A Beginner's Guide to the SDLC
Stuart Robbins, Chief IT Architect
TriNet, Inc.

Author's Note

There are many sources of information on the "best practice" known as the Software Development Lifecycle, from college texts written for software programmers that explain the "waterfall" development methodology to papers extending those principles into contemporary development techniques such as Agile, Scrum, etc. While I will summarize the basic tenets of SDLC theory whenever appropriate, this paper is not a substitute for established research in the field. Instead, this paper concentrates upon strategies for introducing SDLC-style processes where none exist, and tactics for addressing two of the major risks to SDLC initiatives in such early-stage implementations.

Introduction

Let's imagine the following two scenarios. In the first, you are asked to host Thanksgiving dinner for your entire family less than one week before the event, and you've never cooked for more than five people in your entire life. In the second, you are invited to attend your neighbor's Thanksgiving meal, a tradition they have happily fulfilled for more than a decade.

Of course, you try to do the best you can in the first scenario, given your other time commitments and your limited budget, with much of the work tackled at the last-minute with numerous urgent trips to the store for missing ingredients, utensils, napkins. The results are somewhat entertaining but far too chaotic, with too much anxiety and not enough turkey, over-cooked stuffing and relatives who plan to find a restaurant after they leave your house. In the second scenario, you sit at the precise time on the

invitation to a well-organized menu (even an alternate dish for the vegetarian cousins) including two different salads, coordinated wines for each stage of the meal, and perfectly-time Baked Alaska for dessert. The hosts are relaxed and charming, without a single stain on their clothing, where the biggest crisis of the evening involved their cat and some spilled soup. (They had soup!)

It should not be difficult to identify the reasons why one meal was successful and the other was a nightmare that in-laws will find amusing for years to come: planning, execution, and delivery. In scenario #1, with no previous experience and too many distractions, the "ad hoc" event suffered from poor timing, inadequate supplies, and a main course that triggered a health panic among those who are concerned about under-cooked poultry. In scenario #2, based upon experience and supported by tried-and-true recipes along with advanced knowledge of certain guests' special needs, the meal itself was the denoement when many different actions came together in a coordinated manner with sound results.

Simple anecdote, obvious conclusion: if presented with a choice, most of us would prefer to attend the well-executed meal.

And yet, in large and small companies around the world, we suffer from software products of poor quality delivered late and failing to meet even the basic expectations. We may know how to organize a dinner for twenty-five people but fail to apply that same wisdom to software development, which (for many) remains an "ad hoc process" that suffers from poor requirements (how many people are coming and what do they want to eat), poor planning (frozen turkeys purchased on the previous day require more cooking time), and low marks for customer satisfaction (at least we didn't get sick but we're never coming back).

For the purposes of this paper, the Software Development Lifecycle is summarized (not unlike a complicated meal for many guests) as composed of five primary stages:

Plan -> Design -> Build -> Test -> Launch

As previously noted, there are many detailed expositions on these primary stages, with various naming conventions and alternate definitions, however, the objective (in the following sections) is not to declare one superior to another but, rather, to provide a basic primer for those organizations who want/need to move from scenario #1 to scenario #2 (Getting Started) and offer antidotes for two of the major risks to long-term adoption, once the new process is underway.

<u>Life Before Project X</u>

In the case study outlined herein, the following organizational characteristics were present before and during the launch of "Project X" that should be considered environmental factors or boundaries within which the SDLC methodology emerged. The trigger for SDLC initiatives may originate from many sources (Quality complaints, Customer requirements, Compliance mandates) however, the following recognizable characteristics are important to understand with regard to this case study:

- Limited Project Management resources with hundreds of open projects to lead

- A small QA group

- A web development team burdened by support for existing products while new development was launched without testing

- An ERP team with rigid protocols and fully-documented processes

- An infrastructure team that rarely learned about a request for new servers until the day they were needed to be operational

- Ten ways to submit a request to IT

- No agreed-upon governance for budgeting or prioritization

Everyone worked as hard as possible on as many projects as possible, often in the evening and on weekends. Management understood the need for a more orderly environment, but the inevitable "fire drills" of unexpected requests from the business kept their teams, and the entire organization, in "spinning plate" mode – attending to the most urgent issues, juggling multiple projects with inadequate resources, often resolving conflicts in the hallway between meetings with dissatisfied constituencies.

It was in this challenging environment that the company's IT Audit staff, preparing for the pending arrival of SAS auditors, that 10-12 senior developers and analysts attended an exploratory meeting to discuss the Compliance requirement for SDLC documentation. Previous audits focused solely on methodologies within the ERP team, however, the Director of Compliance was now requesting policy and procedure documentation that established IT policy for the entire organization.

<u>The First Meeting</u>

Participants were identified by the IT Management Team based upon a broad set of guidelines: all groups must be represented, seniority (based upon experience with a variety of projects in this environment) was balanced by those who are predisposed to cross-functional communication and collaboration. We decided to drive adoption of basic SDLC practices "from the ground up" by creating a task force of individual contributors, on the theory that successful adoption is proportionally related to the degree that employees felt involved in the decision-making process.

Our objective was to draft a proposed policy/procedure document that would be presented to the management team for approval. Our deadline was the next scheduled visit from the auditors, less than three months away.

We opened the meeting with a clear outline of the task force objectives. Following that was a "venting session" that lasted for two hours, each attendee identifying barriers and roadblocks that their projects suffered: too many ways, most informal, for a new project to begin; no standards for gathering requirements or clarifying scope, which changed constantly; inadequate time to complete the project, schedules often preventing the limited QA staff from having any substantial impact; hallway decisions that often shifted the direction of projects midstream with no escalation path to clarify the inevitable confusions; teams frequently working at cross purposes, underlying platform (ERP) development not synchronized with either the application developers or the server and storage teams who rarely learned about the project until the eleventh hour.

An SDLC policy, even the most rigorous and well documented, would not resolve these challenges.

However, it was clear – from the group's enthusiasm and the mandate from the IT Audit department – that there was an opportunity to "shine a light" on each of the above-mentioned challenges; using the SDLC process as a vehicle for identifying other IT processes in need of improvement, the task force eagerly committed not only to the initial objectives, but to the broader ambition of improving the overall operational capability of the IT organization.

Each of the representatives recognized the benefits to their particular function: the web developers aimed for tighter coordination with the ERP team, and the ERP representatives recognized the chance to expand their rigorous procedures in ways that would influence all major initiatives; the QA staff anticipated stronger sanctions for test cycles, the infrastructure representatives also seeking inclusion in project planning and advanced notification of procurement or configuration requirements. The latter change would, by itself, contribute to better budgeting alignment. Everyone agreed to take the issue back to their teams and return, in two weeks, with three specific steps their functional teams wanted in the policy document's first draft.

One of the attendees, within minutes of the first meeting, declared that this was the first time they felt hopeful that things could get better. Everyone's energy and commitment was palpable.

The result, in three months and without impact on any of the individual's existing responsibilities, was a policy document of enough substance to meet address audit requirements and drive agreement at the management level that the SDLC proposal should be implemented for all new projects.

The high-level workflow documented by the task force emulated SDLC "best practices" yet was quickly embraced because it was "organically" constructed through collaboration by teams and individuals who found immediate value in the model.
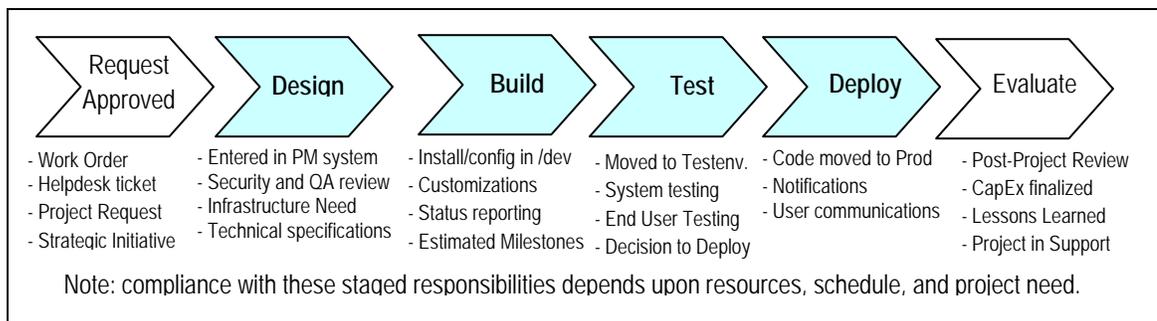

Request Approved	Design	Build	Test	Deploy	Evaluate
- Work Order - Helpdesk ticket - Project Request - Strategic Initiative	- Entered in PM system - Security and QA review - Infrastructure Need - Technical specifications	- Install/config in /dev - Customizations - Status reporting - Estimated Milestones	- Moved to Testenv. - System testing - End User Testing - Decision to Deploy	- Code moved to Prod - Notifications - User communications	- Post-Project Review - CapEx finalized - Lessons Learned - Project in Support

Note: compliance with these staged responsibilities depends upon resources, schedule, and project need.

Diagram 1: Proposed Project workflow[2]

---

[2] Selection from <u>Systems Development Lifecycle: Proposal for Management Consideration</u>, TriNet, Inc., August 2008, based upon "best practice" models in the public domain.

<u>The Experiment</u>

It looked good on paper, sounded good to management, and was applauded by various business groups who hoped to model their new processes (i.e., product management) upon IT's direction. The explicit objectives of the task force were complete, initial audit requirements (demonstrable progress) had been fulfilled, but the task force members were not ready to disband. Moreover, they insisted upon continuing our weekly meetings with a new and more focused goal: implementation. Among the many barriers facing the SDLC team as they sought to be compliant, not only in principle but in practice, was the immense and complicated list of multi-year projects and initiatives that plagued IT, each attendee involved in too many "in flight" projects to consider adding yet another to their workload.

They needed a pilot project with sufficient urgency to capture management's attention and sufficient breadth to demonstrate the value of an SDLC-style process, yet a project that required no one's *time*.

Project X was formally documented as an urgent tactical request from executives, entered in the company's Portfolio Management system, and milestones were tracked with status reports delivered at the SDLC team's weekly meetings. The Company's CIO, in support of the pilot project, agreed to serve in the role of the project's Executive Sponsor and immediately sent a priority email to the team with an all-too-familiar demand: Project X, estimated in the initial project plan as a 4-6 month effort, must be delivered "to the market" in four weeks.

For years, dramatic changes in scope and hurry-up schedules were the norm. The CIO was confronting the SDLC task force to face ten years of habit, and they wilted in the face of executive pressure. Their next meeting began with a flurry of rapid and emotional proposals to meet the new deadline by eliminating all documentation and testing requirements, and re-defining any Project X objective as unimportant if it could not be delivered by the end of the month.

Organizational habit, and the absence of any conceivable alternatives, had transformed their passionate drive for Change into yet another unmanaged and under-resourced race to an irrational deadline over which they had no control. Their deflated morale was almost palpable in the conference room.

The First Obstacle

In any work environment, and particularly in the technology and service sectors, hierarchical authority is reinforced economically, politically, and socially. In the absence of alternate governing principles offering the "check and balance" function, and in time-constrained situations that favor "moving fast," it is human nature to opt for the default momentum of "command and control" decisions. A supervisor says "just do it" and his/her employees "just do it", often spending evenings and weekends in a relentless effort to accomplish the impossible. The results of such environmental dynamics are well-known: schedule delays, poor product quality, over-spending.

To compensate, a governance framework (tools, policies, templates, and distinct exception management guidelines) is a foundational component, and the SDLC policy (which had been approved in principle by the management team) simply needed someone to enforce its basic themes.

A template was designed, and an Escalation Meeting was called for the following morning.[3]

Everyone knew the sponsor's demand was unreasonable, yet there were no clearly-defined methods for business case analysis (justification) or project review (document approval), no agreed-upon roles and responsibilities (chartered accountability for cross-functional oversight) nor a shared vocabulary for communicating urgent issues. They knew it could not be done, they simply had no method for explaining that potential failure to the business. With a template, and an agenda for discussion with the Executive Sponsor, the team now had the means for review and feedback, an essential component of lifecycle methodologies: at each stage, those with the relevant expertise have checkpoints, gates, and channels for communicating risk whenever and wherever it appears.

Based upon the team's one-page escalation document, the Executive (CIO) lifted the schedule constraint he had previously imposed. In doing so, his team learned a secondary lesson about the need for professional candor and good process as precursors to good decisions.

---

[3] It is beyond the scope of this article to identify the key factors that lead to successful executive escalations, however, one or two aspects are appropriate to note in the context of Project X. Escalations, like any other business process, can be done efficiently or awkwardly, and the subsequent results are a microcosm of an institution's process health. When efficient, they can be vehicles for clearer and more accurate decisions.

Only in its Design stage, Project X had already accomplished a key objective by emphasizing that attention to process, howsoever lightweight and timely, increases the likelihood of successful delivery by quickly responding to unexpected changes that would otherwise increase risk.

The Second Obstacle

There is an even greater barrier, in the early establishment of an SDLC-style governance for IT projects, than the absence of governance principles. This informal case study would be incomplete without identifying this challenge that looms larger than most "change management" issues, yet can be traced to any organization's resistance in the face of dramatic transformation.

On a purely tactical level, it is the challenge of orchestration. Once a new development methodology has been a) defined, b) embraced, c) sanctioned, and d) successfully prototyped, it can be very difficult (if not impossible) to identify a "good time in the company's schedule" to implement new rules: it isn't advisable to introduce new methodologies in the middle of a project's timeline (i.e., the futility of re-designing the airplane mid-air), and yet, there are always "in flight" projects that need to be granted exception status. This, however, reinforces the very habits (urgency, "just do it", etc) that the new methodology aims to address. Therefore, upon being sanctioned by the company (particularly in regulate industries wherein compliance with published policy is mandatory), the management team must carefully identify the roadmap for implementing new guidelines.   In some cases, a complete moratorium on new development may be required (2-3 months) to ensure that the SDLC takes root and is given its well-needed foundation.

On a strategic level, there is the broader challenge of organizational architecture. New teams will be needed, new management principles will be required. Behaviors that, for many years, have been successful and even rewarded (code without embedded comments, testing completed by the source engineer) will become less important to the overall health of the organization than the newer skills of planning, prioritization, communication. Ensuring that the organization's architecture is re-designed in coordination with its processes can not only complement but enable the objectives of any SDLC: planned and well-managed execution of projects, consistently, within projected budgets and with reliable levels of quality that are incrementally improved over time.

* * *

Epilogue

Corporations mature in standard developmental cycles – launched by entrepreneurs like the innocence of small children at play among skyscrapers, struggling through the awkward adolescence of stronger competition. I would no more recommend the adoption of a software development methodology to venture-backed startups than would I encourage algebra in kindergarten. Each stage of a company's growth brings, with it, the opportunity to shed behaviors no longer relevant.

The introduction of process-oriented governance methods (like the SDLC) should be similarly considered, after a candid self-assessment by executives who, themselves, may be holding on to behaviors that may have once been, but are no longer, useful.

"We must re-architect ourselves" to the degree that we yearn to re-design our systems and methods.[4]

I had an opportunity to use this metaphor and will share this anecdote in closing, to explicitly re-iterate the importance of thoughtful management (parenting) in the move toward SDLC-style policies and procedures. The conversation occurred during the early weeks of the Task Force, before the value of review/oversight was widely shared. The Compliance Team (in the face of an impending audit) was insisting upon complete (documented) evidence that each production-ready server had been "hardened" to meet accepted security standards.

The manager of the company's small but resourceful UNIX team challenged the benefit of lengthy checklists, standardized templates, submission with signatures, on the accurate prediction that it would slow down his team's ability to rapidly respond to requests. He noted that they were highly skilled and capable of moving very quickly, as the Company needed in its early years.

I asked if he had children, and he acknowledged a young daughter and younger son with a smile. With this in mind, I offered another perspective for him to consider:

We all know that very young children (2-3 years of age) become enamored with Mommy and Daddy's car keys, and they often become a favorite toy. Perhaps it is the jingle, or the cool touch of metal on their skin, and perhaps it is because the keys are objects that serve as a connection with the parents. Whatever the reason, the manager of our UNIX team agreed that a good set of car keys is a great way to keep a baby happy at a restaurant. I observed that using the keys in this fashion was not only convenient, but appropriate in those circumstances.

However, as time passes and the baby becomes a young girl, nearly sixteen and stridently independent, what parent would calmly hand over the car keys for fun? Same girl, same parent, same tool, yet as the family matured, different challenges present themselves, requiring different boundaries, different rules. What was once appropriate and convenient can become thoughtless and even dangerous, over the course of time.   The health and welfare of a company may require a much stricter code of conduct as it strives to be compliant, meet contractual commitments, reach for even more growth.

Good governance, like good parenting, is as much a matter of timing as it is wisdom. If the company needs discipline and rigor that process-oriented methods provide, we should not, as executives, neglect those needs.

---

[4] Steve Yatko, Director of IT R&D, Credit Suisse (NYC), 2002.