The University of Resources. Since 1765.

WEB3D 2012 - 17th International Conference
on 3D Web Technology

Model-Driven Iterative Development of 3D Web-Applications Using SSIML, X3D and JavaScript

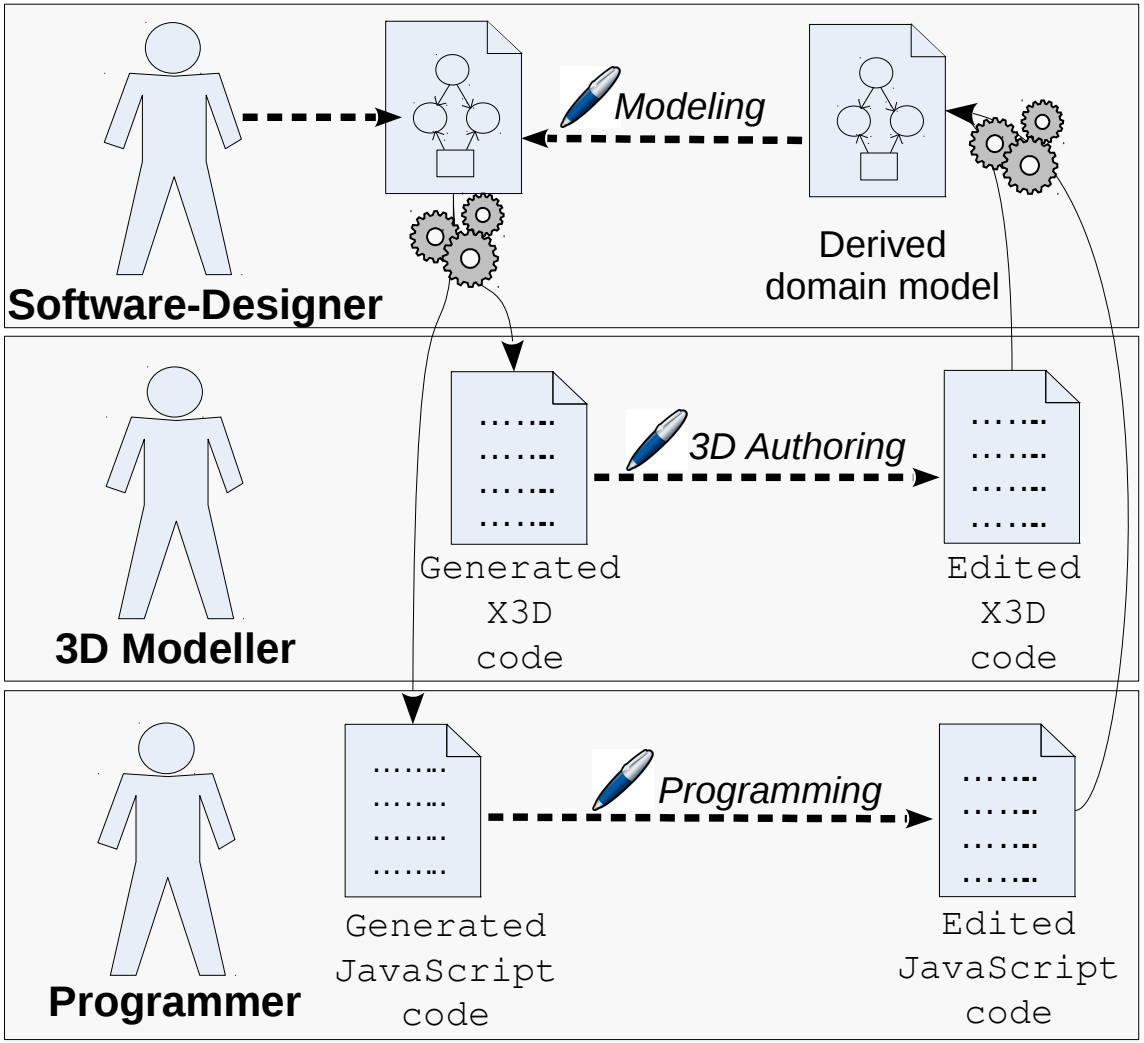
Matthias Lenk Technical University Freiberg, Germany
Arnd Vitzthum University of Cooperative Education Dresden, Germany
Bernhard Jung Technical University Freiberg, Germany

Funded by DFG Deutsche Forschungsgemeinschaft

Table of Contents

1.1 Preliminaries: Motivation

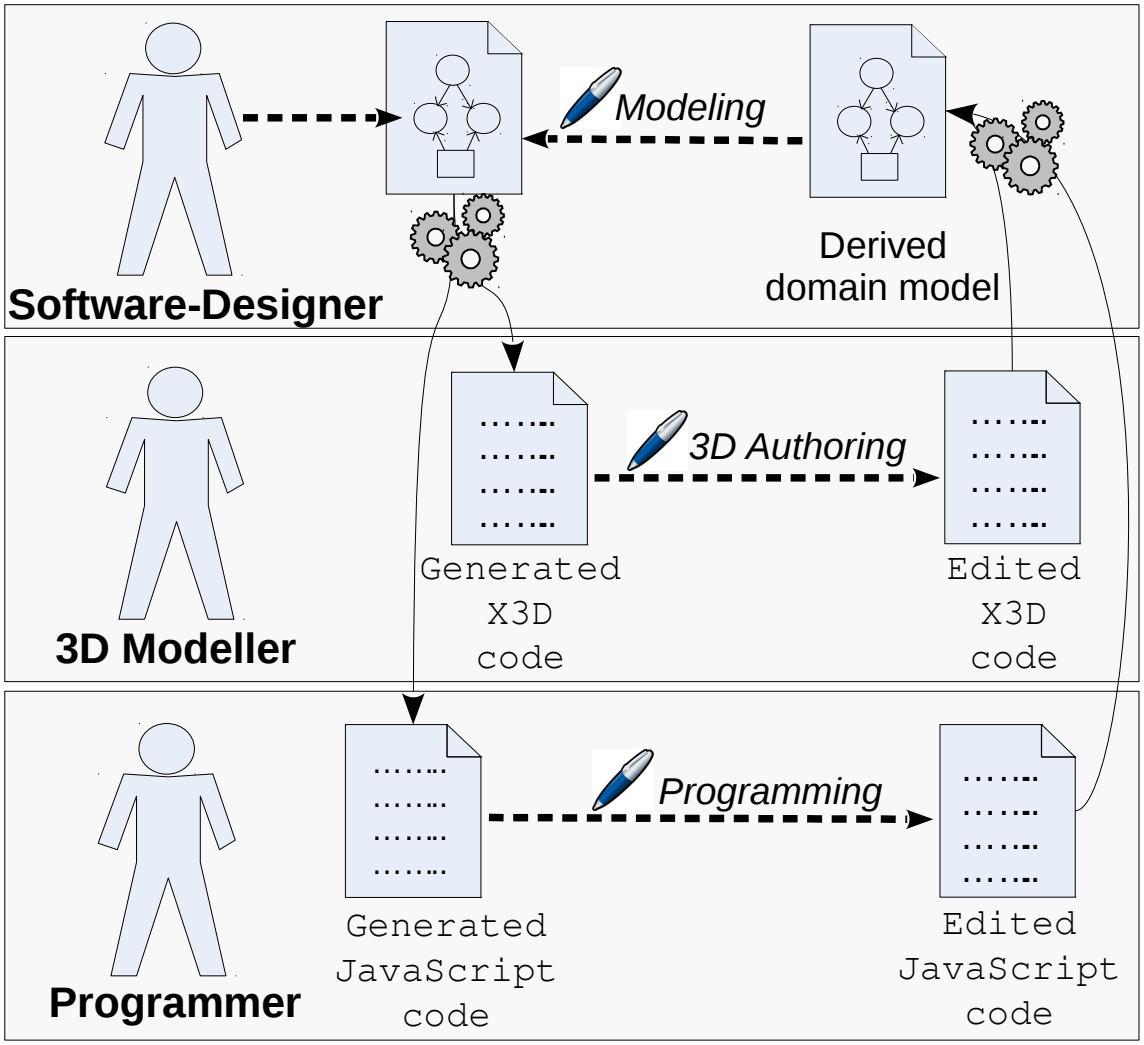
- **Application logic** and **3D content** → **two** different developer groups: 3D modelers, programmers
- E. g. X3D and JavaScript
- Traditionally: code-centric and **iterative** development



1.1 Preliminaries: Motivation

- **Iterative** development
- Improvements with **model-driven** engineering [1] :
  - communication aid
  - contract
  - code generation
  - ➔ **concurrent development**

→ **Round-trip engineering** [2]



$\sum$

Forward engineering

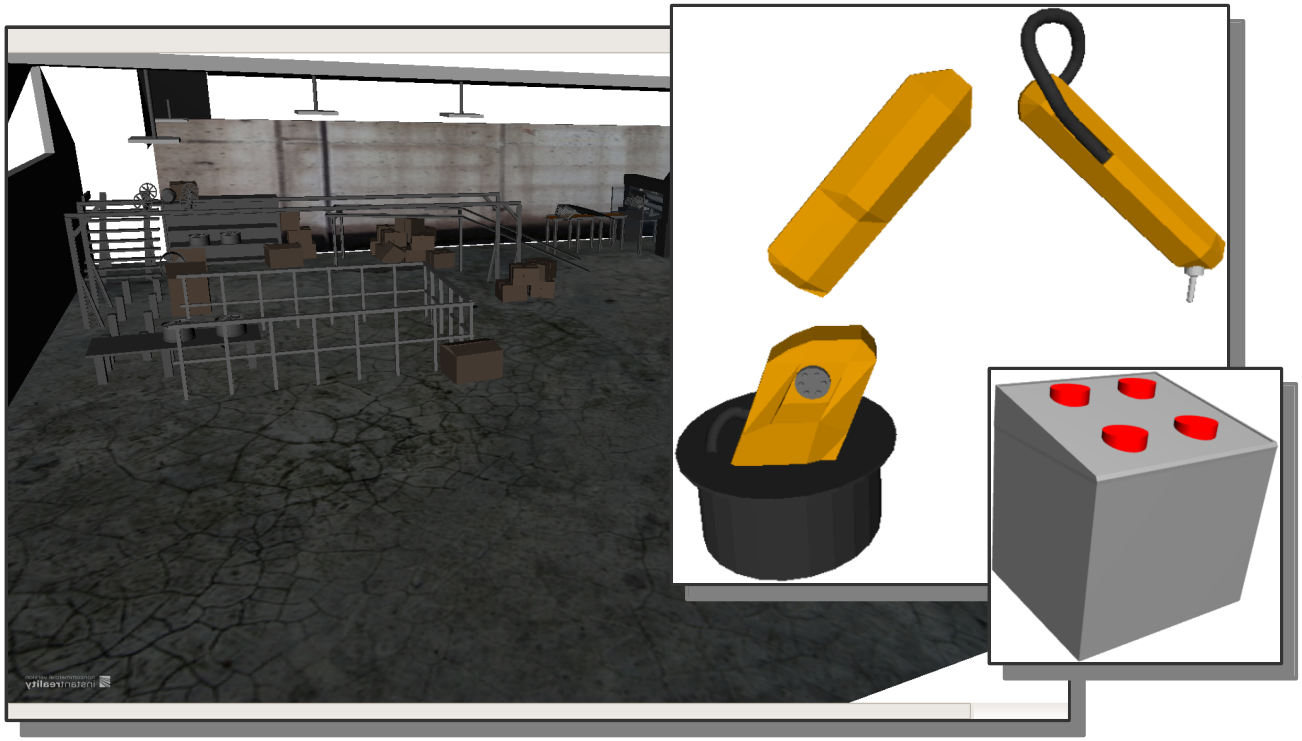
Reverse engineering

Preserve modifications

Table of Contents

1.2 Preliminaries: Running example

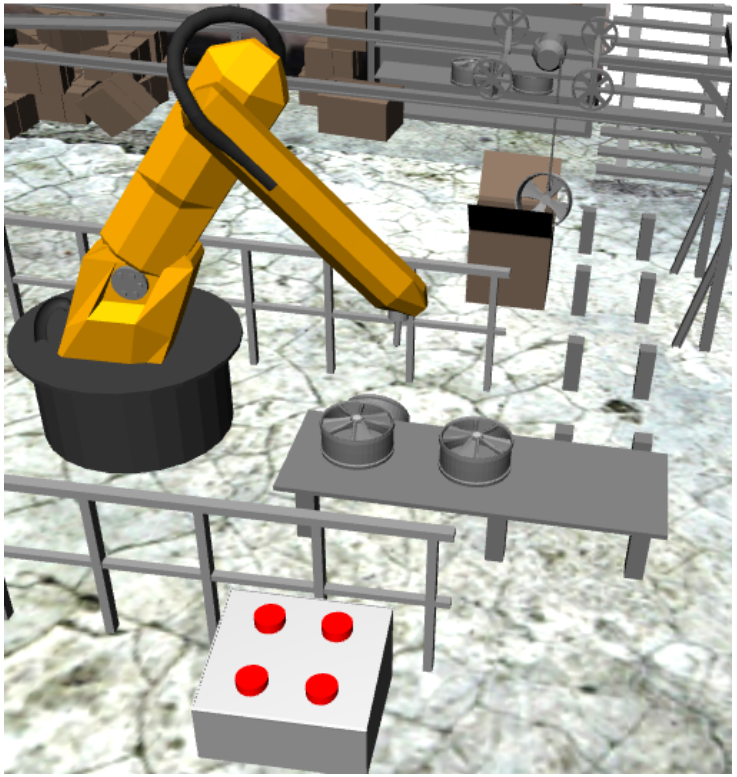
- **3D content** and **application code** → multiple target languages (TL): X3D/ X3DOM, JavaScript
- Example application:



```
window.onload = init;

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

    this.button0_CLICKED = function( obj ) {
        var rot = transform0.getAttribute("rotati
        var angle = parseFloat(rot.split(" ")[3],
        angle += 0.04;
        transform0.setAttribute("rotation", "1 0
    }
    this.button1_CLICKED = function( obj ) {
    var angle = parseFloat(transform0.getAttribut
        angle -= 0.04;
        transform0.setAttribute("rotation", "1 0
    }
    this.button2_CLICKED = function( obj ) {
        var angle = parseFloat(transform1.getAttr
        angle += 0.04;
        transform1.setAttribute("rotation", "1 0
    }

    this.button3_CLICKED = function( obj ) {
        var angle = parseFloat(transform1.getAttr
        angle -= 0.04;
        transform1.setAttribute("rotation", "1 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



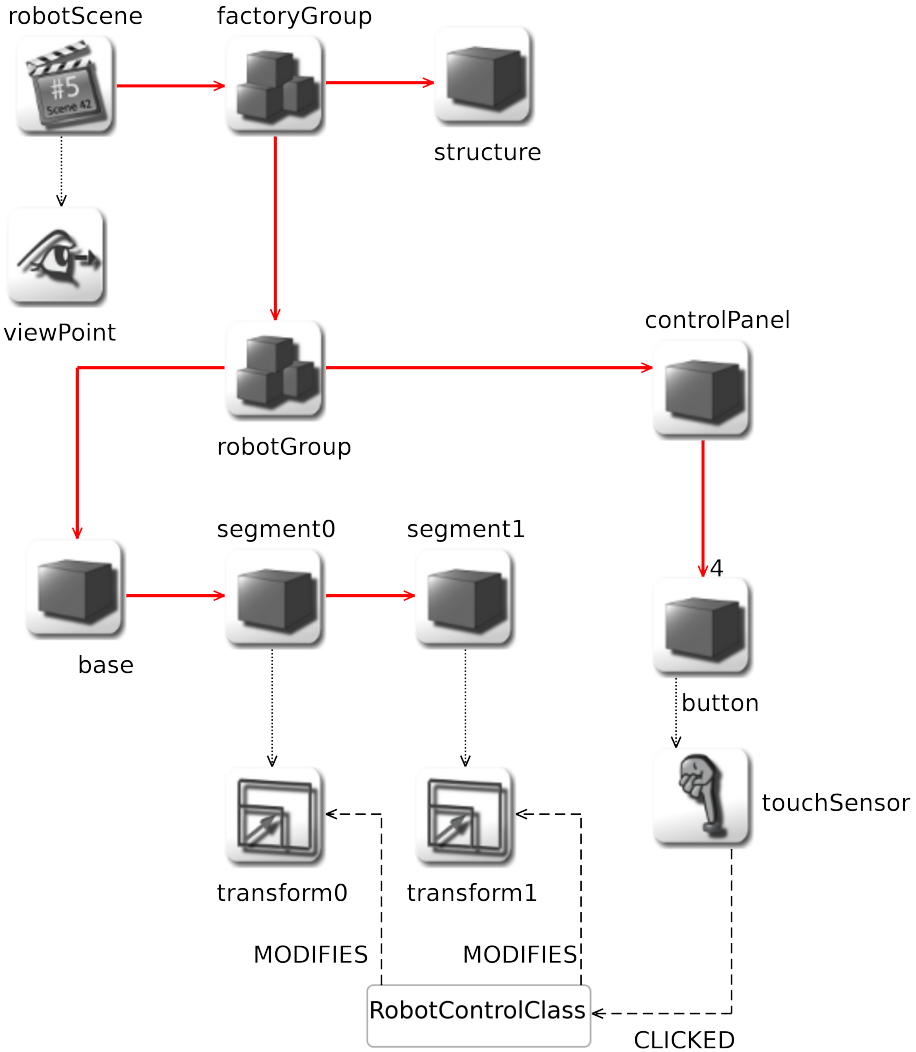


Table of Contents

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

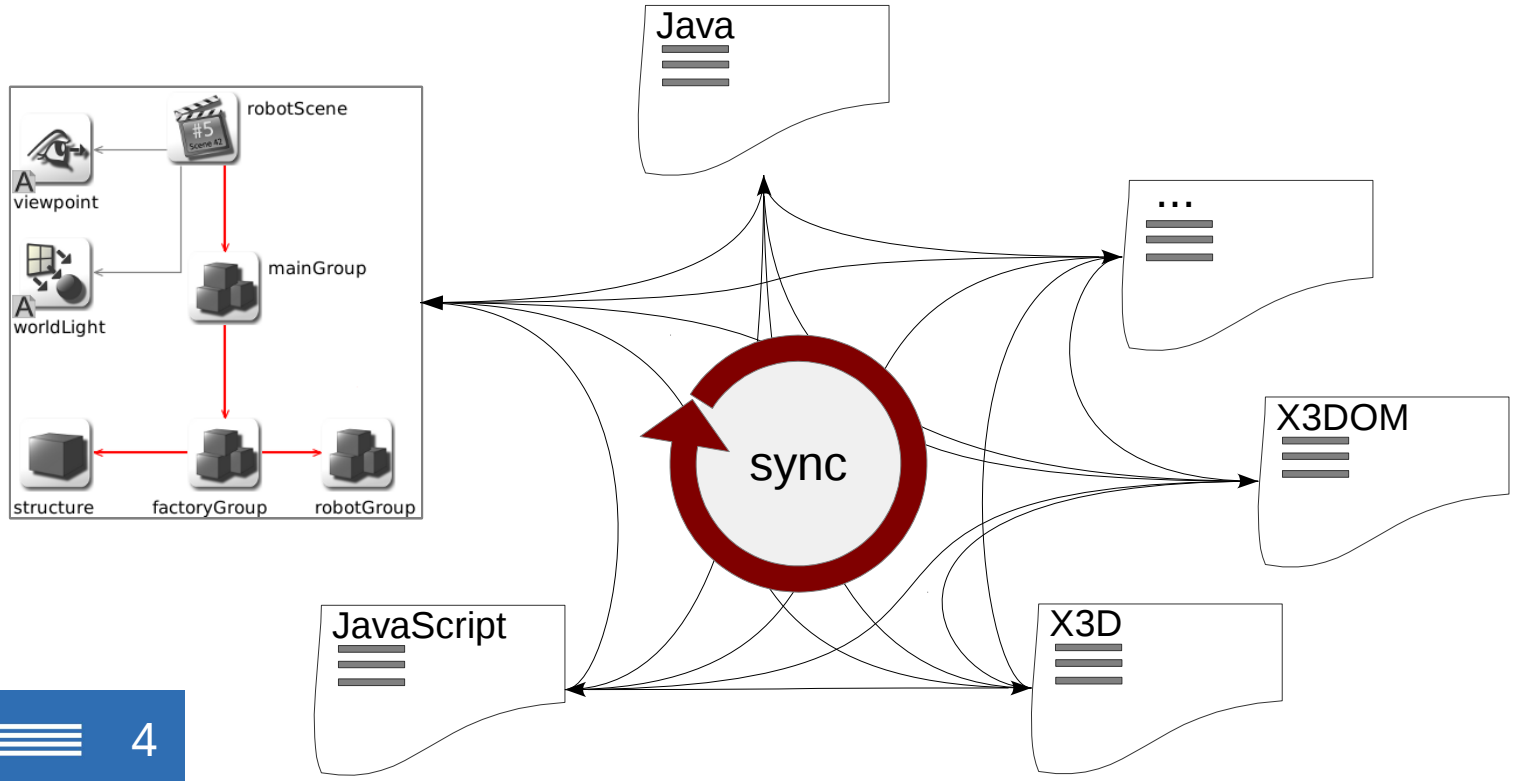
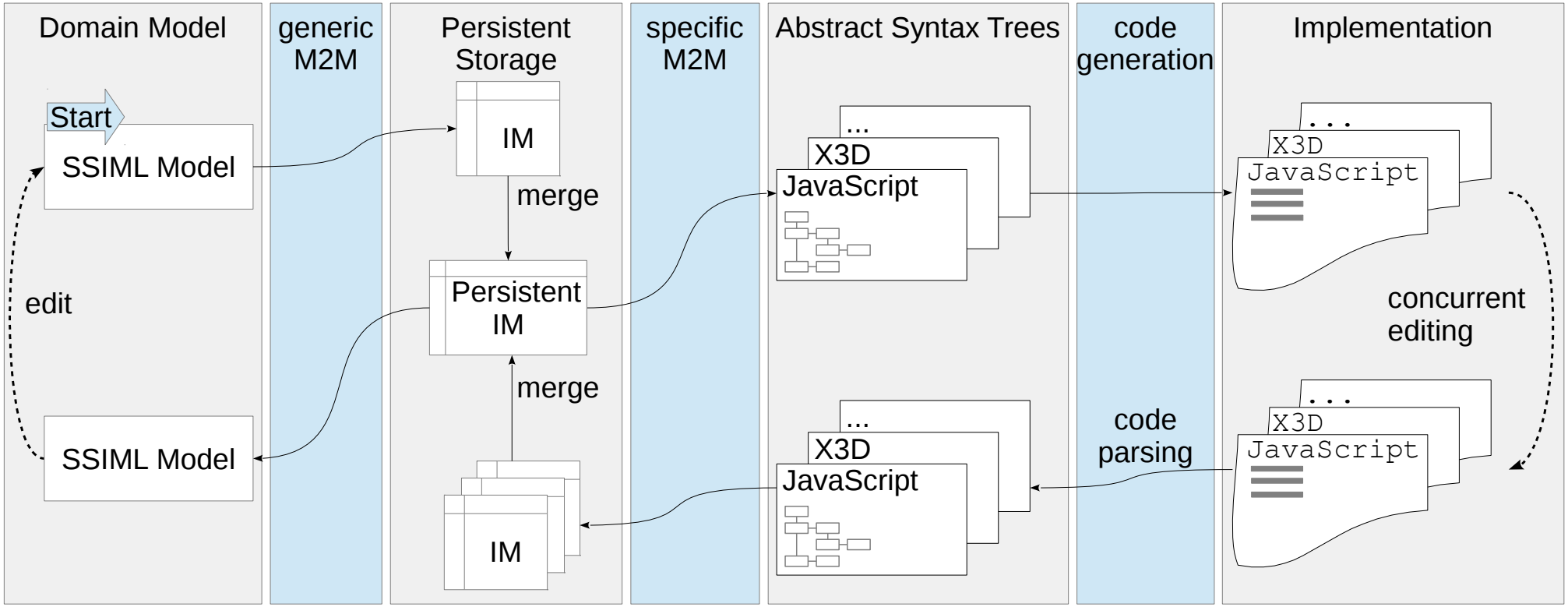


Table of Contents

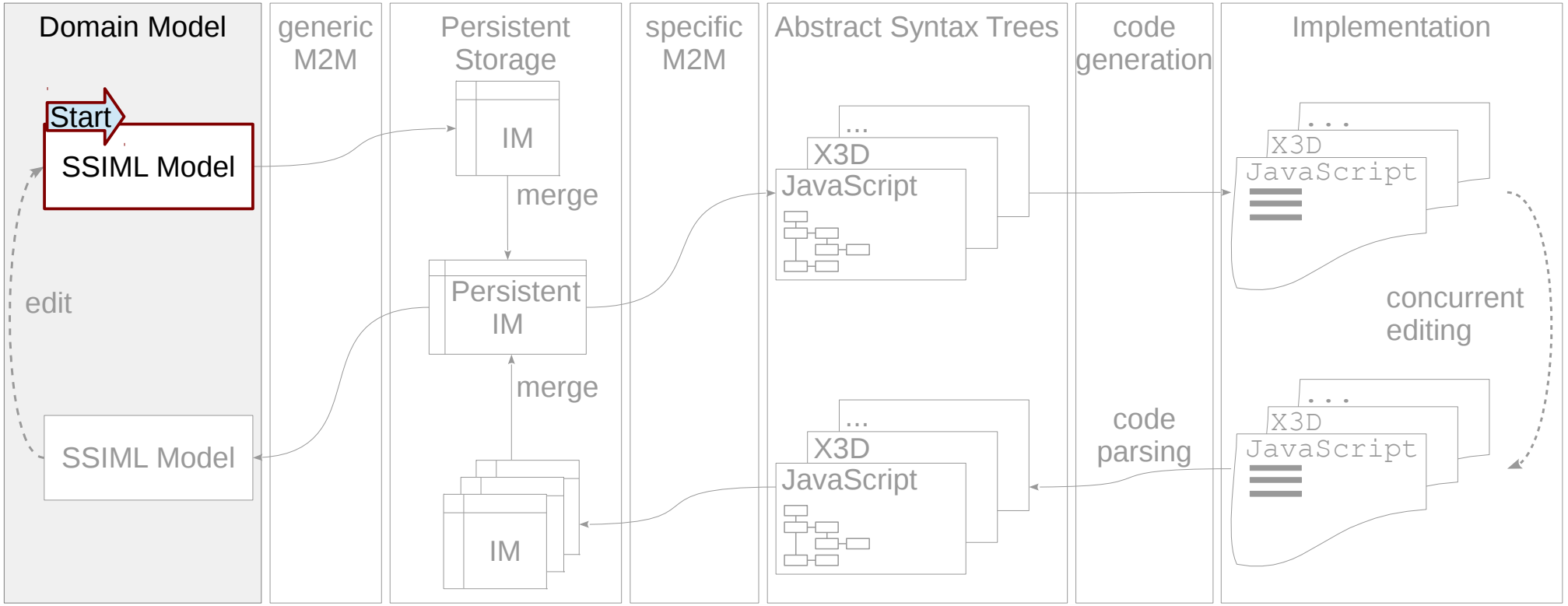
2.2 Roundtrip3D: Overview RTE Approach

- One common **intermediate** representation for languages and model
- Managable transformations:
  - **Multi-tiered** approach
    - **Task specific** transformation languages



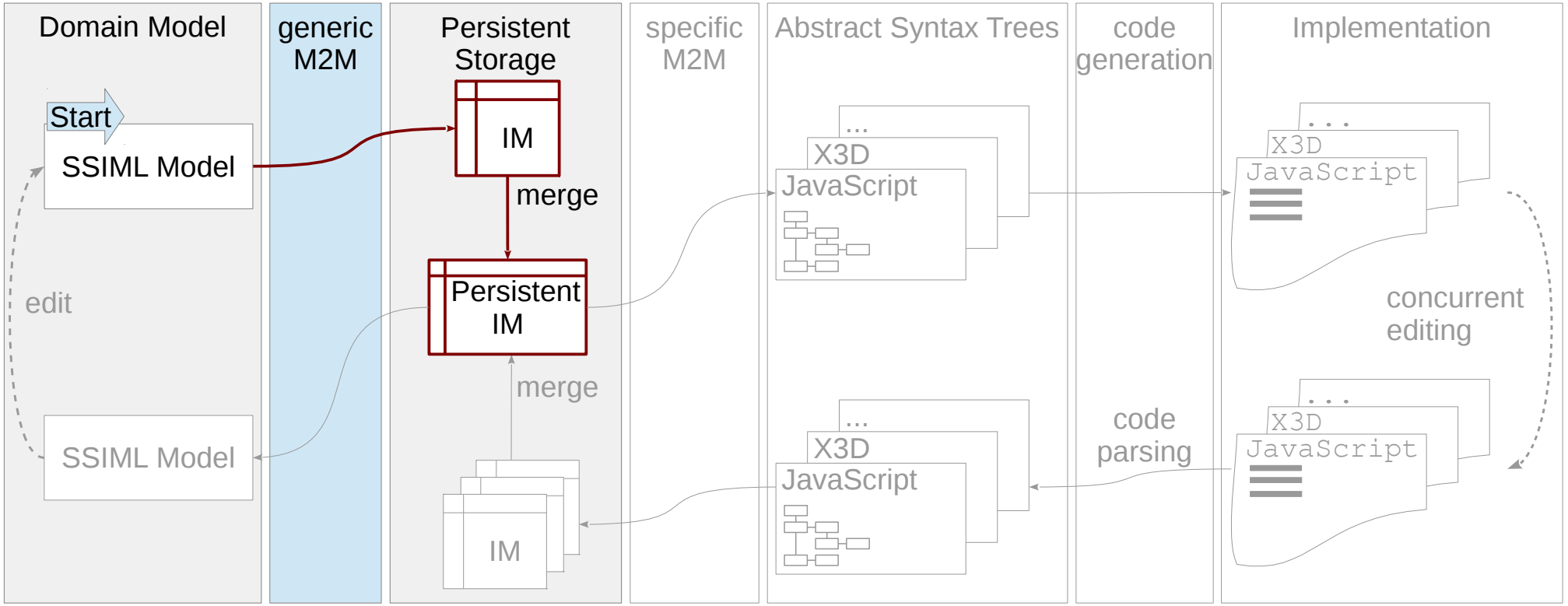
2.2 Roundtrip3D: Overview RTE Approach

- Modeling 3D applications with SSIML



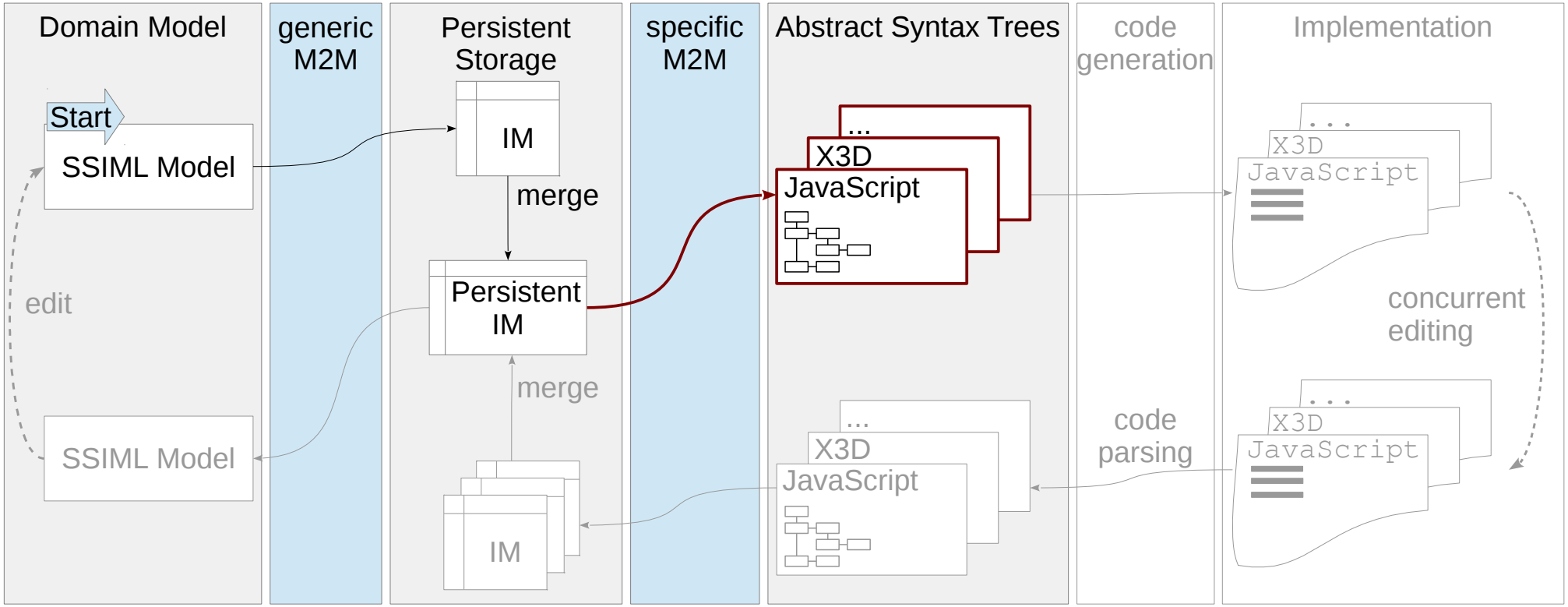
2.2 Roundtrip3D: Overview RTE Approach

- Model-based generic storage in an intermediate model (IM)
- (Merge intermediate model with persistent intermediate model from previous iterations)?



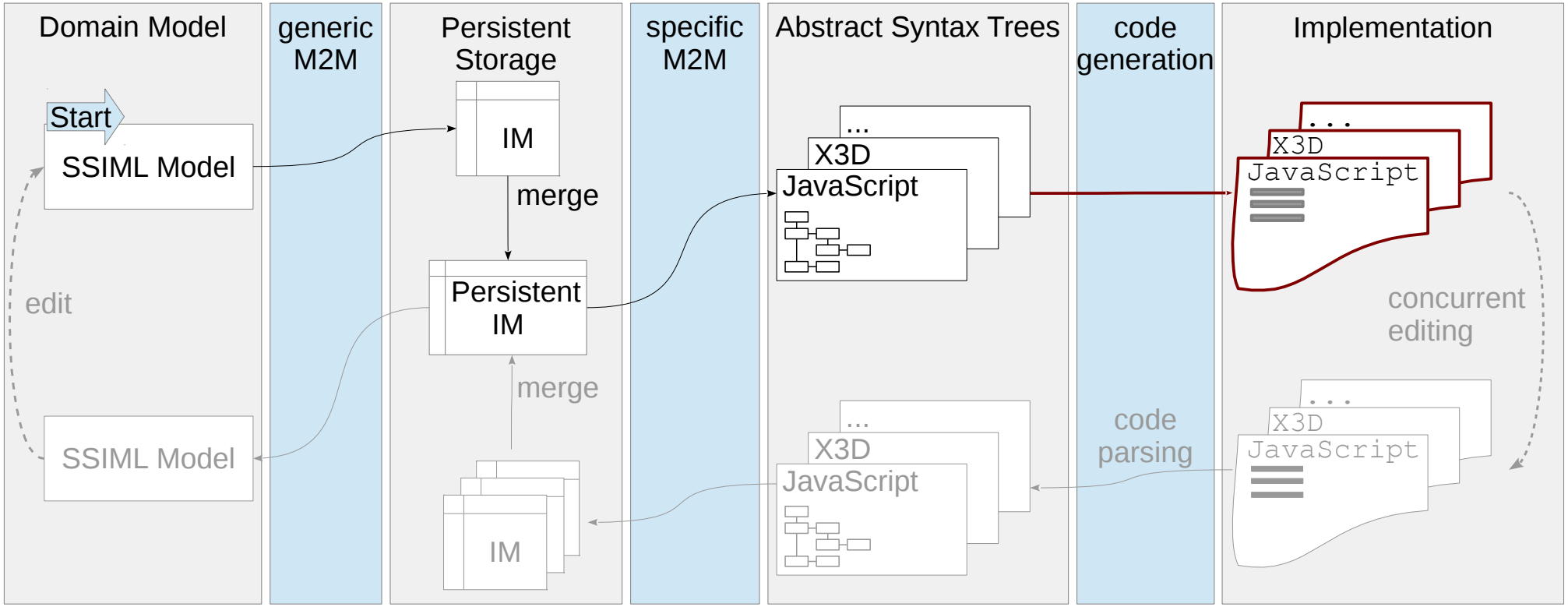
2.2 Roundtrip3D: Overview RTE Approach

- Build abstract syntax trees (ASTs) from intermediate model



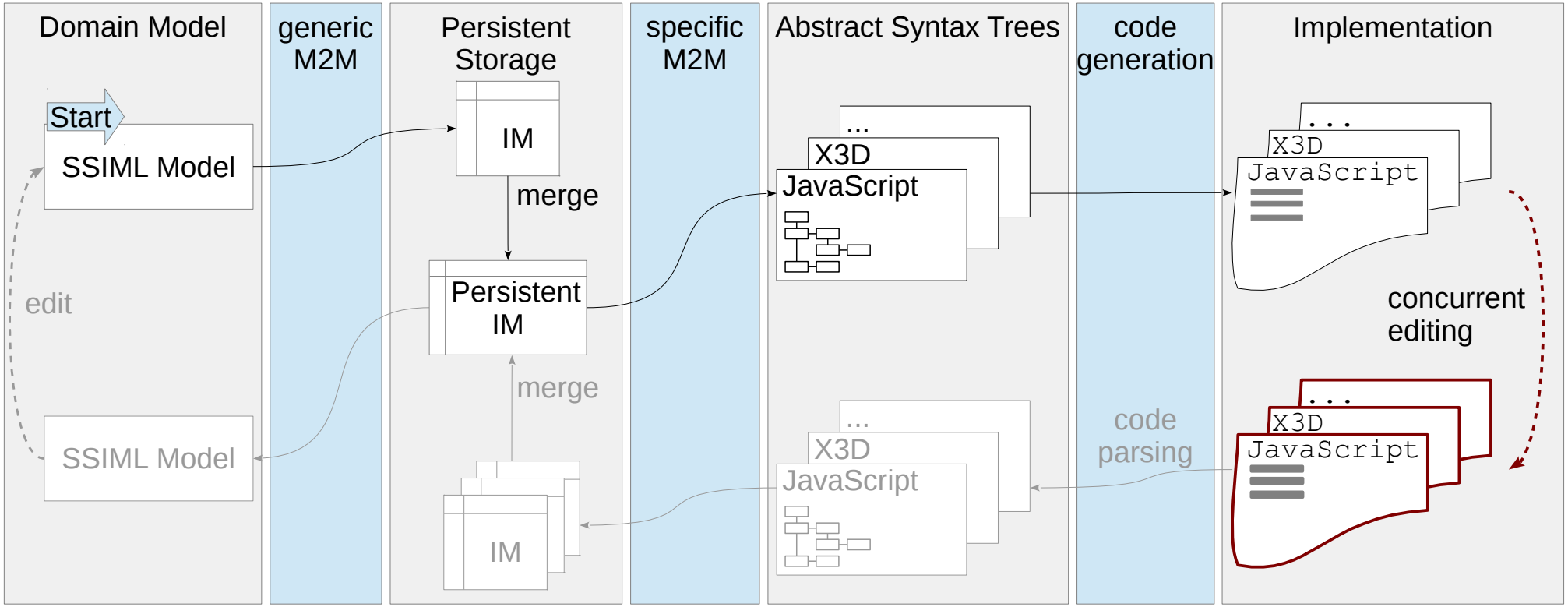
2.2 Roundtrip3D: Overview RTE Approach

- Code generation for each target language from abstract syntax trees



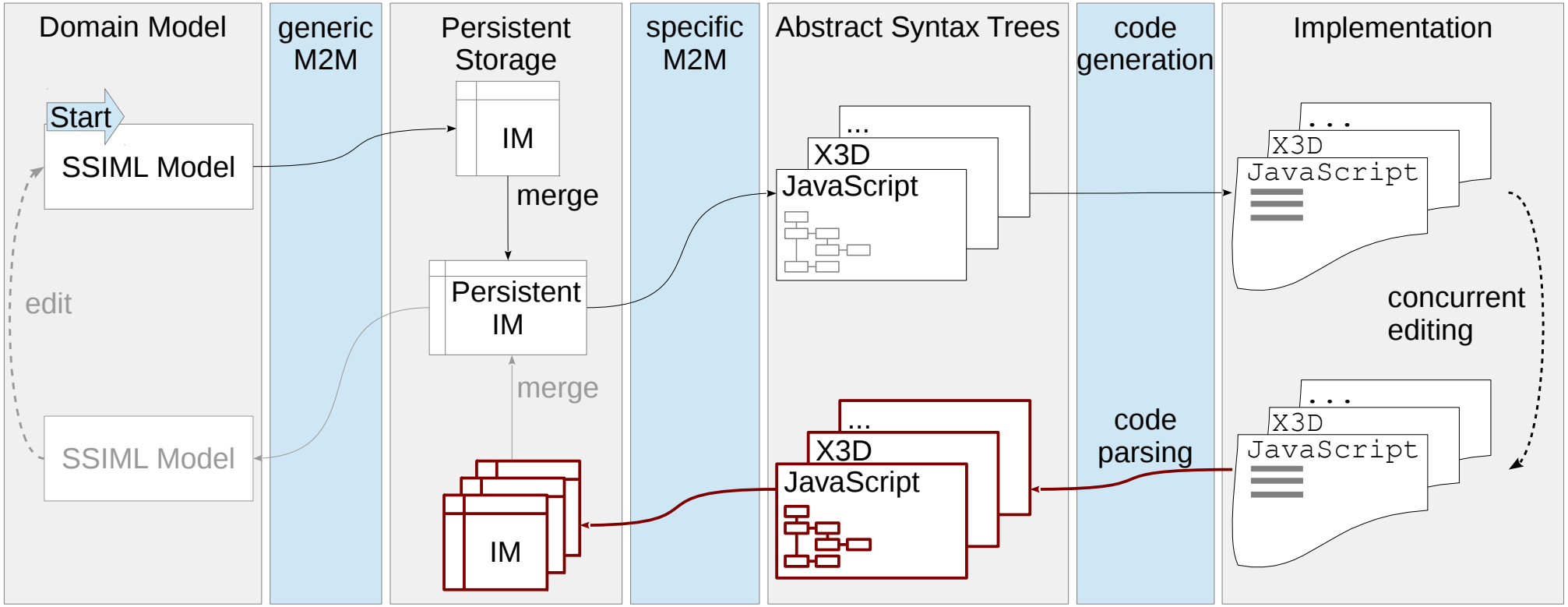
2.2 Roundtrip3D: Overview RTE Approach

- **Concurrent** editing of code bases, i. e. 3D design and implementation of application logic



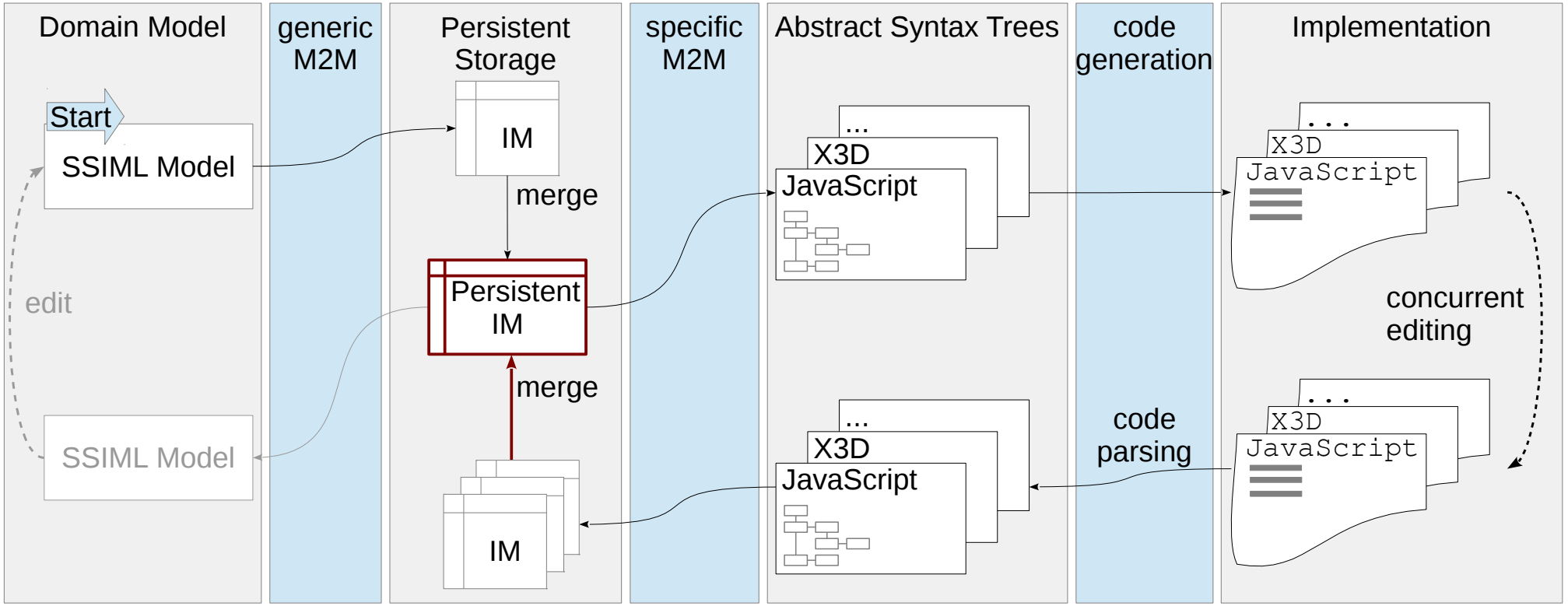
2.2 Roundtrip3D: Overview RTE Approach

- Reverse steps: Code parsing, built abstract syntax trees
- Transformation of each abstract syntax tree to intermediate model



2.2 Roundtrip3D: Overview RTE Approach

- 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

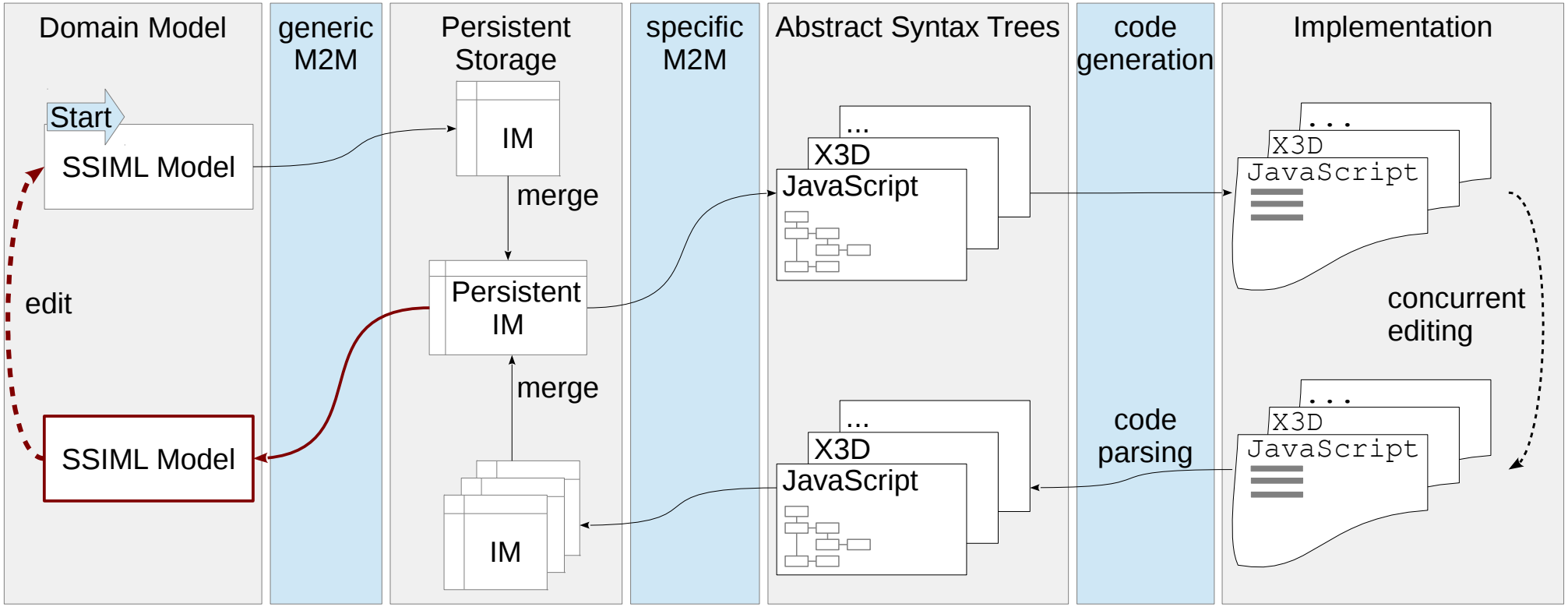
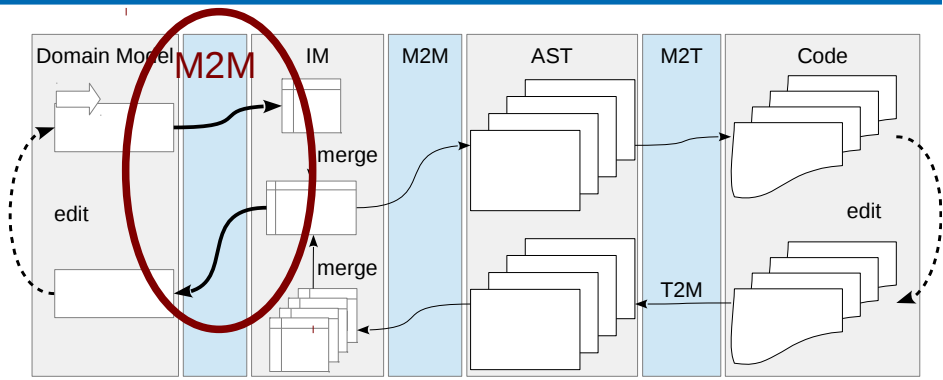
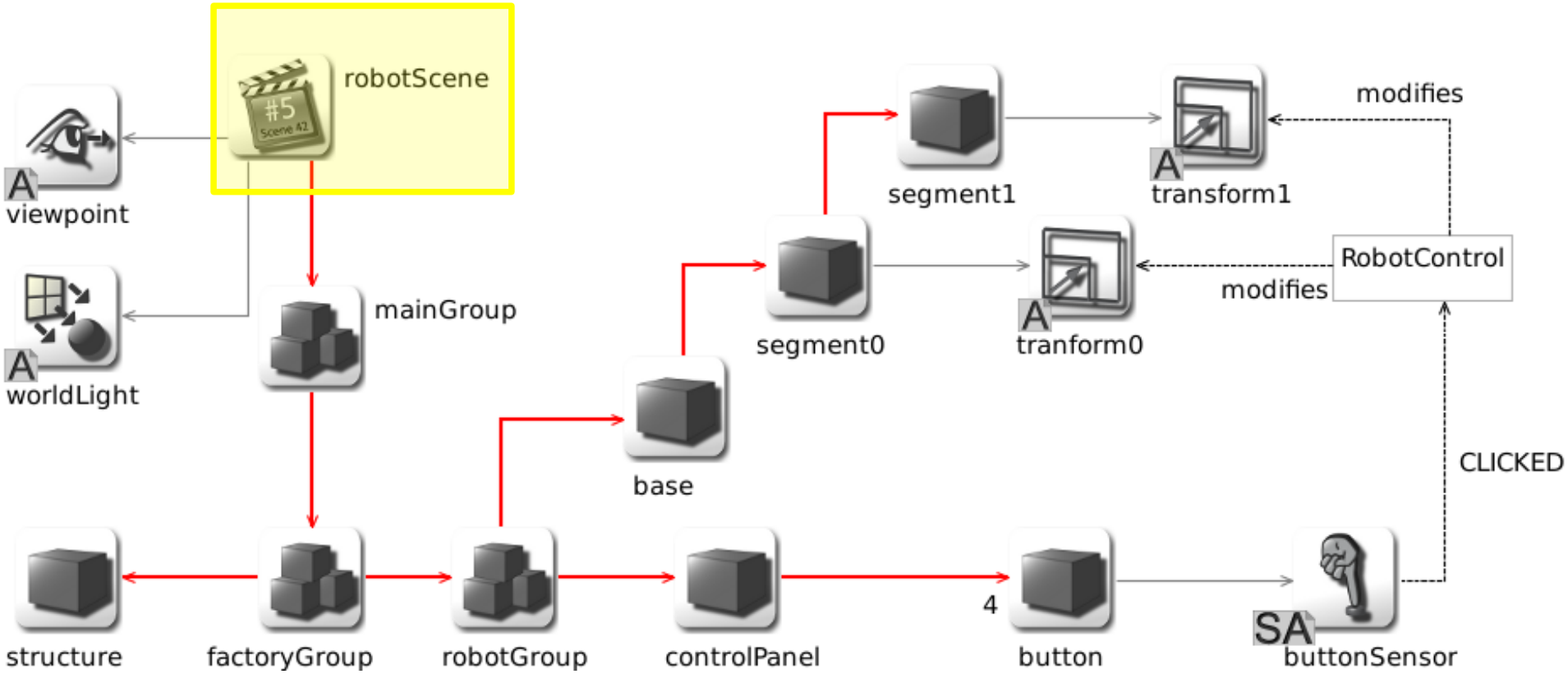


Table of Contents

2.3 Roundtrip3D: SSIML → Intermediate model

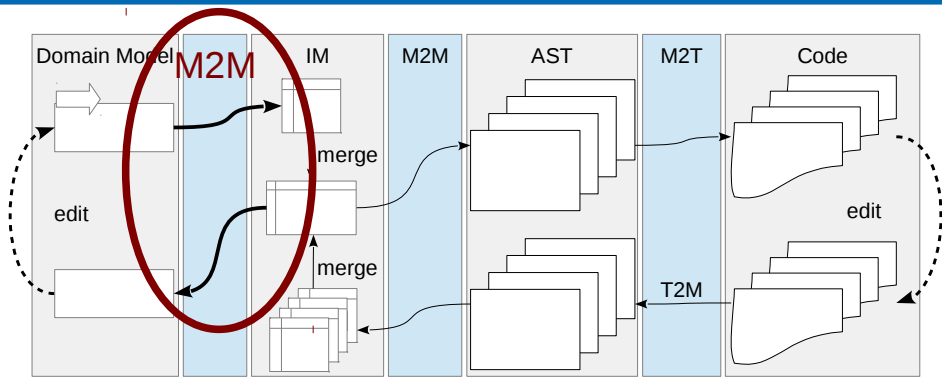
- Generic transformations, with Java
- EMF reflective API





2.3 Roundtrip3D: SSIML → Intermediate model

- Generic transformations, with Java
- EMF reflective API

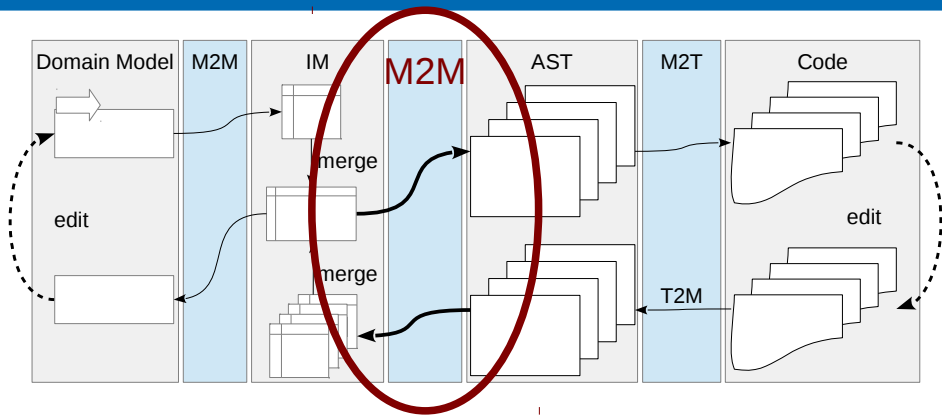


- SSIML Model
  - Value Access Relationship
  - Touch Sensor Event Relationship
  - Value Access Relationship
  - Value Access Relationship
  - Application Class RobotControl
  - Scene robotScene
  - Group factoryGroup
  - Object structure
  - Primary Parent Child Relationship
  - Group robotGroup
  - Primary Parent Child Relationship
  - Object base
  - Object segment0
  - Object segment1

- Model
  - Object ValueAccessRelationship
  - Object TouchSensorEventRelationship
  - Object ValueAccessRelationship
  - Object ValueAccessRelationship
  - Object ApplicationClass
  - Object Scene
    - Attribute id
    - Attribute name
    - Attribute model
    - Attribute attributes
    - Attribute childContent
  - Object Group
  - Object Object

2.3 Roundtrip3D: Intermediate model ↔ AST

- Rule based, hybrid transformation language
- External mapping model [4]



- Mapping Model SSIML to X3D
  - Source Element SSIMLModel
    - Target Element X3D
  - Source Element Group
    - Target Element Group
  - Source Element Transformation
    - Target Element Transform
  - Source Element Object
    - Target Element Inline

+

- Model
  - Object Transformation
    - Attribute id
    - Attribute name
    - Attribute angleAxis
    - Attribute scale
    - Attribute translation
    - Attribute model
    - Attribute owner
    - Attribute actionSources
    - Attribute successor
    - Attribute predecessor
  - Object Scene
  - Object Group
  - Object Object

+

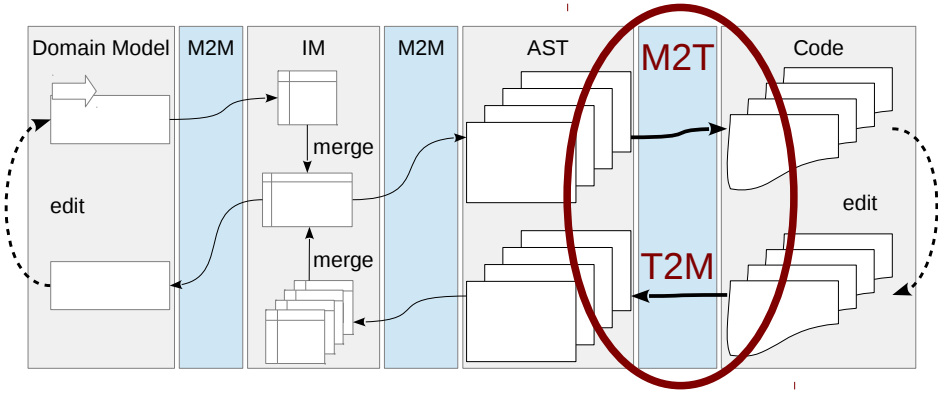
```
rule SSIML_Node_to_X3D_Node
    transform source : IM!IntermetiateObject
    to target : Any
{
    ...
    target.name = source.getMapping();
    target.id   = source ...;
    ...
}
```

- X3D X3D
  - X3D Attribute version
  - X3D Attribute profile
  - Scene Scene
    - Transform Transform
      - X3D Attribute DEF
      - X3D Attribute translation
      - X3D Attribute rotation
      - X3D Attribute scale
      - X3D Attribute center
      - X3D Attribute scaleOrientation
    - Group Group
      - X3D Attribute DEF
      - Transform Transform
      - Transform Transform
    - Viewpoint Viewpoint
      - X3D Attribute DEF
      - X3D Attribute position
      - X3D Attribute orientation

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



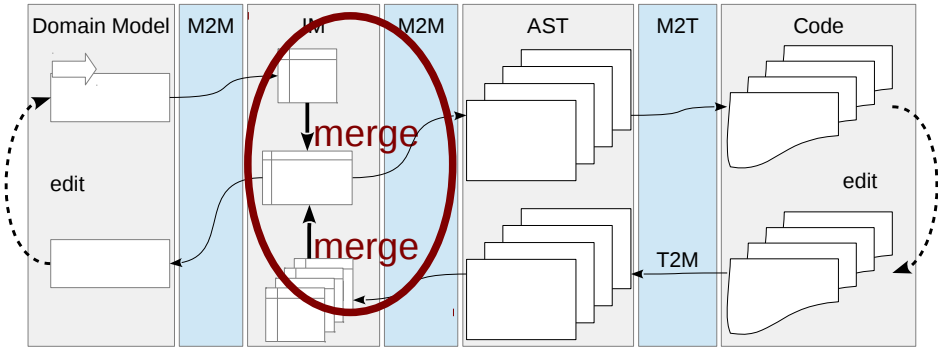
X3D code skeleton

```
<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>
    </Transform>
    ...
      <Inline DEF="button0" url="button.x3d">
      </Inline>
    ...
  </Viewpoint>
 </Scene>
</X3D>
```

JavaScript code skeleton

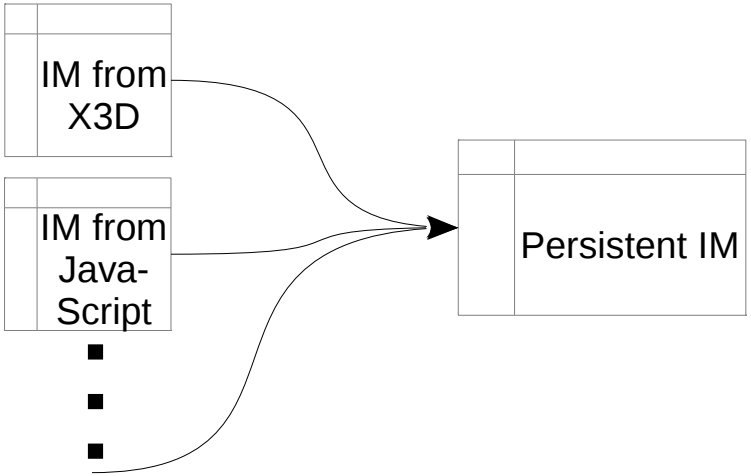
```
/**
 * @id: ApplicationClass 381af861-5437-403d-9293-d3807e441005
 */
function RobotControlClass( ) {
    ...
    var transform0 = document.querySelector( '[DEF="transform0"]' );

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

/**
 * @id: Init 1218cda5-76ca-438f-9b49-4c68219a0637
 */
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

1. Comparison: Operations: Update, Insert, Delete (, Move) [5]
2. Conformance checking
3. Interactive merging: **one** merge algorithm
4. Reconciliation



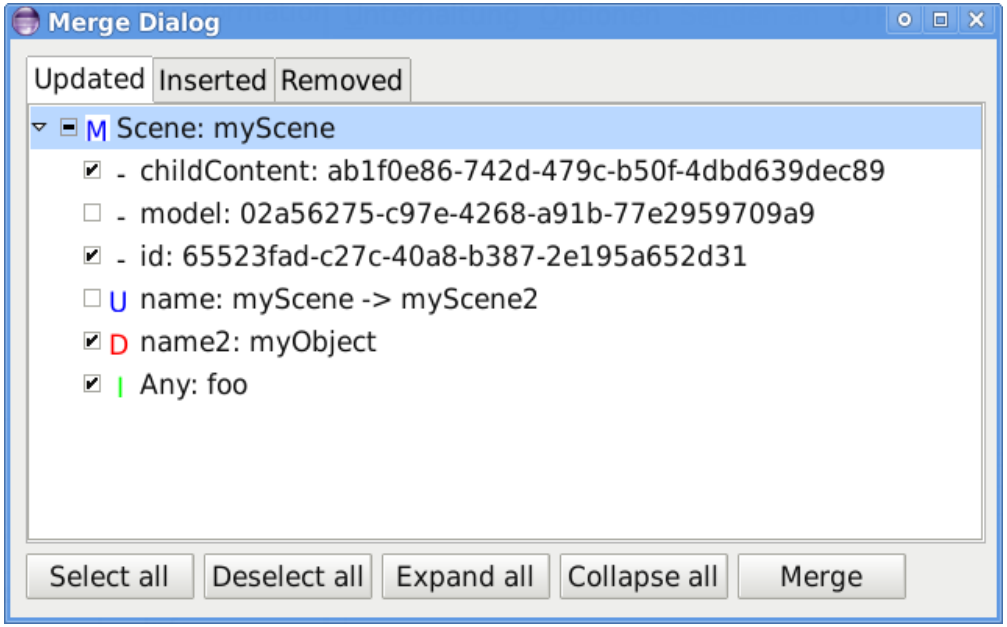
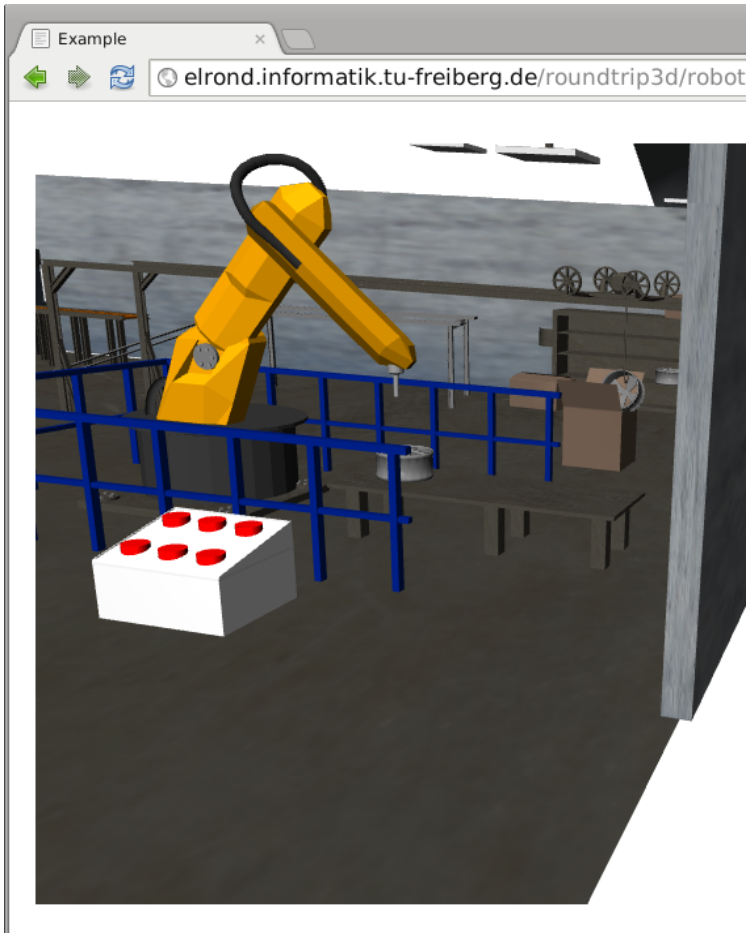


Table of Contents

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

Table of Contents

3.2 Conclusion: Outlook

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

Questions

Thank You!

mail: lenk@informatik.tu-freiberg.de

References

[1] VITZTHUM , A. 2008. Entwicklungsunterstützung für interaktive 3D-Anwendungen. Ein modellgetriebener Ansatz. Dissertation, Ludwigs-Maximilians-University München. In German.

[2] VITZTHUM , A., AND JUNG , B. 2010. Iterative model driven VR and AR development with round trip engineering. In Proc. SEARIS Workshop at the IEEE Virtual Reality 2010 Conference, Shaker.

[3] FIGUEROA , P., GREEN , M., AND HOOVER , H. J. 2002. Intml: a description language for vr applications. In Web3D '02: Proceedings of the seventh international conference on 3D Web technology, ACM, New York, NY, USA, 53–58.

[4] MILLER , J., AND MUKERJI , J. 2003. Mda guide version 1.0.1. Tech. rep., Object Management Group (OMG).

[5] CHAWATHE, S.S., RAJARAMAN, A., GARCIA-MOLINA, H., WIDOM, J.: Change detection in hierarchically structured information. SIGMOD Rec. 25, 493–504 (June 1996)

[6] ROSE, L.M., PAIGE, R.F., KOLOVOS, D.S., POLACK, F.: The epsilon generation language. In: Schieferdecker, I., Hartman, A. (eds.) Model Driven Architecture - Foundations and Applications, 4th European Conference, ECMDA-FA 2008, Berlin, Germany, June 9-13, 2008. Proceedings.
Lecture Notes in Computer Science, vol. 5095, pp. 1–16. Springer (2008)

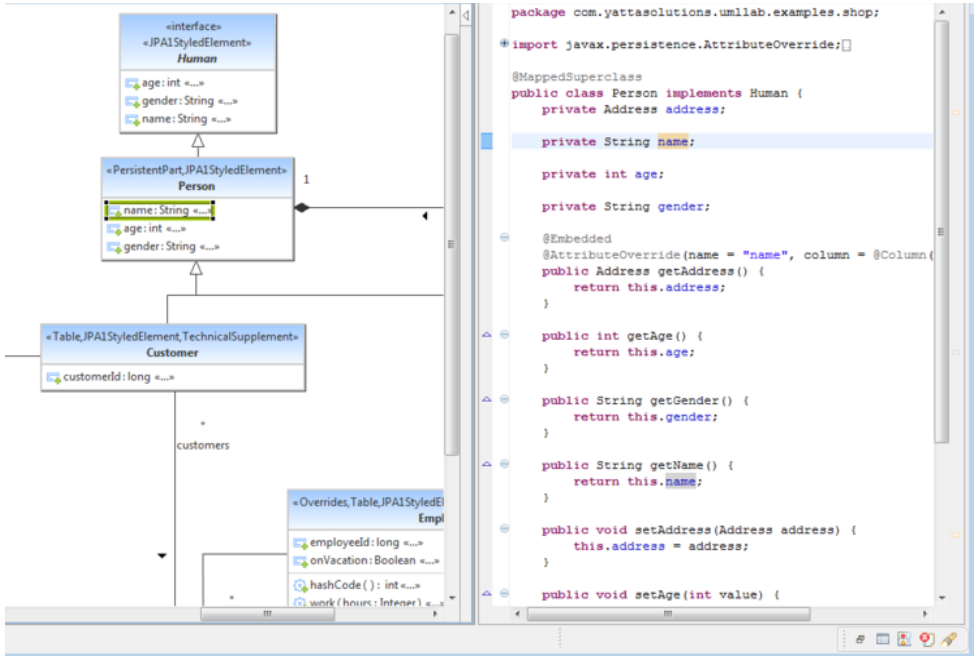
[7] OLDEVIK, J.: MOFScript Eclipse Plug-In: Metamodel-Based Code Generation. In: Proceedings of the Eclipse Technology eXchange workshop (eTX) (2006)

[8] JONES, J.: Abstract syntax tree implementation idioms. Pattern Languages of Program Design (2003), proceedings of the 10th Conference on Pattern Languages of Programs (PLoP2003)

Backup: Related work

Round-trip tools:

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



source: http://www.uml-lab.com/de/uml-lab

Tools for high level 3D development:

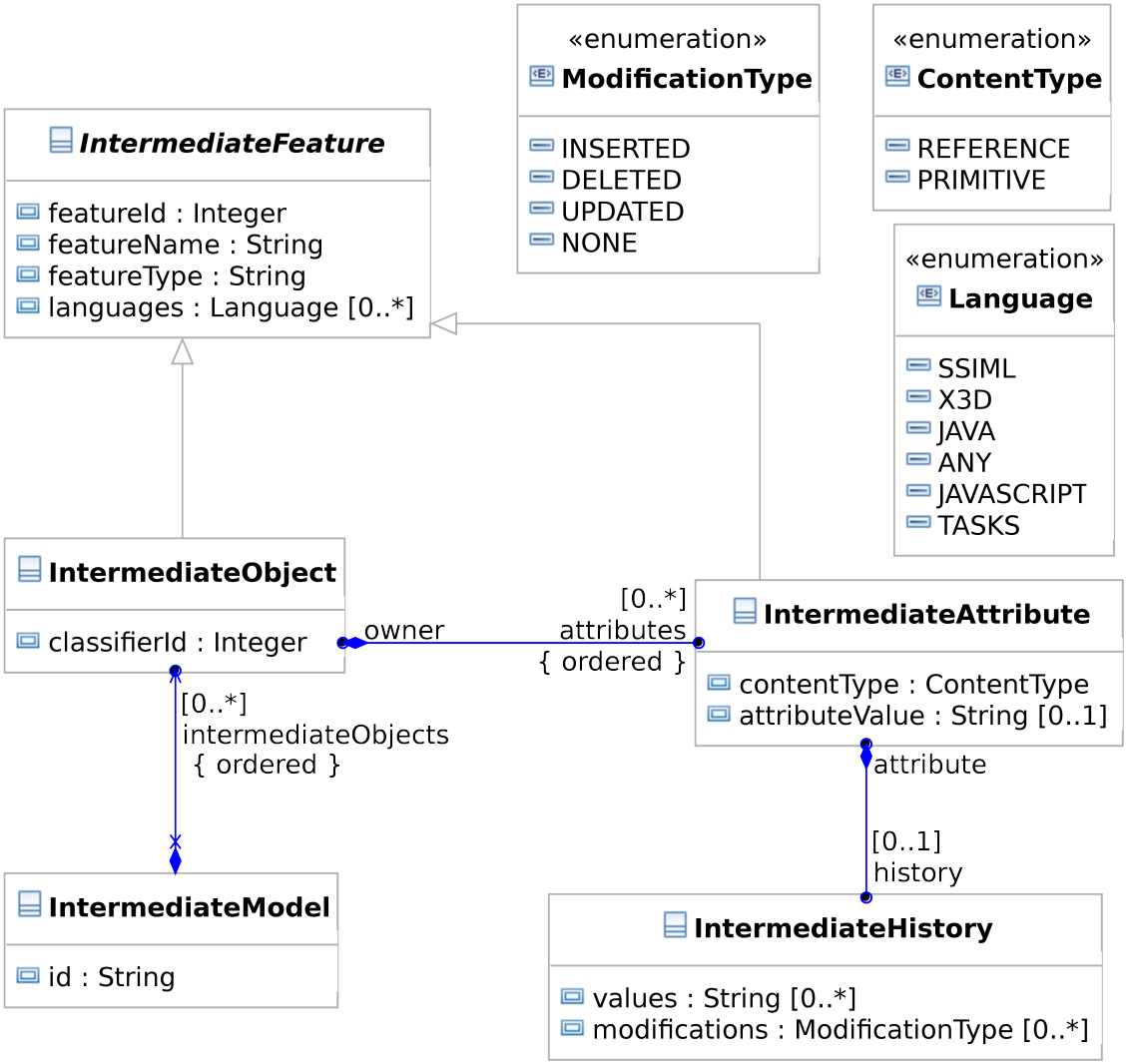
- High level dev. hardly supported
- Simplify programming task
- Rapid prototyping
- E. g. InTML [3] : describe different input devices, output devices, virtual objects, and 3D interaction techniques at high level, XML-based

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

Backup: Intermediate Model

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



Backup: Eclipse Framework and Editor

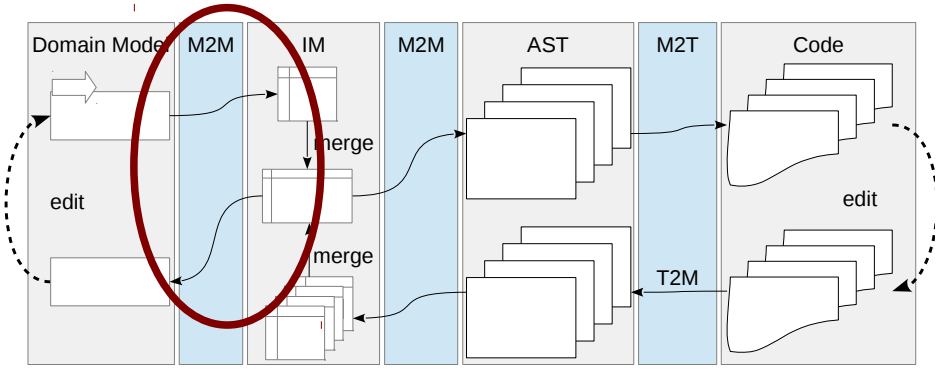
Eclipse Modeling Tools:

- (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"?>
<ssiml:SSIMLModel xmi:version="2.0" xmlns:xmi:
  <interrelations xsi:type="ssiml:ValueAccess
  <interrelations xsi:type="ssiml:TouchSensor
  <interrelations xsi:type="ssiml:ValueAccess
  <applicationElements xsi:type="ssiml:Applic
  <elements xsi:type="ssiml:Scene"
    id="a52a010e-596c-4437-99b6-6bef63d338ef"
    name="robotScene"
    attributes="//@elements.23"
    childContent="//@elements.21"/>
  <elements xsi:type="ssiml:Group" id="404ed.
  <elements xsi:type="ssiml:Object" id="4a31
  <elements xsi:type="ssiml:PrimaryParentChil
  <elements xsi:type="ssiml:Group" id="8bf539
  <elements xsi:type="ssiml:PrimaryParentChil
  <elements xsi:type="ssiml:Object" id="3e162
  <elements xsi:type="ssiml:Object" id="4e3af
```

```
<intermediateObjects featureId="6"
  featureName="elements" featureType="Scene" classifierId="83">
  <languages>SSIML</languages>
  <attributes featureId="0" featureName="id"
    featureType="java.lang.String" contentType="PRIMITIVE"
    attributeValue="a52a010e-596c-4437-99b6-6bef63d338ef">
    <languages>SSIML</languages>
  </attributes>
  <attributes featureId="2" featureName="name" featureType="java
    <languages>SSIML</languages>
  </attributes>
  <attributes featureId="1" featureName="model" featureType="ER
    <languages>SSIML</languages>
  </attributes>
  <attributes featureId="4" featureName="attributes" featureTyp
    <languages>SSIML</languages>
  </attributes>
  <attributes featureId="6" featureName="childContent" featureT
    <languages>SSIML</languages>
  </attributes>
</intermediateObjects>
```