

Testing Concurrent Interacting OO Finite State Machines

Jeff Offutt

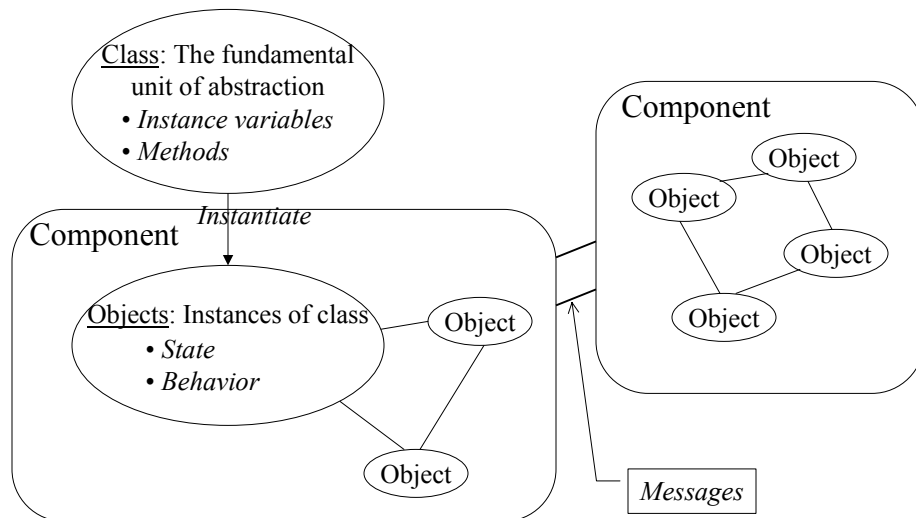
Information & Software Engineering
George Mason University
Fairfax, VA USA
www.ise.gmu.edu/~ofut/
ofut@ise.gmu.edu

Joint research with:

Len Gallagher
Tony Cincotta
Julie Zanon
National Institute of Standards and Technology

Supported by NIST and NSF.

Object-oriented Software



Modeling Classes as FSMs

Single Class Example – Engine

- States : { S_0 , S_f , ON, OFF }
- Variables : { int speed, boolean KeyOn }
- Methods : { Engine(), ~Engine(), setKeyOn (boolean in),
Start (int S), Stop(), setSpeed (int S), int getSpeed() }
- Transitions : (*source* → *target*, *trigger*, *guard*, {*actions*})
 - t1 : ($S_0 \rightarrow$ OFF, Engine(), true, {speed=0, KeyOn=false})
 - t2 : (OFF → ON, Start(), KeyOn==true \wedge $0 \leq S \leq 110$, {speed=S})
 - t3 : (ON → OFF, Stop(), true, {speed=0})
 - t4 : (OFF → OFF, getSpeed(), true, {return speed})



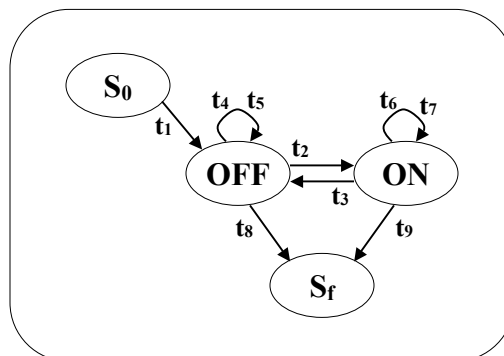
October 2003

© Offutt 2003. All Rights Reserved.

3

Modeling Classes as FSMs

Finite state machine for class Engine



October 2003

© Offutt 2003. All Rights Reserved.

4

Testing OO Software

- 1) **Intra-method testing**: Testing individual methods within classes
- 2) **Inter-method testing**: Pairs of methods within a class are tested in concert
- 3) **Intra-class testing**: Testing a single class, usually using sequences of calls to methods within the class
- 4) **Inter-class testing**: More than one class is tested at the same time (integration)

This research is focused on inter-class testing.

October 2003

© Offutt 2003. All Rights Reserved.

5

Inter-class FSM-based Testing

- Model each class as a finite class state machine (CSM)
- Create combined class state machine from all CSMs
 - Identify mutator methods and their parameters
 - Identify accessor methods
 - Inter-class method calls are messages
 - Add messages as edges to the CSM to create the CCSM
- CCSM is stored in a relational database

October 2003

© Offutt 2003. All Rights Reserved.

6

Component Flow Graph

- A component flow graph (CFG) represents control and data flows among classes and state variables
- Create a CFG
 - Nodes are:
 - States in CCSM that are sources or targets of messages
 - Transitions in the CCSM
 - Guards on transitions
 - Edges represent control flow and data definition-use pairs
 - Intra-class
 - Inter-class
- CFG represents synchronous and asynchronous interactions among classes

October 2003

© Offutt 2003. All Rights Reserved.

7

Generating Test Requirements

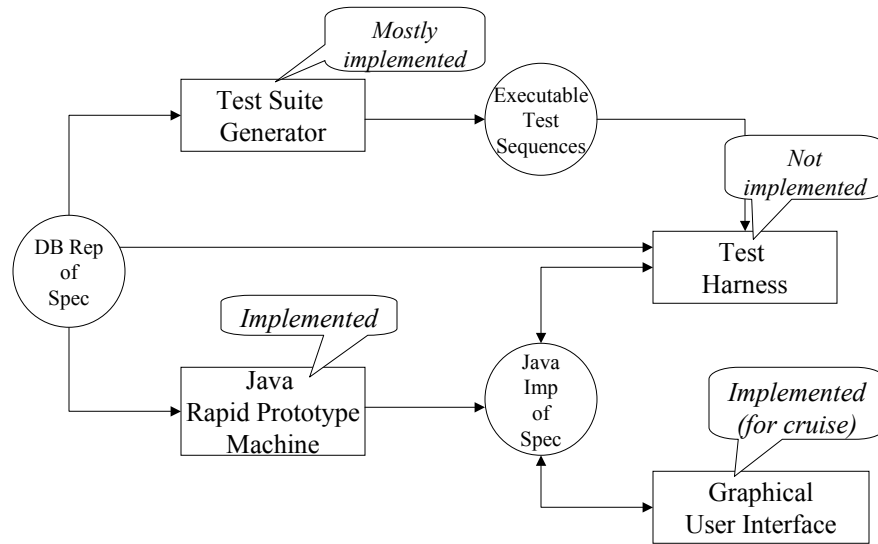
- All-uses on the CFG : Test every definition to every use in the CFG
- Candidate Test Paths : A path from a definition to a use in the CFG
 - must be def-clear : state variable is not changed directly or indirectly (remember asynchronous interactions)
- Candidate test paths are our test requirements
- Executable Test Path is a CTP with parameter values
 - feasible : an input exists that will cause it to be executed
- Executable test cases are user-level inputs that satisfy executable test paths

October 2003

© Offutt 2003. All Rights Reserved.

8

Object-oriented Inter-class Test Process



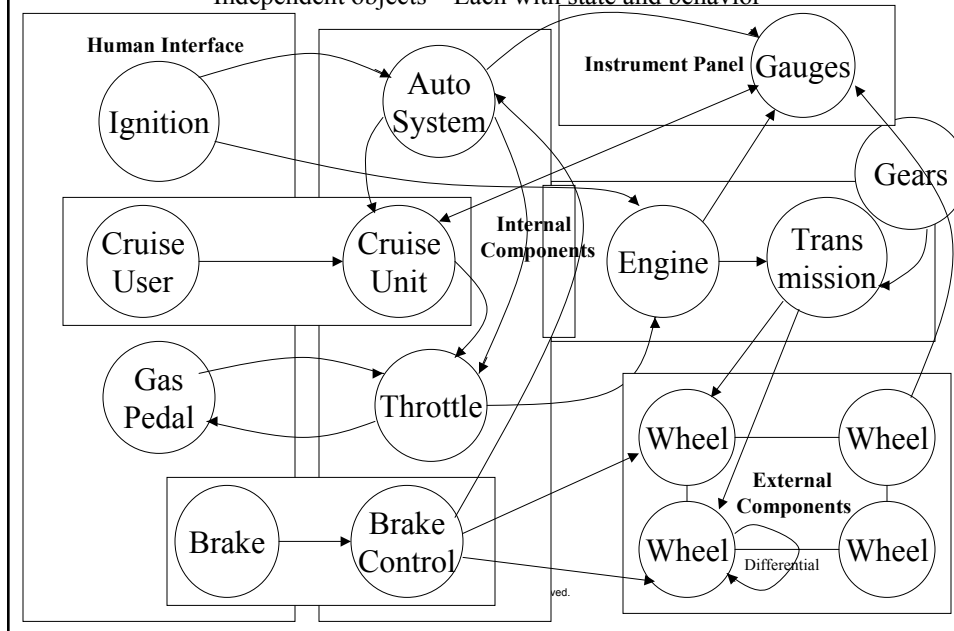
October 2003

© Offutt 2003. All Rights Reserved.

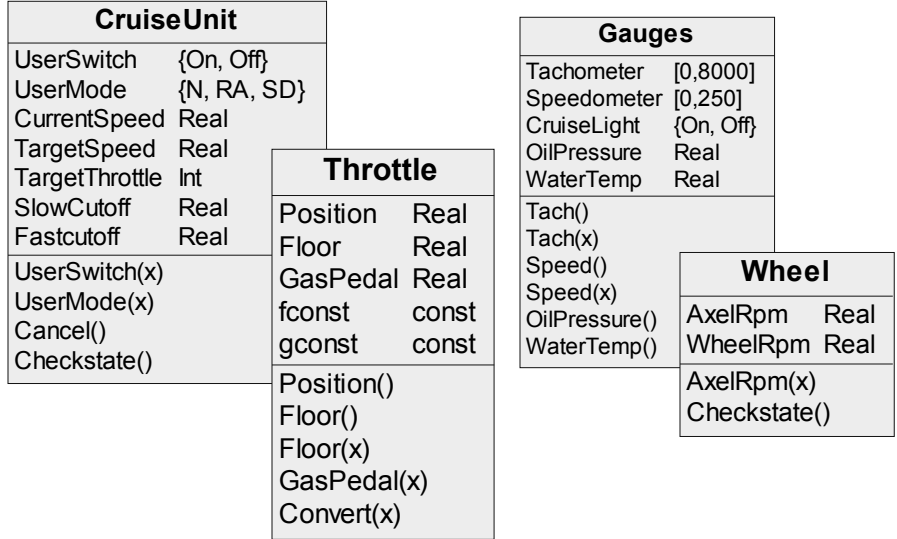
9

Example – OO Functional Specifications

Independent objects – Each with state and behavior



Classes, Objects, State Variables, Methods



October 2003

© Offutt 2003. All Rights Reserved.

11

Define State Predicates

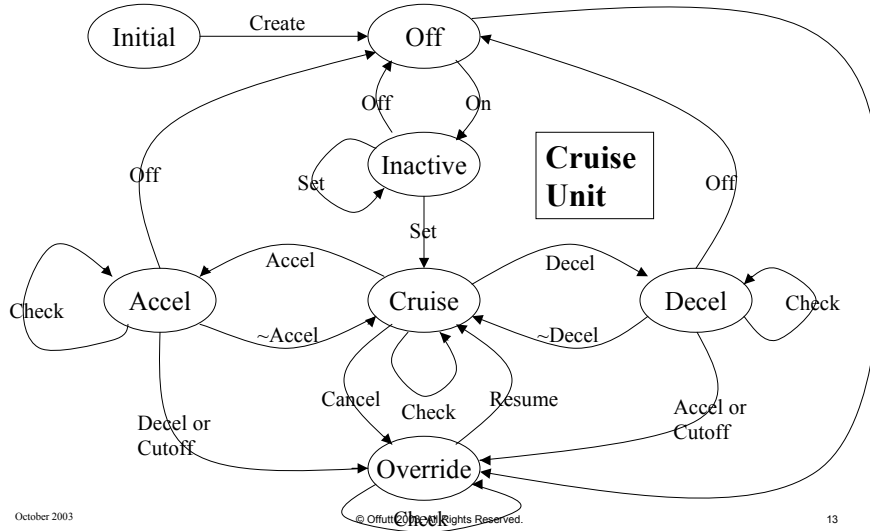
ClassAlias	ClassId	StateId	StateName	DefnPredicate
Throttle	c10	s00	Initial	Undefined
Throttle	c10	s01	Idle	Position=fconst
Throttle	c10	s02	Manual	fconst<Position<=gconst & Position>Floor
Throttle	c10	s03	Automatic	fconst<Position<=gconst & Position=Floor
Throttle	c10	s04	Danger	GasPedal>gconst

ClassAlias	StateId	StateName	DefnPredicate
Gauges	s00	Initial	Undefined
Gauges	s01	Normal	Speed<180 & OilLight=Off & WaterTemp<100
Gauges	s02	Danger	Speed>=180 OR OilLight=On OR WaterTemp>=100

ClassAlias	ClassId	StateId	StateName	DefnPredicate
CruiseUnit	c05	s00	Initial	Undefined
CruiseUnit	c05	s01	Off	UserSwitch=Off
CruiseUnit	c05	s02	Inactive	UserSwitch=On & Gauges.Cruise()=Off & TargetSpeed=0 & UserMode=NULL
CruiseUnit	c05	s03	Cruise	UserSwitch=On & UserMode=NT & Gauges.Cruise()=On & SlowCutoff<TargetSpeed<FastCutoff
CruiseUnit	c05	s04	Accel	UserSwitch=On & UserMode=RA & Gauges.Cruise()=On
CruiseUnit	c05	s05	Decel	UserSwitch=On & UserMode=SD & Gauges.Cruise()=On
CruiseUnit	c05	s06	Override	UserSwitch=On & Gauges.Cruise()=Off & SlowCutoff<TargetSpeed<FastCutoff

Finite Class State Machine Model of Each Object

(sourceState, targetState, method, guard, action)



October 2003

© Offutt 2003. All Rights Reserved.

13

Determine Guards and Actions for Each Transition

ClassID	TransID	SourceState	TargetState	FunName	Guard	Action
e05	t016	Inactive	Cruise	UserMode(x)	x=NT & UserMode=SD & (SlowCutoff < Gauges.Speed() < FastCutoff) & AutoSystem.BrakeActive()=false & AutoSystem.ClutchActive()=false	UserMode:=NT; CurrentSpeed:=Gauges.Speed(); TargetSpeed:=CurrentSpeed; TargetThrottle:=Throttle.Position(); Call Gauges.Cruise(On); Call Throttle.Floor(TargetThrottle); Put CheckState() on Call Queue;
e05	t025	Accel	Override		ABS(TargetSpeed-CurrentSpeed)>=10 &	



GUARD

X = NT & UserMode = RA &
 (SlowCutoff < Gauges.Speed() < FastCutoff)
 & AutoSystem.BrakeActive() = false &
 AutoSystem.ClutchActive() = false

ACTION

Call Throttle.Floor(TargetThrottle);
 Call Gauges.Cruise(On); UserMode:=NT;
 Pause; CurrentSpeed:=Gauges.Speed();
 Put CheckState() on Call Queue;

e05	t063	Override	Override		speed()>=FastCutoff	UserMode:=x;
e05	t064	Off	Off	UserMode(s)	x=NT & UserMode=SD & (SlowCutoff < Gauges.Speed() < FastCutoff) & AutoSystem.BrakeActive()=false & AutoSystem.ClutchActive()=false	CurrentSpeed:=Gauges.Speed(); TargetSpeed:=CurrentSpeed; TargetThrottle:=Throttle.Position(); Call Gauges.Cruise(On); Call Throttle.Floor(TargetThrottle); UserMode:=NT; Put CheckState() on Call Queue;
e05	t065	Override	Cruise	UserMode(s)	x=NT & UserMode=RA & (SlowCutoff < Gauges.Speed() < FastCutoff) & AutoSystem.BrakeActive()=false & AutoSystem.ClutchActive()=false	Call Throttle.Floor(TargetThrottle); Call Gauges.Cruise(On); UserMode:=NT; Pause; CurrentSpeed:=Gauges.Speed(); Put CheckState() on Call Queue;

October 2003

© Offutt 2003. All Rights Reserved.

14

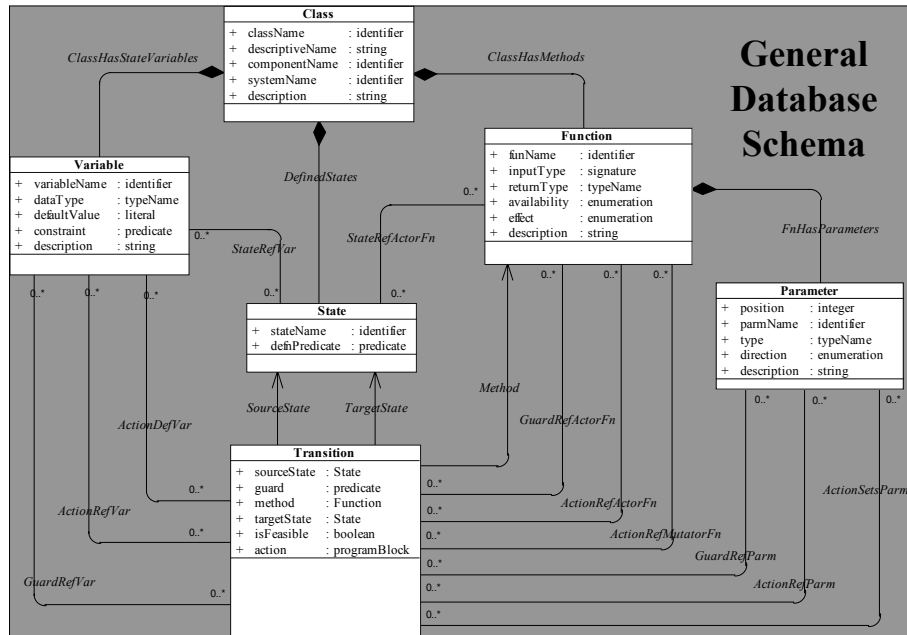
Database Representation

ClassId	ClassAlias	ClassName	ComponentName	SystemName
001	AutoSystem	AutoSystem_ADT	SystemControl	Automobile
002	BrakeControl	BrakeControl_ADT	Brakes	Automobile
003	BrakeUser	BrakeUser_ADT	Brakes	Automobile
004	ClutchUser	ClutchUser_ADT	Clutch	Automobile
005	CruiseUnit	CruiseUnit_ADT	CruiseControl	Automobile
006	CruiseUser	CruiseUser_ADT	CruiseControl	Automobile
007	Engine	Engine_ADT	Engine	Automobile
008	GasUser	GasPedalUser_ADT	Acceleration	Automobile
009	Gauges	Gauges_ADT	InstrumentPanel	Automobile

ClassId	ClassAlias	StateId	StateName	DefnPredicate	FunName	InputType	ReturnType	
001	AutoSystem	000	Initial	Undefined	init()	Boolean	AutoSystem	
001	AutoSystem	001	Inactive	BrakeActive=false & ClutchActive=false & Danger=false	init()	Boolean	Boolean	
001	AutoSystem	002	Active	BrakeActive=true OR ClutchActive=true OR Danger=true	init()	Boolean	Boolean	
002	BrakeControl	000	Initial	Undefined	init()	Boolean	Boolean	
002	BrakeControl	001	Inactive	IsActive=false	off()	Boolean	Boolean	
002	BrakeControl	002	Braking	IsActive=true & WheelsTurning=true	overmor()	Boolean	Boolean	
002	BrakeControl	003	Locked	IsActive=true & WheelsTurning=false	itroll()	Boolean	BrakeControl	
003	BrakeUser	000	ClassAlias	SourceState	TargetState	Features	Guard	Action
003	BrakeUser	001	BrakeUser	Off	Off	ClutchActive	true	Global CruiseUser=New CruiseUser(); UserSwitch=Off; SlowCutoff=25; FastCuts
003	BrakeUser	002	BrakeUser	Off	Off	UserSwitch	true	Return UserSwitch;
004	ClutchUser	000	ClutchUser	Off	Off	UserMode	true	Return UserMode;
004	ClutchUser	001	ClutchUser	Off	Off	SetSpeed	true	CurrentSpeed+=Gauges.Speed;
004	ClutchUser	002	ClutchUser	Inactive	Inactive	UserSwitch	true	Return UserSwitch;
004	ClutchUser	003	ClutchUser	Inactive	Off	UserSwitch	off	UserSwitch=Off;
005	CruiseUnit	000	CruiseUnit	Inactive	Inactive	UserMode	true	Return UserMode;
005	CruiseUnit	001	CruiseUnit	Inactive	Inactive	UserMode	isNT	UserMode=nt;
005	CruiseUnit	002	CruiseUnit	Inactive	Inactive	UserMode	isNT & UserMode=SD & (Gauges.Speed)	UserMode=NT;
005	CruiseUnit	003	CruiseUnit	Inactive	Inactive	SetSpeed	true	CurrentSpeed+=Gauges.Speed;
005	CruiseUnit	004	CruiseUnit	Cruise	Cruise	UserSwitch	true	Return UserSwitch;
005	CruiseUnit	005	CruiseUnit	Cruise	Off	UserSwitch	isOff	Call Gauges.CruiseOff; UserSwitch=Off; UserMode=null;
005	CruiseUnit	006	CruiseUnit	Cruise	Cruise	UserMode	true	Return UserMode;
005	CruiseUnit	007	CruiseUnit	Cruise	Decel	UserSwitch	isSD	TargetSpeed=TargetSpeed-1; UserMode=SD; Put CheckState() on Call Queue;
005	CruiseUnit	008	CruiseUnit	Cruise	Accel	UserMode	isRA	TargetSpeed=TargetSpeed+1; UserMode=RA; Put CheckState() on Call Queue;
006	CruiseUser	000	CruiseUser	Cruise	Override	Cancel	true	Call Gauges.CruiseOff; Call Throttle.Floor();
006	CruiseUser	001	CruiseUser	Cruise	Cruise	SetSpeed	true	CurrentSpeed+=Gauges.Speed;
006	CruiseUser	002	CruiseUser	Cruise	Cruise	CheckState	true	CurrentSpeed-TargetSpeed>1.0 & ThrottCall Throttle.Floor(Throttle.Floor()-1); Pause; CurrentSpeed+=Gauges.Speed; Put
006	CruiseUser	003	CruiseUser	Cruise	Cruise	CheckState	true	ABS(TargetSpeed-CurrentSpeed)>1.0 & CurrentSpeed+=Gauges.Speed; Put CheckState() on Call Queue;
006	CruiseUser	004	CruiseUser	Cruise	Cruise	CheckState	true	TargetSpeed-CurrentSpeed>1.0 & ThrottCall Throttle.Floor(Throttle.Floor()-1); Pause; CurrentSpeed+=Gauges.Speed; Put
006	CruiseUser	005	CruiseUser	Cruise	Cruise	CheckState	true	0.5=ABS(TargetSpeed-CurrentSpeed)-1; CurrentSpeed+=Gauges.Speed; Put CheckState() on Call Queue;
006	CruiseUser	006	CruiseUser	Cruise	Cruise	CheckState	true	ABS(TargetSpeed-CurrentSpeed)>0.5; Pause; CurrentSpeed+=Gauges.Speed; Put CheckState() on Call Queue;
006	CruiseUser	007	CruiseUser	Accel	Accel	UserSwitch	true	Call Gauges.CruiseOff; UserSwitch=Off; UserMode=null;
006	CruiseUser	008	CruiseUser	Accel	Accel	UserMode	true	Return UserMode;
006	CruiseUser	009	CruiseUser	Accel	Override	UserMode	isSD	Call Gauges.CruiseOff; UserMode=null; Call Throttle.Floor();
006	CruiseUser	010	CruiseUser	Accel	Cruise	UserMode	isNT	UserMode=NT; TargetSpeed+=Gauges.Speed; TargetThrottle=Throttle.Position;
006	CruiseUser	011	CruiseUser	Accel	Override	Cancel	true	Call Gauges.CruiseOff; UserMode=null; Call Throttle.Floor();
006	CruiseUser	012	CruiseUser	Accel	Accel	SetSpeed	true	CurrentSpeed+=Gauges.Speed;
006	CruiseUser	013	CruiseUser	Accel	Accel	CheckState	true	Call Throttle.Floor(Throttle.Position)+1; Pause; CurrentSpeed+=Gauges.Speed;
006	CruiseUser	014	CruiseUser	Accel	Override	Cancel	true	Call Gauges.CruiseOff; UserMode=null; Call Throttle.Floor();
006	CruiseUser	015	CruiseUser	Decel	Decel	UserSwitch	true	Return UserSwitch;
006	CruiseUser	016	CruiseUser	Decel	Off	UserSwitch	isOff	Call Gauges.CruiseOff; UserSwitch=Off; UserMode=null;
006	CruiseUser	017	CruiseUser	Decel	Decel	UserMode	true	Return UserMode;
006	CruiseUser	018	CruiseUser	Decel	Cruise	UserMode	isNT	UserMode=NT; TargetSpeed+=Gauges.Speed; TargetThrottle=Throttle.Position;

October 2003

General Database Schema



October 2003

© Offutt 2003. All Rights Reserved.

16

Size of Engine System

- 10 classes
 - 46 class variables
 - 97 methods
 - 76 states
 - 143 transitions
- Component Flow Graph
- 208 Nodes
 - 551 Edges
 - 3433 Def-use pairs

CFG is generated in a few seconds

October 2003

© Offutt 2003. All Rights Reserved.

17

Finding Candidate Test Paths

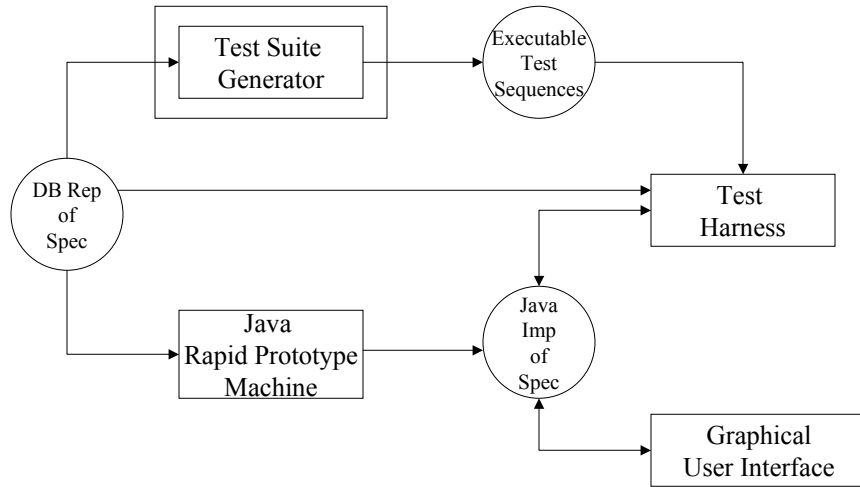
Step	New Paths	New DU-pairs	Unsolved DU-Pairs	Not Def-clear	Partial Paths
1	18	18	3433	99	
2	0	0	3316	0	3316
3	363	363	2948	5	4174
4	69	69	2879	0	15,077
5	291	287	2564	28	49,664
			⋮		
11	231	214	445	6	46,509
12	26	17	428	0	50,822

October 2003

© Offutt 2003. All Rights Reserved.

18

Object-oriented Inter-class Test Process

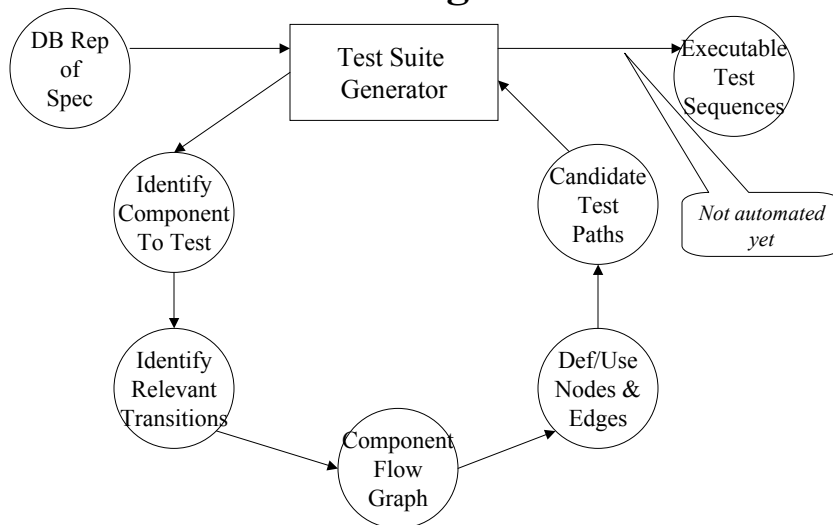


October 2003

© Offutt 2003. All Rights Reserved.

19

Test Suite Generation – Integration Testing

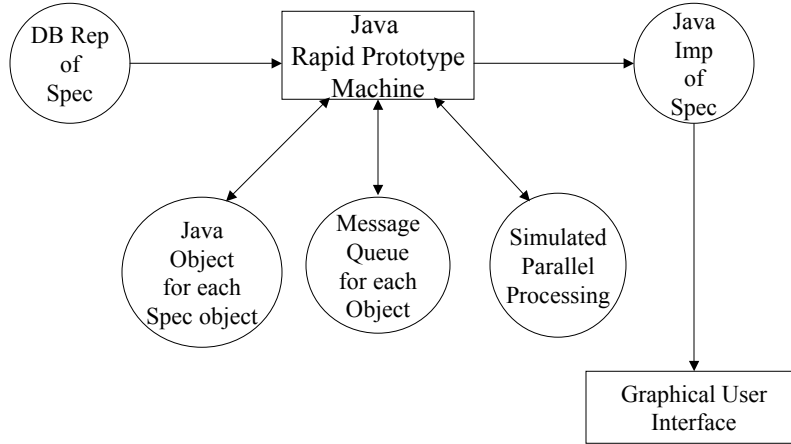


October 2003

© Offutt 2003. All Rights Reserved.

20

Rapid Prototype Machine

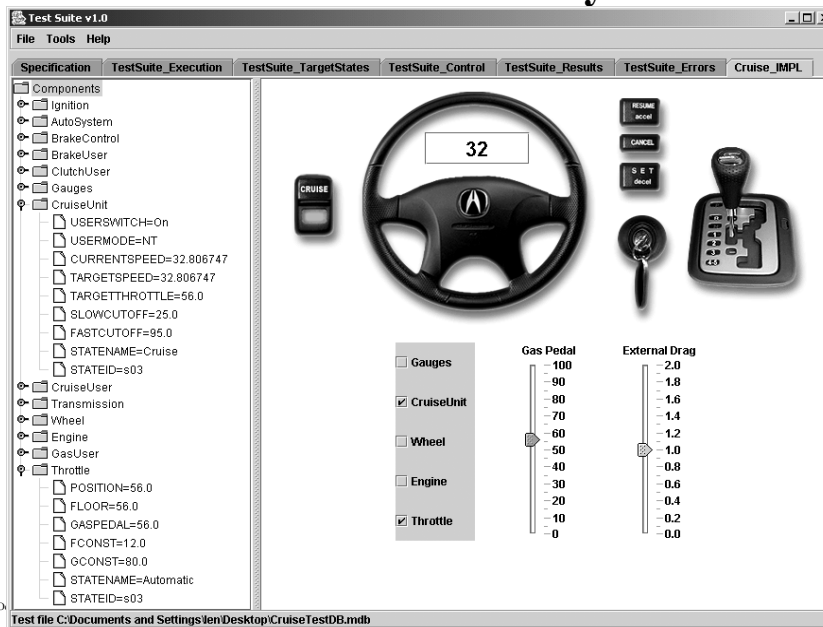


October 2003

© Offutt 2003. All Rights Reserved.

21

Screenshot of Auto System



Summary of Results

- Inter-class testing technique
- Test process for systems comprised of
 - Components that run as separate processes
 - Communication via message passing
 - Object-oriented software
- Tests based on class descriptions
 - Methods
 - State variables
 - Defs and uses of state variables by methods
- Database representation of object state behavior

October 2003

© Offutt 2003. All Rights Reserved.

23

Current and Future Work

- Complete the automation of the executable tests
- Mapping problem
 - What methods must be called to trigger each transition?
 - What user-level actions must be carried out
- Web application interface
- Conveniently capture class information into database
 - Accept and translate UML diagrams
 - Form-based graphical user interface
- Full experimentation with fault study

October 2003

© Offutt 2003. All Rights Reserved.

24