Safety Analysis of a Hemodialysis Machine with S#
Johannes Leupolz, Axel Habermaier, Wolfgang Reif
Institute for Software & Systems Engineering, Augsburg, Germany
Outline

Overview of S#

Case Study Hemodialysis Machine

Safety Analysis

Conclusions and Future Work
Overview of S#

S# is an integrated, tool-supported approach for modeling and analyzing safety-critical systems

- Models are **executable**
  - Simulation, visualization, testing, debugging

- **Formal safety analyses** using **model checking**

- Domain Specific Language for safety-critical systems integrated into **C#**
  - Hierarchical **decomposition** into components
  - **Fault** modeling
Related approaches

SysML
- executability
- analyzability

S#
- nondeterminism
- formal safety analysis
- fault modeling
- expressiveness

Real world Implementation

Formal models
```csharp
int x = 3;
bool b = false;

while (true)
{
    int y;
    bool c;
    c = x <= 5;
    y = Choose(0, 3);
    if (y==3 && c)
        ++x;
}
```
Micro-Macro-step semantics
Hemodialysis Machine
Hierarchical Decomposition

```plaintext
bdd HemodialysisMachine

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ports</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ControlSystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialyzingFluidDeliverySystem</td>
</tr>
<tr>
<td>ExtraCorporealBloop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>references</th>
<th>parts</th>
<th>parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialyzingFluidPreparation : DialyzingFluidPreparation</td>
<td>WaterPreparation : WaterPreparation</td>
<td>ArterialPressureTransducer : Pres</td>
</tr>
</tbody>
</table>
```
public class ControlSystem : Component {
    public int TimeStepsLeft;
    private readonly VenousTubingValve VenousTubingValve;
    /* ... */
    public StateMachine<TherapyPhase> CurrentTherapyPhase = TherapyPhase.InitiationPhase;

    public void ShutdownMotors() {
        VenousTubingValve.CloseValve();
        ArterialBloodPump.SpeedOfMotor = 0;
        /* ... */
    }
    /* ... */
}
public class ControlSystem : Component {
    /* ... */
    public override void Update() {
        CurrentTherapyPhase.Transition{
            from: TherapyPhase.InitiationPhase,
            to: TherapyPhase.EndingPhase,
            guard: TimeStepsLeft <= 0 ||
                VenousSafetyDetector.DetectedGasOrContaminatedBlood,
            action: ShutdownMotors
        };
        /* ... */
    }
}
public class VenousSafetyDetector : Component {
    public readonly BloodFlowInToOut MainFlow = /* ... */
    public bool DetectedGasOrContaminatedBlood = false;
    public VenousSafetyDetector() {
        MainFlow.UpdateForward = SetMainFlow;
        /* ... */
    }
    public virtual Blood SetMainFlow(Blood fromPredecessor) {
        if (!fromPredecessor.GasFree || !fromPredecessor.ChemicalCompositionOk) {
            DetectedGasOrContaminatedBlood = true;
        }
        return fromPredecessor;
    }
    /* ... */
}
Modeling Nondeterminism

class C : Component
{
    public int X { get; private set; } = 0;

    public override void Update()
    {
        X++;
        if (X >= 3)
            X = Choose(0, 3);
    }
}

var c = new C();
Check(FG(c.X == 3), c);
Nondeterministic Faults

State variables
- **Membrane** ∈ \{ok, defect\}
- **Sensed Blood** ∈ \{good, bad\}
- **Venous Valve** ∈ \{open, closed\}

Faults
- membrane ruptures
- safety Detector defect
- valve does not close

Found Hazard with Minimal Cut Set \{m, v\}
class ControlSystem : Component {
   /* ... */
   public virtual void ShutdownMotors() {
      VenousTubingValve.CloseValve();
      ArterialBloodPump.SpeedOfMotor = 0;
      /* ... */
   }
   public Fault SuppressShutdown = new PermanentFault();
   [FaultEffect(Fault = nameof(SuppressShutdown))]
   class SuppressShutdownEffect : ControlSystem {
      public override void ShutdownMotors() {
      }
   }
}
### Results of the Deductive Cause Consequences Analysis

<table>
<thead>
<tr>
<th>Hazard</th>
<th>Minimal Cut Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>H1: Dialysis unsuccessful</strong> (blood is not cleaned and dialysis finished)</td>
<td>(1) {DialyzingFluidPreparationPumpDefect}</td>
</tr>
<tr>
<td></td>
<td>(2) {WaterHeaterDefect}</td>
</tr>
<tr>
<td></td>
<td>(3) {PumpToBalanceChamberDefect}</td>
</tr>
<tr>
<td></td>
<td>(4) {UltrafiltrationPumpDefect}</td>
</tr>
<tr>
<td></td>
<td>(5) {BloodPumpDefect}</td>
</tr>
<tr>
<td></td>
<td>(6) {DialyzerMembraneRupturesFault}</td>
</tr>
<tr>
<td><strong>H2: Blood entering the vein of the patient is contaminated</strong></td>
<td>(1) {SafetyBypassFault, WaterHeaterDefect}</td>
</tr>
<tr>
<td></td>
<td>(2) {DialyzerMembraneRupturesFault, SafetyDetectorDefect}</td>
</tr>
<tr>
<td></td>
<td>(3) {DialyzerMembraneRupturesFault, ValveDoesNotClose}</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

- **Executable**, extended subset of **SysML**
- Specifically tailored for **safety critical systems**
- Enables **analysis** of **complex** case studies

- Released as **open source** (MIT license)
  - [http://safetysharp.isse.de](http://safetysharp.isse.de)

- **Quantitative analysis** under development