Managing Consistency of Distributed Documents

Anthony Finkelstein

With: Ernst Ellmer, Wolfgang Emmerich, Torbjoern Revheim, Danila Smolko, Andrea Zisman

Software Systems Engineering Group
Department of Computer Science
University College London

Outline

- Introduction
- Our approach
- Consistency rules & consistency links
- Classification of the consistency rules
- Architecture
- Demonstration
- Conclusion and future work

Introduction

- Large number of documents
- Distributed document generation
- Multiple actors
- Different perspectives and forms
- Use of heterogeneous applications

Inconsistent and conflicting documents

Goal

- Practical support for managing consistency in this setting
- Characteristics
  - Simple
  - Lightweight
  - Scaleable
  - Can be used in conjunction with existing tools
  - Applicable to software engineering documents

Our Approach

- Uses XML *and* related technologies
- Source documents are represented in XML
- This is a reasonable assumption as there are a wide range of emerging standards for document exchange that use XML
  - *Including the area of software engineering*
- A large range of tools now export XML, an increasing number use XML as an internal data representation
  - *Including the area of software engineering*

Running Example

- Fragments of a specification of a "Meeting Scheduler"
- Documents expressed in UML (using Rational Rose) and in Z
- Made available in:
  - XMI
  - UXF
  - ZIF

Key Components

Consistency Rule

expresses relationships that should exist between elements of distributed documents

Consistency Link

associates related elements and identifies either consistent or inconsistent elements

You will see examples as we proceed

Consistency Rule (Example)

```
<ConsistencyRule   id = "r0"   type = "CT">
<Description> For every instance in a collaboration
   diagram there must exist a class in a class diagram
   with the same name. </Description>
<Source><XPointer>
   root().child(all,Package).(all,CollaborationDiagram).
   (all,Collaboration).(all,Instance)
</Xpointer></Source>
<Destination
   dest_id="dest1"><XPointer>root().child(all,Package).(
   all,ClassDiagram).(all,Class)</Xpointer>
</Destination>
<Condition expsource= "origin().attr(CLASS)"
                    op="eq" dest_ref="dest1"
                    expdest="origin().attr(NAME)"
</Condition>
</ConsistencyRule>
```

Things to Note

- There is a consistency rule DTD, in other words consistency rules are defined in XML too
- "source" and "destination" elements are identified by XPointer expressions
- There is a "condition" given by a condition expression which defines the relationship which should hold between the elements there may be many condition expressions
- Each rule has a "type" which defines the nature of the consistency link generated between the elements

Rule Type

- Three types of rule: CT, CF, and IF.
  - C or I indicates whether two elements are linked because they are in a consistent or inconsistent state, respectively.
  - T or F indicates if the consistency rule is or not mandatory. If the value is T, then for every Source element it is necessary to have a Destination element. If the value is F and there is no Destination element for a Source element, it does not mean that an inconsistency has occurred.

Consistency Link (Example)

```
<ConsistencyLink xml:form="extended" inline="false"
   ruleid="r1">
  <State>consistent</State>
  <Locator xml:form="locator"
   href="C:\home\uml\associate_participant_collaboration
   _diagram.xml#id(i4)”/>
  <Locator xml:form="locator"
   href="C:\home\uml\business_entities_classdiagram.xml#
   id(class4)” />
</ConsistencyLink>
```

Things to Note

- There is a consistency link DTD, in other words consistency links are defined in XML
- The links can exploit the full potential of XLink and XPointer
- Links can be made to elements or to the consistency rules themselves

Consistency Rules - Expressiveness

T1: ***Existence of related elements in different documents***

T11: **Mandatory**

For every association that appears in two or more different UML models, the classes related by this association must have the same name.

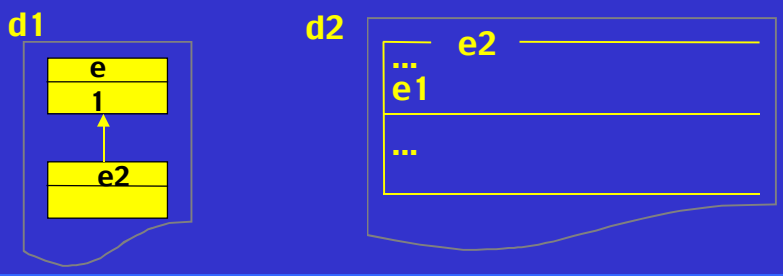
T12: **Non-mandatory**

Classes with the same name in different class diagrams, of the same UML model, are considered to be identical.

Consistency Rules - Expressiveness

T2: ***Mandatory Cross-reference***

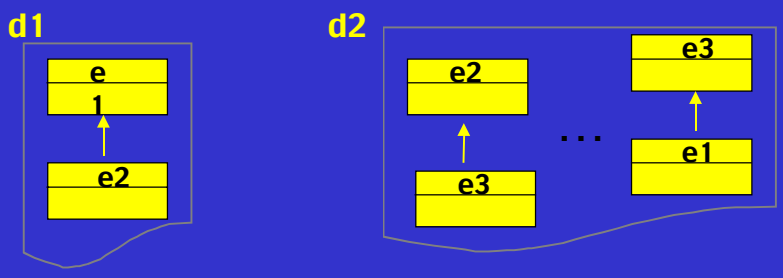
For every class e1 with subclass e2 in a UML class diagram d1, if there is a schema e'2, in a Z document d2, with the same name as subclass e2, then there must exist an inclusion e'1 in schema e'2, with the same name as class e1.



Consistency Rules - Expressiveness

T3: ***Non-existence Cross-reference***

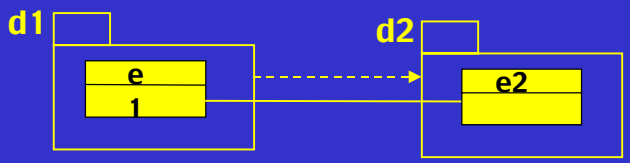
For every class e1 and subclass e2 in a UML class diagram d1, e2 should not be a superclass of e1 in any other class diagram d2, of the same UML model, for any level of nesting.



Consistency Rules - Expressiveness

T4: ***Dependent relation***

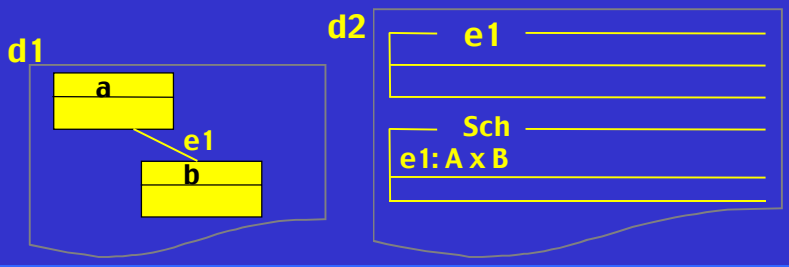
For every association between two classes e1 and e2, in two different package diagrams d1 and d2, there must exist a dependency between these two packages.



Consistency Rules - Expressiveness

T5: ***Associativity***

For every association e1 in a UML class diagram d1, there must exist either a schema e2 in a Z document d2, with the same name as the association e1, or a variable e3 in a schema in a Z document d2, with the same name of the association e1, and the variable must be of type relation or cartesian product.



Consistency Rules - Expressiveness

T6: ***Mandatory Non-existence***

A primitive process e1 in a DFD d1 should not be decomposed into another DFD d2.

T7: ***Existence of related documents***

For every non-primitive process e1 in a data flow diagram d1, there must exist an associated decomposition DFD (document d2).

T8: ***Existence of isolated elements/documents***

For every product being developed there must exist a document d1 related to the SRS of the product

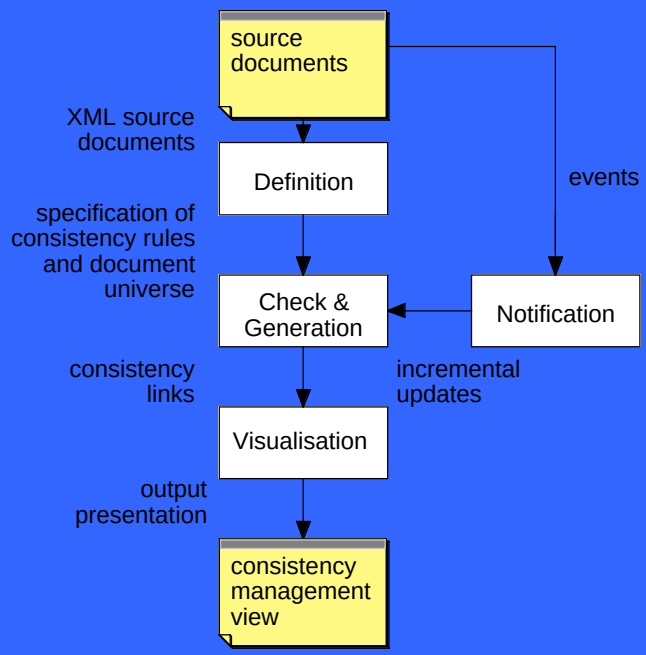
Consistency Rules - Expressiveness

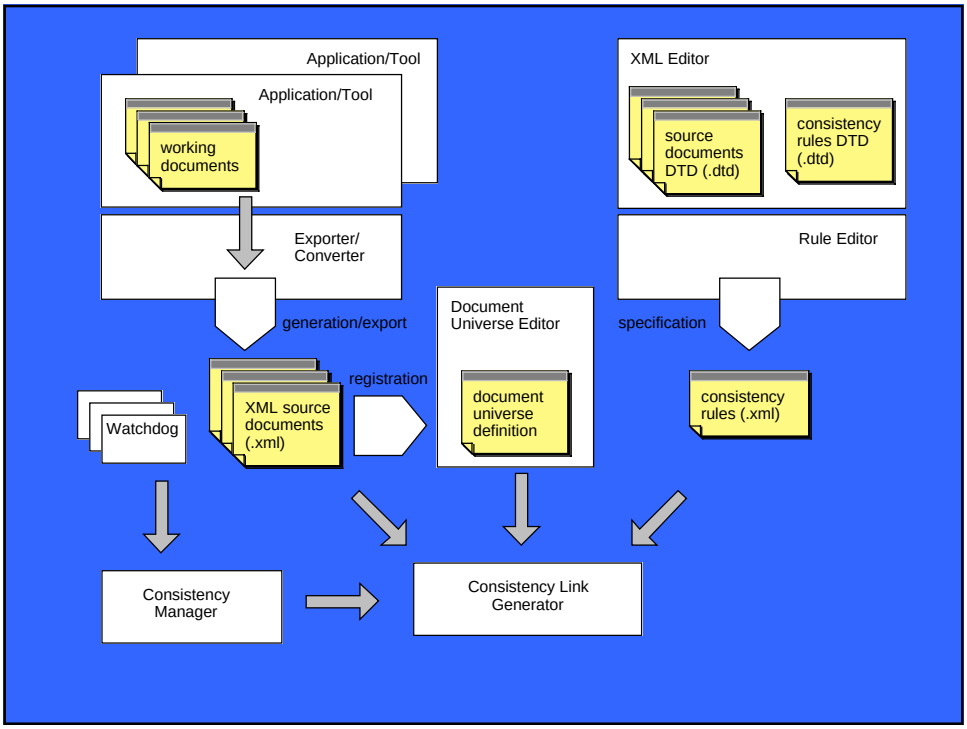
T9: ***Existence of constant elements in different documents***

For every method e1 of a class in a UML class diagram d1, there must exist a schema e2 in a Z specification d2, with the same name of the method, and the purpose of the schema e2 needs to be of value "operation".

T10: ***Elements related to an operation***

The total amount e1 spent in a day during a trip must be the result of the sum of the values ei in the receipts di (2 ≤ i ≤ n).

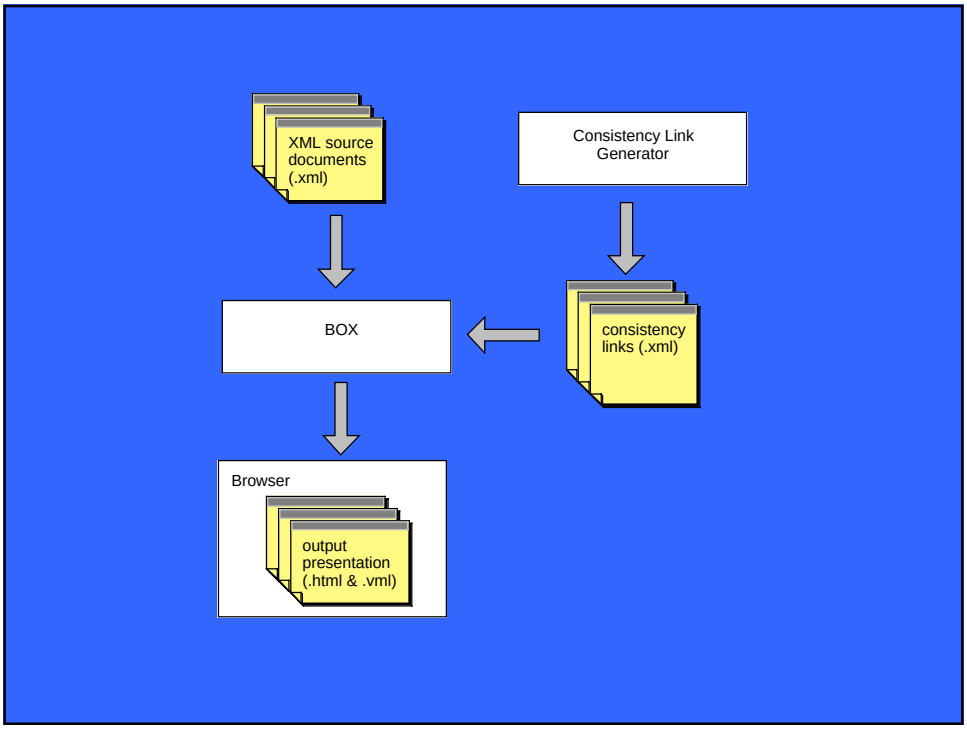




Sometimes Rules can get Large

```
<ConsistencyRule  id = "r11"  type = "CT">
<Description>For every association that appears in two...</Description>
<Source><XPointer>
   root().child(all,XMI.content).(all,Model_Management.Model).(all,Foundation.
   Core.Namespace.ownedElement).(all,Foundation.Core.Association)
</Xpointer></Source>
<Destination   dest_id="dest1"><XPointer>
   root().child(all,XMI.content).(all,Model_Management.Model).(all,Foundation.
   Core.Namespace.ownedElement).(all,Foundation.Core.Association)
</Xpointer></Destination>
<Condition expsource="origin().child(1,Foundation.Core.ModelElement.name)"
            op="eq"
            dest_ref="dest1"
            expdest="origin().child(1,Foundation.Core.ModelElement.name)"
 </Condition>
<Operator  value="AND"/>
<Condition expsource="id(origin().child(1,Foundation.Core.Association.connection).
 (all,Foundation.Core.AssociationEnd).(all,Foundation.Core.AssociationEnd.type).
 child(1,#element,xmi.idref,*)).child(1,Foundation.Core.ModelElement.name)"
            op="eq"
            dest_ref="dest1"
            expdest= ="id(origin(). child(1,Foundation.Core.Association.connection).
 (all,Foundation.Core.AssociationEnd).(all,Foundation.Core.AssociationEnd.type).
 child(1,#element,xmi.idref,*)).child(1,Foundation.Core.ModelElement.name)"
 </Condition> </ConsistencyRule>
```

Which is why you need a rule editor!



BOX (Browsing Objects in XML)

- A special purpose visual front-end
- Uses output from Rational Rose exporter
- Translates XMI to VML (Vector Graphic Markup Language)
- Adds HTML and javascript overlay
- Will display consistency links

Toolkit

JDK 1.2.2
XML parser for Java
(IBM Alphaworks)
Internet Explorer 5

Demo ...

Example (Meeting Scheduler)

- Documents:

- business_entities_classdiagram.xml
- create_meeting_collaboration_diagram.xml
- associate_participant_collaboration_diagram.xml

UXF

meeting_scheduler_i2 (Rose 98).xml
meeting_scheduler_i4 (Rose 98).xml
meeting_scheduler_i5 (Rose 98).xml

XMI

> zifex.xml

ZIF

Example (Meeting Scheduler)

Rules:

R1: *For every instance e1 in a collaboration diagram d1, there must exist a class e2 in a class diagram d2, with the same name.*

R2: *For every class e1 and subclass e2 in a UML class diagram d1, e2 should not be a superclass of e1 in any other class diagram d2, of the same UML model, for any level of nesting.*

R3: *For every association e1 in a UML class diagram d1, there must exist either a schema e2 in a Z document d2, with the same name as the association e1, or a variable e3 in a schema in a Z document d2, with the same name of the association e1, and the variable must be of type relation or cartesian product.*

As seen before

Achieved

- Consistency detection of distributed documents on the WWW
- Simple and relatively lightweight
- Leverages XML and related technologies
- Applicable in a large variety of "Web Information Management" settings
- Readily extensible and acts as an effective testbed
- Scaleable

Immediate Challenges

- Harden and refine our implementation
- Engage in some large-scale experimentation and use
- Migrate to XPath
- Finish off the notification and integrate with WebDAV

Long Term Challenges

- Refine our scaleability strategies
- Extend our experiments in other domains (for example syllabus management)
- Develop "inconsistency handling" strategies
- Extend visualisation
- Develop "consistency management" based web services