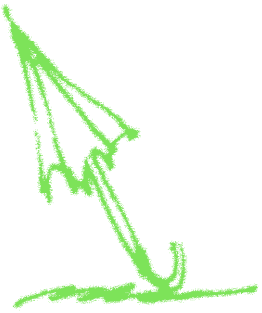


Heuristic Risk Assessment Using Cost Factors

RAYMOND J. MADACHY, *Litton Data Systems and University of Southern California*

The author describes an expert-system tool that can be used to analyze patterns of cost-driver ratings submitted for a Cocomo cost estimate. These results help users determine and rank associated sources of project risk.

Good software risk management requires human judgment, and is often difficult to implement because of the scarcity of seasoned experts and the unique characteristics of individual projects. At project inception, for example, managers who are inexperienced or who lack sufficient time to do a thorough analysis may have a vague idea that a project is risky, but will not know exactly which risks to mitigate and how. They may overlook project planning discrepancies, fail to notice risks during cost estimation, or violate consistency constraints and cost model assumptions.

Approaches for identifying risks are usually separate from cost estimation. We can improve our risk management practices by leveraging on existing knowledge and expertise during cost estimation activities through the use of cost factors to detect patterns of project risk. The automated, heuristic technique described here identifies risks in conjunction with cost estimation to help create mitigation plans based on the relative risk severities and provided advice. An automated tool identifies individual risks that an experienced software manager might recognize but often fails to take into account. It also helps calibrate and rank collections of risks, a process which many managers wouldn't do otherwise.

0740-7459/97/$10.00 © 1997 IEEE

COCOMO

The Constructive Cost Model, or Cocomo, is a widely accepted and thoroughly documented software cost model originally developed by Barry Boehm.[1] Cocomo 81 is a multilevel model that provides formulas for estimating effort and schedule using cost driver ratings to adjust the estimated effort, and is incorporated in many of the estimation tools used in industry and research. The Cocomo II research project has been underway to update the 1981 model for new development processes and products, and incorporates a revised set of cost drivers and scale drivers.[2]

The Cocomo model estimates software effort as a nonlinear function of the product size and modifies it by a geometric product of effort multipliers associated with cost driver ratings. The cost driver variables include product attributes, platform attributes, personnel attributes, and project attributes. In Cocomo II, scale drivers are attributes used to set the size exponent. Scale drivers replace the notion of development modes in Cocomo 81, and were originally used in Ada Cocomo.[3] See Figure 2 in this article for the complete set of scale drivers and cost drivers.

The simplified post-architecture Cocomo formulation for calculating effort and schedule is shown below:

$$\textit{Effort}\ (\textit{person-months}) = \mathtt{a}\ (\textit{size})^{b} \prod_{i=1}^{\textit{no. of cost drivers}} \mathrm{EM}_i$$

$$\textit{Schedule}\ (\textit{months}) = \mathtt{c}\ (\textit{effort})^{(.33+.2(b-1.01))}\ (\textit{schedule ratio})$$

where

- a is a constant derived from project data, provisionally set to 2.5,
- *size* is expressed in thousand lines of source code, function points, or object points with appropriate conversions,
- EM_i is the effort multiplier for the ith cost driver,
- *schedule ratio* refers to the compression or expansion of the nominal schedule time, and
- b is an exponent for the diseconomy of scale dependent on additive scale drivers according to

$$b = .001 \sum_{j=1}^{\tau} \mathrm{SF}_j, \text{ where } \mathrm{SF}_j \text{ is a scale factor.}$$

The geometric product results in an overall effort adjustment factor to the nominal effort.

- c is a constant provisionally set to 3.0. Effort in the schedule equation excludes the effect of the schedule constraint effort multiplier.

As Cocomo II continues to evolve, these formulations will become more extensive to reflect different processes.

REFERENCES

1. B. Boehm, *Software Engineering Economics*, Prentice Hall, Englewood Cliffs, N.J., 1981.
2. B. Boehm et al., "Cost Models for Future Software Life Cycle Processes: Cocomo 2.0," *Annals of Software Engineering Special Volume on Software Process and Product Measurement*, J.D. Arthur and S.M. Henry, eds., J.C. Baltzer AG, Science Publishers, Amsterdam, 1995, pp. 57-94.
3. B. Boehm and W. Royce, "Ada Cocomo and the Ada Process Model," *Proc. 5th Int'l Forum Cocomo and Software Cost Modeling*, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, Oct. 1989.
4. *Cocomo II Model Definition Manual, version 1.4*, Univ. of Southern Calif., Mar. 1997.

The technique is an extension of the Constructive Cost Model, or Cocomo,[1] which supports project planning by identifying, categorizing, quantifying, and prioritizing project risks (see the accompanying boxed text). The extended method also detects cost estimate input anomalies and provides risk control advice in addition to conventional Cocomo cost and schedule calculation.

In previous research work at the University of Southern California, I developed a prototype knowledge-based assistant that focused primarily on risk assessment, provided a user-friendly interface, served as an intelligent front end to a simulation model, and performed quantitative validation. I later incorporated the technique into a cost estimation tool at Litton Data Systems.[2] The latest work builds upon those results by updating the rule base for Cocomo II cost factors,[3] refining the quantitative risk assessment scheme, and developing an application for use on the World Wide Web.

The method described here is encapsulated in a tool called Expert Cocomo II, and is intended to be used in conjunction with other project management techniques. It is being used in Litton Data Systems' intranet environment as part of its Software Engineering Exemplar process and in other industrial locations, and is currently being integrated with a suite of public-domain software engineering tools at USC to assist in system acquisition, project planning, and risk management.

COST ESTIMATION AND RISK MANAGEMENT

Cost models are often used for project planning and estimation to predict both the person effort and elapsed time of a project. Cocomo is a widely used cost model incorporating various cost factors to calculate effort. These factors can be used to assess risk, using sensitivity analysis or Monte Carlo simulation, but the approach described here uses them to infer specific risk situations. See the boxed text "Cocomo" for more details.

Risk is the possibility of an undesirable outcome or a loss. The discipline of risk management attempts to identify, address, and eliminate software risks before they become either threats to successful software operation or major sources of software rework.[4] This activity involves both risk assessment and risk control[4,5] and is ongoing throughout a development project.

Risk management tries to balance the triad of cost–schedule–performance.[4,5] Though cost, schedule, and product performance risks are interrelated, they can also be analyzed independently. Some methods used to quantify these risks include table methods, analytical methods, knowledge-based techniques, and questionnaire-based methods. In practice, risks must be identified as specific instances to be manageable.

Cost estimation and risk management are strongly connected:

- Cost estimates are used to evaluate risk and perform risk trade-offs.
- Risk methods such as Monte Carlo simulation can be applied to cost models.

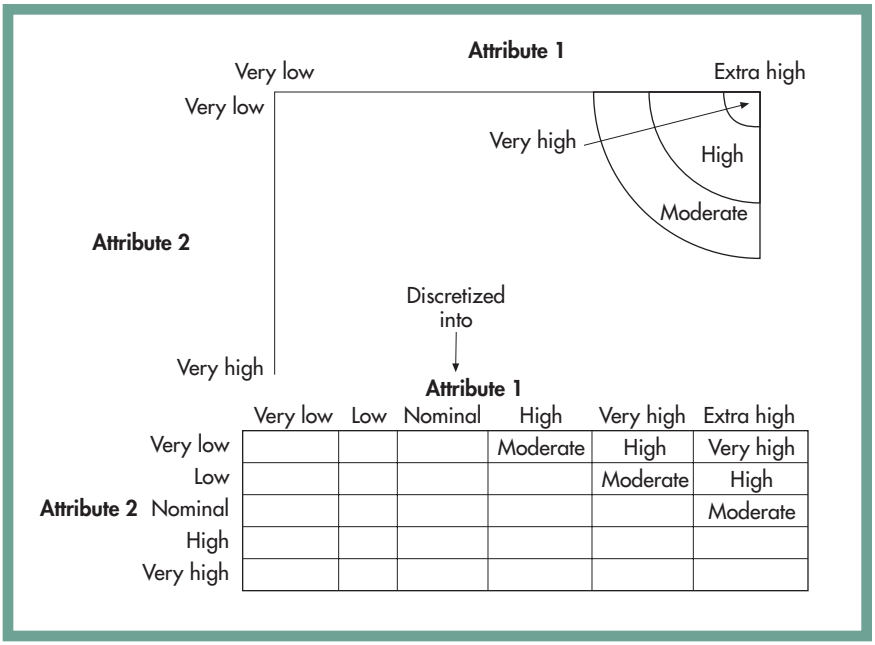


Figure 1. *Typical assignment of risk levels.*

♦ The likelihood of meeting cost estimates depends on risk management.

Another technique for combining risk assessment with cost estimation is described elsewhere in this issue, in "Integrating Risk Assessment with Cost Estimation" by Kari Känsälä. His method differs from mine, as it uses cost drivers to compute indirect risk exposure for risks that are among the cost drivers of the cost model.

Knowledge-based assistance. Knowledge-based assistants for software project management can be classified by their uses: diagnosis and classification, interpretation and analysis, system configuration, anomaly detection, design, or planning and scheduling. Expert Cocomo II is primarily for project analysis and input anomaly detection, and is being extended to provide advice for risk management planning.

Much past work has focused on automating coding activities, but recent research encompasses all software life cycle activities.[6] Improvements have been made in transformational methods, but there has been much less progress towards accumulating knowledge bases for large-scale software engineering processes. There is great potential for capturing expertise to assist in project management functions such as cost estimation and risk management. Incorporating expert-system rules can place considerable added knowledge at the disposal of software project planners and managers to help avoid high-risk development situations and cost overruns.

KNOWLEDGE ENGINEERING

Our knowledge engineering work involved choosing appropriate abstractions for formulating heuristics, iteratively eliciting expert knowledge, representing that knowledge for diagnosis, and testing the expert system. We also devised a risk quantification scheme. Cost drivers in the Cocomo model were identified very early as a complete set of attributes for project risk diagnosis. Starting with a matrix of these drivers, we identified risk situations as a combination of multiple cost attributes, and then formulated the risks into a set of rules. As such, the risk assessment scheme represents a heuristic decomposition of cost driver effects into constituent risk-escalating situations.

Knowledge was acquired from written sources on cost estimation[1,7] and risk management,[4,5] and by interviewing domain experts such as Barry Boehm and Walker Royce and some senior Litton software personnel. After several iterations of the prototype, we again engaged the experts to help refine and quantify the risks. This effort has continued during Cocomo II development.

Risk description. A *risk situation* can be described as a combination of extreme cost driver values indicating increased effort with a potential for more problems, whereas an *input anomaly* is a violation of Cocomo consistency constraints. Risks are identified, quantified, prioritized, and classified depending on the cost drivers involved and their ratings. Interactions of cost attributes that are essentially orthogonal to each other are not identified as risk situations.

One risk example is a project condition whereby the schedule is tight and the staff's applications experience is low. Cost and/or schedule goals may not be met, since time will have to be spent understanding the application domain, and this extra time may not have been planned for. An associated rule would be

> IF ((required development schedule < nominal)
> AND (applications experience < nominal))
> THEN there is a project risk.

In the next level of detail, different rating combinations are evaluated to determine the level of risk. Follow-on advice rules would provide suggestions for relaxing the schedule and improving the staffing situation.

A typical risk situation can be visualized in a two-dimensional plane as shown in Figure 1, where each axis is defined as a cost attribute rating range. As seen in the figure, the continuous representation is discretized into a table. A risk condition corresponds to an individual cell containing an identified risk level. The rules use cost driver ratings to index directly into these tables of risk levels. The tables constitute the knowledge base for risk situations defined as interactions of cost attributes.

An example of an input anomaly would be

> IF ((size > 500,000 lines of source code)
> AND (precedentedness = very low)
> AND (product complexity = very low))
> THEN there is an input anomaly.

This is because a large application that has never been developed before probably necessitates a complex solution. The size

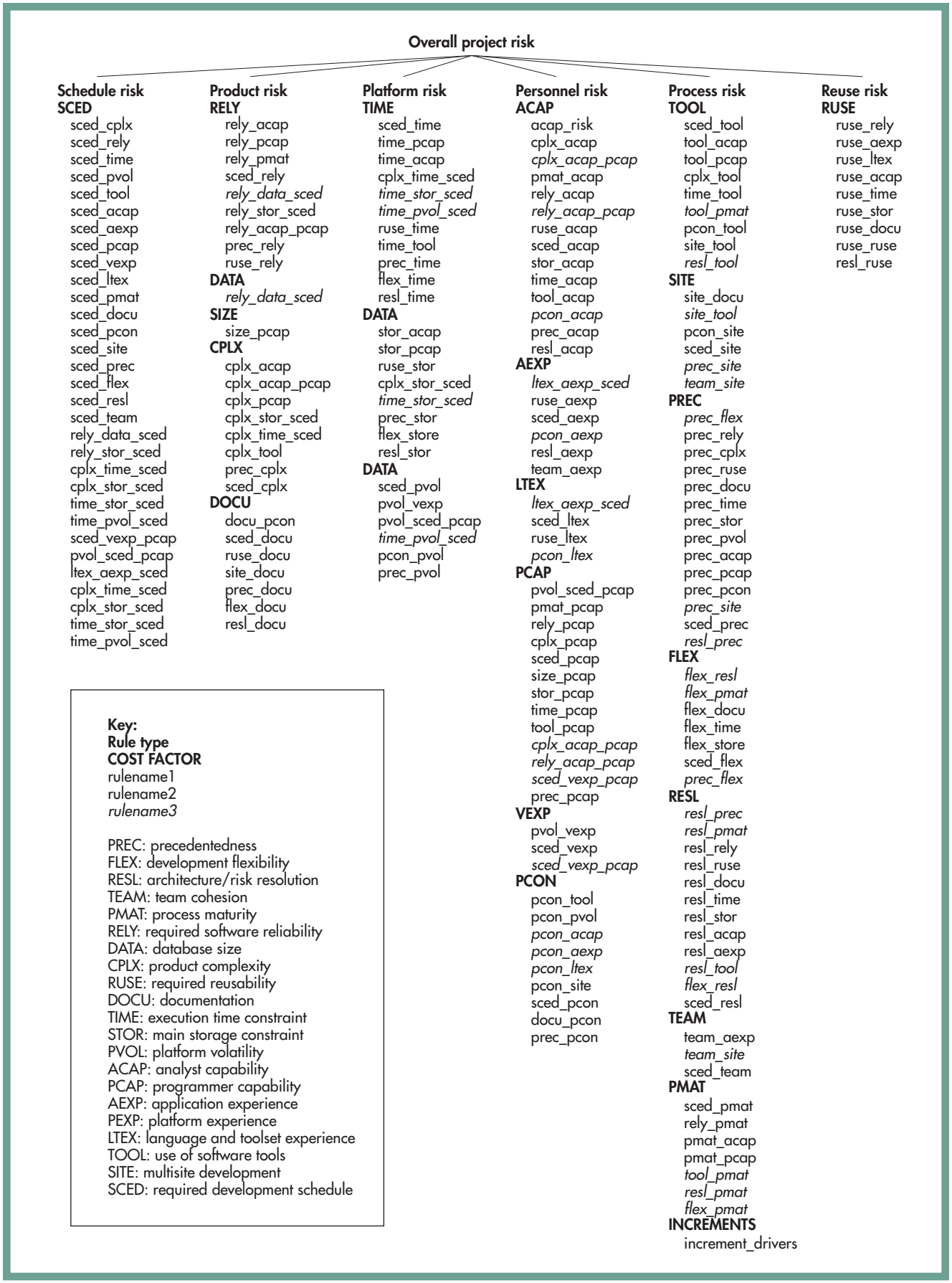
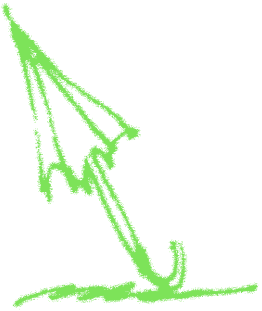


Figure 2. *The Cocomo II rule taxonomy. The rule types appear in bolded upper and lower case; the cost factors are in all caps and bold type. Rule names are listed below each cost factor. Italics denote that a rule shows up more than once within a category.*



alone induces inherent complexity. The above rule is an example of a cost estimation risk, since the inputs are inconsistent.

A RISK TAXONOMY

Each risk category is associated with various major risks. The categories are generally aligned with the cost attribute categories in Cocomo; for example, risks involving product cost factors fall into the product risk category. Most risks are identified as an intersection of cost factors in different categories, and show up in all categories involved. For instance, the risk with tight schedule and low applications experience is a component of both schedule and personnel risk.

Schedule warrants a risk category all by itself, as it is a major component of the risk triad (cost–schedule–performance). Project attributes have been lumped with scale drivers to designate process risks, and a category for reuse risks has been added. The category for cost estimation risk refers to the risk of a poor estimate, and covers those conditions found by input anomaly rules. The following are summary lessons learned.

♦ *Schedule risks.* Of all the cost factors in Cocomo, the required development schedule is involved in more risks than any other. As described below, problems trying to develop a difficult product or with less-than-desired staff are seriously exacerbated when there isn't enough time to do so. Something has to give—some combination of increased costs, overrunning the mandated schedule, or delivering an unsuitable product will occur. All other factors that might impose process time delays are identified as co-contributors to schedule risk.

♦ *Product risks.* Achieving high reliability and developing a highly complex product are difficult tasks, and the situation is seriously compounded with staffing shortfalls or shortened schedules. Solid analysis and programming skills are needed in these situations, otherwise the product goals will not be met and there will likely be cost and schedule overruns due to rework. Adequate schedule time must also be allocated to achieve reliability goals, since there are increased verification and validation activities. Incorporating a large database may entail risk due to the inherent processing complexities, especially if the problem is coupled with achieving high reliability in a compressed schedule.

♦ *Platform risks.* Volatility of the development platform can induce many headaches and rework. If the operating system, development environment, or custom hardware is immature or changing frequently, recompilation or workarounds will be necessary. New and unique errors will often surface due to "upgrades" and will require fixing. Sometimes the new problems are subtle and hard to find. In extreme cases, even changes in the product design are called for.

Since Cocomo 1981, changes in technology have minimized some of the platform risks. Memory has become a cheap commodity and computers have become much faster, so the storage constraint and time constraint factors are not of vital concern (except in certain hardware-limited situations). Additionally, turnaround time has been eliminated as a factor since batch operations are not used in modern environments.

♦ *Personnel risks.* The aggregate personnel factors have the greatest swing on overall productivity, so they are also a primary source of project risk. Interaction of personnel shortfalls with other factors are identified as risks in the other attribute categories and don't need to be repeated here. Essentially, good and experienced people should be in place for the smoothest development. An overly tight schedule will compound personnel problems tremendously.

♦ *Process risks.* Risks due to the factors in the Cocomo project attribute category have been renamed as process risks in this assessment scheme. Additionally, scale drivers in Cocomo II fall in this class of risks. The use of tools and work methods have a large influence on meeting project goals. Risk escalates if tools are inadequate or proven work methods are not used. Not finding defects early will affect large costs and schedule in the final testing phases. A practice such as formal inspections will help to find and fix defects early when they are cheaper to do so compared to testing. If development takes place at multiple sites, essential communication becomes more difficult.

Per various process improvement models, ad hoc processes on the low end of the maturity scale are responsible for high-risk development. With a less defined and measured process, there will be much uncertainty and variations relative to initial estimates. This situation almost always results in cost and schedule overruns.

Developing an unprecedented system is also a risky proposition, especially when coupled with other substandard attributes. By definition, a low value of architecture/risk resolution means there are major uncertainties remaining, since many interfaces are not yet defined. A lack of development flexibility may impose constraints that severely hamper what is accomplished. Good team cohesion is also needed for timely and rational communication, and problem resolution.

Risk impact, or risk exposure, is defined as the probability of loss multiplied by the cost of the loss.

♦ *Reuse risks.* Reuse risk situations are those that may impact the development of reusable assets, and analysis of reuse strategies is made possible with the addition of the reuse factor in Cocomo II. A reuse goal may be distinct from or only adjunct to project commitments. Reusable assets require high reliability, so specifying a low-reliability product is

$$\text{Project risk} = \sum_{j=1}^{No.\,categories} \sum_{i=1}^{No.\,category\,risks} \text{risk level}_{ij} \times \text{effort multiplier product}_{ij}$$

Figure 3. *The risk weighting scheme. A risk level of 1 is moderate, 2 is high, and 4 is very high.*

adverse to this principle. Documentation needs are likewise high. High applications experience and analyst capability as well as a good toolset are required for reuse success. If reuse is to be performed across platforms, then timing and storage constraints may present problems.

♦ *Cost estimation risks.* Inconsistent inputs point to an estimate that needs review and revision. Using a poorly done estimate for planning may preload a project with unnecessary risks. With the replacement of scale drivers for development modes, many input anomalies for Cocomo 81 are no longer relevant. However, there are various combinations of scale driver ratings and other cost factors that indicate inconsistency.

Rule base. Currently, we have identified 94 rules dealing with project risk, 15 rules dealing with input anomalies, and a handful that provide advice. There are nearly 600 risk conditions, or discrete combinations of input parameters, that are covered by the rule base.

Knowledge is represented as a set of production rules for a forward-chaining inference engine. Figure 2 shows the Cocomo II rule taxonomy and corresponding risk taxonomy, as previously described. The four-letter identifiers in the rule names are standard abbreviations for the Cocomo cost drivers and scale drivers. For each risk category, the cost drivers involved for the particular risk type are shown in boldface. Note that most rules show up in more than one category. Not shown are the cost factors and rules for input anomalies (cost estimation risk) and advice. We are now attempting to extend our assessment technique by generating specific advice for each project risk condition.

RISK QUANTIFICATION

Risk impact, or risk exposure, is defined as the probability of loss multiplied by the cost of the loss. A quantitative risk weighting scheme accounts for the nonlinearity of the assigned risk levels and cost multiplier data to compute overall risks for each category and for the entire project according to the equation in Figure 3, where effort multiplier product = (driver 1 effort multiplier) × (driver 2 effort multiplier) × … × (driver *n* effort multiplier). If the risk involves a schedule constraint, then the effort multiplier product = ((schedule constraint effort multiplier) / (relative schedule)) × (driver 2 effort multiplier) × … × (driver *n* effort multiplier).

In cases where a risk involves a scale driver, the effective effort multiplier for the scale driver is of the form $size^{.01 \times SF_i}/size^{.01 \times 3}$, where SF_i corresponds to the scale driver ratings from extra high to very low. Risks associated with scale drivers will correspond to conditions when those drivers are rated very low or low, so SF_i will take on a value between 4 and 5. The denominator is for the nominal condition. Thus, the delta exponent will be on the order of either .01 for low or .02 for very low.

The risk level corresponds to the nonlinear relative probability of the risk occurring, and the effort multiplier product represents the cost consequence of the risk. The product involves those effort multipliers involved in the risk situation. Individual effort multipliers range from .75 to 1.66, with values above the nominal multiplier of 1.0 being involved in the risk calculations. When the risk involves a schedule constraint, the product is divided by the relative schedule to obtain the change in the average personnel level, since the staffing profile is compressed into a shorter project time. The risk assessment calculates general project risks, indicating a probability of not meeting cost, schedule, or performance goals.

The risk levels were normalized to provide meaningful relative risk indications. Sensitivity analysis was performed to determine the sensitivity of the quantified risks with varying inputs, and extreme conditions were tested. A normalized scale from 0 to 100 with benchmarks for low, medium, high, and very high overall project risk was developed as follows: 0–5 low risk, 5–15 medium risk, 15–50 high risk, and 50–100 very high risk. A value of 100 denotes that each cost factor is rated at its most expensive (an unlikely project to be undertaken). The different risk categories are also normalized relative to their maximum values.

IMPLEMENTATION

The most recent implementation is the WWW version at http://sunset.usc.edu/COCOMOII/expert_cocomo/expert_cocomo.html. It consists of an HTML interface posting data to a C program via a common gateway interface, and requires no special browser add-ons to execute. There is also a tailored version on the Litton intranet. They are consistent with the March 1997 initial public release of the Cocomo II.1997 cost and scale drivers. The model definition can be found at http://sunset.usc.edu/Tools.html.

An earlier working prototype based on Cocomo 81 runs on a Macintosh using HyperCard. Initial versions used an expert-system shell, but the prototype was recoded to eliminate the need for a separate inference engine. The risk and anomaly rule bases were updated to correspond with the new set of cost factors and model definitions in Cocomo II.

The Litton Data Systems' Software Engineering Process Group also ported the rule base to a Windows environment and has incorporated the risk assessment technique into standard planning and management practices. The tool, Litton Cocomo, encapsulates cost equations calibrated to historical Litton data for different product lines and was written in Visual Basic as a set of macros within Microsoft Excel. It is now replaced by the intranet version.

The Expert Cocomo prototype and Litton Cocomo tools evaluate inputs for risk situations or inconsistencies and perform calculations for the intermediate versions of Cocomo 81[1] and Ada Cocomo.[7]

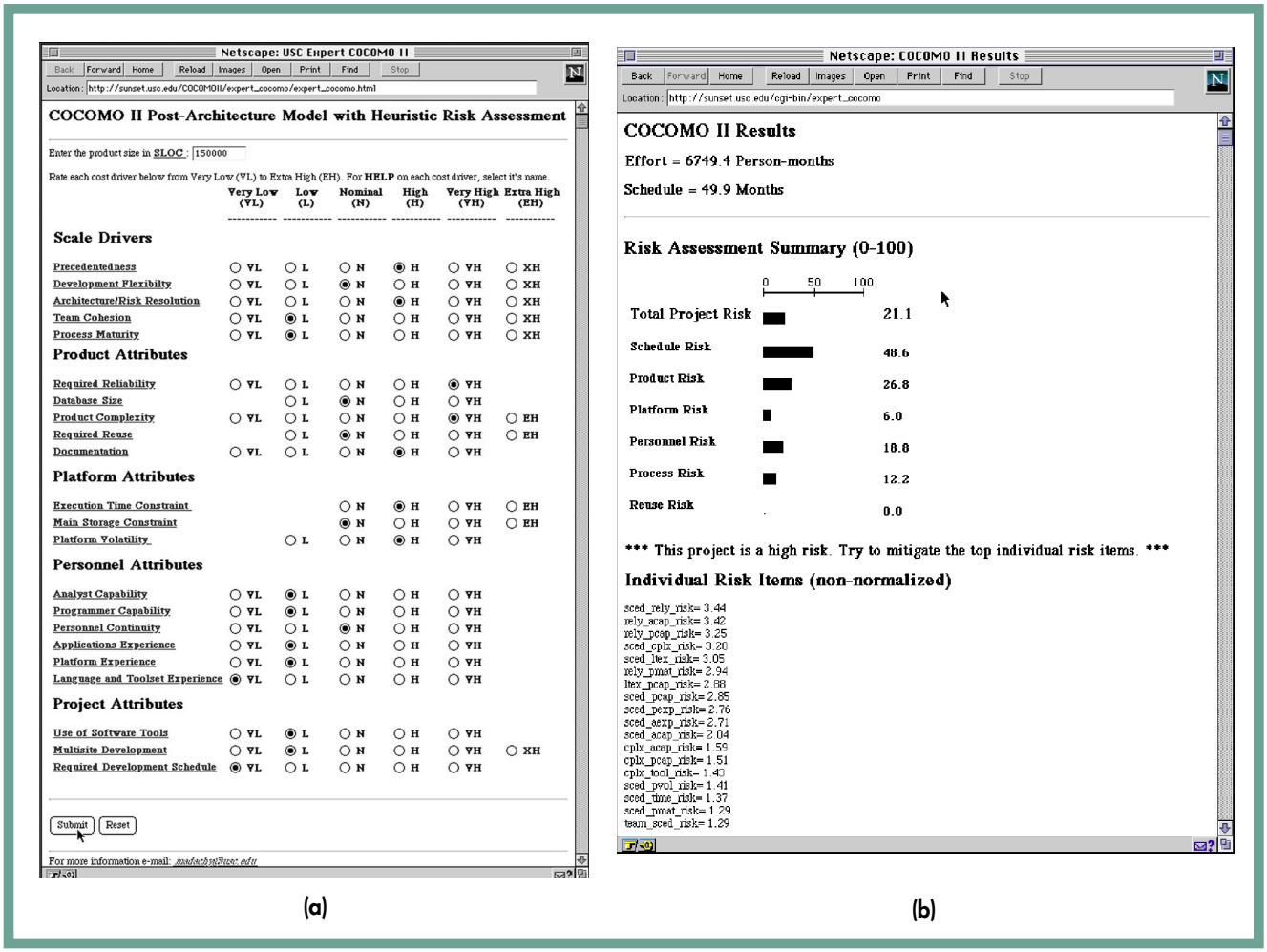


Figure 4. *(a) Sample input screen. (b) Sample risk outputs.*

The WWW version implements Cocomo II exclusively. They all operate on project inputs, encoded knowledge about the cost drivers, associated project risks, cost model constraints, and other information received from the user.

The screen snapshots in subsequent figures are from the latest WWW tool; the other implementations have similar interfaces. The graphical user interface provides an interactive form for project input using radio buttons, hypertext help, and access to the knowledge base, and provides output in the form of warnings, risk summary tables and charts, calculated cost, and schedule. The original prototype also has multiple windowing, graphs of effort phase distributions, hypertext help utilities, and several operational modes. The risk weight tables (seen at the bottom of Figure 1) are user-editable and can be dynamically changed for specific environments, resulting in different risk weights.

The following example is for a risky project where several cost drivers are rated at their costliest values. Figure 4a is the input screen showing the rated attributes for the project. This data also constitutes the input for a cost estimate. Explanatory help is suppressed; a description of the model, input definitions, and rating guidelines are also provided via hypertext links. In this example, the project has a tightly constrained schedule as well as some stringent product attributes and less-than-ideal personnel attributes. With this input data, the expert system identifies specific risk situations and quantifies them according to the formulas described earlier.

The individual risks are ranked and the different risk summaries are presented in a set of tables. The interface supports embedded hypertext links, so the user can click on a risk in a list to traverse to a screen containing the associated risk table and related information.

An example output is seen in Figure 4b, showing cost and schedule estimates, the overall project risk, risks for subcategories, and a prioritized list of risk situations. The leading subcategories of risk here are schedule, product, and personnel. Other outputs include prioritized risks in each category and a list of advice to help manage the risks. The highest risks in this example deal with schedule and reliability, and Cocomo would give the user appropriate advice.

VALIDATION

We tested and evaluated the expert system against the Cocomo 81 project database and other industrial data, and are continuing to do so with Cocomo II project data. The risk quantification is partially based on heuristic judgment, and is partially supported by statistical testing. In one test, correlation is performed between the quantified risks versus actual cost and schedule project performance. It is assumed that risks that became problems will be borne in the realized cost. Using the rule set on the Cocomo 81 database shows a correlation

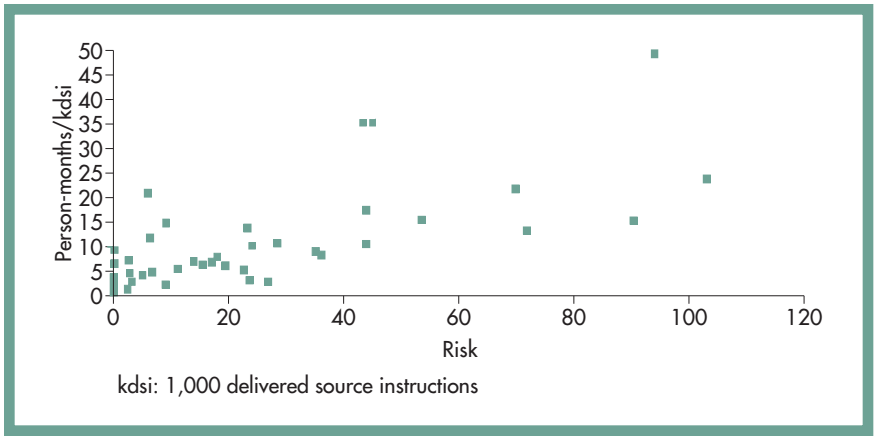


Figure 5. *Correlation against actual cost.*

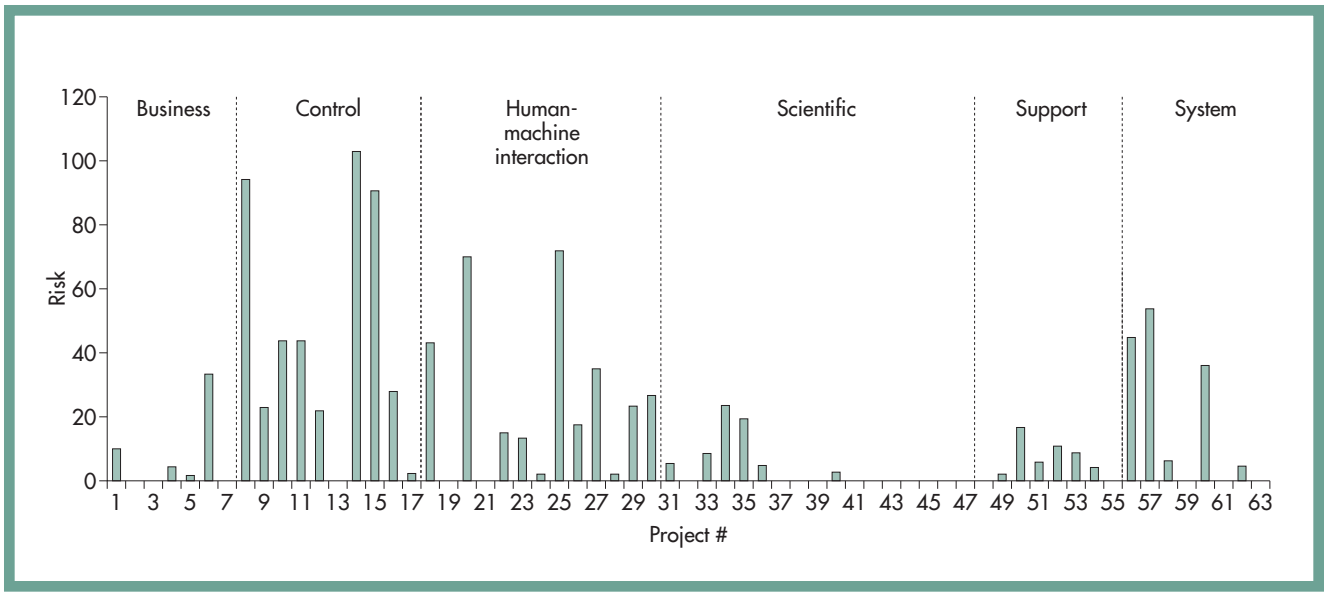


Figure 6. *Risk by Cocomo project number and grouped by project type for the Cocomo database. As you would expect, control applications are typically riskier than business or support applications.*

coefficient of .74 between the calculated risk and actual realized cost in person-months per 1,000 delivered source instructions, as shown in Figure 5. Note that the quantified risks are non-normalized in this case.

Figure 6 shows risk by project number and grouped by project type for the Cocomo database. This depiction also appears reasonable and provides confidence in the method. For example, a control application is on average riskier than a business or support application. There are updated application categories in Cocomo II, and this analysis will be repeated.

We are also using industrial data from Litton and other affiliates of the USC Center for Software Engineering for evaluation, where the calculated risks are compared to actual cost and schedule variances from estimates. We are still collecting data and performing correlation against the actual cost and schedule variance from past projects. In another test, the risk taxonomy is being used as a basis for postmortem assessments of completed projects.

Software engineering practitioners have been evaluating the system and providing feedback and additional project data. At Litton, nine evaluators consisting of the Software Engineering Process Group and other software managers have unanimously evaluated the risk output of the tool as reasonable for a given set of test cases, including past projects and sensitivity tests.

The Cocomo cost factors served well as a core set of abstractions for project risk assessment. The completeness of the attribute set for cost estimation was vital for generating a critical mass of rules from them. Common inputs between the expert system and cost model also ensured unambiguous mapping from project data to the rule set, a necessary condition for successful knowledge-based software engineering applications.[8]

Explication of risky attribute interactions helps illuminate underlying reasons for risk escalation as embodied in cost drivers, thus providing insight into software development risk. Analysis has shown that risk is highly correlated with the total effort multiplier product associated with the cost drivers, and the value of this approach is that it identifies specific risk situations that need attention.

More refined calibrations are needed for meaningful risk scales, as well as

consistency with other risk taxonomies and assessment schemes.[9] The knowledge base at Litton is being extended for specific product lines and environments to assist in consistent estimation and risk assessment, and is being updated to Cocomo II. It is part of the defined risk management process, and is used as a basis for risk assessment of ongoing projects.

Additional risk data from industrial projects will be collected and reported on during the Cocomo II project, and the technique will continue to be enhanced and refined as analysis dictates. One goal is to have the method better supported by statistical validation tests. The domain experts will continue to provide feedback and clarification.

The rule base will be extended to handle distinct development processes (incremental, evolutionary, reuse-based, and so on), cost model constraints, rating of consistency violations, and advice. Substantial additions are expected for advice to control the risks. Program updates will be provided at the WWW address.

Expert assistance can also be used to estimate size or rate cost drivers. For instance, organizational knowledge can be captured for sizing, or depicted in rating profiles for individual product lines so that estimators only need to do relative assessment of cost factors.

This work is also coordinated with other relevant research at the USC Center for Software Engineering. A working hypothesis for Cocomo II is that risk assessment should be a feature of the cost model. Toward this, graduate students at USC are incorporating the Cocomo II-updated rule base into the next revision of the public-domain USC Cocomo tool. Another risk feature in the model is the output of uncertainty ranges in lieu of point estimates.

The assessment scheme will also be incorporated as an adjunct tool to the WinWin spiral model prototype[10] to support negotiation based on Cocomo parameters. Stakeholders collaborate with the tool and negotiate project win conditions, and their trade-off decisions are supported by cost and risk analyses.

The method described here solves only part of the puzzle for project planning. Though no tool can totally replace humans in the loop for risk management, this technique goes a long way to minimize effort by killing two birds—cost estimation and risk management—with one stone, and help prevent risks from falling through the cracks. ◆

ACKNOWLEDGMENTS

I thank Barry Boehm for his guidance and inspiration in this work in addition to serving as a domain expert, Walker Royce for his time and expertise, Prasanta Bose for his comments and suggestions on earlier work, and Brad Clark for programming assistance. Thanks also to the Litton Data Systems Software Engineering Process Group personnel and management for their support.

REFERENCES

1. B. Boehm, *Software Engineering Economics*, Prentice Hall, Englewood Cliffs, N.J., 1981.
2. R. Madachy, "Knowledge-Based Risk Assessment and Cost Estimation," *Automated Software Engineering*, W. L. Johnson and A. Finkelstein, eds., Kluwer Academic Publishers, Hingham, Mass., Sept. 1995, pp. 219-230.
3. B. Boehm et al., "Cost Models for Future Software Life Cycle Processes: Cocomo 2.0," *Annals of Software Engineering Special Volume on Software Process and Product Measurement*, J.D. Arthur and S.M. Henry, eds., J.C. Baltzer AG, Science Publishers, Amsterdam, 1995, pp. 57-94.
4. B. Boehm, *Software Risk Management*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1989.
5. R. Charette, *Software Engineering Risk Analysis and Management*, Intertext Publications/Multiscience Press and McGraw-Hill, New York, 1989.
6. C. Green et al., "Report on a Knowledge-Based Software Assistant," Tech. Report RADC TR83-195, Kestrel Inst., Rome Air Development Center, N.Y., 1983.
7. B. Boehm and W. Royce, "Ada Cocomo and the Ada Process Model," *Proc. 5th Int'l Forum Cocomo and Software Cost Modeling*, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, Oct. 1989.
8. B. Boehm and P. Bose, "Critical Success Factors for Knowledge-Based Software Engineering Applications," *Automated Software Engineering*, W. L. Johnson and A. Finkelstein, eds., Kluwer Academic Publishers, Hingham, Mass., Sept. 1995, pp. 193-202.
9. G. Toth, "Automated Method for Identifying and Prioritizing Project Risk Factors," *Automated Software Engineering*, W. L. Johnson and A. Finkelstein, eds., Kluwer Academic Publishers, Hingham, Mass., Sept. 1995, pp. 231-248.
10. B. Boehm et al., "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach," *Proc. 17th Int'l Conf. Software Eng.*, IEEE Computer Soc. Press, Los Alamitos, Calif., Mar. 1995, pp. 243-253.

Raymond J. Madachy is a senior engineering specialist in the Software Technology group at Litton Data Systems. As a member of the Software Engineering Process Group, he is the division lead for software metrics, cost estimation, and risk management. He is also an adjunct assistant professor in the Computer Science and Industrial and Systems Engineering departments at the University of Southern California, and a research collaborator with the USC Center for Software Engineering.

Madachy received a PhD in industrial and systems engineering from USC, an MS in systems science from the University of California, San Diego, and a BS in mechanical engineering from the University of Dayton. He is a member of the IEEE Computer Society.

Address questions about this article to Madachy at the Center for Software Engineering, University of Southern California, Los Angeles, CA 90089-0781; madachy@usc.edu; http://www-rcf.usc.edu/~madachy.