CITS5501 Workshop 8 – Quality assurance strategies

Arran Stewart

1 Overview

Our quality assurance (QA) strategy for Stercom Technologies comprises four elements:

- careful validation of system requirements
- thorough and continuous testing
- secure development processes
- formal verification of critical components

2 Strategy Elements

2.1 Careful validation of system requirements

Validation is "the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders" [1]. We should determine the business needs of the customer, and verify that the proposed system does indeed meet those needs. In this case, we might ask what business purpose is served by using blockchain technology (which permits multiple, mutually untrusting users to store state [2]). If standard approaches to ordering services (e.g., a web-facing system, with appropriate encryption of traffic and back-end data) could instead meet the customer's business needs, then development of the system could be made simpler and cheaper (since finding developers and QA staff with experience in blockchain systems is likely to be expensive and difficult).

2.2 Thorough and continuous testing

Thorough testing should be performed at all levels, from unit tests through to integration tests and system tests, and a continuous integration (CI) system used to run unit and integration tests whenever code is committed to version control. Full system tests are likely to take longer to run, and might only be run perhaps once per day. Quality of tests can be assured through (a) formal review of the tests, and (b) the use of techniques such as mutation testing (though it may not be possible to perform this for the entirety of a large system).

The system will be used by non-technical staff of Stercom's clients, and must have high availability, so it will be particularly important to:

- carry out performance testing, to see how the system performs under high loads; and
- evaluate the user experience of the software, for instance by performing specialised UI testing (carried out by trained UI testing experts), and running pilot tests of the software, in which a small number of end users trial the system.

We would also recommend a staged roll-out of the software – perhaps to different segments of clients at a time – so that if performance issues are found, there is a chance to correct them before full roll-out.

2.3 Secure development processes

Given that security is a particular concern in this case, we would develop a separate Secure Development Plan, outlining the methodologies that will be employed in designing, developing and testing the system so that security requirements are met.

This could include activities such as:

- performing security reviews of both project designs and code, to ensure that secure coding and design techniques are applied
- potentially, adopting a secure coding standard, such as the OWASP Secure Coding Practices [3]
- conducting penetration testing (probably by a specialised penetration testing firm) once the system is close to completion, and before release
- formal verification of critical components (detailed further in the next section).

2.4 Formal verification of critical components

Although it may not be feasible or cost-effective to formally verify the entirety of the system, we should be able to apply formal methods to components of the system we identify as being particularly critical (for instance, the blockchain component of the system, if blockchain technology ends up being used).

We would recommend that for those critical components, it should be formally verified that the components meet their specifications, using an interactive theorem-prover system (such as Coq or Isabelle).

3 Automation

Some of the elements of the strategy outlined above lend themselves to easy automation. For instance, unit tests, integration tests, mutation tests and system tests can all be run automatically. The process of devising a secure development process cannot be automated, but aspects of secure development certainly can – for instance, static analyses could be run to check for common security errors and anti-patterns.

Other elements, however, are inherently manual tasks – for instance:

- validation of user requirements
- formal reviews of project artifacts
- procuring the services of a penetration testing firm

4 Conclusion

Each of the elements above assist in delivering a high-quality system. Validation of requirements ensures that the system meets the customer's needs; thorough and continuous testing aims to discover and remove defects; our secure development process aims to minimize security issues with the delivered product; and formal verification of critical components will ensure that they precisely meet their requirements.

References

[1] IEEE. *IEEE Guide – Adoption of the Project Management Institute Standard: A Guide to the Project Management Body of Knowledge*. IEEE, 2011.

[2] Karl Wüst and Arthur Gervais. Do you need a blockchain? Cryptology ePrint Archive, Report 2017/375, 2017. https://eprint.iacr.org/2017/375.

[3] OWASP. OWASP Secure Coding Practices Checklist. https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_Checklist.