

# Architecture Based Model Driven Software and System Development for Real-Time Embedded Systems

Bruce Lewis

US Army Avionics and Missile Systems Command

[Bruce.A.Lewis@us.army.mil](mailto:Bruce.A.Lewis@us.army.mil)

## Abstract.

Architecture Description Languages provide significant opportunity for the incorporation of formal methods and engineering models into the analysis of software and system architectures. A standard is being developed for embedded real-time safety critical systems which will support the use of various formal approaches to analyze the impact of the composition of systems from hardware and software and which will allow the generation of system glue code with the performance qualities predicted. The standard, the SAE Architecture Analysis & Design Language (AADL), is based on the MetaH language developed under DARPA and US Army funding and on the model driven architectural based approach demonstrated with this technology over the last 8 years. The AADL standard will include a UML profile useful for avionics, space, automotive, robotics and other real-time concurrent processing domains including safety critical applications. The paper provides an overview of the concepts supported in MetaH and the AADL as examples of the architecture based model driven paradigm and notes several new model based approaches becoming available.

## The Need for an Industry Solution

Complex embedded real-time systems are very expensive to develop, to maintain and to evolve. When system lifecycles are long compared with the commercial lifecycle of its components and when system capabilities must change to keep up with advances, evolvability becomes a key component in reducing the cost of keeping systems up to date. Families of systems also need a simple approach for evolving new members of the family from a baseline configuration. Component changes impact system performance in many areas such as speed, schedulability, reliability, and safety and must be made quickly and predictably against these cross cutting dimensions of system performance to keep costs reasonable. Predictive models in the past have been very difficult to apply due to the complexity and size of the systems and a

lack of an approach which integrates the models with system development so that the models are updated with the system, accurately reflecting the implementation.

## **A Solution with Potential for Radical Impact**

An architecture based, model driven approach provides the foundation to make computer based system changes and predict the impact. It provides analysis at the natural level for change, the software and hardware component. Component level changes that affect system performance are represented through component attributes necessary to perform the system level analysis. Component level analysis is handed with separate tools, component level development is by conventional methods, hand coded, reused or generated. The challenge is to develop sound engineering models for a class of behaviors that are at the heart of large complex real-time systems. Some very helpful models exist now that can be applied to system development. The benefit is the rapid construction to specification with predictable behavior for the development and evolution of trusted systems. An additional challenge is the extension of these concepts to new domains of system architecture.

The capture of the architectural specification and the component attributes is accomplished with an Architecture Description Language (ADL). The language must be capable of expressing the architectural requirements for the domain of application in a way that engineers working in the domain can easily specify and modify systems. The ADL and tools must permit partial specification to allow for early analysis and prototyping, with incremental completion as additional information or analysis become available. Analysis must be efficient, adequate and automated so that it will always be performed. The analysis and system building capabilities must provide support in major areas of concern in the domain. The ADL must support multiple forms of analysis so that a change in a component or a change in the architecture can check multiple areas of possible impact without changing to alternative specification languages. For instance, moving a component from one processor to another can impact scheduability on the bus, scheduability on the processor, system optimization, multi-level safety relationships and system reliability.

The MetaH ADL and toolset provides such an environment. Our experience using MetaH over the last 8 years has convinced us that architecture based model driven approaches can revolutionize the development of real-time systems.

## **Defining the Needed Models**

An engineering model is defined for the purposes of this paper as a body of knowledge that provides a clear and concise notation for describing significant system behaviors; that applies to a broad class of systems within the domain; that provides repeatable methods without custom tuning for producing good implementations; and

that is supported by analytic methods to predict and verify behavior. These criteria were used in the development of the MetaH language and code generation to select the methods that would be applied to system architectural analysis. Examples of such methods are preemptive real-time scheduling theory, fault tree and Markov chain models, and feed-back control theory, and real time message passing and timing models. These are widely used in successful engineering practice within the domain. Each was used in MetaH to define the language, construct the toolsets to analyze the specification and construct the code generator used to build the executive and to integrate the system. Good engineering models are typically produced by a combination of theoretical and empirical work. Additional models, beyond foundational models that are reflected within the ADL itself, can be added as additional analysis tools. The emerging AADL standard (based on MetaH) permits extension to the set of standard component attributes to enable the user or vendor to add new modeling approaches and analysis tools.

Engineering models are essential for many of the “ilities” we desire in system development: scalability, evolvability, dependability, composability, and reusability. Engineering models are an essential basis for requirement analysis, specification languages, and for tools to automate various development activities such as design analysis and code generation.

## **MetaH as an Example**

Since our insight into model based real-time development is through MetaH, the following sections provide some history, a high level overview and a comparison of the current typical development process to the model based process using MetaH. The paper also includes some data on the success of the approach.

## **History**

MetaH was developed over two DARPA programs and has greatly benefited from additional funding from the Open Systems - Joint Task Force. The first, Domain Specific Software Architectures (DSSA), was concerned with using domain specific system engineering knowledge to build Architecture Description Languages (ADLs) that could specify software architectures and analyze architectural properties. MetaH leverages the rapid construction of systems further by adding the automatic integration of hardware and software in accord with modeling used to analyze the system. The second, the Evolutionary Design of Complex Systems (EDCS), was focused on our ability to build systems that could be rapidly evolved with predictable impact. A third DARPA program, Dynamic Assembly for System Adaptability, Dependability and Assurance (DASADA) is leveraging MetaH as an embedded systems ADL. The Open Systems - Joint Task Force (OS-JTF) has supported analysis of open standards using MetaH as well as the standardization of the Avionics Architecture Description Language (AADL) based on MetaH.

Dr. Steve Vestal of the Honeywell Technology Center has been the principal investigator. Bruce Lewis has served as technical POC on both DARPA programs and has led the US Army Aviation and Missile Command, Research Development and Engineering Center, Software Engineering Directorate (SED) laboratory demonstrations and technology integration with MetaH since 1993. The US Navy, US Air Force, the Ada Joint Program Office, and the US Army Space and Missile Defense Command have also funded MetaH related projects.

## **A High Level Overview**

The MetaH language provides a system designer a simple but precise language for specification of architectural requirements. From the specification, the toolsets extract the formal modeling parameters for multiple analyses. MetaH was designed to integrate the multiple domains of application software in avionics on a generated architectural backplane based on formal scheduling and implementation methods while guaranteeing a hard real-time schedule. It comprehends both hardware and software components. It will generate the architecture integrating the hardware and software components into a system compliant with the modeled behavior. It provides a means to port hard real-time systems across execution environments rapidly. Current architectural analyses include schedulability, reliability and safety/security.

The language and toolset were developed to meet the requirements for building state-of-the-art modular multiprocessor systems with multilevel safety, security, and fault management. It provides for the specification and generation of dynamic multi-mode behavior across multiple processors under the real-time constraints of flight control. It also can build from the specification advanced space and time partitioned systems enabling very significant reductions (estimated at 80%) in re-validation and re-testing costs when changes are made to an avionics or mission critical system. The approach of using a combination of time and space partitioning is an emerging commercial avionics technology, now part of several fielded commercial avionics systems. Space and time partitioning is also valuable in military systems for the same reasons, especially with the recent requirements on some military avionics systems for FAA certification.

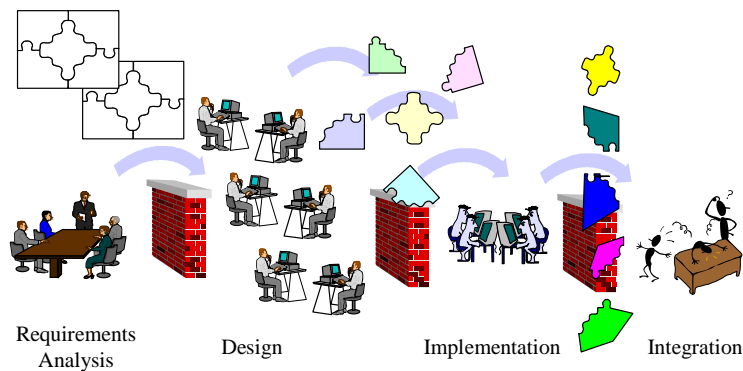
## **Model Based Development Process**

Model based development has many advantages over conventional software development and hardware / software integration. To illustrate this, we compare the current software development approach to the approach we recommend based on MetaH's capabilities for integrated modeling, architecture description, and system integration.

Figure 1 shows the current typical software development paradigm with its specification of requirements and design in various media, case tools, prototypes, paper, dis-

connected models, spreadsheets etc. Integrated Project Teams alleviate some of the communication problems in this “Over-The-Wall” approach, but this approach is characterized by specification for human interpretation and system construction. Evaluations of architecture may occur with requirements modeling tools and simulations but the results are reduced again to paper for impact on the final system software. Modeling results tend to be disconnected from the next phase and from each other. Multiple complex modeling languages are required, one for each system analysis area requiring high levels of expertise. Traceability of the models to the implementation is usually rapidly lost during development. Integration of components into a system is manual and yet must be traceable to the models if they are used. Integration is often difficult, complex and very expensive. Code generation for system or component analysis is typically for prototyping and requirements are again specified for human development of a traceable, testable integrated system.

## Current Software Process



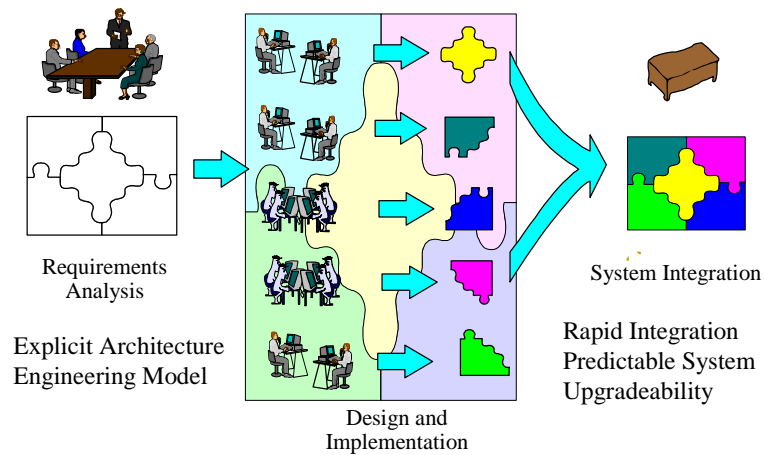
Manual, paper intensive, error prone, resistant to change

Figure 1

Figure 2 shows the architecture based model driven approach [1][8][9] which provides the ability to specify an architecture (consisting of software and hardware components, their interfaces and the system execution behavior), analyze its properties, populate it with source components, integrate hardware components, and then automatically build the system to the specification. First, using specified attributes for the hardware and software components and connections within and across processors, the architectural specification is used to model and analyze schedulability, reliability

(fault handling), and safety/security dependencies. Modeling is incremental; the user fills out the attributes required for the modeling desired. For instance if safety level is not filled out for the components, the multilevel safety analysis is not performed. The analysis for each model is provided each time the architecture is modified and regenerated so the impact can be noted in multiple areas of concern. The system is checked against these models, warning the user if any used analysis is out of limits. Different versions of software or hardware components can be stored on the “bookshelf” and easily substituted in the architecture. Architectural changes are easily made and analyzed if the models for the components (as expressed as attributes of system level interest) are provided.

## Architecture Based Model Driven AADL Process



**Figure 2**

Additional model analyzers can be added as they are defined or improved to provide modeling approaches which allow specification of more complex or larger systems. We are currently working with the University of Illinois to increase the size of the reliability models we can analyze by two orders of magnitude. The AADL draft standard provides for attribute extension allowing users to add attributes and analyzers.

For real-time architectures, the issues of schedulability, reliability and partitioning need to be understood early in safety and time critical systems [2][4][5][8]. Once the systems engineer is satisfied with the architecture, the components can be developed, reused from another project, or generated in parallel with incremental automated integration of the system with MetaH. The system is easily re-integrated through re-generation from the specification. Typically, early integrations may be on a workstation where behavior and system output can be validated. The final system would be automatically integrated from the specification and components, hardware and software, on the target platform where execution behavior and results can again be validated.

A major benefit is that the architecture and execution behavior specified is captured, not in paper or the heads of the designers, or in scattered databases but in one specification that integrates the final system and generates the executive that drives its execution. Also a single architectural specification is used for multiple formal analyses and therefore the system is generated compliant with each of the models used for analysis.

If, in the process of development, the system needs to be changed, the specification is updated and the system is rebuilt. Specification changes allow scaling across multiple processors, adding new modes, changing the execution environment (new processor(s), compilers, O/S), moving processes across processors, tuning execution rates, message passing, event propagation, substituting new components, changing process interfaces, changing error models, safety or security levels, etc. Since the processor, buses, or other hardware devices are part of the architecture they can quickly be changed to any of a predefined set. The defined set is user expandable. Execution environment dependencies reside in the toolset rather than the application code allowing rapid ports to new environments based on the toolset porting cost, not the size or complexity of the application.

The systems engineer benefits from the power of the modeling approach without having to be an expert in it or having to implement the system in conformance to it manually. The timing and data movement semantics required to implement control applications are part of the semantics of the language. So are the elements required for execution and data movement for minimal latency. The engineer has to understand how and where to specify what he needs. He constructs the architecture by specification and analyzes the impact using the models provided.

## **Evidence of Success**

The SED developed a generic missile architecture and used it to re-engineer the on-board software of an Army missile system. MetaH was used to specify the architecture interfaces and timing and to integrate re-engineered components and the dual 80960 embedded hardware together to create an executable system [3]. In addition, a 6 Degree of Freedom (6DOF) Software-In-The-Loop missile flight environment

simulation was developed to allow us to evaluate flight characteristics. It was also re-engineered into MetaH so it could be executed in real-time.

During this first development, the MetaH model based approach reduced the total effort to re-engineer the system and fly software-in-the-loop by 40 %. (Several prime contractors put the estimated savings at 66%). We later ported the application to a new execution environment using Green Hills, PowerPC and VxWorks.

### Effort Saved Using MetaH With Port

Total project 50%, Port Phase 90%

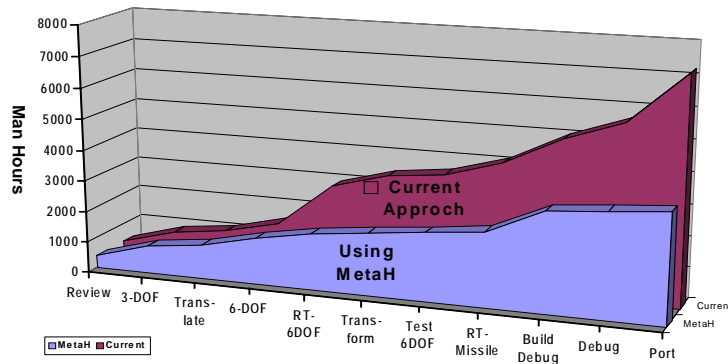


Figure 3

Our Software Engineering Directorate lab accomplished the MetaH toolset port in less than a week and the missile flew correctly in a simulation the first time it executed on the new embedded environment. Our estimated savings moved to 50% [9].

Since the missile architecture was not safety critical we did not experiment with the reliability and safety analysis. We expect that the savings provided by early analysis of reliability and safety partitioning will also provide significant benefit. Several studies of helicopter avionics architectures have been performed by Honeywell. A significant avionics experiment using the additional modeling capabilities is needed, along with other complementary technologies, to demonstrate more convincingly the benefits.

## Rising Support for Architecture Based Model Driven approaches

Based on experience from a number of experiments and applications of the technology on various DARPA programs since 1993 and based on avionics and space requirements from industry, MetaH is being used as the baseline definition for the specification of the AADL under the Society of Automotive Engineers, Avionics Division. See figure 4.

### MetaH History

1991 DARPA DSSA program begins  
 1992 First partitioned target operational (Tartan MAR/i960MC)  
 1994 First multi-processor target operational (VME i960MC)  
 1998 Portable Ada 95, POSIX executive configurations  
 Example evaluation and demonstration projects  
 Missile G&C reference architecture (AMCOM SED)  
 Missile Re-engineering demonstration (AMCOM SED)  
 Space Vehicle Attitude Control System (AMCOM SED)  
 Reconfigurable Flight Control (AMCOM SED)  
 Hybrid automata formal verification (AFOSR, Honeywell)  
 Missile defense (Boeing)  
 Fighter guidance SW fault tolerance (DARPA, CMU, Lockheed-Martin)  
 Incremental Upgrade of Legacy Systems (AFRL, Boeing, Honeywell)  
 Comanche study (AMCOM, Comanche PO, Boeing, Honeywell)  
 Tactical Mobile Robotics (DARPA, Honeywell, Georgia Tech)  
 Advanced Intercept Technology CWE (BMDO, MaxTech)  
 Adaptive Computer Systems (DARPA, Honeywell)  
 Avionics System Performance Management (AFRL, Honeywell)  
 Ada Software Integrated Development/Verification (AFRL, Honeywell)  
 FMS reference architecture (Honeywell)  
 JSF vehicle control (Honeywell)  
 IFMU reengineering (Honeywell)

**Fig. 4.**

Both European and US companies/agencies are participating in the standardization effort. The standardization committee also has a liaison relationship with NATO's Aviation Standard's Committee and we informally coordinate with the Open Group and with UML committees as we have opportunity to attend the meetings. See figure 5 below.

## AADL Standard Participants

- Bruce Lewis: Chair, technology user
- Ed Colbert: AADL & UML Mapping
- Peter Feiler: Secretary, Co-author, Editor, technology user
- Steve Vestal: MetaH Originator, Co-author

### Members

- Boeing, Rockwell, Honeywell, Lockheed Martin, Raytheon, Smith Industries
- NIST, NAVAir, OSJTF, British MOD
- Dassault Aviation, Airbus, European Space Agency, COTRE

**Fig.5.**

The same concepts that provide an Architecture Based Model Driven approach in MetaH will be part of the AADL. The standard provides a means for the commercial production of tools with a common AADL language interface. The UML profile, a specialization providing AADL semantics, will allow the application of formal analysis and code generation tools through a UML graphical specification, enabling the use of currently available UML tools for specification. The UML profile is being developed in parallel with the AADL standard and will be provided as an appendix. We also plan to provide an XML specification for the AADL language once the first version of the language standard is completed. These capabilities will provide an early interface for developing new analysis approaches. We expect to start balloting the standard by the end of 2003.

The AADL Standardization Subcommittee also has a liaison relationship with a French research consortium, COTRE, headed by Airbus. COTRE has adopted the AADL for research into new tools, development and analysis methods to support aviation system development requirements. The AADL plays a significant role in a future software and systems development approach described by Airbus and COTRE in a recent paper[10]. Other US and European companies and agencies are evaluating and experimenting with MetaH.

Architecture based, model driven approaches are also beginning to appear in the general software engineering domain. UML 2.0, the Model Driven Architectures Ini-

tative [11], will provide a new layer to UML to directly support a generalized Model driven architecture based approach. See figure 6. It is expected that multiple profiles for different domains will be defined as specializations of UML 2.0. UML 2.0 is expected to be released in mid 2003. The AADL UML profile will incorporate new architecture description capabilities from UML 2.0 when its released.

## **The Model Driven Architecture (MDA) Initiative**

- Based on the success of UML, the OMG has formulated a vision of a method of software development based on the use of models
- Key characteristic of MDA:
  - The focus and principal products of software development are models rather than programs
  - “The design is the implementation” (i.e. UML as both a modeling and an implementation language)
- UML plays a crucial role in MDA
  - Automatic code generation from UML models
  - Executable UML models
  - Requires a more precise definition of the semantics of UML (UML 2.0)

Source: Bran Selic, Rational

**Fig. 6.**

The University of Southern California, Center for Software Engineering, lead by Barry Boehm, has announced the development of Model-Based Architecting and Software Engineering (MBASE) approach [12]. This approach currently is being developed to be compatible with several Architecture Description Languages, one being the AADL.

## **Conclusions**

Initial demonstrations of the architecture based model driven approach with MetaH has shown significant promise for real-time avionics and related domains. Larger demonstrations and technology extension, including research and development of new modeling approaches will add to the paradigm's power for rapid, low cost development and evolution of computer based systems. The power of the approach is strongly related to both the models and automation of the system building process

which is expected to increase through the availability of the AADL Standard.. The standardization of the AADL with extendable attributes and a UML profile will make adding analysis approaches broadly available as well as provide the foundation for new toolsets. The release of UML 2.0 will focus significant industry interest in the model driven approach. Significant research into modeling focused on predicting system level properties from component attributes is needed and can be significantly leveraged by Architecture Based Model Driven methods to reduce the cost of system development and change.

### References:

1. Pam Binns, Matt Englehart, Mike Jackson and Steve Vestal, "Domain Specific Software Architectures for Guidance, Navigation and Control," Honeywell Technology Center, Minneapolis, MN, International Journal of Software Engineering and Knowledge Engineering, Vol6, No. 2, 1996, pages 201-227.
2. Mark H. Klein, John P. Lehoczky and Rangunathan Rajkumar, "Rate-Monotonic Analysis for Real-Time Industrial Computing," IEEE Computer, January 1994.
3. David J. McConnell, Bruce Lewis and Lisa Gray, "Reengineering a Single Threaded Embedded Missile application onto a Parallel Processing Platform using MetaH," 5<sup>th</sup> Workshop on Parallel and Distributed Real Time Systems, 1996.
4. Farnam Jahanian and Aloysius K. Mok, "Modechart:A Specification Language for Real-Time Systems," IEEE Transactions on Software Engineering, V20, N12, December, 1994.
5. Andrew L. Reibman and Malathi Veeraraghavan, "Reliability Modeling: An Overview for Systems Engineers," IEEE Computer, April 1991.
6. Software Considerations in Airborne Systems and Equipment Certification, RTCA/DO-178B, RTCA, Inc., Washington D.C., December 1992.
7. Pam Binns, "Scheduling slack in MetaH," Real-Time Systems Symposium, December, 1994.
8. Jonathan W. Kruger, Steve Vestal, Bruce Lewis, "Fitting the Pieces Together: System/Software Analysis and Code Integration Using MetaH, " Digital Avionics Systems Conference, 1998.
9. Peter H. Feiler, Bruce Lewis, Steve Vestal, "Improving Predictability in Embedded Real-time Systems," Carnegie Mellon Software Engineering Institute, CMU/SEI-2000-SR-011, October 2000.
10. Patrick Farail, Pierre Dissaux, "COTRE a Software Design Workshop", DASIA 2002, May 2002.
11. Bran Selic, "Performance Oriented UML", Tutorial, 3<sup>rd</sup> International Workshop On Software and Performance, July 2002.
12. Barry Boehm, Overview, Mini Tutorial, <http://sunset.usc.edu/research/MBASE/>