The Role of Feedback in Continuous Integration, Continuous Delivery and Agile ALM
feedback /ˈfɛd,bak/  

Noun: Information about reactions to a product, a person’s performance of a task, etc., used as a basis for improvement.

The modification or control of a process or system by its results or effects, e.g., in a biochemical pathway or behavioral response.

“Information produced during the software development process, provided to process participants and used to improve the product of that process.”
Continuous Integration (CI) at its core is about feedback. It’s about implementing a process that facilitates feedback so that problems can be identified and corrected early in the development process – when it is easier to diagnose and correct them. Continuous Integration feedback is developer-focused, which means that feedback pertains to things like coding or architectural issues, build failures, test status and file release uploads.

The goal of CI is to provide rapid feedback.
Continuous Delivery (CD) extends the Continuous Integration process from build to include the test and deploy phases. One of the primary goals of CD is to ensure fluid communication and coordination between all participants within the delivery process – managers, developers, testers, and operations personnel – and in some cases, even business users and customers. As such, CD is also a key underpinning of DevOps. Since CD involves an extended pipeline, it’s important to identify issues early on. The further downstream issues are identified, the more difficult it is to resolve them.
Since Continuous Delivery involves all process participants, feedback mechanisms should provide visibility to the entire team across the delivery process. Feedback goes hand in hand with traceability. Continuous feedback provides visibility to the entire team so they know the status of what is being delivered and are alerted to issues quickly, while traceability helps identify root cause for faster resolution. This might involve tracing an issue back to the original source code or configuration change that ultimately initiated the broken build, or tracing downstream to a unit test or acceptance test that failed.
Integrating processes and tools into an ALM platform like CollabNet TeamForge automates feedback loops and automates the capturing of events and decisions made throughout the application lifecycle. It facilitates alignment of processes and stakeholders up front to avoid mismatches downstream.
To optimize CI/CD processes, feedback should be...

- **Fast and early**: Notifications of critical events, like failures or negative feedback should be raised immediately.

- **Visible and accessible**: If feedback is not conspicuous or cannot be easily accessed, chances are it will be missed or ignored. Critical feedback should be pushed to teams, not pulled.

- **Actionable**: Adequate information must be provided, so that the team can act on it – especially if it is negative feedback.
Feedback should NOT be...

**too little**

Teams need adequate information to act rapidly and appropriately. “The test failed during the acceptance process” is not enough. Feedback should specify which test failed, how it failed, and provide links to supporting information.

**too much**

Information overload often results in people ceasing to pay attention or critical tidbits of information becoming lost in the sea of information overload. Only provide necessary information that will help teams identify, diagnose and resolve problems.

**too late**

Information should be provided immediately after an event has happened to avoid latency. If a build fails all team members must be notified immediately, so they don’t continue to check in on the broken build. It will only compound the problem.

Remember!

Feedback is only useful if it’s acted upon.
Teams should develop rules for handling various types of feedback and be held accountable to those rules.

- **broken build**: Do not check in, stop and correct!
- **drop in code coverage**: Do not introduce new changes until test code coverage exceeds threshold!
- **failed smoke test in production**: Rollback to previous version and correct immediately!

Be sure all important feedback is timely and any actions taken to resolve issues are visible to the entire team.
Types of feedback mechanisms for communicating feedback to process participants.

**Asynchronous or “pulled”**

These feedback mechanisms usually involve non-critical events or continuing trends: Commit logs, dashboards, pipeline views, VCS tags, deployment reports and code metrics for code quality.

**Synchronous or “pushed”**

These feedback mechanisms usually involve critical events or negative feedback. They are pushed to teams so that they can respond to immediately: email, instant messaging, IDE monitors, RSS, Social media. Integration with issue tracking systems – or even better an ALM platform like TeamForge with TeamForge Orchestrate – lets you continuously capture feedback from your delivery pipelines and set up notification triggers.

**Extreme Feedback**

These feedback mechanisms usually involve the use of physical or hardware devices, such as lava lamps, rocket launchers, LED displays, or the classic red and green orbs, that turn red when the build breaks. LED displays can show other types of data like code quality metrics, grids of builds statuses, or system health monitoring data.
For a simple delivery pipeline consisting of a commit stage, a testing stage* and a deployment (release) stage, these are some of the events that will need to be captured:

Commit Stage

Commit notifications, build status, build results, unit test results and code metrics for code quality.

Testing Stage

Non-functional test results for performance, acceptance test results, manual test results (via ALM or tracker integration).

Deploy Stage

Smoke tests (in staging), production performance monitoring and deployment reports.

For events that are non-critical and simply informational, an asynchronous (pull) mechanism or monitoring display should be used. For instance, teams will not want to be notified of every commit. However, if a build fails, the entire team should be notified immediately via a synchronous (push) mechanism.
The testing stage shown in this delivery pipeline consists of acceptance and exploratory testing.

*The testing stage shown in this delivery pipeline consists of acceptance and exploratory testing.*
Integrating Feedback Loops with ALM

“TeamForge provides feedback across the lifecycle in the form of tracker notifications, source code and commit monitoring, build and test status and notification, file release uploads and more.”

TeamForge is an Agile ALM platform that supports CI/CD processes and lifecycle feedback from end to end. TeamForge integrates all development and delivery tools into one common management and security platform and enables association and traceability all the way from initial requirements to deployment. TeamForge integrates feedback mechanisms into ALM and gives everyone involved in the development and delivery processes visibility into the feedback they need – when they need it, regardless of where they are located.

“Traceability is a requirement for quick response to critical feedback.”
Implementing Enterprise Cloud Development with CollabNet

Team communication is important for teams to function effectively and produce quality products. Feedback is fundamental not only to CI and CD, but to the entire ALM process. Feedback should be fast, and actionable and delivered in a way that lets teams respond rapidly. Traceability throughout the lifecycle is required for rapid response. Tailor your feedback process to fit your team. If an existing solution doesn’t fit or becomes outdated, create a new one that works. To learn more visit www.collab.net/deploy

Learn More
Want to learn more about Source Code Management, Agile, Continuous Delivery, Application Lifecycle Management or DevOps?
Visit CollabNet’s free resource library. It’s packed full of whitepapers, analyst reports, videos, and webinars.
www.collab.net/resources
Topics trending now

Many of the latest technology announcements have implications for PaaS and cloud development that will serve agile businesses everywhere.

• Enterprise Cloud Development, www.collab.net/ecd

• Continuous Integration, www.collab.net/getci

• 5 Things your Development Team need to be doing now, www.collab.net/5things

TeamForge Orchestrate will help you get the most value out of your CI/CD processes. Orchestrate acts as the control center for delivery pipelines, capturing vital process information in a central system of record, and automating lifecycle traceability across tools and clouds. Choose from pre-built integrations or quickly integrate your own. To learn more visit: www.collab.net/products/teamforge/orchestrate#why

CollabNet Deploy is a TeamForge DevOps solution for facilitating real-time collaboration between development and operations teams. CollabNet Deploy will transform your manual deployments into lean, secure and efficient automated processes that take minutes instead of hours or days. To learn more visit: www.collab.net/deploy
About CollabNet

CollabNet is a leading provider of Enterprise Cloud Development. With more than 10,000 global customers, the company provides a development platform and services to address three major trends disrupting the software industry: Agile, DevOps and hybrid cloud development. CollabNet TeamForge® ALM, ScrumWorks® Pro Agile project management and Subversion Edge and Enterprise Git source code management can be deployed separately or together, in the cloud or on-premise. CloudForge® enables cloud development through a flexible platform that is team friendly, enterprise ready and integrated to support TeamForge and leading third party tools. CollabNet complements its technical offerings with industry leading consulting and training services for Agile and cloud development transformations. Many CollabNet customers improve productivity by as much as 70 percent, while reducing costs by 80 percent.

For more information, please visit www.collab.net