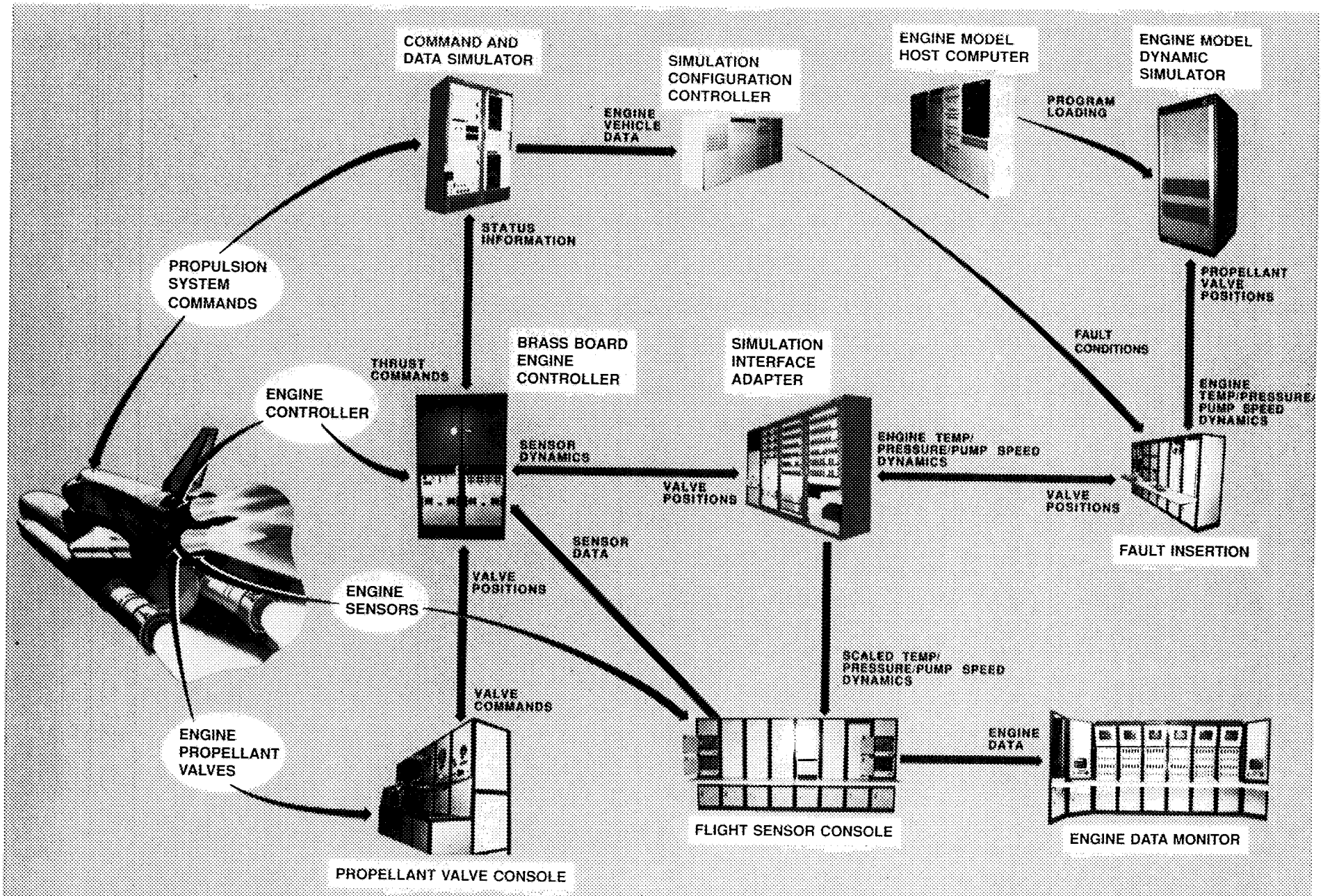
Automated Test Environment for a Real-time Control System

Agenda

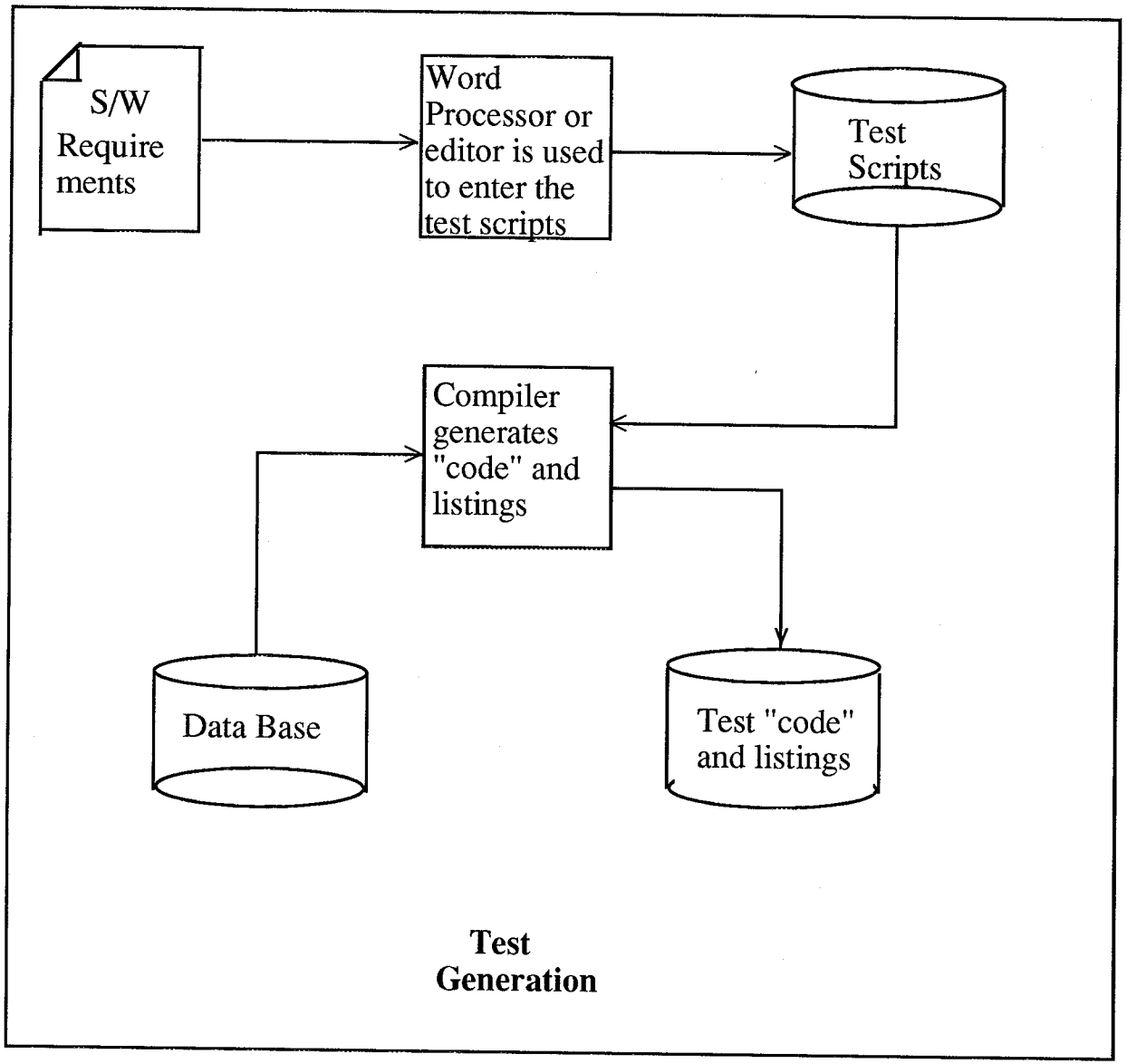
- Problem Domain
- Test Environment
- Test Process
- Advantages of Automation
- Conclusions

Problem Domain

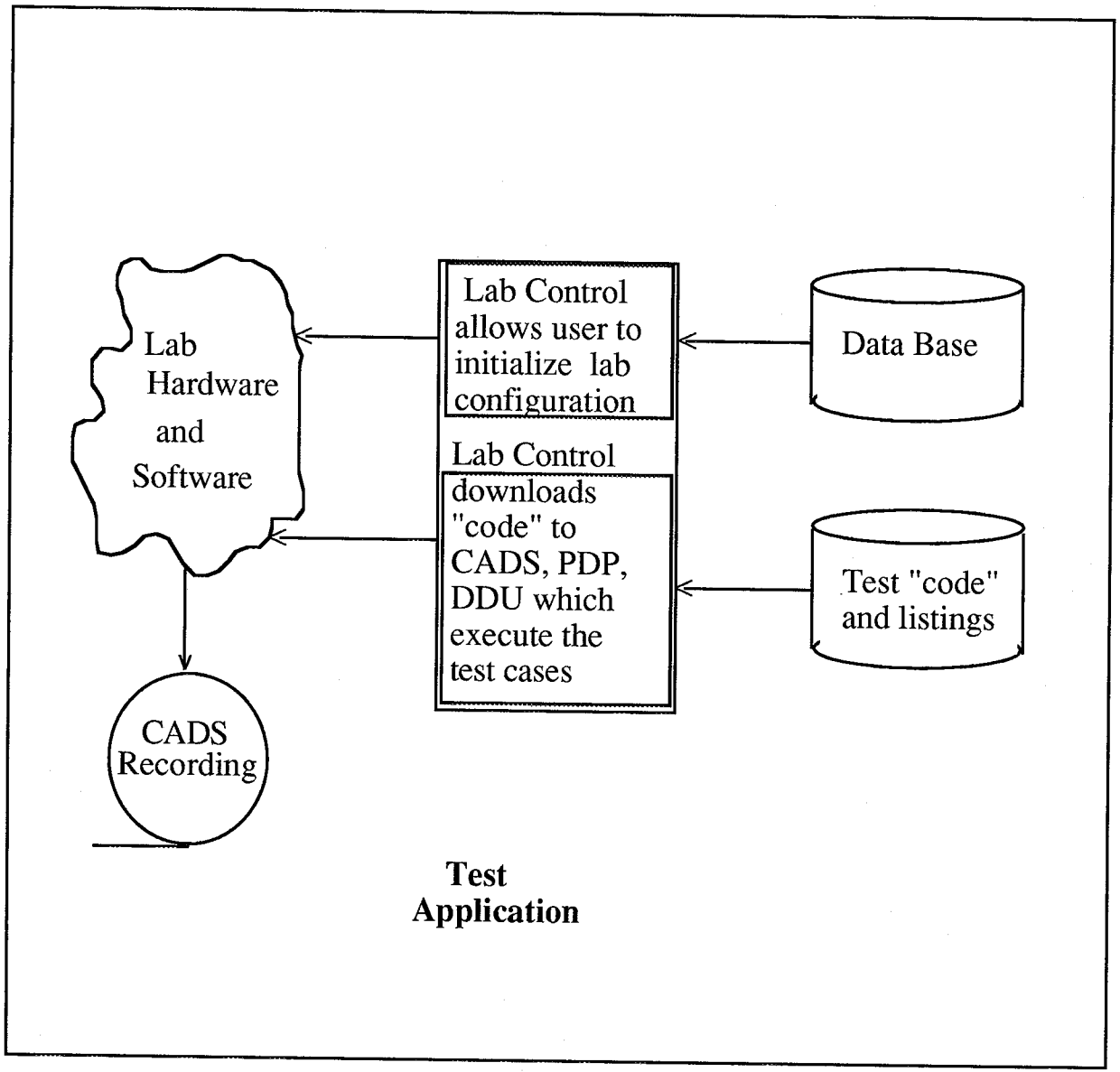
- Space Shuttle Main Engine Control System
- Simulation Requirements
- Test Suite
- Automation



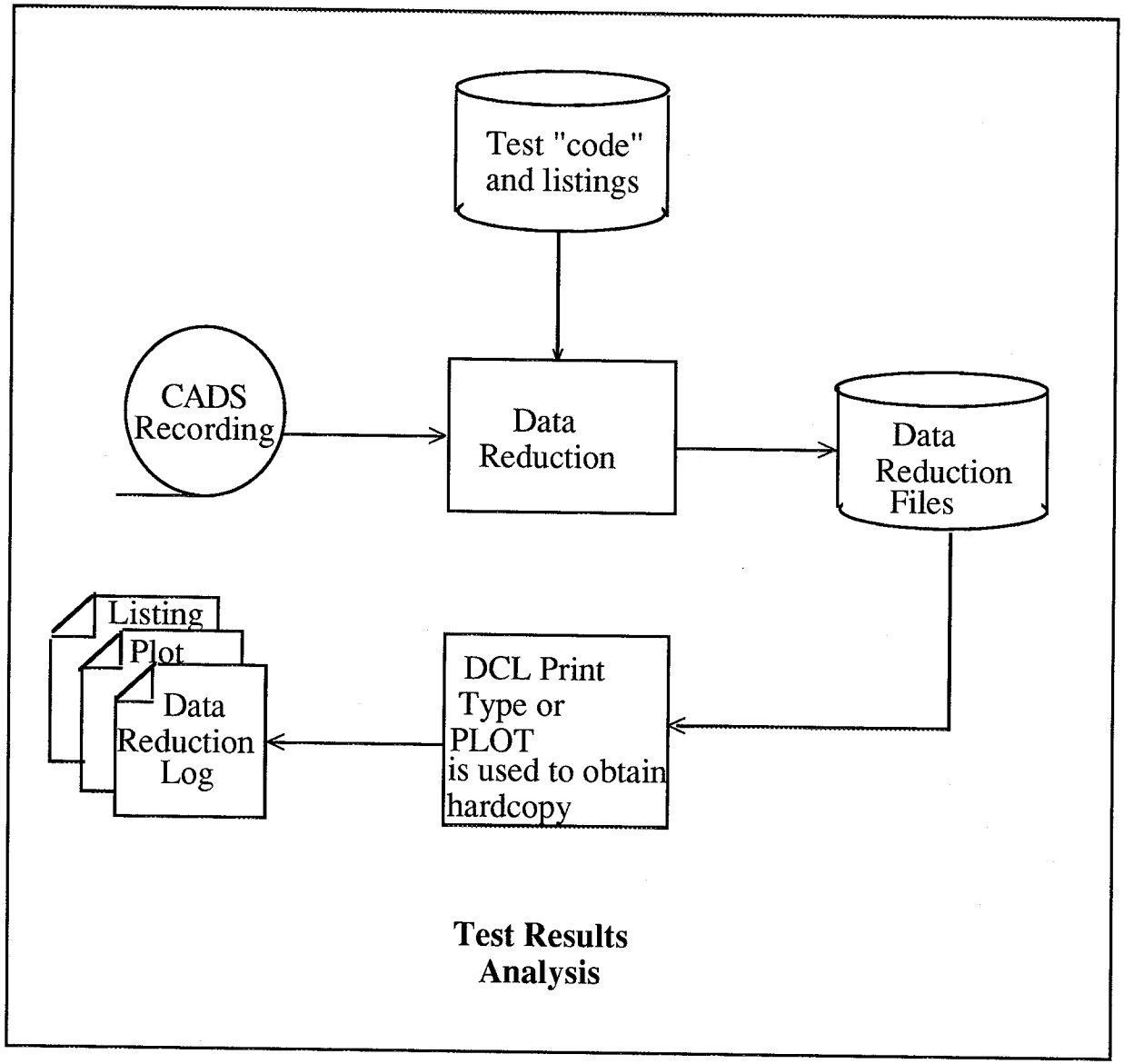
SPACE SHUTTLE MAIN ENGINE HARDWARE SIMULATION LABORATORY



Test Generation



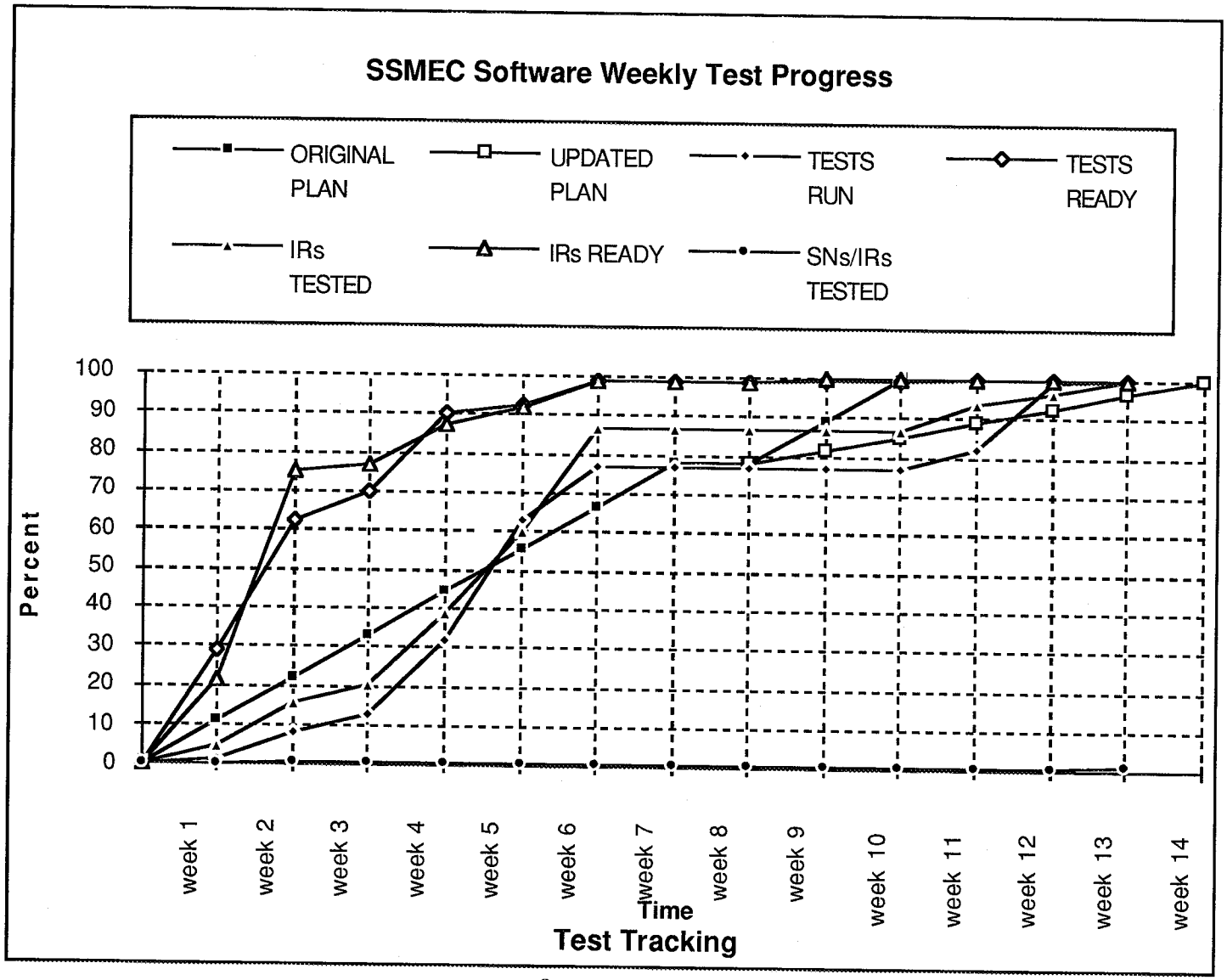
Test Application



Test Results Analysis

Advantages of Automation

- Test Planning
- Test Performance
- Test Tracking
- Test Management


SSMEC Software Weekly Test Progress
ORIGINAL PLAN
UPDATED PLAN
TESTS RUN
TESTS READY
IRs TESTED
IRs READY
SNs/IRs TESTED
Percent
100
90
80
70
60
50
40
30
20
10
0
week 1
week 2
week 3
week 4
week 5
week 6
week 7
week 8
week 9
week 10
week 11
week 12
week 13
week 14
Time
Test Tracking

Conclusions

- Confidence Builder for Software Products
  - Test Volume
  - Test Documentation
  - Visibility of Requirements Mapping
- Throughput of Lab Optimized
- Tests Easily Repeatable
- Concept Adaptable to Other Environments

TITLE

AUTOMATED TEST ENVIRONMENT FOR A REAL-TIME CONTROL SYSTEM

AUTHOR

Ronald Owen Hall
Rocketdyne Division

ABSTRACT

An automated environment with hardware-in-the-loop has been developed by Rocketdyne Huntsville for test of a real-time control system. The target system of application is the man-rated real-time system which controls the Space Shuttle Main Engine(SSME). The primary use of the environment is software verification and validation, but it is also useful for evaluation and analysis of SSME avionics hardware and mathematical engine models. It provides a test bed for the integration of software and hardware.

The principles and skills upon which it operates may be applied to other target systems, such as those requiring hardware-in-the-loop simulation and control system development. Potential applications are in problem domains demanding highly reliable software systems requiring testing to formal requirements and verifying successful transition to/from off-nominal system states.

This paper will provide an overview of the environment and the advantages provided.

INTRODUCTION

The need for a high fidelity simulation facility was recognized early in the SSME program. Any undetected control system flaw could be disastrous, either on a test stand or shuttle orbiter flying a Space Transportation System(STS) mission. NASA/MSFC started the initial design of such a facility and the original version became operational at the MSFC in 1975. The facility provides a program unique capability for simulating SSME system operations at nominal and off-nominal conditions. The intent is to perturb variables of software, hardware, or internal engine operations to determine the SSME system response in all modes of operation.

Rocketdyne assumed responsibility for the facility, known as the Hardware Simulation Laboratory(HSL), in 1977. Rocketdyne's experience in operating the lab indicated the need for automation. As the system matured, an extensive test suite and testing process was developed to determine if the target system functions per requirements. The leverage automation could bring to the process became evident.

When the target system was enhanced to a Motorola 68000 series central processing unit, the go-ahead was given to build an automation environment into the HSL. This effort culminated in 1987 with the release of the first version of the environment. Subsequent versions have been released which have extended and refined the capability.

TEXT

The laboratory contains a maximum complement of flight-type hardware (actuators, igniters, sensors, solenoids) and alternately uses component simulation. The SSME is represented with a high fidelity real-time software engine model. Computers induce in real-time off-nominal conditions through failure activation hardware. Test data is accumulated as testing progresses and is automatically analyzed later in non real-time.

The laboratory hardware is shown in figure 1. Provisions are made to allow the control of events, the modulation of inputs and the recording of responses. The configuration includes a VAX 6410 which manages the test environment, a PDP which controls the lab hardware, a CADS which replaces the orbiter, a DDU which overwrites the SSMEC dual port memory, a DAS which traps SSMEC memory histories and simulation events, engine component hardware and simulations, lab hardware, an AD100 realtime engine model and the SSMEC.

ENVIRONMENT CONTROL SYSTEM

Test application is automatically controlled via a software system distributed across the laboratory. A unified collection of test tools allows the specification of tests and data reduction in a simple scripting language. Different versions of the target system may be tested in the environment with no change to the environment control software.

The heart of the system is a compiler and database. The compiler parses scripts generated by users and generates code for the various devices and for test results analysis. The database provides flexibility by maintaining definitions for devices and symbols required by the compiler. The resulting functionality is that of a state-of-the-art facility providing high density test histories generated in a flexible, repeatable, comprehensive test environment.

TEST PROCESS

The test process consists of three distinct overlapping stages: test generation, test application and test results analysis. Test generation is an off-line activity using a text editor to construct test cases in the scripting language. This activity is essentially a mapping of the individual software requirements(IRs) into test scripts. Failure scenarios are scripted and stored in files along with expected results.

Test application is the actual running of the tests in the laboratory. Test cases are grouped into logical divisions and run as units. Test results are captured for subsequent analysis.

Expected test results are embedded in the test scripts. Test results analysis is the off-line application of the expected results segment of the test scripts to the test results. Test results are automatically analyzed to determine if they are what is expected. Result logs are recorded in files to maintain documented evidence of whether the test results are as expected.

The test process is shown in figure 2. The formal test process originates from the SSMEC software requirements document(SRD). The test team analyzes each requirement to write test scripts which challenge the requirements. The automation of the process begins at this point.

The test scripts are compiled into test code to control the laboratory devices and the reduction of test data. The laboratory is initialized to the appropriate configuration. Tests are run, results gathered and reduced and hardcopies generated.

SKILLS

The development of the environment has attracted and produced skills across the entire software life cycle, plus skills in hardware fabrication, design and maintenance, configuration management, MSFC security and quality control. Total quality management concepts are implemented, including the application of continuous process improvement activities and the application of process metrics. A wrap of engineering and programming expertise has been developed which is particularly applicable to the development and operation of automated test environments.

ADVANTAGES OF AUTOMATED TEST ENVIRONMENT

The primary benefit of the automated test environment is that it facilitates a better testing process. Testing is better in terms of test volume, throughput, flexibility, repeatability, exhaustive exercise of the target system and the precise control of the environment. Advantages are incurred in test planning, test performance, test tracking and test management.

TEST PLANNING

The planning of the test cycle is facilitated by the way the environment structures the testing process. The work of test generation may be planned to support subsequent testing and analysis. Laboratory time may be scheduled as the queue of test cases ready for test increase. Test results analysis may be partitioned among personnel and workstations. Personnel and resources may be mixed and matched as the test cycle moves to completion. Feedback from the process stages allows the test planning to adjust to the work flow as shown in figure 3.

TEST PERFORMANCE

The performance of testing is advantaged by the environment at all three stages of the test process. The flexibility and repeatability provided translate into a dramatic reduction of the test period. A 50% time savings for a complete end-to-end verification of the target system has been realized.

In the test generation stage the construction of test cases is made direct and natural via scripting: Software requirements are reformulated as test cases in the scripting language, in the vocabulary of the system. The language allows the writing of test scripts in terms naturally used in defining tests. The constructs include conditionals and assignments. The verbs reflect the action to be taken.

The compiler and database combine to provide this functionality. The database maintains meaningful names for device and software addresses. These names are used in the test scripts in test case specification. The use of physical addresses is avoided, meaning the compiler is independent of laboratory hardware changes and shifting software addresses.

The principle test results output is the binary 128 word(16 bits) vehicle data table(VDT). The VDT is routinely output by the SSMEC 25 times per second. The database maintains meaningful names for each word of the table, yielding verification scripts in terms of the parameters of the target system. This rendering of the verification scripts enhances the specification and analysis of the data reduction process.

Multiple environment definitions may be maintained in the database to allow the flexibility of using different laboratory configurations and versions of the target system. The functional level of the target system is addressed, avoiding the intricacies of the laboratory. The use of raw data are avoided, as data may be entered and expected results expressed in engineering units.

In the test application stage the environment reduces the scope of the work effort to be the application of selected test suites in selected time periods. In each session the environment is initialized to the appropriate laboratory configuration and target system version, allowing anomalous system states to be repeated as necessary. Thus questionable tests results may be quickly and easily duplicated. Technicians may run the suites, allowing test engineers to concentrate on test generation and analysis.

In the test results analysis stage data reduction is preprogrammed to determine if results are as expected. Engineer activity at this stage is usually reduced to scanning the output logs for the phrase "Results as expected." When results are not as expected failure tables are generated to show actual results. Figures 4 - 7 show example scripts with corresponding data reduction log.

TEST TRACKING

Tracking is essential to the test process to provide visibility to management and customer. Tracking is advantaged by the environment in allowing a view of the test process in terms of what is being accomplished in relation to what is planned. The environment facilitates work progress to be quantified in terms of the essential outputs of the process: test cases/software requirements ready for test as well as those completely tested with results analyzed. The quality of the target system is quantified by the ratio of the number of software notes(problem reports) to the number of requirements tested. Figure 8 depicts a typical test tracking chart.

TEST MANAGEMENT

Test management is advantaged by the environment from the perspectives of target system validation and change management. The environment produces documented evidence of whether the target system functions per specification. This validation information is necessary to the formal release of the target system for field use. Directories of data reduction logs are maintained to document test results. These logs are automatically scanned for the phrase "Results Not Expected" to identify testing problems.

As the target system evolves, change management becomes a central focus. Requirements are added, taken away and modified. The environment facilitates the configuration management of the changing test case suite over multiple releases of the target system. As the test cases reside in computer files, they may be managed in a code management system in the same manner as programming source code. Only those test cases need change whose corresponding software requirements change; reuse is facilitated. Additional test cases are added to the suite as new requirements are added.

An essential task of test management is to provide traceability between software requirements, test scripts and data reduction logs. It is necessary to ensure that all requirements are incorporated and satisfactorily tested and that all tests are driven by requirements. This task is accomplished by identifier mappings via identification/naming conventions.

The requirements change notice(RCN) defines changes to software requirements. The RCN identifiers are mapped to the SRD IR identifiers to link each IR to the definitive RCNs. The SRD paragraph numbers are mapped to test script file names to link each test script file to the definitive SRD paragraph. The SRD IR identifiers are mapped to test case names and data reduction log names to link each test case and log to the definitive SRD ID. These mappings are illustrated in figure 9.

SUMMARY

This paper provides an overview of an automated test environment for a real-time control system. The advantages of such an environment are numerous. The principles and skills upon which it operates may be applied to other target systems such as those requiring hardware-in-the-loop simulation and control system development. Potential applications are in problem domains demanding highly reliable software systems requiring testing to formal requirements and verifying successful transition to/from off-nominal system states.

BIOGRAPHICAL SKETCH

Ronald Owen Hall, Engineering Specialist, Rocketdyne Division, Rockwell International Corporation, Southeastern District Operations,555 Discovery Drive Suite 1,Huntsville, Alabama 35806. Currently involved in developing and maintaining laboratory control software and delivery system software for the SSME controller.

APPENDIX

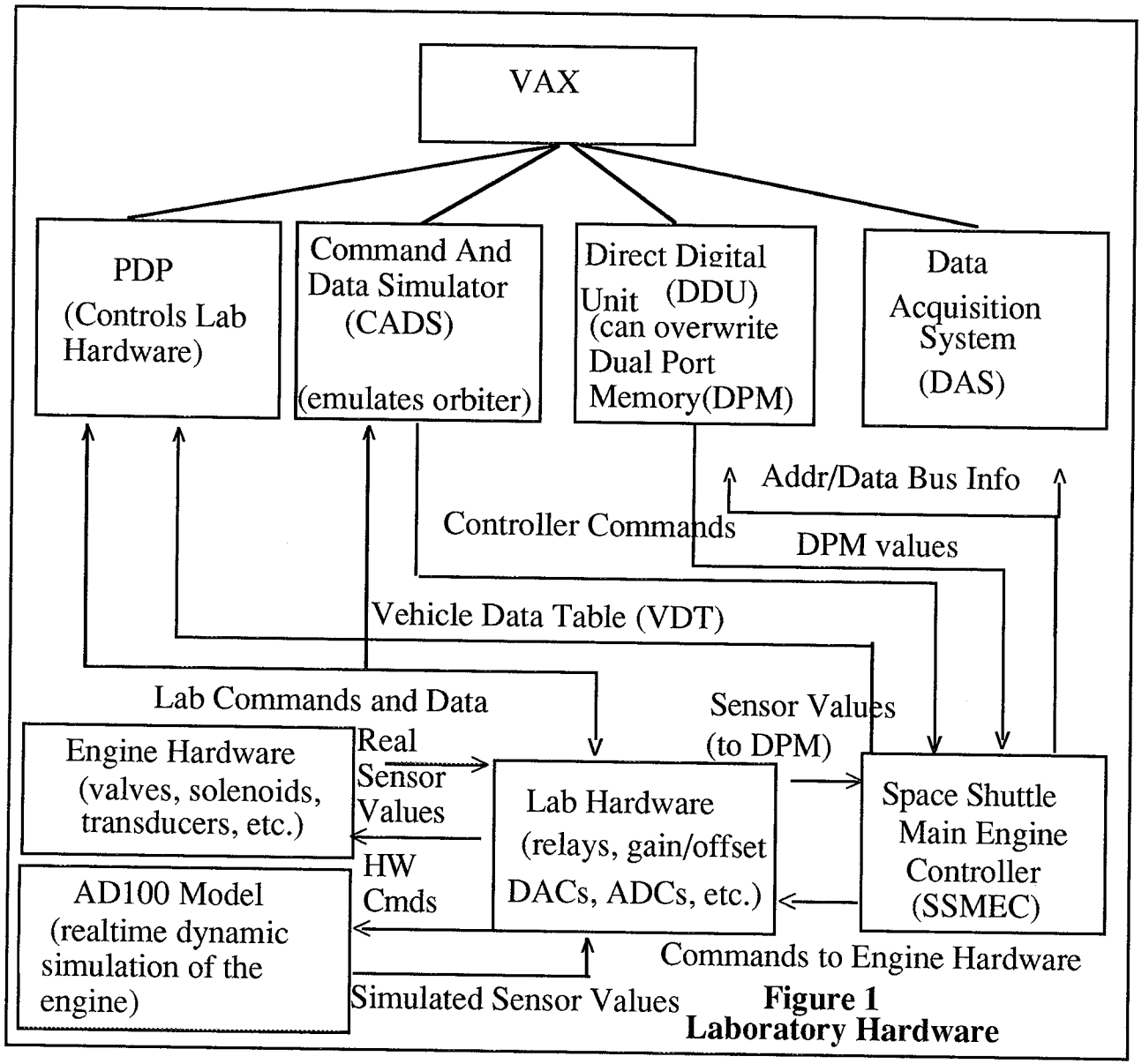


Figure 1
Laboratory Hardware

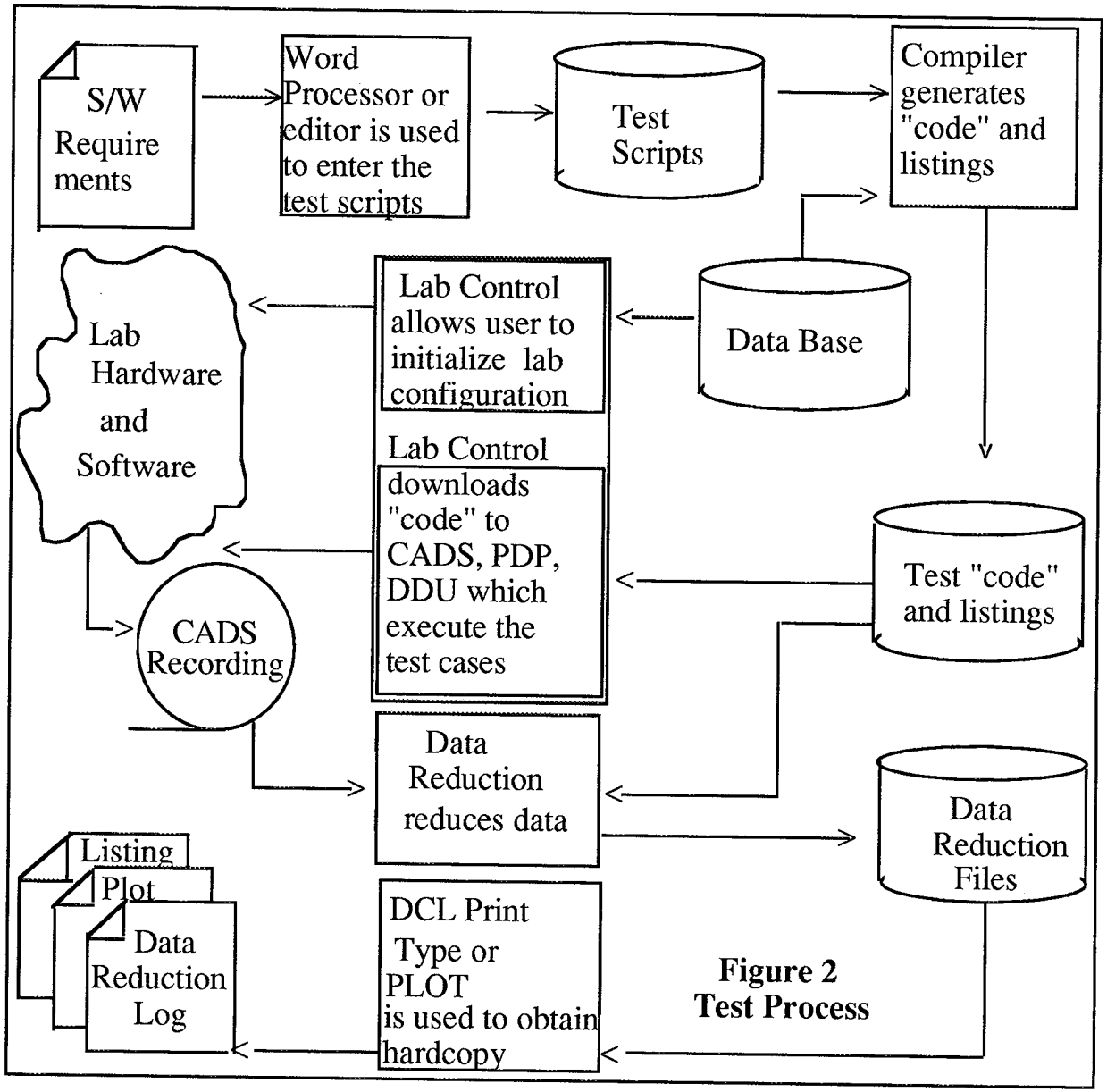


Figure 2
Test Process

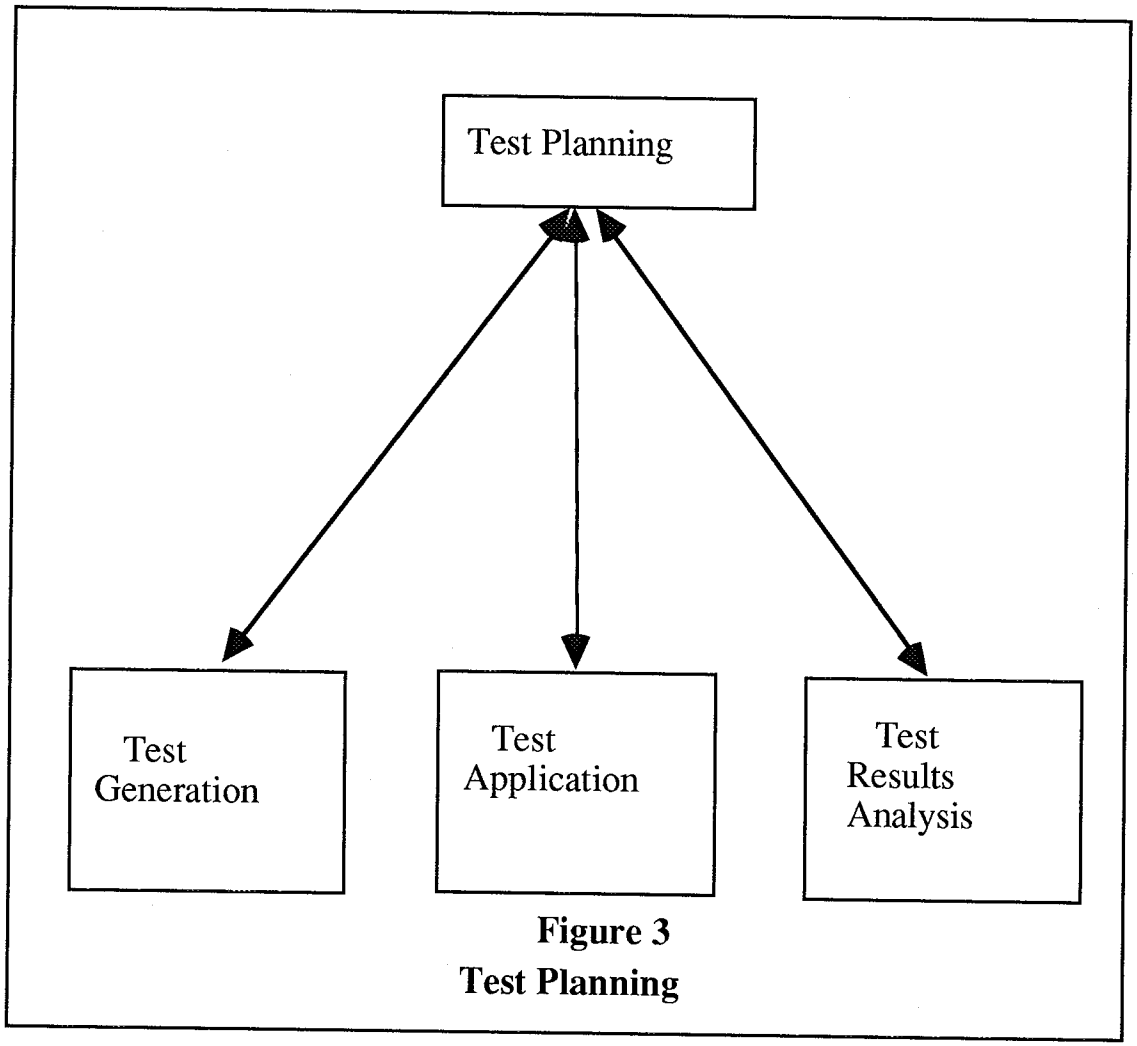


Figure 3
Test Planning

```
This test script verifies the sensor second channel disqualification
failure response for the main combustion chamber pressure
intra-channel test in the start engine phase.

BEGINDDU

/* FAIL THE MCC PC INTRA-CHANNEL TEST FOR
CHANNEL B, THEN CHANNEL A IN THE START ENGINE
PHASE PRIOR TO IGNITION CONFIRMATION */

 FROM ST:*:0.2 TO PS:*:*  /* FAIL CONTINUOUSLY */

  SET MCC_PC_CH_B1_INPUT = 100 /* FAIL CH B */

  SET MCC_PC_CH_B2_INPUT = 176 /* FAIL CH B */
 ENDC

 FROM ST:*:0.6 TO PS:*:*  /* FAIL CONTINUOUSLY */

  SET MCC_PC_CH_A1_INPUT = 200 /* FAIL CH A */

  SET MCC_PC_CH_A2_INPUT = 276 /* FAIL CH A */
 ENDC

ENDDDU
```

Figure 4
Example DDU Test Script

This CADS script controls the simulation by commanding the SSMEC to the appropriate state and by writing the VDT data to a magnetic tape.

```
BEGIN CC
BLAF, HSL_TEST  /* BUILD TEST FILE */
RSME, 100M
CTRS, 100M
EFLT,100M
PSN1, 1S
PSN3, 5S
PSN4, 14S
STEN, 100M
TAPE, 4,0 'IR767:5895;1AF', 100M /* WRITE HEADER TO TAPE */
TAPE,1,100M   /* BEGIN RECORDING */
STRT,2S    /* FAILURES OCCUR DURING START */

TAPE,0,100M   /* STOP TAPE RECORDING */
TAPE,3,16S    /*  WRITE TAPE EOF AFTER SHUTDOWN */
CTRS, 1S
CLER
\
BEGINGO
EXTR, HSL_TEST   /* EXECUTE TEST FILE */
ENDGO
ENDCC
```

Figure 5
Example CADS Control Script

This data reduction script directs the capture of the essential test results.

```
BEGINDR

/* VERIFY CORRECT FAILURE PROCESSING */
 WHEN VDT_WORD_100 == %O11201  /* CH A FAILURE */
  VERIFY FAILURE_COUNTER == 2
  VERIFY VDT_103_F3 == -76.0
  VERIFY _SELF_TEST_STATUS == _MCF

  WHEN _DELAY >= 1 /* DELAY ONE MAJOR CYCLE */
   VERIFY _PHASE_MODE == _THROTTLING_TO_ZERO_THRUST
   WHEN (MCC_PC_REF == 0) && (_CHECKED_ONCE == 0)
    VERIFY MCC_PC == _PREV_MCC_PC_REF
    SET _CHECKED_ONCE = _TRUE
   ELSE
    VERIFY MCC_PC == MCC_PC_REF
   ENDC
  ENDC

  SET _DELAY = _DELAY + 1

 ENDC
 SET _PREV_MCC_PC_REF = MCC_PC_REF

ENDDR
```

Figure 6
Example Data Reduction Script

```
 WHEN VDT_WORD_100 == %O11201
 VERIFY FAILURE_COUNTER == 2
...RESULTS AS EXPECTED

 VERIFY VDT_103_F3 == -76.0
...RESULTS AS EXPECTED

 VERIFY (ENGINE_STATUS_WORD  &  %X0003) == 2
...RESULTS AS EXPECTED

 WHEN _DELAY >= 1
  VERIFY (ENGINE_STATUS WORD  &  X00FC)  ==  %X00A8
...RESULTS AS EXPECTED

  WHEN (MCC_PC_REF == 0) && (_CHECKED_ONCE == 0)
   VERIFY MCC_PC == _PREV_MCC_PC_REF
...RESULTS AS EXPECTED

   SET _CHECKED_ONCE = 1
  ELSE
   VERIFY MCC_PC == MCC_PC_REF
...RESULTS AS EXPECTED
  ENDC
 ENDC
 SET _DELAY = _DELAY + 1
ENDC
SET _PREV_MCC_PC_REF = MCC_PC_REF
CASE SUMMARY =   PASS 6     FAIL 0
```

Figure 7

Example Data Reduction Log

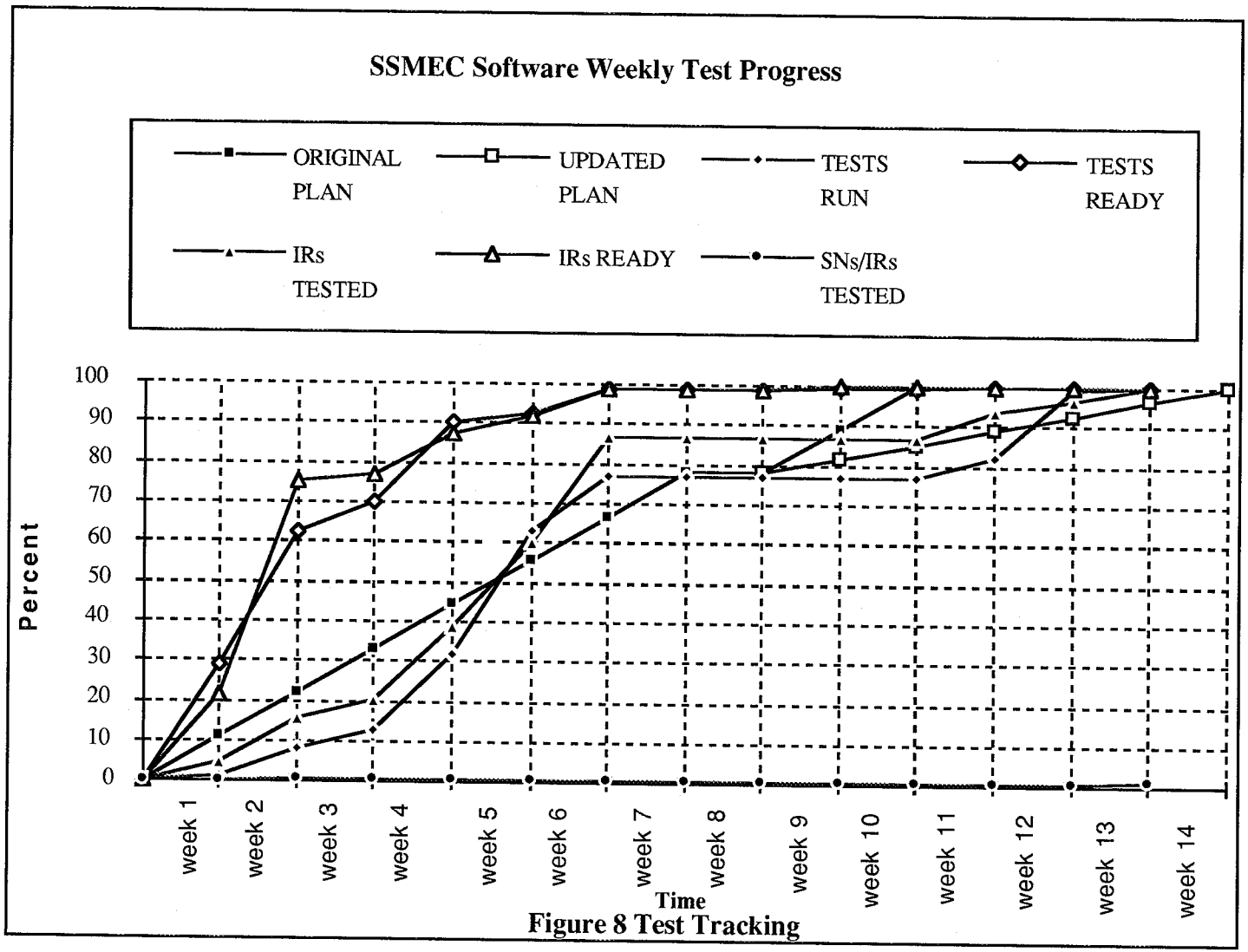


Figure 8 Test Tracking

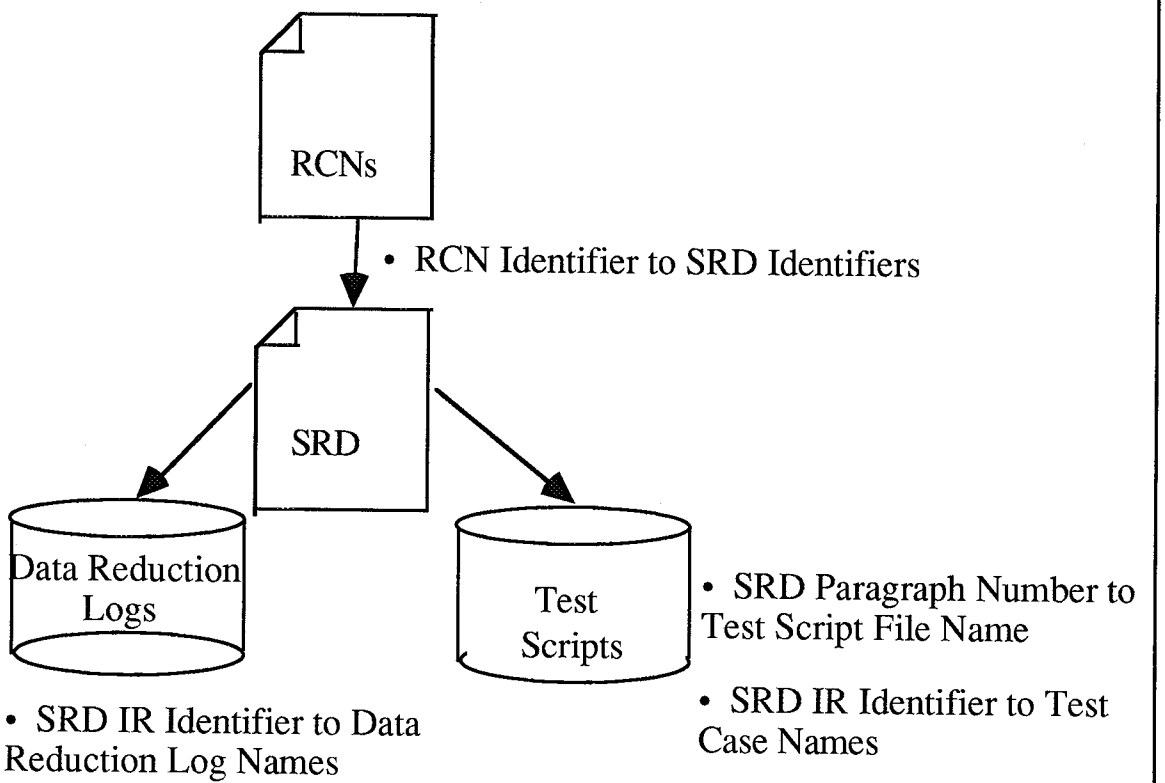


Figure 9

Test Management Identifier Mappings