Critical

Robustness and Stress Testing RTEMS 4.5.0

Final Presentation

Dependable Technologies for Critical Systems

Critical

Study Contract

Safety and Dependability Evaluations (RAMS)

ESTEC/Contract Nº 16582/02/NL/PA

Call-Off Order 02:

Robustness and Stress Testing RTEMS 4.5.0

esa — **Maria Hernek**

Critical — **Diamantino Costa, Ricardo Maia**

2

Presentation Outline

- Introduction
  - Why this study?
  - Study objectives
- Work Plan
  - Study logic
  - Technical Deliverables
- Study Activities
  - Robustness Testing
  - Stress Testing
- Conclusions

3

Why this study?

- Space missions have been lost due to software failures
- Software in Space systems is more and more used in critical functions
- Software dependability and safety needs careful analysis
- Software RAMS requirements are needed in addition to system RAMS

4

Why this study?

- PASCON WO12
  - Identified and analysed typical RAMS requirements applicable to space software
  - Identified RAMS techniques, methods and procedures applicable during software development
  - Provided introductory support on the application of such techniques, methods and procedures

5

Why this study?

- ESA's Initiative on Software Dependability and Safety Evaluations
  - Evaluate dependability and safety of software to reduce the risk of mission failures
  - Help assuring correct implementation of system PA requirements
  - Evaluate the applicability and usefulness of selected techniques
  - Identify improvements on the techniques

6

Study objective

- Assess the applicability and usefulness of:
  - Robustness Testing
  - Stress Testing
- Case Study: RTEMS 4.5.0

7

Study outcome

- Outcome
  - List of problems found during robustness testing
  - List of problems found during stress testing
  - Metrics on the effectiveness and applicability of the methods and techniques
  - Possible improvements of the methods and techniques

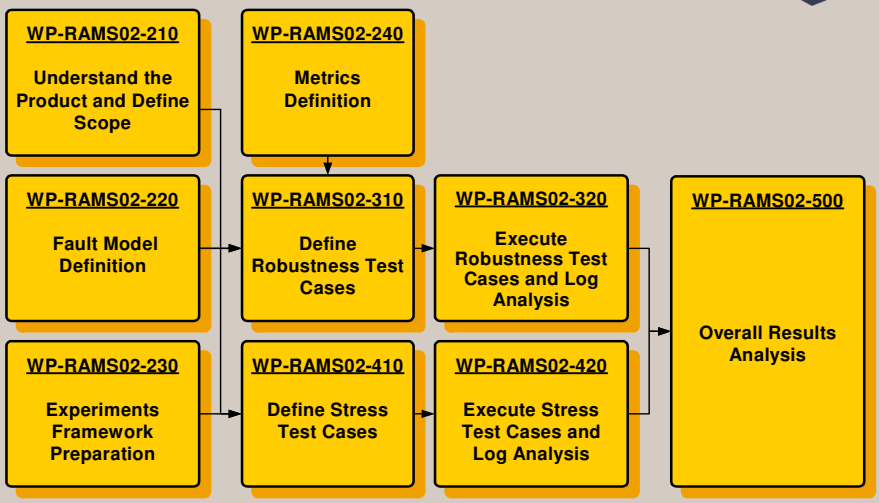
8

Presentation Outline

- Introduction
  - Why this study?
  - Study objectives
- Work Plan
  - Study logic
  - Technical Deliverables
- Study Activities
  - Robustness Testing
  - Stress Testing
- Conclusions

9

Study Logic



10

Technical Deliverables (I)

Deliverable	Document title	Work package	Version	Date
DL-RAMS02-01	Evaluation Report:		2.0	07/11/03
	RTEMS 4.5.0 Description and Scope Definition	**WP-210**		
	Fault Model	**WP-220**		
	Experiment Framework Description	**WP-230**		
	Metrics Definition	**WP-240**		
	Problems Found and Potential Improvements	**WP-500**		

11

Technical Deliverables (II)

Deliverable	Document title	Work package	Version	Date
DL-RAMS02-02	Robustness Testing Report:		1.2	17/09/2003
	Test Cases Definition	**WP-310**		
	Test Results	**WP-320**		
DL-RAMS02-03	Robustness Testing Workloads	WP-310	1.0	17/09/2003
DL-RAMS02-04	Stress Testing Report:		2.0	30/10/2003
	Test Cases Definition	**WP-410**		
	Test Results	**WP-420**		
DL-RAMS02-05	Stress Testing Workloads	WP-410	1.0	30/10/2003

12

Presentation Outline

- Introduction
  - Why this study?
  - Study objectives
- Work Plan
  - Study logic
  - Technical Deliverables
- Study Activities
  - Robustness Testing
  - Stress Testing
- Conclusions

13

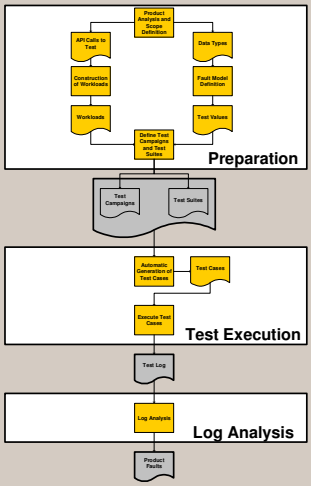
Robustness Testing >> Goal

- "To test RTEMS services against non-nominal parameters exercising consistency checking and error handling mechanisms"

14

Robustness Testing >> Methodology Overview

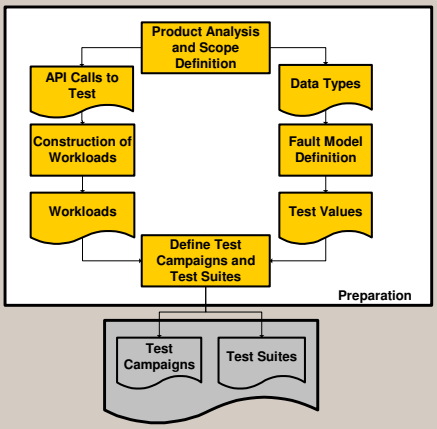
- It comprises 3 phases:
  - Preparation
  - Test Execution
  - Log Analysis



15

Robustness Testing >> Preparation

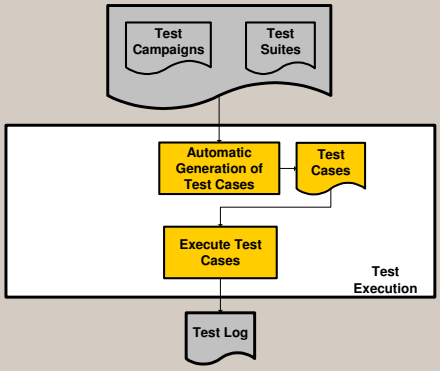
- Analyze the product and selection of the API calls to test
- Definition of the values to use for each data type
- Implementation of applications to exercise the product
- Definition of the test suites to be used for automatic test cases generation



16

Robustness Testing >> Test Execution

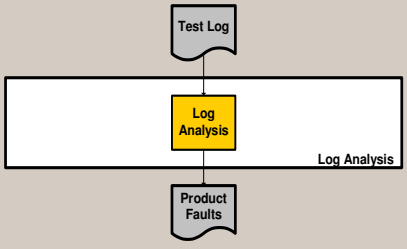
- Automatically generate test case from the test suites
- Execute the test cases and collect the output in a database



17

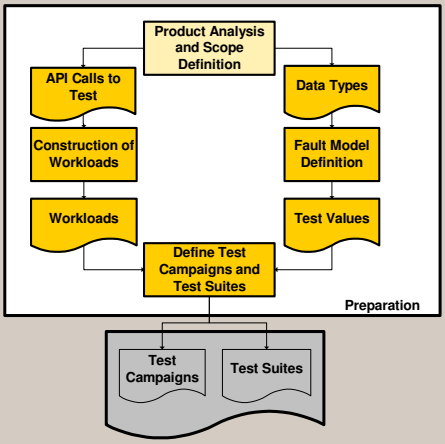
Robustness Testing >> Log Analysis

- Compare the obtained results against the expected values
- Further analysis may be required (e.g. execution in debug mode, analysis of the source code, etc.)



18

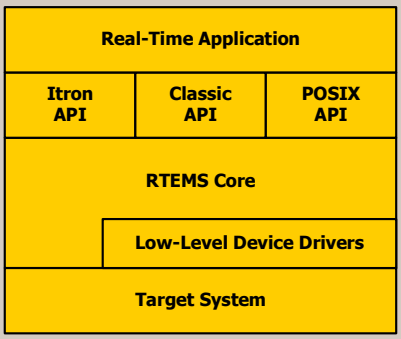
Robustness Testing >> Preparation: Product Analysis and Scope Definition



19

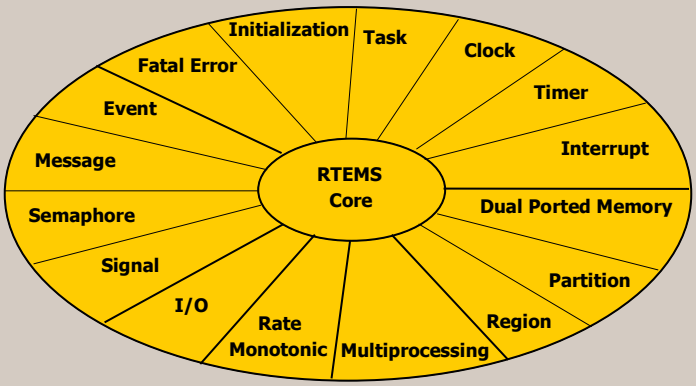
Robustness Testing >> Preparation: Product Analysis – RTEMS Architecture

- RTEMS 4.5.0



20

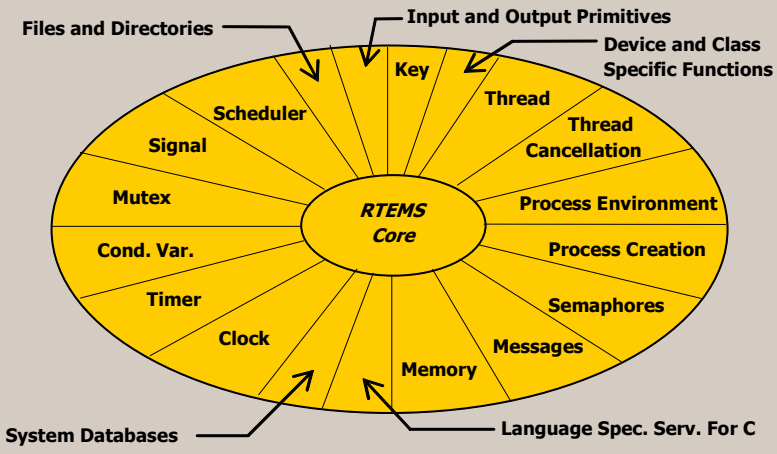
Robustness Testing >> Preparation: Product Analysis – Classic API



- ~100 Directives (e.g. System Calls)

21

Robustness Testing >> Preparation: Product Analysis – POSIX API



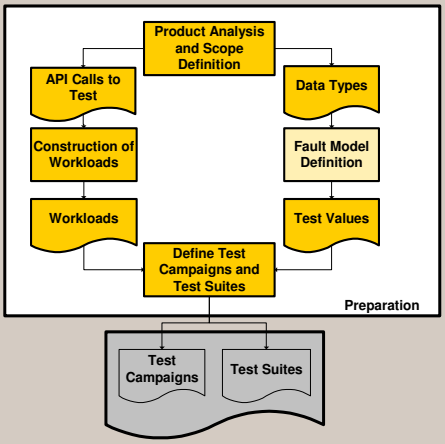
22

Robustness Testing >> Preparation: Scope Definition

- Classical API
- POSIX API
- Executive Core
- Interface with the Low-Level Device Drivers

23

Robustness Testing >> Preparation: Fault Model Definition



24

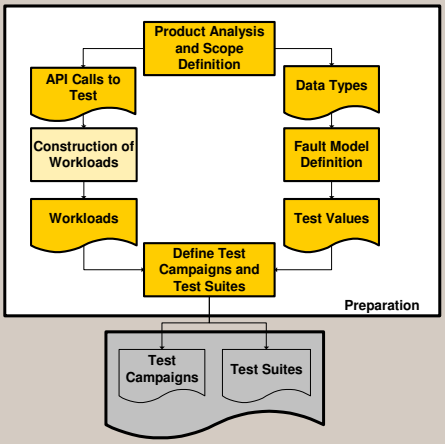
Robustness Testing >> Preparation: Fault Model Definition

- Define test values for basic types according previous experiences
- Analyse each RTEMS data type
- Define test values for each RTEMS data type.

Type Name	Test Values
char	0, 255
signed char	0, -128, 127
int	0, 1, -1, 2147483647, -2147483648
unsigned int	0, 1, 4294967295
short int	0, 1, -1, 32767, -32768
unsigned short int	0, 1, 65535
long	0, 1, -1, 9223372036854775807, -9223372036854775808
unsigned long	0, 1, 18446744073709551615
pointers	NULL

25

Robustness Testing >> Preparation: Construction of the Workloads



26

Robustness Testing >> Preparation: Construction of the Workloads

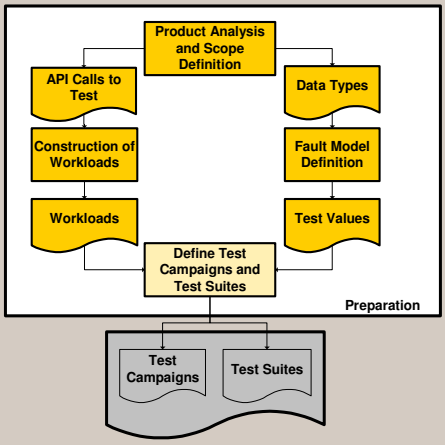
- One workload for each manager
- Calls to every directive under test
- Synthetic output to ease the result analysis

<Function Name>(): <Return Value>; <Assertion Name>: {success|failure};

API	Workloads
Classic	14
POSIX	5
Total	**19**

27

Robustness Testing >> Preparation: Define test campaigns and test suites



28

Robustness Testing >> Preparation: Test Campaigns Definition

- One Campaign for each RTEMS resource manager

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-RGN
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the region manager.
Workload File:	rtems-cmp-cl-rgn.c
Test Suites:	1.RTEMS-TS-CL-RGNCRT 2.RTEMS-TS-CL-RGNGSG 3.RTEMS-TS-CL-RGNGSS
Workload Description: This workload only has one main task. This task performs all tests of the region manager . It executes the following region manager related operations: •Create a region; •Get a segment from region; •Return the segment to region; •Extend region; •Delete region.	

29

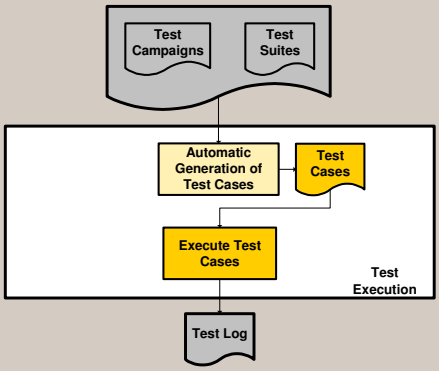
Robustness Testing >> Preparation: Test Suites Definition

- One Test Suite for each directive

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNGSG
Purpose:	To test rtems_region_get_segment by invoking it with the entire range of test values for each of its parameters.
Injection Location(s):	Source file: rtems-cmp-cl-rgn.c Lines: [155 - 159] `returnStatus = rtems_region_get_segment (regionId, requestedSize1, option, timeout, ptsegment1);`
Test Item:	rtems_region_get_segment (rtems_id *id, rtems_unsigned32 size, rtems_option option_set, rtems_interval timeout, void **segment)
Generated Test Cases:	17

30

Robustness Testing >> Test Execution: Automatic Test Cases Generation



31

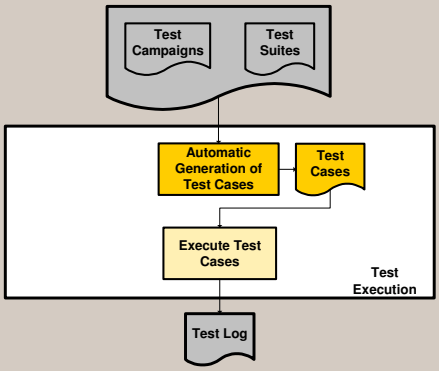
Robustness Testing >> Test Execution: Automatic Test Cases Generation

```
rtems_region_get_segment (
        rtems_id *id,
        rtems_unsigned32 size,
        rtems_option option_set,
        rtems_interval timeout,
        void **segment)
```

Type Name	Test Values
rtems_unsigned16	0, 1, 65535
rtems_unsigned32	0, 1, 4294967295
rtems_unsigned8	0, 1, 255
rtems_vector_number	0, 1, 4294967295

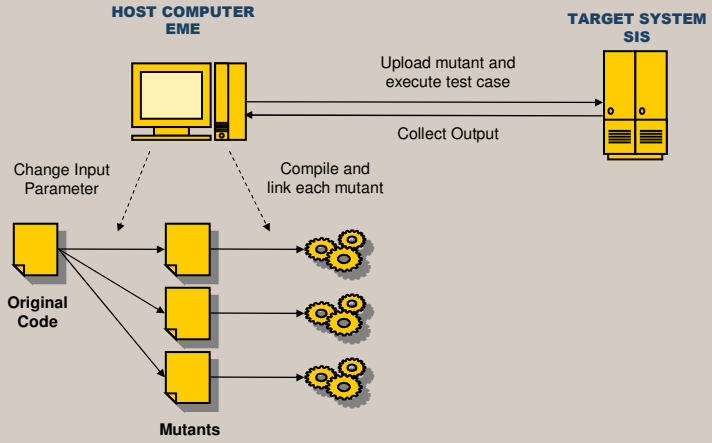
```
requestedSize1 = 4294967295;
     returnStatus = rtems_region_get_segment (
                                        regionId,
                                        requestedSize1,
                                        option,
                                        timeout,
                                        ptsegment1);
```

Robustness Testing >> Test Execution: Execute Test Cases



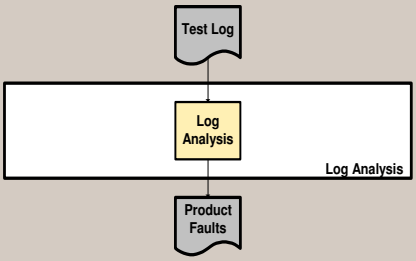
33

Robustness Testing >> Test Execution: Execute Test Cases



34

Robustness Testing >> Log Analysis: Log Analysis



35

Robustness Testing >> Log Analysis: Log Analysis

- Only test cases which uncover faults are shown

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNGSG-022 (same results obtained in RTEMS-TCR-CL-RGNGSG-024)
Input Specification:	

`requestedSize1 = 0;`
`returnStatus = rtems_region_get_segment (regionId,`
`requestedSize1,`
`option,`
`timeout,`
`ptsegment1);`	

Failure Description:	
A Memory Exception occurs while attempting to retrieve a segment of size zero. The same happens when attempting to retrieve a segment of size 4294967295.	
Notes:	

The simulator returns the following output:
`Memory exception at fffffffc (illegal address)`
`Unexpected trap (0x09) at address 0x0200aaac`
`Data access exception at 0xfffffffc`	

36

Robustness Testing >> Results Summary: Overall Results Summary

- A total of 1055 test cases were defined and executed
- A total of 49 test cases failed

API	Test Cases	Test Cases Failed
Classic	527	34
POSIX	528	15
Total	**1055**	**49**

37

Robustness Testing >> Results Summary: Classic API (I)

Manager	Test Cases	Test Cases Failed	Test cases Failed / Total Test Cases
Clock	68	0	0%
Event	18	0	0%
Fatal Error	3	0	0%
Interrupt	5	0	0%
IO	50	6	12%
Message	83	8	10%
Partition	27	2	7%
Rate Monotonic	24	1	4%
Region	67	7	10%
Semaphore	33	1	3%
Signal	10	1	10%
Task	55	4	7%
Timer	67	3	4%
User Extensions	17	0	0%
Total	**527**	**33**	**6%**

38

Robustness Testing >> Results Summary: Classic API (II)

Manager	Critical	Low	Total
Clock	0	0	**0**
Event	0	0	**0**
Fatal Error	0	0	**0**
Interrupt	0	0	**0**
IO	5	1	**6**
Message	2	6	**8**
Partition	0	2	**2**
Rate Monotonic	0	1	**1**
Region	4	3	**7**
Semaphore	0	1	**1**
Signal	0	1	**1**
Task	2	2	**4**
Timer	2	1	**3**
User Extensions	0	1	**1**
Total	**15**	**19**	**34**

39

Robustness Testing >> Result Summary: POSIX API (I)

Manager	Test Cases	Test Cases Failed	Test Cases Failed / Total Test Cases
Clock	32	0	0%
Message	122	3	2%
Mutex	223	4	2%
Signal	122	5	4%
Timer	29	3	10%
Total	**528**	**15**	**3%**

40

Robustness Testing >> Result Summary: POSIX API (II)

Manager	Critical	Low	Total
Clock	0	0	0
Message	2	1	3
Mutex	1	3	4
Signal	1	4	5
Timer	0	3	3
Total	**4**	**11**	**15**

41

Robustness Testing >> Metrics – Error Handling Coverage

- There was no tool available for collecting measures on code coverage (gcov was not ported to RTEMS 4.5.0)
- Code coverage was obtained manually (very time consuming)
- Code coverage was computed concerning the the robustness testing of only one Manager

Module	LOC	Total Error Handling Code (LOC)	Executed Error Handling Code (LOC)	Error Handling Coverage
Message Manager	959	56	49	~75%
Message Manager + CORE	3046	114	63	~55%

42

Presentation Outline

- Introduction
  - Why this study?
  - Study objectives
- Work Plan
  - Study logic
  - Technical Deliverables
- Study Activities
  - Robustness Testing
  - Stress Testing
- Conclusions

43

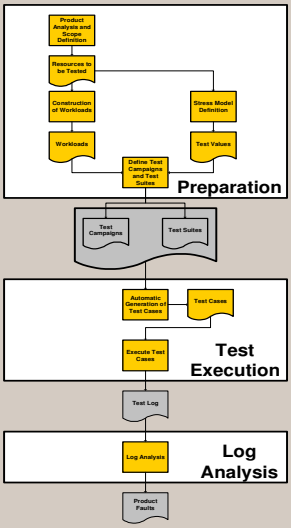
Stress Testing >> Goal

- "Test RTEMS to extreme conditions (e.g. exceptional high workload) to identify vulnerable points or to trigger rarely executed code such as error handling and recovery"

44

Stress Testing >> Methodology Overview

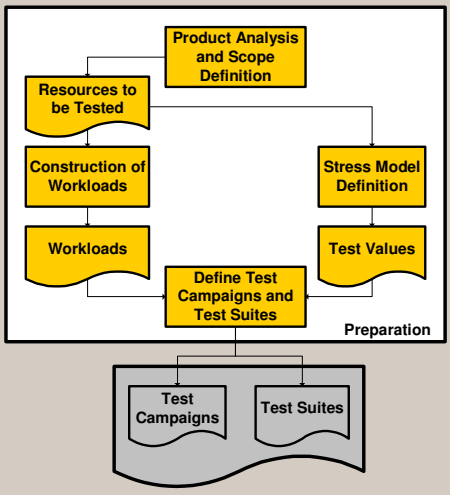
- It comprises 3 phases:
  - Preparation
  - Test Execution
  - Log Analysis



45

Stress Testing >> Preparation

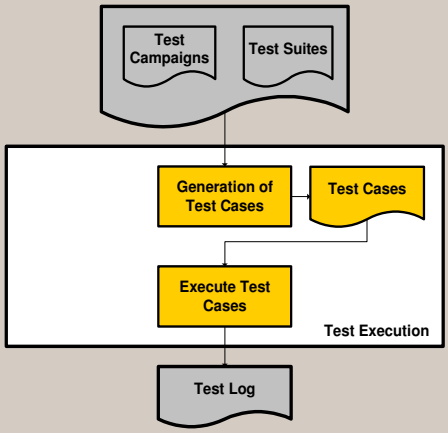
- Analyze the product and selection of the resources to be stressed
- Definition of the desired level of resource usage
- Implementation of applications to stress the system resources
- Definition of the test suites to be used for test cases generation



46

Stress Testing >> Test Execution

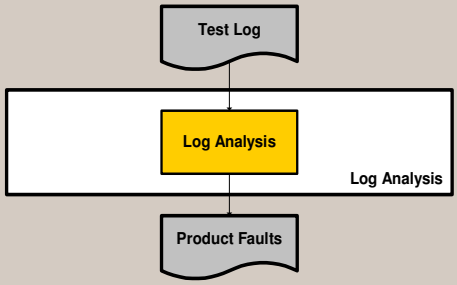
- Generate test cases from the test suites
- Execute the test cases and collect the output in a database



47

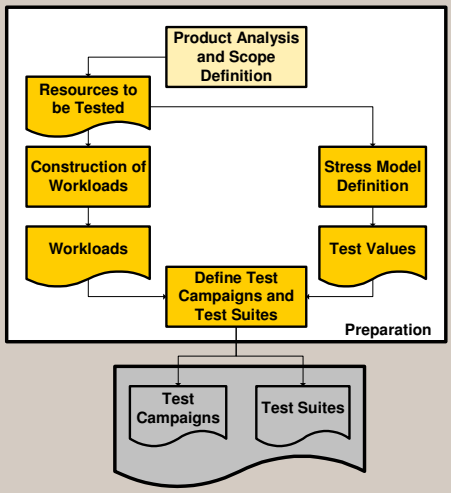
Stress Testing >> Log Analysis

- Compare the obtained results against the expected values
- Further analysis may be required (e.g. execution in debug mode, analysis of the source code, etc.)



48

Stress Testing >> Preparation: Product Analysis and Scope Definition



49

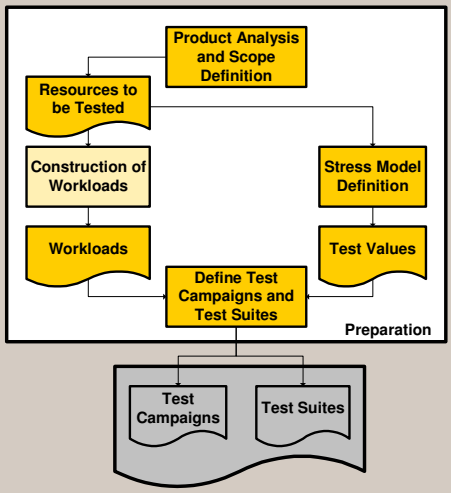
Stress Testing >> Preparation: Product Analysis and Scope Definition

- Types Resources to be Stressed:
  - Physical resources
  - Logical resources

Physical	Logical
CPU Time	Tasks
Memory	Semaphores
	Events
	Interrupts
	Signals
	Message Queues
	Partitions

50

Stress Testing >> Preparation: Construction of Workloads



51

Stress Testing >> Preparation: Construction of Workloads

- One workload for each RTEMS resource Manager

API	Workloads
Classic	7

- Workloads implement a scalable producer/consumer algorithm that uses the selected manager resources.
- Degree of load of an RTEMS resource manager can be adjusted by several parameters.

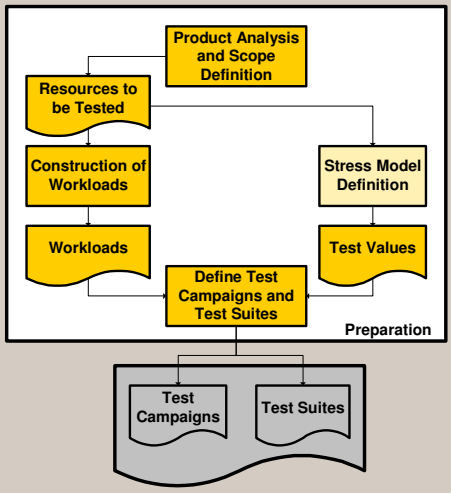
52

Stress Testing >> Preparation: Construction of Workloads

Data Type	Parameter Name	Description
rtems_unsigned32	NUMBER_OF_PRODUCERS	Number of producer tasks
rtems_unsigned32	PRODUCERS_TASK_STACK_SIZE	Producers tasks stack size
rtems_task_priority	PRODUCERS_PRIORITY	Producers priority
rtems_mode	PRODUCERS_TASK_MODE	Producers tasks mode
rtems_attribute	PRODUCERS_TASK_ATTR	Producers tasks attributes
rtems_unsigned32	NUMBER_OF_CONSUMERS	Number of consumers tasks
rtems_unsigned32	CONSUMERS_TASK_STACK_SIZE	Consumers tasks stack size
rtems_mode	CONSUMERS_TASK_MODE	Consumers tasks mode
rtems_attribute	CONSUMERS_TASK_ATTR	Consumers tasks attributes
rtems_task_priority	CONSUMERS_PRIORITY	Consumers priority
rtems_unsigned32	NUMBER_OF_SYSTEMS	Number of systems (producers/consumers systems)
rtems_interval	TEST_TIMEOUT	Number of ticks to wait until workload finishes

53

Stress Testing >> Preparation: Stress Model Definition



54

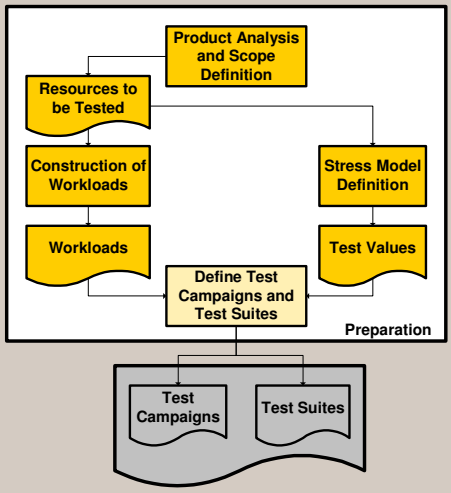
Stress Testing >> Preparation: Stress Model Definition

- Non-linear distribution of test values, in a way that most of the values are small values and only a small percentage of them are large values.

Test Stress Values
4
16
256
65536

55

Stress Testing >> Preparation: Construction of Workloads



56

Stress Testing >> Preparation: Test Campaigns Definition

- One Campaign for each RTEMS resource manager

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-TSK
Purpose:	To stress test the task manager.
Workload Files:	rtems-cmp-cl-tsk.c, rtems-cmp-cl-tsk.h, tsk-producer.c, tsk-consumer.c
Test Suites:	1.RTEMS-TS-CL-TSK-001 2.RTEMS-TS-CL-TSK-002
Workload Description: This workload will create a semaphore for each system. The consumers of a specific system will all wait in the system semaphore, while the producers will periodically signal that semaphore to unblock the consumers. The stressing will consist on the modification of the number of threads created (consumers and producers), using two different priorities: an higher priority for the consumers and a lower priority for the producers.	

57

Stress Testing >> Preparation: Test Suites Definition

- One Test Suite for set of Parameters

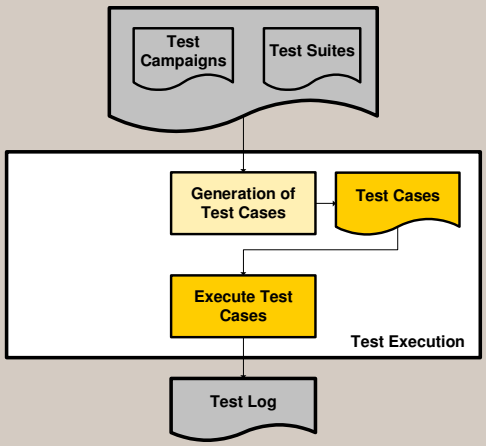
Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TSK-001
Purpose:	To stress the task manager by creating as much consumers and producers as possible.

Fault Location(s):	Source file: rtems-cmp-cl.tsk.h
	Lines: [50, 65]
	`#define NUMBER_OF_PRODUCERS 64`
	`#define NUMBER_OF_CONSUMERS 64`

Generated Test Cases:	16

58

Stress Testing >> Test Execution: Test Cases Generation



59

Stress Testing >> Test Execution: Test Cases Generation

```
/* Number of producer tasks */
#define NUMBER_OF_PRODUCERS 64

/* Producers priority */
#define PRODUCERS_PRIORITY 3

[ . . . ]
```

Test Stress Values
4
16
256
65536

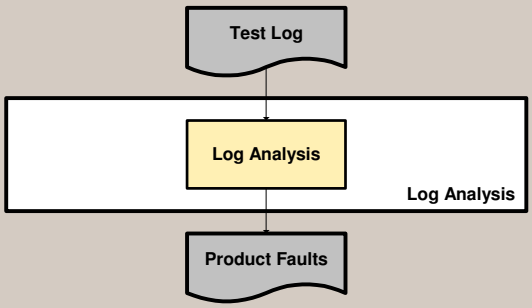
```
/* Number of producer tasks */
#define NUMBER_OF_PRODUCERS 65536

/* Producers priority */
#define PRODUCERS_PRIORITY 3

[ . . . ]
```

60

Stress Testing >> Log Analysis: Log Analysis



61

Stress Testing >> Log Analysis: Log Analysis

- Only test cases which uncover faults are shown

Test Case Results	
Test case result identifier:	RTEMS-TCR-CL-TSK-001-004 (same results obtained in RTEMS-TCR-CL-TSK-001-008, RTEMS-TCR-CL-TSK-001-010, RTEMS-TCR-CL-TSK-001-011, RTEMS-TCR-CL-TSK-001-012)
Input Specification:	
`#define NUMBER_OF_PRODUCERS 256` `#define NUMBER_OF_CONSUMERS 4`	
Failure Description:	
Not enough RAM to execute the application. The simulator returned the following output: `bspstart: Not enough RAM!!!`	
Notes:	

62

Stress Testing >> Results Summary (I)

- A total of 452 test cases were defined and executed

RTEMS Manager	Number of Test Cases
Task Manager	20
Semaphore Manager	68
Message Manager	80
Signal Manager	68
Interrupt Manager	20
Event Manager	68
Partition Manager	128
Total	**452**

63

Stress Testing >> Results Summary (II)

- Three type of failures were identified:
  - Application linkage: linker fails to create a binary image saying that the region ram is full for a specified binary section (the .bss section).
  - RTEMS initialisation failure in two different ways:
    - after detecting that there was not enough RAM memory to initialise the application
    - after trying to initialise the application accessing an invalid (inexistent) memory address.
  - RTEMS objects creation: the application fails on the creation of the specified resources due to lack of memory.

64

Stress Testing >> Results Summary (III)

- A total of 378 test cases failed

RTEMS Manager	Application Linkage	RTEMS Initialisation	RTEMS Objects Initialisation	Total
Task Manager	0	16	0	16
Semaphore Manager	19	32	0	51
Message Manager	19	37	18	74
Signal Manager	31	25	0	56
Interrupt Manager	3	11	0	14
Event Manager	31	25	0	56
Partition Manager	19	40	52	111
Total	**122**	**186**	**74**	**378**

65

Stress Testing >> Results Summary (IV)

- A high number of were related with the same problem: an invalid memory access during the RTEMS initialisation
- A *Critical* criticality was assigned to this problem

RTEMS Manager	Critical	Minor	Total
All	1	0	1
Partition Manager	0	1	1
Total	**1**	**1**	**2**

66

Presentation Outline

- Introduction
  - Why this study?
  - Study objectives
- Work Plan
  - Study logic
  - Technical Deliverables
- Study Activities
  - Robustness Testing
  - Stress Testing
- Conclusions

67

Conclusions >> RTEMS Problems Found – Classic API

- Unexpected Change of the Control Flow
  - Unexpected change of the control flow of the application occurred
- Data Access / Memory not Aligned / Illegal Instruction Exceptions
  - Test cases ended up with unhandled traps.
- No Error Code
  - Although providing an invalid parameter no error code was returned.
- Wrong Error Code
  - RTEMS return an error code different from the expected (e.g. specified by the documentation)

68

Conclusions >> RTEMS Problems Found – POSIX API

- POSIX Compliance
  - The behaviour of the RTEMS POSIX API differs from the specified by the POSIX 1003.1 Standard
- Kernel Crash
  - The application terminates an the kernel outputs an "assertion failed" message

69

Conclusions >> Robustness Testing Methodology

- Definition and execution of a large number of test cases with a reduced effort
- A considerable number of robustness problems were uncovered given the effort spent.

API	Test Cases	Faults	Effort (hours)	Effort per Test Case (hours)	Effort per Fault (hours)
Classic	527	34	160	0,30	4,7
POSIX	528	16	80	0,15	5
Total	**1055**	**50**	**240**	**0,23**	**4,8**

- High Coverage of the Error Handling Code was achieved:
  - ~75% for the API error handling code
  - ~55% for total error handling code

70

Conclusions >> Robustness Testing Methodology

- Very valuable when used to evaluate the degree of "protection" of an API.
- Exercises the Error Handling code
- Very straightforward concerning the robustness testing of system calls or libraries.
- Analysis of the output is still manual

71

Conclusions >> Stress Testing Methodology

- Definition and execution of a large number of test cases with a reduced effort
- Considering the number of faults uncovered results achieved by the stress testing were not as promising as the results of the robustness testing
- Stress testing would also benefit from the knowledge of internal RTEMS architecture. This knowledge could be used to identify sensitive resources of the kernel and to figure out ways of stressing them.

72

Conclusions >> Stress Testing Methodology

- Focus on the complete system not only on the API
- Exercises the Error Handling code
- Effectiveness must be improved
- Analysis of the output is still manual

73