Strategies for Improving Software Productivity and Quality

Minder Chen  
*George Mason University*

Ko-En Chao  
*Ming-Chuan College of Management*

Jeffrey Chi  
*National Central University*

Follow this and additional works at: [http://aisel.aisnet.org/pacis1993](http://aisel.aisnet.org/pacis1993)

**Recommended Citation**

Chen, Minder; Chao, Ko-En; and Chi, Jeffrey, "Strategies for Improving Software Productivity and Quality" (1993). PACIS 1993 Proceedings. 60.  
[http://aisel.aisnet.org/pacis1993/60](http://aisel.aisnet.org/pacis1993/60)

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 1993 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Strategies for Improving Software Productivity and Quality

Minder Chen, Department of Decision Science and MIS
George Mason University, Fairfax VA 22030, USA

Ka-fin Chan, Department of Information Management
Ming-Chuan College of Management, Taipei, ROC

Jeffrey Yan-ping Chi, Department of Information Management
National Central University, Chung-li, Taiwan

Abstract

This paper proposes a framework to study software productivity. Several emerging software and organizational technologies have been identified and their relationships to software productivity are discussed. Future research and development in software productivity are discussed.

1. Introduction

Information-based organizations rely on their ability to manage knowledge and information effectively and efficiently to attain their strategic goals and to increase the competitiveness of their enterprises. Software productivity has not been increased over the last 30 years. The software personnel has a 4% annual increase while the demand for it is 12%. The high cost of maintaining software has prohibited organizations to allocate resources to new applications or to migrate to open systems environments.

There are several driving forces behind the search of software productivity:
1. Incentive information systems support for products and services and increased competition have made IS a critical element in business strategies and initiatives.
2. Shorter life cycles for new products and services demand on-time delivery of information systems to meet business goals.
3. Information-based organizations rely on real-time information delivery to plan, control, and manage.
4. Engineering approaches to systems development are enforced by the use of highly integrated CASE tools to support systems development processes.
5. Distributed IS functions, data-oriented design and development, and high level front-end development tools have made end-user information systems a reality.
6. The downsizing and outsourcing trends, and the scrutiny of IS budgets by business managers have pressured IS organization to measure and improve their productivity to justify its existence.

There are several major initiatives launched in the U.S. governments, private sectors, and research organizations to improve software productivity. These initiatives can be classified into two categories: software technologies and organizational technologies. Software technologies include: CASE tools, data base technologies, software reuse and object-oriented technologies. Organizational technologies include: measurement programs, software capability maturity model, business reengineering, information engineering and rapid application development. The success in implementing emerging software technologies to improve software productivity cannot be achieved without the introduction of related organizational technologies.

In this paper, we first propose a framework for studying software productivity and quality. Several emerging software and organizational technologies are then introduced. Future research and development in software productivity are discussed to conclude the paper.

2. A Framework of Studying Software Productivity and Quality

Software productivity issues should be addressed from different levels as listed in Table I: individual software developers, IS organizations, business functions, and the enterprise levels. The goals, means, and players at each level are different, however, we have to align their efforts to gain real benefits.

<table>
<thead>
<tr>
<th>Focused Level</th>
<th>IS Personnel</th>
<th>IS Organization</th>
<th>Business Function</th>
<th>Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>Improve individual productivity and quality</td>
<td>Deliver quality applications on time</td>
<td>Use IT to improve business processes</td>
<td>Enhance strategic advantages of the enterprise</td>
</tr>
<tr>
<td>Rates</td>
<td>Individual software developers and project managers</td>
<td>CIO, MIS directors</td>
<td>Functional area managers, Professional staff</td>
<td></td>
</tr>
<tr>
<td>Means</td>
<td>Integrated CASE tools, Software measurement program, Data base technologies</td>
<td>Software measurement programs</td>
<td>Business reengineering, End user information systems, Information chain analysis</td>
<td></td>
</tr>
<tr>
<td>Measurement</td>
<td>ELOCCM: FP/MM</td>
<td>Development/maintenance cost ratio</td>
<td>Return on Contribution to the value chain, Support to enterprise strategies</td>
<td></td>
</tr>
</tbody>
</table>

At the enterprise level, top management has to be involved in setting the goal of the enterprise and allocate proper resource for information systems organization. Information strategic planning is an important tool to link business strategies to information technologies.

At the business function level, the major players at this level are professional staff and line managers. Their primary goals are to use IT to improve business processes and ensure the availability of managerial information. Business reengineering and end user information systems are means to achieve these objectives. Performance at this level can be measured by return on management or activity-based costing (Strassmann, 1990).

At the IS organization level, the major players are CIO and MIS directors. Their primary concern is how to deliver quality applications on time and within budget. Building an enterprise wide information systems architecture, institutionize a software measurement program, and encourage software reuse are means to improve software productivity at this level. The performance of IS organizations are measured by
development/maintenance cost ratio and productivity and quality metrics.

3. Emerging Software Technologies for Improving Software Productivity

Several emerging software technologies that may have major impacts on software productivity are discussed in this section. They are CASE tools, data base technologies, object-oriented technologies and software reuse.

3.1 CASE Tools

System development activities generate a large amount of complicated design information that needs to be captured and analyzed. A CASE environment allows systems developers to "document and model an information system from its initial user requirements through design and implementation and let them apply tests for consistency, completeness, and conformity to standard [Chikofsky and Rubenstein, 1988, p. 13]." The use of CASE technologies is becoming a critical factor for the success of large scale systems projects.

CASE tools developed in the early 1980s are mainly designed to support structured methods that use graphical notations, such as Data Flow Diagrams for structured analysis and Structure Charts for structured design. They incorporate better analysis functions to enforce rules of the methods. Detailed design information captured in structured graphics is described in a project dictionary such that it can be shared by other CASE tools in the same environment. However, this integration of CASE tools was limited to those from the same vendor and generally within the same project. A CASE environment can be loosely integrated with its add-on tools through utilities that import or export design information to or from the project dictionary. In a few cases, vendors may link their tools by mutually-accepted formats of the interchanged data or through proprietary application program interfaces.

IE- and repository-based CASE became available in later 1980s. They offer an enterprise-wide repository system and a set of integrated toolkits that support information systems planning, analysis, design, and construction (i.e., code generation). However, IE-based CASE products are tied up with IE method and only support the development of business applications systems.

1990s will be an era of systems integration through open systems. Business needs to develop highly integrated and complicated strategic information systems faster to stay competitive. Integrated CASE environments empower IS organizations to deliver such systems to meet business needs in time and to migrate the target systems to multiple platforms in an open systems architecture. An integrated CASE environment should base on a flexible framework that provides a cost-effective tool integration mechanism, encourages portability of tools, facilitates exchange of design information, is adaptable to future methods, and is extensible to other engineering disciplines [Chen and Sibley, 1991].

3.2 Data Base Technologies

Most modern day MIS applications are centered on data base technologies. Data bases allow various MIS applications to share common data. To achieve the sharing of application data, one has to emphasize the design of an enterprise-wide data model and using the model to guide the development of each application. Data have to be model from an enterprise viewpoint instead of from an application specific viewpoint.

Currently, many data base vendors are providing fourth generation languages and data base design CASE tools to support the development of data base applications. Data base servers are becoming the key technology in a client-server environment. There are "standard" SQL access APIs that allow users to develop application using several front-end and development tools. The back-end data base servers can also be from several different vendors' products. There are so called middle-ware products, such as Pioneers Q-B, that support several data base servers. It is now possible to build new applications that are integrated with existing application via accessing data from existing data bases. With end user-oriented front-end development tools such as Microsoft ACCESS, the movement to end user information systems development will be accelerated.

3.3 Software Reuse and Object-Oriented Technologies

Software reuse has been considered as one of the most promising factors in improving the quality and productivity of software. The notion of reuse includes software components such as requirements specifications, process models, data models, and reusable codes. Organizations have to allocate resources to encourage the development of reusable components. The use of repository-based CASE tools is important to make software reuse a reality.

Object-oriented technology offers an effective mechanism for the creation and adaptation of reusable components through inheritance and encapsulation. Artificial intelligence techniques such as analogical reasoning and case-based reasoning can be applied in assisting the identification and selection of software components for reuse. Classes can be reused by simply using the public interfaces to classes. Functionalities of a class can also be extended through subclassing. Design of reuse classes is the key to increase reuse. C++ has become an industry standard in implementing object-oriented systems.

Software reuse has been recognized as one of the most promising approaches to achieving significant productivity gains. Using reusable software components not only reduces development cost, but also increases development speed and the quality of the products. The reuse concept should be extended from reuse of code to high level specifications, and to software processes.

The increasing trend of distributing IS function into business areas or of using external systems integration services (i.e., outsourcing) has created pressure for IS organizations to build up their credibility. Integrated CASE environments will be a key strategic component in support IT organizations to face their new challenges. All techniques may enable integrated CASE environments to incorporate domain specific knowledge in assisting end users to develop and maintain their own information systems using high level specification languages or diagramming tools. Eventually end users may acquire and modify high level reusable business models (i.e., data and process models) and plug them into the integrated CASE environment to generate applications.
4.1 Software Measurement

Software metrics should be designed to measure software products or software processes. Organizations should not use software measurement programs as mechanisms for individual personnel reward. They should be used to evaluate the effects of software productivity improvement programs and for software project planning. Most importantly, it can be used by the IS organizations as a justification for IT investment to top management. Software measure programs should be customized, therefore, establish historical project data base is the foundation of software measurement programs.

Software matrix can be used to measure the productivity and quality of the software sizing, staffing, and scheduling. Source Line Of Code (SLOC) and function points are two most popular software matrices. Function points analysis is a mapping of inputs, outputs, interfaces, queries, and files to estimate the size of a software project. It has been very popular matrix for MIS applications. However, it is not a reliable matrix for other types of systems. Table 2 is a comparison of SLOC and function points.

<table>
<thead>
<tr>
<th>Function Point Analysis</th>
<th>Line Of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measures the functionality of the software to the user</td>
<td>Measures the size of the software</td>
</tr>
<tr>
<td>Can be estimated at the analysis phase of the SLOC</td>
<td>Can not be measured until the program has been written</td>
</tr>
<tr>
<td>Is a unitless measure which has no physical meaning</td>
<td>Has a physical meaning</td>
</tr>
<tr>
<td>Subjective: Needs human judgment in calculating FPs</td>
<td>Objective: Can be counted by a program</td>
</tr>
<tr>
<td>Mainly applicable to business data processing applications</td>
<td>Counting LOC for nonprocedural languages is difficult</td>
</tr>
</tbody>
</table>

The following are criteria for selecting software matrix:
1. Metrics are meaningful to managers. They can be used to determine attributes of software and the software process. Metric should precisely define the measure to be used to quantity each attribute.
2. Data are reasonable to collect. For example, they can be easily obtained from project documentation or project repository.
3. Metric calculation can be easily automated.
4. Can be used to measure progress and predict results.

4.2 Software Capability Maturity Model

Software capability maturity model has been developed by Software Engineering Institute [Humphrey, 1989]. The model classifies software development process into 5 levels based on the maturity of the software processes. The characteristics and key process areas of the five levels in the Capability Maturity Model for software are summarized in Table 3. Most of organizations are at the level 1 and 2 of the model. Only few of them are at the level 3 of the model. Future U.S. government software contractors may be required to be at least at the level 2 of the model. This is one reason why this model has drawn a lot of attention.

<table>
<thead>
<tr>
<th>Level</th>
<th>Characteristics</th>
<th>Key Process Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>5: Optimizing</td>
<td>Improvement feedback into process</td>
<td>Defect prevention Technology innovation Process change management</td>
</tr>
<tr>
<td>4: Managed</td>
<td>Measured process (quantitative)</td>
<td>Process measurement Quality management</td>
</tr>
<tr>
<td>3: Defined</td>
<td>Process defined and institutionalized (qualitative)</td>
<td>Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination</td>
</tr>
<tr>
<td>2: Repeatable</td>
<td>Process dependent on individuals (intensive)</td>
<td>Requirements Management Project management Project planning Configuration management Software quality assurance Subcontractor management</td>
</tr>
<tr>
<td>1: Initial</td>
<td>ad hoc/chaotic process</td>
<td>None</td>
</tr>
</tbody>
</table>

4.3 Business Reengineering

Business reengineering employs a structured methodology built around specific steps containing clearly defined goals, deliverables and measurements that determine completion for each phase (Dastation, 1992). Most likely objectives for business process redesign include cut down costs, shorten cycle time, improve output quality, enhance quality of work life (QWL), encourage learning, empower human potentials, and improve customer satisfaction. The five phases of business reengineering are:

Phase 1: Define Business Goals

This phase begins with definition of business goals and identifying business drivers required to achieve those goals. Current business structure, organization and processes are studied. An organization then measures how much change to the business process is necessary to achieve business goals.

Phase 2: Analyze The Business Process

Phase 2 establishes a current baseline by describing the business process model. Each business process is described in term of how people actually spend their time, inputs, outs, controls, and supporting mechanisms, and resources consumption. Hidden time and nonvalue-added activities are revealed. Costs can be assigned based on actual activities. Productivity and profitability can be measured in terms of task, product and customer type.

Phase 3: Redesign The Process

There are usually three steps involved in redesigning work processes:
1. Simplification: Work processes are simplified by elimination of nonvalue-adding activities and revision of business rules to respond to current conditions and goals.
2. **Integration:** Redesigned steps into a logical and effective process that may cross functional boundaries. Increase effective interdepartmental coordination and communication. Restructuring the organization if it is necessary.

3. **Automation:** Automate business processes only when process simplification and integration have been achieved.

**Phase 4: Implement The New Process**

Often occurring in discrete stages, implementation of re-designed business processes are quantified and measured against the baseline established in Phase 2. IT lever are identified and used to support the implementation of new IT-enabled business processes [Davenport and Short, 1990].

**Phase 5: Measure The New Process**

Measurement of change against the baseline continues well after the new business process is fully in place. Reengineering is a process that begins; Reengineering is a management technique and should be an ongoing effort.

Business process reengineering is becoming the front-end of process information systems processes. Under the Corporate Information Management (CIM) initiative, DOD is requiring the reengineering of business processes before any major investment in information systems development.

### 4.4 Software Quality Standards

There are several standards for software quality including ISO 9000 series, IEEE standards, and several DOD standards (e.g., DOD-STD-2157A). Currently there are 52 countries adopted the ISO 9000 standards including Hong Kong, Malaysia, Singapore, ROC, and mainland China. Currently European Community has decided to require the compliance to ISO 9000 in its public sector procurement. The following are a series of publications related to the ISO 9000 standards:


ISO 9000-3 is the standard that is developed specifically for software. It emphasizes the importance of software verification and validation because we can view software development as contracts between customers and suppliers. The suppliers can be internal MIS departments or outside system integrators. Verification for software is the process of evaluating the software items of a given phase of a system development life cycle to ensure the consistency and correctness with respect to the software products specification and standards provided as inputs to that phase. [ISO 9000-3, 1991]. Validation for software is the process of evaluating software to ensure its compliance with specified requirements [ISO 9000-3, 1991].

Suppliers of software services should establish and maintain procedures for internal review to ensure the scope and specifications of the contract are documented in a mutually agreed terminology. Cost and risk in contract changes should be identified. ISO 9000-3 suggested several contract items that are added to the quality of a software project in a software contract [ISO 9000-3, 1991]:

1. Criteria for software acceptance testing.
2. Changes management in purchaser's requirements during the development processes.
3. Responsibilities of the purchaser in requirements specification, acceptance testing, and software installation.
4. Deliverables at each of the major milestone of the SDLC and standards and procedures to be used creating these deliverables.

### 4.5 Information Engineering and RAD

Information engineering is the application of an interrelated set of formal techniques for the planning, analysis, design, and construction of information systems on an enterprise-wide basis or across a major sector of the enterprise [Martin, 1990; 1998]. It is supported by an interrelated set of automated tools in which enterprise models, data models, and process models are built up in a repository and are used to create and maintain information systems. Therefore, the use of reusable components is encouraged. IE creates a framework for developing an information-based enterprise and ensures the alignment of an organization's business goals and strategies with its information systems. It also facilitates the long-term evolution of information systems development. End users deeply involved at each stage of the IE life cycle. However, IE should not be regarded as one rigid methodology but as a generic class of evolving methodologies.

IE consists of seven stages [11, 1990]:

1. Information Strategy Planning (ISP), in which a broad view of the information requirements of the whole business is established.
2. Business Area Analysis (BAA), in which a more detailed analysis is performed on a particular segment of the business.
3. Business System Design (BSD), in which an application system supporting a segment of a particular Business Area is described in detail without regard to the particulars of the target computing environments.
4. Technical Design (TD), in which the results of Business System Design are tailored to a specific target computing environment including hardware, OS, DBMS, and communications systems.
5. Construction, in which all of the executable components of a system are created. These components include programs, data bases (DDL & DML), screen formats, JCL, transaction definitions.
6. Transition, in which a newly constructed application system is installed in a production environment in an orderly manner.
7. Production/Maintenance, in which the enterprise realizes the full benefit of the application system as it executes to satisfy some portion of the business requirements identified during ISP.

Since following the rigid IE life cycle to implement information systems may take too long, Rapid Application Development (RAD) has been suggested as a new way of implemented applications identified in the ISP stage. RAD is a methodology designed to develop high quality information systems faster [Martin, 1991]. It capitalizes on software automation, rigorous methodology, and team-oriented approach. There are four elements in RAD:
1. Tools: Use powerful systems development tools including integrated CASE tools, desktop code generators, and rapid prototyping tools.

2. Methodology: Apply the most effective family of formalized techniques including rapid prototyping, joint application design, and timbox.

3. People: Use small SWAT teams of well-trained and highly motivated people and have high degree of end user participation.

4. Management: Impose time-pressure and give proper motivations include obtaining line-of-business managers' involvement and support and ensure top management supports to eliminate bureaucratic and political obstacles.

Table 4 lists the differences between IE and structured methods.

Table 4. Comparisons between IE and Structured Method

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Information Engineering</th>
<th>Structured Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Life cycle support</td>
<td>Full life cycle: Planning, analysis, design, and construction</td>
<td>Partial life cycle: Analysis and Design</td>
</tr>
<tr>
<td>Scope</td>
<td>Enterprise-wide applications</td>
<td>Project-by-project</td>
</tr>
<tr>
<td>Planning approach</td>
<td>Top-down planning and bottom-up implementation</td>
<td>Project-based planning</td>
</tr>
<tr>
<td>Modeling aspect</td>
<td>Data-oriented</td>
<td>Process-oriented</td>
</tr>
<tr>
<td>Applications</td>
<td>Transaction processing systems, Management reporting systems, Executive information systems, End-user computing applications</td>
<td>Transaction processing systems, Management reporting systems</td>
</tr>
<tr>
<td>Focus</td>
<td>Business and systems analysis</td>
<td>Systems analysis</td>
</tr>
<tr>
<td>Developer's role</td>
<td>Architect/Engineer</td>
<td>Technician/Artist/craftsman</td>
</tr>
<tr>
<td>Starting point</td>
<td>Enterprise modeling</td>
<td>Functional decomposition</td>
</tr>
<tr>
<td>Top mgmt. support</td>
<td>Essential</td>
<td>Optional</td>
</tr>
<tr>
<td>Tools</td>
<td>Use integrated CASE tools to link diagrams with code generators and GDL.</td>
<td>Use non-integrated CASE tools.</td>
</tr>
</tbody>
</table>

5. Conclusions

There is no panacea for improving software productivity. We need to invent in developing higher level development tools and put them into an integrated environment. Development of integrated methods to guide the development and use of tools in emerging environments are as important as tool building. Business and IS organizations also need to rebuild the overall software development environments and infrastructures, to renew their thinking about IS services within the enterprise, and to invest in their people by providing them with proper training of new technologies.

We need to conduct research to assess the impact of various factors on software productivity and to understand the interactions among these factors. From these research projects, new paradigms may emerge that help us understand more about software productivity.

6. References