

Web Development: Estimating Quick-to-Market Software

Donald J. Reifer, President
Reifer Consultants, Inc.

Over the years, I have become confident in my ability to estimate software costs and schedules. I guess that such confidence comes to us in the industry who have developed hundreds and hundreds of estimates. Of course, we have inserted mature processes, metrics and estimating models to accomplish this feat in a disciplined and repeatable manner. We have invested in data collection and tuned the processes, refined the metrics and calibrated the models in order to achieve our goals of predictability and control.

During the past few months, I found a way to get rid of my self-assurance. It was simple; I tried to estimate the cost and schedule for a web development. Developers seem to generate software for the web using hypertext markup language (html), Java applets/script and visual programming languages in a flash of an eye. Actually, such projects defeat my processes, defy my models and make my size metrics obsolete. They generate and make thousands of web objects operational over periods of just a few months. As we move to the web and embrace electronic commerce business models, how do we estimate the software project costs and schedules? How do we examine the breakeven points and justify the investments needed to bring in the bounty from electronic commerce? More important, how do we adapt our existing processes, size metrics and models and make them work? The focus of this paper is to provide answers to these questions.

1. Characterizing Web Development Projects

I noticed a banner on my web site that announced that last year electronic commerce reached \$3 billion in sales last year. That's a marvelous achievement. But, in good news there is also bad. And the bad news is that the headline heralds in a change to the way we go about developing software. Table 1 summarizes these changes. It highlights the move to component-based software development, systematic reuse and visual technologies. It identifies a move to quick paced developments. Instead of developing software from requirements via the waterfall, these modern web development projects glue building blocks and reusable components together using rapid application development methods and continuous prototyping. You will also notice that web developments seem deficient in the areas of process and estimating. That's because the momentum to move to the web is new and technology like the Software Engineering Institute's Capability Maturity Model (CMM)³ has not been adapted to address such projects.

In the area of estimating, you will also notice that Software Line Of Code (SLOC) and function-point estimating models like COCOMO⁴, PRICE-S⁵, SLIM⁶ and SEER⁷ are the mainstay tools used for traditional developments. The reasons for this are simple and include:

- The phenomenology associated with development and the parameters that drive cost have been extensively studied,
- Estimating models that take this phenomenology into account have been developed, validated and commercialized over a period of 20+ years, and
- The models have been calibrated using historical data to accurately predict cost and schedule.
- Processes that incorporate the models into the planning and control processes firms use to manage their businesses have been developed, refined and optimized.

Table 1 Characteristics of Traditional Versus 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)
Typical timeline	12-18 months	3-6 months
Development approach employed	Classical, requirements-based, phased and/or incremental delivery, use cases, documentation-driven	Rapid application development, gluing building blocks together, prototyping, Rational Unified Process ¹ , MBASE ²
Primary engineering technologies used	Object oriented methods, generators, modern programming languages (C++), CASE tools, etc.	Component-based methods, 4 th and 5 th 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	Object-based systems, many reusable components (shopping carts, etc.), few external interfaces, relatively simple
People involved	Professional software engineers with lots of experience	Graphic designers, less experienced software engineers
Estimating technologies used	SLOC or function point-based models, WBS approach for small projects	Wing it

2. Addressing the Estimating Challenges

We in the estimating community currently have not agreed on how to develop estimates for web-based projects. The trouble is that the characteristics of the web-based projects that we listed in Table 1 make it difficult for estimators to adapt and put existing processes, metrics and models to work operationally. To highlight the challenges that we in the community face in the area of web estimation, I've constructed Table 2. For comparative purposes, this Table also identified the approaches traditional projects use to develop their estimates.

Table 2 Web-Based Estimating Challenges

	Traditional Approach	Web-Based Challenges
Estimating process	Most use analogy supplemented by lessons learned gleaned from past experience	Job costing done ad hoc based upon 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, building blocks, 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 by breaking the job down into tasks and identifying what is needed to do the work. Little history is available.
Schedule estimation	Schedule is estimated using a cube root relationship with effort.	Schedules estimates using cube root relations are an order of magnitude high.
Model calibration	Measurements from past projects are used to calibrate models to improve accuracy ⁸	Measurements from past projects are used to identify folklore (too few to be used yet)
“What if” analysis	Estimating models are used to perform “what if” and risk analysis. They are also used to compute return-on-investment (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 remain an open challenge.

3. New Size Metrics

Many professionals state they would like to use the more traditional processes, metrics and models for estimating web projects. However, as noted by Table 2, these traditional approaches don't seem to address the challenges facing the field. Estimators state that their major concern is size because it drives most of their models. But, new size metrics are needed to accurately scope the work involved in projects that currently cannot be accurately estimated with Source Lines of Code (SLOC) or Function Points (FP).

The first major question that needs to be resolved is "How do I measure size?" Most working web projects agree that SLOC's may not be suitable for early estimation because they are design-based and FP's may be inappropriate because applications do more than transform inputs to outputs. In response, dozens of size metrics have been proposed for web development⁹ (object points, application points, multimedia points, etc.). The only finding that researchers in the field seem to agree upon is that they can't reach agreement on which of these is best.

It is now time for me to cloud the water even more. Based upon my research, I believe I have developed yet another size metric that I believe resolves the current controversy. My proposed metric, web objects, computes size by taking each of the many elements that make up the web application into account. The metric computes size using Halstead equation¹⁰ for Volume (i.e., a proposed measure of size that is language independent and is related to the vocabulary used to describe it in terms of operands and operators) as follows:

$$V^* = N \log_2 (n) = (N_1^* + N_2^*) \log_2 (n_1^* + n_2^*)$$

Where: N = number of total occurrences of Operands/Operators
n = number of distinct Operands/Operators
 N_1^* = total occurrences of Operand estimator
 N_2^* = total occurrences of Operator estimator
 n_1^* = number of unique Operands estimator
 n_2^* = number of unique Operators estimator
 V^* = volume of work involved represented as web objects

Using the predictors listed in Table 3 to compute the number of web objects, we have been able to predict the size of a web application in a repeatable and robust manner. These predictors allow us to take into account the elements that contribute to the size of a web application. Each predictor can be represented by the unique number of operands and operators that they contribute to the application. Like function points, the key to developing repeatable predictor counts is a well-defined set of counting conventions. We can achieve consistency across organizations and resolve conflicts as size estimates are formulated using such standards. Such counts by their very nature must clearly separate operands from operators because the latter represent what we do to an object, not what the object does. Table 3 also provides examples of operands and operators to clarify what is and isn't counted as we develop our web object estimates. In addition, we have provided a worksheet as Table 4 that we developed to weight the predictors to reflect the actual data we collected on 32 completed projects. The worksheet weightings were developed in a manner similar to that employed by Alan Albrecht and John Gaffney when they validated how well function points fit their data using a software science approach¹¹.

Table 3 Web Based Predictors of Size

Web Object Predictors	Example Operands	Example Operators
# Building blocks	Fine grained components (ActiveX, DCOM, OLE, etc.), widgets, etc.	Create, apply, call, dispatch, interface, terminate, etc.
# COTS components (includes any wrapper code)	Programs, library routines, etc. (supplied by commercial vendors)	Initiate, terminate, apply, bind, customize, export, wrap, etc.
# Function points (unadjusted)	Input screens, output reports, logical files, external interfaces, etc.	Transform (inputs to outputs), access, generate, interface, etc.
# Multimedia components	Text, video, sound, 3D objects, plug-ins, meta-tags, html lines, etc.	Create, cut, paste, clear, edit, animate, broadcast, etc.
# Object or application points ^{1,2}	# server data tables, # client data tables, percent reuse, etc.	Transform (inputs to outputs), access, generate, modify, etc.
# Query language lines	# high level query language statements	Browse, find, search, retrieve, optimize, etc.
# Reusable components	Algorithms, designs, programs, etc.	Create, aggregate, apply, call, interface, terminate, etc.
# Visual language scripts or macros	Aggregation, delegation, containers, run-time support, etc.	Create, store, distribute, serialize, edit, generate, etc.
Other (templates, sgml, etc.)		

Table 4 Web Object Calculation Worksheet

Web Object Predictors	Low	Complexity-Weight Average	High
# Building blocks	1	2	4
# COTS components	2	4	6
# Function points (unadjusted)	*	*	*
# Multimedia components	1	2	4
# Object or application points	*	*	*
# Query language lines	2	3	4
# Reusable components	2	4	6
# Visual language scripts & macros	1	2	3
Other	2	4	6

* We assume weights have already been applied (would result in double counting)

To use this worksheet, we must first identify the elements that contribute to the job. We would start by selecting those items listed in the predictor column that apply. If the item we need to size doesn't appear in the column, then we would use "other" to account for it. Next, we would determine how each of these elements contributes to the total size by counting the unique number of operands (i.e., the objects) and operators (i.e., actions that can done to the object) involved in the application. Then, we would classify each set of operands/operators in terms of its complexity and enter the number into the appropriate column in the worksheet. Next, we would apply the complexity ratings and compute the total number in each column. Finally, we would compute the number of web objects by summing the sums across the three columns.

Computing size in this manner provides us with important advantages. First, the metric used has a solid mathematical foundation. Second, it can be easily extended to include new predictors as new elements are introduced for web applications (e.g., video markup languages, motion). Finally, the approach allows us to address the unique characteristics of web-based developments.

4. New Models

Having a realistic metric for size is just the first step in developing a model that accurately estimates web development costs and schedule. The mathematical issues associated with predicting effort and duration need to be reconciled before such models are launched. The major issues revolve around the form of the mathematical equations and the schedule law. Analysis of data reveals 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 seem to accurately predict web development schedules¹³. Dr. Boehm of the University of Southern California (USC) is looking with a square root relationship¹⁴. Larry Putnam has published several papers where he argues that such relationships can be represented by a fourth power trade-off law.^{15,16} Our initial data analysis reveals that the square root relationship exists for projects whose size is less than 100 web objects. For larger projects, the cube root relationship seems to result in a better fit. Such a variable schedule law relationship based on effort is expected as software science scales effort as it is computed as a function of length and volume to predict duration. One of the open issues we will continue to research as we gather additional project data is how the equations scale for different sized projects.

Now that these mathematical issues are out of the way, let's take a good look at the model that we propose for estimating web development costs. The model was developed using a mix of expert judgment and data from 32 projects using regression analysis. Its mathematical formulation is based upon parameters from both the COCOMO II and SoftCost-OO¹⁷ software cost estimating models. Exponents for both equations have been computed by segmenting the estimating trends into the following three domains: web-based electronic commerce, financial/trading applications and information utilities. The estimating equations for effort (in person-months) and duration (in calendar months) are as follows:

WEBMO Model	
8	
Effort = A $\prod_{i=1}^8$ cd_i (Size)^{P1}	Duration = B(Effort)^{P2}
Where: A and B = constants	cd _i = cost drivers
P1 and P2 = power laws	Size = # web objects

The values for all of the parameters in the model except the cost drivers are summarized by relevant domain in Table 5. Tables 6-9 provide a quick rating scheme for each of our cost drivers, while Table 10 provides the values for each of the settings. The cost drivers include:

- PERS – Personnel Capability (skills, knowledge and abilities of the workforce)
- PREX – Personnel Experience (the breadth and depth of the team's experience)
- TEAM = Teamwork (the ability to synergistically as a team)
- CPLX – Product Complexity (complexity of application being developed)
- PEFF – Process Efficiency (streamlined for the business)
- PDIF – Platform Difficulty (volatility of platform and network servers)
- FCIL – Facilities (tools, equipment and co-located facilities)
- SCED – Schedule (degree of risk taken to shorten duration)

You will notice that we do not convert our size estimates from web objects to SLOC in the equation for effort. The reason for this is that we used our initial data set to calibrate directly the relationships that existed between size in web objects and effort in person months. Another of the open issues that we will research is whether it makes sense to develop backfiring ratios from web objects to/from SLOC as the function point community has done¹⁸.

Table 5 Web Development Model Parameter Values

	A	B	P1	P2
Web-based electronic commerce	2.3	2.0	1.03	*
Financial/trading applications	2.7	2.2	1.05	*
Information utilities	2.1	2.0	1.00	*

* Either 0.5 or 0.33 depending on the scaling.

Table 6 Rating Scale for PERS, PREX, TEAM, PEFF and PDIF

Driver	VL	L	N	H	VH
PERS	35 th percentile	50 th percentile	65 th percentile	75 th percentile	90 th percentile
PREX	≤ 3 months	6 months	1 year	3 years	5 years
TEAM	Limited	Some	Supportive	Cooperative	Synergistic
PEFF	Totally confused and ad hoc	Bureaucratic	Best industry practices	Streamlined	Optimized for the business
PDIF	- Many changes - Unstable net - Poor connectivity	- Frequent changes - Unstable net - Frequent loss of connectivity	- Stable - Stable net - Good connectivity	- Few changes - Speedy net - Rare loss of connectivity	- Rare changes - Speedy net - Best possible connectivity

Table 7 Rating Scale for CPLX

Driver	VL	L	N	H	VH
CPLX	- Client side - No distribution - Invocation - Simple math - Simple I/O - Simple internal interfaces	- Client/server - Limited distribution - Adaptation - Standard math - File management - Simple external interface	- Client/server - Fully distributed - Integration - Statistics - DBMS - Lots of interaction	- Client/server - Broad distribution - Wrapping - Math intensive - Distributed DB - Intensive interaction	- Client/server - Widespread distribution - Binding - Soft real-time - Persistent DB - Highly coupled

Acronyms: DB = Database DBMS = Database Management System

Table 8 Rating Scale for FCIL

Driver	VL	L	N	H	VH
FCIL	- International - Phone/fax - Ad hoc methods - Language tools - Essentially no collaboration	- Multi-site - Phone/email - Phase dependent methods - Basic CASE - Some collaboration	- One complex - LAN - Life cycle methods - Tools support methods - Collaboration	- Same building - WAN - Integrated methods - Integrated toolset - Collaboration	- Co-located - Broadband - State-of-the-art method - Toolset supports collaboration

Table 9 Rating Scale for SCED

Driver	VL	L	N	H	VH
SCED	75% of nominal	85%	100%	130%	160%

Table 10 Values for Cost Drivers

Driver	VL	L	N	H	VH
PERS	1.55	1.33	1.00	0.70	0.55
PREX	1.35	1.17	1.00	0.87	0.69
TEAM	1.45	1.27	1.00	0.75	0.63
CPLX	0.75	0.83	1.00	1.30	1.47
PEFF	0.70	0.85	1.00	1.15	1.30
PDIF	0.75	0.87	1.00	1.12	1.27
FCIL	1.41	1.21	1.00	0.83	0.71
SCED	1.35	1.15	1.00	1.07	1.15

Several of these rating differ greatly from the original models. For example, SCED is flat when extended past its estimated duration in COCOMO II. But, our data shows that schedule adheres to a bell shaped curve. This is consistent with the similar factor in the SoftCost-OO model. In other words, it costs more to compress and to elongate the estimated duration.

5. Next Steps

There are still a large number of open issues. We plan to address them by gathering and analyzing more data from completed web development projects. We plan on working with clients to collect the data we need to resolve these issues and improve the model's estimating accuracy. We can currently estimate web development projects within our database within 30 percent, 60 percent of the time by applying the segmented databases. Our goal is to improve this accuracy incrementally using project data to refine ratings developed via expert opinion. This will take us about a year to accomplish.

6. Summary and Conclusions

As we have shown, estimating the cost and duration of web developments is challenging and subject to error. To cope with the changes afoot in the industry, new size metrics and estimating models are needed. We have developed such a metric, web objects, and cost model, WEBMO, and are in the process of validating, calibrating and commercializing potential products that use these innovations as their basis. WEBCO uses a variable power law that is a function of effort and size in web objects to predict duration. We have also prepared an initial calibration for the model by combining expert opinion and actual data from 32 completed web development projects using regression analysis. But the accuracy of the model must be improved for commercialization. Our goal is to gather data on at least another 30 projects so that we can improve the model's accuracy from within 30 percent of the actuals at least 60 percent of the time to within 20 percent of actuals at least 80 percent of the time. Additional papers are planned to share the results of our continuing research with the community.

References

1. Kruchten, P., *The Rational Unified Process*, Addison-Wesley, 1999.
2. Boehm, B.W., "Transitioning to the CMMI via MBASE," *SoCal SPIN Meeting Presentation*, University of Southern California, January 2000.

3. Paulk, M. C., Weber, C. V., Curtis, B. and Chrises, M. B., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, 1995.
4. Boehm, B.W. et. al., *Software Cost Estimation with COCOMO II*, Prentice-Hall, to be published in June 2000.
5. Park, R. E., "The Central Equations of the PRICE Software Cost Model," *4th COCOMO User's Group*, November 1988.
6. Putnam, L.H. and Myers, W., *Measures of Excellence*, Prentice-Hall, 1992.
7. Department of Defense, *Parametric Cost Estimating Handbook*, Fall 1995.
8. Ferens, D. V. and Christensen, D.S., "Does Calibration Improve Prediction Accuracy?," *CrossTalk*, Vol. 13, No. 4, April 2000, pp. 14-17.
9. Cowderoy, A.J.C., "Size and Quality Measures for Multimedia and Web-Site Production," *14th International COCOMO Forum*, October 1999.
10. Halstead, M. H., *Elements of Software Science*, Elsevier North Holland, 1977.
11. Albrecht, A.J. And Gaffney, J.E. Jr., "Software Function Points, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, Vol. SE-9, No. 6, 1983, pp. 639-47.
12. *COCOMO II Reference Manual*, University of Southern California, 1999.
13. Brown, A. W., "CORADMO," *13th International COCOMO Forum and Focused Workshop on COCOMO II Extensions*, October 1998.
14. Boehm, B.W., "COCOMO II Overview," *14th International COCOMO Forum*, October 1999.
15. Putnam, L. H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering*, Vol. SE-4, No. 4, July 1978, pp. 345-61.
16. Putnam, L. H. and Putnam, D. T., "A Data Verification of the Software Fourth Power Trade-off Law," *International Society of Parametric Analysts Conference*, 1984.
17. Reifer, D. J., *SoftCost-OO Reference Manual*, Reifer Consultants, Inc., 1993.
18. Jones, T.C., *Estimating Software Costs*, McGraw-Hill, 1998.

About the Author

Donald J. Reifer is a teacher, change agent, consultant, contributor to the fields of software engineering and management and author of Tutorial on Software Management, 5th Edition. He is President of Reifer Consultants, Inc. and serves as a visiting associate at the Center for Software Engineering at the University of Southern California. Contact him at d.reifer@ieee.org.