Software Technology Support Center

by Reuel S. Alder

Hill Air Force Base, Utah

ABSTRACT

The newly formed Software Technology Support Center (STSC) at Hill Air Force Base Utah is designed to act as an Air Force information center to advocate and support the development of advanced software tools and methods, to sustain these tools and methods during their operational use, and increase compatibility and efficiency among support environments. As a focal point the STSC will collect and disseminate information to assist the developer in creating methods, tools and environments, and to assist the user in selecting the appropriate methods, tools, and environments for his application. The experience of Air Force users with specific software products will be collected and tabulated. The STSC will then provide an unbiased reference point from which other users can make better informed decisions in selecting their tools and developing plans for future improvements.

INTRODUCTION

There is a software crisis in American industry which continues to grow. This crisis can be measured by the amount of generated software and the rate of its growth. Software created for Mission Critical Computer Resources (MCCR) increases by as much as 30% per year. This means that in three years software size almost doubles. Meeting this demand is difficult enough without the added exponential relationship between code size and complexity. The combined effect of these two facts is alarming. The number of software professionals to address this demand is increasing by 4% per year while their productivity is increasing another 4% per year. Clearly, a shortfall exists.

Government and industry are attempting to solve this problem by creating advanced tools, methods, and environments for software development and maintenance. The development of the Ada language standard is an example of this effort. A major stumbling block to success is the hiding of information due to the sheer magnitude of it available and by industry's need to defend trade secrets. While we cannot divulge industry secrets we can better manage the volume of information available.

Software methods are a disciplined technique or systematic procedure used in the software process. Object-oriented design is an example of a design method used in the design phase of software development. Tools take the form of hardware or software. They are automations of specific methods. Computer Aided Software Engineering (CASE) tools are an example of a tool that assists in requirements definition, system analysis and design, and software design phases of software development.

Tools and methods combine to make software environments. Complete environments designed to support the entire software process do not exist and may never be complete. For an environment to be complete it must include tools and methods to support all phases of the software process, software management, software personnel, and facilities used to create or maintain software. Essentially, the environment must be an ecosystem to be complete. This is an area of rapid development which needs time to mature.

The creation of methods, tools, and environments which do not support each other or play together is an example of industry's lack of communication. To be useful and increase productivity the output of one tool must flow into the input of the next tool needed in the software process. The delivery of a software product for maintenance includes all the software and hardware tools used by the user.
in its development. Each time a product is delivered a new set of tools is delivered. This proliferation of tools and environments is a greater problem than the multiplicity of languages that the Ada language standard is intended to solve.

PROBLEM SOLVING

Progress in any complex human problem solving activity is seldom accomplished in a strictly straightforward manner. When we are presented with a new problem to solve, we proceed in phases ranging from apparently unorganized groping for workable solutions, to a phase of individual artistry, to a phase of scientific development. Our concern here is MCCB, but we share a common thread of concern with all software development and maintenance.

Before suggesting any specific ways to solve software problems, let us focus on problem solving in general so that we might borrow from useful general techniques. The first phase of general problem solving can be characterized as being similar to the title of a Clint Eastwood movie: "The Good, the Bad, and the Ugly." During this phase, there are many spurts of apparently unorganized and sometimes inspired activity. Typically, sometime later in this phase the ground is prepared for the transfer into the artistic phase by people generally called reporters, historians or theorists, who attempt to separate the good from the bad and ugly. This group tries to determine the rules, trends or major concepts from heaps of data and observations. An example of this separation of the good from the bad may be seen in the military context in the way General Grant approached the problems on the Civil War campaign. When asked by a reporter for his strategy for winning the war, the General responded that he did not have a strategy, he just tried to find something that worked and did it again.

The second phase of problem solving is the artistic phase of development. During this phase certain practitioners develop skill far beyond the ordinary individual. The de Vinci class or superprogrammer gains hero status with inspiration and limitless creativity while the rest of us gain journeyman status. While it would be nice to clone this group of superprogrammers, pressures of real world programming are forcing us to search for transition into the next phase - the scientific phase.

We can look outside the present systems development problem solving activity to find a corresponding example of real world forces dictating a transition to a scientific phase of problem solving. At the turn of this century, the Bell telephone was faced with a similar problem of real world workload growth outpacing the growth of supply of phone operators. At that time the growth of the number of phones and call volume was predicted to be sufficient to employ every woman in America by the end of the 1920s. Clearly, the solution had to be found in a revised way to solve the problem. The solution for the Bell System was to automate the phone switching, or to transition from the artistic to scientific phase. While it might be nice to wish that a large class of superoperators would be discovered, it was not going to happen. A new solution had to be found.

This is the stage we are in for software development. The real world workload is outpacing the foreseeable supply of programmers. An example of this is the SDI program that projects a code size in the neighborhood of a billion lines of code. When asked to translate their code size guess to an understandable format, the response was it would require the entire productive output of all programmers in the United States for the next 5 to 10 years.
We must transition into the scientific phase of software development. As an aid to this transition, the Software Technology Support Center (STSC) will act as a focal point for MCCR technology sharing. The focus of the STSC will be on methods, tools, and environments.

SOFTWARE TOOLS SUPPORT THE PROCESS

The software development process can be broken down into six basic steps. They are: requirements identification, system level analysis and design, software design, coding, test and integration, and distribution. Requirements identification involves users defining their needs. System level design is a translation of needs to system behavior. The software portion of this is documented in a system design specification. Software design is a translation of system behavior into software logic structure. If system design and software design are of good quality, then present and future user needs are much easier to meet. Coding can now follow to give body to the software logic structure. Unit level and integration tests are conducted, before distribution is initiated. No amount of testing can ensure the quality of the product. Quality must be built in through the methods, tools, procedures, and practices used in the process.

In the artistic phase of problem solving, only the superprogrammer might be using the most powerful methods and tools. In the scientific phase, we analyze, qualify, and specify tools, methods and environments. In this phase we identify the best tools, methods, and environments, and then we can share, mass produce, reuse, and improve them.

Anticipated productivity increases with the use of improved methods, tools, and environments has created a flurry of activity. Selection of the proper tools, however, is not an easy task. In many cases it is not even possible to point to a shopping list of desirable features that a given tool or environment should possess. Contractors who supply the government with deliverable software are under a tight schedule. They do not have the time to evaluate all the products that are available to them. The choices they face are (1) study the problem and evaluate several products, (2) develop their own tools, or (3) stay with their old brute force techniques. None of these options are desirable. What the users need is a fourth choice. The fourth choice is an independent study and evaluation by the STSC.

ESTABLISHING THE STSC

The Software Technology Support Center has been established to act as a central focal point for proactive management of mission critical computer system support tools and environments. The center will provide for development of the expertise, methods, and procedures to manage and guide the future evolution of mission critical software support. Initial operational capability for the STSC will be 1992. A functional concept has been determined by Air Force Program Management Directive and an initial capability exists. The STSC has been organized to address the following objectives: (1) advocate and support the development of advanced software tools and methods, (2) sustain these tools and methods during their operational use, and (3) increase compatibility and efficiency of operations among support environments.

How the STSC performs its integrating function can be described in the choice of mascots for software technology. I would recommend a rhinoceros. This animal is big, ugly, and ill tempered. It is prone to charge at the least provocation and when it does, it has little sense of direction because of poor vision. How can such a beast be controlled? The answer, while simple in concept, is difficult in practice: burrow
through the thick hide and into the nerve centers of the brain and grab hold of the communication lines to the body. Influence over the direction of the beast can then be achieved. Communication is the method by which the STSC can help to steer software technology to a converging solution.

To be successful the STSC must integrate the needs of developers, maintainers, buyers, and users. Government, industry, and academic communities must be involved. Information provided by the STSC must be accessible, understandable, and useful to all participants.

The alternative approach to the voluntary sharing of information is to create an emperor. The emperor represents institutional power, but he would also have to have academic recognition and a bag of gold. The problem with such an approach is that creative people resist the blind and demeaning regulation of bureaucracy. Creativity cannot be forced but must flow from a sense of self worth and a desire to contribute. Money motives, but it creates problems of its own. This is evidenced by the marketing of software products to fit a problem instead of solving the problem.

Dogmatic approaches to problem resolution tend to be idealistic. These solutions give little regard to technical details which must be resolved to make them workable. The laws of logic and reason do not respond to dictators, but exist in a sphere independent from our own desires and must be diligently sought. Government standards therefore should not be applied to soon in the development process. An attitude of exploration and investigative searching is required to identify a set of valid alternatives, with the understanding that the design process is never complete. An initial solution is picked from a range of alternatives and then continuously refined into a better solution. The collection, cataloging, and rapid distribution of information is essential to this evolutionary process.

**STSC FUNCTIONS**

To achieve the objectives of the program the STSC will carry out eight functions. They are: (1) institutional interfaces, (2) information exchange, (3) consolidated planning, (4) development projects, (5) project advocacy, (6) test and qualify products, (7) field support, and (8) configuration tracking.

Interfaces are required to maintain an awareness of software technology development activities and products available and in development. The STSC will glean information, technology and products from existing programs. The objective is to identify and transition new technologies to the software support process. Where gaps exist or users needs are not addressed, the STSC will encourage developers and acquisition programs to improve the software tools, methods, and environments.

Institutional interfaces will be maintained with participating, operating, and support commands in the Air Force, development labs, major programs, industry, academia, DoD, and other government agencies. The STSC will accomplish this task by participating in conferences, working groups, professional societies, and technical interchange meetings designed to improve software engineering practices.

Information exchange is the primary mechanism by which the STSC will accomplish its mission. The STSC has created a number of products to act as a catalyst in the collection and distribution of information for the Air Force. Examples are: Product Book, Project Book, Software Engineering Handbook, Software Support Activity Guidebook, Tool Evaluation Catalogue, Crosstalk newsletter, Ada reuse library,
and the Software Technology Information System.

The Software Technology Information System was conceived to allow search of information by computer or personal contact. Users can link by modem using tool-free numbers to a computer database of all the products the STSC creates. The information system includes the Product Book, Project Book, Software Support Activity Guidebook, Software Engineering Handbook, etc. A demonstrated capability now exists in the Electronic Bulletin Board System that the STSC currently maintains.

The Project book is designed to be used by managers to become informed about development projects and avoid duplication of effort. Software technology development projects that have a link to MCCR will be catalogued. A description of the development project and goals will be included along with points of contact to obtain additional information. This book as well as other products created by the STSC will be continuously updated and be available in hard copy or through computer search.

The Product Book will catalog the available tools, methods, and environments currently available in the marketplace. It will be accessible by computer or in hard copy. A description of each product and its intended application will be given. Tools are an automation of a particular method with a specific part of the software process for its intended application. Tool applications will be addressed in the Product Book and their domain of application described. Software support environments vary in size and scope. A description of environments, their scope and application will be included in the Product Book. Points of contact for suppliers of products and users of products will be included. This book is intended as a resource to both developers and maintainers to obtain a quick summary of products, their intended use, and users' evaluations.

The Software Support Activity Guidebook is intended for use by managers to develop a detailed software support plan. The Software Engineering Handbook is used by Software Engineers as a source of techniques and standards for software development and maintenance. All information products are generated and maintained in the development project function of the STSC.

Much effort is wasted if there is no goal or plan for achieving that goal. What objectives are currently being sought by the developers of software tools, methods, and environments? Do their objectives meet users' needs? If we stumble through the brush in a purely random fashion we will eventually find the treasure, but will it be found in time? We need a software tool and environment philosophy, with goals, and a plan to implement. This should be a consolidated master plan which includes the needs of developers and maintainers. Such a plan does not exist and will save valuable time and resources once created.

The master plan should encourage the use of off-the-shelf software products where possible. Compatibility of products should be emphasized with the interfaces between tools well defined. The master plan may include several subplans such as an Ada implementation plan, Reusability plan, Tool compatibility plan, Software Support Environment plan, Resource Management plan, etc.

Development projects are designed to integrate existing tools, methods, and environments, and to create the information products supplied by the STSC. Specific items that need to be integrated are CASE design tools, documentation systems, rapid prototyping tools, configuration management tools, project management tools, test tools, hardware platforms, etc.
The information products defined in the information function will be developed and updated as software engineering practices mature. Product availability will begin this year and continue for the life of the STSC.

The STSC cannot test and certify all software tools, methods, and environments. Support from other organizations will be solicited. Participation of government agencies, academia, and industry is vital to the successful creation of compatible tools and environments. The products currently under development and evaluation are being done by support groups such as SAIC who is developing a reuse library, SEI who is integrating a documentation system into F-16 software maintenance, and the Air Force Avionics Laboratory who is working on a programmable test environment.

Tools, methods, and environments are tested and qualified in this STSC function. Testing and qualification procedures must be developed for this activity to occur. A software development philosophy with respect to tools and environments will provide the reference point for product evaluations. The objectives we have for tools in general and the specific features that are desirable for a given set of tools will be defined. A shopping list of desirable features for specific tool types will be created. Tools will be evaluated on the basis of compatibility. Does the output of an evaluated tool flow into the input of the next tool in the software process? What changes need to be made to improve the utility and compatibility of each evaluated tool. Finally what domains will a given tool support.

The STSC will provide support in the transition of technology that has been tested and qualified as supportable software technology. This support will include consultation and training for those who will use the products, documentation for the installation and use of qualified products, and a software support group for the resolution of problems in the field. Enhancements and upgrades to software products will be distributed to all users. This function is scheduled to begin in 1990.

Products supported by the STSC will be tracked for usability, functionality, productivity improvements, and reliability enhancements. This function is estimated to begin in 1990.

SUMMARY

Software is distilled human intelligence which exists as an ether in computer hardware. Its function, behavior, and structure is limited only by the imagination of those who create it and the capacity of the hardware which gives it physical reality. This non-dimensional abstraction is transparent to all human senses and its creation defies the boundaries of closed form solutions. Our appetite for this elusive elixir is increasing at an exponential rate and is rapidly becoming the most complex and expensive human endeavor.

A similar explosion has occurred in the hardware world with the rapid expansion of stand-alone personal computers. The people who stimulated this explosion were called "hackers". This was a term used to describe people who were obsessed with expanding the use of computers. The term existed before the advent of protected code breakers who are also referred to as hackers. A key element to their philosophy which contributed to their success was the belief that all information should be free. It is the goal of the STSC to adopt their philosophy of information sharing to solve the software crisis. The STSC will stimulate the free flow of information were it is reasonable and does not jeopardize the developers need for...
protection. Not all information can be made public, but a status of existing and developing capabilities can be maintained. Weak areas can also be identified so that developers can learn users needs. Essential to this process is the united development of a Software Technology Master Plan. A roadmap of where the technology should proceed. Such a plan requires the involvement of all players in the software development and maintenance process.

The STSC offers products and services to stimulate the exchange of information and the evaluation of methods, tools, and environments to support the software development or maintenance process. If the STSC were to stimulate a modest 10% improvement in productivity, the result would be a savings of more than 3 billion dollars per year.