Concurrent Engineering for Automatic Test Station Development

James C. Lisonbee
Software Engineering Division (TISAD)
Ogden Air Logistics Center
Hill Air Force Base, Utah 84056
801-775-4442
LisonbeJ@software.hill.af.mil

Kenneth C. Chisolm
Software Engineering Division (TISAD)
Ogden Air Logistics Center
Hill Air Force Base, Utah 84056
801-775-4445
ChisolmK@software.hill.af.mil

Mano Sithivong
Software Engineering Division (TISAD)
Ogden Air Logistics Center
Hill Air Force Base, Utah 84056
801-775-4502
SithivoM@software.hill.af.mil

Abstract - During the planning stage for development of a new test station or upgrade, one becomes very aware that there are many tasks to accomplish in a short period of time. It is often important to maintain a short development cycle while accomplishing both hardware and software tasks such as hardware selection and driver development. Concurrent Engineering allows for a simultaneous activity of hardware and software personnel during test station development. This paper will discuss the application of concurrent engineering during development of the F-16 Analog Test Station Sustainment (FATSS) project and how these principles can easily be applied to other development efforts.

INTRODUCTION

The F-16 analog depot test stations are in critical need of modification in order to maintain repair capability for the F-16 aircraft. Most items in the test stations are both obsolete and unsupportable. As vital parts in the test stations fail, the stations will be down indefinitely.

The FATSS project came about as a result of the need to support the F-16's life cycle through the year 2020. The goal of the FATSS Project is to re-establish supportability and to maintain depot repair capability.

When a project is first looked at, several problems exist. One of the most important problems during development of a new test station is the limited time to complete the entire task. Another problem occurs when there are few people available to work on the project. A well thought out and documented development plan is needed to schedule tasks.

During the development plan, Concurrent Engineering principles can be used to task simultaneous activity between hardware and software personnel. This coordinated effort can drastically reduce developmental time.

PRELIMINARIES

There are a couple preliminary tasks that need to be accomplished before the bulk of the project can be started. These tasks can be accomplished at the same time utilizing both hardware and software personnel to their fullest capacity.

These tasks include hardware and software selection. Be prepared for the hardware selection phase as described below to take much longer than the software selection task.

Hardware Selection

When making the decision of what hardware needs to be replaced in the new test station, it is a good idea to remember the life cycle of the test station. Because of this, the FATSS project team wished to use Commercial Off The Shelf (COTS) instruments. We desired to use as much VXI equipment as would meet or specification instead of using GPIB instruments that may not be supported into the new millennium. Many vendors that were called mentioned that their GPIB instrument would only be supported for a couple more years. This poses significant discouragement from choosing their instruments.

Our hardware selection was based on Critical Item Product Function Specifications. Where the hardware specifications were not met, design philosophies in the software were needed to bridge this gap. In the case of a voltage peak to peak we could not find a single instrument that would measure these voltages across the entire range required by the FATSS hardware specification.

To solve this problem, we would modify instrument-handling subroutines within the translator to assign voltage peak to peak requests to hardware with sufficient capabilities. If a single instrument becomes available later to perform the task, we could make a change to a single subroutine in the Translator and be prepared to move on.

The coordination of the hardware and software personnel regarding availability of software drivers facilitated in software tools selection. Once instruments were found that met our specifications, we narrowed our selection to those that included instrument drivers and electronic manuals. If these were not readily available we asked that they be provided in a format that met our software needs. Electronic manuals would be used for an Electronic TO.

Once the FATSS group decided on test station replacement hardware a comprehensive conversion plan was developed. Hardware was ordered and purchased for our prototype. This gives us a way to verify that the hardware really met the design specifications of the vendor.

The FATSS hardware selection team was always mindful of having suitable substitutes available for every instrument possible. We did find some instruments that did not meet vendors specifications. For these cases we would turn to our suitable substitute and go through the same process of verification of vendor specifications for it.

Software Tools Selection

When choosing software, the FATSS software design group found it best to use COTS software. This saves the design team from writing their own software that is readily available.

The selection of software is imperative to accomplish at the beginning of the development phase. This decision drives the remaining software design throughout the project.

Once the software selection is complete, the software selection team can begin working on the software tasks.

SOFTWARE TASKS

Once software personnel are moved to the software tasks, the most important task to begin is the creation of the Interface Control Document (ICD). While many software personnel are tasked to create this document, additional personnel may begin writing or modifying a Test Executive.

Interface Control Document

The ICD allows driver development and translator development to happen simultaneously with minor changes. This task can take place while hardware selections are still being made.

The ICD is simply a means for mapping the output of the Translator to the input of the drivers. The software team must design a common interface that the Translator can produce so the drivers can easily gather needed information from.

The FATSS software design team chose a common interface to be used with every driver that was to be written [1]. The translated code calls a single public access function within a given driver. This common interface to the public access function allows for variable amounts of information to be passed in. To facilitate closing multiple relays we need only make one call to the relay driver rather than multiple calls to the same driver.

The ICD is the key for development of the Translator and the Drivers. The software design team must take a look at the legacy code and become familiar with it from an instrument point of view. The ICD is developed purely from instrument parameters given in the legacy code. There is no other place to get these parameters. It is easy to see how to develop the ICD parameters because ATLAS parses nicely to legacy hardware.

The desired data to be passed to the software driver was placed in the ICD in a FATSS standardized format. This format allowed multiple software personnel to design pieces of the ICD specific to the hardware they were in charge of.

Another purpose of the ICD is to specify what the translated run-time code will use as parameters to pass to the Test Executive. With this information the Translator can produce code that is needed to interface with the Test Executive.

Translator Development

Converts legacy code to a generic and specified format from the ICD. The Translator can push all the input/output concepts into driver public access calls. These calls can be written to facilitate windows messaging to the Test Executive [2].

Driver Development

It is imperative that the software personnel coordinate activities with the hardware personnel. Before drivers can be completed, hardware must be selected. To test the drivers it is best to have hardware on loan from the vendor or already purchased.

When following a completed ICD for a given driver, development can be done quickly. The ICD allows for efficient coding of the driver. The inputs are specified in the same format for each driver making it possible to find similarities between drivers that allow for copy and paste or code reuse.

The driver is responsible to take the specified list of parameters and convert them to something that the vendor's supplied drivers can recognize. This conversion process is done in two layers of the developed driver [1]. The public access function simply moves information from parameters that are passed in to a function call that we call the hardware driver layer. This layer then organizes its parameters to call vendor supplied driver's function calls.

Information passed to the driver is necessarily limited to what the legacy code had as parameters. For this reason, all parameters passed to the hardware driver layer that are not available to the public access function must be hard coded using default values. The driver developer needs to be aware of this possible gap. An example of this is when a dc-offset is not present in the legacy code. The vendor's supplied driver may need a dc-offset as a parameter to one of its function calls. The driver developer must supply a default value of 0.0 for this case.

Hardware initially selected for a given task may not work properly. Do not worry! Driver development using multiple layers has a limited impact when hardware is changed. It may take only a few days to swap out a function generator. It could possibly take a couple of weeks to swap out to a poorly documented piece of hardware.

Test Executive

The Test Executive is the operator's interface to the test station. We were able to take portions of legacy hardware and implement it in software.

Many Test Executive decisions can be made prior to the ICD being complete. The operator's interface can nearly be completed without much of this information.

The Translator and Test Executive need only understand what kind of objects they are writing to and reading from. This is the information provided in the ICD that is needed.

OTHER TASKS

These other tasks can be worked on mostly after hardware selection is made. System documentation should be developed throughout the project.

Cable Design and Build

Wiring diagrams can be worked on without a complete hardware selection with the knowledge that connections are to be determined. This is not recommended since it may cause a lot of rework. It would really be best to wait until the hardware is selected.

There are two main problems that exist in cabling to accomplish the task in a timely manner. First, the design of the cable needs to be done as soon as the hardware is selected. Second, the building of a test cable for driver testing should be done as soon as the hardware is known.

A cabling group can be created or utilized to build a mock-up of the test station. This can be expensive in the use of manpower, but will aid in integration of instrument placement. Remember that extra connectors will need to be purchased for a mock-up to work properly. If your group desires to use the prototype as the real life mock-up then these connectors are not needed. You may need to coordinate with the people that have the instrument in their possession at times to measure for cable lengths and various other things. Using the prototype may prolong the time for cable build for this reason.

Once instruments are placed whether on the prototype or mock-up, cable lengths can be determined and documented for later use in kit proofing.

System Integration

During system integration placement of the instrumentation is finalized, cabling is completed, testing instruments through the translated code using developed drivers is validated & verified and all the design ideas fall apart on you.

Be prepared during this phase to modify everything imaginable to facilitate making the hardware and software work together as planned.

System Documentation

This can be accomplished during all phases of development. When people become available to work on documentation, they can gather information needed for creating things like an Electronic TO. Pay attention to these electronic manuals during hardware selection.

Remember that documentation takes considerable amounts of time. If you can create a process to document as you go you will be time ahead.

Kit Proofing

Once all of the tasks are complete, kit proofing can be completed. The kit proofing task can be started during the cable build task and when system integration is close to completion. As with system documentation, document as you go.

DETAILS

When looking at various aspects of your project, remember that the key to success is to follow a process that enables you to multi-task. Concurrent Engineering through coordinated efforts of both software and hardware personnel aids in completion of the task on schedule and on budget.

REFERENCES

[1] L. Vuu and A. Khan, *Instrument Driver Design,* 1999 IEEE

[2] J. Evans, J. Lisonbee and L. Allred, *Using Windows Messaging to Control Automatic Test Equipment*, 1999 IEEE