Sample E² QA Assessment

AGENDA

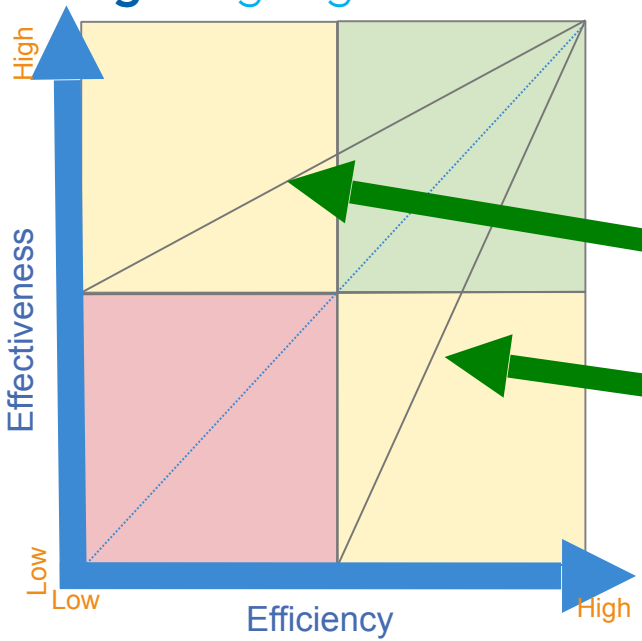
- The Assessment Methodology
- QA Assessment Results
- Gap Analysis & Road Map
- Q & A
- Appendix – Assessment Details

AGENDA

- The Assessment Methodology
  - Areas of Assessment
  - Measures
  - Assessment Effort
- QA Assessment Results
- Gap Analysis & Road Map
- Q & A
- Appendix – Assessment Details

Areas of Assessment

- Effectiveness – **Doing** the right things
- Efficiency – **Doing** things right



Pitch of Corrective Direction Provides Primary Focus

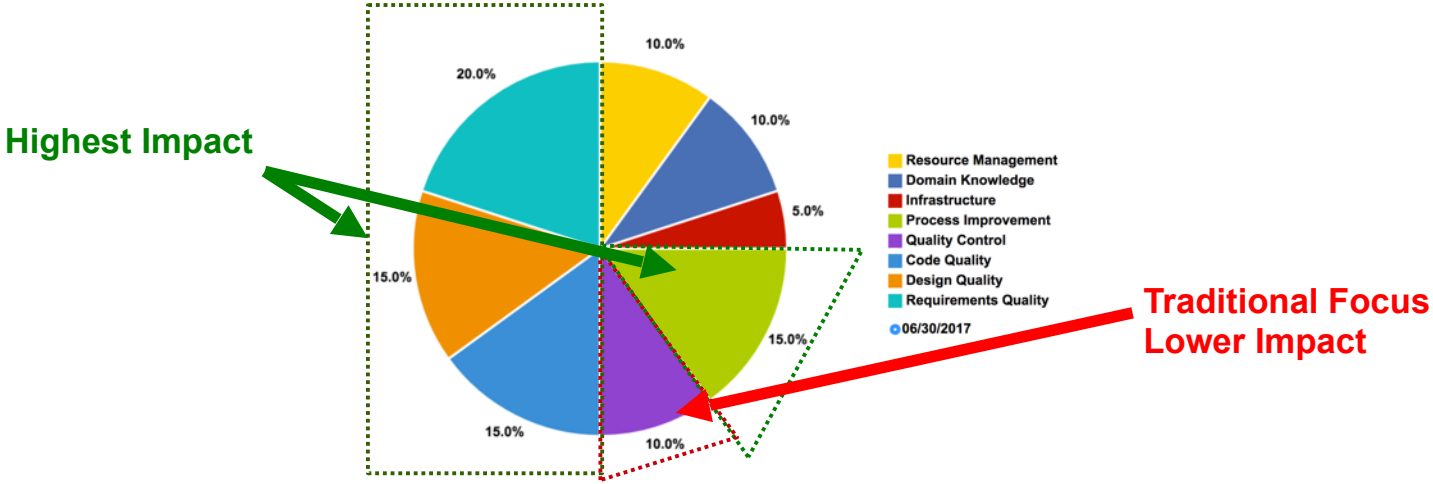
Efficiency Primary Focus vs.

Effectiveness Primary Focus

THE ASSESSMENT PROCESS - METHODOLOGY

Measurements

- **Requirement Quality** (20%)
- **Design Quality** (15%)
- **Code Quality** (15%)
- **Quality Control** (10%)
- **Process Improvement** (15%)
- **Infrastructure** (5%)
- **Domain Knowledge** (10%)
- **Resource Management** (10%)

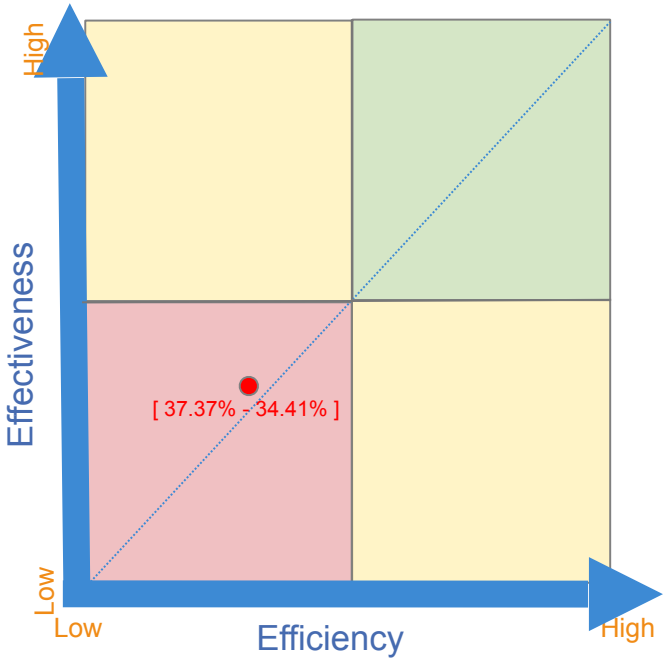


During the assessment we:

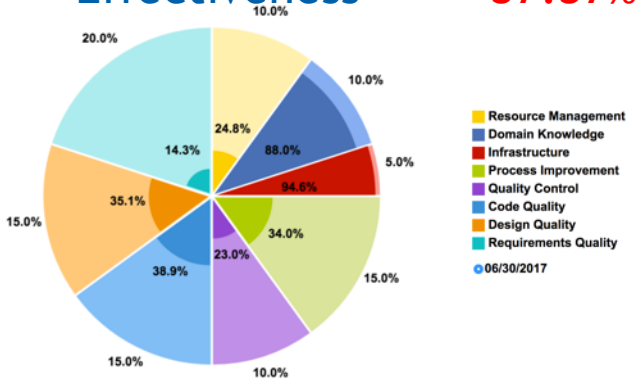
- Conduct Interviews with QA Team Members
- Conduct Interviews with other Team Members
- Review Team Artifacts
- Participate in Meetings
- Review Reports
- Review Tools
- Review Automation Effort and Practice
- Review Team Coordination
- Review Team Collaboration

EXAMPLE RESULTS – QA EFFECTIVENESS/EFFICIENCY SCORE CARD

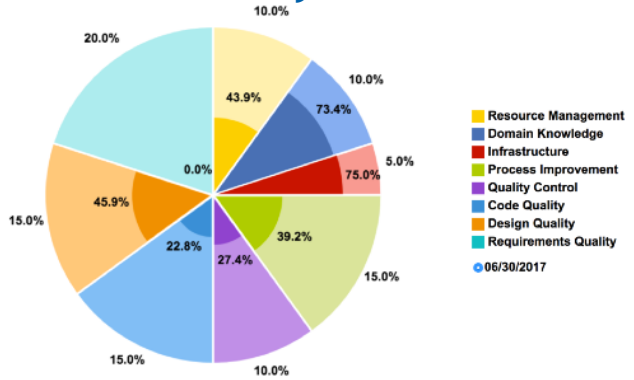
Sample



Effectiveness 37.37%



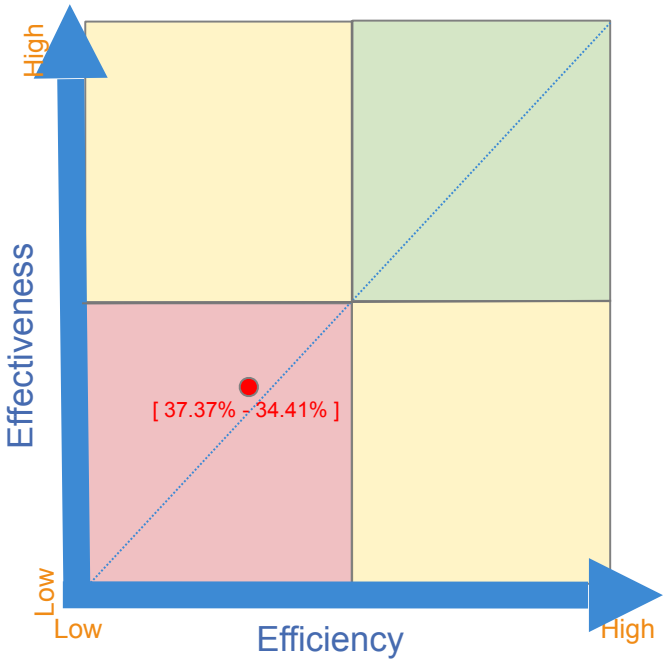
Efficiency 34.41%



SQA²

EXAMPLE RESULTS – QA EFFECTIVENESS/EFFICIENCY SCORE CARD

Sample



- 37.37% Effectiveness Factor
  - \+ Strong documentation tools & culture helps
  - \+ Well defined infrastructure helps
  - \- Emphasis is on finding defects
  - \- Little to no focus in preventing defects
  - \- Large gaps in process
  - \- No API Testing
- 34.41% Efficiency Factor
  - \+ Good cross browser testing coverage
  - \- No enterprise standard process
  - \- Large focus on Critical rather than Important tasks
  - \- Automation not in QA's hands

SQA²

EXAMPLE RESULTS – QA EFFECTIVENESS/EFFICIENCY SCORE CARD

Scored high in the following:

- Domain Knowledge – QA team has established strong practice of providing domain knowledge
- Infrastructure – Environments are cleanly delineated and easy to setup with the deploy tool

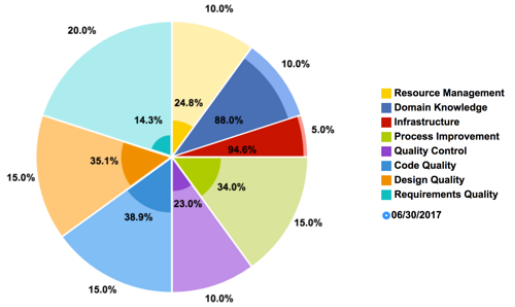
Each of these areas have lower influence factor

Scored low in the following:

- Requirements Quality – QA team does not routinely participate in early review of requirements
- Process Improvement – Large amount to be improved. No metrics kept.
- Design Quality – No QA involvement
- Code Quality – Low QA involvement

Each of these areas have highest influence factor

Effectiveness 37.37%



Measurement	Score
Resource Management	24.8%
Domain Knowledge	88%
Infrastructure	94.6%
Process Improvement	34%
Quality Control	23%
Code Quality	38.9%
Design Quality	35.1%
Requirements Quality	14.3%

SQA²

EXAMPLE RESULTS – QA EFFECTIVENESS/EFFICIENCY SCORE CARD

Sample

Scored high in the following:

- Domain Knowledge – Team has strong understanding of domain knowledge and documents it well
- Infrastructure – Environments are cleanly delineated and generally well understood

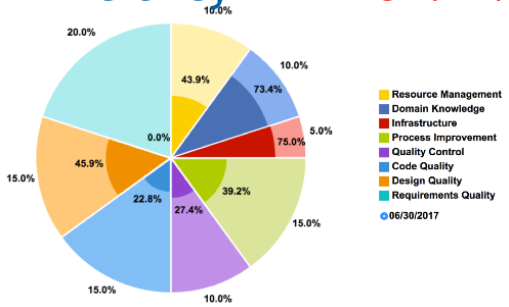
Each of these areas have lower influence factor

Scored low in the following:

- Requirements Quality – Test cases barely cover requirements. No critical analysis done.
- Process Improvement – No QA involvement
- Code Quality – Low QA involvement

Each of these areas have highest influence factor

Efficiency 34.41%



Measurement	Score
Resource Management	43.9%
Domain Knowledge	73.4%
Infrastructure	75%
Process Improvement	39.2%
Quality Control	27.4%
Code Quality	22.8%
Design Quality	45.9%
Requirements Quality	0%

SQA²

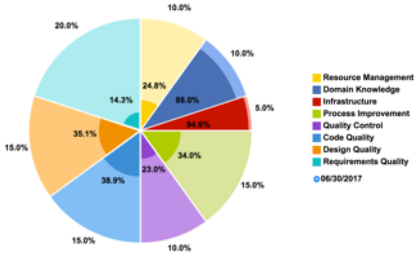
EXAMPLE RESULTS – QA EFFECTIVENESS/EFFICIENCY SCORE CARD

Sample

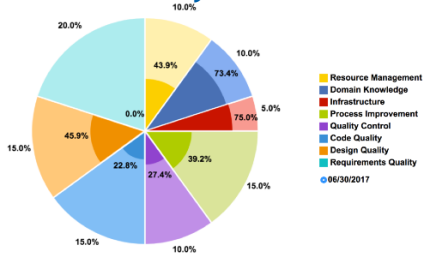
Comparatively:

- **Resource Management** – Team members routinely focus on critical rather than important activities. OT and weekend work is common. This leaves no room for growth and development. Result: This will burn out your QA associates.
- **Quality Control** – Typically, Quality Control tends to be higher for most organizations. All scores were significantly impacted by the lack of API testing, exploratory testing, performance testing, bug deep dives, edge case testing, adequate test case identification, and not reviewing test cases with business and developers. Result: Contributes to the high number of defects in live code.

Effectiveness 37.37%



Efficiency 34.41%



Measurement	Effectiveness Score	Efficiency Score
Resource Management	24.8%	43.9%
Domain Knowledge	88%	73.4%
Infrastructure	94.6%	75%
Process Improvement	34%	39.2%
Quality Control	23%	27.4%
Code Quality	38.9%	22.8%
Design Quality	35.1%	45.9%
Requirements Quality	14.3%	0%

SQA²

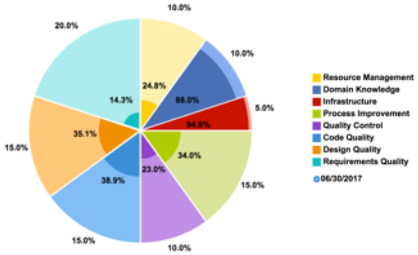
EXAMPLE RESULTS – QA EFFECTIVENESS/EFFICIENCY SCORE CARD

Sample

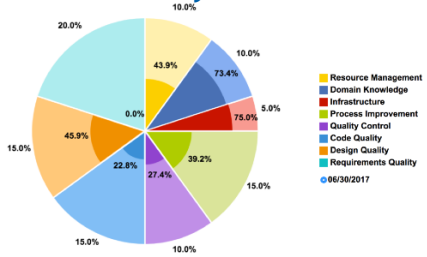
Comparatively:

- Code Quality – Very little involvement of QA in ensuring standards are in place and being followed by development team. Code Quality not considered in test case process – no review of test cases with developers. Result: This contributes to higher defect counts.
- Requirements Quality – With the lack of Effectiveness, there is little to no opportunity for Efficiency. Of the very few processes that do exist (Doing the right things), poor execution on those processes (Doing things right) resulting in a low score. Requirements Quality not considered in test case process – no review of test cases with product owners. Result: This contributes to higher defect counts.

Effectiveness 37.37%



Efficiency 34.41%



Measurement	Effectiveness Score	Efficiency Score
Resource Management	24.8%	43.9%
Domain Knowledge	88%	73.4%
Infrastructure	94.6%	75%
Process Improvement	34%	39.2%
Quality Control	23%	27.4%
Code Quality	38.9%	22.8%
Design Quality	35.1%	45.9%
Requirements Quality	14.3%	0%

SQA²

Gap Analysis

- QA teams need to improve both in Effectiveness & Efficiency
- Automation by the QA associates needs to be established
- Areas of improvement with greatest opportunity for gains are:
  - Requirements Quality
  - Design Quality
  - Code Quality
  - Process Improvement

Quality control will be improved through these four, but QC is lower than we're used to seeing and needs work as well

Road Map

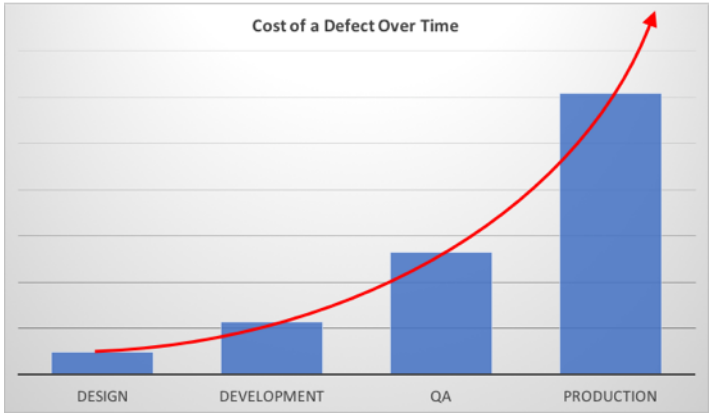
- Establish New QA Defining Principles
- Execute QA Process over Testing Practice
- Establish QA Support Model
- Establish Automation Effort
- Facilitate Continuous Improvement Practice for QA Team

Defining Principles

- Become a full SDLC QA Organization
- Automation success is achieved by a QA mindset
- QA drives Agile alignment and best practices
- Small, Incremental, Measurable Continuous Improvement

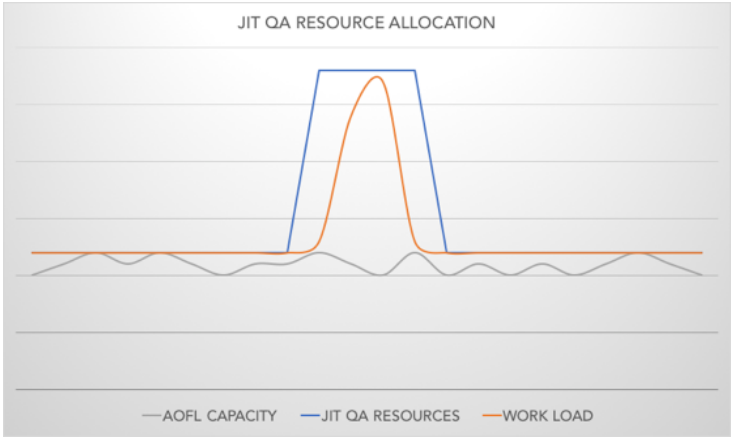
QA Practices

- Core QA Team Focus on QA Practices
  - QA Team needs to think like **QA**, not testers
  - QA Team participates in the following areas:
    - Requirements Quality
    - Design Quality
    - Code Quality
  - Remember:
    - Cost of a defect



Support Model

- To Facilitate QA Focus on QA Principles
  - Current 2:1 Dev to QA Ratio Too High
    - Not Sustainable
  - Implement JIT QA Resources
    - Just-in-time QA / Testers
    - Available High-Burst Period



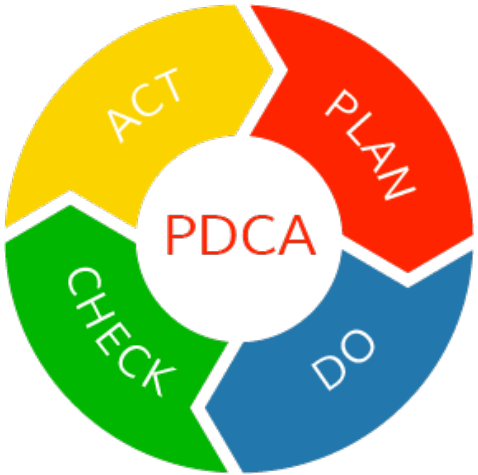
  - Free up QA associates to focus on High Value
    - Bring in QA contractors to focus on repetitive, time-consuming activities
    - Focus on process and improving it
    - Focus on Automation Framework

Automation Effort

- QA associates to lead and manage the automation effort
  - Best growth path for QA professionals
  - QA Contractors can free up QA associate's time to grow in automation
  - SDET path not recommended (Mismanagement of resource, upside down value to cost)
- Adopt Maintainable Automation Framework
  - Framework based on BDD
    - Behavior Driven Development
  - Adopt QA Centric Automation Framework
    - Behavior Based Testing

QA Continuous Improvement

- QA Team to Establish Continuous Improvement Practice
  - Plan / Do / Check / Act
  - Manage Like Scrum
    - Backlog
    - Stories in Sprint
    - Retro



AGENDA

- The Assessment Methodology
- QA Assessment Results
- Gap Analysis & Road Map
- Q & A
- Appendix – Assessment Details

Q & A

Appendix

QA Assessment Details

- Description of Current State
- Adherence to QA Practices
- Completeness and Efficiencies of Test Cycles
- Test Capabilities
- Automation Capabilities and Test Tools
- Application Lifecycle Management
- JIT Testing Resources
- Training Needs
- Captured Concerns

Current State

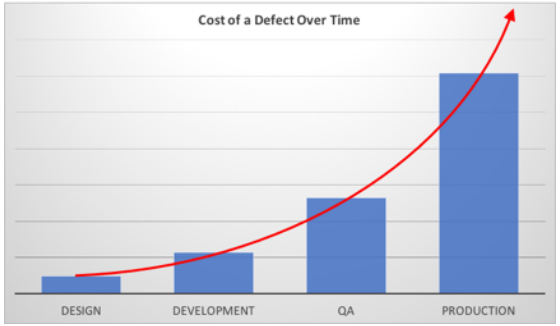
- 4 Scrum Teams with upwards of 2:1 Dev to QA ratio
  - Practicing Scrum/Kanban with 2 week sprints
  - Focus on projects for primary app behind paywall
    - Supporting US release

Current State

- 2-3 QA associates per scrum team
  - Primarily focus on QC (Testing) efforts
  - Little to no process implemented
  - No automation taking place
    - Some team members have expressed interest
- QA Analysts doing tester work
  - Need support so they can focus on high value work
    - Automation
    - Process Improvement
    - Bug Deep Dives
    - Exploratory & Edge Case Testing
    - Performance Testing

Adherence to QA Practices

- QA associates to serve QA role
  - Prevent defects, not just find them
  - Cost of defect
- QA practices call for an organization to continuously improve process
  - QA team not engaged
- Automation belongs with QA
  - Current team is not engaged in automation
  - Team should be building and maintaining automation
  - Right mindset for automation vs. dev alternative



Completeness and Efficiencies of Testing Cycles

- For Quality Control (testing) the team is not effective
- Areas of concern:
  - Little to no process
  - No process improvement
  - Only detecting defects / No defect prevention
  - No API testing taking place
  - Very little effective performance testing taking place

Automation Capabilities / Test Tools

- Enterprise solution should be implemented
  - Minimizes coding
  - Maximizes coverage
  - QA Associates expressed interested in learning
- QA associates to focus on learning and implementing automation
  - Support resources take on mundane work
  - Will achieve increased productivity

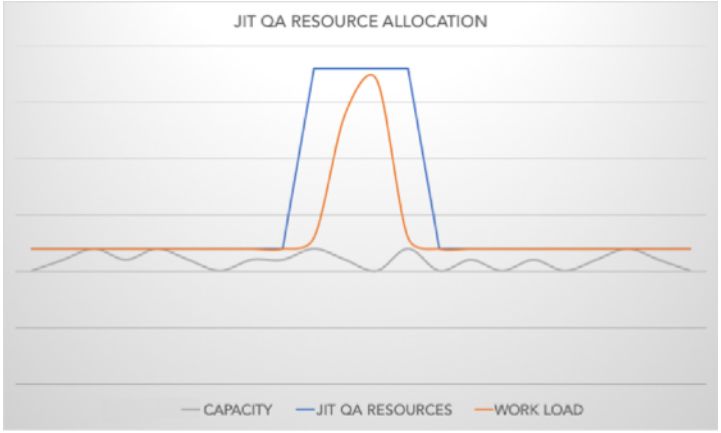
Application Lifecycle Management

- Currently use a mix of multiple tools for ALM:
  - Trello
  - Jira
  - Spreadsheets
  - Confluence
- One tool should be selected and used organization wide
  - **Jira** can handle all involved aspects ALM-based activities
    - One source of record
    - Metrics easily captured
    - Reports and trends easily generated
    - Integration with many other tools (Gitlab, automation reporting, etc.)
  - Confluence continue to handle documentation

Just-In-Time Testing Resources

- QA Efforts fluctuate given the number of projects QA supports
- QA team needs a flexible outsourcing model that allows for ebbs and flows of demand on QA
- Just-In-Time (JIT) Testing is a solution that combines on-site QA contracts that optimize demand for JIT resources
- Enough JIT Testing knowledge is built to support bursts in demand

Conclusion: Implement JIT Testing solution to allow core QA team to focus on QA.



Training Needs

- Core QA is a solid group of QA professionals
- Core QA team needs to transition towards primary QA role versus testing role
- Mentoring is needed to help QA team focus on QA responsibilities
- Mentoring will facilitate implementation of Continuous Improvement focus on QA activities

Conclusion: Training should be conducted in a mentoring model.

RESULTS – CAPTURED CONCERNS (RESOURCE MANAGEMENT)

Captured Notes	Comment
No Continuous Learning	• No training on new technology • No new skill set development • Team members will become stagnant in their skills
QA value reporting	• No regular capture of QA-centric metrics • No daily reporting of value generated by QA team to management • Will lead to misunderstood costs in QA
Iteration Planning	• Half of meeting used on non-planning activities • Expensive use of an entire team's worth of time • Unorganized and non-centralized planning made use of Trello, Jira, and spreadsheets at the same time • Multiple discussions flare up over each other without being stopped
Quality of Life	• Regular request of team members to commit to overtime • Team members asked to come in over the weekend to work for releases
Unrealistic timelines	• US testing on the weekend indicates poor timeline planning

RESULTS – CAPTURED CONCERNS (INFRASTRUCTURE)

Captured Notes	Comment
Testing Coverage	• No testing performed on anything other than GUI • Must consider API, performance testing, integration testing, etc.

RESULTS – CAPTURED CONCERNS (PROCESS IMPROVEMENT) P.1

Captured Notes	Comment
Undeveloped Process	• Many missing key processes that will heavily boost success of projects
Enterprise standard process	• Efforts in each scrum team are independent with no enterprise standard • No documented enterprise standard
Right-fit methodology	• Moving in the right direction with selection of Scrum • Still a lot of work to do to get close to an efficient Scrum process
Scrum meetings ineffective	• Scrum meetings do not follow proper format • Occasionally exceed 15 minute limit • Multiple Scrum meetings overlap, limiting management involvement
QA value reporting	• No end of day updates • No measurement and reporting of QA efforts to management
QA is limiting velocity	• This is backwards. QA has become a roadblock.
Swarm on QA work	• The need for Development and POs to pitch in on QA indicates productivity/resource issues

RESULTS – CAPTURED CONCERNS (PROCESS IMPROVEMENT) P.2

Captured Notes	Comment
Regular Communication of completed tasks	• QA value is not communicated on a daily basis • Scrum meetings are only time where completed work is shown
Dev/QA communication breakdown	• Devs often hand off iterative work, but don't let QA know what's done • Results in defects written for missing features that are not built yet

RESULTS – CAPTURED CONCERNS (QUALITY CONTROL)

Captured Notes	Comment
No Backend Testing	• QA only works in the GUI • No API testing occurring **(LARGE RISK!)**
Documentation of testing	• No evidence of "How-to's" documented in Confluence • Testing results are documented in spreadsheets
Critical vs. Important	• Team constantly working on Critical work • No time left for Important work like exploratory testing, performance testing, bug deep dives, edge case testing **(LARGE RISK!)**
Test Case Coverage	• No quantifiable measure of test case coverage
Test Case Process	• QA writes just enough test cases to cover requirements **(LARGE RISK!)** • No review of test cases with POs or development **(LARGE RISK!)** • No demo of stories to PO at a story-by-story basis
Defects in Production	• Jira shows large number of defects found in the live product • Direct result of all concerns, but majorly influenced by large risks noted above

RESULTS – CAPTURED CONCERNS (CODE QUALITY)

Captured Notes	Comment
QA & Dev coordination	• Developers do not review QA test cases • Code reviews not occurring • Coding standards are not followed • Developers not utilize patterns • No unit testing/integration testing
Lack of Automation	• No automation occurring
Large story breakdown	• Large stories occasionally get broken down into pieces where some are “untestable” according to team members
Performance Testing	• No effective performance testing is done on anything
Large defect output	• High volume of defect output per iterations (20-60 depending on type of stories in the iteration)

RESULTS – CAPTURED CONCERNS (DESIGN QUALITY)

Captured Notes	Comment
QA Involvement	• QA not involved in design process • QA not considering design when creating test cases • QA needs to provide input in design quality so that testing can be improved
Design Reliability	• Live support issues are common

RESULTS – CAPTURED CONCERNS (REQUIREMENT QUALITY)

Captured Notes	Comment
Test Case Identification	• Thorough test case identification not occurring
Review with Business	• No business review of test cases
Test Case Coverage	• Test cases only cover ”just enough” of the requirements • When team swarms on QA, requirements serve as test cases
Requirements Documentation	• Team often gets requirements that have no acceptance criteria • 40% of the time, requirements are changed after work has begun • Team is occasionally blocked by poorly written requirements