Team Test for Testability

1. If you ask the team to change their codebase do they react positively?

   > *If everyone actively recoils from change to all or a specific part of the system and there's always a reason not to tackle this. It's a no.*

2. Does each member of the team have access to the system source control?

   > *Everybody should be able to see the code. More importantly a lot of collaboration occurs here. Access for all is part of closeness as a team.*

3. Does the team know which parts of the codebase are subject to the most change?

   > *A codebase generally has hotspots of change. Knowing this can inform what to test, how to test and where observation and control points are.*

4. Does the team collaborate regularly with teams that maintain their dependencies?

   > *Teams have both internal and external dependencies. The testability of dependencies affects your testability. Close collaboration such as attending their sprint planning helps.*

5. Does the team have regular contact with the users of the

system?

> *To test a system effectively one must have deep empathy with those who use it. If you have internal users you invite them to demo's, for external customers your team is exposed to customer contacts.*

6. Can you set your system into a given state to repeat a test?

> *Some tests need data and configuration to be set. Can you reliably set the state of these assets to repeat a test again?*

7. Is each member of the team able to create a disposable test environment?

> *Sometimes programmers have environments to make changes in something representative of the world. Testers and operations focused people should have this too to encourage early testing, reduce dependency on centralised environments and setup time of tests.*

8. Is each member of the team able to run automated unit tests?

> *If you don't have unit tests, you probably haven't even tried to guess what a unit of your code is. Immediate no. If you do each member should be able to run these tests when they need them.*

9. Can the team test both the synchronous and asynchronous parts of their system?

> *Systems often have scheduled tasks which lifts and shifts*

> *or replicates data and other operations which occur on demand. Can you test both?*

10. Does each member of the team have a method of consuming the application logs from Production?

> *Its important that all teams members can see below the application, so we aren't fooled by what we see. If you don't have an organised logging library which describes system events you should get one, plus eventually some form of centralisation and aggregation.*

11. Does the team know what value the 95th percentile of response times is for their system?

> *If you don't know this you either haven't been able to performance test, gather data for analysis, or realised that you can analyse your performance data without outliers.*

12. Does the team curate a living knowledge base about the system it maintains?

> *This is anything that describes your system, a wiki, auto generated docs, anything. If someone wants to know what something does, where do you go? Your team has many members, not that many Product Owners can read code.*