KDAB's Software Development Best Practices

The Basics

KDAB

These tips won't be any surprise for most professional software engineers. But if you're just getting started it can be a handy reminder. If you're not already doing everything here, you should be.

Keep it simple. Complexity is the enemy of reliability and maintainability. It's often tempting to over-engineer a feature, building in much more flexibility or configurability than the requirements demand. We know it's a natural inclination – developers are trained to avoid hardcoding things to create reusable code. Just remember that this can be taken too far – remember K.I.S.S. You can always refactor things later to add more configurability in the future.

Use version control. Programming any project of substance without using a version control system (VCS) is like free solo rock climbing – technically, it can be done, but it's so incredibly risky that only the most foolhardy would attempt it. A VCS is your safety net so you can make code changes in a testable and repeatable way, a natural way to collaborate in a team, and a tool that carefully tracks and marks your progress. (If you're not using one now, you can't go wrong with Git.)

Adopt a coding standard. Having a consistent coding style helps ensure that the software looks the same, improving readability and understandability. Consistent expression of programming paradigms can also help you avoid cybersecurity issues. While it's better if your coding standards are aided by your tool chain (either statically or within your IDE), an agreement of how file headers, function blocks, loop structures, and variable definitions should be written and documented can prevent code messiness from getting too out of hand.

Incorporate static analysis. Static analysis tools can help you find bugs and potential problems in your code before you even run it. This includes some great KDAB contributed/created tools such as clazy and community projects like clang-tidy. They can also help you comply with coding standards (see the point above). Like version control, static analysis is an additional safety net that once you start using it, you'll wonder how you did without.

Formalize your testing. Of course, you test. But you should be able to perform unit tests, integration tests, and system-level tests automatically so you're confident updates won't introduce new regressions. It takes some extra

time to set up test scaffolding systems (such as Qt Test, Boost, Catch2, doctest, and Google Test), but it's time well spent in ensuring you can produce a quality product and quickly update it.

Document well. Everyone has gotten lectures about writing good comments, but it doesn't really hit home until you read code you've written before and forgotten why you wrote it that way. Good documentation doesn't say what the code is doing as much as why it's doing it that way. It's something that your teammates and your future self will highly appreciate.

Formally track bugs. Every bug that's known should be kept in a bug database; feature requests and enhancement ideas should go here too. A bug tracking tool like Jira, Basecamp, or Assembla provides an overview of the quality of your project, shows your progress towards release, and lets the team focus their effort in areas that matter.

Track dependencies. Keep track of all the dependencies and third-party libraries that your system relies on. Make sure you regularly check to keep them up to date and reintegrate and rerun your regression and integration tests when they change. Keep track of all licensing agreements that any open-source software requires so you can be sure you're properly adhering to their terms. Tools like dependabot exist so you don't have to do this manually.

Code reviews. Having peers review your code before it's submitted to the codebase is a significant ingredient for building quality software. An extra set of eyes can always help point out boundary circumstances, remote implications, or even downright bugs that you can be too close to see. Code reviews also carry a set of best practices so you can make the most of them while not burning out the team.

What is KDAB's Software Development Best Practice series?

This series of whitepapers captures some of the hard-won experience that our senior engineering staff has developed over many years and projects. Offered up as a grab bag of techniques and approaches, we believe that these tips have helped us improve the overall development experience and quality of the resulting software. We hope they can offer the same benefits to you.

View all three parts of this whitepaper series online at: ***www.kdab.com/publications/bestpractices/***

About the KDAB Group

The KDAB Group is the world's leading software consultancy for architecture, development and design of Qt, C++ and OpenGL applications across desktop, embedded and mobile platforms. KDAB is the biggest independent contributor to Qt and is the world's first ISO 9001 certified Qt consulting and development company. Our experts build run-times, mix native and web technologies, solve hardware stack performance issues and porting problems for hundreds of customers, many among the Fortune 500. KDAB's tools and extensive experience in creating, debugging, profiling and porting complex applications help developers worldwide to deliver successful projects. KDAB's trainers, all full-time developers, provide market leading, hands-on, training for Qt, OpenGL and modern C++ in multiple languages.

www.kdab.com