Quality Assurance Software Inspections at NASA Ames
Metrics for Feedback and Modification

Greg Wenneson, Informatics General Corporation

Software Inspections are a set of formal technical review procedures held at selected key points during software development for the purpose of finding defects in software documents. Inspections are a Quality Assurance tool and a Management tool. Their primary purposes are to improve overall software system quality while reducing lifecycle costs and to improve management control over the software development cycle. The Inspections process can be customized to specific project and development type requirements and are specialized for each stage of the development cycle.
For each type of Inspection, materials to be inspected are prepared to predefined levels. The Inspection team follows defined roles and procedures and uses a specialized checklist of common problems in reviewing the materials. The materials and results from the Inspection have to meet explicit completion criteria before the Inspection is finished and the next stage of development proceeds. Statistics, primarily time and error data, from each Inspection are captured and maintained in a historical database. These statistics provide feedback and feedforward to the developer and manager and longer term feedback for modification and control of the development process for most effective application of design and quality assurance efforts.

HISTORY

Software Inspections were developed in the early mid-1970s at IBM by Dr. Mike Fagan, who was subsequently named software innovator of the year. Fagan also credits IBM members O.R.Kohli, R.A.Radice and R.R.Larson for their contributions to the development of Inspections. In the IBM Systems Journal [1], Fagan described Inspections and reported that in controlled experiments at IBM with equivalent systems software development efforts, significant gains in software quality and a 23% gain in development productivity were made by using Inspections based reviews at the end of design and end of coding (clean compile) rather than structured walkthroughs at the same points. Fagan reported that the Inspections caught 82% of development cycle errors before unit test, and that the inspected software had 38% fewer errors from unit test through seven months of system testing compared to the walkthrough sample with equivalent testing. Fagan also cites an applications software example where a 25% productivity gain was made through the introduction of design and code inspections. As further guidelines for using Inspections, IBM published an Installation Management Manual [2] with detailed instructions and guidelines for implementing Inspections.

Inspections were introduced to NASA/Ames Research Center in 1979 by Informatics General Corporation on the Standardized Wind Tunnel System (SWTS) and other pilot projects. The methods described by IBM were adapted to meet the less repetitious character of Ames applications and research/development software as compared to that of IBM's systems software development. Though not able to duplicate IBM's controlled environments and experiments, our experience at Ames of gains in quality and productivity through using Inspections have been similar. From a developed Wind Tunnel software application which had been reviewed in structured walkthroughs and then later was rewritten and reviewed using

Inspections, the Inspected version had 35-65% less debug and test time and about 40% fewer post-release problems. Inspections implemented prior to unit test have been shown to detect over 90% of software's lifetime problems. Inspection results have been sufficiently productive in terms of increased software quality, decreased development times, and management visibility into development progress, that Inspections have been integrated into Informatics' development methodology as the primary Quality Assurance defect removal method.

When Inspections were first implemented at Ames, only design and code Inspections were introduced. The scope and usage has expanded so that currently, Inspections are used to review both system level and component level Goals (requirements) Specifications, Preliminary Design, Detailed Design, Code, Test Plans, Test Cases, and modifications to existing software. Inspections are used on most Informatics staffed development tasks where the staff level and environment are appropriate. Inspections implementation and usage at Ames are described in NASA Contractor Report 166521 [3]. Within Informatics contracts outside of the Ames projects, Inspections are also used to review Phase Zero (initial survey and inventory of project status), Project Goals, and Requirements Specifications generated through structured analysis.

PARTICIPANTS

The Inspectors operate as a team and fill five different types of roles. The Author(s) is the primary designer, developer, or programmer who prepares the materials to be inspected. The author is a passive Inspector, answering questions or providing clarification as necessary. The Moderator directs the flow of the meetings, limiting discussion to finding errors and focusing the sessions to the subject. The moderator also records the problems uncovered during the meetings. A Reader paraphrases the materials, to provide a translation of the materials different from the authors' viewpoint. One or more additional Inspectors complete the active components of the team. A limited number of Observers, who are silent non-participants, may also attend for educational or familiarizing purposes. Of the team members, the moderator and a reader are the absolute minimum necessary to hold an Inspection.

Team composition and size are important. Composition using knowledgeable designers and implementors having similar background or from interfacing software enable cross training of group members; understanding is enhanced and startup time is lessened. However, team members must be sufficiently different so that alternate viewpoints are present. Fagan recommends a four member team composed of a moderator and the software's designer, implementor, and tester. Our experience is that the most effective team size seems to be three to five members, exclusive of author and observers; more than this is a committee, less may not have critical mass for the process. We also try to keep the team together for all of the software's Inspections.

TOOLS

Written tools are used by the participants during the Inspections process to assist in the preparation, the actual sessions, and the completion of the Inspection. Standards are necessary as guidelines for preparing both design and coding products. The Entrance Criteria for inspection materials define what materials are to be inspected at each type of Inspection, the level of detail of preparation, and other prerequisites for an Inspection to occur. Checklists of categories (Data Area Usage, External Linkages, etc.) of various types of problems to look for are used during the sessions to help locate errors and focus attention on areas of project

concern. The Checklists are also used by the author during his preparation of materials and by the inspectors while they are studying the materials. Exit Criteria define what must be done before the Inspection is declared complete and the materials can proceed to the next stage of development. Each of these tools will have been customized for each projects type of development work, language, review requirements, and emphasis that will be placed on each stage of the development process.

PROCEDURES

An Inspection is a multi-step sequential process. Prior to the Inspection, the Author prepares the materials to the level specified in the Entrance Criteria (and to guidelines detailed in the project development or coding standards). The moderator examines the materials and, if they are adequately prepared, selects team members and schedules the Inspection. (IBM lists these preparations as the Planning step.) The Inspection begins with a short educational Overview session of the materials presented by the author to the team. Between the overview and the first Inspection session, Preparation of each Inspector by studying the materials occurs outside of the meetings. In the actual Inspection sessions, the Reader paraphrases while the Inspectors review the materials for defects; the Moderator directs the flow of the meetings, ensures the team sticks only to problem finding, and records problems on a Problem Report form along with the problem location. Checklists of frequent types of problems for the type of software and type of Inspection are used during the preparation and Inspections sessions as a reminder to look for significant or critical problem areas. After the Inspection sessions, the moderator labels errors as major or minor, tabulates the Inspection time and error statistics, groups major errors by type, estimates the rework time, prepares the summaries, and gives the error list to the author. The author Reworks the materials to correct problems on the problem list. Follow-up by the moderator (or re-inspection, if necessary) of the problems ensures that all problems have been resolved.

In certain cases, a desk Inspection or "desk check" may be a more effective use of time than a full Inspection. Desk Inspections differ from normal Inspections in that during the preparation period each inspector individually records errors found and a single Inspection session is held to resolve ambiguities in the problems. The moderator compiles all collected error reports to produce a single report. All other Inspection steps proceed normally. Desk Inspections can be appropriate for code or design that the team is familiar with and that has already been through previous Inspections. Desk Inspections do not have the group synergy generated during "normal" Inspections. The SWTS Inspections database for FORTRAN code Inspections indicates that the desk check has an 80% error detection rate but only takes 40% of the time required of a full Inspection.

STATISTICS

The statistics captured from the Inspection and tabulated by the moderator consist of time and error values. The time statistics are average per person preparation time (excluding the author) and Inspections sessions meeting time, both normalized to a thousand lines of code (KLOC). The error statistics are the numbers of major and minor errors detected, also normalized to a KLOC. As part of the tabulating and summarizing process, error distributions of major errors by Checklist headings are recorded and summarized for the Inspection as a whole. The tabulated statistics are entered into a database as weighted averages by size in lines of design or code and keyed by expected implementation language and type of Inspection. The SWTS Inspections database currently contains almost 250 entries of data for FORTRAN and Assembler languages for the Goals (Functional Requirements), Preliminary

Design, Detailed Design, and Code (desk and non-desk check) types of Inspections held on developed Wind Tunnel System software from 1980 through 1985. Over half of the entries are for code Inspections. Figure 1 contains summary figures from the database. The database summaries provide guidelines from which general conclusions and assumptions can be drawn. The database was generated as a development and management tool from several related SWTS project's Inspections and not from tightly controlled experiments. As such, when comparing individual Inspections figures to the database figures, variances from one-half to twice the average amounts summarized from the database are not considered extraordinary.

STATISTICS USE

The Inspections statistics in their raw and weighted forms can be used by the author, the design team and manager, the project manager, and Software Engineering as feedback, feedforward, and control mechanisms for individual, team, project and Inspections process behavior modification for future work to achieve better results. In addition, the statistics can be used in the current project and for future work and projects for tracking, estimating, planning, and scheduling of development and QA work.

The author uses the statistics to determine immediately what is deficient in inspected design or code and, over the longer term, patterns and general problem areas on which to focus attention for future work. The problem list, besides providing a working list of detected problems, includes locations of what needs to be fixed before the next development stage can proceed. Additionally, a distribution of major errors by checklist category across each module provides warning signals of error prone modules and high or higher density error rates by error type. A history of high error rates of certain error types also provides a pointer to design areas which need more work or training to develop or better understand.

The programming team and manager use error distribution by type and module from individual Inspections and Inspections of related software to locate common problem areas and thus focus future work and communication to diminish these. Error rates higher than normal for the group as a whole or error distributions in particular areas may indicate a group misunderstanding or a misstatement of the requirements. Higher error densities in modules interfacing to existing (or new) software, for example, can alert and direct effort to understanding the interface or provide warning to another group to clarify or improve that interface. For the designer and the team manager, lines of design (or lines of code, depending on development stage) and complexity per module give immediate feedback for design considerations of module size, cohesion, and coupling; this additionally provides an opportunity to ensure that modules are not proliferating from one design stage to the next. The completion of any individual Inspection along with module quantity and sizing gives quantitative and qualitative feedback for validity of component estimating, scheduling, and tracking information.

The Project Manager utilizes the statistics to help locate trends in various problem categories and help the team improve performance through group meetings or education. The statistics provide a quantitative evaluation of software correctness and allow prediction, based on Inspections held, of error prone sections of design or code, in order to concentrate development, QA, and testing resources on the most important areas. Additionally, each Inspection's results can be "validated" to ensure proper procedures were followed and the results are legitimate as compared to the project database. As an example, for a FORTRAN detailed design inspection, time

SUMMARY OF INFORMATICS SWTS PROJECT INSPECTIONS STATISTICS

Type of Inspect'n	Lang.	Total Number Held	Total No "Lines" Inspected	DENSITY-OF-PROBS. Per 1000 Lines Major	Minor	Total	TIME-PER-PERSON Per 1000 Lines Meet'g	Prep'n	Total
CODE - NON-DESK	ALL Lang	94	51186	22.0	59.9	81.9	4.6	4.0	8.7
Only	FORTRAN	90	49389	22.4	60.4	82.8	4.6	4.1	8.7
	ASSEMBLY	4	1797	10.1	44.5	54.6	5.0	2.6	7.7
CODE - DESK	ALL Lang	47	23206	21.0	51.3	72.3	3.9	-	3.9
	FORTRAN	43	21308	19.1	48.1	67.2	3.7	-	3.7
	ASSEMBLY	4	1898	42.6	87.6	130.3	6.3	-	6.3
DETAILED DESIGN	ALL Lang	44	10349	76.74	144.6	221.3	14.5	9.8	24.3
	FORTRAN	40	9205	83.1	143.4	226.5	14.5	9.2	23.7
	ASSEMBLY	4	1144	25.3	153.9	179.2	14.3	14.4	28.7
PRELIMINARY DESIGN	ALL Lang	43	13268	68.1	107.5	175.7	10.8	5.4	16.1
	FORTRAN	41	12570	54.3	89.8	144.1	9.1	5.5	14.6
	ASSEMBLY	2	698	316.6	426.8	743.4	39.8	3.7	43.6

This chart summarizes the statistics from Informatics inspections on the NASA Ames SWTS project. The statistics are weighted averages, each inspection being weighted by its size, in lines of design or code.

Figure 1
SWTS Inspections Database Summaries

guidelines are 23 hrs/KLOD (Thousand Lines of Design) per person for preparation plus meeting time and the team can expect to find 83 major and 143 minor problems per KLOD. Meeting times and error rates significantly different should be examined to determine their cause. A trend toward increasing error rates may mean that not enough attention is being directed to proper design. A decreasing error rate may mean design is becoming more effective or, when accompanied by decreasing preparation and meeting times, may mean Inspections are becoming less effective.

The statistics are also used to modify the Inspection process itself or its application. At the beginning of the project, the entrance and exit criteria, the checklists, and the methodology and standards are specialized to the project's particular development environment, languages, and review requirements. As statistics are compiled, evaluations of the data may lead to modifications to the entrance criteria to change the level of materials preparation, to the checklists to alter the attention given to certain design or code areas, and to the project standards to remove ambiguity or set new standards as necessary. Removing software components from an Inspection requirement or adding or deleting an Inspection as a quality gate at a particular design stage to more optimally use available time are options made more apparent by the statistics.

DATABASE ANALYSIS

Examination and analysis of the SWTS Inspection database indicate correlations between preparation time, meeting time, inspection rate, and errors detected. These correlations and others allow the overall Inspections procedures to be modified and guidelines established for the optimal conduct of Inspections within a project.

For FORTRAN code Inspections, errors detected are related to inspection rate (LOC inspected per hour), figure 2. Most sessions inspected code at the rate of 100 to 300 LOC per hour and detected between 10 and 80 major errors/KLOC. When the Inspection rate is too rapid, the error detection rate falls gradually. When the Inspection rate is excessively slow, there is a wide range of error densities. For excessively slow Inspection rates, we believe this wide range of error densities results from Inspecting two types of materials: "Difficult Materials" where the materials are complex and require a slower Inspection rate to evaluate but result in a normal to above normal error density; and "Poorly Prepared Materials" which were not ready for Inspection, but were still inspected and thus generated a large number of errors, were difficult to understand, and slow to inspect. The inspection of "Poorly Prepared Materials" represent abnormal situations which the moderator is supposed to prevent prior to scheduling or holding an Inspection. To this end, there are also cut-off limits before and within the Inspection, if the Inspected materials are too hard to understand and/or are producing too many errors, that is, they are probably not ready to be Inspected, the Inspection is stopped and the materials are returned to the author to be properly prepared.

There is a linear correlation between inspection rate and preparation rate (LOC/hr), figure 3. Materials requiring a slower preparation rate also experience a slower Inspection rate, and vice versa. We believe the correlating factor is complexity of materials, more "difficult" code takes more inspector preparation time and more inspection time (lower inspection rate).

Errors Detected vs. Inspection Rate

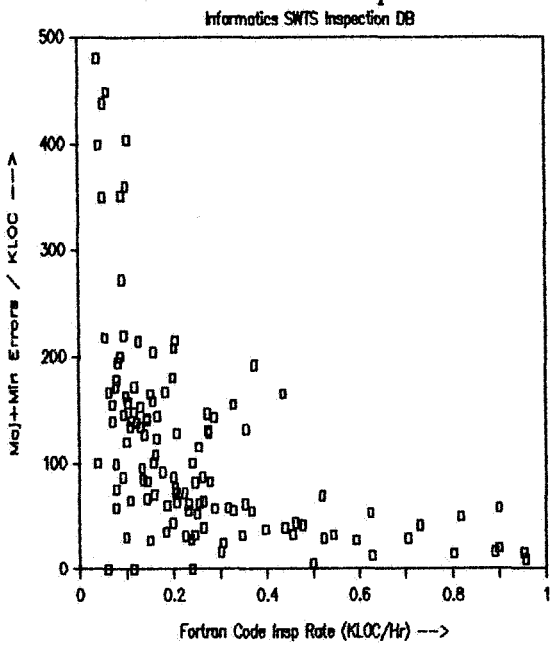


Figure 2

Inspection Rate vs. Preparation Rate

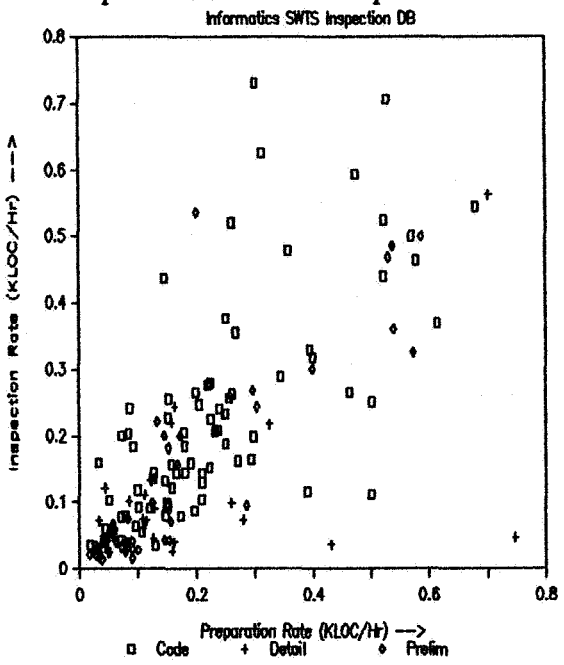


Figure 3

Of any Inspection, we believe the Preliminary Design Inspection is the most critical Inspection to hold, as it helps find modularization errors, data definition errors, and can help to emphasize software re-usability before unit development begins. Based upon major error detection rate and translating preliminary and detailed design lines of design (LOD) to implemented lines of code (LOC), the preliminary design Inspection detects (and removes) a greater number of errors. The translation from lines of design to lines of code is based on a development methodology that requires a preliminary design modularization with logic development where 1 LOD can eventually be coded by 15 to 20 LOC; detailed design logic development is where 1 LOD can be coded by 3 to 10 LOC. Using major errors normalized to estimated implemented LOC, the preliminary design Inspection finds and fixes about 1000 errors per KLOC, the detailed design Inspection locates about 600 errors per KLOC, while the code Inspection is least effective by detecting a mere 20 errors per KLOC. Using the generally accepted cost to repair of an order of magnitude for errors between successive development steps further emphasizes these figures for cost savings purposes: a few ounces of prevention are worth pounds of cure. The SWTS environment uses walkthroughs for reviewing functional requirements specifications; for environments that uniformly use Structured Analysis to generate specifications, the Requirements Specification Inspection would undoubtedly supercede the Preliminary Design Inspection in importance.

Experience in performing Inspections is cumulative and if applied can have an effect on the Inspections process. Over the first two years on the SWTS project, the error rates were widely scattered. In the second year, an examination of the Inspections process resulted in changes in error definition, Inspections procedures, and staff education. Consequently error rates dropped significantly and today remain in a much smaller range.

CONCLUSION

Inspections are not a panacea for Quality Assurance defect removal. They are technical review procedures and may not be appropriate for some situations such

as those needing heavy user interaction (such as user interface definition). They should be used in conjunction with (but probably not as a substitute for) military PDR/CDR large reviews. In appropriate situations, they have been proven to be effective and efficient error detection methods which have extremely important and beneficial "side effects" of accurate planning, scheduling, and tracking for project management and control. The primary effect of Inspections is to move error detection and correction to the earlier (and less costly) development stages. As such, this front-loads the project schedule, but the time is more than recovered during the coding and implementation phases. Consequently, Inspections usage on a project requires proper education, scheduling, and implementation and should not be used on schedule driven projects where the customer understands only two development phases: code and test.

At NASA Ames, based on experience gained using the original IBM model on pilot projects, Inspections have been modified and specialized for numerous projects, development phases, and environments. At Ames, Inspections are expected to play an increasingly major role as a Quality Assurance tool in software development. Some of the directions this can be expected to take are expansion to cover new software languages, incorporation of new structured development methodologies, and modification of the methodologies for the Ames environment based on information gained during Inspections of software developed using those methodologies. Inspections are a significant Quality Assurance tool in their own right and flexible enough to be integrated and implemented with other tools, especially defect prevention, to provide a comprehensive Quality Assurance environment to approach zero defect products.

REFERENCES

1. M.E.Fagan, "Design and Code Inspections to Reduce Errors in Program Development", IBM Systems Journal, Vol.15 No.3, 1976
   (This article can be ordered as a reprint, order no. G321-5033)
2. "Inspections in Application Development - Introduction and Implementation Guidelines", Installation Management Manual GC20-2000-0, IBM Corporation, 1977
3. "Guidelines for Software Inspections", NASA Contractor Report 166521, August 1983, NASA Ames Research Center, Moffett Field, Calif. 94035

THE VIEWGRAPH MATERIALS

for the

G. WENNESON PRESENTATION FOLLOW

SOFTWARE INSPECTIONS AT NASA AMES

METRICS FOR

FEEDBACK

AND

MODIFICATION

GREG WENNESON

INFORMATICS GENERAL CORPORATION

WHAT THEY ARE (AND ARE NOT)

INSPECTIONS :

FORMAL REVIEW PROCEDURES
FOR ERROR DETECTION ONLY
DEFINED TEAM MEMBER ROLES
SPECIFICALLY DEFINED TOOLS
HELD AT SELECTED POINTS IN DEVELOPMENT CYCLE
DEFINED INPUT
DEFINED OUTPUT

INSPECTIONS ARE NOT :

DESIGN SESSIONS
WALKTHROUGHS
EVALUATIONS OF THE AUTHOR
RUBBER STAMP PROCEDURES

HISTORY

AT IBM

MIKE FAGIN, PUBLISHED - 1976
ALSO - O.R.KOHLI, R.R.LARSON, R.A.RADICE

FORMAL GUIDELINES - 1977, 1978

PRODUCTIVITY GAIN 23%

ERROR DETECTION 82%

ERROR REDUCTION 38%

AT NASA AMES

PILOT PROJECTS BY INFORMATICS - 1979
(ALSO COMMERCIAL PILOT PROJECTS)

STANDARDIZED WIND TUNNEL SYSTEM (SWTS)

PRODUCTIVITY GAIN 40%*

ERROR DETECTION 90%*

ERROR REDUCTION 40%*

(* - INCLUDES MAJOR METHODOLOGY CHANGES)

NOW USED ON MOST INFORMATICS AMES PROJECTS

INSPECTION COMPONENTS

DEFINED TOOLS

- STANDARDS
- CRITERIA FOR MATERIALS PREPARATION
- CHECKLISTS FOR ERRORS
- EXIT CRITERIA
- WRITTEN RECORDS AND STATISTICS

TEAM MEMBERS

- MODERATOR
- READER
- INSPECTORS
- AUTHOR

INSPECTION PROCESS

- TEAM SELECTION (PLANNING)
- OVERVIEW
- PREPARATION
- INSPECTIONS SESSIONS — DESK INSPECTION
- REWORK
- FOLLOW-UP

PROBLEM AND STATISTICS RECORDING

PROBLEM RECORDING

MODULE INSPECTION PROBLEM REPORT

"GENERAL" PROBLEMS REPORT

PROBLEM STATISTICS

MODULE PROBLEM SUMMARY

MODULE TIME AND DISPOSITION REPORT

INSPECTION STATISTICS

INSPECTOR TIME REPORT

INSPECTION GENERAL SUMMARY

OUTLINE OF REWORK SCHEDULE

G. Wenneson
Informatics General Corp

INSPECTIONS DATA BASE FOR SWTS

- SUMMARIES -

SUMMARY OF INFORMATICS SWTS PROJECT INSPECTIONS STATISTICS

Type of Inspect'n	Lang.	Total Number Held	Total No "Lines" Inspected	DENSITY-OF-PROBLEMS Per Thousand Lines Major	Minor	Total	TIME-PER-PERSON Per Thousand Lines Meet'g	Prep'n	Total
CODE - NON-DESK Only	ALL Lang	94	51186	22.0	59.9	81.9	4.6	4.0	8.7
	FORTRAN	90	49389	22.4	60.4	82.8	4.6	4.1	8.7
	ASSEMBLY	4	1797	10.1	44.5	54.6	5.0	2.6	7.7
CODE - DESK	ALL Lang	47	23206	21.0	51.3	72.3	3.9	0.0	3.9
	FORTRAN	43	21308	19.1	48.1	67.2	3.7	0.0	3.7
	ASSEMBLY	4	1898	42.6	87.6	130.3	6.3	0.0	6.3
DETAILED DESIGN	ALL Lang	44	10349	76.74	144.6	221.3	14.5	9.8	24.3
	FORTRAN	40	9205	83.1	143.4	226.5	14.5	9.2	23.7
	ASSEMBLY	4	1144	25.3	153.9	179.2	14.3	14.4	28.7
PRELIMINARY DESIGN	ALL Lang	43	13268	68.1	107.5	175.7	10.8	5.4	16.1
	FORTRAN	41	12570	54.3	89.8	144.1	9.1	5.5	14.6
	ASSEMBLY	2	698	316.6	426.8	743.4	39.8	3.7	43.6

This chart summarizes the statistics from Informatics inspections on the NASA Ames SWTS project. The statistics are weighted averages, each inspection being weighted by its size, in lines of design or code.

G. Wenneson
Informatics General Corp

STATISTICS USE

AUTHOR

PROBLEM REPORTS
MODULE PROBLEM SUMMARY
PREVIOUS INSPECTION STATISTICS

DESIGN TEAM AND MANAGER

PROBLEM REPORTS
MODULE PROBLEM SUMMARY
OUTLINE OF REWORK SCHEDULE
MODULE TIME AND DISPOSITION
INSPECTION GENERAL SUMMARY
PREVIOUS INSPECTION STATISTICS

PROJECT MANAGER; TEST GROUP; QA GROUP

MODULE PROBLEM SUMMARY
INSPECTION GENERAL SUMMARY
PREVIOUS INSPECTION STATISTICS

SOFTWARE ENGINEERING

MODULE PROBLEM SUMMARY
INSPECTION GENERAL SUMMARY
PREVIOUS INSPECTION STATISTICS

CODE INSPECTION SUMMARIES
NEW FORTRAN CODE, MODIFICATIONS, AND BOTH

SUMMARY OF INFORMATICS SWTS PROJECT INSPECTIONS STATISTICS

Type of Inspect'n	Lang.	Total Number Held	Total No "Lines" Inspected	DENSITY-OF-PROBLEMS Per Thousand Lines Major	Minor	Total	TIME-PER-PERSON Per Thousand Lines Meet'g	Prep'n	Total
CODE - NON-DESK CHECK									
	FORTRAN	90	49389	22.4	60.4	82.8	4.6	4.1	8.7
	/New	46	25981	26.3	68.3	94.6	5.5	4.9	10.3
	/Mods	13	7019	17.2	42.4	59.6	3.0	3.2	6.2
	/Both	31	16389	18.5	55.6	74.1	3.9	3.3	7.2
CODE - DESK CHECK									
	FORTRAN	43	21308	19.1	48.1	67.2	3.7	0.0	3.7
	/New	8	4121	26.3	51.7	78.0	4.9	0.0	4.9
	/Both	25	14453	18.6	50.1	68.7	3.4	0.0	3.4
	/Mods	10	2734	10.6	32.2	42.8	3.8	0.0	3.8

This chart summarizes the statistics from Informatics inspections on the NASA Ames SWTS project. The statistics are weighted averages, each inspection being weighted by its size, in lines of design or code.

G. Wenneson
Informatics General Corp

INSPECTIONS DATA BASE

"MAJOR" PROBLEM DISTRIBUTION, BY PERCENT

PRELIMINARY DESIGN

Category	FORTRAN	ASSEMBLER
SPECIFICATION	10%	13%
CLARIFICATION	17	1
DATA	18	21
LOGIC	21	21
I/F	5	20
LINKAGES	20	
PERFORMANCE	4	3

DETAILED DESIGN

DETAIL	9	29
LOGIC	29	66
DATA	20	1
LINKAGES	22	1
RETURN CODES	5	

CODE

FUNCTIONALITY	9	4
DATA	19	37
CONTROL	18	22
LINKAGES	24	23
READABILITY	17	2
REG. USE		12

PREVIOUS INSPECTIONS EFFECT ON MAJOR ERROR RATES

STAGE OF DEVELOPMENT	NUMBER OF PREVIOUS INSPECTIONS 0	1	2	3
CODE NON-DESK	17.7	30	32.6	38
CODE DESK	15.1	27	30	21
DETAIL DESIGN	95	79	54	-
PRELIM. DESIGN	58	45.6	-	-

Major Errors Per KLOC

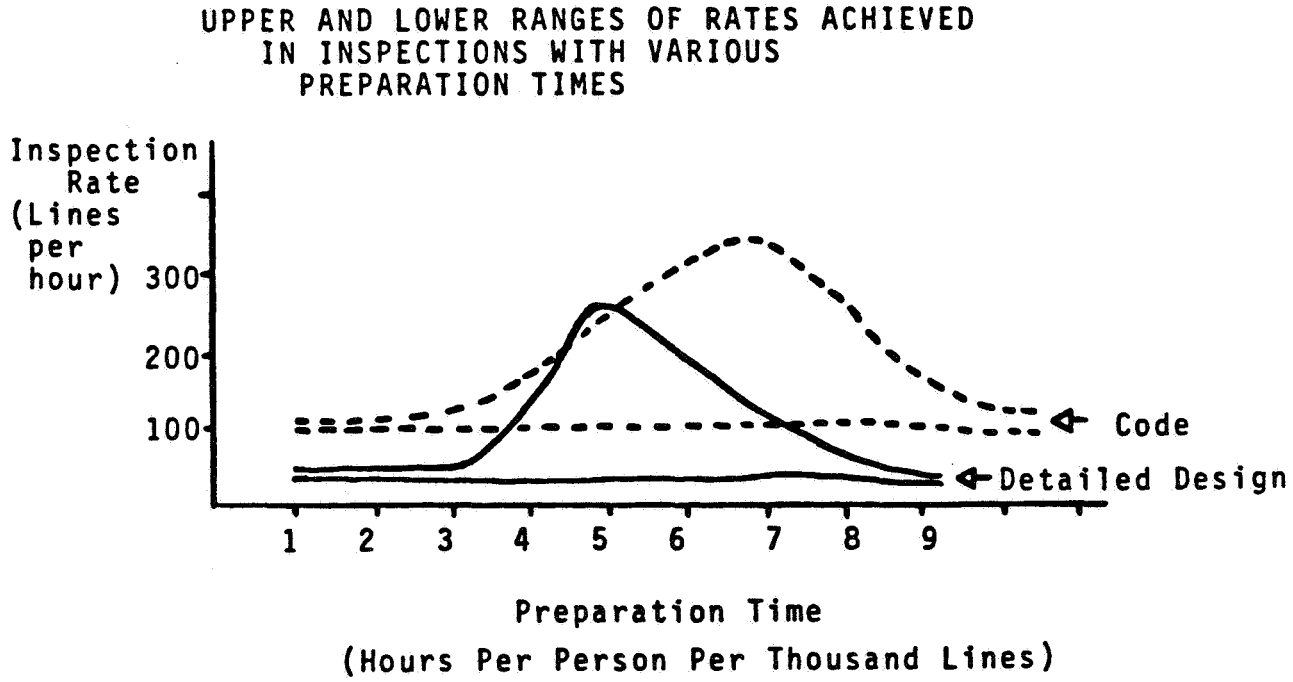
AND ON PREPARATION AND MEETING TIME

STAGE OF DEVELOPMENT	NUMBER OF PREVIOUS INSPECTIONS 0	1	2	3
CODE NON-DESK	8.2	9.2	9.1	10
CODE DESK	4	3.2	3.5	2.5
DETAIL DESIGN	27.7	23.0	9.5	-
PRELIM. DESIGN	14.7	14.4	-	-

HOURS of Preparation plus Meeting time Per KLOC

INSPECTIONS RATE AND PREPARATION TIME RELATIONSHIP

An important area of consideration is the amount of preparation time required in order to allow the participants to proceed at a reasonable rate in the inspection meeting. The graph below, based on the individual inspections to date, suggests that preparation times of 4-7 hours per 1,000 lines may allow the team to proceed at an optimum rate in the meetings. Less preparation time will cause the meeting to slow down because of poor understanding and many questions. More preparation time may have a negative impact on the rate because of over-emphasizing minor problems or discussing the functionality or goals during code or design inspections.



INSPECTIONS AS A PROJECT COORDINATION TOOL

INSPECTIONS CAN INTEGRATE THE FOUR MAJOR PROJECT FACTORS:

PROJECT MANAGEMENT

METHODOLOGY

QUALITY ASSURANCE

STAFF PERFORMANCE

THRU:

REINFORCEMENT OF METHODOLOGY AND STANDARDS
MAJOR MILESTONE TRACKING INFORMATION MATCHING WBS
DETAILED TRACKING AND ESTIMATING INFORMATION MATCHING WBS
DETAILED ERROR AND DESIGN NEEDS AT EACH DEVELOPMENT STAGE
EASY EXTRACTION OF TECHNICAL INFORMATION ABOUT COMPONENTS
INDICATIONS OF TRAINING AREAS NEEDING ATTENTION ACROSS THE PROJECT
INDICATIONS DIRECTLY TO INDIVIDUAL STAFF MEMBERS OF THEIR TRAINING NEEDS

G. Wenneson
Informatics General Corp

ALMOST THE END

CAUTIONS

DOESN'T SUBSTITUTE FOR THINKING
MUST BE SCHEDULED AT BEGINNING - CAN'T BE "TACKED" ON
PARTICIPANTS MUST BE PROPERLY TRAINED
NEED CUSTOMER UNDERSTANDING AND SUPPORT
MANAGEMENT DIRECTION AND SUPPORT CRUCIAL
STATISTICS ARE FOR BETTER SOFTWARE AND MANAGEMENT,
NOT A NUMBERS EXERCISE

WHERE TO GO FROM HERE

EXPAND TO NEW LANGUAGES AND DESIGN TECHNIQUES
EXPAND TO NEW METHODOLOGIES AND SUPPORT TOOLS
FEEDBACK TO CURRENT METHODOLOGIES
EXPAND TO OTHER APPLICABLE COMPANY/CONTRACT AREAS