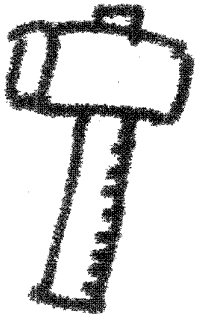


The Impact of Tools on Software Productivity

TILMANN BRUCKHAUS, McGill University
NAZIM H. MADHAVJI, McGill University
INGRID JANSSEN, IBM Canada
JOHN HENSHAW, IBM Canada

It's common knowledge that to stay competitive, your software organization must continuously improve product quality and customer satisfaction, as well as lower software development costs and shorten delivery time—all critical variables for software engineering practice.[1] You can pursue these challenges in many ways. One such way is to improve your development processes and organizational structures using frameworks such as the SEI's Capability Maturity Model,[2] Total Quality Management,[3] the Quality Improvement Paradigm,[4] or the Process Cycle.[5] Another option is to adopt appropriate software tools. Tools have long been recognized as an effective way to improve software development variables such as productivity and product quality. However, to make effective use of specific tools you should first understand how a tool will affect these critical variables in your project.

0740-7459/96/$05.00 © 1996 IEEE

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

TABLE 1
THE IMPACT OF A TOOL ON PRODUCTIVITY OF PROJECTS USING THE ADVANCED PROCESS

Fixed process (*advanced process*)–Varying project size (*5 features and 80 features*)

Requirements planning process activities	Small project: *5 features*						Large project: *80 features*					
	without tool			with tool			without tool			with tool		
	unit effort	execu-tions	total effort	unit effort	execu-tions	total effort	unit effort	execu-tions	total effort	unit effort	execu-tions	total effort
Capture raw requirements	53	12	636	13.5	12	162	248	12	2976	36	12	432
Resolve raw requirements	6.8	23	156	13.2	23	303	6.8	5141	34958	13.2	5141	67861
Resolve problem statement	10.9	11	120	3.2	11	35	10.9	716	7804	3.2	716	2291
Resolve features	97.4	5	487	37.7	5	188	97.4	80	7792	37.7	80	3016
Develop release plan	90	2	180	34	2	68	1440	2	2880	544	2	1088
Total effort			1579			757			56411			74688
Productivity*	7.6			15.8			3.4			2.6		
Productivity Impact**	+ 107.9 %						- 23.5 %					

* measured in number of features that can be processed in one 5-day week (2,400 minutes):

$$\text{Productivity} = \frac{\text{Number of Features in Project}}{\text{Total effort in minutes}} \times \frac{\text{2,400 Minutes}}{\text{Week}}$$

** the change in productivity as a percentage of the original productivity

Tools can help improve your development processes by facilitating activities you didn't practice before. For example, a testing tool can help introduce new testing activities such as branch-coverage analysis. Tools can also help increase productivity by supporting software development activities that are usually carried out with little or no tool support. However, in certain situations, the introduction of a tool can also decrease your productivity. This can happen when a tool increases your effort on specific activities or introduces new activities into the process, such as generating and maintaining new data.

Because we don't know how to analyze a tool's impact on specific projects,[6,7] we generally adopt them based on an intuitive understanding of their expected impact. In many cases, the actual results of this practice are disappointing.[8] The problem is aggravated because tool adoption often brings considerable costs; in addition to acquisition costs, there are many other related expenses that can amount to 5 to 10 times the price of the tool itself.[9]

BEYOND INTUITION

The industry has made great strides in making more development tools available. It's now time to find ways to consistently, objectively evaluate a tool's utility and appropriateness.

—*Theme of* IEEE Software's *May 1992 special issue on tools assessment*

Four years have passed since Elliot Chikofsky made this call for consistent and objective tool evaluation methods. During that time we have been involved in a collaborative research effort in software productivity between McGill University and IBM Software Solutions Toronto Laboratory. As part of that research, we did a case study on the impact of tool insertion in ongoing software projects. The result of our case study was a method that organizations can use to assess the impact of tool insertion on software productivity.[10,11]

We wanted to find out how using specific requirements-management tools affected the productivity of requirements planners in several projects. The projects were of different sizes and used different processes. Our goal was to see which projects would benefit from tool insertion.

Our results show that, depending on project characteristics, the same tool can have vastly different effects on productivity. In one case, due only to differences in project size, the tool's impact ranged from a productivity decrease of more than 80 percent to a productivity increase of almost 75 percent. In another case, differences in the development process alone resulted in a productivity decrease of more than 80 percent on one project and an increase of almost 600 percent on another.

There were other surprising results as well. For example, a tool's performance can peak when you use it on projects of a particular size, but it can be less productive in larger as well as smaller projects. Also, depending on the complexity of the development process, using a tool may require more effort than working with a less sophisticated tool (or technology) such as operating-system commands or simply paper and pen. Further, we found that when tools are used to support a rather simple development process, they tend to be more appropriate for small projects.

Our results indicate that to increase productivity, you should select a tool with your project size and development process in mind. Ignoring these factors can lead to higher software costs and slower time to market.

CHOOSING THE RIGHT TOOL

A tool's impact is not solely governed by its inherent properties, but also by the characteristics of the adopting project. Two characteristics—project size and development activities—are particularly important because they govern which tool functions you use and how often you use them. Thus, a tool's effect on productivity will vary depending upon the specifics of your project.

Tables 1 and 2 show two examples from our case study. Both examples

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

TABLE 2
THE IMPACT OF A TOOL ON TWO PROJECTS USING DIFFERENT DEVELOPMENT PROCESSES

Fixed project size (*10 features*)–Varying process (*regular process* and *advanced process*)

Requirements planning process activities	Simple process: *regular process*						Complex process: *advanced process*					
	without tool			with tool			without tool			with tool		
	unit effort	executions	total effort	unit effort	executions	total effort	unit effort	executions	total effort	unit effort	executions	total effort
Capture raw requirement	46	12	552	9	12	108	66	12	792	15	12	180
Resolve raw requirements	0.3	89	26.7	10.4	89	925.6	6.8	89	605.2	13.2	89	1174.8
Resolve problem statement	7	32	224	2.9	32	92.8	10.9	32	348.8	3.2	32	102.4
Resolve features	8.3	10	83	3.2	10	32	97.4	10	974	37.7	10	377
Develop release plan	0*	2	0	0	2	0	180	2	360	68	2	136
Total effort			855			1158.4			3080			1970.2
Productivity	28.04			20.71			7.79			12.18		
Productivity impact	- 26.1%						+ 56.4%					

* this activity does not involve subactivities affected by tool insertion.

contain data from the requirements-planning process we studied. This process has five subactivities, of which each subactivity also has subactivities, omitted here for simplicity.

The tables show effort and productivity data on two pairs of projects before and after tool insertion. Each table includes the effort required for one execution of an activity (*unit effort*), the number of executions of an activity throughout a release cycle (*executions*), the total effort required for all executions of an activity during one release cycle (*total effort*), and productivity and impact data.

Size. Table 1 shows data from two projects of different sizes that use the same development process. The data on the left is from a requirements-planning process for a small product where five features are added per release cycle. The data on the right is from a larger project, with about 80 added features per release. Clearly, for the larger project more data must be processed, and thus activities must be carried out more frequently. However, as the "executions" column shows, although the number of executions for the listed activities differs across the two projects, it is constant for each project before and after the tool insertion.

The effort for each individual execution of an activity ("unit effort") differs in each of the projects because of the tool insertion, as can be seen by comparing the "without tool" and "with tool" columns. Finally, the unit effort also differs from the small to the large project when comparing the respective "without tool" columns as well as when comparing the respective "with tool" columns. This is because the subactivities are carried out with varying frequency depending upon project size.

The difference in project size affects the impact of the tool: In the smaller project, productivity increased by more than 100 percent because of the tool insertion; in the larger project productivity declined by almost 25 percent.

It is tempting to think that a tool that improves productivity in a small project would lead to a similar—if not greater—increase in a larger, similar project. However, even though the tool introduction facilitates some activities, others may require more effort when the tool is used. In our example, more effort is required to resolve a raw requirement (13.2 versus 6.8 minutes) while all other activities require less effort. Thus, productivity will improve only if you can save more effort on the facilitated activities than you have to invest in activities that require more effort. This is the case in the smaller project in Table 1. If the activities that require more effort are carried out often in a project, as in the larger project in our example, then the tool may curtail productivity.

As our results show, project size can tip the scales of success or failure in tool adoption.

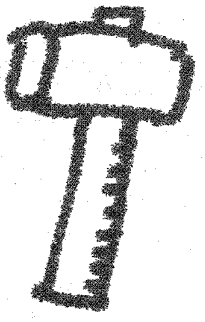
Process. Table 2 shows the importance of the development process in determining the effects of tool insertion. Here, both projects are of the same size (10 features) but use different development processes. The first project uses a simple process, regular process, while the second project uses advanced process, which is more complex. (The specific differences between these processes are in the subactivities which are not shown.)

In the project with the more complex process, you generally have to carry out a larger set of subactivities to accomplish the listed activities (see "unit effort"). However, in these projects, each of the listed activities is carried out with the same frequency (see "executions"). Here, due only to the different development processes, the effect of tool insertion on productivity is again drastically different. Whereas the project with the simpler process showed a productivity drop of more than 25 percent, the project with the more complex process showed an increase of more than 50 percent.

Choosing wisely. Our study gave us a wealth of data on which tool is right for which project. If your main goals are short time to market and low development cost, you can select the tool that promises the greatest productivity boost. When you select a tool for other reasons, you can make sure that using the tool does not lower productivity to the point where you cannot afford to use it. For example,

- a tool that is used in the upstream part of the development process may locally decrease productivity but help improve product quality or increase productivity downstream;
- an analysis tool may help you better understand and document software requirements;
- a reverse engineering tool may

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.



provide new information about a product, such as design abstraction, that simply did not exist prior to the insertion of the tool.

Can you improve both process and productivity by adopting the right tool or technology?

Such tools may also improve quality and reduce rework in your design and coding processes. To reap these benefits, it is important that you understand the tool's impact on productivity.

Regardless of which is more important in your project—quality or productivity—quantitative analysis of a tool's impact can help you decide if you should adopt a tool or pass it by.

CASE STUDY

We chose to do a case study rather than an experiment because we had no control over the industrial environment where we planned to investigate the impact of tool insertion on software productivity.[12] Also, because there is little quantitative data on the effect of tools on software productivity, our study was necessarily exploratory rather than aimed at replicating prior studies or probing existing theories.

Our hypothesis was that the size of a project and the development process practiced can influence the effect of tool insertion. Our goal was to test this hypothesis. We also wanted to know if improvement in process and productivity could be achieved at the same time. That is, we were interested in finding out if it was possible to adopt a more rigorous process (which typically requires more resources), and at the same time increase productivity by supporting that process with an appropriate tool.

Study specifics. Our unit of analysis was the insertion of a requirements-management tool into a requirements-planning process. The subunits of analysis were the alternative development processes, project sizes, and technologies, and their impact on software productivity. They thus constituted our three independent variables and software productivity was our dependent variable.

Our case study focused on the requirements-planning process of 17 ongoing projects. These projects were different with respect to their size and the processes and tools they used; we studied three different processes, five project sizes, and four tools.

- Of the three processes, one process was in use in some of the studied projects; management considered the other two for adoption after a new tool was inserted.
- The five project sizes represent the typical project sizes in our 17 sample projects.
- Of the four technologies, one—referred to as "OS"—was in use in some of the studied projects and one—simply called "Tool"—was being considered for adoption. The two remaining tools—"Prototype" and "Tool+"—were alternative versions of Tool; we studied these versions to better understand how different tool characteristics affect productivity.

For the analysis, we used the Software Productivity Analysis Method (SPAM)[10,11] which let us model different project sizes, development processes, and technologies separately. We then combined these partial models into productivity models, which we analyzed for effort requirements and productivity.

To develop our size, process, and technology models we used field studies consisting of participant observation, direct observation, document analysis, and various types of interviews.[12] To model processes and technologies not in use, we relied on field pilot studies and laboratory studies: professional planners and researchers used the new processes and tools for a limited time on actual project data.

We then used SPAM to combine the 12 (3 + 5 + 4) partial models into 60 (3 x 5 x 4) productivity models and to compute effort requirements and productivity data. Thus, we obtained data on 60 different ways of requirements planning. The effort and productivity figures were validated by the requirements planners and managers in the participating projects as well as by the individuals who had participated in the field pilots and laboratory studies. We are thus confident that the 60 datapoints we obtained are valid and useful to support decision making.

Context. The requirements-planning process determines which requirements to add in a product's future releases to best meet market needs. In this process, planners gather requirements from a variety of sources such as customers, user groups, advisory councils, competitive analyses, conferences, and literature. They then analyze raw requirements and create problem statements to describe different sets of related raw requirements in a more formal manner.

To address the most important problem statements, the planners then suggest new system features. The most valuable of these features will be incorporated into future product releases. In the organization we studied, requirements planning had been done using the AIX file and operating system utilities, and data was also captured by taking notes on paper or simply by remembering it.

The planning managers wanted a tool that would help requirements planners process, document, and access requirements information in a structured and convenient way. They also wanted a tool that would let the plan-

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

TABLE 3
NUMBER OF ACTIVITIES IN EACH PROCESS

Activity type	Process		
	Regular	Advanced	Complete
Documentation	5	12	15
Access	1	4	18
Sharing and QA	1	5	49
Subtotal	7	21	82
Other activities	27	33	65
Total activities	34	54	147

TABLE 4
NUMBER OF REQUIREMENTS-PLANNING ARTIFACTS IN PROJECTS OF DIFFERENT SIZES

Type of artifact	Project size in features				
	5	10	20	40	80
Raw requirements	23	89	344	1331	5141
Problem statements	11	32	89	253	716
Features	5	10	20	40	80

ners cooperatively build and maintain a comprehensive database of requirements information that could be used to make decisions about future product releases. Thus, the tool should also facilitate information sharing among all planners and across different phases of product development.

The organization preselected a requirements-management tool with integrated problem-tracking facilities. The tool facilitated capturing requirements data in a database, managing versions of requirements items, controlling the process of resolving requirements, reviewing requirements, querying for and communicating requirements information, and retaining relationships among different requirements items.

The company's main objectives were to render the requirements process more structured and to improve the quality of the deliverables. Yet, it was not clear whether using the tool would let them meet these improvement goals while maintaining a high level of productivity. A key concern was that the process changes could lead to many new activities, and thus more effort would be required to plan requirements. In this situation, the data from a quantitative productivity analysis was just what was needed.

Modeling the problem. To begin, we had to first model and analyze the industrial setting. The SPAM method allowed us to separately model the development process, project size, and technology. We then used SPAM's productivity evaluation algorithm to calculate the productivity of all possible combinations of process, project size, and technology.

The effort and productivity data we derived concerned only activities that were directly affected by tool insertion—planning activities that are related to documenting, accessing, and sharing requirements information. However, planners must expend substantial effort on activities that are not supported by the requirements-management tool, such as discussing product strategies and meeting customers and vendors. Thus, the data we obtained represents only a specific part of the overall effort required for planning requirements.

Processes. We expected—and intended—that inserting the requirements-management tool would change the process in three main areas:

- documentation of requirements information,
- accessing requirements information, and
- sharing requirements and quality assurance information.

The three alternative processes incorporated activities in these areas to varying degrees. Thus, each process was modeled separately as regular process, advanced process, and complete process.

- *Regular process*: planners document, access, and share requirements data to a minimal extent; a lightweight process to maximize productivity.
- *Advanced process:* represents an intermediate solution; all vital requirements information is documented, shared, and accessed, making it more rigorous than the regular process but requiring less effort than the complete process.
- *Complete process:* all requirements information is documented, accessed, and shared whenever it appears useful; designed to ensure maximum product quality regardless of the required effort.

Table 3 shows how many activities are included in different processes in the areas mentioned above (documentation, access, and sharing of requirements information).

Project sizes. Project size varied widely across projects. We measured project size by counting how many features planners expected to add to a product during one release cycle. To represent typical project sizes, we chose project sizes of 5, 10, 20, 40, and 80 committed features. We also specified the number of raw requirements as a measure of the input amount and modeled the number of problem statements as a third measure of project size. Table 4 shows the resulting project-size models. The impact analysis method SPAM uses these project size specifications to assess how often process activities are executed.

Tools and technologies. To support the requirements-planning process , we selected four technologies: OS, Prototype, Tool, and Tool+.

With *OS technology*, planners use AIX operating-system commands, the AIX file system, and paper and pen. They document requirements information by appending ASCII files, e-mail messages, or notes on paper. This

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

TABLE 5
EFFORT ANALYSIS RESULTS IN PERSON-WEEKS OF EFFORT

Effort analysis

Process	Technology	Project size in features: 5	10	20	40	80
Regular	OS	0.22	0.37	0.73	1.66	4.15
	Prototype	0.23	0.58	1.85	6.52	24.02
	Tool	0.16	0.48	1.68	6.22	23.40
	Tool+	0.12	0.43	1.57	5.91	22.45
Advanced	OS	0.66	1.28	2.95	7.86	23.50
	Prototype	0.50	1.13	3.17	10.26	35.95
	Tool	0.32	0.82	2.53	8.63	31.12
	Tool+	0.24	0.69	2.24	7.87	28.73
Complete	OS	14.08	19.35	36.57	96.06	305.70
	Prototype	4.42	5.98	10.90	27.60	85.69
	Tool	2.54	3.60	7.01	18.69	59.66
	Tool+	1.93	2.72	5.24	13.86	44.04

TABLE 6
PRODUCTIVITY ANALYSIS RESULTS IN FEATURES PER WEEK

Productivity analysis

Process	Technology	Project size in features: 5	10	20	40	80
Regular	OS	23.01	28.04	27.33	24.05	19.28
	Prototype	21.60	17.10	10.82	6.13	3.33
	Tool	31.82	20.71	11.87	6.43	3.41
	Tool+	40.05	23.24	12.71	6.76	3.56
Advanced	OS	7.59	7.79	6.77	5.08	3.40
	Prototype	9.92	8.84	6.31	3.89	2.22
	Tool	15.84	12.18	7.91	4.63	2.57
	Tool+	20.88	14.47	8.91	5.08	2.78
Complete	OS	0.35	0.51	0.54	0.41	0.26
	Prototype	1.13	1.67	1.83	1.44	0.93
	Tool	1.96	2.77	2.85	2.14	1.34
	Tool+	2.58	3.67	3.81	2.88	1.81

is a simple, straightforward technology that does not require much effort to document requirements information. However, accessing this information is quite cumbersome.

The *Tool* technology was preselected by the organization for requirements-management. It allows planners to carry out all documentation, access, and communication activities by selecting one or more functions from the tool's graphical user interface. The tool stores the requirements information in a relational database that planners can conveniently query for specific information on any item. The main drawback of this technology in terms of productivity is that planners must enter requirements information into the tool. This includes rekeying the notes they make on raw requirements when they don't have access to the tool. On the other hand, Tool makes accessing requirements information almost effortless.

The *Prototype* technology consists of a collection of AIX shell scripts. The scripts let planners document requirements information in a database of structured plain-text files stored in the AIX file system. Because the file system is used as the database and because the prototype has a somewhat clumsy command-line interface, carrying out activities requires more effort than with Tool. The *Tool+* technology improves on Tool in that, while it has the same functionality and interface as the regular tool, performance is improved and thus activities require less user time.

With each technology, a planner must carry out various steps to complete a planning process activity. On average, OS requires 5.5 steps using OS commands; Prototype requires 4.9 steps using AIX shell scripts; and Tool and Tool+ each require 2.3 steps using the GUI controls.

Data analysis and raw results. For the effort and productivity analysis, we combined the three processes, five project sizes, and four technologies in all possible permutations. From this we derived 60 distinct productivity models, each of which describes a different method of planning requirements for a project of a specific size, using a specific process and technology. We then analyzed each productivity model for the user effort required and productivity yielded. In this analysis, the process specifies the detailed activities that must be carried out for requirements planning, the project size governs how often these activities are carried out, and the technology determines how much effort is required to accomplish these activities using a certain tool.

Table 5 shows the raw effort data, with the five columns representing the project sizes in number of features for each (5, 10, 20, 40, and 80). In each of the process sections—regular, advanced, and complete—are four rows that show the technology used. The table cells show the effort, measured in person-weeks, required to carry out the requirements-planning activities with each combination of process, technology, and project size.

From this effort analysis, we could

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

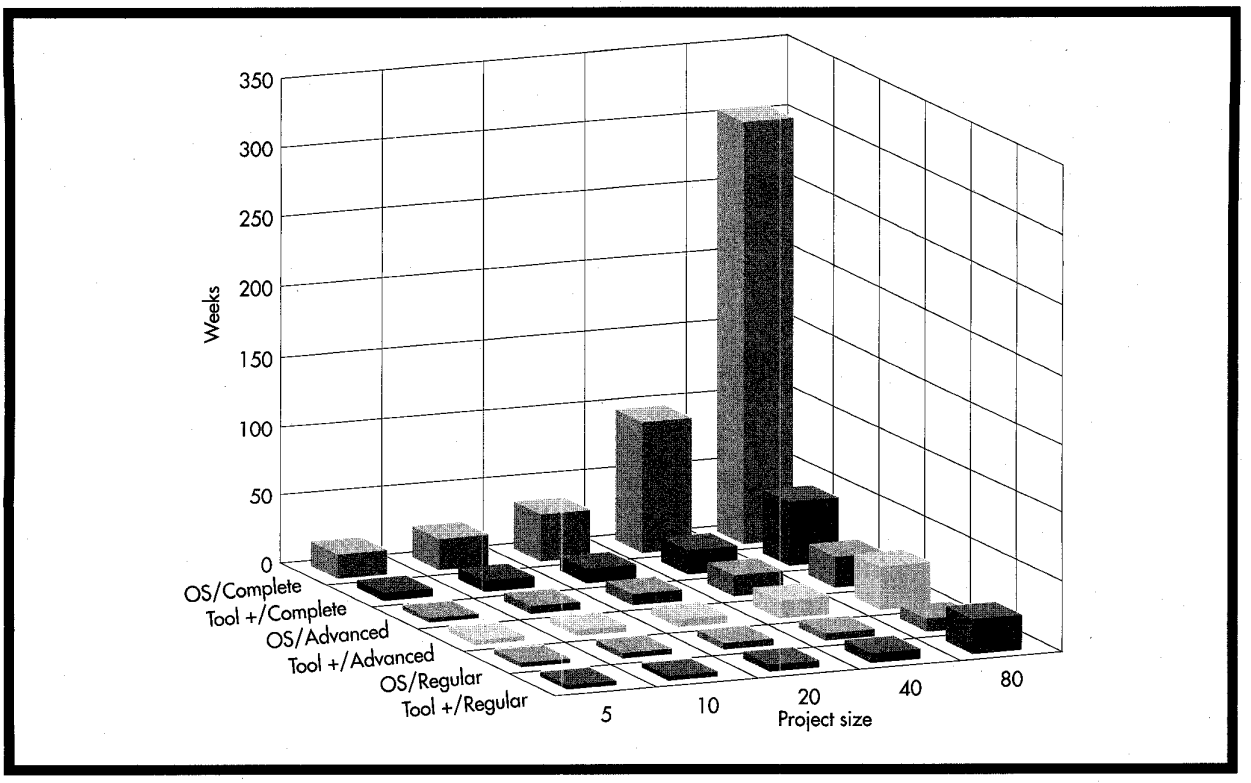


Figure 1. *Effort analysis using OS and Tool+: an overview of 30 data points.*

easily produce productivity data. We derived the productivity figures, shown in Table 6, by dividing the number of features produced by the effort required and normalizing to a features-per-week measure.

This effort and productivity data is the raw result of our case study. If you examine this data from different viewpoints, you can gain insight into a tool's impact on different situations. For example, you can analyze how different processes and technologies affect specific projects, how the productivity of a specific process is affected by different technologies and project sizes, or how different processes and project sizes affect the productivity of a specific technology.

RESULTS INTERPRETATION

We focus our interpretation on comparing effort requirements of alternative processes and technologies to project-specific resource constraints. Thus, after evaluating the 60 productivity models, we set out to determine which process and which technology would be appropriate for projects of a given size with specific quality requirements and resource constraints.

We anticipated that moving from a simple to a more complex process would increase resource consumption—thus, we expected to see a trade-off between productivity and process sophistication. However, we also believed that tool insertion would counteract the expected productivity loss and therefore make such process improvements less costly.

Figure 1 shows an overview of the effort data for OS and Tool+. Given these results, OS technology seems grossly inadequate for the complete process and the project size of 80 features; when Tool+ is used with the same process and project size, the investment is only one-seventh as much. This is also true for all other project sizes.

Data for Prototype and Tool tend to be similar to that of Tool+: Tool always requires more effort than Tool+, but less effort than Prototype. The same is not true for OS; depending on project size and development process, it can require more or less effort than Tool+, as Table 5 shows.

Figure 2 shows all 60 data points converted to productivity in features per week. From both the effort and productivity data, we observed two expected trends: required effort increases with project size, and productivity decreases with process complexity. However, we did not foresee that when you use OS technology with any process, or the complete process with any technology, that productivity would peak at medium-sized projects; nor, conversely, that when you use Prototype, Tool, or Tool+ with the regular or advanced process that productivity would peak for small projects.

Application. When it's time to decide on a tool and you know the size of the project, you first calculate the effort required by the various technologies and processes you are considering. Based on the actual resource constraints, you can then eliminate process/technology combinations that exceed these constraints.

Figure 3 shows the effort required with different processes and technologies for a five-feature project. Based on this, you would select Tool+; it demands the least effort for all three processes. If Tool+ is unavailable, Tool should be selected because it requires less effort than either OS or Prototype.

Using the analysis data, you can also select the most sophisticated process feasible under your project's resource constraints. For instance, if you do not want to invest more than one person-week into documentation, access, and

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

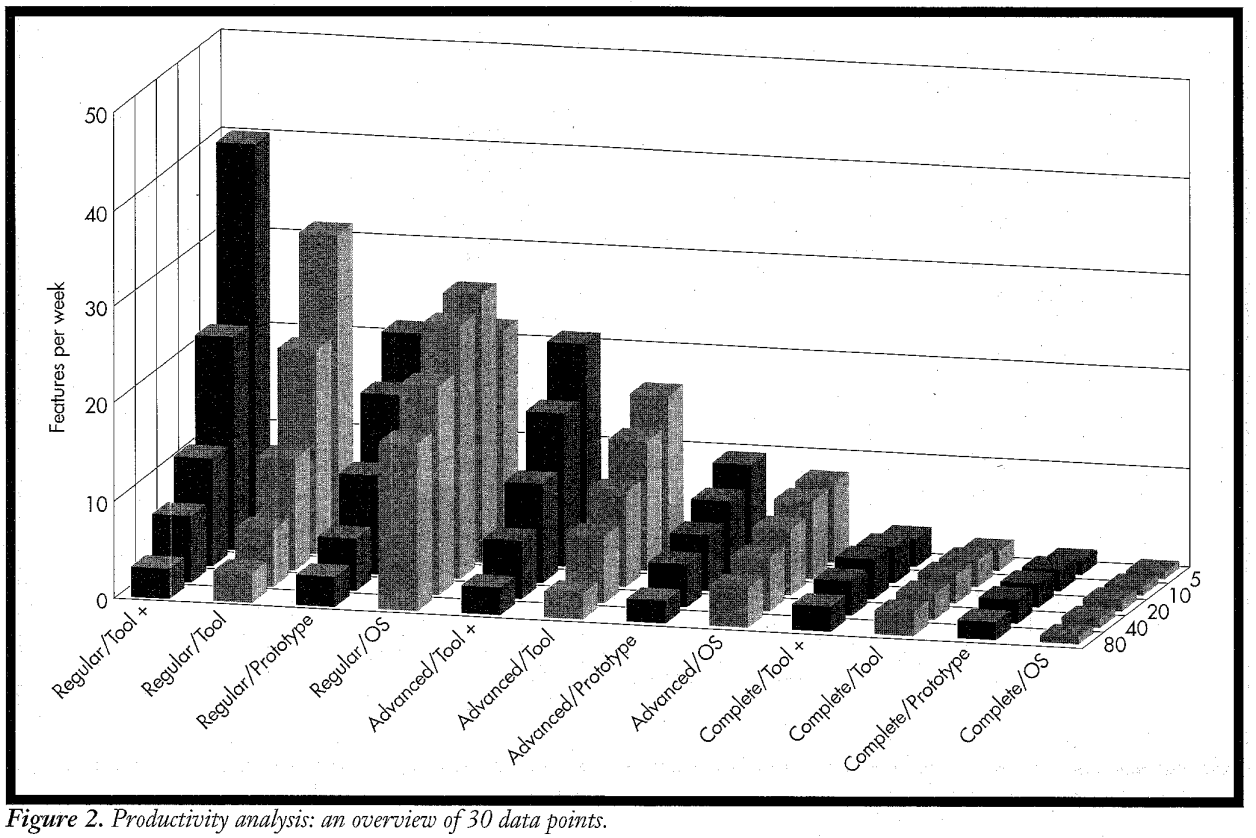


Figure 2. *Productivity analysis: an overview of 30 data points.*

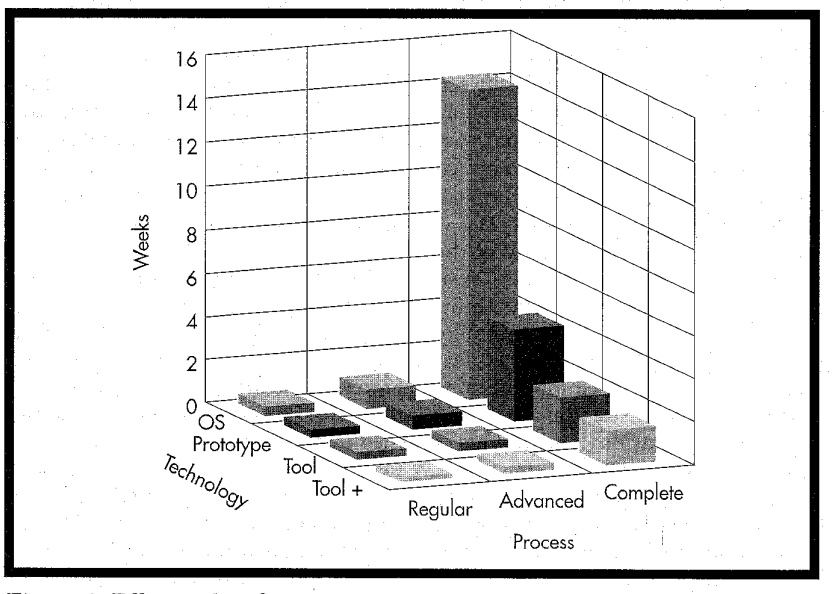


Figure 3. *Effort analysis for project size 5.*

QA in the planning process, and if you prefer a process that is more sophisticated than the regular process, then you should adopt the Tool or Tool+ technology and use the advanced process.

Because you are adding only five features, you can more or less maintain the same level of effort (0.21 weeks) required by OS and the regular process if you move to the advanced process (0.23 weeks with Tool+). Adopting Tool+ lets you move to the advanced process with about 10 percent more effort; whereas the same process improvement without a tool requires a 210 percent effort increase. This shows that adopting Tool is a good choice when you want to improve your process, even though productivity suffers a slight decrease.

This data highlights the danger of adopting tools without studying process changes. You might not be aware that the adoption of Tool+ triggered a process improvement, and might thus discard the tool because of the productivity loss. But, although using Tool+ would cost you an additional 10 percent of effort, not using it would triple your effort. Knowing this, you may choose to use Tool+ in spite of the productivity loss.

On larger projects, we realized that the choice of the best tool changes with project size. In a project with 10 added features, we discovered an interesting phenomenon: for the regular process, using the OS technology would be less expensive than using Tool! It costs less to do the same activities using the regular process with OS than with any of the three "more advanced" technologies. Although it is true that tools partially automate software development, the conclusion that adopting a tool will always save effort or increase productivity is a fallacy.

In a project with 20 added features,

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

the simple OS technology continues to gain ground against the more sophisticated tools when you use the regular or advanced process. If you select the complete process, however, Tool+ is more efficient. This trend was amplified as project size increased. For a project with 40 added features that uses the advanced process, the OS technology demands the same resources as Tool+. When using the advanced process with 80 added features , OS is even cheaper than Tool+.

Overall, if you choose the regular process, OS demands the least effort (with the exception of a project adding only five features); with the complete process, Tool+ will save you the most effort on all project sizes. Results using advanced processes were more variable: OS is better for large projects and the more sophisticated technologies are more efficient for small projects. Thus, it's important to keep in mind that the development process will affect tool choice differently depending on the size of your project.

TABLE 7
COST OF PROCESS IMPROVEMENT WITH AND WITHOUT TOOL INSERTION

Improvement: *regular process to advanced process*

	Project size in features				
	5	*10*	*20*	*40*	*80*
Cost of Regular/OS	$630	$1,080	$2,190	$4,980	$12,420
Additional cost of Advanced/OS	$1,320	$2,760	$6,660	$18,600	$58,080
Additional cost of Advanced/Tool+	$60	$990	$4,530	$18,600	$73,770
Savings through insertion of Tool+	$1,260	$1,770	$2,130	0	-$15,690
Savings through insertion of Tool+	95%	64%	32%	0	-27%

Improvement: *regular process to advanced process*

	Project size in features				
	5	*10*	*20*	*40*	*80*
Cost of Advanced/OS	$1,950	$3,840	$8,850	$23,580	$70,500
Additional cost of Complete/OS	$40,260	$54,210	$100,830	$264,570	$846,600
Additional cost of Complete/Tool+	$3,840	$4,290	$6,870	$18,000	$61,590
Savings through insertion of Tool+	$36,420	$49,920	$93,960	$246,570	$785,010
Savings through insertion of Tool+	90%	92%	93%	93%	93%

PROCESS IMPROVEMENT COSTS

One of the key questions we wanted to investigate was: Can we make a process more rigorous while maintaining or increasing productivity through tool insertion? To find an answer, we first examined two types of process improvement using OS: moving from the regular to the advanced process and moving from the advanced to the complete process. We then compared the additional labor costs of the improvement without tool insertion to the costs of the same improvement accompanied by the adoption of Tool+. (This comparison can also be made for Prototype or Tool.)

Table 7 shows the data we gathered on process improvement, derived by multiplying the effort data with a cost index of $75 per staff hour. The first row of each half of the table is the "base cost": the cost of the regular and advanced processes using OS.

As the data in the top of the table shows, when you move from the regular to the advanced process effort levels will increase, even with Tool+. In the three smaller projects, it is cheaper to adopt a tool than to advance without one; the savings through tool insertion are quite substantial. However, when you add 40 features or more, tool insertion does not result in any savings; at 80 features, process improvements actually cost more with tool insertion. Thus, for larger projects, tool insertion brings a loss in productivity.

In the change from the advanced to the complete process, the results were slightly different. Here too, you cannot avoid expending more effort when you advance to a more sophisticated process. However, across all project sizes—even those with 40 and 80 added features—this process improvement is substantially less expensive when Tool+ is inserted. As the lower portion of Table 7 shows, inserting Tool+ can consistently save you between 90 and 93 percent of the additional labor cost of process improvement.

EMERGING PRINCIPLES

On the basis of our case study results, we formulated four tentative rules.

- The less complex your process and the larger your project, the higher the probability that tool insertion will curtail your productivity.
- When you adopt a more rigorous process, be prepared to increase resources even if you are improving tool support at the same time.
- A tool's performance can peak when used in projects of a certain size, while being less productive in larger or smaller projects.
- In terms of effort, adopting a rigorous process can be substantially less expensive if you also adopt the appropriate tools.

Tool insertion seems best justified when you plan to adopt a sophisticated process and when you are prepared to put in the extra effort that process requires. How much more effort is required can be determined through a quantitative analysis of tool-insertion impact. Without this, your only choice is to wait and see.

The results we present here pertain to a specific case study; more studies are needed before such experiences can be generalized into principles. However, it is clear that quantitative process modeling and analysis can be valuable tools for making decisions about tool assessment and adoption. Such methods can help make your software process more effective and efficient. ◆

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.

ACKNOWLEDGMENT

We are indebted to Jacob Slonim, John Botsford, Jack Dawson, John Robichaud, David Benjamin, Allan Friedman, and Sam Dalal of IBM Canada Ltd. for facilitating this research. This work would not have been possible without their time, effort, and skills.

REFERENCES

1. R.S. Pressman, *Software Engineering—A Practitioner's Approach*, 3rd ed., McGraw-Hill, New York, 1992.
2. M. Paulk et al., "Capability Maturity Model, Version 1.1," *IEEE Software*, July 1993, pp. 18-27.
3. G.G. Schulmeyer and J.I. McManus, *Total Quality Management for Software*, Van Nostrand Reinhold, New York, 1993.
4. V. Basili, "Software Development: A Paradigm for the Future," *Proc. 13th Int'l Comp. Software Applications Conf.*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 471-485.
5. N.H. Madhavji, "The Process Cycle," *IEE/BCS Software Eng. J.*, Sept. 1991, pp. 234-242.
6. L.M. Garro, *Computer-Aided Software Engineering (CASE) and Productivity*, master's thesis, The American Univ., Washington, D.C., 1993; available through UMI: at 1-800-521-0600.
7. W. Scacchi, "Understanding Software Productivity," in *Software Engineering and Knowledge Engineering: Trends for the Next Decade*, Vol. 4, World Scientific Press, Singapore, 1995; available at http://cwis.usc.edu/dept/ATRIUM/Software_Productivity.ps.
8. E. Chikofsky, D.A. Martin, and H. Chang, "Assessing the State of Tools Assessment," *IEEE Software*, May 1992, pp. 18-21.
9. C.C. Huff, "Elements of a Realistic Case Tool Adoption Budget," *Comm. ACM*, Apr. 1992, pp. 45-54.
10. T. Bruckhaus, "Analyzing CASE Impact," *Proc. 1994 Centre for Advanced Studies Conference*, IBM Canada Ltd. and National Research Council Canada, Toronto, 1995, pp. 179-194.
11. T. Bruckhaus, *A Quantitative Approach for Analyzing the Impact of Tools on Software Productivity*, doctoral dissertation, McGill Univ., Montreal, forthcoming.
12. R.K. Yin, *Applied Social Research Methods Series: Case Study*, Sage Publications, London, 1994.

SOFTWARE ENGINEERS

Northrop Grumman is recognized for its leadership and achievement in the design and development of real-time, embedded software and is a major midwest manufacturer of advanced electronics systems. We are an SEI level 3 organization committed to further development in software process and technology.

Applications include active RF and IR systems. Positions at various levels from individual contributor to project leaders. BS Physics/EE/CS/Math and a minimum of 3 years "C" programming, software development for real-time multi-tasking/multi-processor, embedded systems experience required. Electronics warfare industry experience a plus.

ADA & C Software Developers Experience in design, implementation, test, integration of real-time embedded and non-embedded software. ADA or C language required. Knowledge of modern software design methodologies necessary.

Software Architects To initiate design projects and perform requirements analyses, algorithm development and high level design. Requires an advanced technical degree and a minimum of 5 years experience in the development of large-scale real time embedded software systems. Knowledge of computer hardware architectures, performance simulation, modeling and exposure to knowledge based expert systems and object-oriented programming techniques, a plus.

Embedded Software Developer Detailed design, coding, testing and integration of embedded real-time software implementation experience using modern software design methodologies.

For detailed additional openings, view us through the net: http://www.cweb.com/resources/northropgrum/ or send resumes as ASCII text to: resumes@dsd.northrop.com or fax (847) 590-3189 or mail resume to: Manager- Staffing, Northrop Grumman EIWS Dept MM/T91, 600 Hicks Road, Rolling Meadows, IL 60008

We offer a competitive salary/benefits package including: Health/major medical/dental/life insurance, 401(k) and pension plans. Excellent relocation package. We are a smoke-free workplace. U.S. citizenship required for most positions.
An equal opportunity employer m/f/d/v.

NORTHROP GRUMMAN

Tilmann Bruckhaus is a software process analyst at Sun Microsystems. His research interests are in software quality and productivity, and software engineering tools and environments. He has designed and implemented learning systems, modeling and optimization tools, management information systems, and graphical user interfaces. He is currently a PhD candidate in computer science at McGill University, Montreal, Canada, where he held a Centre for Advanced Studies fellowship from IBM Canada.
Tilmann received an MSc in computer science from Rheinisch-Westfälische Technische Hochschule in Aachen, Germany, in 1992.

Nazim H. Madhavji is an associate professor at McGill University. His research interests are in software engineering processes and technologies. He is chair of the IEEE TCSE committee on software process and was a guest editor of special issues on software process in IEE/BCS *Software Engineering Journal* (Sept. 1991) and *IEEE Transactions on Software Engineering* (Dec. 1993). He also served on the advisory editorial board of the *Journal of Software Maintenance*. From 1993 to 1995, Madhavii was research director of the Software Process Programme at Centre de Recherche Informatique de Montreal.

Ingrid Janssen is manager of the application and data solutions team at the IBM Canada Toronto Laboratory. She has more than 18 years' experience in application and software development in various IBM divisions. Her areas of interest are software engineering, project management, and women in computer science.

John Henshaw is a development manager in the Networked Applications Development Center at the IBM Canada Laboratory. Prior to this assignment, he managed the lab's software engineering process group. His technical interests are in reverse engineering, software engineering, and programming languages and environments.
Henshaw recieved an MSc in computer science from the University of Western Ontario.

Address questions about this article to Bruckhaus at Sun Microsystems, 2550 Garcia Ave., M/S UMPK17-307, Mountain View, CA 94043; 415-786-7229; fax, 415-786-5734; Tilmann.Bruckhaus@Eng.Sun.Com; or Madhavji at the School of Computer Science, McGill University, 3480 University Street, Montreal, Quebec, Canada, H3A 2A7; 514-398-6730; fax, 514-398-3883; madhavji@cs.mcgill.ca.

Authorized licensed use limited to: MAIN LIBRARY UNIVERSITY OF ZURICH. Downloaded on November 30, 2009 at 02:35 from IEEE Xplore. Restrictions apply.