Code Review Questionnaire

Rainer Stropek (software architects)

October 28, 2020

Contents

Introduction

Background

We regularly review code for customers who want to have an external assessment about software quality. Typical reasons for code review projects include:

- M&A (acquisition of a company or source code of a product)
- Establishment of a strategic partnership between companies
- New management team wants external opinion
- Customer-vendor relationship
- Large customers buys software that is strategic for their business
  - Frequently: Large customer, small vendor
- Team wants/needs external advice
  - As reviewers, we are a kind of external coach

The answers to the questions in this questionnaire should provide a general overview about the nature of a project. It accompanies the review on source code level to ensure a deeper understanding of the context in which the code has been created and is maintained.

Note that it is important that you do not just describe the current status of your project. If you know of upcoming changes or if related development projects are currently going on, describe those changes.

Core Review Principles

1. Be objective
   - Based on widely accepted best/worst practices defined by e.g. programming platform vendors
   - Clearly state if something is a subjective opinion
2. Be realistic
   - Every project has technical debts[1]
   - Every project as resource constraints
3. Be honest
   - Be polite and appreciative, but be clear about weaknesses

[1] https://en.wikipedia.org/wiki/Technical_debt

    - Don’t just talk about bad things, call out good practices, too

4. Value software craftmanship

    - Aim for professionalism, technical excellence
    - Death of the production line and *factory workers* attitude

5. Avoid being overly smart

    - Code is more often read than written
    - Obvious code first
    - Know your languages and platforms, but avoid being too smart

6. Value legacy code and its maintainers

Key Figures

Introduction

This section contains questions about key figures regarding the software that should be reviewed.

History

- When did the development of the software start?
  - If there were important technology-related milestones, describe them (e.g. switch from one base technology to another).
- How many person-years have been invested in the development of the software
  - Ballpark estimation is sufficient.

Technologies

- Which technologies (e.g. programming languages, frameworks, important component libraries, etc.) is the software based on?
  - Specify version numbers. Are they current or outdated?
  - Backend and UI
- Which external systems (e.g. databases, external APIs, message brokers, cloud services, etc.) are important?
  - Specify version numbers. Are they current or outdated?

Metrics

- Number of customers/users/installations
  - Goal: Get a feeling for the software's market position
- Number of modules/libraries/projects
  - Goal: Get a feeling for the size of the project
- Number of lines of code per module/library/project
  - Goal: Get a feeling for the structure of the software system (e.g. monolith vs. microservices)

Software Architecture

Introduction

Typically, the questions in this chapter are discussed personally in a software review workshop. If you have a written documentation of your software system's architecture and design, provide it before or during the workshop.

Overall Architecture

- Describe the overall architecture of the software solution.
  - Which components/sub-systems/modules do you have? How do they relate to each other?
- Do you have guiding principles (e.g. design patterns, 12factor app[2], microservices) when designing your software architecture?
  - If you have, describe them or provide links to external design principles that you follow.
- Describe characteristic features of your software architecture/design.
- Do you know of specific technical debts and/or overengineering regarding software architecture?
  - If you do, describe your plans to fix them (if you already have a plan for that).

> On a scale from 1 (worst) to 10 (best), how would you rate your software's overall architecture?

Distributed System

- Describe important communication patterns (e.g. protocols, standard, asynchronous vs. synchronous communication, etc.) in your software system.
- Do you have design guidelines for communication protocols (e.g. web API design guidelines, guidelines for choosing communication protocols)?
  - If you have, describe them or make them available.

> On a scale from 1 (worst) to 10 (best), how would you rate your software's communication architecture?

[2] https://12factor.net/

Security Architecture

- Describe security-related aspects in your software system.
  - Authentication
  - Authorization
  - Protocols and standards (e.g. *OpenID Connect*)
  - Used components (e.g. middleware in web apps/APIs)
- If you software uses cloud computing, describe the cloud-related security architecture.
  - Network design
  - Firewalls, reverse proxies, etc.
- Describe your concepts for encryption of data in transit and data at rest.

> On a scale from 1 (worst) to 10 (best), how would you rate your software's security architecture?

Software Distribution

- Do you ship your software to customers or do you offer it as SaaS?
- Do you use cloud computing?
  - If you do, describe your use of the cloud environment (e.g. cloud vendor, services used, etc.).
- Do you use container technology?
  - If you do, describe your use of container technology.

Examples

- Describe an aspect of your software system's architecture/design that has become a technical debt.
- Describe an aspect of your software system's architecture/design that you are proud of because it has proven valuable.

Quality Culture

Introduction

This section contains general questions about team culture influencing code and software architecture quality.

Organization

- Who is responsible for which aspects of software quality?
  - Do you have dedicated teams responsible for software quality, testing, and/or quality assurance?
  - What quality-related responsibilities does each individual developer have?
- Does your team do internal code reviews?
  - If you do, describe the process and tools for code reviews.
- Does your team follow guiding principles when writing code (e.g. *clean code* principles, *SOLID* principles)?
  - If you do, describe them or make them available.

> On a scale from 1 (worst) to 10 (best), how would you rate your organizational readiness for building high-quality software?

Software Quality Management

- Do you make code quality visible for all team members?
  - Do you publish results of automated code quality analysis (e.g. with a product like SonarQube[3])?
  - Do you publish key performance indicators (KPIs) from you support team (e.g. number of support tickets)?
- Do you track and manage quality improvements over time?
- Do you actively manage technical debt?
  - If you do, describe the process and tools you use for it.
- How do known quality issues influence your product backlog?

[3] https://www.sonarqube.org/

> On a scale from 1 (worst) to 10 (best), how would you rate your team's maturity in terms of management of software quality?

Automated Software Quality Analysis

- Do you use tools for ensuring proper software quality (e.g. profilers, benchmarking tools, dependency analyzers, test coverage reporting, etc.)?
  - If you do, name the tools and describe their use.
  - Is their use automated (e.g. in an CI/CD pipeline) or manual?
  - Do you regularly use them or only in case of specific problems?

> On a scale from 1 (worst) to 10 (best), how would you rate the equipment of your team regarding tools for software quality analysis?

Software Design

- Do you have a written description of your software's architecture and/or design?
  - If you have, make it available. Are the documents current or outdated?
- Do you use tools (e.g. modelling tools) for designing your software's architecture?
  - If you do, name the tools and describe their use.

> On a scale from 1 (worst) to 10 (best), how would you rate the maturity of you team with regards to software architecture/design?

Source Code Handling

Introduction

This section contains questions about source code management.

Version Control System

- Do you use a version control system?
  - If you do, describe it.

> On a scale from 1 (worst) to 10 (best), how would you rate the quality of your teams' use of your version control system (e.g. use all the available features, quality of checkins/pull requests/comments, etc.)?

Integrated Product Management

- Do you have a product management system (e.g. backlog management, milestone management, etc.) in place that is integrated with your version control system?
  - If you do, describe it.

> On a scale from 1 (worst) to 10 (best), how good is tool support for product management of your software system?

Contributions

- Do you have guidelines for forking/branching/pull requests (e.g. git flow, GitHub flow)?
  - If you do, describe the process and - if applicable - the tools.
- Do you have documented contribution guidelines (e.g. *done* checklist)?
  - If you do, describe them or make them available.

Coding Guidelines

Introduction

This section contains general questions about coding guidelines.

Existing Guidelines

- Do you have documented coding guidelines in place?
  - If you do have them, make them available. Are they current or outdated?
  - Do all developers know about them and adhere to them?
  - Are they specific for your team or do you refer to external coding guidelines (e.g. from programming language vendor)?

> On a scale from 1 (worst) to 10 (best), how would you rate the quality of your coding guidelines and your processes to enforce them?

Tooling

- Do you use tools to check for coding guideline violations (e.g. code analyzers, linters, specific settings related to compiler warnings, etc.)?
  - If you do, name the tools and describe their use.
  - Is their use automated (e.g. in an CI/CD pipeline) or manual?
- Do you use tools for supporting developers in writing good code that adheres to your coding guidelines (e.g. tools suggesting code changes in your IDE based on coding guidelines)?)
  - If you do, name the tools and describe their use.

> On a scale from 1 (worst) to 10 (best), how would you rate the equipment of your team regarding tools for code quality?

Examples

- Name a class/module/namespace/project with large technical debt in terms of coding quality (e.g. old module that has been extended a lot over time without ever refactoring it)
- Name a class/module/namespace/project which has recently been developed and therefore reflects your team's current code quality the best.

Dependency Management

Introduction

This section contains questions about how you handle dependencies.

Package Management System

- Do you use a package management system (e.g. NPM, NuGet) for distributing internally developed components?
  - If you do, describe the process and tools.
- Do you use public package feeds (e.g. NPM, NuGet) or do you have private servers/services for package distribution?
- Describe your versioning policy (e.g. Semver[4]) if you have one.

> On a scale from 1 (worst) to 10 (best), how would you rate your team's ability to create and distribute reusable code fragments via dependency management?

External Dependencies

- Name the most important (max. 10) external dependencies that your software system uses (e.g. frameworks, core libraries, UI component libraries, etc.).
- Do you use outdated versions which are no longer supported, no longer maintained, or obsolete?
  - If you do, describe the reason and potential plans to upgrade/replace them.
- Do you have a defined process for keeping dependencies up to date (e.g. defined interval for updating core dependencies, defined responsibilities to monitor the activity on important open source dependencies)?
  - If you have, describe it.
- Does your software contain abstractions to isolate your code from changes in external dependencies?
  - If it does, describe it.
  - Do you use dependency injection? If you do, describe it.
  - Did you have a situation in the past where those abstractions were helpful or problematic?

[4] https://semver.org/

> On a scale from 1 (worst) to 10 (best), how would you rate your team's maturity in terms of dependency management?

Documentation

Introduction

This section contains questions about written documentation related to your software system. End-user documentation is not in scope. We focus on documentation related to software architecture and design as well as code documentation.

Conceptual Documentation

- Imagine I am a completely new developer in your team, what do you give me to read?
  - Conceptual documentation
  - Documentation of the development process (e.g. source control, CI/CD, etc.)
  - Documentation of the development environment (e.g. IDE, tools)
  - Documentation of the system's architecture and design
  - Coding guidelines
  - Design principles
  - Etc.
- How do you write conceptual documentation technically (e.g. Word documents stored in SharePoint, Markdown, Wiki)?
- Does your documentation reflect the current status of your software or is it outdated?
  - If it is outdated, describe potential plans to fix that.

> On a scale from 1 (worst) to 10 (best), how would you rate your existing documentation of design principles, coding guidelines, and development environment?

Code Comments

- Do you use available standards in your programming language (e.g. C# XML code documentation, *JSDoc* annotations) to document code and APIs?
- Do you use available standards for documenting communication protocols (e.g. *Open API Specification*)?
- Do you generate a human-readable API documentation from your API documentations?
  - If you do, describe the process and tools (e.g. DocFX[5], *Swagger UI*).

[5] https://dotnet.github.io/docfx/

- Do you have a written guidelines or a common understanding in your team when to add comments to code?

> On a scale from 1 (worst) to 10 (best), how is the quality of your API and code documentation?

Automated Testing

Introduction

This section contains questions about automated testing.

Automated Tests

- Do you write automated *unit* tests?
  - If you do, how would you rate the quality of your unit tests (completeness, meaningful assertations, etc.) on a scale from 1 (worst) to 10 (best)?
  - Do you automatically run unit tests in your CI/CD pipeline?
- Do you write automated *integration* tests?
  - If you do, how would you rate the quality of your integration tests (completeness, meaningful assertations, etc.) on a scale from 1 (worst) to 10 (best)?
  - Do you automatically run integration tests in your CI/CD pipeline?
- Do you write automated *UI* tests (aka end-to-end tests)?
  - If you do, how would you rate the quality of your UI tests (completeness, meaningful assertations, etc.) on a scale from 1 (worst) to 10 (best)?
  - Do you automatically run UI tests in your CI/CD pipeline?
- Do your write other kinds of automated tests (e.g. performance benchmarks, load tests)?
  - If you do, describe them.
- Do you apply the same coding guidelines for testing code that you apply to the rest of your code base?
- Do you use a framework for mocking?
  - If you do, describe the concept and tools.

Manual Tests

- Describe manual testing processes for your software system.
  - Frequency
  - Responsibilities
  - Documentation of test results
  - Test cases

> On a scale from 1 (worst) to 10 (best), how would you rate the manual test processes in your team?

Test in Production

- Do you have a plan for testing in production (e.g. A/B testing, preview versions, feature flags)?
  - If you do, describe it.
- How do you gather logs and telemetry data when testing in production?

> On a scale from 1 (worst) to 10 (best), how would you rate your skills for testing in production?

Performance and Scalability

Introduction

This section contains questions regarding performance and scalability

Performance

- Do you currently know of specific performance issues in your software system?
  - If you do, describe them.
- Do you use performance profilers?
  - If you do, describe the process and tools.
- Do you have automated performance benchmarks?
  - If you do, describe them.
- Do you gather and analyze performance-related telemetry data in production?
  - If you do, describe the process and tools.

> On a scale from 1 (worst) to 10 (best), how would you the performance of your software system?

Examples

- Describe code samples where you fight with performance problems.
- Describe an example for an optimization that you did based on a profiling session.
- Describe an example for an optimization that you did based on telemetry findings.
- Describe examples of performance optimization using parallel programming.

Security

Introduction

This section contains security-related questions. Note that an in-depth security review is out-of-scope.

Automated Security Scanning

- Do you use tools for automatic security scan (e.g. scan source code, Docker images, etc.)?
  - If you do, describe the process and tools.
- Do you use tools for automatic vulnerability scanning (e.g. OWASP tools[6])?
  - If you do, describe the process and tools.

> On a scale from 1 (worst) to 10 (best), how would you rate the maturity of your team regarding automated security scanning?

Handling of Secrets

- Describe your concepts for storing secrets (e.g. database credentials, API keys).

[6] https://owasp.org/www-community/Vulnerability_Scanning_Tools

Automation

Introduction

This section contains questions regarding automation of development processes.

Build Automation

- Do you have a system in place for automated building of your software system (*continuous integration*, short *CI*)?
  - If you do, describe the process and tools.
- Describe additional steps running during CI (e.g. automated tests, code scanning, building of documentation portals, etc.)?

> On a scale from 1 (worst) to 10 (best), how would you rate the maturity of your CI process?

Release Automation

- Describe your staging system (e.g. dev, test, prod).
- Do you have a system in place for automated deployment of your software system (*continuous deployment*, short *CD*)?
  - If you do, describe the process and tools.
- Describe additional steps running during CD (e.g. build installers, handling of stages, confirmation processes, etc.)?
- If you use any kind of cloud computing, do you have automation scripts (aka *Infrastructure as Code*, short *IaC*) for setting up and maintaining your cloud environment (e.g. Azure ARM templates, *Terraform* scripts)?
  - If you do, describe the process and tools.

> On a scale from 1 (worst) to 10 (best), how would you rate the maturity of your CD process?