Productivity - The Role of the Tools Group

Rudy Bazelmans

ABSTRACT

This paper addresses the issue of productivity in the software industry. It discusses the expected benefits of software tools and techniques. It describes the role of the tools group in this regard and, finally, discusses the characteristics of members of a tools group.

1. Introduction

Over the years, there has been a general lack of appreciation within the industry for the value of software tools and techniques. The solution to project overruns has typically been to reduce functionality, require overtime, add additional people, reduce the amount of documentation produced, emphasize coding activities, reduce testing, and apply stricter management controls. The blame for the disasters which are created in this process is usually attributed to poor testing, poor quality assurance, the lack of maintenance personnel and poor development personnel.

Rather than fix problems after they have arisen, it seems more appropriate to provide the tools and techniques beforehand which place the quality into the product from the beginning. This paper addresses the issue of productivity in the software industry and discusses the expected benefits of software tools and techniques. It describes the role of the tools group in this regard and, finally, discusses the characteristics of members of a tools group. The recommendations made in this paper are based on the experiences of the author.

2. Some Ways to Increase Productivity

Before discussing what a tools group should do to help productivity, let us survey the literature for insight into the software productivity problem.

It has been projected by Boehm, in his book entitled *Software Engineering Economics*, that by 1985 the cost of producing a computer system will be 90% software and 10% hardware [BOE 81]. Of the software costs, about half of that effort will be devoted to maintenance rather than development. From these figures it is obvious that the way to reduce costs is to improve the development productivity and reduce the amount of maintenance being done.

Boehm's research on software cost estimation has produced a cost model for software projects called COCOMO (COnstructive COst MOdel). Many practitioners believe that this model is the best technique available for forecasting software costs. Boehm's model uses 14 cost drivers to estimate the cost of projects:

- Language Experience (1.20)
- Schedule Constraints (1.23)
- Data Base Size (1.23)
- Turnaround Time (1.32)
- Virtual Machine Experience (1.34)
- Virtual Machine Volatility (1.49)
- Software Tools (1.49)
- Modern Programming Practices (1.51)
- Storage Constraints (1.56)
- Applications Experience (1.57)
- Timing Constraints (1.66)
- Required Reliability (1.87)

- Product Complexity (2.36)
- Personnel/Team Capability (4.18)

These cost drivers are listed in increasing order of impact. The numbers in parentheses indicate the relative weight of the cost drivers. Notice that the Personnel/Team Capabilities are almost three times as important as the other cost drivers. Of particular interest in this paper is the impact that a tools group can have on productivity. The quality and quantity of software tools and modern programming practices are ranked in Boehm's model on a scale which includes Very Low, Low, Nominal, High and Very High. Using the COCOMO model it can be shown that, if an organization can improve the tools and techniques used by two levels (i.e., from low to high), the difference in productivity would be over 40% (20% for the Software Tools and 24% for the Modern Programming Practices). More improvements can be achieved if the improvements exceed two levels. Boehm's model shows a direct correlation between productivity and the tools and techniques being used.

Boehm also says that with the use of modern programming practices, "most installations can increase their software development and maintenance productivity by an additional 100% in three to four years, and by an additional 400% in six to eight years". Besides modern programming practices, other areas which can be addressed by a tools group are the experience areas (Language, Virtual Machine, Applications). The experience of existing personnel can be improved by training and education.

In the book, *The Mythical Man-Month*, Brooks says that the chief programmer needs an understudy, an administrator, secretarial help, an editor, a program clerk, a toolsmith, a tester and a language expert. Brooks likens the programming team to a surgical team. Brooks later states, "The manager of a project, then, needs to establish a philosophy and set aside resources for the building of common tools. At the same time the manager must recognize the need for specialized tools, and not begrudge his working teams their own tools building" [BRO 75]. Brooks identified this need for development support ten years ago!

The computer industry in the Republic of China (ROC) has recognized a problem with productivity. In order to improve their situation, a number of universities in the ROC have banded together to develop an integrated development environment. They are also working on the problem of transferring technology from the research and academic environments to the industrial environment [YAN 83].

The General Accounting Office has observed a need for additional development support. In a study on methods of increasing software development productivity, the National Bureau of Standards found that development could be reduced 11 to 30 percent over a four year period by introducing software tools and techniques [HEC 82]. There are other benefits to improved tools and techniques; these include increased motivation, job satisfaction and a feeling that management really cares about how enjoyable the job is. Although these benefits make intuitive sense, they are difficult to quantify.

Knowing that tools, techniques, and training are important, how many resources should be allocated to this task? The Japanese have for some time, been recognized for their ability to optimize the manufacture of products; software is no exception. The Fujitsu software factories have been noted for their high software productivity. Fujitsu spends around 10% of their budget on tools development [KOB 84].

The ability to improve productivity is based, in part, on the support of management. This support can come in several ways: the encouragement to try something new, the willingness to provide resources (both financial and human) and if necessary to make edicts. Boehm went as far as to say, "If managers do not genuinely want improved productivity, the organization will not get software productivity" [BOE 81].

In a 1984 study of the development environments 25 US and Japanese companies, Zelkowitz et al, identified several major reasons for the lack of tools support within industry: 1) The hardware engineering background of most managers causes them to unsympathetic to the need for software tools, 2) Most corporations lack an organization whose charter it is to evaluate, select and develop tools, 3) The lack of reuse of tools, 4) The abundance of incomplete or poorly documented tools [ZEL 84]. Let us now discuss how the tools group can help increase productivity.

3. The Charter for a Tools Group

In an attempt to combat the high cost of software, the shortage of good software engineers and the poor quality of software products, it is necessary for corporations to address the broad issue of productivity. The dictionary defines a tool as "something that aids in the accomplishment of a task". This definition is much broader than the common industry definition in which a tool is a program used by developers. Although programs are a great help in increasing productivity, there are numerous other tools which also yield great benefits.

Since the activities of the tools group are highly integrated, they become extremely difficult to categorize. For the purposes of this discussion, the responsibilities have been broken down as follows:

- Provide software tools.
- Write and maintain documentation for software tools.
- Investigate and transfer new technologies.
- Establish and support development standards.
- Communicate technical information.
- Provide support for hardware and software purchases.
- Provide technical support to the department.
- Catalog software available within the department.
- Collect publications related to department development.
- Distribute documentation.
- Administer the Development computer.

One question which comes up immediately is, "Who is the customer?" Is it the corporation, a department, or is it a particular project or group? There are several considerations in this regard. If the customer base is large (whether the entire corporation, a large department or a large project) the following advantages exist for having a large tools group:

- There is economy of scale.
- The tools group can handle problems more easily because of the flexibility inherent in larger groups.

On the other side, if the customer base consists of a single department or project with its own tools group, the following advantages exist:

- The tools group is considered a part of the team.
- The tools group has a better idea of the needs of the group.
- The tools group can focus on fewer needs (those of a single department or project).
- Prioritization of tools work is more easily handled since there are fewer requests.

It is preferable to have a tools group dedicated to a particular department or project. The optimal size of the customer base seems to be between 100 and 200 developers. An exception would be users with similar development environments and similar needs; these groups can easily be larger. In this paper, the term "department" is used to describe the collective customer base. The reason for referring to the other department members as "customers" is simple: it emphasizes the fact that the tools group provides a service. The tools group is really a microcosm of a software corporation since it does market research, product planning, software development, technical writing, sales and support. Let us take a closer look at each of the tasks of the tools group.

3.1. Provide Software Tools

The first step in providing tools is investigating the needs of the department. The ideas for tools are identified in numerous ways:

- Listening for ideas which surface.
- Soliciting suggestions from department members.
- Recognizing your own needs in the course of developing tools for others.
- Analyzing the complaints of the department members.
- Observing others in the course of their work.
- Collecting and analyzing data on the development environment
- Reading books and magazines to see what others in the industry have identified.
- Talking to outside organizations.

After identifying the needs of the department, potential solutions must be sought. This includes investigating the procedures and tools used by other departments in the company and in the industry, as well as investigating the commercial products available.

The tradeoffs between purchasing, modifying and building the tools should be carefully considered. Tools should only be built if there is an overwhelming advantage over purchasing a new tool or modifying an existing one. Software Engineers are often too eager to build yet another software tool.

After the software has been purchased, modified or written, it should be thoroughly tested and then distributed to a limited number of interested users. By introducing the tool to a limited number of users prior to mass distribution, you develop a network of experienced users and potential salesmen; this greatly aids in the infusion of new tools and techniques into the department. These initial users should be given the maximum amount of support possible and their ideas should be included whenever possible.

After successful use by the tools group and several users, the new tool should be introduced to the entire department. This introduction should be done with the aid of reference manuals and short classes on its use (one hour is good). These classes should be as short as practical in order to encourage attendance. These classes are important since they introduce the fundamentals, outline the advanced features, generate the excitement necessary to attract initial use and increase the general awareness of available tools. The initial use by the department is the pivotal point for success or failure of the tool. It is imperative that the tools group provide maximum support during this introduction.

Follow-up analysis should be performed to insure that users are taking full advantage of the tool and that the tool provides the most functionality possible.

The types of software written to enhance productivity are limited only by your imagination. Some of the categories include the following:

- Project management tools, like scheduling, cost estimations, and spreadsheets.
- Office automation tools, such as word processing, calendars, and electronic mail.
- Tools to aid in testing and debugging.
- File modification tools such, as text editors and data base editors.
- Code generation tools, like application generators, and compilers.
- Static and dynamic analyzers.
- Configuration management and bug tracking tools.
- Tools to aid and check the adherence to standards.

If any of these tools are a commercial product of the corporation it may be wise to establish a dedicated application group for the tool. Tools group members are specialists in software tools and not typically specialists in specific software products.

An advantage of having a tools group develop tools, rather than having individuals within the department write ad-hoc tools is that the tools group has the resources to develop larger tools and as a group, they can more easily write well integrated and consistent tools.

3.2. Write and Maintain Documentation for Software Tools

Although software tools are *necessary* for the development of software, a mere set of tools is not *sufficient* to guarantee success [MAR 85]. Software tools need good documentation.

Documentation should be written and maintained for the operating system, software tools, departmental procedures and standards. The types of documentation can include tutorials, reference manuals, summary sheets and others. The documents should be written for the audience; software developers most frequently refer to the documentation when they have a problem. This predominant use should guide the organization of the documentation.

Not only should documentation be written for locally produced tools and procedures but imported tools should also be documented if the documentation is found to be inadequate. The documentation should be well written and up to date. Well written documentation pays for itself through reduced time spent by developers finding answers to questions. Well written does not necessarily mean that it meets technical writing quality standards, but that it is concise, well organized and grammatically correct; in other words, it is functional.

3.3. Investigate and Transfer New Technologies

The ability to apply new hardware and software technologies to the development of software is essential to remain competitive within the industry. These technologies include not only new programs but also new methods and concepts. These technologies should be carefully evaluated for their expected benefits and expected costs.

The usefulness of a technology is strongly influenced by the availability of tools (both hardware and software) to support it. These tools should be investigated and purchased, modified or built. Among the technologies which exist today are: hardware and software emulation, debugging aids, configuration management techniques, high level programming languages, structured design methods, testing techniques and many others.

The accurate and widespread transfer of these technologies is essential to the success of the technologies. Mini seminars and classes on selected tools and techniques provide a useful transfer mechanism. These seminars and classes must be short, well prepared and fast paced in order to maintain interest among the more advanced users in the department. By having the classes on a regular basis and allowing any department member to present their ideas or share their experiences, the classes will become an institution. Demonstrations of new tools (whether internally or externally developed) is also effective in sparking interest. Video taping the seminars and demonstrations is probably worthwhile for new members to the department and for others who could not attend the class. Be aware that when a class is going to taped, the instructor will naturally spend additional time polishing up the presentation. This can be considered both good and bad.

Another method of transferring technology is to encourage individuals to write internal papers on concepts, procedures and methodologies in which the individuals have gained some useful insight. These papers could be collected, cataloged and distributed by the tools group. The administrative support for this activity could also come from the corporate level.

Of course the tools group should always be supportive when asked to consult on a particular topic, or tool. User questions and problems should also be given high priority.

3.4. Establish and Support Development Standards

In a paper entitled *Standards Can Help Us*, Gordon Bell wrote that "standards are constraints that ensure the evolution - not revolution - of computing" [BEL 84]. Standards should be developed in order to increase productivity. Standards development also plays a key role in the reusability of software components, which is known to greatly influence productivity [KOB 84].

All standards which are developed should be as simple and pragmatic as possible. Whenever possible, standards should be supported by tools and the enforcement automated. Among these tools are comment header templates with interactive entry, static analyzers to check adherence to documentation and programming standards, libraries of routines and standard include files. If tools are available to aid in the adherence to standards, they are more likely to be used.

The need for standards must be recognized before standards can be established. Then, it must be shown that the established standards address those needs. A group of active department members should develop and maintain the standards. The membership should be based on knowledge, availability, and interest. Members should be knowledgeable of the development methods used and be respected by their peers.

3.5. Communicate Technical Information

The communication flow between the tools group and the rest of the department is essential to the success of the department. In addition to the informal methods of communicating which were described in other parts of this paper, there are numerous effective, more formal methods of communicating, such as:

- Department Status Meetings
- Tools Group Guild Meetings
- Software Tools Advisory Committee Meetings
- Tools Group Newsletter
- New Employee Orientation

The tools group (or the leader of the tools group) should be invited to all department status meetings. By attending these meetings the tools group can stay in touch with the activities of the department, be forewarned of problems, get involved in solutions, and provide support. The other members of the department also become more aware of the activities of the tools group.

Tools Group Guild meetings involve representatives from the other tools groups within the company, collectively discussing their activities. These meetings serve as a vehicle for the exchange of tools and ideas among the various groups. Sharing tools, procedures and ideas between the groups greatly reduces the corporate cost of tools. These meetings should probably be held once a month.

Software Tools Advisory Committee meetings involve the tools group leader and representatives of the other groups within the department. The representatives should be non-managers who volunteer to meet on a regular basis to discuss issues of productivity. These issues typically include discussions of appropriate features in new and existing tools, issues of standards and numerous other user preference issues. The representatives are expected to survey users in their respective groups and to represent their views in these meetings. Weekly meetings seem to work well, since they provide the frequency necessary to address the issues which regularly arise. Besides the benefits of better representation of decisions made by the tools group, these meetings provide the tools group with a set of department members who appreciate the difficulties in many of these decisions, and will support ideas when they are made public. The concept of a Software Tools Advisory Committee is similar to the concept of a Quality Circle.

The Tools Group Newsletter is a medium for the dissemination of timely information to the department. These newsletters contain such items as the announcement of new tools and features, changes to the development system, hot new tools projects, upcoming events and new hardware, software and documentation acquisitions. The newsletter should be no more than one page in length in order to encourage department members to read it.

The goal is succinct, useful information published every few weeks, as needed. Besides the benefit of communicating information to the department, this mechanism provides visibility which is sorely needed by most tools groups. These newsletters also provide a useful history of events for new department members. It be useful to expand the content of the newsletter to include department news items, thus making it a department newsletter.

Besides the ongoing forms of communication mentioned above, one extremely useful mechanism is a departmental new employee orientation. The tools group is in an ideal position to give new employees a perspective on the department, tell them about the available services within the department and introduce them to the key department members. This would bridge the gap between corporate orientation programs and real life.

3.6. Provide Support for Hardware and Software Purchases

The tools group can provide administrative, technical and initial legal support for hardware and software purchased by the department. The technical support includes the investigation of available products, the evaluation of the feasibility and appropriateness of the product and initial installation and testing of the product. The administrative tasks include helping develop the justification for a product and interfacing with Purchasing, the vendor, and the customer. Minimal legal/technical support can be provided to the Legal department.

By supporting the purchase of products within the department, the department benefits from the technical and purchasing expertise of the tools group and their knowledge of the activities with the company and the industry. The tools group can also free up the developers for other tasks. The tools group benefits because it is further involved with the needs of the department.

3.7. Provide Technical Support to the Department

The tools group should provide technical support for the department whenever necessary. This includes finding information, tracking down a bug in a program or the operating system, solving a programming problem, demonstrating the use of a tool and locating a tool.

These activities should be analyzed, since they point out problems in programs, documentation and procedures which are in place. There is usually a large list of new programs to be written, bugs to be corrected, documentation to be improved and enhancements to be made.

The tools group should also be available if consulting is requested. Some organizations may also find it helpful to hire a part-time consultant to discuss ideas, solve problems and teach seminars. The consulting personnel should be coordinated by the tools group.

3.8. Catalog Software Available Within the Department

There are usually a large number of useful programs which have been written within the department but are not generally available. One way of making these programs available is to develop a Software Catalog. The purpose of this catalog is to list all the software available within the department and to cross-reference the entries, in order to aid in the location of software for a particular purpose. A DBMS/forms entry package should be utilized so that anyone in the department can add entries or query the database.

Each entry in the catalog should contain the following information:

- The name of the program or system of programs.
- The version of the software.
- The date of submission.
- The name of the author of the program (optional).
- Indication of whether the program is supported or not (tools group programs are always supported!).
- The host hardware and operating system.
- The source language.
- Any hardware requirements.
- Whether sources are available.
- Whether documentation is included.
- A brief description of what the software does.
- A list of pertinent file names.
- References to other programs of interest or of related purpose.
- Instructions on how to get the software and documentation.

The entries included in the catalog should do not include the programs which are normally included with the manufacture's hardware (unless resources permit) since these programs are already documented and cataloged. In an informal survey of two companies which maintain these catalogs, over 1,000 internal programs were identified. These tools ranged in size from one staff-hour to multiple staff-month projects. The ratio of collected tools to engineers seems to be greater than 1:1. The number of uncollected tools is unknown.

Everyone should be encouraged to improve an existing program and re-release it. The tools group will distribute the software (if not available on-line) as requested. The tools group will often be asked to support one of the programs by fixing, enhancing, re-writing or documenting a particular program.

3.9. Collect Publications Related to Department Development

There are numerous books, magazines and other publications which are of direct interest to the development department. These could include the following items:

- Language and operating system reference manuals.
- Tutorials, reference manuals and cards on software used within the department.
- Video and audio tapes of seminars help along with copies of the overhead slides and handouts.
- Books and magazines of particular interest to the department.
- Copies of internally distributed memos and documentation.
- Documentation on hardware in use within the department.
- Hardware and Software sales literature related to competitive products and potential purchases.
- Catalogs of available software within the industry.

The last three categories listed are primarily maintained for use by the tools group when evaluating and designing tools. All of the materials should be made available for reference, loan, and in some cases, permanent use. This library of materials should not compete with the corporate library (if one exists); its goal is specific to the needs of a particular department.

Internal memos should be organized into categories and assigned memo series numbers within each category. Doing so makes locating and referencing internal memos easy. If the storage and distribution of these memo series are centralized in the department, it is easy for new and old employees to obtain the information they need. Individual groups may also find it useful to maintain group memo series for the same reasons.

Hard-copy documentation should be available for reference or for personal use (if the user prefers), summary pages and/or reference cards should be readily available and, in some cases, tutorial documentation should be made available. Engineers find reference cards and locally written tutorials extremely useful. All documentation should also be available on-line if practical.

Another useful tool is a documentation roadmap. This roadmap should be an on-line database or a hard-copy document which describes what documentation is available and where it is located. The fields of the roadmap are similar to the fields used in the Software Catalog described previously and are used to aid in information retrieval:

- Brand name of the item being documented.
- Version number of the documentation (or item if applicable).
- Keywords which describe the document.
- Operating system or harware in which the document applies.
- Type of document, e.g., tutorial, reference card, reference manual.
- Organization that generated the document (or product).
- Name of the author.
- Location of document, e.g., on-line (where), library (which one).

This roadmap can also be used to help locate internal documents such as memos, newsletters and magazines. This could be done by adding fields such as project, and department number.

3.10. Distribute Documentation

There should always be well written and up to date documentation available for members of the department. The documentation system should be simple and well organized. If the system is too complicated, ill-documented or time-consuming to use it is less useful.

The documentation which is necessary to do productive work within the department should be organized so that it can easily be distributed. One convenient method of distribution is to group the documentation on the third party and local tools in three-ring binders with the names on indexes. Depending on the volume of material, these binders can be split and grouped by host processor, local versus third party tools, or even by their domain of usefulness (documentation, programming, language etc).

The names of the individuals who have received documentation which is likely to be updated, should be recorded so that updates can be distributed in an organized manner. If updates are necessary, they should be performed by the librarian. The most cost effective and efficient method of updating documentation is to have the librarian update everyone's documentation personally and give each individual a memo describing the improvements which have been made. Programmers will rarely update manuals and if the manuals become out of date, they become useless.

Hard-copy documentation should be freely given to everyone who is interested. Some users prefer hard-copy over on-line documentation. If users have a copy in their office it encourages them to learn more about the tools at their disposal while waiting for compiles, printouts, etc.

3.11. Administer the Development Computer

The tools group should provide technical support to the operations group. The operations group is typically responsible for the daily activities of backups, archiving, hardware installation, repair, preventive maintenance, new user additions and removal of old users. As active developers and members of the project team, the tools group should provide guidance in system tuning, system maintenance, software installations, tools support, technical training and assistance whenever appropriate.

Because of the perspective of the tools group, they should represent the needs of department to the operations group. If the department can't justify an operations group, the operations responsibility should be assumed by the tools group.

4. The Structure of the Tools Group

Given the functions which have been discussed in the previous sections, it is necessary to distribute these activities among the members of the tools group. The ideal tools group would be made up of four *types* of individuals: a group leader, a technology specialist, a toolsmith and a software librarian. The experience requirements, skill requirements and responsibilities of these roles are outlined below.

The number of individuals in each role is based on the number of people in the department being served, the variety of development and target environments in use and the diversity of applications being developed within the department. The most likely requirement for additional personnel will be in the toolsmith role but the need for additional librarians may also become apparent. The additional personnel need not have the same level of experience as the core individuals.

Additional personnel beyond the core group can come from temporarily idle applications personnel. These people are great for helping with the endless list of tasks that need to be done. By using these people, the tools group gets additional respect within the department since more individuals have done work for the tools group. The tools group leader should be careful not to give all the lucrative projects to the temporary personnel because the permanent personnel will get upset.

The key to running the tools group is in cross-training the individuals whenever possible. This includes not only cross training the toolsmiths but also cross-training the toolsmiths with the technology specialist. The job descriptions are flexible enough to support this breadth of activities. By cross-training the individuals within the department, the group is more flexible and thus can handle more varied requirements as imposed by the department. Group reviews of projects within the tools group should be encouraged.

4.1. Tools Group Leader

The leader of the tools group should provide the group with direction and provide technical input into the activities of the members of the group. Since there is (intentional) overlap between the responsibilities of the toolsmith and the technology specialist, the group leader must set priorities and assign tasks. The group leader should have the following qualities:

Experience:

- 5 years experience in software development
- 2-3 years of group leadership experience (tools preferred)
- Bachelors or Masters degree in Software Engineering or equivalent

Skills:

- Proven technical skills
- Strong inter-personal skills
- Adept at public speaking
- Proven leadership experience

Tasks:

- Prioritize and assign the work within the group
- Supervise the work of others within the group
- Report to upper management
- Publish the tools group newsletter
- Attend project status meetings
- Make technical contributions

4.2. Technology Specialist

This individual is primarily interested in infusing new technologies into the department. One strong benefit of this role is that the output is quite portable since it usually consists of procedures, methodologies and ideas. It not uncommon for porting efforts take as little as 1/25th of the original development effort. Besides the enormous salary benefits of faster development, there is a substantial indirect benefit to the new project since the technique is usable much sooner on the new project.

The qualities of the technology specialist are as follows:

Experience:

- 5 years of software development experience
- Masters degree in Software Engineering or equivalent
- Broad exposure to software engineering methodologies
- Group leadership experience is a plus

Skills:

- Demonstrated technical abilities
- Work well with others
- Leadership skills
- Open minded
- Strong writing and presentation skills
- Well formed opinions
- Confident personality

Tasks:

- Develop and maintain internal development standards and methodologies
- Develop and acquire software tools to support standards and methodologies
- Train department members in the use of standards and methodologies
- Analyze productivity within the department

4.3. Toolsmith

The project toolsmiths are responsible for the acquisition and development of software tools for the department. Toolsmiths need the following qualifications:

Experience:

- 5 years experience in software development
- 2-3 years experience in developing software tools
- Bachelors degree in Computer Science or equivalent

Skills:

- Work well with others
- Good writing skills
- Creative and strong problem solving skills

Tasks:

- Develop, evaluate and acquire software tools
- Document tools and procedures
- Train others in the use of tools
- Provide user support

4.4. Software Librarian

The tools group typically has a large number of low level technical and clerical responsibilities. If the department secretaries are utilized, the volume of work will likely overload the secretaries causing animosity between the secretarial help and the tools group. The result is that the engineers in the tools group spend a large amount of their time on work that should delegated to non-engineers.

The librarian's role is to perform those low-level technical and clerical tasks which have to be done; thus off-loading the engineers so they can do other work. The amount of support provided by the secretarial staff is virtually eliminated.

Experience:

- 2-3 years clerical/secretarial experience (librarian experience preferred)

Skills:

- Well organized
- Some technical interests

Tasks:

- Organize and maintain software library
- Organize and distribute documentation
- Maintain software catalog and distribute related software
- Distribute software developed and acquired within the group
- Provide clerical support for the group
- Provide new employee orientation

5. Final Comments

In order to increase the success of our companies during the software shakeout of the next few years; we need to pay more attention to the productivity of your software developers. Future success will depend on the use of software engineering methodologies, educated software engineers, and experienced software managers. If properly supported and run, the tools group can be a big help.

6. Acknowledgements

I would like to thank Kyle Geiger, Bob Pellegrino, Dave Ressler, and Gail McCarthy for their helpful evaluations of my paper. Dr Susan Gerhart and Dr Al Thompson of the Wang Institute also helped by allowing me to write this paper for my course work and by giving me feedback. I would also like to thank my wife, Dawn, for patiently waiting for me to complete yet another "important" project.

7. About the Author

The author is currently a member of a tools group at Wang Laboratories. Previously, he was the supervisor of the corporate tools group at Sykes Datatronics.

The author holds a Bachelors degree in System Software from the Rochester Institute of Technology and is pursuing a Masters degree in Software Engineering at the Wang Institute of Graduate Studies. He has written several articles on software tools and techniques.

You can contact the author at Wang Laboratories, One Industrial Avenue, M/S 1989, Lowell Massachusetts, 01851, by phone at (617)967-2609 or on UUCP at decvax!wanginst!bazelman.

References

[BEL 84] Bell, C. Gordon.
Standards Can Help Us.
IEEE Computer , June, 1984.

[BOE 81] Boehm, Barry W.
Software Engineering Economics.
Prentice-Hall, 1981.

[BRO 75] Brooks, Frederick P.
The Mythical Man-Month.
Addison-Wesley, 1975.

[HEC 82] Hecht, Herbert & Houghton, Raymond C.
The Current Status of Software Tool Usage.
In *IEEE Computer Society's 6th International Computer Software and Applications Conference*, pages 1-8. IEEE, 1982.

[KOB 84] Kobayashi, Kaname.
Fujitsu Software Factory Approach.
In *The Small Computer (R)evolution, Proceedings of COMPCON 84*, pages 132. IEEE, 1984.

[MAN 83] Manley, John H.
Report of the Group on Technology Transfer Process and Vechicle.
In *IEEE CS Workshop of Software Engineering Technology Transfer*, pages 9-10. IEEE, April, 1983.

[MAR 85] Martin, Nancy, Ligett, Dan and Kirby, James.
The Wang Institute Software Environment.
Technical Report, The Wang Institute of Graduate Studies, March, 1985.

[MAT 82] Matsumoto, Yoshihiro.
Software Education in an Industry.
In *IEEE Computer Society's 6th International Computer Software and Applications Conference*, pages 92-94. IEEE, 1982.

[PHI 84] Phillip, Howard C.
Software Tools Improve Programming Productivity.
Small Systems World :24-27, May, 1984.

[TAY 83] Taylor, Bruce J.
Patterns of Technology Transfer in a Development Group.
In *IEEE CS Workshop of Software Engineering Technology Transfer*, pages 94-98. IEEE, April, 1983.

[YAN 83] Yang, Chen-Chau.
A Status Report on the Development of Software Tools in the Republic of China.
In *Productivity in the Information Age, Proceedings of the 46th ASIS Annual Meeting*, pages 157-160. IEEE, 1983.

[ZEL 84] Zelkowitz, Marvin V., Yeh, Raymond T., Hamlet, Richard G., Gannon, John D., and Basili, Victor R.
Software Engineering Practices in the US and Japan.
IEEE Computer , June, 1984.