Two-Step Methodology to Reduce Software System Requirement Defects

Presented to

NDIA Systems Engineering Conference

21 October 2008

Presented by

Robert J. Kosman
Operational Systems Division/1552
(401) 832-8571, robert.kosman@navy.mil
“Typical”
Software System Development

- Waterfall / Incremental model
- Spiral model similar for a spiral
- Implies a sequential process to resolve problems (defects)
- Does not provide an adequate illustration of defect impacts
Software System Development

“Realistic”
Software System Development

- Added links backwards to reflect origin of defects
- Omitted links other than those back to the first phase – software system requirements development
- Rework caused by defects can impact cost and schedule

COST TO FIX
SCHEDULE

Defect Detection
DEFECTS AND REWORK

$ Rework caused by defects can impact cost and schedule

$ The later a defect is found, the greater the cost to correct

$ Defects found and fixed in later phases of development can cost up to 100x the cost to correct if detected in early phases
  - Software Specifications
  - S/W designs, code, test, documentation
  - Integration, T&E plans and procedures
  - Integrated Logistics Support (ILS) products (Operator / User manuals, Training materials, etc)
  - Distribution costs
  - Change documentation

REQUIREMENT DEFECTS

- Impacts all phases and products ("Negative Ripple Effect")
- Most costly to correct
- Cause delays in schedule and product delivery
- Initial system may have reduced capability and functionality, and most likely operational limitations
- Usually require formal documentation to correct, e.g., Engineering Change Proposal (ECP)

DEFECT CORRECTION EXPENDS RESOURCES AND FUNDS REQUIRED FOR PLANNED SYSTEM CAPABILITIES
S/W System Requirement Defects

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

☐ When:

- Focus on software development phase of acquisition; initial development or maintenance phase
- Prior to Software Specification Review (SRR) and Preliminary Design Review (PDR)
  - Low-level, defect detection process prior to high-level, program milestone review
  - Process generates better products input to SRR and PDR, or an Engineering Change Proposal (ECP) during life-cycle maintenance phase
- Used during system software specification generation, i.e., during translation of high level Performance Specification and user requirements (CONOPS) or User Requirements Document into low-level Software Requirement Specifications (SRSs)
- Systems Engineering (SE) organizes and runs the defect detection process
  - SE oversees technical aspects of the entire system acquisition, including processes to find defects in ALL products
S/W System Requirement Defects

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

☐ How:

- Analysis on past defects identifies two basic types of s/w system requirement defects
  - The defect that is unintentionally introduced into the s/w system requirement specifications during specification generation
    - Ambiguous text
    - Equation errors (algorithms)
    - Figure errors (functional and processing flows)
    - Table errors (wrong units, input ranges, etc.)
    - Connectivity and inconsistency issues
    - Missing or incomplete requirements
  - The defect that causes effort to be expended producing unnecessary, incorrect or unwanted functionality
    - “Bells and whistles”
    - Inadequate graphical user interface (GUI)
      - Systems are becoming more user interface driven (COTS) so the proposed GUI should be included in the s/w specification

Need to eliminate user comments like, “system should work this way”

CAUTION

S/W engineers will fill in the ‘holes’ and ‘gaps’
S/W System Requirement Defects

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

☐ How:

- Develop methodology/process to address both types of s/w system requirement defects
- First, tackle the mistakes made translating P-Spec and User specifications/CONOPS into functional flows and the GUI
  » “Bells and whistles”
  » Unnecessary, incorrect or unwanted functionality
- Second, tackle the mistakes made generating the s/w system requirements specifications
  » Usual mistakes made producing specifications, e.g., ambiguous text, etc.
PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

☐ Introduce a two-step methodology for s/w system requirements clean-up

1: Operational Demonstration (OP-DEMO) of the User Requirements
   » Visual demonstration of proposed GUI and functional flows
   » Allows evaluation of system functionality prior to development

2: S/W Inspection conducted on software requirement specifications
   » Rigorous review originally developed for s/w but can be applied to any “readable” products
Step 1: OP-DEMO

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

- **Visualization of the User Requirements**
  - Operability and functional flow
  - Graphical User Interface (GUI)
  - Target Machine or other

- **Different levels of OP-DEMO**
  - Operability features and functional flow
  - Operability features and functional flow with limited processing (e.g., algorithms)

- **Form of Software Rapid Prototyping**
  - Disposable code
  - Developed FAST using appropriate tools
  - User involvement early – during s/w requirements phase
Step 1: OP-DEMO

- **Wrong Concept of OP-DEMO (prototyping)**
  - Target machine is always utilized
  - Deliverable code
  - Considered ‘full’ system operability
  - User involvement in later phases
  - Fix problems in maintenance phase

**CAUTION**

OP-DEMO is Similar to Prototyping and Prototyping
Means Different Things to Different People
OP-DEMO Benefits

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

- Involves the User during the early phases, as opposed to the later phases or after system delivery
- Eliminates unnecessary and incorrect functionality and helps prioritize remaining functionality
- Provides a working model of intended operation for reference, as well as tool to allow parallel development of operator/training materials
- Identifies areas of uncertainty for risk management
- Promotes faster and more accurate s/w system specification writing
Step 2: Requirement Inspection (RI)

**PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS**

- “Software Inspection” applied to the Software System Specifications
- Not like an informal ‘Code Walkthrough’
- Formal, intensive review process designed to detect errors
  - Ambiguous text
  - Equation errors (algorithms)
  - Figure errors (functional and processing flows)
  - Table errors (wrong units, input ranges, etc.)
  - Connectivity and inconsistency issues
  - Missing or incomplete requirements

- **Basic characteristics**
  - Team approach, with assigned roles (reader, moderator, author)
  - Standards of conduct
  - Collect metric data
  - Criteria for Quality

Documented results indicate up to 85% of design and code errors can be detected by “Software Inspections”
Step 2: Requirement Inspection (RI)

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

- Team Members
  - Software Engineer (Lead)
  - System Engineer
  - User (or ILS person)
  - Test Engineer

- Multiple teams (2 or 3) detect more defects (N-Fold Inspection)
  - Small % of duplicate defects found between multiple teams

Multiple discipline involvement ensures consistent interpretation of software system requirements across phases
Requirement Inspection Benefits

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

- Ensures User requirements are accurately specified
- Ensures developer requirements are accurately specified
- Real-time metric data collection identifies areas of improvement w/ specification generation
- Errors corrected in single pass versus iterative correction process
- Detects errors associated with all phases of the Development
- Low cost / defect ratio
- Reduces software development costs by detecting errors early, avoids REWORK
Requirement Inspection Benefits

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

Impact of RI on Development (modified from [1])

Case Study

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

- Two extensive upgrades to an existing system – approx 100 KSLOC each
  - Existing system was really a “prototype/experimental” system delivered as a production system; so had to fix in Maintenance phase via ECPs
  - First upgrade did not use 2-Step Methodology to reduce Software System Requirement Defects; second upgrade did
  - Software System Specifications for first upgrade were developed by SE with only informal reviews, and significant portion of user interface was “TBD/TBS”
  - Software development team was already using Software Inspection during development so extensive defect metric data was collected during both upgrades
  - Causal analysis was conducted on all defects found to determine origin of defect
  - Both types of OP-DEMO were utilized on second upgrade (algorithms); 2-Fold RI also used on second upgrade
Case Study

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

Upgrade 1 Observations

Requirement Defects By Phase - UPGRADE 1

- Informal reviews found some defects but not enough
- Defects found during Design and Code could have been found by RI
- Defects found during computer-based Test and Post-delivery could have been found by OP-DEMO
- Rework caused schedule delays and end product had reduced functionality
- Defects required multiple updates to s/w system spec
Case Study

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

Upgrade 2 Observations

Requirement Defects By Phase - BOTH

OP-DEMO significantly reduced defects in computer-based Test and post-delivery phases

RI significantly reduced defects in Design and Code phases

S/W Requirement Spec had a “positive ripple effect” on development

Significantly less rework for 2nd upgrade and product was delivered on schedule w/ full functionality

Req defects were less severe and were easily fixed
Summary

PROPOSED METHOD TO REDUCE SOFTWARE SYSTEM REQUIREMENT DEFECTS

- Software system requirement defects can impact cost, schedule, and delivered functionality due to REWORK
- OP-DEMOS are useful in reducing defects that would be identified during computer-based Test and Deployment phases
- Requirement Inspections are useful in reducing defects that would be identified during Design & Code phases
- Improved s/w requirement specifications can cut costs in ALL s/w system development phases, including life-cycle maintenance
- Combining OP-DEMO and Requirement Inspection is a low-tech approach to reducing s/w requirement defects; is simple to apply and requires minimal training