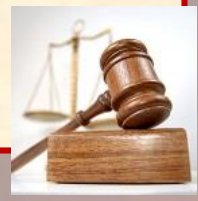
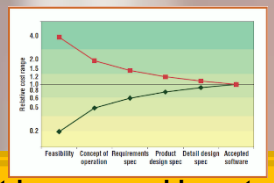
Anatomy of a Real System/Software Development Project Going to Litigation



Large state system is mandated by Feds. State engages Tier 1 software developer/integrator to build/operate new system. Project is 5 years over original estimate when terminated.
Key Issues: estimating; staffing; status/progress reporting; deviations from industry standard SDLC; requirements creation/management; test strategy; risk mgmt.; high vs. low quality system.

1. Customer enters agreement in good faith, expecting Integrator to do same. Contract contains, among other things, roles, responsibilities, deliverables and conduct of each party. Any hidden problems, issues, liabilities should be discussed & a plan to manage those risks developed and agreed to. Integrator makes material false promises to secure the job which Customer relies on to award job.



2. Estimates are NOT based on relevant historical metrics, comparable systems development, industry standard parametric models, or Delphi process. Estimates and Go-live Date(s) are overly optimistic to avoid late penalties imposed by Feds. Integrator hides calendar concerns, hoping to make up time by trying an agile method – a false assumption.

3. Promised Integrator's "A Team" w/ deepest IT, PM & domain expertise never shows up. Assigned staff is unqualified &/or assigned part time & can't perform "best practices" & methods. Best staff gets reassigned. Integrator throws incomplete, poor quality work over fence to Customer to approve/finish.

4. Project Status reports are "rosier than actual" and don't report actionable metrics to get/keep project on track. Earned values are overestimated. Critical path not managed. As milestones are missed , it becomes apparent that the risk of not-meeting the Go-Live date with a suitable, high-quality, maintainable system is very high. Schedule is extended 4x; Cost more than doubles. Integrator NOT learning from mistakes.

5. Unwise SDLC shortcuts taken which will cause damage to the system later, i.e., component & end-to-end requirements/design are unclear/contradictory/incomplete; code not reviewed for standards, logic, test coverage, complexity; unit tests are not standardized/performed/documented; critical tasks deferred (i.e., designing forms, screens, reports); JAD sessions poorly executed without prototyping.

6. Testing cut short at every level – but code still promoted to next test level. Certified test pros not used. The #/kinds of test data/scenarios are insufficient to test typical errors. Defect symptoms are addressed – not root causes. Static Analyses used intermittently. Automated Regression Test Suite not created, used or available for Maintenance Team. Interfaces/converted data untested in production environment. Error Messages/GUI not tested. High key staff/manager turnover w key unfilled positions.

7. Systems Test riddled w errors and stopped. Many end-to-end tests fail because individual components improperly tested earlier. Unit, integration, functional, end-to-end, regression, usability, security & performance tests cut short w/o proper customer knowledge/notification/addressing of risks. Items in Defect Mgmt System are improperly closed to give appearance of progress. Status reports conceal gravity of poor reqm'ts management, coding, certification tests & that project is in crisis. Integrator unwilling/unable to fix.

8. Neither party knows current system status/schedule, how much fallback, re-work, resources and effort remains; Integrator can't reliably estimate a Go-Live date and denies false claims it made before contract signed and throughout the project. Parties unable to reach mutually agreeable plan on how to resource, regress, rework & proceed. Finger pointing rampant. Termination & lawsuits.

9. Everyone loses!