2

RADC-TR-90-414
In-House Report
December 1990

KNOWLEDGE-BASED SOFTWARE ASSISTANT (KBSA) TECHNOLOGY TRANSFER CONSORTIUM: STATUS REPORT #1

Donald M. Elefante

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

RADC-TR-90-414 has been reviewed and is approved for publication.

APPROVED: [illegible]

SAMUEL A. DINITTO, JR.
Chief, C2 Software Technology Division
Directorate of Command & Control

APPROVED: Raymond P. Urtz Jr

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: Raposo

RONALD S. RAPOSO
Directorate of Plans & Programs

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1990	In-House

4. TITLE AND SUBTITLE
KNOWLEDGE-BASED SOFTWARE ASSISTANT (KBSA) TECHNOLOGY TRANSFER CONSORTIUM: STATUS REPORT #1

5. FUNDING NUMBERS
PE - 63728F
PR - 2532
TA - PR
WU - OJ

6. AUTHOR(S)
Donald M. Elefante

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rome Air Development Center (COES)
Griffiss AFB NY 13441-5700

8. PERFORMING ORGANIZATION REPORT NUMBER
RADC-TR-90-414

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Air Development Center (COES)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
RADC Project Engineer: Donald M. Elefante/COES/(315) 330-3565

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

RADC's Knowledge Based Software Assistant (KBSA) program sprung from a 1983 Rome Air Development Center (RADC) technical report entitled "Report on a Knowledge Based Software Assistant". That document integrated key ideas on how artificial intelligence (AI) might be used to design, develop and maintain software over the complete life cycle. The report's general intent was to suggest a program that could bring an order-of-magnitude productivity complex systems involving mission-critical computer resources. Since then, RADC has initiated and completed the first of three multiple-contract phases intended to develop both a KBSA and its supporting technologies. The second phase is well under way.

The success of the KBSA program has meant that RADC has had to devise effective measures for transferring its early technology to potential defense contractors. Thus, the Knowledge Based Software Assistant Technology Transfer Consortium (KBSA TTC) was inaugurated. Because the consortium was founded on quid-pro-quo exchanges not directly involving money, this technology transfer approach has been as much of an experiment in method as an exercise in necessity. This report takes stock of the consortium's experiences and achievements over the first 1-1/2 years.

14. SUBJECT TERMS
Technology transfer, knowledge-based software engineering, artificial intelligence

15. NUMBER OF PAGES
52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

PAGE	CONTENTS
1	Introduction
1	Birth of the KBSA Technology Transfer Consortium
2	Operational Concept Agreement
2	Consortium Goals and Rationale for its Existence
3	Responsibilities of Consortium Members
3	All Members
4	KBSA Developers
4	Early Prototype Evaluation (EPE) Sites
6	Rome Air Development Center
6	Qualifications of Early Prototype Evaluation (EPE) Sites
7	Consolidated Reports
7	Andersen Consulting
12	Boeing Computer Services
14	McDonnell Douglas
15	Rockwell International
18	Honeywell
19	Information Sciences Institute
20	Lockheed Sanders
22	Kestrel Development Corporation
22	Software Options
23	Missing Reports
23	Discontinued Participation
23	Rome Air Development Center Assessment

Appendix

Overview of RADC's Knowledge Based Software Assistant Program

INTRODUCTION

RADC's Knowledge Based Software Assistant (KBSA) program sprung from a 1983 Rome Air Development Center (RADC) technical report entitled "Report on a Knowledge Based Software Assistant" (NTIS AD# A134-699). That document integrated key ideas on how artificial intelligence (AI) might be used to design, develop and maintain software over the complete life cycle. The report's general intent was to suggest a program that could bring an order-of-magnitude productivity improvement to the business of software development and maintenance--particularly for large, complex systems involving mission-critical computer resources. Since then, RADC has initiated and completed the first of three multiple-contract phases intended to develop both a KBSA and its supporting technologies. The second phase is well under way. (Appendix A is a paper describing RADC's KBSA program, circa June 1989).

The success of the KBSA program has meant that RADC has had to devise effective measures for transferring its technology to potential defense contractors. This is especially significant in the current world climate where the break-neck pace of political and economic developments is placing great demands on how well and how quickly we harvest pay-back for our technology developments. And so it is within this climate that Rome Air Development Center inaugurated the Knowledge Based Software Assistant Technology Transfer Consortium (KBSA TTC)--as much of an experiment in method as an exercise in necessity. This report takes stock of the consortium's achievements over the first 1-1/2 years.

BIRTH OF THE KBSA TECHNOLOGY TRANSFER CONSORTIUM

The KBSA TTC was born late in 1988 after RADC advertised in the Commerce Business Daily for businesses and universities who might be interested in playing an active role in the early development and transition of KBSA technology. To keep the consortium at a reasonable size, the advertisement stated that from three to seven large businesses (greater than 500 employees), from two to five small businesses, and from two to five universities would be competitively chosen to participate. Prospects were asked to submit qualification packages that addressed the following six evaluation criteria:

a. Personnel -- people capable of working, and committed to working, with KBSA technology.

b. Equipment -- both appropriate hardware and software to run one or more KBSA applications.

c. Expertise -- familiarity with the technology in general; for example, software engineering, AI, software language theory.

d. Training -- commitment to acquire the training needed to learn about a particular KBSA application.

e. Fiscal commitment -- past or intended future investments in the technology in general.

f. Support for Technology Transfer -- the likelihood of eventually integrating KBSA products into daily operations or products; that is, the technology would eventually be used somehow and not just archived.

When the qualification packages were received and evaluated, eleven organizations were found to meet the criteria. These organizations--initially (and inaccurately) called Alpha Test Sites[1]--were then invited to participate. The selected large businesses were (1) Rockwell International Electronics Operations, Anaheim, CA, (2) Lockheed Missiles & Space Co., R&D Division, Palo Alto, CA, (3) Arthur Andersen & Co., Chicago, IL, (4) Texas Instruments Central Research Laboratories, Dallas TX, (5) Martin Marietta Information & Communications Systems, Denver, CO, (6) McDonnell Douglas Astronautics Co., Huntington Beach, CA, and (7) Boeing Computer Services Government Information Services, Seattle, WA. Four other large businesses were not selected.

Only two out of ten small businesses were selected, since small business had a much more difficult time meeting all six of the qualification criteria. Those selected were Reasoning Systems of Palo Alto, CA, and Software Productivity Solutions of Melbourne, FL. Two universities also responded and each was invited to participate: Texas A&M Knowledge Based Systems Laboratory of College Station, TX, and Rochester Institute of Technology, Rochester NY.

The first consortium meeting inviting the above selectees took place at the 3rd annual KBSA Conference held in August, 1988. At that time the initial draft of the consortium's "Operational Concept Agreement" (OCA) was discussed and more concrete agreements reached. The OCA's integrity and currency are given a great deal of attention by RADC in order to accurately reflect the unique technology transfer arrangement and experiment that has been undertaken. The following section presents details of the current OCA, which succeeds several earlier versions.

OPERATIONAL CONCEPT AGREEMENT

The OCA outlines the goals of the KBSA Technology Transfer Consortium, explains the rationale for its existence, and defines the voluntary responsibilities of each of three classes of participants: KBSA Developers, Early Prototype Evaluation (EPE) Sites, and RADC. The OCA is a living document whose contents evolve as time and experience bring a greater focus to the consortium's needs and function; it represents a recent snapshot of the way consortium members have agreed to work together.

Although formal contractual arrangements have not been ruled out, EPE-site participation is not contractually binding right now. Yet, each site is expected to pull its own weight by providing feedback data to developers and technology-transfer data to RADC. In return for their efforts, EPE sites also benefit in ways explained below.

Consortium Goals and Rationale for its Existence

Allowing potential users to evaluate unfolding KBSA technology does not guarantee the effective transfer of that technology. The greatest benefit can come only when a continual, two-way information flow between KBSA developers and likely users is in effect. Developers need objective evaluations of evolving KBSA prototypes and supporting technologies. They also want to be aware of potential user demands and needed functionality. Potential users, on the other hand, want to be educated in the new technology as it unfolds and eventually becomes available. They benefit from being in on the ground floor of KBSA's exploratory development

[1]With the publication of this report, and in the interests of fostering better communication, the term "Alpha Test Site" will be abandoned in favor of "Early Prototype Evaluation Site" (or EPE Site).

and helping to shape the program and technology decisions. Among these benefits are the early and gradual introduction of KBSA concepts into their organizations, the opportunity to voice their projected user requirements, and spinoff technologies such as reverse engineering that can enhance their operations in the near term.

In the past, the desired two-way information flow was achieved by KBSA Developers and potential users simply agreeing to cooperate. Such cooperation was encouraged by RADC and was also modestly successful for technology transfer. Yet, it had three major limitations.

The first limitation was a total lack of formality. The incentives needed for two parties to casually work together must be substantial for the relationship to work well. Further, even when such relationships do work, benefits are typically confined to only the participants. Second, the technology transfer process is seldom documented well (if at all) under casual arrangements. However, such documentation is important to RADC, for it is essential in pointing the direction for future tech transfer efforts, as well as for evaluating the overall success of the KBSA program. Finally, RADC needs to participate in user/developer communications to learn from the ongoing dialogue. The consortium was set up to mitigate these limitations and thereby enhance the KBSA technology transfer process.

One of the consortium's goals is to provide the early release to EPE Sites of KBSA demonstration systems as well as consulting hours (when requested by the site). Compensation for consulting hours is paid by the EPE Sites to developers at openly available rates. If a developer and an EPE Site wish to set up another form of compensation or some hybrid arrangement, that is their prerogative. Such arrangements are not precluded by consortium membership.

The unrestricted right to access and use KBSA-developed software is a crucial consortium concept. Because KBSA software is embodied largely in early prototypes, it cannot be resold to the government without significant development first taking place. At this point in KBSA's development, it is the concepts embodied in the early prototypes that are of greatest value to EPE Sites--moreso than the software itself; and it is these concepts that are the current objects of technology transfer. However, as the evolving prototypes become more complete and robust, technology transition into the advanced development arena will result.

Though KBSA prototypes are not themselves restricted, the software or supporting environments upon which they run may be proprietary or restricted. Rights to these support environments must be secured from the appropriate parties by EPE Sites.

Another consortium goal is to get feedback from EPE Sites and developers in order to track technology transfer. As stated earlier, this feedback is of much value to both RADC and the developers. In addition to dialogue, this feedback is acquired through annual reports--prepared by the EPE Sites--which RADC consolidates and distributes to consortium members and other interested parties. RADC also distributes pertinent technical reports developed under KBSA contracts.

Responsibilities of Consortium Members

All members will:

Attend the annual KBSA conference where, starting in 1991, annual consortium meetings will be scheduled.

KBSA Developers will:

a. Develop KBSA facets and supporting technologies.

b. Provide early and ongoing opportunities for EPE Sites to comment on developer work; also, take such comments into consideration during development.

(1) For all KBSA contracts let after September 30, 1990, provide for an arrangement wherein one or more EPE sites will be invited to evaluate and comment on the developer's work over the course of the contract. Specific sites needn't be called out at the time the arrangement is proposed to RADC. In the spirit of the consortium, EPE sites should be expected to chip in with their participation expenses.

c. Supply each EPE Site with appropriate KBSA product documentation and prototypes on a timely basis, i.e. source code and (at least) the documentation delivered to the government.

d. Provide software and KBSA-concept training for up to two representatives from each EPE Site. The EPE Sites will bear the cost of training, and the developers will directly bill the EPE Sites for such training. Developers will announce training costs and dates well in advance to permit EPE Sites adequate time for planning. Sixty days is preferred, and 45 days should be considered the minimum.

e. Provide consulting hours to each requesting EPE Site. A minimum number of consulting hours will be provided to each site at openly advertised rates and quantities. Arrangements for more than the minimum number of consultation hours can also be made by mutual agreement. Consulting is intended for conceptual assistance rather than extensive software support.

f. Apprise RADC of all significant or unusual EPE-Site requests, of agreements or interactions, or of important issues that arise. This includes any activity or interaction that RADC should know about relative to its consortium coordination and facilitation role, or its need to keep track of the technology transfer process.

g. Provide a yearly report, covering the period 1 July through 30 June, that assesses the consortium's activities and effectiveness, and provides suggestions for updates to the OCA. The report is to be delivered to RADC by 15 August so that it can be included in the annual consortium status report prepared by RADC and distributed at the KBSA conference.

h. Participate in the Consortium's on-line forum.

KBSA Early Prototype Evaluation Sites will:

a. Have access to KBSA Developer interim products, early prototypes, and relevant documentation.

b. Evaluate KBSA technology with an eye toward helping developers find the best directions for subsequent development (thus helping insure that developed products have the right functionality).

(1) As a condition of accepting a developer's invitation to evaluate and comment on the developer's work over the course of the contract, contribute to participation expenses in the spirit of the consortium's goals.

c. Infuse KBSA concepts and technology, where worthwhile, into current or future technological products, planning, or other related activities.

d. For prototypes or products of interest to the EPE site, send one or two key individuals to product training sessions provided by the developers. Costs of training will be borne by the EPE sites. For any training sessions not attended although available, prepare a statement for RADC (for the purpose of tech-transfer tracking) describing the factor(s) that mitigated against attendance, e.g. expense, perceived value or utility tradeoffs, scheduling conflicts.

e. Be entitled to a minimum number of developer consulting hours at openly advertised rates and quantities. These consulting hours are intended for learning about KBSA products and support environments. Availability of additional consulting hours beyond the minimum will be a function of developer/EPE-Site concurrence.

f. Secure the rights to, and bear the costs of, proprietary software needed to use and evaluate KBSA prototypes at their own site. Otherwise, prepare a report for RADC describing why the expense or effort to procure outweighs the value of the prototype(s) to their organization.

g. Document and demonstrate technology transfer via status reports, papers and demonstrations. Generate a yearly status report according to the following schedule and guidelines:

(1) The status report will cover the period from 1 July through 30 June and may contain proprietary information to assist RADC's understanding of the technology transfer process. This status report will be delivered to RADC by 15 August.

(2) The information contained in this status report (and the reports submitted by the other EPE Sites) will be employed by RADC to create a consolidated consortium report intended for open distribution at the annual fall KBSA conference. Therefore, any proprietary information included in the status report should be appended in a separate section to facilitate easy identification, removal, and protection by RADC.

(3) Status reports will contain:

A. The evaluation of strengths and weaknesses of KBSA technology as it pertains to the site's projected needs; also development recommendations.

B. Technical-level descriptions of how the site ultimately plans to use KBSA technology.

C. An evaluation of developer/RADC/EPE-Site communications and interactions in respect to their suitability for supporting early technology transfer.

D. A general, catch-all discussion, to include such things as (1) the effectiveness of the consortium and OCA to date, and what improvements can be made, (2) the effectiveness of the transfer of KBSA-technology concepts to the site and what could be done to improve it, and (3) the effectiveness of training, including comments and recommendations. Training evaluations should include comments about the product tools, the training received, and initial impressions of the technology.

h. Participate in the Consortium's on-line forum.

i. Designate a primary point of contact for all consortium-related activities.

RADC will:

a. Serve as facilitator and coordinator for all significant KBSA Technology Transfer Consortium activities.

b. Prepare a periodic consortium newsletter for consortium participants.

c. Make final decisions concerning the consortium's composition.

d. Serve as approval authority or mediator for all activities deemed by consortium participants to require such approval authority or mediation. Except for the consortium's ongoing composition, RADC will not otherwise make unilateral policy decisions for the consortium.

e. Maintain the KBSA Technology Transfer Consortium's Operational Concept Agreement.

f. Annually prepare a consortium status report containing a consolidation of yearly EPE-Site and developer status reports. The consortium status report is intended for distribution to all consortium members and other interested parties. Any clearly marked and appended proprietary information included in EPE-Site status reports will be protected.

g. Maintain and moderate an on-line "net news" forum that permits consortium members to publicly interact with each other (non-real-time), and provides a place for newsworthy items to be posted.

h. In general, serve as an enabler: work to facilitate activities and decisions that promise to further the consortium's goals.

Qualifications of Early Prototype Evaluation (EPE) Sites

EPE Sites are evaluated for their qualifications before being invited by RADC to participate in the consortium. Each is asked to submit a qualifications package that addresses the following eight areas:

a. Personnel -- Are people capable of working, and committed to working, with KBSA technology?

b. Equipment -- Does the site have both the hardware and software needed to run and evaluate the KBSA applications of interest to them?. (Sun 4's and Sun SparcStations with 16-32 MB of memory and 500+ MB disks are becoming the standard. REFINE and/or CLF software licenses may also be needed.)

c. Expertise -- Is the site familiar with the technology in general? For example, software engineering, AI, software language theory.

d. Training and Travel Commitment -- Is the site willing to get the training needed to learn about one or more KBSA applications and also attend the annual KBSA Conference?

e. Fiscal Commitment -- What investments, past or planned, have been made for purchasing or expanding technology related to KBSA.

f. Support for Technology Transfer -- How much does the site need to be in on the ground floor of KBSA development, and what is the likelihood of using mature KBSA products or their spinoffs?

g. Internet Access -- Does the site have ready access to the Internet for reading the consortium news and participating in the discussion forum on a regular basis?

h. CCAL Status -- Is the site registered on the Defense Logistics Services Center (DLSC) Certified Contractor Access List (CCAL)? An organization can apply to become registered with the DLSC by submitting a certification application (DD Form 2345, April 1986; Militarily Critical Technical Data Agreement) to the DLSC at the following address:

Attn: DLSC-FEB
United States/Canada Joint Certification Office
Defense Logistics Services Center
Federal Center
Battle Creek, MI 49017-3084
(616) 961-4358 or (800) 352-3572

CONSOLIDATED REPORTS

What follows are consolidations of (or major extractions from) three types of reports submitted by consortium members: EPE-Site status reports, developer assessments of the consortium, and a hybrid report prepared by Andersen Consulting. All reports were as of March 1990.

The EPE-Site status reports followed the reporting format called for in the Operational Concept Agreement in effect at that time, a format that designated four reporting areas. In addition to those four areas, RADC requested a fifth on a one-time basis. It was a questionnaire designed to help developers get an early look at issues that might arise as KBSA moves toward greater standardization. However, only two questionnaires were submitted, leaving little trend information to work with.

Developer reports were prepared as a result of RADC asking them to submit their to-date assessments of the consortium's activities and effectiveness. Although this reporting was not a developer role cited in the OCA that preceded the current one, most developers agreed to do so anyway--to everyone's benefit.

The Andersen Consulting report is a hybrid because, at the advent of the Consortium, Andersen served only as an EPE Site. Later on, they won the KBSA Concept Demonstration Contract and acquired a KBSA Developer hat. Thus wearing both hats, Andersen prepared a report exhibiting both characteristics.

As a final point of introduction, RADC requested that EPE Sites and developers review and comment on the OCA as it read at the close of 1989. Andersen Consulting was the only consortium member to submit substantive comments.

Andersen Consulting Report (Developer and EPE Site)

Andersen generally agreed with the 8 December 89 version of the OCA, but also offered a few minor suggestions or amendments for each of the three responsibility areas:

Developer Responsibilities: Suggest formalizing early and ongoing opportunities for EPE Sites to comment on developer work. Also, take such comments into consideration during development. For example, each developer might be required to form and administer a "user group" of EPE Site personnel who have had experience with the project software. Periodic project reviews would be presented to that group.

EPE Site Responsibilities: Suggest that item f be amended to read "... at their own site, or else prepare a report describing how the expense and/or effort required to do so clearly outweighs the value of the actual assistant for their organization." The way item f currently reads, too much is expected of those sites who may have no interest in certain of the facets.

RADC Responsibilities: Suggest RADC adopt the role of making unilateral policy decisions concerning the Consortium's ongoing composition.

All of Andersen's suggestions were accommodated in some way in the latest OCA.

Next are comments made by Andersen about specific KBSA products.

Knowledge Based Specification Assistant (KBSpecA)/ARIES: "As a result of our work on the Concept Demonstration Systems project, we have spent a great deal of time studying and using the Specification Assistant developed by ISI, as well as its successor technology currently being developed under the ARIES project. This work provides the user with tools to develop and reuse specifications which are represented using the Gist specification language. Gist allows for the description of a wide range of specification information, ranging from abstract descriptions of the domain to executable descriptions of program behavior. Elaboration of the Gist specification is supported through user-directed transformations known as *high level editing commands* (HLEC). The HLEC's represent a step towards the goal of machine-assisted formal implementation, and form the basis for developing higher level plans or strategies for specification development. The Specification Assistant also provides valuable analysis tools that support specification evaluation. The static, resource, and ontological analysis tools provide straight-forward but critical consistency checking mechanisms.

"The Specification Assistant demonstrates that paraphrasing is a useful and realizable technique for specification understanding. We found that the paraphraser worked well, generating natural text in a reasonable amount of time. We found the paraphraser useful in clarifying the meaning of Gist specifications. Paraphrasing helps to alleviate the cognitive complexity of formal notations. This is important not only to support the specification developer, but also to make the specification understandable to others (e.g., end-users, managers, and maintainers) involved in the system development process.

"The symbolic evaluator uses a special purpose theorem prover to generate behaviors from a high level Gist specification. This supports the analysis of Gist specifications which are too abstract to be compiled. However, in our experiments we could not successfully apply the symbolic evaluator to even small specifications. It either failed to terminate after running for several minutes or aborted.

"We believe that the problems with the symbolic evaluator have more to do with the technique in general than with the current implementation in the Specification Assistant. As with many applications which utilize theorem proving, we feel that this approach has not yet been proven viable for realistic problems in specification analysis."

Knowledge Based Requirements Assistant (KBRA): "In our evaluation, the two most important concepts in KBRA are the presentation-based architecture and the 'catch as catch can' approach to capturing and formalizing requirements. The presentation-based architecture provides the user with a powerful method for the evolution of requirements.

"The 'catch as catch can' approach to capturing requirements in the KBRA has both positive and negative points. This approach conforms with our belief that the requirements gathering process cannot be completely well-structured or predefined. No tool can have the complete domain or process knowledge necessary to turn requirements gather into a 'cookbook' process. The tool must therefore provide a 'safety net' which allows the analyst to input information that the tool does not understand.

"Its weakness is that it provides too little structure. The KBRA lacks the facilities to provide feedback to the user on the consistency and completeness of the requirements. It also provides little assistance to help guide the analyst through the requirements analysis process. This lack of support results from the absence of either a formal requirements gathering process model or a formal requirements language.

"The input mechanisms developed in KBRA included a valuable experiment in the use of natural language processing (NLP) techniques for requirements gathering. The Intelligent Notepad utilizes a case-frame NLP scheme to interpret natural language requirements. We concluded from our own experiences and from the results of the KBRA that the natural language approach will not scale up to industrial problems in the near future. An NLP requirements tool that has to work robustly on industrial size problems would require an unrealistically complete knowledge base of both application and requirement analysis domain concepts."

KBSA Framework: "The KBSA Framework addressed the integration of individual facets into a single framework. It developed a distributed object protocol using metaclass programming in CLOS. The Framework project also discussed various approaches to integrating different tools and showed that for KBSA, the only feasible approach was 'deep integration,' i.e. reimplementing in order to share the same underlying parts of KBRA. The experience of the Framework developers in these reimplementation projects will be extremely valuable in future integration efforts, including the Concept Demonstration.

"Perhaps one of the most important accomplishments of the Framework was to point the way toward higher-level compatibility standards. It will be essential for future KBSA facets to be written using similar knowledge representation and interface tools. It is not enough for the facets to all use common lisp. As long as they use different supporting tools (AP5, Loglisp, Refine, Socle, etc.) the overhead in communication and in size of the lisp image will make it impossible to have several facets cooperating in the same lisp environment. The framework project took steps towards a higher level of standardization through its use of CLOS and CLUE (Common Lisp User Environment)."

Project Management Assistant (PMA): Concerning the Project Management Assistant (as of September 89), "we were impressed rather favorably with its high-quality interface and reasonably good response time. We remain uncertain as to the degree of functionality that it provides beyond current generation project management support packages."

Since the time of Andersen's status report, a later version of PMA, which evolved in both functionality and user interface, was shipped to Andersen for evaluation and comment. The final version of PMA was delivered to RADC in August, 1990--the completion of Kestrel Development Corporation's PMA contract with RADC. Andersen will be incorporating the final version for the preliminary design review to be held at the the fifth annual KBSA Conference.

Performance Estimation Assistant (PEA): Andersen's exposure to PEA occurred in a class environment at Kestrel Development Corporation created for Consortium members. The meeting demonstrated the functionality and progress on the PEA software, and provided hands-on experience for participants under the guidance of the developers (Kestrel).

Concerning that class: "The class was focused on the transformational development of efficient program implementations from formal specifications using the PEA software. An explicit process model was presented to describe the relationships among the processes of algorithm design, optimization and prototyping. Algorithm design encompasses the process of determining an appropriate sequence of operators sufficient to transform the state of a system from a specified initial state to a specified desired state. The process of algorithm optimization involves the selection of data structures appropriate to the algorithm and the application of specific program derivation tactics that transform specifications into efficient implementations. Execution or simulation of the algorithm is always possible during the optimization process."

Activity Coordinator (AC): Andersen saw a very informal demonstration of a voting system as of December 1989. In summary:

"The Activity Coordinator's objective is to develop a model and software infrastructure capable of supporting cooperative work on multiple, heterogeneous hardware platforms. This involves the development of a protocol that will support interaction over long periods of time. The AC prototype is scheduled to run in early 1991; its target environment is UNIX. Software Options wants to think of the AC as a program with which the user interacts and which will run forever. While no piece of hardware will run forever, the AC has to be able to survive both hardware and system software crashes as well as a cut-over to new versions of its own applications. (An application might be the expected series of interactions planned to occur over a given project, i.e. a project plan. But if the plan is changed halfway through the project, the AC has to be able to relate the two plans to each other.

"The only running software related to the Activity Coordinator project is a prototype voting system, in which a registrar determines that a vote should be held, nominates a set of voters, describes the issue, and then starts the voting process. The voters receive notice that a vote is going on, and can vote or wait until the vote's 'timeout' function fires, terminating the polls. The registrar is notified when the vote is complete (either everyone has voted, or the vote has timed out). This system appeared to run quite fast. It used Emacs windows for its interface. In the current version, the program ran on multiple machines, consisting of code in Emacs/Lisp, Common Lisp, C, and machine code, all embedded in a LaTex document.

"We feel the Activity Coordinator, although currently in a very developmental stage, adds well-deserved attention to several important dimensions in the overall KBSA research program. The first is its explicit recognition that software development is an activity which almost always involves the interactions of multiple agents, thus requiring coordination. Further, the project's recognition of the importance of persistent data and long-lived transactions, as central to the software development process itself, if very important in our view. We see the Activity Coordinator project as one whose products will clearly further the KBSA program's overall goals."

* * *

Andersen also reported on their KBSA plans for 1990, and provided general comments on their experience in the Consortium (which was asked of each developer). The details follow.

Technical Plans for KBSA Technology Use in 1990: "In early 1990, we anticipate receiving a current version of the PMA, which we will evaluate extensively in connection with the Concept Demo project. As part of the Concept Demo, we hope to integrate and leverage both ideas and software from the existing KBSA ARIES, PMA, PEA, Development Assistant, and Framework facets. We regard this as an important step in the evaluation of KBSA technology, since evaluation to date has been focused necessarily on individual KBSA facets outside of the intended context of integrated support for the entire software development process.

"While there are no specific technical plans for their use, our Human Systems Integration Laboratory's Collaboration Technology Project has expressed interest in the Activity Coordinator and Framework technologies."

General Comments on Andersen's Experience in the Consortium: "Andersen Consulting is very interested in continuing to participate in the KBSA Technology Transfer Consortium, both as a developer of the Concept Demonstration and as an EPE Site. We note, however, that while some KBSA technology has been transferred to our research center, little (if any) of it has been transferred further into Andersen. Given the current state of KBSA art, this may be appropriate, as much of the work done to date has been at the 'proof of concept' level. In order for the transfer process to extend further, into our commercial CASE tool products and/or widespread use in our systems development consulting practice, the following issues need to receive greater attention in KBSA research in the next three to five years:

- capabilities for handling persistent data and long-lived transactions;
- demonstration of the technology's capability to scale-up to problems of industrial size;
- improved usability of KBSA functionality and articulation of process models which support its application in industrial software development;
- coexistence and interaction of KBSA-developed software with previously developed software;
- support for data-intensive applications;
- support for the evolution of systems once they have become operational (and are far more heavily constrained with respect to modification and enhancement);
- support for programming in the large, i.e., the complex interactions between teams and individuals involved in large-scale software development using distributed workstations; and
- support for design and optimized implementation at the system (as well as the component) level.

Boeing Computer Services (EPE Site)

A. The evaluation of strengths and weaknesses of KBSA technology as it pertains to EPE-Site needs.

Boeing sent personnel to all KBSA facet trainings offered since the consortium began. They received the Specification Assistant from ISI and the Framework from Honeywell. At the time of their report, they had not received the Requirements Assistant from Sanders or the Performance Assistant from Kestrel. Below is a brief synopsis of evaluations derived from Boeing training-class attendance.

1. **Requirements Assistant** (evaluated by Philip Newcomb)

"The strong suit of the Requirements Assistant is its knowledge representation language: the Structured Object Constraint Language Environment (SOCLE). SOCLE is however, the part of the Requirements Assistant which is proprietary software and consequently not available to Boeing.

"Of the various tools built around SOCLE, which make up the Requirements Assistant, the spreadsheet-like editor for specifying and testing constraints looked to be the most useful and stable.

"The natural language parser for requirements capture and the various graphical editors were fragile and not particularly exciting when compared with commercial offerings of similar ilk.

"The overall set of tools along with SOCLE do constitute a conceptually viable approach to requirements capture which is considerably advanced over pen-and-pencil approaches. It is doubtful, however, that the Requirements Assistant can compete with KEE, ART, Knowledge Craft, Refine and several other commercially available and mature modeling and simulation languages which can accomplish much the same thing.

"Perhaps the instructional materials, technical reports and presentation materials will in the long run prove to be the Requirements Assistant artifacts of greatest value."

2. **Specification Assistant** (evaluated by Philip Newcomb).

"The Specification Assistant has some serious problems which impair its usefulness as a specification language. Let's use the term 'Tool Leverage Principle' to represent the basic guideline that a tool should leverage human effort to amplify output.

"The implementation of the human-directed transformational paradigm in the Specification Assistant violates the Tool Leverage Principle. The basic structured refinement approach of the Specification Assistant is not objectionable. The refinement of an abstract high-level specification into a low-level executable specification is a good method of expressing a concept model at successive levels of detail. The process by which the user accomplishes this task in the Specification Assistant is hopelessly unusable. The application of transforms by the user is an unguided and complex search task which exceeds the patience of virtually any user."

3. **Framework** (evaluated by Jim Fulton).

"Honeywell's framework was viewed as irrelevant to Boeing needs or interests. The evaluator advised against Boeing making any effort to use the technology."

4. Performance Assistant (evaluated by Jim Patterson & Philip Newcomb).

"The Performance Assistant was viewed as a well thought out specification optimization tool of high potential value to Boeing.

"The general consensus reached by Boeing personnel exposed to Kestrel Institute developed KBSA facets and to Reasoning System's REFINE is that these tools are most likely to be usable by Boeing.

"Boeing has conducted several projects with Reasoning Systems and the Kestrel Institute in software reverse-engineering and reengineering with good preliminary results.

"Considerable interest has been expressed in the Development Assistant and the Project Management Assistant from a variety of Boeing Software Organizations."

B. Technical level descriptions of how the site plans to use KBSA Technology.

"Boeing is not currently using any KBSA technology. The two facets we have used are not deemed usable. We are however using KBSA spinoff technologies.

"Our technical work involves development of parsers and translators with Reasoning Systems and the Kestrel Institute for several computer languages and the development of interactive workbenches for reengineering computer systems data models and programs.

"Boeing is likely to find uses for the Performance Assistant as part of our software reengineering efforts aimed at analyzing and optimizing existing codes.

"The Boeing Advanced Technology Center is planning to sponsor a one-day seminar on KBSA technology at the end of May, 1990. This seminar will be prepared by the Boeing focal point (Philip Newcomb). The KBSA facets we have received will be demonstrated and an hour lecture will be prepared for each of the KBSA facets."

C. An evaluation of developer/RADC/EPE-Site communications and interactions in respect to their suitability for supporting robust technology transfer.

"You have to have a robust technology to achieve robust technology transfer. As pointed out above, most of the KBSA facets are not robust. The current level of communications is appropriate. As the KBSA technology matures it may become something to shout about."

D. General Discussion.

"Generally, the training has been excellent. In spirit the OCA and Consortium are well conceived. The biggest headaches have come in getting license agreements and actual delivery of facet software. If the RADC had taken on the task of distributing the KBSA software some of the problems with accessibility to the software might have been overcome.

"The Advanced Technology Center itself could be more vigorous in promoting the KBSA software. However, given that most of the software has been found to be non-robust the more conservative approach we've adopted has been sound. We have focussed upon using KBSA spinoff technologies on high valued problems. We think this is a sound approach.

"In the future, our technology transfer approach will put greater emphasis upon education of the Boeing community about the KBSA facets. We will continue to promote the more mature facets and production quality spinoff technologies."

McDonnell Douglas (EPE Site)

A. The evaluation of strengths and weaknesses of KBSA technology as it pertains to EPE-Site needs.

"Strengths: The strengths of KBSA technology have thus far been in the ability of RADC and KBSA software developers to recognize promising new directions within software engineering or weaknesses within initial KBSA concepts and to take corrective steps. Examples are: Evolution to an object oriented design approach within an X-Windows environment; the decision to combine the requirements assistant and the specification assistant into a single software facet; and the decision to contract for an integrated working prototype containing the paradigm's proven features. While it was not possible to anticipate the need for the trends within the industry, a flexible and adaptive viewpoint has kept the research on track with industry needs.

"Weaknesses: The following comments were introduced during the annual KBSA Conference, but are repeated here for the record.

"--KBSA envisions an environment wherein the customer becomes the software developer, supported by a "smart" software assistant (analogous to evolution of telephone operator support over the past 60 years). In providing this support, KBSA seeks to avoid imposing a disciplined methodology on the customer-developer. We agree with the broad vision but not with the concept of method-free support. The discipline of methods is fundamental to sound engineering, project continuity through personnel changes, etc. Furthermore, we think implementation of a method would enhance progress on defining the support to be provided by the software assistant. Many of the shortfalls of the Phase I Requirements Assistant would be much easier to solve within the framework of a method (a prime example is the divergence of views of the emerging system between the developer and the software assistant). We recognize that an optimal method supporting the KBSA vision does not exist today, but we believe it will evolve and adoption of a current method would permit the research to focus on more important issues such as generating early simulations and evolving them into software system prototypes, effective use of graphics to generate the compilable design specifications, etc.

"--More attention should be given to graphical support for requirements simulations and to graphical representations of the compilable specification language.

"--The choice of a compilable specification language, not commercially available, makes technology transfer a more difficult task."

B. Technical level descriptions of how the site plans to use KBSA Technology.

"The Phase I KBSA prototype software is not sufficiently robust to justify a rigorous evaluation within industry. Accordingly, we have confined our efforts to familiarization with the implemented concepts and provision of feedback to RADC and the developers. If the integrated prototype being developed by Andersen Consulting [of Arthur Andersen & Co.] is sufficiently robust and will scale to industrial problems, we will use it on a small parallel development effort and proceed as indicated by the results."

C. An evaluation of developer/RADC/EPE-Site communications and interactions in respect to their suitability for supporting robust technology transfer.

"MDC interest in KBSA is as a tool user, not as a tool builder. KBSA technology has not reached the level of maturity needed for a tool user to consider robust technology transfer actions. We regularly convey our interest to tool vendors (within the Consortium and outside of it) in participating in a commercial implementation of the KBSA paradigm. The level and type of MDC participation would be subject of negotiation. Examples of participation might be evaluation of alpha software, support through venture capital, etc."

D. General Discussion.

"We believe the Consortium and the governing Operational Concept Agreement have been effective to date.

"Recently, developers have been discussing the merits of shifting KBSA research to hardware which supports a UNIX, X-Windows environment. The trade-off appears to be the advantages of access to commercial software environments and compatibility with the leading edge of hardware technology versus the disadvantages of converting from the current environment. We think the switch will ultimately be driven by KBSA's need for maximal hardware processing capability. If this is true, then the sooner it is done, the lower the impact.

"Training has improved steadily from the first offering to the most recent. The training provided by Kestrel was the best to date. Perhaps the single feature which made it so was the provision of detailed tutorial handouts which will permit efficient transfer of the training lessons to our home environment. One serious weakness in all training received to date is the long time lag between training and delivery of the software. This seriously erodes the learning curve."

Rockwell International (EPE Site)

A. The evaluation of strengths and weaknesses of KBSA technology as it pertains to EPE-Site needs.

"KBSA introduces new software paradigms to Rockwell software development. The most valuable features are the object-oriented environment, the capture of software design at the beginning of software development, and the replay mechanism (e.g., Performance Estimation Assistant). Although Ada offers abstract data types that provide for object oriented programming, it has no inferencing capability and doesn't handle property inheritance.

"KBSA tools are knowledge based. They provide inference mechanisms and rule generation to assist in automatic program generation. Some of the CASE tools (e.g., TEAMWORK) offer early-stage design capture and provide assistance to the user in designing the system in an implementable pseudo code. However, they are not knowledge based and do not support replay following design modification.

"KBSA concepts and their applicability to small-scale problems were demonstrated at the technology-transfer classes during 1988 and 1989. Overall, KBSA technology is still at an early stage. The current tools that represent different facets of the software development cycle are quite independent, having been built on different platforms. User interfaces to tools differ in both form and syntax; and the tools have different underlying systems. Although the Framework integrated KBRA and PMA, a maintenance problem currently occurs when either facet is updated. This is attributable to built-in macros used to translate the source code. This problem will be eliminated in the next phase of KBSA because the different tools will be built on top of the common framework rather than upon independent foundations.

"The four KBSA tools that Rockwell has limited hands-on experience with are discussed briefly in the following:

"1. **Knowledge Based Requirements Assistant (KBRA):** From what has been learned in the training class and demonstrations, KBRA is a useful tool that allows system engineers to enter their requirements through different representation formats, making modifications incrementally. The user needs to know SOCLE (Structural Object and Constraint Language Environment) to develop domain-specific reusable components for the library. Since KBRA generates a DOD-STD-2167A document, the software will need modification as the standard changes.

"2. **Specification Assistant** (SA): With little knowledge about Gist (a high level specification language) and limited hands-on experience with the SA, it is difficult to apply SA. An interface may have to be created for SA to enable an average engineer to adapt to using the Gist formal representational language. Transformations are hard for the user to understand and difficult to apply at the right time. However, SA has good features, such as reusability, the theorem prover, and the English Paraphraser. These concepts can be applied to various projects. For example, software reusability has been applied to most of the projects in the Expert Systems and Diagnostics group. A simple theorem prover has been implemented in the Expert System shell developed by Rockwell to provide consistency checking and derive new implications of fact. And the English Paraphraser concept may be applied to the Expert System to output a text-form explanation of rule traces.

"3. **Framework:** Rockwell received a Symbolics tape containing the Framework software and the demonstration examples of PMA (Project Management Assistant) and KBRA. The tape was loaded onto the Symbolics Genera 7.2 operating system. Unfortunately, the executable files were a problem during the loading process. Several attempts were made to delete the executable files and recompile all the LISP files. Although the source was compiled, problems occurred in loading the "make file" that sets up the KBSA Framework. The Framework developer was informed of this problem. Because of Rockwell's Symbolics configuration, a hard copy of the loading process is not available for further debugging.

Framework is not a transparent tool to the user with limited experience. Without a good understanding of metaclass programming in the Common Lisp Object System (CLOS)--especially the way distributed objects are handled--it is difficult to develop application programs using the Framework as a tool. However, the user interface environment (KUIE) is not difficult to adapt to since a set of predefined "contacts" is available.

"4. **Performance Estimation Assistant (PEA):** The user needs a fairly good understanding of PERFORMO or REFINE to write a functional specification in PEA. Once a formal specification is completed, the user can start a PEA session. It was very helpful that users had handouts of the examples at the PEA training class. The handouts enabled the class to follow the scripts for entering commands and therefore obtain the expected results. It is not clear to the user, however, why or when certain KIDS (Kestrel Interface Development System) commands should be applied. This knowledge may be gained as familiarity with the tool grows.

"Small examples have been demonstrated on PEA. If the user can develop the formal specification of a function, PEA can generate executable code. Rockwell anticipates Kestrel's implementation of the Air Traffic control system in PEA this year."

B. Technical level descriptions of how the site plans to use KBSA Technology.

"Rockwell plans to install future KBSA tools on a Sun 3 or Sun 4 workstation to continue evaluating KBSA software. Rockwell will also continue trying to resolve the licensing problems with the existing tools.

"Rockwell has an on-going, two-year study plan to compare the KBSA tools with Microelectronic Computer Technology Corporation (MCC) tools. The first study report, Rockwell IR&D project 830, was issued at the end of the fiscal year 1989. In that report KBSA's Framework and MCC's Leonardc Deli were compared as development environments. KBRA, SA and Leonardo Gibis were discussed in the design aids section.

"The KBSA User Interface Environment (KUIE) developed as part of the Framework is a portable system for constructing user interfaces and provides a collection of predefined user interface components and protocols. Similar to KUIE, Rockwell developed a generalized graphics package called the Rockwell Interactive Graphics System (RIGS). This package provides reusable user interface functions on the IBM PC. Both KUIE and RIGS are portable and provide modularity, and user interfaces can be implemented by calling the interface functions without accessing the underlying systems. The differences between the two systems are summarized in the following:

"1. KUIE uses the Common Lisp User Environment (CLUE), which builds on the X-Windows system. RIGS is written in C, provides its own window system manager, and uses the lower level graphics system, HALO, from Media Cybernetics. A benefit of RIGS is that it has no underlying lower level window system to worry about. KUIE runs on Symbolics and Sun workstations while RIGS can only be used on IBM PC's.

"2. KUIE is designed within an object-oriented environment. The human interface functions (e.g., menus, forms, and buttons) are represented by the contact object class. In KUIE the functions of the application programmer and the contact programmer are distinct. In RIGS the application programmer has to define and draw windows, menus, and buttons using RIGS functions.

"3. In addition to having the capability to display drawings "on the fly," RIGS also allows a user to display drawings generated by CAD/CAE.

"The KBSA tools that Rockwell is considering for use in a project are the Performance Estimation Assistant and the Project Management Assistant. Once Rockwell obtains a copy of REFINE and PEA, the software group will use the tools to generate executable Ada code from small functional descriptions of diagnostic problems.

"Based on the PMA demonstration, it seems that task and component planning for a software project can be accomplished. A Computer Software Configuration Item (CSCI) structure can be defined using PMA; and the Top Level Computer Software Components (TLCSCs) and the Lower Level Components (LLCSCs) can be added as the project unfolds. PMA also provides project scheduling and monitoring capabilities that will assist in software project management. When PMA is available to the EPE Sites, Rockwell will use it to implement part of an on-going expert system project.

"We utilize internal organizations such as ASCG (Anaheim Software Coordination Group), the company-wide AI Tech Panel, and internal briefings (such as IR&D progress meetings) to report KBSA TTC activities to other software organizations within the company.

We are also considering co-authoring a paper at a future Rockwell Software Engineering Symposium, an annual activity where all Rockwell organizations participate."

C. An evaluation of developer/RADC/EPE-Site communications and interactions in respect to their suitability for supporting robust technology transfer.

"Rockwell has not had many interactions with the KBSA Developers because Rockwell has only one distribution tape. And based on our experience with this tool, it is difficult to load new software with only telephone line support. The hardware configuration and other supporting software are also important factors in loading and executing the KBSA software. The user documentation should specify software and hardware configurations required for operation. It would also be most helpful if detailed loading and testing procedures were included with distribution tapes.

"RADC has played a very important role in monitoring the KBSA development and informing Consortium members of development progress. The active involvement of RADC is very helpful for the KBSA community."

D. General Discussion.

"The KBSA Technology Transfer Consortium represents all three sides of KBSA development--contractor, developer, and industrial users. With suitable technology transfer and feedback from the users, KBSA will evolve into a real world system. At this time, Rockwell feels strongly that standards need to be set on both software and hardware configurations. KBSA is intended for multiple users; therefore, the issue of host computer availability needs to be addressed. The second issue is software support. KBSA Developers are not obliged to provide immediate support for their tools. Finally, licensing issues remain to be resolved before these tools can be made available to the public.

"The Expert systems and Diagnostics group at Rockwell has issued several trip reports and distributed them internally. This is our primary way for transferring KBSA concepts to the organization aside from internal management briefings. Once KBSA tools are established and a working example has been implemented, the group will demonstrate the system to interested parties within Rockwell.

"The technology transfer classes are very informative. However, there tends to be too much material to cover in the two-day training. Decisions should be made whether classes should emphasize tool-design concepts or tool use. Rockwell recommends emphasizing tool use for real-world applications. And since the technical concepts are well documented, it may be to a student's benefit to receive documents and read them before attending training classes. With some knowledge about the tool already in hand, the lab session could be more extensive. Following training, the user should be able to implement a small example with minimum assistance from the developer."

Honeywell (Developer)

"As with any group, there have been some good choices and some others. I believe that the selection of Arthur Anderson [Andersen Consulting] and McDonnell Douglas (specifically Doug Abbott were very good choices. It appears that both organizations have a commitment to incorporate KBSA technologies into their businesses. In my opinion, getting industrial support for/use of KBSA is critical to the success of the KBSA program, especially given the current political environment. On the other hand, several consortium organizations either made poor selections for their representatives, or had little commitment to the consortium (it struck me as strange that I had to explain how to call a function in lisp to one of the people installing the Framework). I have no idea how to minimize this sort of thing, but it is fairly clear that given the state of the prototypes being delivered by the developers that people without a fairly good understanding of the underlying languages and concepts of the facets will not gain much by being part of the consortium at this time.

"Preparing for non expert users has caused several of the developers to put effort into making experimental prototypes into products. This of course diverted some effort away from the research issues involved, but probably caused some thought to be given to issues like a consistent user interface, and facet interoperability that would likely not have been raised quite so soon.

"As for our involvement in the consortium, we have received very little feedback from the members, neither comments nor questions, other than regarding installation of the Framework."

Information Sciences Institute (Developer)

"This is a summary of the impact so far of the KBSA Technology Transfer Consortium on the KBSA work at ISI. The consortium has had a clear impact on our research program. Our participation in the consortium has been beneficial to us, and, we hope, to the consortium as well. I would also like to make a suggestion regarding how to improve future interaction with the consortium.

"Our first major interchange with the KBSA Technology Transfer Consortium was in conjunction with the training course that we gave for the KBSA Specification Assistant in February 1989. Although we had given an earlier version of the training course the previous year at RADC, giving the course to consortium members afforded us the opportunity to further polish the course and introduce KBSA to people who were not as familiar with the project as the KBSA program personnel at RADC.

"This exposure turned out to be very important, for a number of reasons. When people evaluate software engineering research, they often ask what experience the researchers have had in applying their ideas. Since it is too early to consider using KBSA prototypes on mainline software development projects, such practical experience is hard to come by. Providing a training course to industrial representatives seems to be an excellent way of getting feedback.

"We carried away the following lessons from training courses. Two of the technologies that we exhibited in the Specification Assistant, namely evolution transformations and natural language paraphrasing, proved their worth. Students found that a good way to learn about Gist was to modify specifications and paraphrase the result, to see the impact that such changes would have. The evolution transformations provided some assistance in this modification process.

"At the same time, we were surprised by the effort required to put together a really good training course, and a system that could support it. Students wanted a system that was broad enough in scope that it could be used in undirected activities. We attempted to provide this, and largely succeeded. As a result, we got somewhat better reviews than the Sanders people got for their course. At the same time, students also wanted detailed scripts that took them through a specification development in a keystroke-by-keystroke fashion. We thought that we had provided such a script, but it turned out not to have as much detail as the students demanded.

"Another useful source of feedback for us were the papers presented at last year's KBSA conference by Chunka Mui [Andersen Consulting] and Doug Abbott [McDonnell Douglas]. It was useful for us to see clear statements of these consortium members' reactions to our work. The papers provided much more useful feedback than the evaluation sheets that course participants handed in. I hope that consortium members will be encouraged to present similar papers at future conferences.

"The consortium has been of some help in providing contact with other industrial organizations, Arthur Andersen in particular. We hope that other consortium organizations will be motivated to establish bilateral contacts with us.

"The one area where we have not yet received much feedback is regarding on-site evaluations of the Specification Assistant. We have delivered the Specification Assistant to a number of organizations. This turned out to be a very painful process, both because of licensing problems and because of hardware problems in producing tapes. So far we have received little word from people as to what further experience they had with the system.

"Although we have been encouraged by our interactions with the consortium, we do see possible room for improvement. One thing that we need to do is to make clearer the kind of evaluation that is expected from consortium members. Educational computing technologists make a distinction which might be useful: they distinguish **formative** evaluation from **summative** evaluation. Formative evaluation is an initial review of a prototype of the system to be built. The purpose of this evaluation is to ensure that the system has the right functionality. It results in constructive suggestions of changes, which can then be made before the implementation is complete. Summative evaluation is a controlled field test of the software, demonstrating that the software has a positive effect. It should be made clear both to consortium members and KBSA Developers that formative evaluations are what is needed. In other words, the consortium can help most by providing suggestions early on in the development process of needed functionality. KBSA Developers should use such evaluations as a motivation for subsequent system development. A good way of doing this in the future will be to allow consortium members a chance to review rapid prototypes of KBSA systems.

Lockheed Sanders (Developer)

"The KBSA Technology Transfer Consortium plays many technology transfer roles. Importantly, the consortium helps to steer KBSA technology development. Concepts are identified, debugged, and expanded. In addition, members learn how to match the capabilities and goals of KBSA to the needs of particular application areas and they receive training in the promotion of the technology, in developing KBSA-like components, and in specific mechanisms that are used within KBSA components. We believe that the consortium has been very successful in helping to direct KBSA technology development (I will provide a few examples of this below). We believe there have been many examples of successful technology transfer through discussions, demonstrations, and hands-on sessions. The one weak area has been in promotion of the technology. The prototype systems (our own included) are not production quality systems. They are by definition only facets of a full KBSA and are limited in portability and robustness. In time, these prototypes will develop maturity and help us to gain wide support for the technology.

"As developers of KBSA technology, we provided consortium members with information on the work we have undertaken. The first KBSA tutorial session was held at our site and focussed on KBRA, the prototype KBSA Requirements Assistant. Most of the comments in this report are based on these training sessions.

"Recognizing that detailed issues might best be dealt with at a follow-up session (or through individual contacts between ourselves and other consortium members), we titled the session 'KBRA Technical Exchange,' presented KBRA as a discussion starter, and allowed considerable time for discussion during the two-day exchanges. Post-session participant evaluation forms were very positive and agreed with this orientation for a first encounter with a KBSA component.

"Discussions at the technical exchange were as much about concepts as they were about architecture, mechanism, metamodel, how to run a demonstration, or how to extend the prototype into a particular application area (although the emphasis varied depending on the interest of the participants). These discussions were a highly effective and valuable consortium work product. I know that many observations made during the session have helped us steer software assistance work at Lockheed (on subsequent KBSA sponsored work and other efforts as well). A few specific examples can help highlight this contribution.

"1. Presentation: Additional presentations that would be valuable for requirements work were identified. For example, entity-relationship diagrams are very effective for describing database systems (e.g., a library system). Also, analysis tools which complement KBRA's calculator/spreadsheet would be helpful. One envisions families of tools for different types of non-functional properties of specifications.

"2. Reusability: We have just started on the path toward effective reuse of requirements information. In addition to the functional taxonomies that are present in the current KBRA library, one should be able to find and mix-in reusable descriptions of real-time systems, reliable systems, secure systems, and systems representing other orthogonal dimensions in reuse.

"3. Domain modelling is very important: Many participants expressed an interest in moving the operating point--expanding the amount of knowledge and the variety of presentation modes--within a particular application area. We demonstrated the current support for this (a DEFMODULE primitive and the KBRA metamodel). An important growth area would be in tightly integrating domain acquisition with requirements elicitation in a requirements tool. That is to say, the user should be able to uniformly interact with graphical and textual presentations for both types of work.

"4. Process Model: A process model is necessary to provide guidance to users. In fact, several process models need to be formalized. KBRA is intended for use from the first day of a project and needs to support very unstructured creative explorations. On the other hand, much real-world requirements analysis is a team effort which benefits greatly from committing to a particular methodology. An important issue that surfaced in the discussions is the tension between providing support for current methodologies and process models and the potential for developing a new KBSA methodology as we gain experience with the paradigm.

"Based on our own observations and participant evaluation forms, we found that the technical exchange succeeded by allocating time for discussions, by providing participants with resource material prior to the technical exchange, and by spiraling in on details.

"In all, our assessment is that the consortium activities--technical exchanges, tutorials, meetings, critiques--have helped to direct KBSA growth, have lead to an exchange of important information, and are essential for the success of KBSA."

Kestrel Development Corporation (Developer)

"The consortium has shown signs of life. Most members attended the KBSA conference and the various training courses on the initial KBSA facets. A small number of EPE Sites even experimented on their own with some of the facets and provided useful feedback to the developers.

"The feedback provided seemed to indicate that the consortium members appreciated the ideas underlying the facets but felt the realization of these ideas in the prototypes was too immature to allow for experiments that go beyond the toy example stage--for instance, "shadow projects" in those companies. The best feedback we received at Kestrel came from Andersen Consulting; they took a very close look at our KBSA facets as part of their Concept Demonstration contract.

"For the KBSA consortium to achieve its purpose, i.e., to facilitate the transfer of the KBSA ideas and technology into industry, there needs to be a technology push from the KBSA Developers and a technology pull from the industrial consortium members, especially the EPE Sites. Owing to the immaturity of the technology, the technology push has not been strong enough to jolt the EPE Sites into anything more than taking a few interested looks at the KBSA facets. To my knowledge, there have been very few, if any, sustained interactions between the facet developers and the EPE Sites, so the technology pull has been almost non-existent.

"As mentioned above, we did have, and continue to have, some very good interaction with Andersen Consulting. Of course, Andersen receives impetus and funding for these activities from its Concept Demo contract. This point, unfortunately, seems to be crucial. Perhaps understandably, considering the immaturity of the technology and the associated risk, the industrial consortium members have been reluctant to commit more substantial resources to their KBSA activities. Overcoming this reluctance will be necessary if increased fruitful interactions between technology developers and recipients are to take place in the future.

"The cooperation between developers and other consortium members need not be limited to existing KBSA facets. For instance, one of the consortium members has induced us to apply KBSA technology to reengineering problems. This sort of "consortium fall-out," I believe, is desirable, and presents a perhaps very effective technology transfer path. Perhaps it is just this kind of opportunistic interaction, rather than the planned consortium activities, that will produce the major benefits of the program."

Software Options

"Our involvement in the KBSA Technology Transfer Consortium came at a time when our work (in activity coordination) was not as far along as that of the other developers in their respective areas. As a consequence, we have had less interaction with, and less of an impact on, other members of the consortium than have the other developers. However, the support provided by our participation has had a pivotal impact on our work on activity coordination, and present support for this work (SBIR funds coming through the Naval Ocean Systems Center) is going toward an implementation of ideas that evolved during our KBSA contract.

"Briefly, our work consisted first of examining two activity coordination problems: software trouble reporting in a multi-contractor setting, and an analysis of the ESD System Acquisitions process. Out of these studies grew the conviction that computer-aided activity coordination can be of immense value. Part of this stems simply from the exercise of trying to formalize, even to a very gross level of detail, what happens in the activity and, part, from the automation and attention to detail that only computers can provide. However, for a system to be of value in real-world situations, it must be unlike most software systems. First, it must be long-lived, because projects go on for days or years. Second, it must be widely distributed. We noticed, for example, that one of the steps in the ESD process is to send a document to some high official in Washington and, if nothing happens in 20 days, to proceed. Ordinary programming languages and systems do not provide simple statements having such semantics.

"Our analysis of these examples provided the starting point for the development of a formalism on which to build activity coordination specifications, called "transaction graphs". There is a technical report on this subject, and this is not the place for even a short summary. Suffice it to say that transaction graphs address the issue of activities connected by potentially slow communication and provide a way of composing hierarchical specifications of activity patterns using a graphical notation. It is an implementation of transaction graphs that is mentioned above as being underway.

"In the longer term, our goal is to combine our work on transaction graphs with other work that we have done on configuration management to achieve a system whose scope ranges smoothly from that of an individual programmer to that of a nation-wide, or even world-wide, organization."

Missing Reports

The following EPE Sites did not submit status.reports:

Lockheed
Reasoning Systems
Rochester Institute of Technology
Texas A&M

Discontinued Participation

The following EPE Sites discontinued their participation in the consortium:

Martin Marietta
Software Productivity Solutions
Texas Instruments

ROME AIR DEVELOPMENT CENTER ASSESSMENT

The KBSA Technology Transfer Consortium was founded on voluntary participation (at least for EPE Sites) tied to a non-monetary "quid pro quo" principle. In some ways this has been a bold experiment because the exchange of dollars for services is the normal way of doing business in the DoD. Yet, the thought is very appealing of being able to foster customer/developer communications in the early stages of KBSA's evolution without first having to jump through legal and other resource-intensive hoops to make it work well. It's clear from the status reports that some success, although hard to measure quantitatively in most cases, has been achieved. It's also clear that there's room for improvement.

Because of what we've learned from consortium meetings, status reports, and one-on-one discussions, we now have a better idea of the challenges we should focus on in the coming year or two in order to get the greatest payoff. At the forefront is EPE-Site participation, the very heartbeat of the consortium. Yet, in this era of dwindling budgets, we see a growing threat to continued participation and commitment.

As industrial and government managers and accountants set their new-age priorities and wield their surgical pens, competition for money gets harder and harder--money needed to do the basic things we know must be done to run an effective and efficient program. And since strategic thinking in management becomes scarce in hard times, participation in a consortium whose current products are largely exploratory may not sit well with some.

Therefore, our first challenge is to become more adept at clarifying and demonstrating the inherent value of early industry-developer cooperation in the KBSA program. Following that, our next challenge is to become more creative in allocating and using any resources that we still have left after the inevitable surgery has taken place. A good start would be a reduction in travel costs for consortium activities.

Since prototype trainings have been completed for the near term, there will be no need for additional travel or training-registration fees before late 1991 or 1992. However, as developments proceed and updated trainings or evaluations become desirable, we will keep our eyes open for travel-consolidation opportunities.

The first deliberate action RADC is taking to reduce travel budgets while also increasing participation and synergy is to hold one consortium meeting per year instead of two, and to do it in conjunction with the fall KBSA conference. Special planning for this will be effected for 1991. To compensate for the reduced number of meetings, RADC will set up and moderate an on-line (but not real-time) news service and discussion forum, access to which will be required for consortium membership.

If certain kinds of cost-sharing arrangements were available, EPE Sites--and perhaps developers not currently under contract--could be more effective in contributing to the consortium's tech-transfer goals. Further, they would find management approval for their participation easier. We have only one sample to work with so far: Andersen Consulting's tech-transfer activities as a paid EPE-Site/developer (under the Concept Demonstration contract, where a portion of the management costs are being shared by Andersen). To date, this effort has yielded benefits beyond expectations. Other types of cost-sharing arrangements might involve items such as consulting hours, travel, software licensing, researcher/commercial-developer arrangements, or training internships at RADC. A new vehicle, called the "Cooperative Research and Development Agreement" (CRDA), is also available to enable RADC and industry to participate in R&D programs without money exchange but with sufficient quid-pro-quo incentives to make the arrangement viable.

KBSA contracts are now being written with a requirement for consortium participation. RADC is emphasizing flexible, contractor-proposed arrangements wherein EPE Sites can contribute early comments and evaluations of KBSA technology on a cost-shared, as-requested basis. In addition, the search for other flexible contractual vehicles devoted entirely to early evaluation and tech transfer will be made in the coming year. However, because any cost-sharing approach could impact the administrative and legal simplicity[1] that the consortium has so far enjoyed, any such approach will have to be closely studied for its possible adverse side effects.

The licensing problems that are preventing some EPE Sites from gaining access to prototypes have no easy solutions. Some corporate legal structures have proven themselves to be quite inflexible about licensing proprietary technology. The consortium's best solution for the future--one that RADC supports--is a calculated effort at shifting toward commercially

[1]Except for licensing problems related to proprietary software development platforms.

available development platforms and other higher-level software standards. In the meantime, RADC will continue to write new KBSA development contracts with an eye toward minimizing such problems. For example, the KBSA Concept Demonstration contract was written to require the delivery of 5-year transferrable licenses along with the demonstration system.

Another notable problem that EPE Sites have reported is the slow delivery of prototypes following training (and its impact on the learning curve). This can be eliminated in the future through better planning by trainers.

The developer report prepared by the Information Sciences Institute raised an important issue (page 29) about the types of evaluations EPE Sites are expected to take part in. The bottom line is that we, for the most part, are still in the "formative" evaluation stage for most KBSA products. Per ISI's request, we intend to make this as clear as possible to prospective consortium members. In addition, we would like to see developers work toward mobilizing EPE Site "user groups" as suggested in the Andersen Consulting report (page 11).

Concerning new membership in the consortium, there are three issues that need addressing in the near term: the consideration of limited new membership, membership qualification criteria, and the accommodation of "old" members who are no longer able to place suitable priorities on the responsibilities they agreed to at the outset.

The latest version of the OCA contains membership qualification criteria for the first time. Furthermore, these criteria have been updated from their original state to better reflect the consortium's present direction. Therefore, the OCA is now in a form to serve as a self-sufficient document for current members, membership candidates, or other interested parties.

Although the invitation may be withdrawn at any time to limit consortium size, the consortium is now open to new members (fall 1990). Candidates must submit packages showing how they meet the qualification criteria given in the OCA. RADC will evaluate the packages, possibly asking current consortium members for recommendations. Therefore, submitted packages should contain no proprietary data. They should also be sized and structured to permit easy photocopying and distribution.

Since the consortium embodies a voluntary, quid pro quo arrangement (except for one or two developers who currently have a contractual responsibility to participate), members are under no legal obligation to live up to the OCA. Consequently, some haven't. This current lack of sufficient formality is at once a consortium strength and weakness, with RADC having the task of balancing the dichotomy. In the coming year, RADC will lean more toward limiting consortium membership to only those players who are in a position to contribute. Furthermore, greater clarity in the OCA is expected to help alleviate misunderstandings about participation responsibilities.

To recap eight (non-prioritized) points made in the Andersen report concerning issues that need greater attention in KBSA research in the next three to five years:

1. capabilities for handling persistent data and long-lived transactions;

2. demonstration of the technology's capability to scale-up to problems of industrial size;

3. improved usability of KBSA functionality and articulation of process models which support its application industrial software development;

4. coexistence and interaction of KBSA-developed software with previously developed software;

5. support for data-intensive applications;

6. support for the evolution of systems once they become operational (and are far more heavily constrained with respect to modification and enhancement);

7. support for programming in the large, i.e., the complex interactions between teams and individuals involved in large-scale software development using distributed workstations; and

8. support for design and optimized implementation at the system (as well as the component) level.

In addition to the above eight points, the following three were added at a consortium meeting held at Kestrel Development Corporation (Palo Alto, CA) on 12 July 1990:

9. support for design and optimized implementation at the system (as well as the component) level;

10. support for the design of large systems;

11. approaches to integrating existing methods in software reengineering.

As a final point, feedback from the consortium has made it clear that rapid prototyping, with extensive early review by potential users, is unequivocally the most effective way to develop software. This is only fitting, for the KBSA paradigm subsumes rapid prototyping. The Concept Demonstration contract was written to employ rapid prototyping, with the contract's preliminary design review having taken place at the 1990 KBSA conference. The first-year rapid prototype was explained and demonstrated at that time, and from all accounts, the public PDR was a notable success.

APPENDIX A

KNOWLEDGE BASED SOFTWARE ASSISTANT PROGRAM (KBSA)

Donald M. Elefante

Rome Air Development Center (COES)

Griffiss AFB, NY 13441-5700

Abstract

In 1983, Rome Air Development Center (RADC) published "Report on a Knowledge-Based Software Assistant." [14] That report integrated key ideas on how artificial intelligence (AI) might be used to design, develop and maintain software over the complete life-cycle. Since then, RADC initiated the first of three multiple-contract iterations intended to develop both a Knowledge-Based Software Assistant (KBSA) and its supporting technologies. This paper provides a KBSA overview and describes the most interesting results to date.

What is KBSA?

The Knowledge-Based Software Assistant is a formally based, computer-mediated paradigm for the specification, development, evolution, and long term maintenance of computer software. The paradigm captures system evolution history, providing a corporate memory of how parts interact, what assumptions were made and why, the rationale behind choices, and how requirements were satisfied. It also features an explanation facility that supports developers in grasping the project's state at any time and tying it to the specific application at hand. The current, evolving version of KBSA achieves these functions with a set of software modules or "facets" that can be thought of as sub-assistants. In the future, the facet functionality will be integrated through a common, underlying "framework" that coordinates life-cycle activities. Facets (and their progeny) currently under development support project management, requirements definition, system specification, program development and performance maximization. Facets under consideration for future development will support documentation, testing, and other areas yet to be defined.

The KBSA concept formalizes all activities and products of the software life-cycle, and all life-cycle activities are machine mediated and supported. In this connection, it exhibits four main features which depart from the most common software engineering paradigm, the "waterfall" paradigm (and its close derivatives).

Distinguishing KBSA Features

The first distinguishing feature is the use of incremental, executable and formal specifications. "Incremental" means that the specifier may gradually add detail to the specification as pertinent knowledge becomes available. This detail can be added by interacting with appropriate specification presentations that help manage process complexity. One need not start out with a complete, or even an accurate, specification. "Executable" means that the specification "runs" as a high-level prototype and portrays the states and activities that will occur in the modeled system as it is exercised. This portrayal helps the specifier validate the specification against user intent. "Formal" means that the specification is expressed in a language exhibiting precise semantics. This avoids natural language ambiguity and renders a specification accessible to machine reasoning.

The second distinguishing KBSA feature is formal implementation. This means that all decisions made during system implementation are captured in formal descriptions and justified by analyzing those descriptions. This formality supports the use of meaning-preserving transformations for creating lower and lower implementation levels. The result is a verified implementation, where implementation validity arises from the very process by which the implementation is developed.

The third distinguishing KBSA feature is a project management policy that is formally stated and enforced. That is, project policy defines relationships between various software development objects (e.g., requirements, specifications, code, test cases, bug reports, etc.), and KBSA ensures that the relationships are held throughout development.

The last main, distinguishing feature of KBSA is that maintenance is done at the requirements and specification levels. This is important since maintenance activities are normally a result of new or better-defined user requirements. Requirements and specification maintenance then drives changes in the lower-level code through meaning-preserving transformations of the specifications. This high-level maintenance approach permits integrating old and new design decisions and validating the package before new code is generated. In addition, the formal capture of requirements, specifications and high-level maintenance activities permits simply "replaying" the development after modifying specifications or refinement decisions. Retention of the original design history helps to identify which parts of a fielded system must be modified. This makes the maintenance and upgrade of fielded systems simpler and more reliable.

These four features indeed capture the larger vision of the KBSA concept. However, system integration is also a major component of the larger vision. Additionally, supporting technologies and their evolution have played a critical role in KBSA development success.

System Integration

System integration in KBSA will involve defining and building a support environment that supplies full life-cycle-support services to KBSA users. The environment's role will be to embody the framework, facet, user-interface and other support functionality deemed necessary for a KBSA. However, this should be accomplished with a seamless, well integrated package that eliminates arbitrary facet distinctions. Therefore, an ideal support environment interface will have five characteristics. First, it will be implementation independent. Second, it will exhibit a well-conceived interface and communication protocol suite. Third, its services will be present at many granularity levels to support both commonality in usage and specialization. Fourth, it will exhibit quality explanation and help facilities. Finally, it will support extendibility to provide the integration of new functionality in a uniform way. These service characteristics are largely analogous to those provided by a kernelized, "portable" operating system.

Supporting Technologies

Evolving technologies that have been instrumental in supporting KBSA development fall into three main categories: the wide-spectrum language, general inferential systems and domain-specific inferential systems.

An ideal wide-spectrum language (WSL) is a single language that lets the user capture the formal semantics--at any level of detail or at any step in the development cycle--of the system under development. It permits uniform expressibility regardless of what is being described (e.g., requirements, specifications, code, test cases, project management policy, etc.), and it does so in a way that is syntactically and semantically consistent at and between all abstraction levels. The WSL is intended to cover the spectrum from requirements to implementation, from management to maintenance. A mature WSL for KBSA should also cover graphic and schematic system representations as well as conventional, written representations. A great deal of headway has been made on all these fronts.

A general inferential system is one that supports reasoning and is applicable to all problem domains. We must be concerned in KBSA with this reasoning efficiency and the data representations used. We need modular expansion facilities to accept domain-specific inference modules or more efficient decision procedures. And the user interface must make the system useful for practical applications. Mechanisms that support inheritance, constraint propagation and planning must be present, and theorem provers may be integrated with the inference engines used to perform meaning-preserving transformations of specifications. All of these qualities will be seen in the KBSA facets described later.

Domain-specific inferential systems extend general inferential systems to address the unique software development features within a given domain. Three dimensions comprise the character of a domain-specific inferential system: the application domain, the life-cycle phase being addressed, and the functional area (e.g., presentation domain, structured text domain, evolving system description). For each problem domain, special reasoning and solution-finding rules apply. We know that inferential systems inherently based on special domain rules are more efficient than general inferential systems applied to encodings of the special domain. In KBSA, this technology area has focused on achieving good knowledge representation of software development objects and inference rules; also, how they can be formally represented and used for further reasoning. Therefore, three sub-areas of investigation are also pertinent here: formal semantic models of the software development environment, the knowledge representation and management of software development objects in the domain, and specialized inferential systems that incorporate rules tuned to a specific problem or task.

When the KBSA report [14] first came out in 1983, supporting technologies were not adequately developed. Neither was the software development process well understood. To address these shortfalls, an evolutionary, three-iteration contractual approach was undertaken. The next section describes the basic approach and key accomplishments of each contract, and shows that KBSA supporting technologies have matured along with the facet developments.

KBSA Facet Developments and Related Work

The first of the three iterations was initiated in 1985 and is now largely complete. Some overlap with the second is also taking place. Iteration 1 has focused on designing individual KBSA facets and pulling the supporting technologies as needed. Universal solutions haven't been sought. Rather, solutions unique to each facet have been pursued to help facet developers understand and formalize their chosen life-cycle phases. Because of the need to better understand the software-development process with respect to KBSA, initial facet partitionings were intentionally chosen to parallel phase designations of the waterfall paradigm. However, over the long haul, KBSA will unveil its own identity and adopt the broader vision of an AI-inspired computer assistant shorn of unnecessarily restrictive notions about life-cycle partitionings or phased development.

Although the formalizations derived under iteration 1 have centered on the products of individual phases (e.g., requirements, specification, and code), they have more significantly focused on the process of how these products came about. The sections that follow elaborate on this.

The first-iteration KBSA efforts are: Project Management Assistant (PMA) 1 and 2, Requirements Assistant (RA), Specification Assistant (SA), Performance Assistant (PA), Development Assistant (DA) and KBSA Framework (KF). Efforts marking the start of iteration 2 are the Requirements/-Specification Assistant (Req/Spec), and the KBSA Concept Demo.

Project Management Assistant

Work on a formalism definition for PMA and the construction of a prototype began in 1985 [8,9,23] by Kestrel Institute. The goal of PMA within the software development life-cycle was to provide knowledge-based help to users and managers in project communication, coordination, and task management.

PMA capabilities fall into three categories: project definition, project monitoring and user interface. Project definition consists of decomposing the project into individual, manageable tasks and then scheduling and assigning them. Once manageable tasks have been defined for the project, the project must be monitored. In PMA, this monitoring primarily takes the form of cost and schedule constraints, but the enforcement of specific management policies (e.g., DOD-STD-2167, rapid prototyping, KBSA, etc.) is also included. Finally, the PMA user interface supports interactions involving project monitoring and definition, and uses direct queries and updates, Pert charts, and Gantt charts.

Although the above capabilities are important, we would expect them of any project management tool. What sets PMA apart from its predecessors is its expressibility and flexibility. Not only does PMA handle user defined tasks, but it also understands their products and the implicit relationships between them (e.g., components, tasks, requirements, specification, source code, test cases, test results, and milestones). Also captured in PMA are software development objects unique to programming-in-the-large: versions, configurations, derivations, releases, and people.

From a technical perspective, the advances made in PMA include: the formalization of the software development objects enumerated above, the development of a powerful time calculus for representing temporal relationships between software development objects, and a mechanism for directly expressing and enforcing project policy.

PMA formalizes software development products (e.g., components, tasks, milestones, requirements, specifications, test cases etc.) from a project management perspective, and provides a language to describe the process by which these products came about. Current software engineering literature calls this concept "process programming," [29] but it has been a part of PMA since its inception. Thus, PMA demonstrates that a software project can be described formally and reasoned about.

PMA also demonstrates WSL applicability within the project management domain. The WSL used for the PMA prototype is Reasoning Systems' REFINE. REFINE is best described as a programming language that provides constructs for doing specification in a variety of styles (e.g., functional, logical, procedural, and object oriented). These constructs include the following list: user-defined object classes, set-theoretic data types, constructs from first order logic, relations defined by assertions, transforms, a pattern language, and conventional programming constructs. [20]

In addition to the above, REFINE supports the definition of new constructs in PMA, i.e., assertion types used to maintain knowledge base integrity. One example is CHECKING, a trigger that looks for particular conditions and then flags the system when they arise. A second is COMPLAINING, which is often used with CHECKING. It interacts with the project manager to explain what has gone wrong. MAINTAINING is a third assertion type that automatically insures that a given condition remains true. In general, PMA has shown that a single WSL can uniformly handle both software programming constructs and process programming constructs. This is a consequence of REFINE's expressibility and extensibility.

The PMA effort spent considerable resources developing a highly expressive, generalized time calculus. The calculus, an extension of James Allen's interval calculus for reasoning about time, [1] was developed by Peter Ladkin. Allen's relational algebra has 13 atoms and is therefore of finite size. PMA's time calculus, on the other hand, uses an infinite algebra [25] whose temporal logic primitives are implemented directly in REFINE. It is the infinite algebra that gives the calculus its unique expressive power in terms of time representation.

Some of the advantages cited by Ladkin include capturing the metaphysics of time and presenting a natural way for representing time. This includes allowing for the joining of seconds into minutes and days into weeks or months such that there is a linear hierarchy of standard units.

The initial PMA effort was completed in 1986. [23] A follow-on PMA contract was initiated in November 1987. Its goals are two-fold: First, the evolution of PMA with the purpose of expanding the formalized knowledge of project management and providing enhanced capabilities. Second, implementation of PMA as an integral part of a full-scale software engineering environment called SLCSE (Software Life-Cycle Support Environment).

Requirements Assistant

The work on the Requirements Assistant (RA) [6,7,31] began in 1985 by Sanders Associates. The greatest RA challenge was to adapt to the informal nature of the requirements process. It also had to allow users to enter requirements in any desired order or level of detail. Therefore, RA's responsibility is to do the necessary bookkeeping to support user requirements manipulation, and to maintain consistency among requirements as they become known. RA's key features are (1) complexity management through the use of multiple presentations, including formal and informal expression modes, (2) reusability at the requirements level, and (3) the use of constraint propagation technology for supporting requirements traceability and trade-off analysis.

RA features a convenient user interface that spotlights multiple presentation modes, any of which can be used at any time. The user chooses the mode or modes that seem the most natural for the task. Internally, only a single representation exists to maintain consistency, for multiple presentations are typically used to capture and validate user requirements.

The presentation modes available within RA are:

a. Intelligent note pad -- Diary notes on requirements collection. Support is provided through limited natural language, structure editing, and feedback on word recognition in a lexicon.

b. Graphical presentations -- These include context diagrams, system function diagrams, internal interface diagrams, functional decomposition diagrams, data flow diagrams, state transition diagrams, and activation tables (which provide an active-function list for each system mode).

c. Calculator-spreadsheet -- The tabular display of non-functional properties for each system function (tied to an underlying constraint propagation system for bi-directional propagation, retraction and explanation).

d. Requirements document -- Natural language requirements document.

Support for reusable domain knowledge components are of particular note in RA. As the user enters requirements, the knowledge base is updated. However, requirements needn't be described entirely from scratch. RA provides mechanisms for merging "reusable requirements components" into the evolving system description. Requirements components are analogous to Rich's and Water's cliches in KBEmacs. [32] Like cliches, reusable requirements components are manually encoded with descriptions of standard system requirements derived from specific application domains (in the case of Sanders' effort, the air-traffic-control domain). When merged into the evolving system requirement, they become a source of system expectations. Expectations support the critiquing of the evolving specification. Capabilities associated with critiquing include: detecting inconsistencies and duplicate objects, checking for reasonable values, and detecting missing information.

RA provides two mechanisms for merging specific reusable requirements components into the evolving system requirements. The first mechanism employs inheritance along user defined relations. Here, when the user defines an object as an instance of some supertype, supertype expectations are inherited (supertypes are examples of reusable components).

The second mechanism employs automatic classification. Here, the system tries to establish supertype relations of the most specific type possible, thus inferring the greatest amount of information possible and providing the most specific expectations on the evolving system. RA does automatic classification by consulting built-in tables within each supertype to determine appropriateness of that supertype. These tables are generally driven by requirements analysis tradeoffs, i.e., time, space, and other resource considerations.

Unique requirements constructs were necessary for RA to have the capabilities described thus far. Sanders used SOCLE, [15] their wide-spectrum-language based on frame and constraint inference, and derived from Roberts' and Goldstein's FRL, and Steele's constraint based language. Expressed in SOCLE, three classes (realms) of requirements constructs were necessary: Presentations, Structured Text, and Evolving System Description.

A presentation architecture [3] handles objects that contain information about (1) how to display requirements in any of the presentation modes, (2) the presentation mode semantics, and (3) the impact of editing these presentations. The requirements/presentation boundary is crossed by presenters and recognizers. Presenters are concerned with the object's physical properties (i.e., the object being presented, screen coordinates, sizes, alignments, labels, and relevant graphical editing options), while recognizers are concerned with updating the evolving system description when the user edits the current presentation.

Structured Text is an object class that allows informally and formally related textual information to be managed as such. Groupings of related text are not necessarily dependent on the presentation mode, though presentation modes rely on structured text to represent text strings and their relationship to the current presentation. Initially, structured text objects are created interactively as the user enters more information. The groupings between these text objects evolve as the system is better defined. Finally, additional text objects are automatically created to paraphrase requirements statements first expressed in other non-textual presentations.

The Evolving System Description is the requirements repository in RA. Presentations and Structured Text merely reflect the Evolving System Description and, as a result, are tightly tied to it. Within this requirements construct class is an object-types hierarchy. The most general is the requirements object. A requirements object may have no more information than an associated Structured Text Unit that contains an uninterpreted text string and recognized key words.

At the next level down are activity, event, data, transition, and constraint object types. Each of these object types have more expectations on the requirements being expressed. Farther down the hierarchy, and thus more specialized, are specific reusable requirements components. Evolving-system-description components are type instances in this taxonomy.

The RA work was completed in October 1988.[16] A prototype was delivered which is now serving a technology transfer function as well as feeding in to the new Requirements/Specification Assistant effort (to be briefly described below).

Specification Assistant

The main goal of the KBSA Specification Assistant[2,21,22,24] is to yield a formal specification of the system under development and then validate it against user intent. Formal specification development must be supported in an incremental fashion, modeling the way developers typically construct specifications. Validation must be done by exposing the specification to the user at the earliest opportunity and continuing the exposure throughout the construction process. The effort to develop a KBSA Specification Assistant began in 1985.

In SA, a user begins specification development by describing a high-level GIST[21] specification of the intended system. This specification is highly idealized, is usually incomplete, and does not include exceptional-case behavior. SA then provides the user with high-level transformation commands (HLTC's) to refine the high-level specification to a low-level specification where: (1) exceptional behavior is described, (2) all agents within the application are enumerated, (3) data boundaries are clearly defined (i.e., removal of the perfect knowledge assumption by each agent), and (4) all necessary functionality is enumerated and described in a semantically complete fashion.

HLTC's are similar to transforms in transformational systems, but are not necessarily meaning-preserving. In fact, most high-level transformation commands intentionally modify the semantics of the specification under development to achieve desired revisions. These semantics-modifying commands are referred to as evolutionary transformations. They explicitly formalize and carry out the evolution process which is an essential KBSA feature. Evolutionary transformations would typically be used to define new specifications, modify existing specifications, or merge existing specifications into larger ones.

When an evolutionary transformation is executed, it determines the impact of the change elsewhere in a specification and propagates further changes accordingly. HLTC's can thus assist in the process of integrating new specification components, such as those that might be retrieved from a library of cliches, into an existing specification. At present, over 100 high-level transformation commands exist.

An important effect of the user's applying HLTC's is that they formally capture specification evolution. Furthermore, typographical errors are less common because specification modification is automated. The real payoff for using HLTC's will come in the form of more mature application strategies surfacing earlier in the development cycle. Examples of new questions that can now be sensibly asked are: What does the developer want to do? Does a selected operation make sense within the current context? This work is just beginning.

In addition to incrementally refining and elaborating the specification under development, a user often wishes to incorporate previously developed specifications or portions of specifications. SA provides the user a package facility, called "views," that encapsulates specification components. Views are used both for reusing previously developed specifications and for focusing SA analysis tools.

View extraction is essential for removing unnecessary detail lingering from original definitions. It is accomplished when the user identifies critical specification definitions to be reused (with SA adding additional definitions discovered while tracing along dependency links). SA can also extract the transitive closure of any dependencies (e.g., retrieving all invoked procedures along with procedures that they, in turn, invoke).

Once a view has been extracted, it may be reused by merging it into a specification under development. Current merge capabilities are fairly simple. They can handle views containing no conflicting object definitions. However, new analysis tools are needed to deal with views containing conflicting definitions. In particular: When an object is unconstrained in one view and deterministically constrained in the other, under what condition can these constraints be reconciled when merged?

To simplify specifications, a focusing capability takes advantage of SA's view extraction facilities plus some HLTC's. Once a view has been defined it may be passed to any analysis tool. Such a tool will analyze the extracted view only. This allows the user to focus on small parts of the overall specification without being overwhelmed by the details of the full specification.

So far, only the sequential development and elaboration of specifications has been described. This deals effectively with a series of events, one occurring after another. But many times the sequential ordering is arbitrary--a function of when the developer happened to handle particular problems. In contrast, SA's parallel elaboration supports non-sequential specification development. In those cases where no interaction between parallel activities exists, the solutions to supporting parallel elaboration are trivial. However, when minor interactions are called for, SA can integrate parallel development activities.

SA achieves parallel elaboration by customarily saving the development histories of the HLTC's used to arrive at a specification elaboration associated with any sequential elaboration path. If a user wants to integrate multiple elaboration paths, SA retreats to the common starting point and tries to interleave each development history. User guidance is needed and, in some cases, certain HLTC's may have to be reapplied in light of development histories interaction.

During specification development, SA permits the user to add more information about the specification--information that can't be adequately expressed in GIST. Rather than revising GIST, SA supports an annotation language. One example of its use: annotating specification statements according to whether they are requirements or goals. The distinction is that requirements are inviolable constraints, whereas goals describe general behavior that need not always be accurate and, in some cases, may involve exceptional cases not currently covered by the goal. Therefore, HLTC's first look at specification statement annotations to see whether they denote requirements or goals. Requirements will only be edited by HLTC's that are meaning preserving so that requirements satisfaction is insured. Goals, on the other hand, might be "compromised" by HLTC's to handle exceptional cases. This of course implies that non-meaning-preserving transformations are allowed to operate on statements annotated as goals.

SA is built on top of an integrated support environment called the Specification Service. The Specification Service was, in turn, built on top of three tools previously developed at ISI: AP5, [5] ISI's wide-spectrum language; CLF, [4] an object base built on top of AP5; and POPART, [33] a tool for building and manipulating language parse trees given the language's BNF grammar definition (in this case, GIST). Within the SA effort, incompatibilities between these three tools were overcome so that all three could simultaneously work together on the same specification. SA also provides several integrated analysis tools for validating and debugging the specification. These include a symbolic evaluator, a behavior explainer, an influence graph generator, a GIST paraphraser, a static type analyzer and a resource analyzer.

Performance Assistant

The Performance Assistant [10,11,12,13] work began at Kestrel Institute in 1985. Technical issues addressed so far in the effort are (1) data structure selection (DSS) using symbolic and heuristic techniques, and (2) the development of PERFORMO, a functional specification language with set-theoretic data types. PERFORMO is similar to VAL, [27] developed at MIT, and SISAL, [28] developed at Lawrence Livermore Laboratory. PERFORMO is intended for DSS work, but is expressive enough to be a good initial specification language for further research on performance and implementation strategies.

Long term goals for a performance assistance are to guide software performance decisions at many levels in the software development cycle--from requirements specifications in very high-level programs to low-level code. The strategy is to combine heuristic, symbolic, and statistical approaches to provide capabilities for symbolic evaluation, data structure analysis and advice, and algorithm design analysis and advice. This effort is focusing on data structure selection and optimizations which include finite differencing, iterator inversion, flattening, operator elaboration, membership-test removal, loop fusion, and dead variable and copy elimination.

Briefly, finite differencing calculates a new expression-value within a loop by using the old expression-value in conjunction with a delta function (rather than recomputing the new expression-value from scratch). Iterator inversion is a special case of finite differencing that focuses on duplicate iterator expressions that can often be reformulated into a single iterator expression. Flattening, which is necessary for both finite differencing and iteration inversion, removes all nested function calls and provides names for all possible subexpressions. Operator elaboration transforms high-level operations into lower-level operations. Membership-test removal eliminates explicit membership tests when they can be inferred a priori. Loop fusion transforms multiple loops into a single loop to reduce overhead code and (sometimes) iteration requirements. Dead variable and copy elimination is done following automatic transformations, which often leave behind useless or redundant code.

PA's data-structure-selection strategy involves supplying refinement decisions to the implementation generator while reducing PERFORMO specifications to efficient Ada-class implementations. When PA needs a refinement decision, it determines which program properties are necessary to make a satisfactory selection. Properties include how a specific variable will be used and some its characteristics (like size and containment). Properties of a variable could include whether it is random access, ordered, enumerated, dynamic, or possibly empty. Based on such properties, specific implementation decisions can be made.

DSS's approach is to use basing as the representation methodology, when possible--particularly for complex data structures. Basing is a representation method used within the SETL data structure optimizer. Basing introduces a dual representation consisting of the object (data structure) and an accessor to the object. Multiple objects may share the same base. The base may also be some intermediate object upon which data access can be efficiently carried out. To determine the applicability of basing and what the appropriate base should be, the following analysis techniques are employed: Containment, 1-1, Bound, Sparseness, and Operation Analyses.

Framework

The KBSA Framework (KF) effort [17,18,26] brings a global perspective to KBSA. Initial work on KF began at Honeywell Systems and Research Center in early 1986. This effort seeks two goals: (1) to develop an integrated KBSA demonstration and (2) to propose the specification of a framework through which all facets must interact, communicate and eventually be built upon. The purpose of (1) is to provide a concept demonstration that would be intuitively obvious to the most casual observer. The purpose of (2) is to promote a tightly coupled interaction between facets. KF will provide a common reference for each facet developer and allow information sharing. KF interaction will be a requirement for all iteration-2 contracts. The future result will be a more tightly integrated KBSA.

The main technical challenges in this effort are to (1) define the minimum functionality that the framework must provide to all facets, (2) define a common interface for the facets to interact with each other, (3) extend the framework functionality to include a distributed environment, (4) support programming-in-the-large concepts like configuration control, traceability, and access control, and (5) provide consistent user interface services.

The first KF iteration will yield a demonstration of two independent facets tightly coupled to the KF. The facets may exist on a separate machine but will communicate via the KF. KF will be responsible for maintaining traceability between software development objects and keeping facets updated about the status of those objects. The two facets, on the other hand, will be responsible for establishing the initial traceability (e.g., the relationship between requirements objects and specification objects) via the use of services supported by KF.

The difficult issue within the KF has been to characterize what indeed constitutes a framework definition. At the conceptual level, where agreement exists between all developers, KF should have or support the following characteristics:

a. An object base.

b. Logical relations and inference mechanisms that map to the object base.

c. A distributed knowledge base.

d. Access control.

e. Configuration control.

f. Transactions (i.e., collecting related operations into a single transaction to prevent the recognition of temporary knowledge base inconsistencies resulting from intermediate operations)

g. Object permanence.

h. A flexible user interface.

No deeper-level descriptions for the KF have been agreed upon to date. However, such a description must become available under the second iteration phase.

Early in the facet-development efforts, the importance of a common example was recognized. For the KBSA Framework demonstration, the domain will be substantial in that the air traffic control (ATC) problem (also analyzed by Sanders Associates and covered in the RA and SA) will be addressed. Although the demonstration will not solve the ATC problem or focus on its full scope, the rich set of requirements inherent in ATC will serve as an excellent driver for KBSA. ATC involves a variety of real world issues such as real time requirements, data base management, user interaction, interaction with the outside world, and changing or not-too-well-defined requirements.

Planning and Plan Recognition

As part of the Northeast Artificial Intelligence Consortium, which is sponsored by RADC, the University of Massachusetts has conducted research in planning and plan recognition as it relates to the software development process. The primary role that this research will play in future KBSA will be in the areas of activities coordination and the "replaying" of developments.

A "plan" is a collection of partially ordered actions or operations that achieve one or more goals, given an initial state of the world. Each operator is tied to (a) one or more pre-conditions that define the state which must hold for the action to be legal, and (b) one or more post-conditions that define the state changes which result from performing the action.

"Planning" offers a mechanism for formalizing the software development process. It can be used to assist the programmer in achieving some end goal, to explain the current project status in terms of completed or partially completed plans, or to automatically trigger an action for which all of the pre-conditions are satisfied.

"Plan Recognition" uses plans to infer user end goals from the actions that are being performed. The plan recognizer is able to act as an automated apprentice that "looks over the user's shoulder," correcting simple programmer errors and suggesting alternative actions. This mechanism will enable the construction of user interfaces that reduce the complexity of determining what operations to apply in a given situation. It can also minimize errors that result from performing inappropriate operations.

Development Assistant

The Development Assistant (DA) effort began in November 88 by Kestrel Institute. This effort will define a formalism that captures the knowledge and decision making processes used by software programmers and designers. The effort's goal is a knowledge-based system that derives efficient, executable code from a completed specification, automating wherever possible (via automatic transformation), and capturing user supplied design decisions otherwise. DA's assistance will include providing search trees of possible implementations, and permitting the user-in-the-loop to fine-tune his implementation on the basis of various performance factors, maintenance considerations, code reusability, etc.

Requirements/Specification Assistant

This iteration-2 contractual effort was awarded in May 1989 to the Information Sciences Institute of the University of Southern California (who also subcontracted with Sanders Associates). The effort's mission is to join the activities of requirements analysis and program specification by using a common knowledge base. This will provide a smooth transition from informal user requirements to formal, low-level specifications, thus enabling greater tracing and requirements validation. This has grown out of the recognition that both early requirements validation and the need for informal methods to support specification evolution are important.

Activities Coordination Formalism Design

In large software projects (and similar cooperative-work programs), a major share of the effort is spent either on coordinating the activities of participants or on repairing the damage caused by poor communication and coordination. Either way, costs grow explosively as projects grow larger. This contract, awarded to Software Options, Inc. of Cambridge, MA in October 88, will design a view formalism for project communication (which, here, means any exchange of information between project participants or agencies that could reasonably be computer mediated, even if it is not handled electronically in present practice). Views in this formalism will allow project participants to see the project's established communication and coordination protocols without having to study the detailed process model.

Because of its specific focus on project communication and the enactable model derivation, work under this effort will complement the activity coordination and project management work already under way in KBSA. One of the main objectives is to show that activity coordination by communication management is a useful technique throughout the military systems acquisition process, from the pre-award stage in which an RFP is generated to fielded systems maintenance. Therefore, protocol sketch notation will be made accessible to a broad spectrum of project personnel. Another central objective is to enable managers and other non-programmers to play a direct role in the creation and customization of enactable process models, and in their evolution over a project's life. To that end, techniques will be developed to elaborate and refine protocol sketches into detailed process models. The design will show how tools could be provided to partially mechanize this derivation. However, human participation will still be needed as well, both to fill in details that are not represented in the sketch abstractions and to establish links with process model aspects that do not bear directly on communication.

KBSA Concept Demonstration

Targeted for contract award in the third quarter of 1989, the KBSA Concept Demonstration will first design then implement a "broad" and "shallow" software life-cycle processing system based on the KBSA paradigm. This system will be broad in the sense that it will demonstrate support for the entire system life-cycle, including both development and evolution. It will be shallow in the sense that it will provide less powerful assistance and functional completeness than will be required of eventual productized versions of a KBSA.

This effort will differ from the Framework effort in that the loose coupling and facet integration is not a goal. Rather, Concept Demo will provide essential experience with the KBSA paradigm as a whole, particularly the gathering of insights into requirements for an integrated high-to-medium-level interface suite, and high-to-medium-level process mediation and coordination. Furthermore, it will provide an additional vehicle for technology transfer to members of RADC's KBSA Technology Transfer Consortium, which was instituted in 1988 to promote KBSA technology transfer to industry. Therefore, the KBSA Concept Demonstration might be thought of as a vanguard for iteration 2 in the sense that it will attempt to give us insight into the software development process that takes us farther, still, from the waterfall paradigm and closer to the goals expressed in the KBSA vision. In particular, it should provide critical insight for formulation of the iteration-2 framework definition.

Standards and Technology Transfer

Throughout the KBSA program, efforts have been made to encourage the use of common conventions that will increase components reuse during KBSA development, and ensure a smooth technology transfer. Three communities have emerged that play a role in defining these conventions and serve as vehicles for transferring the technology:

1. KBSA developers -- Meetings of all the players that directly contribute technology to the program (government, industry, university) are held on a regular basis. This ensures coordination and participant awareness of new technologies and prototype components that might be shared.

2. KBSA Consortium -- RADC established the KBSA Technology Transfer Consortium to ensure a fertile base to transfer KBSA technology. The consortium, coordinated by RADC, is comprised of KBSA developers and a competition-derived set of "alpha" sites from industry, small business and academia. Each consortium member is committed to active technology transfer through reports and KBSA prototype evaluations, and the feedback of technical results from trial applications and studies. The actual technology transfer process as it relates to this consortium is as of much interest to RADC as the results because the operational agreement that participants signed up to is based upon synergy and perceived benefit rather than a legal arrangement or the exchange of funds for service.

3. External Community -- External KBSA program awareness is maintained through yearly KBSA Conferences that focus on KBSA and related knowledge-based technologies. Four such conferences have been held to date.

Within each of these technology transfer communities, the use and development of standard components has been emphasized. Each community has been a source of influence in developing "common" components that are referred to as "KBSA conventions."

To minimize effort duplication, the KBSA developers have been identifying and discussing common components that might be shared among two or more KBSA component developers. To date, standardization efforts have focused on user interface components, with the following three areas being discussed:

1. Editor standard -- Text manipulation is performed by the SA, KBRA, and the KF. A common editor convention is being defined based on GNU Emacs.

2. Natural Language Generation -- Explanation of the knowledge-based state is a necessary capability of several KBSA components. An initial natural language generation facility developed by one of the facet developers will be used as a basis for this convention.

3. Graphics toolkit -- The user interface to the KBSA will be highly graphical. Each of the existing facets and framework make extensive use of graphics and were developed using advanced workstations. A graphics toolkit based on the X-Windows package from MIT is now being proposed. The toolkit will provide a highly object oriented interface using many concepts originally defined in the Common Lisp User Environment (CLUE).

To facilitate technology transfer to consortium members, a limited set of hardware platforms have been identified for use as initial demonstration platforms. This set consists of: Sun 3, Symbolics 3600, TI Explorer II, VAX 11/780, and Macintosh II. These platforms have emerged from the initial platforms available to RADC and the current KBSA developers. The limited list is intended to reduce platform diversity and the number of hardware configurations needed to demonstrate the KBSA.

Currently, all KBSA developers have constructed systems using Common Lisp. With the inclusion of the Common Lisp Object System (CLOS) into the ANSII Common Lisp Standard, the KBSA developers are evaluating the CLOS adoption as a basis for future development. Currently, only the KF has been constructed using CLOS. Tentative plans call for CLOS to be used as well in future releases of the SA and the PMA.

Conclusion

Though significant progress is being and has been made in providing machine-in-the-loop assistance for specific life-cycle activities, we are still short of the technology needed for a true KBSA. In iteration 2, we will be focusing on more closely coupled assistants that do not fall neatly within the arbitrary life-cycle boundaries one sees in the waterfall model or between iteration-1 assistants. Instead, the second iteration will focus on continued development of a framework specification and assistants that are targeted at specific users--assistants that span the entire software life-cycle.

Acknowledgements

I extend my appreciation to Mr. David Harris of Sanders Associates, Mr. Steve Huseth of Honeywell, and Dr. Lewis Johnson of the University of Southern California Information Sciences Institute--all participating KBSA developers--who provided edifying comments and text to help insure that this paper is as accurate and up-to-date as possible. Special thanks go to Mr. Kevin Benner, a defacto collaborator in this paper. Kevin was an enthusiastic and tireless purveyor of the KBSA philosophy during his commission as a junior officer at RADC, and is responsible for gathering and organizing the bulk of the technical information found herein. He is now pursuing his Ph D at the University of Southern California and is employed part time by the Information Sciences Institute of USC.

References

1 Allen, J.F., "Maintaining Knowledge about Temporal Intervals," Comm. A.C.M.26(11), November 1983, 832-843.

2 Balzer, R. et al., "Knowledge-Based Specification Assistant," Interim Technical Report, RADC, Griffiss AFB, NY, Dec., 1986.

3 Cicarelli, E., "Presentation Based User Interfaces," TR 794, MIT Artificial Intelligence Lab, 1984.

4 CLF Project, "CLF Overview," USC/Information Sciences Institute, Mar, 1986.

5 Cohen, D., "AP5 Manual," User Manual, USC/Information Sciences Institute, Oct 19, 1987.

6 Czuchry, A. J. Jr., "Where's the Intelligence in the Intelligent Assistant for Requirements Analysis?," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.

7 Czuchry, A., Harris, D., "The Knowledge-Based Requirements Assistant: A New Paradigm for Requirements Engineering," IEEE Expert, Nov. 1988.

8 Gilham, L., "KBSA-PMA Program Specification," Documentation, RADC Contract F30602-84-C-0109.

9 Gilham, L., "KBSA-PMA User Manual," Documentation, RADC Contract F30602-84-C-0109.

10 Goldberg, A. and Smith, D., "Towards a Performance Assistant," Interim Technical Report, RADC, Griffiss AFB, NY, Nov., 1986.

11 Goldberg, A. and Smith, D., "Performance Estimation for a Knowledge-Based Software Assistant," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.

12 Goldberg, A., "Technical Issues for Performance Estimation," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.

13 Goldberg, A., et al, "KBSA Performance Estimation Assistant Program Specification," Documentation, RADC Contract F30602-86-C-0026.

14 Green, C. et al., "Report on a Knowledge-Based Software Assistant," RADC Tech. Report TR-83-195, RADC, Griffiss AFB, NY, Aug, 1983.

15 Harris, D., "A Hybrid Structured Object and Constraint Representation Language," Proceedings of National Conference on Artificial Intelligence, Aug. 1984.

16 Harris, D., Czuchry, A., "Knowledge Based Requirements Assistant," RADC Final Technical Report TR-88-205 (two volumes), October 1988.

17 Huseth, S., "Analysis of Knowledge-Based Frameworks, Interim Technical Report, RADC/COES, Griffiss AFB, NY, Nov, 1987.

18 Huseth, S. and King, T., "A Common Framework for Knowledge-Based Programming," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.

19 Huff, K., Lesser, V., "Plans and Meta-plans in an Intelligent Assistant for the Process of Programming", 2nd KBSA Conference, August 1987.

20 Huseth, Steve et al. (Honeywell, Inc.), "KBSA Framework (Phase I)," RADC Final Tech. Report, TR-88-204, October 1988.

21 Johnson, W. J., "Overview of the Knowledge-Based Specification Assistant," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.

22 Johnson, W. J., "Turning Ideas into Specifications," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.

23 Jullig, R., et al., "KBSA Project Management Assistant," Final Technical Report, RADC, Griffiss AFB, NY, July 1987, TR-87-78 (two volumes).

24 "Knowledge-Based Specification Assistant User Manual," Documentation, RADC Contract F30602-85-C-0221, June, 1988.

25 Ladkin, P., "Primitives and Units for Time Specification," AAAI-86, Philadelphia, PA, Aug. 11- 15, 1986.

26 Larson, A. and Huseth, S., "KBSA Common Framework Implementation," RADC, 2nd Annual KBSA Conference, Utica, NY, Aug 18-20, 1987.

27 McGraw, J. R., "The VAL Language: Description and Analysis," ACM Transactions on Programming Languages and Systems, Jan., 1982.

28 McGraw, J. R., et. al, "SISAL: Streams and Iteration in a Single Assignment Language," Technical Report M-146, Lawrence Livermore Laboratory, Mar., 1985.

29 Osterweil, L., "Software Processes Are Software Too," Proceedings, 9th Annual ICSE, Monterey, CA, 30 March - 2 April 1987.

30 Reasoning Systems, "REFINE User's Guide," June 15, 1986.

31 Sanders Associates, "Knowledge-Based Requirements Assistant," Final Technical Report, RADC Tech. Report, TR-88-205 (two volumes), Griffiss AFB, NY, Oct., 1988.

32 Waters, R., "KBEmacs: Where's the AI?," The AI Magazine, Spring, 1986.

33 Wile, D., "POPART: Producer of Parsers and Related Tools Systems Builders' Manual," USC/Information Sciences Institute, July 1987.