Estimating Web Development Costs: There Are Differences

Donald J. Reifer
Reifer Consultants, Inc.

This article discusses the need for new metrics and models to estimate the effort and duration for Web development projects. It then describes a new size metric, Web objects, and a new cost estimation model, WEBMO, that have been developed to satisfy these needs. Most importantly, this article identifies differences between traditional and Web projects that managers need to be aware of when developing estimates prior to ending with a current status of the effort.

This past year, electronic commerce reportedly reached $5 billion in sales. Considering that this was during a recession, it is a marvelous achievement. You are probably thinking, "How was that achieved with the technology bubble bursting and Internet start-ups failing right and left?" The answer is simple. The larger businesses took the place of the smaller businesses. They moved to the Web with speed and enthusiasm, often for good business reasons. For example, General Electric reportedly saved $40 million in a single month compared with the same month in the previous year by moving its travel onto the Web [1].

Just as importantly, this move to the Web is heralding in a major change in the way we in the software community do business. For example, the large projects that we worked on in the past are being replaced by many small Web developments. These small projects are being done using different technology as well. Table 1 characterizes these changes to give some insight into the current trends. It highlights the move to agile methods [2], extreme programming methods [3], components [4], multimedia, and visual technologies by Web shops.

These trends are motivated by the move to quicker-paced developments. Instead of developing software from requirements, these Web development projects are gluing components together using agile instead of traditional methods. They build prototypes and refactor [5] them instead of focusing on design. From Table 1, you will see that Web developments seem deficient in the areas of process, discipline, and estimating. That is not entirely true. As Mark Paulk recently pointed out, process improvement and extreme methods are not incompatible [9]. However, many of the large firms with which my firm has recently worked seem to have abandoned process paradigms and the Software Engineering Institute's Capability Maturity Model® (CMM®), Software Capability Maturity Model® (SW-CMM®), and Capability Maturity Model Integration℠ (CMMI℠) [10, 11] in their quest to speed time-to-market as they move to the Web.

Those of us in the estimating community currently have not agreed upon how to address Web-based projects. The trouble is that the characteristics of the Web-based projects listed in Table 1 make it difficult for estimators to adapt and put their existing processes, metrics, and models to work operationally. Web projects are different. To highlight the challenges involved in the area of Web estimation, we have constructed Table 2 (see page 14). For comparative purposes, this table also identifies the approaches that we currently use to develop estimates for traditional software projects.

Most estimators would like to use the more traditional processes, metrics, models, and tools for estimating Web projects. They are mature, and many of us in the field have confidence in their ability to accurately predict project costs and schedules. We also have a great deal of experience using these metrics, models, and tools and feel comfortable with them and their outputs. However, as noted by Table 2 (page 14), these traditional approaches do not address the challenges that we face with Web projects. The two major challenges are accurately estimating size and duration. New size metrics are needed to cope with Web objects like shopping carts, Java scripts, and building blocks like Cookies, ActiveX controls, and Component Object Model components. New duration-estimating equations are needed to address the fact that the cube root laws used by most estimating models just do not seem to work for the Web.

New Web Applications Sizing Metrics Needed

Because Web cost can be treated as a function of size, a meaningful size predictor is needed for Web projects. Those working such projects agree that the popular size metrics, function points (FP) and source lines of code (SLOC), are not suitable for Web estimation because they do not take all of the Web objects (buttons, templates, etc.) into account. Luckily, the research community has not been idle. It has proposed several size metrics for Web developments (object points [12], application points [13], etc.). However, the only find-

Table 1: *Characteristics of Traditional vs. Web Development Projects*

Characteristics	Traditional Developments	Web Developments
Primary objective	Build quality software products at minimum cost.	Bring quality products to market as quickly as possible.
Typical project size	Medium to large (hundreds of team members).	Small (3-5 team members the norm, 30 the largest).
Typical timeline	12-18 months	3-6 months
Typical cost	$ millions	$ thousands
Development approach employed	Classical, requirements-based, phased and/or incremental delivery, use cases, documentation driven.	Agile methods, extreme programming, building block-based, demo-driven, prototyping, Rational Unified Process [6], MBASE [7].
Primary engineering technologies used	Object-oriented methods, generators, modern programming languages (C++), CASE tools, etc.	Component-based methods, 4th and 5th generation languages (html, Java, etc.) visualization (motion, animation), etc.
Processes employed	CMM-based	Ad hoc
Products developed	Code-based systems, mostly new, some reuse, many external interfaces, often-complex applications.	Object-based systems, many reusable components (shopping carts, etc.), few external interfaces, relatively simple.
People involved	Professional software engineers with 5+ years of experience.	Graphic designers, less-experienced software engineers, Java specialists.
Estimating technologies used	Source line of code or function point-based models, Web-based systems approach for small projects.	Design-to-fit based on available resources, Web-based systems for small projects.

Note: Table is an adaptation of the author's previously published work [8].

® Software Capability Maturity Model and SW-CMM are registered in the U.S. Patent and Trademark Office.
SM CMM Integration and CMMI are service marks of Carnegie Mellon University.

	Traditional Approach	Web-Based Challenges
Estimating process	Most use analogy supplemented by lessons learned from past experience.	Job costing, if done, performed ad hoc using inputs from the developers (too optimistic).
Size estimation	Because systems are built to requirements, SLOC or function points are used. Separate models are used for COTS and reused software (equivalent new lines).	Applications are built using templates using a variety of Web-based objects (html, applets, components, etc.). No agreement on size measure reached yet within the community.
Effort estimation	Effort is estimated via regression formulas modified by cost drivers (plot project data to develop relationships between variables).	Effort is estimated via analogy using job costing practices and experience as the guide. Little history is available.
Schedule estimation	Schedule is estimated using a cube root relationship with effort.	Schedules estimated using cube root relations are an order of magnitude high.
Model calibration	Measurements from past projects are used to calibrate models to improve accuracy.	Measurements from past projects are used to identify estimating knowledge base.
What if analysis	Estimating models are used to perform quantitative *what if* and risk analysis. They are also used to compute ROI and cost/benefits.	Most *what if* and risk analysis is mostly qualitative because models don't exist. ROI and cost/benefit analysis for electronic commerce remains an open challenge.

Note: Table is an adaptation of the author's previously published work [8].
SLOC = source lines of code, COTS = commercial off-the-shelf, ROI = return on investment

Table 2: *Web-Based Estimation Challenges*

ing that researchers in the field currently agree upon is that they cannot reach agreement on which of these is best. Based upon our experimentation, we believe that we have developed a size metric that can resolve the current debate. The metric, Web objects, predicts size by taking each of the many elements that make up the Web application into account as size is estimated.

You are probably asking, "What are Web objects?" Like function points, Web objects are defined to be a metric that provides users with an indication of the relative size of an application [14]. In our case, the applications run on the Web. Web objects predict size by permitting its users to bound the functionality of their applications in terms of the following five groups of predictors:

- Function points.
- Links.
- Multimedia files.
- Scripts.
- Web building blocks.

As indicated, Web objects extend function points to encompass groups of functions present in Web applications. For example, Web objects allow estimators to take Web building blocks like shopping carts and the number of XML language lines needed to link the application to a Web accessible database into account as they develop their estimate. Such extension is needed because traditional function points predict size using more traditional application characteristics like number of inputs and outputs.

Using the size predictor groupings listed in Table 3 to compute the number of Web objects, we have been able to repeatedly predict the size of a Web application with what we believe to be reasonable accuracy. These predictors allow us to take into account all of the different elements of applications that contribute to size, including those specific to the Web. We devised this list initially based upon the opinions of experts. For one year, we have applied the metric to develop estimates, collect project data, and refine our counting conventions based upon experience. Based upon analysis of 64 completed Web projects in five application domains, we have shown that these predictors can be used along with FPs to develop accurate estimates.

Like function points, the key to developing repeatable and consistent sizing predictor counts is a well-defined set of counting conventions. To help on our pilot projects, we have developed a white paper to explain our initial counting conventions [15]. We plan to update this and develop a more detailed counting manual for Web objects later this year. That manual, a version of which will posted on our Web site at <www.reifer.com>, will provide those interested in using Web objects with a consistent set of experience-based conventions for dealing with most situations they will likely encounter when sizing their Web applications.

We have also developed the worksheet in Table 4 to show you how to use the information gathered on predictors along with function points to size a typical Web application. Using an actual Java program being developed for a Web portal as an example, the size estimate developed in Table 4 provides you with an indication of how big the program would be once fully developed. It uses the five groupings of predictors that we have discussed to develop weighted counts that allow us to size a Web application based upon its unique characteristics.

We have also empirically developed backfiring ratios to convert from Web objects to SLOC. If the example shown in Table 4 were done in Java, the 356 Web objects would be the equivalent of 11,392 source lines of Java code, assuming a conversion ratio of 32 Java lines per Web object. Backfiring is important to us because we plan to use a modified version of the COCOMO II early design model to estimate effort and duration.

The COCOMO II model uses SLOC as its underlying size metric. However, after much experimentation, we calibrated our new cost estimation model WEB model (WEBMO) using Web objects instead of SLOC. The reason for this was that such calibration improved our estimating accuracy by as much as 12 percent in two of our five application domains. However, to remain compatible with COCOMO II until we can calibrate all five domains, we plan to continue to use SLOC in our formulas. This may change in the future once we gather more data and can more precisely calibrate our estimating model.

A New Estimation Model

Having a realistic size metric is just the first step in developing a model for accurately

Table 3: *Web Object Predictors*

Web Object Predictors	Description
Number of function points	Traditional metrics used to predict the size of non-Web applications using number of inputs, outputs, files, inquiries, and interfaces as the basis of estimate.
Number of XML, HTML, and query language links	Takes into account the effort to link applications, integrate them together dynamically, and bind them to the database and other applications in a persistent manner.
Number of multimedia files	Takes into account the effort required to insert audio, video, and images into applications.
Number of scripts	Takes into account the effort required to link HTML/XML data with applications and files and generate reports.
Number of Web building blocks	Takes into account the effort required to develop Web-enabled fine-grained building block libraries and related wrapper code needed to instantiate them.

estimating Web application effort and duration. The mathematical issues associated with making such predictions need to be reconciled before such models are transitioned into use.

The major issue revolves around the schedule law used by the model. Analysis of data we have collected to date confirms that the equations can be expressed as regressions. However, the traditional cube-root relationship that exists between effort and duration in most estimation models does not accurately predict Web development schedules [16].

Dr. Barry Boehm of the University of Southern California has proposed a square-root relationship for small projects [17]. Larry Putnam has published several papers arguing that such relationships are accurately represented by a fourth power tradeoff law [18, 19]. Our initial data analysis reveals that a square-root relationship exists for Web projects. However, this mathematical relationship tends to breakdown when the number of Web objects exceeds 300. Therefore, the schedule law used in our model needs to be scaled accordingly.

To estimate Web project costs, we have developed the WEBMO cost model; its mathematical formulation is shown in Figure 1. As stated, this model is an extension of the COCOMO II early design model [20]. The WEBMO model was developed using a mix of expert judgment and actual data from 64 projects using linear regression techniques. It allows users to take the characteristics of Web projects identified in Table 1 into account via adjustments that they make to its cost drivers.

WEBMO's mathematical formulation builds on the COCOMO II model's extensive data analysis of more than 161 projects to address Web issues. We compute exponents for its effort and duration equations, P1 and P2, using the following five application domains: Web-based electronic commerce, financial/trading applications, business-to-business applications, Web portals, and information utilities. As shown in Figure 1, the WEBMO estimating equations for effort (in person-months) and duration (in calendar months) assume size is provided in Web objects. To predict duration, the model assumes a square root instead of a cube-root relationship between duration and effort for small projects.

The current version of the WEBMO estimation model differs from the original COCOMO II model by having nine instead of seven cost drivers and a fixed instead of a variable effort power law.

Web Object Predictors	Low	Average	High	Notes
Traditional function point predictors				
• Internal logical files		2x10	1x15	From specification: 3 files
• External interface files		2x7		From specification: 2 interfaces
• External inputs		4x4	6x6	From specification: 10 inputs
• External outputs		3x5		From specification: 3 outputs
• External inquires				From specification: none
Number of XML, HTML, and query lines		16x4		From specification: 16 HTML lines
Number of multimedia files	1x4	13x5	1x7	Operands: audio file, 13 multimedia files, help file
	3x4			Operators: open, close, save
Number of scripts		1x3		Operands: animation script
	3x2			Operators: open, go (forward), close
Number of Web building blocks		10x4	5x6	Operands: 15 building blocks from library including 9 buttons, 1 cast, and 5 secure server icons
	3x3			Operators: find, add, and insert
TOTAL	**31**	**237**	**88**	

Table 4: *Web Object Calculation Worksheet*

Application Domain	A	B	P1	P2
Web-based electronic commerce	2.3	2.0	1.03	0.5 or 0.32
Financial/trading applications	2.7	2.2	1.05	0.5 or 0.32
Business-to-business applications	2.0	1.5	1.00	0.5 or 0.32
Web-based portals	2.1	1.8	1.00	0.5 or 0.32
Web-based information utilities	2.1	2.0	1.00	0.5 or 0.32

Table 5: *WEBMO Parametric Values*

While our goal is to be as compatible with COCOMO II as possible, we had to deviate because of observed colinearity between cost drivers when we performed our regression analysis. Such colinearity means that some of the cost drivers can not be assumed to be independent from others. In response, we have treated them and COCOMO's scale factors differently in our mathematical formulation.

The constants in the effort and duration equations and power laws for each of the five application domains that we have studied are summarized in Table 5. A brief explanation for each of the nine cost drivers used by the model is provided in Table 6 (see page 16). The values for the driver ratings used in the model are also provided in Table 6. Those interested in more detail on the model are referred to the WEBMO model definition manual that will be issued late this year. At that time, a version of this manual will also be made available at <www.reifer.com>.

As expected, the choice of value for the duration power law, P2, is based on the relative size of the application. For small applications less than 300 Web objects, the square-root relationship between effort and duration seems to hold (e.g., P2 = 0.5). For larger Web applications, the cube-root relationship should be used (P2 = 0.32).

The nine cost drivers replace those in the original COCOMO II model. Most represent combinations of the original factors in the early design version of the model. However, teamwork and process efficiency are new and different. They represent scale factors in the COCOMO II model that have been shown to have a statistical effect on Web estimation. However, instead of using them as power law factors in the effort estimating equation, we include them as effort multipliers to simplify the mathematics. Once we can verify the WEBMO calibration statistically using scale factors, we will revert back to the more standard version of the COCOMO II model.

Let us continue with the Java example we used previously for sizing a Web application. The 356 Web object count in Table 4 represents the size of the program that would be required for this Web application. If this were done entirely using the Java language, the program would take the equivalent of 11,392 SLOC to develop, test, and transition into operations using the Language Expansion

Figure 1: *WEBMO Estimation Equations*

$$\textbf{Effort} = A \prod_{i=1}^{8} cd_i\,(\text{Size})^{P1} \qquad \textbf{Duration} = B(\text{Effort})^{P2}$$

Where: A and B = constants cd$_i$ = cost drivers
P1 and P2 = power laws Size = # SLOC

Cost Driver	Ratings				
	Very Low	Low	Nominal	High	Very High
Product Reliability and Complexity (CPLX)	Client only, simple math and I/O, no distribution, reliability not a factor.	Client/server, some math, file management, limited distribution, easy to recover.	Client/server, full distribution, databases, integration, moderate recovery goals.	Client/server, wide distribution, math intensive, high losses due to errors.	Client/server, full distribution, collaborative, soft real-time, errors dangerous.
Values	**0.63**	**0.85**	**1.0**	**1.30**	**1.67**
Platform Difficulty (PDIF)	Rare platform changes, speedy net, no resource limitations.	Few platform changes, fast net, few resource problems.	Stable platform, net performance all right, must watch resource usage.	Platform often changes, slow, lack of resources a problem.	Platform unstable, poor performance, resources limited.
Values	**0.75**	**0.87**	**1.00**	**1.21**	**1.41**
Personnel Capabilities (PERS)	15th percentile, major delays due to turnover.	35th percentile, minor delays due to turnover.	55th percentile, few delays due to turnover.	75th percentile, rare delays due to turnover.	90th percentile, no delays due to turnover.
Values	**1.55**	**1.35**	**1.00**	**0.75**	**0.58**
Personnel Experience (PREX)	≤ 2 months, limited tool, language, and platform experience.	≤ 6 months, some tool, language, and platform experience.	≤ 1 year, average tool, language, and platform experience.	≤ 3 years, above average tool, language, and platform experience.	≤ 6 years, lots of tool, language, and platform experience.
Values	**1.35**	**1.19**	**1.00**	**0.87**	**0.71**
Facilities (FCIL)	International, no collaboration, language tools.	Multisite, some collaboration, basic CASE, some methods.	One complex, teams, life-cycle methods, good tools.	Same building, teamwork, integrated tools and methods.	Co-located, integrated collaborative method/tools, etc.
Values	**1.35**	**1.13**	**1.00**	**0.85**	**0.68**
Schedule Constraints (SCED*)	Must shorten, 75% of nominal value.	Must shorten, 85% of nominal value.	Keep as is, nominal value.	Can relax some, 120% of nominal value.	Can extend, 140% of nominal value.
Values	**1.35**	**1.15**	**1.00**	**1.05**	**1.10**
Degree of Planned Reuse (RUSE)	Not used.	Not used.	Unplanned reuse.	Planned reuse of component libraries.	Systematic reuse based on architecture.
Values	**--**	**--**	**1.00**	**1.25**	**1.48**
Teamwork (TEAM)	No shared vision, no team cohesion.	Little shared vision, marginally effective teams and teamwork.	Some shared vision, functional teams.	Considerable shared vision, strong team cohesion.	Extensive shared vision, exceptional team cohesion.
Values	**1.45**	**1.31**	**1.00**	**0.75**	**0.62**
Process Efficiency (PEFF)	Ad hoc, rely on heroes.	Project-based process, rely on leadership.	Streamlined process, rely on process.	Efficient process, best way to do job.	Effective process, people want to use it.
Values	**1.35**	**1.20**	**1.00**	**0.85**	**0.65**

*Schedule differs from COCOMO II, which is bell shaped instead of flat past its nominal value.

Table 6: *WEBMO Cost Drivers and Their Values*

Factors (LEF) listed in Table 7. Please note that the values in this table differ from those currently endorsed by the International Function Point Users Group (IFPUG). We developed these numbers empirically using our Web applications database because the IFPUG numbers did not seem to be consistent with our current experience. This count includes the volume of work required to program Java scripts and beans on both the client and server, assuming that an appropriate Java environment were available for this distributed application.

The LEF factors are used to backfire between FP and SLOC estimates. The one convention that we impose on backfiring is that only languages in the same family can be used in conjunction with each other. For example, you would not mix C and C++ counts because their syntax and semantics are quite different. Our data indicate that there is a 10 percent to 40 percent error in counting when languages are mixed across language families. As noted, HTML and System Query Language (SQL) are considered 4GL and can be mixed using the conventions that we developed.

Why is backfiring to SLOC important? The COCOMO II model uses SLOC as the basis for all of its estimates. Therefore, conversion to SLOC is required to use this popular model out of the box.

Let us run WEBMO with all of its drivers set to nominal within the Web portal domain as an example. The effort estimate assuming size is 11.4 thousand SLOC is about 24 person-months, while the duration estimate is 5.0 calendar months assuming the cube-root relationship holds because size is greater than 300 Web objects.

For comparison purposes, let us run the early design version of the COCOMO II model out of the box with all of its cost drivers set to nominal. However, to use the model, we need to calibrate the scale drivers. We will assume the following values for these parameters:

- Precedence – largely familiar.
- Development flexibility – general goals.
- Architecture/risk resolution – often (60 percent).
- Team cohesion – basically cooperative.
- Process maturity – level 1 (upper half).

With these values, COCOMO II estimates the effort at 38.8 person-months during a period of 11.4 calendar months. "Which estimate is right?" you are probably asking. As expected, neither answer is right on the mark. The actual for this project was six people for four months. Clearly, the WEBMO formulas have better predictive accuracy for this project.

Summary and Significant Research Findings

Estimating the cost and duration of Web developments has a number of challenges associated with it. To cope with these challenges, we developed new size metrics, Web objects, and an estimating model, WEBMO. We have also validated and calibrated the metric and model in anticipation of building potential products based upon them.

We prepared an initial calibration for WEBMO by combining expert opinion and actual data from 64 completed Web projects. Our goal is to improve the accuracy of our models by collecting data on at least another 30 projects during Phase II.

The following significant results/findings were outputs of our initial research efforts:

- We validated that Web objects have better predictive accuracy (r^2) than tra-

ditional function points when counted using conventions developed for that purpose. These counting conventions allowed us to extend the excellent work done by the IFPUG so that we can better handle the sizing of Web applications.
- We increased the statistical accuracy of our WEBMO estimating model from 30 percent of the actual experience at least 60 percent of the time (using a 32-project database of actuals) to 20 percent of the actual experience at least 68 percent of the time (using our expanded 64-project database of actuals).
- We validated that a square root instead of a cube-root relationship exists between effort and schedule for Web application projects whose size was less than 300 Web objects.

These results are substantial because they indicate that the Web objects and the WEBMO estimating model can help address the gaps in the estimating technology that we summarized in Tables 1 and 2.

Acknowledgment

A major part of the research reported within this paper was funded by the National Science Foundation under Grant DMI-0060006. Our partners also provided the data used to calibrate the model and refine its mathematical formulation.

Currently, we are looking for beta test sites and partners for our research. Anyone interested in participating in this capacity can inquire through e-mail at <info@reifer.com>.

Information on COCOMO II

For those unfamiliar with the COCOMO II, refer to the University of Southern California (USC) Web site at <http://sunset.usc.edu/research>. USC has much literature and a public version of the model available on this site.

References

1. Pelz, James P. "GE Takes to the Net to Lower Company Costs." Los Angeles Times 9 Oct. 2000: C1-C5.
2. Highsmith, Jim, and Alistar Cockburn. "Agile Software Development: The Business of Innovation." IEEE Computer Nov. 2001: 120-122.
3. Beck, Kent. Extreme Programming Explained. Addison-Wesley, 2000.
4. Heineman, George T., and William T. Councill. Component-Based Software Engineering. Addison-Wesley, 2001.
5. Fowler, Martin, Kent Beck, John Brant, William Opdyke, and Don Roberts. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999.
6. Kruchten, Philippe. The Rational Unified Process. Addison-Wesley, 1998.
7. Boehm, Barry W. "Transitioning to the CMMI via MBASE." Southern California SPIN Meeting Presentation. University of Southern California, Jan. 2000. Available at: <http://sunset.usc.edu>.
8. Reifer, Donald J. "Web Development: Estimating Quick-to-Market Software." IEEE Software Nov./Dec. 2000: 57-64.
9. Paulk, Mark C., "Extreme Programming from a CMM Perspective." IEEE Software Nov./Dec. 2001: 19-26.
10. Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrisis. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, 1995.
11. Ahern, Dennis M., Aaron Clouse, and Richard Turner. CMMI Distilled. Addison-Wesley, 2001.
12. Lorenz, Mark, and Jeff Kidd. Object-Oriented Software Metrics. Prentice Hall, 1994.
13. Boehm, Barry W., Chris Abts, A. Winsor Brown, et al. Software Cost Estimation with COCOMO II. Prentice Hall, 2000: 192-6.
14. Dreger, J. Brian. Function Point Analysis. Prentice Hall, 1989: 5.
15. Reifer, Donald J. "Web Objects Counting Conventions." Reifer Consultants, Mar. 2001. Available at: <www.info@reifer.com>.
16. Brown, A. W. "CORADMO." 13th International COCOMO Forum and Focused Workshop on COCOMO II Extensions. Oct. 1998.
17. Boehm, Barry W. "COCOMO II Overview." 14th International COCOMO Forum. Oct. 1999.
18. Putnam, L. H. "A General Empirical Solution to the Macro Software Sizing and Estimating Problem." IEEE Trans. Software Engineering. SE-4 July (1978): 345-61.
19. Putnam, L. H., and D. T. Putnam. "A Data Verification of the Software Fourth Power Tradeoff Law." International Society of Parametric Analysts Conference. 1984.
20. Boehm, Barry W., Chris Abts, A. Winsor Brown, et al. Software Cost Estimation with COCOMO II. Prentice Hall, 2000: 51-55.

Language	LEF
1GL default	320
C	128
2GL default	107
COBOL (ANSI85)	91
FORTRAN 107	107
PASCAL	91
3GL default	80
C++	53
Java for Web	32
LISP	64
ORACLE	38
Visual Basic	40
Visual C++	34
Web default — visual languages	35
OO default	29
EIFFEL	20
PERL	22
Smalltalk	20
Web default — OO languages	25
4GL default	20
Crystal Reports	20
Program generator default	16
HTML	15
SQL for Web	10
Spreadsheet default	6
Excel	6
Screen Painter	6
5GL default	5
XML	6
MATHCAD	5

Table 7: *Language Expansion Factors (LEF)*

About the Author

Donald J. Reifer is one of the leading figures in the fields of software engineering and management, with more than 30 years of progressive experience in government and industry. In that time, he has served as chief of the Ada Joint Program Office and the director of the Department of Defense Software Reuse Initiative. He is currently the president of Reifer Consultants, Inc., which specializes in helping clients improve the way they do business. Reifer's many honors include the American Institute of Aeronautics and Astronautics Software Engineering Award, the Secretary of Defense's Medal for Outstanding Public Service, the NASA Distinguished Service Medal, the Frieman Award, and the Hughes Aircraft Fellowship.

P.O. Box 4046
Torrance, CA 90505
Phone: (310) 530-4493
Fax: (310) 530-4297
E-mail: d.reifer@ieee.org