Defect Prevention and Process Impirovement

Outline

- Defect prevention
  - Basic concepts
  - Approaches
    - Error blocking
    - Error source removal
  - Techniques
- Process improvement
- Conclusion

QA Alternatives

- Defect and QA:
  - Defect: error/fault/failure.
  - Defect prevention/removal/containment.
  - Map to major QA activities
- Defect prevention (this lecture):
  - Error source removal & error blocking
- Defect removal: Inspection/testing/etc.
- Defect containment: Fault tolerance and failure containment (safety assurance).

Generic Ways for Defect Prevention

- Error blocking
  - Error: missing/incorrect actions
  - Direct intervention
  - Error blocked

    => fault injections prevented (or errors tolerated)
  - Rely on technology/tools/etc.
- Error source removal
  - Root cause analysis

    => identify error sources
  - Removal through education/training/etc.

Defect Prevention: Why and How?

- Major factors in favor of defect prevention:
  - Super-linear defect cost⇧ over time
    - ✓ early faults: chain-effect/propagation
    - ✓ difficulty to fix remote (early) faults
    - ✓ in-field problems: cost⇧ significantly
  - Other QA techniques for later phases
  - even inspection after defect injection
- Basis for defect prevention:

Causal and risk analysis

  - Analyze pervasive defects
  - Cause identification and fixing
  - Risk analysis to focus/zoom-in

Defect Cause and Actions

- Types of causal analyses:
  - Logical (root cause) analysis by expert for individual defects and defect groups
  - Statistical (risk) analysis for large data sets with multiple attributes
    - ✓ Model: predictor variables => defects
    - ✓ # defects: often as response variable
  - Cause(s) identified via either variation
- Actions for identified causes:
  - Remedial actions for current product
  - Preventive actions for future products:
    - ✓ negate causes or pre-conditions

Common Causes/Preventive Actions

- Education/training to correct human misconceptions as error sources:
  - Product domain knowledge,
  - Development methodology,
  - Development process, etc.
  - Act to remove error sources
  - Cause identification: mostly through root case analysis.
- Formal methods:
  - Formal specification: to eliminate imprecision in design/implementation (error source removal)
  - Formally verify fault absence.

Common Causes/Preventive Actions 2

- Technologies/tools/standards/etc.:
  - Based on empirical evidence
  - Proper selection and consistent usage or enforcement
  - More error blocking than error source removal
  - Cause identification: mostly statistical
- Process improvement:
  - Integration of many factors in processes
  - Based on empirical evidence or logic
  - Define/select/enforce
  - Helping both error blocking and error source removal
  - Cause identification: often implicit

Education and Training

- People: most important factor to quality
  - e.g., vs. impl. Languages
- Development methodology knowledge:
  - Solid CS and SE education
  - Methodology/process/tools/etc.
- Product/domain knowledge:
  - Industry/segment specific knowledge
  - Type of products: new vs. legacy etc.
    - ✓ legacy product: inter-operability
  - General product environment, etc.
- Means of delivery:
  - formal and informal education + on-the-job training.

Other Techniques

- Appropriate software technologies:
  - Formal methods
  - Cleanroom: formal verification + statistical testing
  - Other technologies: CBSE, COTS, etc.
- Appropriate standards/guidelines:
  - Mis-understanding/mis-communication#
  - Empirical evidence for effectiveness
  - Appropriate scope and formality
- Effective methodologies:
  - As package technologies/standards/tools/etc.
  - Empirical evidence
  - Match to the specific product domain

Tools for Error Blocking

- Programming language/environment tools:
  - Syntax-directed editor to match pairs.
  - Syntax checker/enforcer.
  - General tools for coding standards, etc.
- Other tools:
  - Design/code and version control
    - ✓ { examples: CMVC, CVS, etc.
  - Tools for indiv. development activities:
  - testing tools
  - requirement solicitation tools,
  - design automation tools, etc.
- General tools or tool suites for certain methodologies, e.g., Rational Rose.

Process Improvement

- Integration of individual pieces for defect prevention => process improvement
- Selecting appropriate development processes:
  - Process characteristics and capability
  - Match to specific product environment
  - Consideration of culture/experience/etc.
- Process definition and customization
  - Adapt to specific project environment
  - e.g., IBM's PPA from Waterfall
- Process enforcement and ISO/9000:
  - “say what you do”
  - “do what you say”
  - „show me"

Process Maturity for Improvement

- SEI/CMM: Focus on defect prevention
  - maturity level: focus/key practice area
    - ✓ 1. ad-hoc: competent people/heroics
    - ✓ 2. repeatable: project management proc.
    - ✓ 3. defined: engr-proc./org. support
    - ✓ 4. managed: prod./proc. quality
    - ✓ 5. optimized: continuous process improvement
  - expectation: maturity⇧ ➔ quality⇧
  - recently: CMMI for Development (CMMI-DEV, CMMI for Acquisition (CMMI-ACQ, CMMI for Services (CMMI-SVC.
- Other process maturity work
  - SPICE (Software Process Improvement and Capability dEtermination)
    - ✓ international effort
    - ✓ assessment, trial, and technology transfer
  - BOOTSTRAP - ESPRIT programme

TAME: Process/Quality Improvement

- QIP: Quality Improvement Paradigm
  - understand baseline
  - intro. process change and assess impact
  - package above for infusion
- GQM: goals/questions/metrics paradigm
  - goal-driven activities
  - questions related to goals
  - metrics to answer questions
- EF: experience factory
  - separation of concerns
  - EF separate from product organization
  - form a feedback/improvement loop

Summary

- Key advantages:
  - Significant savings if applicable:
    - ✓ avoid downstream problems
  - Direct affect important people factor
  - Promising tools, methodologies, etc.
  - Process improvement: long-lasting and wide-impact
- Key limitations:
  - Known causes of pervasive problems
  - Difficulties analyzing complex problems
  - Difficulties with changing environment
  - Hard to automate

Q
A