Stop the Line!

*I've discovered the **Stop-the-line** culture in **IMPLEMENTING LEAN SOFTWARE DEVELOPMENT. FROM CONCEPT TO CASH**, by Mary and Top Poppendieck. The authors describe this practice as follows: "work is organized so that the slightest abnormality is immediately detected, work stops, and the cause of the problem is remedied before work resumes.". In software development, this culture can be implemented by several practices combined.*

Test-driven development. Automated self-checking tests check the code does what it is required to do. *This enables to detect an abnormality in what the application does.*

Measure code coverage by tests. Any code never exercised by tests is detected. Never-tested code is just as bad as failing code. *This enables to be sure that anormalities may be detected everywhere in the code.*

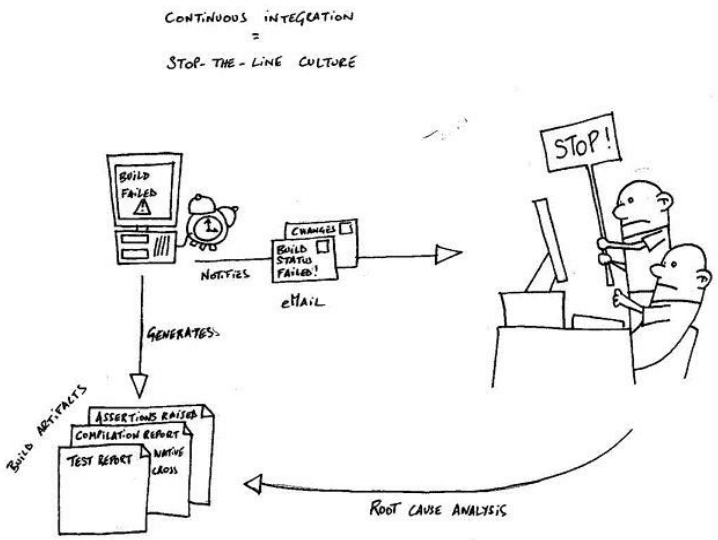
Design-by-contract with assertions. The class responsabilites (operations) are described in terms of pre-conditions and post-conditions. The class state is described in term of a class invariant. The pre-conditions, post-conditions and invariants are checked by assertions coded into the application. Therefore, if a contract is broken at runtime (which means a bug is detected) the application immediately stops and requires fixing. The assertions are exercised by the tests. *This enables to detect a broken contract as soon as it is introduced into the code.*

One-action build and test. A one-action script builds the whole application and runs the automated self-checking tests. *This enables to dispose of test results easily.*

Continuous integration. Developpers deliver and synchronize their work as often as possible. *This enables to always dispose of an up-to-date and shippable application.*

Automated build and test. A continuous integration tool automatically builds and tests the application upon detection of a delivered code or test change. *This enables to automatically detect abnormalities in the build or in what the code does.*

Send abnormality notifications. If the automated build and test fail for any reason (the build fails or a test fails), then a notification is sent to the developpers. The notification may be emails or lava lamps. *This enables the developpers to be aware that an abnormality has been detected.*



These practices enable to detect a defect as soon as possible and to notify the development team. Now the team has to stop and correct the problem before work resumes. The best part is that all these practices have many other benefits!