Best Practices:
for Testing & Monitoring

…in the Agile world of Continuous Integration & Delivery

Stormcloud Consulting & Advisory:
written by: Walter Sturm

In order to make software more responsive to market conditions, and adapt to the demands of digital business, software development has been undergoing an evolutionary shift. Over the past few decades, the process of creating software has been evolving from a series of sequential phases with longer cycles and monolithic requirements definition to a routinely iterative process with much shorter cycles and dynamic requirements enabled by close interaction between business and development teams.

As a result, businesses are realizing higher quality code, which is better aligned to stakeholder interests, in a fraction of the time, and at lower cost.

Some examples:

a. 10 years into a project to overhaul the FBI's Case File System, a team of 300+ developers were ultimately unable to deliver a functioning system after spending $600M using traditional practices. A subsequent effort employing modern methods, completed the project, meeting all requirements with a team of just 45 developers, and replaced the outdated system on schedule for half the allocated budget at a total cost of $99M.

b. Prior to modernizing development methods, Hewlett Packard's Laserjet Division spent only 5% of their time innovating. After a 3-year transformation, HP's Software Development efficiency has improved so much that they now spend 40% of their time innovating.

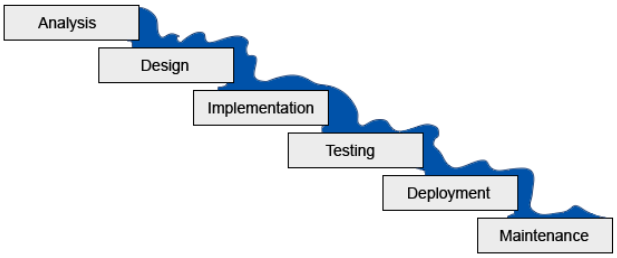
c. There are dozens of additional examples where companies such as Bank of America, Barclays, Labcorp, Spotify, Nordstrom, Sony, Fidelity, Amazon, Etsy, etc… have dramatically shortened delivery times and increased output by applying more modern development practices.

This shift is the adoption of Agile techniques to the Software Development Lifecycle and like any evolutionary process, change is not immediate but occurs over many years and most often in small steps. With this in mind, we've published this paper to help companies assess where they are along this evolutionary path, and identify industry best practices which are realistic to implement given their organizational constraints.

The Old World

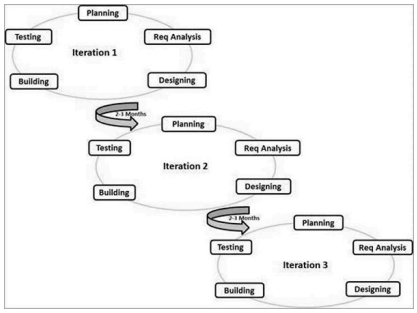
1. The Traditional/Waterfall Process of software development is a highly structured, top down process divided into distinct phases where dates define the boundaries of each phase. A waterfall project begins when detailed requirements are gathered and documented. With fixed requirements set, hardware, and software design decisions outline the environment within which the development team will build. Requirements are then separated into smaller units which are coded and tested. Once all the software units are tested, the component units are integrated into a system and tested as a whole. When the development team is satisfied that the project code meets the documented requirements, business users perform a final acceptance review before release. Any bugs that are discovered or changes that are requested are addressed as patches or minor version releases during the maintenance phase. The typical Waterfall cycle is 6-24 months.

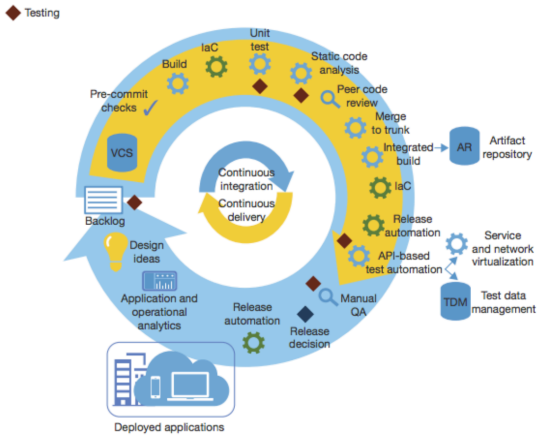
1. Requirement
2. Design
3. Code & Build
4. Testing
5. Deployment
6. Maintenance



The New World

An Agile Project begins with User Stories which describe light weight functional descriptions of why and how a user will interact with the system. User Stories promote greater interaction between the user and developer and focus on a limited scope relative to traditional requirements. Larger projects are separated into smaller builds. Self-organizing, cross functional teams are formed comprised of all members necessary to complete a build as described in the User Story. Each Iteration includes requirements, build, testing & implementation with the cycle repeated for each iteration. Where multiple iterations are built concurrently, Continuous Integration is the process of merging all working copies (iterations) to a shared "Mainline" copy as often as several times daily. A typical Agile Cycle is <1 day - 3 months.





The Real World

In reality, most companies no longer use pure Waterfall methods. Likewise, most have not fully adopted Agile ideals but are somewhere along the evolutionary spectrum in between. We describe the process of selectively choosing component processes along with the frequency of software release cycles as the degree of enterprise "Agility" or "Agile Maturity." Organizations that employ many Agile processes and release code every week are more Agile than companies that have implemented only one or two processes and release code every 6 months.

There are a number of reasons why companies are more or less Agile. Some organizations are pursuing a "Best of Both Worlds" approach by adopting Agile methods where they see benefits and retaining the elements of Waterfall that better suit their business needs. Other organizations are committed to fully adopt Agile but recognize that it will take time to reorganize, retool, and revise their processes.

Like most things in the real world, nothing is best in all cases and while Agile Methodologies have delivered some amazing results, it follows that some heavy-weight, formal, waterfall techniques will better suit some businesses.

Consideration:	Agile	Waterfall
Scope	Time to Market is Key	Complete Project is Key
Client Availability	High	Limited
Funding	Best with Time & Materials	Best with Prix Fix
Team	Routine Collaboration	Defined Handoffs
Feature Priority	Adaptive	Static

Some common reasons teams choose to retain some of the structure of Waterfall methods include:

- Fixed-scope, Fixed-price contracts
- Client or Business Unit do not expect rapid change in scope
- Client or Business Unit enforces a very formal approach on suppliers.
- Client or Business Unit representatives are not routinely available
- Performance-measures based upon delivery date and budget
- Upfront investment is not risky to make …Waterfall = longer time to realize a return
- Mistakes are not recoverable …with Agile mistakes are presumed but quickly corrected
- Work cannot easily be modularized`\```

Conversely, the biggest benefits of a more complete Agile approach will be realized where:

- Time to Market is Critical
- Alignment to Market is Critical (particularly where):
    - Close Collaboration with End users is possible/feasible
    - Adapting to rapidly changing requirements provides a competitive advantage.
    - Higher Quality User Experience provides a competitive advantage via fast feedback loop
- Value and Definition are not well understood.
- Mistakes are recoverable
- Complex Problems particularly where the solution is unknown
- Work can be modularized particularly when each incremental step has value

Agility

As of a 2013 report, Gartner Research found that the majority of enterprise development shops were still working on a 6 month or greater SDLC (Software Delivery Life Cycle). Similarly, according to a 2016 survey, 82% of organizations are functioning at or below a “still maturing” level as it pertains to modernization of IT processes with Agile methods.

Regardless of where your organization falls relative to the industry, there are a variety of maturity models which may be useful to help understand where your organization is currently, as well as which areas you may want focus on next to continue moving forward. Some models focus on culture and organizational maturity while others focus on best practices. Both types of models are beneficial since implementing Agile methods typically requires significant change

and for most businesses, both cultural and organizational challenges will need to be addressed either as a pre-requisite or during the process.

While this paper is focused on Best Practices from a DevOps/QA perspective, a comprehensive self-evaluation will include the following:

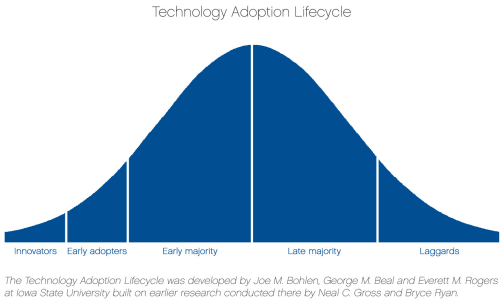
- Organization
- Architecture
- Build Process
- Testing
- Reporting

As you evaluate best practices, if you uncover cross-organizational barriers, we recommend augmenting the guidance in this paper to also address the organizational aspects of Agile Transformation. There are several good books, industry papers and case studies which document real-world lessons learned by organizations that have already undergone the process.

Classifiction:

Following a common technology adoption curve, we've categorized stages of evolution as it relates to your peers:

- ***Laggards:*** Falling Behind
- ***Majority:*** Where most companies are today.
- ***Early Adopters:*** Moderately Mature Agile Practices (some differentiation)
- ***Innovators:*** Very Mature Agile Practices
- ***Bleeding Edge:*** Industry Leading and often costly to implement



The Technology Adoption Lifecycle was developed by Joe M. Bohlen, George M. Beal and Everett M. Rogers at Iowa State University built on earlier research conducted there by Neal C. Gross and Bryce Ryan.

Best Practices for each Stage:

Each stage either builds upon (adds new capability) or evolves from (replaces the less effective method) capabilities of the prior stage.

Laggards:

Agility Gauge: Organizations in this category develop Monolithic Code which is integrated at the end of a Build Phase. Report generation is manual.

Testing Typical Practice: Testing occurs at the end of Integration which might be manual or include scripted tests. Manual Functional and Regression tests are performed in Staging.

Monitoring Typical Practice: Basic up/down infrastructure (server & network) monitoring practices are employed.

Majority:

Agility Gauge: Organizations have deployed a standardized Build Process and Dev Environment with Manual Deployment, which may include Automated Deployment Scripts. The SDLC is measured and routinely reported on.

Testing Best Practices: The majority of organizations have processes where Automated Functional & Regression Tests are employed in Staging with Developers quickly addressing failures. Ad-hoc Capacity Tests are routinely performed with a new release.

Monitoring Best Practices: Most organizations have one APM (Application Performance Monitoring) component employed. Commonly this is Application Monitoring (Application Discovery, Tracing & Diagnostics) solutions or Digital Experience Monitoring (DEM) which comprises Real User Measurements (RUM) or Proactive (Synthetic) Monitoring i.e, Automated Scripts checking Availability, API Function, etc…

Early Adopters:

Agility Gauge: Continuous Build & Integration at scheduled times with Dependency Management Repository along with Push Button Deployment to Test Environment and Production including Disaster Recovery. Build Server reports about code changes, source code analysis, and compilation errors as well as testing results are tracked historically with critical Reports available across teams.

Testing Best Practices: Organizations employing Continuous Builds (scheduled) with Fully Automated Deployment to a Standardized Test Environment. They utilize a Broad Variety of Tests both Functional (Automated & Manual) as well as Source Code Analysis run by a Build Automation System. Limited Capacity Stress Tests are run on Staging Environment for planning.

Monitoring Best Practices: Organizations integrating Server (Application Monitoring) and User (Proactive/Synthetic or RUM) data together into a correlated, cross referenced view. Performance monitoring is automated and routinely tested ad-hoc at scale (capacity testing).

Innovators:

Agility Gauge: Utilizes Scalable Build Cluster with Load Balancing and Fully Automated Deployment to Test Environment followed by Fully Automated Deployment to Production contingent upon Quality Gates. Push Button System Deploy of Coordinated Builds (collection of Web Services) promoted concurrently. Trending Reports assess SDLC Performance over time.

Testing Best Practices: Organizations take a rational approach toward the goal to test 100% of their code (testing all user flows and code pathways). This translates to foregoing some limited number of tests because they are too expensive but ensuring that all realistic scenarios are thoroughly tested. This would include Functional & Unit Testing for all important System functions, Distributed Load Tests at Scale (Load Testing), as well as Performance Testing, increased frequency of Source Code Analysis, Run Time Monitoring, and Vulnerability Probes. Testing Scripts are reused for Operational Monitoring.

Monitoring Best Practices: Organizations correlating data sources from, Application Monitoring, Proactive/Synthetic Monitoring, and RUM (which includes Web User Behavior, Traffic, & Performance, as well as Video Behavior, Traffic & Performance) into an Application Analytics engine which typically consists of Automated Detection of Anomalies, Baseline Metrics, and Visualization across data types.

Bleeding Edge:

Agility Gauge: A Versioned Build Process enables Continuous Deployment scheduled to Production with Fully Automated Testing & Monitoring. Reports tie SDLC performance to process changes.

Best Testing Practices: Organizations forego consideration of cost/difficulty and are strict in their interpretation of testing 100% of expected user Scenarios. Every Line of code is tested via Automation with the assumption that the cost of creating some Tests will outweigh their benefit in return for the reduced risk of 100% coverage. This includes Vulnerability Testing on the Staging Environment.

Monitoring Best Practices: Organizations employ a sophisticated data collection system leveraging all relevant data. In addition to APM (Application, Proactive/Synthetic, RUM) data being correlated, core network, security, and business data are correlated into an Advanced Application Analytics engine which adds Predictive Analysis, and triggers additional automated tests to reduce the steps associated with manual trouble shooting.

Recommendations/Summary:

Identifying how Agile your organization is may help you map out your next steps and while we don't outline specific dependencies, it's safe to expect that moving up to the next level from where you are today is generally a realistic objective.

With this in mind, there are a few important trends we've observed…

The first is that the historic line between Testing & Performance Monitoring is becoming less distinct. It's evolving into a single function differentiated only by whether a check is performed before or after code release. More commonly organizations are testing complex user scenarios on their staging platform and reusing the scripts in production for monitoring. Using the same Scripts Pre and Post Release (Testing & Monitoring) provides consistency between Development and Operations as well as the increased efficiency of creating once and leveraging the script multiple times across the lifecycle.

As this linkage between testing and monitoring becomes stronger, it's evolved into a single continuous process which occurs more frequently as well as shifted earlier into the development lifecycle (i.e. test early, test often). Some firms are integrating functional test modules with regression tests to build scripts and push the concept of reuse earlier into the workflow.

Naturally more frequent testing leads to more automation. So, we're seeing organizations investing to automate as many of the testing components as their business model supports:

- Functional
- Integration
- Regression
- Vulnerability
- Performance

While more testing is sure to improve the quality of code, as many organizations have experienced, component tests/checks don't always predict how a system will perform under load. So, we're also seeing automated Load Tests with realistic user scenarios being performed at scale, concurrent with Proactive/Synthetic Performance Monitoring and/or Real User Measurement data being integrated for analysis. In fact, it's becoming more common for organizations to collect data from increasingly varied sources and aggregate the data with their own tools in order to present a comprehensive view upon which to make intelligent decisions, as well as reduce both Mean Time To Respond and Mean Time To Repair. Sources such as Proactive/Synthetic, Application, Network, Server, and Application Monitoring, with RUM Beacons, along with CDN (Content Delivery Network), and HSP (Hosting Service Provider) Log Files.

Finally, we're seeing organizations re-evaluate the toolsets that support their SDLC, Testing, and Monitoring in order to enable more agile processes, reduce test script development time, and generate more realistic scripts. Tools that are less flexible, and/or poorly integrated require a significantly greater investment of time and return far less value to the business in terms of data or risk mitigation.