Testbeds in Cyber-Physical Systems
Interfaces and Time Scales

Jonathan Sprinkle, PhD
With help from:
Kun Zhang (2015), Dr. Sean Whitsitt (2014), Dr. Xiao Qin (2013), Matt Bunting

THE UNIVERSITY OF ARIZONA

Different timescales*

Lifestyle change

Energy Bills

Gas Prices

Shipping

Giving Birth

Commute

OTC Pain relievers

River flow

Traffic flow

Car Navigation

Stability controller

malloc

Router

Timescale of feedback/coupling

Agriculture/ watering

Surgery

Electricity prices

Water Temperature

HVAC set point Mgmt

Java GC

Chip Thermal Mgmt

Computer Clock

* Not to scale. It's not like I plotted this in MATLAB or anything...

Testbed 1: Full-sized Ford Escape

Down to the wires and back again

- 2008 Ford Escape Hybrid
- Actuated by Torc Robotics
- CAN Bus reader with dedicated CompactRIO for control inputs
- 1.2kW power supply based off the Hybrid battery
- Equipped with
    - pause/stop modes for safety
    - emergency-stop: normally open held closed
    - dead-man's switch: executes e-stop when no message received in time frame

Featuring various hardware additions...


Velodyne 64e
3D lidar (~$80,000)
STATE OF ARIZONA FOR OFFICIAL USE ONLY
HYBRID
ByWire XGV

NoVaTel GPS/
IMU (~$25,000)
STATE OF ARIZONA FOR OFFICIAL USE ONLY
ByWire XGV
five star ford.com
Scottsdale
ESCAPE


STATE OF ARIZONA FOR OFFICIAL USE ONLY
BYWIRE XGV

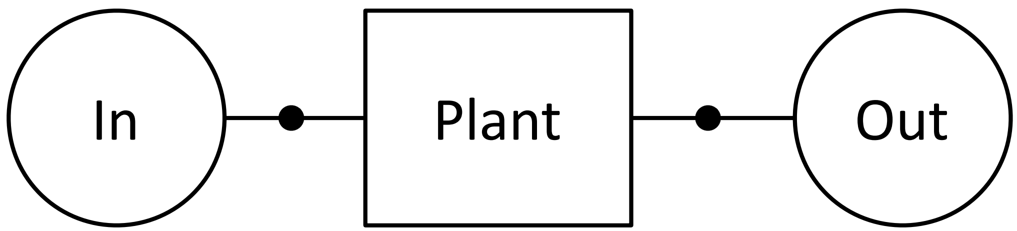
Testbed interface

- Available control inputs:
- open loop
  - throttle
  - steering
  - brake
  - shifter
- Closed loop
  - desired speed
  - desired acceleration
- Devices
  - start/disable vehicle
  - left/right turn
  - horn (yes!)
  - lights

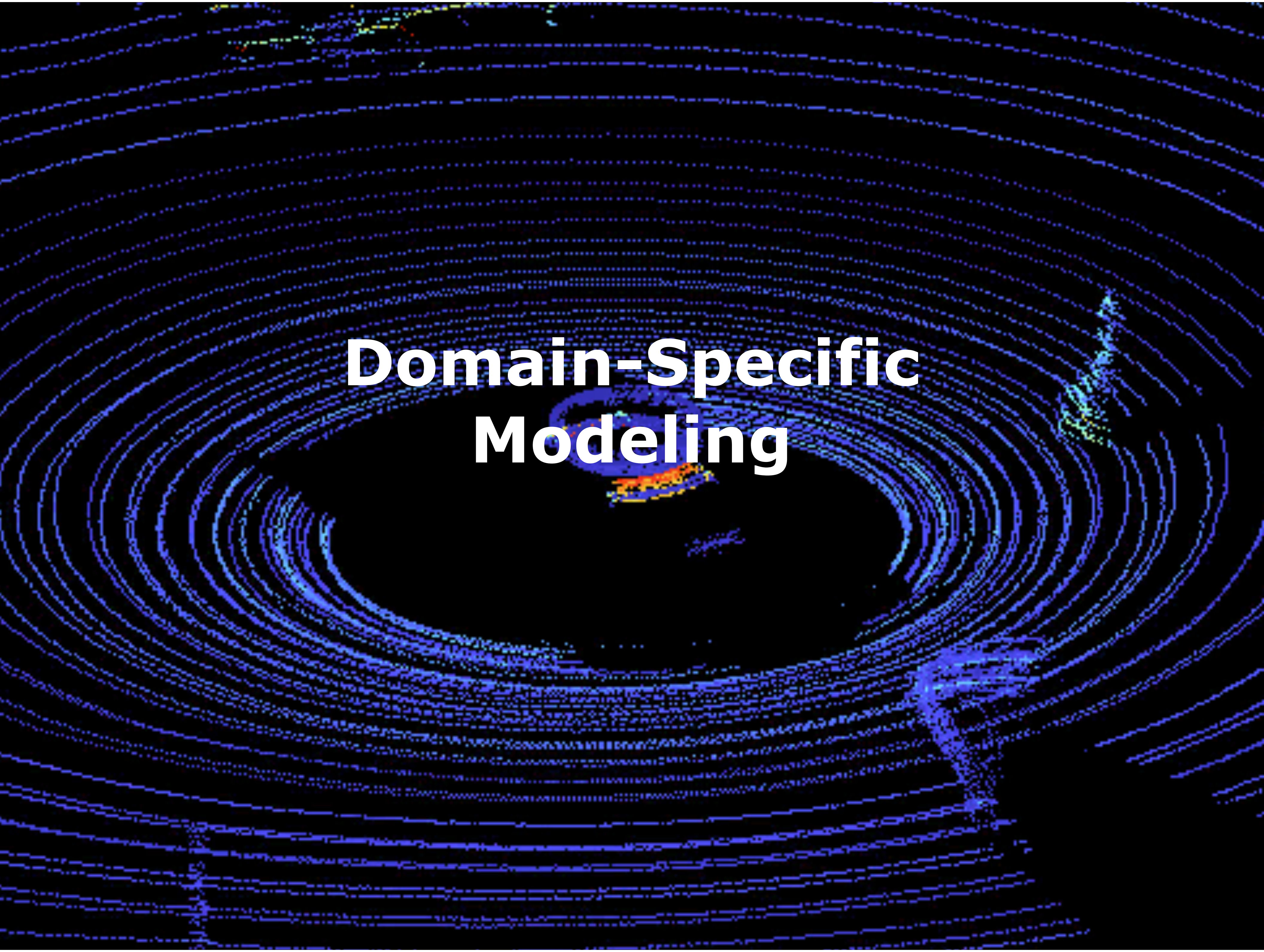
- Available state data:
- Speed
  - individual wheels
  - vehicle speed
- Steering angle
- Throttle percentage
- Brake percentage
- Shift position

Interface:

With interfaces, we can model.

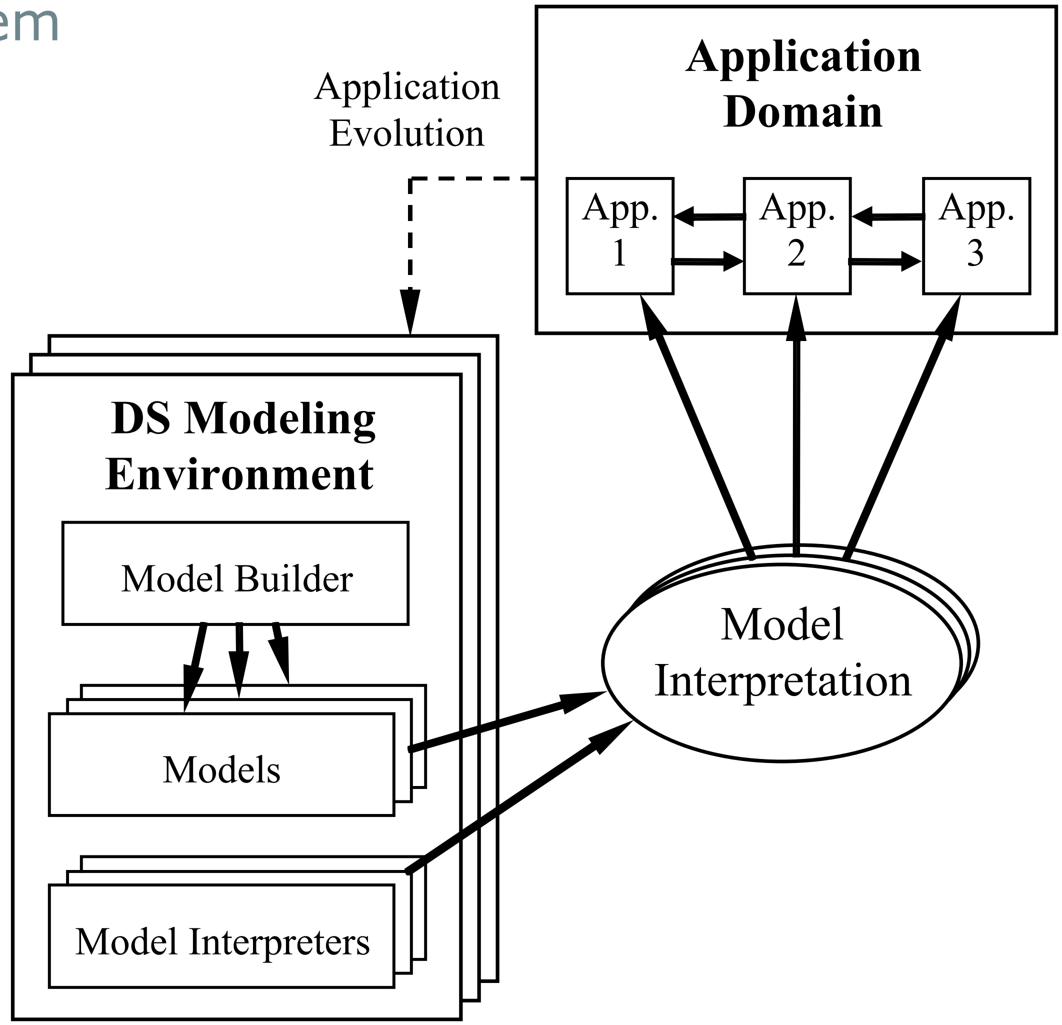


Domain-Specific Modeling



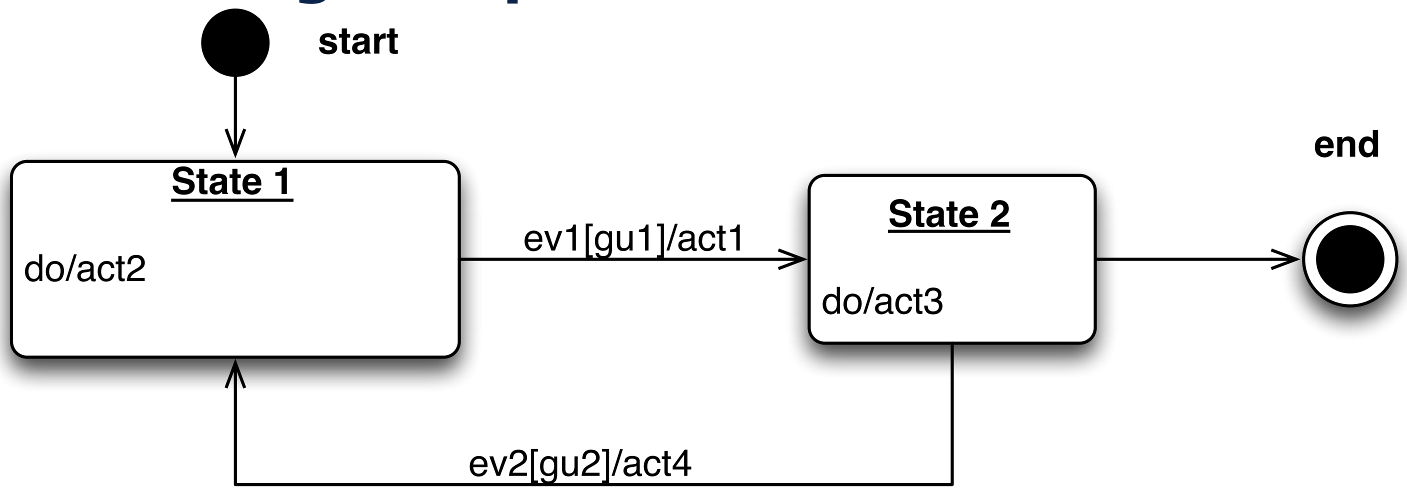
Domain-Specific Modeling

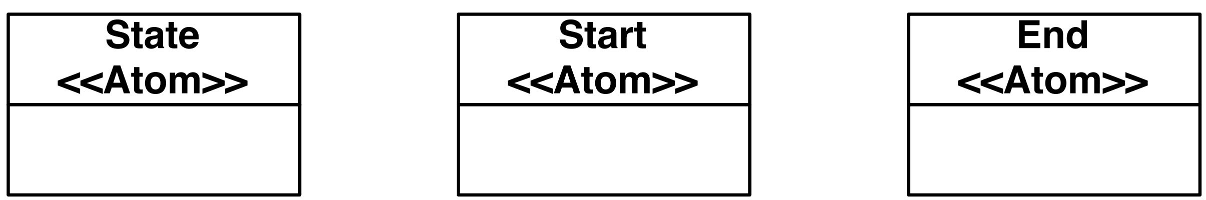
- Create model of the system
- Perform
  - Analysis
  - Architecture exploration
  - Simulation
- Generate
  - Configuration
  - Code
  - Executables
- From the same models!



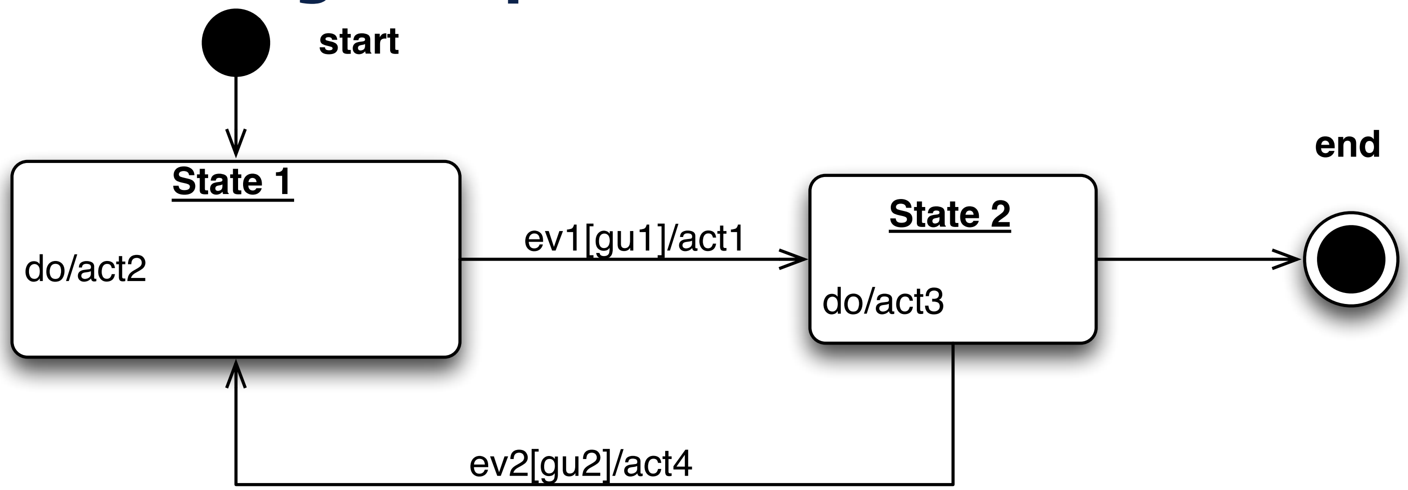
Example Domains & Environments:
- VLSI Layout (e.g., Altera)
- Engg Drawing (e.g., AutoCAD)
- Physical Modeling (e.g., SolidWorks)
- Signal Processing (e.g., LabVIEW)
- Controls (e.g., Simulink)

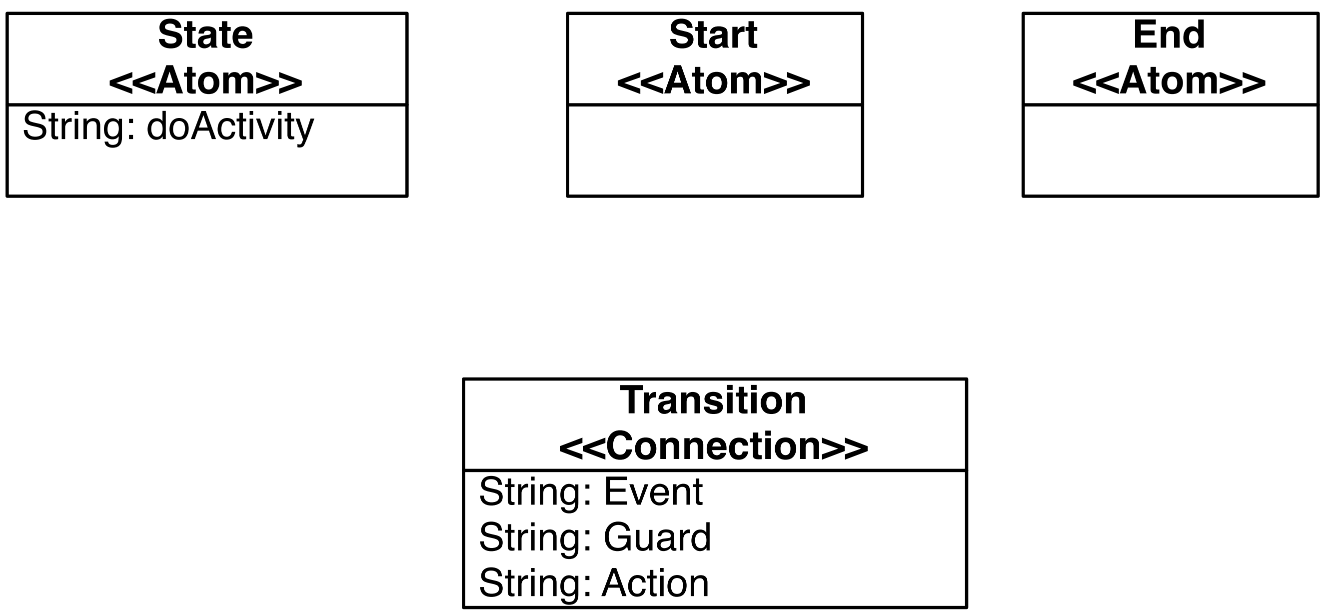
Thought Experiment: State Models



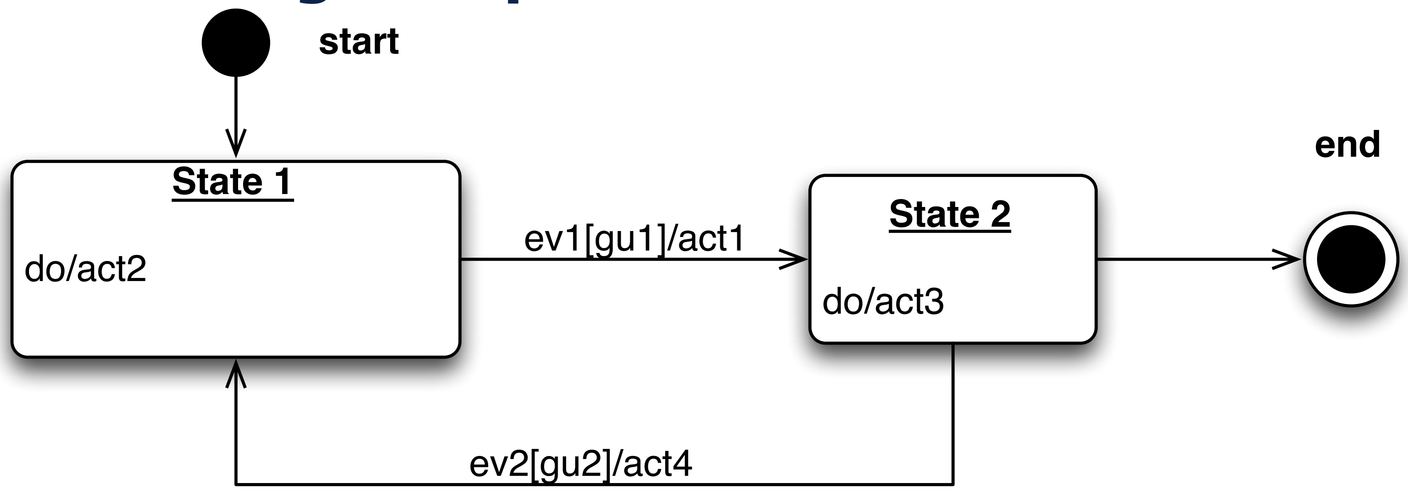


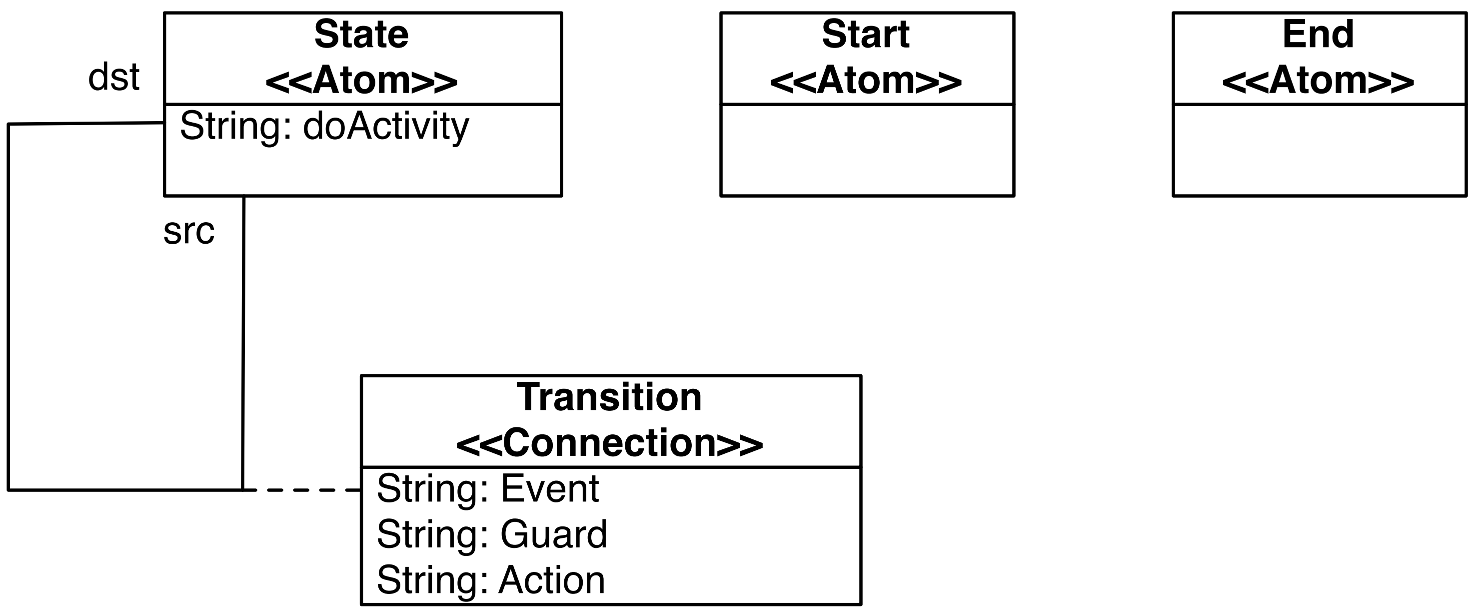
Thought Experiment: State Models



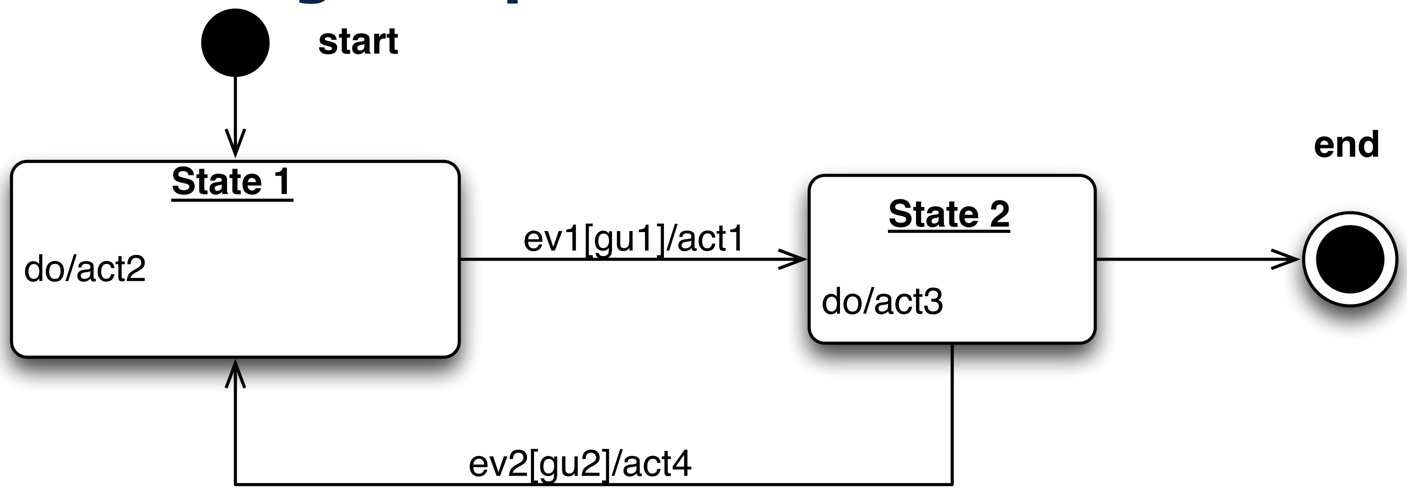


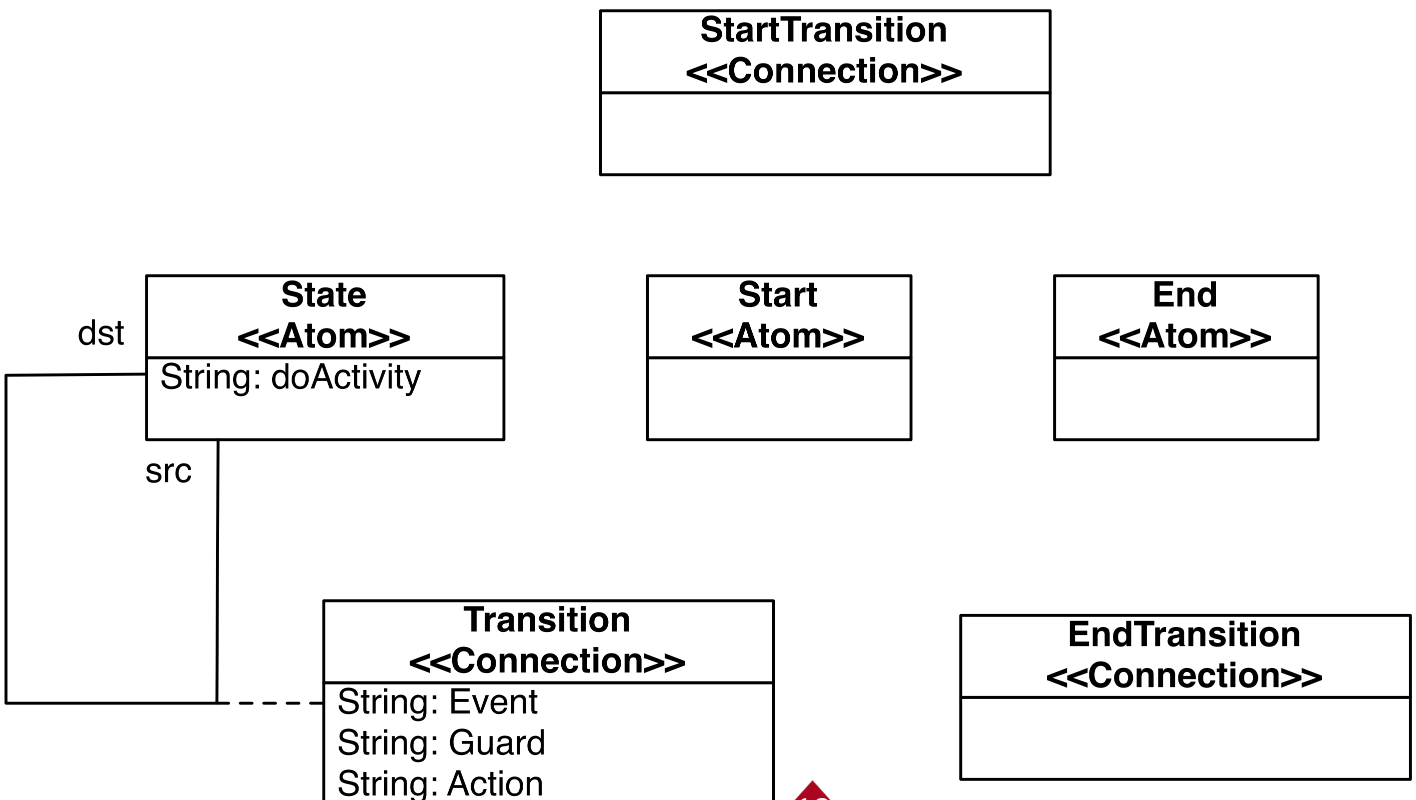
Thought Experiment: State Models



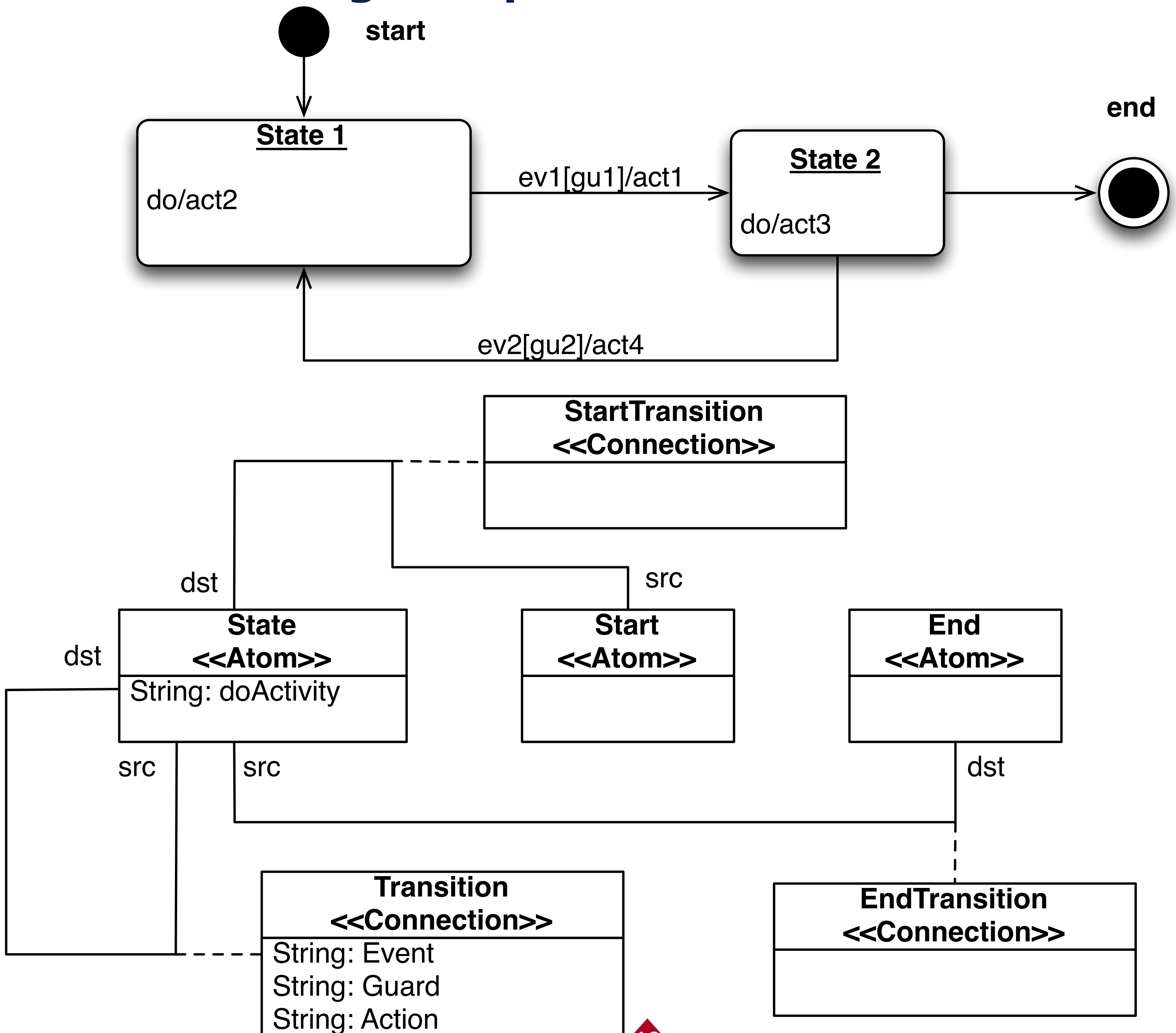


Thought Experiment: State Models

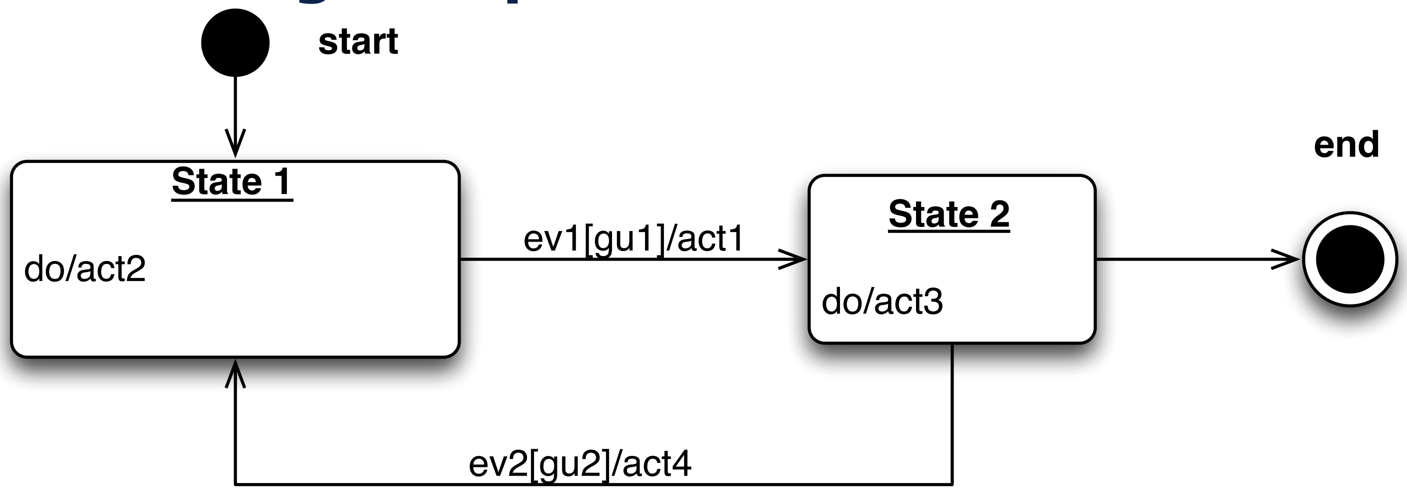


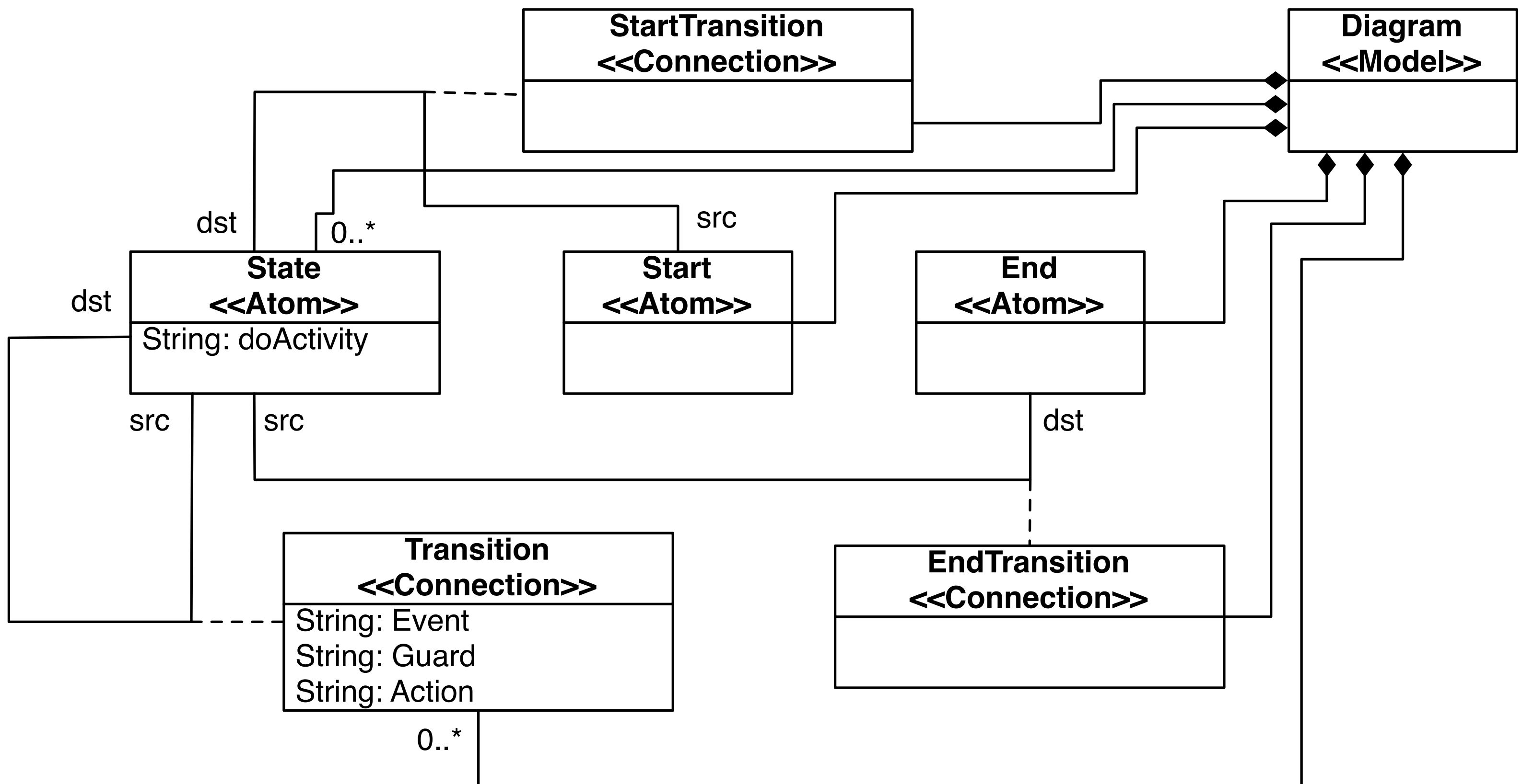


Thought Experiment: State Models



Thought Experiment: State Models





// Template Code Generation

```
/*
 *  #MESSAGE_FUNCTION_NAME#.c
 *  OpenJaus
 *
 *  Created by JausMessageML_Interpreter on #DATE#.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "jaus.h"
#include "#MESSAGE_FUNCTION_NAME#.h"

static const int commandCode = #MESSAGE_COMMAND_CODE#;
static const int maxDataSizeBytes = 0;

static JausBoolean headerFromBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int
bufferSizeBytes);
static JausBoolean headerToBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int bufferSizeBytes);
static int headerToString(#MESSAGE_OBJECT_NAME# message, char **buf);

static JausBoolean dataFromBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int bufferSizeBytes);
static int dataToBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int bufferSizeBytes);
static void dataInitialize(#MESSAGE_OBJECT_NAME# message);
static void dataDestroy(#MESSAGE_OBJECT_NAME# message);
static unsigned int dataSize(#MESSAGE_OBJECT_NAME# message);

// ************************************************************************************************************** //
//                                      USER CONFIGURED FUNCTIONS
// ************************************************************************************************************** //

// Initializes the message-specific fields
static void dataInitialize(#MESSAGE_OBJECT_NAME# message)
{
// generated code
#CODE_DATA_INIT#
// end generated code
}
```

Idea: Output code usually correlates to model structs

```
#includes#

class #class_name# {
    public:
        int messageID;
        #public_vars#

        #public_functions#

    private:
        #private_vars#

        #private_functions#
}
```

⇒

```
#include <math.h>
using namespace std;

class MixedMessage {
    public:
        int messageID;
        double sentSignal;
        double actualSignal;

        double signal();
        double lieAboutSignal();

    private:
        int signalID;
}
```

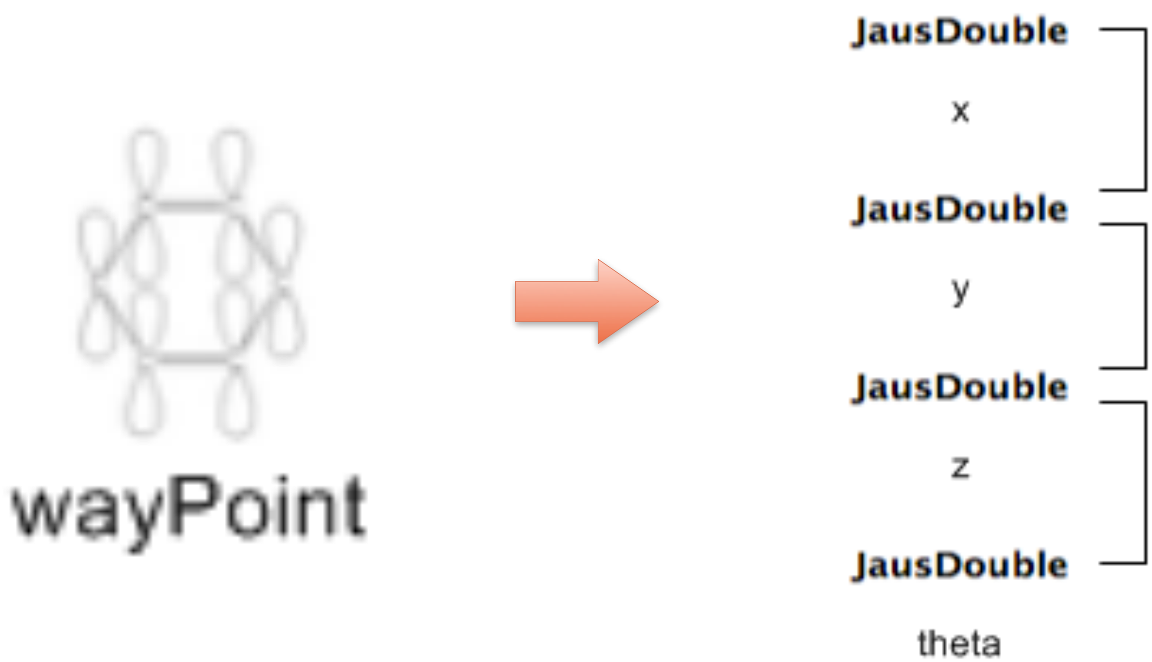
Idea 2: we probably already have output code

```
*****************************************************************************/
// File Name: setDataLinkSelectMessage.c
//
// Written By: Danny Kent (jaus AT dannykent DOT com), Tom Galluzzo (galluzzo AT gmail DOT com)
//
// Version: 3.3.0b
//
// Date: 09/08/09
//
// Description: This file defines the functionality of a SetDataLinkSelectMessage

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "jaus.h"

static const int commandCode = JAUS_SET_DATA_LINK_SELECT;
static const int maxDataSizeBytes = 1;

static JausBoolean headerFromBuffer(SetDataLinkSelectMessage message, unsigned char *buffer, unsigned int bufferSizeBytes);
static JausBoolean headerToBuffer(SetDataLinkSelectMessage message, unsigned char *buffer, unsigned int bufferSizeBytes);
static int headerToString(SetDataLinkSelectMessage message, char **buf);

static JausBoolean dataFromBuffer(SetDataLinkSelectMessage message, unsigned char *buffer, unsigned int bufferSizeBytes);
static int dataToBuffer(SetDataLinkSelectMessage message, unsigned char *buffer, unsigned int bufferSizeBytes);
static void dataInitialize(SetDataLinkSelectMessage message);
static void dataDestroy(SetDataLinkSelectMessage message);
static unsigned int dataSize(SetDataLinkSelectMessage message);

// ...
```

For formulaic code generation…

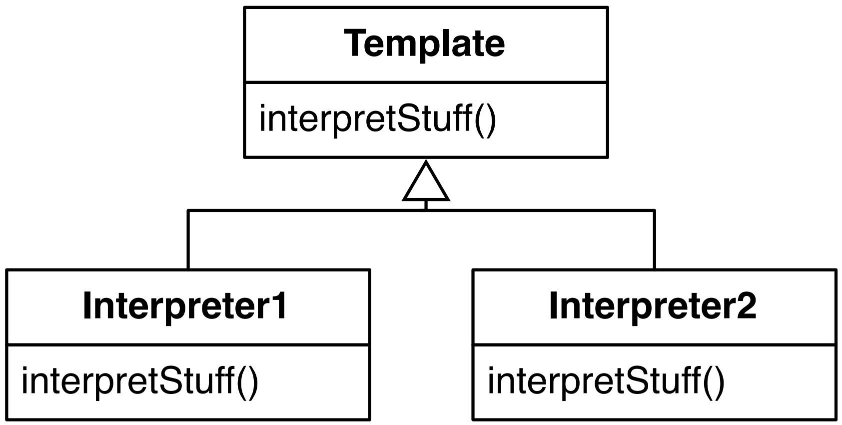
```
/*
 *  #MESSAGE_FUNCTION_NAME#.c
 *  OpenJaus
 *
 *  Created by JausMessageML_Interpreter on #DATE#.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "jaus.h"
#include "#MESSAGE_FUNCTION_NAME#.h"

static const int commandCode = #MESSAGE_COMMAND_CODE#;
static const int maxDataSizeBytes = 0;

static JausBoolean headerFromBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int bufferSizeBytes);
static JausBoolean headerToBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int bufferSizeBytes);
static int headerToString(#MESSAGE_OBJECT_NAME# message, char **buf);

static JausBoolean dataFromBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int bufferSizeBytes);
static int dataToBuffer(#MESSAGE_OBJECT_NAME# message, unsigned char *buffer, unsigned int bufferSizeBytes);
static void dataInitialize(#MESSAGE_OBJECT_NAME# message);
static void dataDestroy(#MESSAGE_OBJECT_NAME# message);
static unsigned int dataSize(#MESSAGE_OBJECT_NAME# message);

// ...
```

A new message with a set structure is now straightforward



What about more complex output s/w architectures?

- Build the parts
- Insert the parts
- Output an artifact
- Repeat



E.g., MATLAB Component

```
% MATLAB Level 2 S-Function #S_FUNCTION_NAME#_matlab
function  #S_FUNCTION_NAME#_matlab( block )
   setup(block);
end

function setup(block)

% Register number of input and output ports
#S_FUNCTION_INPUT_PORTS#
#S_FUNCTION_OUTPUT_PORTS#
  %block.SetPreCompInpPortInfoToDynamic;
  %block.SetPreCompOutPortInfoToDynamic;

  % Set block sample time to variable sample time
  block.SampleTimes = [0 0];

  % Set the block simStateCompliance to default (i.e., same as a built-in block)
  block.SimStateCompliance = 'DefaultSimState';

  % Register methods
  block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);
  block.RegBlockMethod('InitializeConditions',    @InitConditions);
  block.RegBlockMethod('Outputs',                 @Output);
  block.RegBlockMethod('Update',                  @Update);

end

function DoPostPropSetup(block)
#S_FUNCTION_DWORK_VECTORS_SETUP#
end

function InitConditions(block)
#S_FUNCTION_INIT_DATA#
end
```

E.g., JAUS Component

```
//#define DEBUG_QUERY_ONLY_NO_SERVICE_CONNECTION 1

OjCmpt #COMPONENT_NAME#::create(std::string prettyName) {
	OjCmpt result;
	#COMPONENT_NAME#Data *data = NULL;
	JausAddress vcAddr;
	// it is unbelieveable that I have to do this...what a HACK
	char szName[256];
	strcpy(szName, prettyName.c_str());
	// now, we create it using the global (groan) methods
	result = ojCmptCreate(szName, #COMPONENT_ID#, THREAD_DESIRED_RATE_HZ);

	if(result == NULL) {
		// something bad happened...
		std::cout << "Error starting #COMPONENT_NAME#...aborting." << std::endl;
		return result;
	} else {
		// ... omitted for brevity
		data = (#COMPONENT_NAME#Data*)malloc(sizeof(#COMPONENT_NAME#Data));

// begin generated code
#DATA_INIT#
// end generated code

// begin generated code
#SUPPORTED_CONNECTIONS#
// end generated code

// begin generated code
#MESSAGE_CALLBACKS#
// end generated code

// begin generated code
#ESTABLISH_SC#
// end generated code

		jausAddressDestroy(vcAddr)
```

Interpreter looks like…

```
void MessagingModelInterpreter::interpretCPP(std::string projectDirectory, JausMessageML_BON::Component component) {
	// create the skeleton files for each necessary component file
	Console::Out::WriteLine("InterpretComponent CPP Begin");
	Skeleton hfile = Skeleton::Skeleton();
	Skeleton cppfile = Skeleton::Skeleton();
	Skeleton mainfile = Skeleton::Skeleton();
	hfile.load(DEFAULT_COMPONENT_H);
	cppfile.load(DEFAULT_COMPONENT_CPP);
	mainfile.load(DEFAULT_COMPONENT_MAIN);

	// ....

	Console::Out::WriteLine("Supported Service Connections");
	string supportedSC = generateAddSupportedSCs(models);
	cppfile.replace(SUPPORTED_CONNECTIONS, supportedSC);

	Console::Out::WriteLine("Message Callbacks");
	string callbacks = generateMessageCallbacks(models);
	cppfile.replace(MESSAGE_CALLBACKS, callbacks);

	Console::Out::WriteLine("Component Message Includes");
	// generate the message includes code
	string messageIncludes = generateMessageIncludes(models);
	hfile.replace(MESSAGE_INCLUDES, messageIncludes);

	Console::Out::WriteLine("Command Functions");
	set<Command> commands = component->getCommand();
	string commandFunctions = generateCommandFunctions(commands, component);
	cppfile.replace(COMMAND_FUNCTIONS, commandFunctions);

	Console::Out::WriteLine("Command Prototypes");
	string commandPrototypes = generateCommandPrototypes(commands);
	hfile.replace(COMMAND_PROTOTYPES, commandPrototypes);
	...
```

Example: Generating Message Includes

```
std::string MessagingModelInterpreter::generateMessageIncludes(std::set<Model> models) {
	Console::Out::WriteLine("GenerateMessageIncludes Begin.");
	std::string code = std::string();

	// generate code for each model
	for (std::set<Model>::iterator ii = models.begin(); ii != models.end(); ii++) {
		Model model = (Model)*ii;
		std::string out = string("GenerateMessageIncludes: ");
		out += model->getName();
		Console::Out::WriteLine(out.c_str());
		if (hasRole(model, "Port")) {
			Port port = (Port)model;
			std::set<JausMessage> messages = port->getJausMessage();
			for (std::set<JausMessage>::iterator jj = messages.begin(); jj != messages.end(); jj++) {
				JausMessage message = (JausMessage)*jj;
				Attribute attr = message->getAttribute(MESSAGE_TEMPLATE_FOLDER_NAME);
				code += "#include \"../";
				code += attr->getStringValue(true);
				code += "/";
				code += MessageInterpreter::generateFunctionName(message->getName());
				code += ".h\"\n";
			}
		}
	}

	Console::Out::WriteLine("GenerateMessageIncludes End.");
	return code;
}
```

With models, we extend the user base

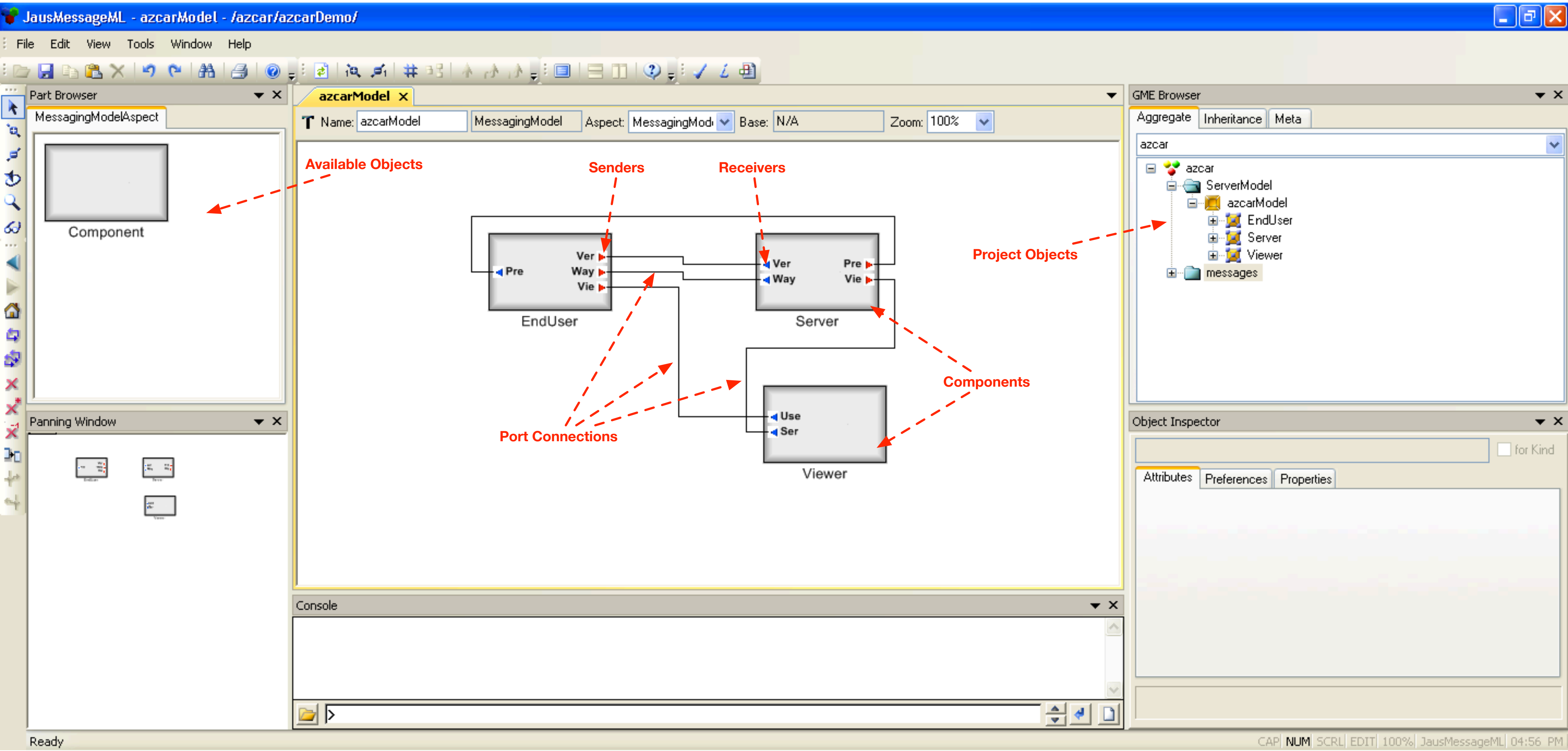
So far, 18 undergraduate REU participants have been able to use the testbed as part of a 10-week NSF Program

CAT Vehicle
2014

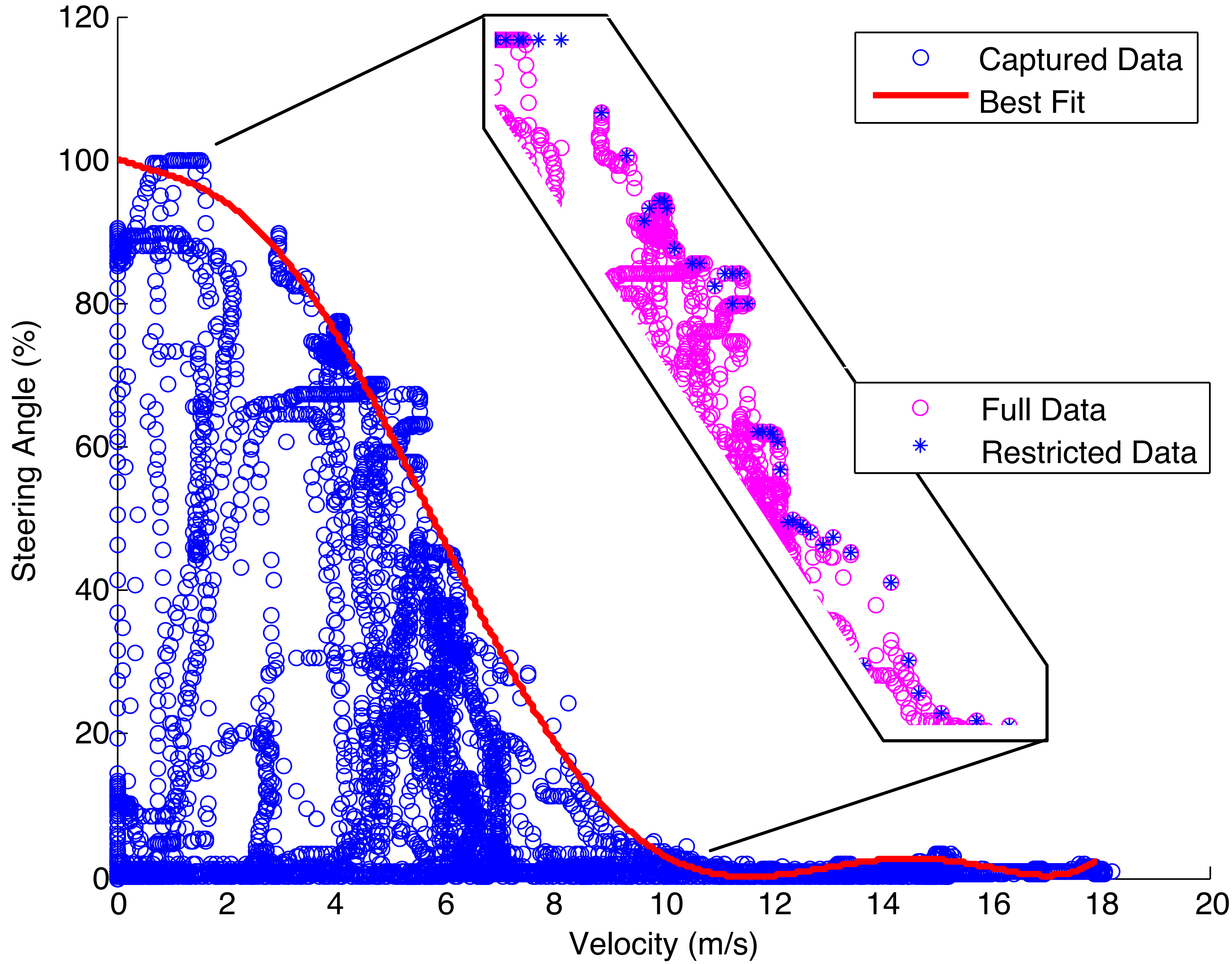
Visit http://catvehicle.arizona.edu/ to learn more.


Testbed for Research!
ByWire XGV

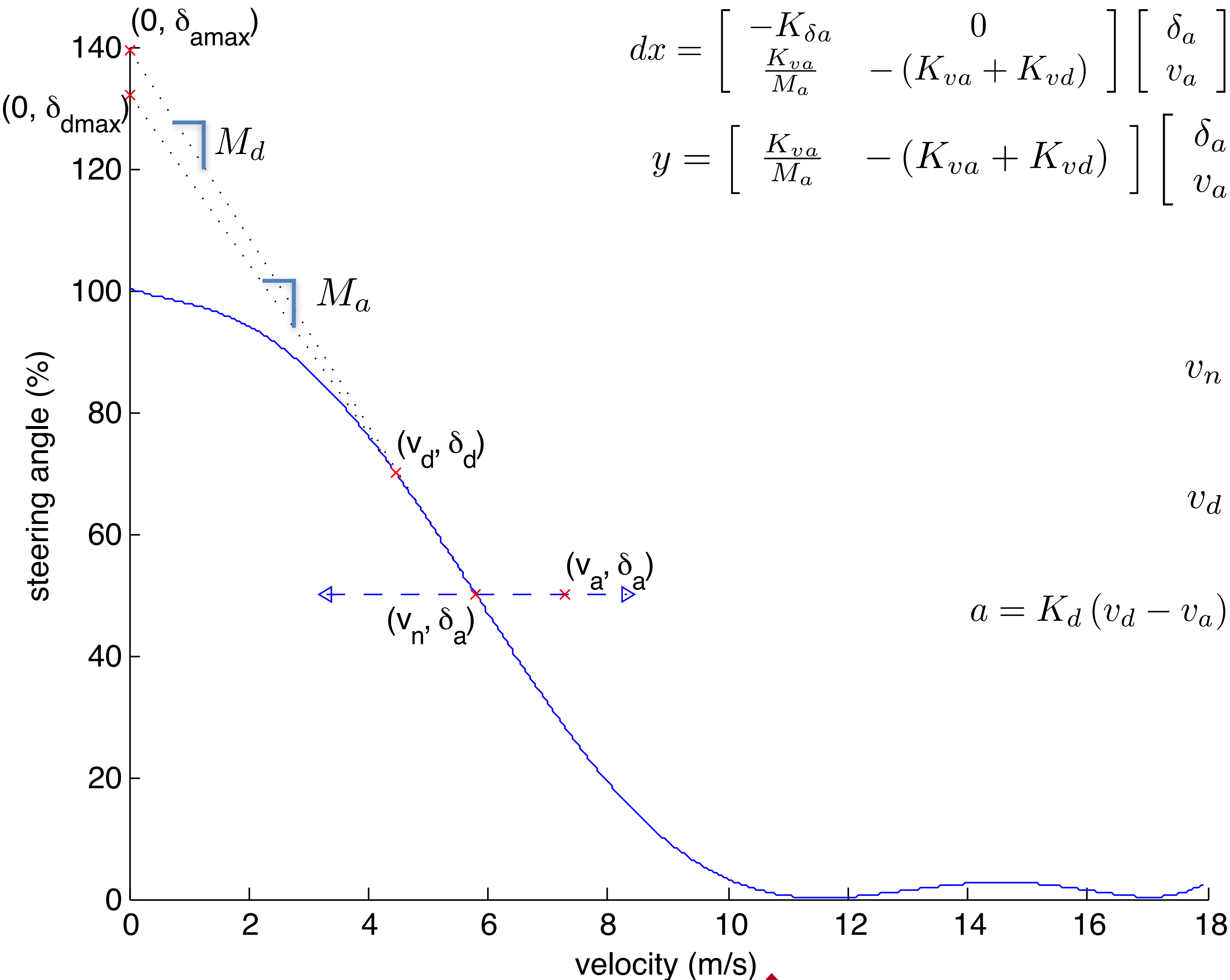
Example model of component interconnection



Gather data about driving behaviors when turning



Take the fit data and utilize linearization techniques



$$dx = \begin{bmatrix} -K_{\delta a} & 0 \\ \frac{K_{va}}{M_a} & -(K_{va}+K_{vd}) \end{bmatrix} \begin{bmatrix} \delta_a \\ v_a \end{bmatrix} + \begin{bmatrix} -K_{\delta d} \\ \frac{K_{vd}}{M_d} \end{bmatrix} \delta_d$$

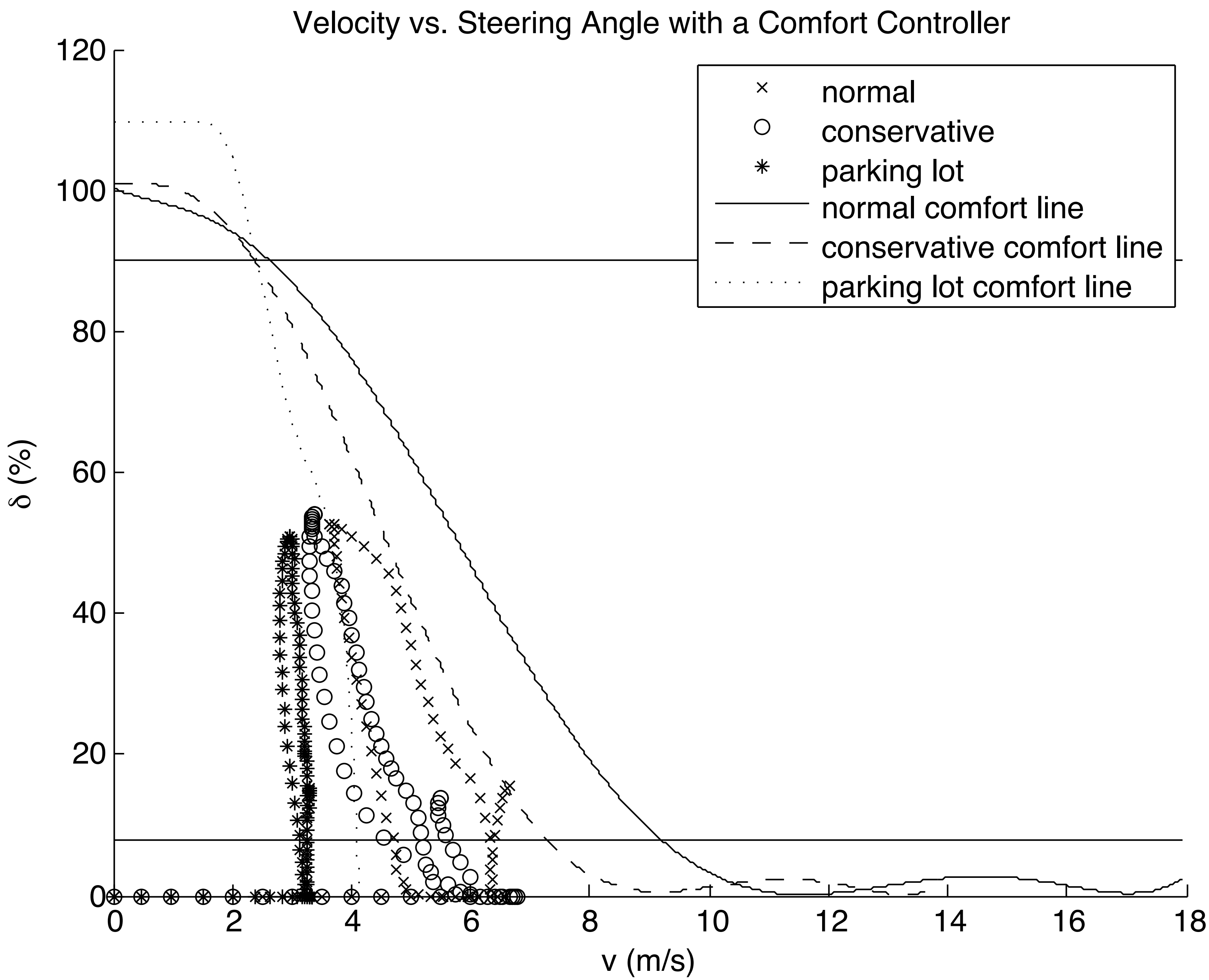
$$y = \begin{bmatrix} \frac{K_{va}}{M_a} & -(K_{va}+K_{vd}) \end{bmatrix} \begin{bmatrix} \delta_a \\ v_a \end{bmatrix} + \begin{bmatrix} \frac{K_{vd}}{M_d} \end{bmatrix} \delta_d$$

$$v_n = \frac{\delta_a - \delta_{amax}}{M_a}$$

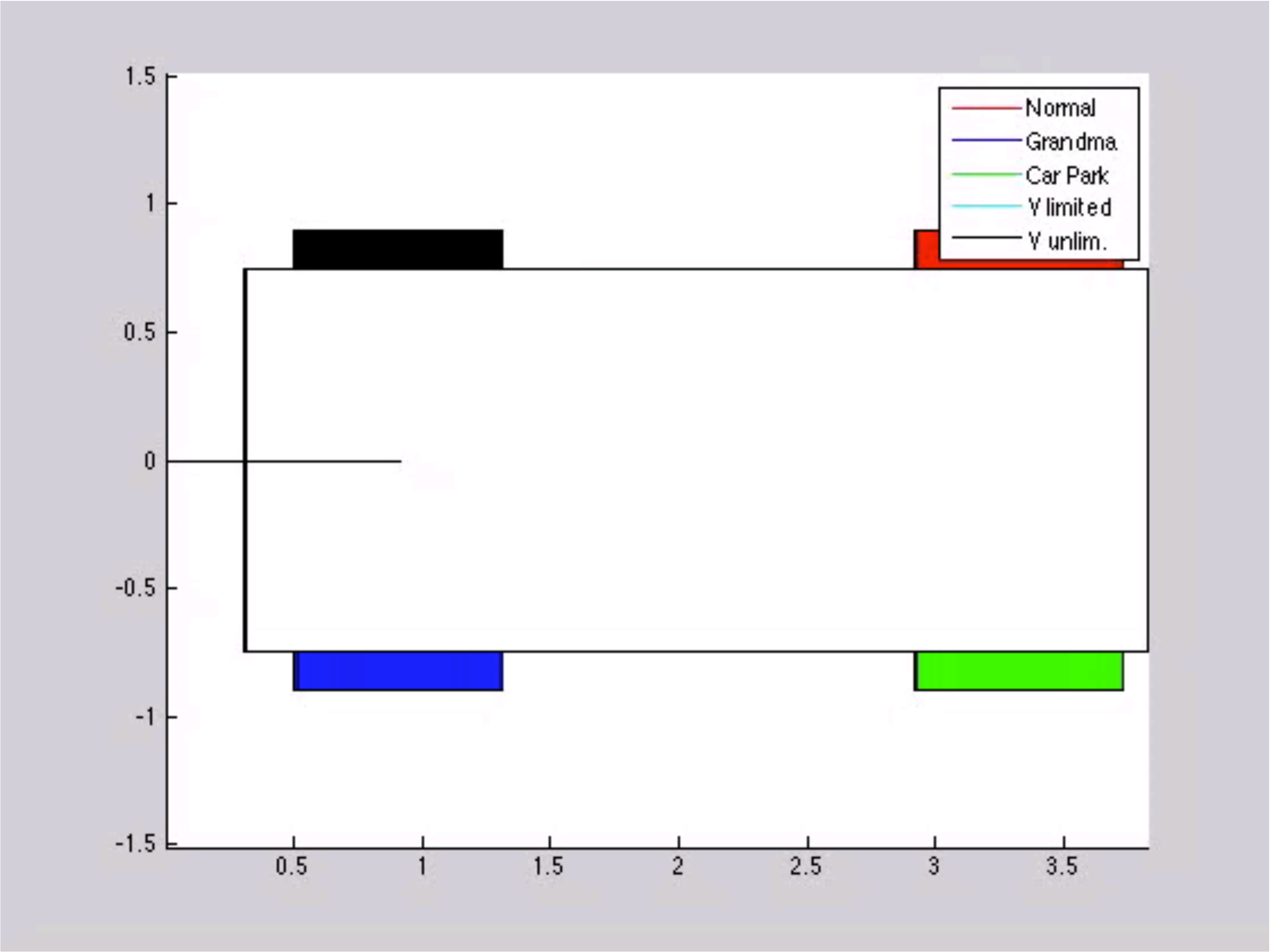
$$v_d = \frac{\delta_d - \delta_{dmax}}{M_d}$$

$$a = K_d(v_d - v_a) + K_a(v_n - v_a)$$

Scatter plot with comfort controller

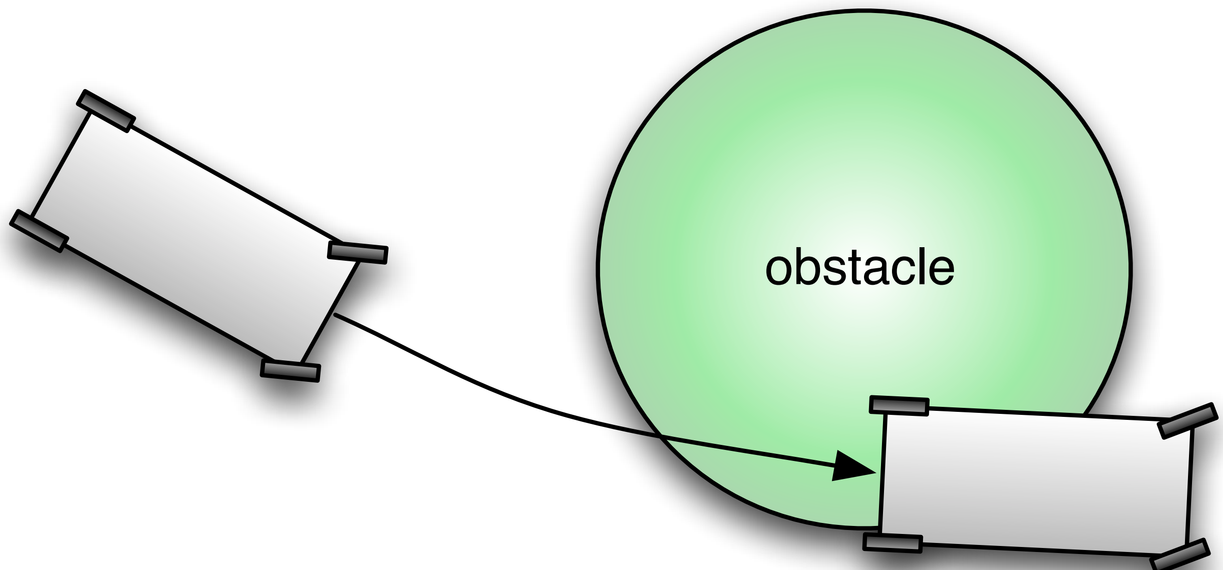


Simulation


Normal
Grandma
Car Park
V limited
V unlim.
1.5
1
0.5
0
-0.5
-1
-1.5
0.5
1
1.5
2
2.5
3
3.5

Problems: MPC return time

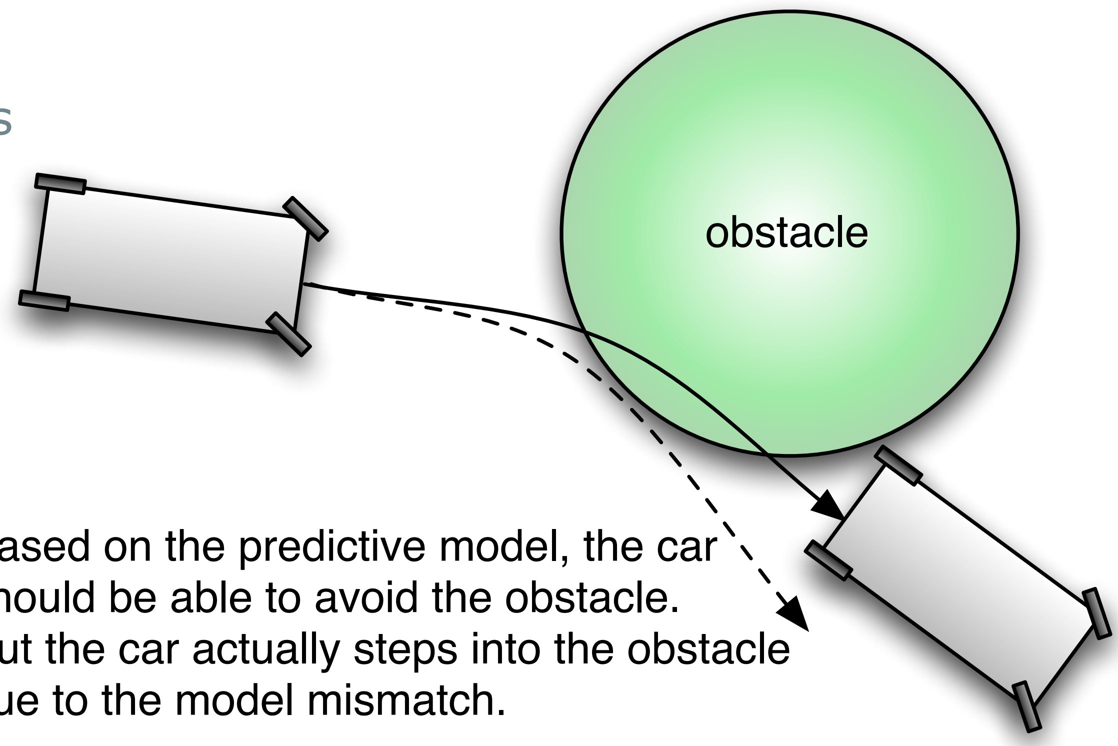
- A complex model may introduce predictive accuracy,
- however, this increases the computational burden.
- Especially under high speed, the system cannot tolerate a slow return rate.



The car is executing the previous control sequence while waiting for the new control sequence

Problems: control accuracy

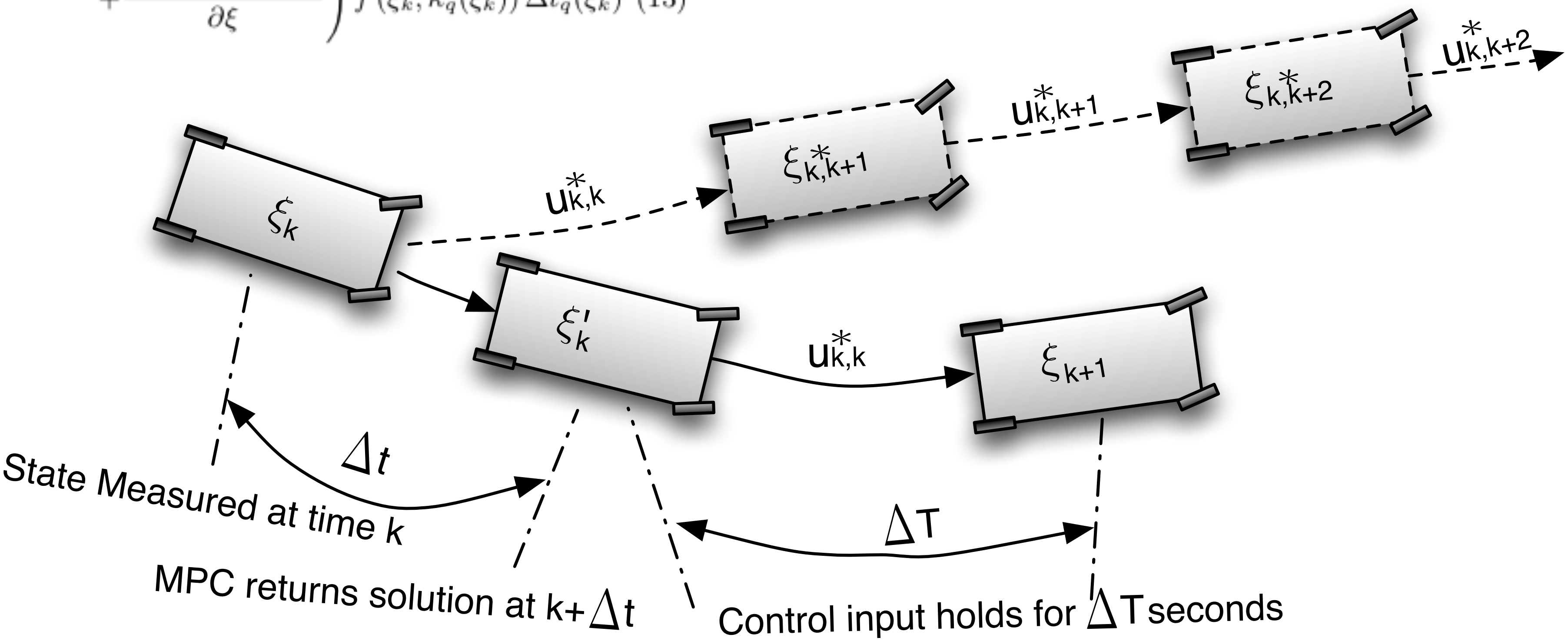
- The return time problem can be addressed via model reduction,
  - potential drawback is higher model mismatch.
- Model mismatch can also introduce problems.
  - Wrong prediction
  - Infeasible trajectories



Problem Modeling

$$\xi'_k \approx \xi_k + f(\xi_k, u^*_{k-1,k})\Delta t$$

$$\begin{aligned}
\xi_{k+1} &= \hat{f}\left(\xi'_k, \kappa_q(\xi_k)\right) = \hat{f}_q\left(\xi'_k, \kappa_q(\xi_k)\right) + \hat{\Gamma}_q(\xi'_k) \\
&\approx \hat{f}_q\left(\xi_k, \kappa_q(\xi_k)\right) + \hat{\Gamma}_q(\xi_k) + \left(\frac{\partial \hat{f}_q\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi}\right. \\
&\quad \left. + \frac{\partial \hat{\Gamma}_q\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi}\right) f\left(\xi_k, \kappa_q(\xi_k)\right) \Delta t_q(\xi_k) \quad (13)
\end{aligned}$$



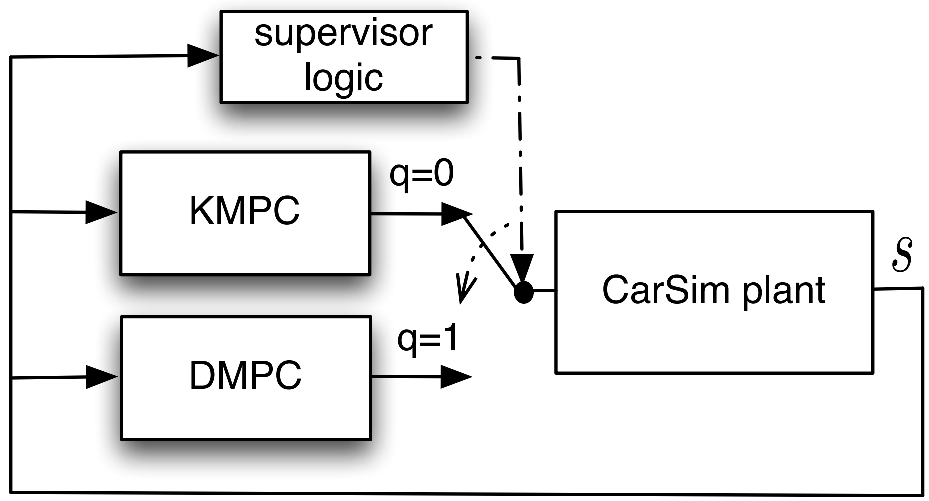
Problem Statement

- Our target is to ensure bounded MPC return time while maintaining the control accuracy via the design of hybrid MPC. The problem is to find such hybrid logic.

  $\|\xi_{k+1} - \xi^{q*}_{k,k+1}\|$

- Uncontrollable Divergence Def:

  $q = \underset{q}{\operatorname{argmin}} \|\xi_{k+1} - \xi^{q*}_{k,k+1}\|$



Problem: *Select a model from the family of vehicle models* $\{\hat{f}_q\}_{q\in\mathbb{Q}}$ *such that the error (or divergence) between the state of the plant* (ξ) *and of the model* (ξ_k) *obtained with the same inputs is minimized.*

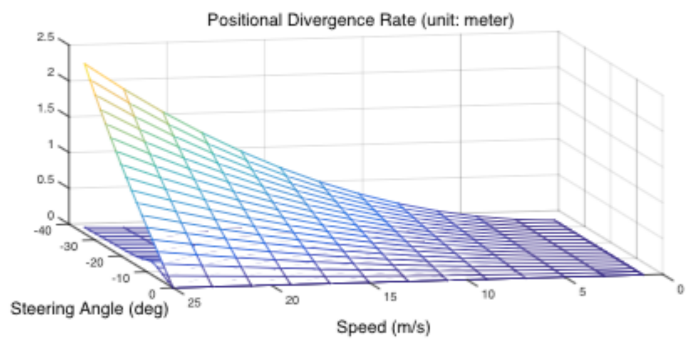
Hybrid MPC Design

- Two models are selected as the predictive model: (i) kinematic model, and (ii) dynamic model. Thus, two MPCs are generated: KMPC and DMPC. They are used in the hybrid MPC.

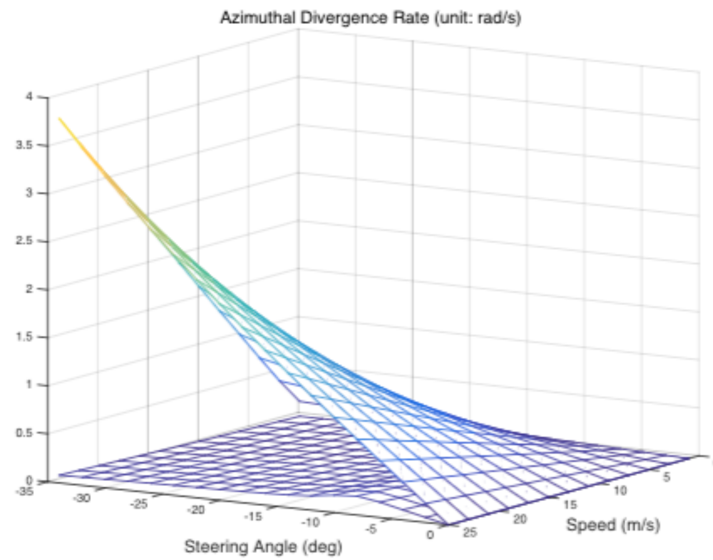
$$
\begin{aligned}
&\left\|\xi_{k+1}-\xi_{k,k+1}^{q*}\right\| \leqslant \left\|\hat{\Gamma}_q(\xi_k)\right\| \\
&+\left\|I+\frac{\partial f\left(\xi_k,\kappa_q(\xi_k)\right)}{\partial \xi}\Delta T\right\| \left\|f\left(\xi_k,\kappa_q(\xi_k)\right)\right\| \Delta t_q(\xi_k) \\
&\approx \left\|\hat{\Gamma}_q(\xi_k)\right\| + v\Delta t_q(\xi_k)\sqrt{1+\left(\frac{\tan\delta}{L}v\Delta T\right)^2} \qquad (16)
\end{aligned}
$$

- Take car as the specific example, we have the hybrid logic. Implementation of such logic needs estimation of model mismatch and return time of both MPCs.
- $q=\underset{q}{\operatorname{argmin}}\left\|\xi_{k+1}-\xi_{k,k+1}^{q*}\right\|$

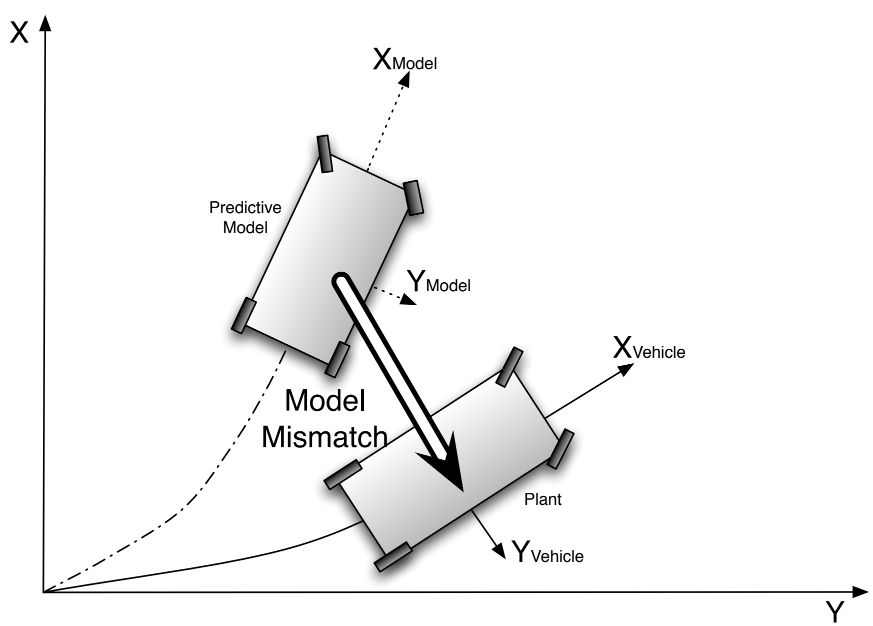
Model Mismatch & Return Time

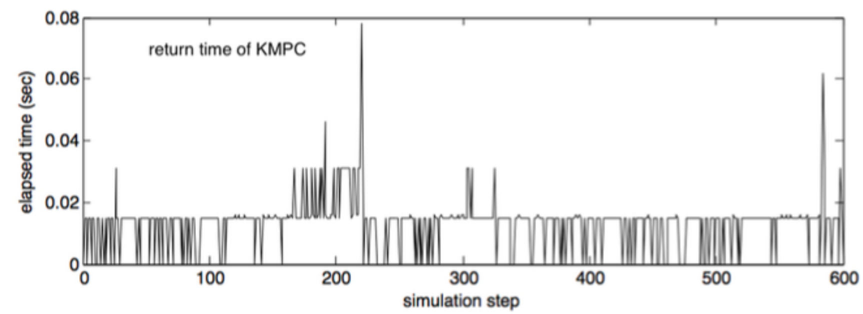


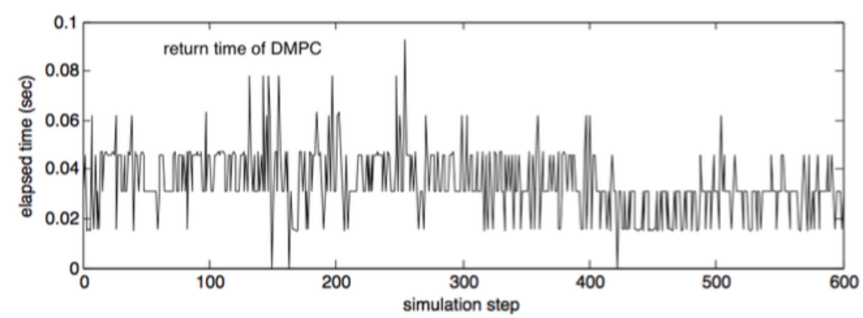
(a)



(b)

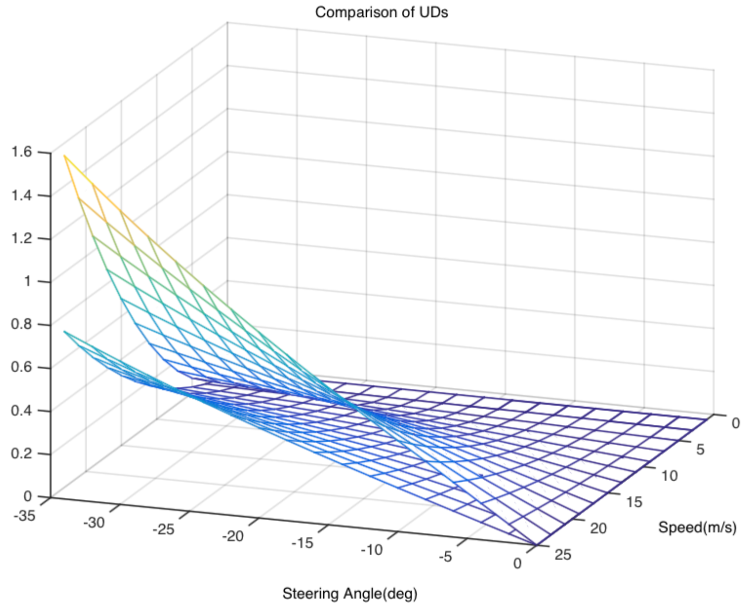


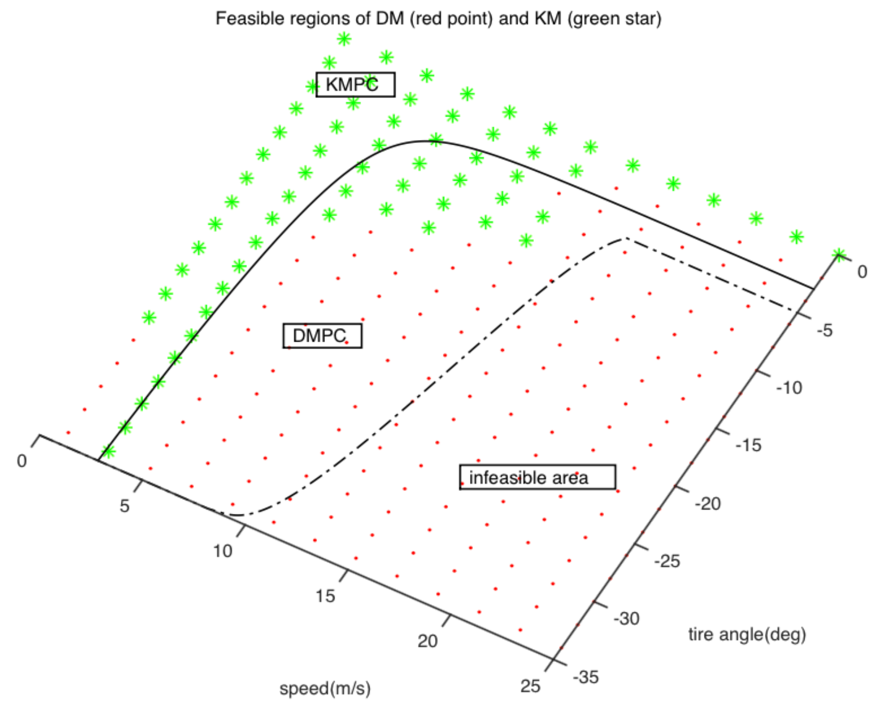




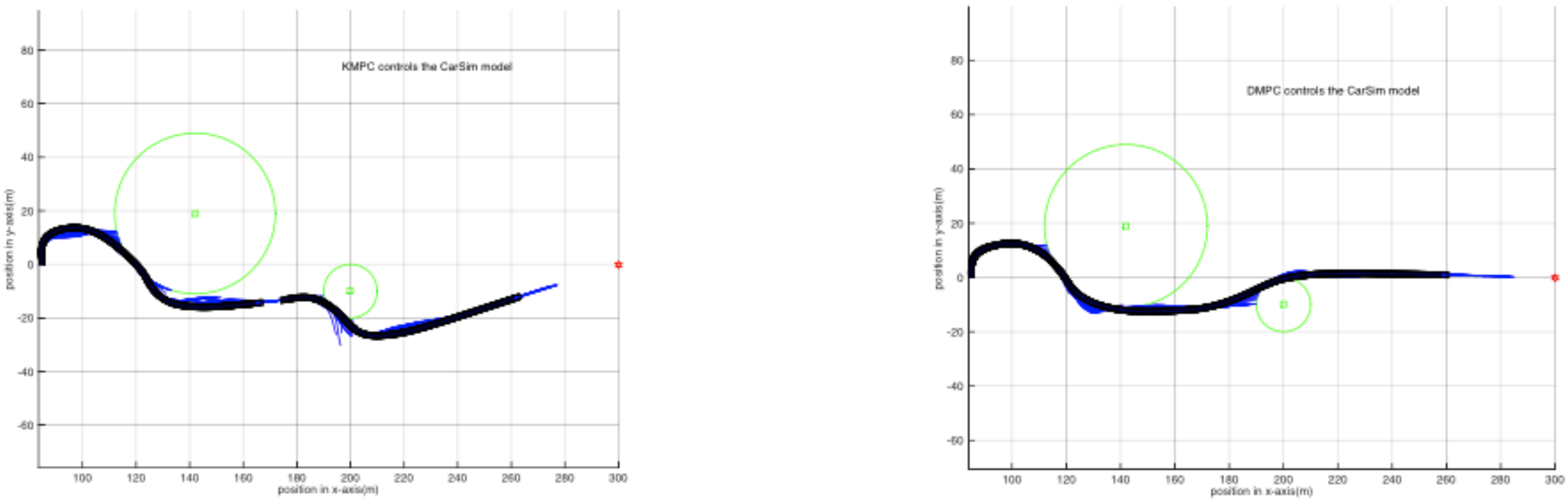
Hybrid MPC Design

- By plotting out UDs of both MPCs, we know the explicit switching boundary.





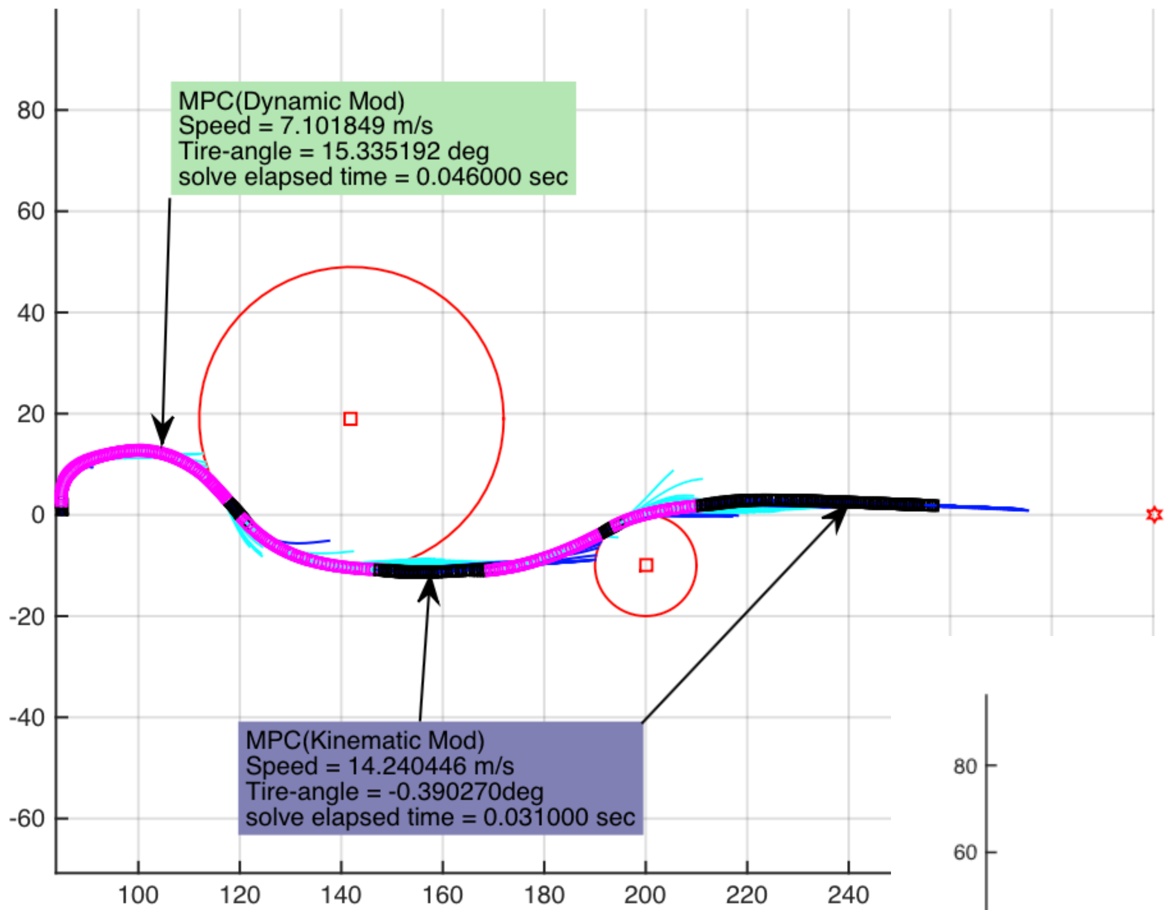
Simulation Result

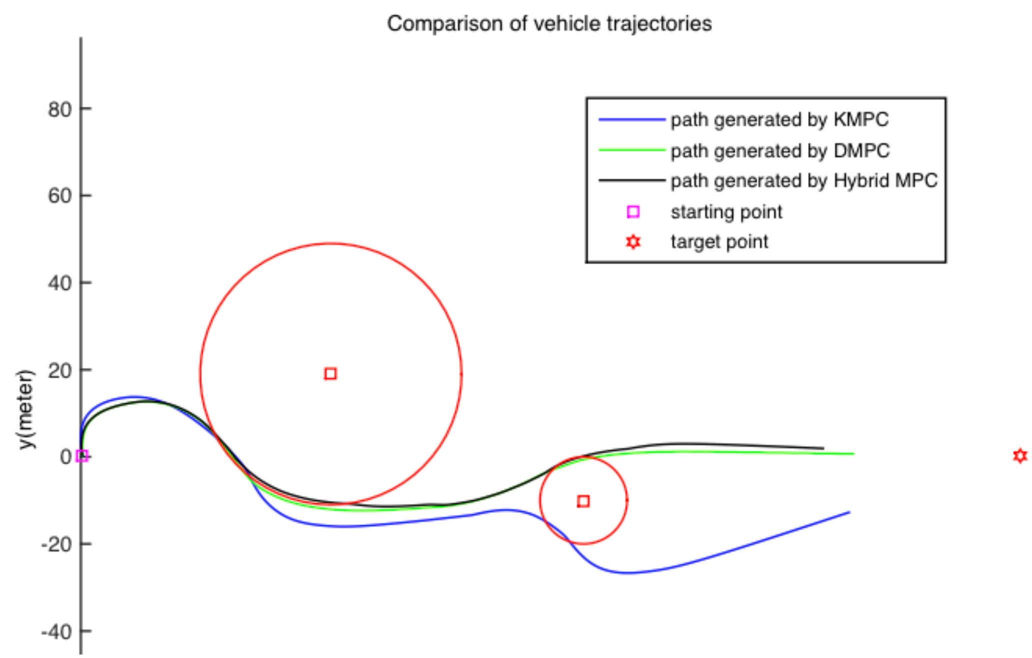


(a) (b)

Figure 8: (a). Kinematic MPC ($q = 0$) only; there is significant divergence of the predicted from the plant model, which results in a path that is unable to navigate between the two obstacles. (b). Dynamic MPC ($q = 1$) only; here the selected trajectory and its tracking are much more aggressive.

Simulation Result





Simulation Result

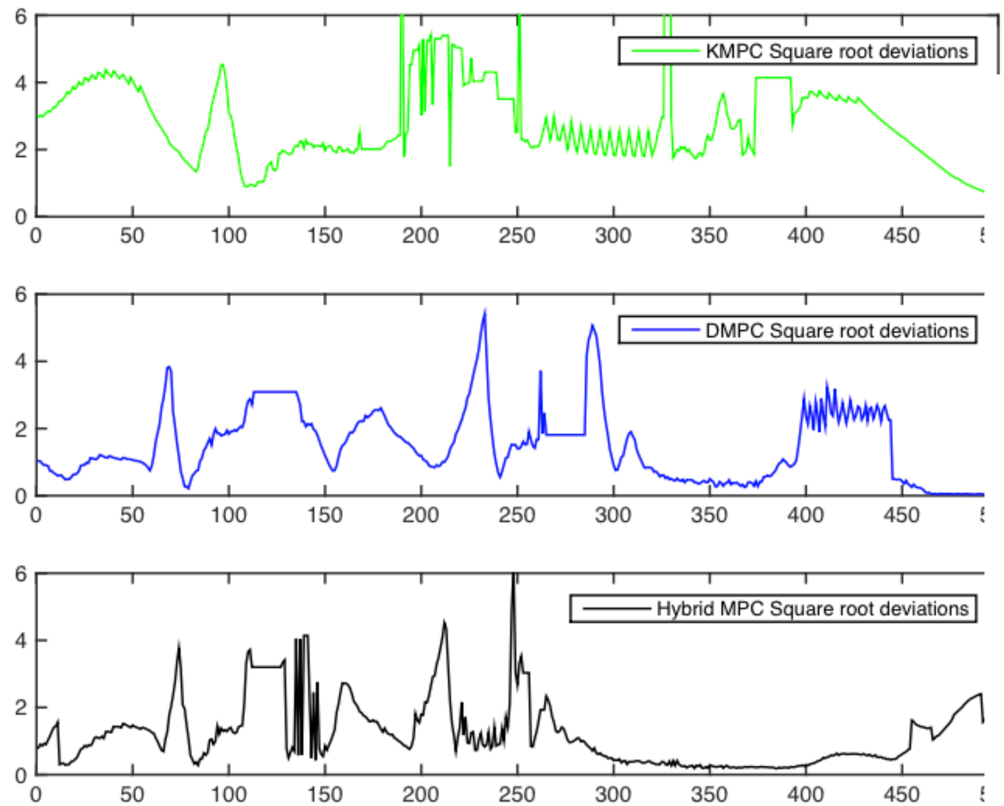
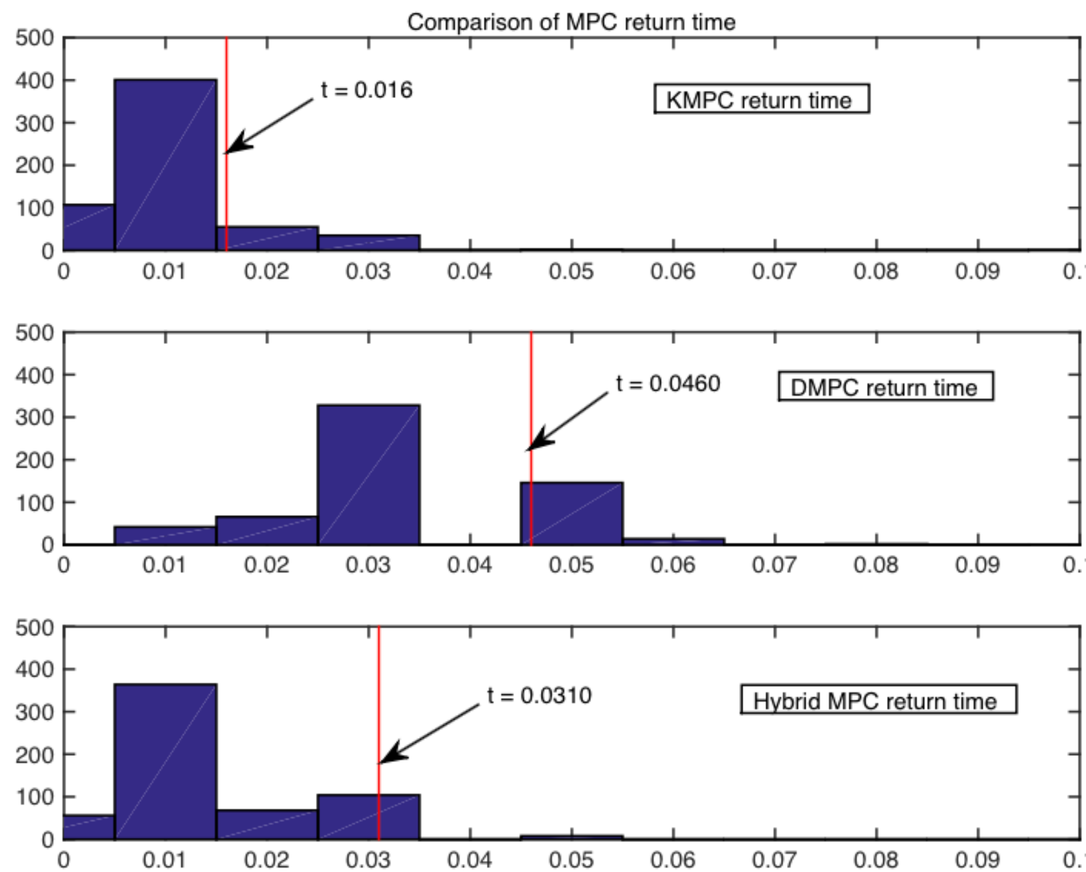


Figure 12: Square root deviations between plant t jectories and predictions. The last several samp are truncated for the reason that predictions alwa run ahead of the current state.



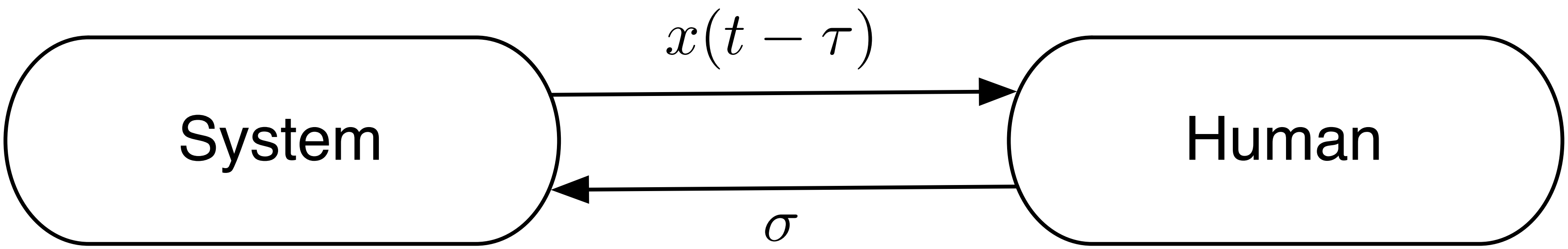
Back to safety…of interface code

Project	Project Type	NOA	LOC	Bytes
	Generated	342	113140	3462891
OpenJAUS Messages	Tests	318	16568	824577
	Manual	319	119752	4038053
	Generated	26	5178	148768
Drive-by-iPad	Tests	14	502	13883
	Manual	27	6756	207447

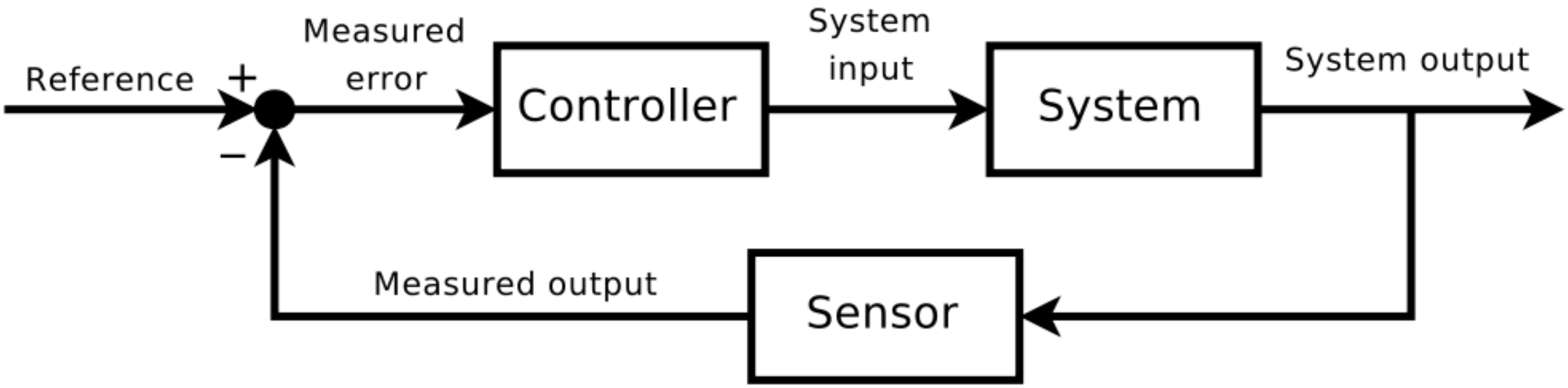
Timescales for Humans-In-The-Loop

THE UNIVERSITY OF ARIZONA

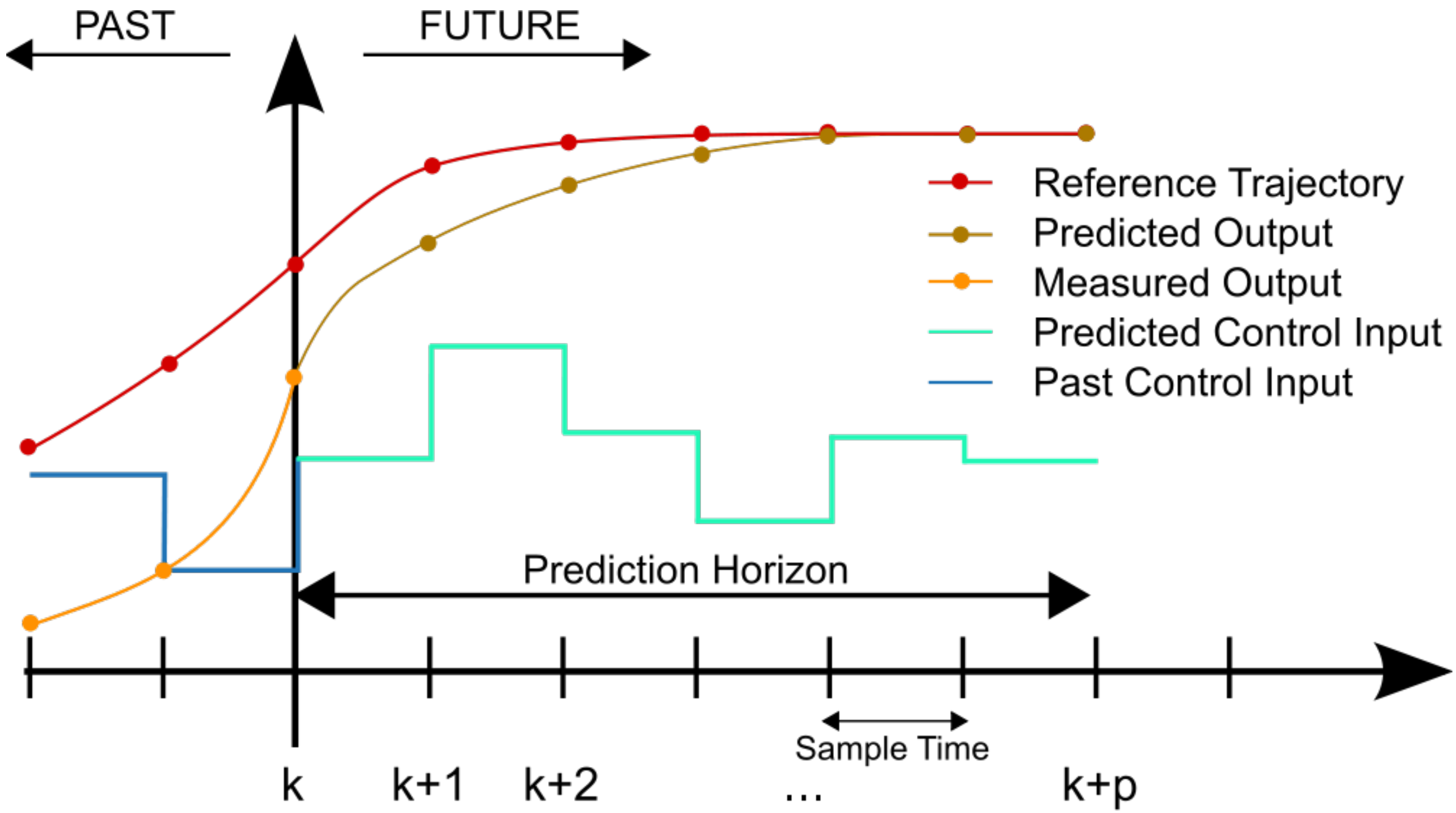
Time delay human control



Conventional control system



Model Predictive Control (MPC)



Market forces vs. behavior change

Lifestyle change

Energy Bills

Gas Prices

Shipping

Giving Birth

Commute

OTC Pain relievers

River flow

Traffic flow

Car Navigation

Stability controller

malloc

Router

Timescale of feedback/coupling

Agriculture/ watering

Surgery

Electricity prices

Water Temperature

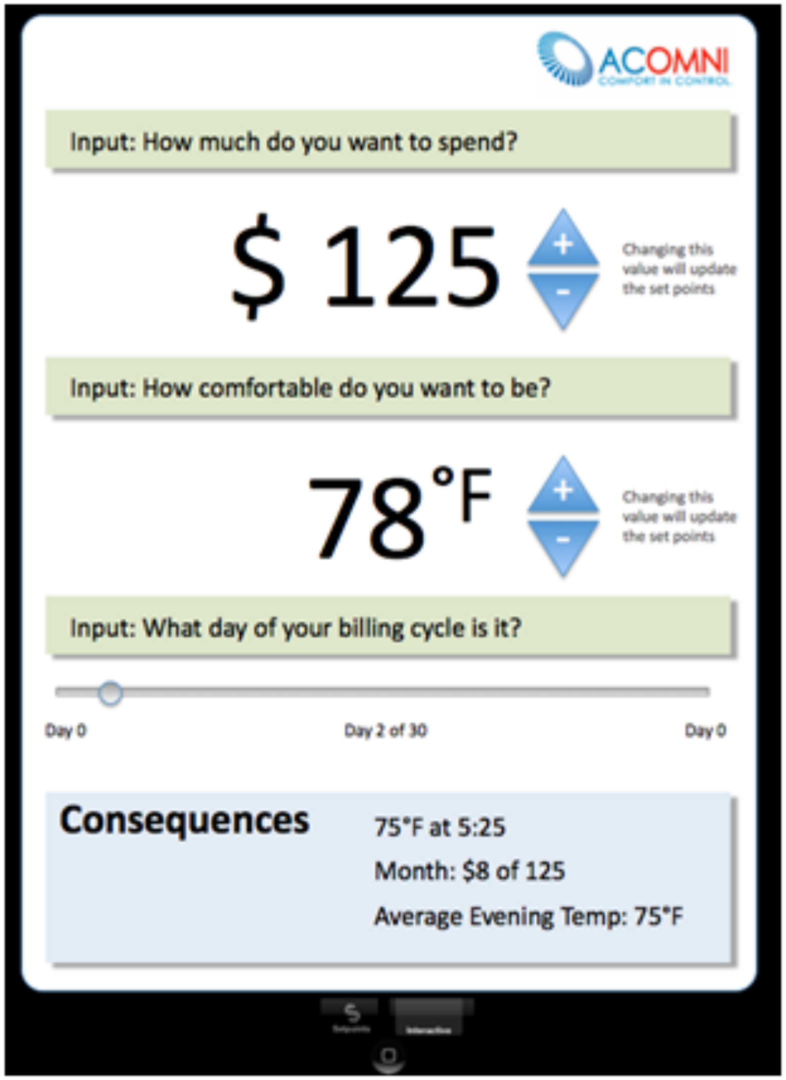
HVAC set point Mgmt

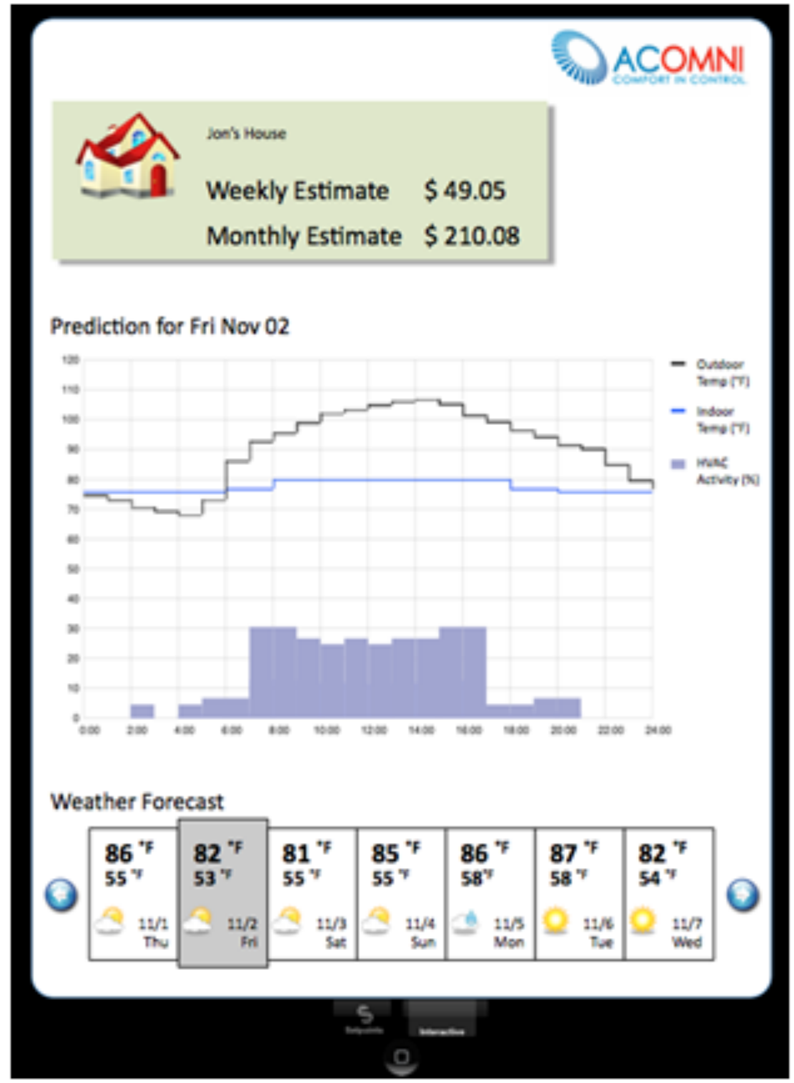
Java GC

Chip Thermal Mgmt

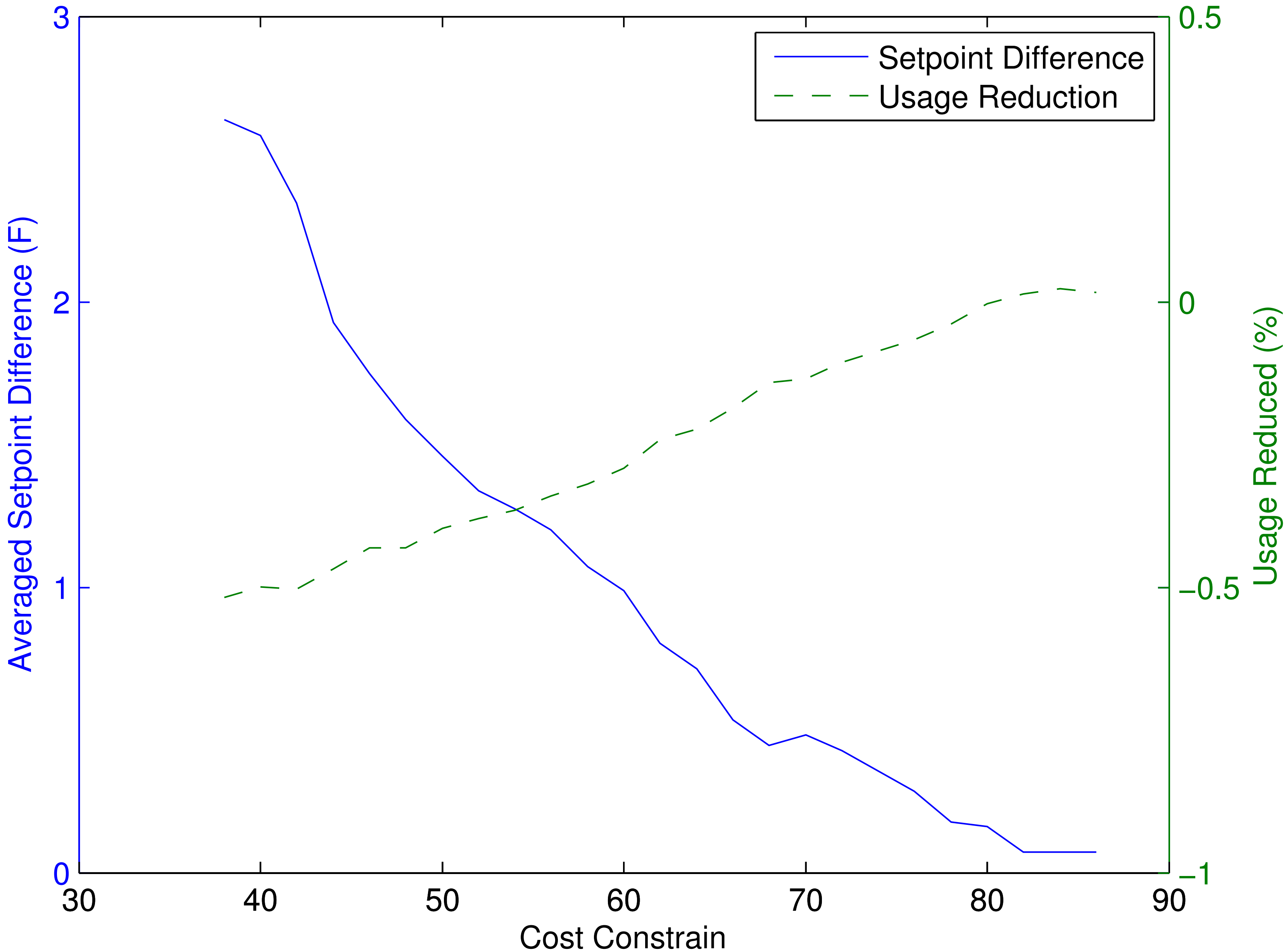
Computer Clock

Idea: Correlate Cost and Comfort in HITL Timescale

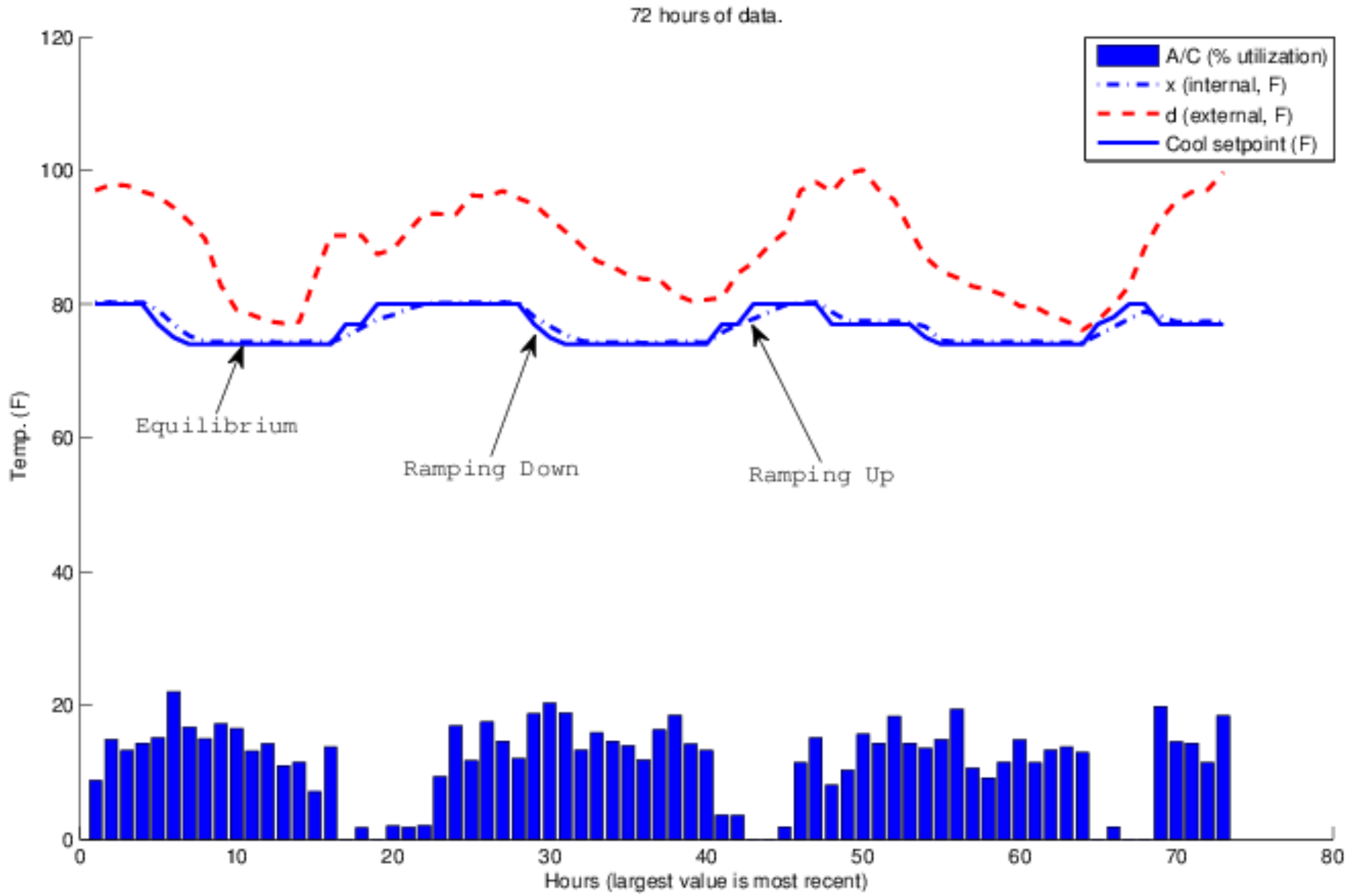




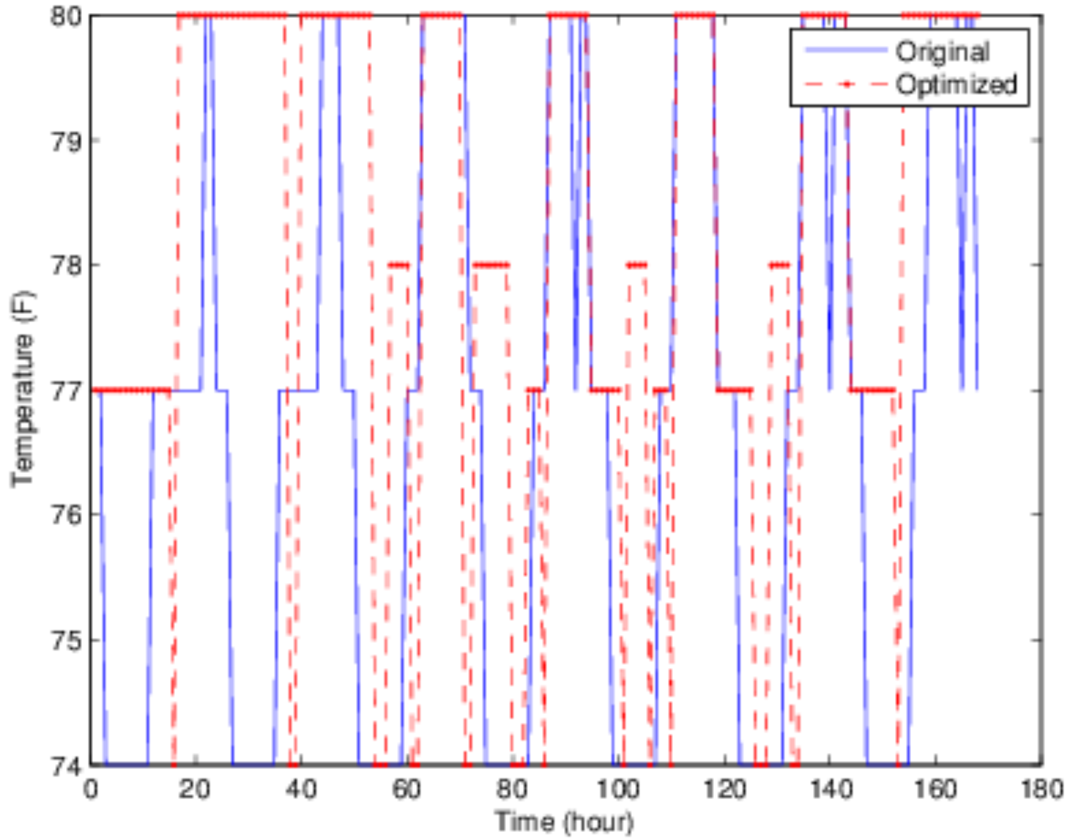
Close the cost/comfort loop by dynamically changing set points



It is possible, but all starts with the need for data



Example results



Cooling in July, 2012

7 day horizon
Cost constraint: $50 ($78 before)
Total change constraint: 28 (4 times per day)
Average temperature sacrifice: 1.3 ˚F

Surprise, I am exceeding my time horizon

The University of Arizona

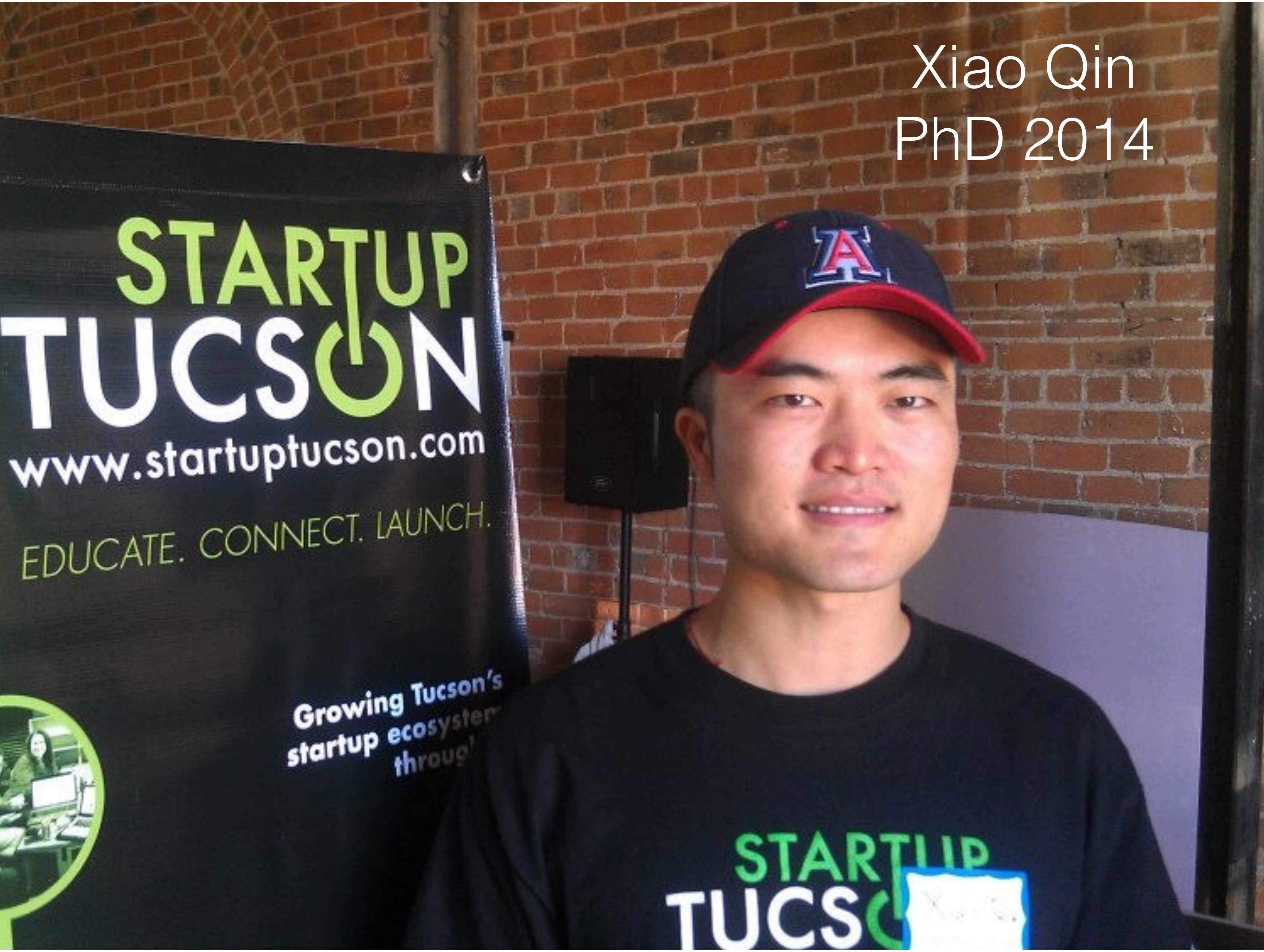
Conclusions

- The opportunity for testbeds is in their ability to grow the user base from adjacent disciplines or new users
- The data **must be realistic;** else runs the risk of perpetuating invalid assumptions
- The timescale of interaction may require supervisory controllers or data gatherers to capture/enforce dynamics for safety and security


Sean Whitsitt
PhD 2014


Kun Zhang
PhD 2015


Matt Bunting
PhD 2016



Xiao Qin
PhD 2014

CAT Vehicle 2013

CAT Vehicle 2014

Thanks for the Support

"Self-Reconfigurable Sensors in River Environments" NSF CNS-0930919, with Sonia Martinez (UC San Diego) and Alex Bayen (UC Berkeley); I-Corps: "A Cost-Limited Home Thermostat (CLD/HT)" NSF IIP-1249175.

"CAREER: Domain-Specific Modeling Techniques for Cyber-Physical Systems" NSF CNS-1253334

"REU Site: CatVehicle: Cognitive and Autonomous Test Vehicle" NSF IIS-1262960

Additional support for awards CNS-1253334 and IIS-1262960 provided by the Air Force Office of Scientific Research

"Control of Vehicular Traffic Flow via Low Density Autonomous Vehicles" NSF CNS-1446435

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or AFOSR.