Model-Driven Iterative Development of 3D Web-Applications Using SSIML, X3D and JavaScript
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Preliminaries</td>
<td>1.1 Motivation</td>
</tr>
<tr>
<td></td>
<td>1.2 MDE for 3D web-applications with example</td>
</tr>
<tr>
<td>2. Roundtrip3D</td>
<td>2.1 Requirements</td>
</tr>
<tr>
<td></td>
<td>2.2 Approach of RTE</td>
</tr>
<tr>
<td></td>
<td>2.3 Transformations</td>
</tr>
<tr>
<td></td>
<td>2.4 Merging</td>
</tr>
<tr>
<td>3. Conclusion</td>
<td>3.1 Summary</td>
</tr>
<tr>
<td></td>
<td>3.2 Outlook</td>
</tr>
</tbody>
</table>
1.1 Preliminaries: Motivation

- **Application logic** and **3D content** → **two** different developer groups: 3D modelers, programmers
- E.g. X3D and JavaScript
- Traditionally: code-centric and **iterative** development

![Diagram showing the process of application logic and 3D content development](image)
1.1 Preliminaries: Motivation

- **Iterative** development
- Improvements with **model-driven** engineering [1]:
  - communication aid
  - contract
  - code generation
  -> concurrent development
  -> **Round-trip engineering** [2]

∑ Forward engineering
∑ Reverse engineering
∑ Preserve modifications
# Table of Contents

1. Preliminaries  
   1.1 Motivation  
   1.2 MDE for 3D web-applications with example  
2. Roundtrip3D  
   2.1 Requirements  
   2.2 Approach of RTE  
   2.3 Transformations  
   2.4 Merging  
3. Conclusion  
   3.1 Summary  
   3.2 Outlook
• 3D content and application code → multiple target languages (TL): X3D/ X3DOM, JavaScript

• Example application:

```javascript
window.onload = init;

function RobotControlClass( ) {
  var transform1 = document.querySelector('[DE
  var transform0 = document.querySelector('[DE

  this.button0.CLICKED = function( obj ) {
    var rot = transform0.getAttribute("rotation");
    var angle = parseFloat(rot.split( "\"\"\")[3],
    angle += 0.04;
    transform0.setAttribute("rotation", "1 0 0 ");
  }

  this.button1.CLICKED = function( obj ) {
    var angle = parseFloat(transform0.getAttribute("
    angle += 0.04;
    transform0.setAttribute("rotation", "1 0 0 ");
  }

  this.button2.CLICKED = function( obj ) {
    var angle = parseFloat(transform1.getAttribute("
    angle += 0.04;
    transform1.setAttribute("rotation", "1 0 0 ");
  }

  this.button3.CLICKED = function( obj ) {
    var angle = parseFloat(transform1.getAttribute("
    angle += 0.04;
    transform1.setAttribute("rotation", "1 0 0 ");
  }
```
1.2 Preliminaries: MDE for 3D applications

- Graphical domain specific language: SSIML (Scene Structure and Integration Modeling Language) [1]

- **Scene graph and application components** at high grade of abstraction
1. Preliminaries
  1.1 Motivation
  1.2 MDE for 3D web-applications with example
2. Roundtrip3D
  2.1 Requirements
  2.2 Approach of RTE
  2.3 Transformations
  2.4 Merging
3. Conclusion
  3.1 Summary
  3.2 Outlook
2.1 Roundtrip3D: Requirements

- Different **granularity** of model and code
- **Heterogeneous** target languages with different **scope**
- Common methods in RTE tools:
  - Model and code stored separately
  - Model derived from source code

**Synchronization issue** between code bases and model
1. Preliminaries
   1.1 Motivation
   1.2 MDE for 3D web-applications with example
2. Roundtrip3D
   2.1 Requirements
   2.2 Approach of RTE
   2.3 Transformations
   2.4 Merging
3. Conclusion
   3.1 Summary
   3.2 Outlook
• One common **intermediate** representation for languages and model

• Managable transformations: ➢ **Multi-tiered** approach
  ➢ **Task specific** transformation languages
• Modeling 3D applications with SSIML
2.2 Roundtrip3D: Overview RTE Approach

- Model-based generic storage in an intermediate model (IM)
- (Merge intermediate model with persistent intermediate model from previous iterations)?
• Build abstract syntax trees (ASTs) from intermediate model
• Code generation for each target language from abstract syntax trees
• **Concurrent** editing of code bases, i.e. 3D design and implementation of application logic
Reverse steps: Code parsing, built abstract syntax trees
Transformation of each abstract syntax tree to intermediate model

2.2 Roundtrip3D: Overview RTE Approach

- Reverse steps: Code parsing, built abstract syntax trees
- Transformation of each abstract syntax tree to intermediate model
• Sequentially merge derived intermediate model with persistent one
2.2 Roundtrip3D: Overview RTE Approach

- Derive domain model → modify
- Next forward iteration: Complete source code re-generation from persistent intermediate model

![Diagram showing the process of Roundtrip3D with models and code generation stages.]

**Domain Model**
- Start: SSIML Model
- Edit: SSIML Model

**generic M2M**
- Persistent Storage
- IM
- Merge
- Persistent IM
- IM

**specific M2M**
- Abstract Syntax Trees
- X3D
- JavaScript
- Merge
- Code generation

**Implementation**
- Concurrent editing
- X3D
- JavaScript
1. Preliminaries
   1.1 Motivation
   1.2 MDE for 3D web-applications with example
2. Roundtrip3D
   2.1 Requirements
   2.2 Approach of RTE
   2.3 Transformations
   2.4 Merging
3. Conclusion
   3.1 Summary
   3.2 Outlook
2.3 Roundtrip3D: SSIML → Intermediate model

- Generic transformations, with Java
- EMF reflective API
2.3 Roundtrip3D: SSIML → Intermediate model

- Generic transformations, with Java
- EMF reflective API

**SSIML Model**

- Value Access Relationship
- Touch Sensor Event Relationship
- Value Access Relationship
- Application Class RobotControl

**Scene robotScene**

- Group factoryGroup
- Object structure
- Primary Parent Child Relationship
- Object base
- Object segment0
- Object segment1

**Model**

- Object ValueAccessRelationship
- Object TouchSensorEventRelationship
- Object ValueAccessRelationship
- Object ApplicationClass
2.3 Roundtrip3D: Intermediate model ↔ AST

- Rule based, hybrid transformation language
- External mapping model [4]
2.3 Roundtrip3D: Code serialization and parsing

- Grammar for each target language
- Coarse granularity of grammar and meta model [8]

→ Derive code serializers and parsers

**X3D code skeleton**

```xml
<X3D version="3.0" profile="Immersive">
  <Scene>
    <Transform DEF="generatedTransform1" translation="0 0 0" rotation="0 0 1 0" scale="1 1 1" center="0 0 0" scaleOrientation="0 0 1 0">
      <Group DEF="factoryGroup">
        <Transform DEF="generatedTransform2" translation="0 0 0" rotation="0 0 1 0" scale="1 1 1" center="0 0 0" scaleOrientation="0 0 1 0">
          <Inline DEF="structure" url="factory.x3d">
          </Inline>
        </Group>
      </Transform>
    </Scene>
</X3D>
```

**JavaScript code skeleton**

```javascript
/** @id: ApplicationClass 381af861-5437-403d-9293-d3807e441005 */

function RobotControlClass( ) {
  ...
  var transform0 = document.querySelector( '[DEF="transform0"]' );

  this.button0_CLICKED = function( obj ) {
    ...
  }
}

function init( ) {
  var robotControlClass = new RobotControlClass( );
  document.querySelector( '[DEF="button0"]' ).addEventListener( "click", robotControlClass.button0_CLICKED );
  ...
}
2.4 Roundtrip3D: Model merging

- **Synchronize** relevant elements between model and code
- **Preserve** manually developed source code

2. Conformance checking
3. Interactive merging: one merge algorithm
4. Reconciliation

<table>
<thead>
<tr>
<th>IM from X3D</th>
<th>IM from Java-Script</th>
<th>Persistent IM</th>
</tr>
</thead>
</table>

![Diagram of model merging process]
1. Preliminaries
   1.1 Motivation
   1.2 MDE for 3D web-applications with example

2. Roundtrip3D
   2.1 Requirements
   2.2 Approach of RTE
   2.3 Transformations
   2.4 Merging

3. Conclusion
   3.1 Summary
   3.2 Outlook
3.1 Conclusion: Summary

- Eclipse based graphical model editor
- Forward & reverse engineering (X3D/ X3DOM, JavaScript) with respect to non-simultaneous development process
- Common intermediate representation and one merge algorithm to overcome synchronization issue

http://elrond.informatik.tu-freiberg.de/roundtrip3d/robot
1. Preliminaries
   1.1 Motivation
   1.2 MDE for 3D web-applications with example
2. Roundtrip3D
   2.1 Requirements
   2.2 Approach of RTE
   2.3 Transformations
   2.4 Merging
3. Conclusion
   3.1 Summary
   3.2 Outlook
3.2 Conclusion: Outlook

- Tree diff algorithms to avoid IDs
- Integrate Augmented Reality (Android devices, Java)
- Integrate C.A.V.E. like systems (C++)
Thank You!

References


Round-trip tools:

- Exist for 'ordinary' software
- E. g. UML Lab, Together, ...
- Simultaneous editing; provide different views

Tools for high level 3D development:

- High level dev. hardly supported
- Simplify programming task
- Rapid prototyping
- E. g. InTML [3] : describe different input devices, output devices, virtual objects, and 3D interaction techniques at high level, XML-based

Traditionally: Active code generation (code weaving, protected regions) [6, 7]

source: http://www.uml-lab.com/de/uml-lab
Model based storage

- Common methods:
  1. Store model and code separately
     - Redundancy
     - Synchronization through domain model; restrictions
  2. Completely derive model from code (render code)
     - Synchronize between TL
     - Many merge processes

- Intermediate model contains data and meta data
Eclipse Modeling Tools:

- (Meta-) Modeling of DSLs: Eclipse Modeling Framework (EMF)
- Graphical DSLs: Graphical Modeling Framework (GMF)
- (Graphical) model editor: ~120t effective lines of code
- Model validation: Object Constraint Language (OCL)
- Model transformations (M2M): ETL, (ATL, QVT)
- Code generation (M2T): JET, Xpand, (MOFScript)
- Generation of code parsers and serializers (M2T, T2M): Xtext
- Change detection, model merging and resolving conflicts: ECL, EML, ETL
Backup: SSIML ↔ IM (XMI)

M2M: SSIML ↔ Intermediate model:
- Generic transformations, with Java
- EMF reflective API

<?xml version="1.0" encoding="UTF-8"?>
  xsi:schemaLocation="http://www.example.com/ssiml http://www.example.com/ssiml/SSIMLMetaModel.xsd">

  <applicationElements xsi:type="ssiml:ApplicationElements">
    <elements xsi:type="ssiml:Scene" id="a52a018e-596c-4437-99b6-6bef63d33ef" name="robotScene">
      <attributes featureId="0" featureName="id" featureType="java.lang.String" classifierId="83">
        <attributeValue value="a52a018e-596c-4437-99b6-6bef63d33ef"/>
      </attributes>
      <attributes featureId="2" featureName="name" featureType="java.lang.String" classifierId="83">
        <attributeValue value="robotScene"/>
      </attributes>
      <childrenContent featureId="1" featureName="children" featureType="java.lang.String" classifierId="83">
        <childrenContent featureId="4" featureType="java.lang.String" classifierId="83">
          <childrenContent featureId="7" featureType="java.lang.String" classifierId="83">
            <childrenContent featureId="10" featureType="java.lang.String" classifierId="83">
              <childrenContent featureId="13" featureType="java.lang.String" classifierId="83">
                <childrenContent featureId="16" featureType="java.lang.String" classifierId="83">
                  <childrenContent featureId="19" featureType="java.lang.String" classifierId="83">
                    <childrenContent featureId="22" featureType="java.lang.String" classifierId="83">
                      <childrenContent featureId="25" featureType="java.lang.String" classifierId="83">
                        <childrenContent featureId="28" featureType="java.lang.String" classifierId="83">
                          <childrenContent featureId="31" featureType="java.lang.String" classifierId="83">
                            <childrenContent featureId="34" featureType="java.lang.String" classifierId="83">
                              <childrenContent featureId="37" featureType="java.lang.String" classifierId="83">
                                <childrenContent featureId="40" featureType="java.lang.String" classifierId="83">
                                  <childrenContent featureId="43" featureType="java.lang.String" classifierId="83">
                                    <childrenContent featureId="46" featureType="java.lang.String" classifierId="83">
                                      <childrenContent featureId="49" featureType="java.lang.String" classifierId="83">
                                        <childrenContent featureId="52" featureType="java.lang.String" classifierId="83">
                                          <childrenContent featureId="55" featureType="java.lang.String" classifierId="83">
                                            <childrenContent featureId="58" featureType="java.lang.String" classifierId="83">
                                              <childrenContent featureId="61" featureType="java.lang.String" classifierId="83">
                                                <childrenContent featureId="64" featureType="java.lang.String" classifierId="83">
                                                  <childrenContent featureId="67" featureType="java.lang.String" classifierId="83">
                                                    <childrenContent featureId="70" featureType="java.lang.String" classifierId="83">
                                                      <childrenContent featureId="73" featureType="java.lang.String" classifierId="83">
                                                        <childrenContent featureId="76" featureType="java.lang.String" classifierId="83">
                                                          <childrenContent featureId="79" featureType="java.lang.String" classifierId="83">
                                                            <childrenContent featureId="82" featureType="java.lang.String" classifierId="83">
                                                              <childrenContent featureId="85" featureType="java.lang.String" classifierId="83">
                                                                <childrenContent featureId="88" featureType="java.lang.String" classifierId="83">
                                                                  <childrenContent featureId="91" featureType="java.lang.String" classifierId="83">
                                                                    <childrenContent featureId="94" featureType="java.lang.String" classifierId="83">
                                                                      <childrenContent featureId="97" featureType="java.lang.String" classifierId="83">
                                                                        <childrenContent featureId="100" featureType="java.lang.String" classifierId="83">
                                                                          <childrenContent featureId="103" featureType="java.lang.String" classifierId="83">
                                                                            <childrenContent featureId="106" featureType="java.lang.String" classifierId="83">
                                                                              <childrenContent featureId="109" featureType="java.lang.String" classifierId="83">
                                                                                <childrenContent featureId="112" featureType="java.lang.String" classifierId="83">
                                                                                  <childrenContent featureId="115" featureType="java.lang.String" classifierId="83">
                                                                                      <childrenContent featureId="118" featureType="java.lang.String" classifierId="83">
                                                                                      </childrenContent>
                                                                                  </childrenContent>
                                                                                </childrenContent>
                                                                              </childrenContent>
                                                                            </childrenContent>
                                                                          </childrenContent>
                                                                        </childrenContent>
                                                                      </childrenContent>
                                                                    </childrenContent>
                                                                  </childrenContent>
                                                                </childrenContent>
                                                              </childrenContent>
                                                            </childrenContent>
                                                          </childrenContent>
                                                        </childrenContent>
                                                      </childrenContent>
                                                    </childrenContent>
                                                  </childrenContent>
                                                </childrenContent>
                                              </childrenContent>
                                            </childrenContent>
                                          </childrenContent>
                                        </childrenContent>
                                      </childrenContent>
                                    </childrenContent>
                                  </childrenContent>
                                </childrenContent>
                              </childrenContent>
                            </childrenContent>
                          </childrenContent>
                        </childrenContent>
                      </childrenContent>
                    </childrenContent>
                  </childrenContent>
                </childrenContent>
              </childrenContent>
            </childrenContent>
          </childrenContent>
        </childrenContent>
      </childrenContent>
    </elements>
  </applicationElements>
</ssiml:SSIMLMetaModel>