COMPAS: A DEVELOPMENT-PROCESS SUPPORT SYSTEM

H. Jack Barnard and Robert B. Collicott

H. Jack Barnard *is a member of technical staff in the Development Technologies Department, and* ***Robert B. Collicott*** *is a supervisor in the Systems Development Department. They are with AT&T Bell Laboratories in Denver, Colorado. Mr. Barnard developed the first version of COMPAS and is now the project leader. He joined the company in 1980, and has a B.S. in mathematics and an M.S. in computer science from the University of Houston. Mr. Collicott joined AT&T in 1969, and was responsible for COMPAS development. He received both a B.E.E. and an M.E.E. from Cornell University.*

COMPAS is a development-process support system that provides quick access to documents and document-related information produced at discrete times throughout the development cycle. The system is designed to aid the development of documentation products; aid the planning, scheduling, and tracking of project progress; support activities that verify and validate development products as they are produced; and provide metrics about the document-development process for controlling the current project and data for planning future projects. As a result of using COMPAS, developers have achieved improvements in productivity and product quality. Also, the development cycle now is more visible to managers, and they have more control over it.

Introduction

The development of documentation for large systems is a complex, time-consuming, and error-prone process. Developers must become familiar with project standards and methods. They must acquire knowledge about requirements, related system components, and system resources. They must communicate information, coordinate their efforts with those of other developers, and participate in the verification of other work products. As systems become more complex, so does the process of developing and controlling these work products.

When support is provided during the early phases of the development process where documentation is produced, the productivity of the entire development cycle and the quality of the final product can greatly improve. More complete and correct documents can reduce the amount of effort required in technical reviews and in integration and system tests. Accurate measures of development work products (like documentation) and verification processes (like reviews) can give managers better control and lead to improvements in those processes.

COMPAS reduces the complexity of the document-development

process, provides support for process control, and provides feedback for process improvement. It helps developers acquire the knowledge they need to produce better specifications more quickly. These specifications are produced at intermediate steps in the development process and entered into COMPAS. The system provides the mechanisms needed for consistent labeling, tracking, and change control of this information. It supports the processes used to verify the quality of these specifications, and helps ensure that verification procedures are performed consistently. Development managers can obtain information about project productivity and product quality through a powerful query and report-generation facility.

In this paper, we present an overview of COMPAS and describe the major components that support the development cycle. We then discuss how COMPAS is used to support document development. Next, we describe the quality-assurance activities and in-process metrics supported for technical documentation. Finally, we address the mechanisms used to help manage the document-development process.

Panel 1. Acronyms and Terms

ANSI	American National Standards Institute
attribute	a value that describes the characteristics of a document
baseline	place a document under change control
form	a window that permits data to be entered into labeled spaces
menu	a list of selectable actions related to the current task
MR	modification request
NCSL	noncommentary source line; an executable line in the code
NOBS	nonobservable software fault
OBS	observable software fault
SABLE	software product administration system; provides change control for a project's code and documents
state	current status of a document (e.g., `draft`), represented by a status attribute
window	a software controlled, rectangular area on a terminal's screen; each window supports a different activity, within the context of an application or system, without terminating other activities

System Overview

This section provides a short history of COMPAS and describes its architecture.

Background. The COMPAS prototype was developed in 1984 as a document-management system to support the development of switching systems. Developers were encouraged to try the system and provide suggestions and feedback on its use. COMPAS quickly gained support from the development community, and users eventually started to request support for other development activities. Managers began to realize that much of the information they needed to track project progress and product quality was either available from or could easily be captured by the system.

COMPAS evolved over the years to its current state as a development-process support system. In Denver where COMPAS was developed, there are 1100 users and the system is accessed over 300 times each day. More than 65 percent of those accesses are to retrieve information. The system is now used by many AT&T entities, and easily interfaces with other development tools—such as relational databases, change-management systems, text browsers, document formatters, and text editors.

The current "look and feel" of COMPAS is a product of several years of prototyping and input from development communities. The system's supported features are implemented through an interactive menu-and-forms system that provides "pop-up" windows, context-sensitive help, and function-key usage for most types of terminals. (Panel 1 defines acronyms and terms used in this paper.) Survey results have indicated that

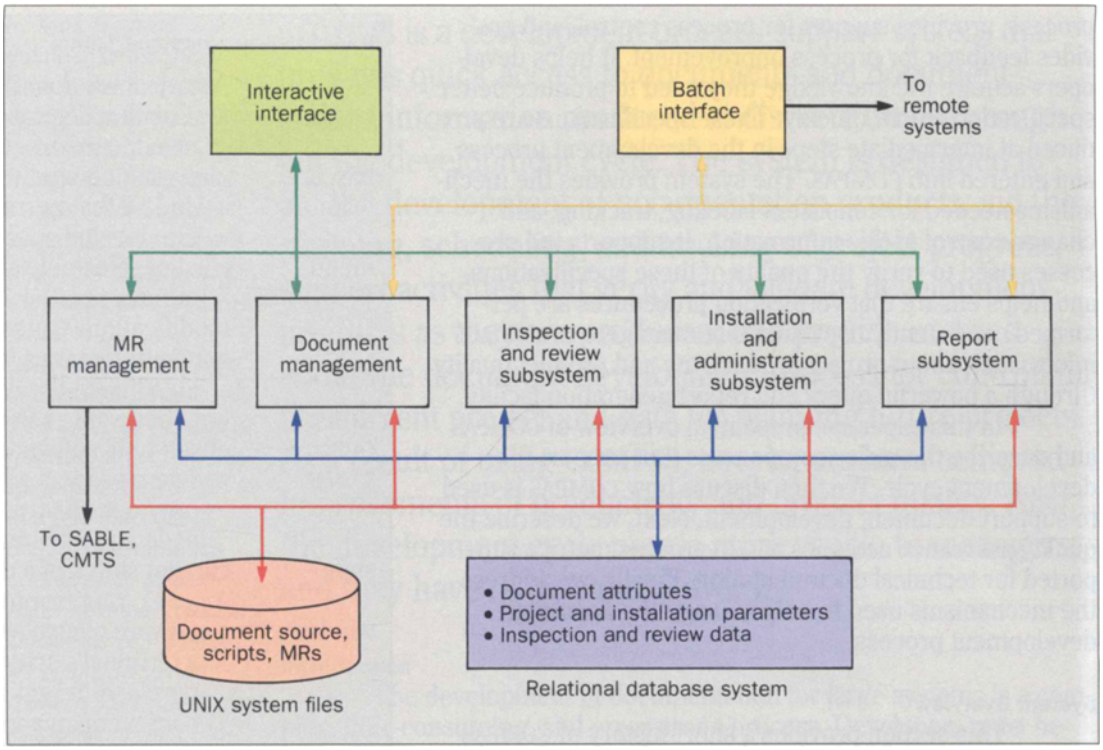


Figure 1. COMPAS architecture. The interactive interface provides a menu and forms oriented, full-screen interface, while the batch interface provides a command-line interface for access from satellite systems. MR management provides an interface with other systems for managing modification requests. Document management provides document storage, retrieval, and printing functions. The inspection and review subsystem provides collection, storage, and retrieval of inspection and review data. The installation and administration subsystem permits project and installation customization. The report subsystem provides report-generation and printing functions.

the COMPAS system is easy to use, and its users require little or no training.

Architecture. COMPAS runs on UNIX® System V and is written in the C programming language. As illustrated in Figure 1, the major components of the COMPAS system are document management, reviews and inspections, administration, report management, and change management. These components can be accessed either through an interactive interface based on menus and forms or through a command-line interface.

The information processed by the major system components resides in UNIX system files and a relational database management system. The *files* contain:

- The scripts for processing electronic mail
- The document-access control files, project-list files,

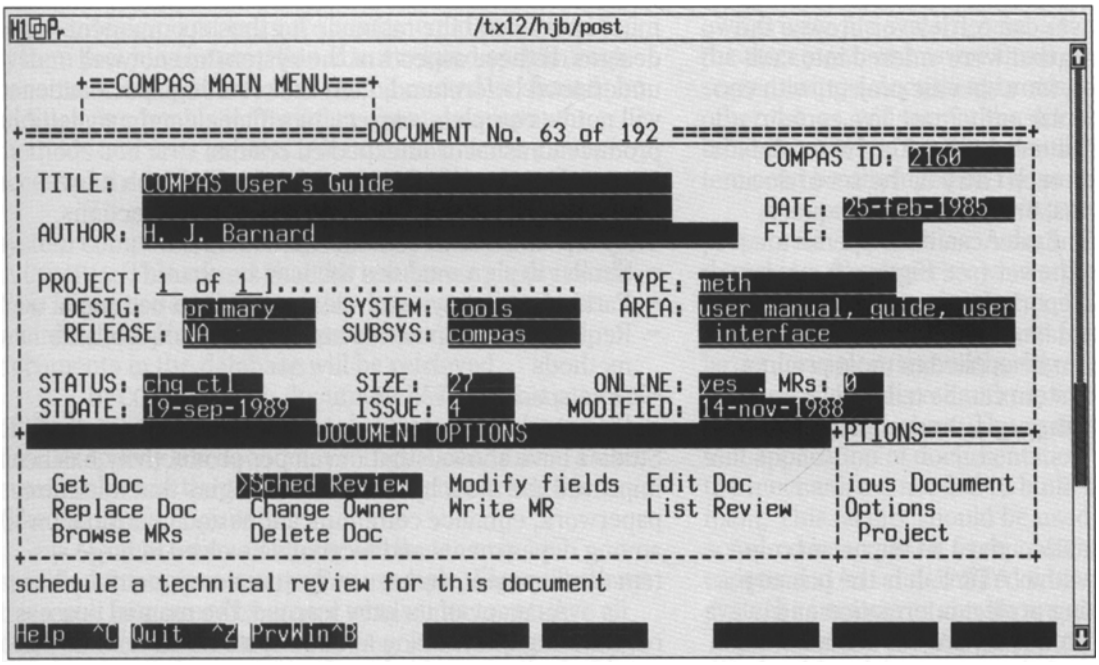


Figure 2. A query result. Highlighted areas in the upper window contain attributes for a document. Some of these attributes matched those specified in the query. The interactive, document options menu in the lower window provides choices for processing the document, changing its data, or executing a related process. As the highlighted action shows, the user wants to schedule a review of this document.

and administration files
- A library of processes invoked by COMPAS
- The document source files.

The *relational database* contains:
- Attributes for each document, review, inspection, and modification request (MR)
- User preferences (for queries and reports)
- Installation parameters.

As a document evolves through states or levels of maturity, COMPAS collects information in the form of text, attributes, and data. The *state* of a document is represented by a status attribute. When document text is entered into COMPAS, a user must specify some of the values for the document attributes, such as the ones on the form in Figure 2. COMPAS automatically sets the status to `draft` and starts to track the document. Additional attributes and data are collected for the document as it evolves through the review and inspection states, is updated, and is put under change control; and as modification requests are written. By the time a document reaches the `finished` or `chg_ctl` (change control) state, information has been collected both automatically and manually about its history.

COMPAS provides a general-purpose query facility that allows a user to retrieve information about documents and their reviews, inspections, or modification requests. The information can be retrieved by formulating queries that specify conditions about attributes.

For example, a user can retrieve or browse the text for a set of documents that were entered into COMPAS between certain dates, for a specific project, with certain keywords, by a particular author, are in a certain "state," and have a fixed pattern in the title. He or she can view the attributes for each entry in the set of documents, reviews, inspections, or modification requests retrieved by this query. The user can either perform operations on an entry in the set (see Figure 2), or generate status or statistical reports for the set of entries.

COMPAS provides data security and data integrity, is easy to maintain, and can be applied to most product life cycles. The COMPAS system can be tailored for projects and individual users through its administration and installation subsystem.

Document Development

Project documentation plays an important role in product development within AT&T. It is the primary medium for communicating project information and development decisions. Producing project documentation requires cooperative effort from many people and organizations. Throughout the development process, these organizations will produce many technical documents that describe what a system will do, how it will work, how it will be developed, how it will be tested, and how it should be used. The development staff spends well over half of its time just producing these documents.[1,2]

The Discovery Process. Producing a project specification is a difficult and time-consuming task. Developers must first acquire knowledge about many pertinent and related system components.[3] In this discovery process, they must:

- Determine the overall architecture of the system.
- Acquire knowledge about all components that will interact either directly or indirectly with their components.
- Acquire knowledge about the system environment.

Developers must know how their components interact with other components and with system resources, and must understand the rationale for those component designs. If these aspects of the system are not well understood beforehand, then the resulting specifications will not be complete, correct, or efficient, and may later produce unusual or unexpected results.

Developers must become familiar with more than just environmental and component interactions. They also need to obtain information about:

- Similar design modules that can be shared
- Parts of earlier system releases that can be reused
- Required development standards, techniques, and methods
- Area specialists
- Plans and schedules.

Studies have shown[2] that developer productivity has been improved the most by those technologies that minimize paperwork, enhance communications and coordination among departments and locations involved in large system development, and reuse system components.

As many of us have learned, the manual process of gathering information about a system can be time consuming, tedious, and imprecise. Commonly used techniques include obtaining needed document information from peers, and tracing other related information through the reference sections of those documents. A developer may also contact an expert and obtain information through informal conversations. However, the use of these and other manual techniques increases the risk that the needed set of information will not be current or complete.

When we consider the magnitude, difficulty, and importance of this discovery process, any improvement could result in shorter development times. Further, it would prevent many potential problems that result from misunderstood or omitted component and environmental interactions.

Information Retrieval. COMPAS provides a method of gathering needed development information through access to a computer data store of product and environmental knowledge. All the information that documents a product's plans, methodologies, requirements, archi-

tecture, design, and tests are contained in COMPAS. The system provides information about related system components or components that can be reused or shared. In addition, it contains information about development methods and area experts. All this information can be accessed through a powerful query facility.

To gather the information needed to document a system component, developers must formulate a query in COMPAS. This means that the user must supply *conditions* about the attributes of the documents he or she wants to find. If no conditions are specified, then all documents in the database will be retrieved.

In COMPAS, each document has an associated set of attributes that describe the document's characteristics. To specify the document retrieval conditions, a user enters values in the fields of a document attribute form (like the one in Figure 2).

Combinations of these attributes may be specified for a given query. For example, Figure 2 shows the result of a query by a user who wants to retrieve all documents that have a state of `chg_ctl`, a type of `meth`, and a pattern like `guide` in the title; were created after 1985 (i.e., `DATE >= 01/01/85`); and are related to the project `tools`. Another example would be a query from a tester who wants to study all test plans for similar features. Given this flexibility, the number of unique query possibilities is virtually limitless.

The set of documents retrieved by a query can be altered. If the set does not include enough documents, the user can query the database again and have the results of the second query appended to those of the first query. On the other hand, if the set of documents that a query retrieves is too large, the user can query that set as if it were the database and, thus, produce a smaller set of retrieved documents.

When a user is satisfied with the set of documents retrieved by a combination of queries, then he or she can obtain the document information. Each document's open MRs will be retrieved automatically with the text.

Users can either copy the document text to their own private file system, print a formatted paper copy of the document, or browse through the document on the screen using a document browser. The browser, among other things, will format the document and display a table of contents. Each section listed in the table of contents can be displayed on the screen.

Producing the Draft. COMPAS does not provide text preparation capabilities. A draft document must be produced in a developer's own word processing or document production environment. This environment may consist simply of an editor and formatter; or it may be a larger, complex, text authoring system. Some of these systems allow several authors to produce separate parts of the same document. The system can check for consistent application of documentation standards, and provides the mechanisms needed to build a completed draft document. This facility should be used after the developer has acquired information from COMPAS about related specifications and has determined what parts, if any, of other system components can be reused or shared.

A draft document is often produced in the developer's environment using an editor. If a "template" exists for the type of specification to be produced for a project, the developer can retrieve that template from COMPAS and place it into his or her private file system. He or she can then retrieve parts of other documents from COMPAS and *reuse* them, using "cut and paste" editor operations. Finally, the developer can complete the remaining sections of the specification template.

Document Entry. The draft document should be entered into COMPAS as early as possible before the technical review. This gives other developers an opportunity to access that draft.

To enter the draft into COMPAS, the developer simply fills in the fields of the document attribute form (similar to the one in Figure 2), and provides the system with a file that contains the document text. The system supports the ability to *share* or link documents among projects by providing information about the link on the document attribute form.

Developers can create entries for a document before they enter its text into COMPAS. Many projects use this capability to improve document tracking. As soon as the project decides what documents are to be produced, entries are created for these documents in COMPAS. This information is used to track completed drafts, because a draft is not considered complete until the text has been delivered to the system.

Quality Assurance

Quality assurance is defined as the system of methods and procedures used to ensure that the product meets its requirements.[4] For technical documents, quality assurance typically includes methods and procedures like reviews, inspections, and change control. Each technical specification for a component of the system should meet the specification's requirements and specify the proper interactions with other system components and with the system environment. In addition, a specification must follow the project standards and conventions. If not, other specifications, the code, test plans, and user documentation will be adversely affected.

The quality of technical documentation must be verified early and maintained throughout the development life cycle. After formal technical review, a document is usually stable enough for its author and other developers to use. This does not mean that the document will not undergo change. As developers produce more detailed system specifications, they may discover that certain parts of earlier specifications were not complete or correct. Changes to these earlier specifications must be made to ensure their quality. When a *control team* approves those changes and the documents are updated, other developers who are already working from those specifications must be notified of the changes.

COMPAS provides support for quality assurance activities that follow the typical development paradigm described by Bersoff.[5] That is, a draft document is produced, reviewed, and baselined (put under change control); subsequent changes are then controlled. Each of these document-state changes is reflected in COMPAS. Of course, there are exceptions to every rule, and COMPAS does not require formal change control for all types of documents.

Reviews. In addition to support for scheduling technical reviews, the system captures review-process quality data. It supports baselining of documents, requests for modification, and modification. Because changes to a controlled document can affect other controlled documents, COMPAS helps the document's developer trace the other documents through relational attributes. The system also supports inspections that are more rigorous than technical reviews.

After a document is produced and delivered to the system, a developer can schedule a technical review. Figure 3 shows a typical COMPAS screen for scheduling a review. Once the developer has completed the form, he or she elects to send the notice (i.e., chooses the last item on the options menu). Then, COMPAS automatically changes the document's state from `draft` to `review`. Notices are sent via electronic mail to the appropriate project staff. Forms for the review meeting will be automatically filled out (using information in the database), formatted, and printed. If the disposition of this review is "re-review," additional reviews are then scheduled for the document. COMPAS will automatically change the document's state to `follow` and keep track of the number of follow-up reviews that were held.

Productivity and quality can be improved through the COMPAS review process. It reduces the effort required to produce, copy, and distribute review notices and review material, which increases developer productivity. It improves the quality of the review process by giving reviews more visibility. A developer can view a calendar of all reviews scheduled for related system components, and arrange to attend a review if appropriate. In addition, development managers can browse all reviews scheduled for a specific project area. Drafts of the documents to be reviewed can be obtained easily from the system.

Because COMPAS helps the development commu-

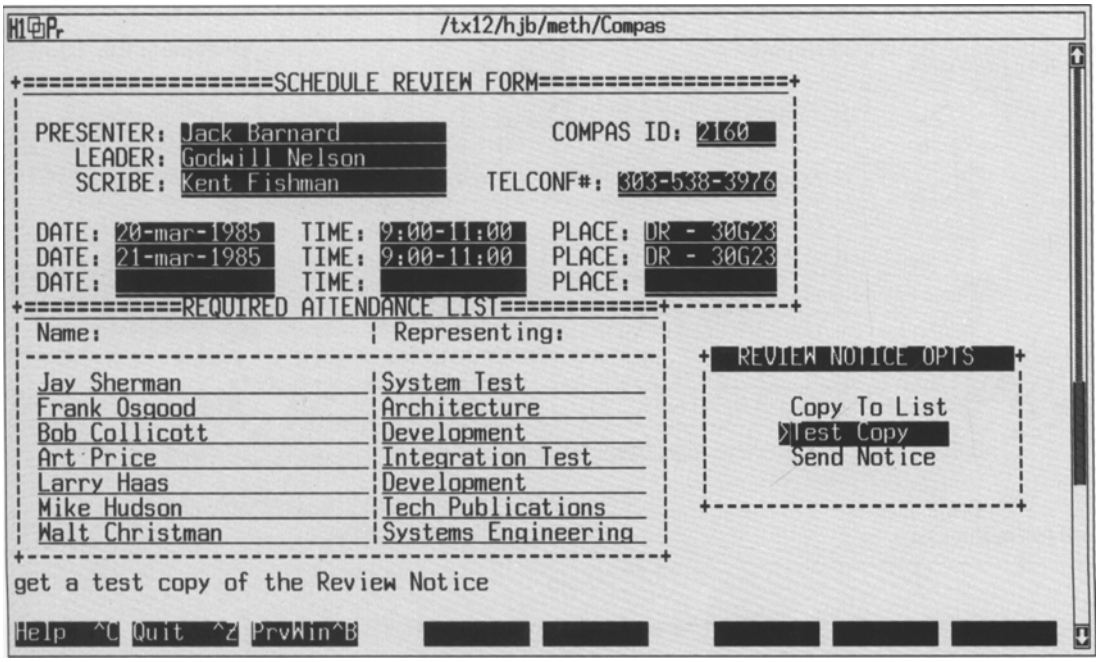


Figure 3. Scheduling a technical review. The window at the top contains data about review meetings for the document in Figure 1, while the one on the lower left displays the attendance list. On the right is the menu of available actions. If the user selects `Send Notice` from the menu, COMPAS sends meeting notices electronically to the attendees and automatically generates and prints the reports needed for the meeting. (`Copy to list` enables a user to add people, other than the required reviewers, to the mailing list for the notice. `Test copy` generates and prints a test copy of the review notice for the user to check before sending the real notice.)

nity comply with the review methodology, additional quality improvements are achieved. The review methodology that COMPAS supports is similar to that proposed in much of the literature.[6,7] However, the number of reviewers and their review roles can be controlled, adequate notification time can be ensured, and project notification lists can be automated. The system is flexible, and changes to review procedures and review requirements can be implemented easily.

Baselining. When the review process is complete and the document is stable enough for others to use, a project librarian will baseline the document in COMPAS by moving its state to `chg_ctl` and entering the review-process and fault data. COMPAS generates several forms that are normally completed at the end of a review and delivered to the librarian. If a project uses additional control before baselining a document, the librarian changes the document's state to `sign_off`, which remains in effect until the control team has indicated its approval.

Although some types of documents do not require a formal baselining procedure, such documents can also be entered into COMPAS as drafts. When the authors have completed the document, they can move its

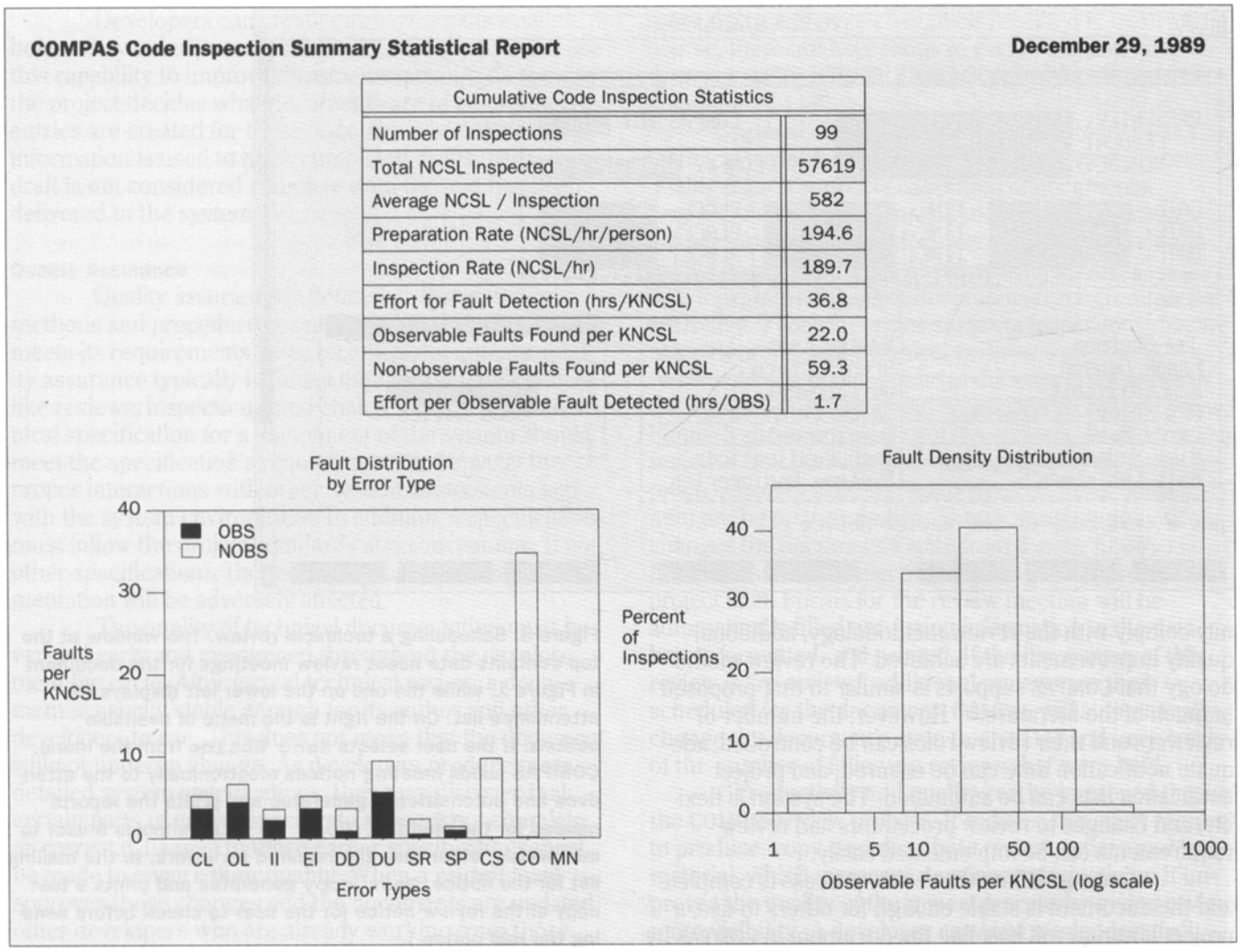
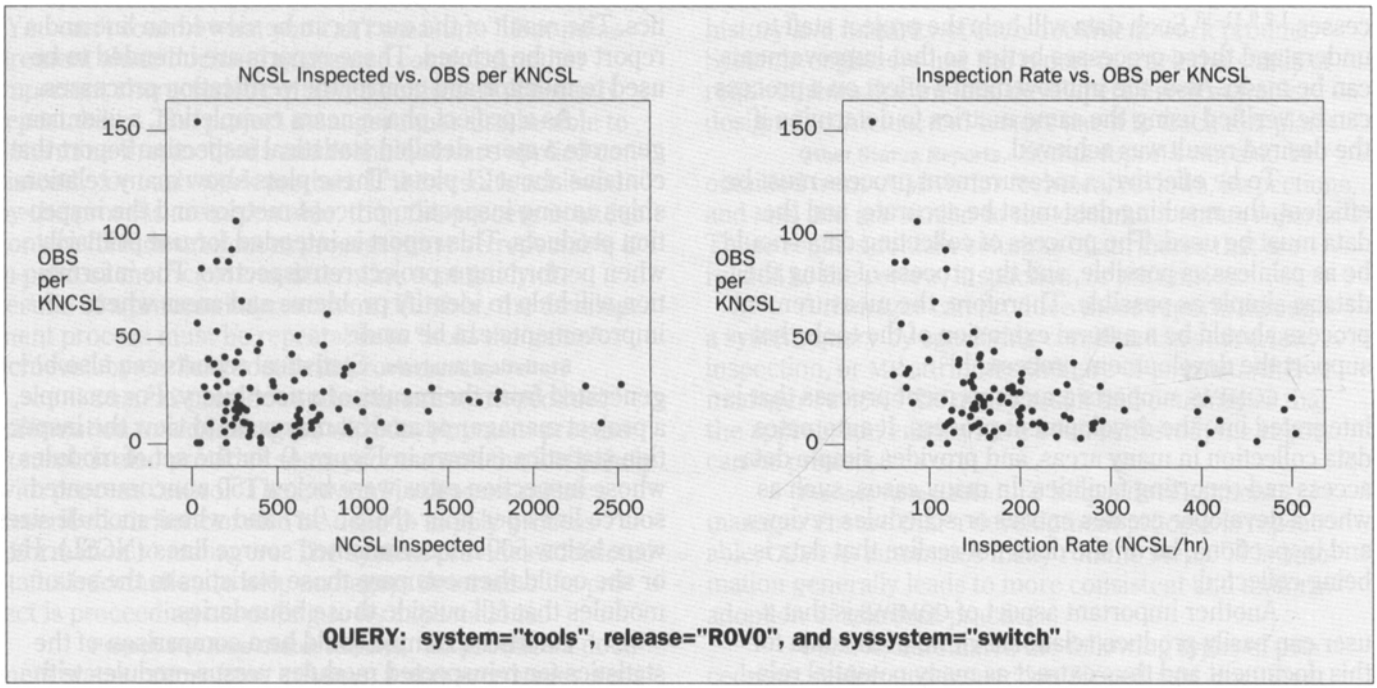


Figure 4. A typical code-inspection summary statistical report. (Above) Statistics about the noncommentary source lines (NCSL) in the document are tabulated at the top. Bar charts present fault distribution by error type (left) and fault-density distribution (right) per thousand NSCL (KNCSL). (Right) Scatter plots show the number of observations (left) and the observation rate (right). At the bottom are the query parameters issued to retrieve this report. (COMPAS produces this as a one-page report. It was split here to fit this publication's requirements.)



state to `finished`. A formal MR is not required to make a change to this type of document.

Modification Requests. COMPAS interfaces with change management systems, such as SABLE, to manage document MRs.[8] (SABLE is an AT&T software-product administration system.) To change a baselined document, a developer locates the document in COMPAS and then fills in the fields of an MR. The document author may resolve the MR, have the resolution approved, and modify the baselined document in COMPAS.

Other specifications that a document change could affect can be traced easily in COMPAS through relational document attributes. For example, suppose a user issues a query for documents with "Call Forwarding" in the title but does not specify the document type. The query would retrieve all requirements, architecture, and design documents that are related to the call-forwarding feature. Less obvious relationships to the feature can be found by using combinations of other document attributes.

Inspections. When code modules are produced from specifications, an entry for each module can be created in COMPAS and tracked to take advantage of the inspection support facility. Like technical reviews, inspections can be scheduled through COMPAS, and a completed set of inspection forms will be generated automatically and sent to the appropriate project staff.

The COMPAS system ensures that the inspection methodology is interpreted consistently. The inspection methodology it supports is similar to that described in much of the literature.[6,9,10]

In-Process Metrics

There is a lot of interest in collecting measurement data about development and verification pro-

cesses.[1,2,9,11–13] Such data will help the project staff to understand these processes better so that improvements can be made. Also, the improvement's effect on a process can be verified using the same metrics to determine if the desired result was achieved.

To be effective, a measurement process must be efficient, the resulting data must be accurate, and the data must be used. The process of collecting data should be as painless as possible, and the process of using the data as simple as possible. Therefore, the measurement process should be a natural extension of the tools that support the development process.[14,15]

COMPAS supports a measurement process that is integrated into the development process. It automates data collection in many areas, and provides simple data-access and reporting facilities. In many cases, such as when a developer creates entries or schedules reviews and inspections, he or she does not realize that data is being collected.

Another important aspect of COMPAS is that a user can easily produce relational views of the data for this document and thus extract as many potential relationships from the data as desired.

Documentation Process Statistics. A user can produce a documentation statistical report from COMPAS by specifying the area of interest and then printing the report. The report will contain a pie chart that indicates the distribution of each type of document for a project, and a histogram on the state of the documents for each month of the development cycle. Also included in this report are statistics on the characteristics of the set of documents.

Review and Inspection Statistics. The review and inspection statistical reports can be very detailed and include information about the quality of development products and of the verification processes.

A review and inspection summary report is obtained by selecting the project or project area of interest through a system query. As Figure 4 shows, this one-page report presents a variety of error and inspection statistics. The result of the query can be viewed on line and a report can be printed. These reports are intended to be used to monitor and control the verification processes.

As a project phase nears completion, a user can generate a more detailed statistical inspection report that contains about 21 plots. These plots show many relationships among inspection process metrics and the inspection products. This report is intended for use primarily when performing a project retrospective. The information will help to identify problems and areas where improvements can be made.

Statistical Analysis. Statistical reports can also be generated from the results of a user query. For example, a project manager or analyst may want to view the inspection statistics (shown in Figure 4) for the set of modules whose inspection rates were below 150 noncommented source lines per hour (NCSL/hr) and whose module sizes were below 500 noncommented source lines (NCSL). He or she could then compare those statistics to the set of modules that fall outside those boundaries.

Another example could be a comparison of the statistics for reinspected modules versus modules with only one inspection.

Process Control

A process is successful if it achieves its planned goals and objectives. Development plans generally include minimum requirements for cost, schedule, and quality. Therefore, a project is well controlled if it is on schedule, within budget, and achieves the quality objective. DeMarco states,[11] "Most project failures are the result of inflated or unreasonable expectations, and are not usually the fault of the project team."

The key factor in a well-controlled process is measurement.[11,15] The project manager must have enough data from each phase of the development process to ensure that a project will meet its expectations. He or she must monitor work products to ensure that they achieve each milestone when expected, and that the expected level of quality is achieved. DeMarco states,[11]

"You can't control what you can't measure." But measurement alone does not guarantee control. Another important characteristic of a well-controlled process is repeatability. The project manager must also be able to determine that methods and techniques are applied consistently for each work product. If the results achieved by the process are repeatable, then the process is under control. The development process must be repeatable if in-process metrics are to determine accurately the results of a process improvement. Further, the development process must be repeatable to "hold the gains" achieved by earlier process improvements.

COMPAS provides statistics and work-product information to aid managers with development process control. It also provides some process automation to help with process control. The COMPAS in-process metrics described earlier can also be used to supply process information to managers. The system provides additional status information to help managers determine if a project is proceeding according to its expectations.

Project and Area Status Reports. To produce a document status report from COMPAS, a manager issues a system query to retrieve document entries for a specific area of interest.

The system query is used to restrict the amount of information included in the report. Therefore, if a manager does not specify conditions for one of the document attributes, this is the same as if he or she specified "all" for that attribute. For example, if no conditions are specified for the system query, all document entries will be retrieved for the report.

The query result can be viewed online and, if desired, a report can be printed. The report's columns represent document attributes. For example, a typical document status report will contain columns for the document identification number, title, and authors; document date; type of document; the project area and size; the document status and status date; and so on.

Document status reports are generally used for tracking milestones, and list information about the history and maturity of each document work product. Systems engineers use the report to track the status of requirements documents, developers use it to track design documents, and testers use it to track test plans.

Other Status Reports. Status reports can also be obtained from COMPAS for technical reviews, inspections, and MRs and are similar to the document status reports. These reports contain columns of attributes that are related to the review, inspection, or MR entries.

A manager can produce these reports through a system query by specifying combinations of review, inspection, or MR attributes that are of interest. After a manager reviews the query result and determines that the appropriate entries have been retrieved, the report can be printed.

Process Automation. To help a development manager ensure that a development process is repeatable, COMPAS automates many routine steps. This automation generally leads to more consistent and uniform adoption of standard practices.

COMPAS automates the following types of procedures: review and inspection procedures, document state changes, project forms and meeting notices, distribution, data collection and statistical algorithms, and document change control.

Future Directions

The COMPAS development staff have plans for new releases of the system that will provide additional capabilities such as more in-process metrics; tailoring of fault types and forms; hypertext-like capabilities; and keyword searches on document contents. (*Hypertext* refers to a scheme that links text in a way that indicates how the text objects are related. It permits "nonlinear reading," something like skipping around in a book to gather information.)

Acknowledgments

We gratefully acknowledge comments from Art Price, Frank Osgood, Walt Christmas, and Jay Sherman

on this paper; and thank Dave Belanger for his valuable comments and support of this paper. We also thank the current members of the COMPAS development team: Kent Fishman, Godwill Akotaobi, Mike Hudson, and Zachary Lewis for their dedication and commitment to COMPAS.

References

1. B. W. Boehm, *Software Engineering Economics*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.
2. C. Jones, *Programming Productivity*, McGraw Hill Book Company, New York, 1986.
3. H. J. Barnard, R. F. Metz, and A. L. Price, "A Recommended Practice for Describing Software Designs: IEEE Standards Project 1016," *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2, February 1986, pp. 258-263.
4. G. G. Schulmeyer and J. I. McManus, *Handbook of Software Quality Assurance*, Van Nostrand Reinhold, New York, 1987.
5. E. H. Bersoff, "Elements of Software Configuration Management," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 1, January 1984, pp. 79-87.
6. *A Standard for Software Reviews and Audits (P1028)*, ANSI/IEEE Standard 1028-1988, IEEE Computer Society Press, New York, 1988.
7. D. P. Freedman and G. M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews*, Third edition, Little, Brown and Co., Boston, Massachusetts, 1982.
8. S. Cichinski and G. S. Fowler, "Product Administration through SABLE and NMAKE," *AT&T Technical Journal*, Vol. 67, No. 4, July/August 1988, pp. 59-70.
9. P. J. Fowler, "In-Process Inspections of Workproducts at AT&T," *AT&T Technical Journal*, Vol. 65, No. 2, March/April 1986, pp. 102-112.
10. M. E. Fagan, "Advances in Software Inspections," *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 7, July 1986, pp. 744-753.
11. T. DeMarco, *Controlling Software Projects*, Yourdon Press, New York, 1982.
12. *Software Metrics*, A. J. Perlis, F. G. Sayward, and M. Shaw (eds.), The MIT Press, Cambridge, Massachusetts, 1981.
13. R. Fairley, *Software Engineering Concepts*, McGraw-Hill Book Company, New York, 1985.
14. S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, The Benjamin/Cummings Publishing Co., Menlo Park, California, 1986.
15. W. S. Humphrey, *Managing the Software Process*, Addison-Wesley Publishing Company, New York, 1989.

(Manuscript received December 29, 1989)