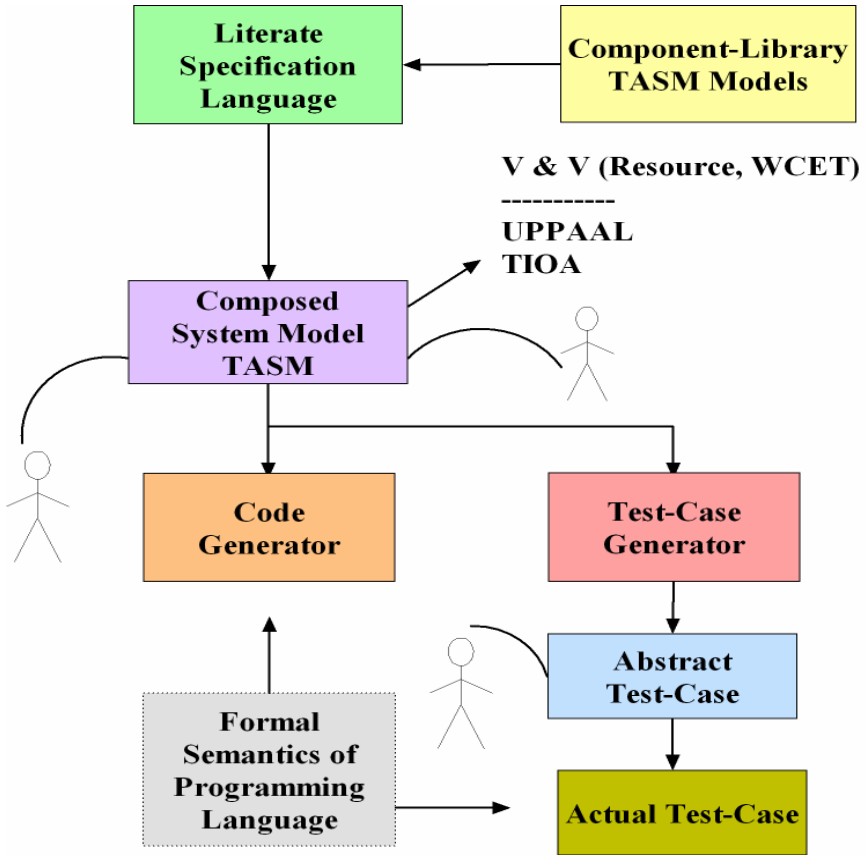
The Timed Abstract State Machine (TASM) Language and Toolset

Martin Ouimet, Guillaume Berteau, Kristina Lundqvist
Embedded Systems Laboratory
Massachusetts Institute of Technology
October 1st, 2006

Introduction

- Hi-Five Framework
  - Specification framework for V & V of Real-Time Embedded Systems
  - Use same language across components and abstraction levels
  - Be “open”
  - Be “usable”



The Timed Abstract State Machine Language (TASM) - I

- Derivative of Abstract State Machines (ASM)

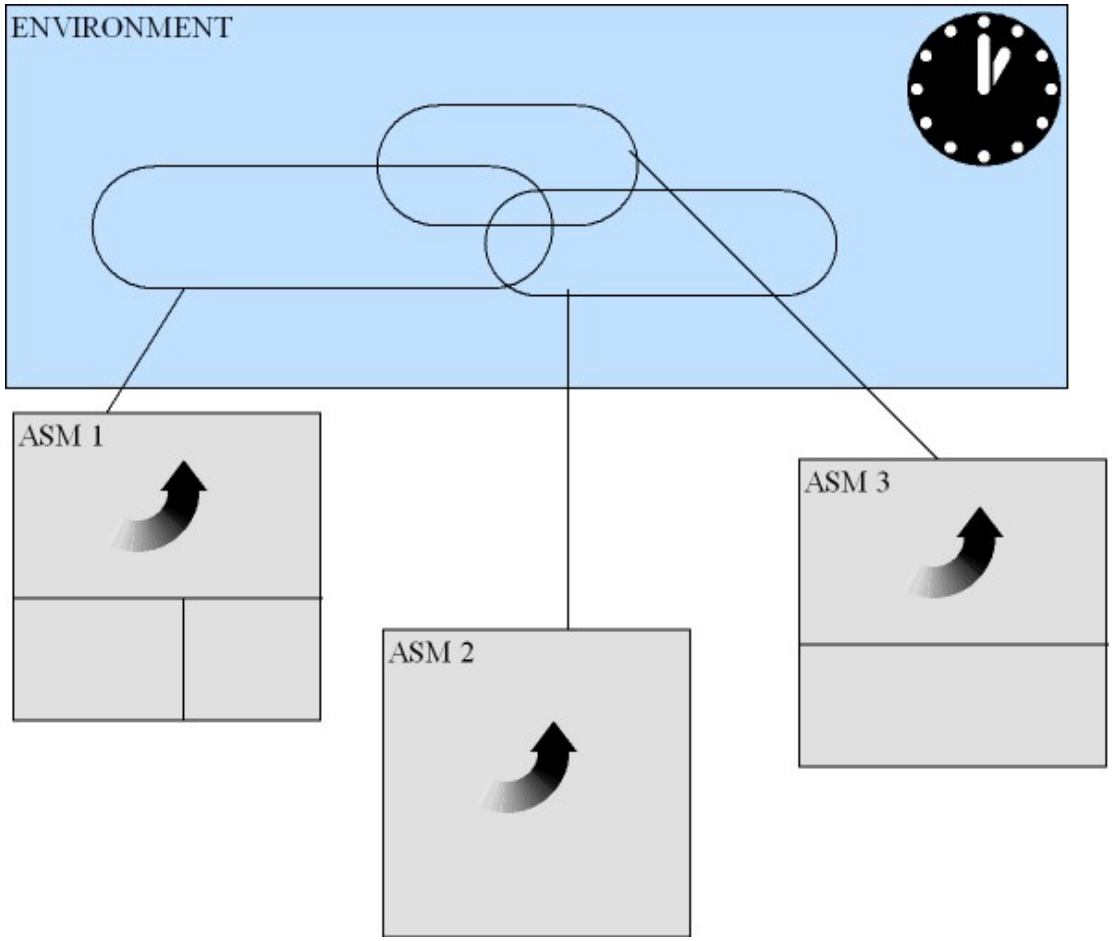
- Extensions to include non-functional behavior alongside functional behavior
  - Time
  - Resource consumption

- “Literate” specification language

The Timed Abstract State Machine Language (TASM) - II

- ASM + Time + Resources
- Parallel and hierarchical composition
- Synchronous and asynchronous systems
- Synchronization model
  - Time
- Communication model
  - Shared variables

TASM – Global View



Introducing ASMs – States and Update Sets

- State = Value of environment variables at a given point in time, a “step”
- Step = ASM produces an update set, that is an assignment of one or more values to one or more global variable
- Run = Sequence of update sets
- State Evolution = Application of update set at each step of the run. $S_i = S_{i-1} @ U_i$
- Example:
  - Run: $U_1, U_2, U_3, U_4, \ldots$
  - States: $S_0, S_1, S_2, \ldots$

Introducing TASMs – State and Update Sets + Time + Resources

- State = Value of environment variables at a given point in time, a “timestamp” on the global time axis
- State Evolution = Application of update set at specific timestamps of the run. $TS_i = TS_{i-1} @ TU_i$
- Example:
  - Run: TU_1, TU_2, TU_3, TU_4, …
  - States: TS_0, TS_1, TS_2, …
  - $(0, S_0)$, (t_1, S_1), (t_2, S_2), …

ASM - Example

```
light_status := {On, Off};
light_status light1 := On;
light_status light2 := Off;
```

Environment

```
R1: Turn On
  If light1 = Off then
    light1:= On;

R2: Turn Off
  If light1 = On then
    light1 := Off;
```

ASM 1

```
R1: Turn On
  If light2 = Off then
    light2:= On;

R2: Turn Off
  If light2 = On then
    light2 := Off;
```

ASM 2

- Initial environment
  - (light1, On),
    (light2, Off)
- Synchronous
  - (On, Off),
    (Off, On),
    (On, Off), etc.
- Asynchronous
  - (On, Off),
    (Off, Off),
    (On, Off),
    (Off, Off), etc.

TASM - Example

```
light_status := {On, Off};
light_status light1 := On;
light_status light2 := Off;
```

Environment

```
R1: Turn On
  t := [3, 5];
  memory := 35;
  If light1 = Off then
    light1:= On;

R2: Turn Off
  t := [1, 2];
  memory := 15;
  If light1 = On then
    light1 := Off;
```

TASM 1

```
R1: Turn On
  t := [1, 2];
  memory := 25
  If light2 := Off
    light2:= On;

R2: Turn Off
  t := [3, 5];
  memory := 35;
  If light2 = On then
    light2 := Off;
```

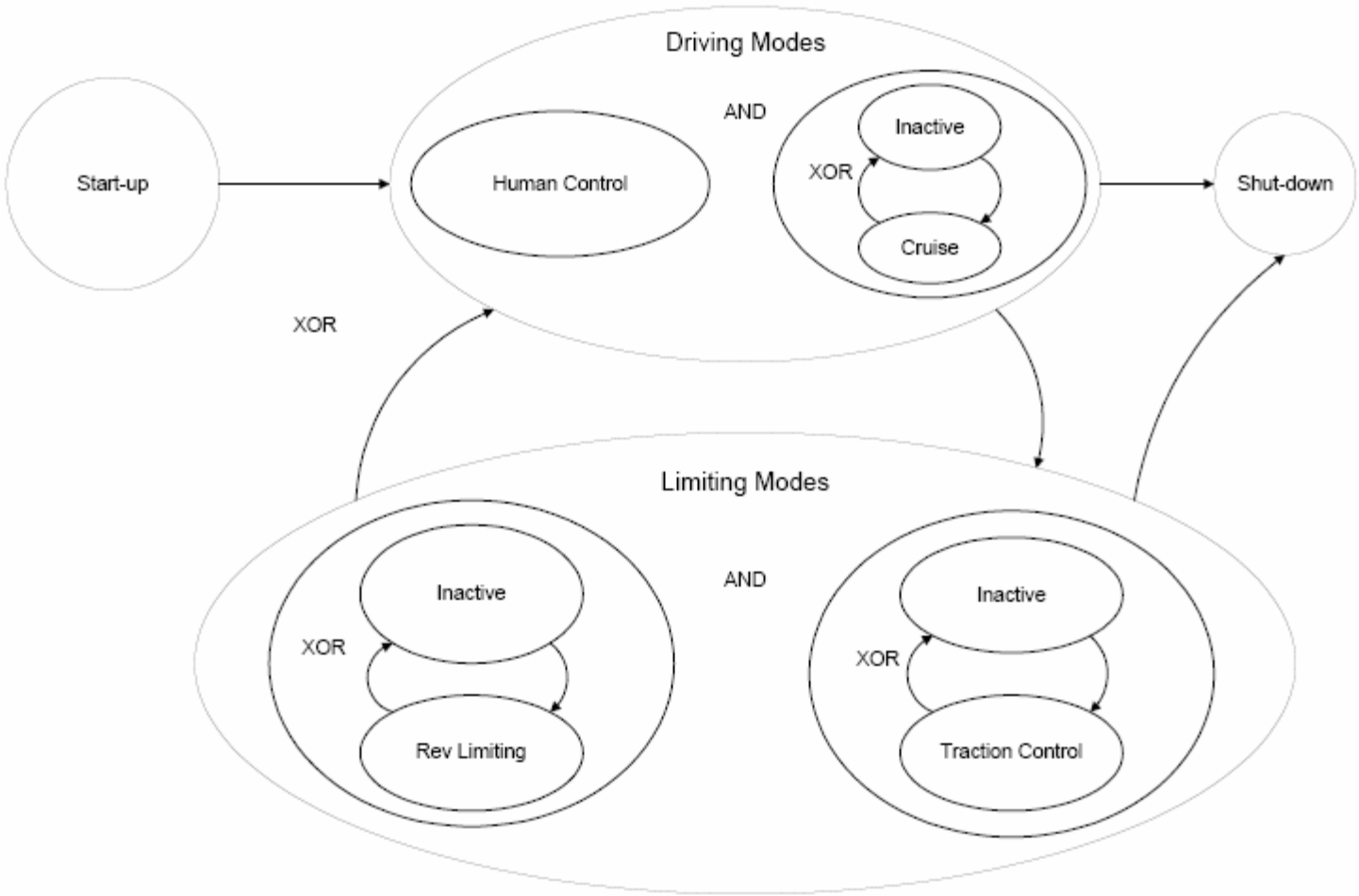
TASM 2

- t = 0
  (On, Off)
  memory = 40
- …
- t = 1
  (On, On)
  memory = 40
- …
- t = 2
  (Off, On)
  memory = 50
- …

A More Substantial Example – Electronic Throttle Controller

- Drive-by-wire system to regulate throttle angle and fuel injection for fuel-efficiency
- Relies heavily on modes to compute commanded throttle voltage
- Multiple combination of modes – rev-limiting, traction-limiting, cruise-control, human-input
- From Paul G. Griffiths (UC Berkeley)

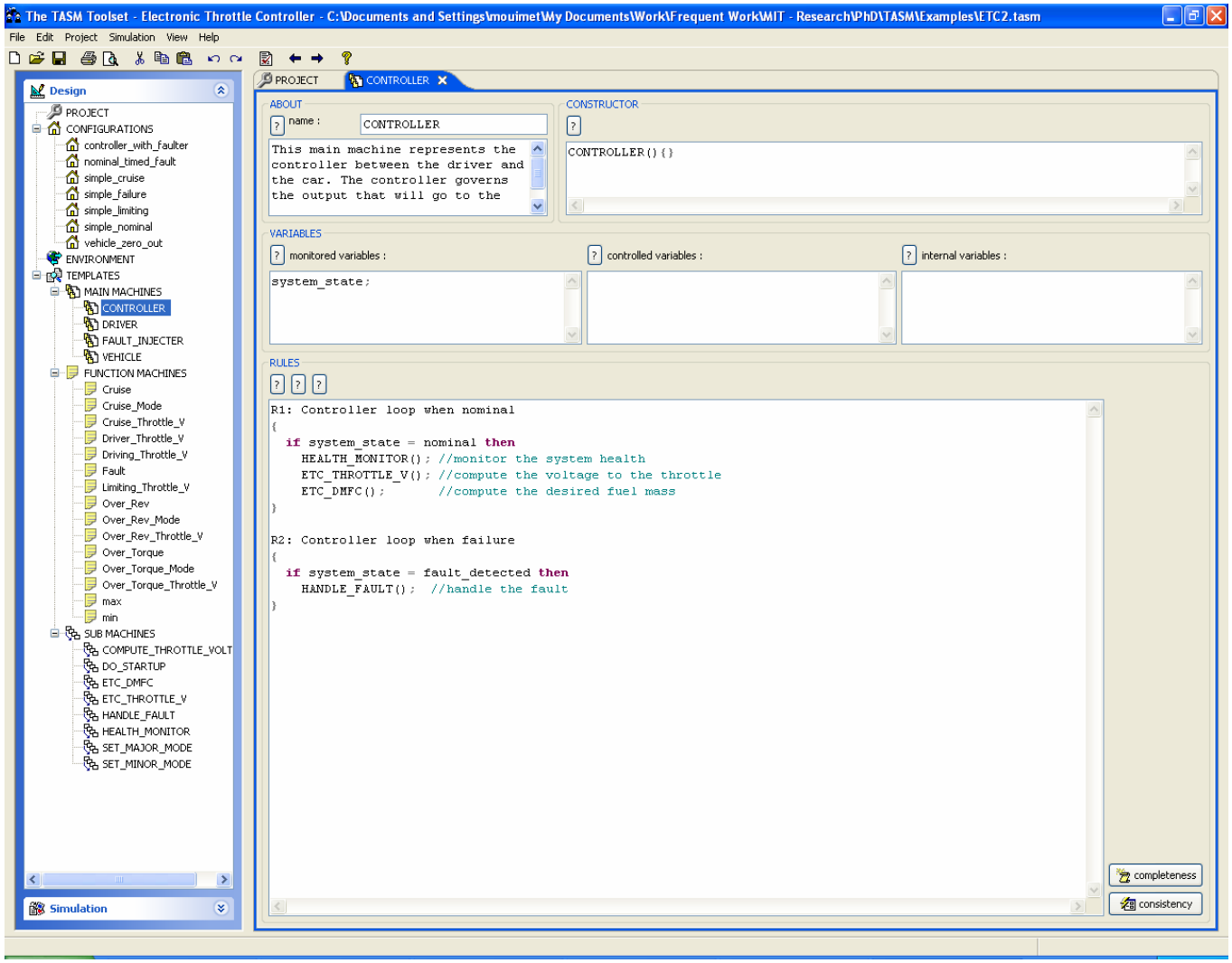
A More Substantial Example – Electronic Throttle Controller



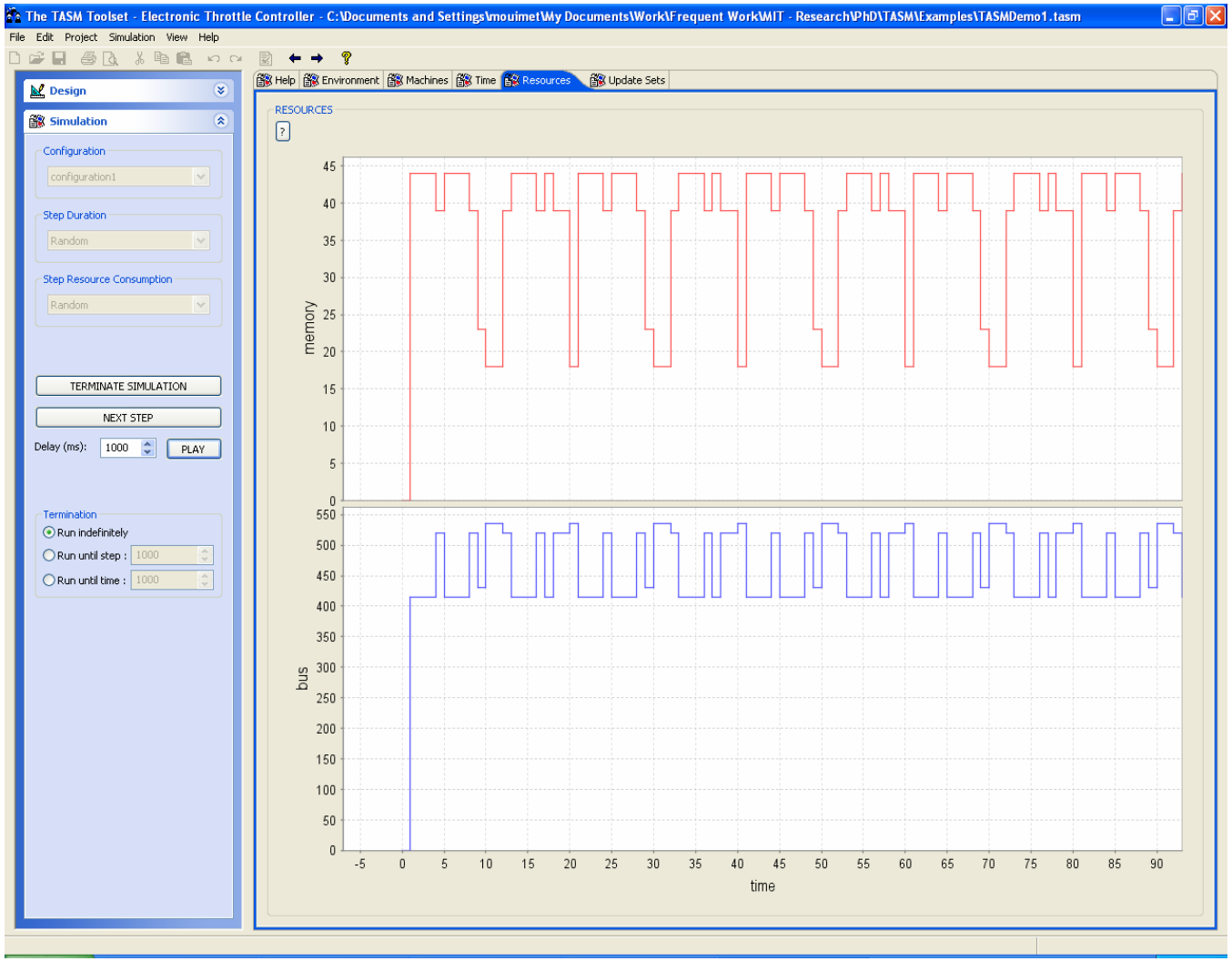
The TASM Toolset

- Tool to:
  - Read and write specifications in the TASM language
  - Simulator to execute specifications
  - Verify consistency of rule sets for a given machine
  - Verify completeness of rule sets for a given machine

A More Substantial Example – Electronic Throttle Controller



A More Substantial Example – Electronic Throttle Controller



TASM Toolset Demo

- Simple Example
- ETC

Research Roadmap

- Create more examples to exercise language
- Map to UPPAAL to verify best-case and worst-case behavior (time, resource)
- Use conversion to CNF to generate test-cases

Summary

- Conclusion
  - TASM language – executable specification language
  - Unified language – function + time + resources
  - Toolset for analysis and simulation

- For more information
  - tasm@mit.edu
  - http://esl.mit.edu/tasm

References

- ASM
  - Börger, Stärk – Abstract State Machines: A Methodology for high-level System Design, Springer-Verlag, 2003.
  - Börger – Why use Evolving Algebras for Hardware and Software Engineering?, LNCS Vol. 1012, 1995.

- ETC
  - Griffiths - Embedded Software Control Design for an Electronic Throttle Body. Master's Thesis, UC Berkeley, 2000.