Benchmarking Software Development Productivity

Determining how to most effectively use multicompany databases to benchmark software development productivity can be challenging. In this article, the authors present results of an analysis performed on the Experience database, with an in-depth look at key productivity factors.

Katrina D. Maxwell, *Datamax*

Pekka Forselius, *Software Technology Transfer Finland*

In response to the widespread need to benchmark software-development productivity, numerous software-metrics databases comprising data collected from companies operating in different business sectors have become available. For effective benchmarking, a company must select the projects in the databases that most resemble their own. However, in addition to the problems associated with multicompany databases, such as measurement comparability, past research has shown that only a few factors affect software-development productivity in any given environment.[1–5] Also, because industry-wide productivity rates vary widely, and there are many different ways to "cut" the data, determining which projects to benchmark against is not easy. As a solution, a company could use benchmarking equations—such as the ones we derived from a productivity-variation analysis we performed on the Experience database—to compare their software-development productivity to that of similar projects. This article presents the results of our analysis, including those benchmarking equations.

Results of Experience—Database Productivity Analysis

The Experience database comprises 206 business-software projects from 26 companies in Finland. For more information on the database, see the "Experience Database" sidebar. Table 1 presents the variables used in our Experience-database productivity analysis. Because our analysis showed that the customer company's business sector is the most important across-company benchmarking variable for this database, we examine productivity-factor differences in the banking, insurance, manufacturing, wholesale-and-retail, and public-administra-

Table 1

Variables Considered in the Analysis

Variable name	Type	Values (or levels)	Definition
			Classification variables
LANG	Nominal	90	Application programming language
APP	Nominal	8	Application type (customer service, MIS, and so on)
HAR	Nominal	5	Hardware platform (mainframe, PC, and so on)
IFC	Nominal	2	User interface (graphical, character)
ACQ	Nominal	5	Development model (waterfall, and so on)
DBA	Nominal	5	DBMS architecture (hierarchical, and so on)
CDB	Nominal	3	Database centralization (in client, server, combined)
CSW	Nominal	3	Software centralization (in client, server, combined)
DBT1	Nominal	22	DBMS tools (Db/2, Idms, Dl/1, Ingres, and so on)
CAS1	Nominal	8	CASE tools (Iew/adw, foundation, and so on)
OPE	Nominal	17	Operating system (DOS, Unix, VMS, and so on)
COMPANY	Nominal	26	Company where project was developed
SECTOR	Nominal	5	Customer's business sector: banking, insurance, manufacturing, retail, public administration
			Quantitative project data
EFFORT	Ratio		Effort from specification to delivery in hours
DUR	Ratio		Duration of project in months
INP	Ratio		Sum of inputs
INQ	Ratio		Sum of inquiries
OUT	Ratio		Sum of outputs
INT	Ratio		Sum of interfaces
FIL	Ratio		Sum of entities
LAS	Ratio		Sum of algorithms
SUMFXN	Ratio		Unweighted Experience 2.0 function points
WSUMFXN	Ratio		Weighted Experience 2.0 function points
YSTART	Interval	1978–1994	Start year of project
NMET	Ratio	0–9	Number of different methods used
NLAN	Ratio	1–4	Number of different languages used
			Productivity factors*[6] *(1: very low, 5: very high)
P01	Ordinal	1–5	Customer participation
P02	Ordinal	1–5	Development environment adequacy
P03	Ordinal	1–5	Staff availability
P04	Ordinal	1–5	Standards use
P05	Ordinal	1–5	Methods use
P06	Ordinal	1–5	Tools use
P07	Ordinal	1–5	Software's logical complexity
P08	Ordinal	1–5	Requirements volatility
P09	Ordinal	1–5	Quality requirements
P10	Ordinal	1–5	Efficiency requirements
P11	Ordinal	1–5	Installation requirements
P12	Ordinal	1–5	Staff's analysis skills
P13	Ordinal	1–5	Staff's application knowledge
P14	Ordinal	1–5	Staff's tool skills
P15	Ordinal	1–5	Staff's team skills

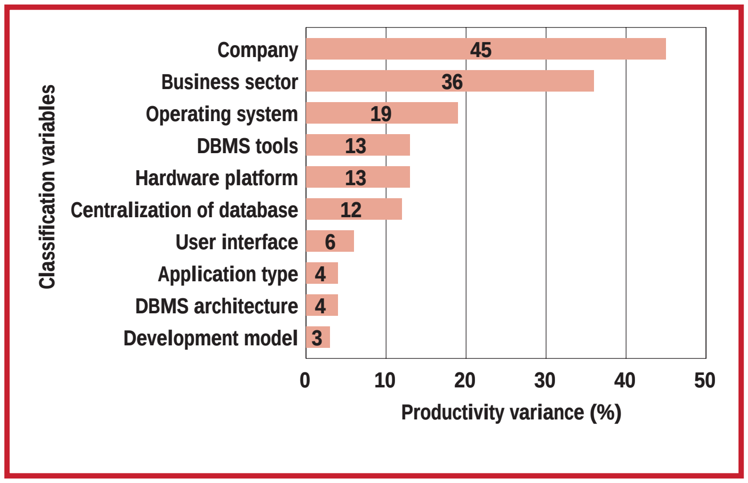


Figure 1. Summary of individual-classification-variable productivity analysis. (For details on the statistical analysis, see the references.[10])

tion sectors in more detail.

In Figure 1, we show the variance accounted for by each significant classification variable.

That the Company variable (company that developed project) accounted for the greatest productivity variance (45%) highlights the need for companies to establish their own software-metrics databases, in addition to benchmarking their data against that of other companies. The next most significant classification variable was the customer company's business sector, which accounted for 36% of the variance. Application programming language, CASE tools, and software centralization did not significantly account for any productivity variation. The insignificance of language runs counter to other studies' results might be due to Cobol's inclusion in most of the language combinations with enough observations to analyze.[2–4,7,8] If data concerning each language's share and role in multilanguage projects had been collected, the number of different language combinations (90 for 206 projects) might have been radically reduced and the results might have been different.

In Figure 2, we break down the mean productivity by business sector. The manufacturing sector has the highest productivity, and the banking and insurance sectors the lowest. This might be because the manufacturing and wholesale-and-retail companies functioned in a more competitive environment than the banking and insurance companies during the timeframe considered. External suppliers, chosen through a bidding process, also undertook most of the public-administration projects. Most companies developed their banking and insurance projects in-house. In addition, the manufacturing, wholesale-and-retail, and public-administration data contained more projects developing administrative-type systems, while the data-processing projects in the banking and insurance sectors were more related to the companies' core operations. Project intents likely affected quality requirements. Furthermore, requirements volatility was higher in the banking and insurance sectors because of merging companies and changes in management. The International Software Benchmarking Standards Group also found that, in their database, banking and insurance had the lowest productivity of these five sectors.[9]

Results of Productivity Analysis by Business Sector

In Table 2, we show the variables most affecting each sector's productivity. The table can be interpreted as follows: for the insurance sector, three variables account for 36% of the productivity variance. Requirements volatility accounts for 19%, the software's logical complexity accounts for an additional 11%, and tools use accounts for an additional 6%. The table also shows the effect on productivity of increasing the value of the variable. For example, increasing tools use has a positive impact on productivity in the insurance sector.

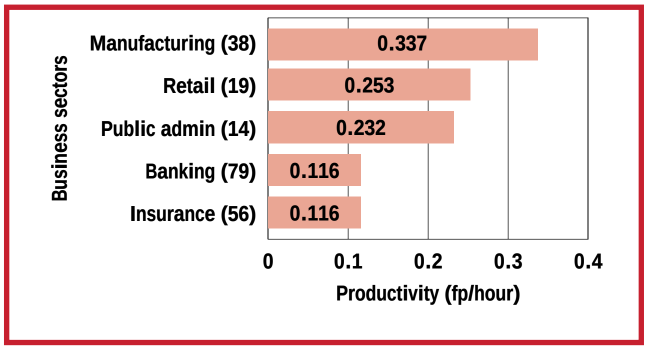


Figure 2. Mean productivity by business sector, in function points per hour. The number after each business-sector label refers to the number of projects.

Banking

In the banking sector, five variables accounted for 46% of the variance in the data. Productivity in the banking sector decreases with increasing efficiency requirements and require-

Table 2

Best Productivity Model by Business Sector

Business sector	Significant variables	Effect on productivity	Variance accounted for per variable (%)	Total variance accounted for (%)
Banking	User interface	Depends on type	17	
	Efficiency requirements	Negative	14	46
	Requirements volatility	Negative	10	
	Weighted Experience 2.0 function pts.	Positive	3	
	Staffer tool skills	Positive	2	
Insurance	Requirements volatility	Negative	19	36
	Software's logical complexity	Negative	11	
	Tools use	Positive	6	
Manufacturing	Hardware platform	Depends on type	45	70
	Requirements volatility	Negative	10	
	CASE tools	Depends on type	7	
	Standards use	Negative	5	
	Staff's tool skills	Positive	3	
Wholesale-and-retail	Staff availability	Positive	16	30
	Number of languages	Negative	14	
Public administration	Number of inquiries	Positive	24	39
	Customer participation	Negative	15	

ments volatility, and increases with increasing staff tool skills and size. This last finding is especially interesting, as it is contrary to most function-point practitioners' belief. As past research had revealed large diseconomies of scale, the trend in the banks was to break large software-development projects into smaller projects. However, these smaller projects' proportionally larger overhead made them less productive. Large customer databases (1–2 million customers), large account databases, and advanced online services typify banking projects. Managing transactions quickly requires a high efficiency level. Some projects' requirements volatility was high because of merging banks and changes in management. As most companies developed their projects in-house, it was also more tempting and easier for the client to change the requirements during the project. Tool skills are also important in the banking sector. Data processing is vital to banks' core business, and banks invest in tools. In the late '80s and early '90s, the banks also heavily invested in training in this area.

Productivity also depends on the user interface. Productivity is twice as high for projects with a graphical, rather than a character, user interface. The graphical-user-interface projects might show higher productivity because they took place more recently. (The combined effects of many factors' evolution over time—such as modern programming practices, tools use, and storage constraints—might explain why more recent projects have higher productivity.) Another explanation might be that GUI-development-tool ease-of-use entices software developers to build more functionality into applications, without much additional effort. This causes the GUI projects to have a higher function-point count and thus a higher productivity.

Experience Database

The Experience database began as a publicly organized cooperative project to launch and support the measurement programs of 16 member companies in Finland. The project grew and is now an STTF-managed commercial activity (www.sttf.fi/html/exppro.html). Project managers use Experience Pro, a tool incorporating the database, for preliminary project planning, including cost estimation, reuse analysis, software-process-capability analysis, risk analysis, and productivity benchmarking. Companies buy the tool and pay an annual maintenance fee. In return, they receive the tool, new software versions, and updated data. Companies can add their own data to the tool; also, STTF offers a maintenance-fee reduction to companies for each project they contribute to the shared database. The data validity and comparability is maximized, as all companies collect data using the same tool and every variable's value is defined (to reduce subjectivity). Users discuss interpretation problems in Experience-user-group meetings.

Table 3

Productivity-Benchmarking Equations by Business Sector

Business sector	Productivity (fp/hour) = (variables defined in Table 1)			
Banking	Text user interface			
	$0.1828 \times WSUMFXN^{0.1622} \times P08^{-0.7973} \times P10^{-0.8530} \times P14^{0.3115}$			
	Graphical user interface			
	$0.4590 \times WSUMFXN^{0.1622} \times P08^{-0.7973} \times P10^{-0.8530} \times P14^{0.3115}$			
Insurance	$0.1757 \times P06^{0.8950} \times P07^{-0.4911} \times P08^{-0.8409}$			
Manufacturing	$2.8157 \times HC \times P04^{-1.0047} \times P08^{-0.5592} \times P14^{0.2872}$			
	where HC = hardware and CASE-tool multiplier (values below)			
	Tools	No CASE tool	IEW/ADW	Other CASE tools
	Network	0.3479	0.2792	no data
	Mainframe	0.2339	0.3585	no data
	Mini	0.9666	0.3493	no data
	Combined	1.0000	no data	no data
	If you used a CASE tool other than IEW/ADW, you can calculate productivity using the following equation, which doesn't depend on a CASE tool: $1.3012 \times HAR \times P08^{-0.5720}$			
	where HAR = Hardware Multiplier (values below)			
	Network	0.3095		
	Mainframe	0.2988		
	Mini	0.5921		
	Combined	1.0000		
Wholesale and retail	$0.1072 \times NLAN^{-0.4627} \times P03^{0.6651}$			
Public administration	$0.2127 \times INQ^{0.1493} \times P01^{-0.3950}$			

Insurance

Productivity in the insurance sector decreases with the software's increasing requirements volatility and logical complexity, and increases with increasing tools use. These three variables account for 36% of the variance. The banking and insurance sectors have many similarities. Like those in the banking sector, insurance-sector companies invest in tools and training because data processing is vital to their core business. In addition, some insurance projects' requirements volatility was high because of merging companies and changes in management. As most insurance companies also developed their projects in-house, the client could change the requirements during the project more easily. The software's logical complexity is important in the insurance sector because the software is very algorithmic.

Manufacturing

The project data in the manufacturing sector mainly comprised administration-and-

Table 4

Comparison of Benchmarking Accuracy between Average Productivity and Productivity Equation

Business sector	Average productivity (fp/hour)	Average productivity MMRE	Average productivity PRED(.25)	Productivity equation MMRE	Productivity equation PRED(.25)
Banking	.116	81%	42%	42%	32%
Insurance	.116	56%	41%	36%	55%
Manufacturing	.337	67%	29%	23%	69%
Wholesale and retail	.253	59%	37%	27%	47%
Public administration	.232	36%	57%	21%	79%

information-service projects that weren't related to the company's core business. Productivity decreases with increasing requirements volatility and standards use, and increases with the staff's increasing tool skills. Requirements volatility was important, as internal software units—where the owner is more likely to change the requirements—undertook the development. The Finnish manufacturing sector had no common IT or software-development standards, so the standards the companies used weren't stable during this time. A couple of large companies in the sector have been investing heavily in tools and training.

Productivity also depends on the combination of the hardware platform and CASE tools used in the project (see Figure 3). Productivity was higher for mini and combined hardware platforms than for network and mainframe platforms. Network and mini projects in which IEW/ADW, a CASE tool for preliminary planning and requirements specification, were less productive than projects that did not. In contrast, the developer's productivity on mainframe projects was higher when they used this tool. This might be because the mainframe environment is older and developers had more experience using the tool. (IEW/ADW was the only CASE tool with enough observations to analyze.) The significant interaction between CASE-tool use and hardware platform is a good example of the danger of making conclusions using one variable's averages. Had we simply calculated the average productivity of CASE-tool use for all manufacturing projects, ignoring the interaction with hardware platform, we would have concluded that projects using IEW/ADW are less productive than projects using no CASE tools. Together, the variables account for 70% of the variance in productivity.

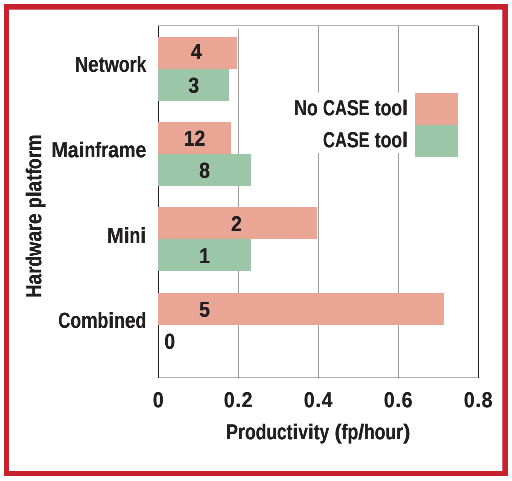


Figure 3. Productivity (fp/hour) by hardware platform and CASE tools. The number in each bar refers to the number of observations.

Wholesale-and-retail

One characteristic of projects in the wholesale-and-retail sector is that they have distributed systems. Shops have cash terminals connected to small computers in the back office, communicating with a mainframe computer at the main office. Thus, developers must develop software for different environments and with very specific languages. This requires specialized knowledge, more interfaces, and complicated testing. Productivity in the wholesale-and-retail sector decreases with the increased number of languages and increases with increased staff availability. These two variables combined accounted for 30% of the productivity variation. Cost efficiency is also important in this sector. Though companies often develop software in-house, which they figure to be cheaper, these companies don't have large software-development staffs. Thus, staff availability is an important factor because the few high-salaried software developers are fully utilized.

Public administration

Productivity in the public-administration sector decreases with increasing customer participation, and increases with an increasing number of inquiries. These two variables account for 39% of the productivity variance. Though high customer participation decreases project productivity, it might result in a more satisfied customer. The inquiry function's high influence might be due to the possibility that if the customer does not participate much, the software developer will develop extra inquiry functions—which are simple to copy—in case the customer asks for them at a later date.

Productivity-Benchmarking Equations

Table 3 contains each business sector's productivity-benchmarking equation. Although most valuable to companies that contributed to the Experience database, we provide the equations here so readers can test them on their own data. For an example of how to use the equations, see the "Productivity Measurement" sidebar. The sidebar also contains a description of the Experience 2.0 method and relates it to the IFPUG4.0 method. Table 4 shows the results of comparing the productivity-benchmark equations' accuracy with the accuracy obtained when the average productivity in each business sector is used as a benchmark for the Experience database projects. The table can be interpreted as follows: the average productivity of projects in the manufacturing sector is 0.337 fp/hour. If we compare the actual productivity of each Experience database manufacturing project to 0.337 fp/hour, the average error (MMRE) is 67%. The value of 0.337 fp/hour is within 25% (PRED(.25)) of the actual productivity for only 29% of the projects. If, instead, we use the productivity equation, the average error is 23%, and 69% of the estimates are within 25% of the actual productivity values. The results show that using the productivity equation improves estimation accuracy, especially in the average error. Thus, these

Productivity Measurement

We define productivity as output divided by the effort required to produce that output. But how should we measure software-development productivity? How do we translate the output—a completed software-development project—into a meaningful measurement? We believe software-project managers should base output measurement on a combination of a project's size, functionality, and quality. However, such a measurement doesn't yet exist. Line-of-code and function-point counts are currently the most common output measurements used. The Experience 2.0 method measures productivity in function points per hour.

Experience 2.0 function-point method

Nearly 40 different function-point-analysis-method variants exist today. Software-project managers use these methods for software functional sizing. From the user's viewpoint, Experience 2.0

- categorizes the software application's functions into six groups: inputs, outputs, inquiries, entities, interfaces, and algorithms (the software's total unweighted function-count is the sum of the count for each category),
- rates the functions in each category by difficulty, on a five-point scale ranging from very easy to very difficult, and then
- multiplies each of these 30 elements (each of the five difficulty ratings for each of the six groups) by an empirically determined weighting factor.

The total weighted function-point count (that is, the software's functional size) is the sum of each weighted element. The biggest difference between the Experience 2.0 method[1,2] and the widely used IFPUG4.0 method,[3] is that IFPUG4.0 also adjusts the count for a number of external complexity factors. Experience 2.0 treats these separately as productivity factors, so they don't affect the function-point count. This is why the unweighted IFPUG4.0 count is approximately equivalent to the weighted Experience 2.0 count. The Experience 2.0 function-point count will also be higher for software containing many algorithms. Client-server systems' interpretation rules are also different from IFPUG4.0.

Using the productivity equations

Software-project managers can use the productivity-benchmarking equations to calculate a productivity value for benchmarking their completed projects against projects in the Experience database, or to estimate a new project's productivity. They can also use the equations to determine the likely impact on productivity of changes in a key factor. Table A contains the definitions of the key productivity factors. (If a productivity factor's value is unknown, assume it is average.) Consider a public-administration project with six inquiries and average customer participation. Plugging INQ = 6 and P01 = 3 into the public-administration-sector equation given in Table 3 results in a productivity of 0.180 fp/hour:

$$\text{Productivity} = 0.2127 \times 6^{0.1493} \times 3^{-0.3950} = 0.180 \text{ fp/hour.}$$

This is the expected productivity for this project type and the benchmark with which to compare the project's actual productivity value.

What if the number of inquiries remained the same (INQ = 6), but the customer participation was very high (P01 = 5)? In a similar manner, we can calculate a productivity of 0.147 fp/hour. Thus, we can determine that the impact of very-high customer participation, with everything else remaining constant, decreases this project's productivity by 18%.

A software-development-productivity-benchmarking service based on these equations is available at www.datamax-france.com.

References

1. www.sttf.fi/html/exppro.html.
2. *Laturi-System Product Manual Version 2.0,* Information Technology Development Center, Helsinki, 1996.
3. *Function Point Counting Practices Manual, Release 4.0,* International Function Point Users Group, Westerville, Ohio, 1994.
4. R. Nevalainen and H. Maki, *Laturi-System Productivity Model Version 1.4,* Tech. Report 30.3.1994, Information Technology Development Center, Helsinki, 1994.

equations better reflect the actual productivity of projects in the Experience database.

Benchmarking projects using the average productivity of one "slice" or dimension of the data is not very accurate, even when used on the underlying data. Many of the variables collected don't even affect productivity, and only a few variables are important in each sector. For example, the hardware platform, operating system, and DBMS architecture don't affect productivity in the banking sector, so there is no reason to limit comparison of banking projects to other banking projects based on these variables. Companies must statistically analyze the data to develop benchmarking equations based on the key productivity factors.

Table A. Definitions of Key Productivity Factors[4]

Productivity factor	Very low (1)	Low (2)	Nominal (3)	High (4)	Very high (5)
Customer participation (P01)	None	Passive; client defines or approves < 30% of the functions	Client defines or approves 30–70% of the functions	Active; client defines or approves > 70% of the most important functions.	Very active participation
Staff availability (P03)	Big problems in key software personnel availability; software requires specialized knowledge	Members involved in other projects simultaneously and have maintenance responsibilities	Key members are involved in only one other project	Members of project are involved almost full time	Qualified software personnel are available when needed and can participate fully in project
Standards use (P04) Quality of standards applied in the project	Project managers must develop standards during project	Some standards are available, but not familiar; project managers must develop more	Project members use generally known standards in known environments	Project members use new standards, which other customers or software houses have applied in the same environment	Project members use new standards, which aren't familiar in the industry, but must be followed; use is controlled
Tools use (P06)	Minimal tools; editors, compilers, testing tools	Basic tools; interpreters, editors, compilers, debuggers, databases, libraries	Development environment, DBMS, support for most phases	Modern tools like CASE, project planning, application generators, standardized interfaces between phases	Integrated CASE environment covers entire lifecycle; all tools can support each other flexibly
Logical complexity (P07) Computing, I/O needs, and user-interface requirements	Only routines; no need for user interface; simple database	Functionally clear; no algorithmic tasks; database solution clear	Functionally typical; normal standard database; no algorithms	Processing more demanding; database large and complex; new requirements for user interfaces	Functionally and technically difficult solution; user interface very complex; distributed database
Requirements volatility (P08)	No new features; standard components; conversions only	Some changes to specifications; some new or adapted functions; some minor changes in data contents	More changes to specifications, but project members can handle them, and their impact is minor (< 15% new or modified functions)	Some major changes, impacting total architecture and requiring rework; 15–30% of functions new or modified	Continuously new requirements; lots of rework; > 30% of the functions new or modified
Efficiency requirements (P10)	No efficiency requirements needing attention and planning	Efficiency goals easy to reach, requirements below average	Capacity level of the software is stable and predictable; response time, transaction load, and turnaround time are typical	Specific peaks in capacity, response time, transaction processing, and turnaround time can be reached by specific design and implementation techniques	Efficiency is essential; strict efficiency goals need continuous attention and specific skills
Staff tool skills (P14) Average project-team experience	< 6 months experience	6–12 months	1–3 years	3–6 years	> 6 years experience

Also, the variables collected in the database accounted for less than 50% of the productivity variation in each business sector (with the exception of the manufacturing sector). This means there are likely to be other variables important in explaining each business sector's productivity variation that weren't in the database. For example, in the banking sector an important factor might be whether the bank is government owned. Additionally, many of one bank's projects had a lower productivity level because they were migration projects, requiring additional effort for converting the old system to the new. Unfortunately, we couldn't observe this effect in more detail because this data didn't exist for all bank projects. Nevertheless, our data-analysis process resulted in a greater understanding of the relationships among variables, and further insight as to why these factors are important.

As the Experience database grows, we plan to periodically reanalyze the data to determine how the key productivity factors are evolving over time. We are currently studying the factors affecting the productivity and cost of annual software-application maintenance in the banking sector.

Our data-analysis process resulted in a greater understanding of the relationships among variables, and further insight as to why these factors are important.

Acknowledgments

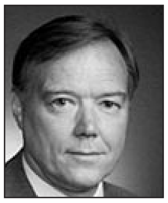
Special thanks to Risto Nevalainen of Software Technology Transfer Finland for defining, collecting, and providing the data; without his initiative this study would not have been possible. Thanks also to Herve Balloux, Barbara Kitchenham, and the reviewers for their comments on an earlier version of this article.

References

1. R.D. Banker, S.M. Datar, and C.F. Kemerer, "A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects," *Management Science*, Vol. 37, No. 1, Jan. 1991, pp. 1–18.
2. B.A. Kitchenham, "Empirical Studies of Assumptions that Underlie Software Cost-Estimation Models," *Information and Software Technology*, Vol. 34, No. 4, Apr. 1992, pp. 211–218.
3. K. Maxwell, L. Van Wassenhove, and S. Dutta, "Software Development Productivity of European Space, Military and Industrial Applications," *IEEE Trans. Software Eng.*, Vol. 22, No. 10, Oct. 1996, pp. 706–718.
4. K. Maxwell, L. Van Wassenhove, and S. Dutta, "Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation," *Management Science*, Vol. 45, No. 6, June 1999, pp. 787-803
5. T. Mukhopadhyay and S. Kekre, "Software Effort Models for Early Estimation of Process Control Applications," *IEEE Trans. Software Engineering*, Vol. 18, No. 10, Oct. 1992, pp. 915–924.
6. R. Nevalainen and H. Maki, *Laturi-System Productivity Model Version 1.4*, Tech. Report 30.3, 1994, Information Technology Development Center, Helsinki, 1994.
7. K. Maxwell, L. Van Wassenhove, and S. Dutta, "Benchmarking: The Data Contribution Dilemma," *Proc. 1997 European Software Control and Metrics Conference*, The ESCOM Conference, 30 Willow Tree Glade, Calcot, Reading RG31 7AZ, UK, 1997, pp 82–92.
8. C. Jones, *Applied Software Measurement: Assuring Productivity and Quality*, McGraw-Hill, New York, 1991.
9. ISBSG, "Worldwide Software Development—the Benchmark", Release 5, International Software Benchmarking Standards Group, Mar. 1998, Warrandyte, Victoria, Australia, p. 63.
10. K. Maxwell, "Benchmarking Software Development Productivity: Statistical Analysis by Business Sector," *Process Control for 2000 and Beyond*, R. Kusters et al., eds., Shaker Publishing, Maastricht, The Netherlands, 1998, pp. 33–41.

About the Authors

Katrina D. Maxwell is a Datamax–*Adding Value to Data* co-founder. Her research interests include applied data analysis, software productivity, and effort estimation. She has taught courses in quantitative methods at the University of Illinois, INSEAD and the ESCP-Paris School of Management. She received a BS in civil engineering from the University of Illinois and a PhD in mechanical engineering from Brunel University. She is the program chair of the European Software Control and Metrics Conference for 2000 and 2001, and an IEEE Computer Society member.



Pekka Forselius is director and project management consultant at Software Technology Transfer Finland. He developed the Experience Pro data-collection concept and is responsible for Experience Pro software development. His research interest is in learning software-development organizations, especially organizational-memory and organizational-understanding concepts. He received an MS in informatics from the University of Helsinki and an executive MBA from the University of Jyvaskyla. He has been a research associate at INSEAD since 1996 and is the secretary of the board of the Finnish Software Metrics Association.

Address questions about this article to Maxwell at Datamax, 14 Avenue Franklin-Roosevelt, 77210 Avon, France; datamax@computer.org.