Service confidence...because code coverage and CRAP scores just don't give you the whole picture

Don Kidd

dkidd@miamioh.edu

Abstract

At Miami University, we have four to six software development teams responsible for the applications and processes that are written to support the members of the Miami community. Formal testing practices were very new to the organization, but we wanted to get an idea of the quality of our applications. One strategy would be to conduct a static analysis of code to obtain a score, but since we are in higher education, we wanted something more. Thus, we developed the idea of "Service Confidence"—as a group of stakeholders, we discuss and score three factors for our services: Context, Confidence, and Complexity. While the grade that we give our applications is important, we have discovered that the discussion determining the grade is just as important. I will be sharing how these conversations are going, and what we are learning as we go through this scoring exercise of Service Confidence, along with some timing considerations when planning on scheduling these discussions. While this doesn't fully replace the quick analytics obtained from code static analysis, we can show that there is more to Quality than CRAP scores and code coverage.

Biography

Don Kidd was a software developer for 16+ years and then made the switch to the quality side about 6 years ago, where he is now tasked to improve quality efforts at Miami. He is currently working with various stakeholders at Miami to understand what "Quality" means to them, so that the IT department can design and implement quality measures to reach those objectives. He also coaches software delivery teams on tools and practices to help build solutions they can be confident in, while making the users of these systems more efficient in the work they are doing.

1. Introduction

Quality is a word that is used everywhere. In almost every situation, someone is always inquiring about the quality of something. If you are buying a car, I'm sure that you want a quality one. If you are selling something that is handcrafted, you say that it is "quality crafted." When a company creates software, they want the software that is developed to be a quality solution.

When I began my position as a Quality Analyst, I was the first such position at Miami, and it was really the first time in our organization that we decided that quality was an important attribute of the output from our organization. So, what does quality mean, and how am I going to measure the level of quality we are producing in our organization?

2. Problem: What Determines Quality?

If I were to be responsible for quality, I needed to get an understanding on our current quality so that I can help us improve it.

Our IT group writes and maintains all sorts of software. Some would be considered a small application (written normally in PHP), and we have an assortment of other scripts and software written in various languages, such as Perl, Python, PL/SQL, etc. (I think we finally got rid of our last COBOL script a few years ago!). The biggest challenge I had was how to measure the quality of each of these services. To simplify this range of development, we refer to each solution that we generate as a service. We define a "service" as an application or solution component created, deployed, or operated by our department that directly or indirectly fulfills user needs.

No matter where you look, everyone says they make quality widget, or their widget is made with quality parts. But what does that mean? Who is defining quality? According to the Merriam-Webster, Quality is "an inherent feature or a degree of excellence." and the Oxford English Dictionary, says that is "the standard of something as measured against other things of a similar kind; the degree of excellence of something." So that helps us get a basic idea of quality, but what are these "similar kinds"?

One simple way was to start was for us to create the groups of "similar kind" based on the language our "services" are written in.

3. Language-Based Groupings

By grouping our services by language, we can run a variety of tools specific to the language that the service was written in.

3.1. PHP

- Unit tests and feature tests are run to check on the code to make sure that the tests pass and to see how much of the code is covered with the tests.
- The Change Risk Anti-Patterns (CRAP) Index is calculated based on the cyclomatic complexity and code coverage of a unit of code. Code that is not too complex and has an adequate test coverage will have a low CRAP index. The CRAP index can be lowered by writing tests and by refactoring the code to lower its complexity.
- The code run is analyzed using PHPCS, which is a code sniffer that reviews and detects violation of coding standards.

3.2. Shell

- Shell Check is used to find basic bugs in the shell script.

3.3. Perl

- Perl::Critic is used to apply coding standards to Perl and check to see which standards are followed.

3.4. What we learned

The great thing about tools based on language is that there are already tools built and scripts that can be run to do the analysis on our code to get values that we can use and compare. The downside of this grouping is that we can only compare things that were written in the same language, and we would like to look at services that are developed regardless of the language.

4. Quality Grades

The language-based analyses of quality seemed like a good start and one that would work for each specific language, but I was looking for a way to compare services regardless of the language they were written in. While those values generated from language-based analysis will help and are a factor in assessing quality, there must be a way to start comparing all our services.

4.1. Testing Quadrants

As a group that is attempting to develop using an Agile mindset, I started doing research on Agile testing and how that might fit into our structure. Crispin and Gregory (2008) put forth the concept of Agile "Testing Quadrants," which I investigated and tried to apply to the academic setting.

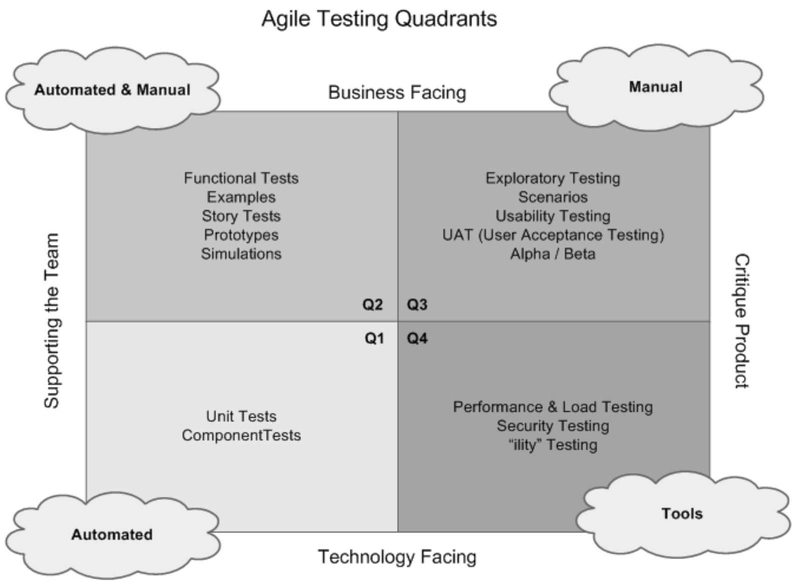


Image Source: http://agiletester.ca/

While the quadrants are really guides to help the teams talk about testing, it got me thinking. A lot of our PHP applications teams were beginning to focus on items in Q1 & Q2, specifically, the Unit Tests; they were also starting to get into more Function Tests. Both are great, but if we would test from all 4 quadrants, then wouldn't it be a great piece of software? How could I encourage the development teams to start spending time on in Q3 and Q4 as well? If an application were to be tested in all the Quadrants, it would likely be a high-quality application, but if our application isn't used by many people, is it worth the effort to spend the time to create all the tests?

4.2. Testing Maturity Model

Our Vice President of Technology at the time was very focused on Capability Maturity Models (CMM). I was wondering if there were something similar related to testing, and came upon the Testing Maturity Model (TMM). While it was originally developed in 1996 at the Illinois Institute of Technology, it is an idea that is still quite relevant today. The TMM clearly outlines the levels that your testing follows. Each level adds upon the success of prior levels; you can't just jump to Level 4 without first working on Level 2. If an application is in Level 5 of TMM, then some might consider it to be a high-quality piece of software. Like the four testing quadrants, achieving TMM Level 5 may be great—but does everything need to be a Level 5? There needs to be a discussion as to what TMM Level of software is expected.

Since I work at an institution of higher education, we are always talking about grades and how students are always trying to achieve an "A" in a class, but what if a "B" is good enough? A good student could get a "B" if they determined that the cost to get the "A" was too much, or maybe they didn't need the "A" to maintain a particular GPA. When you combine the ideas from the testing quadrants and convert the concept of TMM into a grading scale, Grading of Services is created.

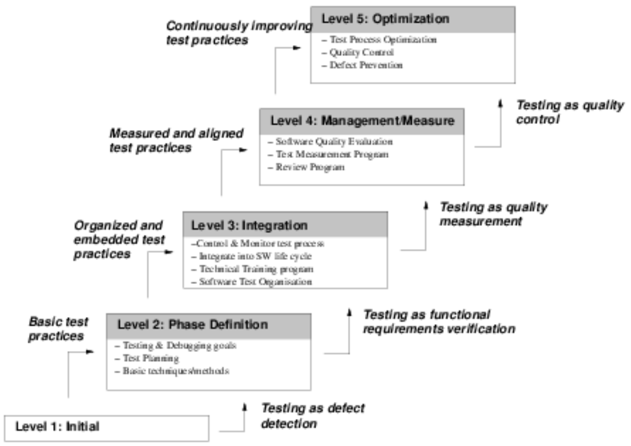

Continuously improving test practices
Level 5: Optimization
– Test Process Optimization
– Quality Control
– Defect Prevention
Testing as quality control
Measured and aligned test practices
Level 4: Management/Measure
– Software Quality Evaluation
– Test Measurement Program
– Review Program
Testing as quality measurement
Organized and embedded test practices
Level 3: Integration
–Control & Monitor test process
– Integrate into SW life cycle
– Technical Training program
– Software Test Organisation
Testing as functional requirements verification
Basic test practices
Level 2: Phase Definition
– Testing & Debugging goals
– Test Planning
– Basic techniques/methods
Testing as defect detection
Level 1: Initial

Figure 1: Maturity levels and goals of the Testing Maturity Model TMM.(Moll 2000)

4.3. Grading of Services

Since I work in an institution of higher education, I decided to use a grading scale to help us talk about quality. Now, we may always strive for an A, but getting one is not always an achievable goal. There may be other things that are of greater value then getting an A. Maybe it was determined that getting an A was just too hard, so we focused on getting a solid B grade. I've tried to create the chart below as a way for us to discuss the grade of quality that we are striving to get.

Our confidence in our code can be measured in many ways, from a CRAP (Change Risk Anti Pattern) score to Unit Test code coverage. While it is true that we should always try and improve the quality of our code and strive for "*perfect, bugless code,*" that is not an achievable goal. There are times when the cost to find or prevent bugs outweighs the value of the bug being searched for.

In addition to the confidence level, we also need to gauge the usage of this service. If this Service is going to be used by a lot of users, then we should spend a lot of time on the service to get as good of a grade—and be as confident in our code as we can. But perhaps, if this solution will only be used by a few people, do we need to strive as hard to have high confidence in our solution. Not that it should be terrible and inefficient code, but it shouldn't take inordinate amounts of time and effort to improve it.

I've created this chart to help us visualize the grade of quality in our code. This is by no means a definitive chart or gauge to our level of quality, but two axes of data we can use to help us think about a desired level of quality in our application.

	Small Audience to use Service	Medium Audience to use Service	Large Audience to use Service
High Confidence in Change	A	B	C
Medium Confidence Change	B	B	C
Low Confidence in Change	C	C	D

4.4. What we learned

While this was a good concept overall, we needed more than just letter grades to show our confidence. What went into our confidence grade? We felt like we needed more points of data to help us accurately calculate the confidence. After many discussions with team members and my manager, we came up with the Service Confidence Framework.

5. Service Confidence Framework and Process

The Service Confidence Framework describes processes intended to generate conversation and visualizations related to our confidence in sustaining services operated by the department. The framework is intentionally "loose," providing suggested signals to consider when discussing our level of confidence in a service, but not prescribing a fixed set of indicators. The value of this framework is in the conversation and resulting insights.

Our IT department defines a "service" as an application or solution component created, deployed, or operated by our department that directly or indirectly fulfills user needs. This may be an application directly accessed by end users, such as CAS(Central Authentication Service), or it may be APIs that are consumed by other applications. A key characteristic of a service is that we make changes to it, and failures resulting from changes have an impact to users.

We understand that "service" is an overloaded term in our industry. For the purposes of this framework, we suggest a pragmatic approach, with a goal of enabling useful conversation without overburdening the organization. For example, scoring “authentication” as a service is likely to be an effort in frustration, whereas scoring CAS and Shibboleth would be appropriate. Some services may also only make sense in aggregate. When considering Canvas, for example, it makes sense to combine the hosted platform with our data integration scripts.

The Service Confidence Framework describes three factors for a service:

- Context - the impact of the application on the University
- Confidence - our trust in the service reliability and our ability to restore the service in the event of an outage
- Potential impact - combines the context and confidence to represent likely "blast radius" of a disruption

When considered together, these factors provide a useful summary which allows comparison of multiple services, identifies critical services with low confidence, and informs efforts to improve service stability.

There are three primary activities in the Service Confidence Framework process: score, visualize, and act.

5.1. Score

The process of scoring services is not highly prescribed, but it is also not lightweight. The goal is not to simply have a checklist where checking 90% of the boxes gets you 9 out of 10 on the scale. A wide range of signals influence our confidence. Not all apply to every service or have the same weight between services. The framework provides examples of signals to consider when scoring, but the participants should consider any relevant indicators.

The framework requires three scores be generated for each service: context, direct confidence, and complexity. The potential impact value is calculated from these scores. These are then combined by an algorithm to produce the final context and confidence scores. All scoring is done on a scale from 1 (low) to 10 (high).

The framework does not describe who should participate in or the exact procedure for scoring a service. We can offer some general guidelines, however.

- Diverse perspectives are critical, so consider developers, operations, infrastructure, and anyone with responsibility for how the service is developed, deployed, and operated.
- Consider including business stakeholders but beware that they may overestimate the business criticality of services.
- Experience tells us that this scoring conversation is the most valuable part of the process, so ensure an open approach and document the conversation and outcomes.
- Operations of and changes to a service may be different (a service can be very stable except when changes are being made), so consider both while scoring.
- Consider reviewing the scoring outcomes for similar services (if available).

5.1.1.Context

The service context describes the value and availability needs of the service from the user perspective. In some cases, these will be direct (when users directly interact with the service) and in other cases, indirect (the service supports other services). The context of a service can be thought of as factors outside of our control.

Signals to consider for context include:

- Frequency of use
- Number of users per unit of time
- Number of times used per unit of time
- Business function criticality

One of the more interesting signals we should consider is “time to critical." In other words, how long will the business tolerate the service being disrupted? The concept of "time to critical" encapsulates many of the other signals and provides a useful overall measure. However, the "business" must be defined as the institution rather than a single unit.

5.1.2.Direct Confidence

The direct confidence in a service is determined by operational characteristics from development through runtime. Signals to consider for direct confidence include:

- Efficiency of resource utilization
- Automated test coverage
- Resiliency
- Change management
- Documentation (runbooks, operational procedures)
- Observability (monitoring, alerting, logs)
- Positive track record (change success rate, incident rate)

The direct confidence of a service can be thought of as factors we have explicit control over, at least to some extent, and the presence of which is beneficial.

5.1.3. Complexity

The complexity of a service represents how hard a service is (to understand its exposure) as well as external factors. Complexity acts as a detractor against direct confidence (more complexity results in lower confidence). Signals to consider for complexity include:

- Code complexity (i.e., CRAP scores)
- Dependencies (number, stability, up to date)
- Influencers (shared services)
- Architecture
- Deployment process (automated, etc.)
- Integrations (diversity, sources, and sinks)
- Contracts with other services

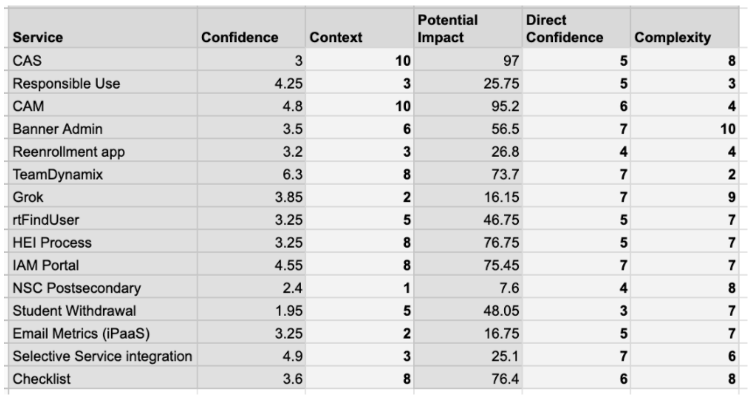
The complexity of a service can be thought of as factors we can influence but not necessarily control.

5.2. Visualize

Once scoring is complete, we will record the service scores in a common location so they can be visualized and compared.

5.2.1. Record Scores

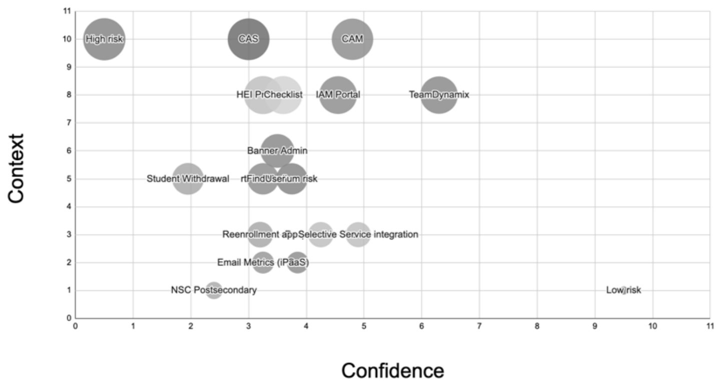
The first iteration of the Service Confidence Framework uses a Google Sheet to record scores. A Google Sheet provides a simple, sharable solution while we evolve the process.



Service	Confidence	Context	Potential Impact	Direct Confidence	Complexity
CAS	3	**10**	97	**5**	**8**
Responsible Use	4.25	**3**	25.75	**5**	**3**
CAM	4.8	**10**	95.2	**6**	**4**
Banner Admin	3.5	**6**	56.5	**7**	**10**
Reenrollment app	3.2	**3**	26.8	**4**	**4**
TeamDynamix	6.3	**8**	73.7	**7**	**2**
Grok	3.85	**2**	16.15	**7**	**9**
rtFindUser	3.25	**5**	46.75	**5**	**7**
HEI Process	3.25	**8**	76.75	**5**	**7**
IAM Portal	4.55	**8**	75.45	**7**	**7**
NSC Postsecondary	2.4	**1**	7.6	**4**	**8**
Student Withdrawal	1.95	**5**	48.05	**3**	**7**
Email Metrics (iPaaS)	3.25	**2**	16.75	**5**	**7**
Selective Service integration	4.9	**3**	25.1	**7**	**6**
Checklist	3.6	**8**	76.4	**6**	**8**

5.2.2. Plot Services

While the scoring conversation is a valuable way to understand our confidence in an individual service, we gain additional insight by comparing services. The first iteration of the framework is a bubble chart plotting context and confidence on the axis and using potential impact as the bubble size.



5.3. Act

With the services scored and visualized, we can gain insight into actions which will reduce risk and improve operational outcomes. We consider two aspects: a "zone" of services in a high-risk situation, and strategies for shifting service confidence.

5.3.1. The "Zone"

Our high-risk "zone" can be considered the upper left quadrant of our bubble chart. That quadrant contains services with high context and low confidence scores, indicative of service at high risk of disruption. Within that zone, we could further refine our view by considering services with a higher potential impact as being of higher priority to address.

We should note that being in the zone, while not desirable, is not in and of itself bad. Some services may simply be of such high context, and our ability to change the direct confidence or complexity so constrained, that it is not practical to move. This should not be seen as a negative, only an indication that we need to take extra precautions with that service.

So, what does it mean to be in the zone? Being in the zone implies a high context, so the business would not tolerate significant disruption, as well as low confidence for changes or operations. In these cases, changes may require extra scrutiny and planning; responsibilities may not "shift left" for operations, or any number of other stipulations to reduce potential disruption.

Since there are consequences for a service in the zone, we should take opportunities to make changes that result in services moving out of the zone when possible. We said previously that we can't really change a service's context, so that leaves direct confidence and complexity as means to move a service. There are any number of ways to improve direct confidence, including improved testing, writing operational documentation, or improving the deployment process. Complexity changes may take more effort, but they also realize significant improvements overall.

5.3.2. Improve

The overall picture of our service confidence scores helps us build a roadmap for improvement. We can assume that increasing our confidence will lead to better outcomes for users and lower operational burden for us. We can use the output of the Service Confidence Framework to identify services we should target for improvement and leverage the scoring conversations to select specific improvement activities.

For example, we may wish to improve our confidence in a service with moderate context. The benefits would be reduced overhead in the change process (moving out of the zone), willingness from the business side for more frequent updates, or any number of things. The conversation around scoring may indicate that the service lacks automated testing and operational documentation, both easily actionable improvements and within the ability of Solution Delivery to carry out.

6. What we learned

Utilizing the Service Confidence framework, we are able compare a variety of the services that we support, regardless of the language. We've been able to look at different services that have been created using different languages. We can leverage the Service Confidence scores and discussions to determine where more effort might be needed to improve the confidence of our services.

6.1. Conversation

While talking about what the appropriate values for Context, Complexity, and Direct Confidence, often, very good conversations occur. During these conversations, we have learned about documentation that was previously hidden in a department shared drive. When discussing our Direct Confidence of a service and talking through some of the signals, we were able to ask questions to the development team about their service. The conversation brought to light a scenario that the team had not yet thought of. It allowed them to add stories to the project backlog and add features to improve the confidence in their service.

6.2. Timing

One thing that really surprised us was that the timing of the Service Confidence discussion can have different results. We were under the assumption that the best time to go over the Service Confidence was towards the end of the update to the service, or when the service was almost complete. In some situations, we had the discussion earlier, and this allowed the team to have a conversation about their design earlier in the process with other stakeholders, which allowed questions to be raised and discussions to be had that would allow the confidence in the service to be improved. But if the conversation were too early, some signals that are discussed might not yet be complete, and the score would be lower at that time. This was very hard for some teams to accept because they felt that the service they were working on should have a higher score than it did.

Given these observations, we are still trying to determine the best time within the development lifecycle to have these discussions.

6.3. Overall

While each application is given a confidence score, this is in no way a judgment of the application. Our goal is to map out the services that we provide and help the organization understand the relative strengths and weaknesses that exist. This will eventually help us know where to allocate resources so we can strengthen the services that most need it.

7. Conclusion

As I began the position of Quality Analyst at Miami University, I determined that we needed a way to quantify the quality of services that we were developing for use across the university. We were developing services that were very different, yet we needed a way to have all scored in a similar manner. Some of the services that we developed might only be used to help a single office of three to five people. Then there were others that would be used by all faculty and students across the entire University, which might be 23,000 people. With the large variety of both audience and languages, we looked to come up with a way that we could measure the quality of all and compare the confidence we have in each of these services.

By utilizing this Service Confidence Framework, we have been able to compare the services and determine the value of or the necessity of putting more effort into improving the quality of some services, and minimal effort in others. It has been a valuable tool that we will continue to improve upon going forward.

References

Crispin, Lisa and Janet Gregory. 2008 *Agile Testing: A Practical Guide for Testers and Agile Teams* Addison-Wesley Professional

Jacobs, Jef & Moll, Jan & Stokes, Tom. (2000). *The Process of Test Process Improvement.* 8.

Acknowledgements

This paper would not have been as well written as it was if it were not for my beautiful wife, Corrine Kidd, who also happens to be my editor. She is the writer in the family, but she has encouraged me and pushed me to finish this paper and made me explain things that just didn't make sense to her. So, with her asking of questions and cleaning up of my sentences this is a much better paper because of her.

I'd also like to thank Dirk Tepe, my manager, who helped form the ideas and thoughts that we have put together to form this Service Confidence Framework into what it is today.

Lastly, I want to thank Miami University, all my co-workers at in the Solution Delivery Group of IT Services, specifically the other members of the 4-Facets, as you have continued to push the use and growth of this framework and are helping us all to see the potential values of using this methodology.