
Microsoft® Research
Faculty Summit
10
YEAR ANNIVERSARY

The Foundations and Tools for Software Engineering Lab

Department of Computing, FCEN,
University of Buenos Aires, Argentina

Sebastian Uchitel

You are here. We are here.

About us

- Research:
  - Foundations and Tools for Software Engineering
- People
  - Directors: Victor Braberman and Sebastian Uchitel
  - 3.5 Staff
  - 1 Postdoc
  - 6 PhD Students
  - Several master’s level research assistants

About us

- Ongoing Collaborations
  - Microsoft, University of Toronto, Imperial College London, University College London, University of Louvain-la-Neuve, CNRS-France
- Consultancy
  - Kodak UK, Polo IT Buenos Aires, HP, Telco's, Pragma, MS Corp, Argentine Government, etc...
- Teaching
  - Undergraduate, Graduate and Industry

About us

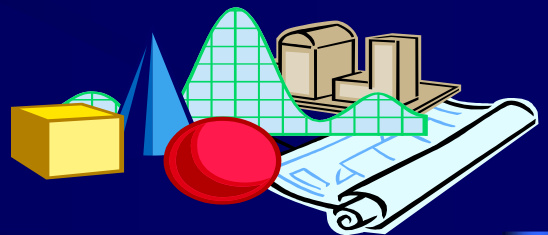
- Publication track record
  - Journals: TOSEM, TSE, FMSD, STTT, ASEJ, ...
  - Conferences: ICSE, FSE, RTSS, ASE, TACAS, CAV, ...
- Grant track record (currently over 2.3 million USD)
  - ANCPYT, ECOSUD (Argentina/France), CONICET, UBACYT, EPSRC (UK), EU-FP6 (EU), CECYT-MAE (Argentina/Italy)
- International Recognition
  - Program Committees: ICSE (2005, 2007, 2008), ISSTA 06, FASE (2006-2007), ASE (2003-2006), ICTAC 05, FSE (2005-2007), RE (2005, 2007), ...
  - Program Chairs: SCESM 2004, ASE 2006, ICSE 2010.
  - Journal Editorial Boards: TSE (2006-), REJ (2007-)
  - Awards: Microsoft Research, IBM, Leverhulme Trust, Nuffield Foundation, CESSI, Argentine National Academy of Science...

Overview

- Technical areas
  - Model Extraction
  - Static Analysis
  - Memory usage prediction
  - Dynamic Analysis
  - (Distributed) Model Checking
  - Test-case generation
  - Test-guided model checking
  - Quantitative Modeling and Analysis
  - Machine learning
  - AOP
  - Model Synthesis
  - Partial Behaviour Models
- Application Domains
  - Real time systems
  - Service Oriented Architectures
  - Distributed and Concurrent systems
  - Object-oriented programs
  - Embedded systems
  - Dynamic and reconfigurable systems
- Software Engineering Activities
  - Requirements Engineering
  - Software Architecture
  - Testing
  - Design

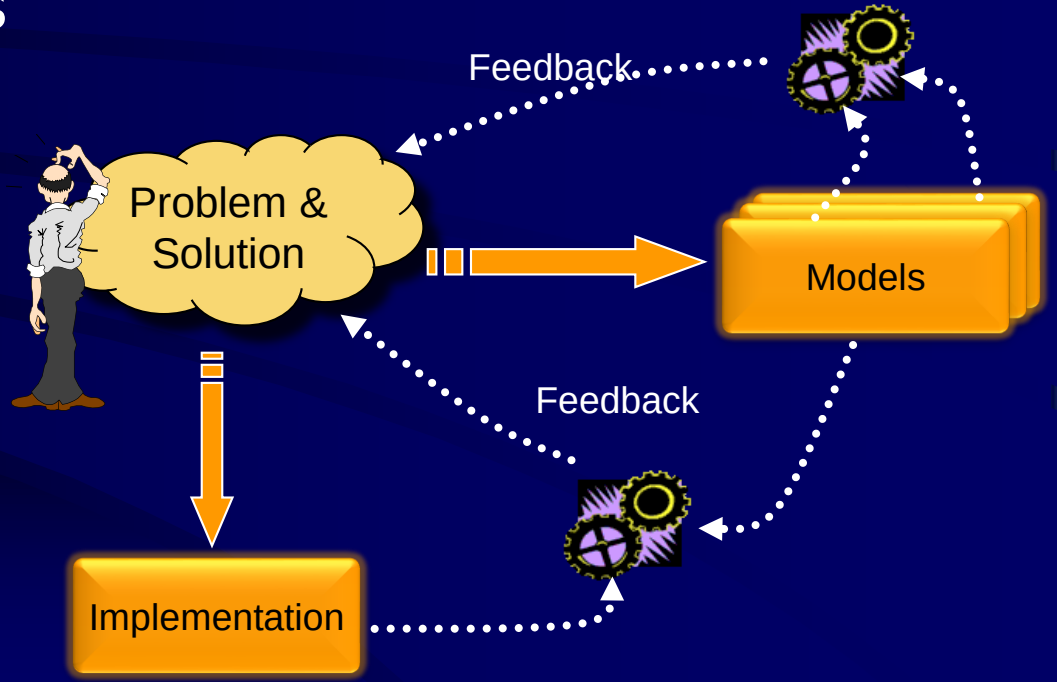
Our vision: We believe that...

- Models should play a central role in software engineering.
- Traditional engineering approach
  - Abstract & Precise
  - Amenable to analysis.
  - Complexity: Model **<<** System.
- Pre-development analysis of behaviour
  - Prevent consequences
  - Early detection -> cheaper fix
- Costs << Benefits



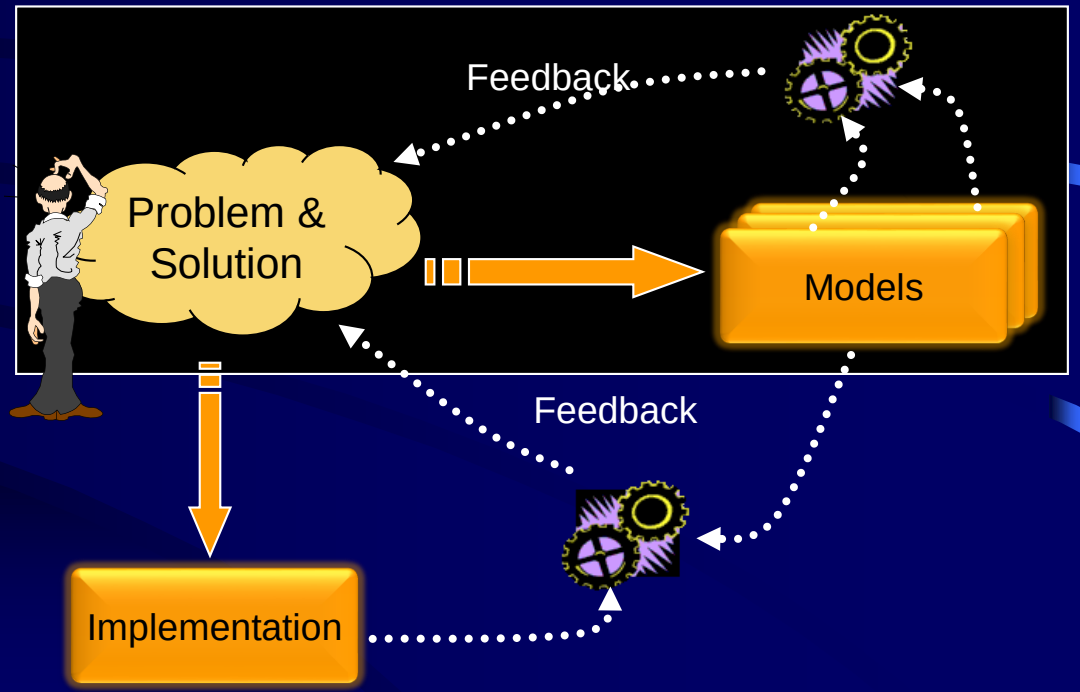
Our Research Focus

- Models
- Automated Analyses
- Verification and Validation



Theme 1: Validation

- How do I know I've modelled the right thing?



Theme 1: Validation of Contract Specifications

- Contract specifications
  - Pre/Post-conditions + invariants

  appear in a variety of software artefacts
  - Specification (Z, Design by Contract, Use Cases)
  - Code (Spec#-C#, Eiffel, Java)
  - Output of Analysis tools (Daikon, DySy)
- However, they are far from trivial to understand

Contracts are hard to validate

```
contract CircularBuffer
  variable a : array[element]
  variable w, r : integer

  invariant : 0 ≤ r < |a| ∧ 0 ≤ w < |a| ∧ |a| > 3

  start : |a| > 3 ∧ r = |a| − 1 ∧ w = 0

  action write(element e)
    pre : w < r − 1 ∨ (w = |a| − 1 ∧ r > 0)
    post : r' = r ∧ w' = (w + 1) % |a| ∧ a' = store(a, w, e)

  action element read()
    pre : r < w − 1 ∨ (r = |a| − 1 ∧ w > 0)
    post : a' = a ∧ w' = w ∧ r' = (r + 1) % |a| ∧ rv = a[r']
```

Validation Strategies

- Visualise state space
  - Even simple contract specifications are infinite state
- Execute / Simulate
  - Very partial exploration
  - When do we stop?
  - No big picture
- Prove properties (model check)
  - Which properties?
  - Do we have them all?
  - Must validate the properties…

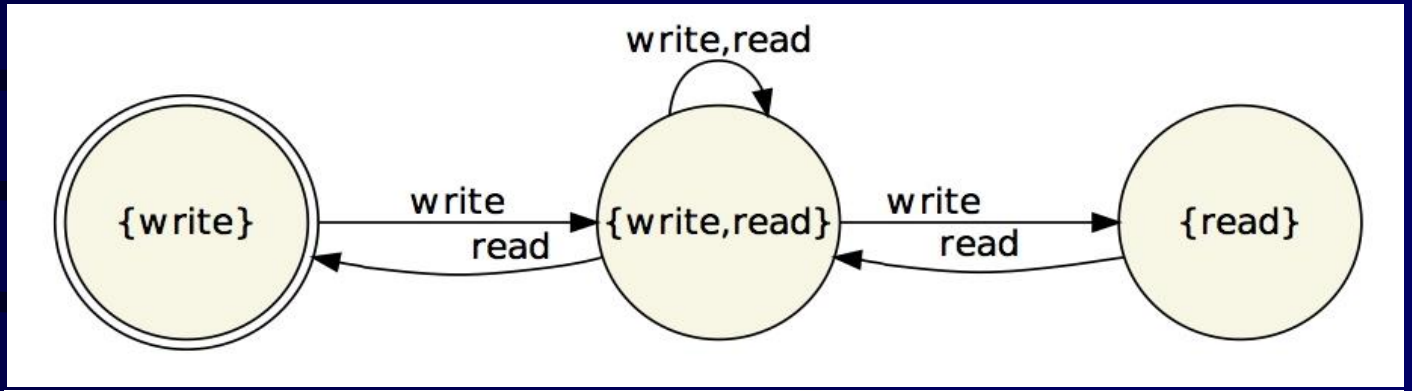
Our validation strategy: Abstraction

- What is the right abstraction of an infinite state space that will aide validation?
  - Precision vs. Size trade-off is key
- A: Finite State Machine that preserves action enabledness
  - Two concrete states are in the same abstract state if and only if they allow the same set of actions
    (i.e. preconditions that hold for both
    are the same)

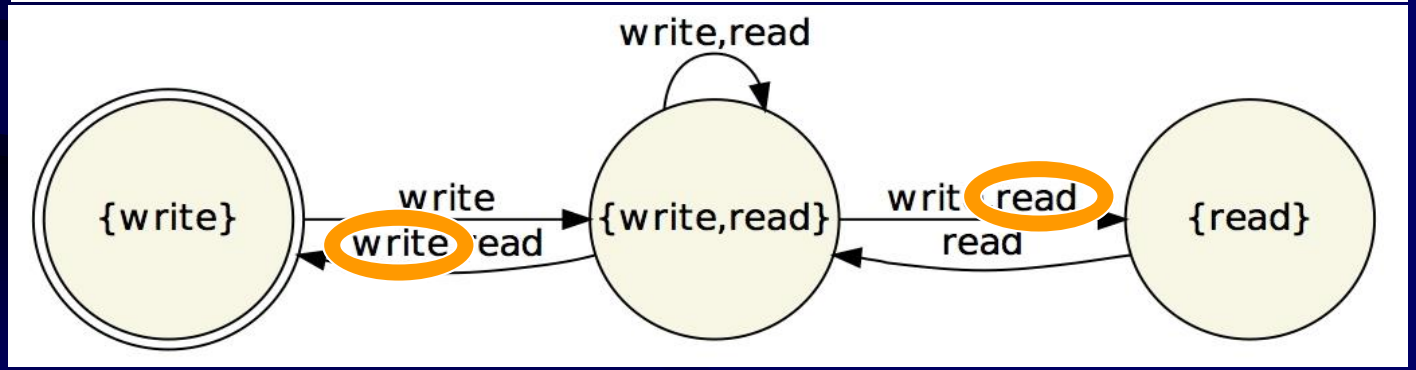
[ICSE 09]

Enabledness Preserving Finite State Machine

Model A



Model B

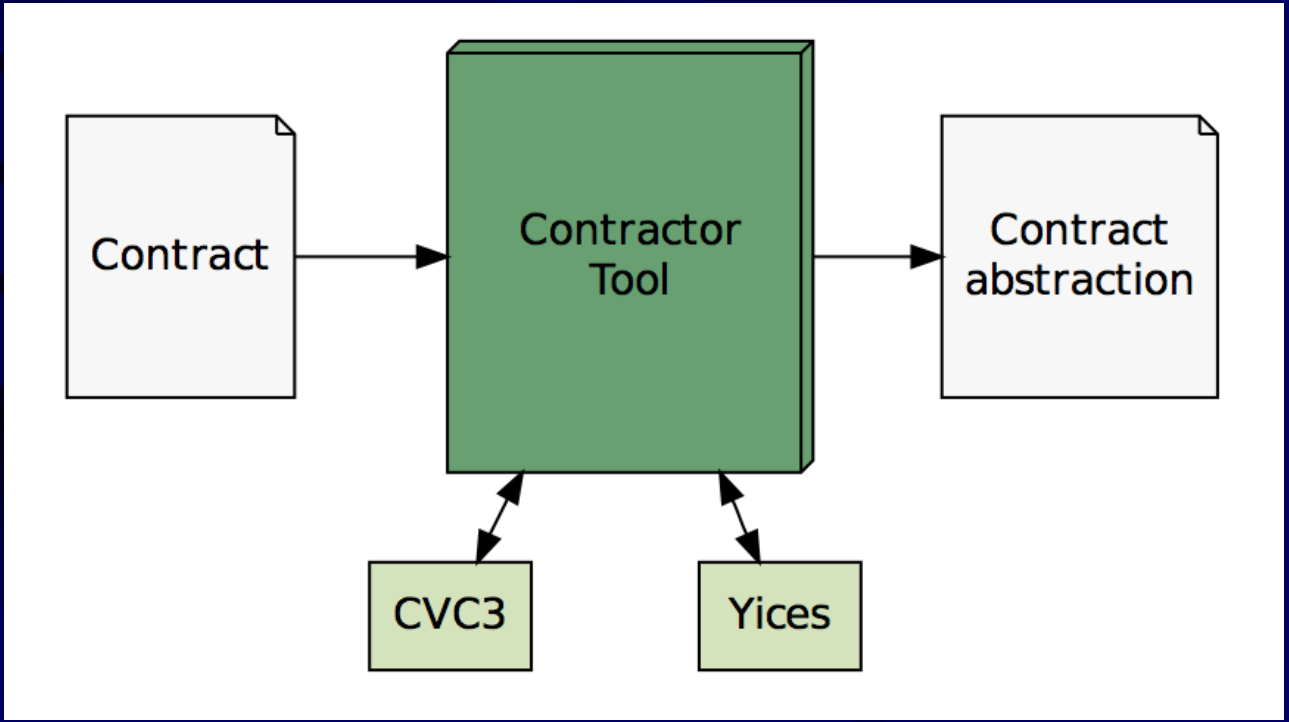


Circular Buffer has an error

“(r != w)” is missing from the invariant

Tools Support

Open source available at http://lafhis.dc.uba.ar/contractor

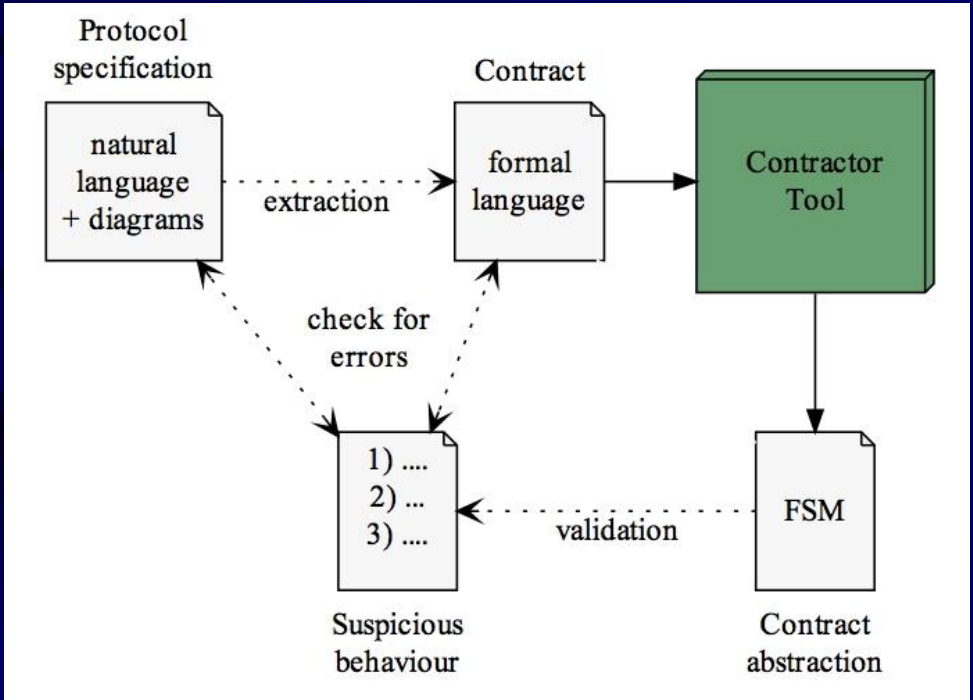


Validating Windows Server protocols

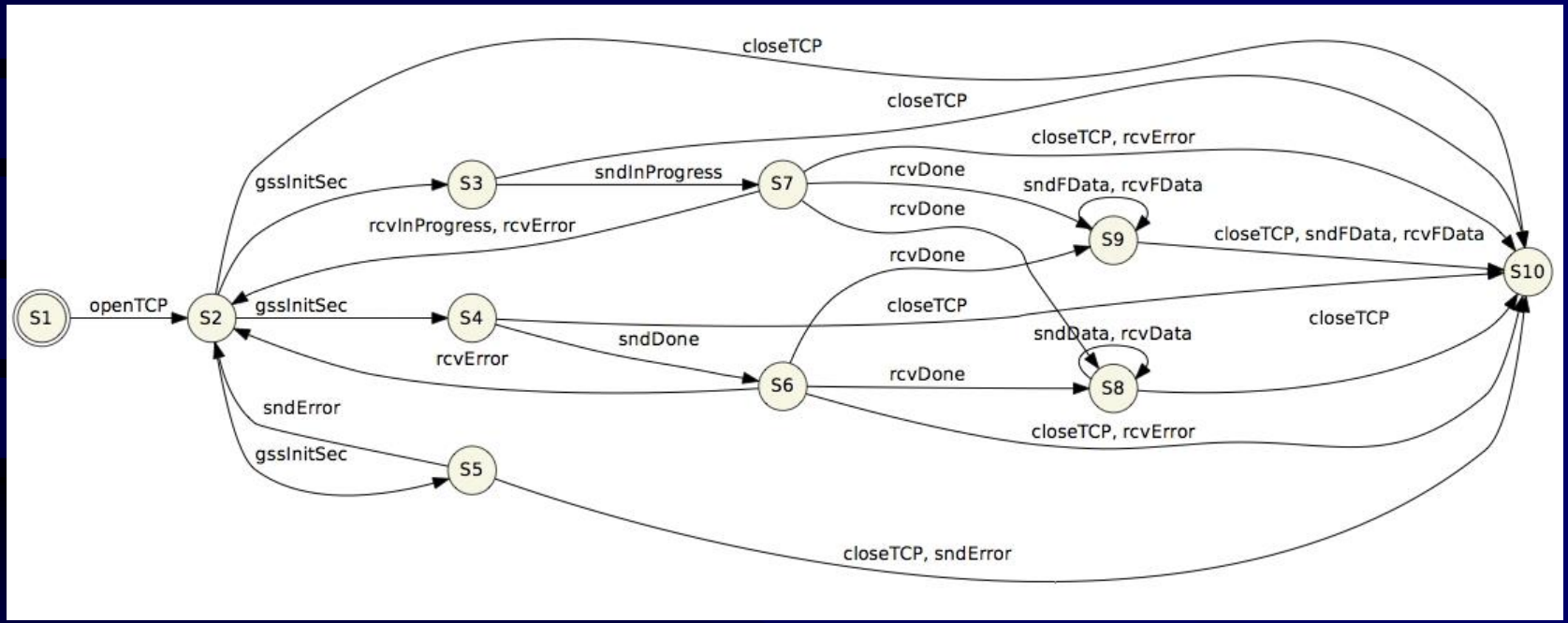
- Negotiate Stream Protocol
  - A protocol for the negotiation of credentials between a client and a server over a TCP stream
  - 13 operations, potential state space of 2^13 = 8192
  - Challenge: Will the size allow for manual validation?
- WINS Replication and Autodiscovery Protocol
  - Governs the process by which a set of name servers discover each other and share their records in order to keep an up-to-date vision of the name mappings
  - 33 operations, potential state space of 2^33 = 8 Billion
  - Challenge: Can we build it, let alone validate?

Windows Negotiate Stream Protocol 2.0

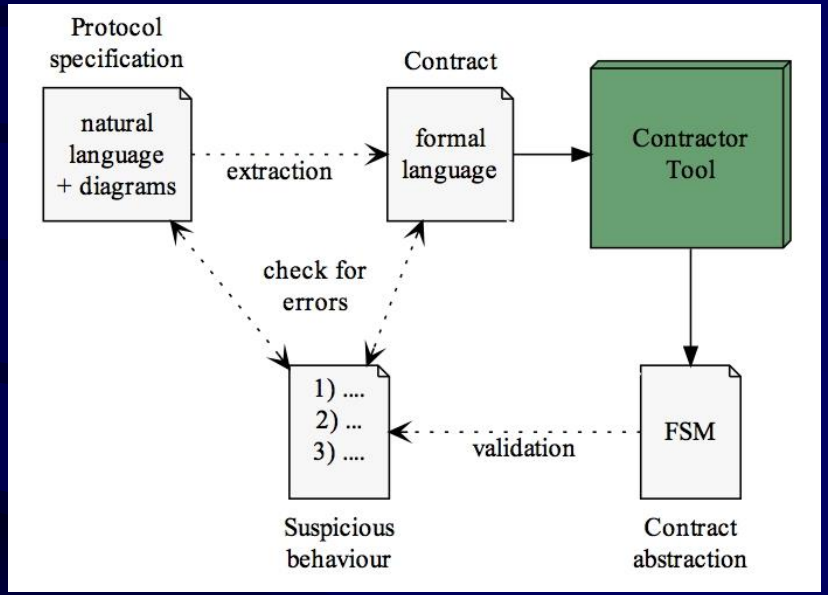
Experimental Setup



Windows Negotiate Stream Protocol 2.0



Windows Negotiate Stream Protocol 2.0



Various problems were found in the TD 2.0.
These problems were fixed in TD 3.0

Case studies

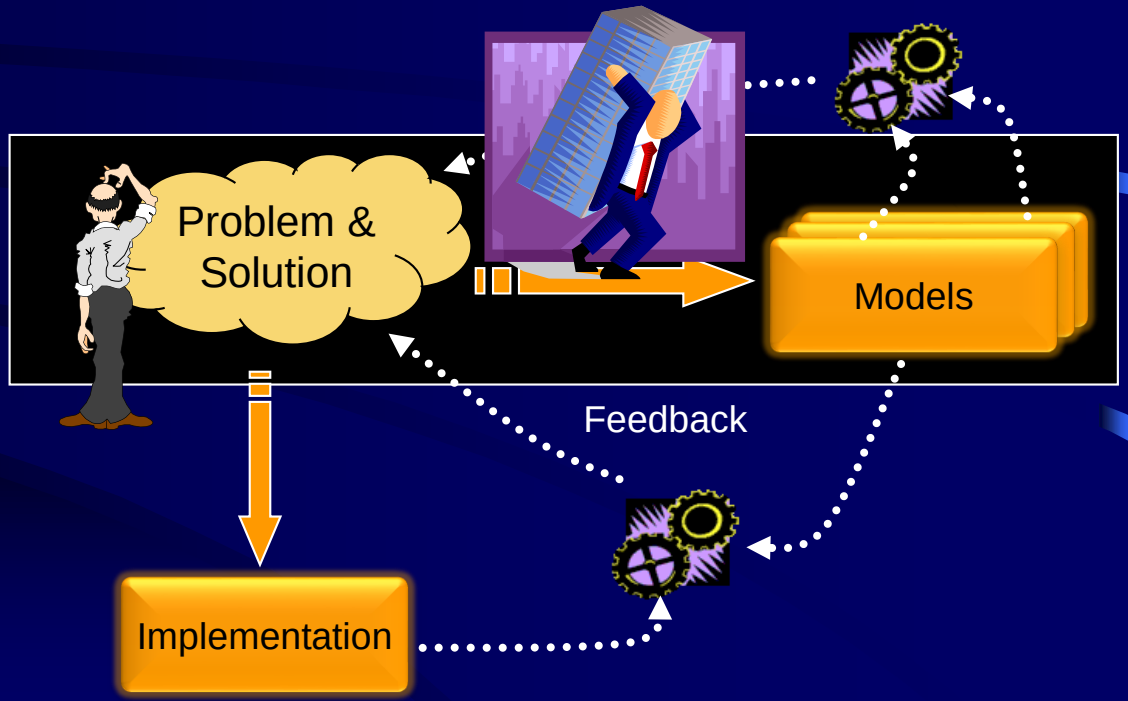
	Operations	Reachable states	Execution time (seconds)
Web Fetcher [de Line 2004]	4	2	0.3
ATM [Whittle 2000]	8	6	5
MS-NSS	13	10	4
MS-WINSRA	33	39	97

Future Work

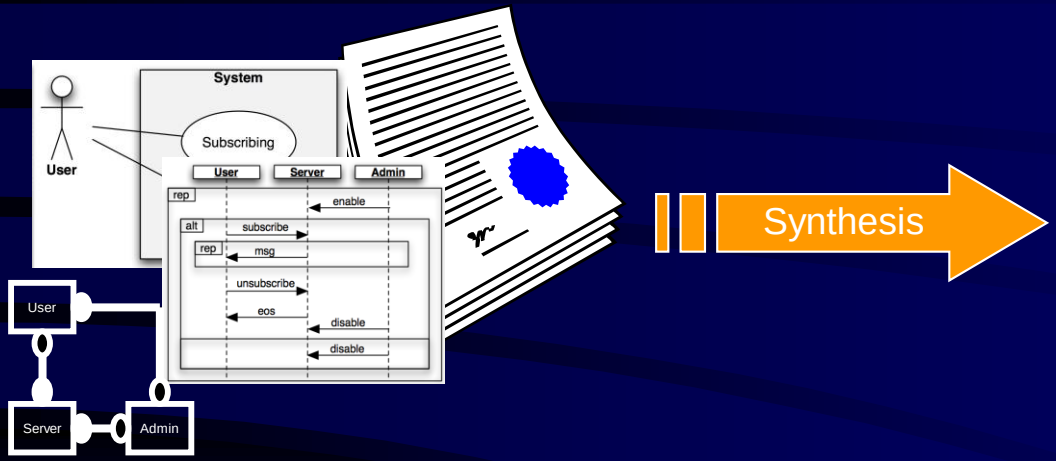
Talking to the Microsoft Protocol Engineering Team

Theme 2: Model Construction and Elaboration

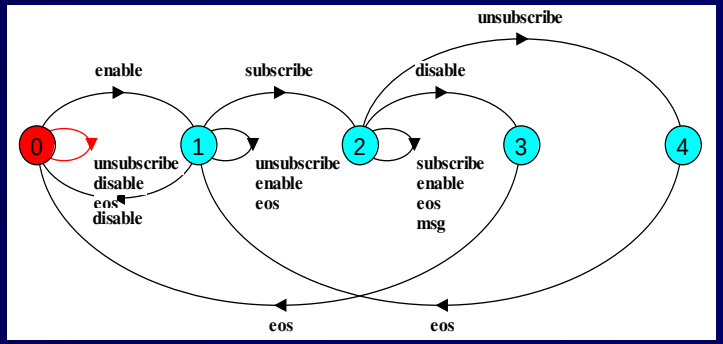
- Models are hard to build!



Synthesis from Heterogeneous Partial Specifications



Synthesis



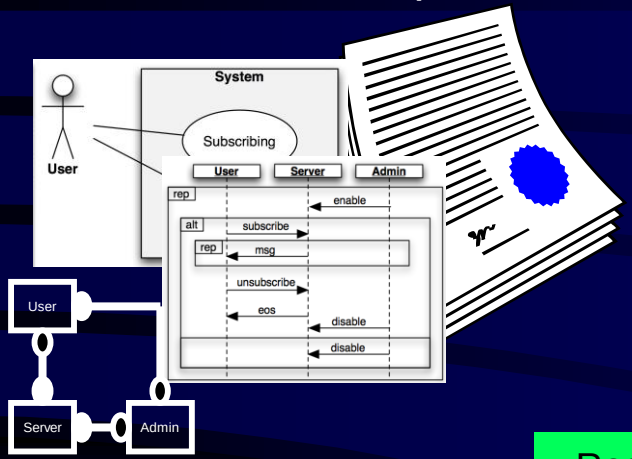
Use cases, Scenarios,
Architecture, Requirements,
Class Diagrams, Contracts,…

Behaviour models
Eg. Labelled Transition Systems

[TSE03][FSE04][TOSEM04]

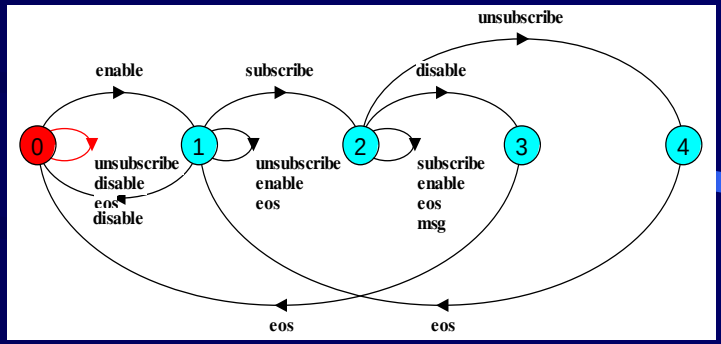
Semantic Mismatch

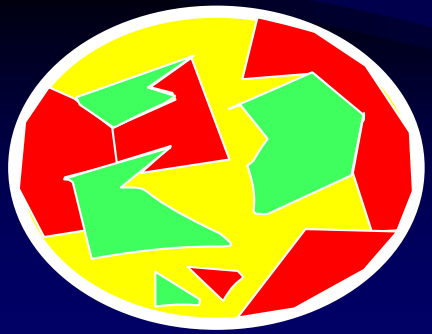
Partial Description



Synthesis

Complete Description

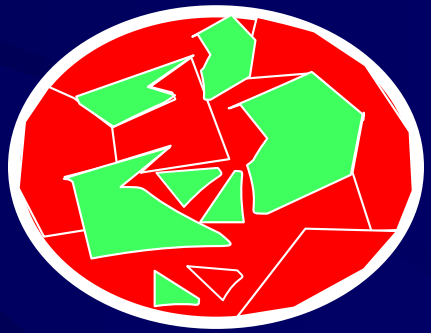




Required Behaviour

Undefined Behaviour

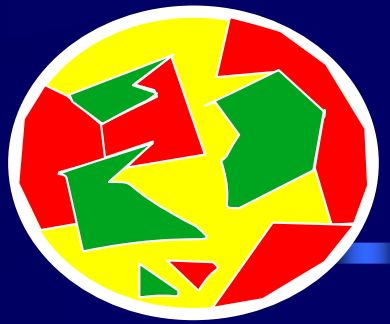
Proscribed Behaviour



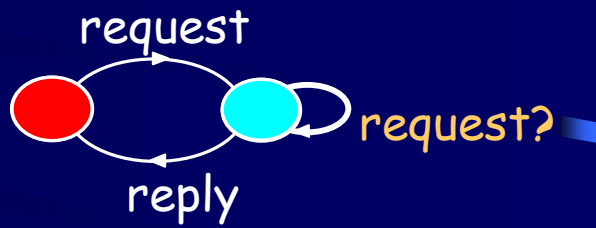
Required Behaviour

Proscribed Behaviour

Solution: Partial Behaviour Models



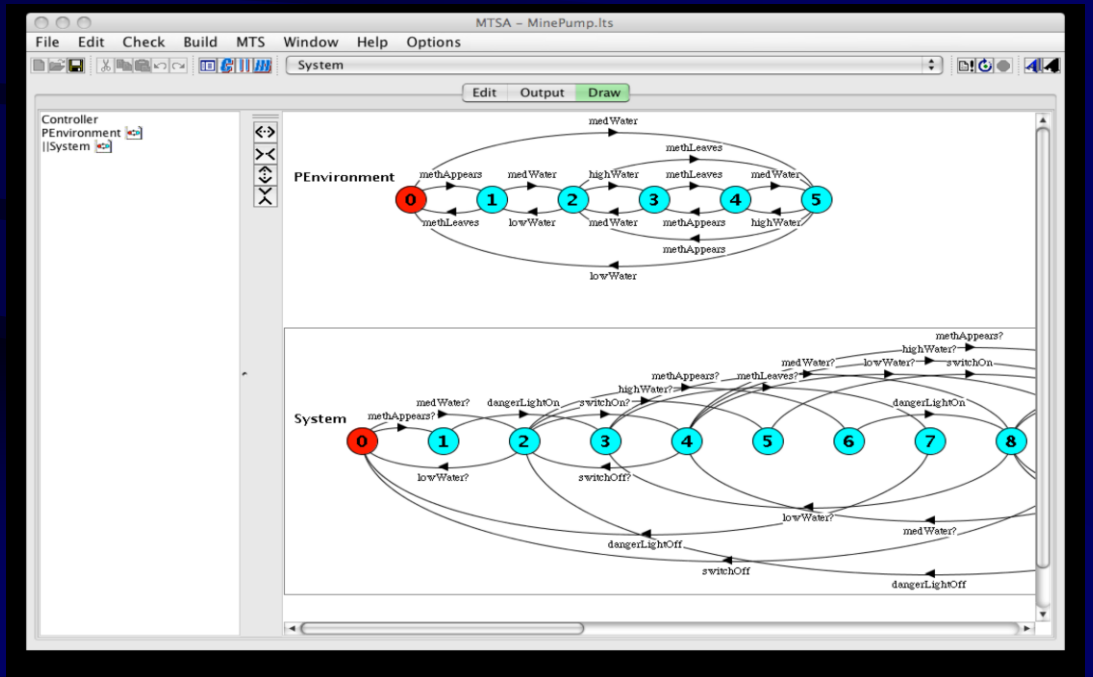
- Capable of distinguishing Required, Proscribed and Unknown behaviour
  - Eg. Modal Transition Systems
- Research threads
  - Refinement
  - Model Checking
  - Synthesis
  - Merge and Composition



[TSE09][FSE08][ASE08][ICTAC09][FM06]

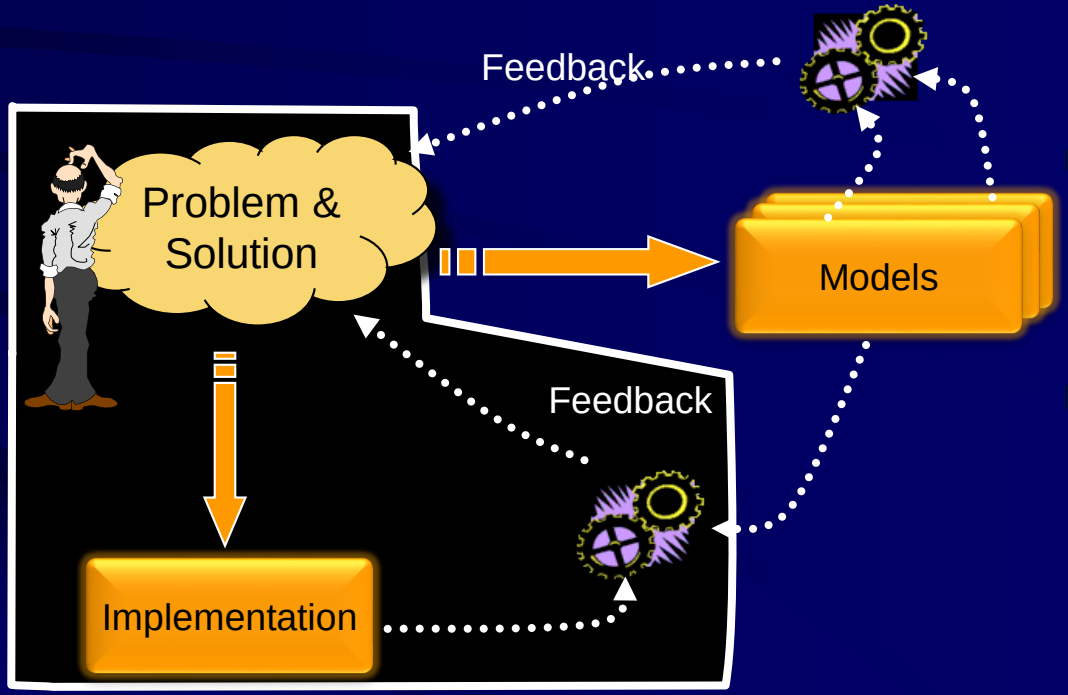
Tool Support

- MTS Model Checker
- Open source: http://sourceforge.net/projects/mtsa/

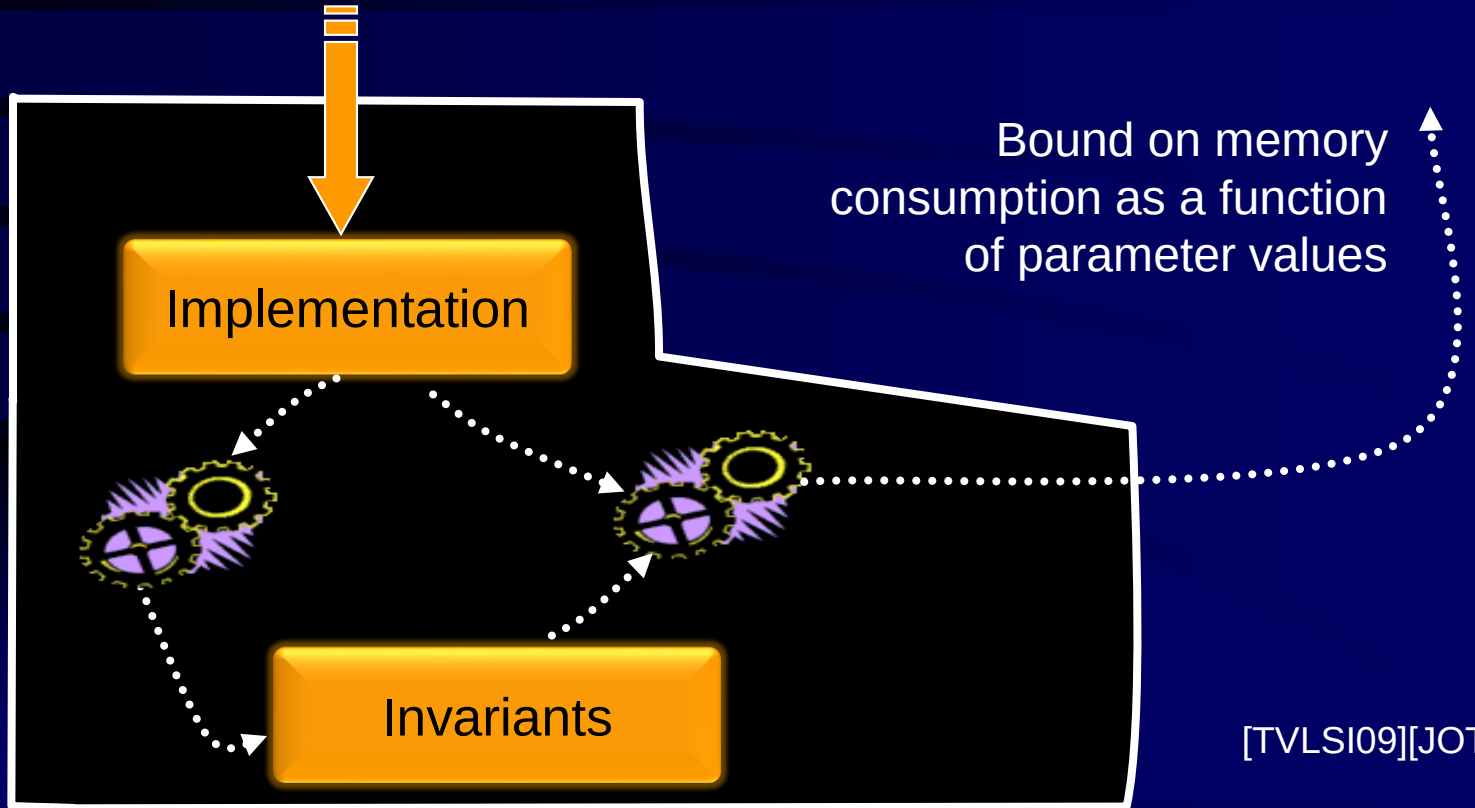


Theme 3: Program Analysis

- What can be said about the code?



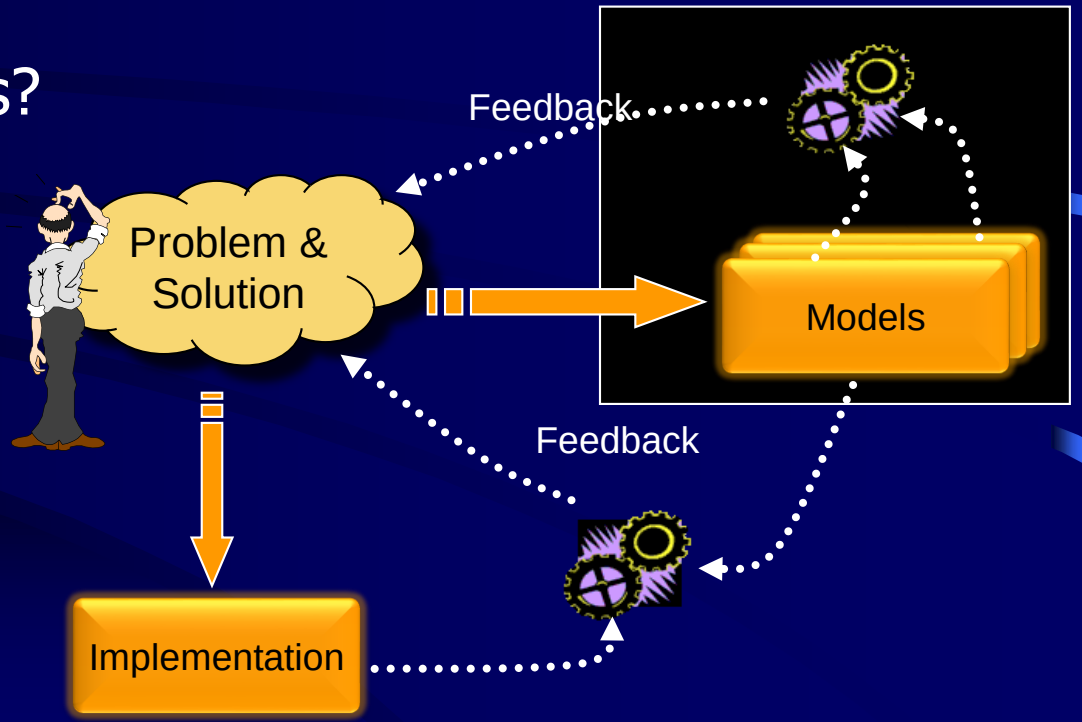
Automatic Generation of Memory Consumption Certificates



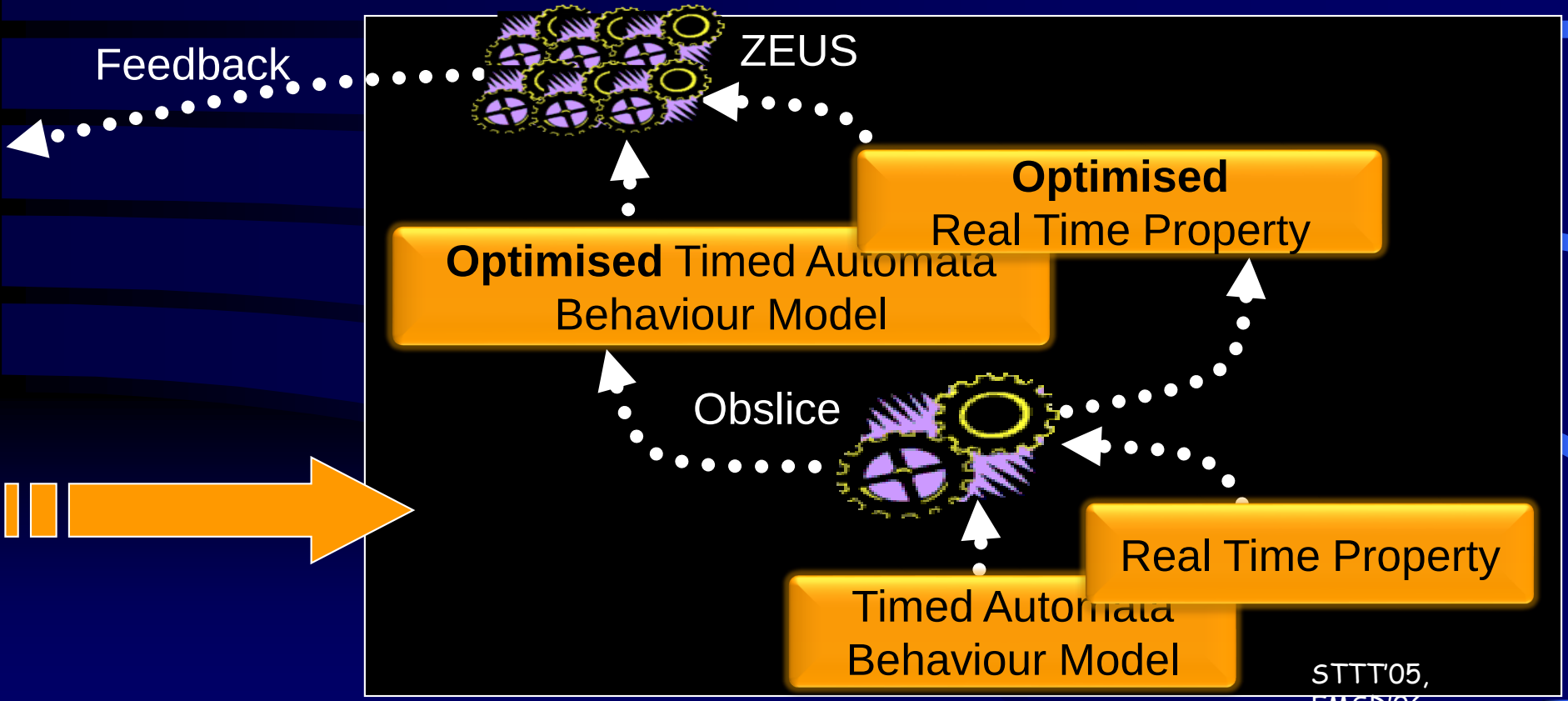
[TVLSI09][JOT08][JOT06]

Theme 4: Model Checking

- Can we increase scalability of model checking procedures?



ZEUS: Real Time Distributed Model Checking



Overview

- Technical areas
  - Model Extraction
  - Static Analysis
  - Memory usage prediction
  - Dynamic Analysis
  - (Distributed) Model Checking
  - Test-case generation
  - Test-guided model checking
  - Quantitative Modeling and Analysis
  - Machine learning
  - AOP
  - Model Synthesis
  - Partial Behaviour Models
- Application Domains
  - Real time systems
  - Service Oriented Architectures
  - Distributed and Concurrent systems
  - Object-oriented programs
  - Embedded systems
  - Dynamic and reconfigurable systems
- Software Engineering Activities
  - Requirements Engineering
  - Software Architecture
  - Testing
  - Design

The Foundations and Tools for Software Engineering Lab

Department of Computing, FCEN,
University of Buenos Aires, Argentina

Sebastian Uchitel

Submit to ICSE

Deadline for submissions to the technical track: September 6

http://www.sbs.co.za/ICSE2010/