

SEQUENCE-BASED SPECIFICATION OF CRITICAL SOFTWARE SYSTEMS

Stacy J. Prowell and W. Thomas Swain
sprowell@cs.utk.edu · swain@cs.utk.edu

Software Quality Research Laboratory
University of Tennessee
Knoxville, TN 37996-3400, USA

<http://www.cs.utk.edu/sqrl/>

Software-Based Control and Protection

Benefits

- Improved measurement accuracy

- Operational flexibility

Market reality

- Fewer analog instruments available

Downside

- Regulatory scrutiny

- Operational reliability concerns

How Can We Ensure Correct Operation of Critical Software?

Testing?

- A sampling process

- As complexity increases, a sufficient sample becomes impractical

Peer Review of Documentation/Code?

- Often subjective

- Labor intensive

- No objective criteria for sufficiency

The Only Practical Alternative: Rigorous Specification and Design

Basic Premise

A software program is a rule for computing a mathematical function that maps all possible stimulus histories to all possible responses

Approach

Derive mathematically rigorous specification and design from requirements

Prove critical design properties using basic mathematics of software

Formal Methods and Tools

Z

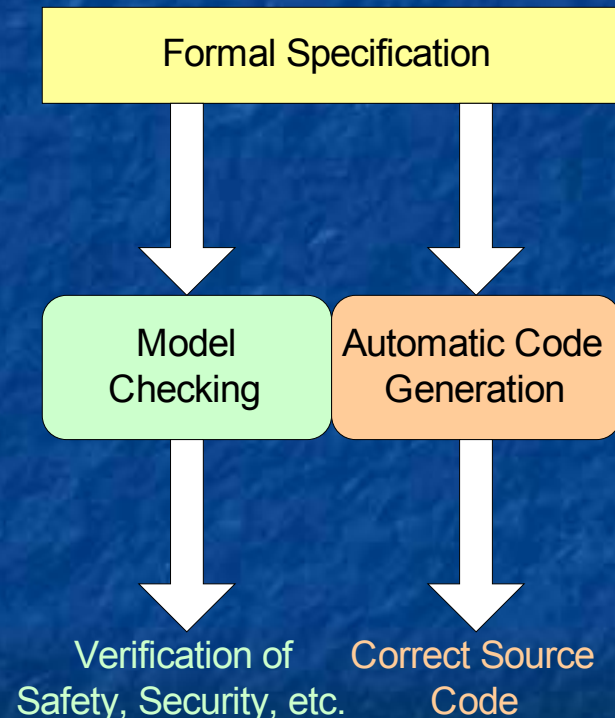
Software Cost Reduction
(SCR)

Trace Assertion Method
(TAM)

Calculus of Concurrent
Systems (CCS)

Communicating Sequential
Processes (CSP)

Applicative Common Lisp
(ACL2)



Problem: How Do We Produce the Formal Specification?



Specification Objectives

Completeness

a response is defined for every stimulus history

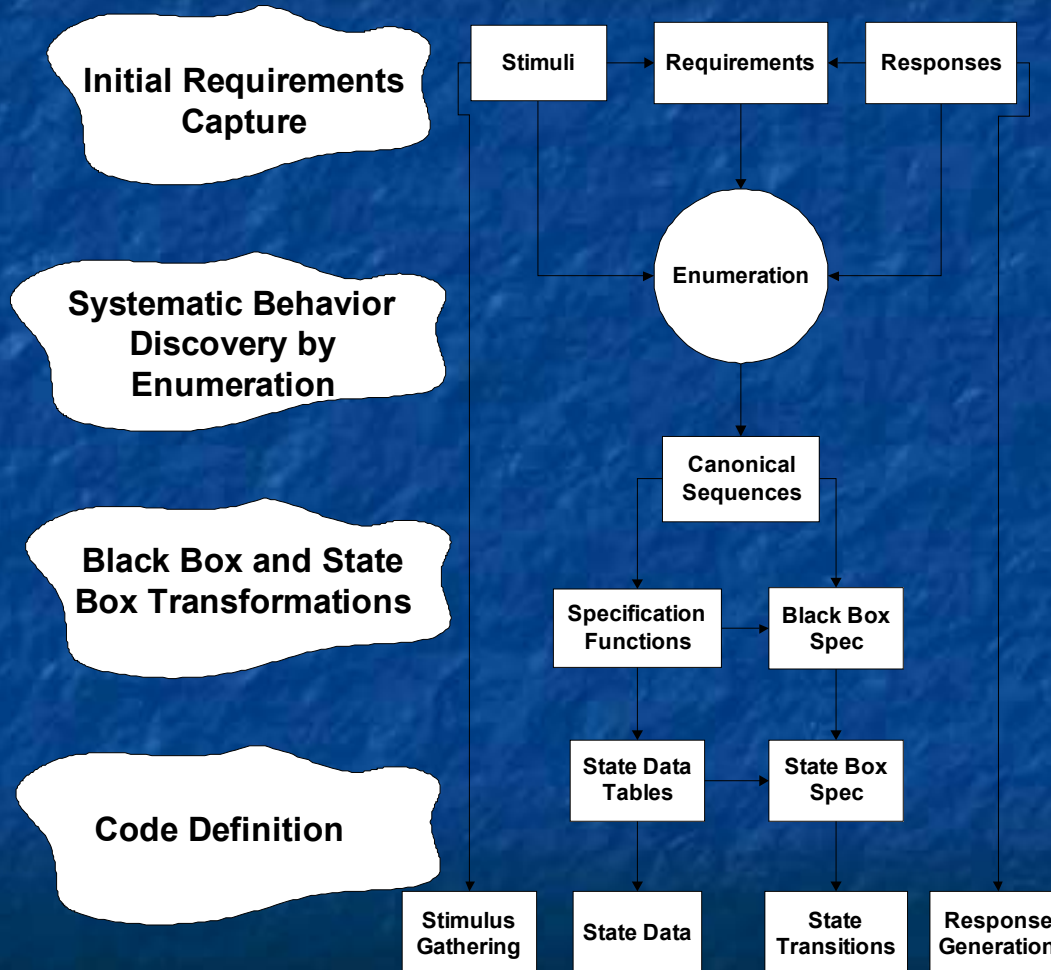
Consistency

each stimulus history maps to only one response

Correctness

the specification is explicitly traceable to the requirements

Specification Process Overview



Rigorous Specification Process

Establish system boundary.

Define human/software/hardware interfaces.

Itemize stimuli.

Itemize responses.

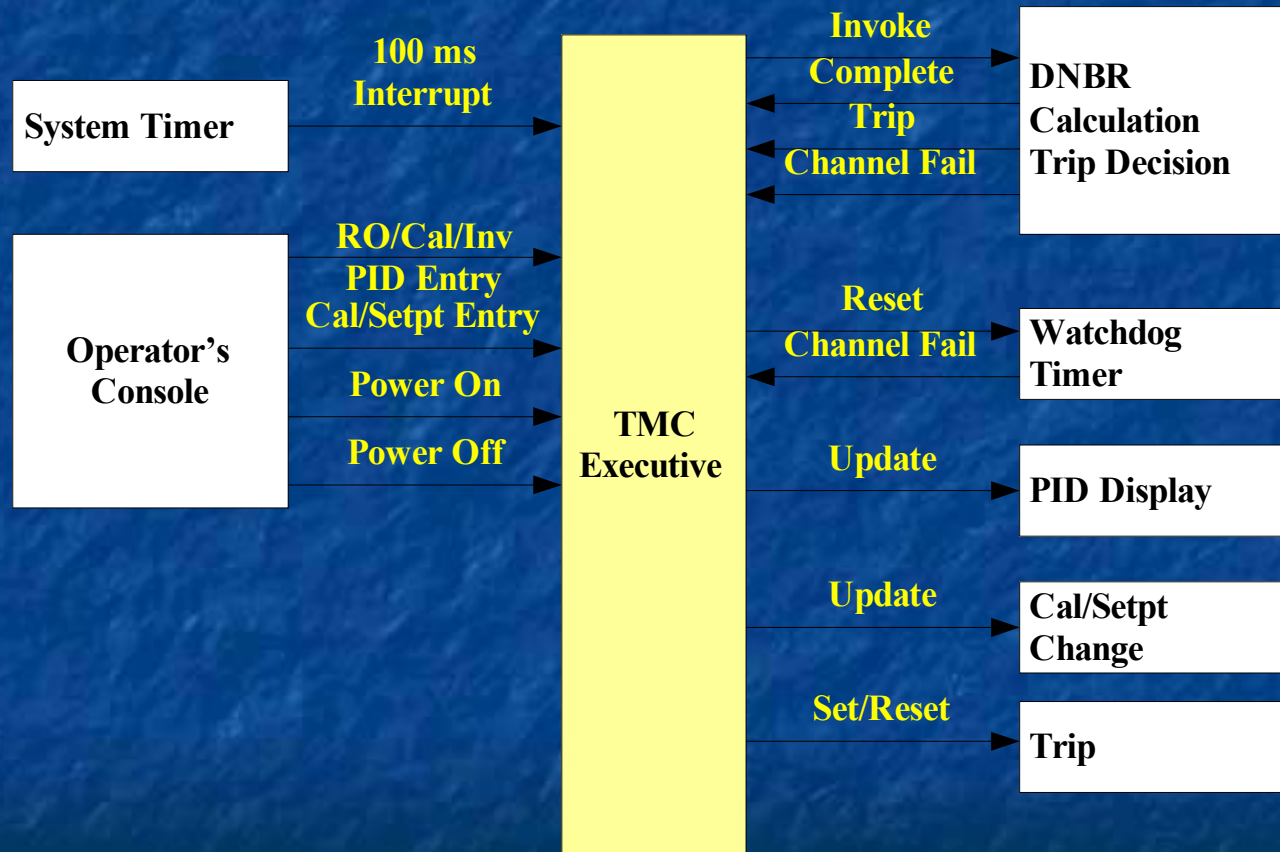
Perform enumeration.

Perform canonical sequence analysis.

Generate state machine specification.

Transform to formal notation (e.g., ACL2).

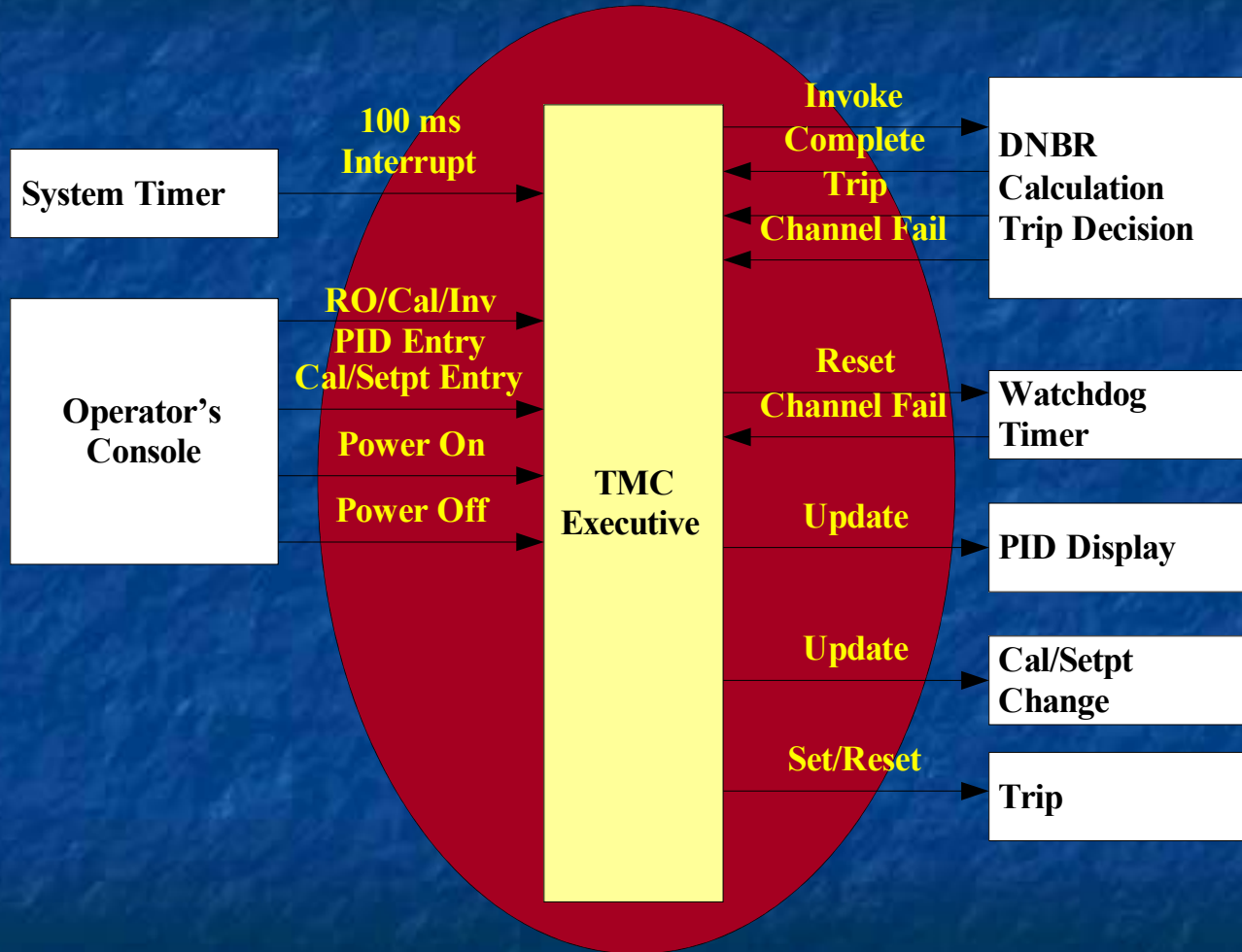
Example: Hypothetical PWR Thermal Margin Calculator (TMC)



Initial TMC Requirements

Tag	Requirement
1	The TMC shall compute an accurate, but conservative, <i>static</i> estimate of DNBR every 100 ms. Sensor inputs for the DNBR calculation are pressurizer pressure (p), neutron flux at selected locations (ϕ), primary coolant mass flow rate (F), cold leg temperature (T_c), and hot leg temperature (T_h).
2	The DNBR, D_d , is compared to a trip setpoint, D_t , each time it is computed. If D_d is less than the setpoint, then the trip contact output is set.
3	All sensor inputs are checked for validity when sampled. If a sensor value is invalid, the channel failure indicator is set.
4	The operator interface has the following features: <ul style="list-style-type: none"> • Numeric displays for the point ID and value of sensor inputs, calculated values, calibration constants, and setpoints. • An alphanumeric display of the current point ID. • A numeric keypad and associated function keys for entry of point ID selections and new values for calibration constants and setpoints. • Indicator lights for trip status and channel failure status. • A keyswitch to turn power on and off.
5	If a non-existent or invalid point ID is entered, the point ID and value displays revert to their previous values.
6	If an invalid calibration or setpoint value is entered, the value display reverts to the original value.
7	A watchdog timer must be reset at least every 100 ms. Otherwise the channel failure indication is set and a reboot command is issued.
8	When a TMC is powered up or rebooted, all sensor inputs are read and used to initialize the DNBR calculation, the watchdog timer is reset, and the trip contact output is set.
9	If a TMC is powered down or rebooted the trip contact output must be set.

System Boundary and Interfaces



Important Definitions

Stimulus - an event resulting in information flow from the outside to the inside of the system boundary

Output – externally observable item of information flow from inside to outside the system boundary

Response – occurrence of one or more outputs caused by a stimulus

TMC Executive Stimuli

Interface	Stimulus	Description	Trace
Clock interrupt	C1	100 ms interrupt	1
Operator console	Pr	Selection of read-only PID	4
	Pc	Selection of PID for calibration constant or setpoint	4
	Pn	Entry of non-existent or invalid PID	5
	Cs	Entry of new calibration constant or setpoint	4
	Ci	Entry of invalid calibration constant or setpoint	6
Power	Pi	Power on (initialize)	8
	Pe	Power off (exit)	9
Task control	Tc	Task complete with no trip	1
	Tt	Task resulted in trip event	2
	Tf	Task reported a sensor or channel failure	3,7
System	R	Restart executive	7,8,9

Abstract Stimulus

Definition - Single stimulus representing multiple events.

Examples - Pr, Pc, Pn, Cs, Ci each represent multiple keystrokes required to display PIDs or modify setpoints or calibration constants.

Abstractions must be defined by disjoint predicates representing all possible stimuli.

TMC Executive Outputs

Interface	Output	Description	Trace
Task control	T1	Invoke 100 ms task	1
Watchdog timer	Wr	Reset watchdog timer	7
Operator console	Ud	Update PID display	4
	Uc	Update calibration constant or setpoint	4
	Fa	Activate failure indicator	2,4,8,9
	Fd	Deactivate failure indicator	2,4,8,9
	Ta	Activate trip indicator	3,4,7
	Td	Deactivate trip indicator	3,4,7
Trip logic	Ts	Set trip output	2,8,9
	Tr	Reset trip output	2,8,9
System	Ri	Initiate reboot	7,8,9

Enumeration Overview

Define response to all possible stimulus sequences by considering them in order of length.

Continue until all sequences of a given length are either illegal or equivalent to previous sequences.

Identify canonical sequences.

Important Definitions

A stimulus sequence is

illegal by definition if it is impossible for the system to observe it or for the environment to generate it

illegal by design if the system design is intended to prevent either generation or observation of it

More Important Definitions

A sequence is

equivalent to another sequence if their responses to the same future stimuli are identical

reduced if it has been declared equivalent to a previous sequence

canonical if it is legal and unreduced when enumeration is complete

Enumeration Mechanics

For each stimulus sequence of length n :

If illegal, mark it illegal and do not extend further.

Document correct response based on requirements.

If no requirement found, create derived requirement.

Record requirements trace.

Check for equivalence with previous sequences.

Extend only those sequences that are not illegal or equivalent.

TMC Executive Enumeration through Length 2

Prefix	New Stimulus	Response	Equivalence	Trace
Length 1				
λ	Pi	Wr,Ud,Fd,Ts		8
	Pe		ω	8
	C1		ω	8
	Pr		ω	8
	Pc		ω	8
	Pn		ω	8
	Cs		ω	8
	Ci		ω	8
	R		ω	8
	Tc		ω	8
	Tt		ω	8
	Tf		ω	8
Length 2				
Pi	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1		1
	Pr	Ud	Pi	4
	Pc	Ud		4
	Pn	null	Pi	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts		9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3

TMC Executive Enumeration for Length 3

Length 3				
PiC1	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiC1	4
	Pc	Ud		4
	Pn	null	PiC1	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td		1,7,4
	Tt	Wr,Ta,Ts	Pi	1,2,4
	Tf	Wr,Fa	PiR	1,3
PiPc	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1	PiC1Pc	1
	Pr	Ud	PiPr	4
	Pc	Ud	PiPc	4
	Pn	null	PiPc	5
	Cs	Ud,Uc	Pi	4
	Ci	null	PiPc	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3
PiR	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1		1
	Pr	Ud	PiR	4
	Pc	Ud		4
	Pn	null	PiR	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3

Derived Requirement

Derived Requirement

Tag	Requirement
D1	If a second 100 ms clock interrupt is received before task completion, declare a channel failure and initiate a reboot.

The discovery and documentation of derived requirements is a natural and desirable part of the sequence enumeration process.

TMC Executive Enumeration for Length 4

Length 4				
PiC1Pc	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiC1	4
	Pc	Ud	PiC1Pc	4
	Pn	null	PiC1Pc	5
	Cs	Ud,Uc	PiC1Pc	4
	Ci	null	PiC1Pc	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td		1,7,4
	Tt	Wr,Ta,Ts	PiPc	1,2,4
	Tf	Wr,Fa	PiRPc	1,3
PiC1Tc	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1		1
	Pr	Ud	PiC1Tc	4
	Pc	Ud	PiC1PcTc	4
	Pn	null	PiC1Tc	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3
PiRC1	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiRC1	4
	Pc	Ud		4
	Pn	null	PiRC1	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td	PiC1Tc	1,7,4
	Tt	Wr,Ta,Ts	PiR	1,2,4
	Tf	Wr,Fa	PiR	1,3
PiRPc	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1	PiRC1Pc	1
	Pr	Ud	PiR	4
	Pc	Ud	PiRPc	4
	Pn	null	PiRPc	5
	Cs	Ud,Uc	PiRPc	4
	Ci	null	PiRPc	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3

TMC Executive Enumeration for Length 5

Length 5				
PiC1PcTc	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1		1
	Pr	Ud	PiC1Tc	4
	Pc	Ud	PiC1PcTc	4
	Pn	null	PiC1PcTc	5
	Cs	Ud,Uc	PiC1PcTc	4
	Ci	null	PiC1PcTc	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3
PiC1TcC1	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiC1TcC1	4
	Pc	Ud	PiC1PcTcC1	4
	Pn	null	PiC1TcC1	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td	PiC1Tc	1,7,4
	Tt	Wr,Ta,Ts	Pi	1,2,4
	Tf	Wr,Fa		1,3
PiRC1Pc	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiRC1	4
	Pc	Ud	PiRC1Pc	4
	Pn	null	PiRC1Pc	5
	Cs	Ud,Uc	PiRC1Pc	4
	Ci	null	PiRC1Pc	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td	PiC1PcTc	1,7,4
	Tt	Wr,Ta,Ts	PiRPc	1,2,4
	Tf	Wr,Fa	PiRPc	1,3

TMC Executive Enumeration for Length 6

Length 6				
PiC1PcTcC1	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiC1TcC1	4
	Pc	Ud	PiC1PcTcC1	4
	Pn	null	PiC1PcTcC1	5
	Cs	Ud,Uc	PiC1PcTcC1	4
	Ci	null	PiC1PcTcC1	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td	PiC1PcTc	1,7,4
	Tt	Wr,Ta,Ts	PiPc	1,2,4
	Tf	Wr,Fa		1,3
PiC1TcC1Tf	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1		1
	Pr	Ud	PiC1TcC1Tf	4
	Pc	Ud	PiC1PcTcC1Tf	4
	Pn	null	PiC1TcC1Tf	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3

TMC Executive Enumeration for Length 7 & 8

Length 7				
PiC1PcTcC1Tf	Pi		ω	4
	Pe	Ts	λ	9
	C1	T1		1
	Pr	Ud	PiC1TcC1Tf	4
	Pc	Ud	PiC1PcTcC1Tf	4
	Pn	null	PiC1PcTcC1Tf	5
	Cs	Ud,Uc	PiC1PcTcC1Tf	4
	Ci	null	PiC1PcTcC1Tf	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc		ω	1
	Tt		ω	1,2
	Tf		ω	1,3
PiC1TcC1TfC1	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiC1TcC1TfC1	4
	Pc	Ud	PiC1PcTcC1TfC1	4
	Pn	null	PiC1TcC1TfC1	5
	Cs		ω	4
	Ci		ω	4
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td	PiC1Tc	1,7,4
	Tt	Wr,Ta,Ts	PiR	1,2,4
	Tf	Wr,Fa	PiC1TcC1Tf	1,3
Length 8				
PiC1PcTcC1TfC1	Pi		ω	4
	Pe	Ts	λ	9
	C1	Ri,Wr,Ud,Fa,Ta,Ts	PiR	D1
	Pr	Ud	PiC1TcC1TfC1	4
	Pc	Ud	PiC1PcTcC1TfC1	4
	Pn	null	PiC1PcTcC1TfC1	5
	Cs	Ud,Uc	PiC1PcTcC1TfC1	4
	Ci	null	PiC1PcTcC1TfC1	6
	R	Ri,Wr,Ud,Fa,Ta,Ts	PiR	8,9
	Tc	Wr,Fd,Td	PiC1PcTc	1,7,4
	Tt	Wr,Ta,Ts	PiR Pc	1,2,4
	Tf	Wr,Fa	PiC1PcTcC1Tf	1,3

Canonical Sequence Analysis

Construct a table and list the canonical sequences in the order enumerated.

Define one or more state variables such that each canonical sequence corresponds to a unique combination of values.

TMC Executive Canonical Sequences*

Sequence	TaskPending	CallPending	Fail	Trip	Sequence	TaskPending	CallPending	Fail	Trip
Pi	False	False	False	True	Pi C1 Pc Tc	False	True	False	False
Pi C1	True	False	False	True	Pi C1 Tc C1	True	False	False	False
Pi Pc	False	True	False	True	Pi R C1 Pc	True	True	True	True
Pi R	False	False	True	True	Pi C1 Pc Tc C1	True	True	False	False
Pi C1 Pc	True	True	False	True	Pi C1 Tc C1 Tf	False	False	True	False
Pi C1 Tc	False	False	False	False	Pi C1 Pc Tc C1 Tf	False	True	True	False
Pi R C1	True	False	True	True	Pi C1 Tc C1 Tf C1	True	False	True	False
Pi R Pc	False	True	True	True	Pi C1 Pc Tc C1 Tf C1	True	True	True	False

*other than λ

State Machine Specification Generation

For each stimulus,

For each canonical sequence,

Determine response and any changes to state variable(s) after stimulus is applied

Tabulate results

Remove illegal transitions

Combine rows for which possible state variable changes and responses are identical.

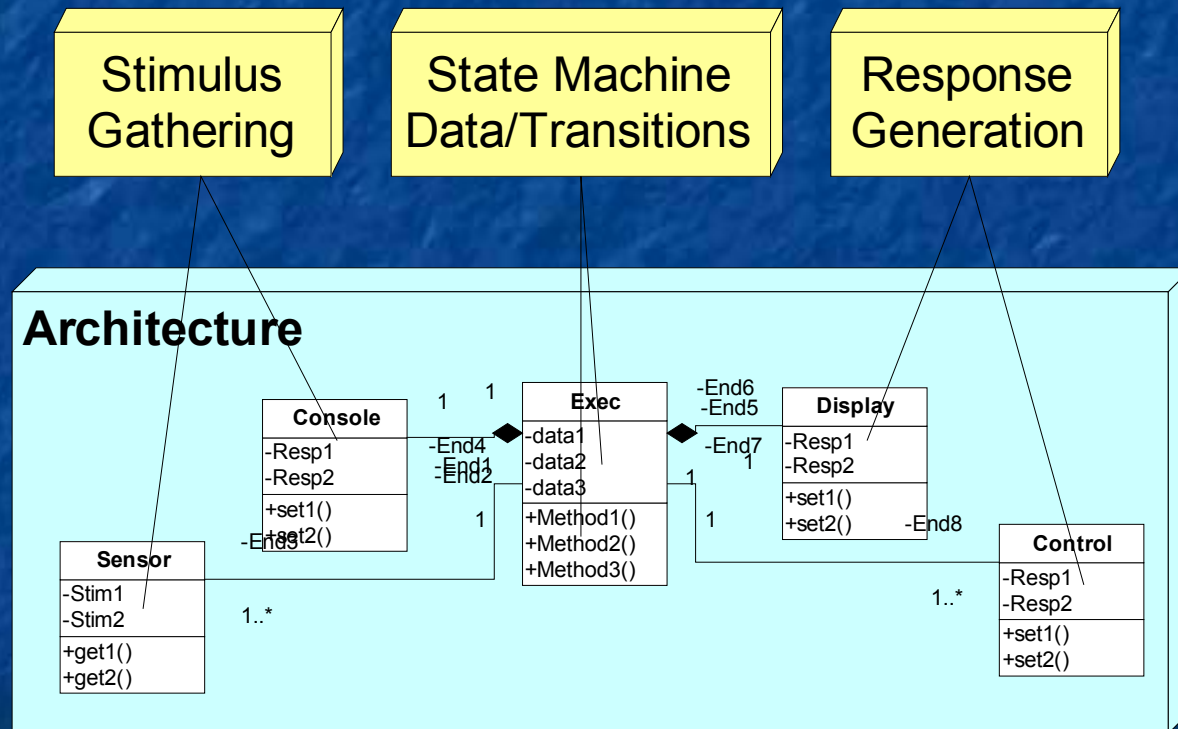
TMC State Machine Excerpt for Stimulus Tc

Beginning State				End State				Response
TaskPending	CalPending	Fail	Trip	TaskPending	CalPending	Fail	Trip	
No	No	No	Yes					illegal
Yes	No	No	Yes	No	No	No	No	Wr,Fd,Td
No	Yes	No	Yes					illegal
No	No	Yes	Yes					illegal
Yes	Yes	No	Yes	No	Yes	No	No	Wr,Fd,Td
No	No	No	No					illegal
Yes	No	Yes	Yes	No	No	No	No	Wr,Fd,Td
No	Yes	Yes	Yes					illegal
No	Yes	No	No					illegal
Yes	No	No	No	No	No	No	No	Wr,Fd,Td
Yes	Yes	Yes	Yes	No	Yes	No	No	Wr,Fd,Td
Yes	Yes	No	No	No	Yes	No	No	Wr,Fd,Td
No	No	Yes	No					illegal
No	Yes	Yes	No					illegal
Yes	No	Yes	No	No	No	No	No	Wr,Fd,Td
Yes	Yes	Yes	No	No	Yes	No	No	Wr,Fd,Td

Reduced TMC State Machine Table for Stimulus Tc

Initial State	Next State	Response	Trace
TaskPending = False		Illegal	1
TaskPending = True	TaskPending = False, Fail = False, Trip = False	Wr,Fd,Td	1,7,4

State Machine Code



Transformation to Formal Notation

General Approach

Encode State Machine **or**

Direct Transformation of Enumeration

Candidate Notations: Z, SCR, CSP, TAM,
CCS, ACL2

Transformation to Formal Notation

ACL2 Example

Assign a number, f , to each Canonical Sequence.

Map each Sequence, s , in the Enumeration to f for the equivalent Canonical Sequence.

Map each (f,s) Pair to the correct response, $resp$.

ACL2 Example

Can be generated
automatically from
enumeration

Used for

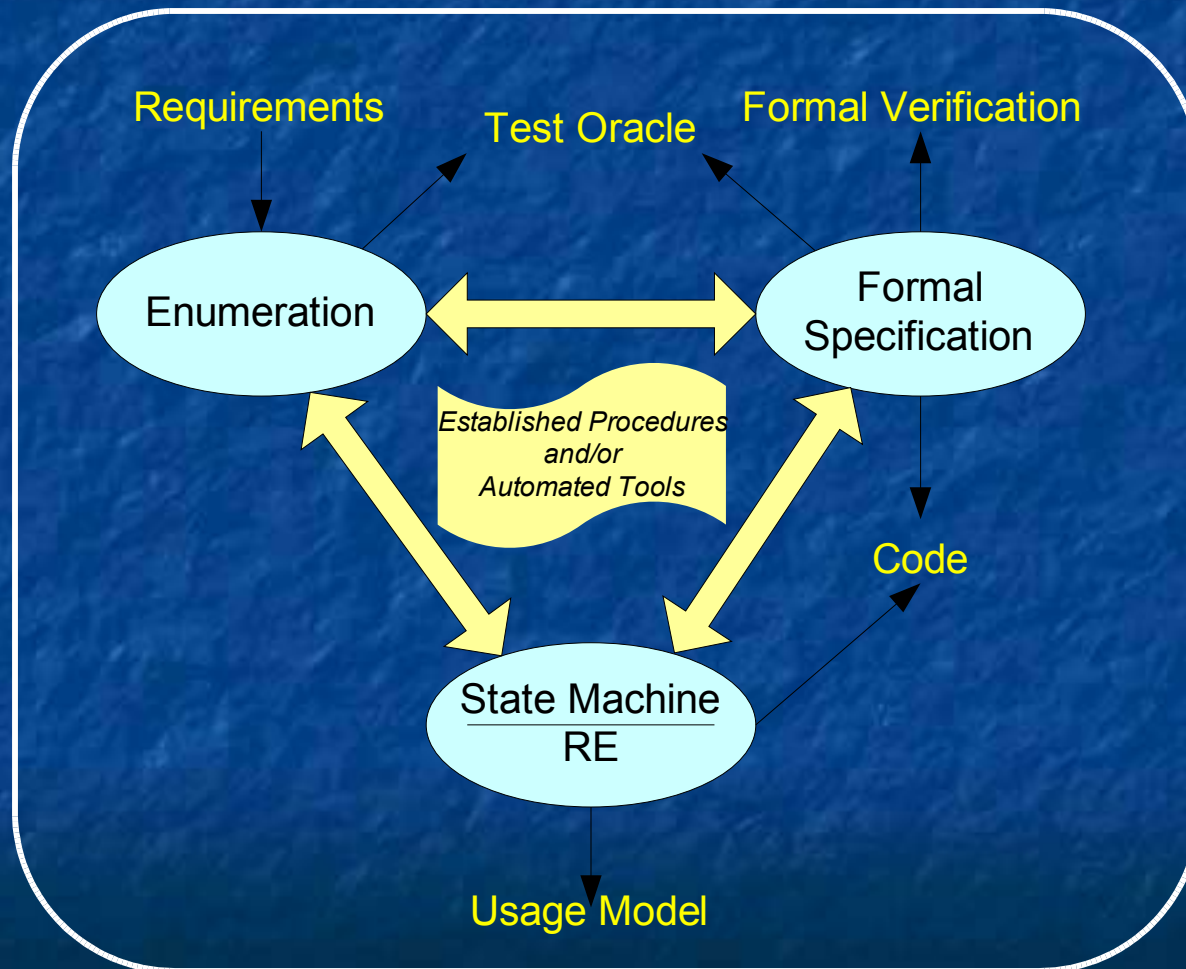
Simulation

Test Oracle

Extended verification
via model checkers
and/or theorem provers

```
; Definition of resp.
(defun resp (s)
  (if (endp s) `null
      (let ((C (f (cdr s))))
        (case (car s)
          ; Other stimuli omitted here
          (Tc (case C
                (1 'illegal)
                (2 '(Wr Fd Td))
                (3 'illegal)
                (4 'illegal)
                (5 '(Wr Fd Td))
                (6 'illegal)
                (7 '(Wr Fd Td))
                (8 'illegal)
                (9 'illegal)
                (10 '(Wr Fd Td))
                (11 '(Wr Fd Td))
                (12 '(Wr Fd Td))
                (13 'illegal)
                (14 'illegal)
                (15 '(Wr Fd Td))
                (16 '(Wr Fd Td))))))))))
```

Overview



Summary and Conclusion

Sequence-Based Specification is a method for producing **consistent, complete, and traceably correct** software specifications.

The method is based on a simple **sequence enumeration** procedure and basic requirements analysis skills.

The results can be freely converted to state machines and other formal representations for

Automatic Code Generation

Formal Verification via Theorem Provers

Test Case Generation and Results Analysis