An approach to the definition and implementation of a software development environment

by JAMES F. ELWELL
TRW Defense & Space Systems Group
Redondo Beach, CA

ABSTRACT

During the past two decades, a marked increase in software costs has been seen. The ingredients are now present to define and implement a software development environment which provides an increase in programmer productivity. The methods used by TRW to identify the goals of such an environment and define the components of the environment are discussed. The resultant TRW Software Office Of The Future is presented and its current status given. Observations relevant to this process are made.
INTRODUCTION

In the past two decades, we have both witnessed and participated in a remarkable expansion of the role of the computer in commerce and industry. For those of us associated with software, it has been particularly significant because during this time the cost for software has overtaken that of hardware (see Figure 1), as predicted by Boehm in 1973.1

Another consequence of this expansion has been the steady increase in demand for the trained computer professional—a demand that has not been matched by the university, which is the main source of new computer professionals.2,3 This trend is expected to continue for the foreseeable future.

The cost implications are clear; unless something is done, software costs will continue to rise at a rapid rate. Increasing programmer productivity is essential to keep software costs down.

During the same period of time, the cost of the computer has shown a marked decrease. For example, in the early 1960’s an IBM 7094 cost approximately $1.5 million, while today a computer with comparable power (e.g., a DEC PDP 11/70) costs $100,000.4 This is the other half of Figure 1, and the prediction by Boehm seems on schedule. In addition to becoming cheaper, the computer is packing increasing power into smaller and smaller packages. In the past few years we have witnessed what can be described only as the dawn of the age of the microcomputer. The computer has now become accessible to the user, no longer requiring the specially equipped rooms of their predecessors. This accessibility, coupled with decreased cost and increased power, suggests that hardware which could be used to support software productivity gains is available. Computer professionals can now off-load portions of their work to hardware, thus “automating” themselves as a way to becoming more productive.

Software expertise has also shown rapid expansion during this period of time. A brief summary of the evolution of software tool systems shown in Table I will exemplify this.

Each generation of these tool systems has provided significant productivity gains. For example, the switch from assembler code to a high-level language increased the number of machine instructions produced per source instruction by a factor of 5.5 There has likewise been a notable advance in software development methodologies, the application of which tends to reduce the cost of software.6,7

The conclusion to be reached is that the ingredients are present to define and build a software development environment that supports programmer productivity gains. The motivation is provided by the increasing software cost and shortage of computer professionals. The means are provided by the decreasing hardware cost, increasing hardware capability, improved software tool system, and the evolution of software development methodologies. In fact, the development of this productivity environment is essential. In the remainder of this paper I would like to describe the steps taken at TRW to define and implement the TRW Software Office Of The Future. I will conclude with a brief status report and some observations which are relevant to this process.

<table>
<thead>
<tr>
<th>Time Span</th>
<th>Tool system</th>
<th>Example</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1950–</td>
<td>Assembler</td>
<td>IBM map</td>
<td>Mnemonic code, macro capability</td>
</tr>
<tr>
<td>1957–</td>
<td>Compiler</td>
<td>FORTRAN, ALGOL</td>
<td>High-level language, functions</td>
</tr>
<tr>
<td>1963–</td>
<td>Operating system</td>
<td>System 360</td>
<td>Multi-language support, utilities, simple job streaming</td>
</tr>
<tr>
<td>1968–</td>
<td>Time-sharing system</td>
<td>IBM TSO, TRW TSS</td>
<td>Remote job entry, collection of independent tools, multitasking</td>
</tr>
<tr>
<td>1974–</td>
<td>Software Environment</td>
<td>UNIX, ALTO</td>
<td>Integrated tool set</td>
</tr>
</tbody>
</table>
LONG-TERM GOALS

The first step in defining an environment that will provide productivity gains is to understand just what gains are desired. To do this, TRW established a Software Productivity Tools Working Group, whose charter was to analyze software tool usage at TRW, assess the current and likely future state of the art in software, and recommend actions leading to improved software productivity within the company. Although the Group concentrated on automated aids to software development, considerations of documentation, facilities, and procedures were central concerns throughout the study. The major conclusion of this study was that software tools are a critical part of an integrated productivity improvement strategy which could increase productivity on large software projects by a factor of two in 1985 and by a factor of four in 1990. In addition to productivity improvements directly related to tools (such as increasing tool usage, increasing interactive development and the avoidance of the cost of retooling), the indirect gains (such as personnel motivation, modern programming practices, tool system experience and stability) were considered.

To achieve these productivity goals, the concept of an integrated TRW Software Office Of The Future (TSOF) was defined. TSOF drew upon the best of environments that are currently defined or operational, modified to meet the unique requirements of TRW. Its goals are to provide the following characteristics:

1. Hardware: low-cost, medium-power personal computer each with individual file-storage capability; high-capacity bus which connects the terminals in a local network to a medium-size mainframe (e.g., DEC VAX 11/780), which supports the network and various peripherals; a gateway from each of the local networks to a large-scale mainframe (e.g., CDC Cyber).
2. Software: an integrated tool set which supports the software development life cycle and is user-friendly, easy to use, portable, expandable, flexible, and secure.
3. Facilities: offices which are approximately 100 square feet with floor-to-ceiling walls and which provide adequate work space, storage, lighting and quiet for a single person.

Thus, the following set of goals was established:

1. Implement a TRW Software Office Of The Future
2. Increase productivity by a factor of two by 1985
3. Increase productivity by a factor of four by 1990

APPROACH TO REALIZING THE GOALS

The realization of the long-term productivity goals and the implementation of the TSOF require a significant level of corporate investment. To justify such an investment, an analysis of the effect of the expected software productivity gains on the cost of delivered software was made using the TRW proprietary Software Cost Estimation Program (SCEP). This showed that the productivity goals were achievable.

A further justification was made on the basis of cost. An investment of $15,000 per programmer to provide the facilities and hardware and software tools that will double (or quadruple) productivity is returned the first year they are in place. This return on investment comes from either the value of the additional software which may be built or from the reduced cost of the production of the same amount of software.

The development approach to the realization of the goals was based on three principles: (1) The resultant environment would be built in increments, (2) data would be collected to use as a guide for the next increment, and (3) the environment would be used to support an ongoing project. This approach enables the builder to define and implement something in the near-term, measure progress, then introduce new or improved hardware and/or software, and measure again. Thus a cycle of building, evaluation, and improving is established. The use of the TSOF to support an ongoing project provides a practical focus for the work.

IDENTIFICATION OF NEAR-TERM OBJECTIVES

The result of establishing long-term goals and defining an approach to their realization was the establishment of a Software Productivity Project (SPP), whose charter is to effect those two charges. As a practical matter, the building of the TSOF is a project that will take many years. A set of near-term objectives is therefore necessary to direct the work on a more manageable basis. For 1981, work proceeded based upon the following six objectives:

1. Get something up and running in 1981. This “something” was defined to be Increment One. As with any new concept or project, it is important to show results, particularly to a sometimes skeptical audience. This objective also provides an impetus to the work, as does the support for an ongoing project. The builder has promised, and the project needs some results now.
2. Increment One must be a foundation for the future. What constitutes the foundation clearly must be identified. This foundation must process at least one characteristic; the ability to accommodate change must be built in.
3. Increment One must provide some support for all phases of software development. In addition to the usual phases of software development (e.g., requirements, design, code), tools for managers and necessary support functions (e.g., documentation) must also be provided. Some tools take a long time to build. If they are not started early, they won’t be ready by the time they are needed, say in the test phase.
4. Increment One must be amenable to change. With the limited time scale for Increment One, both the hardware and the software will be a subset of that which is ultimately desired. Succeeding increments will add features to that which is built in previous increments. Changes will also occur because of technological innovation and as a result of the measurement process. Ability to accommodate these changes must be built in.
5. The total system aspects of Increment One must be emphasized. Currently, most tools are independent.\textsuperscript{11,12} With some exceptions, each was built with little or no knowledge of the other. Productivity gains accrue from the elimination of duplication of effort and the intrusion in the work of one person by another. Tools, then, should be "systematized"; i.e., the incidental information generated in the performance of one task which is valuable to another should be available to the second tool.

6. Certain aspects of potential productivity gain are deferred to later increments. The funding and time limits placed on SPP required the exclusion of some facets of the software and hardware world. In particular, the security aspects of the system (except those that are part of the foundation), graphics, and CAD/CAM were not to be considered as part of Increment One. The gateway from the local network to other networks was also excluded.

CHOOSING THE TEAM

The charter given to the Software Productivity Project (SPP) required building a team with a wide range of talent. This mix of talent is important in providing a perspective on the needs of the potential users of the resultant system and also in providing the expertise necessary to implement the system. Thus, individuals with software, hardware, and facilities expertise who also possessed managerial, project and/or line organization backgrounds were sought. Representatives from various support areas such as configuration management, data entry, and secretaries also participated in the definition and implementation of the TSOF.

Team members were also chosen because they possessed the qualities of creativity and experience, the former because it is essential for the pioneer, and the building of the TSOF is truly a pioneering effort; the latter because each individual is, in effect, a representative of a class of users. A high level of experience is required to effectively represent these users.

There must also be a careful balance between the academics and the practitioner. Both are necessary because they provide complementary talents, but one group cannot dominate the other without serious consequences for the project. With too many practitioners, an inflexible, one-purpose system may be built; while with too many academics, a sand-box project and perhaps even no system at all could result. In this case, the tie to an ongoing project is a great help, because real deadlines require real results.

The team selected for the SPP consisted of 15 full-time and 6 part-time people. Included in this group were 4 individuals with PhD’s and 10 with master’s degrees. The average amount of experience was 9 years. The team was organized as shown in Figure 2.

Organization is strictly along functional lines, with each functional area responsible for research and development expertise in its area. The system engineer has overall responsibility for the conceptual integrity of the system, its testing, and performance.

SYSTEM DEFINITION

With computational capability available directly to users from two different sources ((1) the medium-size central computer, in this case a DEC VAX 11/780, which supports the local network and (2) the personal computer/terminal), the first task of the system definition process is to partition the work. In the TSOF, the personal computer/terminal initially will be used mainly to off-load the word processing and text editing from the central computer. If a tool were able to be operated on both computers, it would be available on both. Thus, the central computer becomes a repository for the text files used to support a software project, the machine on which the large tools are run, and the storage point for the project database. This partitioning has certain hardware implications, the most important one being the requirement for a high-capacity bus to support the rapid transfer (ideally almost instantaneous) of files between the central and personal computer.

The second step in defining the system was to identify the tool set which was to be built. This tool set is dependent upon the software development methodology used at TRW. To define this tool set, the activities required to develop software according to TRW standards were identified. Then a set of tool types which could be provided to support these activities was defined. The result was a matrix of 110 activities and 25 tool types. To choose the tools to be built in Increment One, an “x” was placed at the point at which a tool type supported an activity. For the tool types which supported the greatest number of activities, the actual tools (if they existed) which provided that capability were surveyed. For this survey, the following criteria were applied to aid in the choice of the tools to be built:

1. Consistency with the near-term objectives: The ability of a tool to satisfy the near-term objectives was scrutinized. The tools which satisfied the greatest number of these objectives were retained as candidates.
2. Availability: If a tool which passed Test 1 was available and porting to the TRW system was reasonable, it was given preference as a candidate.
3. The sum total of the resources required to either build or...
port the tools chosen must be less than our allocated budget.

Clearly this was an iterative process.
Ten tools were chosen as a result of this process. These tools are described below.

**Automated Unit Development Folder (AUDF)**

This tool maintains an electronic representation of a software Unit Development Folder Cover Sheet and provides functions for the addition, modification, and deletion of items on that sheet. Each of the nine sections of a UDF is represented, and an Associated Information Pointer is included for each section to provide access to the text content of the UDF.

**PDL**

The Program Design Language is the software package developed by Caine, Farbe and Gordon, Inc. This provides a structured method for the design of software.

**Automated Office**

The Automated Office consists of a set of facilities to enhance the communications between people in an office environment via electronic means. These facilities include electronic mail, calendar, document templating, word processing, text editing, and UNIX automated office functions.

**Front-End Help**

This tool provides an initial high-level view of the tool set through two levels: (1) a menu-driven conceptual view of the collection of tools available and (2) a help capability for each available tool which describes the tool's nature, documentation, person-in-charge, and invoking command.

**FORTRAN 77 Analyzer**

This tool provides an automated system for the analysis of FORTRAN programs, including those written in ANSI 77 FORTRAN. Analysis data is provided at three levels on the static structure of the code and on the dynamic structure indicated during program testing. It is useful as a code auditor, test effectiveness measurer, and general software development aid.

**Software Requirements Engineering Methodology (SREM)**

SREM is a set of tools and a technique for defining and analyzing software requirements. The technique is built upon a language, Requirements Statement Language (RSL), readable both by computer and by person. The set of tools is collectively termed the Requirements Engineering and Validation System (REVS) and is used to analyze the requirements specified by the user for completeness and display input and output in a variety of ways.

**Automated Test Manager (ATM)**

ATM controls and supports the testing of FORTRAN software units. This includes the specification of test case inputs and expected outputs for the unit, the generation of a test driver for the unit, and the execution and analysis of one or more units using the generated test driver.

**Relational Database Access (RDA)**

RDA allows the user to modify a relational database in a user-friendly way. RDA prompts for inputs, provides help messages, validates the inputs, and provides default values. RDA is designed for a user with some knowledge of relational database and enables such a user to add, delete, modify, or show tuples in a relation.

**Requirements Traceability (RT)**

RT allows the user to trace requirements through software design and test. RT generates reports such as the test evaluation matrix and exception reports.

**Distributed System Design (DSD)**

DSD supports the designers of software development projects utilizing distributed computer architectures. By providing a central location for the design of hardware and software elements, and the interfaces between them, DSD facilitates communication among software designers, systems engineers, and database administrators and encourages an integrated design.

The third step was the choice of the operating system. There were two systems considered: (1) UNIX as distributed by the University of California at Berkeley and (2) VMS, the standard DEC operating system for the VAX. This was not the first time a choice of operating systems was made between these two alternatives. In fact, it seems that this choice has the power to arouse passionate debate. To resolve the problem, an evaluation was performed to determine which of the systems could best support the needs of the large TRW project which would be the first user of the TSOF. A list of 38 capabilities was developed, and the ability of each system to support these needs was evaluated. The major conclusions were that neither system supplied all the project's needs and that each possessed features needed by the project that the other did not. Moreover, with some augmentation, each could support the project. After satisfying ourselves through an industry survey of UNIX use that UNIX could support the software development of a large-scale real-time system, UNIX was chosen. It was chosen primarily because it offered the best approach to achieving the long-term goals of portability, flexibility, conceptual integrity, etc., which were defined for software.

During the process by which the tool set was defined, it was noted that a centralized database was an essential feature of any system that would emerge. Accordingly, a list of ten...
candidate database management systems (DBMS) was prepared and, given certain criteria, evaluated. To provide a foundation for future expansion and to provide for flexibility in use, a relational database was deemed superior to a hierarchical database. The choice was further narrowed by the choice of the UNIX Operating System. Again, the industry was surveyed to determine user experience with the now candidate DBMS's. The result of this was the choice of INGRES as the DBMS to be used by the productivity system. In addition to being the best rated DBMS, the potential to port INGRES programs to a machine, the Britton, Lee IDM-500 existed. The prospect of off-loading a large software data processing activity to hardware was very appealing.

Thus far, only the software component of the TSOF has been considered. There are two more segments which were also studied and defined: (1) the hardware components of the system and (2) the office facilities. There are three major hardware components which were chosen: (1) the bus for the local network, (2) the personal computer, and (3) the terminal. The evaluation of the potential candidates for each of these components was performed in early 1981 and has already been dated by the rapidly changing technology of the industry. The results of this evaluation are shown on Table II.

Of more interest is the resultant hardware configuration, which is shown in Figure 3.

The final component of the TSOF is the office facilities in which the programmer would work. After surveying the existing facilities in industry and the universities and visiting some of them, notably the IBM Santa Teresa facility and Xerox PARC, the basic goals for the office facilities evolved. The TSOF would be a single-person office with a closeable door and floor-to-ceiling walls. It would be self-contained; i.e., each office would be connected to the network and have sufficient power, lighting, and air conditioning to support the potential system hardware configurations. Communication with others would be via the terminal over the network and by a telephone. Sufficient internal space would be provided so that two people could meet in any office.

The IBM Santa Teresa facility strongly influenced both the design and the criteria of the office facilities. However, whereas IBM built a building which embodied their ideas, the TRW buildings were already in place. This necessitated adapting certain criteria to reality; e.g., not every office could be 100 square feet because of existing building constraints. The resultant office characteristics are shown below:

1. Facilities—Each office will be from 80–100 square feet with floor-to-ceiling walls, a closeable door, rug, walls coated with sound absorbent material and lights inlaid in an acoustic ceiling.
2. Furniture—Two chairs; approximately 30 square feet of work surface, a portion of which would be at proper terminal height; wastebasket; and two-drawer security cabinet.
3. Storage—Approximately nine linear feet of hanging files and fifteen linear feet of shelf space in close proximity to the work surface; space for tape and disk storage must also be provided.

Table II—Hardware Components

<table>
<thead>
<tr>
<th>Model</th>
<th>Manufacturer</th>
<th>Characteristics</th>
</tr>
</thead>
</table>
| Bus SPP | TRW DSSG/S&SO Redondo Beach, CA 90278 | • 150K characters/second per channel (minimum)  
• 50 channel capability  
• 100 to 300 users per channel  
• Throughput  
  —960 characters/second  
  (terminal to terminal or terminal to computer)  
  —25,000 characters/second  
  (file transfers between computers) |
| VAX 11/780 | Digital Equipment Corporation Maynard, Mass. | • Medium scale 32 bit computer (4 M bytes RAM)  
• Two 250 M byte disks  
• 30 users  
• UNIX/Operating System |
| Computers DYNASTY | Tymshare/Santa Cruz Operation Santa Cruz, CA | • LSI 11/23 based microcomputer (250 K bytes RAM)  
• 30 M byte Winchester disk and two 8" floppies  
• 4 users  
• DYNIX Operating System (Derived from UNIX) |
| VT100 | Digital Equipment Corporation Maynard, Mass. | VT100 or VT100 emulator:  
• a multitude of keyboard selectable options, e.g.,  
  —2 scroll speeds  
  —scroll/no scroll button  
  —80/132 column selectable format  
  —print white on black or black on white  
  —throughput up to 1,920 characters/sec |
| Terminals VISUAL 100 | Visual Technology, Inc. Tewksbury, Mass. |                                                   |
| CIT 101 | C.I.TOH Electronics, Inc. Los Angeles, CA |                                                   |
4. Lighting—A task light over the work surface in addition to industry-standard overhead lights.
5. Ambience—The office should offer a pleasant work atmosphere and, where possible, be comprised of ergonomically correct components.

To satisfy these criteria, it was necessary to have some of the components custom-designed. By working with various vendors, a full-scale prototype office was built and furnished. A picture of the resulting office is shown in Figure 4. A full-time programmer was assigned to work in this office as a test of the configuration. The feedback from this experience was used in the design of a much larger area, which would be built to house the contingent of programmers who would be the first users of the TSOF.

MEASUREMENT

One of the most important features of the TSOF is the ability to gather feedback from and measure the performance of the users. By measuring the performance, improvements to the system can be identified and their effects quantified. Measurement also provides the hard data which is necessary to convince management of the need to continue their commitment to this concept because it is providing a return on their investment.

There are two types of metrics which will be measured. The subjective metrics will explore the attitudes and impressions of the users toward the TSOF. Such impressions can be used to rate the morale of the user, identify the best features for reinforcement, and spot weaknesses for improvement. The objective metrics which will be used are: time sheets, which will be used to determine work patterns; system accounting; delivered source instructions; and cost model ratings for the TRW SCEP Program. The latter metrics will be used to provide an object measure of work activity and productivity.

The data will be gathered in the following ways: automatically as the users perform their work, and subjectively, by interviewing the users, by filling out questionnaires, and by observation. Only the approach to performance measurement has been done this year. The actual performance measurement and analysis is an effort for 1982.

STATUS

At the time of this writing (November 1981) an area of approximately 6,000 square feet has been configured in two equipment rooms, two secretarial bays, and 37 offices. Construction of this area is complete. The office furniture for this area has been designed and built and is beginning to arrive and be installed.

Designing, building, equipping, and working in a prototype office during the year was completely successful. The feedback received from the occupant and from the many visitors to this office was used to improve the design of the subsequent offices.

The hardware components, with one exception, have been ordered, have arrived, and have been installed in temporary locations pending completion of the offices. Although the full
local network is not yet available, valuable experience has been gained in using the personal computer/terminal in a stand-alone mode. In this mode they are able to support virtually all their assigned functions, the major exception being the transfer of files between the personal computer and the central computer.

The schedule for the delivery of the software tools, again with one exception, is being met. Some of the tools are in operational use on the central computer, others are in integration and test, and one, the Automated Office, is operational on both the personal and central computer.

OBSERVATIONS

The project to build the TRW Software Office of the Future has had a successful first year. However, like all projects, there were deviations from the original plan, oversights, and things done so right that they bear pointing out because of their potential value to others who attempt similar work.

The first observation to be made is that the project expanded virtually from the time it started. The building of the TSOF was a relatively new idea within many areas of TRW, and everyone wanted to have the final product today. Because of this pressure, every facet of the project expanded in scope; the tools had more capability than originally planned, more hardware was required to support this expansion, and the size and diversity of the user group was increased. The staff, too, contributed to this expansion with a wealth of ideas and just plain hard work. All of this is good, but it is easy to lose control of the activities and thereby lose your way. The incremental development approach which was used on the project was very helpful in collecting these ideas and channeling this pressure along constructive, controlled paths.

The next observation to be made is that as different groups of potential users view this system, they feel that it rightfully belongs to them and should reflect only their needs. The system is flexible enough to accommodate that view. In particular, it is difficult to resist tailoring the goals and objectives of the project to those of the first user. With different areas having different aims, control of the project becomes difficult. To mitigate this problem, a high-level program office was established to oversee the direction of the project. This central point of both control and contact with the different parties served to coordinate the work. The main advantage of this centralization is that it provides an opportunity to build only
one system, thus saving the cost of duplication if each of the areas built their own.

The importance and scope of the training necessary to support the use of the tools provided by this project was grossly underestimated during the planning phase. It is difficult to overstate the critical importance of a well-planned, well-rehearsed, thorough series of training sessions. This training must address psychological issues which are not part of the usual training class. After all, the user is being asked to accept a new method of doing business which changes, in some cases radically, the previous method. Training must provide the user not only with technical skills to use the tools but also with the psychological will to use them. The truth of this became apparent well into the project's life, and a crash effort to provide the many courses required by an enlarged user group was instituted. The problem was complicated because the user group was no longer homogeneous. Programmers require different training than, say, managers, and the courses must accommodate that difference.

It is essential to have the firm commitment of upper management to a project such as this. Without it there is no project. With it, it is possible to attract the high quality of staff required to do the work, little effort is used finding funds for equipment, and positive direction comes from high levels. Fortunately, we have this commitment at TRW.

There seems to be real benefit in tying the development of this software environment to the needs of an ongoing project. The system being developed is flexible by design. The potential for working a problem to death to find the "right" solution is great. (It may well be that there is no one "right" solution, but several. This suggests that what needs to be built is a system that is easily configurable to each individual's needs and idiosyncrasies.) Having a project depending upon the completion of tools to meet their deadlines serves to focus the discussion so that results are produced more rapidly. It is worth the risk of tailoring the system too closely to the needs of the project.

Finally, one of the most fruitful areas for increasing productivity seems to lie in building an integrated system. In this type of system, actions which are incidental to performance of one job are captured and become the essential data needed for another. To date, most of the tools which have been built are independent, one being unaware of the existence of another. To build such an integrated system requires extensive front-end analysis to understand the interrelationships between the tasks being performed and the tools being provided. By establishing tool standards early in the life of the project, individual tools may be built which fit both within the integrated system and support one another.

ACKNOWLEDGMENTS

The author wishes to both acknowledge the help given him by J. Hurvitz, M. Imura, P. Bogle, M. Green, and D. Nunez in the preparation of this paper and to thank them for it. Except for the figures, the manuscript of this paper was prepared using the facilities of the system described.

REFERENCES