NCHS Executive Summary

In coordination with NCHS, Altimetrik was requested to lead the Agile / DevOps assessment revolving around understanding and determining improvement opportunities for the engineering team to effectively adopt Agile and DevOps practices to:

- Define standard and consistent end-2-end delivery models leveraging Agile and DevOps to improve developer productivity, product quality, and tool chain integration
- Identify best practices related to delivery frameworks, tool chain architecture, environment strategy, release models to develop a scalable and repeatable product engineering practice driving innovation
- Define a comprehensive target state leveraging existing tool chain, refactored processes, and automated frameworks to ~~accelerate delivery to non-production and production environments to~~ accelerate delivery, instill predictability and shorten time to non-production and production environments.

.

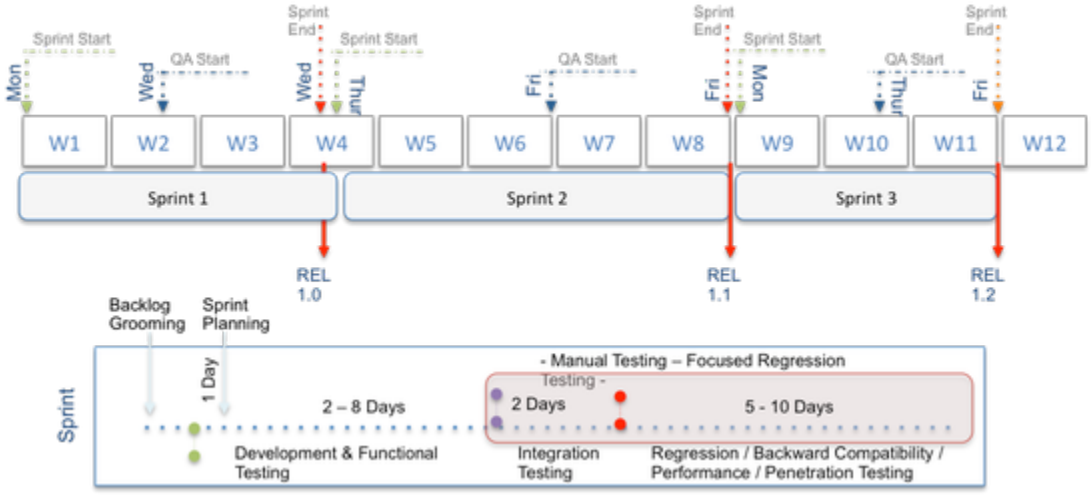
Our Agile/DevOps assessment involved spending time on site with the engineering team to conduct a thorough review of the end-to-end product delivery model and practices related to portfolio management, program prioritization / alignment, scrum best practices, product development life cycles, build/package/publish/deploy practices, tools to develop a scalable and modern product engineering practice driving innovation and productivity.

Key Observations

Based on working sessions across the different teams / team leads, there were common themes identified which need to be addressed to enable a more productive, cohesive, Agile/DevOps working environment. Improvement points have been identified in portfolio management, scrum adoption, product engineering, test engineering, environment management, measurement / metrics, and tool chain adoption.

- **Culture / Creativity -** Engineering team members are caught juggling between the pressures of keeping the lights on and project execution. Team has awareness for need of adopting modern DevOps practices, but ~~don't have the bandwidth~~ / ~~expertise~~ do not demonstrate the bandwidth or proficiency to incorporate these practices in the intermediate phase.
- **Delivery / Release Model -** Adoption of a mini-waterfall sprint model, with limited agile practices incorporated related to program allocation, sizing, product demos, retrospectives, quality gates, release gates, release readiness. Release model based on sprint delivery content - requiring all registered stories / tasks to be developed / validated for a given release event.
- **Modern DevOps Platform** - Need for synergy across process and tools related to continuous workflow, transparency, and reusability that can be shared across the product lifecycle. Engineering practices related to branching, packaging / publish lifecycle management, deployment strategies, and test automation are limited / inconsistently followed. Current environment strategy does not align well with agile practices - due to shared environments (Prod-QA/PROD) and manual deployments to production.
- **Test Governance & Automation -** Overall test practice is focused on quality assurance vs. quality engineering. Test practices are manually driven, with integration and regression testing managed by single resource based on tribal knowledge. Due to lack of automation framework, delivery teams have adopted right-shift strategy with regards to testing.

Current Delivery Model



- **Portfolio / Program / Agile Methodology Overview**
  - Weekly meeting with product management team (Marketing / Telehealth) in reviewing product backlog and complete grooming exercise
  - Engineering engagement w/product teams in product strategy alignment, program categorization and priority managed in weekly call

- Product backlog managed via Jira Epic issue-types with Story issue type to manage feature set/feature
- Team leveraging some features of the agile delivery model - but not consistent in documentation, ceremonies, story sizing, capacity management, retrospectives, and release certification criteria

- **Sprint Delivery Overview**
  - Flexible sprint cadence (start, duration, end) - decision based on previous sprint target completion and set of planned stories / tasks to be delivered along with release certification activities for current sprint
  - Sprint backlog grooming occurs 2 days prior to sprint start - to finalize design, open questions, acceptance criteria
  - **Best Practices**
    - Test Governance / Lifecycle management activities managed via Jira test-management plugin mapped to Story / Tasks.
    - Enabling giving and getting early feedback throughout product life cycle
    - Infrastructure related activities (provisioning new VMs, network changes) managed via ServiceNow tickets
    - Daily Standup - 45 min between Mahathi and internal engineering team on progress, build candidates, blockers
  - **Sprint Adoption -**
    - **Day 1 -** Sprint planning occurs on 1st day of sprint - story broken down into tasks assigned to developer (TO-DO state)
    - **Day 2 - 8**
      - Developer change task state to (IN-PROGRESS) and completes task
      - Functional QA tester executes functional testing of task (if failed, no bug ticket created)
      - Functional testing pass - Developer check-in and commit change to branch
      - Developer request lead code-review, change state to (REVIEW)
      - Code Review completed - change propagated to develop branch and deploy to Staging (QA)
      - Staging QA tester validate change on Staging environment (The Offshore team does not have access to STG Environment. The current Onshore QA Engineer starts regression testing only in Prod-QA environment.)
      - Staging QA tester determine if bug creation required (not in scope), or fail task (lets confirm that is the case)
      - Engineering team determines on Day 8 - if requires more days to complete sprint or create release candidate build
    - **Day 9 - 10**
      - Release candidate build deployed to Prod-QA environment
      - Integration QA tester executes manual integration testing on release build
    - **Day 11 - X**
      - Release certification activities manually executed on release candidate build
      - If developer does not have bug - work on platform activities
      - Focus - regression testing with critical paths validated
      - Performance / Penetration testing occurs on same Prod-QA environment, which is on the same VM as production. The current utilization is max to 50%, however, future releases such as this one is in a shared environment. Also, PROD-QA activities can impact production.
      - Patching / Upgrades / Performance / Penetration / Regression testing scheduled and executed
    - **Day X - Y**
      - Sprint success rate around 90% to become a releasable product - decision made by engineering team based on business commitment, sprint challenges, and quality index (For example, there is no KPI or matrix that measures quality index. The current 45 minute daily standup calls are the only things validating pass/fail test scenarios. In addition, failed test cases about whether a known error exists or there is a production issue in the test lifecycle are transferred into the backlog.
      - Engineering team manual deployment / setup of production release activities (6am - 9am) (I think the development team is responsible for this. The engineering team only manages infrastructure tickets.
      - Smoke test of production environment to determine success / rollback (However, post production smoke tests are very limited.)

- **Tool Chain / Environment Overview**
  - **Tool Chain**
    - Source System - TFS and GIT, automated sync between vendor code base and internal engineering team
      - GIT - 5 Repos separated by applications, sprint branch created as when needed
      - GIT flow branching model - sprint based branching model, adoption of develop and main branch
    - Code Analysis - Visual Studio (ReSharper) extension leveraged on developer workstations to manage code quality, code smells, and enforce coding standards (But certain team members only enforce tailoring to the code on an infrequent basis.)
    - Build Automation - (MSbuild) build definition, dependency management and build
    - 3rd Party Library Management - (NuGet) integrated with MSBuild to retrieve shared 3rd party libraries
    - Deployment Orchestration - (Octopus) orchestration for application related changes (does not incorporate IIS configuration, registries, certificates, middleware)
    - TFS folder cleanup activities are manual.
    - Lockfornet info needed to add.
  - **Environment Strategy**
    - Environment Types - (Staging, Prod-QA, Prod) with Prod-QA and Prod sharing the same VM instances
      - Staging - based on single stack 2 VMs (App Server, DB Server)
      - Prod-QA / Prod - based on dual stack w/load balancer (2 App Server, Prod-QA DB Server, Prod DB Server) with dedicated end-point URLs
    - Release Model -
      - Production Release - Sequenced roll-over approach with one server instance updated and then released, then next server is updated

- ct Management and Engineering team allocation broken into 3 areas focusing on Product enhancements, Technical Improvements and Production Support.

- Current Prod-QA & Production resides on same VM due to firewall constrains for overall product reliability

Open Items

- Review of Jira issue types, workflows and screen schemes to determine alignment with process / practices
- Jira test management plugin review - test case management, and integration with issue types
- TFS project, branching, build, package and publish configuration
- Branching / Release / Build ID mapping - to determine where hot fix needs to be applied
- Long living projects (middleware / framework changes) - how are they managed and shared across team members
- GIT project, branching, build, package, and publish configuration
- Octopus deployment strategy, configuration and automation review, license management limitation.
- App Development - build / packaging and publish model
- Release calendar, guidelines, registration, certification activities, release readiness, and retrospectives
- High Level architecture diagram/environment topology mapping to confirm environment strategy
- Data cleanup - in production testing

Improvement Opportunities

- **Portfolio / Program / Agile Methodology Overview**
  - Portfolio / Program allocation to be broken down into 3 areas (product engineering, production support, and internal engineering initiatives)
  - Product Management / Engineering team to agree on capacity allocation (based on the 3 areas) to manage expectations
  - Engineering team to be engaged in product strategy (3 - 6 month roadmap) to align technology roadmap with product roadmap
- **Sprint Delivery Overview**
  - Adopt standard 2 or 3 week sprint model - incorporating development, testing (unit, functional, integration), sprint demo within sprint period
  - Standup - should not take more than 15 minutes
  - Release Readiness -
- **Tool Chain / Environment Overview**
  - **Tool Chain**
    - Source System - Consolidate source system - project / repo model
    - Branching Model - Incorporate feature based branching, Develop for CI, Release for release candidates, and Main for production
    - Code Analysis - Adopt code coverage, unit test analysis on Develop branch
    - Build Automation - Incorporate build / deploy automation practices beyond application changes
    - Continuous Workflow - Remove manual intervention to change ticket status with integrated tool flows
    - Developer Test - Incorporate test driven development ensuring products are developed with test automation as a pre-requisite
    - Release Certification - Adopt test automation tool, strategy and framework to automate regression testing, cross browser testing, and device testing
    - Deployment Orchestration - (Octopus) orchestration for application related changes including database, configuration, IIS settings, etc
  - **Environment Strategy**
    - Environment Types -
      - Introduce Dev-Int environment to support continuous integration / delivery capabilities before QA engagement
      - Prod-QA / Prod - separate out Prod-QA from Prod environment avoid risk of unplanned outage, access controls
    - Environment Health
      - Incorporate basic acceptance testing / environment health checks to identify non-functional impacts

Development Engagement

- ~~Limited visibility in offshore development life cycle activities which allow for predictable timelines within each project.~~
- Due to the limited transparency in the vendor offshore development team, the product delivery timelines are impacted for each project.