**Project Title:** Design and test of autonomous robots in an active world  
**Lead University:** University of Denver  
**Principal Investigator:** Anneliese Andrews  
**Team Members:** Abdelgawad  
**Budget:** $17,500  
**Schedule:**  
- Start date: 1/1/2015  
- End date: 12/30/2015  

**Objective(s):**  
Develop a design approach for intelligent robots that:  
- Considers an active environment  
- Is test-ready

**Problem Statement:**  
Autonomous agents operate in a dynamically changing environment in the presence of many actors. This is challenging to model, both for design and testing purposes.

**Deliverables:**  
- Initial definition of approach  
- Application to three case studies
Last Review Status:

- Approach definition.
- Application to AGV, HRI, and USAR robots.
- Models & Tools: Petri Nets (PIPE v4.3), CEFSM (CADP toolbox).
- Compare the Efficiency of Test Criteria (All serializations, Rendezvous).
- First paper published (MODELSWARD 2015).

Progress since Last Review:

- Second paper published (MoDeVVa 2015).
- Third paper accepted by HASE 2016 (will be held in Orlando, FL, Jan 7-9, 2016 ).
- Fourth paper submitted to MODELSWARD 2015 (under review).
- Lower level of V-Model (component interaction level), RAMP Project with UNC at Charlotte.
- Test execution (GoogleTest for testing ROS processes), In progress.
### TECHNICAL SUMMARY

- **Applicability (3 applications, 2 models)**

<table>
<thead>
<tr>
<th></th>
<th>UGV on Highway</th>
<th>HRI robot in Shopping arcade</th>
<th>USAR robot in disaster scene</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CEFSM</strong></td>
<td>ModelSWARD2015</td>
<td>MoDeVVa 2015</td>
<td></td>
</tr>
<tr>
<td><strong>Petri Nets</strong></td>
<td></td>
<td>HASE 2016</td>
<td>ModelSWARD2016</td>
</tr>
</tbody>
</table>

- Published
- Accepted
- Submitted
- Empty
TECHNICAL SUMMARY

World Model and Test Process (quick view):

Modeling:
- **Structural Model**: UML Class Diagram.
- **Behavioral Model**: CEFSM, Petri Nets

Coverage:
- Internal Test Paths (Edge, Edge-Pair, and Prime).
- Concurrent Test Paths (All Serialization and Rendezvous).
- Input-Space Partitioning (ECC, ACoC)

Test Generation:
- Concurrent test paths (WBTCs).
- Executable behavioral test cases.
USAR Worlds:

- **Disaster Scene:**

- **Actors:**
  - **Victim:** survivor and non-survivor.
  - **Construction:** posts, stairs, elevators, walls, doors, windows, and ramp
  - **Rubble:** rocks, bricks, and concretes.
  - **Debris:** furniture, woods, ropes, wires, pipes, papers, glasses, and plastics.
  - **Hazard:** temperature, electricity, and explosion.
Structural Model:

- **Class Diagram Disaster Actors:** 
  \{survivor, non-survivor, construction, rubble, debris, hazard\}

- **Lists:** [survivor stimuli, survivor conditions, construction types, construction states, rubble types, rubble states, debris types, debris states, hazard types, hazard severity]
Behavioral Models (Petri Nets) & Internal Path Sets.

1. survivor process, \( TP_1 = \{tp_{11}, tp_{12}\} \)
   \( tp_{11} : \text{unrealized} \xrightarrow{\text{su.isVisual()}} \text{realized} \xrightarrow{\text{su.getCondition()}} \text{reported} \)
   \( \text{su.isAccessible()} \xrightarrow{\text{su.stimulate()}} \text{realized} \)
   \( \text{tp}_{12} : \text{unrealized} \xrightarrow{\text{su.getCondition()}} \text{reported} \xrightarrow{\text{su.isAccessible()}} \text{realized} \)

2. non-survivor process, \( TP_2 = \{tp_{21}\} \)
   \( tp_{21} : \text{unobsered} \xrightarrow{\text{nsu.get().description()}} \text{described} \)
   \( \text{nsu.isAccessible()} \)

3. construction process, \( TP_3 = \{tp_{31}\} \)
   \( tp_{31} : \text{unspecified} \xrightarrow{\text{cons.getState()}} \text{specified} \)
   \( \text{cons.change(newState)} \)

4. rubble process, \( TP_4 = \{tp_{41}\} \)
   \( tp_{41} : \text{unstated} \xrightarrow{\text{rubble.change(newState)}} \text{stated} \)

5. debris process, \( TP_5 = \{tp_{51}\} \)
   \( tp_{51} : \text{unnoticed} \xrightarrow{\text{debris.change(newState)}} \text{noticed} \)

6. hazard process, \( TP_6 = \{tp_{61}\} \)
   \( tp_{61} : \text{unidentified} \xrightarrow{\text{hazard.get().Severity()}} \text{identified} \xrightarrow{\text{hazard.change(newSeverity)}} \text{declared} \)
   \( \text{hazard.isManipulatable()} \xrightarrow{\text{hazard.change(newSeverity)}} \text{identified} \)

T-Coverage
• **High Level Model:**

Covered by 12 Interaction Test Paths (*simple*):

\[
\begin{align*}
ITP_1 : (\text{Construction} : TP_3) & \rightarrow (\text{Hazard} : TP_6) \rightarrow (\text{Survivor} : TP_1) \\
ITP_2 : (\text{Hazard} : TP_6) & \rightarrow (\text{Debris} : TP_5) \\
ITP_3 : (\text{Rubble} : TP_4) & \rightarrow (\text{Survivor} : TP_1) \\
ITP_4 : (\text{Debris} : TP_5) & \rightarrow (\text{NonSurvivor} : TP_2)
\end{align*}
\]

• **Input Space Sets:**

10 input-domains. One block for each:

Survivor stimuli block includes \{motion, body heat, sound, and CO2 emission\}.

Survivor conditions block consists of values \{aware, semi-conscious, and unconscious\}. 
TECHNICAL SUMMARY

1. **Concurrent Coverage Criteria:**

   \[ \text{APSESCC} = \sum_{i=1}^{n} \left( \frac{\text{len}(c_{ij})}{\text{len}(c_{ij})} \prod_{j=1}^{\text{len}(c_{ij})} \text{tp}_{ij} \right)! \]

   \[ \text{RCC} = \prod_{i=1}^{n} (TP_i + 1) - 1 \]

2. **Input-Space Partitioning Coverage:**

   \[ \text{ACoC} = \prod_{i=1}^{Q} (B_i) \]

   \[ \text{ECC} = \sum_{i=1}^{Q} \text{MAX}(B_i) \]
TECHNICAL SUMMARY

• Test Criteria Efficiency:

<table>
<thead>
<tr>
<th>Interaction Test Path</th>
<th>Test-paths C.</th>
<th>Test-data C.</th>
<th>APSESCC with ACoC</th>
<th>RCC with EEC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APSESCC RCC</td>
<td>ACoC ECC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITP₁</td>
<td>23100 11</td>
<td>18480 11</td>
<td>426888000 121</td>
<td></td>
</tr>
<tr>
<td>ITP₂</td>
<td>35 3</td>
<td>2400 15</td>
<td>84000 45</td>
<td></td>
</tr>
<tr>
<td>ITP₃</td>
<td>70 5</td>
<td>540 15</td>
<td>37800 75</td>
<td></td>
</tr>
<tr>
<td>ITP₄</td>
<td>35 3</td>
<td>120 15</td>
<td>4320 45</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>23240 22</td>
<td>42420 56</td>
<td>427014120 286</td>
<td></td>
</tr>
</tbody>
</table>

• Issues:
  • Infeasible Test-path and infeasible Test-data.
• Components interaction testing for RAMP, UNC at Charlotte:

• V-Model
UNC at Charlotte:

Real-Time Adaptive Motion Planning (RAMP):

- Planning trajectories of high-DOF robots (mobile manipulators).
- Generator trajectories and evaluate them.
- Execute trajectories on robot
- Sensing for odometry information of obstacles

All these features intent to work concurrently

• UNC at Charlotte:

RAMP Component Diagram

RAMP Class Diagram
TECHNICAL SUMMARY

1. Planner:

Note: Planner consists of two independent processes.

1. Planning & Control:

Predicates:

\[
P_{11} = (-/ \lfloor n<PC \rfloor / n++ / \text{modify(population)}) // \text{e.g., PC} = 4, \text{for planning cycles}
\]

\[
P_{12} = (-/ \lfloor n=PC \rfloor / n=0 / -)
\]

\[
P_{13} = (-/ \lfloor n<1 \rfloor / n++ / \text{setErrorCorrectionArray}() / \text{send(trajecory); update(TrajectoryRequest(), EvaluationRequest(), startStatePopulation)})
\]

\[
P_{14} = (-/ \lfloor n=1 \rfloor / n=0 / -)
\]

- Red color functions are external messages the transition send once it is executed.
- Green color functions are events received which trigger the transitions.
- Dots square refers to that this component consists of multi-processes
2. Trajectory Generator:

Predicates:

\[ P_{21} = (\neg \neg / [TCP\_Socket = null] / \neg \neg / \neg \neg) \]

\[ P_{22} = (\text{TrajectoryRequest}(\ldots) /[TCP\_Socket != null] / \text{generateTrajectory}() / \neg) \]

\[ P_{23} = (\neg \neg / ![\text{trajectory\_generation\_Done}()] / \neg \neg / \neg \neg) \quad // \text{We can put time-constraint for trajectory generation (e.g., 8 ms)} \]

\[ P_{24} = (\neg \neg / [\text{trajectory\_generation\_Done}()] / \neg / \text{TrajectoryResponse}(): \text{trajectory[]} ) \]
CEFSM for RAMP Components interaction

TECHNICAL SUMMARY

1. Planner

2. Trajectory Generator

3. Trajectory Evaluation

4. Control

5. Path Modification

6. RAMP Sensing
TECHNICAL SUMMARY

- Test Execution is an issue (Platforms & Languages).
- **In progress …** Execute WBTCs on ROS processes *(GoogleTest/gtest).*

```c++
#include <gtest/gtest.h>
#include "rostalker.h"
#include "roslistener.h"

test(TestSuiteName, TestcaseName)
{
  try { talker.publish("Hello");
      EXPECT_TRUE (talker.getTopic.compare ("Hello") == 0);
  } catch(...) { ADD_FAILURE() << "Fatal failure" ; }
}

// Test Runner
int main(int argc, char **argv) {
  testing::InitGoogleTest(&argc, argv); // Initialize gtest
  ros::init(argc, argv, "talker_tester"); // Initialize ros
  return RUN_ALL_TESTS();
}
```
PERFORMANCE SUMMARY

• Applied to AGV, HRI, and USAR domains (Modeling and Test generation).

• 2 Published, 1 Accepted, and 1 Submitted.

• V-Model levels with UNC at Charlotte (RAMP).

• Test Execution, Google Test for ROS, in progress.
PLAN FOR NEXT REVIEW

• Finish the Applicability Investigation (obsolete WBTCs).

• Test Execution.

• Preliminary Exam.

• The Whole Work: Journal Paper (Applicability, Generalizability, and Evaluation).