An Integrated Toolset for Distributed Real-Time Systems Based on Computational Grid

Lichen Zhang
Faculty of Computer Science and Technology
Guangdong University of Technology , 510090 Guangzhou
GuangDong Province, P.R. of China
Email: lchzhang@gdut.edu.cn

Abstract: Advances in networking infrastructure have led to the development of a new type of "computational grid" infrastructure that provides predictable, consistent and uniform access to geographically distributed resources such as computers, data repositories, scientific instruments, and advanced display devices . Such Grid environments are being used to construct sophisticated, performance-sensitive applications in such areas as dynamic, distributed real-time applications. In this paper, we propose a toolset for designing distributed real-time systems based on computational grid.The toolset is based on a new methodology and integrates the models that methodology proposed for designing real-time systems.

1. Introduction

As the associated human community, instruments, and resources required for data processing become increasingly distributed, real-time online instrument systems connected by wide area networks will be the norm for scientific, medical, and similar data-generating systems. Such systems have rigorous Quality of Service (QoS) objectives. They must behave in a dependable manner, must respond to threats in a timely fashion and must provide continuous availability, even within hazardous and unknown environments. Furthermore, resources should be utilized in an efficient manner, and scalability must be provided to address the ever-increasing complexity of scenarios that confront such systems. The difficulties in engineering such systems arise from several phenomena, one of the most perplexing being the dynamic environments in which they must function. Systems which operate in dynamic environments may have unknown worst-case scenarios, may have large variances in the sizes of the data and event sets that they process (and thus, have large variances in execution latencies and resource requirements), and cannot be characterized (accurately) by constants, by intervals or even by time-invariant statistical distributions. This environment gives rise to the need for a variety of capabilities: dynamically schedulable resources, easily administered and enforced use conditions and access control for all elements, systems designed to adapt to varying conditions in the distributed environment, automated control and guidance systems that facilitate remote (in time, space and scale) operations, and a myriad of reservation and scheduling capabilities for all of the resources involved.

Advances in networking infrastructure have led to the development of a new type of "computational grid" infrastructure that provides predictable, consistent and uniform access to geographically distributed resources such as computers, data repositories, scientific instruments, and advanced display devices . Such Grid environments are being used to construct sophisticated, performance-sensitive applications in such areas as dynamic, distributed real-time applications.

It seems fair to say those current methods and tools that aid in the development of complex real-time systems are still underdeveloped. Numerous methods and tools have been proposed for the design of such systems. However, with the current state of practice, none of these methods and tools correctly addresses all kinds of user requirements. Although designers typically commit to one method, they often find that its notation is not rich enough to express some semantic concepts or that it lacks the guidance needed to choose design entities. Consequently, they attempt to borrow useful ideas and notations from other methods.

The objective of our work is to propose an integration toolset, which supports a single design methodology that covers all phases of the life cycle, ensuring that specific real-time requirements of the software will be met, even on Computational Grid as target. A key issue in current distributed real-time system development is the desire to integrate various analysis and design methods and tools that address different aspects of development process. Typically, different authors, each of whom has chosen to focus on a specific part of the overall problem, produce these methods and tools. Users of these methods and tools want them to work together and fully support the user's design and development process. Thus, method and tool integration is intended to provide an approach that support the entire software development life cycle.

This paper is organized as follows. In section 2 we outline the main aspects and characteristics of our methodology supported by the integration toolset. In section 3, we describe the main aspects and characteristics of the integration toolset. In section 4, we summarize the features of our integration tools and discuss current, ongoing work.

2. Methodology Supported by Integration Toolset

The primary goal of a software development methodology is to facilitate the creation, communication, verification and tracing of requirements, design, and implementation. To be truly effective, a modern methodology must also automatically produce implementations from designs, test cases from requirement specifications, analyses of designs and reusable component libraries. Our experience in the development of real-time systems and research into software development methodologies have led us to conclude that existing public-domain methodologies do not permit us to achieve these goals. Most often software development methods offer excellent solutions to specific, partial aspects of system development, providing only little of help for other aspects. Classical methods for specifying and analyzing real-time systems typically focus on a limited subset of system characteristics. RTSA [18], for example, focuses primarily on the functionality and state-oriented behavior of real-time systems. STATEMATE [9] provides tree different graphic or diagrammatic languages. Module charts represent the structural view of the system, activity charts represent functionality, and state charts represent the behavior view of the system. At the other extreme, formal specification and verification methods strive for fool-proof or error-free designs. they can be used to specifying and analyzing some properties of a system. Thus integration between informal specification methods and formal specification methods is desired:

- the integration of the methods used to specify systems requirements.
- the integration of tools that support these methods and
- the integration of the multiple specification fragments produced by applying these methods and tools.

In our opinion, an acceptable real-time system design methodology must synthesize the different views of systems, use a number of different methods and tools, consist of an orderly series of steps to assure that all aspects of the requirement and all the design aspects have been considered. The real-time system design methodology should address these problems:

- supporting specification and analysis of a system from multiple viewpoints to capture different perspectives on a system,
- providing methods and tools that are appropriate for different viewpoints for improved understandability of specifications,
- employing a formal basis to integrate multiple viewpoints and perform analyses with mathematical rigor, and
- providing methods to handle the size and complexity required by large-scale systems.

Hence, the most obvious difference between current methodologies and our approach is that we apply a methodology which explicitly avoid the use of a simple framework to design complex systems. Instead, we advocate a multi-view objects-oriented approach to the development of real-time systems, based on the identification of complementary views on the system as starting point for modeling and design. These five views are: environmental view, the functional view, the behavioral view, the performance view, the implementation view. Each of these views will be discussed in detail below.

The environmental view: The environment in which a real-time system is to operate plays an important role in the design of the system. Embedded computer systems have to react quickly and correctly complex sequence of external events. The entities that produce these external events are collectively named "environment". Many environments are very well defined. Designers think of these as deterministic environments. These give rise to small, static real-time system. However, some environments are dynamic and nondeterministic environment. These give rise to large, complex and dynamic real-time systems. Since the behavior of the system is strongly coupled to the behavior of the environment in this application domain, we propose a view of environment as part of the development of the system. In building this view, we are concerned with:

the identification of objects in the environment of system;
the determination of operations and events related to these objects;
the description of the behavior of each object;
the collection of information from each object and presentation of information within each object;
the specification of the timing constraints of the output of each object;
the knowledge representation of environments is as basis of specifying requirements.

The functional view: The functional view captures the static structure of the system, it addresses questions on functionality of the system, i.e. the input and output, what the subfunctions are, and how these functions are combined. This view is necessary because the objective to construct a system lies in the realization of functionality that an user demands.

The behavioral view: The behavioral view captures the dynamics of the system, i.e., the conditions under which the functionality is performed. The changes of internal behavior of system are caused by the change of the behavior of external environment. Some forms of state machine was useful in analyzing the system behavior. however, this type of analysis is limited since the number of states required for an embedded computer system is too large, it is desirable to use high-level automata in which multiple states can be active at any moment of time, multiple transitions can be fire simultaneously, states can be decomposed hierarchically into lower-level machines, and there exists a rich language for representing logical and temporal dependencies among states and transitions.

The performance view: Performance is an important but often neglected aspect of software development methodologies. Performance refers to system responsiveness: the time required to respond to specific events, or to the number of the events processed in a given time internal. Other than for e.g. information systems, and even for soft real-times, in hard real-time systems, performance issues are correctness issues. In real-time systems, performance requirements are a major concern. Due to the time criticalness of such an application, performance analysis and optimization becomes prominent. Hence, we need a performance view, in which the performance of the artifacts under construction is critically inspected, and where the performance of the resulting system is kept under control within the development process.

The implementation view: Real-time embedded systems are often constrained by the target environment, they are supposed to be implemented on. The ***implementation*** view aims at supporting the process of fitting the specification onto a particular target hardware environment. A developer must realize what hardware is used and how it is used to implement the specification. Therefore, the hardware view aims at providing means for evaluating particular configurations with respect to a particular application. The view can be represented by a pictorial representation of the system showing how the hardware is configured and how the tasks are implemented.

3. Main Aspects and Characteristics of Integrated Toolset

The underlying premise of the integrated toolset is to use the objects-oriented techniques to specify and analyze the real-time systems under development from five separate, but related, point of view: environment, function, behavior, performance, implementation. For these five views, the integrated working environment provides different graphical, diagrammatic languages. These graphical languages are based on a common set of simple graphical editors that check for syntactic validity as the specifications are developed, and more importantly, with formal semantics that are embedded. Especially, UML-RT[4], performance analysis tools and RT-CORBA[3] can be integrated together for designing distributed real-time systems based on computational grid.

UML-RT is an extension of the UML built on concepts from ROOM. It provides different models for specify real-time systems from different points of view. For the external environment of real-time system, this is expressed as an external event context in UML-RT. The external event context is expressed as an object model in which the system itself is treated as a single black-box composite object sending and receiving messages to external actor objects. The functionality of the systems is expressed as the Use Case model. The Use case and scenario models decompose the primary functionality of the system and the protocols necessary to meet these functional requirements. From the point of view of behavior modeling each capsule is associated to a statechart specifying states and states transition of the capsule. A capsule is a specialized active class and is used for modeling a self-contained component of a system. Capsule statecharts are the same as statecharts of UML which originated from Harel's statecharts.

In order to handle dynamic environments, it is necessary to make the some extensions for UML-RT to includes feactures that can be related to dynamic mechanisms. L.Welch [1] proposed the techniques for specification and modeling of dynamic, distributed real-time systems. The techniques are based on a programming-language-independent meta-language for describing real-time QoS in terms of end-to-end paths through application programs. Constructs are provided for the specification and modeling of deterministic, stochastic, and dynamic characteristics of environment-dependent paths. The provision for description of periodic, transient and hybrid paths is also made. Another nover feature of the language is that it allows the description of multi-level timing constraints throught (1) simple, cycle deadlines, and (2) super-period deadlines.

In the integrated too set, the most important aspects are the tools of performance analysis, used to support the analysis of temporal behavior of systems for completeness, correctness, feasibility and predictability. A significant difference between real-time systems and other software systems is the importance of correct timing behavior. Therefore, being able to guarantee meeting of timing constraints is central to any real-time system development method and tool. We use an integration approach that unify and augment the diverse approaches of the timing analysis. Different tools can be used for different stages and different aspects. We prefer to a layer-by -layer timing analysis in which we can achieve both microscopic and macroscopic predictability. In the microscopic view, we can compute the worst case execution time of any object. This is not as simple as it first may see. First, we require a simplified architecture so that instruction times are well-defined. Second, we must be able to account for resource requirements and calls to system primitives made on behalf of this. This can be accomplished via various techniques. Further, the layer-by-layer approach enables a macroscopic view of predictability. That is, first, we require the macroscopic view that all critical objects will always make their deadlines. Second, by planning and through microscopic predictability, at any point in time we know exactly which noncritical but hard real-time objects in the entire system will make their deadlines. Third, it is also possible to develop an overall quantitative, but probabilistic assessment of the performance of noncritical hard real-time objects given expected normal and overloads. We then would need to show that on the average these objects meet the system requirements or redesign or add resources until this is true.

A primary goal of using software component concepts is to promotes software reuse and sharing. A software component is an unit (object) of deployment for a collection of related software, typically with some coherent purpose. The software component is usually designed to be used to compose some larger application. With the recent adoption of the CORBA Component Model (CCM)application designers now have a standard way to implement, manage, configure, and deploy components that implement and integrate CORBA services. CCM standard not only enables greater software reuse for servers, it also provides greater flexibility for dynamic configuration of CORBA application. Thus, CCM appears to be well –suited for real-time applications based on the Computing Grid.

Meeting the QoS requirements of distributed real-time applications requires an integrated architecture that can deliver end-end QoS support at multiple levels in real-time and embedded systems. Distributed object computing (DOC) middleware based on the real-time CORBA (RT_CORBA) offers solutions to some resource management challenges and developers of real-time systems, particularly those systems are designed using dynamic scheduling techniques. Real-time CORBA ORBs preserve efficient, scalable, and predictable behavior end-to-end for higher-level services and application components. For example, a global scheduling service can be used to manage and schedule distributed resources. Such a scheduling service can interact with an ORB to provide mechanisms that support the specification and enforcement of end-end operation timing behavior.

4. Conclusion

In this paper, we have introduced a methodology and integrated toolset for analyzing, designing and implementing real-time systems based on the grid computing. an object oriented integration tool for the development of real-time systems. This integrated toolset is a collection of CASE tools, which enable a user to use objects-oriented techniques to specify , analyze, design system under development from five interrelated points of views, capturing environment, functionality, behavior, performance and implementation. One of the main feature of the our integrated tool is its capability of providing intricate of a set of tools of performance analysis, used to support the analysis of temporal behavior of systems for completeness, correctness, feasibility and predictability.

A number of possible desirable features are not currently implemented. These include some test tools. We expect an approach that combines software engineering principles, increased attention to the user interface, and prototyping in the manner described above to provide a solid foundation for building production-quality, real-time CASE tools. The existence of well-engineered CASE tools should in turn encourage practitioners to use formal methods in the development of real-time systems. We believe the visual formalisms will turn out to be a crucial ingredient in the continuous search for more natural and powerful ways to develop distributed real-time systems. It is our feeling the progress made in Virtual Reality will result in a significant change in the way we carry out our complex real-time system engineering activities.

Acknowledgments

This work is partly supported by the National Natural Science Fund, Natural science Fund of Guangdong province, “Thousand , Hundred, and ten” outstanding person fund of Education Department of Guangdong Province, Natural science fund of Education Department of Guangdong Province.

References

[1] L.R.Welch et al., Specification and modeling of dynamic, distributed real-time systems, Proceedings of the 19th IEEE Real-Time Systems Symposium, 72-81, IEEE Computer Society Press, 1998.

[2] C.Puchol and A.K.Mok, Integrated design tools for hard real-time systems, Proceedings of the 19th IEEE Real-Time Systems Symposium, 72-81, IEEE Computer Society Press, 1998.

[3] D.C.Schmidt et al., The design and performance of real-time object request brokers, Computer Communication, Vol.21, pp.294-324, Apr.1998.

[4] B.Selic, Using UML for modeling complex real-time systems. Lecture Notes in Computer Science, 1474:250-262, 1998.

[5] R.Ammar and C.Rosiene, Visualizing a Hierarchy of performance models for software systems, Software-Practice and experience, Vol. 23(3), March 1993.

[6] N.C. Audsley, et al., STRESS, A simulator for hard real-time systems, Software-Practice and experience, Vol. 24(6), June 1994.

[7] B.Berthomieu and M.Diaz, Modeling and verification of time dependent systems using time Ptri Nets, IEEE Trans. on Software Eng., Volume 17, Number 3, March 1991.

[8] Gene Forte et al., Computer-aided software engineering, IEEE Computer Society Press, 1992.

[9] D.Harel et al., Statement: A working environment for the development of complex reactive systems, IEEE Trans. on Software Eng., Vol 16, Number 4, April 1990.

[10] C.Heitmeyer and D.Mandrioli, Formal methods for real-time systems, Wiley, 1996

[11] Hood Working Group, HOOD Reference Manual, draft B Issue 3.0, June 1989.

[12] J.P. Huang, “Modeling of software partition for distributed real-time application”, IEEE Trans. on software Eng., Vol SE 11, No. 10, 1985.

[13] R.M. Kavi, “Real-time systems: Abstractions, languages, and Design Methodologies”, IEEE Computer Society Press, 1992.

[14] C.Y.Park and A.C. Shaw, Experiments with a program timing tool based on source-level timing scheme, IEEE computer, Vol.24, No.5, May 1991.

[15] J.A. Stankovic and K.Ramarithm, Hard real-time systems, IEEE computer Society, Order Number819, 1988.

[16] S.P. Reiss, ‘PECAN: program development systems that support multiple views’, IEEE Trans. on Software Eng., SE-11, (3), 1985.

[17] A.M.V. Tilborg and C.M. Koob, Foundations of real-time computing: Formal specifications and methods, Kluwer Academic publishers, 1991.

[18] P.T. Ward, S.J.Mellor, Structured development for real-time systems, Yourdon Computing series Yourdon Press-Prentice-Hall 1985.

[19] J.Xu and D.L.Parnas, On statisfying timing constraints in hard real-time systems, Proceeding of the ACM SIGSOFT’91 Conference on Software for Critical Systems, 1991.

[20] L.Zhang, et al., Methodology of real-time system design using multiprocessors, Microprocessors and Microsystems, Vol 17, No 4, May 1993.

[21 L.Zhang and B.Chaib, A design methodology for real-time to be implemented on multiprocessors, the Journal of system and software, April, 1996, 33: 37-56.

[22] L. Zhang, et al., Integrating different views to develop complex real-time systems, Proc of 21st IFAC/IFIP workshop on real-time programming, Elsevier Science , 1996.