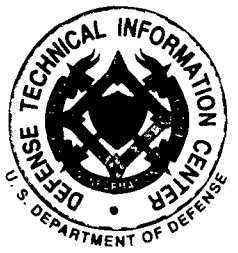


Defense Technical Information Center Compilation Part Notice

This paper is a part of the following report:

• *Title:* Technology Showcase: Integrated Monitoring, Diagnostics and Failure Prevention. Proceedings of a Joint Conference, Mobile, Alabama, April 22-26, 1996.

• *To order the complete compilation report, use:* AD-A325 558

The component part is provided here to allow users access to individually authored sections of proceedings, annals, symposia, etc. However, the component should be considered within the context of the overall compilation report and not as a stand-alone technical report.

TURBINE ENGINE DIAGNOSTICS USING A PARALLEL SIGNAL PROCESSOR

Theodore A. Bapty, Akos Ledeczi, James R. Davis
Measurement and Control Systems Laboratory
Department of Electrical and Computer Engineering
Vanderbilt University
Nashville, TN 37325

Ben Abbott
Department of Electrical and Computer Engineering
Utah State University
Logan UT

Terry Hayes, Thomas Tibbals
Sverdup Technology Inc., AEDC Group
Arnold AFB, TN

Abstract: For the past several years, the Measurement and Computing Systems Laboratory has been working in close cooperation with the United States Air Force at Arnold Engineering Development Center (AEDC), Arnold AFB, to develop techniques for large-scale instrumentation systems. In-depth, on-line analysis of test data from turbine engine testing is critical to ensuring an accurate, timely evaluation and diagnosis of engine performance. Given the complexity of the analysis algorithms and the quantity of data, the computations overrun the capability of the fastest supercomputers.

This paper describes the development of Computer Assisted Dynamic Data Monitoring and Acquisition System (CADDMAS). The CADDMAS is a 48 channel, 50 KHz full-time analysis system, capable of flexible analysis of signals in the time and frequency domains. Data is presented on real-time displays, showing, for example, spectrums, Campbell Diagrams, engine-order tracking. The system (both hardware and software) is synthesized using a novel model-based technique. The approach has been used to generate several systems used for on-line military and commercial turbine engine data analysis at Arnold AFB and for analysis of the SSME for NASA. On-line analysis has had a significant impact on turbine engine testing, reducing the time necessary to meet testing objectives and improving the quality of testing results. Substantial savings have been demonstrated by allowing immediate access to reduced data.

Keywords: Instrumentation, Parallel Processing, Turbine Engine Diagnostics, Turbomachinery, Model-Based Systems, High-Performance Computing, Digital Signal Processing, Real-Time Systems.

INTRODUCTION: For the past several years, the Measurement and Computing Systems Laboratory has been working in close cooperation with the United States Air Force at Arnold Engineering Development Center (AEDC), Arnold AFB, to develop techniques for large-scale instrumentation systems. AEDC is the premier aerospace ground testing facility in the United States. Propulsion testing is of particular importance. Propulsion test facilities allow aircraft engines to be tested at a range of altitudes, mach numbers, temperatures, etc. The entire flight envelope of an engine can be simulated.

A tremendous quantity of data is generated during turbine engine testing. Stress testing exemplifies this problem, where hundreds of sensors must be analyzed over frequencies up to tens of kilohertz.

In-depth, on-line analysis of this data is critical to ensuring an accurate, timely evaluation and diagnosis of engine performance. Given the complexity of the analysis algorithms and the quantity of data, the computations overrun the capability of the fastest supercomputers. Sustained rates of several billion floating point operations per second are required. This computation is in addition to the demands of acquiring and maintaining the incoming data. The solution to this problem is to apply parallel processing. Off-the-shelf hardware is available, allowing construction of massively parallel machines. Unfortunately, the software technology lags behind.

This paper describes the development of Computer Assisted Dynamic Data Monitoring and Acquisition System (CADDMAS). The CADDMAS is a 48 channel, 50KHz full-time analysis system, capable of flexible analysis of signals in the time and frequency domains. Data is presented on real-time displays, showing, for example, spectrums, Campbell Diagrams, engine-order tracking. The flexible architecture allows arbitrary analysis to be added.

DESIGN FACTORS: Construction of large scale instrumentation systems is complex. The factors that drive this complexity are as follows:

1. Many sensors are necessary to effectively instrument a large system. High measurement bandwidths with large numbers of sensors generate a significant computational load. As a result, large-scale parallel processing systems are required to supply the necessary computational resources.
2. Instrumentation systems have real-time specifications, both hard and soft. Building real-time systems, especially using parallel hardware, is a complex task.
3. System requirements change rapidly in many applications. The system performance must scale gracefully over a wide range. Functionality must meet changing requirements. The systems must be upgraded in-place, with minimal down-time. This hardware and software scalability incurs a cost in complexity.
4. System implementation cost is also a driving factor. The increase in cost as a system is scaled-up must be modest. To meet this goal, the computational platform is typically a

low-cost parallel DSP platform. Software development and maintenance costs must be minimized as well.

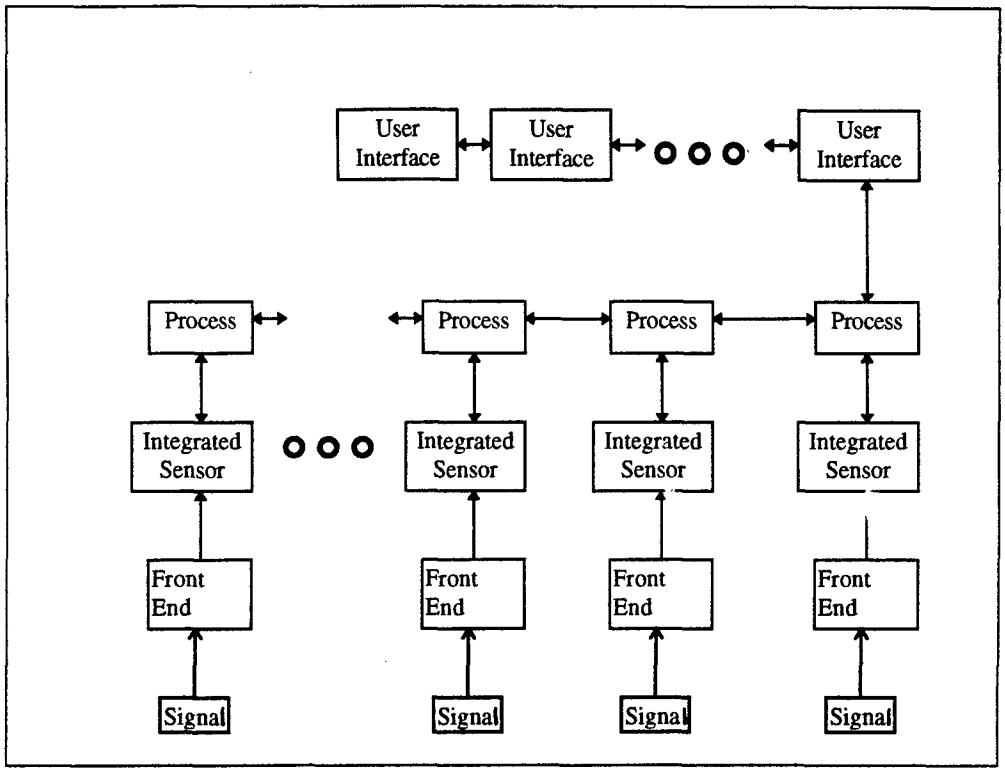
5. The users of instrumentation systems are typically not instrumentation engineers. Interaction with the system must mask the underlying complexity, since the domain users are unlikely to have experience in complex, real-time, reactive, large-scale, scalable, parallel systems. The user is concerned with specifying system requirements and with using the resultant system. Interaction should be in terms of familiar domain concepts.

These factors must be addressed if a successful system is to be constructed. The design strategy must encompass requirements specification, software design, and hardware design in a real-time environment. Cost performance requirements also specify that the target system is a parallel processor.

TARGET DIAGNOSTIC SYSTEM HARDWARE ARCHITECTURE: The systems are constructed with an application specific network of Texas Instruments TMS320C31/40 Signal Processors and/or Inmos transputers. A large number of different systems have been constructed to meet specific processing requirements. The largest system is built using 48 TMS320C31's, 12 TMS320C40's, and 48 Motorola 56002's. The TMS320C31 is a standard Digital Signal Processor, augmented with a primitive communication facility. The TMS320C40 is a newer DSP with integrated communication hardware. The TMS family processors are capable of 40 MFLOPS performance and do the bulk of the analysis, using single precision floating point arithmetic. The Motorola processors operate on integer data and serve as a data acquisition front-end to the system.

The processors are connected in a modified pipeline topology, closely matching the information flow within the application. Figure 1 shows the general architecture.

- The User Interfaces are Intel 486 or Pentium based PC's. They serve to display information, print hardcopies, and control the processing options of the system.
- The "Process" nodes accumulate information for Campbell Diagrams, perform Bicoherence computations, and route information throughout the system.
- The "Integrated Sensor" nodes perform the low-level operations: FFT, Magnitude Spectrum, Spectral Peaks, Engineering Unit Conversion, etc.



MODEL-BASED DESIGN METHODOLOGY: The system (both hardware and software) is synthesized using a novel model-based technique. Using this technique, the engineer building the system represents the goals of the system in terms of **Models** - such as signal flow graphs and processing requirements. These models are interpreted to automatically generate the wiring diagrams of the hardware and the executable software files.

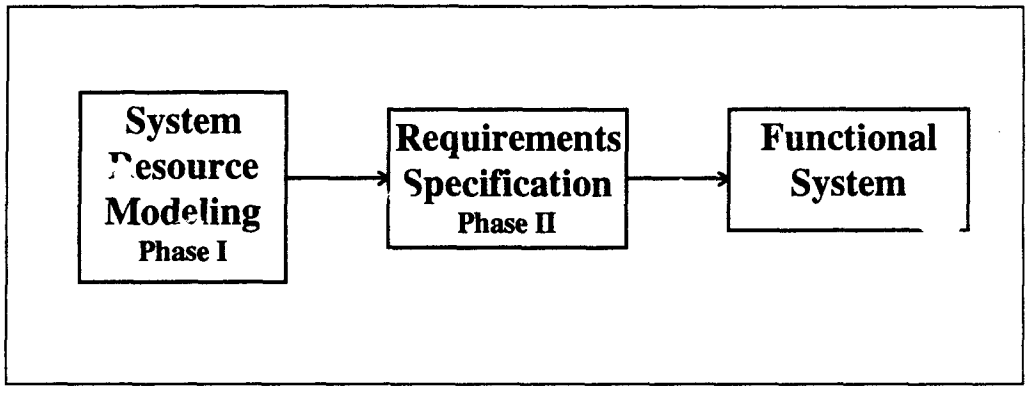
In order to manage the complexity of the effort while satisfying the flexibility and remaining within cost, a careful, structured approach must be used. We are approaching the class of parallel instrumentation systems using multiple-aspect modeling. The overall system architecture for parallel real-time instrumentation is shown in Figure 2. The basic thrust in the model-based approach is to model the system at appropriate levels, in a domain-specific manner. The domain specific nature allows us to represent the system design in concepts that are familiar to the end-user.

The information capture process is a combination of graphical editing and/or textual menu-based forms where appropriate. These models are interpreted various ways tc generate the executable systems and other tools. The proper definition of the paradigms is critical to the success of the approach. In this section we describe the paradigms developed for the domain of parallel real-time instrumentation systems.

The organization of the modeling paradigm is driven by the structure of the entire testing process. This includes the process of specifying and building the systems as well as the users involved. The user interactions and processes of building instrumentation systems can be broken into three phases. In the first phase, the range of system capabilities and resources are defined by the **Instrumentation System Engineer**. The phase I paradigm is called **System Resource Modeling**, where parameterized models specify the available processing & output classes along with implementation structure and parameter options. In addition, the available hardware resources and system inputs are defined.

In the second phase of system construction, **Requirements Specification**, the **Domain Engineer** uses editors automatically generated from phase I to specify requirements for the desired instrumentation system implementation. The characteristics of the system are selected here: inputs, outputs (i.e. plots, alarms, databases), system structure, timing requirements, etc. Note that the domain engineer is not necessarily trained in signal processing, computer architecture, or parallel processing.

The third set of users of the system are the operators, who interact with the automatically generated system resulting from the interpretation of the phase II requirements specification and phase I System Resource Models. The user interface is a window into the capabilities specified and built into the system by the domain engineer, allowing interactive application of these capabilities.



System Resource Modeling Paradigm: The system resources can be broken into two groups: the algorithmic and software implementations & structure and the physical system components. The conceptual computations are defined in **the Signal Flow Aspect**, while the hardware components used in constructing the computational platform, as well as the connections to the external environment is defined in the Physical Resource Aspect.

Signal Flow Aspect: The signal processing engineer uses the Signal Flow Modeling aspect to define the mechanisms for generating the user plots. The processing is described using a signal flow graph representation, commonly used to describe many digital signal processing algorithms. The primary concepts employed in this paradigm are as follows.

- Processing Nodes, representing the elemental large grain signal processing operators, with attributes, specifying run-time behavior, resource requirements (execution time),
- input/output characteristics, etc., information necessary for process allocation and scheduling.
- Connections represent the flow of information between processing nodes.
- Timing Constraints indicate the critical time path in component computations.
- Local Parameters allow control of the behavior of the processing nodes from the high-level builder.

Hierarchy and iterators are used to manage the complexity of large processing structures and to manage highly regular computational structures.

These concepts are extended further to allow the packaging of various signal flow graphs into definitions of user plots. The actual implementation of the processing structure used to generate a specific plot may vary depending on the requirements. Consider, for example, spectral analysis. Many methods exist for computing spectra. The best algorithm depends on the goals. Multiple implementations can be specified, along with the methodology for selection of the proper option and configuration of the internal processing. We therefore add the following concepts:

- The **System Output Class Definition** consists of a grouping of signal flow graphs implementing similar functions. These represent the design alternatives available when generating system outputs. The implementations may differ in computational expense, resolution, accuracy, or time response.
- **Implementation Option Selection Criteria** represent the alternate methods for implementing a function. Criteria are specified for choosing the best implementation given a set of specifications. This information allows the builder to choose between, for instance, an FFT or a filter bank implementation of a spectrum computation. The selection criteria is specified as an algorithm, a function of the set of configuration variables. These configuration variables are specified in this model.
- **Implementation Configuration Methods** allow the builder to tune the implementation to match specifications. For example, in an FFT application, the FFT block size, window function, and overlap ratio must be specified. The implementation configuration is also specified as an algorithm, since this process is specific to the signal processing
- method.

Graphical modeling is a natural paradigm for this system aspect. Signal Flow Graphs have been used historically to design and conceptualize signal processing systems.

Physical Resource Aspect: Hardware Architecture models for defining the set of DSP's Transputers, A/D converters, etc. used in the physical construction, allowing the user to specify where signals enter the system. **Processor Nodes** correspond to a self contained processor, memory, I/O, communications, and special resources. The node attributes to be specified include *processor type, processor speed(MIPS/MFLOPS), memory size*

communication bandwidth, etc. **Connections** define the physical connections between node. Combined with the processor nodes, this defines the system topology. **Signals** are the raw data, the sensors digitized into the system.

Hierarchies of these nodes are used to define preconfigured nodes and connections: sub-assemblies (boards), chassis, racks, and systems. A graphical model is appropriate in this paradigm, since it is natural to draw pictures of system architectures and network topologies.

System Requirements Modeling Paradigm: At the top level, the user of the system interacts only with familiar concepts, directly reflecting the user's view of the system. These concepts include: **Inputs:** The signals available for processing. **Outputs:** The output processing requirements of the system. These include *Plots*, *Result Files*, and *Alarms*. For each of these outputs, the user specifies relevant information, such as type, accuracy's, range, update rates, maximum latencies.

From this vi. v, the user can specify the capabilities and performance of the system that is required for a particular test. The complexities of parallel software, signal processing, and computer hardware are invisible at this level. This is required, since the end-user of the system is an aerospace engineer, with little concern or training on these issues.

MODEL INTERPRETATION: The model interpretation is a multiple phase operation. In the first phase, the signal processing expert models the available classes of system structures and outputs, and the instrumentation engineer defines the signal inputs. These models are interpreted to generate the System Requirements Definition Model Editor. This editor is used by the end-user (the domain engineer) to model/specify the system requirements. The second phase of interpretation takes these requirements and generates an architecture-independent specification for the software structure of the system. The final phase integrates the architecture-independent specification and the hardware model to generate a hardware/software design and implements an executable system. Under typical operation, the first phase will be used infrequently, whenever new features/plots are added to the system. The most common design loop will use phase II and III to generate systems of various sizes, con' gurations, and hardware platforms.

CONCLUSION: The approach has been used to generate several systems used for on-line military and commercial turbine engine data analysis at Arnold AFB and for analysis of the SSME for NASA. On-line analysis has had a significant impact on turbine engine testing, reducing the time necessary to meet testing objectives and improving the quality of testing results. Substantial savings have been demonstrated by allowing immediate access to reduced data.

BIBLIOGRAPHY:

Bapty, T.A., "Model-Based Synthesis of Parallel Real-Time Systems", PhD Dissertation, Vanderbilt University, 1995.

Abbott, B, Bapty T. Biegl, C. Karsai, G., Sztipanovits, J. ``Model-Based Approach for Software Synthesis", IEEE Software, pp. 42-53, May, 1993

Karsai, G. ``A Visual Programming Environment for Domain-Specific Model-based Progrmming.", IEEE Computer, May 1995.

Ledeczi, A., Abbott B.A., ``Parallel Systems with Flexible Topology", Proc. of the Scalable High Perfomace Computing Conference, pp271-276, Knoxville, TN 1994.

Sztipanovits, J., Abbott, B., Bapty, T., Misra, A., "Model-Based Synthesis of Complex Embedded Systems", Proc. of the 1994 Complex Systems Engineering Synthesis and Assessment Technology Workshop, Washington D.C. July 19-20, 1994

Bapty, T.A., Abbott, B.A., Biegl, C. Ledeczi, A. "Parallel Turbine Engine Instrumentation System", Proc. of the 9th AIAA Conference on Computing in Aerospace, San Diego, CA, October, 1993