Failed Implementation of A Real-Time ERP System

Summary of WSRcg Opinion – Testified September 2015
$80-$100 Million Bench Trial
Worked with Integrator's Counsel

This dispute involved a proprietary ERP system developer/Plaintiff Integrator (using a complex real-time transaction capture and processing system, interfacing with mechanical devices, robots, and multiple third parties) and a state agency Defendant Customer with a goal of collecting hundreds of millions of dollars. The Integrator was also contracted to operate the system for three years post Go-Live. The contract called for an 11 month schedule, 6 months of which was to be a freeze period, to be started after the Functional Design Requirements were signed off by the Customer. In fact, the Customer kept making changes to the design for 9 ½ months into the project, leaving only 6 weeks to develop, test, make ready, and implement the system so customer could meet its politically mandated Go-Live date. Customer terminated contract after almost three years. Judgment Pending

Our Overall Opinion: WSRcg opined that the delay by the Customer, in fact, was unsupportable because they were attempting to change the essential, internal, functional and technical design of the system, through hundreds and hundreds of changes and new requirements, in spite of the contract being a "performance contract" with little or no customization allowed.

Mr. Reid, who has extensive experience in software testing, systems readiness, and writing and keynoting on the art of testing large scale systems, testified in the following areas:

- **Opinion: The new mandated project schedule was unreasonable and unsuitable, and would lead to inevitable shortcuts of critical promised SDLC tasks – with predictable and negative consequences.**
    - The impossible crunching of six-months to six-weeks created a predictably chaotic, extended, reactive settling period with numerous emergency releases, bugs, and surprises.
    - The consequence of the Customer truncation continued past the post Go-Live settling period and into production for over 2 years.
    - Many of the errors discovered post Go-Live could have been unearthed and resolved in the contracted-for, six-month freeze period.
    - Many of the deferred items that got further deferred could have been performed and/or created much earlier.

- Industry lessons learned about software and systems testing:
    - No system is perfect! All systems have bugs & will experience a bumpy settling period.
    - Each test looks for different defects/errors. Next level tests do not uncover same errors.
    - Entry/Exit criteria for each test level must be established and met. Perform regression tests to ensure fixes/changes do not cause previously passed code to fail.
    - "Bugs found in later project phases than when first injected cost exponentially more to fix." (Barry Boehm, PhD)

- **Opinion: Approaching the software test process:**
    - The specific goals of testing;
    - What kinds of testing should be considered;
    - What the "best practices testing methodologies have in common; (PMBOK, SWEBOK, IEEE, and Software Engineering Institute, and more)
    - The ways in which a forced Go-Live date cripples projects.

- **Opinion: Mr. Reid's prediction of the defects, workarounds, surprises, operational emergencies, etc. that would result from the six-week crunch, and actually did, many times over:**
    - Report, screen, and form errors; data and table issues; reduced or eliminated negative testing;
    - Already-passed tests fail during later iterations;
    - Performance requirements met intermittently; software not synched to time for robotic responses;
    - Integrator is forced to issue multiple emergency system releases (2x per week → then 2x per month after Go-Live), all but eliminating time to focus on typical errors/surprises, or time to work on and/or complete deferred items or next releases, etc.

- **Opinion: The System worked!**
  Whatever functionality was missing, deferred, or performed in a different way than preferred, was largely because Customer refused to sign off on the Functional Design causing the Integrator to develop and test to a moving, invisible target. Through the Integrator's persistence, commitment, integrity and hard work, the system was kept operational, reliable, and secure: appropriately capturing, billing, escalating, and collecting from patrons; working with third parties and partners; updating accounts; handling patron service; and paying the Customer.