

Integration of an Analysis Tool for Large-Scale Embedded Real-Time Software into a Vehicle Control Platform Development Tool Chain*

Xiaofeng Yin
Department of Automotive Engineering
Xihua University
Chengdu, Sichuan, P. R. China

Daniel L. Kiskis, Daniel Mihalik, and Kang G. Shin
Department of EECS
University of Michigan
Ann Arbor, MI, U.S.A.

Abstract - We present a software tool for high-level design and analysis of large-scale embedded real-time software, which has been integrated into a vehicle control platform development tool chain. The integration framework of the tool chain is based on CORBA services, and the data exchanges among different tools are carried out through an analysis interchange format (AIF) expressed in XML schema. Our tool subscribes the document containing the software design and deployment models via the tool integration framework, performs structural and timing analysis, updates the document with analysis results, and publishes the updated document back via the same integration framework to our analysis result display tool, which is implemented as a plugin to Eclipse. By extracting and analyzing relevant information in the models of automotive embedded control software, our tool can find structural errors, analyze schedulability, and provide design recommendation/automation in the early stages of system design.

Keywords: Analysis, modeling, embedded real-time software, automotive control systems, tool integration.

1.0 Introduction

With the increasing complexity and stringent safety-critical requirements of automotive embedded control systems, domain-specific languages such as Matlab Simulink/Stateflow [12] are widely used in the typical design process of automotive embedded software. Simulink/Stateflow is used to design control algorithms, and then corresponding C code is generated via automatic code generators such as Real-Time Workshop (RTW). The automatically-generated

code can be compiled and downloaded to the target platform, where it is tested and run. These tools can simplify the work of programming and shorten development cycle to some extent.

However, current automotive embedded software development pays little attention to non-functional requirements, such as timing constraints, until the end of the development process – testing code on the target platform. If the system’s ability to meet timing constraints is checked early in the design process, it is possible to avoid costly late-stage redesigns that may cause the project’s deadline to be missed. In addition, there is still a lack of tools for detecting software structural errors in current development process, and structural errors are usually not easily detected through the final system test process.

We address the problem of early-stage checking of non-functional and structural properties of embedded software through the integration of a software analysis tool into an embedded system development tool chain. The software tool, AIRES (Automatic Integration of Reusable Embedded Software), was developed at the University of Michigan to assist in high-level design and analysis of large-scale embedded real-time software. It is targeted towards both automotive embedded control and avionics mission computing domains. It enables the design engineers to perform early analysis on software structure and system timing behavior.

The development tool chain we consider is VCP (Vehicle Control Platform) [3]. It contains a set of tools developed by multiple institutions to support the whole life cycle of embedded real-time software development, targeting automotive embedded control systems with hard real-time requirements and platform-specific optimizations. It covers modeling, analysis, synthesis, verification, simulation, etc., and was developed with the philosophy that integrated

*The work reported in this paper was supported in part by ESCHER Institute and Ford Motor Company.

tools working together contribute to the success of development processes more than individually applying non-integrated tools to different aspects of the design process.

This paper is focused on describing how to integrate the AIRES analysis tool into the VCP tool chain. For further information on domain-specific modeling and detailed analysis algorithms, we recommend the interested reader to refer to [11] and [5], respectively.

The rest of this paper is organized as follows. Section 2 introduces the integration framework developed by the researchers at Vanderbilt University. Section 3 describes an XML-based analysis interchange format (AIF) used for sharing models among different tools and AIRES via the integration framework. Section 4 discusses various algorithms (or functions) fulfilled in current AIRES tool in the VCP tool chain. Section 5 explains AIRES tool integration process. This paper concludes with Section 6.

2.0 Integration Framework

We use WOTIF (Web-based Open Tool Integration Framework), formerly known as OTIF [6], as the integration framework for AIRES analysis tool integration. It is an open framework for integrating design tools for embedded system development. It provides a meta-model driven infrastructure for design tool integration, which facilitates semantic interoperability across the elements of a tool chain. Elements of a (W)OTIF architecture for a simple, 3-tool integration solution are shown in Figure 1 [17]. Various tools can connect to the CORBA-based backplane via tool adaptors. The backplane facilitates data interchange by scheduling and executing appropriate transformations on the data (using translator elements), in a manner that is compliant with a modeled workflow across the tools.

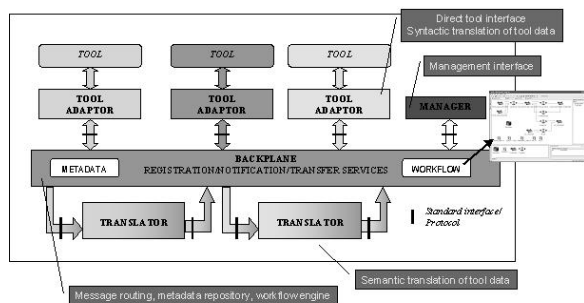


Figure 1. (W)OTIF architecture for a simple, 3-tool integration solution provided by Vanderbilt University.

3.0 An XML-based Analysis Interchange Format - xAIF

To provide a mechanism for sharing models of embedded systems among modeling and analysis tools, we developed an XML-based Analysis Interchange Format – xAIF, which is based on AIF, an interchange format developed for the Avionics Open Experimental Platform in the former DARPA MoBIES (Model-Based Integration of Embedded Software) program. xAIF modifies and extends AIF to include modeling constructs applicable to other embedded system domains, especially for automotive control systems.

xAIF mainly changes AIF in four aspects. First, it modifies the system model to capture important timing and performance information that is unavailable in AIF, but is present in design models. Second, it adds additional modeling constructs to capture system runtime information, including component allocation, thread definitions and scheduling parameters. Third, to reduce the complexity of passing analysis result from AIRES tool to our Eclipse-based display software via (W)OTIF, it adds an entity called *AnalysisReport* at the same level as the *System* entity. Fourth, it adds the ability to model the runtime structure of both avionics and automotive systems. This extension is required to allow for the analysis of models expressed in ECSL-DP (Embedded Control Systems Language for Distributed Processing) [14], a domain-specific graphical modeling language developed at Vanderbilt University for automotive control systems. ECSL-DP uses a different runtime model from that of ESML (Embedded Systems Modeling Language) [7], which is aimed at modeling avionics applications.

We use GME (Generic Modeling Environment) [11] to define xAIF in terms of UML class diagrams, in which the following models are used to represent an embedded system:

- The *DocumentStructure* model defines the sections of the xAIF document as a whole, in which the *RootFolder* corresponds to the xAIF document and is composed of two optional objects, an *AnalysisReport* and a *System*. The *AnalysisReport* encapsulates the message and error reporting content generated by analysis tools. The *System* encapsulates the complete embedded system model, including the *Software*, *Hardware*, and *SwExecution* models.
- The *Software* model (shown as Figure 2) defines the relationships between software components. It is rooted in a *System*, which encapsulates the concept that an embedded software system consists of a set of *Interactions* that operate across

of a set of *Networks* via a set of *Interfaces*. *Interactions* model the control flow in the embedded software. An interaction is a sequence of actions of *ComponentInstances* that are invoked via *Events* or *Method* calls. An *Interface* represents the exposed interface for a remote procedure call, which can be either a *Facet* on the invoked component or a *Receptacle* on the invoking component. Components are modeled by *ComponentInstances*, which come in two subclasses: *ApplicationComponents* and *SupportComponents*. In addition to the direct invocations (*Facets* / *Receptacles*), components also have event-based interactions through publishing or

subscribing *Events* via *EventPorts* (*PublishPorts* / *SubscribePorts*). In xAIF, a timer is modeled by a *TimerComponent* (with attributes *isPeriodic*, *interval*, *maximum* and *resolution*), which acts as a publish port to generate a timeout event at a specific time. The arrival of event on a subscribe port triggers an *Action* that has an associated worst-case execution time (WCET) property. In xAIF, real-time properties such as aggregate worst-case execution time, period, deadline, priority, etc., are captured through two association classes: *InvocationQoSProperties* and *ActionQoSProperties*.

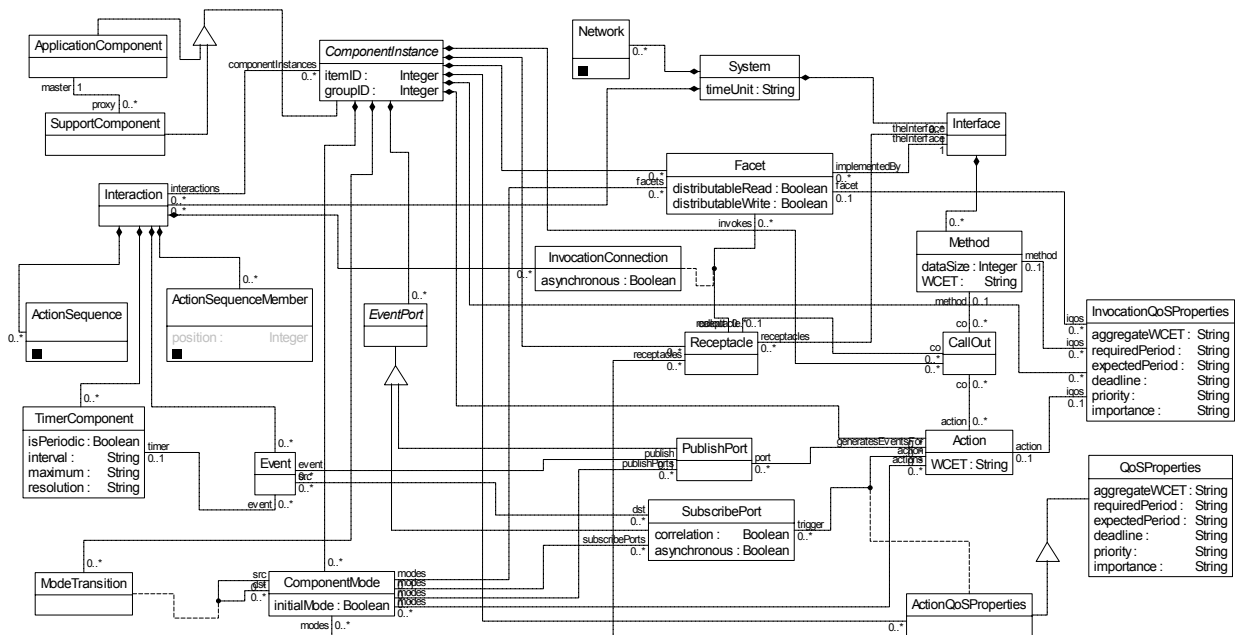


Figure 2. Software model.

The *Hardware* model (shown as Figure 3) defines the execution platform by describing hardware configurations, how components are allocated to processors, and how fault management is implemented. The hardware model is rooted in a *Network* that contains *Processors* that contain *Processes* that contain *Threads* (*Thread* is defined in the *SwExecution* model). *Processors* in the model are connected by communication *Links* (such as a CAN bus) via *CommPorts*. Processes have *ComponentInstances* associated with them through two kinds of associations: *NormalComponent* associations are used when there are no failures in the system, while

BackupComponent associations are used when failures exist; which of them is activated is determined by the *SystemMode* (Normal or Fault mode). xAIF uses *Resource* (with 3 subclasses: *Processor*, *Link*, and *SupportSW*) to define the allocatable resources in the embedded system. *SupportSW* models the underlying supporting software, including OS and communication software for the hardware being modeled. It consists of *TimingService*, *SchedService*, and *CommService*, which model the timing service, scheduling algorithms, and communication mechanisms, respectively.

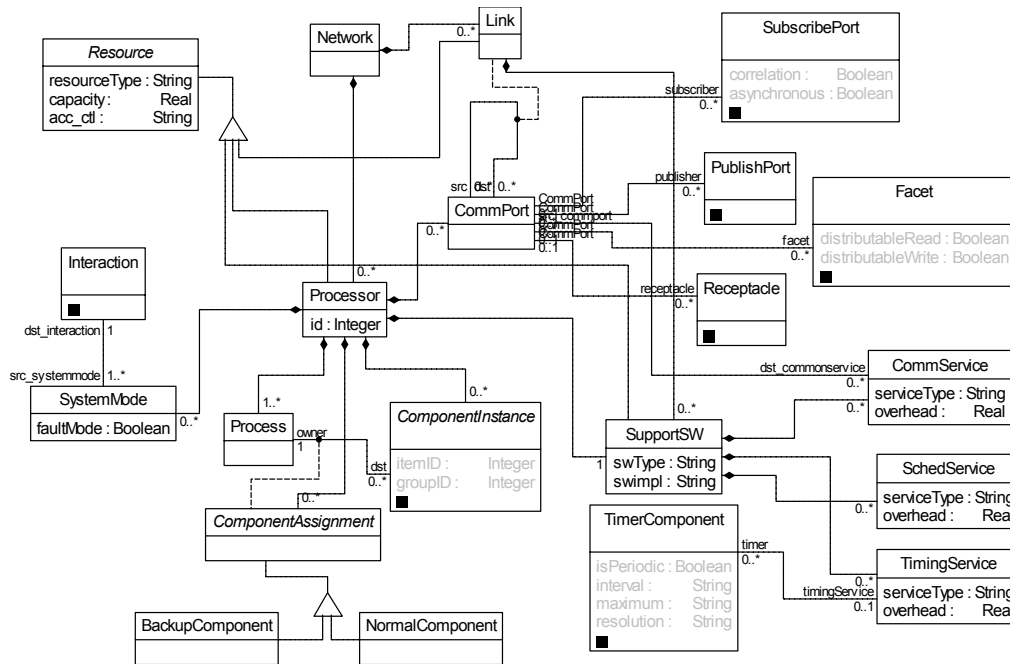


Figure 3. Hardware model.

- The *SwExecution* model (shown as Figure 4) defines the runtime structure of the embedded software. It is modeled as a set of *Processes* that contain *Threads* (with attributes describing priority, deadline, offset) that use block-waiting mechanism and may have a *Trigger* (*TimerComponent* or *SubscribePort*) for their execution. The execution of a thread is also subject to the OS *SchedService*. xAIF uses a *ThreadSequence*, with

attributes describing the end to end period and end to end deadline, to define a sequence of threads that are executed in order. In addition, each thread may have a *ThreadQoSProperties* and/or an *InvocationQoSProperties* object associated with it. *ThreadQoSProperties* has the same set of real-time properties as *InvocationQoSProperties*.

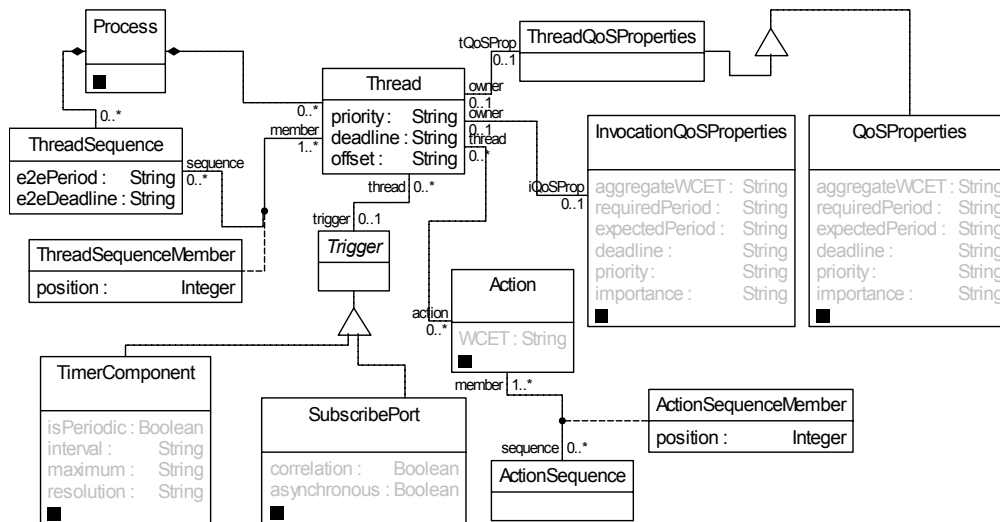


Figure 4. SwExecution model.

- The *Reports* model defines the format of information reports generated by analysis tools. It is rooted in an *AnalysisReport* that contains all textual information, including *Error* messages and *Information* messages, which are passed back to the user.

The xAIF document defined by these UML class diagrams can then be used to generate an XSD (XML Schema Definition) document, by the UDM (Universal Data Model) tool, to provide a sharing mechanism among modeling and analysis tools. For more details on UDM tool, please refer to [1].

4.0 AIRES Tool

The original AIRES tool was developed for the DARPA MoBIES program. The purpose of integrating it into the VCP development tool chain is to meet the goal of ESCHER Project, i.e., enabling the transition of government-funded information technology research results to industry and to the research community [3]. Therefore, the goals of this integration effort are to develop techniques to assist large-scale embedded software design, analysis, and automation; to construct a software toolkit to implement these techniques; and to make the toolkit available and support its usage for real world applications.

There exists another version of the AIRES tool developed by the University of Michigan, which is built based on GME, and covers most aspects of the design process for automotive embedded control systems, such as automotive embedded control software model importing, structural modeling, platform modeling, component allocation, task forming, network message importation, schedulability analysis, design refinement, and code generation. To avoid confusion with the GME-based AIRES, we call the AIRES tool integrated with VCP tool chain the (W)OTIF-based AIRES tool. In this section, we will only briefly describe various analysis algorithms fulfilled in the (W)OTIF-based AIRES tool. For further information about the AIRES tool, such as modeling approaches, detailed (more complex) analysis algorithms, as well as design automation, etc., we recommend the interested reader to refer to [15], [4], [5], [9], [16] and [13].

At present, the (W)OTIF-based AIRES tool carries out three functions: dependency analysis, timing analysis, and design recommendation/automation.

The (W)OTIF-based AIRES tool extracts system information from an xAIF document generated from an ESML model or an ECSL-DP model. It constructs

a direct graph called a Port Dependency Graph (PDG), in which each node is a port (either a publish/subscribe port or a facet/receptacle invocation), and each edge denotes dependencies between ports. The PDG captures both inter- and intra- component dependencies of the modeled embedded software. We also derive a Component Dependency Port (CDG) from the PDG for component-level analysis. Based on the constructed PDG and CDG, the AIRES tool uses traditional graph algorithms to analyze system dependencies and tries to find out if there are any potential structural errors contained in the software models, through: (1) dependency cycle checking to make sure the graphs are acyclic; (2) identifying if there are events present without a subscriber or publisher; (3) identifying unreachable components by checking if there exists any action which is not a part of the CDG; (4) reachability testing by a recursive Depth-First Search (DFS) [2] based rate assignment algorithm.

Timing analysis is used to determine if the tasks (same as threads in this paper) formed from the various actions and their corresponding components are schedulable. The algorithm is shown as Figure 5. The reachability test for dependency analysis is actually implemented in the process of timing analysis.

Algorithm TA: Timing analysis

BEGIN

foreach timer component

perform DFS search on the PDG starting from timer component's publish port and assign the timer component's interval to each visited node;

foreach component

set interval as that of its input port;
calculate utilization as the ratio of its input port's WCET to its interval;

foreach component

if interval is zero then mark it as unreachable;

foreach processor

calculate WCET as the sum of WCET for all tasks running on it;
calculate utilization;

if utilization is greater than the utilization bound of that processor then mark it as unschedulable;

else perform standard RMA analysis [8];

END

Figure 5. Timing analysis algorithm.

The design automation currently implemented in (W)OTIF-based AIRES tool is component allocation, i.e., automatically allocating software components to

the processors in the platform model. We have implemented two simple allocation algorithms, load-balancing and first-fit, with the goals of balancing the utilization of all of the processors and assigning the components to as few processors as possible, respectively. Both of them consider only the utilization bound of the processors, and ignore the communication utilization bound and the memory bound.

5.0 Analysis Tool Integration

Figure 6 shows an example of how the AIRES tool is integrated into the VCP development tool chain using (W)OTIF to produce an analysis workflow for automotive software development. Two steps must be implemented before the analysis can be performed: first, models of the automotive embedded system are constructed using the ECSL-DP meta-modeling language, e.g., in the GME environment; second, the ECSL-DP model is translated to an xAIF model using the ECSL-DP2xAIF translator, a model transformation tool developed at the University of Michigan, based on the ECSL-DP2AIF translator developed by Vanderbilt University.

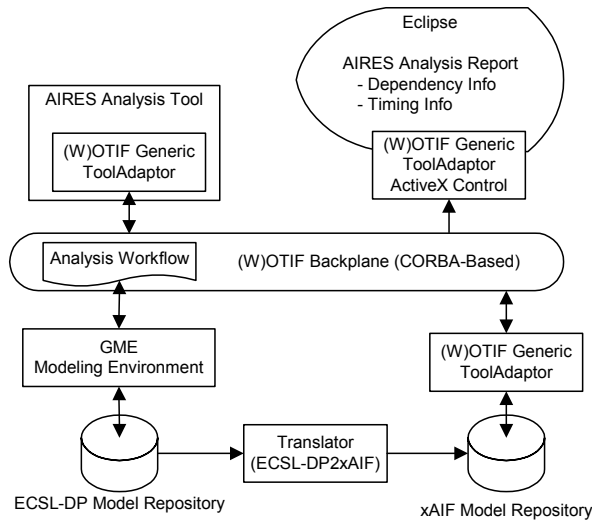


Figure 6. Analysis tool integration.

Once the xAIF model is ready, the AIRES tool can be used for analysis, following the workflow specified in Figure 7. The workflow model is first built using the WFM paradigm [10] in GME, and then registered to the (W)OTIF backplane via the WFM interpreter to define the invocation sequence for different tools in the tool chain. As shown in Figure 7, our analysis workflow defines three different types of tool adaptors. The xAIF_UPDATER corresponds to the tool adaptor

between the xAIF model repository and the backplane. It is used to publish the xAIF document containing the models of the automotive embedded control system. The xAIF_RECEIVER_PUBLISHER is used by AIRES analysis tool to subscribe to the xAIF document and publish the modified xAIF document to the backplane. The xAIF_RECEIVER is used by our analysis result display tool (implemented as an Eclipse *view* plugin) to parse relevant information from the modified xAIF document.

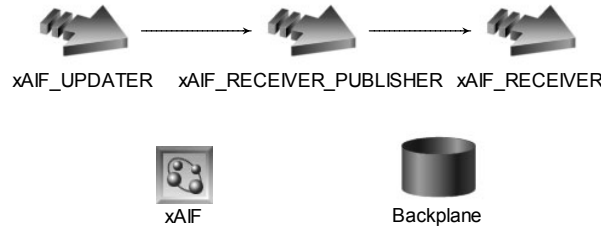


Figure 7. Analysis workflow model.

As a consequence of using open standards such as CORBA and XML, third-party tools will be easily integrated into the VCP development tool chain and made interoperable with the AIRES analysis tool.

6.0 Conclusion and Future Work

We have integrated a structural and timing analysis tool into a vehicle control platform development tool chain. Our approaches provide design engineers in the domain of automotive embedded control with a chance to identify structural errors and analyze real-time characteristics of their automotive embedded control software. They also help them make high-level design decisions at an early design stage that are important for reducing cost and time compared with the current widely-used software development practices. Current practices only deal with non-functional aspects of the design and are generally applied near the end of the development process through time-consuming and usually expensive testing on the target platform.

As part of our future work, we are considering building a tool to translate an xAIF model to an ECSL-DP model such that the user can directly make use of the component allocation provided by AIRES to assist design automation. As a matter of fact, we have already developed a number of algorithms for design automation in the GME-based AIRES tool, such as component allocation with multiple resource constraints [16], priority refinement for dependent tasks [13], etc. Therefore, transferring them into the

(W)OTIF-based AIRES tool and constructing a mechanism convenient for the user to select different algorithms will also be considered in our future work.

7.0 Acknowledgements

We would like to thank the researchers at Vanderbilt University for providing the fundamental framework and other supporting tools for our analysis tool integration.

8.0 References

- [1] Bakay, A., and Magyari, E. *The UDM Framework*. Institute for Software-Integrated Systems, Vanderbilt University, Nashville, TN, October 2004.
- [2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, Second Edition*. McGraw Hill Publishers, 2001.
- [3] Escher Research Institute website: <http://www.escherinstitute.org/toolchains/vcp>
- [4] Gu, Z., Kodase, S., Wang, S., Kim, J. C., and Shin, K. G. Integrated modeling and analysis of embedded control systems with real-time scheduling. *In-Vehicle Networks and Software, Electrical Wiring Harnesses, and Electronics and Systems Reliability (SP-1852), 2004 SAE World Congress*. Detroit, Michigan, March 8-11, 2004
- [5] Gu, Z., Kodase, S., Wang, S., and Shin, K. G. A model-based approach to system-level dependency and real-time analysis of embedded software. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2003)*. Toronto, Canada, May 27-30, 2003
- [6] Karsai, G., Lang, A., and Neema, S. Tool integration patterns. In *Proceedings of Workshop on Tool Integration in System Development, European Software Engineering Conference*. Helsinki, Finland, September 2003
- [7] Karsai, G., Neema, S., Bakay, A., Ledeczi, A., Shi, F., and Gokhale, A. A model-based front-end to TAO/ACE: the embedded system modeling language. In *Proceedings of the 2nd Workshop on The ACE ORB (TAO)*. Arlington, VA, USA, July 19, 2002
- [8] Klein, M. H., Ralya, T., Pollak, B., and Obenza, R. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, 1993.
- [9] Kodase, S., Wang, S., and Shin, K. G. Transforming structural model to runtime model of embedded software with real-time constraints. In *Proceedings of 2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003)*. Munich, Germany, March 3-7, 2003.
- [10] Lang, A. *OTIF Usage Guide*. Institute for Software-Integrated Systems, Vanderbilt University, Nashville, TN, March 2004.
- [11] Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., and Volgyesi, P. The generic modeling environment. In *Proceedings of IEEE International Workshop on Intelligent Signal Processing (WISP'2001)*. Budapest, Hungary, May 24-25, 2001.
- [12] Mathworks website: <http://www.mathworks.com>
- [13] Merrick, J. R., Wang, S., and Shin, K. G. Priority refinement for dependent tasks in large embedded real-time software. In *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2005)*. San Francisco, CA, USA, March 7-10, 2005.
- [14] Neema, S., Karsai, G., and Vizhanyo A. Embedded Control Systems Language for Distributed Processing (ECSL-DP). In *Proceedings of the OMG's First Annual Model-Integrated Computing Workshop*. Arlington, VA, USA, October 12-15, 2004
- [15] Wang, S., and Shin, K. G. Early-stage performance modeling and its application for integrated embedded control software design. In *Proceedings of the Fourth International Workshop on Software and Performance (WOSP 2004)*. Redwood Shores, California, USA, January 14-16, 2004
- [16] Wang, S., Merrick, J. R., and Shin, K. G. Component allocation with multiple resource constraints for large embedded real-time software design. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004)*. Toronto, Canada, May 25-28, 2004.
- [17] (W)OTIF website: <http://www.escherinstitute.org/frameworks/otif>