|  |
| --- |
| **Testability Smells** |
| 1. **Too many production issues**   Does your team feel that too many issues are escaping into production? Is your team’s planned work frequently interrupted and delayed as a result of dealing with production issues? |
| 1. **Pre-release regression cycles**   Does your team have to execute a lengthy regression test cycle before releasing? Does your team often find important issues during this regression cycle? |
| 1. **Lack of automation & exploratory testing**   Does your team frequently check and confirm things that should be done using automation? Does your team overlook exploratory testing? |
| 1. **Hesitance to change code**   Is your team hesitant to make small, regular code improvements for fear it will introduce undetected issues? Does your team feel uncomfortable refactoring the code even when they believe it’s necessary? |
| 1. **Testing not considered during architectural design**   Does your team neglect to involve testers in the architectural design discussions? Does your team neglect the impact on testing when making design decisions? |
| 1. **Team constantly seeking more testers**   Does your team feel like they need to add more testers as a result of mounting workload and complexity? |
| 1. **Too many slow UI tests**   Does your team waste a lot of time preparing, executing and waiting for feedback from slow GUI tests, either manual or automated? |
| 1. **Important scenarios not tested**   Does your team release the system without testing important scenarios because they are either impossible or impractical to test? Are there areas of significant risk that are not being tested? |
| 1. **Ineffective unit and integration tests**   Does your team write unit tests and integration tests that often miss important problems? Does your team endeavour to continuously improve your unit and integration tests? |
| 1. **Cluttered ineffective logging**   Do your logs contain lots of errors and warnings even when the system is considered to be behaving as normal? Can team members quickly and easily isolate and debug issues using the logs? |
| 1. **Flaky nondeterministic automation**   Does your team spend a large proportion of their time investigating failures, debugging and maintaining automation? Does your team re-run automation when it fails expecting it to pass the second time? |
| 1. **Tests that contain duplication & irrelevant detail**   Does your team have tests that contain a lot of duplicate steps (usually setup) in order to get it in a state to perform the essential part of the test? Does your team have tests that contain lots of details that have nothing specifically to do with what you’re actually trying to test? |
| 1. **Issues are difficult to reproduce**   Does your team often encounter issues that are difficult, time consuming or impractical to reproduce either in your test environments or production? |
| 1. **Issues are difficult to isolate & debug**   Does your team struggle to isolate and debug issues when they occur either in your test environments or production? Does it take days of investigation to find the root cause of a problem? |
| 1. **Too much effort spent writing, maintaining and debugging automation**   Does your team rely too heavily on automation written at the UI level?Does your team test business logic through the UI? |