Modernizing Domain-Specific Languages with XMLText and IntellEdit

Patrick Neubauer, Robert Bill, and Manuel Wimmer

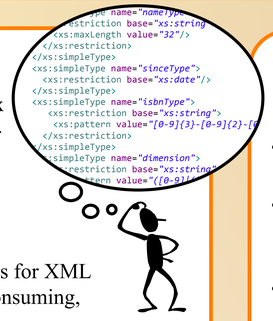
Contact: neubauer@big.tuwien.ac.at

Business Informatics Group
Institute of Software Technology and Interactive Systems
TU Wien

http://XMLIntellEdit.big.tuwien.ac.at

24th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER), February 21-24, 2017, Klagenfurt, Austria

Problem



- Immutable and verbose concrete XML syntax not intended for high human-comprehensibility and maintainability [1]
- Lack of dedicated and advanced editing support for huge corpus of XML Schema-based languages
- Creation and maintenance of advanced editors for XML Schema-based languages is complex, time-consuming, and not automated

Motivation

- State-of-the-art language workbenches like Xtext [3] enable the development of rich and modern modeling workbenches
- Automation of XML Schema-based language modernization
- Establishment of interoperability between XMLware, Grammarware, and Modelware
- Minimization of required language engineering knowledge and workload associated with providing advanced editing capabilities
- Liberation from XML's fixed concrete syntax
- Complete backward compatibility to vast amount of XML Schema documents

Goal

Modernization of XML Schema-based languages by exploiting language definitions for the automated generation of modeling workbenches and advanced editing facilities.

References

[1] G. J. Badros. JavaML: A Markup Language for Java Source Code. *Computer Networks*, 33(1):159–177, 2000.

[2] M. Fowler. Domain-specific languages. Pearson Education, 2010.

[3] M. Eysholdt and H. Behrens. Xtext: Implement your Language Faster than the Quick and Dirty Way. *Companion Proceedings of OOPSLA*, pages 307–309. ACM, 2010.

[4] P. Neubauer, A. Bergmayr, T. Mayerhofer, J. Troya, and M. Wimmer. XMLText: From XML Schema to Xtext. *Proceedings of the 8th International Conference on Software Language Engineering (SLE)*, Springer, LNCS 8706, 2015.

[5] D. Steinberg, et al. EMF: Eclipse Modeling Framework. Pearson Education, 2008.

[6] P. Neubauer, R. Bill, T. Mayerhofer, and M. Wimmer. Automated Generation of Consistency-Achieving Model Editors. *Proceedings of the 24rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, IEEE, 2017.

[7] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals". in Soviet Physics Doklady, 1966.

Approach

Introduction

- Builds on XMLText [4], IntellEdit [6], Eclipse Modeling Framework (EMF) [5], and Xtext [3]
- Offers a unique combination of automated language modernization workflows for tackling shortcommings of XML Schema-based languages
- Significantly lowers required knowledge and effort associated with the delivery of advanced editing capabilities

Modernization Workflow

1a Transformation of *XML Schema* to *Metamodel*

1b Transformation of *Restrictions* to *Formal Constraints*

2 Adaptation of *Metamodel* for effective language grammar production

3 Transformation of *Metamodel* to *Basic Language Workbench* which includes parsers and serializers for instance-level transformations

4 Generation of *Advanced Editing* facilities, including extended validation, content-assist, and quick fix solution, from *Formal Constraint* specifications

Advanced Editor Execution Workflow

The *Runtime Plugin* is continually executed alongside the *Advanced Editor:*

- Identification and representation of error details by the *IntellEdit OCL Interpreter*
- Request and retrieval of valid *autocompletions* and *quick fix solutions* by *Content Assist Provider* and *Quick Fix Provider*, respectively

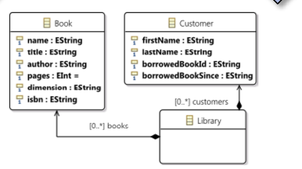
Library Example Language

XML Schema

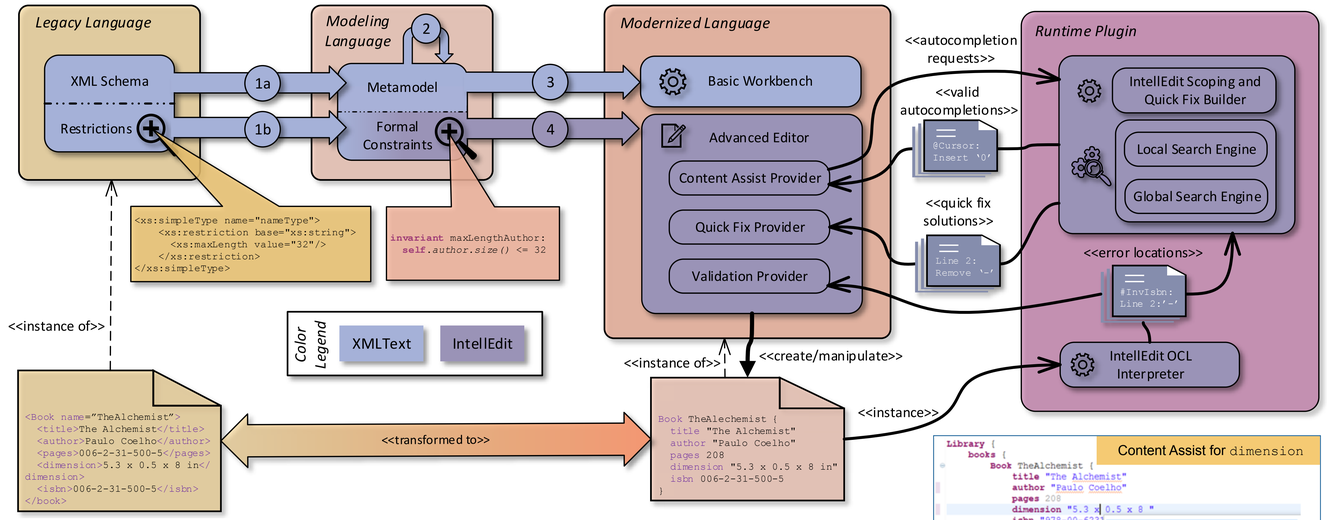
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- definition of simple types -->
  <xs:simpleType name="nameType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="32"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="sinceType">
    <xs:restriction base="xs:date"/>
  </xs:simpleType>
  <xs:simpleType name="isbnType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{3}-[0-9]{2}-[0-9]{4}-[0-9]{3}-[0-9]"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="dimension">
    <xs:restriction base="xs:string">
      <xs:pattern value="([0-9]|([1-9][0-9]+))([.][0-9]+)? x
                         ([0-9]|([1-9][0-9]+))([.][0-9]+)? x
                         ([0-9]|([1-9][0-9]+))([.][0-9]+)?
                         (centimeters|cm|in|inches)"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- definition of complex types -->
  <xs:complexType name="libraryType">
```

1a 1b

Ecore Metamodel




```
class Book
{
    ...
    invariant maxLengthAuthor: self.author.size() <= 32;
    invariant
    patternDimension: self.dimension.matches('([0-9]|([1-9][0-9]+))([.][0-9]+)? x
                                              ([0-9]|([1-9][0-9]+))([.][0-9]+)? x
                                              ([0-9]|([1-9][0-9]+))([.][0-9]+)?
                                              (centimeters|cm|in|inches)');
    invariant patternIsbn: self.isbn.matches('[0-9]{3}-[0-9]{2}-[0-9]{4}-[0-9]{3}-[0-9]')
}
```



Validation and Content Assist

Ranked recursive constraint evaluation by selecting a closed match [7] of a set of expected values with actual values of sub-expressions.

Quick Fix Solutions

Three-stage neighborhood search looking for different kind of concrete changes:

- *Small Local Search* — changes resolving single expression violations
- *Large Local Search* — changes resolving single expression violations but considering all feature values and object instances involved in the expression violations
- *Global Search* — arbitrary changes on entire instance