High-Performance IT

Applications Development Health Check

russell
futcher
.com

Applications Development

Development

The Best practice IT Standard:

The Best practice IT Standard is comprised of:

1. A thoroughly documented and tailored where appropriate, development lifecycle methodology. (Systems Development Lifecycles-SDLC) with associated how-to guidelines and procedures.
2. A vendor supported, integrated applications development toolset.
3. Development and test databases refreshed daily.
4. As the applications portfolio is usually a mix of packaged solutions and in-house developments, both require thorough documentation, especially legacy applications.
5. Development work like enhancements requires rigorous functional gap analysis and review of business processes before work is undertaken.
6. The use of scripts is minimised as they tend to organically grow which makes them problematic and requiring manual intervention to run.

Performance Health Check

Testing

1. What is the approach to testing?
2. Does the testing approach cover people (numbers and skills), capacity, software and tools, overall technical environment, processes (e.g., function test, performance and load test, problem reporting, management and fixing), and the user functions?
3. Does testing cover off usability, reliability, functionality, and performance versus the documented requirements.
4. How is testing planned and documented?
5. How are test statements planned and documented?
6. Do test plans include component tests, integration test, regression tests, system test and acceptance test.
7. How are acceptance criteria for the system deliverables defined?

Quality Assurance

1. What quality management tools and techniques are in use?
2. How is quality assurance integrated with sub-contractors?

Conversion

1. What format do the conversion strategies take?
2. Are conversion rules, programs and final data file definitions defined?
3. How is the original environment described?
4. What if any conversion tools are in use?
5. Are conversion and integrated acceptance test plans defined?
6. How is the clean-up of converted data defined?
7. How is a newly converted system compared with the original to assess identical results?
8. What data integrity issues exist?

Sample Task list

1. Create an end-to-end testing process including the use of test harnesses, scripts and conditions.
2. Review the use of and management of development and testing environments.
3. Review recent application failures (unusable to users), assess cause and ensure mitigations been put in place.
4. Introduce a Rule of ‘fix a problem once’.
5. Split DBAs into Systems and Applications.
6. Locate application DBAs with developers.
7. Create any necessary umbrella processes that involve inter-team workflows including hand-off points and deliverables.
8. Pending how often conversions are undertaken, consider creating a conversions process that includes sub-contractors.
9. Determine extent of data integrity issues and how to resolve.
10. Define and agree a test environment.
11. investigate QA toolsets.
12. Determine further works required and scope out.
13. Breakdown the scope of works to task level, ready for loading into the change management project schedule.

Methodologies

The Best practice IT Standard:

Is that applications development methodologies are guiding all development activities, there are five primary types:

1. The waterfall model: - this is the classic SDLC model, with a sequential process that has goals for each applications development phase. The waterfall model

simplifies task scheduling because it is linear with no iterative or overlapping tasks.

2. Rapid application development (RAD): - based on the approach that solutions can be developed more quickly using development workshops to collect system requirements. Makes use of prototyping and reiterative design testing.
3. Joint application development (JAD): - this model involves the client or end-user in the design and construction of an application through a series of collaborative workshops.
4. Prototyping: - in this model, a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.
5. Synchronize-and-stabilize: - this approach calls for different development teams working in parallel on different parts of an application. Requires regular synchronization of code.

Performance Health Check

1. How often are development databases refreshed from production databases?
2. Are the methodologies in use tailored?
3. What methodology how-to guidelines exists?
4. Are there matching Work Breakdown Schedule templates in use?
5. What is project delivery performance like (timely delivery of all software developed by (and subcontractors) to defined acceptance criteria (i.e. on schedule, within budget, with required function, with required performance and to the specified quality)?
6. Is there an agreed and documented procedure for interfacing with subcontractors?
7. How are subcontractors advised of in-house development standards?
8. Are regular performance reviews of progress including monitoring actual versus plan on a task and effort basis conducted with sub-contractors?
9. Is Quality Assurance involved in code inspections?
10. Is there a clear and unambiguous view of the overall application architecture and, is this conveyed to subcontractors?
11. Is a close and continual liaison with the technical architect maintained to ensure that all matters potentially affecting system capacity and performance are communicated and understood?
12. How are applications security requirements managed?

Sample Task list

1. Review software licenses.
2. Risk analysis of in-use methodologies.

3. Assess methodologies fit for purpose.
4. Document in-house methodologies and customisations to packaged methodologies.
5. Determine further works required and scope out.
6. Breakdown the scope of works to task level, ready for loading into the change management project schedule.

Releases vs Projects

The more complex the design, the bigger the project. But as projects get bigger the risk of failure increases geometrically while the likelihood of substantial delay approaches certainty. Releases bundle discrete enhancements for regression, stress testing, and deployment. Relying on releases instead of projects lets you take advantage of one of the most reliable heuristics of IT management: Enhancements succeed, projects often fail.

Organizing development work into releases still creates delay. Typical scrum sprints are a month in duration, which establishes a monthly pace for business change, not including having to wait for the critical Change Advisory Board (CAB) meeting).

Favouring Interfaces over Integration

New functionality creates new value. But in most IT shops, new functionality takes a backseat to making sure a software change doesn't break the company's spider web of custom-programmed point-to-point batch interfaces. Clean up the Interface tangle with a well-engineered integration system which results in project teams speeding up, testing taking less time, and deployments going more smoothly.

Insisting on 100 percent solutions

IT's instinct is to bulletproof everything. But coding for a case that hits the system once every thousand transactions takes as long as coding for a case that occurs hundreds of times a day. Take a lesson from the ancient days of IT: Program the main cases and kick out the rest as exceptions for manual processing. Computers are good at main cases. Humans are good at exceptions.

Creating Data Warehouses

Pick two words to describe the typical data warehouse project and it would likely be 'behind schedule.' Data warehouses are still chronically late due to the difficulty of designing OLAP data structures optimized to answer questions nobody knows they're going to want to ask. Enter NoSQL. What makes NoSQL interesting isn't only its ability to handle large data volumes. Even more valuable is its ability to accept data now and let analysts figure out its organization later, when the time comes to query it.