Designing Integrated Engineering Environments: Blackboard-Based Integration of Design and Analysis Tools

Susan E. Lander[1], Scott M. Staley[2], and Daniel D. Corkill[1]

[1]Blackboard Technology Group, Inc.
401 Main Street
Amherst, MA 01002
{lander,corkill}@bbtech.com

[2]Ford Research Laboratory
P.O. Box 2053/MD 2122-SRL
Dearborn, MI 48121-2053
staley@carnap.srl.ford.com

Abstract

In the automotive industry, the knowledge, information, and tools required for design, analysis, manufacturing, and testing of company products are distributed throughout the organization. This fragmentation of expertise and information creates impediments to cost-effective design engineering. This article describes: 1) an approach to building integrated engineering environments based on blackboard technology as the integrating infrastructure; and 2) the RRM Engineering Environment, developed at Ford Research Laboratory. The RRM Engineering Environment is a large-scale prototype application comprising thirteen agents, some of which are legacy application programs, written in various languages, and distributed over heterogeneous machines and platforms. Use of integrated environments, such as the RRM environment, is expected to result in increased productivity, fewer delays and errors, and better communication among interacting participants in the design-to-manufacture process.

1 Introduction

At Ford Motor Company, the knowledge, information and tools required for design, analysis, manufacturing and testing of company products are distributed throughout, and even outside, the organization. This fragmentation of expertise and information is typical of automotive and other design and manufacturing industries and creates impediments to cost-effective design engineering. Numerous heterogeneous software tools support the

This work was supported in part by Ford Research Laboratory Project No. 5060300 and RRM Project No. 7.AA. Basic research used in this effort was supported in part by NSF SBIR Contract 94-45. The content of the information does not necessarily reflect the position or policy of Ford Motor Company or the Government and no official endorsement should be inferred.

design-to-manufacture process, some developed in-house and some purchased from commercial vendors. These tools are incompatible in various ways and their usage patterns are unstructured and informal. Engineers must spend considerable time performing manual conversion of model and data representations and manual control of software invocations. Past attempts at integrating operations and results from heterogeneous software packages have been labor-intensive, error-prone, and low-yield. Changing technologies, manufacturing philosophies, and market and legislative demands make the manufacturing environment highly dynamic. Therefore, any integrated environment must be easily adaptable to additions, revisions, and replacements of individual tools.

This article describes: 1) an approach to designing integrated engineering environments based on blackboard technology as the integrating infrastructure; and 2) an engineering environment developed at Ford Research Laboratory (FRL) using the approach. Pan and Tenenbaum [12] suggest that an integrated enterprise computing environment must support information access (the ability to locate appropriate information in a usable form), monitoring and automation (notification of events that affect local computation), cooperative work (people and computers working as a team across multiple development stages and geographical distributions), and system integration (integration of heterogeneous software packages). The blackboard-based integrated engineering environment at FRL captures these requirements and, in addition, has proven to be very adaptable to changes in the tool set.

Use of integrated environments is expected to result in increased productivity resulting from less busy-work and frustration on the part of engineers, fewer errors and delays, and better communication among interacting participants in the design-to-manufacturing process. The prototype integrated environment developed for the Rapid Response Manufacturing Consortium (RRM)[1] provides a proof-of-concept for blackboard-based integration as the enabling infrastructure for Direct Engineering at Ford.

2 Blackboard-Based Integration

Blackboard systems have been in use since the mid-1970's [5]. The metaphor often used to describe the functioning of a blackboard system is that of a set of specialists collaborating on a problem by writing on a blackboard:

> Imagine a group of human specialists seated in a room with a large blackboard. The specialists are working as a team to "brainstorm" a solution to a problem, using the blackboard as the workplace for cooperatively developing the solution.
>
> The session begins when the problem specifications are written onto the blackboard. The specialists all watch the blackboard, looking for an opportu-

[1] RRM is a joint venture managed by the National Center for Manufacturing Sciences, and funded through a NIST Advanced Technology Program Award. Ford is a member of the RRM consortium along with General Motors, Texas Instruments, Inc., Pratt & Whitney Aircraft, Martin Marietta Energy Systems, and a number of CAD/CAM/CAE software suppliers. RRM is focused on pre-competitive technology development and is not a research program.

nity to apply their expertise to the developing solution. When someone writes something on the blackboard that allows another specialist to apply her expertise, she records her contribution on the blackboard, hopefully enabling other specialists to apply their expertise. This process of adding contributions to the blackboard continues until the problem has been solved.

The blackboard model was developed specifically to mirror cooperative problem solving by experts. Although the simple metaphor above does not capture the complexity of representation and control inherent in integrated environments, it nevertheless suggests a powerful paradigm for collaboration. Ineffective implementation of some blackboard applications has led some to believe that blackboard systems are too inefficient for industrial use. However, with today's commercial availability of high-performance blackboard toolkits, the technology is both widely accessible and appropriate for industrial applications. The engineering environments discussed in this paper were developed using the commercial GBB™/NetGBB™ blackboard toolkits developed by Blackboard Technology Group, Inc. (BBTech). GBB provides all the supporting mechanisms for the development of high-performance blackboard applications. NetGBB is a recent addition to the set of blackboard tools, specifically geared toward the development of heterogeneous and distributed environments.

2.1 Agents

Specialist modules in blackboard systems have traditionally been called knowledge sources (KSs). However, the current technology supports the integration of fully autonomous software (and potentially people), more appropriately called agents. Agents differ from KSs in their degree of autonomy: they may be standalone systems, they are persistent, they may have local databases and local control. Agents can be developed independently, bought from commercial vendors, or found in libraries of existing systems. They can be implemented using widely differing approaches and programming languages and geographically distributed across heterogeneous computing platforms. Within an integrated environment, new agents can be added easily and existing agents can be transparently modified, replaced with enhanced versions, or removed. Agents that provide a generic expertise in one application can be reused in new applications.

An agent that is not specifically developed as part of an integrated environment must be *wrapped* in an encapsulation shell in order to anchor it into the environment's infrastructure. BBTech is currently developing mechanisms to support the automatic generation of encapsulation shells using high-level information acquired from a user. The type of information that must be supplied to the infrastructure include physical characteristics of an agent (hostname, pathname, language, etc.), activation requirements (parameters, initializations, entry points, etc.), and I/O requirements (translators, data formats, etc.). Agents may encompass more than one functional capability: for example, an agent might consist of a statistical package that has a single set of physical characteristics but a number of distinct functions. Each function (or operator) in turn has an associated set of I/O and invocation characteristics.

2.2 The Blackboard Database

Any integrated environment will require a shared language in which agents can communicate. The GBB/NetGBB toolkit provides an extensible object-oriented representation for shared information that is fully compatible with existing agent languages (such as KQML [6]). Shared data is normally written in the defined blackboard language and stored on the blackboard database. The database provides a shared repository of problems, partial solutions, suggestions, and contributed information. Because it is intended to be heavily used and highly dynamic, it provides highly efficient access and retrieval mechanisms. Because it is globally visible, agents are able to avoid replicating shared information locally, avoiding not only the costs associated with duplicate data storage, but also problems that stem from maintaining currency in local copies of shared data.

Because the blackboard database is partitioned, it naturally supports the definition of domain-specific ontologies. Each partition can be associated with a particular ontology and any agent that accesses information on that partition must understand the terms defined for that ontology. However, although agents are expected to use the shared language(s) for interaction, they are not required to use that language internally. If environments of heterogeneous agents are to become a reality, it is impossible to expect every agent to be developed under one language standard. Since agents can be legacy systems, it is decades too late to try to enforce language standards for internal coding. It is impractical to expect even newly developed systems to follow a single standard. Therefore, when an agent's internal representations differ from the shared representations, translation capabilities must be provided as an integral part of interaction.

Sharing information through a blackboard is an effective and efficient means of ensuring that the data is seen by all agents that can use it. It enforces common language use and understanding among agents and, because of the structure of the blackboard, supports access to information through a wide range of data characteristics. GBB provides mechanisms for specifying complex data types and multiple "dimensions" of data types, and it supports complex pattern-directed retrieval of data from the blackboard. Furthermore, the actions associated with data creation, modification, and access are events that can be used to control processing (see Section 2.3 below). By abstracting shared information away from the agents that provide and use the data, control can be extremely flexible. For example, if an agent that supplies a particular output is replaced by another agent that supplies the same output, the change is completely transparent to agents using that output: nothing needs to be done to propagate the replacement through the environment.

Despite all the advantages of the blackboard database, however, it is sometimes more reasonable for agents to share large blocks of information directly rather than through a blackboard. For example, legacy systems often expect their inputs to be available in specific, formatted input files and, similarly, write their outputs to specific, formatted output files. In these cases, a second level of sharing is possible. Rather than placing all the information from the files on the blackboard, it is sufficient to place an object there that describes the data and its format and points to the files. The creation of this object is sufficient to ensure that all agents know the data exists and to act as a triggering event for process control. The applications developed at FRL use both levels of data sharing. They also use a "load-or-compute-as-needed" philosophy for creating data on the blackboard database.

2.3 The Control Shell

Just as our eager human specialists in the blackboard metaphor need a moderator to prevent them from trampling each other in a mad dash to grab the chalk, agents need a mechanism to organize their use in the most effective and coherent fashion. Blackboard-system control has traditionally been opportunistic or reactive as in the metaphor; that is, agents react to changes on the blackboard by providing an indication that they are ready to do some activity (raising their hands for instance), then a controller selects one or more of those agents to actually invoke based on some criteria that can be either pre-established or dynamically computed. This style of control can be very effective in certain types of domain, such as domains in which there is a great deal of uncertainty about whether the information currently on the blackboard is correct. For example, in the Hearsay-II domain of speech understanding, speech signals were incrementally processed into syllables, words, and phrases, with the goal of understanding spoken sentences [5]. However, at each incremental step there were many possible solutions and high uncertainty about which was correct. If the only possible course of action was to work linearly from the first detected signal to the last signal, uncertainty about early sounds could lead to much wasted effort. Opportunistic control allowed Hearsay-II to find parts of the utterance that seemed fairly certain, without regard to time orientation, and to work outward from those pieces (called islands of certainty).

In addition to supporting opportunistic processing, the traditional blackboard control style is extremely effective at reacting to external events and exceptional events. External events occur in domains where the environment can be influenced by forces that are not under the control of the application (sensing or monitoring applications for example). Exceptional events can occur in any complex domain: something happens at an unanticipated time or in an unexpected place. An example of an exceptional event in an engineering environment might be the failure of some agent to find a solution under the current problem specification or the violation of a design assumption made by one agent in a solution provided by another agent. Because blackboard agents are triggered directly by events that occur in the environment, external and exceptional events are treated just as any other event and can be dealt with as a natural and inherent part of the application.

However, there can be several difficulties with opportunistic control. First, opportunistic control reevaluates the situation after every operator execution which, without a representation of long-term goals or plans, can lead to fragmented activity (it's hard to see the forest for the trees). This issue has been partially addressed by several researchers by adding some level of goal-directed processing or planning capabilities [1,2,3].

A second problem with opportunistic control is that many implementors find it difficult to envision and structure appropriately. In engineering environments, there are often fairly straightforward sequences of activities that can be applied to accomplish specific goals and it feels more natural to specify those sequences through a workflow-style notation. At

FRL, a methodology description language (MDL) has been developed that allows the user to specify control and data flow as a workflow graph with branching, iteration, and concurrency constructs [14]. The graph is then automatically interpreted to invoke sequences of agent executions and to direct data flow.

MDL control is more natural and easier to visualize than opportunistic control for many people and it ensures coordinated, focused activity among a set of agents. However, it is also inflexible and non-reactive. Ongoing research at BBTech is looking at ways to integrate the two control styles and to provide user support to ease the specification of effective control strategies.

3 The RRM Engineering Environment Prototype at Ford Research Laboatory

This work, supported by the RRM Project, is a prototype of a general integrated engineering design environment [4]. The currently implemented environment contains thirteen agents related to crankshaft design, analysis, and manufacturing feasibility assessment. These agents include legacy Ford analysis code, ICAD KBE applications, an Itasca OODBMS, an ACIS solid modeler, STEP data conversion and repository software, large-scale analysis models (NASTRAN and GESIM), and custom in-house CAE models. The agents are distributed over heterogeneous machines and platforms and are written in Lisp, C, C++, and FORTRAN.

The integrated application is a multiuser, multiprocessing environment. When a user begins work on a design problem, she sees a top-level screen that shows the set of tools available in the environment. Each of these tools may actually comprise one or more distinct agents. Although physical properties of the tools are transparent to the user, they may, in fact, be running on disparate workstations that are geographically distributed across the network and they may be simultaneously in use by other engineers working on other designs.

3.1 Integrated Agents in the RRM Environment

The RRM environment includes two ICAD KBE agents: K-BAM, which produces a manufacturing feasibility assessment given a crankshaft parameter file; and Dunnage Design System (DDS), which produces the design of material handling equipment for shipping and storage of finished (machined) crankshafts from the crankshaft parameter file. The integration of ICAD agents into the RRM engineering environment was accomplished by developing NetGBB code to create and access ICAD parts concurrently using lightweight multiprocessing capabilities. This approach allows KBE rule bases to be shared concurrently among users and other agents in the environment. Data is supplied to the ICAD agent with a user interface. Then a high-level command is entered to connect to the ICAD image running on a remote machine and run the ICAD rule-base on a new set of design information. When computation is complete, another command sends queries to ICAD to return values, for example, of newly designed dunnage.

A previously developed application that has been integrated into the RRM environment is the Advanced Manufacturing Development System (AMDS), a STEP data repository built on the Itasca object-oriented distributed database management system. An AMDS database was developed to hold crankshaft and other engine component design data. Changes to design data in Itasca would trigger messages to other concurrently executing agents and notify them of new parameter values for a design. A user interface was developed to allow engineers to easily make changes to design data as well.

Much of the data required for AMDS is represented by other agents in ACIS rather than STEP format. The use of multiple representations of data within a single environment is very common and is handled by including reusable agents in the environment specifically to translate from format to format. In this case, conversion from ACIS to STEP is done by the STI translator.

A fifth agent in the environment is a Material Properties Database, implemented as a stand-alone distributed blackboard system. Other agents such as a database browser can connect and disconnect to the database as material property data is required, and multiple agents can connect simultaneously. Materials in this database are organized according to typical use categories such as package, substrate, and interconnect. The database uses a "load-on-demand" strategy that allows it to load and initialize quickly with a set of identifiers. The actual data associated with each identifier is loaded only when it is requested. The data may be actual property values or it may be in the form of equations for computing values. Along with each entry in the database is a source which describes where the data was found, for example, a source can be a material reference book or a journal article.

The creation of a new crankshaft design requires analysis to ensure that the part meets specified design performance criteria. The crankshaft is a central part of an engine and significantly affects its noise, vibration, and harshness characteristics. Numerous analyses must be performed and the analyses must be sequenced correctly. The prototype RRM environment includes five CAE analysis agents although in a full-scale implementation, others would be integrated as well. Analysis agents include NASTRAN, General Engine SIMulation (GESIM), Torsional Vibration Analysis (TVA), Front End Accessory Drive (FEAD), and UNBALANCE, which determines the moments and forces created by the rotating crankshaft system. The benefit of using the blackboard approach for building an environment is readily apparent when integrating design and analysis tools which have a large mutual solution space, a variety of input data, and diverse local inferencing methods. Once the underlying structure of the environment is determined, the addition of new agents or replacement of modified agents is relatively straightforward. Due to the modularity of the agent-based approach and the abstraction of physical characteristics of agents from their logical connections, often the only effort needed is to recompile the environment.

3.2 Sharing Data

There are two primary mechanisms for sharing data among agents in the RRM environment: blackbox data sharing and shared-memory data sharing. In blackbox data sharing, the information consumer is assumed to be a completely stand-alone application with some specific data input requirements. For example, the NASTRAN agent requires a formatted "input deck" to be prepared prior to its execution. In this situation, the data required to build the input deck is maintained on the blackboard. A special input-preparation agent monitors the blackboard and, when all necessary information becomes available, writes it out to the input deck and runs the NASTRAN agent on that deck.

The second form of information sharing allows agents to share a single set of data rather than specially formatted copies of the data. This shared-memory model was the predominant method in the RRM environment. Shared-memory integration is accomplished using the Lisp foreign function interface, a powerful tool included in most commercially available Lisp implementations [7,11]. This interface allows the definition of foreign data structures within the Lisp environment which can then be shared between Lisp and the agent's language of choice. Data is stored directly on the blackboard and viewed directly by the agent. There are two primary advantages to this method. First, information need not be explicitly passed to and from agents as I/O since the agent can access it directly. Second, the different agents see current parameter values without update delays. The disadvantage to this form of data sharing is that the agent must represent its shared data as common blocks or other structures that can be shared by the agent and the Lisp environment.

3.3 Status of the RRM Engineering Environment

The RRM prototype environment was implemented in approximately 3 months using GBB and NetGBB as the integration framework. The implementation has been highly dynamic, with agents being added incrementally as needed and as prototyping time permitted. This prototype environment has been presented to the RRM Steering Committee and to Ford as a potential enabling infrastructure for Direct Engineering.

The RRM environment described above is fully implemented and has been highly successful. It meets all of the requirements stated by Pan and Tenenbaum, as noted above, and has exceeded expectations in terms of speed of implementation, flexibility, maintainability, and cost-effectiveness. Legacy and custom tools can be added and modified within the integrated environment without incurring large costs in the overall project. The tools are at the engineers' fingertips, much of the interaction among tools has been automated, and the engineers are able to work concurrently and track each other's progress.

Are we done then? The GBB/NetGBB infrastructure works very effectively to support enterprise integration. The next step is to look beyond infrastructure issues at the problems that arise due to the nature of large, distributed, and complex systems. In the next section, we describe ongoing projects that are designed to ease the construction and use of integrated environments. Being able to create an environment does not guarantee that the environment will be particularly effective; therefore, in the last section, we discuss some long-term directions for optimizing the effectiveness of integrated environments.

4 Current Projects

The ability to create an instance of an integrated environment does not supplant the need to articulate and support a formalized approach to designing new environments. Standardizing the design process is difficult because each application environment is unique and the interacting components are themselves heterogeneous, complex, and, particularly in the case of legacy systems, not well understood. To build design-support software systems, formal models of the design process are required. These are largely lacking (or informal) in industrial practice. Even given a formal design process description, the difficulty remains of producing the integrating software for a particular design domain.

Therefore, one very important direction is to enhance the software-engineering process involved in building integrated design environments. Several projects are currently underway to standardize and support this process. The first project is the development of a graphical user interface to assist a system integrator in specifying a graphical representation of control and data flow in an environment. This representation is then provided to an interpreter for automatic execution of the specifications [14,15]. A second project is the development of agent wrappers for tools that will automatically anchor the tools into the infrastructure of the integrated blackboard environment. Along with agent wrappers for existing tools, we are developing a set of generic reusable agents that will perform routine translation and computation over multiple applications [9]. The primary goal of these projects is to make it easier to build integrated environments.

5 Future Directions

In these projects, rapid progress has been made in meeting the infrastructure requirements for creating integrated engineering environments. However, the future points back to the past: Distributed AI has always been concerned with the problem of how to make the most effective use of fragmented expertise and information. Who should send what information to whom and when? Who should I listen to? What order should be placed on distributed task sequencing and how will the order be maintained? What social rules and conventions should be followed? What is the most appropriate organization of agents for solving a particular problem? These questions are not answered by the infrastructure of an application (although infrastructure considerations may influence the answers).

The creation of effective and efficient environments must therefore look to DAI research and development to provide insight and direction. One preliminary investigation looked at the feasibility and effect of sharing information other than results in heterogeneous environments; for example, not responding to a request with a single solution, but rather sending a solution along with constraining information that defines an abstract search space for other possible solutions. This investigation showed that significant improvement in the costs of designs and in the amount of time required to produce designs could be achieved through information sharing [10]. A second investigation looked at agent organizations and whether machine learning techniques can be applied to learning appropriate organizations for an agent set [8,13]. Preliminary results from that investigation showed that agent organization has a significant effect on the quality of designs and that learning can be used both to migrate toward a good organization dynamically and to characterize situations in which a particular organization is appropriate.

References

[1] Norman Carver and Victor Lesser. The evolution of blackboard control architectures. *Expert Systems with Applications–Special Issue on the Blackboard Paradigm and Its Applications*, 7(1):1–30, January/March 1994.

[2] Daniel Corkill, Victor Lesser, and Eva Hudlicka. Unifying data-directed and goal-directed control: An example and experiments. In *Proceedings of the National Conference on Artificial Intelligence*, pages 143–147, Pittsburgh, Pennsylvania, August 1982.

[3] Edmund Durfee and Victor Lesser. Incremental planning to control a blackboard-based problem solver. In *Proceedings of the National Conference on Artificial Intelligence*, pages 58–64, Philadelphia, Pennsylvania, August 1986.

[4] S.M. Staley (Ed.). Integration technology development. RRM Final Report Project No. 7AA, National Center for Manufacturing Sciences, Ann Arbor, MI, February 1995.

[5] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[6] Tim Finin, Rich Fritzson, Don McKay, and Robin McEntire. KQML: A language and protocol for knowledge and information exchange. Technical report, University of Maryland, Baltimore, MD, 1994.

[7] Franz Inc., Berkeley, CA. *Allegro CL Common Lisp User Guide*, January 1994.

[8] Susan E. Lander. *Distributed Search and Conflict Management Among Reusable Heterogeneous Agents*. PhD thesis, University of Massachusetts, Amherst, Massachusetts, May 1994.

[9] Susan E. Lander. Blackboard-based integration of legacy and reusable software agents. NSF SBIR Final Report Contract No. 94-45, National Science Foundation, July 1995.

[10] Susan E. Lander and Victor R. Lesser. Sharing meta-information to guide cooperative search among heterogeneous reusable agents. Technical Report 94-48, Computer Science Department, University of Massachusetts, Amherst, Massachusetts, June 1994.

[11] Lucid Inc. *Lucid Common Lisp/SUN Advanced User's Guide*, 1991.

[12] Jeff Y.C. Pan and Jay M. Tenenbaum. An intelligent agent framework for enterprise integration. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November/December 1991.

[13] M.V. Nagendra Prasad, V.R. Lesser, and S.E. Lander. Learning experiments in a heterogeneous multi-agent system. In *Proceedings of the Workshop on Adaptation and Learning in Multiagent Systems, IJCAI-95*, Montreal, Canada, August 1995. Also available as CS Technical Report 95-35, University of Massachusetts, Amherst.

[14] Nanxin Wang and Jie Chang. MDL: A methodology description language in a CAE integration environment. In *Proceedings of the ASME Design Automation Conference*, September 1994.

[15] Nanxin Wang and Jie Cheng. EMAT: An engineering methodology application tool. Technical report, Ford Research Laboratory, Ford Motor Company, Dearborn, MI, 1995.