Learning from Our Mistakes with Defect Causal Analysis

April 2001

David N. Card

Based on the article in *IEEE Software*, January 1998

Agenda

- What is Defect Causal Analysis?
- Defect Prevention Key Process Area
- Defect Causal Analysis Procedure
- Action Team Activities
- Summary and Conclusions

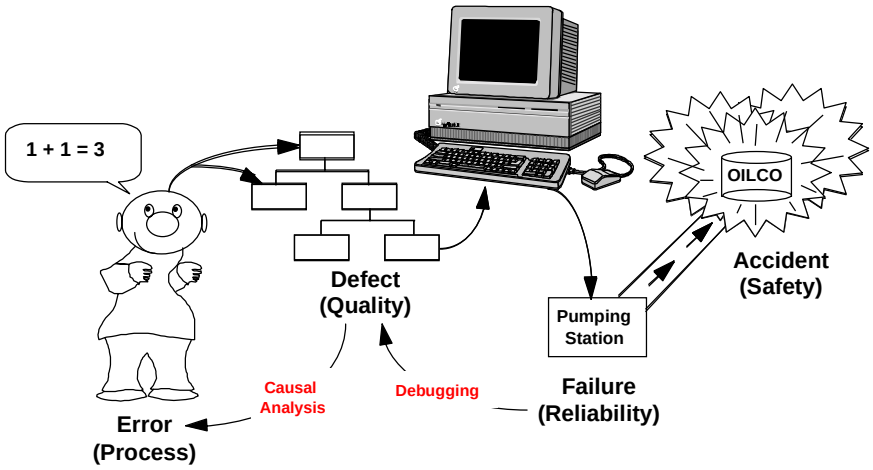
What is DCA?

- Examination of information about problems
- Intent to identify causes of defects so that they can be prevented or detected earlier
- Many different approaches called defect causal analysis or root cause analysis – employ many different techniques
- Performed in response to an “out of control” situation or as part of a continual improvement program

Definitions

- Error - a mistake made by a member of the software team
- Defect - a section of code or documentation that must be changed to correct a failure
- Failure - a situation in which the software fails to execute as intended
- Problem Report - usually documentation that a failure has occurred during testing or use. May also be used to document defects found in inspections and reviews.

The Defect Causal Chain



SOFTWARE PRODUCTIVITY CONSORTIUM

Concept of Causality

- Conditions of causality
  - "Cause" must precede the "effect" in time
  - Mechanism by which the cause produces the effect must be understood
- Assignment of cause in a "human-intensive process" always includes a significant element of subjectivity

SOFTWARE PRODUCTIVITY CONSORTIUM

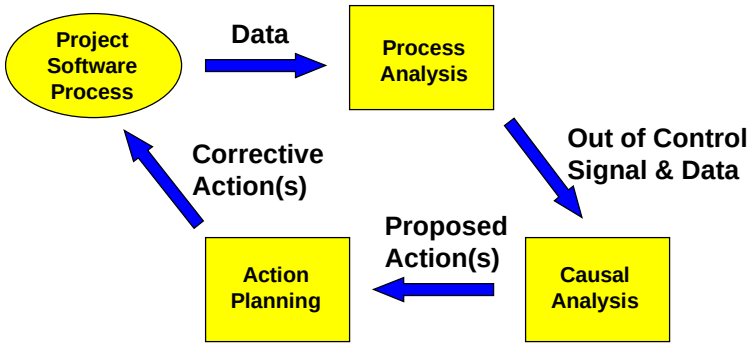
Relationship to CMM

- Level 4
  - May be ad-hoc
  - Performed in response to “out-of control” situations
- Level 5
  - Component of Defect Prevention KPA
  - Systematic approach required – “in accordance with a documented procedure
  - Performed even when process is “in control”

Causal Analysis for Control

- Robust causal analysis process is not required for Level 4, but it can give you a head-start on Level 5
- Causal analysis indicated when “out of control” situations arise
- Use all the data associated with the “out of control” situation as input to the causal analysis
- Control charts may track subgroups of any size for any type of measure
- Causal analysis resulting from monitoring measures of defect data requires same techniques as for continuous improvement

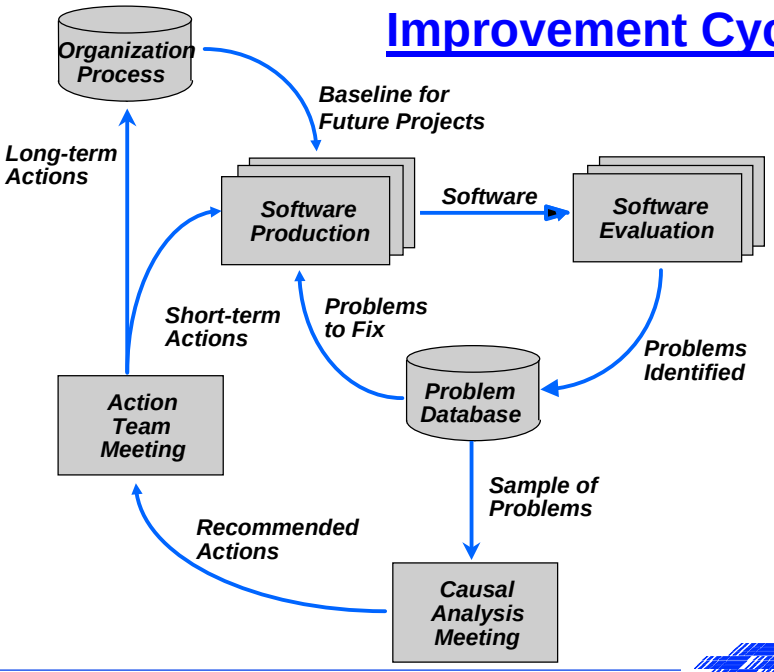
Corrective Action Cycle



Causal Analysis for Improvement

- May be organized within a Defect Prevention context
- Assigns responsibility for causal analysis of a process to the software team
- Bases analysis on a sample of problems rather than an exhaustive study of all problems
- The software team proposes actions to:
  - prevent problems
  - find problems earlier
- Assigns responsibility for implementing proposals to a management action team

Improvement Cycle



SOFTWARE PRODUCTIVITY CONSORTIUM

Defect Prevention KPA

SOFTWARE PRODUCTIVITY CONSORTIUM

Defect Prevention Description

Purpose

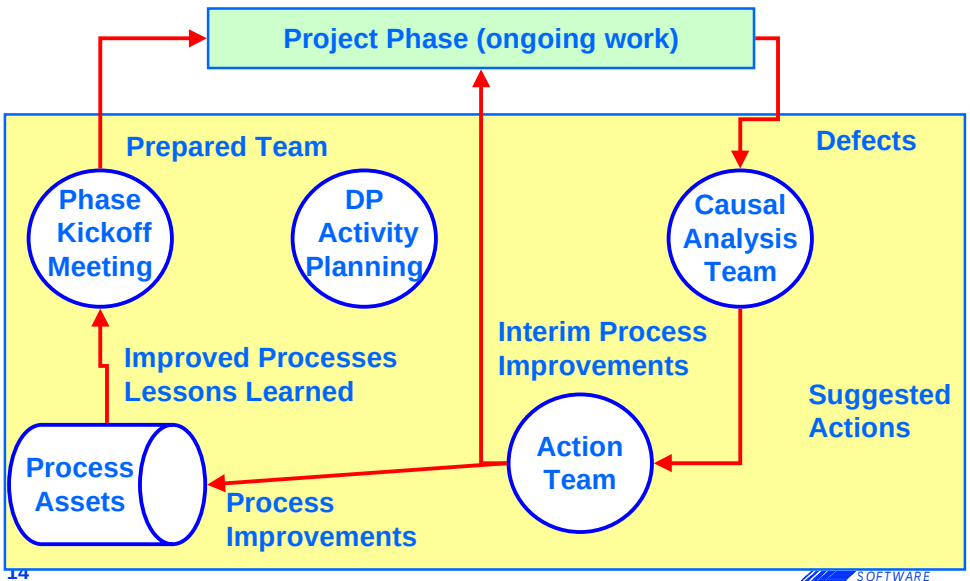
- To identify the cause of defects and prevent them from recurring

KPA goals

- Defect prevention activities are planned
- Common causes of defects are sought out and identified
- Common causes of defects are prioritized and systematically eliminated

Source: ***Key Practices of the Capability Maturity Model, Version 1.1*****, SEI, CMU/SEI-93-TR-25.**

Project DP Process in the CMM



DP Planning

- Defines focus, composition, roles, and responsibilities of defect causal analysis team(s)
- Defines charter, composition, roles, and responsibility of action team(s)
- Based on results of process performance analysis provided by QPM, SQM, PCM activities

Defect Causal Analysis Procedure

Causal Analysis Meeting

- Focus of DCA process
- Held at regular intervals for continuous improvement or when an out of control situation arises
- Involves the entire development or maintenance team - or other group contributing to the “out of control situation”
- Designated moderator (facilitator)
- Managers not present
- Open and constructive, not defensive

DCA Phases

- Meeting Preparation
- Causal Analysis
- Corrective Action Development

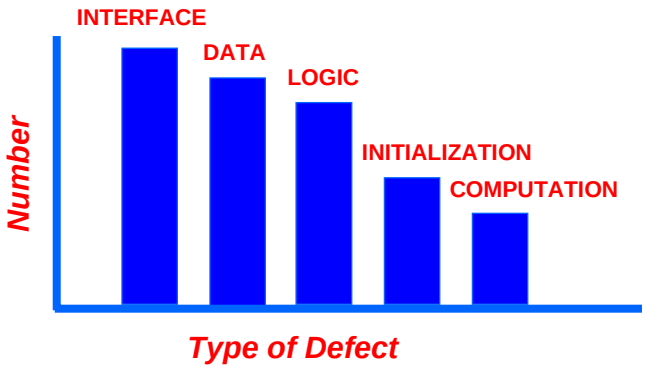
Problem Sample

- Need to reduce the input to a manageable volume, especially for continuous improvement
- Selection and Classification may be done in advance by Moderator
- Select no more than 20 problems for analysis in one session
- Omit obvious duplicates and non-software problems
- Do not select only “high priority” problems
- Do not select problems from just one source (individual or component)

Problem Classification

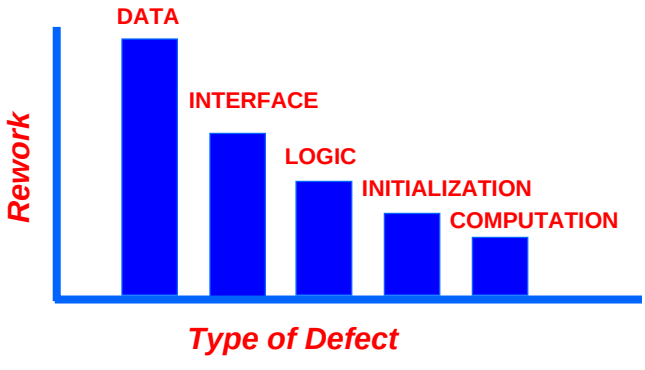
- Problems may be classified by the programmer when analyzing or implementing fixes
- Use local standard classifications:
  - when inserted (activity)
  - when found (activity)
  - type of error made
- Develop Pareto Diagrams or counts for each category
- Revise the classifications as indicated by experience

Pareto Example (Quality)



SOFTWARE PRODUCTIVITY CONSORTIUM

Pareto Example (Cost)



SOFTWARE PRODUCTIVITY CONSORTIUM

Systematic Errors

- "Random" mistakes are expected - focus attention on the least random
- Characteristics of Systematic Errors
  - Same or similar defect repeated
  - Many defects from the same activity
  - Many defects of the same type
  - Few defects captured by an activity
- Look at defects that fall into both the peak source and peak type categories
- Develop problem statements for the Systematic Error

Example of Systematic Error

- Problem Reports from Integration Testing:
  - "unable to locate file"
  - "access not authorized"
  - "device not found"
- Systematic Error – "variations in use of computing environment results in incompatible software components"

Cause Identification

- Ignore the effect of the problem in assigning cause
- Consider
  - classification information
  - symptoms
  - special circumstances
  - departures from usual practice
- Many factors usually contribute - look for the primary cause
- Develop Cause-Effect Diagram if the primary cause is not obvious

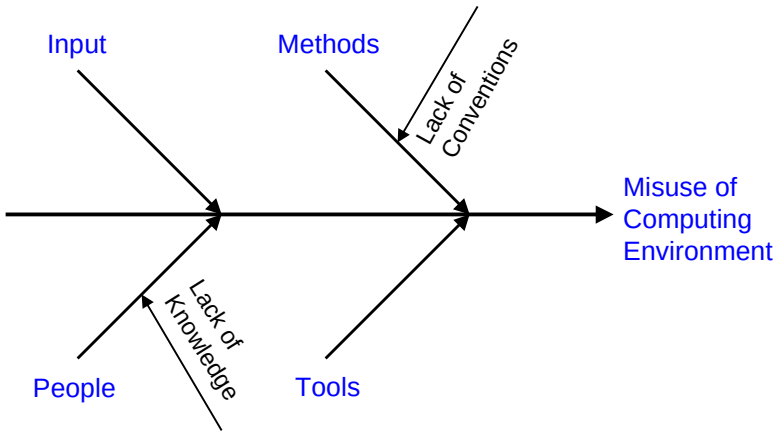
Cause-Effect Diagram

- Simple graphical technique
- Helps to sort and relate many factors
- Developed as a team (facilitated)
- Focus for discussion - not a definitive result
- Also called an Ishikawa or Fishbone Diagram

Diagramming Steps

- State problem (effect) - Use statement of Systematic Error - Draw main branch
- Insert headings for generic causes
  - methods
  - people
  - tools/environment
  - input
- Brainstorm specific causes - attach to appropriate generic causes
- Highlight principal/operative causes(s) - circle

Cause-Effect Example



Action Proposals (1)

- Must address Systematic Errors
- Focus on high payoff actions
- Consider
  - How could we have done things differently?
  - What information did we need to avoid this?
  - How could we detect the problem earlier?
- Actions must be specific/concrete
- Limit actions to four per Systematic Error - one good action proposal is enough!

Action Proposals (2)

- Examples of Actions
  - update common error lists used in reviews
  - provide training in a specific skill
  - regularly disseminate key information
- Avoid general terms (e.g., better, more, as needed, available, enough)
- List specific characteristics of suggested action (e.g., stimulus, frequency, scope, responsibility)
- Focus on you own process - only address the interfaces to other processes

Meeting Documentation

- Records are necessary to ensure that actions get implemented
- Identify
  - meeting event (date, etc.)
  - "out of control" situation (if applicable)
  - systematic error (if identified)
  - problem reports related to systematic error
  - proposed actions
- Problems are the justification for action

Action Team Activity

Action Team Organization

- Meets regularly to consider proposed actions
- Must include management - needs resources
- May include technical personnel - usually DCA moderators
- Multiple DCA teams often feed into one Action Team
- Benefits of DCA are lost without timely action

Action Team Role

- Select and prioritize proposals
- Resolve conflicts and combine related proposals
- Plan and schedule implementation
- Allocate resources and assign responsibility
- Monitor progress and effectiveness
- Communicate actions and status to the teams

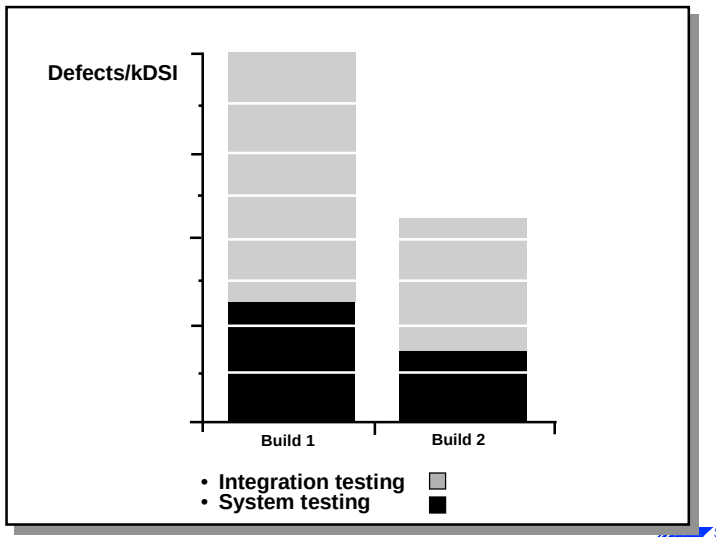
Example of DCA Process

- Problem - inconsistent use of environment by developers resulted in many errors during integration
- Proposal - define operational environment (e.g., directory structures, devices, protections) as early as possible and perform developer testing in this environment
- Results - integration time for subsequent builds was reduced 50%

35

SOFTWARE PRODUCTIVITY CONSORTIUM

Comparison of Defect Rates for Builds



36

SOFTWARE PRODUCTIVITY CONSORTIUM

Sources of Systematic Errors

- Methods: 65%
  - failure to follow defined process
  - failure to communicate information
- People: 15%
- Input: 12%
- Tools: 8%

Key Points

- Don’t study all problems reported - sampling will find systematic errors
- Look beyond the symptoms to the underlying causes
- Do not create an action for each problem - get leverage by attacking the systematic errors
- Focus on fixing the team’s process, not someone else’s
- Benefits take time to realize
- Facilitator training is helpful for moderators
- Action team must follow through

Summary and Conclusions

Maturity-Pull Effect

- DCA is a high-leverage activity
- Relationship to SEI CMM
  - CMM is descriptive, not prescriptive
  - Level 1 organizations usually implement training and SEPGs (Level 3 KPAs) to get to Level 2
  - DCA helps organizations establish themselves at Level 3
  - DCA does not fully satisfy the Defect Prevention KPA of Level 5
- DCA shows the value of an effective defined process
- DCA is of limited value in an ad-hoc process

Summary of DCA Experience

- Easy to implement - common sense approach
- Low cost (about 1.5% of software budget - including implementation of actions)
- Increased awareness of quality, process, and measurement
- Tangibly improved product quality
- Personnel reacted favorably
- Large dollar savings for IBM and Lucent; increased customer satisfaction for CSC

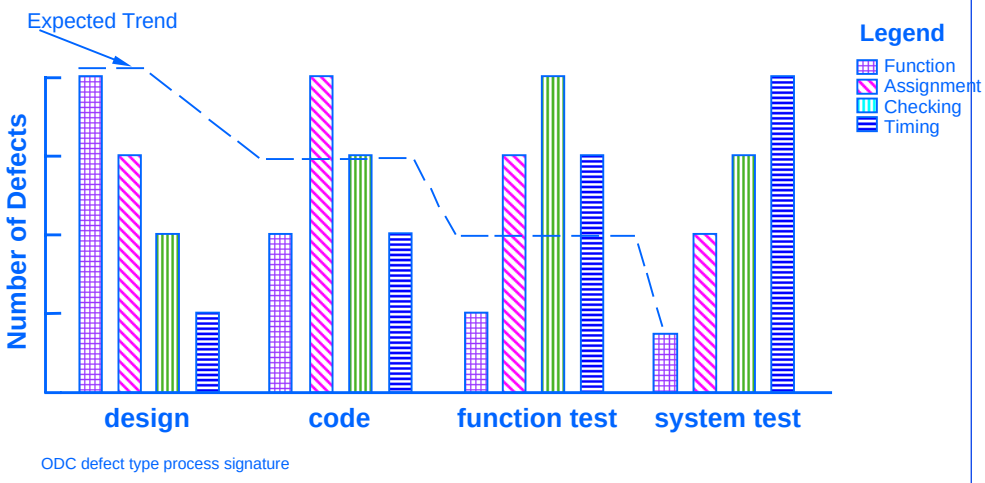
Relationship to Six Sigma

- Additional causal analysis techniques provided in most Six Sigma training programs (e.g, Error Modes and Effects Analysis)
- Defect prevention strategy and team-based approach to DCA usually are not explicit elements of Six Sigma
- CMM approach assumes processes are defined, the need to define processes as part DCA increases the time and effort required

Orthogonal Defect Classification

- Assumption: Distribution of defect types within each phase remains stable while process is stable.
- Data from past projects/builds establishes defect profile.
- More or less defects than expected of any type indicates problem areas.
- Chi-square test can be performed to test significance of difference between current results and expected results.

Example of ODC Trend



ODC defect type process signature

Source: Bridge, N., and C. Miller. "Orthogonal Defect Classification Using Defect Data." *ASQ Software Quality Newsletter*, March 1998.

Summary

- Most organizations with well-defined processes can benefit from some application of DCA
- Maximum benefit obtained from
  - Following a systematic approach
  - Involving the developers/maintainers
  - Pursuing a strategy derived from an objective understanding of improvement opportunities
- DCA can be applied to any process that receives feedback on its defects or failures

References

- Card, D. “Statistical Process Control for Software?” *IEEE Software*, May 1994.
- Chillargee, R., and I. Bhandari, et. al. “Orthogonal Defect Classification - A Concept for In-Process Measurements,” *IEEE Transactions on Software Engineering*, November 1992.
- Mays, R., et al., Experiences with Defect Prevention, *IBM Systems Journal*, January 1990
- Dangerfield, O., et al., Defect Causal Analysis - A Report from the Field, *ASQC International Conference on Software Quality*, October 1992
- Yu, W., A Software Fault Prevention Approach in Coding and Root Cause Analysis, *Bell Labs Technical Journal*, April 1998
- Card, D., Learning from Our Mistakes with Defect Causal Analysis, *IEEE Software*, January 1998