California Institute of Technology

Automated Specification-Based Test Case Generation Using SCR

Allen Nikora
Quality Assurance Office (512)
Jet Propulsion Laboratory

Constance L. Heitmeyer
Head, Software Engineering Section
Naval Research Laboratory

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology. This work was sponsored by the Software Engineering Technology area of JPL's Center for Space Mission Information and Software Systems (CSMISS).

Outline

- Task Objectives
- Work Accomplished
  - SCR Specification of FPE
  - Simulation of FPE, graphical interface for simulator
  - Test Cases
- Issues
- Conclusions
- Future Work
  - Deployment
  - Extension of FPE specification
  - Complete FPE requirements document/user's manual

Task Objectives

- Pilot the use of the SCR-based test case generator on real development efforts. Initial work will be accomplished using Deep Impact FP Engine (FPE)
  - JPL will use different FPE implementations in various spacecraft.
  - JPL will need high reliance in the correctness of each version of the FPE code.
  - Develop a "good" specification of the FPE behavior.
    - Describes precisely and clearly normal and abnormal behavior
    - Avoids both overspecification and underspecification
    - Describes likely ways that the FPE component will change
    - A set of critical properties that the FPE code must satisfy
    - Proof that the spec satisfies the properties and validation that the spec captures the intended behavior
  - Construct a set of test cases satisfying some coverage criteria from the specification.
- Determine how to most effectively use the test case generator and other SCR capabilities on JPL projects

California Institute of Technology

Work Accomplished

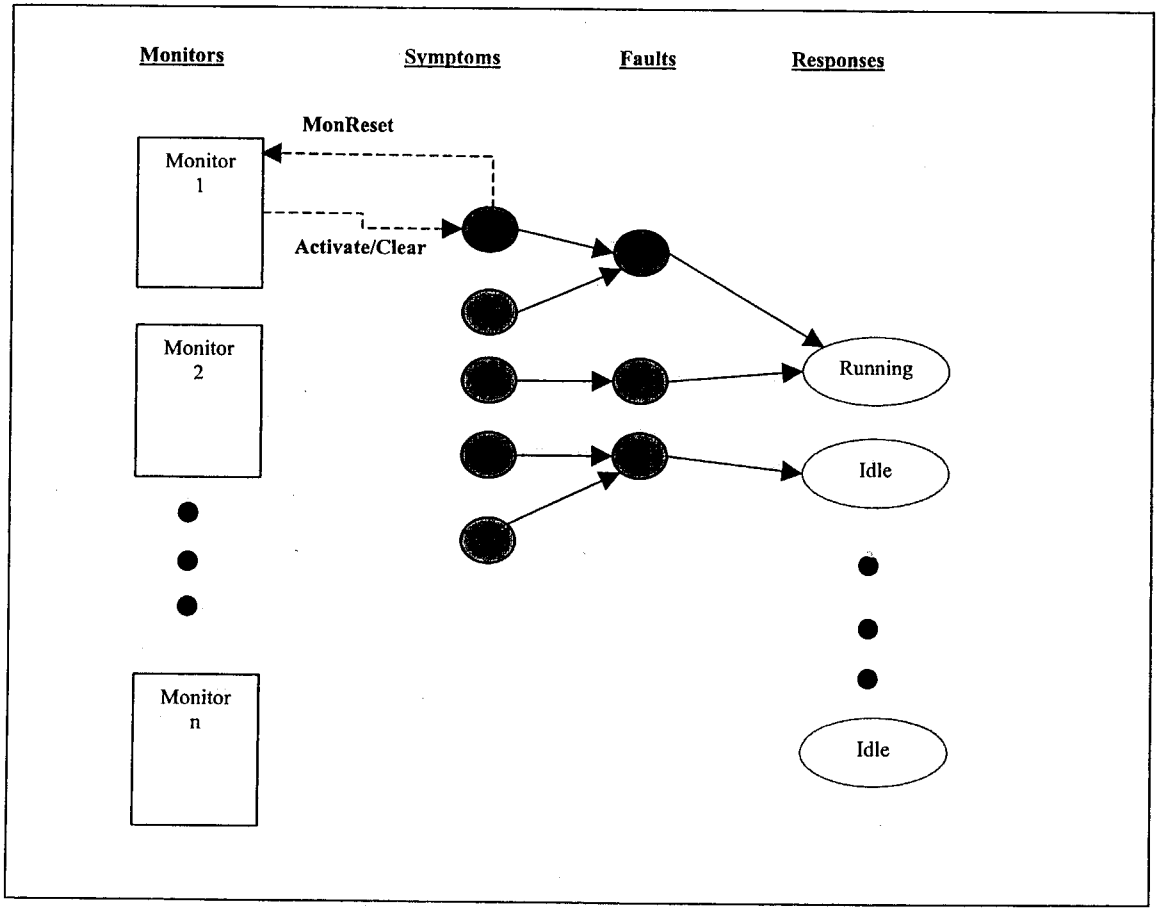
- SCR Specification
- SCR Simulator
- Test Cases

SCR Specification Overview

- SCR specification of Fault Protection Engine based on:
  - Final report of an effort to model the Fault Protection Engine using SDL.
  - Stateflow diagrams for FP engine available from JPL-internal website
  - Deep Impact FP Engine design documentation – available from the on-line Deep Impact project library at JPL

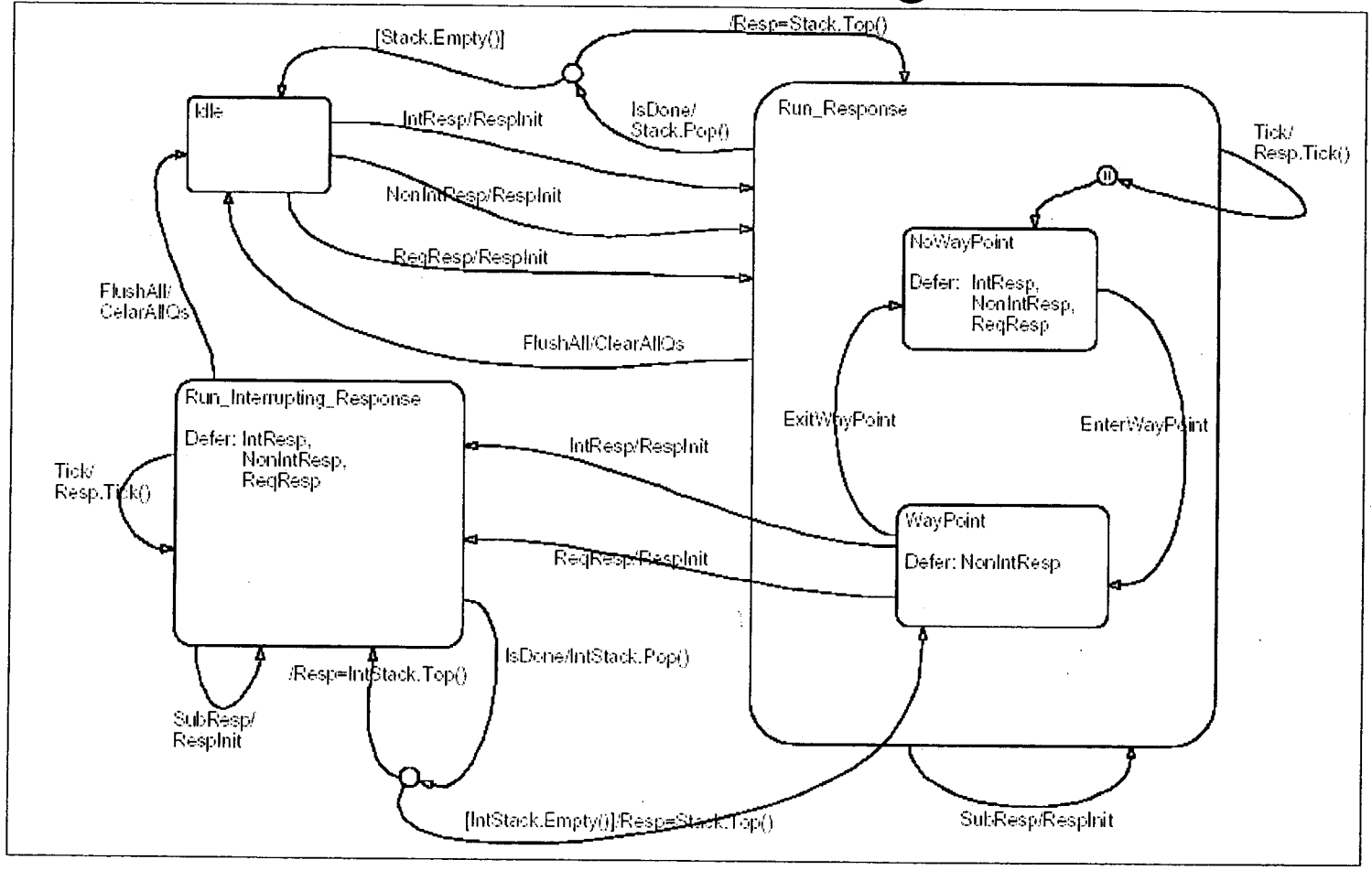
SCR Specification Overview (cont'd)

FPE Symptom/Fault/Response Mapping



SCR Specification Overview (cont'd)

Fault Protection Engine



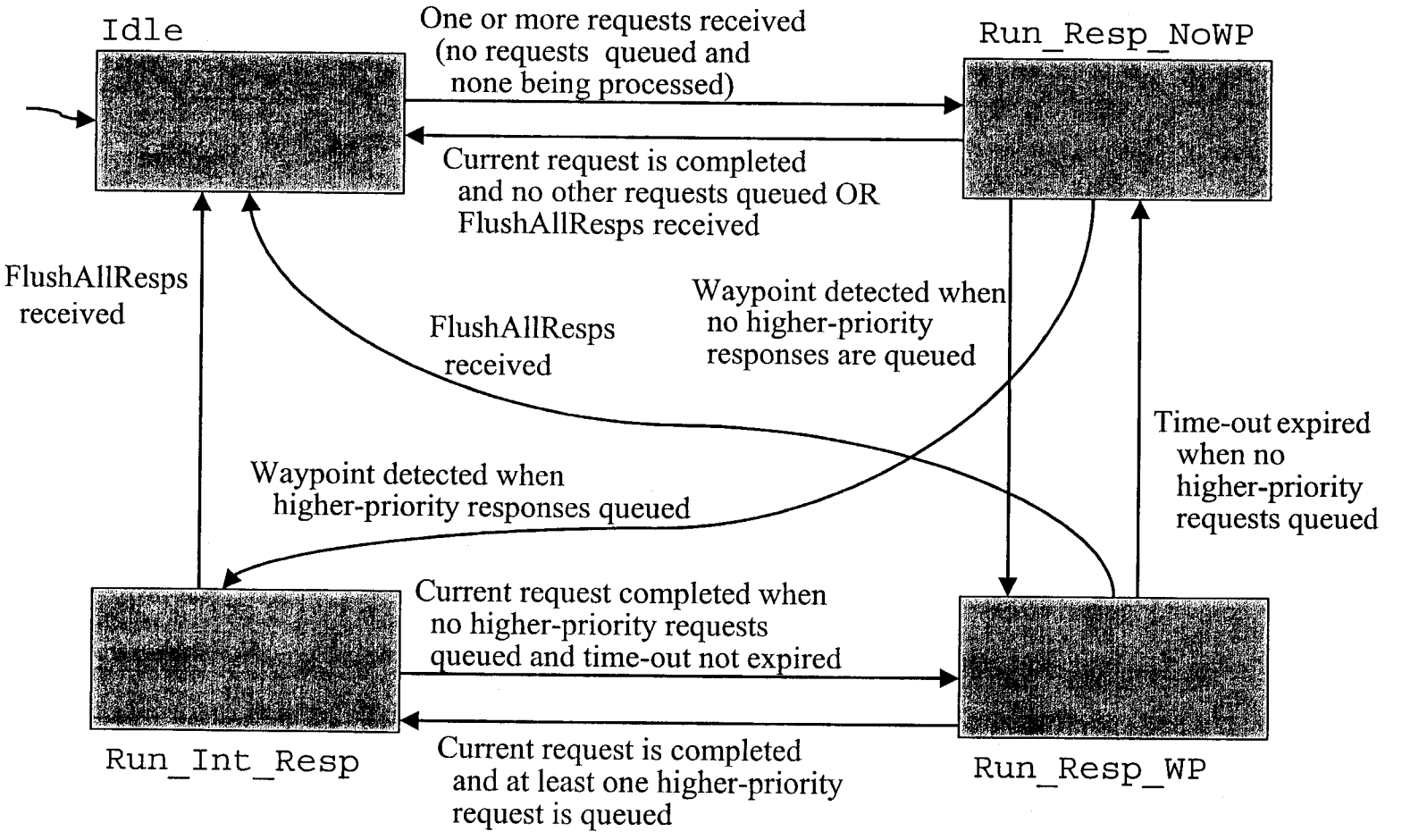
SCR Specification Overview (cont'd)

JPL California Institute of Technology

- Simplifying Abstractions
  - No sub-responses
    - For example, on slide 7, "SubResp/RespInit", from Run_Response to Run_Response. When the FPE encounters a call to a sub-response, it will suspend the currently-running response and cause the named sub-response to begin executing.

    To simplify this version of the SCR specification, we did not include in it this aspect of the FPE.
  - The maximum number of responses of each type is significantly smaller in the specification than it would be in a real spacecraft.
    - Real space missions can have 20 or more different fault responses.
    - Lower number of responses of each type is adequate to accurately model the interactions between the different types of response requests that the FPE could encounter.
  - The response deferral mechanism in the implemented FPE is somewhat more complicated that what is shown in the specification.
    - In the SCR specification, no response deferral queue can have two responses having the same ID.
    - In the implemented FPE, a given fault cannot queue the same response twice.

SCR Specification Overview (cont'd)



STD for FPE specification

SCR Specification Overview (cont'd)

Statechart			SCR Specification		
Begin State	End State	Transition Event	Begin State	End State	Transition Event
Idle	Run_Response	Received request to run a response OR there are one or more deferred responses	Idle	Run_Resp_NoWP	Received request to run a response OR there are one or more deferred responses
Run_Response, Run_Interrupting_ Response	Idle	Received request to flush all responses	Run_Resp_NoWP, Run_Resp_WP, Run_Int_Resp	Idle	Received request to flush all responses
Run_Response	Idle if stack is empty, Run_Response if stack is not empty	-	Run_Resp_NoWP	Idle	Response completed
NoWayPoint	WayPoint	Waypoint encountered in response	Run_Resp_NoWP	Run_Resp_WP	Waypoint encountered in response

Comparison of SCR Specification to FPE statechart

SCR Specification Overview (cont'd)

Statechart			SCR Specification		
Begin State	End State	Transition Event	Begin State	End State	Transition Event
-	-	-	Run_Resp_NoWP	Run_Int_Resp	Waypoint encountered in response AND there are one or more deferred interrupting or ground-requested responses.
Run_Interrupting_ Response	WayPoint	Interrupting or ground requested response completes.	Run_Int_Resp	Run_Resp_WP	Interrupting or ground requested response completes prior to expiration of waypoint.
-	-	-	Run_Int_Resp	Run_Resp_NoWP	Waypoint expired prior to completing interrupting or ground requested response.

Comparison of SCR Specification to FPE statechart (cont'd)

SCR Specification Overview (cont'd)

Statechart			SCR Specification		
Begin State	End State	Transition Event	Begin State	End State	Transition Event
WayPoint	NoWayPoint	Waypoint has expired.	Run_Resp_WP	Run_Resp_NoWP	Waypoint has expired.
WayPoint	Run_Interrupting_ Response	Request for interrupting or ground requested response received OR there are one or more deferred interrupting or ground-requested responses.	Run_Resp_WP	Run_Int_Resp	Request for interrupting or ground requested response received.

Comparison of SCR Specification to FPE statechart (cont'd)

SCR Specification Overview (cont'd)

Statechart	SCR Specification	Comments
Monitored Variables		
IntResp	mRespRequest	mRespRequest is a variable whose value is a three-digit number. The least significant digit represents the ID of a non-interrupting response, the next least significant digit represents the ID of an interrupting response, and the most significant digit represents the ID of a ground-request response. These could have been specified as three separate monitored variables. Since more than one response can be requested at any given time, however, specifying the variable in this manner simplified the specification.
NonIntResp	mRespRequest	See above
ReqResp	mRespRequest	See above
IsDone	MrespDone	A signal indicating that the currently executing response has completed. In the SCR specification, this signal is viewed as coming from the sequencer that actually executes the instructions within a response. The functionality and behavior of the sequencer are not included in the SCR specification.
FlushAll	mFlushAllResps	A signal to the FPE to terminate the currently-executing response and cancel all deferred response requests.

Monitored and Controlled Variables

SCR Specification Overview (cont'd)

Statechart	SCR Specification	Comments
Monitored Variables		
EnterWayPoint	mWayPoint	A signal to the FPE indicating that a (non-interrupting) response has encountered a waypoint. In the SCR specification, this is viewed as a signal from the sequencer actually executing the response's instructions.
ExitWayPoint	mTimeOut	These data items signal the end of a waypoint within a (non-interrupting) response. To make the timeout more visible, we defined separate signals for entering a waypoint and waypoint timeout.
Controlled Variables		
RespInit	cResp_Request	This variable indicates the ID and type of the request that should be executed next. In the SCR specification, the variable is represented as a three-digit non-zero number, where exactly one digit is non-zero, the position of the non-zero digit indicates the response type, and the digit value indicates the response ID.

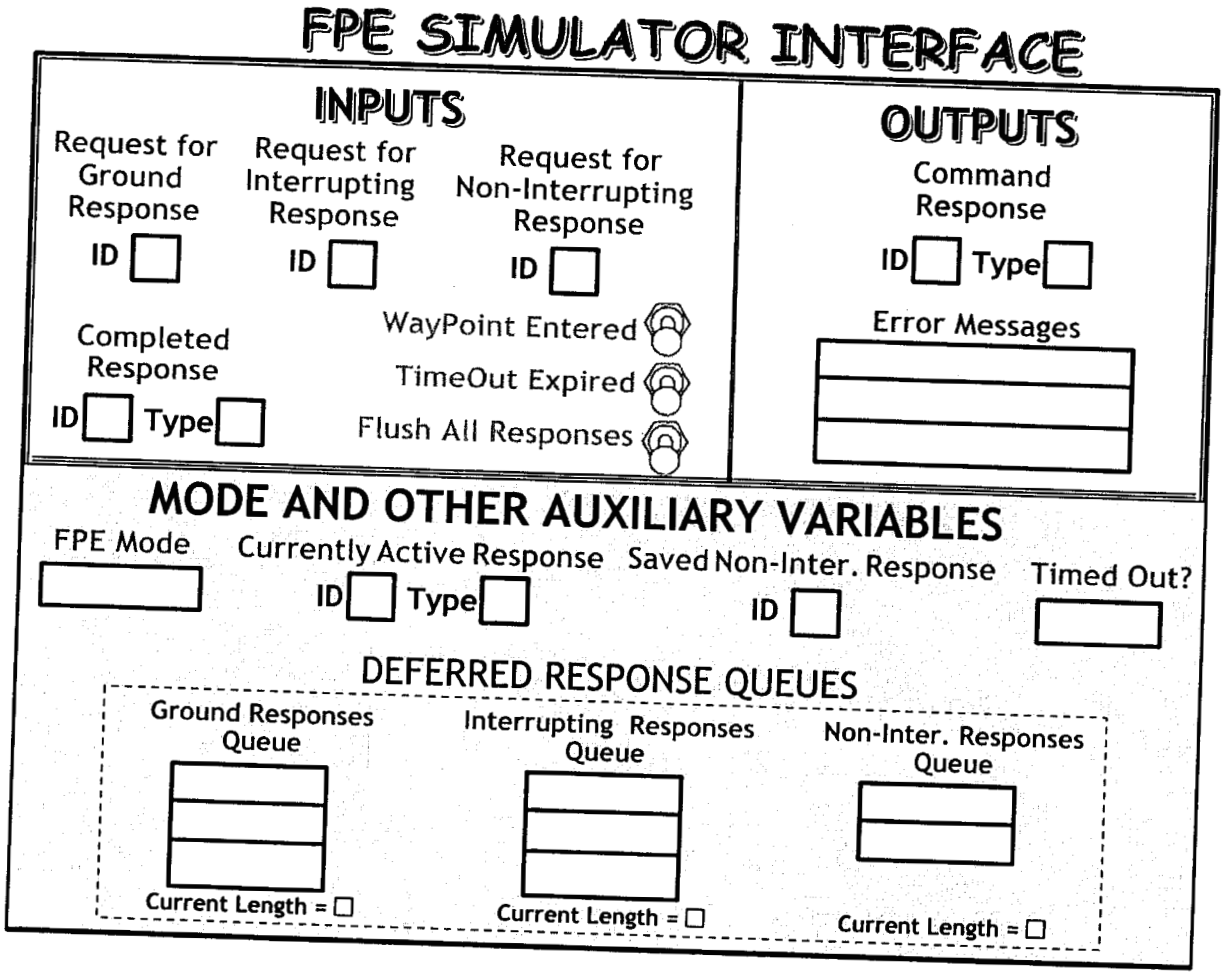
Monitored and Controlled Variables (cont'd)

Simulator Overview

- SCR toolset includes facilities for generating a simulation for a specification
- Created a simulation of the FPE specification to better understand FPE behavior. Simulation was useful for:
  - Detecting faults in the specification
  - Identifying areas of the specification that needed further clarification by the FPE developers
  - Presenting the functionality and behavior of the FPE to management
- Up to four scenarios will be demonstrated
  - One Non-Interrupting, One Interrupting Response
  - Two Non-Interrupting Responses
  - One Non-Interrupting, Two Interrupting Responses
  - One Non-Interrupting, Two Interrupting, Two Ground-Requested Responses

Simulator Overview (cont'd)

California Institute of Technology



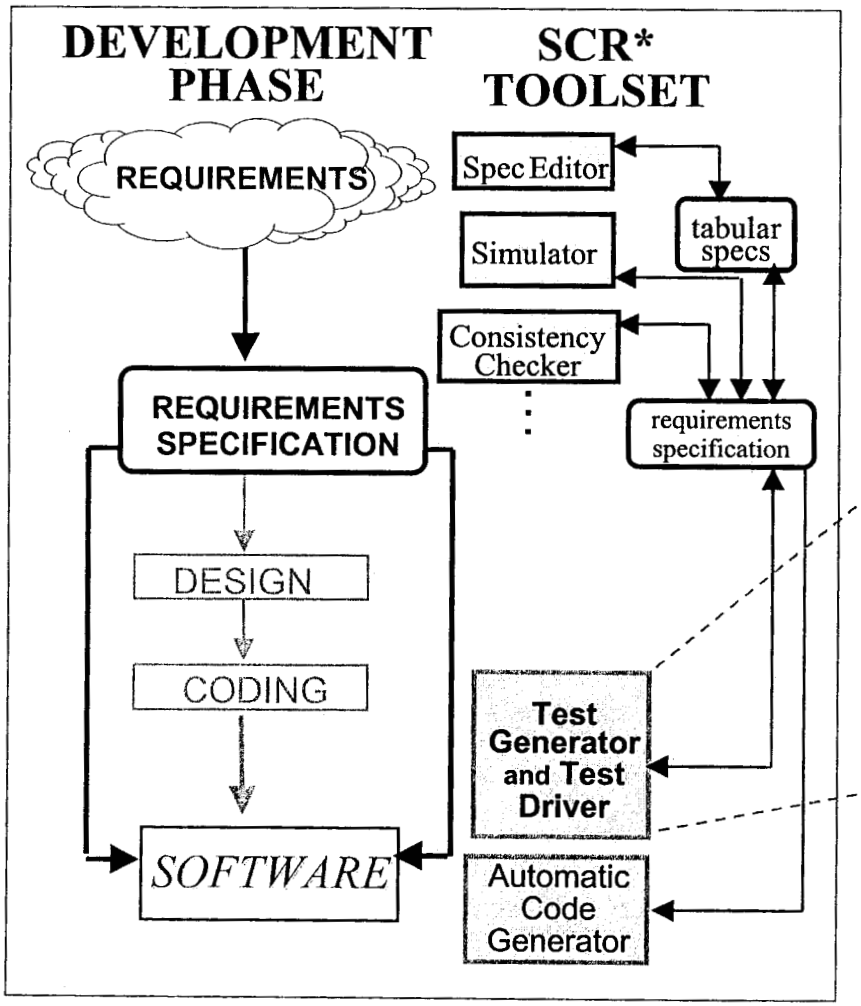
FPE Simulator GUI

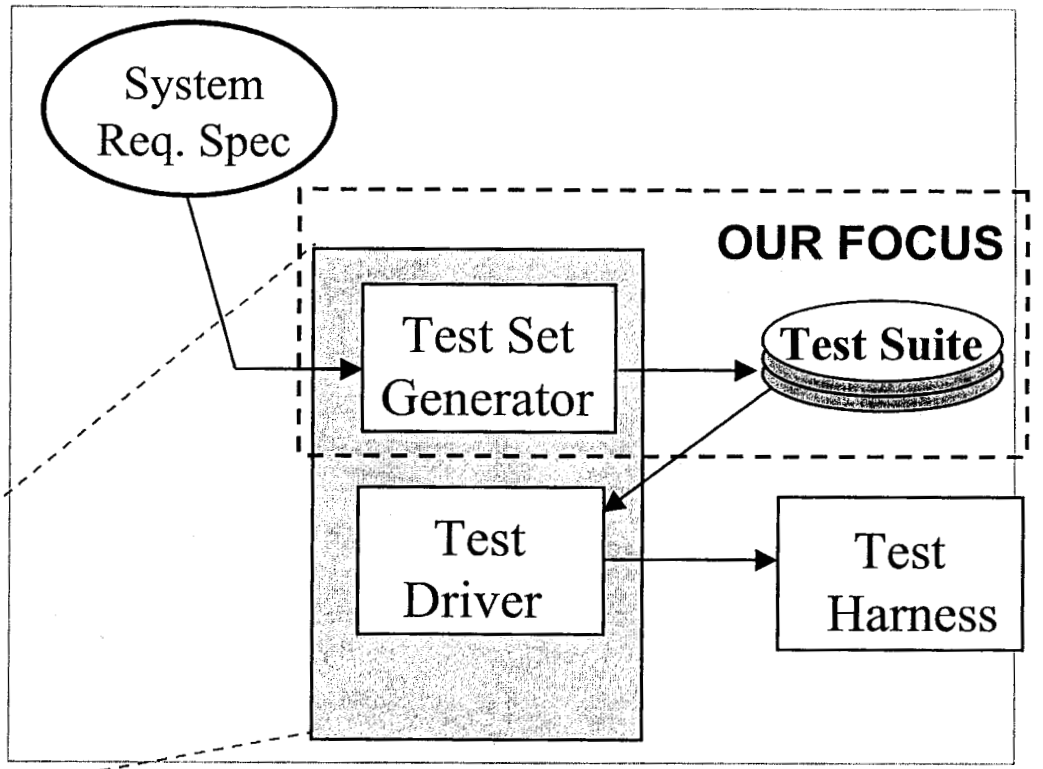
Test Cases Overview

- Generated according to mode transition table defined in specification
- Test cases expressed in terms of externally-visible inputs and outputs
- Test cases cover all transitions defined in mode transition table
  - Nominal behavior
  - Some error behavior

Extending SCR To Automatic Test Set Generation

California Institute of Technology





Our approach to software testing
- specification-based
- blackbox--does the software satisfy the requirements specification?

Example: The Mode Transition Table From The SCR Spec Of The FPE

Old Mode	Event	New Mode
Idle	@C(mResp_Request) AND…	Run_Resp_NoWP
Run_Resp_NoWP	@T(mWayPoint) when tCurrentReqType=NR and tIRq_len=0 and tGRq_len=0	Run_Resp_WP
…	…	…
Run_Resp_WP	@T(mFlushAllResps)	Idle

Table Defining the Value of FPEMode

Example: The Mode Transition Table From The SCR Spec Of The FPE (cont'd)

If

- □ FPEMode = Idle → FPEMode' = Run_Resp_NoWP
  ∧ @C(mResp_Request) AND ...
- □ FPEMode = Run_Resp_NoWP → FPEMode' = Run_Resp_WP
  ∧ @T(mWayPoint) when tCurrentReqType=NR & tIRq_len=0 & tGRq_len=0

...

- □ FPEMode = Run_Resp_WP → FPEMode' = Run_Resp_NoWP
  ∧ @T(mFlushAllResps)
- □ *(else)* → FPEMode' = FPEMode

fi

Total function that the table defines (single else clause)

Constructing Test Cases From A Mode Transition Table (1)

California Institute of Technology

Alternate Representation of the Function with the else Clause Distributed

```
if
❑ FPEMode = Idle
  if
    ❑ @C(mResp_Request) & ...      →  FPEMode' = Run_Resp_NoWP      C1
    ❑ (else)                       →  FPEMode' = FPEMode            C1else
  fi
❑ FPEMode = Run_Resp_NoWP
  if
    ❑ @C(mResp_Done) & ...         →  FPEMode' = Idle               C2
    ❑ @T(WayPoint) & ...           →  FPEMode' = Run_Resp_WP        C3
    ❑ @T(WayPoint) & ...           →  FPEMode' = Run_Int_Resp       C4
    ❑ (else)                       →  FPEMode' = FPEMode            C2else
  fi

❑ FPEMode = Run_Resp_WP
  if
    ❑ ...                          →  FPEMode' = ...                ...
    ❑ (else)                       →  FPEMode' = FPEMode            ...
  fi

❑ FPEMode = Run_Int_Resp
  if
    ❑ ...                          →  FPEMode' = ...                ...
    ❑ (else)                       →  FPEMode' = FPEMode            ...
  fi
fi
```

Constructing Test Cases From A Mode Trans. Table (2)

- Each part of the function definition is called a case
- Each case defines a set of state transitions
- Because each function is total, the set of test cases cover the entire state space
- Because the cases are mutually exclusive, each case is an equivalence class of system executions with the same two final states

For example, case *C1* defines the set of executions whose final two states satisfy the following property:

$$\text{FPEMode} = \text{Idle} \wedge @C(\text{mResp_Request}) \;\&\; \ldots \Rightarrow \text{FPEMode}' = \text{Run_Resp_NoWP}$$

Test Cases Overview (cont'd)

California Institute of Technology

Source Mode	Events	Destination Mode	Test Case
Idle	@C(mResp_Request) AND (mResp_Request' > 0 AND ((tNR_ID' > 0 AND tNR_ID' <= MaxID) OR (tIR_ID' > 0 AND tIR_ID' <= MaxID) OR (tGR_ID' > 0 AND tGR_ID' <= MaxID)))	Run_Resp_NoWP	C1
	ELSE	Idle	C1else
Run_Resp_NoWP	@C(mResp_Done) AND (mResp_Done'= cResp_Request AND tNoReqsQd) OR @T(mFlushAllResps)	Idle	C2
	ELSE	Run_Resp_NoWP	C2else
Run_Resp_NoWP	@T(mWayPoint) WHEN (tCurrent_Req_Type = NR ANDtIRq_len = 0 AND tGRq_len = 0)	Run_Resp_WP	C3
	ELSE	Run_Resp_NoWP	C3else
Run_Resp_NoWP	@T(mWayPoint) WHEN (tCurrent_Req_Type = NR AND(tIRq_len > 0 OR tGRq_len > 0))	Run_Int_Resp	C4
	ELSE	Run_Resp_NoWP	C4else
Run_Int_Resp	@C(mResp_Done) AND (mResp_Done'= cResp_Request AND tIRq_len = 0 AND tGRq_len = 0 AND tTimeOut=false)	Run_Resp_WP	C5
	ELSE	Run_Int_Resp	C5else

Correspondence Between Test Cases and Mode Transitions

Test Cases Overview (cont'd)

California Institute of Technology

Source Mode	Events	Destination Mode	Test Case
Run_Int_Resp	@C(mResp_Done) AND (mResp_Done'= cResp_Request AND tTimeOut=true) **ELSE**	Run_Resp_NoWP Run_Int_Resp	C6 C6else
Run_Int_Resp	@T(mFlushAllResps) **ELSE**	Idle Run_Int_Resp	C7 C7else
Run_Resp_WP	@C(mTimeOut) WHEN (tIRq_len = 0 AND tGRq_len = 0) **ELSE**	Run_Resp_NoWP Run_Resp_WP	C8 C8else
Run_Resp_WP	@C(mResp_Request) AND ((tGR_ID' != tGR_ID AND tGR_ID' > 0) OR (tIR_ID' != tIR_ID ANDtIR_ID' > 0)) **ELSE**	Run_Int_Resp Run_Resp_WP	C9 C9else
Run_Resp_WP	@T(mFlushAllResps) **ELSE**	Idle Run_Resp_WP	C10 C10else

Correspondence Between Test Cases and Mode Transitions (cont'd)

Test Cases Overview (cont'd)

California Institute of Technology

Individual Test Cases

---------C1---------	---------C1---------
mResp_Request 1	cResp_Request 1

---------C2---------	---------C2---------
mResp_Request 1	cResp_Request 1
mResp_Done 1	cResp_Request 0

---------C3---------	---------C3---------
mFlushAllResps TRUE	<>
mResp_Request 1	cResp_Request 1
mWayPoint TRUE	cResp_Request 0

---------C4---------	---------C4---------
mResp_Request 1	**cResp_Request 1**
mResp_Request 10	<>
mWayPoint TRUE	**cResp_Request 10**

---------C5---------	---------C5---------
mResp_Request 1	cResp_Request 1
mResp_Request 21	<>
mWayPoint TRUE	cResp_Request 20
mResp_Done 10	cResp_Request 0

---------C6---------	---------C6---------
mResp_Request 2	cResp_Request 2
mResp_Request 4	<>
mResp_Request 11	<>
mWayPoint TRUE	cResp_Request 10
mTimeOut TRUE	<>
mResp_Done 10	cResp_Request 2

---------C7---------	---------C7---------
mResp_Request 9	**cErrMsgBadID = ID_Out_of_Range**
mResp_Request 3	**cErrMsgBadID = null**
	cResp_Request 3
mResp_Request 13	<>
mWayPoint TRUE	**cResp_Request 10**
mFlushAllResps TRUE	**cResp_Request 0**

---------C8---------	---------C8---------
mFlushAllResps TRUE	<>
mResp_Request 1	cResp_Request 1
mWayPoint TRUE	cResp_Request 0
mTimeOut TRUE	cResp_Request 1

---------C9---------	---------C9---------
mFlushAllResps TRUE	<>
mResp_Done 1	<>
mResp_Request 3	cResp_Request 3
mWayPoint TRUE	cResp_Request 0
mResp_Request 10	cResp_Request 10

---------C10---------	---------C10---------
mResp_Request 4	cResp_Request 4
mWayPoint TRUE	cResp_Request 0
mFlushAllResps TRUE	<>

NOTES

- Test case C1 may be eliminated because it is contained in test case C2.
- In many cases, for example, the first step of test case C3, an input does not generate a change in a controlled variable (above, no change is represented by <>).
- The second input of test case C7 produces changes in two controlled variables.
- Some of the test cases are not the shortest possible tests. For example, the first two steps of test case C9 could be deleted, since they have no effect on the state or on the controlled variables.

Test Cases Overview (cont'd)

California Institute of Technology

Individual Test Cases (cont'd)

-------C1else-------- -------C1else-------- Eliminate --
mFlushAllResps TRUE <> OVERLAPPED BY C9

-------C2else-------- -------C2else--------
mResp_Request 1 cResp_Request 1
mResp_Request 2 <>

-------C3else-------- -------C3else--------
mResp_Request 1 cResp_Request 1
mResp_Request 3 <>

-------C4else-------- -------C4else--------
mResp_Request 1 cResp_Request 1
mResp_Request 7 cErrMsgBadID = ID_Out_of_Range

-------C5else-------- -------C5else--------
mResp_Request 4 cResp_Request 4
mWayPoint TRUE cResp_Request 0
mResp_Request 10 cResp_Request 10
mResp_Request 500 <>

-------C6else-------- -------C6else-------- Eliminate -- OVERLAPPED BY C6
mResp_Request 2 cResp_Request 2
mResp_Request 4 <>
mResp_Request 11 <>
mWayPoint TRUE cResp_Request 10
mTimeOut TRUE <>

-------C7else-------- -------C7else--------
mResp_Request 3 cResp_Request 3
mResp_Request 10 <>
mWayPoint TRUE cResp_Request 10
mResp_Request 2 <>

-------C8else-------- -------C8else--------
mResp_Request 1 cResp_Request 1
mWayPoint TRUE cResp_Request 0
mResp_Request 3 <>

-------C9else-------- -------C9else--------
mFlushAllResps TRUE <>
mResp_Done 1 <>
mResp_Request 1 cResp_Request 1
mWayPoint TRUE cResp_Request 0
mFlushAllResp <>

-------C10else-------- -------C10else--------
mResp_Request 1 cResp_Request 1
mWayPoint TRUE cResp_Request 0
mResp_Request 2 <>

NOTES

- Test cases C1else and C6else may be eliminated because they are contained in test cases C9 and C6, respectively.

Issues

- Unspecified Behavior in Available Documentation
  - The priority of the different types of responses was not specified in the available documentation.
  - Although non-interrupting responses are the only type of response intended to have waypoints, there is no on-board enforcement mechanism. This allows the following possibilities:
    - An interrupting or a ground-requested response could implement way-points
    - A sub-response to an interrupting or non-interrupting response could have waypoints.

    Under these circumstances, the behavior of the FPE is not defined.
  - The descriptive material used as the basis for the specification developed for this task does not define the behavior of the FPE if it receives a request for a non-existent response. A diagnostic message will be displayed during simulation runs if this situation arises.
- Test Case Generation
  - Test cases produced with SPIN can result in very long test cases because of the algorithms SPIN uses in generating counterexamples.
  - Future work will investigate the use of other model checkers to produce shorter test cases.

Conclusions

California Institute of Technology

What have we done

- Demonstrated feasibility of constructing a set of test sequences from an operational req. specification using a model checker
- Have done so in a manner that "covers" all possible system executions described by the requirements specification
- Demonstrated how one can construct from the spec a set of two-state properties (i.e., cases) that describe all possible system behaviors

What next?

- How to improve the scalability of the method
  - Apply abstraction methods to model checking
  - Develop an algorithm to directly build a test sequence from a property
- Our method currently builds one test sequence per property: how can more than one effective test sequence be built from a single property
  - Statistical methods
  - Case splitting
  - A method such as that of Weyuker et al. [TSE, May94].
- Systematically consider fault-tolerant behavior
  - Example – if a response ID does not satisfy its type, the specification does not state how the FPE will deal with this situation.

Conclusions (cont'd)

California Institute of Technology

- SCR test generation facility appears to be appropriate for components or systems at the FPE level of complexity
  - Compositional reasoning will allow us to compose SCR specifications of the different pieces of the FPE.
- ***Almost all effort is in the development of the specification.***
- After gaining familiarity with SCR, development of specs is fairly rapid
  - Mechanics of translating statecharts to SCR specifications is straightforward
  - Information not specified in statecharts must be gathered by interviewing developers (e.g., FP response priorities)
- FP Engine represents a type of system to which SCR has not previously been applied
  - FPE algorithm involves many complex constructs that do not normally arise in embedded systems
    - e.g., feedback loops, queuing, simult. events, priorities, error messages, etc.
  - Does not satisfy Synchronous Hypothesis (i.e., inputs are completely consumed before another input is received)

Conclusions (cont'd)

- SCR specification captures the required behavior in an understandable way
  - Easy to change when errors are detected
  - Easy to change when one needs a different version of the FPE algorithm
  - People can be easily taught to understand the spec language
- The SCR specification is executable, allowing
  - Automatic checking for syntax and type errors, missing cases, unwanted non-determinism, circular definitions
  - Automatic construction of a simulator model of the FPE, which is useful for demonstrating and validation the spec
  - Automatic verification/refutation using model checkers/theorem provers (future)

Future Work

- Deployment
  - Work with other projects to identify appropriate components to which technique could be applied.
- Extend FPE specification to include:
  - Additional priority levels for responses.
  - Priority aging for responses.
  - Waypoints in higher priority responses.
  - Detecting/terminating responses that haven't completed.
  - Multithreading.
- Develop methods of generating more than one effective test case from a single property.
- Investigate other model checkers to produce shorter test cases.

Future Work (cont'd)

California Institute of Technology

- FPE Requirements/User's Manual
  - Any use requires knowledge of SCR; implies formal specification/SCR training will be needed
  - Actual requirements spec will include
    - Precise verbal description of FPE and other components
    - Abstract description of queues
    - What Ids of requests actually are (rather than the placeholders that are currently used)
  - Critical properties that the FPE must satisfy (e.g., deadlock free, etc.)
    - Actual time-outs
    - Subresponses
  - Likely areas of change (none of these are captured in the current documentation)
    - Other classes of requests besides Ground, Non-Interrupting, and Interrupting
    - Queue Lengths