DESIGN OF A CONFIGURATIONAND MANAGEMENT TOOL FORINSTRUMENTATION NETWORKS

John Roach
VP Network Products Division
Teletronics Technology Corporation
Newtown, PA USA

ABSTRACT

The development of network-based data acquisition systems has resulted in a new architecture for supporting flight instrumentation that has the potential to revolutionize the way we test our aircraft. However, the inherent capability and flexibility in a networked test architecture can only be realized by the flight engineer if a sufficiently powerful toolset is available that can configure and manage the system. This paper introduces the concept of an instrumentation configuration and management system (ICMS) that acts as the central resource for configuring, controlling, and monitoring the instrumentation network. Typically, the ICMS supports a graphical user interface into the workings of the instrumentation network, providing the user with a friendly and efficient way to verify the operation of the system. Statistics being gathered at different peripherals within the network would be collected by this tool and formatted for interpretation by the user. Any error conditions or out-of-bounds situations would be detected by the ICMS and signaled to the user. Changes made to the operation of any of the peripherals in the network (if permitted) would be managed by the ICMS to ensure consistency of the system. Furthermore, the ICMS could guarantee that the appropriate procedures were being followed and that the operator had the required privileges needed to make any changes. This paper describes the high-level design of a modular and multi-platform ICMS and its use within the measurement-centric aircraft instrumentation network architecture under development by the Network Products Division at Teletronics.

KEY WORDS

NETWORK, XML, SNMP, DATA ACQUISITION, DATA MINING, SYSTEM MANAGEMENT

INTRODUCTION

During the last 3 years, the flight test community has seen a flurry of interest in the development of instrumentation networks. Some of this activity is driven by the proliferation of data communications networks in the avionics portion of the aircraft and the need to acquire measurements from those systems. Some of the more visible activity is related to the creation of the iNET project and its goals to standardize instrumentation networks. Primarily, though, the interest in developing instrumentation networks is driven by observing the success that communication networks in general has had in improving productivity in other application domains. Much of that success is made possible by the power and flexibility of this technology; its ability to enable rapid dissemination of information and to control processes remotely. An often underrated factor in the success of communication networks has been the availability of user-friendly tools to configure and manage large networks. Distributed networks are inherently a complex structure. Without powerful tools to aid in understanding their operation and managing the configuration complexity, the productivity gains they bring could be lost in the additional time and effort required to troubleshoot and maintain their operation. The same considerations apply when we look to applying communication networking technology to the flight test domain. We need to bring along the same kinds of tools employed to configure and manage networks in commercial applications along with developing specialized tools specific to our flight test communities' needs.

The focus of this paper is to examine the functionality needed when configuring and managing flight test instrumentation networks. We will use these requirements to define the architecture for a toolset that can be used in the management of an instrumentation network. We then identify common network management technologies used in other application domains and how they would be applied to flight test instrumentation networks.

INSTRUMENTATION NETWORKS

The Network Products Division at Teletronics has been involved in the research and development of instrumentation networks since late 2002. For a more complete summary of those efforts, see our companion paper in this conference [1]. During this ongoing engineering development process, we have been providing feedback regarding our experiences to the iNET development team in an effort to aid in the overall success of the iNET architecture. Additionally, we have been using the recommendations published by the iNET development team to guide our own engineering and product development process. Our objective is to realize an aircraft instrumentation implementation for networked data acquisition that is consistent with their proposed draft architecture. Our approach is tempered by the need to maintain backwards compatibility both with our own product lines and with the requirement that the quality of data that the flight test engineer obtains from a networked system is not compromised. While the iNET effort has made significant progress towards the definition of a Telemetry Network System (TmNS), the overall scale of the iNET project has prevented their effort from making any significant progress in the area of defining standards for configuration and management to this point. This lack of progress is expected to change significantly in the next 12 months as the iNET project enters the next phase of standards and protocols definition based on its draft TmNS architecture. The Network Products Division expects to contribute to that definition process while

defining and building its own management and configuration system based on its existing efforts with its iNET-compliant instrumentation architecture. Our goal is to use our existing progress in the development of networked instrumentation systems as a test bench for the development of instrumentation configuration and management systems.

The overall iNET architecture is quite broad and can be logically partitioned into six systems: Telemetry Network System, Peripherals, Network Management, Spectrum Assignment Management, Peripheral Configuration Applications, and Peripheral Management Applications. The relationship between these systems and the domain physical architecture is as follows:
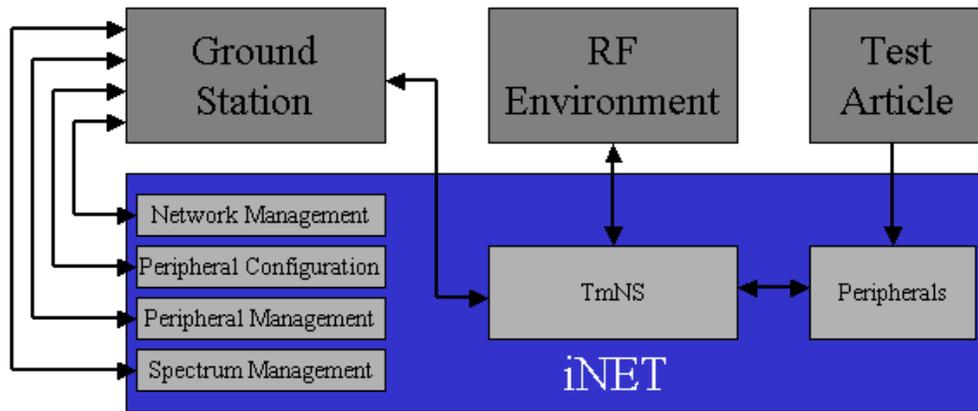
Primarily, the focus of the iNET effort to date has been in the definition of the TmNS. For the purposes of this paper, we have chosen to concentrate on the issues involved in developing software tools for Network Management, Peripheral Management, and Peripheral Configuration. A system not defined within iNET, Network Configuration, also plays a role in our architecture. Before discussing the specifics of our management system, a brief overview of the elements and structure of an instrumentation network is reviewed. For a more complete and in depth discussion of this architecture, see [1][2].

An instrumentation network is defined here as meaning a data acquisition system that uses packetized two-way communication links to exchange measurement data and control messages. In particular, an instrumentation network compatible with the draft TmNS architecture contains the following elements or observes the following rules:

- A network of distributed nodes which communicate using Internet Protocols, either IPV4 or IPV6 (in the future)
- A distributed network structure comprised of Switched Gigabit Ethernet (10/100/1000)
- The use of IEEE 1588 technology for synchronization of time across the network
- The use of UDP multicast for transport of acquisition data
- A dedicated RF channel for streaming serial telemetry from each test article
- A dynamically shared RF channel for all test articles that provides packetized information based on a TDMA-enabled OFDM transport link
- Assignable levels of Quality of Service (QOS) associated to different data streams dependent on bandwidth priority, guaranteed delivery or latency needs.

3

- A recorder with an IP interface and simultaneous read and write access to the media.
- Data acquisition units with an IP interface for configuration and management
- Use of Simple Network Management Protocol (SNMP)

For more details on the TmNS draft architecture, see [1][2][3].

ICMS ARCHITECTURE

The Instrumentation Configuration and Management System (ICMS) is a tool being developed by the Network Products Division at Teletronics and is designed to be used by the flight test engineer to manage the complexity of the aircraft instrumentation network. It can primarily be defined as a collection of software tools sharing a common graphical user interface. Its architecture has been defined and influenced by several factors:

1. The iNET Telemetry Metadata Standard is an attempt to define a measurement meta-model that is vendor-neutral. During the flight test process, there are different points where relevant data is cataloged or created and used to define and configure an instrumentation system or analyze the data after it has been collected. The list of instrumentation activities that this model is targeted for include Describe Sensor Attributes, Describe Measurements, Configure Analog Data Acquisition, Configure Digital Data Acquisition, Publish Bus Catalog, Describe Storage of Measurements, Describe Transport of Measurements, Describe Topology of Instrumentation System and others. Once these activities are defined, they can be transformed using vendor-supplied rules into information compatible with the actual hardware or process supplied by the vendor that carries out that task.
2. The Network Products Division has for the last 3 years worked with Boeing Seattle in the development of an aircraft instrumentation network. Much of our experiences in satisfying their requirements have directed our efforts in this field.
3. Teletronics has, for many years, provided its own configuration software solution for traditional PCM-based data acquisition systems. During the last 3 years, we have worked with Lockheed Martin in the development of an XML-based hardware definition and configuration language for use in automating the flow of bus catalog information into our software tools.
4. The Network Products Division team has many engineers experienced in the development of hardware and software for telecommunication systems. Our previous experiences in developing management software for these systems have influenced our team on the importance of certain user interface considerations.

In addition to these factors, other requirements we have imposed on our efforts are listed below. Note that for the purposes of this list, the word peripheral includes the TmNS Link Manager, all 1588 Switches, the recorder, and all conventional TmNS peripherals (data acquisition units or gateways).

1. Multi-platform support (particularly Windows and Linux) for the host computer.
2. Use of XML for direct configuration of peripherals. In particular, all peripherals will contain their own XML interpreter allowing for direct instance validation and

interpretation. Configuration XML data generated by ICMS will be posted directly to the peripheral or switch.

3. The use of HTML as a visual interface to dynamic parameters. A dynamic parameter is a measurement variable that can be modified in real-time (while the peripheral is actively collecting measurements). All peripherals will contain an HTTP server that serve HTML pages to the ICMS for display and modification by the user.

4. Similarly, the ICMS will contain its own HTTP server that can serve virtual HTML peripheral pages, allowing for a mixed configuration of real and virtual peripherals simultaneously. Changes made to a virtual peripheral can be posted at a later time to an actual device, or used locally with a functional simulator to allow for system validation or measuring network bandwidth utilization.

5. During configuration of a peripheral, the logic needed to compute actual settings will be encoded as embedded JavaScript programs within the HTML pages; the actual execution of these scripts will occur on the ICMS host with the results posted back to the peripheral. This allows not only the specification of declarative knowledge regarding the configuration of a peripheral (for example, the range of a specific parameter), but also procedure knowledge (for example, a change in the voltage gain of the sensor produces a specific change the offset range).

6. Actual HTML pages are stored as XSLT stylesheets; this allows for the actual construction of the visual appearance of each page to occur on the ICMS host as AJAX translations. In fact, the intent is that the configuration pages of any peripheral can be accessed and managed using a simple web browser as needed.

7. A product catalog will act as a knowledge base, containing declarative and procedural information regarding the measurement and network properties of any peripheral. Multiple catalogs can be used at any one time, and catalog updates can occur independent of ICMS software updates.

8. The ICMS will provide two working level abstractions, one at the structure level allowing the assembling of an instrumentation network from physical components, and a second one at the measurement level allowing for definition of an instrumentation network from its inputs and outputs.

9. All peripherals will support a command line interface (CLI) similar in function and compatible with the CLI used by CISCO on their routers/switches.

10. Simple Network Management Protocol (SNMP) will be used for interrogating a peripheral for its real-time status, and for the management of traps generated by the peripheral when user-defined exceptions occur.

The diagram below summarizes the architecture described above.
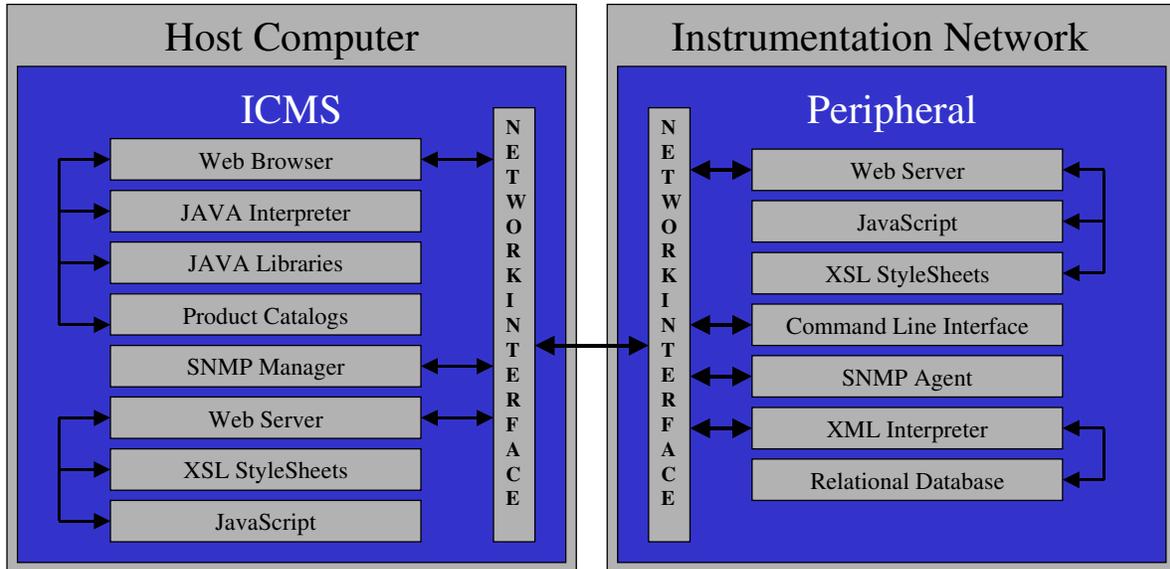
In the following three sections, we focus on providing more discussion and details regarding the function of SNMP, XML, and dynamic parameters in the ICMS architecture.

## SIMPLE NETWORK MANAGEMENT PROTOCOL

SNMP is an IETF standard designed to manage distributed systems connected by an IP network. All elements of our instrumentation architecture (and TmNS) will use SNMP for remote status monitoring and some minimal configuration. Critical events occurring anywhere in the instrumentation network will be forwarded to ICMS for immediate user attention in addition to permanent records of such events being captured by the flight test data recorder. The capabilities exposed thru the use of SNMP are defined by a structure known as a MIB (management information base). A MIB defines objects (like an internal temperature sensor reading) and events (like overheating). MIB objects are identified using a universal hierarchical numbering scheme. Related MIB objects are grouped together by form or function to form a MIB tree.

MIB definitions are written using a subset of the ASN.1 language in plain text files, known as MIB files. Textual names (peripheral variables) are also defined in the MIB files for convenient referral to MIB objects. Several RFCs specify standard MIB trees for IP, Ethernet, and other common networking technology. Private enterprises are allowed to define a proprietary vendor-specific MIB tree. All proprietary vendor MIB trees must attach to a global root hierarchy, specified by RFC 1065 and maintained by the IANA. Each peripheral from Teletronics contains its own unique set of MIBs; all products from Teletronics contain the elements of a vendor-specific MIB tree unique to the company. This MIB tree, along with standard RFC MIBs and MIBs defined unique to iNET will form the basis for management of the instrumentation network.

The set of standard MIBs supported by ICMS include:

RFC 2790	Host MIB
RFC 2863	IF MIB
RFC 2011	IP MIB
RFC 2013	UDP MIB
RFC 2012	TCP MIB
RFC 3418	SNMP MIB
RFC 2273	SNMPv3 Applications

The set of vendor-specific MIBs supported by ICMS are identified by the IANA assigned OID of 23172 under the enterprise branch of the global MIB tree. Our proprietary tree is defined to have the following structure:
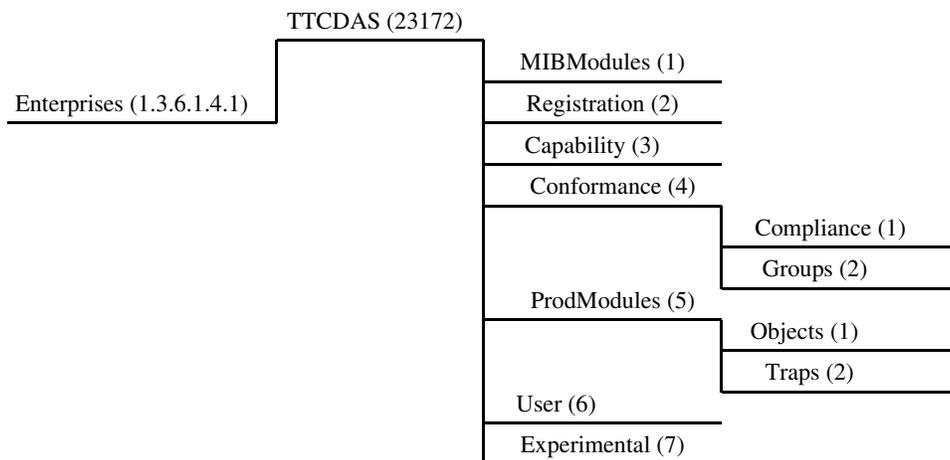
MIBModules	This branch has a flat list of all MIB modules defining our proprietary tree. Each MIB file defines exactly one MIB module.
Registration	This branch has a tree of objects whose IDs represent all TTCDAS network product models like MnVID-2000, NREC-6000.
Capability	This branch has a tree of objects representing all TTCDAS product models. The definition of each object specifies the capabilities of the SNMP agent on a particular multiplexer unit. This tree should follow the same structure as the registration tree.
Conformance	This branch is a customary anchor point for compliance and groups sub-tree. Compliance and object group definitions are described in RFC1904.
Compliance	This branch has a tree of objects representing all TTCDAS product models. The definition of each object specifies the default capabilities of the SNMP agent on a product model. This tree must follow the same structure as the registration tree.
Groups	This branch has a tree of object groups and notification (trap) groups, whose definition specifies a list of IDs of leaf objects or notifications belonging to a product module.
prodModules	This branch has a tree of object and trap (notification) definitions.
Objects	This branch has a tree of trap definitions. They are organized into sub-trees by product modules, like MPPC-500-1 module, MBIM-429-1 module.
Trap	This branch has a tree of trap definitions. They are organized into sub-trees by product modules, like MPPC-500-1 module, MBIM-429-1 module.
User	This branch has a tree of customer specific objects and traps. They are organized into sub-trees by customer names.
Experimental	Experimental objects and traps which may change dramatically or relocate to other sub-trees in the future

A SNMP manager is an SNMP network entity that generates requests to retrieve and modify MIB objects, receives responses to those requests, and receives event reports from other SNMP network entities. A SNMP agent is an SNMP network entity that receives and responds to requests from a SNMP manager, as well as generates event reports to a specific designated SNMP manager. The ICMS will contain a SNMP manager, while all peripherals will support a SNMP agent.

THE ROLE OF XML

XML (Extensible Markup Language) is a set of rules for designing text formats that allow you to structure information. It guides a software program in presenting, understanding, and manipulating information in a uniform manner. ICMS (and the firmware on the peripherals) use XML to organize the information each needs to perform its functions. TTC uses several different classes of XML instance documents to organize and represent different sets of information relevant to the configuration and management of an instrumentation network. A XML Schema is used to validate that a particular XML instance document belongs to a specific class. ICMS defines 5 different XML Schemas for representing the information used for configuration and management. Each class is briefly described below.

- **Product Catalog** – represents the set of Teletronics network products that can be used to build an instrumentation network. Multiple catalogs (including those from multiple vendors) can be used as a source of peripherals for the set of measurements needed. ICMS uses this information to know what design options can be given to the flight test engineer.

- **Network Configuration** – represents the actual physical connection of peripherals, switches, and devices needed to assemble the instrumentation network.

- **Peripheral Configuration** – represents the specific programming instructions for each peripheral in the network that define its configuration and how data is collected.

- **Data Filtering** – specifies the rules that define where data is routed within the network.

- **Measurements** – defines the set of sensor or bus measurements the instrumentation network will accomplish.

A small example of an XML configuration instance document for a 1588 switch is shown below:

```
<TTC>
    <PROJECT>
        <HARDWARE>
            <BOX Name="Switch-1">
                <CARD Name="NSW-MAIN" Type="NSW-MAIN" Slot="1" Bus="L">
                    <SETTINGS>
                        <VLAN Number="1"/>
                        <MANAGEMENT_ADDRESS>192.168.0.0</MANAGEMENT_ADDRESS>
                        <PORT Number="1" Enable="Enabled" Speed="10" Duplex="half"/>
                        <PORT Number="2" Enable="Enabled" Speed="100" Duplex="full"/>
                        <PORT Number="3" Enable="Enabled" Speed="1g" Duplex="full"/>
                        <PORT Number="4" Enable="Enabled" Speed="auto" Duplex="auto"/>
                        <MCAST_RULE Address="10.203.123.201" Vlan="1">
                            <PORT Number="2"/>
                        </MCAST_RULE>
```

```
        </SETTINGS>
    </CARD>
    <CARD Name="NSW-TIME" Type="NSW-TIME" Slot="2" Bus="L">
        <SETTINGS>
            <PORT Number="0" Enable="Enabled">
                <PPS_IN_SOURCE>IRIG AC</PPS_IN_SOURCE>
                <GPS_ANT_PWR>5V</GPS_ANT_PWR>
                <RT_CLOCK_IN>Automatic</RT_CLOCK_IN>
                <UPDATE_METHOD>Automatic</UPDATE_METHOD>
            </PORT>
        </SETTINGS>
    </CARD>
    <CARD Name="NSW-IO" Type="NSW-IO" Slot="1" Bus="L">
        <PORT Number="2" Enable="Enabled">
            <IEEE1588_CLOCK>Boundary</IEEE1588_CLOCK>
            <SYNC_INTERVAL>16</SYNC_INTERVAL>
            <PTP_DOMAIN>DFLT</PTP_DOMAIN>
            <TX_DELAY_COMPENSATION>10</TX_DELAY_COMPENSATION>
            <RX_DELAY_COMPENSATION>10</RX_DELAY_COMPENSATION>
        </PORT>
    </CARD>
        </BOX>
    </HARDWARE>
  </PROJECT>
</TTC>
```

DYNAMIC PARAMETERS

Dynamic parameters refer to any programmable attributes of a peripheral that can be modified while the device is in operation. Not all programmable attributes can be used in this manner; some attributes, if modified during operation of the system, may cause undesirable side effects elsewhere in the instrumentation network. Dynamic parameters are primarily intended to allow the test engineer to fine-tune or revise a measurement from a sensor based on real-time feedback of its operation. A detailed paper on the workings of dynamic parameters was presented at the ITC 2006 conference [4].

The peripheral is initially programmed using the XML-based configuration file generated by ICMS. This file contains settings for all variables associated with the measurement (both dynamic and static) along with configuration parameters for the peripheral itself (such as multicast addresses to use for transmitting the results). During the configuration phase, the peripheral acts as a client to the ICMS, requesting its configuration files from the host computer and interpreting the XML content as modifications to its relational database. In the dynamic parameters phase, its role is temporarily reversed and the peripheral acts as the server to ICMS. The peripheral runs a specialized HTTP server to deliver files upon request of the ICMS. The HTTP server combines a server-side XML parser and XSLT processor to create dynamic web pages. Each page conforms to a common user interface, complete with a form for modifying dynamic measurements along with submission and navigation buttons. The peripheral always stores a copy of its current configuration in a local XML file. As shown below, when the link to a peripheral with dynamic measurements is accessed through ICMS, an HTTP GET request is generated and sent to the peripheral. In response, the peripheral combines its XML file with XSLT stylesheets to deliver the appropriate data entry form to ICMS. Modifications made within ICMS make use of embedded JavaScript (as necessary) to insure procedure knowledge is used to maintain consistency with the existing programming in the peripheral. Once the changes are complete, the modifications are posted back to the peripheral as a new XML programming file.

ICMS
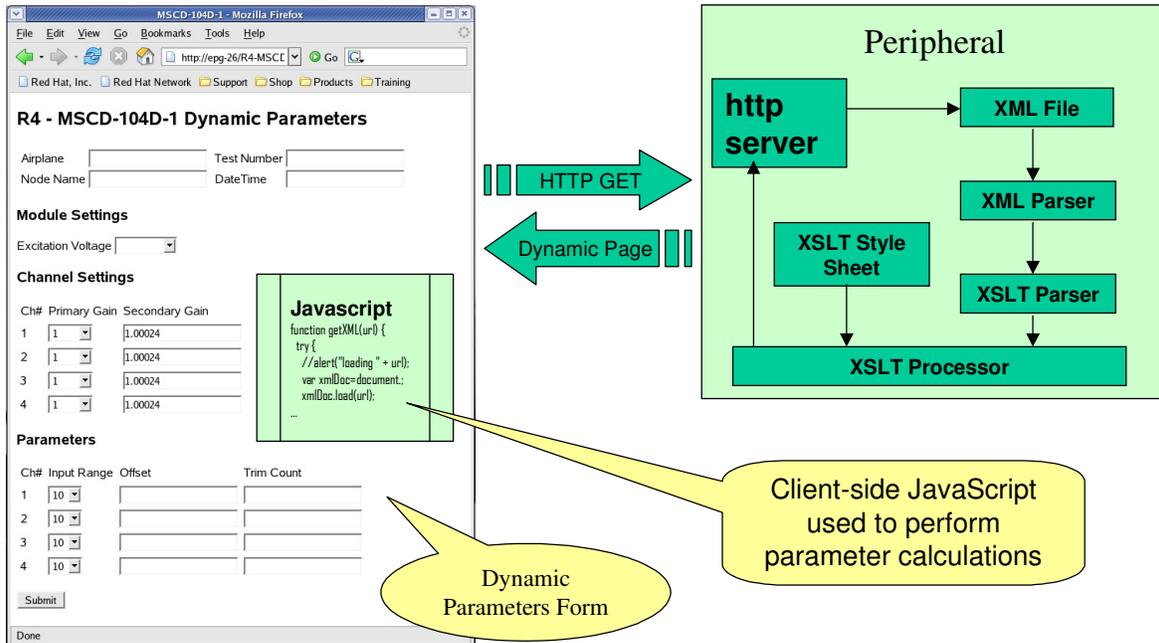
CONCLUSION

The Instrumentation Configuration and Management System (ICMS) is a software tool that aids the flight test engineer in the design, verification, maintenance and analysis of aircraft instrumentation networks. Its design has been influenced by the experience of the Network Products Division in building network-centric data acquisition systems for our customers and by the configuration and management work being done within iNET. We have discussed some design considerations in this paper and described some of the requirements that we expect our system to satisfy. Its viability as a product will be tested when it is delivered as part of an instrumentation network solution for the Future Combat Systems project.

REFERENCES

1. Roach, John; "The Architecture of Aircraft Instrumentation Networks", Proceedings of the International Telemetering Conference (ITC 2007)

2. Roach, John; "vNET Architecture Study: One Vendor's Implementation", Proceedings of the International Telemetering Conference (ITC 2006)

3. Skelley, Daniel; "iNET Project – Special Session", Proceedings of the International Telemetering Conference (ITC 2005)

4. Pesciotta, Eric; "An Implementation of Dynamic Data Acquisition Measurements", Proceedings of the International Telemetering Conference (ITC 2006)