ANALYSIS OF A SUCCESSFUL INSPECTION PROGRAM

Ray Madachy
Linda Little
Sylvia Fan

Software Engineering Process Group
Litton Data Systems
Agoura Hills, CA

ABSTRACT

Litton Data Systems has institutionalized the inspection process, and achieved dramatic results in terms of defect prevention and cost savings thus far. Additionally, several findings have been gleaned from an analysis to optimize the process. Over 300 inspections have been performed over the last two years on many types of documents, and this paper describes some quantitative results to-date from the initial "champion" project.

BACKGROUND

Litton was first trained in inspections by Tom Gilb in 1989. His method differs from Fagan's [Gilb 88], and Litton has subsequently modified Gilb's method for in-house. The success of our program owes much to strong executive support. Inspections are now the cornerstone of our peer review process.

Over 400 software personnel have been trained in inspections, and inspections are now being used on four major development programs. Our software director set project goals to save at least 50% of integration effort by spending more effort during design and coding for inspections. Thus far, we appear to be achieving this goal.

Unique properties of the Litton inspection process include no "reader" role, no discussion on defect category during inspection, a routing process for inspection results, no time limit on causal analysis and the use of a Software Engineering Process Group (SEPG) Peer Review Coordinator. A standard reporting form, as shown in Table 1, has been devised for collecting the inspection data.

Though project management has collected some high-level inspection statistics, the SEPG instituted an inspection database as part of its metrics program to evaluate process improvement. Data from the form in Table 1 goes into the database, and is regularly entered at the end of each week. The database was used for this analysis, and validated against high-level project management data. The provision on the form for defect categories supporting causal analysis is a recent addition, so little data has been collected for defect category analysis up to this point. The following sections describe some results-to-date of our analysis of the inspection data.

Table 1: Typical Data Sheet

INSPECTION STATISTICS

MODERATOR: _____ DATE: 16 November 1993

SUBJECT: DP 18.14 CHUNK: 1 SUBJECT TYPE: Detail Design

PRE INSPECTION MEETING DATA

INSPECTOR	PREPARATION TIME (minutes)	MAJORS	MINORS	TOTAL ITEMS
A	40	1	1	2
B	60	0	6	6
C	60	0	1	1
D	30	0	5	5
E				
F				
TOTALS	190	1	13	14

INSPECTION MEETING DATA

Estimated SLOCs (from FDB): N/A

Changed Pages/Changed Lines Inspected: 550 Start Time: 9:09

Total MAJORS Asserted: 0 Stop Time: 9:40

Total MINORS Asserted: 22 Inspection Time (min): 31

Total Defects Asserted: 22 Defects Asserted Per Minute: .70

Changed Pages/Changed Lines Inspected Per Hour: _____

New Defects Found During Meeting: 5

POST INSPECTION MEETING DATA

Total MAJORS Accepted: 0 Total Minors Accepted: 19

Rework Hours: 4 Hours Working Causal Analysis Items: N/A

Number of Causal Analysis Items Requiring Action: None

Category Totals: 1: 2 2: 1 3: 0 4: 1 5: 4 6: 0 7: 3 8: 1

9: 0 10: 0 11: 0 12: 0 13: 7

ANALYSIS

This analysis concerns both optimization of the inspection process, as well as performing a cost/benefit analysis to determine how much extra effort is used during design and coding for inspections and how much is saved during testing and integration. This effect on project effort is shown in Figure 1 from [Fagan 86].

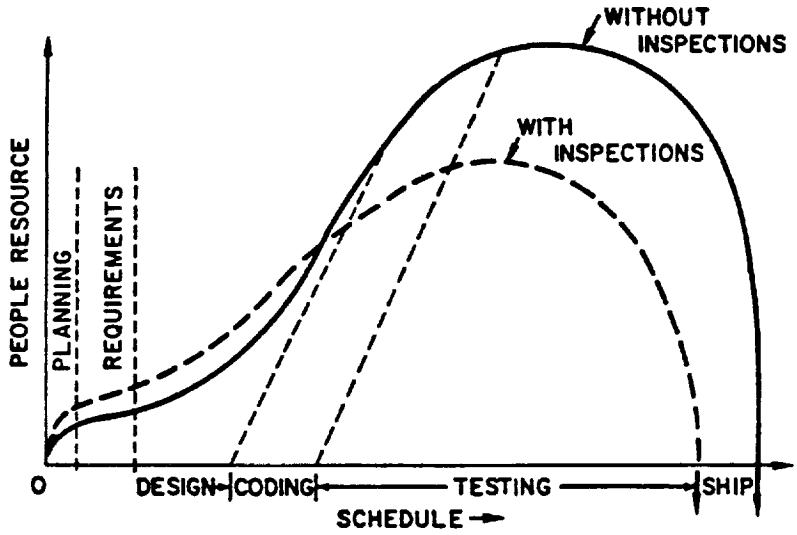


Figure 1: Effect of Inspections on Project Effort (from [Fagan 86])

The following formulations are used in this analysis:

defects found = items from preparation + new items

inspection effort = preparation effort + meeting effort + rework effort

defect removal effectiveness = defects found / inspection effort

finding rate = defects / meeting time

inspection rate = inspected pages / meeting time

meeting effort = (meeting time) * (# personnel involved).

Preparation effort is the total effort for all inspectors. A major defect is defined as an error that would lead to a trouble report during testing and integration. A new item is one found during the inspection meeting that was not identified by any inspectors during pre-inspection preparation. We decided to separate new items discovered at the inspection meeting from defects noted during preparation, as we have observed that certain practices increase the new item finding rate and wish to investigate further.

Several types of documents are inspected: requirements (both requirements description and requirements analysis); design (top-level and detailed design); code; and change requests. Summary statistics are shown in Table 2. The total inspection effort was distributed as follows: preparation effort - 27%, inspection meeting effort - 39% and rework effort - 34%. The last column in Table 2 represents the defect removal effectiveness. As seen, the effectiveness decreases for later documents.

Table 2: Summary Statistics

Subject Type	Total Defects	Total Majors	Inspection Effort	# Pages	LOC Inspected	Major Defects/ Inspection Effort
REQUIREMENT DESCRIPTION	460	72	78	179	0	.923
REQUIREMENT ANALYSIS	2165	177	483	1065	0	.366
HIGH LEVEL DESIGN	2199	188	655	1592	0	.287
DETAILED DESIGN	1550	127	610	1387	19007	.208
Subtotal	6374	564	1826	4223	19007	.309
CODE	4272	432	1742	5047	149361	.248
CHANGE REQUEST	814	27	309	1579	0	.087
Grand total	11460	1023	3877	10849	168368	.264

When the defect density for these document types are ordered by activity, the results show that the defects steadily decrease since the predecessor artifacts were previously inspected. This is shown in Figure 2, overlayed with similar results from JPL [Kelly-Sherif 90]. The trend seems to corroborate the previous results. Code is not shown because of inconsistencies in reporting the document size. These results strongly support the practice of inspecting documents early as possible in the life cycle.

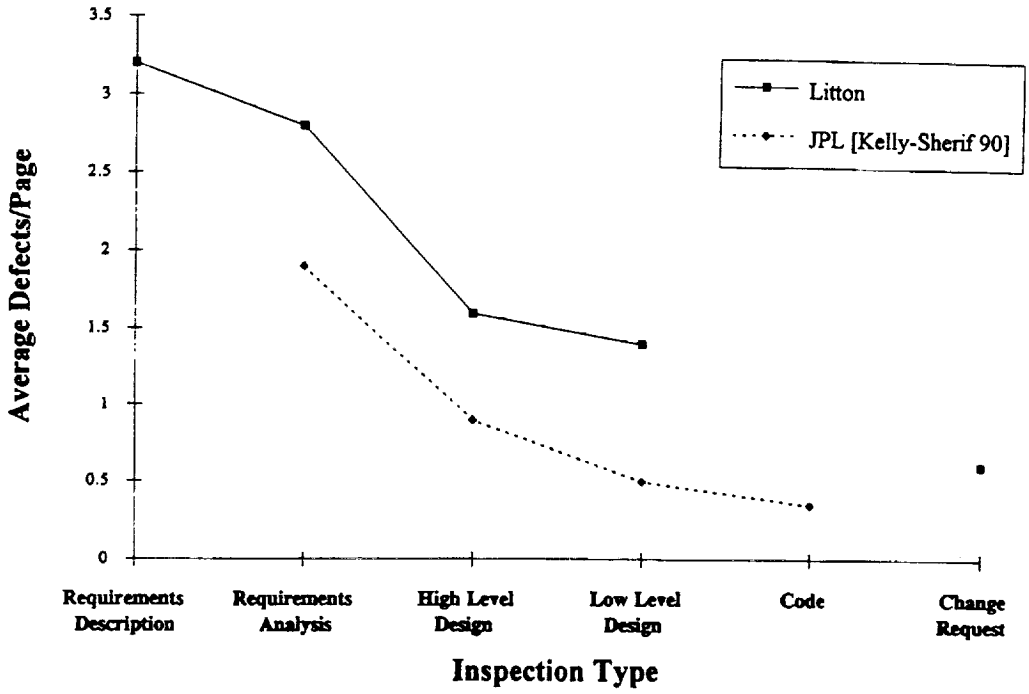


Figure 2: Defect Density per Subject Type

Project Effort for Inspections

We tracked the inspection effort as a portion of the total software development effort over the last year. The effects of schedule pressure were seen on inspection data, much as it is observed for staff coding and integration efforts before a "drop dead" date. This trend is shown in Figure 3, where the percent of project effort dedicated to inspections is plotted. The monthly inspection effort profile shows extreme peaks right before two Technical Interface Meetings where the customer evaluates the inspected documents, and right before a Preliminary Design Review with the customer. For this time period of regular inspections, an average of 2.9% of effort was spent on inspections. Both preparation time and inspection time increased during the peaks, but preparation time increased much more severely. The relatively small increase in actual inspection time indicates that the meeting process remained under control, instead of moderators drastically slowing the pace to find more defects. These dynamic effects on effort due to schedule pressure affect the long term averages and short-term project behavior, and should be kept in mind when planning effort or evaluating project trends since process stability is affected.

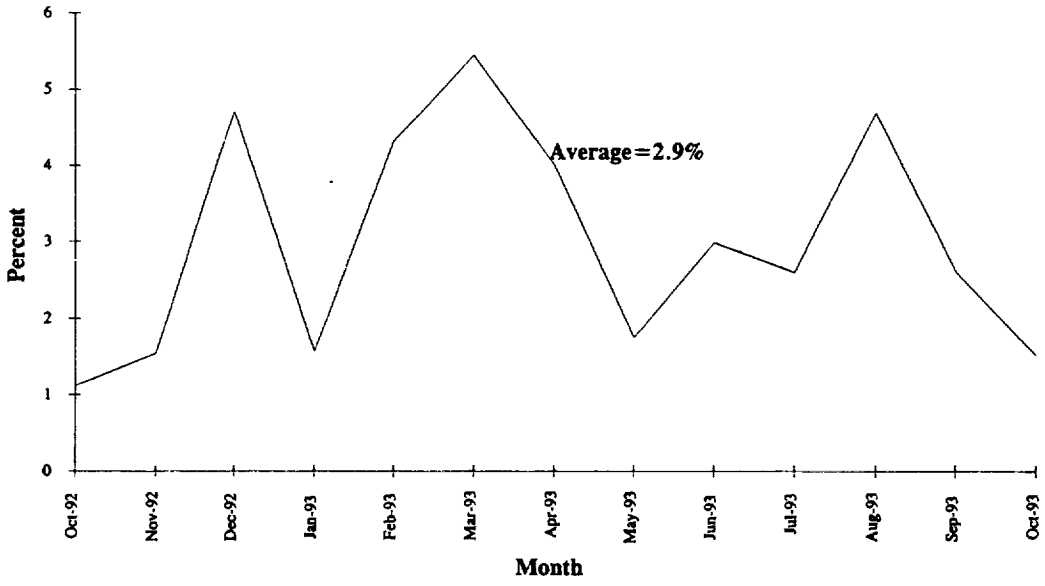


Figure 3: Percent of Project Effort for Inspections

Based on statistics on the inspection effort and knowledge about the process, a bottoms-up inspection costing algorithm has been devised. It identifies effort for pre-inspection, inspection and post-inspection activities based on the type and length of the inspected document(s). The algorithm is being included in a cost model used in the Division.

Return on Investment

The following return on investment (ROI) method of tracking inspection success calculates the difference of testing time saved and inspection effort for each meeting [Grady 92], [Rodriguez 91]. It uses the formulation

ROI= (found defects) * (average effort to fix defect in test) - inspection effort

for each inspection meeting, using major defects only. The rationale for equating test defects with design defects follows from the previous definition of a major defect. At Litton, our historical data on the product line shows an average of 17.6 person-hours is spent to fix defects during testing. Using this value, the ROI for each inspection is shown in Figure 4. Overall, the total return from these inspections has been 14,210 person-hours of effort, with an average return of 63.4 person-hours per inspection. Out of 223 inspections, 139 have provided savings.

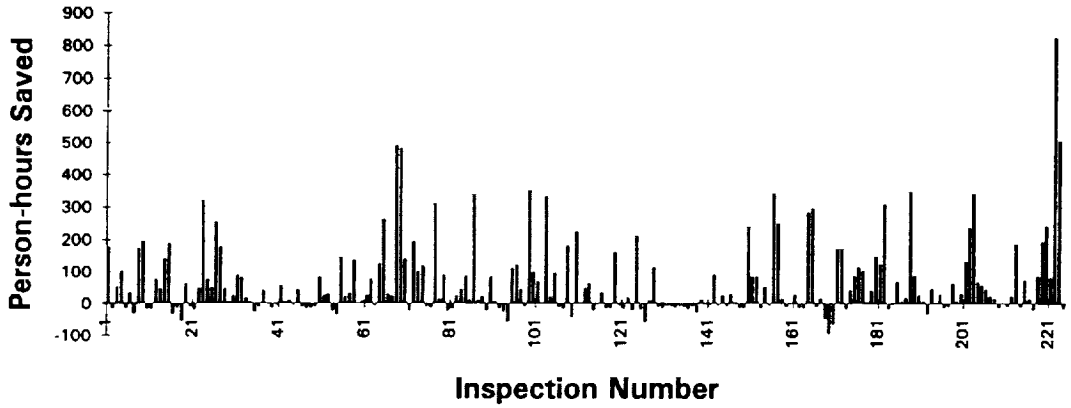


Figure 4: Return on Investment per Inspection

Statistics have been kept for several years on the number of trouble reports encountered during integration and the associated costs to fix them for this particular product line. When comparing trouble report data before and after inspections were introduced, there is a 76% reduction in trouble report density. This appears to be on the high end of reported results for defect reduction. Using the historical data on average efforts to fix inspection defects and trouble reports, about 573 labor-hours per KSLOC have been saved.

Process Control

Figures 5-7 show control charts for defect finding rate, design document inspection rate and code inspection rate. The bands shown on them represent the average values plus or minus a standard deviation for the upper and lower control limits. The overall items/minute for this project is apparently on the low end of the industry standard. The variances of inspection rates are higher relative to the variances of defect finding efficiency due to aforementioned dynamic schedule effects and other phenomena such as "process tweaking" and new personnel.

In Figure 5, it appears that the defect finding rate seems to have come down since the beginning of the program. This trend of project evolution could be due to the earlier documents having higher defect densities per Figure 2. In Figure 6, note that there seems to be a relatively sudden ending to the activity near 5/93. This corresponds to the date when coding started in earnest, and much design documentation was completed at that time.

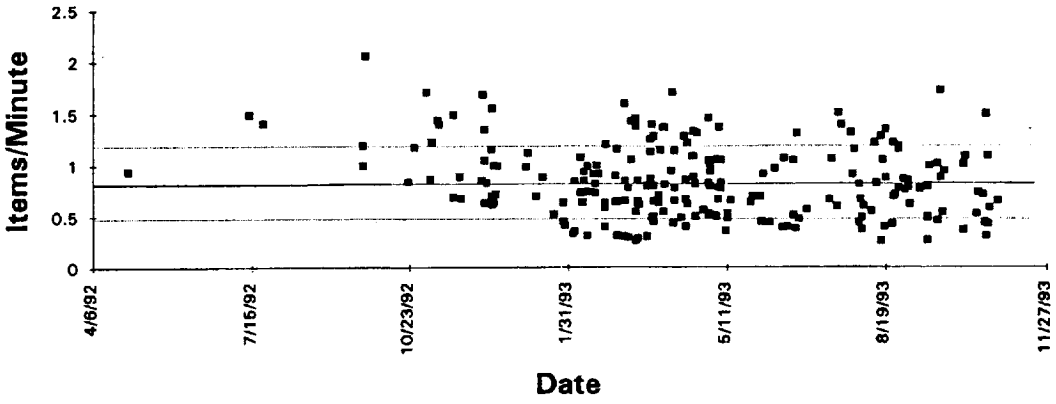


Figure 5: Defect Finding Rate Control Chart

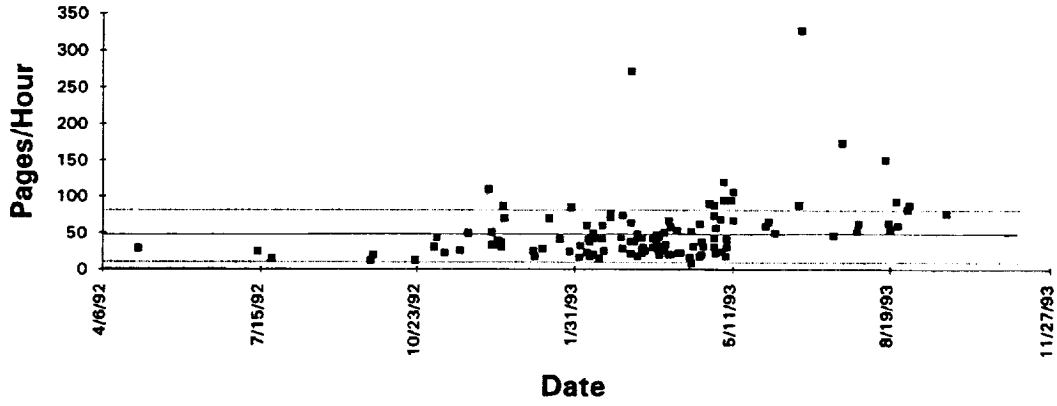


Figure 6: Document Inspection Rate Control Chart

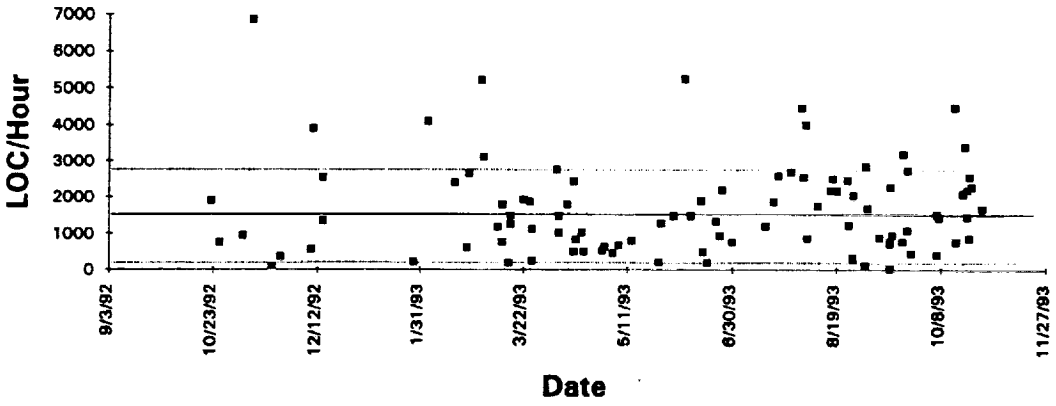


Figure 7: Code Inspection Rate Control Chart

When analyzing the data for adherence to process control limits for inspection rate and item finding rate, several outlying data points were identified. Upon further investigation, it was seen that there was a single moderator who was not particularly well-

suited to the task. This moderator had been previously identified as one who rushed through the documents too fast, and the analysis confirmed that perception.

Along these same lines of inquiry, we wanted to see if outlying moderators could be detected by looking at individual performance. Figure 8 shows the average items found per minute for all moderators, and they all are in the same approximate range. This depiction showed some disparate ranges between moderators earlier in the program, thus we feel that the process has stabilized among moderators over time. This provides confidence that the process is relatively independent of individual moderators used and shows the benefits of good training.

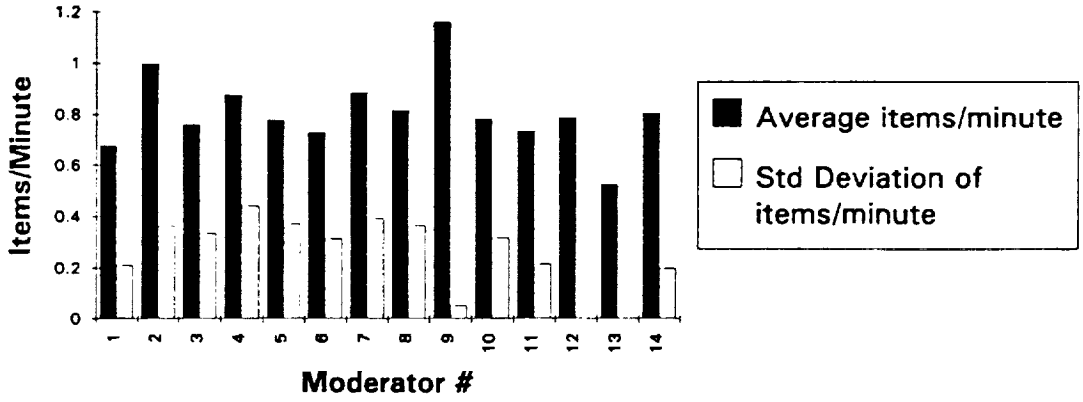


Figure 8: Moderator Finding Efficiency

The inspection rate is an important parameter to optimize. Going too slow may waste time, but going too fast will miss defects. Figure 9 shows the average defect density for different ranges of inspection rate. Note that we have normalized the defects found by the document size. As seen, going faster than about 50 pages per hour seems to substantially decrease the defects found. The overall average is 48 pages per hour, though we are currently trying to slow down the rate at meetings to be closer to 30-40 pages per hour.

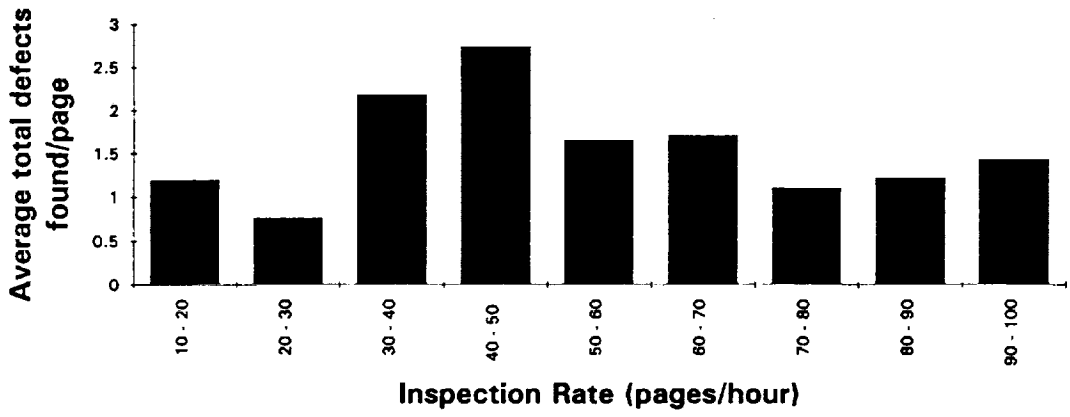


Figure 9: Effect of Inspection Rate on Defects Found

We also wish to know the optimal number of inspectors to maximize the defect removal effectiveness. Other studies have shown that 4-5 inspectors is the optimal number [Grady 92], [Gilb 88], and our data also supports this number. Figure 10 shows the average defect removal effectiveness for the number of inspectors. From our data, the optimum does not appear quite as clear-cut for major defects alone.

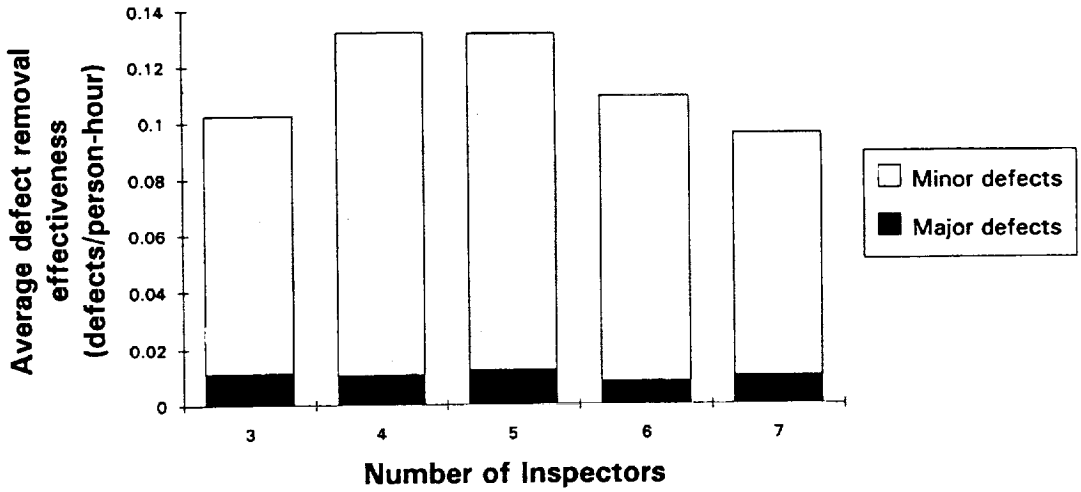


Figure 10: Average Defect Removal Effectiveness vs. Number of Inspectors

Yet another process parameter to optimize is the ratio of preparation time to inspection time. Grady and others [Grady 92] indicate an optimum value greater than 1.75, with some sites averaging about 1.5. Figure 11 shows our results. The optimum ratio appears to be somewhere between .5 and 2.0, with our average ratio being 1.4.

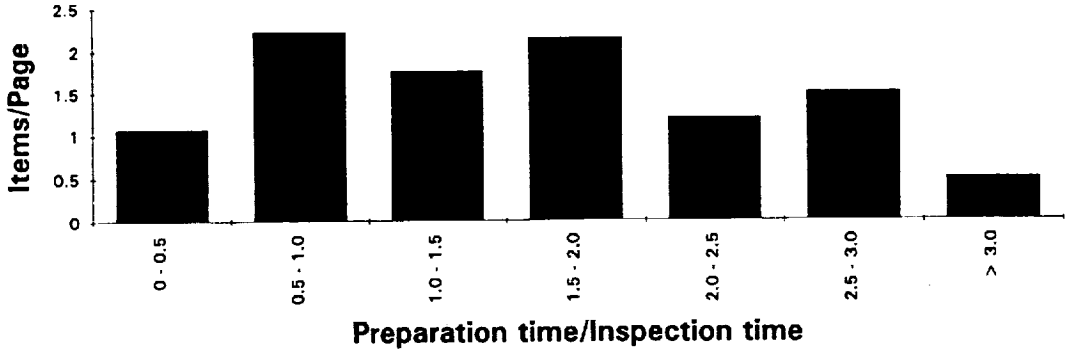


Figure 11: Defects Found vs. Preparation/Inspection Time

One counter intuitive result not previously reported in the literature is a high correlation (.8) between the preparation time (averaged over the inspectors) and new items/page or new items/KSLOC found during the inspection. Instead of catching less new

defects during inspection after more thorough preparation to identify defects before the meeting, the inspectors are more familiar with the subject matter and thus able to find even more new items during the inspection meeting. A scatterplot of this data for all non-code documents is shown in Figure 12.

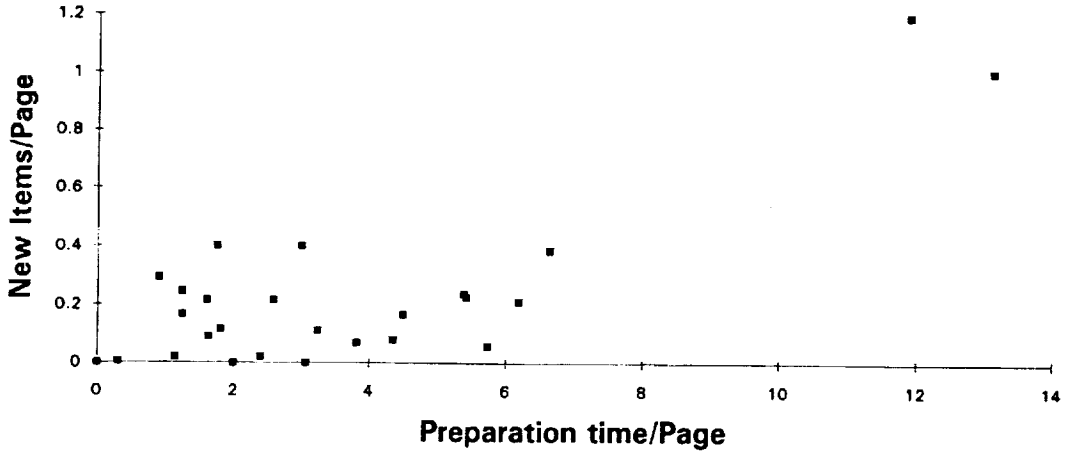


Figure 12: Effect of Preparation Time on New Items Found

As expected, there were also high correlations between preparation time and total items found (pre-inspection and new items) and inspection time versus both total items and new items. These relationships are more stable for design documents as opposed to code documents, due to the reduced clarity and understanding of program code.

Resulting Defect Density During Integration

Inspections are expected to severely reduce the number of problems encountered during testing and integration activities. Though this project is not 100% complete, data from the first couple of builds supports this hypothesis. Figure 13 shows the resulting defect density during integration, as the trouble report density running average by build. The first 10 builds were before inspections started, and the last two are for the current project within the same product line after inspections were mandated. Other project environmental factors are virtually identical except for the use of inspections. We are confident that something is helping to reduce the trouble report density.

Attempts were also made to perform a t-test on individual modules to determine if there are significant differences in defect density during testing due to inspection. The metrics tracking procedures did not lend themselves to such analysis due to intractable mappings between design documents and implemented code functions, actual code sizes could not be mapped at a low level to what was being inspected, and the inability to distinguish new development from modified code.

This experience was a lesson learned. In order to evaluate new techniques in the future for process improvement, the metrics procedures have to be restructured on the program, so that individual modules can be tracked throughout the lifecycle. Recommendations for the changes are being documented.

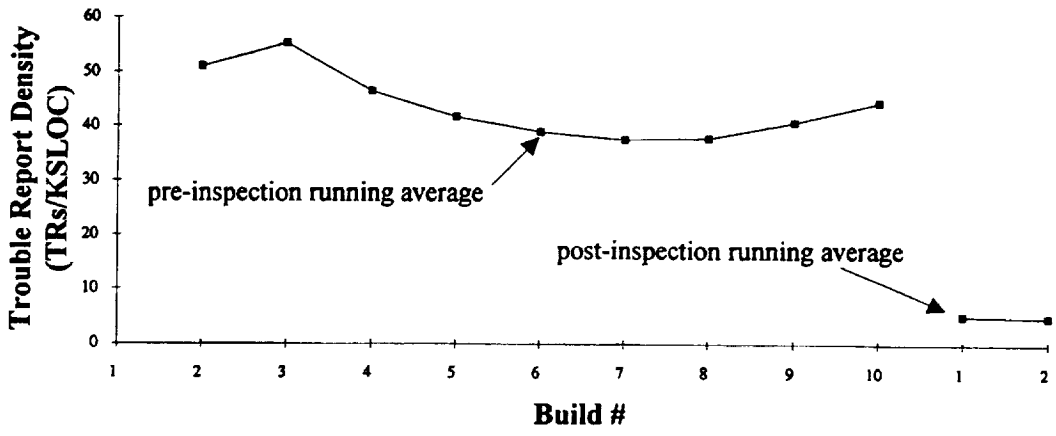


Figure 13: Defect Density During Integration

CONCLUSIONS AND FUTURE WORK

Though this initial major project using an inspection-based process is not complete, the preliminary results indicate a large return on investment. Since inspections began, inspectors have increased their effort and authors are producing higher quality documents, indicating buy-in to the new process.

Some process stabilization occurred during the first year of practice, and the teaching method and the process itself has been modified based on the statistical results. Inspections are being used on more ongoing projects, and the results appear to be repeatable within the company. The process is now mandated on all new projects.

This analysis has helped to identify areas of improvement for software metrics collection. This impetus will lead to revised procedures to enable more thorough analysis of process improvement activities.

Analysis of inspection data will continue in order to understand and account for the confounding factors of inspectors and authors, to continue identifying optimal practices, to perform more detailed cost/benefit analysis and to investigate other related process issues. Analysis of variance will be performed to determine the contribution of different process parameters to overall defect removal effectiveness.

With the recent enhancement to the data form for defect category information, defect metrics will be collected to support causal analysis activities. Additionally, a system dynamics model of an inspection-based process is under development, and will be calibrated to Litton data to assist in process improvement activities.

BIBLIOGRAPHY AND SELECTED NOTES

[Ackerman et al. 84] Ackerman AF, Fowler P, Ebenau R, *Software inspections and the industrial production of software*, in "Software Validation, Inspections-Testing-Verification-Alternatives" (H. Hausen, ed.), New York, NY, Elsevier Science Publishers, 1984, pp. 13-40

Describes inspections as performed at Bell Laboratories and discusses use of inspections in conjunction with other verification and validation techniques.

[Boehm 81] Boehm BW, *Software Engineering Economics*. Englewood Cliffs, NJ, Prentice-Hall, 1981, pp. 383-386

Discusses error removal production functions for inspection, unit test and other error removal techniques. Points out difficulty of overlap between methods for removing different classes of errors.

[Buck-Dobbins 84] Buck R, Dobbins J, *Application of software inspection methodology in design and code,* in "Software Validation, Inspections-Testing-Verification-Alternatives" (H. Hausen, ed.), New York, NY, Elsevier Science Publishers, 1984, pp. 41-56

[Fagan 76] Fagan ME, *Design and code inspections to reduce errors in program development*, IBM Systems Journal, V. 15, no. 3, 1976, pp. 182-210

The original article by Mike Fagan that introduced the IBM inspection experience.

[Fagan 86] Fagan ME, *Advances in software inspections,* IEEE Transactions on Software Engineering, V. SE-12, no. 7, July 1986, pp. 744-751

A more recent article by Fagan provides additional evidence of inspection benefits over the years, indicating slight front-end loading of the development effort and significant reduction in testing and rework effort.

[Freedman-Weinberg 82] Freedman D, Weinberg G, *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Prgoram, Projects and Products*, Little Brown, 1982

Good material on the human and organization aspects of inspections.

[Gilb 88] Gilb T, *Principles of Software Engineering Management*. Addison-Wesley, Wokingham, England, 1988, pp. 205-226, 403-422

Gilb originally taught the inspection method in-house, which was attended by high-level engineering management. Their strong support of the method led to our inspection-based process. This book provides ample detail to start an inspection progam.

[Grady 92] Grady R, Caswell D, *Practical Software Metrics for Project Management and Process Improvement* Prentice-Hall, Englewood Cliffs, NJ, 1992

Good current book on process improvement metrics with relatively brief but worthwhile treatment of inspections. Has an illustrative complete example of calculating inspection savings and cost/benefit on page 180.

[Kelly-Sherif 90] Kelly J, Sherif J, *An analysis of defect densities found during software inspections*, Proceedings of the Fifteenth Annual Software Engineering Workshop, Goddard Space Flight Center, 1990

[Radice-Phillips 88] Radice RA, Phillips RW, *Software Engineering - An Industrial Approach*, Englewood Cliffs, NJ, Prentice-Hall, 1988, pp. 242-261

A good overall treatment and summary of how to do inspections, by someone who helped pioneer inspections at IBM.

[Remus 84] Remus H, *Integrated software validation in the view of inspections /reviews,* in "Software Validation, Inspections-Testing-Verification-Alternatives" (H. Hausen, ed.), New York, NY, Elsevier Science Publishers, 1984, pp. 57-64

[Rodriguez 91] Rodriguez S, *SESD inspection results*, April 1991

The ROI tracking method was used at Hewlett-Packard.

[Scott-Decot 85] Scott B, Decot D, *Inspections at DSD - automating data input and data analysis*, HP Software Productivity Conference Proceedings, 1985, pp. 1-79 - 1-80

[Weller 93] Weller E, *Three years worth of inspection data*, IEEE Software, September 1993, pp. 38 - 45

Weller published a previous article in Crosstalk on the first two years of data at Bull HN Information Systems, and this article improves upon it.

ANALYSIS OF A SUCCESSFUL INSPECTION PROGRAM

Ray Madachy
Linda Little
Sylvia Fan

Litton Data Systems
Agoura Hills, CA

Presented at the Eighteenth Annual Software Engineering Workshop
NASA Goddard Space Flight Center
December 1, 1993

Outline

- Introduction and background
- Defect density versus inspection subject
- Inspection effort
- Return on investment
- Process control
- Defect density during integration
- Conclusions and future work
- References

Litton
Data Systems

Unique Properties of Litton Inspection Process

- No "reader" role (Fagan).
- No discussion on defect category during inspection.
- Routing process.
- No time limit on causal analysis.
- SEPG Peer Review Coordinator serves as moderator.

Typical Data Sheet

INSPECTION STATISTICS

MODERATOR: _____ DATE: 16 November 1993

SUBJECT: DP 10.14 CHUNK: 1 SUBJECT TYPE: Detail Design

PRE INSPECTION MEETING DATA

INSPECTOR	PREPARATION TIME (minutes)	MAJORS	MINORS	TOTAL ITEMS
A				
B	40	1	1	2
C	60	0	5	5
D	60	0	1	1
E	30	0	5	5
F				
TOTALS	190	1	12	14

INSPECTION MEETING DATA

Estimated SLOCs (from PDS): N/A

Changed Pages/Changed Lines Inspected: 550 Start Time: 9:09

Total MAJORS Asserted: 8 Stop Time: 9:40

Total MINORS Asserted: 22 Inspection Time (min): 31

Total Defects Asserted: 22 Defects Asserted Per Minute: .70

Changed Pages/Changed Lines Inspected Per Hour: _____

New Defects Found During Meeting: 8

POST INSPECTION MEETING DATA

Total MAJORS Accepted: 0 Total Minors Accepted: 19

Rework Hours: 4 Hours Working Causal Analysis Items: N/A

Number of Causal Analysis Items Requiring Action: None

Category Totals: 1: 2 2: 1 3: 0 4: 1 5: 4 6: 0 7: 3 8: 1
9: 0 10: 0 11: 0 12: 0 13: 7

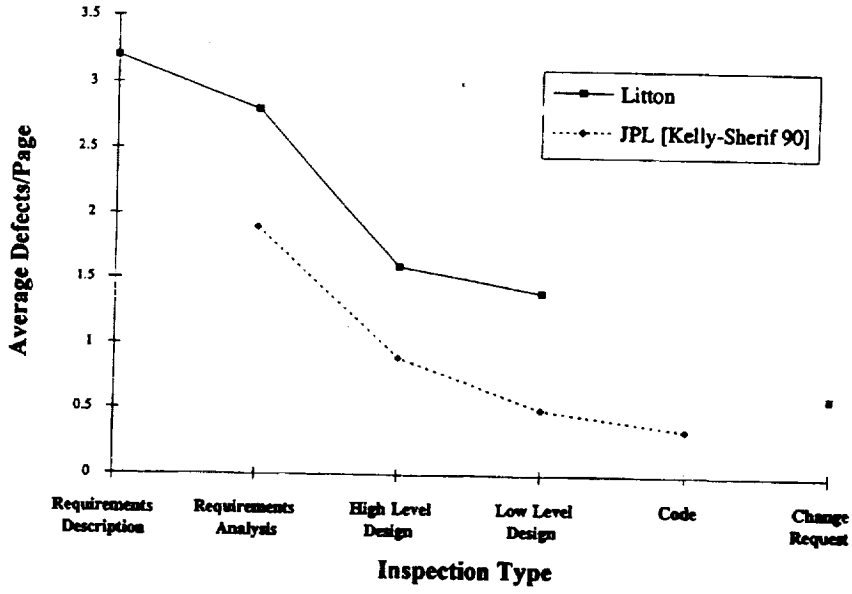
Litton
Data Systems

Summary Statistics

Subject Type	Total Defects	Total Majors	Inspection Effort	# Pages	LOC Inspected	Major Defects/ Inspection Effort
REQUIREMENT DESCRIPTION	460	72	78	179	0	.923
REQUIREMENT ANALYSIS	2165	177	483	1065	0	.366
HIGH LEVEL DESIGN	2199	188	655	1592	0	.287
DETAILED DESIGN	1550	127	610	1387	19007	.208
Subtotal	6374	564	1826	4223	19007	.309
CODE	4272	432	1742	5047	149361	.248
CHANGE REQUEST	814	27	309	1579	0	.087
Grand total	11460	1023	3877	10849	168368	.264

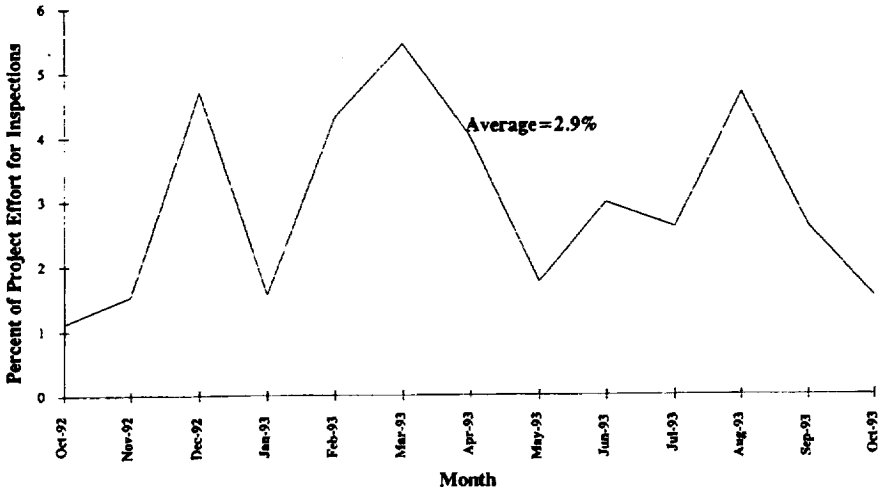
Litton
Data Systems

Defect Density per Subject Type



Litton
Data Systems

Inspection Effort



Return on Investment

- For each inspection, ROI = (test effort saved) - (inspection effort)

  where

  test effort saved =

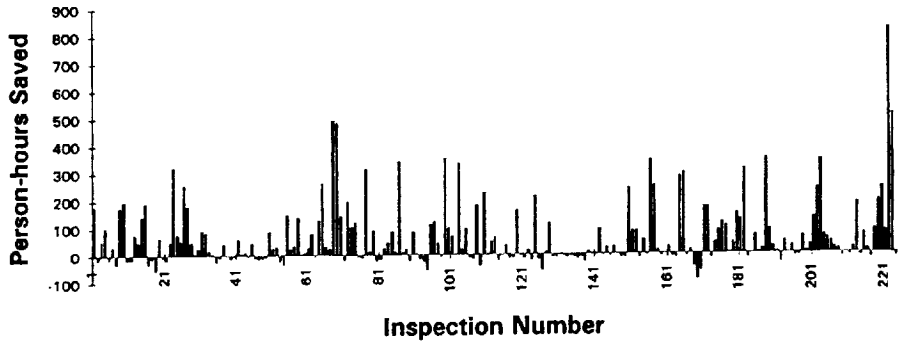
  (# major defects found)*(average effort to fix defect during test)

  inspection effort = preparation effort + meeting effort + rework effort

  = total preparation effort
  + (meeting time) * (# personnel involved in meeting)
  + rework effort

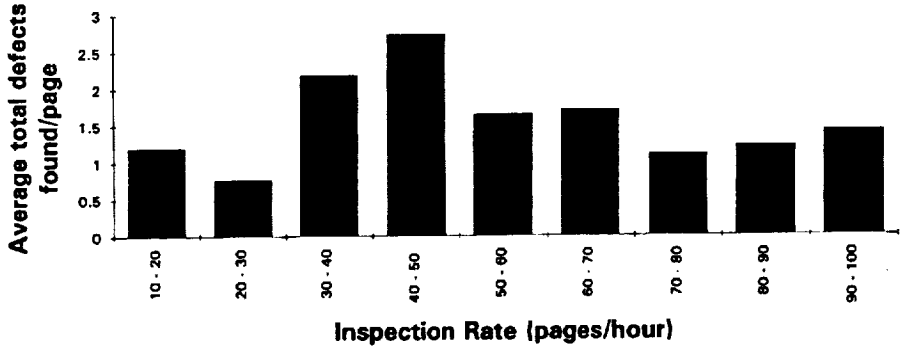
Return on Investment

total return = 14210 person-hours
average inspection savings = 63.4 person-hours
139/223 inspections saved time



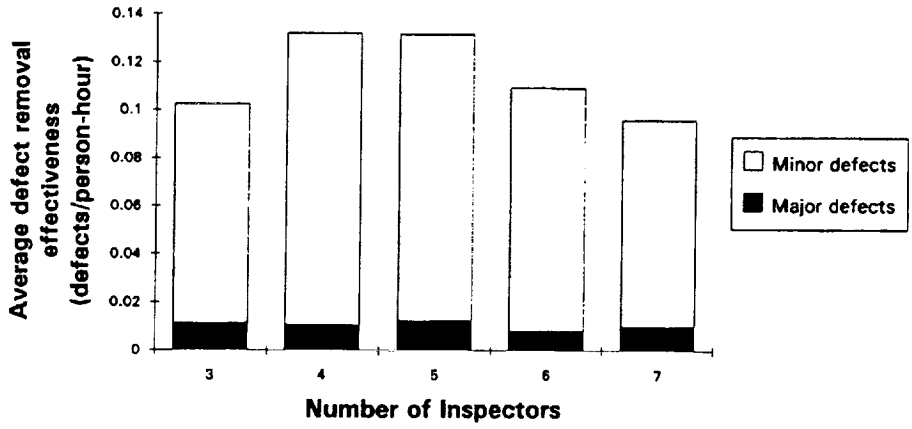
Litton
Data Systems

Effect of Inspection Rate on Defects Found



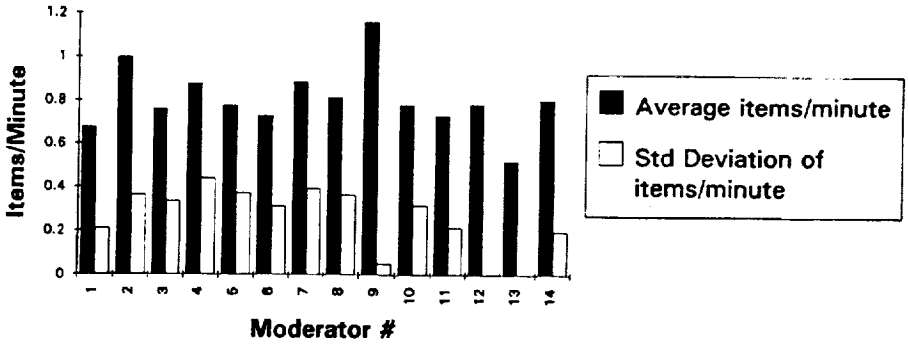
Litton
Data Systems

Defect Removal Effectiveness vs. Number of Inspectors



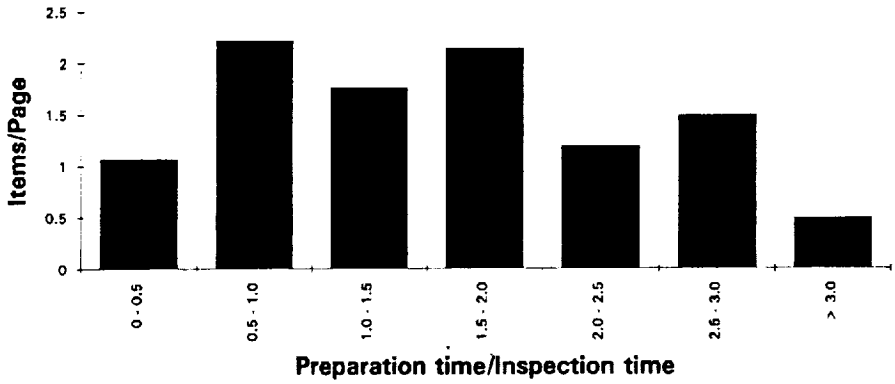
Litton
Data Systems

Moderator Finding Efficiency



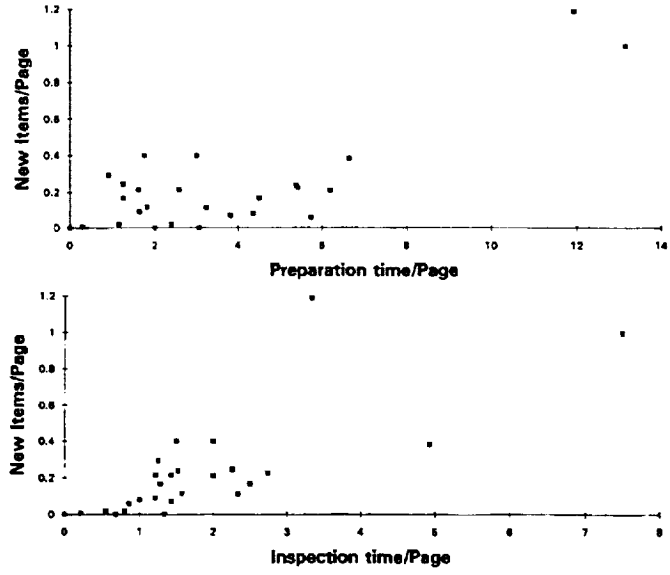
Litton
Data Systems

Effect of Preparation/Inspection Time Ratio on Defects Found



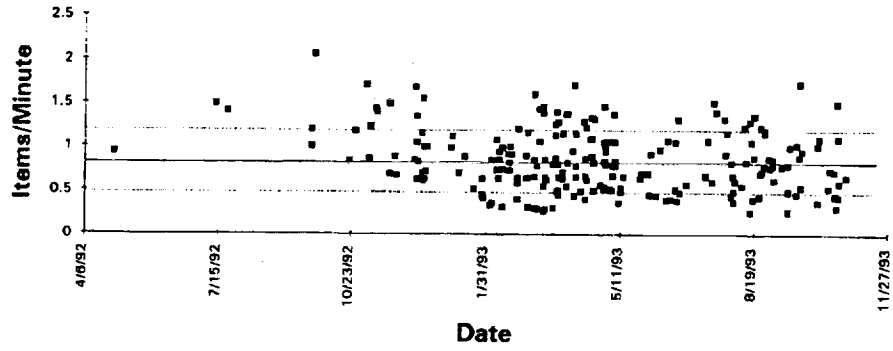
Litton
Data Systems

Effect of Preparation and Inspection Time on New Items Found



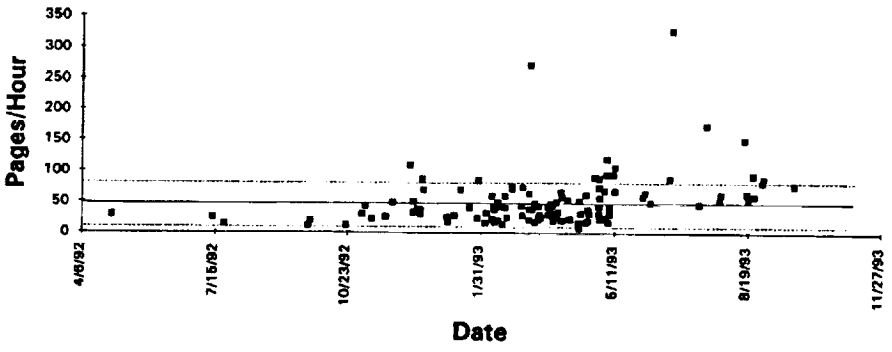
Litton
Data Systems

Defect Finding Rate Control Chart



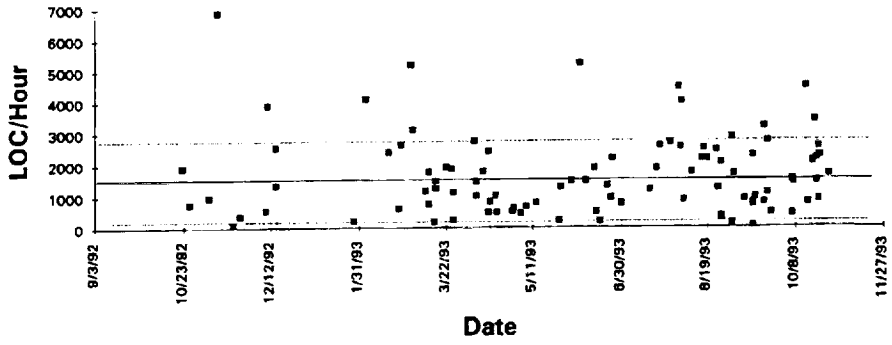
Litton
Data Systems

Document Inspection Rate Control Chart



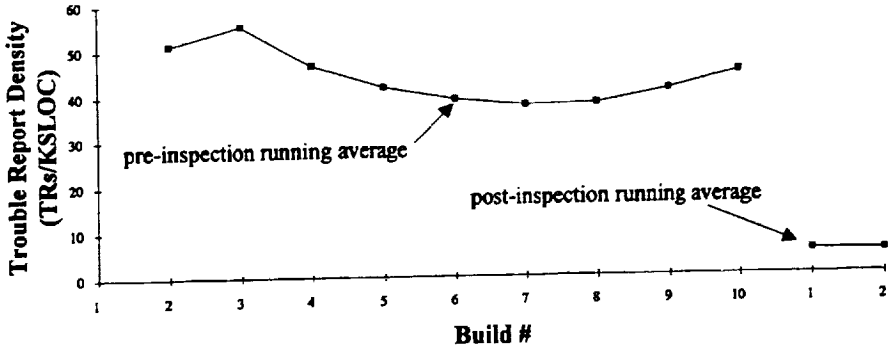
Litton
Data Systems

Code Inspection Rate Control Chart



Litton
Data Systems

Resulting Defect Density During Integration



Litton
Data Systems

Conclusions and Future Work

- Inspections are a worthwhile investment.
- Peer review coordinator essential to keeping process under control.
- Strong correlation between pre-inspection effort and new items found.
- Inspections appear to affect downstream artifacts and eventual system integration.
- Inspectors and authors have improved since inspections began.
- Some stabilization observed during first year of practice.
- Improved teaching method and changed process based on statistical results.
- Inspection analysis has provided impetus for improved metrics tracking procedures.
- Further analysis desired.
  - understand and account for confounding factors
  - defect category metrics and causal analysis
  - process control and optimization
  - ANOVA, other

References

Fagan ME, *Design and code inspections to reduce errors in program development,* IBM Systems Journal, V. 15, no. 3, 1976, pp. 182-210

Gilb T, Principles of Software Engineering Management, Addison-Wesley, Reading MA, 1988, pp. 205-226, 403-422

Grady R, Caswell D, *Practical Software Metrics for Project Management and Process Improvement* Prentice-Hall, Englewood Cliffs, NJ, 1992

Remus H, *Integrated software validation in the view of inspections /reviews,* In "Software Validation, Inspections-Testing-Verification-Alternatives" (H. Hausen, ed.), New York, NY, Elsevier Science Publishers, 1984, pp. 57-64

Weller EF, *Three years worth of inspection data,* IEEE Software, September 1993, pp. 38-45