Defect Prevention Techniques for High Quality and Reduced Cycle Time

An ESSI Process Improvement Experiment (PIE)

Abraham Peled
Leeba Salzman
Abraham Danon
Paul Rogoway

Motorola Communications Israel Ltd.
Tel Aviv, Israel

Background

Motorola Communications Israel Ltd. (MCIL) is an industrial company which develops and manufactures radio equipment and radio related products. MCIL's Development Division is one of Motorola's largest development group outside of the United States, and is involved in many main-line development projects of Motorola's Communication Enterprise (CE).

Motorola's products and systems are becoming software intensive at a remarkable rate. In the last five years, the amount of software in MCIL's products has increased dramatically and the number of software engineers has more than doubled. More and more engineering assets are becoming software intensive, and the company estimates that in the coming years, the software content in its products will grow to 90%, and software will become increasingly vital to its competitiveness and success.

In order to meet the increased market demands in functionality of new products, MCIL needs to constantly apply more engineering resources in the development process.

The company has invested heavily in quality, and is eager and committed to preserve this important asset. With new, sophisticated and complex products and systems, it is becoming increasingly difficult to maintain higher levels of quality while reducing the development cycle time, to "rush" to market.

Project Description

The “Defect Prevention Techniques” Process Improvement Experiment (PIE) was established to investigate patterns of defects that commonly occur in the Software Development Process. The Goal of the PIE was to define and implement techniques to reduce the total number of defects in the Development Life-cycle and to prevent certain classes of defects from recurring. [1]

It is well known that correcting defects in later stages of the Development Life-cycle is more complicated, expensive and time-consuming than correcting them in earlier stages. Preferable even to early detection, is the avoidance of defects altogether - the Defect Prevention method. Defect Prevention is therefore the best way to optimize the development process costs and to shorten the development cycle time. [2]

The objective of the experiment was to define and implement Defect Prevention methods and techniques, for the various phases of the Development Life-cycle, to reduce the quantity of defects and then to determine a Strategy for decreasing the Testing effort needed for development projects.

The PIE has produced a better understanding of common defect types, suggested and implemented solutions to avoid them, and established a mechanism to investigate new / remaining defects with the goal of eliminating them.

Starting Scenario

MCIL is a well-established Level 3 organization, with a phased development process, institutionalized formal reviews, automated defect tracking, and a long history of pioneering state-of-the-art software tools and technologies.

In order to initiate the PIE, we selected a project from our Digital Radio development group as the baseline project. The project selected is TETRA - Trans European Trunked RAdio - a new all-European digital radio system which integrates, in a single subscriber unit, Cellular and Dispatch (two-way) communication, as well as short paging messages.

The project has more than one release, and we used one for determining the reference line and a consecutive one to measure the improvements effected by the PIE. Selecting a multi-generation project, with two releases, rather than two distinct projects, helped neutralize the effects of several important variables, which could have otherwise distorted the findings. Thus we ensured commonality with respect to development environment, application domain, work culture, working methods, engineering team and tools.

Plans and Expected Outcome

The PIE plan was comprised of several steps:

1. Create a reference line of root causes based on defects recorded in the initial project release.

2. Based on the profile of common defects, define techniques to prevent specific problems in each phase of development.

3. Implement and disseminate techniques to engineering team at phase kickoffs.

4. During subsequent project development, support modified software development process (see Figure 1) and ongoing causal analysis of new defects detected.

5. Review root causes, analyze changes in defect trends. Evaluate efficacy of new techniques. Determine strategy to reduce test effort while focusing on the most error-prone areas.

6. Disseminate findings and recommend changes to the OSSP (Organizational Standard Software Process).

The existing CMM Level 3 development process was modified to include Defect Prevention activities. A first-stage causal analysis was added to the defect closing procedure, whereby the engineer handling the error/defect would fill out a new Analysis form. This Analysis form which includes Beizer Taxonomy classification, cause category, root cause analysis, containment method and suggestion, is physically attached to the problem report and remains part of the database.

A second-stage causal analysis was added to verify the correctness of the first stage and identify trends and extreme cases which require attention. Phase kickoffs were added to the process to educate the engineers on the common errors of the phase and on causal analysis techniques.

Based on the trends identified in the second stage of analysis, defect prevention techniques and strategies were recommended and implemented. This activity was performed by the PIE team and managed as a separate process.

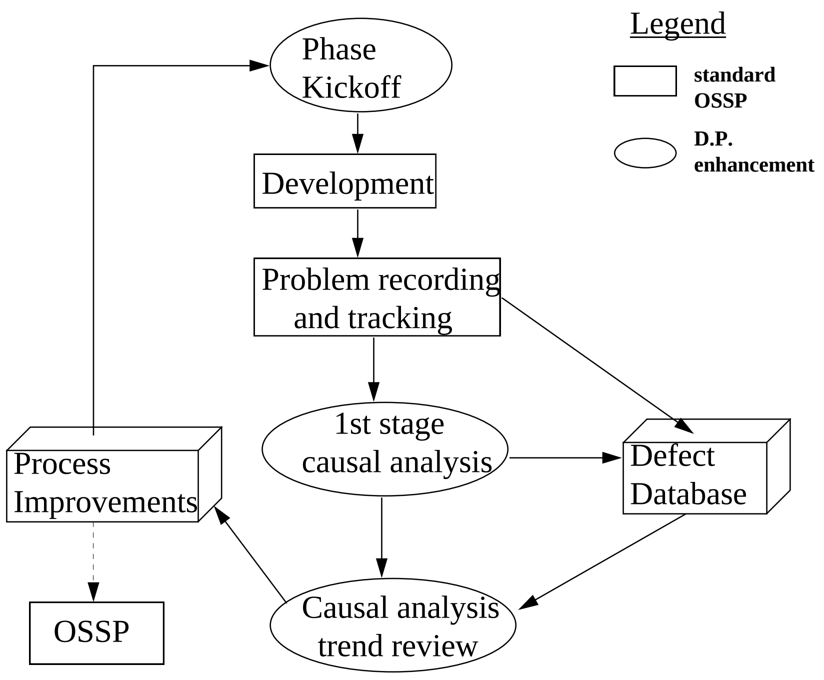


Figure 1 - CMM Level 3 Software Process enhanced for Defect Prevention

The Reference Line: The first step was to perform “Defect Analysis of Past Projects” in order to create a reference line for the PIE. We analysed 1336 defects from the baseline project (TETRA Release 1) and two other projects (to increase the statistical significance). Detailed Root Cause Analysis was performed on all the defects, and the Beizer [3] Taxonomy was used as the “classification vehicle”. Analysis was done for five of the development phases, namely : Requirement Specifications, Architectural Design, Detailed Design, Coding and System Test Case Preparation. Based on this analysis, specific Defect Prevention solutions were determined for each of these phases.

The Beizer Taxonomy used for the classification includes ten major categories, each of which is divided into three levels, resulting in a 4-digit number which specifies unique defects. The ten top level categories are :

0xxx	Planning
1xxx	Requirements and Features
2xxx	Functionality as Implemented
3xxx	Structural Bugs
4xxx	Data
5xxx	Implementation
6xxx	Integration
7xxx	Real-Time and Operating System
8xxx	Test Definition or Execution Bugs
9xxx	Other

The causes of the defects as determined by the engineers doing the classification, fall into four major categories:

* Communication
* Education
* Oversight
* Transcription

In creating the reference line, detailed interviews with 24 software engineers took place, in order to fully understand the reason for each defect, to classify the cause and to understand how the defect could have been prevented. This “data mining” was performed on all the defects, resulting in a series of “classification tables” and a good Pareto analysis of the most common problems.

The following Pareto represents the breakdown (in descending order) of the defect analysis according to the Beizer Taxonomy top level categories :

Requirements and Features	(1xxx)	47.0%
Functionality as Implemented	(2xxx)	13.5%
Structural Bugs	(3xxx)	9.3%
Implementation	(5xxx)	8.3%
Data	(4xxx)	6.9%
Integration	(6xxx)	5.7%
Real time and Operating system	(7xxx)	4.9%
Test definition or Execution bug	(8xxx)	4.3%

Within each development phase in the baseline project, we further classified the defects, based on the Beizer Taxonomy. For example, in the Requirement Specifications Phase, the second level breakdown of the main defects was as follows:

Requirement Completeness	(13xx)	37.5%
Requirement Presentation	(15xx)	34.7%
Requirement Changes	(16xx)	11.2%
Requirement Incorrect	(11xx)	8.7%

The third level breakdown of the main "Requirement Completeness" defects was :

Incomplete Requirements	(131x)	73.4%
Missing, unspecified requirements	(132x)	11.2%
Overly generalised requirements	(134x)	4.6%

The same type of data analysis was performed for each of the development phases selected for the PIE.

The next step was to identify a tool-set of phase-specific improvement activities, based on the root cause analysis, that would prevent the defects from recurring in the next release. Highest priority was given to the most common defect types.

Extensive training and phase kickoff meetings were held to empower the development team to integrate Defect Prevention into the existing process. The improvement activities determined in the analysis phase were then applied by the development team in the different development phases, and ongoing defect recording and measurements were performed.

The final step was to compare the numbers and types of TETRA Release 2 defects with those of the reference line. The effectiveness of the "prevention tool-set" was measured in the quantity and types of defects found in the second release of the project. The prevention actions which were found to be effective could then be integrated into the OSSP to improve quality and cycle time for all the projects in MCIL. The impact on the OSSP, including changes to Review Guidelines and changes to the Phase Kickoffs, are considered to be part of the PIE results.

The Expected Outcomes:

1. A framework for establishing a Defect Prevention program in a software development environment

2. A list of improvement actions to be taken by the TETRA project development group in order to prevent defects, including :

   * Method to number the Requirements in the SRS document
   * A Writing Strategy procedure to reduce the ambiguities in the Requirement Specification phase
   * A utility to support/implement the Writing Strategy
   * Improved Software Requirement Specifications (SRS) template
   * Formalised Context Diagram / Feature Interface Chart for the Requirement and Design phases

* Improved Review Checklists for all phases of the development life-cycle
* Causal Analysis procedures and meeting guidelines
* Improved Kickoff meeting templates and guidelines, for all phases of the development process
* Testing Strategy

3. Improved quality of the Tetra product, including :

   * Decrease in the overall number of defects found in the various development phases
   * Shift in the distribution of defects, by phases
   * Lower development costs
   * Shorter cycle time

Implementation of the Improvement Actions

Kickoff meetings were held for each phase, where the importance of Defect Prevention and causal analysis were explained and emphasised. The improvement actions for the specific phase were presented and discussed. The actions, as suggested by the PIE team, were generally well received by the TETRA development engineers and managers. Techniques such as improved review checklists were applied immediately after the kickoff at formal peer reviews.

In each progressive phase, engineers became more adept at recording the defects using DDTs® - Distributed Defect Tracking System, and at performing causal analysis. They became more open minded about reporting and recording their own defects, understanding the importance of a systematic tracking approach to the quality of the product and the process.

Many TETRA engineers expressed satisfaction with the causal analysis process and kickoff meetings, which made them feel better equipped to prevent defects, and improved their general attitude towards the software process.

The PIE is considered by the technical staff as well as the business staff, to be a positive process, which gives us an advantage in better quality of the products, and reduced cycle time of the development process. As such, the TETRA development group has adopted several changes to its processes, to accommodate the Defect Prevention environment.

Internal dissemination outside of the TETRA development group, has yet to be done and will begin with the presentation of the Defect Prevention method to the SEPG - Software Engineering Process Group, the owner of MCIL's OSSP. This group will analyse the results of the PIE project, and update the OSSP accordingly. The SEPG will also be responsible for deploying the new process and training the other development groups. This will be done through a series of technical meetings with engineers and managers, dealing with Defect Prevention, the PIE and the updated OSSP.

Measured Results

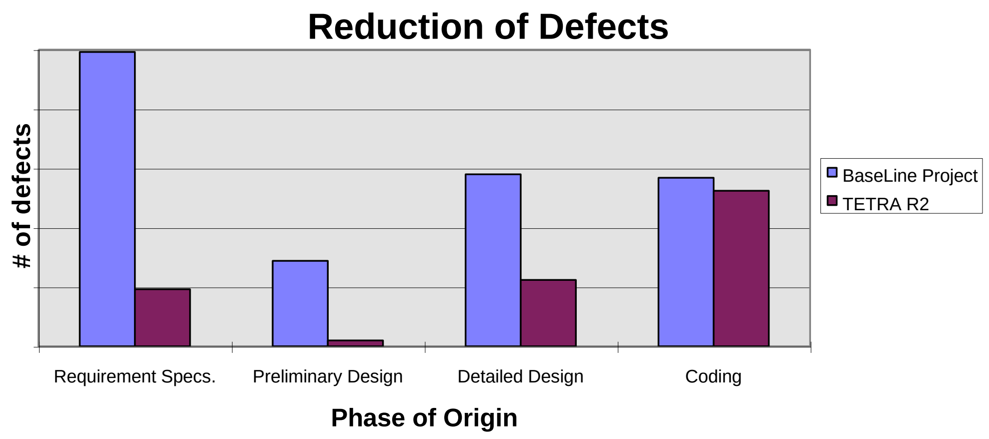
The overall number of defects in Tetra Release 2 has **decreased by 60%,** in comparison to the number of defects detected in TETRA Release 1 (the reference line project). In part, this can be attributed to the fact that Release 2 is a continuation project and not an initial project as Release 1, and that later releases usually have less defects due to more cohesive teams, greater familiarity with the application domain, experience, and fewer undefined issues.

Based on numbers from other MCIL projects, we estimate that half of the defect decrease (30%) can be attributed to the implementation of the PIE.

A breakdown of the defects, by Phase of Origin, shows the following results :

	TETRA Release 2	**Past Projects**	**% Improvement**
Phase of Origin			
Requirement Spec.	**20%**	**40.8%**	**80.6%**
Preliminary Design	**2.5%**	**11.8%**	**93%**
Detailed Design	**23%**	**23.9%**	**61.4%**
Coding	**54.5%**	**23.4%**	**8%**
	100%	**100%**	**60%**

The absolute reduction in defects, which relates to the % improvement shown in the above table, can be observed in the following chart :



The obvious observation is that a higher percentage of the defects "migrated" to later phases of the development process : from Requirement Specifications, Preliminary Design and Detailed Design, to Coding. In Tetra Release 1, 76.5% of the defects are in the Requirement and Design phases and only

23.4% are in Coding, while in Tetra Release 2, only 45.5% of defects are in Requirement and Design and 54.5% are in Coding. This implies that the defect prevention methods employed in the early phases of development were very effective.

The % Improvement column, shows the improvement within each development phase, with respect to the absolute number of defects. This is a different view of the improvement in the number of defects, partially attributable to the Improvement Actions.

Another comparison was made in respect to the Cause category. Following are the results:

Cause category	**TETRA Release 2**	**Past Projects**
Communication	**10%**	**11%**
Education	**13%**	**13%**
Oversight	**74%**	**74%**
Transcription	**3%**	**2%**

The obvious observation here is that the differences are **not significant**. The largest bulk of the defects are caused by human errors.

Lessons Learned

There are several key lessons learned from this PIE project:

1. Although Defect Prevention is considered an SEI/CMM Level-5 KPA, we found that a strong Level-3 organization, with a Defect Prevention infrastructure, can build an effective Defect Prevention Process, and obtain excellent results.

2. The primary cause of defects as classified by the development team is oversight, or human error (almost **75%**). Our experience shows that the term “oversight” is too broad and should be broken down somewhat, probably based on the Beizer classifications of those defects which were categorised as “oversight”.

3. The Timing of the Phase Kickoff meetings is critical. A Phase Kickoff should be planned early and performed as close as possible to the beginning of the phase.

4. In order for the Defect Prevention process to be effective, the software teams need in-depth training and initial support in using the taxonomy and performing the root cause analysis.

5. A tool to input the classification of defects, according to the Beizer Taxonomy is essential. An automatic tool is needed to analyze the defects and to get statistical results. The current vehicle we have for input of cause analysis and defect classification is deficient. A better interface is needed, as well as a mechanism for adding new categories to the

Beizer Taxonomy. Standardized statistical analysis reports are needed for use by all projects for ongoing Defect Prevention and process improvement.

References

[1] R.G. Mays, C.L. Jones, G.J. Holloway, D.P. Studinski, “Experiences with Defect Prevention”, IBM Systems Journal, Vol 29, No. 1, 1990

[2] Watts S. Humphrey, “Managing the Software Process”, Chapter 17 - Defect Prevention, ISBN-0-201-18095-2

[3] Beizer Boris., “Software Testing Techniques”, Second edition, 1990, ISBN-0-442-20672-0

Appendix A - Authors' CV

Paul Rogoway
Director of Software Quality Standards, Motorola

Motorola Experience (16 Years)
Coordinates all Motorola activities relating to software quality standards, including participation in national and international software standards working groups; identification, development and promotion of internal Motorola standards; and dissemination of information to Motorolans concerning risks and opportunities related to software standards.

Serves on Corporate Software Engineering Technology Steering Committee and on various international committees and working groups in areas such as software life cycle models, capability assessment, technology and tools, and process improvement. Israel's delegate to several international software standards organizations, including those responsible for SPICE and ISO 9000-3. Chairperson of the Israel Software Process Improvement Network (I-SPIN).

Member of Motorola's Science Advisory Board Associates (SABA). Winner of Motorola Outstanding Impact Award for contribution to ISO 9000-3 revision project. Authorized Lead Assessor in the SEI Appraiser Program.

Previously established and managed two software development groups, headed a Motorola Senior Executive Program team which produced a methodology for software engineering technology planning, founded and managed "4S" to help Motorola and non-Motorola organizations accelerate their improvement in software development capability, served as process improvement coach/consultant to several Motorola software development organizations and to leading non-Motorola software development organizations in Israel, and was responsible for the first phase of the CASE* project to define an Integrated Project Support Environment for Motorola worldwide.

Other Experience :
Adjunct Professor of Computer Science, Bar-Ilan University, Israel.
Before joining Motorola, Prof. Rogoway held senior technical and management positions at major U.S. and Israel companies, including TRW, Informatics, IBM, Elbit and Tadiran.

Publications :
More than 40 papers, articles, and lectures on various software engineering topics

Abraham Peled
Software Process Improvement Manager, MCIL

Motorola experience (12 years) :
Currently, manager of Software Process Improvement activities in MCIL's Development Division, which recently achieved Level 3 in a SEI CMM assessment. The chair-person of two internal Software Process related committees : The Software Engineering Steering Committee and the Software Engineering Process Group (SEPG). Responsible for and coordinating the activities of the various Process Improvement Groups.

A member of the Software Quality Committee (SQC), a corporate level committee, dealing with software quality issues, with Software Quality Metrics, and with deployment of the Quality System Review (QSR) within Motorola facilities, worldwide.

Previously, established and managed for 9 years the Systems & Software Quality Assurance, MCIL's Box and System Testing group, who is responsible for testing and releasing all the Software/Firmware products of MCIL. Most of the tests performed, were "shifted" from manual testing to the automatic testing environment, developed internally over the years. Also, Manager of the PQE group - who is responsible for Hardware and Environmental testing of all MCIL's released products. Responsible for the planning, installation and maintenance of MCIL's conventional SmartZone System, and for the new Digital Tetra/Dimetra System.

Leeba Salzman
Software Quality Manager, MCIL

Motorola experience (12 years)
Currently Software Quality Manager in the Digital Radio Group, responsible for process adherence, new employee training, and process improvement deployment. Member of the Software Engineering Process Group (SEPG). Responsible for MCIL training on Software Peer Reviews, Problem Tracking and DDTs. Head of MCIL DDTs Empowerment Team.

Abraham Danon
Systems & Software Quality Assurance Manager, MCIL

Motorola experience (16 years)
Currently, manager of MCIL's Software Box/System testing group. Member of MCIL's Software Engineering Steering Committee and editor of MCIL's publication on Software Engineering : "Touch Of Quality".

During years with Motorola, carried on technical and managerial software tasks including responsibility for software process, tools and technology.

Appendix B - Company Profile

Motorola Communications Israel, Ltd. (MCIL) is an industrial company which develops and manufactures radio equipment and radio related products. MCIL's Development Division is one of Motorola's largest development group outside of the United States, and is involved in many main-line development projects of Motorola's Communication Enterprise (CE).

MCIL is a fully-owned subsidiary of Motorola Inc., the world-wide leader in wireless communication. Motorola develops and manufactures components, products and systems in the following domains : Semiconductors, Cellular Systems, Paging Systems, Two-way Radio Communication, Modems and Integrated Management Systems, Automotive Electronics, Government and Space Systems (Irridium), and Multimedia.

According to the latest "Fortune 500" list, Motorola is the 29th largest company in the U.S., with revenues of $30B in 1997, and 150,000 employees worldwide.