WEB-BASED ISSUE TRACKING FOR LARGE SOFTWARE PROJECTS

JOHN R. CALLAHAN AND RESHMA R. KHATSURIYA
West Virginia University
NASA / WVU Software Research Lab
RANDY HEFNER
Intermetrics Inc.
NASA Software IV&V Facility

Many problems are found and fixed during the development of a software system. The Project Issue Tracking System toolkit, a Web-based issue-management tool, can be used to organize issue reports during development and to communicate with different project teams around the world.

Any software project, irrespective of its size, undergoes many transitions during its evolution. Most of these transitions are triggered directly by problems that emerge during the software life cycle. Such problems can take various forms, such as faults, errors, bugs, or requests for additional features and are typically documented in issue reports that can originate from developers or end users. A database of issue reports generated and maintained during development can be analyzed to reveal valuable information about why a particular decision was made or why an error occurred. Such a repository forms a dialogue between parties in which discrepancies can be raised, discussed, and resolved. Analysis of the dialogue leads to more informed decisions as well as improvements in both the product and the organization.

Most large-scale, mission-critical projects implement independent verification and validation functions to find, track, and continuously analyze all major issues raised during the life cycle of a project.[1] IV&V is performed by someone other than the system developer to evaluate the extent to which the delivered product satisfies the user's operational needs throughout the development life cycle. For most of these projects, success depends on effective communication between the developers and IV&V analysts, coordination between the groups, and systematic tracking of the issues. By increasing the availability and timeliness of information in a software project, we can enable interaction among geographically dispersed groups with differing viewpoints. Collaborative tools based on the World Wide

 1089-7801/98/$10.00 ©1998 IEEE

Web can mitigate risk on large projects by enabling more effective communication between designers and analysts.

The Project Issue Tracking System (PITS) is a tool that supports the IV&V effort for two major NASA projects: the Earth Observation System Data and Information System (EOSDIS) and the Earth Science Data and Information System (ESDIS). The EOSDIS IV&V effort teams several companies and organizations at several sites with Intermetrics, Inc., serving as the lead contractor. In this article, we examine the PITS Web-based mechanisms for tracking issue reports.

WEB-INTEGRATED SOFTWARE ENVIRONMENT

PITS was developed to provide an automated mechanism to document and track issues throughout their life cycles in a way that enhances overall IV&V effectiveness. The system is based on the Web-Integrated Software Environment, a tool developed at the Software Research Laboratory in Fairmont, West Virginia.[2] WISE is a Web-based change management system that helps groups of developers manage an issue report database. WISE supports indirect communication between users by providing different views based on user roles in the development process. Each role has a unique view called a to-do list that describes a subset of the project issues pertaining to it.

WISE can be configured to suit a specific development process. The WISE Programming Language (WPL) is a process language that lets users program forms-based views of project issue reports. This is achieved by specifying various roles pertaining to the project personnel and associating these roles with restricted accesses to the form fields. The software process is defined implicitly as the workflow of issue reports. This process is not fixed or globally defined by a single manager but can change based on the evolving roles of the development personnel or a change in the development process. The ability to evolve with changing roles and processes is essential for any collaborative effort.

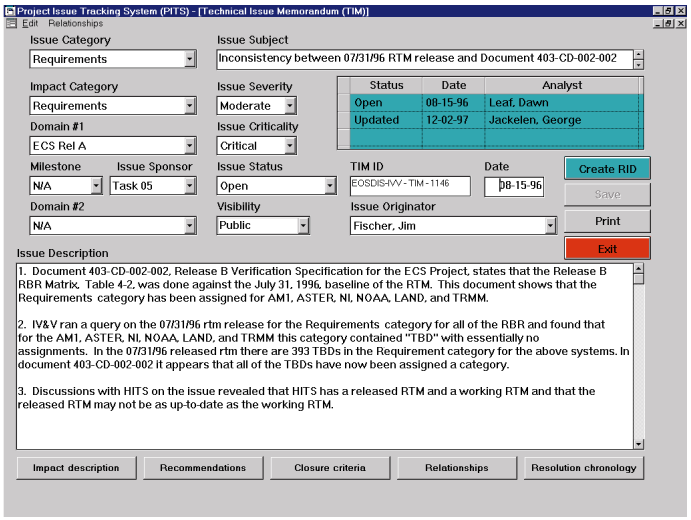


Figure 1. The PITS Technical Issue Memorandum. IV&V analysts enter technical issue memoranda and update issue information using a PC-based client-server interface tool.

PITS TOOLKIT OVERVIEW

The WISE prototype proved the feasibility of using Web-based systems to track issues, manage change requests, and process other electronic forms in projects with significant IV&V activities. The PITS toolkit advances many WISE concepts by supporting a rigorous, repeatable process to facilitate the identification and resolution of important issues during the development process. PITS minimizes the time devoted to metrics collection by automating the management of the issue database. This automation supports sophisticated analysis of process trends. It also allows team members to maintain a record of relationships between various issues.

The PITS toolkit consists of three basic tools:

- the PITS browser,
- the PITS client interface, and
- analysis tools.

The PITS browser is a CGI-based interface that lets software development teams view issues via a standard Web browser. IV&V analysts enter issue reports in the form of technical issue memoranda using a PC-based customized client-server interface tool (see Figure 1). This interface is also used to update issue information. Issue reports are stored in a central database specific to a particular project. Authorized clients can access the issue report database through the Web for query purposes and for viewing individual reports.

Architecture and Implementation

PITS began as a client-server application limited to the wide area network accessible from a NASA-approved client application site in the EOSDIS project. A PITS database can be hosted on any SQL-compliant backend. Currently, the system uses a Sybase relational database management server on a Sun Sparc server and a Microsoft SQL server config-

ured on an Intel x86 platform. The client runs in a local PC Windows environment that connects to the Sybase RDBMS server. The client-server application has been installed on PCs at Goddard Space Flight Center, the Intermetrics Greenbelt office, and the NASA/WVU software IV&V facility at Fairmont.

The current release of the PITS browser lets users select the issue repository pertaining to the two IV&V projects undertaken by NASA. Non-IV&V users have view and query access (that is, selective, read-only) to the issues databases via the Internet. This capability has been extended to be accessible through the WWW via a CGI-based interface restricted by the IP address of the browser site. Figure 2 illustrates different role-based interfaces of the PITS toolkit.

Issue Management

The vehicle for documenting and tracking issues within a PITS repository is the technical issue memorandum. A technical issue memorandum is a named, discrete collection of the project metadata (searchable issue characteristics), descriptive text, prescriptive text, and resolution progress information. This metadata can be customized for any project but typically include such items as issue description, impacts, closure criteria, resolution chronology, and relationships to other issues. Each technical issue memorandum describes a clearly defined set of problems at the some level of importance defined by its criticality and severity.

Technical issue memoranda track issue resolution processes to closure via PITS entries in the resolution chronology. The project characteristics captured in these memoranda can be analyzed to assess the impact of a problem on the development schedule and overall project quality, as well as the effort needed to resolve the problem.

The life cycle of an issue report is specific to the organization it is designed for. Typically, a problem is first identified and documented as an issue. Next, the issue is reviewed and if valid, published. Remedial actions are taken to resolve the issue and, when resolved, the issue is closed. PITS considers an issue closed when it is no longer a problem because it has been either satisfactorily resolved or overtaken by events. PITS is the primary mechanism for documenting the extent to which issues generated at a given moment are still important at a later moment.

Each issue carries a rating of its significance to the project. Issues at all levels of importance (severity and criticality) should be documented; however, marginal issues—those that do not affect success significantly—need not be given a high level of management attention. PITS filters out issues of marginal value so that project management can concentrate on resolving the more critical issues.

The values associated with issue characteristics are specified in issue database tables. Two entities are automatically captured by PITS:

- Issue status takes one of four values: “draft,” “open,” “closed,” or “closed with concerns.” Draft issues are not considered official until they are opened. Closed issues that do not meet the satisfaction of the originating organization are attributed the status of “closed with concerns” with the concerns described in the cor-

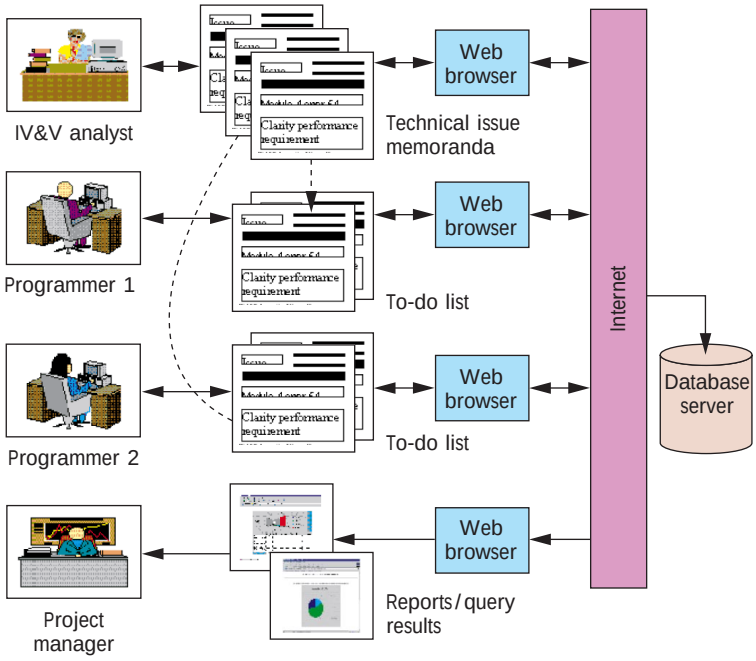


Figure 2. PITS schematic of user access to issue repositories. The PITS browser gives users access via the Internet according to their roles in the development project. Access restriction is based on the IP address of the browser site.

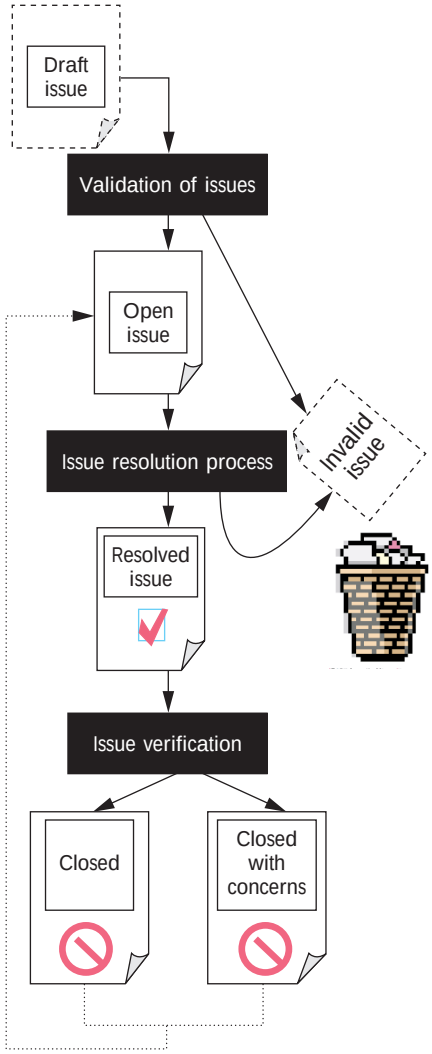


Figure 3. The life cycle of a typical issue.

PITS METADATA

In addition to "issue status" and "issue dates," the two entities automatically captured by PITS, the EOSDIS IV&V team associates values with the following issue categories:

- *Issue and impact categories* are keyed to the system development life cycle and project management. These categories include values such as "requirements," "integration and test," "engineering processes," "programmatics," and "basis of estimates."
- *Issue domains* are keyed to the intrinsic project development activities and phases. For example, in EOSDIS, the domain values are "ECS Release A," "EGS Version 2," and "Ebnet."
- *Issue milestones* are keyed to formal reviews and may be assigned values like PDR and CDR, indicating the preliminary and critical design review milestones in the software development life cycle.
- *Issue severity* classifies the error as, for example, "major," "moderate," or "minor." Classifying issues based on their severity enables management to concentrate on the more important issues. This is particularly useful in safety-critical, high-risk projects.
- *Issue criticality* categorizes a problem that impairs the operation of an important function. This field may take one of the following values: "critical," "essential," or "fulfillment." The combination of issue criticality and issue severity establishes the overall importance of a technical issue memorandum.
- *Issue visibility* can be either "public" or "private." A private technical issue memorandum is visible only to specific repository-affiliated users, while the public memorandum is visible to all PITS environment users if it does not have a draft status.
- *Issue originator* identifies the author of the technical issue memorandum.
- *Issue sponsor* describes activities associated with an issue and may take values like "Run test 144," "Contact Smith," or "Verify with issue 72."

responding technical issue memorandum. Figure 3 illustrates the rough life cycle of a typical EOSDIS issue report.

- Issue dates identify four important stages in the life cycle of an issue: "draft," "open," "closed/closed with concerns," and "updated." This entity provides the basis for studying the resolution chronology of a particular issue.

The additional characteristics used in the EOSDIS effort are listed in the sidebar "PITS Metadata."

A technical issue memorandum can be generated as a stand-alone issue but most issues exhibit close relationships with other EOSDIS reporting entities such as technical analysis memorandum, technical analysis reports, and review item discrepancies. A technical analysis memorandum, for example, can raise issues that necessitate formal tracking because they have more than marginal value. PITS is designed to support multiple relationships among reporting entities. These take the form of hypertext links across project sites. Linkage across multiple repositories within a single PITS environment is a planned enhancement.

PITS WEB BROWSER FUNCTIONALITY

The PITS browser can be accessed from any standard Web browser. At present, PITS uses the browser for dissemination of information only, and

includes query capabilities, reports, and metrics. Future releases will include the technical issue memorandum submission form, additional query capabilities, and more effective real-time reporting mechanisms.

PITS Database Queries

The initial PITS browser query capability permits filtering of the repository issues based on selections of various metadata. The categories are used to characterize each PITS repository issue. As shown in Figure 4, users can select values from metadata list boxes and text search boxes to generate an SQL query to the Sybase server. When building the SQL query, each selection for a metadata category is added in a logical AND and multiple value selections for the same metadata category are viewed as logical OR operations. A sample query result is shown in Figure 5. Included in the result window is a table that briefly describes the technical issue memoranda and provides hyperlinks to each memorandum.

Issue Reporting

The PITS browser page shown in Figure 6 graphically represents the total number of technical issue memoranda in PITS. The graphic consists of a pie chart with sections for "total open," "total closed," and "total closed with concerns." The browser also displays summary reports and aging metrics. The aging report includes a bar graph, which displays the memoranda by number of days open, and a table, which furnishes the number of memoranda divided by issue categories, such as severity, criticality, days open, and status, and highlights the set of critical activities that have to be watched with extra vigilance.

As an error propagates through each phase in the development life cycle, its effect multiplies on other subsystems. Cost overruns and schedule delays are generally more severe the later they are discovered. Since any delay in taking care of these issues will hamper the overall effectiveness of IV&V activity, the summary and aging reports call the most severe and critical technical issue memoranda to the attention of the project managers as early as possible in the development process.

Configuration

For the PITS tool to be effective, the development and IV&V teams must configure it to implement certain policies and guidelines. Some of these are common and applicable to any project, while others differ from project to project.

Roles-based issue management. In PITS, members of the development team can change issue reports in accordance with assigned roles. Understanding and configuring these roles and their associated activities is an important management responsibility.

Access and security. PITS lets authorized team members see all tasks, but update only their own. PITS supports five user access levels identifying the privileges associated with various roles, as depicted in Figure 7.

Access level 0 corresponds to the role of the PITS system manager. Individuals in these roles

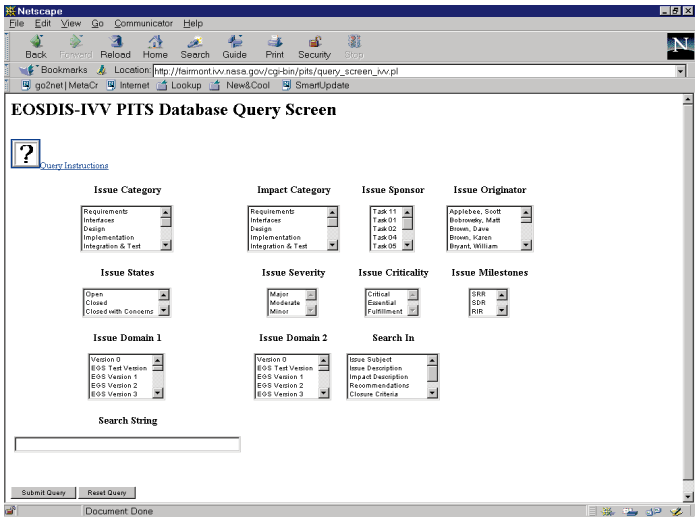


Figure 4. The PITS database query screen lets users select metadata values from the list boxes and enter search strings into the text box.

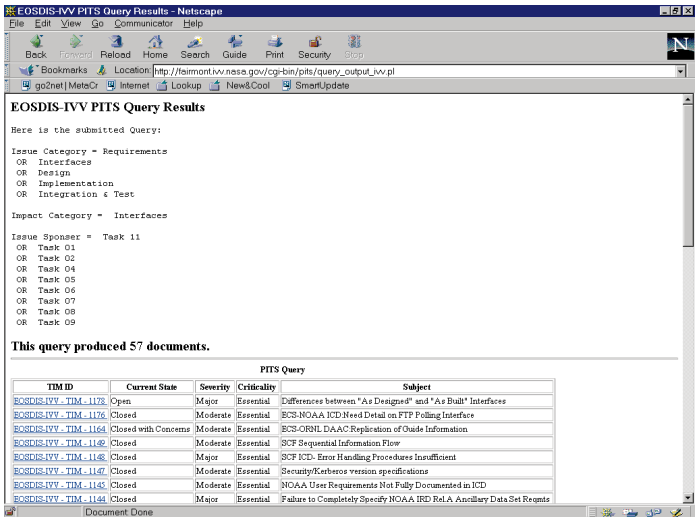
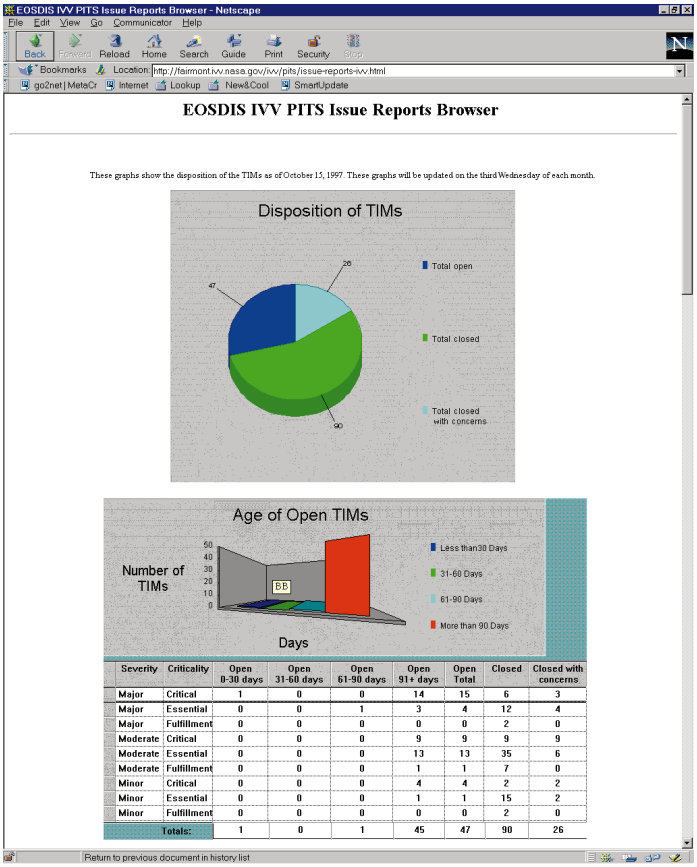


Figure 5. A PITS query results screen lists the technical issue memoranda returned in response to the query. Each memorandum is linked to a detailed description.



Severity	Criticality	Open 0-30 days	Open 31-60 days	Open 61-90 days	Open 91+ days	Open Total	Closed	Closed with concerns
Major	Critical	1	0	0	14	15	6	3
Major	Essential	0	0	1	3	4	12	4
Major	Fulfillment	0	0	0	0	0	2	0
Moderate	Critical	0	0	0	9	9	9	9
Moderate	Essential	0	0	0	13	13	35	6
Moderate	Fulfillment	0	0	0	1	1	7	0
Minor	Critical	0	0	0	4	4	2	2
Minor	Essential	0	0	0	1	1	15	2
Minor	Fulfillment	0	0	0	0	0	2	0
Totals:		1	0	1	45	47	90	26

Figure 6. The PITS issue reports browser page gives graphic representations and tables summarizing status and providing an aging analysis of the technical issue memoranda in the PITS repository.

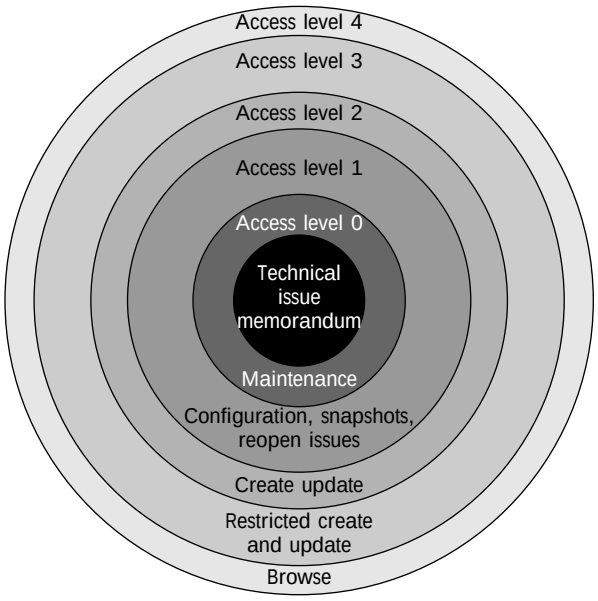


Figure 7. PITS supports five user access levels, each associated with specific privileges.

have maintenance privileges, allowing them to create new user accounts. Program managers are assigned access level 1 and are allowed to create and review database trend analysis snapshots, update PITS metadata values, and reopen closed issues. Task lead/tier-one managers (access level 2 roles) are allowed to create or update an issue regardless of its status. IV&V analysts are assigned access level 3 and can create issues and update those issues while the issues are in the draft state. They can only add resolution chronology entries to open issues. Access level 4 corresponds to the visitor role, which has read-only access for public issues not in a draft state. Technical issue memoranda classified with visibility "private" cannot be viewed by visitors.

Upper access levels automatically inherit privileges of lower access levels.

BENEFITS OF PITS IMPLEMENTATION

PITS provides a quantitative basis for the periodic IV&V findings meetings with NASA's senior project management. It serves the needs not only of mission-critical projects where IV&V is applied, but can be extended to almost any project in public or private sector organizations.

Trend Analysis

Determining the health of a project and keeping up with its schedule are difficult management tasks. PITS allows for continuous monitoring of issue categories that significantly affect the status of an ongoing software development project.

PITS is designed to fully support extensive near- and long-term trend analysis and provide summary reports on overall trends. This distinguishes PITS from other issue-tracking systems that primarily focus on project milestone issues. For example, the Review Item Discrepancy tracking system, which is also used on the EOSDIS project, authors and tracks only those issues identified during major reviews. Other issue-tracking toolkits, such as Rational's Distributed Defect Data Tracking System, allow development organizations to manage issue reports within development teams.

To demonstrate the value of continuous issue reporting and analysis, we examined a body of 116 closed issue reports in the EOSDIS repository (http://hopper.csee.wvu.edu/projects/data.htm). Each issue originated as a draft issue, transitioned to an open issue and was eventually resolved and classified as a closed issue. Several issues were opened and closed on the same day, but PITS

retained such issues in draft form until a formal review. Disagreements over issue classification are discussed during formal reviews and resolved by management. A dispute is resolved by opening the issue or invalidating the item. If a valid draft issue has already been resolved by the development team, it is immediately closed.

A continuous regression analysis on an issue repository can indicate problems early in the development process. For example, an ordinal logistic regression analysis on the selected 116 issue reports showed that criticality, severity, number of relations to other reports, length of the resolution chronology, and impact category type significantly influence the time needed to close an open issue. The category of the error, however, does not play a significant role in determining how long an issue is open. Although the majority of the 116 issues were short-lived, those with high criticality and a large number of resolution chronology entries tended to be amongst the longest-lived open issues. These issues usually correspond to chronic, unresolved problems that may not have an immediate resolution, but must remain open until an adequate solution is determined. Our analysis model is simplistic, but we are building more sophisticated models to help identify additional significant factors on the project. Factors such as severity and impact category may also play a role in determining the time to close an issue (for a full list of issue characteristics, see the sidebar "PITS Metadata.")

Online issue management systems such as PITS can help management with cost and schedule estimates using continuous analysis. For example, we can determine the minimum, average, and maximum number of days needed to close an open issue based on characteristics found to be significant on a specific project. To help schedule a more accurate date of delivery, we can determine the range of the number of days needed to close all remaining open issues. We can plot the range of days on a periodic basis throughout the project to determine if a project's delivery date is realistic. Many questions remain, however, concerning the privacy, security, and appropriateness of continuous analysis.

Process Improvement

The ability to detect faults early in the project's life cycle reduces the mean time and effort to remove them. As Boehm has shown, correcting an error later in the project stream tends to cost 50 to 200 times more than correcting it when it is found.[3] Regular monitoring of the documented issue reports allows team members to detect incorrect, ambiguous, or inconsistent requirements or design specifications that can lead to critical faults during the later stages of development. Tools like PITS and their underlying principles and practices have the potential to trigger process improvement by enabling developers to share and learn from each other's experiences.[4]

PITS enforces formal, early, and continual communication carried out in a well-structured manner—a substantial step toward achieving a better capability maturity model (CMM) level. Most common errors can be isolated from the issues repository and taken care of by analyzing defects for causes. PITS also enables complex projects to preserve the memory of past communications and helps to retain this memory in its knowledge base, allowing developers to look into why a certain decision was made. Moreover, the PITS software metrics reports improve top-level management visibility into the software development process, giving management tighter control over the entire process.

Correcting an error later in the project stream tends to cost 50 to 200 times more than correcting it when it is found.

Because issue-tracking tools like PITS are based on formal and recorded communication, they aid in decision making by involving every member in the project team. This collaboration not only increases the level of confidence in the resultant software, but also leads to a more maintainable software product. Moreover, PITS allows outside users with visitor access to access the defect knowledge base and get answers to problems or analyze available defect information in real time. This read-only capability enhances communication between the system vendors and their customers.

Increased Reliability and Accurate Schedule Estimation

Software for safety-critical systems has to be ultra-reliable. Keeping defects under strict control is a highest priority task and is essential for assessing reliability and predicting development schedule and cost characteristics.

RELATED WORK

Problem-tracking tools support disciplined software development with built-in configuration management, version control, and automatic change notification. Some of these, such as TrackWeb,[1] act as problem-solving systems that help maintenance personnel solve problems by letting them know whether a similar problem has occurred before and how it was solved.

Process-modeling languages provided by systems like Oz, developed at the Programming Systems Lab at Columbia University, use a rule-based approach to define workflow processes. New processes or existing processes can be tailored for a project by specifying a set of rules that govern the flow of transactions. Environments like OzWeb[2,3] provide hypermedia collaboration environments that intertwine support for planning and execution of purposeful work by assisting information workers in generating and exploiting all online materials relevant to a particular application.

ConversationBuilder,[4] Oz, and OzWeb focus on integrated team communication and collaboration but lack change control and tracking mechanisms needed by an IV&V team. Other approaches that help coordinate development activities via the Web include WebMake,[5] Software Dock,[6] NUCM,[7] and CoMo-Kit/MILOS.[8,9] These approaches, however, did not support our design goals in that they do not enable explicit channels of communication between IV&V analysts and development teams concerning the issues that evolve during a project's life cycle.

REFERENCES

1. R. Roybal, "Defect Tracking for the Enterprise: A Case Study," *American Programmer*, 1996, pp. 62–81.
2. G. Kaiser et al., "A Metalinguistic Approach to Process Enactment Extensibility," *Fourth Int'l Conf. on the Software Process*, Brighton, UK, Dec. 1996.
3. G. Kaiser et al., "WWW-based Collaboration Environments with Distributed Tool Services" *World Wide Web J.*, Vol. 1, 1998, pp. 3–25.
4. S.M. Kaplan et al., "Supporting Collaborative Software Development with ConversationBuilder," *Software Eng. Notes*, Vol. 17, No. 5, Dec. 1992. *Proc. Fifth ACM SIGSOFT Symp. on Software Development Environments.*
5. M. Baentsch, G. Molter, and P. Sturm, "WebMake: Integrating Distributed Software Development in a Structure-Enhanced Web," *Proc. Third Int'l World-Wide-Web Conf.*, Darmstadt, Germany, 10–14 Apr. 1995. Available at http://www.igd.fhg.de/www/www95/papers/51/WebMake/WebMake.html.
6. R.S. Hall et al., "The Software Dock: A Distributed, Agent-based Software Deployment System," Tech. Report CU-CS-832-97, Dept. of Computer Science, Univ. of Colorado, 1997.
7. A. van der Hoek, D. Heimbigner, and A.L. Wolf, "A Generic Peer-to-Peer Repository for Distributed Configuration Management," *Proc. 18th Int'l Conf. Software Eng.*, Berlin, Germany, Mar. 1996.
8. B. Dellen, F. Maurer, and G. Pews, "Modeling and Coordinating Design Processes," *Data and Knowledge Eng. J.*, Vol. 23, No. 3, Sept. 1997, pp. 269–295.
9. F. Maurer and B. Dellen, "Internet Based Software Process Support," *Proc. WET-ICE 98*, IEEE CS Press, Los Alamitos, Calif., forthcoming, 1998.

Counting the defects at various levels in the process gives a quantitative handle on how much work the project team has to do before it can release the software. By comparing the number of new defects uncovered to the number of defects resolved each week, developers and analysts can determine how close the project is to completion. A ratio indicating that defects are being corrected faster than they are being found shows that the project is making progress toward completion.[5]

The data on time required to fix defects categorized by type of defect (for example, three hours per design phase defect, two days per requirements specification defect) provides a basis for estimating remaining defect-correction work on current and future projects. The data on phases in which defects are detected and corrected gives a good measure of the efficiency of the development process—for example, if 95 percent of the defects are detected in the same phase they were created, the process is very efficient. On the other hand, if more than 5 percent of the defects are detected one or more phases after the phase in which they emerged, the verification and validation process has a lot of room for improvement.

Defect density, measured in terms of the number of defects per line of source code, speaks of the quality of the development effort. Several other techniques, such as defect pooling and defect seeding, reveal the degree of quality assurance on a project. Defect pooling is a simple defect prediction technique in which defect reports are arbitrarily separated into two pools and the total number of unique defects are estimated based on the overlap of defects reported in the separate pools. Defect seeding is a statistical technique in which the number of defects can be estimated based on the ratio of seeded defects found to indigenous defects found.

FUTURE WORK

Future work will expand PITS to improve data collection and analysis by reducing the integration effort with other project artifacts. For example, issue reports pertaining to design may involve certain diagrammatic representations that need to be communicated using graphical capabilities. In the client-server model of PITS, this issue can be handled by adding links in the technical issue memorandum to the graphical object. In the Web-based PITS browser, this issue can be resolved to some extent by allowing an HTML page to be part of the memorandum. Additional planned features include a relationships interface, password-protected secu-

rity mechanisms based on roles, and an automatic mail-notification mechanism.

We also intend to look further at the similarities between issue management and research in the area of inconsistency management.[6] In many cases, issues identified by the IV&V team cannot be resolved immediately, but are acknowledged by the development team and addressed at later phases in the development life cycle. Project team members who are aware of potential risks associated with issues can keep track of certain types of inconsistencies while progressing in other areas of a system's development. ■

ACKNOWLEDGEMENTS

The authors would like to thank Jim Harner and Magdalena Niewiadomska-Bugaj of the West Virginia University Department of Statistics for their helpful comments and interest concerning the regression analysis on the issue repository. Mike Brown also helped provide preliminary analysis and ideas about future trend analysis models.

REFERENCES

1. J.D. Arthur et al., "Reducing the Mean Time to Remove Faults Through Early Fault Detection: An Experiment in Independent Verification and Validation," Tech. Report TR-96-02, Dept. of Computer Science, Virginia Tech, Va., 1996.
2. J. Callahan and S. Ramkrishnan, "Software Project Management and Measurement on the World-Wide-Web (WWW)," *Proc. Fourth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1995, pp. 76–87. Also available online at http://research.ivv.nasa.gov/docs/techreports/1996/NASA-IVV-96-006.ps.
3. B.W. Boehm, *Software Engineering Economics*, Prentice Hall, Englewood Cliff, N.J., 1981.
4. B. Grady, *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, Englewood Cliffs, N.J., 1992.
5. S. McConnell, *Software Project Survival Guide*, Microsoft Press, Redmond, Wash., 1997.
6. A. Finkelstein et al., "Inconsistency Handling in Multi-perspective Specifications," *Lecture Notes in Computer Science 717*, I. Sommerville and M. Paul, eds., Springer-Verlag, New York, 1993, pp.84–99.

John R. Callahan is associate professor of computer science in the Department of Computer Science and Electrical Engineering at West Virginia University, Morgantown, West Virginia. He also serves as a principal investigator of the NASA/WVU Software Research Laboratory at the NASA Software IV&V Facility in Fairmont, West Virginia.. Callahan received a PhD in computer science from the University of Maryland, College Park, in 1993.

Reshma Khatsuriya is a graduate student at West Virginia University, Morgantown, West Virginia. She received a BS in computer science and an MS in computer management from the University of Pune, India. Her research interests include software engineering, independent verification and validation, and database and Web development.

Randy Hefner is the independent verification and validation tools and technologies lead for Intermetrics, Inc., EOSDIS IV&V Tools Development. His research focus has been the development of information system tools to support, automate, and improve IV&V processes. He received a BS with honors in computer science from West Virginia University in 1986.

Readers can contact Callahan at callahan@csee.wvu.edu.