Technical Debt

4 Dec 2024

This enhanced strategy accounts for the inability to mark false positives as resolved while leveraging CI in pull requests to minimize future debt accumulation. By combining manual oversight, a technical debt registry, and gradual process improvement, this approach ensures sustainable debt reduction.

Challenges

1. **False Positives:**
    - The current tool (SonarQube) is not fully aware of the deployment/infrastructure technologies, leading to false positives that cannot be marked as resolved.
    - Examples include misclassified issues such as random values for centroids being flagged as a possible security breach.
2. **Preventing Future Debt:**
    - Continuous Integration (CI) has been implemented in pull requests (PRs) to block new technical debt from being added to the codebase.

Strategy

Our technical debt strategy prioritizes addressing the most critical and high-priority issues first, ensuring that risks with the greatest potential impact on system reliability, security, and performance are resolved quickly. Once these high-severity issues are mitigated, we shift focus to tackling medium and low-priority items in a phased manner, incorporating them into regular maintenance cycles or dedicated sprints (e.g. Sprint 3). This approach allows us to stabilize the most pressing risks early while progressively improving the overall maintainability and quality of the codebase by addressing less critical concerns over time.

Levels of Risk and Actions

1. High-Risk Items

Characteristics:

- High-severity issues identified by SonarQube and verified as true risks.
- Items directly impacting production reliability, performance, or security.

Actions:

1. **Verification & Manual Review:**
    - Create a dedicated review process for high-severity issues.
    - Pair SonarQube findings with additional tools or manual testing to validate flagged items.

    (E.g. using our git flow and issues system to create and store all information in GitHub or in YT)

2. **Immediate Resolution:**
    - Resolve confirmed issues promptly with a priority on critical business impacts.
3. **Temporary Workaround for False Positives:**
    - Use inline code comments or annotations to suppress irrelevant warnings.
    - Document false positives, if necessary, in a shared debt log to track and communicate these cases.
4. **Tool Customization:**
    - Configure SonarQube's ruleset to better align with your technology stack (e.g., custom rules for your image-processing logic).
    - Explore complementary tools to fill all possible gaps of the main technology

---

2. Medium-Risk Items

Characteristics:

- Issues that may reduce maintainability, scalability, or efficiency but do not present immediate risks.

  (Typically includes unoptimized code and non-critical code smells.)

Actions:

1. **Prioritized Debt Reduction Sprints:**
   - Plan dedicated sprints to address clusters of medium-risk issues.
   - Align with CI processes to ensure no new issues are added to modules being refactored.
2. **Documentation for False Positives:**
   - Tag and group recurring false positives to inform long-term mitigation strategies.
3. **CI Enhancements:**
   1. Configure CI to allow PRs to pass even when false positives are flagged

      (e.g., through manual overrides or tagging known issues as “deferred”, needs a spike to tackle the best solution).

---

3. Low-Risk Items

Characteristics:

- Cosmetic or negligible issues unlikely to have an impact on functionality or maintainability.
- Often includes minor code smells or low-priority warnings.

Actions:

1. **Defer Action:**
   - Log low-risk issues in the technical debt registry but defer addressing them unless bundled with higher-risk fixes.

2. **Educate Developers:**
    - Update coding standards and guidelines to prevent common low-risk issues from reoccurring.
    - Perform in retrospective meeting a brief recup of the bad practises and why they still are used.

Cross-Level Actions (Systemic Improvements)

1. **Future Debt Prevention (CI in PRs):**
    - Integrate SonarQube checks into CI pipelines for every pull request to block the introduction of new technical debt.
2. **Technical Debt Registry:**
    - Maintain a shared, updated registry of all unresolved issues (including false positives).

      (E.g. use issues or a defined board/system of epic and tasks)
    - Use this registry to inform long-term planning and tool calibration efforts.
3. **Tool Accuracy and Supplementation:**
    - Improve features to better address your tech stack's specific needs.
    - Explore supplementary tools to fill gaps in SonarQube's analysis and validate critical findings.
4. **Cultural Emphasis on Quality:**
    - Promote a culture of addressing technical debt incrementally and proactively.
    - Reward developers for reducing debt as part of regular sprints.
