Project #: D-48-617  
Center #: 10/24-6-R6890-0A0
Contract#: DAC88-90-D-0006-0001  
Prime #:  
Subprojects ?: Y  
Main project #:  
Project unit: DEAN ARCH  
Project director(s): CIRCEO L JR

Sponsor/division names: ARMY  
Sponsor/division codes: 102

Award period: 900130 to 901130 (performance) 901130 (reports)

Sponsor amount  
Contract value  
11,074.00
Funded  
11,074.00
Cost sharing amount  

Does subcontracting plan apply ?: N

Title: DEVELOP SPECIFICATIONS FOR KNOWLEDGE WORKER SYSTEM, VERSION 1.0

PROJECT ADMINISTRATION DATA

OCA contact: William F. Brown 894-4820
Sponsor technical contact: MS. BEVERLY COSKUNOGLU (217)373-7284
US ARMY CONSTRUCT. ENGR. RES. LAB.  
2902 NEWMARK DR., P.O. BOX 4005  
CHAMPAIGN, IL 61824-4005

Security class (U,C,S,TS) : U
Defense priority rating : DO-C2
Equipment title vests with: Sponsor X  
ONR resident rep. is ACO (Y/N): N

Administrative comments - MOD. P00001 ADDS ADDITIONAL EFFORT, INCREASES THE PROJECT PRICE BY $11,074, AND EXTENDS THROUGH 11/30/90.
GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 12/31/91

Project No. D-48-617

Project Director CIRCEO L JR

Sponsor ARMY/CON ENG RES LAB, IL

Contract/Grant No. DACA88-90-D-0006-0001

Prime Contract No.

Title DEVELOP SPECIFICATIONS FOR KNOWLEDGE WORKER SYSTEM, VERSION 1.0

Effective Completion Date 901130 (Performance) 901130 (Reports)

Closeout Actions Required:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Y/N</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Invoice or Copy of Final Invoice</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Final Report of Inventions and/or Subcontracts</td>
<td>Y</td>
<td>910624</td>
</tr>
<tr>
<td>Government Property Inventory &amp; Related Certificate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classified Material Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Release and Assignment</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Comments

Subproject Under Main Project No.

Continues Project No.

Distribution Required:

<table>
<thead>
<tr>
<th>Role</th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Director</td>
<td>Y</td>
</tr>
<tr>
<td>Administrative Network Representative</td>
<td>Y</td>
</tr>
<tr>
<td>GTRI Accounting/Grants and Contracts</td>
<td>Y</td>
</tr>
<tr>
<td>Procurement/Supply Services</td>
<td>Y</td>
</tr>
<tr>
<td>Research Property Management</td>
<td>Y</td>
</tr>
<tr>
<td>Research Security Services</td>
<td>N</td>
</tr>
<tr>
<td>Reports Coordinator (OCA)</td>
<td>Y</td>
</tr>
<tr>
<td>GTRC</td>
<td>Y</td>
</tr>
<tr>
<td>Project File</td>
<td>Y</td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
</tr>
</tbody>
</table>

NOTE: Final Patent Questionnaire sent to PDPT.
GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT (SUBPROJECTS)

Closeout Notice Date 12/31/91

Project No. D-48-617
Center No. 10/24-6-R6890-0A0_ 

Project Director CIRCEO L JR_____
School/Lab DEAN ARCH_____

Sponsor ARMY/CON ENG RES LAB, IL_____

<table>
<thead>
<tr>
<th>Project #</th>
<th>DO #</th>
<th>Ctr #</th>
<th>Sponsor</th>
<th>Main proj #</th>
<th>Admin. Unit</th>
<th>Start</th>
<th>End</th>
<th>Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-20-609</td>
<td>DACA88-90-D-0006-000</td>
<td>10/24-6-R6890-0A1</td>
<td>ARMY/CON ENG RES LAB, IL</td>
<td>D-48-617</td>
<td>02.010.116 T</td>
<td>900130</td>
<td>900830</td>
<td>4,761.00</td>
</tr>
</tbody>
</table>

LEGEND
1. * indicates the project is a subproject.
2. I indicates the project is active and being updated.
3. A indicates the project is currently active.
4. T indicates the project has been terminated.
5. R indicates a terminated project that is being modified.
KNOWLEDGE MANAGER MONTHLY PROGRESS REPORT

Prepared Under Contract
DACA88-90-D-0006-0001

February, 1990

Submitted to:
Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared March 8, 1990 by:
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
Accomplishments

During this month George Olive attended a PAX developers meeting held in Washington DC. This meeting was sponsored by John Sheehey (CEMP-P). The purpose of this meeting was to discuss issues related to the developing applications for the PAX system.

Additionally, we delivered the our first report "Requirements for future versions of the Knowledge Worker System". This report summarized the results of the JAD conference held in November, 1989 and incorporated our lessons learned while developing the KWS prototype.

We also investigated a better method to get information from the Congressional Record into the KWS. John Sheehey had mentioned he was purchasing printed versions of the Congressional Record from a private company and scanning this material into a word processor. We have discovered that this material is available directly from the Government Printing Office (GPO) on magnetic tape for approximately $55 per day congress meets.

This month we receive a Compaq 386/20e from USACERL for use in testing the KWS software.

Goals for next month

During March Mike Jones will be starting to work on this project. His first task will be to review the work which was done for the prototype KWS and to learn Actor.

Additionally, we will deliver written specifications for the Hardware and Software required to use for producing Version 1 of KWS. The major problem anticipated in this effort will be finding database software that will run under windows.

Suggestions

We recommend getting the following software for evaluation of its usefulness to producing KWS.

- Knowledge Pro for Windows
- VistaComm
- VistaKit
- New Wave
- Windows 3.0
- Windows 3.0 dev kit

We also recommend getting one Congressional Record tape so we may investigate the impact this data would have on the general database design for KWS.
KNOWLEDGE MANAGER MONTHLY PROGRESS REPORT

Prepared Under Contract
DACA88-90-D-0006-0001

March 1990

Submitted to:
Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared April 3, 1990 by:
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
Accomplishments

During March, Michael Jones reviewed the prototype KWS functionality, and source code, so that he will be able to start programming on version one of KWS. Additionally, we reviewed our software options for implementing KWS version 1. It was our hope that Actor 2.0 running under Windows 3.0 would free enough memory to continue to develop KWS using Actor. However, we decided that the current unavailability of Windows 3.0 precludes its from consideration for use with KWS version 1. Furthermore, Actor 2.0 alone did not provide the required amount of free memory required. Therefore, we suggest Version 1 of KWS should be written in either C or Objective-C.

In the course of testing Actor 2.0 under Windows/386 we discovered an interesting problem. On the compaq 386/20e on loan to Ga. Tech, the prototype version of KWS would not run. However, on a compaq 386/20e configured identically at USACERL (except for the type of network card) the prototype did run. It was first thought that the difference in the network cards should make no difference since neither machine was using them, but when the network card was removed from the Ga. Tech machine KWS would run. Windows/386 (or Actor) appear to be sensitive to the mere presence of certain types of network cards on the machine bus. This sensitivity should be remembered when testing version 1 of KWS.

Goals for Next Month

We will start writing KWSv1 in C. Additionally, if we get Objective-C we will make some preliminary tests as to its fitness for KWSv1. The testing of Objective-C will run concurrently with the writing of KWSv1 in C. This will allow us to deliver KWSv1 at the earliest possible date should we decide not to write KWSv1 in Objective-C.

Both George Olive and Mike Jones will be attending the JAD follow-up workshop at Ft. Belvoir April 18,19. Additionally, we will start to explore the possibility of obtaining G++ (the Free Software Foundation version of C++), and the NIH C++ class libraries. These will be evaluated for fitness for use in versions of KWS beyond version 1.
KWS Prototype Development
Monthly Status Report
April 1990

Accomplishments

During the month of April Mike Jones and George Olive prepared for and attended at JAD workshop at Ft. Belvoir.

Plans for May

Install Xwindows software on the Sun 4.
Install Oracle on Sun 4, and install Vista-host software for testing.
Get Windows 3.0 for evaluation
Continue the conversion of KWS prototype to C.
Beverly,

Here is the status report. I still don’t have the compaq back, but when I get it I’ll reformat it in word perfect. Until then I hope this is ok.

------Cut

KWS Prototype Development
Monthly Status Report
May 1990

Accomplishments

During the month of May we accomplished the following:

Installed X-Windows software on the Sun 4 designated for evaluating Unix software.

Installed Oracle on the Sun 4, and installed the CDC vista-host host software. This will allow us to evaluate Vista-host as a potential database for KWS. Additionally, we received the PC version of Vista-host which is required to connect PCs to the database.

Received Windows 3.0 for evaluation.

The conversion of KWS source to the C language is continuing. This month the following sections of code were completed:
- KWS Logo Screen
- Event Scheduler (add, modify, delete)
- Task Scheduler (add, modify, delete)
- Subtask Scheduler (add, modify, delete)
- Subtask Steps (add, modify, delete)
- Todo List (add, modify, delete public subtasks)

Plans

Evaluate Gupta software and return loaned copy.

Continue work on C version of KWS.

Meet with USACERL personell to discuss future directions of KWS development.

Determine if Windows 3.0 caused the Compaq computer to break down.

Problems

The compaq computer loaned to Ga. Tech. has been in for repair and has not been fixed yet.
Beverly,

Here is the status report. I still don’t have the compaq back, but when I get it I’ll reformat it in word perfect. Until then I hope this is ok.

------Cut

KWS Prototype Development
Monthly Status Report
June 1990

Accomplishments

Evaluated Gupta software and return loaned copy.
Continued work on C version of KWS.
Met with USACERL personnel to discuss future directions of KWS development.
Determined Windows 3.0 did not cause the Compaq computer to break down. The Compaq computer was received from the repair shop at the end of June.

Plans

Start evaluating additional software for use in implementing KWS.
Concentrate on the database requirements for KWS, this effort should be concluded in August.
Start Neural-Net study.

Problems

The compaq computer loaned to Ga. Tech. broke down again, 3 days after receiving it from the repair shop.
KWS Prototype Development  
Monthly Status Report  
July 1990

Accomplishments

During the month of July, we began evaluating the various database options available for use with KWS Version 1. The options under consideration are, Gupta SQLbase, Microsoft SQL server, FOCUS, Oracle, and Sybase.

Plans for August

Finalize design of KWS Version 1. At the end of August we will present our final design for KWS to John Sheehy, and USACERL personnel in Washington DC.

Additionally, during August, George Olive will attend a Windows 3.0 conference in Boston.
KNOWLEDGE WORKER SYSTEM
MONTHLY STATUS REPORT
August, 1990

Prepared Under Contract
DACA88-90-D-0006-0001

Submitted to:
Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared November 2, 1990 by:
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
KWS Prototype Development  
Monthly Status Report  
August 1990

Accomplishments

During the month of August, George Olive attended a Windows conference in Boston. Additionally, he presented the file KWS Version 1 design to USACERL personnel in Washington, DC. Unfortunately, John Sheehy was sick and unable to attend the briefing.

Plans for September

During September, we George Olive will travel to Washington, DC to brief John Sheehy.

Pending acceptance, Programming will begin on KWS Version 1.
Accomplishments

During September, we George Olive travelled to Washington, DC to brief John Sheehy on the design for KWS Version 1. John expressed concern over the use of a local area network as the central communications scheme in KWS. After some discussion, he authorized us to proceed with the plan as it was presented.

John Sheehy asked Jerry Goudelocke to comment on CEMP's current network setup. Jerry said that it was an unacceptable system.

While in Washington, it was decided by USA-CERL that Georgia Tech should send a person to review the CEMP local area network.

Plans for October

Code KWS Version 1, using Gupta Technologies, SQLbase.

Travel to Washington DC to review CEMP lan.

Problems

The Gutpa Gateway to Oracle turns out to be in beta test, and is not available as a product.
KWS Product Development  
Monthly Status Report  
October, 1990

Accomplishments

During October, Beverly Thomas came to Georgia Tech to discuss the project. In attendance were Beverly Thomas from CERL and George Olive, Mike Jones, and Rich Erwin from Georgia Tech. A schedule of development and deliverables was discussed and agreed upon. The current status of the KWS software was discussed. Three trips to Washington, D.C. were planned for George Olive.

Reports describing the project milestones for three projects were created and delivered. Reports describing hardware platforms, the software testing plan, the short range development plan for the project, and a specifications checklist were prepared and delivered.

- George Olive travelled to Washington and evaluated CEMP’s local area network.
- KWS version 1.0 coding continued in C, accessing Gupta Technologies’ SQLbase.
- Sample data from McClendon Automation was received for testing purposes.
- Gupta Technologies’ SQLRouter/Oracle software for gateway communication to Oracle SQL*Net was received.

Plans for November

- Travel to Washington, D.C. for executive meetings.
- Finish initial coding of KWS, incorporating remote DB communications.
- Start to write the users’ manual and technical guide.

Problems

The SQLRouter/Oracle does not work. For some reason, it will not allow connection to the remote database server. The error message presented by Windows 3.0 states that this is a memory problem. We are in contact with Gupta Technologies and are trying to fix the problem.
REQUIREMENTS FOR FUTURE VERSIONS OF THE KNOWLEDGE WORKER SYSTEM

Prepared Under Contract
DACA88-90-D-0006-0001

Task 1-1

Submitted to:

Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared February 22, 1990 by:
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
Overview

During the previous year, a prototype of the Knowledge Worker System (KWS) was developed by Georgia Tech, working under the direction of USACERL. This prototype was presented to the KWS User Group in a Joint Application Development (JAD) workshop at the Ft. Belvoir Fusion Center in November 1989.

The purpose of the workshop was to bring system developers and the targeted users together in order to develop a common understanding of how the Knowledge Worker System should operate. The prototype was intended to model one possible implementation of KWS and to provide a starting point for discussion about system design.

This paper will summarize the results of the JAD workshop as well as incorporate lessons learned during the development of the KWS prototype. The discussions in the JAD workshop focused primarily on what features to include in KWS and the user’s interaction with KWS. The lessons learned section primarily includes technical considerations that emerged while developing the KWS program.

Both the technical and user concerns will be organized into three categories: very short term, short term, and long term. The very short term category includes those features/modifications that can be made to the KWS prototype with minimal changes to the existing code and can be implemented in less than 6 months. The short term category includes features/modifications that can be incorporated if the prototype code were rewritten and covers the period of 6 months to 1 year from now. The long term category covers all the features/modifications that appear to be very difficult to implement and will take more than a year to incorporate.
The JAD Workshop

The focus of the discussions in the JAD workshop focused on the users' point of view of how KWS should work. Discussion centered on the user interface, the functionality required from the system in order to assist knowledge workers (KW) in their daily work ["required functionality"] and future KWS capabilities that will facilitate knowledge workers ["nice-to-have functionality"]. The resulting requirements from the JAD workshop are summarized in Appendix I. The requirements are stated as questions and have been reorganized into major categories.

The categories into which user comments seem to naturally divide include:

<table>
<thead>
<tr>
<th>Category</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>How the KW can receive help both in the form of help messages generated by the system and via the organization tasked to support KWS.</td>
</tr>
<tr>
<td>Communications</td>
<td>How the KW can access data stored somewhere other than on his own personal computer.</td>
</tr>
<tr>
<td>Data Organization</td>
<td>How the KW can access data in KWS that does not match the normal structure of KWS data; how the KW can keep track of the timeliness of data.</td>
</tr>
<tr>
<td>Automatic Execution</td>
<td>What tasks KWS can/should perform for the user.</td>
</tr>
<tr>
<td>Scheduling</td>
<td>All issues that affect how the tasks that appear on the ToDo lists can be manipulated.</td>
</tr>
<tr>
<td>The User Interface</td>
<td>All issues that affect how the user interacts with KWS.</td>
</tr>
</tbody>
</table>

HELP

The two major features that must be provided via the KWS help facility are on-line help and the means to contact a support organization for help with problems that require more assistance than on-line help can provide.

The on-line help should, at a minimum, be context sensitive; that is, HELP should produce a message pertinent to the activity being attempted by the user when she selects HELP. Additionally, in the longer term, HELP should provide templates for filling in information about steps, subtasks, or forms, etc. In the long term, on-line HELP should provide an artificial intelligence facility that can review what the KW was attempting to do and, based upon information in the KWS knowledge base, suggest a solution.

As a fall-back position, the users requested that KWS provide the means to ask a human expert how to solve the problem they had encountered. This problem-solving HELP may be as simple as providing telephone support; it may require that KWS automatically prepare an electronic submission of the request for additional assistance.

Communications

This category covers a broad range of features. Users stated that Knowledge Worker System should provide the capability to: access other computers; send messages to other KWS; and enter data from paper copies.

Access to other computers encompasses simple terminal sessions, file transfers, and database access.
Users require terminal sessions and file transfer capability in the very short term. Database access should ultimately include the ability to search a database on another computer as easily as if the database were on the local computer. The computers KWS must access were divided into two categories by users: those used by knowledge workers (primarily PAX) and those used by other agencies.

Users expressed a great need for electronic messaging capability, including both notification and electronic mail. Notification entails informing the user when a remote job is completed or when an important task is added to the ToDo list. Electronic mail is more involved; this capability will allow the KW to distribute reports to non-KWS knowledge workers and computers. In either case the messaging system must not be intrusive; that is, system-generated notifications must not interfere with any work the KW is doing.

The final communications ability requested by users is the means to access KWS from a remote location. This access ranges in complexity from the ability to use any computer, anywhere and obtain full access to their entire system to the simple capability to get important messages from a remote site.

Data Organization

The KWS prototype presented data to the user in a rigid hierarchical organization. The top end of the hierarchy consists of an event; the base of the hierarchy includes the documents and programs required to accomplish one step of an event. Users require a data organization that allows the accessing and structuring of data in a more flexible manner. Both Schedule data, as well as non-Schedule data (documents and programs), must provide this flexibility. This section of the document will deal only with non-Schedule data; Schedule data will be discussed in more detail under a separate Scheduling section.

Areas of concern include: how to keep track of various versions of a document; how to find a document or program when it is not attached to the step the user is currently accessing; information archival; and how users can determine if it is safe to delete a specific document or program.

The KWS prototype provides the ability to keep several versions of a document, however, version control is completely manual. Each step can have multiple attachments and the user can attach more than one version of a document to a step. Users requested that this process be automated so that KWS tracks document versions, rolling back all previous versions whenever a new version is generated.

The capability to search the entire KWS knowledge base for a relevant document is a feature strongly requested by users. In the short term, this feature could consist of a simple key word search of all documents. In the long term, the system should incorporate an intelligent algorithm that determines the similarity of a current task to other tasks in the system. This capability should also be generalized so that it can assist a knowledge worker in locating any procedure (to perform an ad hoc task, e.g.) that another user may have already entered into the knowledge base.

In order to keep the on-line storage of the knowledge base to a reasonable size, there must be a method to archive older data. Furthermore, KWS must automatically keep track of the request to archive data so that any data in use will remain in the knowledge base until the last user archives it.

Automatic Execution

One ultimate goal of the KWS is to perform the majority of repetitive tasks that a knowledge worker must do each day. Areas for automation as suggested by users included: automating the daily verification of the contents of PAX databases; automating the extraction and subsequent updates of PAX
databases; automating the process of entering the steps involved in performing a subtask; and automating the production of documents. In the long term, virtually every task the knowledge worker performs should be automated to some degree.

Scheduling

Scheduling issues that were discussed in the JAD workshop can be divided into three major categories: Supervisory issues - project scheduling and project tracking; User issues - project information, daily time management, job information; Issues that concern both supervisors and users - privacy and the tracking of ad hoc tasks. It was also stressed that a KW may be a supervisor when working on one task and a user at other times.

Several useful project scheduling features were discussed as critical success factors for the Scheduler. The software must make it easy to add scheduled items to the schedule. It must be easy to reassign tasks to different KWs, both temporarily and permanently. It should allow the scheduling of TDYs and vacations.

The scheduling system of KWS should also provide common project tracking information: what is late? where should resources be applied? It must also provide the information a supervisor requires to fill out manpower forms, e.g., the ability to track work by functional area or by duration; status information for all tasks currently being worked on. The ability to track a project should also allow a project to be tracked across divisions.

The type of information KWS must provide a non-supervisory user focused more on the ToDo list. KWS must provide information telling how long a task is likely to take. KWS should also allow the user to inquire about parts of a task on which the user is not working on and answer questions such as: Is this task late? am I impacting someone else? who should I coordinate with before rescheduling? do I have all the necessary information to start a task? Users stressed that KWS should flag any updates to the ToDo list so these changes can be easily noticed by the KW.

The set of issues common to both classes of users involved the following: the privacy of data, job information, and the handling of ad hoc requests.

The user group felt that it was important to be able to designate parts of the schedule as private to varying degrees (only the individual, only others in the group, everybody). Furthermore, the ability to change a due date should be similarly restricted.

Another useful function requested by the user group was the ability to summarize a knowledge worker job description. The events, task and subtasks for which a knowledge worker is the proponent should be listed in the system-generated job description. As well, the system should identify any other duties assigned to the knowledge worker.

The user group discussed how the Scheduler should handle ad hoc tasks. The knowledge worker should be able to add an ad hoc task to the ToDo list. Furthermore, KWS should help locate any information in the knowledge base which may be useful to an individual knowledge worker performing the task. Finally, KWS should be able to promote an ad hoc task to the master schedule should the KW decide that was appropriate.

The User Interface

Due to a lack of time, general discussion of the KWS user interface was cut short. The windowing approach was considered to be appropriate, however, several improvements were suggested.
Users were very interested in being able to view the schedule data in a variety of different ways. Particularly, the user group felt that windows should be provided to allow the data to be organized by week, month, and quarter.

Additionally, users requested a provision for graphical views of the KWS data. Users asked for two views: A view that shows the hierarchy of the KWS knowledge base from event down to the documents, along with the linkages which define their relationships; and, a view that shows the schedule information in a traditional graphic format, e.g., a Gantt chart.

In addition to graphical views of schedule data, the users need the capability of linking graphic information into KWS documents. The kind of information a user might include could be charts or scanned images.

A request was made for a way to define short cuts that allow a user to jump from one position in the KWS knowledge base to any other, without traversing the intervening positions.
Technical Considerations

Four major areas where technical changes could improve the Knowledge Worker System are the 1) addition of an underlying database; 2) generalization of the hypertext structure; 3) porting to a compiled language; and 4) addition of a KWS programming language. Of these areas, the addition of an underlying database is essential to the further development of KWS; the generalization of the hypertext structure will provide an enormous gain in usefulness.

Adding a Database

The nature of the KWS scheduling requirements demands that the schedule data be available to all knowledge workers simultaneously. While the schedule data has a degree of locality, any change that a knowledge worker (KW) can make to the master schedule will affect other users. Furthermore, some of the requested improvements suggested by the user group require KWS to quickly access large sections of the schedule data. For instance, the request to access the schedule data in some manner other than in the hierarchical way in which it will be entered will require KWS to form a new view of the data on the fly. There are a number of data base programs already on the market that offer this capability.

This database must be accessible directly to KWS as a library of subroutine calls. Some of the desired AI capabilities will be precluded if the data can not be accessed in such a manner. One type of AI program that would be essentially impossible to write without such a database is one that requires the use of the master schedule as one of its knowledge bases/rule resolution.

The database chosen to support KWS must be a completely distributed database so that 1) the data can be placed nearest to the group of workers that use it the most, 2) all knowledge workers can the KWS data. Use of a distributed database will generate a savings in communications costs; the access time will be minimized as well.

Generalizing the Hypertext Structure

The current hypertext structure of KWS includes certain assumptions about how the user will want to organize the KWS knowledge base. A task can only be linked to a subtask; a subtask to a step; and only steps can be linked to documents and programs. This restriction is not necessary.

Assuming the addition of an underlying database, a more generalized form of hyperlink can be defined. This hyperlink can link any item in the knowledge base to any other item(s) in the knowledge base. This capability will require a change to the user interface to allow a user to request specific linkages. While this capability can not be made in the very short term, it should be considered for the short term.

The generalization of hypertext links will allow the users to organize KWS data more closely to their current way of thinking rather than changing their thinking to follow the organization of the data. One user requested design change (the ability to add notes any place in KWS data) would be possible with a more general hypertext structure.

Porting KWS to a Compiled Language

So far Actor has proven to be a good language choice for the original implementation of KWS. Its interactive development environment has allowed the development process to proceed quickly and allows changes to be made easily. Additionally, new releases of Actor are becoming available in time to remain useful for KWS development.
However, there will come a time when KWS development begins to stabilize. In the long term the design will cease to change as quickly as it has in these initial stages. At this time, the advantages in execution speed and program size will start to favor a compiled language over an interpreted language such as Actor.

During this transition, it will still be desirable to use an object-oriented language. One advantage of this object-oriented approach is the production of a well-structured program. In the long-run, a well-structured program will be easier to maintain than a poorly-structured one.

KWS Programming Language

In order to allow the user the greatest amount of control over how the KWS system is used, a KWS programming language is required. This language will allow users to directly (i.e., to the system) specify how to perform tasks. In this way the users themselves can shape future versions of KWS.

The KWS programming language should allow the KW to name objects in the KWS knowledge base either by name or by attribute. It also must allow the user to create and destroy KWS objects. It must allow the user to form new relations between KWS objects or perform modifications to them. This language should also allow the user to define new generalized relationships which KWS will remember and allow other knowledge workers to apply to their data.

This language must also allow the user to create new window types by simply describing the window and the relation used to put objects into it.

Throughout this process KWS should "watch over the users shoulder" to ensure that the user does not define invalid relationships within the scope of the knowledge base.

This language is a feature for the long term to very long term development of the Knowledge Worker System. However, it will be the first step in allowing knowledge workers to specify enough information about a process so that KWS can fully automate an entire task.
Summary

The following table summarizes future enhancements to KWS according to category and time frame.

Key:  
P indicates partial implementation  
C indicates complete implementation

<table>
<thead>
<tr>
<th>Category</th>
<th>very short</th>
<th>short</th>
<th>long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context sensitive</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electronically submit request to person</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide templates</td>
<td>P C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalized AI</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communications</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminal sessions</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File transfer</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Access (simple)</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Access (complex)</td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electronic notification</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electronic mail</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote access (messages)</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote access (complete)</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Organization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version control</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document search by key word</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document search by similarity</td>
<td>P C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Archive</td>
<td>P C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>very short</td>
<td>short</td>
<td>long</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td><strong>Automatic Execution</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start another program</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAX Reports¹</td>
<td>P</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Entering KWS data</td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>Production of documents</td>
<td>P</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td><strong>Scheduling²</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>User Interface</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allow schedule data to be viewed by week, month, etc.</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphical view KWS hierarchy</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphical view schedule data</td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>&quot;hot keys&quot;</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Technical Changes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributed database</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalized hypertext</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port to compiled language</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KWS programming language</td>
<td></td>
<td></td>
<td>C</td>
</tr>
</tbody>
</table>

Notes:

1. This work is currently being performed by McClendon Automation
2. These tasks can not be specified until after a second JAD conference scheduled for early spring 1990.
### Appendix I - Results of JAD Workshop

See the **Functional Design Requirements, Question/Decision/Product (QDP) section, p. III-1 through III-66** for more information.

**General**

**Communications**

<table>
<thead>
<tr>
<th>Category</th>
<th>QDP Num</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>46</td>
<td>How will the knowledge worker get help from the KWS Support Office for using the system itself (on-line, context-sensitive)?</td>
</tr>
<tr>
<td>Communications</td>
<td>28</td>
<td>Can this information or report be reformatted and distributed to others?</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>Can the KWS scan, digitize and broadcast messages or documents (posting news of major events)? To everyone? To specific people to trigger tasks?</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>Should the KWS provide access to other systems, agencies, private industry (construction specs)? What are they?</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>Can I access the system from outside the office? Answer messages?</td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>Can the system keep messages and notify me in an intelligent way without interrupting my work? Can I pull mail from other existing mail systems?</td>
</tr>
<tr>
<td>Category / Viewing</td>
<td>QDP Num</td>
<td>Question</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>Data Association</td>
<td>2</td>
<td>How do I determine whether subtasks, steps, attachments or automated tools are outdated? Can I search through steps to determine if changes have to be made? (e.g., Search all references to ARI and tell me what steps have to be modified, addressed and/or reaffirmed.)</td>
</tr>
<tr>
<td>Data Integrity</td>
<td>3</td>
<td>How can I keep track of the various versions of a product?</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>How do I know if it's OK to delete an item?</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>How do I archive old information so it can be retrieved later?</td>
</tr>
<tr>
<td>Automatic Execution</td>
<td>16</td>
<td>What are the subtasks, steps, or automated aides that are related to a subtask? Can I find them? How can I find them?</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>What information is related to this ad-hoc subtask or task? How do I find it?</td>
</tr>
<tr>
<td></td>
<td>39</td>
<td>How do I know that the system has a task with steps to support routine and ad-hoc requests? (Capability similar to a register of routine versus calendar driven.)</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>Can I search documents regardless of task association? Can the system make new associations automatically with confirmation (artificial intelligence)?</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>Can the system facilitate or trigger analysis data integrity (versus today's manual checks and re-input into other data bases)?</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>Can the system assist in preparing the steps for similar subtasks? (e.g., a tool box of templates/Dolts for generic tasks.)</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>Can the system automate the preparation of documents and forms with the extraction of data from other attachments?</td>
</tr>
</tbody>
</table>

Appendix I - 2
<table>
<thead>
<tr>
<th>Category</th>
<th>QDP</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Scheduling</td>
<td>6</td>
<td>How do you add new events?</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Can the supervisor easily and efficiently reassign subtasks or assign ad-hoc tasks to different knowledge workers? Can knowledge workers reassign tasks and subtasks for both temporary and permanent reassignments?</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>Can we plan schedules for leave and TDY's?</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>What's the impact of external changes on the schedule?</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>Can the system generate/support performance reviews, reports, charts and briefing materials (e.g., DoIt function) ?</td>
</tr>
<tr>
<td></td>
<td>42</td>
<td>Does the system prioritize my work? Of what I have to do today, what has the highest priority? By critical path? By priority? By value of product?</td>
</tr>
<tr>
<td>Project Reporting / Tracking</td>
<td>10</td>
<td>What items are now critical? To what do I have to devote most of my resources or it will be late?</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>How does the system help me determine whether I'm staffed for a new task or subtask?</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>What tasks can, or must, I reschedule?</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>Can we track work by functional area? By duration time?</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>How can I flag an item as submitted but not approved (i.e., items that aren't really finished until approved)?</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>How do I reverse a completed task? How do I change task status and amount completed? How do I notify others effected?</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>Can the KWS track tasks and subtasks in-house between divisions?</td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>What is currently being worked on and its status? How much is completed? What is the status of a subtask? Supervisor information?</td>
</tr>
<tr>
<td>Project Information</td>
<td>8</td>
<td>How long will it take to do this subtask? How long did it take in previous years? How long did it actually take versus how long I said it would take? What is the normal time duration? How much slack time is incorporated? When will a subtask become critical?</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>What new items have been added, deleted or modified in my TODO list since my last inspection?</td>
</tr>
<tr>
<td>Number</td>
<td>Task Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>What is the status of a subtask that's been assigned to another knowledge worker (that is a part of my task)? Is the status current? When was the last status update?</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Who are the knowledge workers (and others) with whom I must coordinate before rescheduling?</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Do I have all the dependent subtasks to start my subtasks?</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Who is dependent upon the completion of my subtasks? What is the impact on them if I delay the start? Can I decide to notify them electronically if I choose to delay?</td>
<td></td>
</tr>
<tr>
<td>Privacy / Control</td>
<td>What part of my TODO list is private? Who can see my scheduling information? (Private, for my supervisor, co-workers, public.)</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Who should be allowed to make schedule changes? How does the schedule get changed? Who controls it?</td>
<td></td>
</tr>
<tr>
<td>Daily Time Management</td>
<td>What tasks and subtasks must I finish today? Start today? When should they be completed?</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>What is the impact of my not being available in the office? (i.e., sick, unexpected meeting)</td>
<td></td>
</tr>
<tr>
<td>Job Information</td>
<td>For what functions, events, tasks and subtasks am I proponent? What are other duties and responsibilities?</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>How do I adjust my schedule to respond to this ad-hoc request?</td>
<td></td>
</tr>
<tr>
<td>Ad-Hoc tasks</td>
<td>How do I handle on-going tasks (no end dates or routine tasks)?</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Can I easily convert an ad-hoc task to a routine task? Can I add master tasks on an ad-hoc basis?</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SOFTWARE OPTIONS FOR FUTURE VERSIONS OF KWS

Prepared Under Contract
DACA88-90-D-0006-0001

Task 1-2

Submitted to:
Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared April 25, 1990 by:
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
Introduction

This report will discuss the various options for extending the prototype version of KWS; both in the short term, i.e. to include the amount of features necessary to make KWS usable as quickly as possible; and, in the long term, to produce the final version of KWS.

The options discussed will be based on the use of one of three different operating systems: Microsoft/IBM DOS (DOS); Microsoft OS/2 (OS/2); and some version of Unix\(^1\). DOS is considered because it is the operating system the KWS prototype is currently running under. OS/2 the operating system designed to replace DOS at some time in the distant future. Unix is an operating system in wide use on both large and small computers. These three operating systems are the only ones that are in wide enough use to be considered for use in future versions of KWS.

The choice of a Graphical User Interface (GUI) determines what operating systems, languages, development tools, and applications are available to the user. Therefore, the GUI will be the major criteria considered; and the languages, tools and applications will be discussed within the context of the chosen GUI.

\(^1\) Unix is a registered trademark of AT & T.
Graphical User Interfaces

The choice of which graphical user interface (GUI) to use for future versions of KWS is largely determined the choice of operating system chosen to implement KWS. Table 1. shows the graphical user interfaces considered in this sections and the operating systems they can run under.

<table>
<thead>
<tr>
<th>OS</th>
<th>GUI</th>
<th>DOS</th>
<th>OS/2</th>
<th>A/UX</th>
<th>Unix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation Manager</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MultiFinder</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X Windows</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Smalltalk</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 1. - Graphical User Interfaces and Associated Operating Systems.

Windows

Microsoft Windows is the GUI which runs under MS DOS. The Windows/386 version allows multitasking and multiple DOS windows. Windows is a large application running under the already cramped memory constraints of DOS, therefore, it severely limits the maximum amount of memory an application can use. Microsoft claims this memory limit will be removed when they release the new version of Windows (version 3.0).

There is a growing number of applications available for MS Windows.

Presentation Manager

Presentations Manager (PM) runs under OS/2. Because OS/2 is a multitasking operating system, all applications running under presentation manager can be fully multitasking. Additionally, Presentation Manager programs are limited only by the size of OS/2 programs which is 16MB. Currently, OS/2 allows only one DOS program to run along with the OS/2 programs, and the user must switch screens away from the PM to see the DOS program.

There are not very many applications available which run under Presentation Manager. Furthermore, the inability to
run multiple DOS applications limits the users choices considerably.

**MultiFinder**

MultiFinder a multitasking operating system which runs on the Apple Macintosh. It provides a completely integrated graphical user interface. MultiFinder Applications are force to a very rigid standard for interfacing to the operations system. This standard forces applications to have a very consistent look and feel to the user.

The latest release of Apple's version of unix (A/UX) can run multiple MultiFinder applications concurrently with unix applications.

There is a very large number of applications which can run under MultiFinder.

**X Windows**

X Windows by itself is not a GUI, it is merely a standard for applications to describe windows on a terminal. This allows X/Window applications to run on both workstations, and remote terminals (as long as they support the X Windows protocol).

There are additional libraries, called Widgets, which provide the additional functionality expected in a GUI. There are two competing positions for the set of standard Widgets for providing the GUI. These are OpenLook, and Motif.

X Windows is particularly interesting because it runs on a large number of platforms. As table 1. shows the only Operating system it is not currently running on is OS/2, but it can be reasonable expected that an OS/2 implementation of X Windows will be available soon. This makes X Windows a good choice when considering the issues of portability.

**Smalltalk**

Smalltalk is an entire programming environment. It provides a GUI, tools to manipulate the environment, and defines a language to program with. This means that any program running inside the smalltalk environment will be able to interact seamlessly, programs running outside smalltalk are not under the control of the smalltalk programs.

Smalltalk is an interpreter. This means programs written under smalltalk will run more slowly than similar programs written in a compiled language.
The fact that Smalltalk is available on all the platforms being considered makes it very attractive; the amount of memory it requires, runtime overhead it extracts, and its detachment from the operating system makes it a poor choice for a production system.
Languages

The language chosen to implement a program is judged by to factors. It must be able to manipulate the environment the program is trying to control, in this case the Graphical User Interface and the operation system of the computer. Also, it should make encoding the actions on the environment as easy and clear as possible.

The common language for manipulating all the GUIs discussed so far is C. This language supports the first requirement. There is, however, a set of languages called object-oriented languages which better support the second requirement. Two of these, Objective-C and C++, are translated to C as an intermediate step in the compiling process, and can therefore be considered for use in writing KWS.

C

The C language is the common denominator in all of the GUIs discussed. It is also the most basic of the languages, and would require the largest amount of source code. The large amount of source code would increase the effort required to maintain the code, as well as the amount of effort to rewrite the program to work with another GUI.

Both the memory and execution speed problems could be solved by writing version 1 of KWS in the C language. This transition would work even without upgrading to the Windows 3.0 environment.

C++

C++ is an object-oriented version of C. It would be reasonable to expect a C++ program to require more memory than a C program, but the execution speed should be comparable.

Because C++ is an object-oriented language, programs written in it should tend to be more structured. This should increase the programs portability to other operating system, as well as improve the maintainability of the code.

There are public domain libraries which provide a large amount of the code required to produce X Windows. Additionally, there is commercially available libraries to aid in the development of applications for the other GUIs.

There are numerous vendors selling C++ compilers. This assures the language will be well supported.
Objective-C

Objective-C is a more recent language than C++. However, it is supported on a large number of platforms. Additionally, there subroutine libraries available for Objective-C to support the Graphical User Interfaces discussed (particularly, X Windows).
Operating System

The prototype version of KWS is written in the Actor language and runs under Microsoft Windows (a graphical user interface (GUI) for DOS.) This choice allowed rapid development of the prototype KWS, but has several problems which make it undesirable as an actual production version of KWS. These problems are: we ran out of memory; and, the prototype KWS ran slowly.

DOS

Both of the problems encountered in the prototype are inherent in DOS itself, but were also aggravated by the additional overhead required by Windows and Actor. The latest version of MS Windows (version 3.0) was not available before writing this paper, however, Microsoft claims that version 3.0 will help greatly with the memory problems encountered using version 2.1.

A newer version of Actor (version 2.0) was tested to see if it would reduce the memory requirements. Version 2.0 of Actor did in fact use less memory. Unfortunately, running with Windows 2.1, lack of memory was still a problem. This problem might be lessened if Actor 2.0 is used in conjunction with Windows 3.0.

OS/2

OS/2 is a multitasking, single user operating system, which is designed to work in a networked environment. It runs only on 286 and 386 based microprocessors. It was designed to provide DOS users with an upgrade path for large applications. OS/2 requires a large amount of memory, 4MB is recommended just to run the operating system and presentation manager.

Currently, OS/2 allows only one DOS program at a time to run. Furthermore, this program will not execute while it is switched to the background. Like many of the Microsoft products, the next version (it is promised) will fix this problem.

There are only a limited number of OS/2 applications available. And, while support by vendors is growing for OS/2 it appears this dearth of applications will not be alleviated soon.

Unix

Unix is a multiuser, multitasking operating system originally developed at AT&T Bell Laboratories. It is currently marketed by a large number of vendors. This
assures that the operating system will evolve with technology, and be well supported for years to come.

Unix runs on a huge number of machines, from microprocessors to mainframes. Furthermore, it allows different machines all running unix to network seamlessly.

The major problem with unix is, there are several different versions which are not entirely compatible with each other. These problems are being addressed by government, industry and professional organizations to develop a standard which will insure compatibility both with source code and connectivity.

Some versions of unix, especially those running on 386 machines, allow users to run multiple DOS programs as concurrently with unix programs.
Tools

There are two types of software tools which will be useful in developing future versions of KWS, programmer productivity tools and program support tools.

Programmer productivity tools include:

**Browsers**, programs which allow a programmer to view and edit object-oriented source code from within the context of the class hierarchy.

**Interface Builders**, programs which aide the programmer in designing the user interface.

**Profilers**, programs which allow the programmer to determine where a program is spending most of its time, so these sections of the program can be concentrated on for optimization.

**Debuggers**, programs to aid in debugging the code.

Program support tools include:

**Database Servers**, KWS will require a program which can keep track of the KWS information and manage the sharing of this information among the various Knowledge Workers.

**Hypertext Servers**, As the number of documents and other information increases, KWS will need external support to keep track of this information and its interrelationship.
Conclusions and Recommendations

In the near term, we must continue to support the applications currently being used by the targeted knowledge workers. This implies we must, for the time being, stick with DOS. We should move away from DOS as the KWS platform as quickly as possible because of the constraints it places on what KWS can do.

In the long term we should target KWS to the largest number of systems practical. This eliminates Multifinder and OS/2 because the proprietary nature of the Operating System limits what hardware a user must have.

This leaves unix as the operating system of choice, and mandates X Windows as the GUI. This will allow kws to run on; DOS machines as a terminal (with an X Windows package installed) to a KWS server, mackintoshes (with A/UX), many unix workstations, mini- and main-frame computers.

For the first production version of KWS we should:

- Continue to support microsoft windows;
- Develop in C or one of its object-oriented descendants;
- Use a database external to the KWS program.

For a version later then the first version, we should:

- Move to X Windows;
- Develop in an object-oriented language;
- Target Unix as the supporting operating system;
- Incorporate database and hypertext servers.
KNOWLEDGE WORKER SYSTEM MILESTONES REPORT

Prepared Under Contract
DACAR88-90-D-0006-0001

Submitted to:
Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared October 22, 1990 by:
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
KWS Delivery/Development Schedule
and Milestones Report

The following milestones are proposed for CERL contract DACA88-90-D-0006-0001 (GT contract D48-617):

- Sep 28 90: Analyze requirements for options for KWS database
- Oct 01 90: Analyze requirements for future KWS iterations
- Oct 08 90: CERL will provide comments about future KWS iterations
- Oct 01 90: Review software options for future iterations of KWS
- Oct 08 90: CERL will provide comments about software options
- Oct 01 90: Review hardware options for future iterations of KWS
- Oct 08 90: CERL will provide comments about hardware options
- Oct 15 90: White paper draft about dynamic scheduling techniques
- Oct 15 90: CERL will provide comments about database requirements
- Oct 29 90: Final database report
- Oct 29 90: CERL will provide comments about dynamic scheduling
- Oct 29 90: Draft of checklist of specifications for KWS
- Oct 29 90: Draft of short term development plan for KWS
- Nov 12 90: CERL will provide comments about development plan
- Nov 12 90: CERL will provide comments about specifications list
- Nov 19 90: Final version of white paper
- Nov 26 90: Final development plan incorporating CERL comments
- Nov 26 90: Final specifications checklist incorporating CERL comments
- Feb 04 91: End of task briefing
- Mar 18 91: Investigate the possibility of neural networks and KWS
- Apr 01 91: CERL will provide comments about the neural network report
- Mar 18 91: Evaluate software packages applicable to KWS
- Apr 01 91: CERL will provide comments about software packages
KNOWLEDGE WORKER SYSTEM SPECIFICATIONS CHECKLIST

Prepared Under Contract
DACA88-90-D-0006-0001

Submitted to:
Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared October 31, 1990 by:
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
Knowledge Worker System Specifications Checklist

This document presents a checklist of specifications for the Knowledge Worker System, Version 1.0. The specifications have been divided into 12 categories which describe the functionality, hardware, and software of the system. These categories are: help, communications, user interface, data organization, automatic operation, Todo list, modifications to the master schedule, viewing the master schedule, task schedule, control issues, the software platform, and the hardware platform.

A. Help
   [ ] Context sensitive
   [ ] Electronically submitted
   [ ] On-line tutorial
   [ ] 4 week phone support

B. Communications
   [ ] Terminal sessions
   [ ] File transfer capability
   [ ] Electronic notification of schedule changes
   [ ] Remote access to messages from other knowledge workers
   [ ] Remote access to all KWS functionality

C. User Interface
   [ ] View Todo list
   [ ] Colors used as visual clues to important information
   [ ] 'Hot keys' to invoke KWS functions
   [ ] Require user acknowledgement on important changes
   [ ] Pop-up reminders to be used

D. Data Organization
   [ ] Version control for multiple versions of attached documents
   [ ] Document can be searched by keyword
   [ ] Automatic backup of data

E. Automatic Operation
   [ ] Automatic invocation of supporting software

F. Todo List
   [ ] Show tasks for which each knowledge worker is responsible
   [ ] Show when tasks are due
   [ ] Show priority of each task
   [ ] Show status of each task
   [ ] Show tasks that are in danger of becoming late
   [ ] Identify tasks that have a due date that is simply an estimate
   [ ] Show how much of task has been completed
   [ ] Show when a task should be started
   [ ] Allow scheduling of personal items
   [ ] Show "work drivers" (why and for whom the work is to be done)

G. Viewing of Master Schedule
   [ ] Tabular Display showing event, tasks, and subtasks
   [ ] View predecessor tasks and status in STEP window
H. Modification of Master Schedule
- Separate user interfaces
- Add/Delete/Modify capability
- Automatically maintain logic of schedule
- Convert Ad-hoc tasks to routine tasks
- Cyclical tasks need to be entered only once

I. Status
- Maintained by knowledge workers
- Subtask priority will be maintained
- Latest start time will be sorting criteria
- Knowledge workers can view predecessor task status
- Color will be used to show important information
- Temporarily reassigned subtasks can be seen on both knowledge workers' Todo lists

J. Control
- Knowledge workers will be able to add private items to Todo list
- Meeting/Appointment management will be left to existing packages
- Automatic backup of servers will be supported
- All knowledge workers will be allowed to make schedule changes

K. KWS Software Platform
- Microsoft Windows 3.0
- Customized Scheduler
- VistaComm
- Custom windows software written in C
- 3+Open
- 3+Open Internet
- 3com TCP/IP (DDN Protocol)
- Gupta SQL/Windows/Oracle for SCO Unix
- SCO Unix

L. KWS Hardware Platform
- Compaq 386/20e with 5 megabytes of memory
- Unisys 386 with
  - 14 megabytes memory
  - 330 megabyte hard disk
  - Tape backup unit
  - Uninterruptable power supply
- Gateway machine - Unisys 386
KNOWLEDGE WORKER SYSTEM
SHORT TERM DEVELOPMENT PLAN

Introduction

This document describes the short term development plan for the Knowledge Worker System, Version 1.0 (KWS). This plan will describe the steps that will be undertaken by the Georgia Institute of Technology during the process of developing a working version of the Knowledge Worker System. Version 1.0 will be based on a prototype version that was developed for CERL in 1988. This document will describe the changes that will be made in the prototype and the general plan for development until February 1991. The plan will cover such issues as the software platforms that will be needed to run the Knowledge Worker, the hardware platforms, changes that must be made since the development of the prototype, the user interface, user help facilities, the organization of documents, communications and networking, scheduling, and the series of events which will constitute the delivery and set up of the product. Information for this report was compiled from a review of the JAD Workshop, a report describing the lessons learned from the prototype, and a presentation given by George Olive of Georgia Tech.

Hardware Platform

The hardware required to run the Knowledge Worker System can be divided into three types: 1) hardware required by each user at his individual workstation, 2) that which is required at the central server location, 3) hardware required for communications between the knowledge worker (workstation) and the central database site. Each knowledge worker wishing to use the KWS must have a 386 class machine (Compaq 386/20e) with 5 megabytes of memory. In addition, the user machine must possess a hard disk with at least a 60 megabyte memory capacity, a mouse for the use of windowing utilities, and a VGA monitor. A math co-processor is recommended for speed considerations. A server database will be maintained at a central location. At this location, a Unisys 386 machine with 14 megabytes of memory is required. Also, the server machine will need at least a 330 megabyte hard disk, a tape backup unit, and an uninterruptable power supply. For communications purposes, a second Unisys 386 will be used as a gateway machine.

Software Platform

The software used for the KWS will be used to integrate a variety of functions, including the user interface, data base utilities, and communications. The KWS will run under Microsoft Windows version 3.0 and use customized windows software developed in the C programming language by Georgia Tech. For communications, KWS requires VistaComm communications software, as well as 3+Open, 3+Open Internet, and 3Com TCP/IP (DDN Protocol). For accessing data base functions from a window environment, KWS will use Gupta SQL/Windows and Oracle. SCO Unix will be required for the data base server.

User Interface

The KWS will run under Microsoft Windows 3.0 and use the windowing environment for the graphical user interface. The graphical viewing facility will allow and provide for the viewing of the KWS data in a hierarchical fashion, as well as viewing the schedule data and attached documents. While viewing the scheduling data, the user will have the ability to group the data and view it by week, month, year, etc. While the KWS is running, the keyboard will be set up to activate "hot keys." Hot keys will provide the user with the ability to invoke an often called KWS function at the touch of a single program key.
Help

The help that is available to the KWS user will come in a variety of forms. First, a corpus of hard copy manuals will be provided with the delivery of the software. These will include a users' manual, training guides and a technical manual. In addition to the printed manuals, on-line help will also be available to the user. This help will be context sensitive, and the messages provided will depend upon the current mode of KWS. There will also be provisions made for human aided help. For the two weeks after installation of version 1.0 (February 6 - 19, 1990), a Georgia Tech representative will be on-site to provide support and answer questions. Following that for an additional four week period (February 19 - March 19, 1990) Georgia Tech will provide support by electronic mail and telephone.

Automatic Execution

The ultimate goal of the KWS is to perform as much of the repetitive tasks as possible a Knowledge Worker must do each day. Since many of the tasks a user must do are cyclical, or must be done periodically, KWS will provide a facility for the automatic invocation of programs. This facility will allow for a knowledge worker to start external programs and supporting software from within the Knowledge Worker System environment. McClendon Automation will also provide a facility which will automatically write PAX reports. In addition KWS will eventually provide methods for automating the process of entering the steps involved in performing a subtask, as well as automating the production of documents and reports.

Communications

The Knowledge Worker System consists of a data base server/host machine which can be accessed by different Knowledge Workers with PCs in (possibly) different offices or locations over a 3+Open local area network. The communications provided by KWS will allow Knowledge Workers to access all of the centralized data base information as well as communicate with any other Knowledge Worker. The Knowledge Worker System will also accommodate Knowledge Workers working from a remote site by using 3Com and 3+Open communications software. The ability to access other computers will be provided by simple terminal sessions, as well as file transfer capability. The users will be able to communicate with one another by the way of an electronic message facility. At first, only a notification daemon will inform the user when a remote job is completed or an important task is added to the ToDo list. A more complicated electronic mail facility will be added in the more distant future.

Document Organization

In contrast to the prototype, the KWS will provide a less structured way of handling documents. The KWS will provide the facility to relate any document to any task (event, task, subtask, or step) in the system. Besides providing a more flexible interrelated document structure, KWS will also provide support for the automatic version control of documents when ever a new version of an existing document is created. KWS will also provide the ability to search for any document in the system based on a keyword. A document search by similarity of tasks (to the current task) and the ability to archive documents are planned to be included in the long term development scheme.
Prototype Technical Changes

According to the reviews of the prototype and its usability, there are four main issues that must be changed when developing the actual KWS. These have already been discussed in some detail. The four issues are 1) adding a distributed database, 2) generalizing the hypertext structure, 3) implementing KWS in a compiled language, and 4) the elimination of DOS as the underlying operating system.

The database chosen to support KWS is a completely distributed database. The data will be placed in a central location and will be able to be accessed by all KW users. The hypertext structure was described in the above section on document organization. In the prototype, documents and programs could only be linked to steps. This restriction is not necessary and will be removed in Version 1.0. KWS will provide the ability to link documents and programs to steps, tasks, subtasks, and events. To increase the usability and speed of the system, the production version of KWS will be implemented in the C programming language with customized interface to the Microsoft Windows system, eliminating the need for DOS support.

Implementation Events

The following describes the logical sequence of implementation events for the Knowledge Worker System.

- Delivery of initial KWS Version 1.0 to CERL
  November 19, 1990

- Two one-week testing periods by CERL
  November 19, 1990
  January 14, 1991

- Installation of KWS Version at CEMP
  February 02, 1991

- Workshops demonstrating usability of KWS
  January 14, 1991
  February 04, 1991

- Training of KW users
  February 05, 1991

- Support Periods (2 weeks on site, 4 weeks phone/e-mail)

Equipment Summary

Hardware:
- Compaq 386/20e with
  * 5 megabytes memory
  * 60 megabyte hard disk
  * mouse
  * VGA monitor
  * math co-processor
- Unisys 386 with
  - 330 megabyte hard disk
  - tape backup unit
  - uninterruptable power supply
- a second Unisys 386 for gateway communications

Software:
- Microsoft Windows 3.0
- VistaComm
- 3+Open
- 3+Open Internet
- 3Com TCP/IP (DDN Protocol)
- Gupta SQLWindows / Oracl for SCO Unix
- SCO Unix
DEVELOP SPECIFICATIONS FOR KNOWLEDGE WORKER SYSTEM, VERSION 1.0

Prepared Under Contract DACA88-90-D-0006-0001, Task 15

Task 2-2

A White Paper on Dynamic Scheduling Issues

Submitted to:

CONSTRUCTION ENGINEERING RESEARCH LABORATORY
Champaign, Illinois

Prepared May 10, 1991

by

GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
DYNAMIC SCHEDULER ISSUES

Introduction

The purpose of this white paper is to discuss the issues associated with the development of the dynamic scheduler of the Knowledge Worker System in the light of discussions between John Sheehy, CERL, and Georgia Tech. Included in this discussion are items raised by the two JAD workshops for the Knowledge Worker System User Group.

There are two possible configurations of the dynamic scheduler. One configuration is an internal scheduler using custom code within the Knowledge Worker System. The other configuration would include off-the-shelf scheduling software for the master schedule. These two configurations are discussed separately below.

Internal Schedule

In the internal schedule approach, the master schedule resides within the Knowledge Worker System. Schedule logic, task duration, and other necessary information would be part of the Knowledge Worker System knowledge base.

The basic structure of the master schedule will follow the previously established "Event - Task - Subtask - Step" hierarchy. The basic elements of this hierarchy are restated here for clarity.

- Event - Starting or ending date for a task.
- Task - An activity with a definite ending date and duration for which a supervisor is responsible.
- Subtask - An activity which is a sub-component of a task. An activity at the subtask level is assigned to no more than one Knowledge Worker.
- Step - Items to be done in order to complete a subtask.

The "Event - Task - Subtask - Step" hierarchy is shown graphically on Figure 1. Generally, all PPBES tasks selected for the master schedule will connect to one another according to their interdependence thus forming a continuous network of tasks. For example, for the two tasks shown in Figure 1, the end event for Task A would be the beginning event for Task B. Since tasks are interconnected, changes in task due dates are automatically propagated throughout the schedule. Updated task due dates will be passed to Knowledge Worker's by way of the ToDo list. There could be isolated instances of tasks not connected to the network of tasks. However, these isolated tasks would still be related to the accomplishment of some milestone and the associated due dates would be reflected on the ToDo list.

Subtasks will be included as separate subnetworks under a "parent" task as shown in Figure 1 under Task A. Accordingly, changes in due dates of tasks are also propagated to subtasks. Moreover, subtasks of different tasks can be linked. The linkage, in this case, is
informational only. That is, the link will show that a subtask of one task is related to a subtask of another task, but changes in due dates are not passed between these subtasks.

The alignment of subtasks will be according to their late finish time. Consider Task A in Figure 1. Here, the subtasks are assumed to occur sequentially. The first subtask is due at the end of day one, the second at the end of day two, and the third at the end of day three. Since the first two subtasks require less than eight hours to complete, there will be some slack time associated with them. On the figure, slack time is shown at the beginning of the subtask rather than at the end. This is because users seemed to want to know when a subtask was due rather than when it should start.

Although the sequencing of the subtasks of Task A do not require more than an eight hour day to complete, it is possible that a combination of subtasks could exceed eight hours. This is depicted under Task B of Figure 1. In this case, the time for the ad hoc tasks added to

Figure 1 – Event-Task-Subtask-Step Hierarchy
time for the subtasks could exceed the time available. These instances will be "flagged" to the Knowledge Worker or supervisor so they may do whatever rescheduling is necessary. It is important to observe that whenever several subtasks occur on the same day, no attempt is contemplated to indicate when during the day the subtasks should be started. It is felt that this type of "micromanagement" would be counterproductive.

The master schedule will initially be developed using "dummy" tasks to demonstrate proof-of-concept. As we receive the actual tasks with their predecessors and successors, we will incorporate them into the master schedule.

The internal version of the master schedule is evaluated below in the context of the following general issues: ability to change the master schedule, ability to view the master schedule, how to determine task/subtask status, job management items, coordination items, privacy, ad hoc tasks, automation, supervisory issues, and control issues. These general issues were extracted from user comments generated in the JAD workshop in November, 1989 and the Knowledge Worker workshop in April, 1990.

*Change Master Schedule:* This issue concerns the ability of a Knowledge Worker or supervisor to add, delete, or modify tasks and subtasks. Clearly the ability to change schedule data will depend on the authority of a person to make changes. Given the proper authority, items such as event due date, task/subtask duration, knowledge worker assigned, priorities, and so forth can be changed through an appropriate menu.

A separate scheduling interface will be available to make changes in the master schedule. This interface will automatically maintain the logic of the schedule upon addition or deletion of subtasks.

"What-if" analysis is not included in the current version of Knowledge Worker. In future versions this analysis will be done by downloading a local copy of the master schedule. Any of the changes that are possible with the master schedule will be possible with the local copy.

Converting an ad hoc task to a master schedule task also depends on the authority to change the master schedule. On the other hand, permanently adding an ad hoc task to one's personal calendar will be possible through the ToDo menu.

One user had requested an ability to integrate their schedule with another for dependent tasks. It is assumed that the logic of the entire function will explicitly include this integration. Further, it is anticipated that some of this information will be part of the ToDo list. See *Job Management* and *Coordination* below.

*View Master Schedule:* In the short term, the ability to view the schedule will be limited to tabular displays showing dependency lists. It is not anticipated that this version of Knowledge Worker will have extensive or polished graphical displays of the master schedule in the form of, say, a bar chart. However, a knowledge worker will be able to "walk" forward and backward through the schedule by clicking on an item in the ToDo list and getting to a subtask window.
This subtask window can have additional schedule information related to the subtask. For example, a knowledge worker, while viewing the subtask window, will be able to access a list of predecessors and successors for that subtask. Clicking on a predecessor or successor will take the knowledge worker to that particular subtask and its associated information.

In this way, a knowledge worker can see the progress of others who must complete work before the knowledge worker's subtask can begin. The knowledge worker can also see who, down the line, depends on their work. This type of information would also be available as part of subtask data, but some traversing to other subtasks would be necessary for additional detail.

Since the schedule will be organized around a data base, it should also be possible to view events or tasks and list only the steps for which a knowledge worker is responsible. Tracking by functional area and divisions should also be possible using this approach as well as organizing subtasks by date and by knowledge worker.

Task/Subtask Status: Status will be maintained by individual knowledge workers. They will enter percent complete, OBE, submitted but not approved or other status as appropriate. The updated status will be clearly "flagged" within the ToDo list at each new logon.

Job Management: Job management takes place primarily within the ToDo function of the Knowledge Worker System. Subtasks can be displayed with an indication of their criticality or priority. The latest time a subtask can be started will be displayed on the ToDo list. It appears from user responses that late start times rather than early start times are preferred.

Subtasks which are late will be shown in a distinctive color or pattern. Changes since last logon will also be displayed in a distinctive manner. Subtasks in danger of being late due to overdue predecessors will be flagged on the ToDo list. More detail on predecessors can be shown in the subtask window described above under View Master Schedule.

A "look ahead" feature can be included to permit a knowledge worker to scan forward in time to see those subtasks due in the future.

Status of work reassigned to other Knowledge Worker will be available from the original Knowledge Worker's ToDo list. Naturally, the reassigned work will appear on the ToDo list of the delegated Knowledge Worker.

Coordination: Coordination information can be displayed in the subtask window described above in the section on View Master Schedule. Information in this window includes items such as who depends on my subtasks and what subtasks must be complete before I can begin a subtask. The completion of all predecessor subtasks as well as players for each task and a point of contact for the event can also be displayed in this window.
Privacy: This is primarily a personal calendar issue. In the current version of Knowledge Worker, privacy is not available since one Knowledge Worker can log on as another and will have access to the first Knowledge Worker’s ToDo list.

Ad Hoc Tasks: Ad hoc tasks can be added to the ToDo list by the Knowledge Worker. These are tasks which are not normally added to the master schedule but would remain in their personal calendar until removed. Ad hoc tasks will not affect the master schedule calculations.

Automation: A limited degree of automation is possible in this version of Knowledge Worker. For example, historical schedule information (that is information on how long it took to complete a subtask) can be automatically recorded. However, merely recording elapsed computer time from between when a subtask is opened until it is closed may not accurately reflect the time it actually took to do the subtask. Knowledge Workers will need to be careful to record actual task and subtask duration.

True learning from schedule performance, in the sense that the computer interprets and saves actions by Knowledge Workers, will not be possible. On the other hand, the system will save updated steps and documents that were modified or added during a session.

Limited versions of job descriptions could be generated. Conceptually, these job descriptions would be a listing of subtasks/steps and selected ad hoc items which can be retrieved from the knowledge base. Items could be "flagged" and printed as outlines to a job description package. This capability is not expected in Version 1.0 of Knowledge Worker.

It is not expected that this version will generate or support performance reviews. Future versions of the system could be tailored to extract pertinent information from a Knowledge Worker’s file to support performance reviews.

Reports, charts, and briefing materials will be available to the extent they are already available in the Knowledge Worker System.

Automatic periodic backup will be a part of this version of the Knowledge Worker System.

One user response requested that the system be able to do group scheduling by locating blocks of free time and electronically notifying participants. It is possible the personal calendar portion of the system could accommodate this feature, but it is not included in the current specifications. There are a number of packages currently available that could be used to perform this function.

Supervisory Issues: Supervisory items requested include the ability to plan for leave or TDY, advise on available staffing for new task/subtask, and advise on the impact of external changes. Implicit in these items is a requirement for a daily resource loading in hours. It should be possible to scan the schedule and personal calendar data base and accumulate hourly tasking per Knowledge Worker per day.
Management reports of time and resources expended should be reasonably available by accessing the schedule and personal calendar data base.

One request was for the impact of an absent Knowledge Worker. An associated request was for a listing of the tasks of an absent Knowledge Worker. This type of information is available by calling up an individual Knowledge Worker’s ToDo list.

**Control Issues:** A protocol regarding who, how, and when to make schedule changes will be required. Supervisors have authority over changes to the master schedule. A change window as part of the ToDo list will be the vehicle for changes. Schedule changes and status updates could take place at the end of the day. A controlled queuing system will be established to insure changes take place in an orderly fashion. After these changes/updates are incorporated into the master schedule, they would be available to supervisors and Knowledge Workers at their next logon.

**External Schedule:**

In the external schedule configuration, the master schedule resides outside the Knowledge Worker System. Conceptually, the results of critical path calculations would be passed to a Knowledge Worker System database by way of a transfer file. The resulting due dates and other information would then be available for use by the Knowledge Worker System. Any time a change is made in schedule data from the Knowledge Worker System, the change is passed back to the master schedule by file transfer. The schedule would then be updated and the results returned to the Knowledge Worker System.

It is assumed that an external scheduler would do all the normal critical path calculations one expects from any commercially available schedule package. Beyond these basics, minimum criteria for selection of an external scheduler include the following.

- Must transfer data to and from the Knowledge Worker System.
- Must be networkable.
- Must permit "what if" scheduling.
- Activity durations must be in either hours or days.
- Not be limited to number of activities.
- Must indicate percent complete.
- Must handle subprojects.
- Must accommodate some form of resource allocation.
- Must handle milestone type activities.
- Must integrate personal activities.
- Must incorporate individualized status reports.
- Must be able to view the schedule.
Several commercial scheduling packages meet most of the above criteria. Among these packages is Microsoft Project for Windows. This package was because it operates in the Microsoft Windows environment. Microsoft Project for Windows is a mid-range package in terms of capability and price. Additionally, Microsoft Project for Windows would work on the 386 class machines currently envisioned for the Knowledge Worker System.

Microsoft Project for Windows meets most of the above listed criteria. Data transfer in the form of CSV (Comma Separated Values) is possible. Therefore passing data from the Knowledge Worker System to the external scheduler should present no difficulty. Networking is possible, should that be necessary. "What-if" scheduling is available by maintaining several copies of the master schedule. Activity durations may be in days or hours. There is no limit, except for available memory, on the number of activities. Percent complete is indicated as well as duration to complete. Subprojects are possible. Resource allocation is limited to informing the user when required resources exceed available resources. The system does not automatically shift resources so as not to exceed a given limit. Milestone activities are possible. Personal activities are not possible except those a Knowledge Worker inputs into the external schedule. Users can view the schedule in Gantt and CPM format. Prints and plots of the schedule are also available. Finally, individualized status reports are possible.

One potential advantage of an external scheduler is in its graphic presentation of the schedule. This advantage is almost totally offset when one considers that the time scale for most of the Knowledge Worker tasks is less than a day. If the time scale is in hours, a schedule spanning several years would be nearly unmanageable with respect to viewing the schedule on-screen.

Other disadvantages with Microsoft Project for Windows are in resource allocation. The user is informed when enough resources for a task are not available, but the system does not shift activities to allocate resources to a given limit.

A further disadvantage is that much of the flexibility of the internal scheduler would be lost when one has to work within the capabilities of the external scheduler. Additionally, all Knowledge Workers would have to be familiar with whatever scheduling package is chosen in order to view schedules and conduct "what-if" analyses.

Although Microsoft Project for Windows could work in a somewhat restricted manner, it seems inappropriate for the dynamic scheduler. While other scheduling packages might overcome some of these restrictions, no package is designed to the special requirements of the dynamic scheduler. The reason for this is that the networks of the Knowledge Worker System are not in the form of traditional scheduling networks for which off-the-shelf packages are intended. Thus, the highly specialized nature of the Event-Task-Subtask-Step hierarchy seems to indicate an internal scheduler which can be tailored to its unique requirements.

Summary

Based on an evaluation of Microsoft Project for Windows, it does not appear that off-the-shelf scheduling packages are not suitable for the dynamic scheduler.

The current version of the Knowledge Worker System will include the following capabilities. Users can add and delete tasks as well as modify existing tasks through menus in the ToDo list. Ad-hoc tasks can also be added to the ToDo list. All changes to the schedule
will be consolidated at an appropriate time and the master schedule will be updated. These changes will be highlighted to Knowledge Workers the next time they log on to the system. Knowledge Workers will not be able to see a graphical presentation of the schedule network in this version, however they will be able to move forward and backward through the schedule by "clicking" on predecessors or successors. By moving forward and backward through the schedule, Knowledge Workers will be able to see the status of tasks assigned to other Knowledge Workers. "What-if" analysis of changes to the master schedule is not included in this version.

Status of tasks and subtasks will be shown on the ToDo list. Knowledge Workers enter the percent complete of subtasks. Subtasks will be displayed by priority and by due date. Additionally, the latest time a subtask can be started will be displayed on the ToDo list. Daily resource loading in terms of hours required of each Knowledge Worker will be available.

A degree of automation is included in this version of the Knowledge Worker System. Job descriptions, limited to a listing of subtasks and steps, can be generated. Automatic periodic backup will be included. True learning of Knowledge Worker's performance is not included, nor was it contemplated. However, it might be an useful future enhancement.