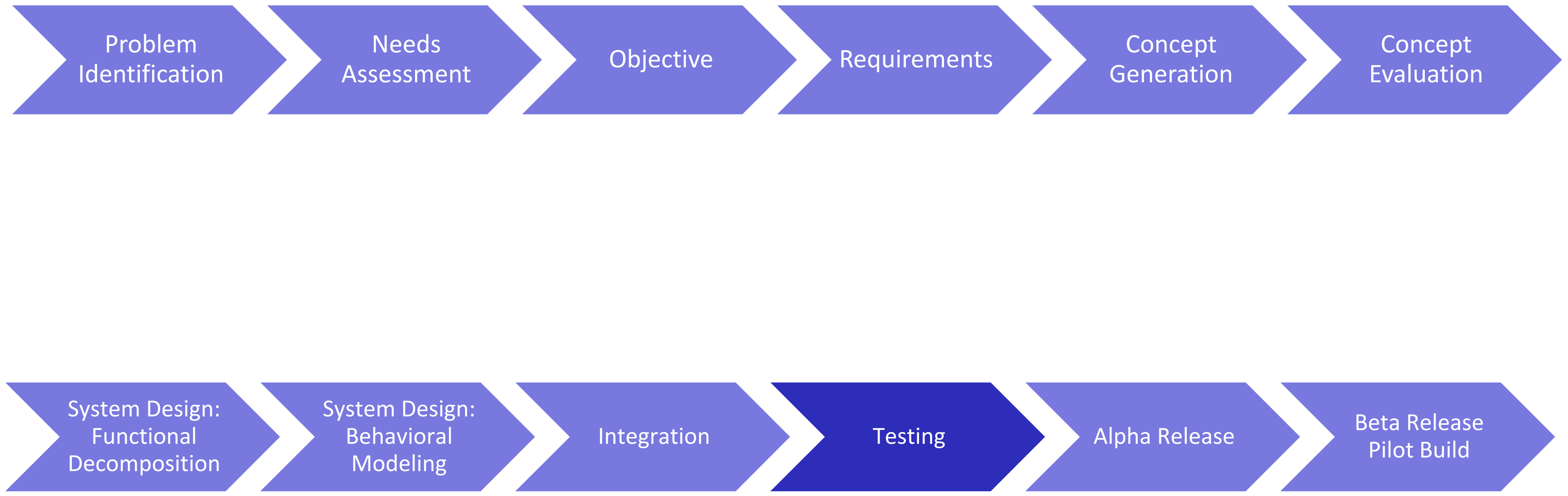
ECE 411
Engineering Practices
“Testing and Documentation”

Portland State University
Mark G. Faust

Outline

- Testing Principles
- Concurrent Test
- Key Terms and Concepts
- Test Documentation
- Testability
- Black Box and White Box Testing
- Stubs
- Debugging
- Examples
- Documentation

Process


Problem Identification
Needs Assessment
Objective
Requirements
Concept Generation
Concept Evaluation
System Design: Functional Decomposition
System Design: Behavioral Modeling
Integration
Testing
Alpha Release
Beta Release Pilot Build

Testing Principles

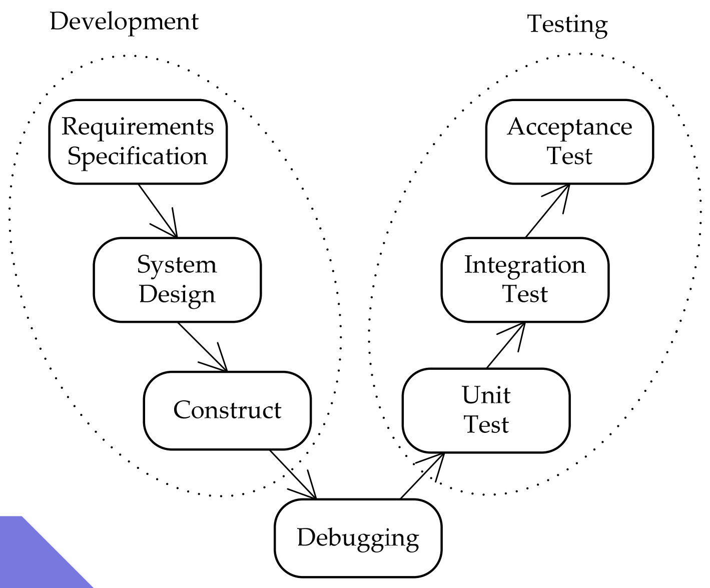
- Different interpretations
  - Design verification: “Is my design correct?”
    - Goal: debug design
    - Involves: human error, misinterpreted requirements or specifications, bugs in tools…
  - Manufacturing test: “Is this a faithful copy of the design?”
    - Goal: detect manufacturing defects
    - Involves: process-specific failure modes (often modeled more simply)
      - PCB manufacturing: wrong part, part backwards, bent/broken pin, bad solder joint (stuck at 0/1, open/short model)
      - IC manufacturing: mask misalignment, contaminants

Testing Principles

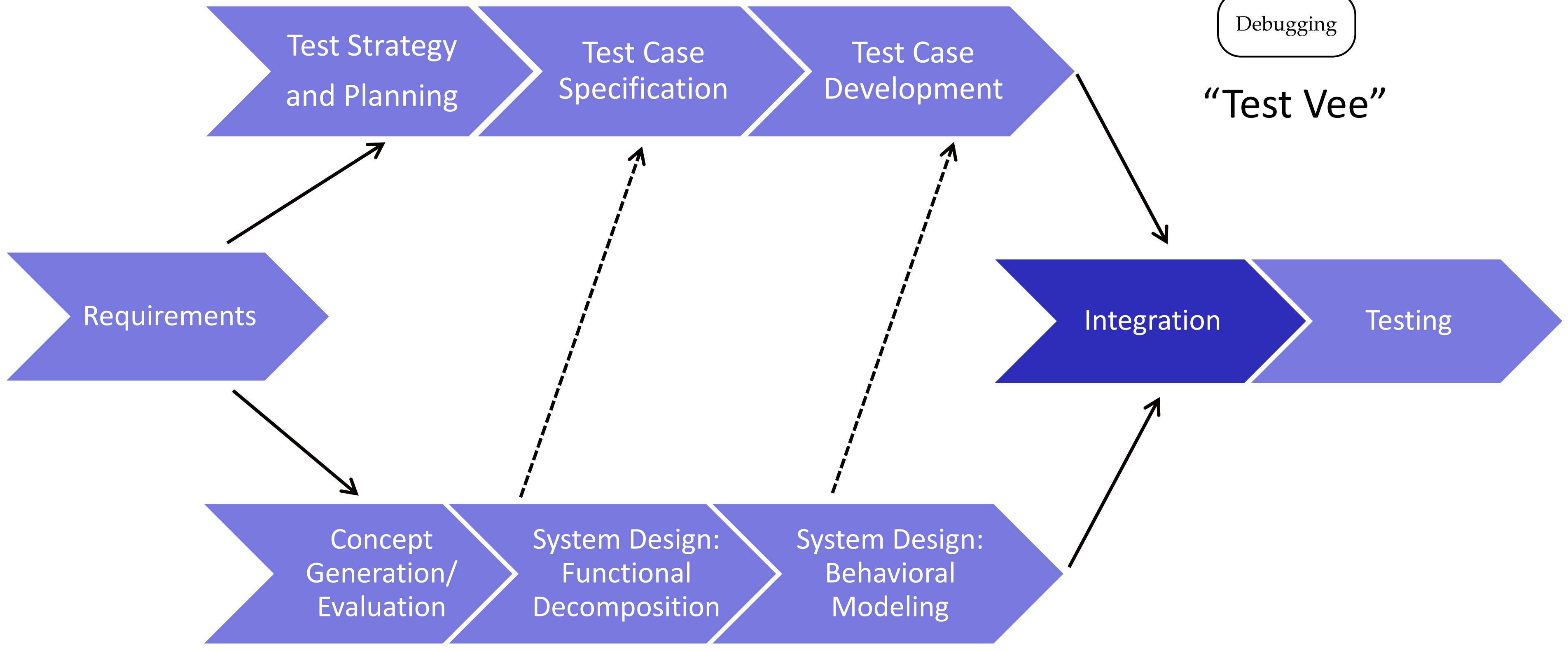
- Detecting and correcting bugs early saves money
  - Rule of 10!
  - Depending upon nature of bug and cohesion of system, fix in one module may affect other modules, requiring re-design and re-test of them
- Testing proceeds in parallel with the design process
  - “Concurrent Engineering”
  - Test planning during requirements development phase
    - May revise based on further design and implementation decisions
  - Specify test cases during preliminary design (specification)
  - Write/develop test cases during detailed design
  - Perform/apply test cases during implementation
  - Increasingly larger subsystems tested as they are integrated

Test Documentation

- Test Plan
- Test Case Descriptions
- Test Cases
- Test Results



“Test Vee”



Test Strategy and Planning
Test Case Specification
Test Case Development
Requirements
Integration
Testing
Concept Generation/ Evaluation
System Design: Functional Decomposition
System Design: Behavioral Modeling

“Concurrent Engineering”

Unit/Module Test

- Test of individual module (unit) using module’s specification (or model) as description of desired behavior. Depending upon cohesion and connectivity, may require significant test “fixture” – e.g. stubs

Integration Test

- Test of two or more modules or hardware/software together.

Acceptance Test

- Testing of system for acceptance by client. Must adhere to functional and performance specifications

Parametric Test

- Testing to determine or confirm detailed parameters (e.g. noise margin, THD, setup/hold times, Voh, Vol, power consumption) of a completed design.

Functional Test

- Test to confirm basic functionality (usually excludes parametric test)

Exhaustive Testing

- Testing every possible input/state. Often not feasible. Tradeoff involving test cost (time, $). Need for a test strategy.

Stress Testing

- Testing to determine system performance (go/no-go or degradation) when stressed (e.g. boundary conditions at which system intended to perform).

Alpha Testing

- Testing by small number of users (often internal) through actual use. May involved preliminary version of system

Beta Testing

- Testing by larger number of (external) users through actual use.   Usually involves final configuration/version of system built as though in actual production.

White Box Testing

- Testing a module or system with detailed knowledge of its internal organization/implementation or fault model.   May also take advantage of access to lower level/internals.

Black Box Testing

- Testing a module or system only with knowledge of its external specification (e.g. inputs, outputs, functionality) and using only inputs and outputs

Installation Testing

- Ability to install system using only supplied documentation and permitted tools

Use Testing

- Task-oriented testing by “typical” user with access to available documentation

Error Testing

- Are (anticipated) errors handled correctly?   May be incorporated into functional testing

Environmental Testing

- Testing against environmental specifications (e.g. temperature, humidity, vibration, noise)

Regression Testing

- Applying the same tests to successive versions of a system to ensure no unintended changes in behavior or performance were introduced

Importance of Test Documentation

- Avoid “random” testing (e.g. just playing with it, “smoke test”)
- Employ a strategy to
  - Detect most bugs
  - Confirm adherence to requirements
- Ensure tests are applied so that they actually achieve their goal
  - Test conditions
- Ensure tests are repeatable
  - With different personnel
  - At different times
  - With different versions (regression testing)
- Ensure adequate resources available for testing (not just what testing can be accomplished in time that remains!)
  - Personnel
  - Equipment
  - Time

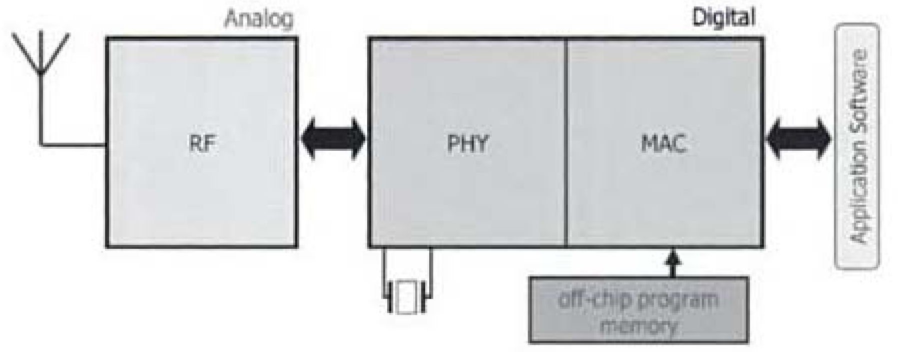
Importance of Test Documentation

- Allows for review and comment
  - By designer
  - By test engineers
  - By client
- Allows more concurrent engineering
  - Well documented tests can be performed by others in parallel
- Valuable for design process
  - Many bugs uncovered just through review of test plan or test case
- Provide documentation
  - Field returns
  - “Did we ever test that…”
- Legal or regulatory requirements
  - May require retention of test results as well as plans, cases

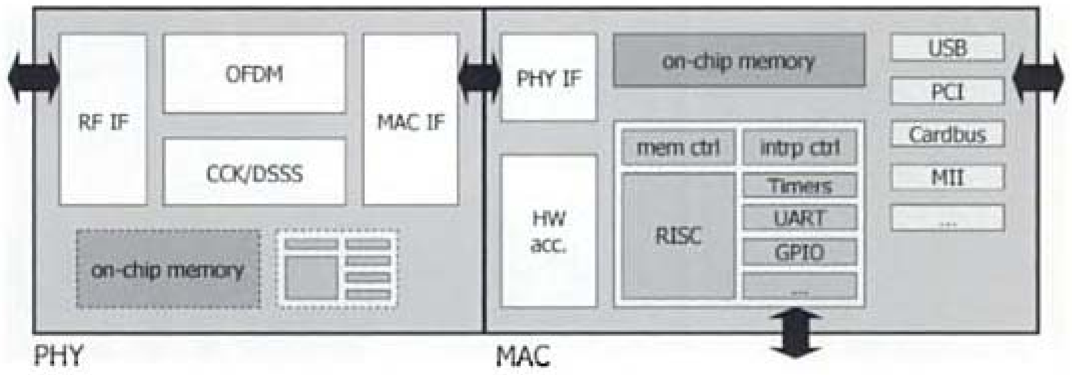
Testability

- Testability
  - Design is "testable" if a failure can be quickly located
  - Strategy: improve observability and controllability
- Observability
  - Ability to observe any node of the system
  - How will an error or bug manifest itself at the system's outputs or observable state?
- Controllability
  - Ability to control or set any node to a prescribed value
  - What combination of inputs (over time) required to set up condition where bug can be detected?
- Black box testing
  - Low observability and controllability (only at module inputs and outputs)

Black Box Testing

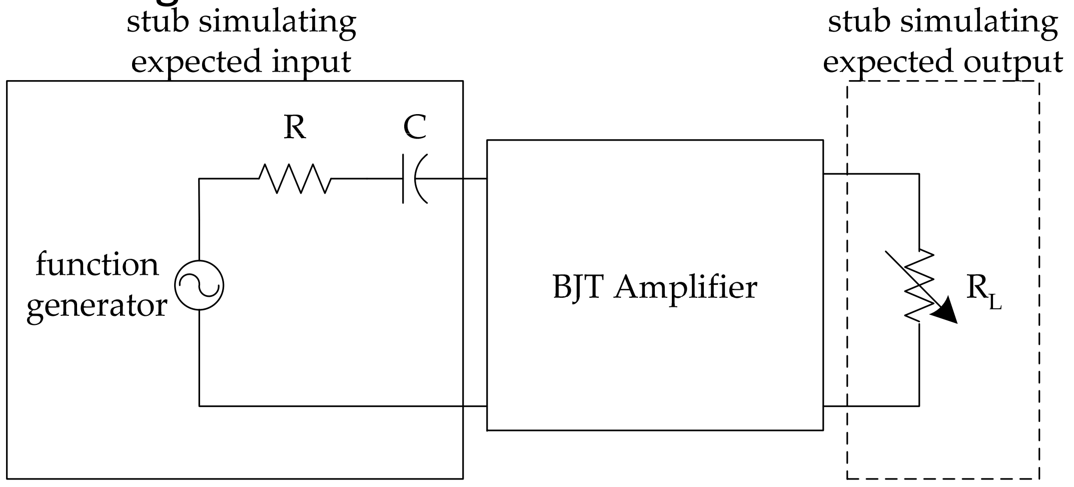


White Box Testing

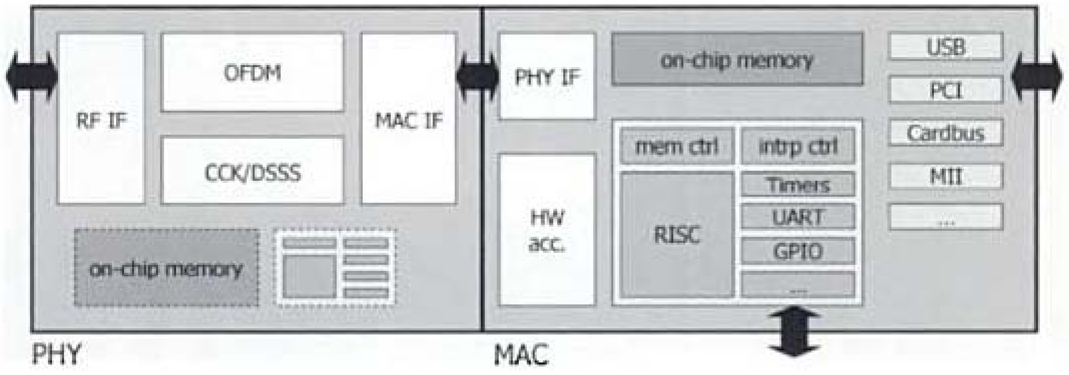


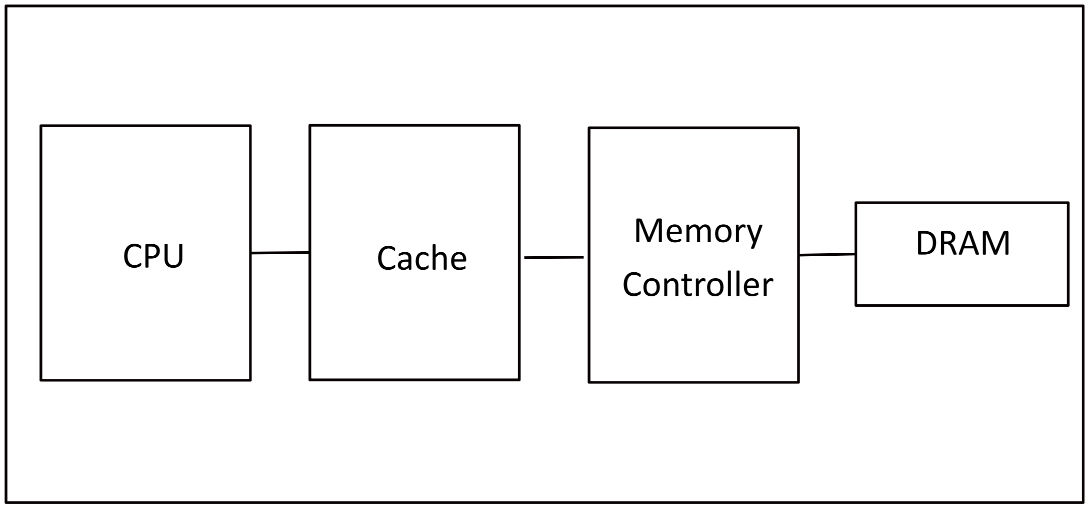
Stubs

- Placeholder for module or subsystem which interfaces to UUT
  - Simulates inputs or monitors outputs
  - Permits individual modules to be tested prior to completion of others
  - May not need complete functionality
- Examples
  - Function generator for audio input
  - sscanf() with ASCII data file instead of actual file reader
  - High level Verilog behavioral model


stub simulating expected input
stub simulating expected output
R
C
function generator
BJT Amplifier
R_L

Stub Examples





Test Case Properties

Accurate

The test case should check what it is supposed to and exercise an area of intent

Economical

Tests should be performed in minimal number of steps

Limited in complexity

Tests should consist of moderate number of steps (e.g. 10-15)

Repeatable

Test should be able to be performed and repeated by another person

Appropriate

The complexity of the test should allow it to be performed by others assigned the testing task

Traceable
The test should verify a specific requirement

Self-cleaning
The system should return to the pre-test state after the test is complete

Debugging

- Bohr bugs
  - Electrons have a definite position
  - Bugs are reliably reproducible
  - Error always in same place
  - Usually yield to typical debug/diagnosis methods
- Heisen bugs
  - Electron position is probability function
  - Bug appears/disappears with innocuous, unrelated changes to inputs
  - Errors move around the system
  - Software: memory allocation errors, spurious interrupts, bad pointers
  - Hardware: bad grounds, bad memory, power supply
  - May require thinking outside box, brainstorming

Debugging Process

- Observe the problem under different operating conditions
- Form a hypothesis as to what the potential cause is
  - What could cause the behavior observed?
  - Would it also cause behaviors not observed?
  - Which causes are most likely?
- Conduct experiments to confirm or eliminate the hypothesized source of the problem
  - Isolate the UUT
  - Control for all other sources except hypothesized one
  - Confirm the altered condition/source
- Repeat as necessary

Debugging

- Check easiest problems first
  - Others may go away as these are found and fixed
- Start at lowest levels of abstraction
  - Upper levels rely on lower levels
- Examples
  - Is the system powered up?
  - Is the testing equipment adjusted properly?
  - Are the bus lines being correctly manipulated?
  - Has the system been properly initialized?

Unit Tests

- Test of functionality of module in isolation
  - Traceable to detailed design
  - Consists of family of test cases
  - Each test case establishes that system performs some functionality correctly according to specification
  - Test cases expressly written to uncover defects
- Write unit tests during implementation of module
  - Just thinking about test situations and conditions may lead designer to uncover or avoid errors before they're design into system
  - Unit tests written with understanding of internal organization of the module (best understood during development)
  - Review process is symbiotic
  - Concurrent engineering!

Example Test Case

Test Writer: Sue L. Engineer			
Test Case Name:	Finite State Machine Path Test #1	**Test ID #:**	FSM-Path-01
Description:	Simulate insertion of money with a mix of nickels and dimes. Verifies FSM, outputs candy in response to a total deposit of $0.30.	**Type:**	☑ white box ☐ black box
Tester Information			
Name of Tester:		**Date:**	
Hardware Ver:	1.0	**Time:**	
Setup:	Make sure that the system was reset sometime prior and is in state $0.00.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Strobe Nickel	State should go to $0.05				
2	Strobe Dime	State should go to $0.15				
3	Wait	State should remain $0.15				
4	Strobe Nickel	State should go to $0.20				
5	Strobe Dime	State should go to $0.25				
6	Nothing	State should go to $0.00				
	Overall test result:					

Matrix Tests

Repeated tests with different parameters

Test Writer: Sue L. Engineer			
Test Case Name:	ADC function test	**Test ID #:**	ADC-FT-01
Description:	Verify conversion range and clock frequency. Output goes to 0 in presence of null clock.	**Type:**	☐ white box [✓ black box]
Tester Information			
Name of Tester:		**Date:**	
Hardware Ver:	1.0	**Time:**	
Setup:	Isolate the ADC from the system by removing configuration jumpers.		

Test	V_T	**Clock**	**Expected output**		**Pass**	**Fail**	**N/A**	**Comments**
			Decimal	**Hexadecimal**				
1	0.0V	10kHz	0	0x000				
5	2.0V	0Hz	0	0x000				
Overall test result:								

Integration Tests

- What are the different paths through the system?
- Are all modules exercised at least once during testing?
- Have all interfaces been tested?
- Have all interface modes been exercised?
- Does the system performance meet specification?
  - Throughput, bandwidth, timing, etc.

Acceptance Tests

- May be contractual (legal) document
- Typically written along with requirements document
- Traceable to engineering requirements
- Identifies scope
  - How much of system and functionality to be tested
- Identifies level
  - How deep will testing be performed

Automating Tests

- Facilitate performing tests
  - Scripts or programs to create necessary conditions, test cases, input files/DB
  - Scripts or programs to execute tests (controlling inputs, collecting outputs)
  - Scripts or programs to compare acquired results with expected or known good results
- Particularly useful for repeated tests
- Regression testing
  - Perform regression test suite on new versions/releases
  - Compare with prior known good results
  - Investigate differences
    - Bugs introduced
    - Variance due to new components or subsystems
      - New supplier
      - Change in process

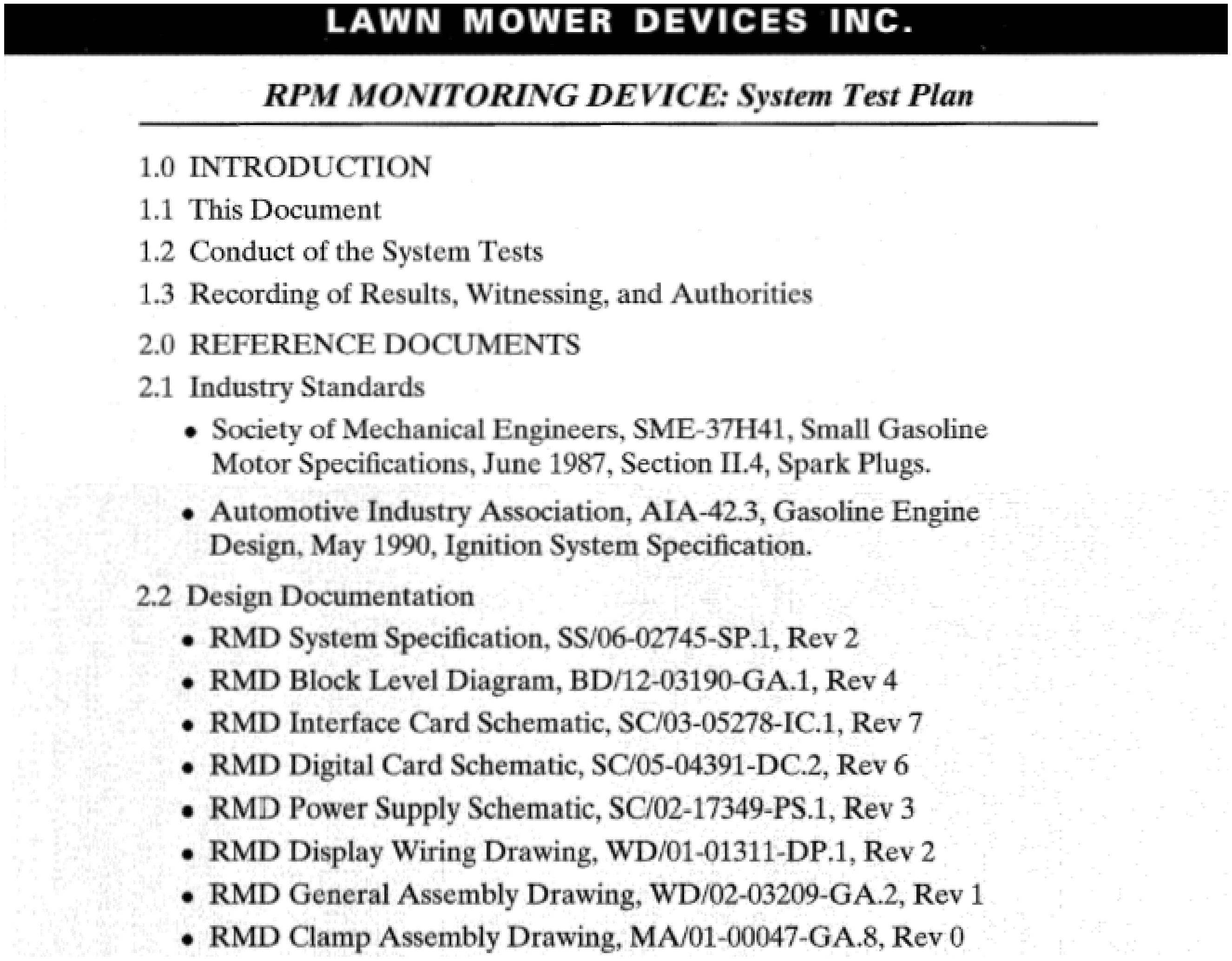
Test Plans

- Identify specific system being tested
  - Relevant specifications/requirements and versions being tested against
- Identify objectives of test (e.g. acceptance, stress, functional)
- Identify resources required
  - Number of copies/instances of UUT
  - Personnel: Training, expertise, number
  - Additional equipment, software, lab instruments, specialized test fixtures
    - Environmental chambers (noise, temp, humidity, vibrations, EMI)
    - Interfaces (e.g. software, drivers); Specify versions if necessary
    - Some may be specific to groups of tests
- Describe each test case (outline form)
  - Organize hierarchically, grouping tests by purpose and functionality covered
  - Specify: objective, preconditions, test procedure, anticipated results
- Review the test plan!   Who?

Remember: All Documentation

- Title Page: Authors, Date, Revision Number.
- Revisions: Update revision number, date. Consider using change bars to assist reviewers (nobody wants to read the entire specification after each revision). Applies to all documents (test plans, internal specs, etc)
- Change history: Consider a change history so you can track changes to the specification (either embed in document or use revision control system.
- Revision Control System: Consider using a revision control system to help manage revisions.
- Document niceties: Use a table of contents, page numbers, section numbers, and page headers that reflect section number/name to aid readability and assist reader in locating passages.

Test Plan Example: RPM Monitoring Device



LAWN MOWER DEVICES INC.

RPM MONITORING DEVICE: System Test Plan

1.0 INTRODUCTION

1.1 This Document

1.2 Conduct of the System Tests

1.3 Recording of Results, Witnessing, and Authorities

2.0 REFERENCE DOCUMENTS

2.1 Industry Standards

- Society of Mechanical Engineers, SME-37H41, Small Gasoline Motor Specifications, June 1987, Section II.4, Spark Plugs.
- Automotive Industry Association, AIA-42.3, Gasoline Engine Design, May 1990, Ignition System Specification.

2.2 Design Documentation

- RMD System Specification, SS/06-02745-SP.1, Rev 2
- RMD Block Level Diagram, BD/12-03190-GA.1, Rev 4
- RMD Interface Card Schematic, SC/03-05278-IC.1, Rev 7
- RMD Digital Card Schematic, SC/05-04391-DC.2, Rev 6
- RMD Power Supply Schematic, SC/02-17349-PS.1, Rev 3
- RMD Display Wiring Drawing, WD/01-01311-DP.1, Rev 2
- RMD General Assembly Drawing, WD/02-03209-GA.2, Rev 1
- RMD Clamp Assembly Drawing, MA/01-00047-GA.8, Rev 0

2.3 Other

- Ignition Spark Simulator, Operations Manual, ISS-OM/12, Rev 3
- RMD Draft Operations and Maintenance Manual, RMD-OM/D2, Rev 2

3.0 RMD OVERVIEW

3.1 Operational Description

3.2 Definition of Terminology

3.3 Computational Methods

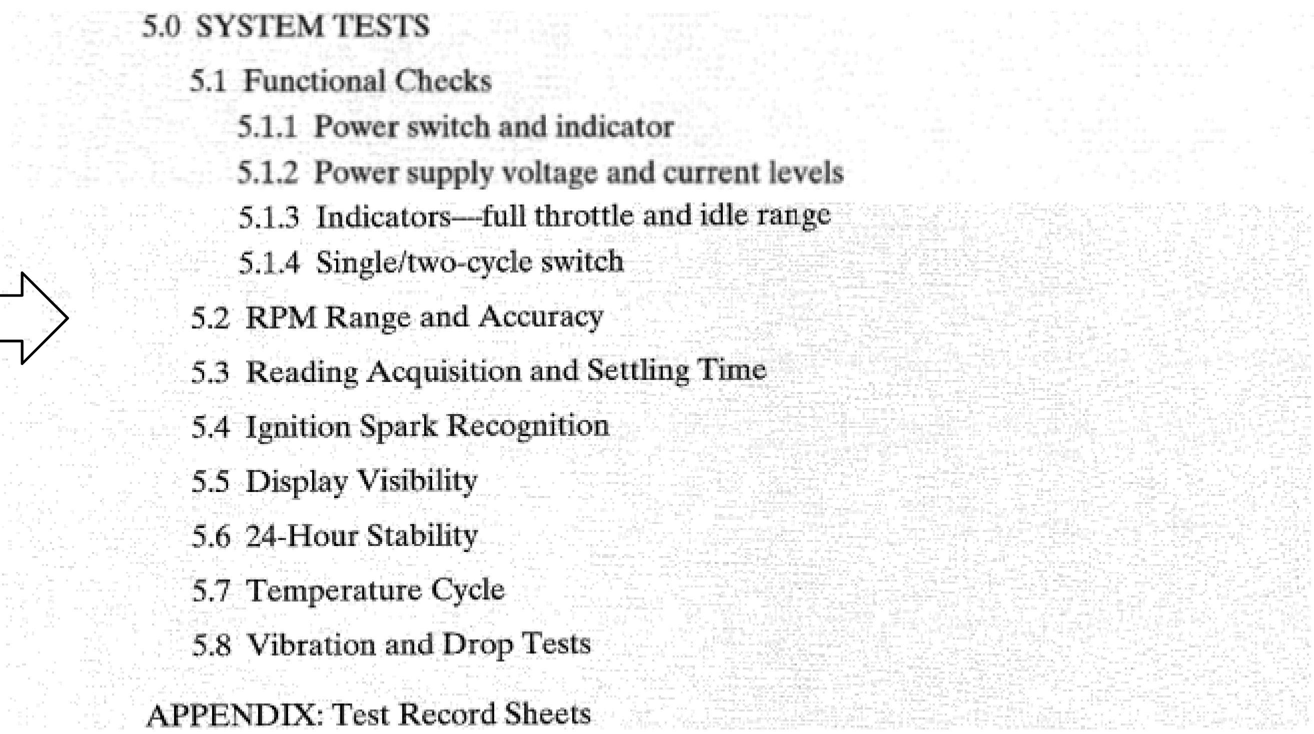
- Peak to RMS voltage conversion
- Temperature cycle to aging equivalent
- System reliability calculations

4.0 PRETEST PREPARATION

4.1 Test Equipment

- Frequency Counter PH417
- Oscilloscope Extron5203C
- Multimeter Lufk415
- Ignition Spark Simulator, LMD Inc., ISS, model 02
- Temperature Chamber, Cycle Inc., TCC model 01
- Vibration Simulator, Luntian Enterprises,
- VibroGen, model 01

4.2 Test Setup and Calibration



Individual test case descriptions (identified by name/number) go here

Parking Meter Example

- Materials or equipment required
  - Two parking meter stations
  - One base station
- Installation Test
  - Additional equipment required
- Environmental Test
  - Additional equipment required
- Use Test
  - Additional equipment required: 1 roll each of quarters, dimes, nickels; 1 valid credit card
  - Additional equipment/materials required
  - Transaction with coins
  - Transaction with valid credit card

Parking Meter Example

- (User) Error Test
  - Additional equipment required: 1 Canadian quarter, 1 invalid credit card
  - Bogus coin
  - Invalid credit card
  - Transaction during holiday or hours when fee not required
  - Transaction when coin box full
- (System) Error Test
  - Parking communication when base station not up
- Stress Testing
  - Too many parking stations for base station

Parking Meter Example

- Module Tests
  - Power (battery, solar panel, charging)
  - Communications
  - Printer
  - Coin mechanism
  - Credit Card reader
  - LCD Display
  - Keyboard
- Don’t Forget …
  - Low battery power (perhaps indication of malfunctioning or blocked solar panel)
  - Low printer paper
  - Full coin box
  - Low printer ink

Example: Autonomous Robot

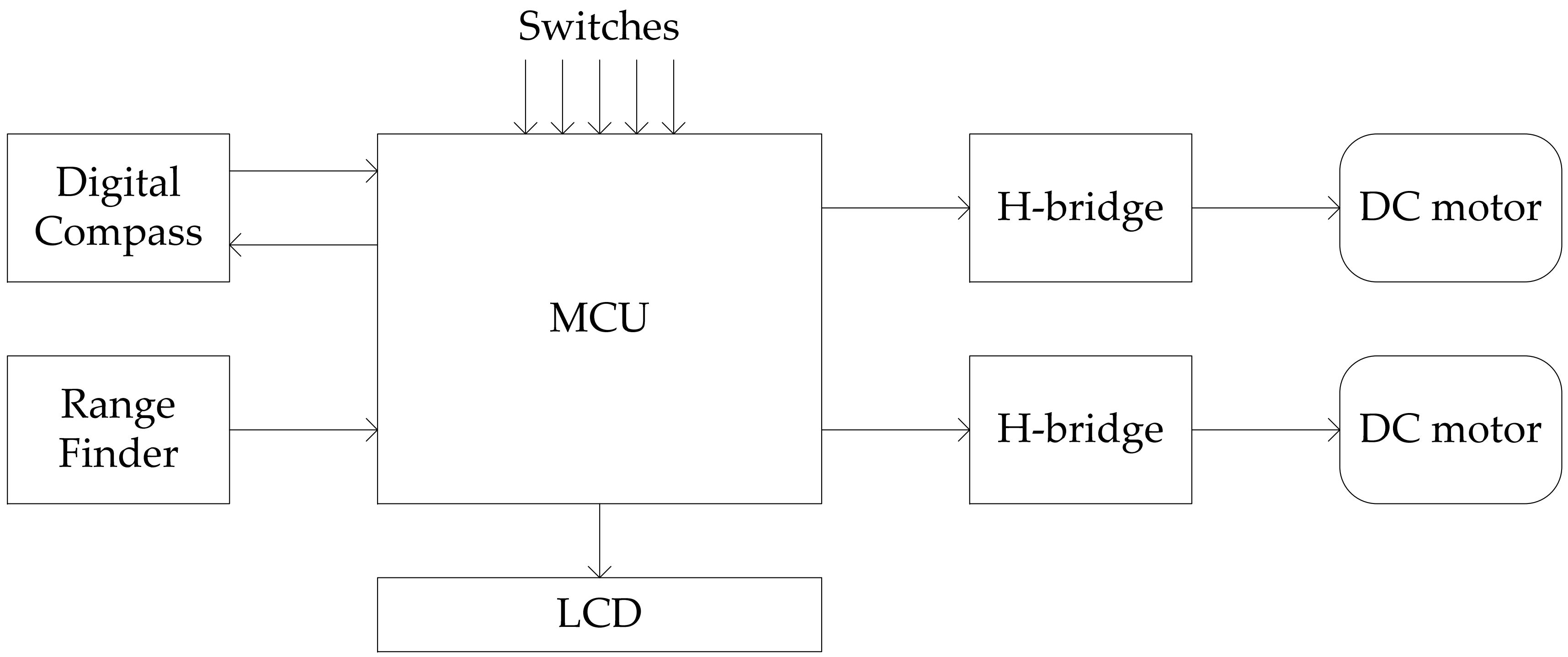
- Autonomous navigating robot
- Key engineering requirements
  - The robot’s center must stay within 12-18 cm of the wall over 90% of the course, while traveling parallel to a wall over a 3 meter course
  - The robot’s heading should never deviate more than 10 degrees from the wall’s axis while traveling parallel to a straight wall over a 3 meter course

Robot Acceptance Test

Test Writer: Sue L. Engineer			
Test Case Name:	Robot acceptance test #1	**Test ID #:**	Robot-AT-01
Description:	Checks the engineering requirement: *The robot's center must stay within 12 to 18 centimeters of the wall over 90% of the course, while traveling parallel to a wall over a 3 meter course.*	**Type:**	☐ white box ☑ black box
Tester Information			
Name of Tester:		**Date:**	
Hardware Ver:	Robot 1.0	**Time:**	
Setup:	Completed robot should be fully charged and placed on 3 meter test track.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Write a program to monitor the robots position from the wall.	Program should be statically tested to verify accuracy. Should sample wall at a sufficient rate depending on speed.				
2	Put robot on test track, run test, and download data.	The robot should travel down the entire length of the test track and then stop.				
3	Plot test data in a spreadsheet program.	Plot of position vs. time should be within 12 – 18 cm 90% of the time.				
	Overall test result:					

Robot Architecture



- Unit Test Possibilities
- Integration Test Possibilities
  - MCU + motors + bridge + switches
  - Chassis + range finder + MCU
- Stubs/Test Modes/Automation
  - Control compass headings to MCU
  - Control range finder to MCU
  - Display output to motors on LCD

Unit Test: Digital Compass

Module	Digital Compass – Geosensor version 2.3
Inputs	- Earth's magnetic field: An orientated field of magnetic force beginning and ending at the earth's magnetic poles. - SClk – Clock signal to clock data through the module. Maximum Frequency is 10Mhz. - SDIn – Serial data input to send data into the compass module. Date is valid on positive SClk edges.
Outputs	- SDOut – Serial data output from the compass module. Data is valid on negative clock edges.
Functionality	Senses the earth's magnetic field and determines the orientation of the compass with respect to the field. This orientation is stored in an internal register and can be retrieved through the SPI interface.
Test	Comp-UT-01

Unit Test: Digital Compass (Matrix)

Test Writer: Sue L. Engineer			
Test Case Name:	Compass unit test #1	**Test ID #:**	Comp-UT-01
Description:	Checks that the compass returns correct angular measurements to the MCU. Test program is in ./test/compass_unit_test_1.c	**Type:**	white box [✓ black box]
Tester Information			
Name of Tester:		**Date:**	
Hardware Ver:	Compass Module - Geosensor version 2.3	**Time:**	
Setup:	Compass module should be wired to the MCU through the SPI interface pins. The MCU should be connected to an RS232 terminal through its SCI interface. The terminal should be configured to run at 9600 baud. Cardinal directions map should be aligned using the magnetic compass.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Compile compass.c in /test directory	IDE should generate no warnings or errors.				
2	Download	MCU should report "download successful"				
3	Execute	MCU should display compass splash screen on terminal interface.				
4	Orientate compass to 0 degrees.	Terminal interface should display 0 degrees +/- 10 degrees.				
5	Orientate compass to 30 degrees.	Terminal interface should display 30 degrees +/- 10 degrees.				
6	Orientate compass to 45 degrees.	Terminal interface should display 45 degrees +/- 10 degrees.				
…	…	…				
12	Orientate compass to 315degrees.	Terminal interface should display 315 degrees +/- 10 degrees.				
	Overall test result:					

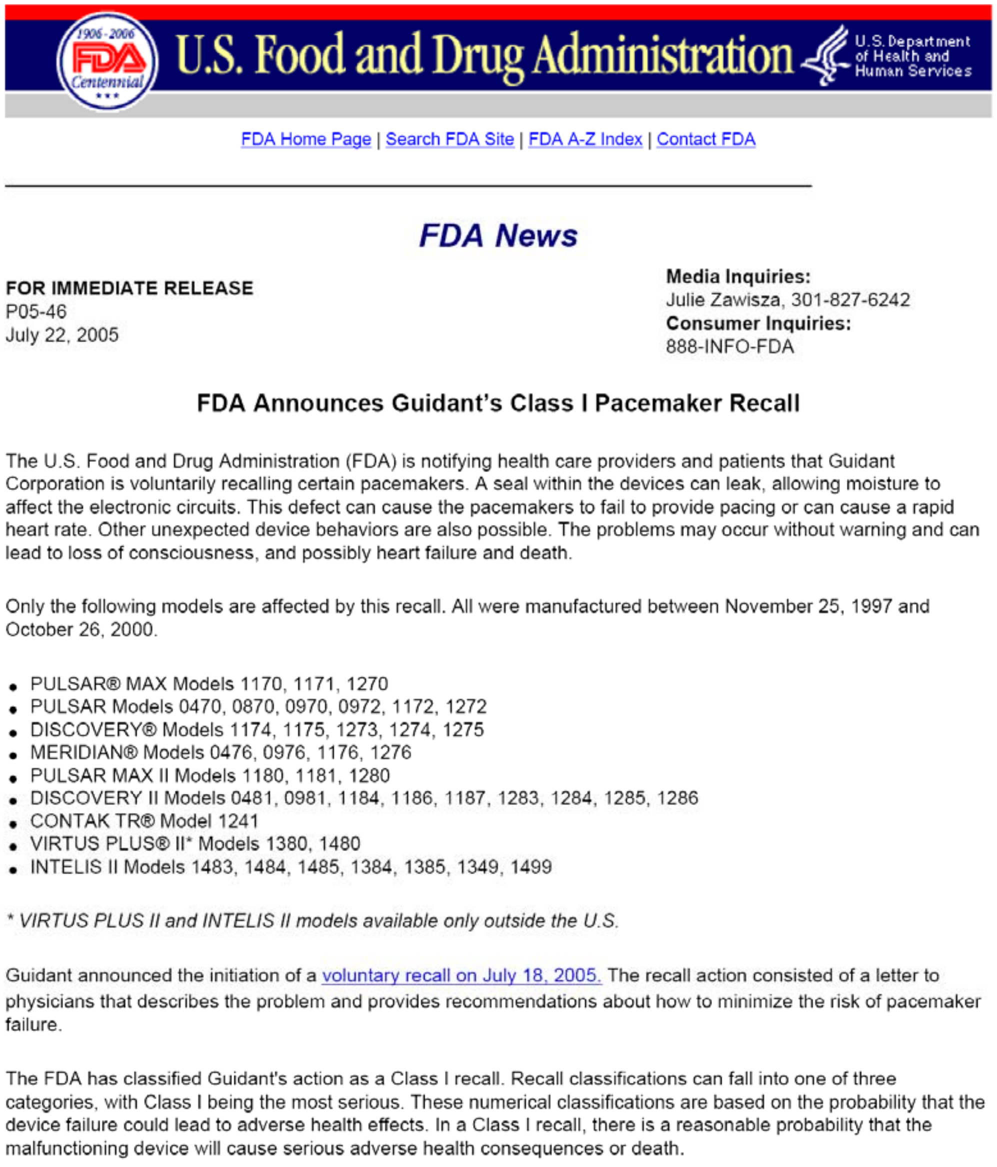
Step-by-Step Integration Test

Test Writer: Sue L. Engineer			
Test Case Name:	Robot integration test #1	**Test ID #:**	Robot-IT-01
Description:	Checks interaction of DC motors on the magnetic compass.	**Type:**	white box [✓ black box]
Tester Information			
Name of Tester:		**Date:**	
Hardware Ver:	Robot 1.0	**Time:**	
Setup:	A wooden turn-table should be placed on top of the cardinal direction map. This map should be aligned with a magnetic compass. There should be no metal present while the alignment is being performed. Next, the partially assembled robot should be placed on the turn-table. The MCU should be connected to a terminal to observe and record data.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Write program to spool compass readings while simultaneously driving motors.	Program should be statically tested to verify accuracy. Should sample compass at a sufficient rate depending on speed.				
2	Run acceptance test	Test program should prompt user to turn the robot to an orientation and then run the motors up to full speed.				
3	Plot spooled data in spreadsheet program.	Plots should be analyzed to see if compass deviated any more than 10 degrees from set point.				
	Overall test result:					

Testing

- Take it seriously
  - Cost
  - Time to market
  - Health/Safety
  - Legal/Ethical
- Plan it
- Perform throughout
- Schedule it


1906-2006 FDA Centennial
U.S. Food and Drug Administration
U.S. Department of Health and Human Services
FDA Home Page | Search FDA Site | FDA A-Z Index | Contact FDA

FDA News

FOR IMMEDIATE RELEASE
P05-46
July 22, 2005

Media Inquiries:
Julie Zawisza, 301-827-6242
Consumer Inquiries:
888-INFO-FDA

FDA Announces Guidant's Class I Pacemaker Recall

The U.S. Food and Drug Administration (FDA) is notifying health care providers and patients that Guidant Corporation is voluntarily recalling certain pacemakers. A seal within the devices can leak, allowing moisture to affect the electronic circuits. This defect can cause the pacemakers to fail to provide pacing or can cause a rapid heart rate. Other unexpected device behaviors are also possible. The problems may occur without warning and can lead to loss of consciousness, and possibly heart failure and death.

Only the following models are affected by this recall. All were manufactured between November 25, 1997 and October 26, 2000.

- PULSAR® MAX Models 1170, 1171, 1270
- PULSAR Models 0470, 0870, 0970, 0972, 1172, 1272
- DISCOVERY® Models 1174, 1175, 1273, 1274, 1275
- MERIDIAN® Models 0476, 0976, 1176, 1276
- PULSAR MAX II Models 1180, 1181, 1280
- DISCOVERY II Models 0481, 0981, 1184, 1186, 1187, 1283, 1284, 1285, 1286
- CONTAK TR® Model 1241
- VIRTUS PLUS® II* Models 1380, 1480
- INTELIS II Models 1483, 1484, 1485, 1384, 1385, 1349, 1499

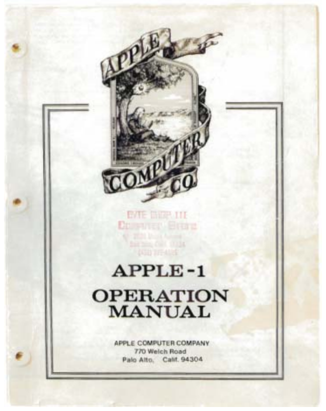
** VIRTUS PLUS II and INTELIS II models available only outside the U.S.*

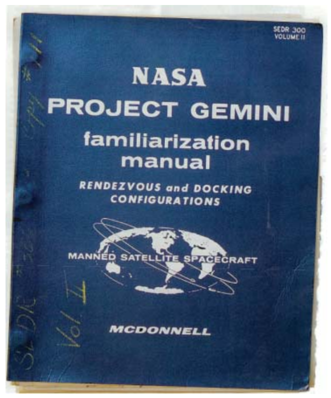
Guidant announced the initiation of a voluntary recall on July 18, 2005. The recall action consisted of a letter to physicians that describes the problem and provides recommendations about how to minimize the risk of pacemaker failure.

The FDA has classified Guidant's action as a Class I recall. Recall classifications can fall into one of three categories, with Class I being the most serious. These numerical classifications are based on the probability that the device failure could lead to adverse health effects. In a Class I recall, there is a reasonable probability that the malfunctioning device will cause serious adverse health consequences or death.

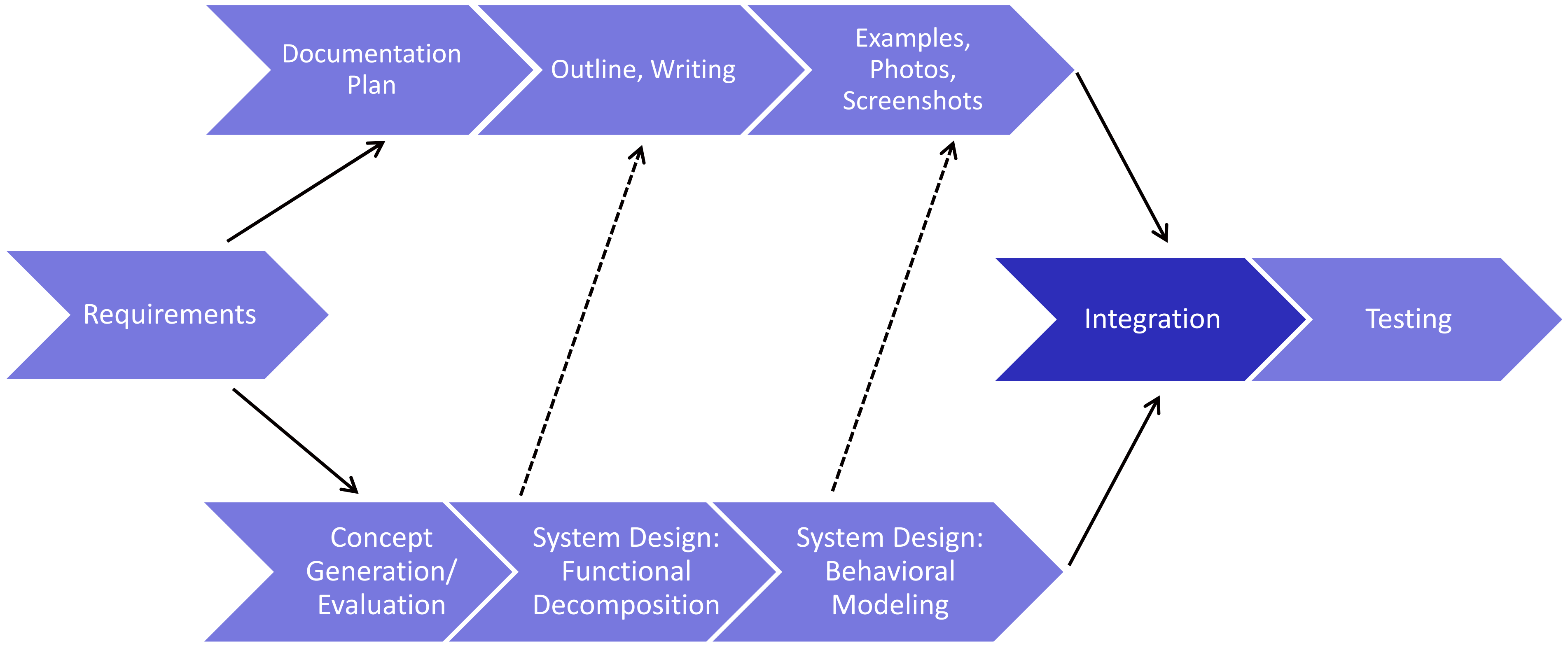
Documentation Plan

- Who are the "users" of the product?
- What are their needs?
  - Content
  - Type (installation, reference, user, tutorial, training, maintenance)
  - Method of delivery
    - Printed, PDF, CD, Video, WWW
    - Browse-able/searchable?
  - Internationalization needed?
- ATE Example
  - Test Engineer (programming)
  - Operator (operation)
  - Test Technician (interfacing, probers/handlers, data formats)
  - Product Engineer (troubleshooting, yield enhancement)
  - Service/Maintenance engineer (installation, PM, troubleshooting, calibration)





Documentation

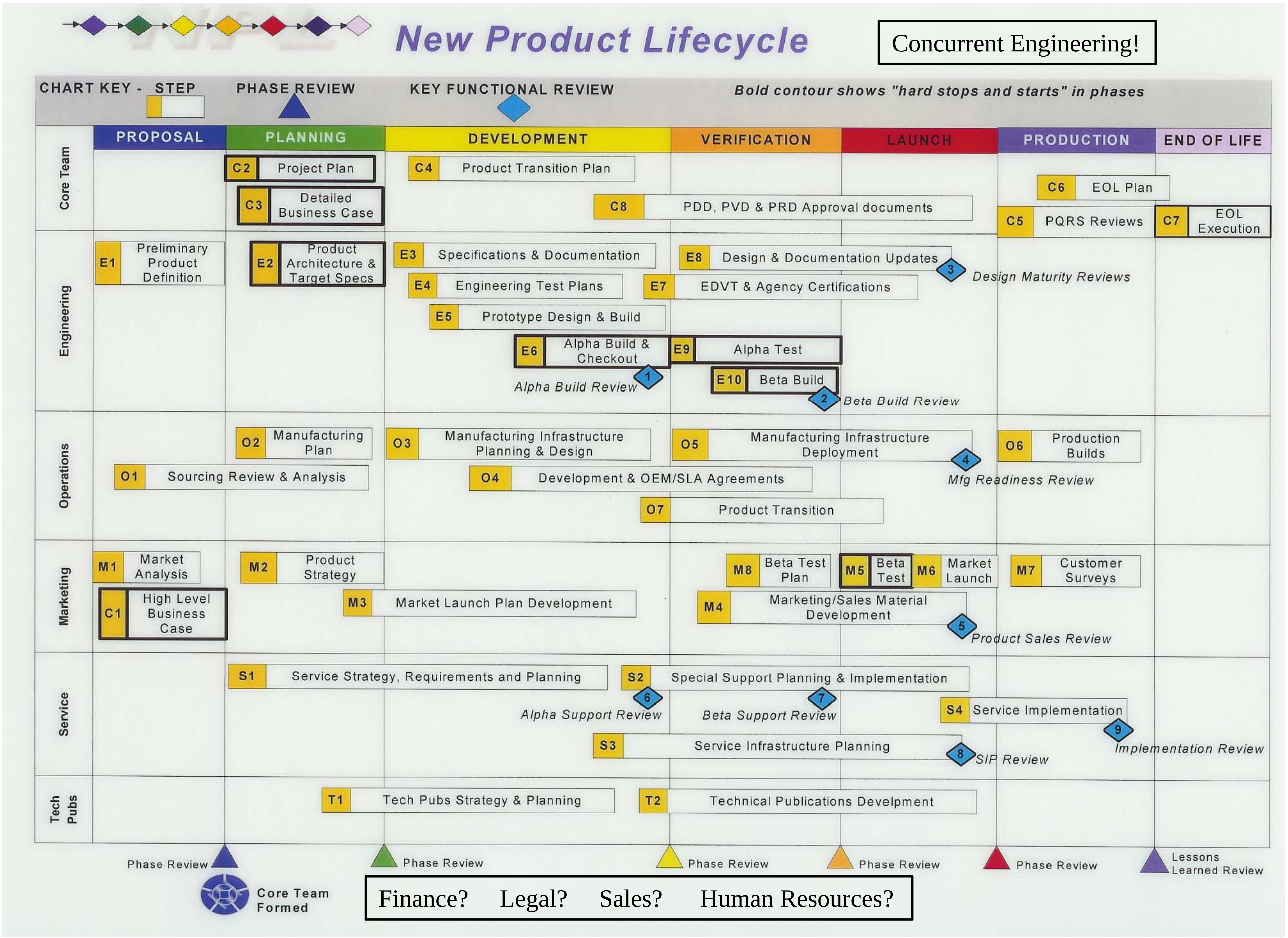

Documentation Plan
Outline, Writing
Examples, Photos, Screenshots
Requirements
Integration
Testing
Concept Generation/ Evaluation
System Design: Functional Decomposition
System Design: Behavioral Modeling

"Concurrent Engineering"

Service and Training Plan

- Who needs to be trained?
  - Internal
    - Applications engineers, field service engineers
  - External (Customer)
    - Installers, programmers, service/maintenance engineers, users
- What tools, instrumentation, spares are required
  - On site
  - Depot
  - Field offices
  - Home office

Concurrent Engineering!


New Product Lifecycle
CHART KEY - STEP
PHASE REVIEW
KEY FUNCTIONAL REVIEW
Bold contour shows "hard stops and starts" in phases
PROPOSAL
PLANNING
DEVELOPMENT
VERIFICATION
LAUNCH
PRODUCTION
END OF LIFE
Core Team
C2 Project Plan
C3 Detailed Business Case
C4 Product Transition Plan
C8 PDD, PVD & PRD Approval documents
C6 EOL Plan
C5 PQRS Reviews
C7 EOL Execution
Engineering
E1 Preliminary Product Definition
E2 Product Architecture & Target Specs
E3 Specifications & Documentation
E4 Engineering Test Plans
E5 Prototype Design & Build
E6 Alpha Build & Checkout
1 Alpha Build Review
E8 Design & Documentation Updates
3 Design Maturity Reviews
E7 EDVT & Agency Certifications
E9 Alpha Test
E10 Beta Build
2 Beta Build Review
Operations
O2 Manufacturing Plan
O1 Sourcing Review & Analysis
O3 Manufacturing Infrastructure Planning & Design
O4 Development & OEM/SLA Agreements
O5 Manufacturing Infrastructure Deployment
4 Mfg Readiness Review
O7 Product Transition
O6 Production Builds
Marketing
M1 Market Analysis
C1 High Level Business Case
M2 Product Strategy
M3 Market Launch Plan Development
M8 Beta Test Plan
M5 Beta Test
M6 Market Launch
M4 Marketing/Sales Material Development
5 Product Sales Review
M7 Customer Surveys
Service
S1 Service Strategy, Requirements and Planning
S2 Special Support Planning & Implementation
6 Alpha Support Review
7 Beta Support Review
S3 Service Infrastructure Planning
8 SIP Review
S4 Service Implementation
9 Implementation Review
Tech Pubs
T1 Tech Pubs Strategy & Planning
T2 Technical Publications Develpment
Phase Review
Phase Review
Phase Review
Phase Review
Phase Review
Lessons Learned Review
Core Team Formed

Finance? Legal? Sales? Human Resources?