Chapter XIV

Costs and Benefits of Software Engineering in Product Development Environments

Sorel Reisman, California State University, Fullerton

EXECUTIVE SUMMARY

A computer-based cost benefit (CBFM) forecasting model was developed to investigate possible long term effects of improved productivity that might be realized from the use of modern software engineering tools. The model was implemented in the development environment of Company X, a multinational corporation that manufactures embedded processor-based control system products. The primary purpose of the model was to generate comparative data to answer "what-if" questions posed by senior corporate management attempting to understand possible overall effects of introducing the new software development methodologies. The model provided comparative data regarding programmer labor costs, probably this company's most visible yet least understood line item in their monthly status reports. For Company X, the assumptions that were used to develop the CBFM were tailored to senior management's own priorities. Hence the model produced comparative summaries that ultimately allowed Company X to make the decision to begin implementing new software engineering strategies for product development.

BACKGROUND

Information technology plays an important role in the survival of an organization (Drucker, 1988). Coupled with the plummeting costs of computer hardware, it becomes a vital source for deriving efficient and cost-effective solutions to many of today's business problems. A well managed information system enhances a firm's ability to compete favorably and it minimizes the assumptions and guesswork in decision making that could lead to unsatisfactory performance.

In many companies information technology also shapes the process of product development (Abdel-Hamid, 1990). Organizations that are able to adapt new information technologies into their development process have often seen increased productivity and improvements in product quality. Many companies have investigated the utility of such information-based processes as CASE (computer assisted software engineering) methodologies, hoping to realize faster product development cycles, shorter production schedules, higher quality products, and lower overheads.

The cost of software development systems, like any information systems, stems directly from the cost of the resources required to provide and support the functions of the systems. The adoption of software development methodologies can be a serious strategic change. Therefore, "before management can support [software engineering] tool implementation, it must have a realistic understanding of the costs and benefits of the tools" (Smith & Oman, 1990). Cost-benefit analyses usually weigh the relationships between the costs and values of a system (Ein-Dor & Jones, 1985). Like any capital investment, the benefits must exceed the costs to justify the expense.

The economics of software engineering has often focused on software cost estimation, essentially a consideration of the costs related to single development projects. For example, software cost estimation techniques and models have tended to link software development costs to a project's size, its functional complexity, manpower requirements, and ultimately to the duration of the software development. Examples include SLIM (Putnam, 1980), COCOMO (Boehm, 1981), function point analysis (Dreger, 1989), ESTIMACS, and Price-S (Kemerer, 1987).

Because the successful implementation of software development tools requires a critical shift in senior management philosophy, it is essential to be able to justify the upheaval likely to result from these shifts. Within corporate environments such a justification is usually based on economic issues and related benefits, i.e., a sound business case. The economics of justifying these new methodologies to senior management requires a more global view of their costs and benefits than is typically found in the more detailed techniques mentioned above. That view must cut across projects, and focus on the effects of the methodologies on the totality of the development environment. In fact, as software engineering products (e.g., CASE tools) evolve into more integrated systems, they should begin more and more to address the needs of all phases of the system development life cycle.

For companies that may consider software engineering as a process that can improve the development of their own marketable products, a commitment to it should require the cooperative involvement from such diverse areas of the organization as sales, marketing, and customer service. For companies that instead consider that the methodologies may benefit their own internal systems development, the effect on the companies' own internal end user community can be substantial. In either event, the decision to