Designing user interface for FTSyn

Presented By: Maulik Patel

Problem Statement

- Designing a user interface for Automatic Addition of fault tolerance.

- Fault-Tolerance Synthesizer (FTSyn) which is a framework for adding fault tolerance to parallel/distributed programs.

- Develop a GUI tool for FTSyn that generates FTSyn input file from UML diagram file.

FTSyn

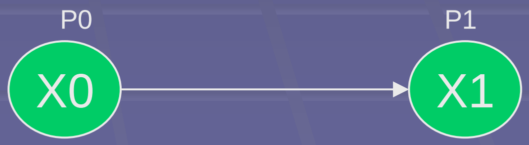
- Developed to allow users to automatically add fault-tolerance.
- The input consist of:
  - Fault-intolerant program
  - Invariant
  - Safety specification
  - Initial state
  - Class of faults

FTSync

- Dijkstra's guarded commands are used for input fault-intolerant program.
  - *g* → *st*
  - *g*: state predicate *st*: statement

- Example:

```
process  P1
begin
        (X0 != 0) ⟶ (X1= X0 % 2);
read  X0, X1;
write  X1;
end
```



FTSync

- Faults are modeled as a set of guarded commands.

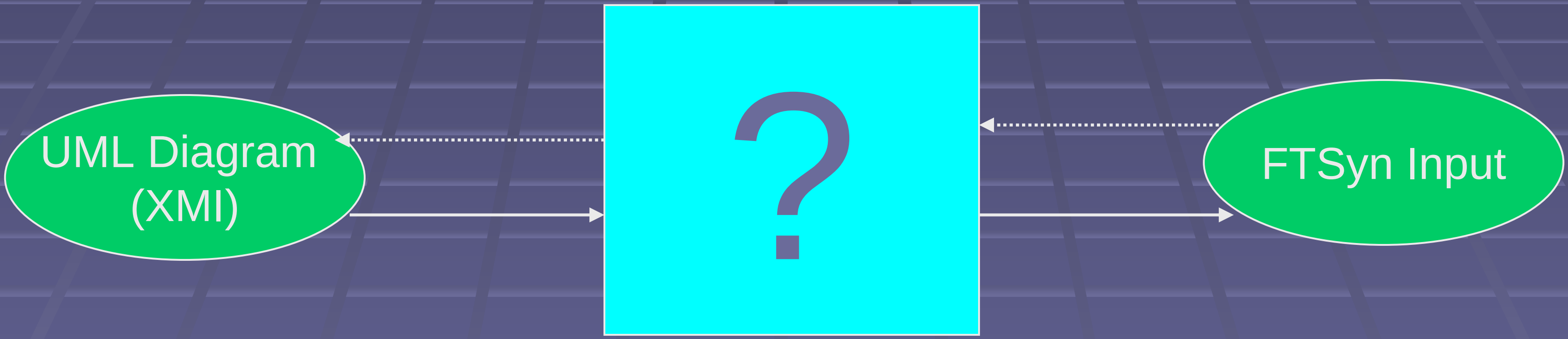
```
fault  process_corruption
begin
        (X1 > 10) ⟶ (X1 = -1)
end
```

- Safety-specifications and invariant are represented as boolean expression.

- Initial states:

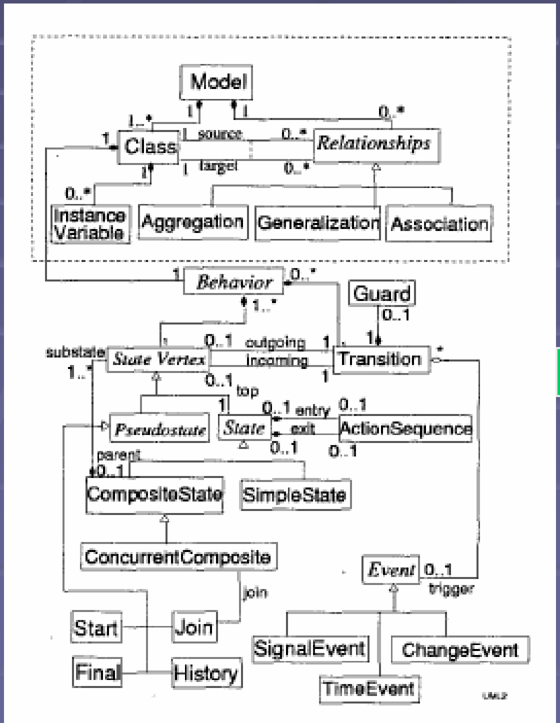
```
init
state X0=1, X1=1;
state X0=2, X1=1;
```

Goal



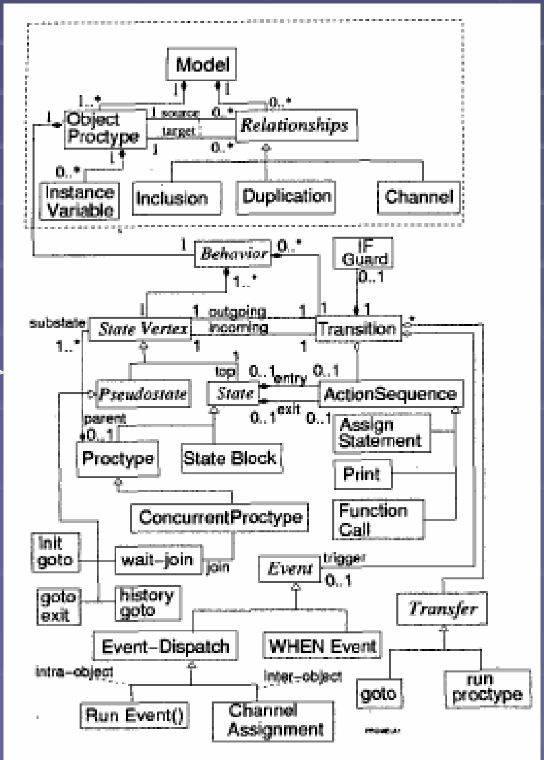
Mapping

- *General Framework for Fromalizing UML with Formal Languages* (Authors: William Mcumber and H.C. Cheng)



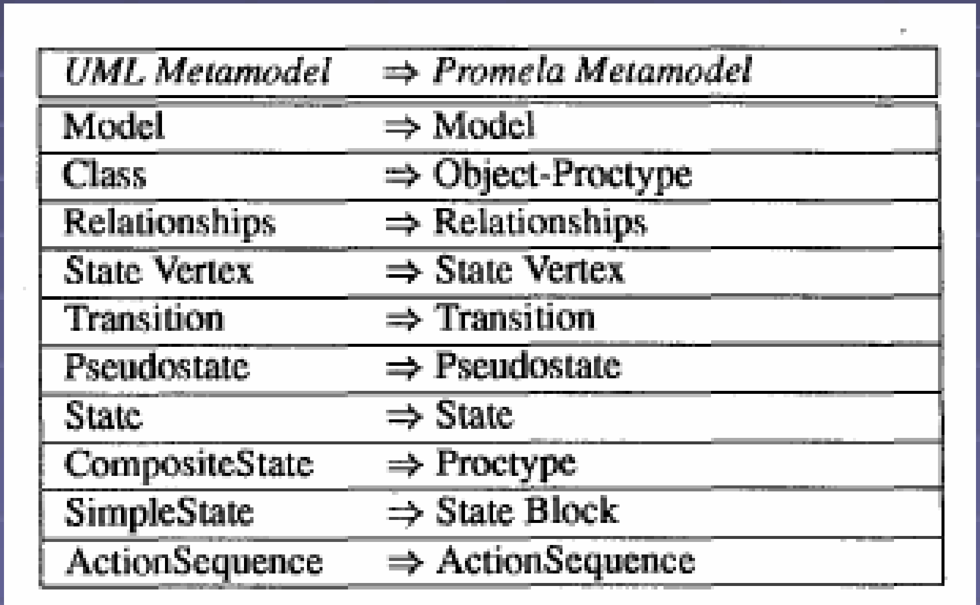
UML Metamodel

Homorphic Mapping



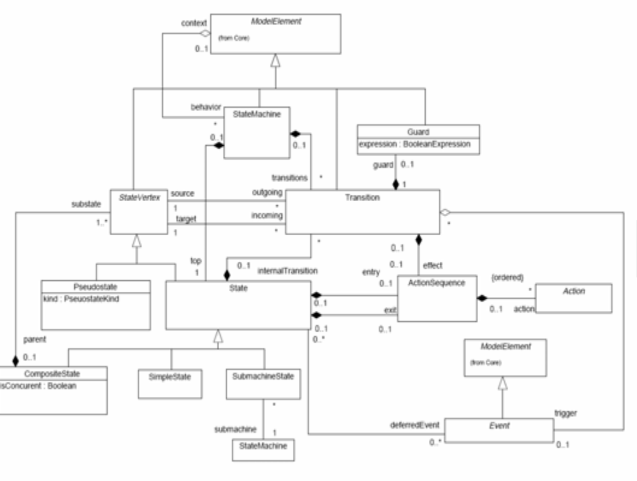
SPIN/Promela Metamodel

Mapping



UML Metamodel	⇒	*Promela Metamodel*
Model	⇒	Model
Class	⇒	Object-Proctype
Relationships	⇒	Relationships
State Vertex	⇒	State Vertex
Transition	⇒	Transition
Pseudostate	⇒	Pseudostate
State	⇒	State
CompositeState	⇒	Proctype
SimpleState	⇒	State Block
ActionSequence	⇒	ActionSequence

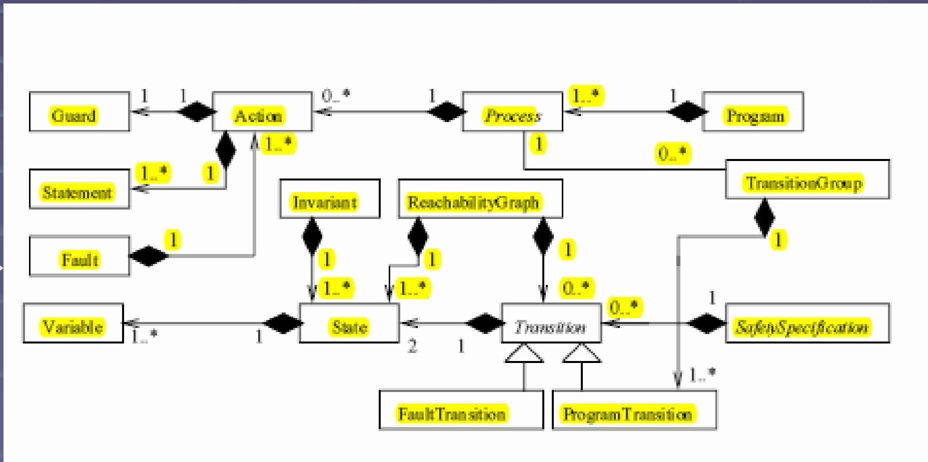
Mapping



UML Metamodel

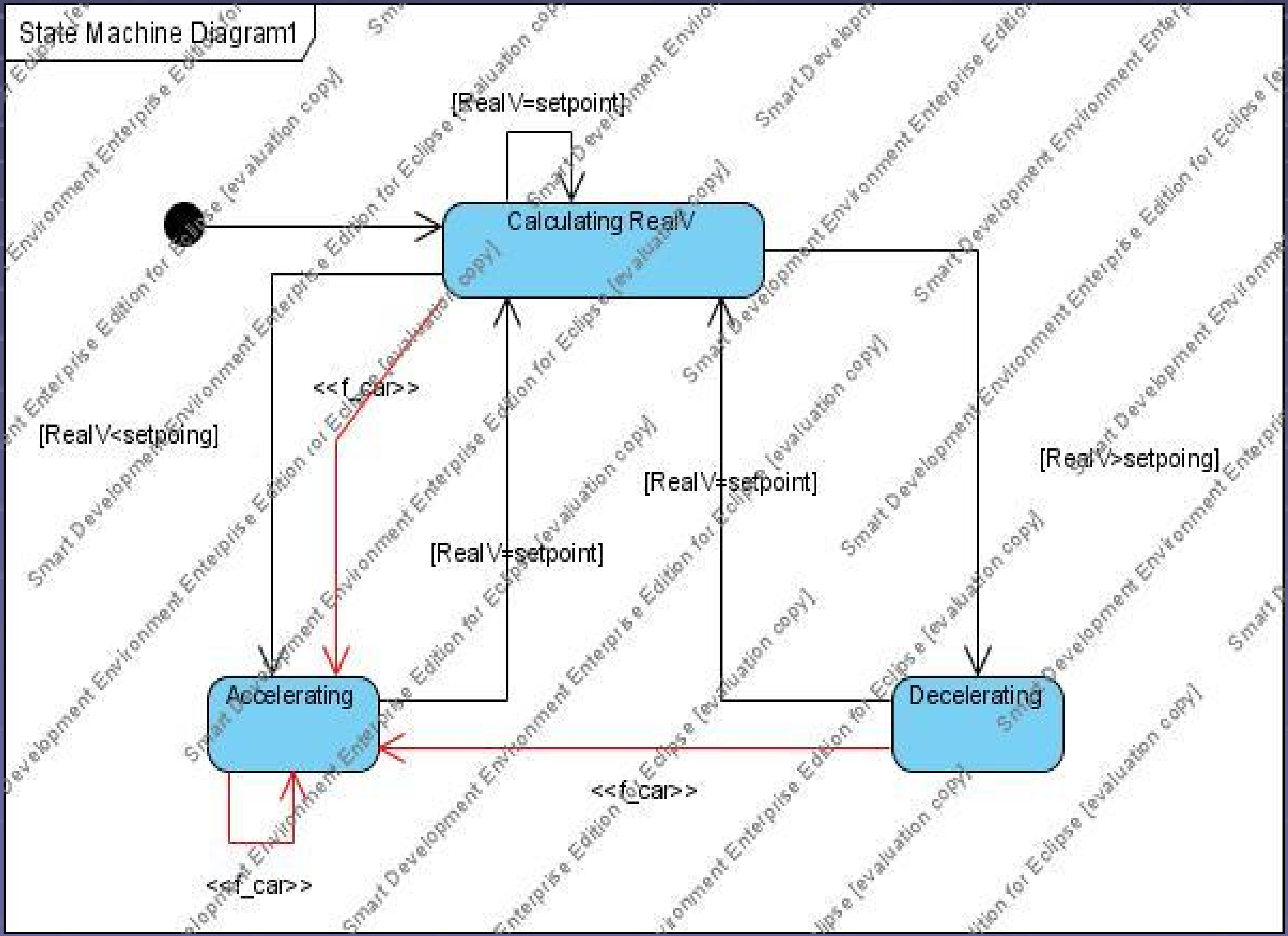
Homorphic Mapping

?

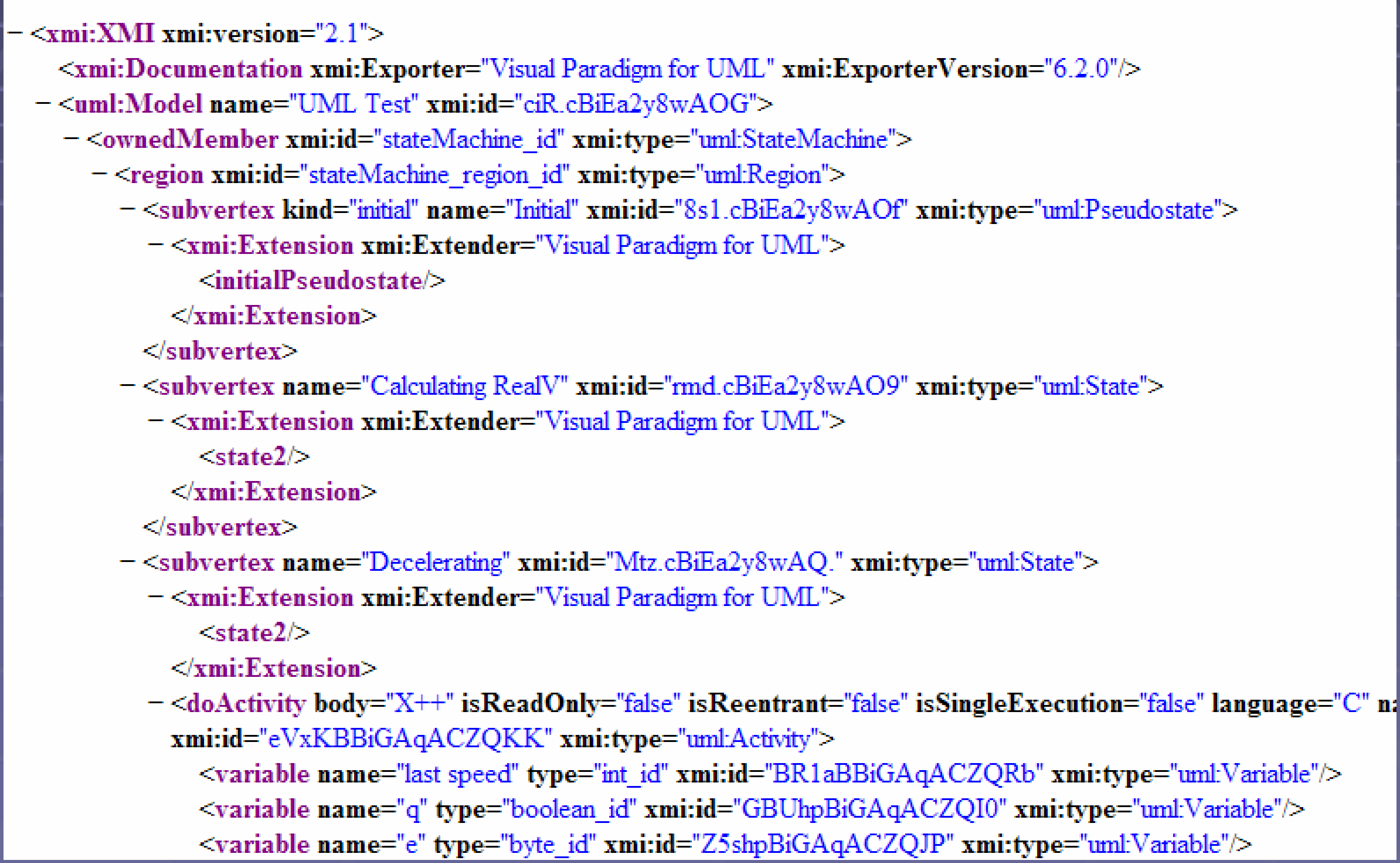


FTSyn Metamodel

Implementation



Implementation



```
− <xmi:XMI xmi:version="2.1">
    <xmi:Documentation xmi:Exporter="Visual Paradigm for UML" xmi:ExporterVersion="6.2.0"/>
  − <uml:Model name="UML Test" xmi:id="ciR.cBiEa2y8wAOG">
    − <ownedMember xmi:id="stateMachine_id" xmi:type="uml:StateMachine">
      − <region xmi:id="stateMachine_region_id" xmi:type="uml:Region">
        − <subvertex kind="initial" name="Initial" xmi:id="8s1.cBiEa2y8wAOf" xmi:type="uml:Pseudostate">
          − <xmi:Extension xmi:Extender="Visual Paradigm for UML">
              <initialPseudostate/>
            </xmi:Extension>
          </subvertex>
        − <subvertex name="Calculating RealV" xmi:id="rmd.cBiEa2y8wAO9" xmi:type="uml:State">
          − <xmi:Extension xmi:Extender="Visual Paradigm for UML">
              <state2/>
            </xmi:Extension>
          </subvertex>
        − <subvertex name="Decelerating" xmi:id="Mtz.cBiEa2y8wAQ." xmi:type="uml:State">
          − <xmi:Extension xmi:Extender="Visual Paradigm for UML">
              <state2/>
            </xmi:Extension>
          − <doActivity body="X++" isReadOnly="false" isReentrant="false" isSingleExecution="false" language="C" na
            xmi:id="eVxKBBiGAqACZQKK" xmi:type="uml:Activity">
              <variable name="last speed" type="int_id" xmi:id="BR1aBBiGAqACZQRb" xmi:type="uml:Variable"/>
              <variable name="q" type="boolean_id" xmi:id="GBUhpBiGAqACZQI0" xmi:type="uml:Variable"/>
              <variable name="e" type="byte_id" xmi:id="Z5shpBiGAqACZQJP" xmi:type="uml:Variable"/>
```

Thank You!