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
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

Monitors

 Symptoms

 Faults

 Responses

MonReset

Activate/Clear

Monitor 1

Monitor 2

Monitor n

Running

Idle

Idle

May 9, 2003
SCR Specification Overview (cont’d)

Fault Protection Engine
• 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 than 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)

- **Idle**: One or more requests received (no requests queued and none being processed)
- **Run_Resp_NoWP**: Current request is completed and no other requests queued or FlushAllResps received
- **FlushAllResps received**: Waypoint detected when higher-priority responses are queued
- **Run_Int_Resp**: Current request completed when no higher-priority requests queued and time-out not expired
- **Run_Resp_WP**: Current request is completed and at least one higher-priority request is queued

STD for FPE specification

SEHAS 2003

May 9, 2003
### SCR Specification Overview (cont’d)

<table>
<thead>
<tr>
<th>Begin State</th>
<th>Statechart</th>
<th>Transition Event</th>
<th>SCR Specification</th>
<th>Transition Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Run_Response</td>
<td>Received request to run a response OR there are one or more deferred responses</td>
<td>Idle</td>
<td>Run_Resp_NoWP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Received request to run a response OR there are one or more deferred responses</td>
</tr>
<tr>
<td>Run_Response, Run_Interrupting_Response</td>
<td>Idle</td>
<td>Received request to flush all responses</td>
<td>Run_Resp_NoWP, Run_Resp_WP, Run_Int Resp</td>
<td>Idle</td>
</tr>
<tr>
<td>Run_Response</td>
<td>Idle if stack is empty, Run_Response if stack is not empty</td>
<td>-</td>
<td>Run_Resp_NoWP</td>
<td>Idle</td>
</tr>
<tr>
<td>NoWayPoint</td>
<td>WayPoint</td>
<td>Waypoint encountered in response</td>
<td>Run_Resp_NoWP</td>
<td>Run_Resp_WP</td>
</tr>
</tbody>
</table>

### Comparison of SCR Specification to FPE statechart
### SCR Specification Overview (cont’d)

<table>
<thead>
<tr>
<th>Begin State</th>
<th>Statechart End State</th>
<th>Transition Event</th>
<th>SCR Specification Begin State</th>
<th>End State</th>
<th>Transition Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Run Resp_NoWP</td>
<td>Run_Int_Resp</td>
<td>Waypoint encountered in response AND there are one or more deferred interrupting or ground-requested responses.</td>
</tr>
<tr>
<td>Run_Interrupting_Response</td>
<td>WayPoint</td>
<td>Interrupting or ground requested response completes.</td>
<td>Run_Int_Resp</td>
<td>Run_Resp_WP</td>
<td>Interrupting or ground requested response completes prior to expiration of waypoint.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Run_Int_Resp</td>
<td>Run_Resp_NoWP</td>
<td>Waypoint expired prior to completing interrupting or ground requested response.</td>
</tr>
</tbody>
</table>

### Comparison of SCR Specification to FPE statechart (cont’d)
### SCR Specification Overview (cont’d)

<table>
<thead>
<tr>
<th>Begin State</th>
<th>Statechart</th>
<th>Transition Event</th>
<th>SCR Specification</th>
<th>Transition Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>WayPoint</td>
<td>NoWayPoint</td>
<td>Waypoint has expired.</td>
<td>Run_Resp_WP</td>
<td>Run_Resp_NoWP</td>
</tr>
<tr>
<td>WayPoint</td>
<td>Run_Interrupting_Response</td>
<td>Request for interrupting or ground requested response received OR there are one or more deferred interrupting or ground-requested responses.</td>
<td>Run_Resp_WP</td>
<td>Run_Int_Resp</td>
</tr>
</tbody>
</table>

### Comparison of SCR Specification to FPE statechart (cont’d)

SEHAS 2003

12

May 9, 2003
## SCR Specification Overview (cont’d)

### Monitored Variables

<table>
<thead>
<tr>
<th>Statechart</th>
<th>SCR Specification</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IntResp</td>
<td>mRespRequest</td>
<td>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.</td>
</tr>
<tr>
<td>NonIntResp</td>
<td>mRespRequest</td>
<td>See above</td>
</tr>
<tr>
<td>ReqResp</td>
<td>mRespRequest</td>
<td>See above</td>
</tr>
<tr>
<td>IsDone</td>
<td>MrespDone</td>
<td>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.</td>
</tr>
<tr>
<td>FlushAll</td>
<td>mFlushAllResps</td>
<td>A signal to the FPE to terminate the currently-executing response and cancel all deferred response requests.</td>
</tr>
</tbody>
</table>

### Monitored and Controlled Variables

SEHAS 2003

May 9, 2003
SCR Specification Overview (cont’d)

<table>
<thead>
<tr>
<th>Statechart</th>
<th>SCR Specification</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnterWayPoint</td>
<td>mWayPoint</td>
<td>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.</td>
</tr>
<tr>
<td>ExitWayPoint</td>
<td>mTimeOut</td>
<td>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.</td>
</tr>
</tbody>
</table>

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)

FPE SIMULATOR INTERFACE

INPUTS

- Request for Ground Response
  - ID
  - Type
- Completed Response
  - ID
  - Type
- Request for Interrupting Response
  - ID
- Request for Non-interrupting Response
  - ID
- WayPoint Entered
- TimeOut Expired
- Flush All Responses

OUTPUTS

- Command Response
  - ID
  - Type
- Error Messages

MODE AND OTHER AUXILIARY VARIABLES

- FPE Mode
  - ID
  - Type
- Currently Active Response
  - ID
- Saved Non-Inter. Response
  - ID
- Timed Out?

DEFERRED RESPONSE QUEUES

- Ground Responses Queue
  - Current Length
- Interrupting Responses Queue
  - Current Length
- Non-Inter. Responses Queue
  - Current Length

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

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

SEHAS 2003
Example: The Mode Transition Table From The SCR Spec Of The FPE

<table>
<thead>
<tr>
<th>Old Mode</th>
<th>Event</th>
<th>New Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>@C(mResp_Request) AND...</td>
<td>Run_Resp_NoWP</td>
</tr>
<tr>
<td>Run_Resp_NoWP</td>
<td>@T(mWayPoint) when tCurrentReqType=NR and tIRq_len=0 and tGRq_len=0</td>
<td>Run_Resp_WP</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Run_Resp_WP</td>
<td>@T(mFlushAllResps)</td>
<td>Idle</td>
</tr>
</tbody>
</table>

Table Defining the Value of FPEMode
Example: The Mode Transition Table From The SCR Spec Of The FPE (cont'd)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPEMode = Idle ∧ @C(mResp_Request)AND ...</td>
<td>→ FPEMode' = Run_Resp_NoWP</td>
</tr>
<tr>
<td>FPEMode = Run_Resp_NoWP ∧ @T(mWayPoint) when tCurrentReqType=NR &amp; tIRq_len=0 &amp; tGRq_len=0</td>
<td>→ FPEMode' = Run_Resp_WP</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>FPEMode = Run_Resp_WP ∧ @T(mFlushAllResps)</td>
<td>→ FPEMode' = Run_Resp_NoWP</td>
</tr>
<tr>
<td>(else)</td>
<td>→ FPEMode' = FPEMode</td>
</tr>
</tbody>
</table>

Total function that the table defines (single else clause)
Constructing Test Cases From A Mode Transition

Table (1)

Alternate Representation of the Function
with the else Clause Distributed

```plaintext
if FPEMode = Idle
  if @C(mResp_Request) & ...
    FPEMode' = Run_Resp_NoWP
  else
    FPEMode' = FPEMode
fi

FPEMode = Run_Resp_NoWP
if @C(mResp_Done) & ...
  FPEMode' = Idle
else
  FPEMode' = Run_Resp_WP
fi

FPEMode = Run_Resp_WP
if ...
  FPEMode' = ...
else
  FPEMode' = FPEMode
fi

FPEMode = Run_Int_Resp
if ...
  FPEMode' = ...
else
  FPEMode' = FPEMode
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:

$$FPE\text{Mode} = \text{Idle} \land @C(\text{mResp\_Request}) \land ...$$

$$\Rightarrow FPE\text{Mode}' = \text{Run\_Resp\_NoWP}$$
<table>
<thead>
<tr>
<th>Source Mode</th>
<th>Events</th>
<th>Destination Mode</th>
<th>Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>@C(mResp_Request) AND (mResp_Request' &gt; 0 AND ((tNR_ID' &gt; 0 AND tNR_ID' &lt;= MaxID) OR (tIR_ID' &gt; 0 AND tIR_ID' &lt;= MaxID) OR (tGR_ID' &gt; 0 AND tGR_ID' &lt;= MaxID))) ELSE</td>
<td>Run_Resp_NoWP</td>
<td>C1</td>
</tr>
<tr>
<td>Run_Resp_NoWP</td>
<td>@C(mResp_Done) AND (mResp_Done'=cResp_Request AND tNoReqsQd) OR @T(mFlushAllResps) ELSE</td>
<td>Idle</td>
<td>C1else</td>
</tr>
<tr>
<td>Run_Resp_NoWP</td>
<td>@T(mWayPoint) WHEN (tCurrent_Req_Type = NR AND tIRq_len = 0 AND tGRq_len = 0) ELSE</td>
<td>Run_Resp_NoWP</td>
<td>C2</td>
</tr>
<tr>
<td>Run_Resp_NoWP</td>
<td>@T(mWayPoint) WHEN (tCurrent_Req_Type = NR AND (tIRq_len &gt; 0 OR tGRq_len &gt; 0)) ELSE</td>
<td>Run_Resp_NoWP</td>
<td>C3else</td>
</tr>
<tr>
<td>Run_Int_Resp</td>
<td>@C(mResp_Done) AND (mResp_Done'=cResp_Request AND tIRq_len = 0 AND tGRq_len = 0 AND tTimeOut=false) ELSE</td>
<td>Run_Resp_NoWP</td>
<td>C4</td>
</tr>
</tbody>
</table>

Correspondence Between Test Cases and Mode Transitions

SEHAS 2003

May 9, 2003
### Test Cases Overview (cont’d)

<table>
<thead>
<tr>
<th>Source Mode</th>
<th>Events</th>
<th>Destination Mode</th>
<th>Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run_Int_Resp</td>
<td>@C(mResp_Done) AND (mResp_Done' = cResp_Request AND tTimeOut=true) ELSE</td>
<td>Run_Resp_NoWP</td>
<td>C6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Run_Int_Resp</td>
<td>C6else</td>
</tr>
<tr>
<td>Run_Int_Resp</td>
<td>@T(mFlushAllResps) ELSE</td>
<td>Idle</td>
<td>C7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Run_Int_Resp</td>
<td>C7else</td>
</tr>
<tr>
<td>Run_Resp_WP</td>
<td>@C(mTimeout) WHEN (tIRq_len = 0 AND tGRq_len = 0) ELSE</td>
<td>Run_Resp_NoWP</td>
<td>C8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Run_Resp_WP</td>
<td>C8else</td>
</tr>
<tr>
<td>Run_Resp_WP</td>
<td>@C(mResp_Request) AND ((tGR_ID' != tGR_ID AND tGR_ID' &gt; 0) OR (tIR_ID' != tIR_ID AND tIR_ID' &gt; 0)) ELSE</td>
<td>Run_Int_Resp</td>
<td>C9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Run_Resp_WP</td>
<td>C9else</td>
</tr>
<tr>
<td>Run_Resp_WP</td>
<td>@T(mFlushAllResps) ELSE</td>
<td>Idle</td>
<td>C10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Run_Resp_WP</td>
<td>C10else</td>
</tr>
</tbody>
</table>

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

SEHAS 2003

24

May 9, 2003
## Individual Test Cases

<table>
<thead>
<tr>
<th>C1</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>mResp_Request 1</td>
<td>cResp_Request 1</td>
</tr>
<tr>
<td>mResp_Request 1</td>
<td>cResp_Request 1</td>
</tr>
<tr>
<td>mResp_Done 1</td>
<td>cResp_Request 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C2</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>mResp_Request 1</td>
<td>cResp_Request 1</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C3</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>mFlushAllResps TRUE</td>
<td>⇐</td>
</tr>
<tr>
<td>mResp_Request 1</td>
<td>cResp_Request 1</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C4</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>mResp_Request 1</td>
<td>cResp_Request 1</td>
</tr>
<tr>
<td>mResp_Request 10</td>
<td>⇐</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C5</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>mResp_Request 1</td>
<td>cResp_Request 1</td>
</tr>
<tr>
<td>mResp_Request 21</td>
<td>⇐</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C6</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>mResp_Request 2</td>
<td>cResp_Request 2</td>
</tr>
<tr>
<td>mResp_Request 4</td>
<td>⇐</td>
</tr>
<tr>
<td>mResp_Request 11</td>
<td>⇐</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 10</td>
</tr>
<tr>
<td>mTimeOut TRUE</td>
<td>⇐</td>
</tr>
<tr>
<td>mResp_Done 10</td>
<td>cResp_Request 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C7</th>
<th>C7</th>
</tr>
</thead>
<tbody>
<tr>
<td>mResp_Request 9</td>
<td>cErrMsgBadID = ID_Out_of_Range</td>
</tr>
<tr>
<td>mResp_Request 3</td>
<td>cErrMsgBadID = null</td>
</tr>
<tr>
<td>mResp_Request 13</td>
<td>⇐</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 10</td>
</tr>
<tr>
<td>mFlushAllResps TRUE</td>
<td>cResp_Request 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C8</th>
<th>C8</th>
</tr>
</thead>
<tbody>
<tr>
<td>mFlushAllResps TRUE</td>
<td>⇐</td>
</tr>
<tr>
<td>mResp_Request 1</td>
<td>cResp_Request 1</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 0</td>
</tr>
<tr>
<td>mTimeOut TRUE</td>
<td>cResp_Request 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C9</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td>mFlushAllResps TRUE</td>
<td>⇐</td>
</tr>
<tr>
<td>mResp_Done 1</td>
<td>⇐</td>
</tr>
<tr>
<td>mResp_Request 3</td>
<td>cResp_Request 3</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 0</td>
</tr>
<tr>
<td>mResp_Request 10</td>
<td>cResp_Request 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C10</th>
<th>C10</th>
</tr>
</thead>
<tbody>
<tr>
<td>mResp_Request 4</td>
<td>cResp_Request 4</td>
</tr>
<tr>
<td>mWayPoint TRUE</td>
<td>cResp_Request 0</td>
</tr>
<tr>
<td>mFlushAllResps TRUE</td>
<td>⇐</td>
</tr>
</tbody>
</table>

**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)
Individual Test Cases (cont’d)

-----C1else---- mFlushAllResps TRUE Eliminate -- mResp_Request 2
mResp_Request 1 cResp_Request 1
mResp_Request 2

-----C2else---- mResp_Request 1 cResp_Request 1

-----C3else---- mResp_Request 1 cResp_Request 1
mResp_Request 3

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

-----C5else---- mResp_Request 4 cResp_Request 4
mWayPoint TRUE mResp_Request 10
mResp_Request 500

-----C6else---- cResp_Request 2
mResp_Request 2
mResp_Request 4
mResp_Request 11
mWayPoint TRUE
mTimeOut TRUE

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

-----C8else---- mResp_Request 1 cResp_Request 1
mWayPoint TRUE mResp_Request 3

-----C9else---- mResp_Request 0 cResp_Request 0
mWayPoint TRUE mResp_Request 1
mResp_Request 10
mFlushAllResps TRUE

-----C10else---- mResp_Request 1 cResp_Request 1
mResp_Request 2

NOTES
- Test cases C1 else and C6 else may be eliminated because they are contained in test cases C9 and C6, respectively.

SEHAS 2003 26 May 9, 2003
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

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)

- 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)

- 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