TOWARDS FULLY AUTOMATED INSTRUMENTATION TEST SUPPORT

Charles H. Jones, PhD
Edwards Air Force Base, CA
charles.jones@edwards.af.mil

ABSTRACT

Imagine that a test vehicle has just arrived at your test facility and that it is fully instrumented with sensors and a data acquisition system (DAS). Imagine that a test engineer logs onto the vehicle's DAS, submits a list of data requirements, and the DAS automatically configures itself to meet those data requirements. Imagine that the control room then contacts the DAS, downloads the configuration, and coordinates its own configuration with the vehicle's setup. Imagine all of this done with no more human interaction than the original test engineer's request. How close to this imaginary scenario is the instrumentation community? We're not there yet, but through a variety of efforts, we are headed towards this fully automated scenario. This paper outlines the current status, current projects, and some missing pieces in the journey towards this end. This journey includes standards development in the Range Commander's Council (RCC), smart sensor standards development through the Institute of Electrical and Electronics Engineers (IEEE), Small Business Innovation Research (SBIR) contracts, efforts by the integrated Network Enhanced Telemetry (iNET) project, and other projects involved in reaching this goal.

KEYWORDS

Test Automation, Standards, XML, Wireless Communications, Networks

INTRODUCTION

Here is an ideal end state scenario that many in the instrumentation field are working towards. A test vehicle shows up to be tested. The ground station equipment automatically interacts with the vehicle to set up data acquisition requirements. The test is conducted with required data telemetered to control rooms and recorded on board. After the test, data is automatically distributed to the appropriate test engineers. From an instrumentation point of view, the only involvement of the test engineer is the original input of data requirements and reception of the data after the test. This paper looks at where we are at in terms of realizing this scenario. It will do so by looking at general technology trends as well as referencing specific projects that the author is involved with or knows enough about to discuss. (There are certainly other projects related to this scenario.) This paper will concentrate on this specific test setup scenario, but the topics and technology discussed here do not exist in a vacuum Not addressed will be such things as how the sensor suite and data acquisition system (DAS) were initially designed and installed on the test vehicle. Nor will things such as real-time interaction between engineers and the vehicle nor data archiving and retrieval be addressed. All of which are necessary for complete test support.

Figure 1 will be used throughout the paper to discuss technology related to the scenario. The bolded boxes and lines in figure 1 represent the basic instrumentation structure from sensor or bus through the DAS and transmission to ground stations and engineers. The dashed boxes represent projects. (Acronyms will be spelled out as they are discussed and are listed at the end of the paper.) The lines between elements of the basic instrumentation structure are lines and not arrows for a reason. As of today, the flow of data is almost strictly from sensor to recorder or ground station. However, in order to achieve our ideal scenario, these lines of communication will have to be bidirectional.

The discussion will utilize three general concepts: Connectivity – how do the pieces physically talk to each other? Abstraction Languages – how is the data described so the different pieces know what is being transferred and what each piece can do? And Requirements Management – how do the pieces satisfy the end user? For each of these areas, I will list what I think are the key elements along with my opinion of their developmental status, and then discuss each element in more detail.

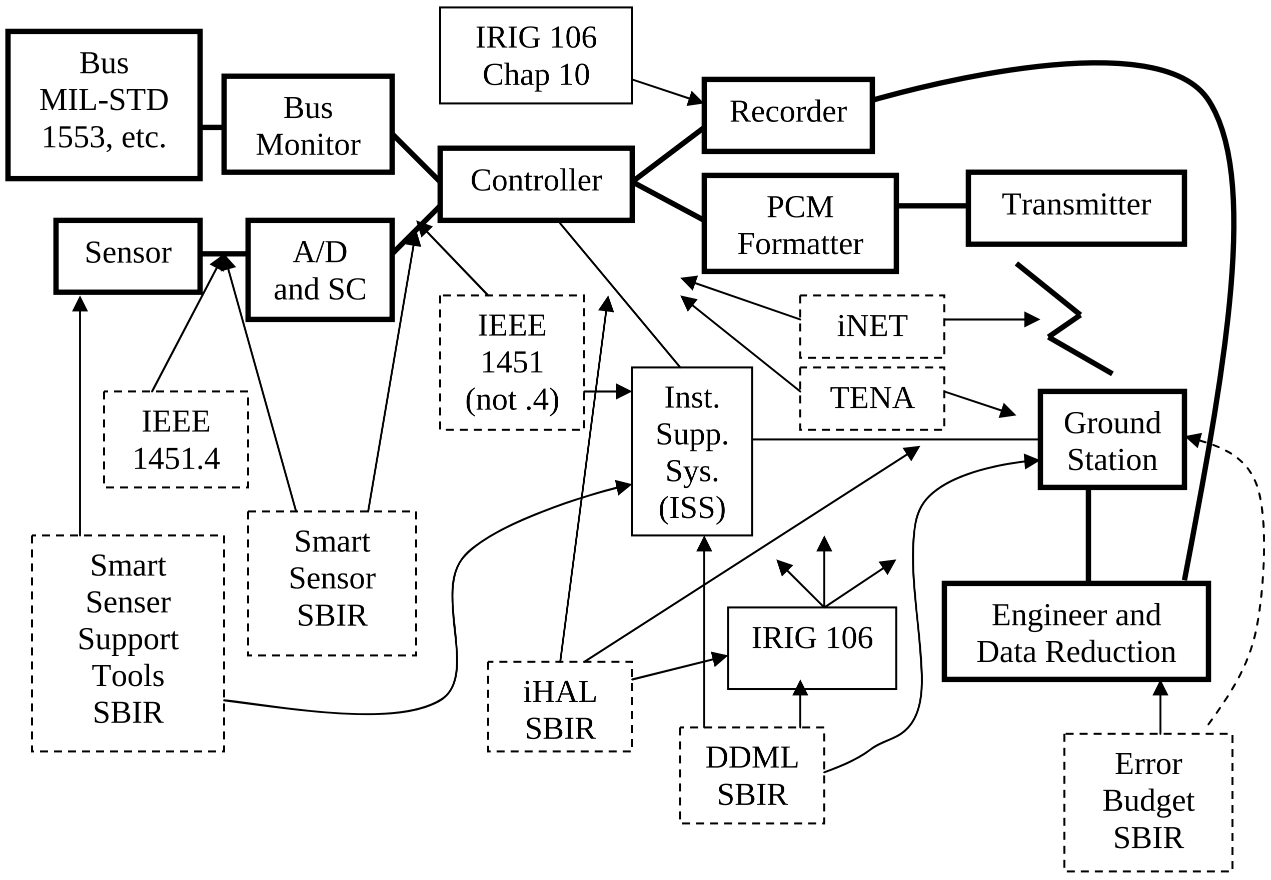

Figure 1 Basic Instrumentation Structure and Related Projects

CONNECTIVITY

In a word, connectivity in this context comes down to: network. The objective scenario requires that every piece of equipment be able to communicate with every other piece of equipment. Or, at the very least, know what capabilities are present on the vehicle and on the ground. This includes everything from high-level controllers down to individual sensors. Being a little more specific, the following elements are included:

1. The network – especially uplink. *Status*: being worked on, deployment will take a while.
2. Smart, network-enabled instrumentation. *Status*: being worked on, awaiting the network.
3. Smart sensors. *Status*: emerging in industry, need to adopt.
4. Middleware. *Status*: some exists, awaiting smart instrumentation.
5. Support systems. *Status*: expect continued adaptation and upgrade as the above mature.

The current data acquisition architecture is very much a one way street. Data flows from sensors and buses to the recorder and ground stations. Our scenario requires the ability to wirelessly communicate from the ground stations to the vehicle. The integrated Network Enhanced Telemetry (iNET) project [1] is the vehicle through which this network communications link is expected to be developed. However, there also needs to be a network on the test vehicle connecting all the data acquisition devices as well. Both of these network elements will require modifying the vehicle at some level. Test articles are not usually taken out of service just to update the instrumentation and such upgrades tend to be piecemeal. There are still test articles with 30-year-old test equipment. There is no reason to think that 100 percent deployment of network technology will move any faster than replacing previous generation equipment has moved.

In order to query an instrument over a network, the instrument must be network enabled. It must also have some level of intelligence to respond to the query. Adding an Ethernet card to an instrument is not difficult, but you need to know that it is Ethernet that the instrument is going to be connected to. Further, until you start having network functionality, it is difficult to implement higher order functionality in a device. In this sense, implementing a network structure is an enabling technology that will allow manufacturers to increase the functionality of individual devices.

It would be possible to lump smart sensors under smart instrumentation and, ultimately, we may view sensors as simply nano-sized instruments. However, sensors perform a distinctly different function from, say, a controller or bus monitor. They are also more ubiquitous and more distributed. The Institute of Electronics and Electrical Engineers (IEEE) 1451 family of standards is emerging as a leading nonproprietary smart transducer standard. (The IEEE 1451 working groups discovered there are over a 100 proprietary, or at least corporate, smart sensor standards.) The IEEE 1451.4 standard [1], which allows communication to the transducer in either digital or analog modes, has been approved for several years and is taking off in industry. The IEEE 1451.0 [3], defining common commands and common transducer electronic data sheets (TEDS), and the wireless version, IEEE 1451.5 [4], have recently been approved and have strong industry support as well.

A critical part of our scenario is to retrieve TEDS-like information from every sensor on the vehicle. As the vehicle networks mature, the Test and Evaluation (T&E) community should plan on adopting industry standards for smart sensors. Towards this end, I have initiated several Small Business Innovation Research (SBIR) projects. A current SBIR effort by NVE Corporation [5] is developing hardware to interface IEEE 1451.4 compliant sensors with the Common Airborne Instrumentation System (CAIS). A SBIR effort by Nomadics Corporation [6] is developing smart sensor support tools. Although smart sensors will make everybody's life easier, they force a change in operational scenarios. It is necessary to develop new support tools to fully realize these benefits.

Just because a device has some intelligence, doesn't mean it can talk to every homebody that comes along. Developing middleware that standardizes instrumentation interfaces is a solid approach to enabling wide spread communications. The Test and Training Enabling Architecture (TENA) project [7] continues to develop this type of middleware. Although TENA has been used more by the training community than the T&E community, expect TENA to develop these key interface modules as iNET and related technologies mature.

Currently, I make a distinction between Instrumentation Support Systems (ISS) and Ground Support Systems (GSS). An ISS may do some of the function of a GSS but is more focused on "touching the vehicle". Alternatively, a GSS is generally in a control room with intent to support many engineers real time during a test. That is, an ISS configures the DAS on the vehicle and communicates that configuration to a GSS. The GSS is then the central system during the test itself. At Edwards, Iliad [8] is a major ISS that provides multi-vendor instrumentation support. Similarly, the Interactive Analysis and Display System (IADS) [9] is the main GSS in the control room. Most vendors provide a basic level of support for these functions. However, our scenario requires vendor-neutral support. Our scenario will also cause the distinction between these types of systems to blur. These types of systems have necessarily upgraded to support new technology. There is no reason to think they won't continue to evolve as necessary, but they need the new technology in place, or at least well defined, before doing so.

ABSTRACTION LANGUAGES

Metadata is data about data. An abstraction language is the language that metadata is written in. We can take one more step and point out that eXtensible Markup Language (XML) schemas are becoming the default mechanism for describing abstraction languages. For example, we might have a datum of "25". In order to describe the meaning of this datum we need to know the metadatum "degrees Celsius." The concept of "unit" is part of the abstraction language that encompasses "degrees Celsius" and other units. Within an XML schema, a tag "<Units>" might be used to encapsulate the abstracted concept of "unit".

About a decade ago, Lee Gardner and I discussed what abstraction languages were needed for data acquisition. The list has grown some from that time, but that original list allowed instigating some of the projects in Figure 1. My current list (for data acquisition) is:

1. Data maps. *Status*: in good shape, need to add network protocols.
2. Data displays. *Status*: in good shape, will continue to need improvements.
3. Hardware. *Status*: being worked on, will always need improvements.
4. Physical units. *Status*: standards exist outside T&E, need to adopt.
5. Project configuration. *Status*: every ISS does this, need a standard.
6. Derived measurand algorithms. *Status*: every ISS does this, need a standard.
7. Measurand naming. *Status*: not even close to existing.

Data comes packaged in myriad formats: PCM, MIL-STD 1558, IP, etc. A data map describes the data in one of these packages. The Telemetry Attributes Standard (TMATS) in IRIG 106 [10] is a prime example of a data map abstraction language. It allows describing PCM formats and other data maps. In general, these types of abstraction languages are in good shape because you have to have the mapping in order to use these protocols at all. In particular TMATS is a very mature abstraction language which has been recently updated to an XML schema and certainly Chapter 10 of IRIG 106 is a solid example of a higher level description of data packaging since it describes recorded data with extreme detail. Within our context of moving to a networked telemetry environment, there are some new elements, such as IP (internet protocol), that need to be added to our existing abstraction languages. The iNET Metadata task [11] is doing exactly that.

There are a variety of systems that allow display of data. (By display we mean via strip charts, bar charts, dials, etc.) Within our scenario, one of the things that could be retrieved from the DAS is a recommended data display format. This would require a standard abstraction language that could be understood by disparate display systems. The RCC Telemetry Group has recently adopted the Data Display Markup Language (DDML) [12] developed under a SBIR by Knowledge Based Systems, Inc. (KBSI) as part of IRIG 106. This puts this language in good shape. Although, the DDML is not a 100 percent solution and we should expect enhancements over the years.

As part of our scenario we need to be able to retrieve a complete description of the hardware making up the DAS. This would be helpful in troubleshooting, understanding the limits of the system, and, post test, possibly providing the ability to track down data errors. KBSI is working under another SBIR to develop the instrumentation Hardware Abstraction Language [13]. They are working with the RCC TG and iNET Metadata working group to ensure consistency between related efforts and to work towards iHAL's inclusion in IRIG 106 in the future.

Developing a standard for describing physical units is a non-trivial undertaking. The International Standards Organization (ISO) has a method for defining all physical units in terms of nine base units. This is fine and dandy, but things get a little complicated when you add in dimensionless units such as Mach number, units such as "seconds per second", and the fact that not every one uses the metric system. The IEEE 1451, as part of fully describing transducer characteristics, has developed a method that starts with the ISO method, adds a little bit, and allows for conversion from ISO units into user defined units. It would be reasonable to adopt this, or some other equivalent approach, rather than reinvent a language for physical units.

Configuration management is an everlasting battle. Things change. It is important to keep track of how the overall system is and was configured for setup and troubleshooting both during and after a test. Every ISS maintains a project configuration at some level and, in its essence, this is not difficult. But a standard language and approach is needed.

The need for standard abstraction languages might be codified by the word "measurand". I have spent many hours describing and defining what this word means to different people all of which have different definitions. The point that needs to be made here is that sensors provide some raw value representing a measurement. This raw value is then converted into engineering units (EU). This EU value is then combined with other EU values (often via an equation or algorithm) to derive another value. For our purposes consider every one of these different values a "measurand". Again, every ISS has some level of ability to describe the transformation functions and derivation algorithms. A standard approach is needed. The IEEE 1451 provides some approaches to this and there are standards such as MathML (which is XML-based) that provide a starting point for such a standard.

Part of our scenario requires that the GSS and vehicle be able to automatically determine what measurements (measurands) are to be collected during the test. Within our ideal scenario, the test engineer should be able to identify these by location and function rather than by some obscure name of a particular sensor (which is the current state of affairs.) For example, an engineer might want to say give "me temperature off the right wing tip" rather than "please collect measurement M205". Having a standard naming convention would help test engineers move between vehicles as well as aid the automation of test setup. Even though many people have told me this is a great idea, it is not clear to me how this could be done and it is very clear to me that this would be extremely contentious.

REQUIREMENTS MANAGEMENT

Fundamental questions before every test include: What data do you record on board? What data do you telemeter to the GSS? What measurands need to be derived real time? And how do you display the data? Let us be clear that the requirements *analysis* itself is outside the scope of this paper as is display development. We are assuming that test engineers know what their requirements are and that they have developed their displays. The question before us is how we automate the specification into the GSS and not require the engineer to know specifics about the DAS. I identify these main elements:

1. What to measure and where to send it. *Status*: every body does it, no standard, automation highly dependent on many elements discussed above.
2. Sensor suite displays. *Status*: some software efforts in progress, no standard, automation dependent on ability to fully query the vehicle.
3. Automated matching of requirements to sensor suite. *Status*: not even close to existing.

Although I've separated them out, the second element, sensor suite displays is potentially part of the solution to the first element, what to measure. If the support systems can provide a detailed and accurate 3-dimensinal (3D) graphic of what sensors are available and where they are, then the engineer can do a lot of pointing and clicking to identify the measurements they're interested in. Similarly, if all the displays have been developed (or pulled from the vehicle) a series of menus or graphical icons could allow drag and drop specification of what displays to use and possibly what measurements attach to what displays.

Given all the necessary inputs, developing software to implement this type of graphical interface is doable. In fact, an initial version of such an interface has been developed as part of the smart sensor support tools SBIR contracted to Nomadics Corp. KBSI (and probably other companies) also has software that can do some of this. The difficulty is in getting the required input – both the 3D graphic of the vehicle and the exact placement of the sensors on the vehicle. Even if you have access to a generic model of a given vehicle, the model probably does not contain all the modifications a specific vehicle has undergone; vehicle configuration is not static, especially in the T&E environment. Smart sensors and a network to retrieve TEDS will aid in automating the location, but the original location specification of a sensor may depend on human input – and thus introduce human error.

Another approach to measurand specification is through the measurand naming abstraction language. A test engineer may have some very standard tests that are performed and analyzed on different vehicles. It might be possible for an engineer to generically specify what measurands are required for such standard tests – without regard to a specific vehicle. Ideally, the test support systems could take such a generic specification and match it against the sensor suite as retrieved off the vehicle.

If all required measurands specifically exist on the vehicle, than automatically matching requirements to the sensor suite could be fairly easy. However, even if the engineer has used the sensor suite display approach to selecting measurands, there is more to measurand specification than location and physical phenomenon to be measured. In particular, sampling rate and accuracy are usually of concern. Thus, the automated analysis has to take into consideration whether or not the sensor can perform at the level required.

Within our scenario, where we do not have the option of adding new sensors, the chances of finding a perfect match are slim. So we would want the automated process to recommend a best fit of the sensor suite to the requirements. This might include applications of data fusion; sometimes mixing data from multiple sensors can meet requirements instead of placing a particular sensor at a particular point on the vehicle. A key aspect of such a best fit is overall system accuracy; every component in the data acquisition chain introduces error. Automatically analyzing this overall error budget is non-trivial. A SBIR to develop system-level error budget analysis software is in progress with Sysense Corp. [14]. Although I have included this in Figure 1, I have left the link to the ground station as a dashed line because the link is tenuous at this point. Although some of the ideas and software might be part of the automated best fit analysis, the current effort is of more value to post-test analysis and initial design of the DAS. In general, the complexity of this analysis points to the need for some level of artificial intelligence. I am not aware of any effort addressing this issue.

DISCUSSION

We are at the point in technology maturity where we can conceive a system which provides the seamless automated setup that is the core of our ideal scenario. Most of the technology even exists and the T&E community (especially through iNET) is in the process of implementing the core functionality required – the network. In the sense that many of the elements of our scenario are dependent on the network, implementing the network is truly an enabling technology. We need the network to advance on to more sophisticated applications. A strong example of this enablement is smart sensors. Our scenario is critically dependent on being able to retrieve current information and status directly from the sensors. The network must go down to the sensor level. Since network technology is very mature, the real hurdle at this point is agreeing on what we want to do and implementing the standards to do it.

Another way of looking at our ideal scenario is that we want to make the instrumentation invisible to the test engineers. We want the test engineers to be able to concentrate on data requirements and data analysis and not have to worry about the minute details of how to get the data. One concern in all this is that, as the instrumentation becomes more and more invisible, so do the instrumentation engineers. There is a lot of work to be done to implement our ideal scenario and there will continue to be more work to upgrade instrumentation systems after we reach that point. Let us not forget that just because you don't see it, doesn't mean it isn't critical to the job and doesn't mean that a lot of people didn't put a lot of effort into automating a complex system.

REFERENCES

[1] Kip Temple and Daniel Laird, *An Airborne Network Telemetry Link*, Proc. International Telemetering Conf., Vol. XXXXII, (2006) Paper 06-08-02.

[2] IEEE Std 1451.4, "Draft Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats." Institute of Electrical and Electronic Engineers.

[3] IEEE 1451.0 (IM/ST) "Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats," Institute of Electrical and Electronic Engineers.

[4] IEEE 1451.5 (IM/ST) "Standard for a Smart Transducer Interface for Sensors and Actuators - Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," Institute of Electrical and Electronic Engineers.

[5] R. Sinclair, R. Beech, K. Jones, and C. Jones, *Old Sensors Mix It Up With IEEE P1451.3 & .4 Standards*, International Test & Evaluation Association Workshop, Ridgecrest, CA, (2003).

[6] Mac Reiter and Charles H. Jones, *Human and Workflow Issues With Smart Sensor Networks*, Proc. International Telemetering Conf., Vol. XXXXI, (2005) Paper 05-21-04.

[7] Gene Hudgens, *DoD Range Interoperability and Resource Reuse Achievable Through the Test and Training Enabling Architecture*, TENA, Proc. International Telemetering Conf., Vol XXXXII, (2006) Paper 06-01-01.

[8] Michael T. Lockard and James A. Garling, *Managing Multi-Vendor Instrumentation Systems with Abstraction Models*, Proc. International Telemetering Conf., Vol XXXXII, (2006) Paper 06-06-05.

[9] Symvionics' Interactive Analysis and Display System (IADS™), http://www.symvionics.com/products/iadshome.htm/.

[10] Telemetry Standards, IRIG STANDARD 106, Secretariat, Range Commanders Council, White Sands Missile Range, New Mexico, http://www.jcte.jcs.mil/rcc/PUBS/pubs.htm.

[11] Thomas Grace and Clay Fink, *Metadata For Range Telemetry*, Proc. International Telemetering Conf., Vol XXXXII, (2006) Paper 06-10-04.

[12] Charles H. Jones, John L. Hamilton, Ronald Fernandes, Michael Graul, and Jon Morgan, *Data Display Markup Language Standardization Effort*, Proceedings International Test & Evaluation Association Workshop, Lancaster, CA, (2006).

[13] John L. Hamilton, Ronald Fernandes, Paul Koola, and Charles H. Jones, *An Overview Of An Instrumentation Hardware Abstraction Language*, Proc. International Telemetering Conf., Vol. XXXXII, (2006) Paper 06-10-01.

[14] Walton R. Williamson, Jason L. Speyer, and Charles H. Jones, *Detection and Isoloation of Instrumentation Failures Applied to GPS and Inertial Navigation*, International Test & Evaluation Association Workshop, Ridgecrest, CA, (2005).

ACRONYMS

A/D	– Analog to Digital conversion
CAIS	– Common Airborne Instrumentation System
DDML	– Data Display Markup Language
GSS	– Ground Support System
IEEE	– Institute of Electrical and Electronics Engineers
IADS	– Interactive Analysis and Display System
iHAL	– instrumentation Hardware Abstraction Language
Iliad	– When I coined this name, I came up with a phrase so it could be considered an acronym. Since then I have decided the phrase doesn't add information and I now promote the idea that "Iliad" is the name of the system and doesn't stand for anything.
iNET	– integrated Network Enhanced Telemetry
IRIG	– The standards portion of the RCC
ISO	– International Standards Organization
ISS	– Instrumentation Support System
PCM	– Pulse Code Modulation (also used to reference the bit stream definition)
RCC	– Range Commanders Council
SBIR	– Small Business Innovation Research
SC	– Signal Conditioner
TEDS	– Transducer Electronic Data Sheet
TENA	– Test and Training Enabling Network Architecture
XML	– eXtensible Markup Language