When Go-Live Falls Short: Lessons in How Software Engineering and Acquisition Best Practice Could Have Saved the Day

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Patricia Oberndorf, John Hawrylak, Bryce Meyer
2 April 2014

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0000924

Outline

The System and the Situation

The Go-Live Experience

Facts Behind the Failures

Best Practices That Could Have Made A Difference

Epilogue

A Few Caveats …

- We will not reveal the exact system on which this is based.
  - However, what we are reporting has been observed in countless systems in our collective experience.
- It is generally based on an IT system that is interoperable across multiple defense and non-defense agencies.
  - But the problems observed here can happen in any sort of system.

The System and the Situation

The basic function of the system is to

- Accept both real-time and batch inputs, which may be less than pristine
- Compare them to prior inputs stored in a repository
- Add them to the repository
- Report back to the user on the results of the comparison

The system originated several years ago as a quick-reaction capability.

- Expedient design
- Expedient contract features

There are a number of COTS products available that support the main mission of the system (the comparison).

There are a large number of users in the field who submit the inputs.

Coordination with several other agencies may be required to complete the overall mission of the system.

The Go-Live Experience

The system went to go-live, and things started to happen:

- The system came up, but soon was not keeping up with the workload.
- The users were not getting responses.
- Results were falling on the floor, and submissions were not being entered into the repository.
- The users were soon frustrated.

Result:

- Program was forced to revert to the previous system version while trying to sort out the problems.

Facts Behind the Failure – 1

Further investigation revealed:

- Tests prior to go-live had assumed the inputs (and input formats) expected by the developer
  - Developer made a change in the version of the interface specification
  - Provided change to the input contractor
  - Input organization had chosen not to upgrade
    - Unknown to the Program Office and the development contractor
- There were no end-to-end tests of the actual system flow.
  - Unknown by the developers (or Program Office): some of the coordinating agencies were pre-processing user inputs
    - Both manually and with automated scripts that ran on the platform.
      - Scripts not included in the system build
  - Also unknown by the developers (or Program Office): some coordinating agencies provided personal assistance and service to users
  - The requirements process was out of control

Facts Behind the Failure – 2

Further investigation also revealed:

- Process and skill flaws:
  - No end-to-end use cases or user process flows
  - Tendency to see system as a set of point solutions rather than stepping back to determine common solutions to multiple needs
- Management flaws:
  - Technical staff who had only a minimal understanding of the system and how it worked
  - Failure to appreciate the need for system documentation, including the general process as well as architecture and modeling tools
  - Poor communication between PMO and stakeholders
  - Political in-fighting
    - Among coordinating agencies
    - Among contractors

Best Practices That Could Have Made A Difference – 1

Test:

- Never “assume” anything about what others in the overall process may (or may not) have implemented
  - Always conduct at least some of the tests with actual inputs from other participants.
- Always include end-to-end tests – starting and ending with the user in the field
  - The proof is in the total flow, not in the smaller pieces that are often the basis for tests before full system test.
  - End-to-end use cases are key to this overall system understanding.

Best Practices That Could Have Made A Difference – 2

Process documentation:

- Document user processes
  - Undocumented user processes are problems just waiting to happen
  - Knowledge of end-to-end user processes is essential
  - May be documented as use cases or by other means
    - Thorough documentation of the complete process, covering the entire route from user submission through return of a response to a user
    - Include all user/coordinating agency processes

Proven development processes:

- Use disciplined acquisition and development processes
  - E.g., CMMI covers such topics as Lifecycle Models (in the Project Planning Process Area), Organizational Process Definition (OPD), Involve Relevant Stakeholders, and System Transition.

Best Practices That Could Have Made A Difference – 3

Requirements processes:

- Institute and respect a bona fide requirements generation and approval process
  - All parties participate
  - Documentation on all requirements is clear and shared
  - Requirements changes are controlled
  - Requirements are vetted through a proper approval process

Best Practices That Could Have Made A Difference – 3

System documentation:

- Government personnel need insight into every aspect of the system
  - End-to-end process flows
  - Architecture and design information
  - Implementation and test plans and results
- System documentation must be created and delivered to the government
  - Government personnel must know what to do with it
    - Technically qualified to
      - Ask the right questions
      - Assess the answers provided
    - Must be willing and able to act on their technical assessments
      - E.g., to decide whether to accept or reject a deliverable and justify that decision

Best Practices That Could Have Made A Difference – 4

Contractual vehicles:

- Contracts need to support:
  - Creation and delivery of system documentation
  - Holding the contractor(s) accountable for its content and quality
- Government personnel must be qualified to oversee them
  - Knowing when to defer to the contractor – and when not to

Best Practices That Could Have Made A Difference – 5

Stakeholder communication:

- Document relationships with coordinating agencies
  - Something akin to SLAs, MOAs, etc.
- Ensure that
  - They truly cover everything
  - Everyone honors them

Epilogue

The government's response?

Once massive test failures were encountered, the Program responded with a classic set of Firefighting[1] decisions

- Redirecting all personnel to getting to the bottom of the go-live problems
- Putting work on the next version of the system on hold
- Thus risking subsequent problems in the next version's future

1 See http://resources.sei.cmu.edu/asset_files/whitepaper/2008_019_001_29209.pdf for more information on the Firefighting Archetype.

Questions?

Contact Information Slide Format

Patricia Oberndorf
Principal Engineer
Software Engineering Institute
Phone: 412-973-3459
Email: po@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Bryce Meyer
bmeyer@sei.cmu.edu

John Hawrylak
harlac@sei.cmu.edu

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm