AttributeList	1	1	0	1
DTD.AttributeType	0	0	1	0
DTD.Children	0	1	3	0
DTD.Choice	1	2	0	1
DTD.DefaultValue	1	2	0	2
DTD.Element	0	1	2	0
DTD.ElementContent	0	0	2	0
DTD.Entity	0	1	0	0
DTD.Enumeration	0	1	0	0
DTD.Leaf	0	2	0	0
DTD.Mixed	0	1	0	0
DTD.NoDefaultValue	1	2	0	2
DTD.Notation	0	1	0	0
DTD.OtherAttributeType	0	0	0	0
DTD.RestrictedElement	0	2	0	1
DTD.Sequence	1	2	0	1
DTD.TypeDescription	0	0	5	0
Class	**WRC**	**DIT**	**NOC**	**CBO**

Metrics for Object-Oriented Design on Package elements

	MOOD						
Package	**AIF**	**OIF**	**AHF**	**OHF**	**BPF**	**CCF**	**ICF**
	=4.1	!=78	!=0	>1.4	!=0	!=0	=0
java.lang	0%	0%	0%	0%	0%	0%	0%
java.util	0%	0%	0%	0%	0%	0%	0%
org.eclipse.core.internal.preferences.legacy	91.6%	78%	0%	0%	72%	0%	0%
org.eclipse.core.internal.runtime	4.1%	18.7%	0%	2.1%	2.6%	0%	0%
org.eclipse.core.runtime	0%	0%	7.4%	1.4%	0%	0%	0%
org.eclipse.core.runtime.jobs	0%	0%	0%	0%	0%	0%	0%
org.eclipse.osgi.util	0%	0%	0%	0%	0%	0%	0%
Package	**AIF**	**OIF**	**AHF**	**OHF**	**BPF**	**CCF**	**ICF**

Figure 2: Samples of XHTML file

2. Metamodels

2.1. Measure

The Measure meta-model is used to stored the data collected after a model measurement.

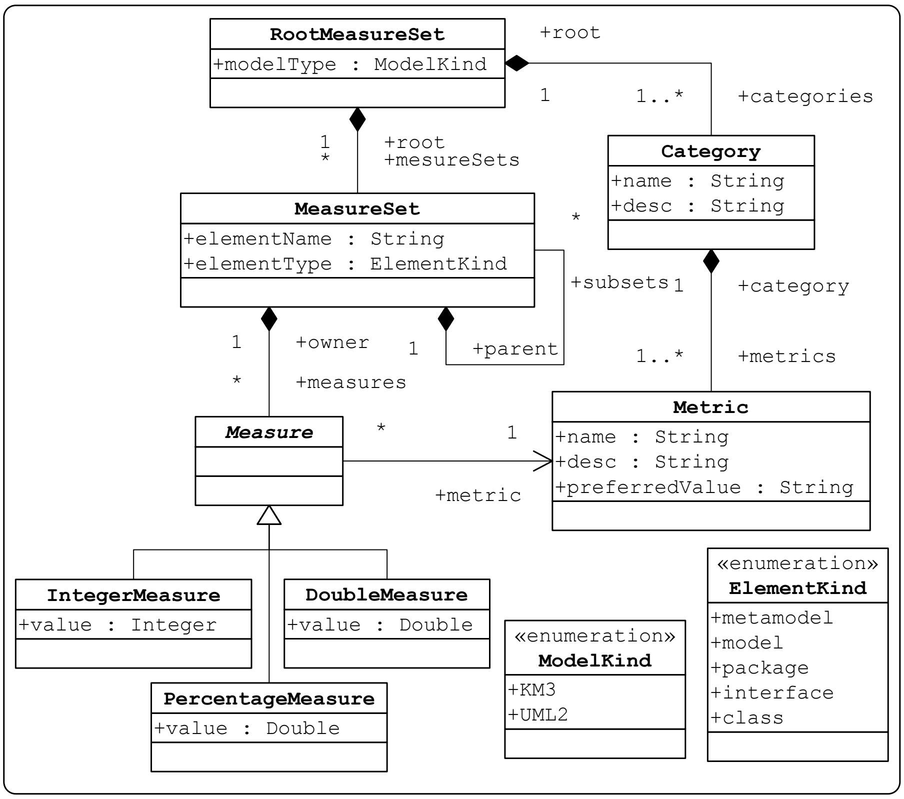


Figure 3: Measure meta-model

A measure model is in the following way made up: the root is a set of measure (*RootMeasureSet*) which contains information on the type of measured model (*modelType* among *KM3* or *UML2*), a set of categories of metric and sets of measure for each model element measured. A category (*Category*) corresponds to a metric set with a *name* and a description (*desc*) (an acronym and its definition). A category gathers one or more metric (*Metric*) also defined with a *name* and a description. A default predicate is also associated (*preferredValue*), it is the desired values for the metric (for example ≠ 0 or > 75). A set of measure (*MeasureSet*) described measurements performed on a model element (*elementName*) of a given type (*elementType* among *meta-model*, *model*, *package*, *interface* or *class*). The set of measure are structured between them, for example a set of measure on a package will contain the set of measure of the classes that this package contains. A measure (*Measure*) is associated to a metric and is declined in several versions. Measures with an integer, real or percentage value (respectively *IntegerMesasure*, *DoubleMeasure* and *PercentageMeasure*).

2.2. XHTML

The source metamodel of XHTML represents the XHTML language in version 1.1, and is base on the Strict DTD [3] [4].

3. Transformation from Measure to XHTML

3.1. Rules specification

These are the rules to transform a Measure model to a XHTML model.

- For each not empty sequence of MeasureSet of the same element type, the following element is created:
  - A Table element.
- For each Table element created, the following elements are created:
  - A header Row element.
  - Several value Row elements.
- For each header Row element created, the following elements are created:
  - A first Cell element with the kind of the model element measured.
  - Several Cell elements with the name of each metric defined for this kind of model element.
- For each value Row element created, the following elements are created:
  - A first Cell element with the name of the model element measured.
  - Several Cell elements with the value of each metric defined for this kind of model element.

3.2. ATL code

This ATL code for the Measure2XHTML transformation consists in 8 helpers and 23 rules.

The transformation uses the library defined in section 0.

The attribute helpers *useCSSFile* and *useJSFile* are flags for using external CSS and JavaScript files or the embedded version provided by the helpers *getStyleSheet* and *getJavaScript*.

The attribute helper *metricDefLocation* returns the URL of the metrics definition page.

The attribute helpers *modelKind* and *elementKind* are a mapping between the corresponding enumeration and the desired string value.

The attribute helper *html* is the root tag of the document.

The rule *RootMeasureSet*() creates a the XHTML document by calling different called an lazy rules.

The called rule *HTML* allocates a HTML element. The rule creates a HTML element (“html”) which is composed of a HEAD and BODY elements. The HEAD element is composed of a TITLE element, a META element, a LINK or STYLE element, a SCRIPT element and a NOSCRIPT element.

The called rule *LINK* allocates a LINK element. The rule creates a LINK element (“link”) which is linked to the CSS file stylesheet.css.

The called rule *STYLE* allocates a STYLE element. The rule creates a STYLE element (“style”) which includes the CSS returned by the helper *getStyleSheet*.

The called rule *SCRIPT* allocates a SCRIPT element. The rule creates a SCRIPT element (“script”) which is linked to the JavaScript file script.js.

The called rule *EmbeddedSCRIPT* allocates a SCRIPT element. The rule creates a SCRIPT element (“embeddedScript”) which includes the JavaScript returned by the helper *getJavaScript*.

The called rule *NOSCRIPT* allocates a NOSCRIPT element. The rule creates a NOSCRIPT element (“noscript”) for preventing users to activate JavaScript in their browser.

The called rule *ACRONYM* allocates an ACRONYM element. The rule creates a nACRONYM element (“acronym”) with a title and a value.

The called rule *LI* allocates a LI element. The rule creates a LI element (“li”) with a text.

The called rule *BR* allocates a BR element. The rule creates a BR element (“br”).

The called rule *HowToDIV* allocates a DIV element. The rule creates a DIV element (“howtoDiv”) which is composed of a FIELDSET element and two BR elements. The FIELDSET element is composed of a LEGEND element, P and UL elements. The general content of this DIV is the usage of the interactive version of the report.

The called rule *TABLE* allocates a TABLE element for a sequence of MeasureSet and for a Category. The rule creates a TABLE element (“table”) which is composed of CAPTION, THEAD, TFOOT and TBODY elements.

The called rule *FirstHeaderRow* allocates a TR element for a Category. The rule creates a TR element (“tr”) which is composed of two TH elements. The first TH is empty and the second one is composed of an A element and for the *colspan* given. The A element is composed of an ACRONYM element with the name and description of the category given.

The lazy rule *HeaderRow* allocates a TR element for a MesureSet and a Category. The rule creates a TR element (“tr”) which is composed of first header cell and header cells.

The called rule *FirstHeaderCell* allocates a TH element. The rule creates a TH element (“th”) with the element type given.

The lazy rule *HeaderCell* allocates a TH element for a Metric. The rule creates a TH element (“th”) which contains the name and description of the Metric given as an ACRONYM in an A element.

The called rule *InputTextRow* allocates a TR element for a sequence of Mesure. The rule creates a TR element (“tr”) which is composed of a first empty TH element and input text cells.

The called rule *InputTextCell* allocates a TH element for a Mesure. The rule creates a TH element (“th”) which contains an INPUT element which default value is the *preferredValue* of the Measure.

The lazy rule *DataRow* allocates a TR element for a MesureSet and a Category. The rule creates a TR element (“tr”) which is composed of first data cell and data cells.

The called rule *FirstDataCell* allocates a TD element. The rule creates a TD element (“td”) with the element name given.

The lazy rule *DataCell* allocates a TD element for a Metric. The rule creates a TD element (“td”) which contains the string value of the measure.

The called rules *CDATA* allocates a CDATA element for a string. The rule creates a CDATA element which value is the given string.

The called rules *PCDATA* allocates a PCDATA element for a string. The rule creates a PCDATA element which contains a CDATA with the string given.

4. ATL Library MeasureHelpers

4.1. ATL code

This ATL code for the MeasureHelpers library consists in 18 helpers.

The two helpers *metricName* returns the metric name prefixed by the name of the category of the metric.

The helper *allMetrics* returns all the metrics which category name is given.

The helper *absoluteName* returns the absolute name of the measured model element for this measure set.

The helper *allMeasureSets* returns all the measure sets for a given element type and sorted by element name.

The helper *categories* returns all the categories used by the measures of the measure set.

The helper *allMeasures* returns all the measures of the measure set and sorted by metric full name.

The helper *allMeasures* returns all the measures of the measure set for a category.

The helper *getModelKind* returns the String value for model type of the root measure set.

The helper *simplifiedValue* returns the value with at most one digit after the dot.

The helper *getElementKind* returns the String value for element type of the measure set.

The helpers *stringValue* returns a string value for a measure with an integer, a real or a percentage value.

The helper *toNumber* returns a real value for a percentage measure.

The helper *matchPreferredValue* returns true if the value match the predicate for the preferred value.

5. XHTML Projector

The XHTML projector is a transformation from XHTML to XML followed by the predefined XML extractor.

This can be done in this way, because XHTML is a XML-like language.

The XHTML HTML element is mapped to the XML Root element.

Other XHTML mark-up are mapped to XML Element element.

Each attribute of a XHTML mark-up is mapped as a XML Attribute element.

The XHTML CDATA or PCDATA are mapped to XML Text element.

6. References

[1] ATLAS (ATLantic dAta Systems) Official Webpage: http://www.sciences.univ-nantes.fr/lina/ATLAS/

[2] The Atlantic Zoo: http://www.eclipse.org/gmt/am3/zoos/atlanticZoo/

[3] XHTML 1.0 Strict DTD: http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd

[4] XHTML 1.1 - Module-based XHTML - Second Edition: http://www.w3.org/TR/xhtml11/