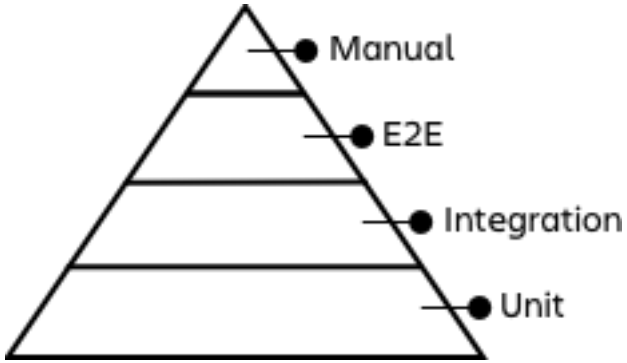
ATLASSIAN

Quality Health Monitor Sample

Attribute	Definition	Checkpoint discussion examples HEALTHY BIT SICK SICK
Shared QA responsibility	“Quality is everybody’s responsibility.” The team doesn’t rely on the QE for Quality Activities. The entire team contributes to the definition of Quality Processes and continuously improves these.	• Everyone's just improving being a Quality Champion. • Some team members have identified opportunities to improve Quality processes. • QE is focusing on the "bigger picture" now instead of the Quality rituals
Bare minimum manual testing	We know when to automate and when to manually test. Our manual testing effort is minimal yet effective and efficient due to our skills and experience. Guide questions: • How confident is the team of their testing skills? • Do we often miss edge cases? • Do we automate checks instead of manually executing them?	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

ATLASSIAN

Balanced test automation pyramid	We have the right distribution of automated tests at different levels. These tests are understandable, maintainable and uncomplicated to run. Guide questions • Does the team have good test coverage? • Does the team's different tests resemble the testing pyramid?  • Is the team automating more than they should? • Can the team run tests locally with ease?	• We have way more UI tests vs unit tests. • Some of the devs do not know how to run some of the tests in their locals.
Performance testing at scale	Our products meet (or establish) customers' expectations on performance. We do not ship changes that deteriorate the products' current performance. Guide questions • Does the team have performance benchmarks on their product/features? • Does the team regularly execute performance testing? • Is there an established way of monitoring performance regressions within dogfooding environments? • Is there a process to resolve degraded performance?	We have performance tests in our build pipeline and these are run towards the end of the release cycle. However, we have only noticed that the recent test results indicate a decline in performance. We need to improve our process in monitoring the details of the results more closely or improve our test reports to flag risks earlier.

ATLASSIAN

Stable CI/CD	Good automation of the CI/CD workflow contributes to the team's speed of delivery. Automated tests and checks are baked in the build and release pipelines with clear, actionable results for any failure. Guide questions • Are we running automated tests/checks at the right stage of the build pipeline? • Is the team's build pipeline reliable and non-flaky? • Is there a notification system for build failures? • Does the team have a process for handling flaky tests?	The team always addresses flaky builds/tests immediately. We time box validation of failures and then add these in our backlog. So far, most of these backlog items have been prioritised and included in the following sprint.
Acceptable bug rate	There will always be bugs in Production but we do not make releases that result to the degradation of product features or have a major impact to the end user flows. Guide questions • Is the total count of Production bugs rising per release? • Is the team shipping more bugs than they can fix? • Does the team allocate time in fixing Production bugs? • Does the team have a process in reviewing and responding to JAC issues? • Does the team learn from these bugs to continuously improve (i.e. lower bug rate, 0 regression, not repeating the same mistakes, etc.)?	We almost shipped a release with a potential SEFCON. We caught it by luck. Let's schedule a brainstorming session on how to prevent this from happening again.

ATLASSIAN

Balanced speed and quality	We know the importance and urgency of our deliverables so we commit to flagging risks and blockers early to mitigate these. However, we also know that the quality of deliverables should not be compromised for the sake of speed. Guide questions • Are there there too many re-work iterations (due to bugs, misunderstanding of requirements, scope change/creep, etc.)? • Is the team finding more bugs at later stages of the development cycle (e.g. uncovering bugs at Blitzes)? • Are there defined processes to ensure quality deliverables? • Does the team adhere to the processes that they have defined to ensure quality output?	There was a piece of work wherein it took more than a sprint to complete. We could have identified and mitigated risks at the earliest stage possible. Let's start doing Quality Kickoffs.
Maintainable codebase	Our codebase is easy to maintain so developers understand it and can contribute to it. This also aids in easier and faster peer reviews. Guide questions • Is there enough documentation in the repository to guide people on how to use it and/or contribute to it? • Is the knowledge of the codebase distributed amongst the team members? • Is the peer review process taking too long? • Are there checks in our dev process to help catch potential errors (e.g. linting rules, pre-commit hooks, etc.)? • Is the current codebase easy to extend? • Does the team regularly address tech debt?	It takes too long for a new developer to confidently submit a pull request because they're unsure that they might break things.