Consortium for Information & Software Quality™

How Do You Measure Software Resilience?

What is Software Resilience?

There is rough but imperfect agreement about what the term 'software resilience' means. Some representative definitions of software resilience include the following.

CISQ Cyber Resilience Forum – *Cyber Resilience is the ability to architect and build software that can withstand malicious attacks and continue to operate in unexpected circumstances.* https://www.it-cisq.org/cyber-resilience/

Samir Nassar, IBM – *Software solution resilience refers to the ability of a solution to absorb the impact of a problem in one or more parts of a system, while continuing to provide an acceptable service level to the business.* *https://www.ibm.com/developerworks/websphere/techjournal/1407_col_nasser/1407_col_nasser.html*

Vilas Veeraraghavan, Walmart Labs – *The goal of Resilience Engineering is to help sustain an application ecosystem where unexpected failures in infrastructure and dependencies cause minimal disruption to the user experience*. https://medium.com/walmartlabs/charting-a-path-to-software-resilience-38148d956f4a

Adrian Hornsby, Amazon Web Services – *Resilient systems embrace the idea that failures are normal, and that it's perfectly OK to run systems in what we call 'partially failing mode'*. https://medium.com/@adhorn/patterns-for-resilient-architecture-part-1-d3b60cd8d2b6

Because of its focus on sustaining user functionality, 'software resilience' is often extended to include the quick recovery of systems after a disruptive incident. Scanning the numerous definitions of software resilience leads to a combination of the following attributes. A resilient software-intensive system can:

1) experience a failure in one or more of its constituent components (hardware, software, network, etc.), and/or
2) encounter unexpected inputs or external conditions, and/ or
3) be under malicious attack from internal or external sources,

and yet

a) continue to provide a useful level of functionality to the user, and

b) recover disrupted functions quickly after a disruptive incident.

How Does Resilience Relate to Software Quality Standards?

'Resilience' has not been defined in any of the standard software product quality models. However, there are synonymous quality attributes in most of them. The dominate model for software and system product quality is ISO/IEC 25010, soon to be revised as ISO/IEC 25010-2. ISO/IEC 25010 includes eight quality characteristics (Figure 1), each elaborated into subcharacteristics. 'Reliability' is the ISO/IEC 25010 quality characteristic under which the concept of resilience best falls. The subcharacteristics under Reliability most aligned with the typical descriptions of resilience are 'Availability', 'Fault-Tolerance', and 'Recoverability'.

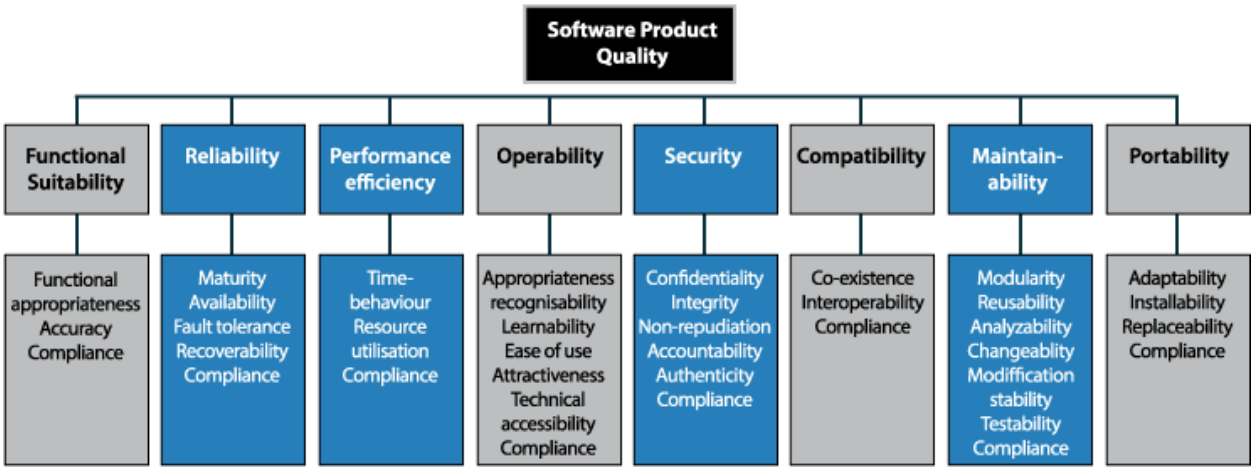

Figure 1. ISO/IEC 25010 software product quality model.

ISO/IEC 25023 is the standard that provides measures for the subcharacteristics of the quality model. It only lists measures related to the external behavior of a system such as downtime, incidents, and recovery speed. It does not provide measures related to system flaws that degrade resilience, or to architectural components that enhance resilience. Consequently, the ISO standards do not provide internal, structural measures of resilience, that is, a way to detect and measure attributes relevant to resilience from analyzing the software system itself.

The Consortium for Information & Software Quality (CISQ) has developed specifications approved by OMG as international standards for measuring four of the eight quality characteristics in ISO/IEC 25010 (shown in blue in Figure 1), namely Reliability, Security, Performance Efficiency, and Maintainability. These measures are developed from detecting and counting severe violations of good architectural and coding practice. The current best measures of a software product related to resilience are a combination of CISQ's Reliability and Security measures. These measures evaluate flawed structures inside a software system that

can reduce its resilience. CISQ measures include flaws related to many of the architectural components that would be included to improve resilience.

Is Resilience Measurable?

An assessment of software system resilience requires that measures of system performance be compared to resilience thresholds. Resilience thresholds can be established from at least these three elements:

1) which transactions or services are critical to sustain for users,
2) how fast disrupted services can recover, and
3) the proportion of requests that can be lost during a disruptive incident.

Measuring a software-intensive system's capability for resilient performance requires a combination of two different analyses. The first is a static analysis of the existence of architectural elements that enhance resilience and avoidance of flaws that degrade it, while the second is a dynamic analysis of a system's performance under disruptive events. A full measure of 'Resilience' is beyond the current state of static measurement technology because automated analysis scripts for some architectural components required for resilient systems have not been fully defined or developed. However, many scripts are within the state of the art of software analysis, although some require system-level architectural analysis, especially those relevant to multi-layer distributed systems. Currently the analysis of components yet to be covered by automated scripts is most often accomplished through manually conducted formal inspections of the architecture and/or source code of a system. However, there is a continuing shift of manual capabilities to automation.

Since software resilience is defined as sustainable behavior, a full measure of resilience requires that static analysis of the system's internal structure be coupled with dynamic analysis of system behavior. Static analysis determines a resilience element's properly designed existence in the code, while dynamic analysis evaluates its ability to sustain resilient performance. The results of dynamic execution and related testing are limited by their ability to adequately simulate actual disruptive conditions. An approach often called 'chaos engineering' includes guidelines, methods, and processes for engineering and evaluating system resilience.

What Architectural Attributes Affect Resilience?

Advances in measuring resilience will require analysis and measurement of the architectural components of a software-intensive system. Some, but not all, of the architectural attributes needed to ensure the resilience of a software-intensive system, especially those with a distributed architecture, include:

- *Redundancy* – duplication of components (or services) in order to increase availability in case of failure in one or more system components.
- *Load balancers* – checks services to determine if they can handle requests and spreads processing loads across them.

- *Innate scaling* – ability to scale dynamically to meet changes in demand or load.
- *Infrastructure as code* – the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.
- *Immutable infrastructure* – components are swapped/replaced for every deployment, rather than being updated in place.
- *Stateless application* – the application treats all client requests independently of prior requests or sessions and should never store any information on local disks or memory. This also includes freeing up resources (e.g. memory) quickly.
- *Backoff algorithms* – increases the rate at which retries are performed gradually, thus avoiding network congestion and ultimate meltdown.
- *Timeouts* – termination of connections that degrade performance rather than freezing because the connection pool has run out of connections.
- *Idempotent operations* – an operation that can be repeated without duplicating results, side effects, or failure of the application.
- *Circuit breakers* – monitors the number of consecutive failures between a producer and a consumer. If the number of failures passes over a threshold, the circuit breaker object *trips*, and all attempts by the producer to invoke the consumer will fail immediately or return a defined fallback.
- *Caching* – a hardware or software component that stores data so future requests for that data can be served faster and can be configured to improve resilience during denial of service attacks.
- *Input vetting* – evaluation of all inputs to ensure they are not harmful or malicious.

In summary, the measurement of resilience requires a combination of three quantitative elements:

1) a determination of resilience thresholds,
2) a static analysis of resilience capabilities and flaws, and
3) a dynamic analysis of system behavior under realistic disruptive conditions.

At the current rate of advancement in static software analysis, the detection and measurement of most architectural structures related to resilience should be automated within 3-5 years. The good news is that some aspects of software that affect resilience are detectable using existing standards and technology. Thus, the more advanced current technologies can provide measures that are reasonable estimators of software system resilience.