Technology for Rapidly Adaptable Command & Control (C2) Systems

Roger J. Dziegiel, Jr.
AFRL/IFTD,
525 Brooks Rd.
Rome, NY 13441
(315) 330-2185
dziegielr@rl.af.mil

Capt. Jonathan C. Clough
AFRL/IFSA
525 Brooks Rd.
Rome, NY 13441
(315) 330-3782
cloughj@rl.af.mil

Abstract

Future Coalition Forces Commanders in forward-deployed locations will need robust, flexible, and intelligent command and control infrastructure to effectively employ aerospace power rapidly with fewer resources and a smaller footprint. With the ever-widening scope of operational environments in which air power is being employed, Air Force C2 nodes must provide commanders with the appropriate functionality to support a variety of operational scenarios from Major Regional Conflicts (MRC) to Operations Other Than War (OOTW). Given the mandate for rapid expeditionary deployments into these evolving crisis environments, the C2 infrastructure must be one that is able to adapt dynamically to the unfolding scenario without system downtime or the need for large contingents of system administrators to reengineer the computing or communication architectures. The limited resources available early in a deployment must be able to support a considerable variety of application tools and permit non-discontinuous transitions among them as the situation dictates. This dynamic system reconfigurability should encompass both the physical and functional realms. Hardware footprint must be variable-sized based on what equipment can be airlifted into theater in a given amount of time, and the functionality provided by the application software must be able to evolve seamlessly to respond to changing operational needs. Hardware, software, and communications infrastructures need to adapt and evolve to the changing nature of the mission without breaking the stride of the battle rhythm. Intelligent, scaleable resource-aware distributed architectures are necessary to make this vision a C2 reality.

1.0 Introduction

Future Coalition Forces Commanders in forward deployed locations will need a robust flexible, adaptable, and intelligent command and control infrastructure in order to effectively employ aerospace power rapidly with fewer resources and a smaller footprint. Complicating matters is the ever-widening range of operations in which air power is being utilized, spanning the breadth

of the spectrum of conflict. The command and control (C2) systems of the future must be able to evolve and adapt to changing operational environments without requiring time-consuming and labor-intensive configuration changes needed today. Furthermore, with the rise of split-Air Operations Center (AOC) operations and the need for timely, accurate information exchange between forward and rear nodes, effective collaboration environments are absolutely essential. The explosion of available information has also spawned the need for its presentation in a meaningful and relevant fashion to the individual(s) making time-critical decisions.

System design technology emerging from the Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) sponsored Evolutionary Design of Complex Software (EDCS) program, and the vision of future C2 as embodied in the AFRL Configurable Aerospace Command and Control (CACC) program, are combining to provide just such capabilities to Coalition Forces Commanders. One such EDCS technology is Model-Integrated Computing (MIC), which addresses development and evolution problems of software-integrated systems. The key element of this approach is the extension of the scope and usage of models such that they form the backbone of a model-integrated system development process.

2.0 The Challenge

Information processing is increasingly becoming an integral part of today's and tomorrow's systems. It dramatically increases the potential interactions among physical components and processes, generates complex dynamics, and establishes component interdependencies unknown in legacy systems. The tight integration of physical and information processes creates tremendous challenges for the software technology. First of all, the conceptual construct of the software is inextricably combined with the conceptual construct of its external environment. Consequently, the software cannot be static – it must adapt, evolving together with its environment. Another well-known difficulty in the design and implementation of embedded information systems is that software is becoming a component of a physical system. The overall system behavior can only be understood if information, material and energy transfer processes are modeled and analyzed together. This means that software artifacts need to be modeled in their context, using a modeling language – or modeling paradigm – that is meaningful for the design, analysis and operation of the whole system. An additional challenge that must be answered is that of criticality. Software directly impacts the operation of physical processes and failure may cause unacceptable social or economic damage. Thus the software technology must offer methods and tools for verifying and maintaining dependability requirements.

3.0 Concept Of Model Integrated Computing (MIC)

Model Integrated Computing (MIC) addresses the challenges described above by providing rich, domain-specific modeling environments including model analysis and program synthesis tools. This technology is used to create and evolve integrated, multiple-aspect models using concepts, relations, and model composition principles routinely used in the specific field, to facilitate systems/software engineering analysis of the models and to automatically synthesize applications from the models.

Model Integrated Computing is being used for building embedded software systems. The key element of this approach is the extension of the scope and usage of models such that they form the "backbone" of a model integrated system development process. In Model Integrated Computing models play the following roles:

- Integrated, multiple-view models capture the information relevant to the system to be developed. Models can explicitly represent the designer's understanding of the entire system, including the information processing architecture, the physical architecture and the environment it operates in.
- Integrated modeling allows the explicit representation of dependencies and constraints among the different modeling views.
- Tools analyze different, but interdependent characteristics of systems (such as performance, safety, reliability). Tool specific model interpreters translate the information in the models to the input language of analysis tools.
- The integrated models are used for the automatic synthesis of the target software application. The model-integrated program synthesis process utilizes model interpreters to translate the models into executable specifications.

Using MIC technology, one can capture the requirements, actual architecture, and the environment of a system in the form of high-level models. Requirement Models allow the explicit representation of desired functionalities and/or non-functional properties. Architecture Models represent the actual structure of the system to be built. Environment Models capture what the outside world of the system looks like. These models act a repository of information that is needed for analyzing and generating the system under development.

In Figure 1, the Multigraph Architecture (MGA) provides a unified software architecture and framework for building domain-specific tools for: (1) building, testing and storing domain models, (2) transforming the models into executable programs and/or extracting information for system engineering tools, and (3) integrating applications on heterogeneous parallel/distributed platforms. The MGA is comprised of three levels: 1) application level, 2) model-integrated program synthesis (MIPS) level, and 3) the meta-level [Nordstrom et al., 1999].

The Application Level represents the synthesized adaptable software applications. Components may include configurable COTS software tools, different communication platforms, and fine-grain adaptable applications specified in terms of the Multigraph Computational Model (MCM), a macro-dataflow model which models the synthesized programs as an attributed, directed, bipartite graph. The runtime support for MCM, the Multigraph kernel, is implemented as an overlay above operating and communication systems.

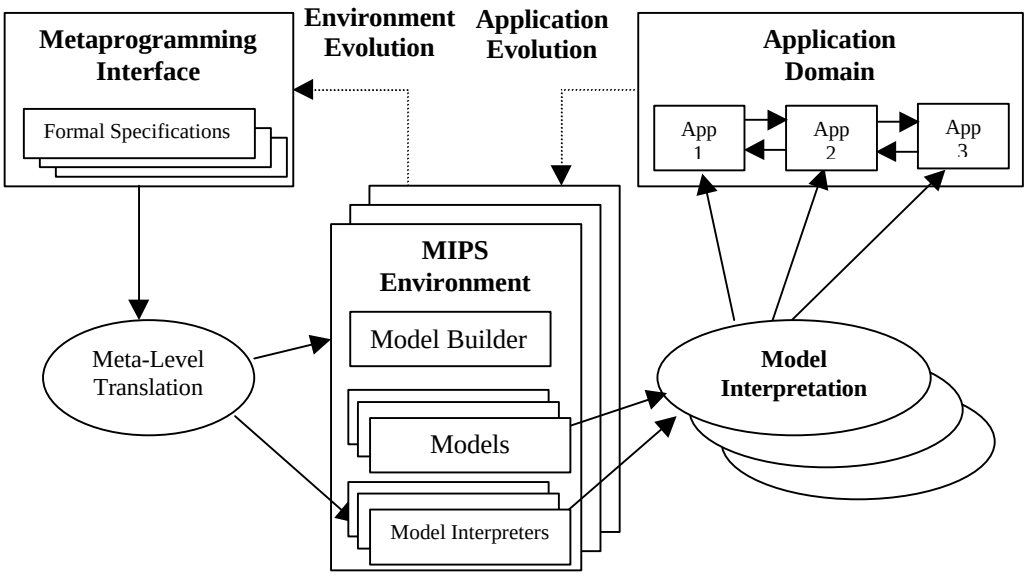

Figure 1: The MultiGraph Architecture

The Model Integrated Program Synthesis (MIPS) Level includes generic, but customizable tools for model building, model analysis, and application synthesis. The generic components of the architecture are: (1) a customizable Graphical Model Editor (GME), (2) a database layer for storing and accessing models, (3) model analysis tools and external tools, and (4) model interpreters that synthesize applications ("executable models") or translate models into input data structures of the analysis tools ("analysis models") [Ledeczi et al., 1999].

The Meta Level of MGA provides a metaprogramming interface for the components on the MIPS Level. The Metaprogramming Interface includes: (1) support for the formal specification of domain-specific modeling paradigms and model interpreters using formal languages, (2) meta-level translators to generate configuration files for the GME from the modeling paradigm specification, (3) meta-level program synthesis tools for generating model interpreters from their formal specification, and (4) support for the validation and verification of the meta-models.

One key feature of MIC is the ability to perform meta-modeling -- the modeling of models. The added advantages include support for the formal specification of domain-specific modeling paradigms and model interpreters using formal languages, fast adaptation of applications through automatic re-synthesis of running applications from models, and evolution of applications through the modification of models and re-synthesis of applications. Because the environment generation system is model-based, target modeling environments can easily adapt to changes in requirements. Whenever a new kind of system is needed, the meta-model is updated and the new modeling environment is generated.

The Unified Modeling Language (UML), along with the Object Constraint Language, have been used in defining MGA models. UML is a graphical modeling language for specifying, constructing, visualizing, and documenting the artifacts of a software system. UML supports the four-layer meta-modeling architecture, and can be used to model other modeling languages.

Formalism is added to MIC using the Multigraph Constraint Language (MCL). MCL is a formal language that allows the specification of complex syntactic and semantic constraint relationships during the meta-modeling process. Because MCL is a formal language, computer-based verification and consistency checking can be performed before the target environment is generated, ensuring the target system properly constrains the modeler to create only valid models, eliminating syntactic modeling errors. MCL is based on UML's Object Constraint Language (OCL), an industry standard [Gates 1998], [Melewski, 1998].

4.0 Current Applications Of MIC

The success of a technology depends on how well it scales up to real world problems. MIC has been applied to real world problems and has resulted in significant savings. It is in use at Saturn Corporation for the Saturn Site Production Flow (SSPF) System and has to date resulted in an increase in throughput by 10 percent. In addition, the USAF Arnold Engineering Development Center (AEDC) used MIC to develop a flexible Engine Test and Data Analysis System called Computer Assisted Dynamic Data Monitoring and Analysis System (CADDMAS) which analyzes large quantities of high bandwidth signals during testing of aircraft engines. The Air Force has verified savings of $10 million annually through its application.

4.1 Saturn Corporation

One of the driving forces industry faces today is the need to be competitive. Industry needs to increase throughput while keeping costs down. This requires an engineering process involving day-to-day and long term examination and analysis of the functioning of the plant and operations, identification of bottlenecks in the production, analysis of capacity, and identification for improvements.

To tackle the problem of being competitive and increasing throughput, the Saturn Site Production Flow (SSPF) system was developed. SSPF is a Manufacturing Execution System which is an integral and enabling component of the business process employed (see Figure 2). The Saturn Site in Spring Hill, TN, consists of three plants – Powertrain, Body, and Assembly. There are approximately 10,000 operations such as stamping, molding, fabrication, casting, machining, assembly, fluid fill, etc. Saturn brings these operations together in an integrated manufacturing system designed around "just-in-time" principles.

The Saturn manufacturing site, is a network of processes and buffers. Processes represent the operations required to build a car. An operation, in turn, can involve actions (such as the casting, machining, and welding of car parts) or assemblies (such as transmissions, engines and the final car). Each process has associated measurements that indicate its productivity. These measurements – critical information in making business decisions – include entities such as cycle time, production count, work in process, and downtime. Buffers lie between processes. A buffer holds parts and/or subassemblies produced by an upstream process for consumption by a downstream process. The interconnectivity of processes and buffers captures the sequences of operations required to produce a car; it also defines the interdependence of processes on each other and on buffer capacities. A process may have to remain idle if an upstream process in not producing enough parts (starving). On the other hand, a process may be forced to stop producing

if a downstream process is not consuming enough parts (blocking). Starving and blocking conditions may result from mismatches between process cycle times, downtime, and the capacities of buffers between processes [Misra et al., 1997].

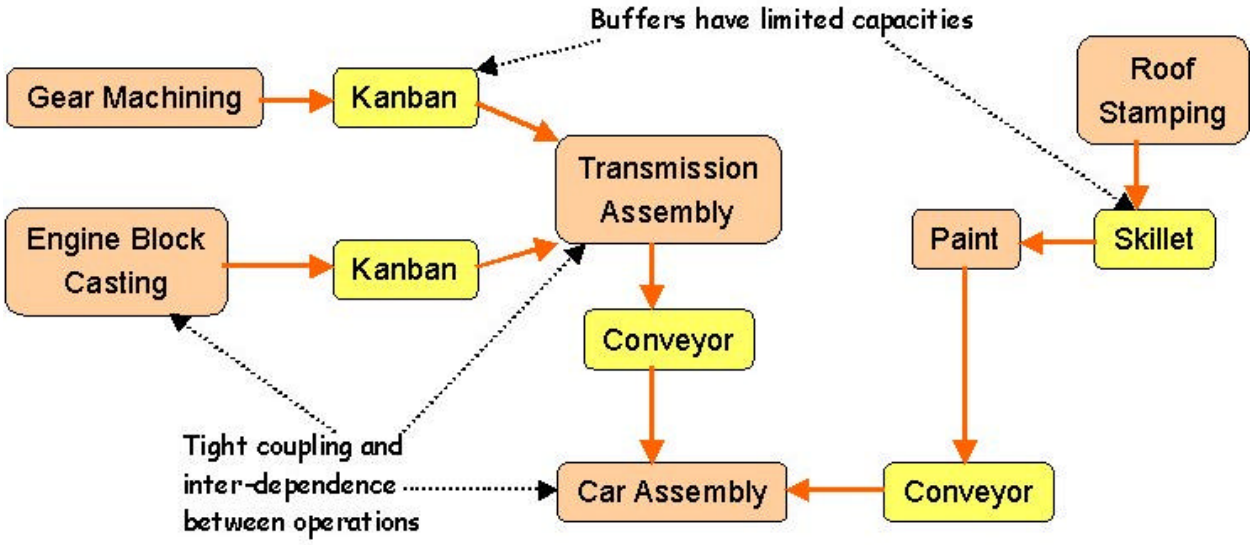

Figure 2: SSPF Business Process

In order to visualize and analyze the throughput characteristics of the plant, and make business decisions for throughput improvement, SSPF implements an engineering process (see Figure 3). The engineering process is a cyclical process that is implemented by the SSPF application, which in turn consists of many COTS and custom developed components and the Model-Integrated Computing (MIC) Environment [Long et al., 1998].

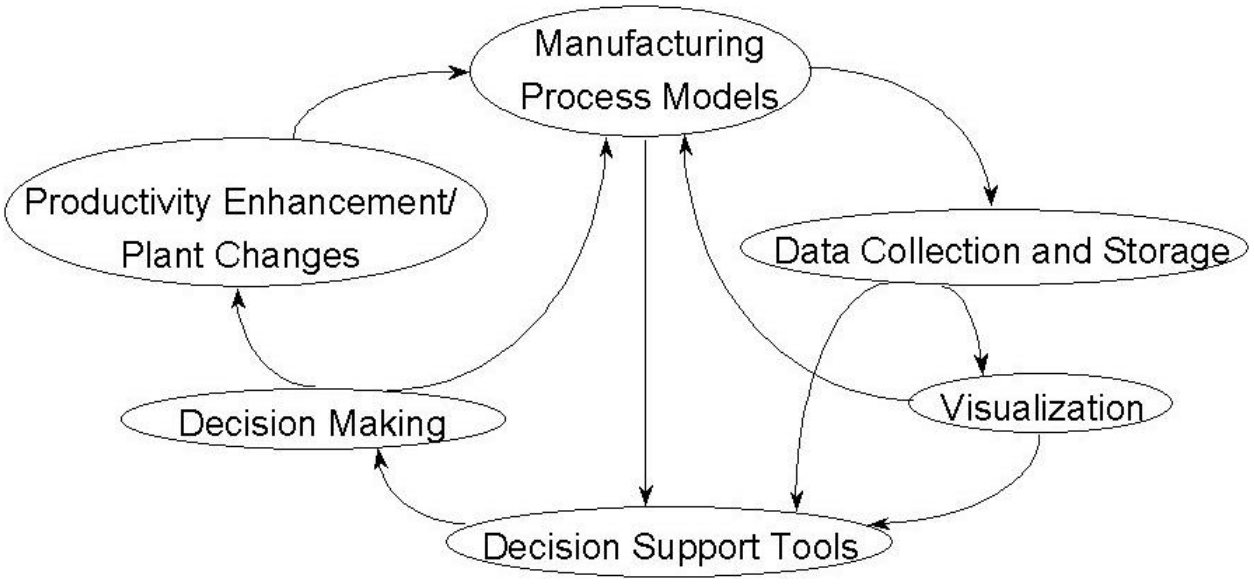

Figure 3: SSPF Engineering Process

The SSPF engineering process, is an ongoing, cyclical process that begins by modeling production data. The modeling defines what data should be collected and how it should be stored. The process then calls for visualizing and analyzing current and historical data, which supports business decisions. At this point, more data collection, visualization, and analysis might be required before changing the plant or process. If a decision changes business practices to increase throughput, such changes could affect the data model, as in the case of changes to production flow or plant configuration.

The SSPF engineering process collects, presents, stores, retrieves, and analyzes data. Saturn's data-rich environment is based on traditional process monitoring and control, and the Spring Hill site monitors and/or controls an estimated 80,000 points of live data, including production counts, downtime, and other assembly data. The majority of these data points come from programmable logic controllers (PLCs) on machines within the plant. The traditional information system collects, logs, and presents data to users via status screens configured using the Cimplicity™ data acquisition and display package.

For the SSPF engineering process, MIC was used to model the plant and its data. Entities in the plant, and their relationships and properties were identified that have a bearing on the production flow and the software used to measure and analyze that flow. The screen shot of the Graphical Modeling Editor (GME) in Figure 4 describes these entities and relations.

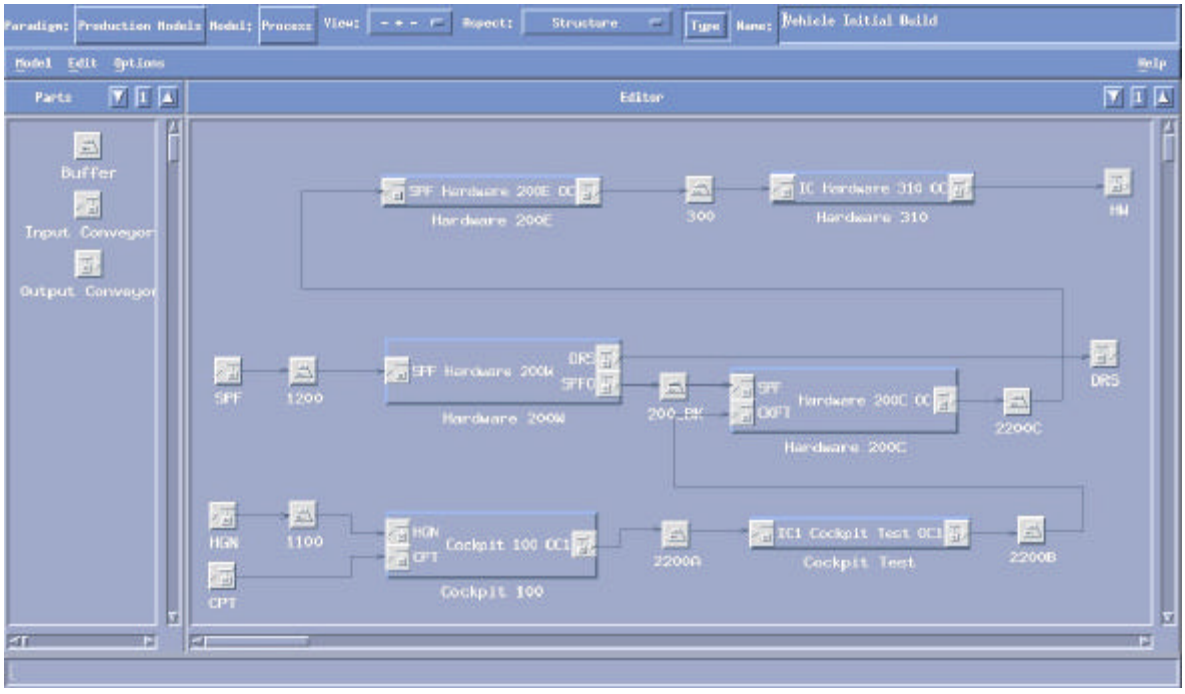

Figure 4: SSPF Model

The plant processes were modeled hierarchically, allowing the abstraction of relevant production flow information at higher levels in the hierarchy. The figure above shows the model for Vehicle Initial Build, a section of the Saturn general assembly plant. This process has five subprocesses: Cockpit 100, Cockpit Test, Hardware 200W, Hardware 200C, Hardware 200E,

and Hardware 310. The icons for the subprocesses appear in the figure. The interface points on the icons represent conveyor systems that deliver parts or subassemblies from buffers. The buffers between processes also appear (labeled 2200A, 2200B, 300, etc) The connectivity between buffers and processes represents the production flow.

Once modeled, the data must be collected in real-time. Using models to guide data acquisition facilitates the implementation by providing a structured view of the data. The plant models are used to configure the interface to Cimplicity™ and, in turn, the programmable logic controllers (PLCs). The use of models helped bring structure to the "chaos" of plant data.

Time is an essential dimension in understanding the dynamics of production flow. Some production flow dynamics remain within the bounds of a single production shift. Others involve multiple shifts, weeks, or months of history. Thus, to understand the dynamics of production flow, data collected in real-time needs to be stored for later retrieval and analysis. The stored data contains detailed histories for every process and buffer in the plant.

Data needs to be stored in a structured manner, so it can easily be retrieved and analyzed according to the structure of the plant and the production flow, which the plant models capture. Plant models are used to generate the underlying schemas for storing data, to provide data retrieval mechanisms in custom-developed and COTS components, and to ensure interoperability.

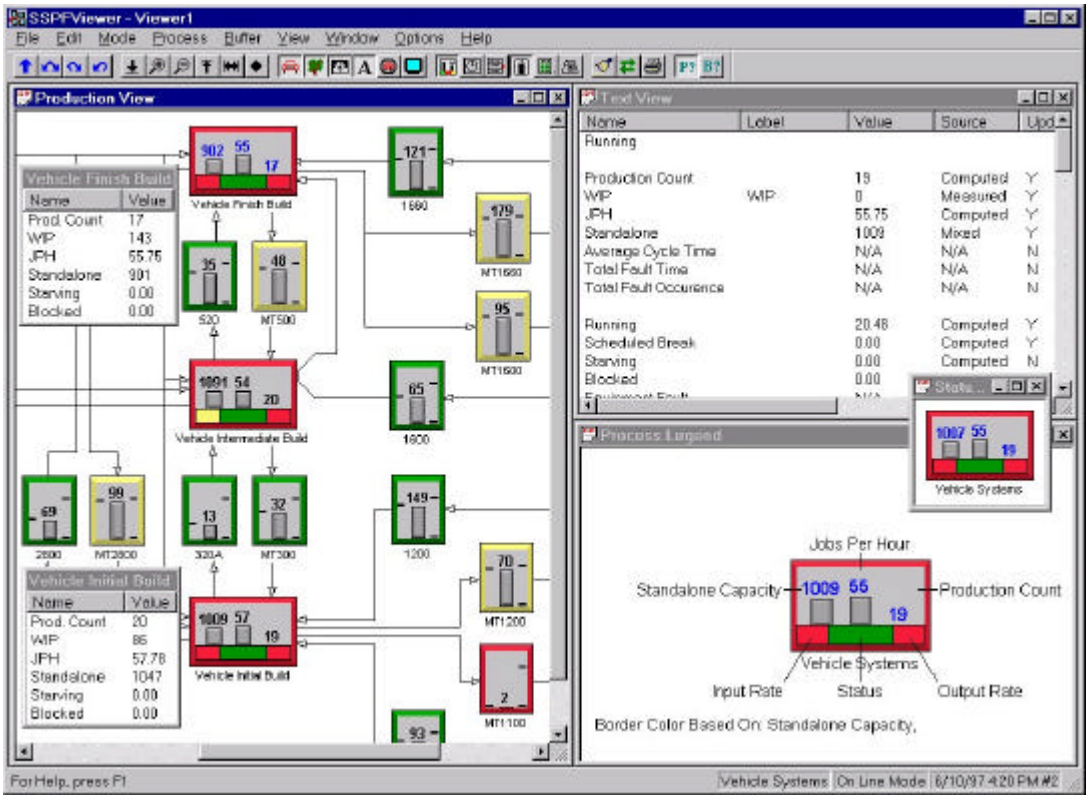

Figure 5: SSPF Viewer

The throughput task force used many different types of visualization techniques to analyze throughput for both real-time and historical data. Rather than being set in stone, it was expected that the visualization process would evolve with time. To provide a visualization tool that presents a uniform view across the plant and also changes as the plant evolves, the plant models are used to configure the visualization. The models provide the structured, uniform view of production flow that facilitates better visualization (see Figure 5).

Saturn team members use the visualization methods daily to make decisions, so the system presents data to users in a succinct and problem-oriented manner. For example, if users wish to examine downtime trends for a process or set of processes over a few weeks, they can readily access the information in visual form, using the SSPF system and tools such as Microsoft Access and Excel. The enormity and complexity of the data doesn't overwhelm users; instead, they can easily navigate to the desired data. Users need only specify the desired data for the desired process(es) and/or buffer(s), and the tools make it available.

4.2 USAF Arnold Engineering Development Center

The Computer Assisted Dynamic Data Monitoring and Analysis System (CADDMAS) was developed for USAF Arnold Engineering Development Center (AEDC) for use in ground-based, turbine engine stress testing. CADDMAS is a massively parallel, real-time, signal processing system, designed to allow flexible analysis of large quantities of high bandwidth signals.

The design goals for CADDMAS were:

- Scalability: The CADDMAS can be scaled up from 4 channels, in increments of 4. Input bandwidth is up to 50 KHz. Adding more processors and analog front-ends scales up the system.
- Performance: Large CADDMAS systems deliver over 800 MFLOPS of sustained useful computations. This processing occurs concurrently with high bandwidth input and graphical output.
- Real-Time: The system operates in a gap-less mode, processing each data packet. Display updates are soft real-time, with plot rates of more than 25 per second.
- Low Cost: The CADDMAS is implemented is low cost parallel processing technology.
- Flexibility: The algorithms supported can be expanded to meet user demands.
- Ease of Use: Mechanical and Aerospace Engineers operate the CADDMAS. The user interface needs to be designed to minimize the training required to operate the system.

Construction of the hardware to build CADDMAS systems was a relatively simple task. There were several commercial sources for this type of hardware. The difficulty of building these systems was in the management of:

- Changing System Requirements: The CADDMAS requirements change daily. No two systems are alike: in size, bandwidths, or processing requirements.
- Real-Time Requirements: The systems must satisfy not only numerical accuracy requirements, but input rate and time-to-completion specifications.

- Hardware Complexity: There can be a large number of processing nodes in the system, and a larger number of possible connections defining topologies. Furthermore, node characteristics will vary (in speed, memory, I/O)
- Software Complexity: For each hardware node, there can be many software processes, managing thousands of software components and their related, real-time communications.

In order to make these systems adaptable and configurable, software tools were necessary. A Model-Based approach for graphical specification and automatic generation of these systems was developed. A real-time kernel was developed to allow portability across architectures. The real-time kernel supported transputers, PC and Unix workstations. The kernel supports real-time processing and communication on a parallel network [Sztipanovits et al., 1998].

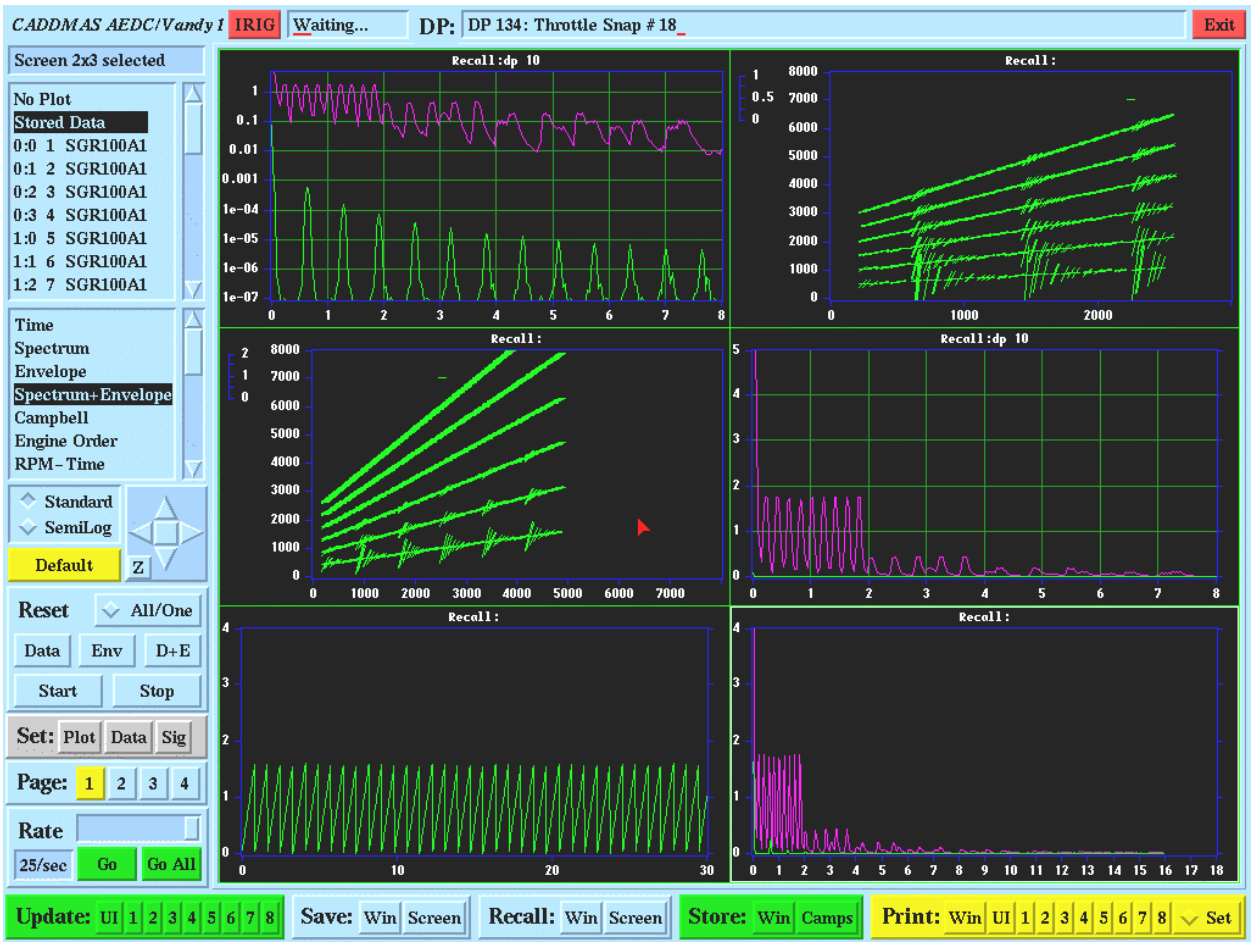

Figure 6: CADDMAS

The benefits of the CADDMAS include the capability of on-line delivery of the processed results in real-time (see Figure 6). Engines are put through rigorous testing that sometimes pushes the engines into untested performance areas. The result of exceeding limitations could be catastrophic, resulting in damage of an expensive prototype engine and/or personnel injure. To minimize the expense and potential loss, the data from the sensors must be analyzed in real-time,

with immediate feedback to the user. Information that took weeks/days to be processed is now given to the user in seconds.

5.0 Application to the C2 Domain

The evolution of the USAF into the Expeditionary Air Force (EAF) construct is driving some fundamental changes in the command and control (C2) infrastructures that enable it to apply aerospace power rapidly and more effectively in diverse operational environments. Increasingly, the National Command Authorities are directing air power employment into an ever-widening range of operations that span the breadth of the spectrum of conflict. These new crisis environments are often very dynamic in nature and demand robust and flexible C2 systems that can evolve and adapt to changing operational demands without requiring time-consuming and labor-intensive configuration changes needed today. Complicating matters further is the need for lighter, leaner forces that can be deployed into a theater and be prepared to conduct operations within hours, not days or weeks. This drive for a faster, smaller forward "footprint" has led to the concept of distributed C2, in which multiple geographically separated nodes linked together with robust networks form a "virtual" Air Operations Center (AOC), rather than a single, large, vulnerable facility, as was the case in Operation Desert Storm. So, with less equipment and fewer personnel, C2 nodes in forward locations must be able to assume multiple and evolving C2 functions as crisis environments evolve and develop.

AFRL is attempting to address many of these emerging needs in a new research and development effort entitled "Configurable Aerospace Command and Control" (CACC). The objective of this program is to develop a dynamically reconfigurable and adaptive computing and communications infrastructure that will enable expeditionary C2 with distributed operations across the spectrum of conflict. Some of the challenge problems faced by the program include:

- Physical "plug-and-play" scalability of computing and communications systems
- Global resource awareness and intelligent distributed network management
- Automated dynamic system reconfigurability
- Functional adaptivity of software tools and communications systems
- Support for legacy and emerging software systems (e.g. Theater Battle Management Core Systems [TBMCS]) as well as the future Joint Aerospace Applications (JAA).
- Interoperability with coalition and allied C2 infrastructures

To address the problem of scalable, functionally adaptive architectures, the CACC program is investigating component-based design as a foundational technology for implementation. In order for a component oriented framework to intelligently adapt and reconfigure for new or changing functional demands, some type of intelligent resource awareness and management capability will be needed to drive the configuration changes. This capability would involve some form of system monitoring to track status, load, failures, etc. of computing hardware, communications equipment, and software applications and a means to then manage their allocation and use to facilitate the functional requirements imposed on the system.

MIC technology has potential as a means to address the system monitoring functions as well as the management of the envisioned component-based architecture. C2 systems make use of large

quantities of information gathered from various sources using resources available to users, both within a command center and externally, whose interests and roles vary widely. C2 systems must integrate data from hundreds of data sources (real-time and legacy), manage complex, dynamically changing data relationships, and adapt to rapidly evolving user requirements. Limited resources must be managed and utilized efficiently. Performance and security must be verified, maintained, and monitored on-line.

MIC provides the opportunity to solve the problem(s) in real-time. MIC addresses the challenges C2 systems need for (re) configurability and adaptability to changing environment and end-user needs. MIC technology prescribes the use of models throughout the lifecycle of the system following a spiral model approach. In MIC, the models capture all the information (mission, processing, and resources) on a computer system configuration, for a particular set of requirements, and within a set of well-defined constraints. The concept of "self-adaptive" or "self-configuring" systems will require rapid system design/redesign and analysis to ensure viable configurations can be found before implementing the changes. MIC could also potentially identify problem areas, bottlenecks, or critical failure points in the current design, suggest changes for operator approval, and then instantiate the modifications.

Given some of the early examples of success in linking legacy systems (e.g. Saturn), MIC technology could serve to enable the Air Force to identify ways to integrate existing C2 applications to serve the unique functional needs of a given operational environment. Presently, most C2 systems are deployed in an "all or nothing" manner, mandating the forward deployment of considerable hardware, software and communications resources to accomplish a mission; regardless of whether all the functionality provided by the full system is necessary. Specific tailoring of resources to the particular functional requirements or airlift and time constraints is very difficult at best. The new expeditionary model demands this type of tailoring; consistent with the concept of distributed operations. MIC may be able to provide the tools needed for system administrators and JICOs (Joint Interface Control Officers) to rapidly design, prototype, and field tailored C2 software architectures to satisfy operational needs without violating lift, resource, and timing constraints.

Another critical area in which MIC technology could serve the CACC efforts is in the realm of allied and coalition interoperability. It has become conventional wisdom that U.S. forces will not employ autonomously for most future operations. The past decade has borne this out clearly, with most military operations being characterized by cooperative ventures through diverse coalition partnerships. In order for such endeavors to succeed, interoperability amongst disparate C2 systems is imperative. Often this is a very time-consuming and daunting task, if it is achieved at all. With appropriate data on allied and coalition C2 architectures, MIC technology could serve a critical function in rapidly modeling, designing, and prototyping interfaces and linkages between these dissimilar systems, protocols, and data formats. Expediting and facilitating such interoperability would be invaluable to achieving the levels of partnership, coordination, and effective communication needed to execute a successful coalition operation.

In summary, MIC technology has the potential to be a true enabler for the type of rapid and dynamic configurability envisioned for future C2 systems under the Air Force's new expeditionary paradigm. From the high-level issues of allied and coalition interoperability to bits

and wires level system adaptability and configuration, MIC is a technology with clear utility and value for realizing vital capabilities for command and control in the 21st century.

6.0 References

[Misra et al., 1997] Amit Misra, Gabor Karsai, Janos Sztipanovits, Akos Ledeczi, and Michael Moore, *A Model-Integrated Information System for Increasing Throughput in Discrete Manufacturing*, Proceedings of the International Conference and Workshop of Engineering of Computer Based Systems, March 1997

[Gates, 1998] Lana Gates, *A Sampling Of UML Analysis And Design Tools,* Application Development Trends, Volume 5, Number 10, October 1998

[Melewski, 1998] Deborah Melewski, *UML Gains Ground*, Application Development Trends, Volume 5, Number 10, October 1998

[Sztipanovits et al., 1998] Janos Sztipanovits, Gabor Karsai, and Ted Bapty, *Self-Adaptive Software for Signal Processing*, Communications of the ACM, Volume 41, Number 5, May 1998

[Long et al., 1998] Earl Long, Amit Misra, and Janos Sztipanovits, *Increasing Productivity at Saturn*, IEEE Computer, August 1998

[Nordstrom et al., 1999] Greg Nordstrom, Janos Sztipanovits, Gabor Karsai, and Akos Ledeczi, *Metamodeling – Rapid Design and Evolution of Domain-Specific Modeling Environments*, Proceedings of the IEEE ECBS'99 Conference, April 1999

[Ledeczi et al., 1999] Akos Ledeczi, Mikos Maroti, Gabor Karsai, and Greg Nordstrom, *Metaprogrammable Toolkit for Model-Integrated Computing*, Proceedings of the IEEE ECBS'99 Conference, April 1999

[Clough 1999] Jonathan Clough, *Configurable Aerospace Command and Control (CACC) Program Overview*, January 1999