Lecture 9 – Design for Test [LVTraining_2011]

Prelims

- Discuss CLAD Review
- Discuss CLAD Registration (TLG)
-

Overview

Part 1 – System Design and Analysis

1. *Design for Test – Overview*
2. *Design for Test - Principles*
3. *Problem Specification*
4. *Design Specification*
5. *Translating to Code Pt. 1 – Functional Specification*
6. *Translating to Code Pt. 2 – Organization into Modules & Interface Specification*
7. *Pt. 3 – Identification of Risks and Hazards*
8. *Selected Design*
9. *Analysis*

Part 2 – System Implementation

1. *System Code Architecture*
2. *GUI / Control Thread*
3. *Bath Thread*
4. *Rack Thread*
5. *Log Thread*
6. *Status Thread*
7. *Test 1 Thread*

Suggested Reading

A. LabVIEW For Everyone

- P.x-y ()
- P.x-y ()
- P.x-y ()
- P.x-y ()

Part 1 – System Design and Analysis

We'll start with the problem specification, digest and internalize it and then translate it into a design specification (i.e. our interpretation of what needs to be built). We'll then take the intermediate steps to turn this 'spec' into an outline of something we can actually build, and then analyze this definition.

In Part 2 we'll then take the steps to implement it in actual code, staying aware of the consequences of each implementation decision we make along the way (i.e. should the thread abort on an error, or try to recover? When do we turn off the heater? Etc.

I. Design for Test – Overview

II. Design for Test - Principles

- The simplest design wins
- Flexibility in implementation can be a valuable asset but usually at the tradeoff of extra complexity
- Complexity kills any good application
- The longer a system must last the more important the front-end design becomes. Especially against lock-in.
- Considering the lifetime of the application is important and necessarily changes how you implement your solution.
- Uniformity and design patterns
- If the code exists, but no documentation exists, the system doesn't exist. Period. Design *for* Test, not *after* Test.

III. Problem Description

- **Customer:** Joe Kaizen, brazen, pioneering boss of process flow Q&A.

Bench #19 on the **superWidget** Assembly Line can save 30 minutes if automated, and the boss assigns you to the task. There are two stages, *a* and *b* (denoted #19a and #19b, respectively). There is one *operator* for the bench which holds the following equipment:

- Controlling Application and PC: Application that aids the operator in automating the process and records.
- Temperature Bath: A 100C temperature bath which preheats the devices
- Test Harness: A test harness which monitors the widget and performs the testing
- Test Rack: A special rack where devices are placed after heating in the bath.

. The process flow for this bench goes as follows:

1. Operator receives incoming **superWidget**, and then connects it to a *testHarness*
2. The operator records the device ID and then inserts into a 5min temperature bath
3. Operator monitors devices in the temperature bath, and removes devices after 5 min when alerted by the controlling application
4. After removal from the temperature bath the operator moves the **superWidget** to a test rack where a series of I-V tests are executed via the *testHarness*
5. Operator monitors devices on the test rack, and removes the devices when prompted by the controlling application. Devices that pass the tests are then transferred by the operator to Bench #20. Devices that fail the test are moved to a remediation pile, where another worker will come collect them for analysis and repair.

IV. Problem Specification

Joe got excited and fabricated the entire bench for you, it's already in place. Joe needs you to develop the controlling application using LabVIEW to allow for control of the device flow into and out of the temperature bath, monitoring of device's in each stage, logging of device results, and general process flow instructions and enhancements to aid the user. See Use Cases and Functional Specification below for more details.

Use Cases

A. Actors

- Controlling PC
- Operator
- Device
- Bench #18

B. Use Cases

- Operator receives a new device from Bench #18 and inserts it into the system while also logging it into the controlling application. Device is placed into temperature bath.
- A device has sat been in temperature bath for 5 min and needs to be removed. This is indicated to the operator.
- Operator removes device from temperature bath and transfers to the test rack, logging into Controlling Application.
- A device completes the Test Rack test and needs to be removed. This is indicated by the controlling application to the Operator.

C. Exceptions

- Temperature Bath Failure
- Operator Absence
- Device Removed Prematurely or Lost
- Invalid Operator Inputs into the System
- Repeated Device

V. Design Specification Pt. 1 – Functional Specification by Module

A. General Organization

GUI Thread which will serve as application control. Will have a globalControl queue which it periodically checks to see if someone requested a shutdown. After shutdown of any thread the GUI thread is signaled to exit. After exit, the GUI thread then signals all other threads to exit.

- System Initialization
- System Shutdown
- Control of Temperature Bath
- Control of Test Rack
- Execution of Tests
- Logging of Data
- System Control
- UI

VI. Design Specification Pt. 2 – Organization into Modules

A. Operator Interface

- Standardized Interface (UI) with intuitive layout and clear communication and signaling
- Ordered Flow, Organized Layout
- Single Interface for entire bench
- No task shall block any of the major control or monitoring stations (e.g temperature bath or rack)

B. Temperature Bath

- Initialize, Maintain, and Shutdown Bath's Heating Element
- Allow input of devices into bath
- Monitor devices in bath at a rate no less than 10 Hz
- Indicate Device Status in Bath to Operator
- Cause system to abort on the case that a device overheats
- Maintain active record of all devices in bath

C. Testing Rack

- Transfer Devices from bath onto rack
- Execute Tests on Device
- Indicate Test Status to Operator
- Provide Operator Interface to Remove Device from Rack

D. Testing Harness

- Provide unambiguous, secure connection to device
- Allow connection of multiple devices at same time

E. Device Data Logging

- Maintain record of active devices on bench
- Input new values for devices on bench
    - Device Name / Date / Operator Date
    - Bath Temperature and Dwell Time
    - Test Status and Test Data
- Record device test information to file after completion of tests
- Provide easy interface for other threads to throw data to it

F. Status Reporting

- Provide easy interface for other threads to write to the status bar on the GUI.

G. System Control

- Handle Operator UI interaction
- Handle Operator action requests (e.g insert into bath), delegate actions to other threads
- Handle global shutdown of all threads

VII. Design Specification Pt. 2 –Interface Specification

This system presents a large amount of inter-thread communication, and thus it is critical to start implementation with the inter-thread communication clearly defined.

- This is where you would define the communication channels between each thread, along with corresponding datatypes and message definitions
- General Words of Wisdom: Make your interface here flexible for the addition of future fields into the messages!

A. Control->Bath Messaging

B. Bath->Rack Messaging

C. Status Messaging

D. Datalog Messaging

E. Test 1 Messaging

F. Test 1->Rack Messaging

VIII. Design Specification Pt. 3 – Identification of Risks and Hazards

A. Temperature Bath Failure

Status: Handled with safety shutdown procedure

B. Operator Absence

C. Device Removed Prematurely or Lost

D. Invalid Operator Inputs into the System

E. Operator Aborts the Process or Attempts to Restart a Step of Process

Status: Not Handled or Designed Against

Analysis: This is a tough, complicated and messy scenario to design against. We choose not to design against until either (i) the customer requires it or (ii) operator's are attempting to restart in a particular step often. Handlng restarts for every step of the way would be ideal, but incredibly complex and hard to manage. This is one of those code lock-in topics.

F. Repeated Device

IX. Selected Design

<This should be a very detailed section…>

X. Analysis

A. Preventing Code Lock-In

B. Complexity Balance

C. Unexpected Use Cases

D. Does design support potential expansion into future needs?

- Only allow the user to cancel if all line is empty
- Also perform a shutdown routine.
- Only let devices with unique IDs into the queue.

E. Is Design Testable?

Part 2 – System Implementation

XI. System Code Architecture

Each module will be designated with its own thread, and queues will serve as the communication between each. Slave threads will generally follow the producer consumer model, and handle actions on T.O. while waiting for new messages.

F. VI Hierarchy

The VI Hierarchy 'View->VI Hierarchy' is an useful tool for viewing the linking of your code. Below is a snapshot from our project today.

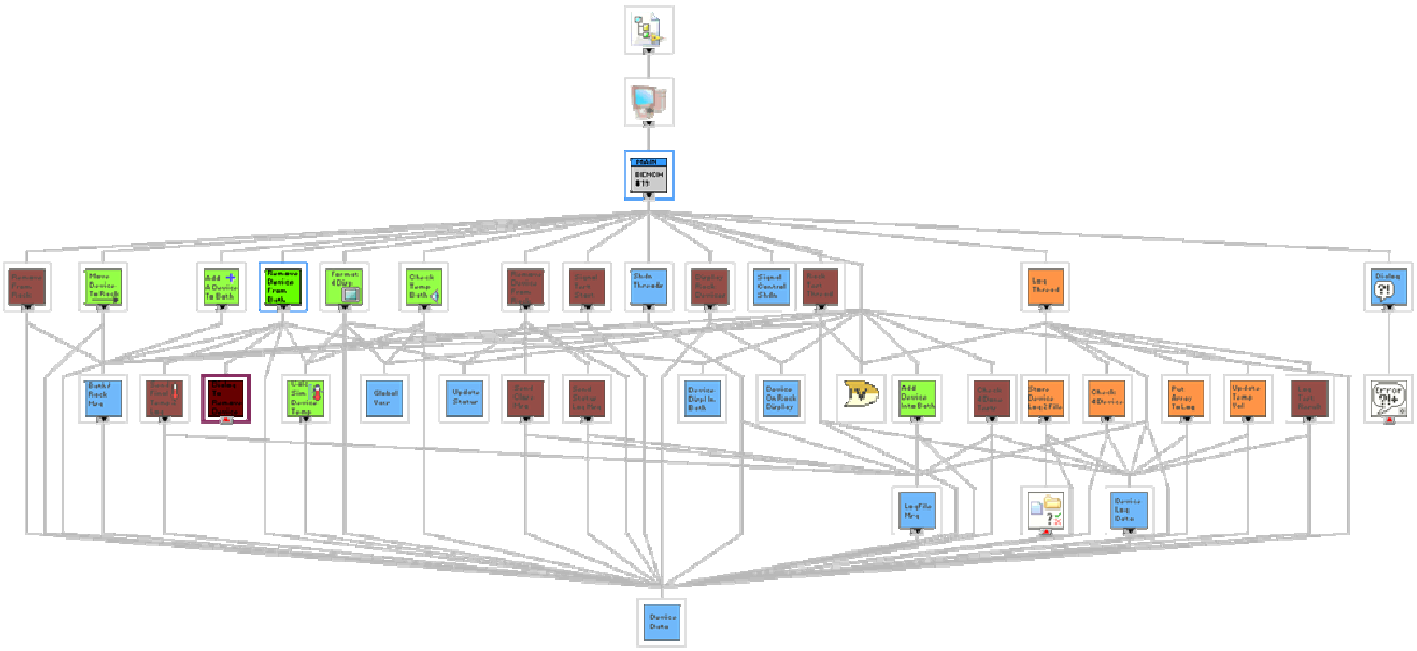

Figure 1: *VI Hierarchy for the project easily shows function dependancies. Also color coding your modules also can quickly show these dependencies too.*

XII. GUI / Control Thread

XIII. Bath Thread

XIV. Rack Thread

XV. Log Thread

XVI. Status Thread

XVII. Test 1 Thread

XVIII. Error Codes

- 5000: Device was over temperature