Software Development Life Cycle (SDLC)

Table of Contents

Revision History

Version	Date	Changes
1.0	2017-05-12	Initial Release
1.1	2017-10-31	Added Revision History
1.2	2017-11-08	Secure Coding Program details
1.3	2018-01-04	Monthly Roll Out Update
1.4	2018-04-05	Monthly releases have transformed to numbered releases which are independent of months.
1.5	2018-05-14	Expanded security section and added development terminology section
1.6	2018-06-04	Expand, integrate new terminology
1.7	2018-10-01	Add change management process
1.8	2018-11-26	Update for Firmware
1.9	2018-11-26	Updates for Firmware
1.10	2019-09-26	MyAdmin is included in the security test environment, Microservices are included in security scans
1.11	2020-10-20	Removed old MyGeotab versioning descriptions, minor day of week change for deployments
1.12	2020-11-03	Change “master branch” to “main branch”
1.13	2021-11-17	● Update MyGeotab build number conventions. ● Drop MyGeotab windows performance metrics as we now only deploy to Linux production servers. ● Add new metrics from load test dashboard ● Maintenance is now called Server Operations
1.14	2022-02-04	● Annual review ● Add table of contents ● Updated copyright in footer
1.15	2022/03	● Firmware uses Parasoft to scan
1.16	2022/08	● Rm Zendesk reference ● Add rare case for features to go into patches ● Add dependency scans ● Less specifics on training
1.17	2023/01	Annual Review
2.0	2024/02-03	Refocus document and annual review
2.1	2024-05-15	Annual Review
2.2	2025-10-07	Annual Review

Summary

High-Level Overview

The software development process follows a continuous integration methodology. Our approach is multifaceted, from examination of individual lines of code to comprehensive system testing. Geotab produces releases, which introduce new features and fixes, as well as patches, which aim to fix issues that cannot wait for a new release.

At a high level, our quality assurance (QA) process encompasses both functional testing (unit, integration and system tests) and non-functional testing (performance, security, usability).

Secure Coding Program

Geotab has developed and implemented an industry best practices development program around source code and product development. Program functions and processes are described below:

- All Geotab developers undergo regular security training.
- Geotab's development process requires that all commits are peer reviewed and approved by others, prior to being merged into the production release repository. More details are in the Branch Testing / Quality Assurance section.
- Geotab has developed a fully-automated build process, which pushes the compiled applications through a huge library of highly custom unit tests and data validation tests. Any detected issues will mark the build as a failed build and the necessary development teams will be alerted. Changes must be made to create a passing build, as only passing builds can be promoted into production.
- Geotab has partnered with various security vendors. The security scanners provide details of all discovered, potential vulnerabilities. Code scan reports are reviewed and any valid issues are identified, logged and tracked in Geotab's internal vulnerability program. We sometimes switch vendors. Products use a subset of the following:
    - Veracode (www.veracode.com) and its Secure Application Development program, which does an in-depth binary scan of the compiled applications.
    - Fortify by MicroEdge (previously Fortify was an HP product). Fortify is a source code scanner.
    - Parasoft Static Analyzer analyzes source code for coding violations and software weaknesses that expose vulnerabilities.
- Geotab repositories are scanned by a third party dependency tool, which looks for vulnerabilities in underlying components. Identified vulnerabilities are reviewed, logged and tracked in Geotab's internal issue tracking system.
- Release candidates are pushed to Geotab's Security Test environments. The candidate is then scanned by a variety of industry-leading vulnerability scanners that look for additional potential issues. All issues are reported to Geotab's Security Department for review and mitigation, prior to any product release.
- Geotab has annual external penetration testing on all network appliances, applications and Geotab GO devices.
- We also conduct systematic vulnerability testing on all our production servers (continuous scanning). These tests use a variety of industry scanners.
- Detailed procedures for developers are available to internal users or those who have signed Non-disclosure Agreements (NDAs). This includes roles and responsibilities, separation of duties, and developer activities to support security.

Development Workflow

Overview

Major features closely follow the iterative and incremental model, similar to a series of mini-waterfalls. The frequency of the steps may vary, depending on the product.

1. Requirements
2. Design

3. Development
4. Testing
5. Release Engineering
6. Change Management
7. Deployment
8. Support

Requirements

Requirements are documented in **tickets** in our **issue tracking software**. New feature requests are reviewed based on security, customer needs, and compliance testing. New features are prioritized according to existing backlogs.

Planning / Design

Large features are broken into smaller parts, which are either implemented sequentially in a series of releases (each sub-component being fully functional), or in parallel by multiple developers. Each of these parts follows its own mini-waterfall approach.

Development / Implementation

During the development cycle, developers create **development branches** based on either a **main** or **release branch**. Commits are made to the development branches. Pushes cause the development branch to compile builds and run **automated tests** on our **continuous integration (CI) pipeline**. These builds are not approved by default, i.e. they are not deployable builds. If the CI build fails, the developer can make changes and repeat. Once satisfied, the developer can pass the ticket (and branch) to one or more code reviewers.

Bug Fix Patches

Patches are changes to an already-released version of the software aimed at fixing issues that cannot wait for the next release. Important issues are brought to the attention of the Development team through our Support team and our 24/7 on-call system.

The goal of a patch is to quickly fix critical issues while minimizing the risk of creating new issues. On a rare exception basis new features may go into patches upon approval of a team lead and release manager. New features must go through more extensive testing that will not fit into the timeline of a patch. Patches more closely follow the agile methodology, which is based on iterative and incremental development. Patches are applied only to the servers that require them and the results of the patch changes are carefully monitored.

Testing/Quality Assurance

Development Branch

Code reviews and automated tests are performed on each development branch.

- Code Reviews — Each code change is examined by a developer, different from the author of the change, and reviewed to determine if it should be patched into any other releases. Unit tests are updated or added for the code change to prevent regression issues.
- Automated Tests — Tests run automatically on our Continuous Integration (CI) pipeline. These include:
  - Unit testing on small pieces of code, such as individual functions, in isolation.
  - Integration testing for how parts of the system work together.
  - Functional UI testing (browser testing) to verify the UI is functioning correctly, the data is being processed, and translations have been provided where needed.

Branch Merge

After a ticket passes the manual code review, CI automated tests, and optional manual testing, the code reviewer merges the development branch to either the main or release branch, as appropriate.

The new merge runs through the same automated tests that the development branch was run through. Test failures are addressed immediately. These are typically few, if any, as the branch merged already passed the same tests on its own.

Additional Testing

Products are subject to extensive testing, according to their needs. Not all products require every type of additional testing. Releases are subject to more tests than patches.

- Automated Testing — Each new release must pass the automated test suite described earlier.
- Performance — New releases are monitored for server health metrics such as CPU, memory utilization, thread count, bad file counts and GBQ uploader data volume. Additional internal monitoring tools such as performance reports, logs, and notifications are also reviewed.
- Trend Analysis — Results from Big Data trend analysis are examined for any performance anomalies.
- Regression Analysis — Results from Big Data regression analysis are examined for any performance / data anomalies.
- Focused Manual Testing — Parts of the system which cannot be automatically tested are manually tested for each release.
- Integration Testing — The cumulative set of code changes for a release is reviewed to identify major functional areas of change. Black box testing is then performed for the identified functional areas.
- Internal Breadth Testing — New releases are previewed internally, allowing Geotab employees (from multiple departments) to test new features and general usability.
- Security Testing — Each new release must pass extensive security tests. The application is run through multiple scanners, and findings are evaluated and addressed as required. Third-party frameworks, modules and libraries are checked for required security upgrades.
- ProdTest/Load Testing — New releases are installed on a set of copied servers and their performance is compared with older releases on original servers. Both sets of servers receive the same vehicle data in order to test whether live data processing is impacted when moving to a new release. This testing does not duplicate user UI actions.

Release Engineering

Once testing is complete, a decision is made by the Release Engineer to accumulate all of the commits from the main or release branch since the last approved build, and create a new build which is approved for deployment.

There are two types of **deployable builds**: **releases** and **patches**. Patches contain fixes that cannot wait until the next release. As a company, we try to balance fast support for our customers with the more productive release approach.

For a given **branch**, there is an initial release, followed by patches.

There are two parts to the **build approval process**: **approving** the deployable build, and **notifying** other departments. The release engineer sends out a release or patch notice to an internal email list. The note contains the branch, build number, QA status, priority, and any special deployment notes (e.g. if a patch is required for a particular server).

Deployable builds are approved, and then deployed to servers. Each build has an associated build number.

Release Artifacts

Release artifacts are maintained. Below is a representative list typical for Geotab products.

- Build Labels — Each ticket that is part of a release or patch is tagged with the build in which it was first deployed. The build tags can be used as filters in our issue tracking system, so we can always cross reference tickets with a deployable release/patch.
- Security Scan Results — Results from various security scans.
- Analysis and Statistics — A complete list of release and patch builds, the corresponding approval date, and the percentage of features versus bugs, bug sources, and backlog.

Change Management

All changes and associated reviews, approvals, and other activities shall be documented and tracked, as appropriate, within the version control or change management tools and retained for a minimum of 1 year.

Security impacting and architectural changes to production components shall be coordinated and overseen by the Geotab Change Advisory Board (CAB). The CAB shall review all proposed standard and significant changes to the Geotab Telematics systems and related features. The CAB reviews proposed changes on a frequent basis. Geotab shall maintain a CAB Charter outlining the roles and responsibilities of the CAB.

Geotab shall test, validate, and document changes to the Geotab production components before enabling the changes in a production environment.

Deployment

Servers can be categorized as test, security_test, prod_test and production and deployments follow the same order. In addition, large scale deployments follow a staged/staggered production deployment schedule, with monitoring at each stage If any critical issues are found at any state, the schedule is put on hold. Once the issues have been resolved, the deployment continues from where it left off.

Deployment schedules take into account the users impacted. For example, test servers might be updated during the workday, whereas customer servers would be updated off hours.

Support for All Products

Important issues are brought to the attention of the Development team through our Support team, and our 24/7 on-call system.