Platform-Independent Runtime Quality Metrics for Integrations

RIK SMALE, University of Twente, The Netherlands

One way of measuring the quality of software systems is through the usage of metrics. Having a higher quality of software is needed to be able to differentiate in the current market and allows quicker development of new features and reduces the time spent on maintenance. Just like with software systems in general this holds for integrations as well. Integrations play a big role in the digital transformation by connecting various systems together to create more efficient processes. While general runtime quality metrics have been proposed in the past, none have looked specifically into measuring the quality of integrations this way. This paper does so by first identifying specific ISO 25010 (sub)characteristics, like time behaviour, resource efficiency and reliability and provides metrics from previous research shown to work for software systems in general. It then validates these through an expert to provide a list of runtime quality metrics that can be used to measure the quality on an integration.

Additional Key Words and Phrases: System Integration, Quality Metrics, Runtime Quality

1 INTRODUCTION

Over the last years many businesses have been looking into how to digitally transform their processes. The first step of that is to remove the usage of paper and keep track of that information digitally. Since companies have many different processes this information is stored in various systems. As these systems become less monolithic, separate systems might come into use which will increase the complexity of the landscape of a company even more. To tackle this complexity systems might share their data through integrations. With integrations become more important, the quality of these systems should also be able to be held to some standard. Therefore research needs to be done into measuring the quality of these integrations.

2 PROBLEM STATEMENT

While runtime (quality) metrics have been researched in various settings before [10, 13, 18], no specific research has been done into metrics for integrations. Integrations have different goals than other systems so this research will look into whether these metrics are suitable for usage in integrations. This paper hopes to identify these metrics and determine which are suitable for usage.

2.1 Research question

Using the problem statement the following research question can be composed;

Which runtime metrics are suitable for usage in measuring the quality of integrations?

To answer this research question, the following sub-question will be used;

Which runtime metrics have been identified for measuring quality of software systems in earlier research?

3 METHODOLOGY

The methodology of this research is inspired by the Design Science methodology[7]. The first step is researching runtime quality metrics aimed at general software projects identified in the past. After that an artefact will be designed, which will be a set of possible metrics. Those will be validated by an expert. The metrics that will be reported in this research will be done according to the Goal Quality Metric method [16] and mapped to (sub)characteristics of ISO 25010 [8].

4 RELATED WORK

4.1 Runtime Quality

ISO 25010 [8] defines two models that can be used for software quality; a quality in use model and a product quality model. The product quality model has eight characteristics, divided up in thirty-one subcharacteristics, that can be used to categorise product quality attributes. This research focuses on the quality of the runtime of a program, which is the seventh and final stage of the program life cycle. [12, 17] The runtime is when the program is actually executing. Measuring metrics on the runtime of a program allows for more platform-independent metrics since the other stages might differ in programming languages and platforms. Most of the subcharacteristics of ISO 25010 can be used to define runtime quality.

4.2 Goal, Question, Metric approach

One possible way of defining metrics is the Goal, Question, Metric approach, first proposed by Basili and Weiss (1984) [2]. The measurement model that is the result of this approach has three levels [1]; The first level is the conceptual level (Goal) in which a goal is defined from various points of view. It details the purpose, the issue, the object (or process) and the viewpoint. The second level is the operational level (Question) in which questions are used to characterise how the goal can be assessed and/or achieved. These are done from a chosen viewpoint. The third and final level is the quantitative level (Metric) in which a set of data, associated with every question, is used to answer the question quantitatively. This data can either be objective or subjective, depending on if they only depend on the object that is measured or also the viewpoint which was selected for the operational level. As shown in figure 1, metrics can be used for multiple questions.

4.3 Integrations

In order to be able to measure the quality it is needed to know what the object is that is being measured. This paper looks into the integration of enterprise applications defined as "unrestricted sharing of data and business processes among any connected applications and data sources in the enterprise" [11] . The systems that do this integration have some basic characteristics [4] ; they

TScIT 38, February 3, 2023, Enschede, The Netherlands

Fig. 1. Visualisation of relation between Goals, Questions and Metrics

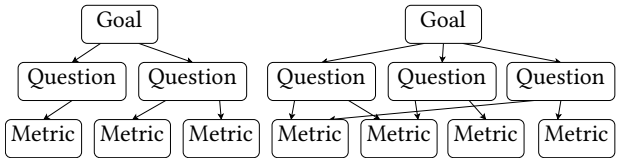

Table 1. Resulting ISO 25010 subcharacteristics

Performance Efficiency
Time Behaviour
Resource Utilisation
Reliability
Maturity
Availability
Fault Tolerance
Recoverability

integrate at the business level including business and data processes; they re-use applicable business processes and data; they involve no real understanding of specific system functions; they do not require source code or code administration rights to any of the applications to the integrated; and they generally require no changes to the hardware infrastructure of the applications to be integrated. Since these integration systems run separately from the applications to be integrated it is possible to do runtime measurements independently as well.

4.4 Runtime Quality of Integrations

Garg (2022) [5] has identified two characteristics and four subcharacteristics of ISO 25010 that might provide insights in the quality of integrations. The characteristics are reliability and security. The four subcharacteristics are time behaviour, resource utilisation, modularity, and modifiability. Modularity and modifiability, relating to the characteristic of maintainability, show the degree to which a program has their components separated and modifications can be made without impact to the product quality. These are more related to the edit time of a program. Therefore these two subcharacteristics will not be taken into account for this research. The security characteristic is also not taken into account for this research for the same reason. The resulting subcharacteristics are summarised in Table 1

5 RESULTS

The following section contains a non-exhaustive list of runtime quality metrics that might be able to be used to measure the quality of integrations. This list is non-exhaustive since some metrics found are not able to be measured and/or have not been proven to be suitable for usage in general software systems yet. These metrics are mapped to (sub)characteristics of ISO 25010 [8] and will be discussed in their respective sections. Combined with the ability to change the time span of the measurements, these metrics, can tell much about the quality on an integration according to the expert.

Table 2. Goal Question Metric model for Performance Efficiency metrics

Goal	Purpose Issue Object Viewpoint	Measuring the runtime quality of an integration as a software engineer
Question Metrics	Q1	Time Behaviour
	M1	Mean response time [9]
	M1a	Response time under various load conditions [6]
	M2	Response time adequacy [9]
	M3	Mean turnaround time [9]
	M4	Turnaround time adequacy [9]
	M5	Mean throughput [9]
	M6	Capacity [6]
Question Metrics	Q2	Resource Utilisation
	M7	Mean processor utilisation [9]
	M8	Mean memory utilisation [9]
	M9	Mean I/O devices utilisation [9]
	M10	Bandwidth utilisation [9]
	M11	Network usage [6]
	M12	Disk usage [6]

Some metrics in specific, as discussed below. Different methods of displaying a metric can also be used for other metrics, like the change under various load conditions or the adequacy

5.1 Performance Efficiency

Table 2 defines the GQM model for metrics related to the Performance Efficiency characteristic, mapped to the subcharacteristics. The metrics are explained in the section of their subcharacteristic.

5.1.1 Time Behaviour. Metrics related to Time Behaviour look into response and processing times and throughput rates of the integration. [8] The response time is the time that the system takes to respond to a specific task [15] and the turnaround time is the time that the system takes to complete the task [15]. Both of these attributes can be used to calculate the following metrics;

- The mean time in a specified time period. For the response time this is calculated by the sum of the time taken divided by the number of responses. For the turnaround time this is the sum of the times between starting and completing the task divided by the number of tasks.
- The mean time under various load conditions in specified time periods. These metrics are taken from a subset of data of the previous metric. Haindl and Plösch [6] propose to use the normal load and the peak load, where the biggest amount of requests come in.
- The time adequacy. Some integration might have a target response time. This target might be from a user, internally decided or contractually obligated. This metric uses the mean time as calculated before divided by the target time.

The throughput is the amount of tasks that are completed in a certain time period. From this the mean can be used, but also the maximum amount can be seen as the capacity of the integration [6].

Table 3. Goal Question Metric model for Reliability metrics

Goal	Purpose Issue Object Viewpoint	Measuring the runtime quality of an integration as a software engineer
Question	Q1	Maturity
Metrics	M1	Mean time between failure (MTBF) [9]
	M2	Failure rate under various load conditions [6]
Question	Q2	Availability
Metrics	M3	System availability [9]
	M4	Mean down time [9]
	M5	Number of reliability incidents [6]
Question	Q3	Fault Tolerance
Metrics	M6	Mean fault notification time [9]
Question	Q4	Recoverability
Metrics	M7	Mean recovery time [9]
	M8	Backup data completeness [9]

5.1.2 Resource Utilisation. While metrics in the Time Utilisation subcharacteristic look into how long it takes to do a certain thing, metrics in the Resource Utilisation subcharacteristic are about how many resources are needed to do that certain thing. These include the processor (CPU), memory (RAM), I/O devices (e.g. hard disk) and transmission bandwidth (over a network). The mean utilisation of these are metrics. Network usage can be measured by two metrics, how much bandwidth is used in a second but also per day. The disk usage can also be measured by several metrics. One is to measure the time that the disk is active, another is the rate on which data is being transferred, and a third is the amount of disk space that is being used up. In talking to the expert, it became clear that there are differing opinions on what is better for the quality of an integration. Integrations that receive the same number of requests throughout the day might have 80%-90% CPU usage and still be of good quality while integrations that receive a varying number of requests might time out in a mean usage of 80%. One metric that could provide more insights to the quality of an integration according to the expert is the usage of I/O devices. Since integrations facilitate processing a request from one system to another, I/O devices like hard disks are often not needed.

5.2 Reliability

Table 3 defines the GQM model for metrics related to the Reliability characteristic, mapped to the subcharacteristics. The metrics are explained in the section of their subcharacteristic.

5.2.1 Maturity. The maturity is defined as the "degree to which a system, product or component performs specified functions under specified conditions for a specified period of time". [8] Metrics that can measure this maturity relate to how often errors or other failures in the system occurs. The main metric in this subcharacteristic is the mean time between failure (MTBF) which can be calculated by dividing the operation time by the number of occurring failures. [15] Another metric derived from this information is the failure rate under various load conditions, as also discussed in subsubsection 5.1.1.

5.2.2 Availability. The availability subcharacteristic has metrics that relate to how often the system is available for use. The system availability can be measured by dividing the actual operation time of the system by the operation time as specified in the operation schedule. [15] This operation schedule can exclude planned downtime due to maintenance and like the adequacy metrics can be determined by e.g. internal agreements or a Service Level Agreement. The expert noted that proving the actual availability can be difficult since it is not possible to know the system is running if there are no requests to it. A metric that can be measured more easily is the mean down time [15] which show how long, on average, a system is unavailable for when a breakdown of the system occurs. The number of times that a breakdown, or another interruption of the operation of the integration occurs can also be used as a metric for the number of reliability incidents.

5.2.3 Fault Tolerance. Fault Tolerance deals with how much errors can occur while the system does not get any downtime. One runtime metric identified in this category is the mean fault notification time. This measures how long the system takes to report a fault until it is detected. Measuring this is more difficult and depends on if you are able to do monitoring on your monitoring systems. The expert mentioned that often this can be measured retroactively since manual efforts are required to figure out when the fault occurred.

5.2.4 Recoverability. Metrics for the recoverability subcharacteristic show if a system can recover data in case of an error or breakdown and how quickly it can do that. The latter can be derived to the mean recovery time. Another metric is the backup data completeness. Unlike the other metrics this metric is not suitable for use in integrations since integrations do not store data, they just process it between other systems.

6 DISCUSSION

6.1 Validation & Limitations

The major limitation of this paper is in the validation. While the metrics have been found from earlier research into software quality, this research validates the usability of these metrics in the specific domain of integrations through one expert. The metrics are shown to have validity through the underlying theory and the usability has validity through content validity through the discussion with the expert. Both these validity criteria are for internal validity as per Meneely et al. [14] Do note that usability in the sense of this paper does not correspond with the usability validation criteria identified by Cavano and McCall [3] since that deals with cost-efficiency instead of suitability of the usage. These metrics can further be validated in several ways. One that was kept in mind for runtime metrics is the trackability in which the metrics change together with the quality factor which it is supposed to measure. Better content validity can also be achieved by consulting more experts.

6.2 Future work

This research only provided a first step in the ability of measuring the runtime quality of integrations. A second step would be

to validate these metrics using data of actual integrations, something that could not happen in this research due to time constraints. After the metrics are validated it would be possible to research possible reference values for these metrics. These values would most likely only work for specific configurations, programming languages and/or platforms, but future research could look into generalising these. Other research could be done to other quality metrics using the identified (sub)characteristics mentioned in subsection 4.4. This could use the other stages of the program lifecycle, mainly the edit time (also called design time). A third possibility is to see if runtime quality can be improved by introducing further best practices and checklists instead of measuring at the runtime. These final two possibilities could especially be used in future research to the security characteristic since these are less related to the other (sub)characteristics researched in this paper.

7 CONCLUSIONS

This paper researches various runtime quality metrics that have been identified in the past and discusses them with an expert to see if they are suitable for usage in measuring the quality of integrations. It starts off with an explanation on the importance of this under-researched topic and gives insights into runtime quality, quality metrics, the Goal, Question, Metric approach and integrations in general before diving into some information on the runtime quality of integrations. Various papers were used to identify runtime quality metrics in six previously identified subcharacteristics of the ISO 25010 standard. The result of this is a set of runtime metrics that have been identified for measuring quality of software systems used to answer the research sub-question. These resulting 20 metrics discussed with an expert. Almost all of these metrics were found to be suitable for usage in measuring quality of integrations at runtime in various degrees. All metrics from Table 2 and the ones from Table 3 apart from backup data completeness are therefore the result to the main research question. Future research can look into visualizing these metrics in a dashboard, amongst other topics identified in this paper.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Lucas Meertens, for his enthusiasm in discussing this topic and keeping up with me throughout the process. I also would like to thank Samet Kaya for his feedback and insights on the resulting metrics.

REFERENCES

[1] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. 1994. The Goal Question Metric Approach.
[2] Victor R. Basili and David M. Weiss. 1984. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering* SE-10, 6 (1984), 728–738. https://doi.org/10.1109/TSE.1984.5010301
[3] Joseph P. Cavano and James A. McCall. 1978. A Framework for the Measurement of Software Quality. In *Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues*. Association for Computing Machinery, New York, NY, USA, 133–139. https://doi.org/10.1145/800283.811113
[4] Naveen Erasala, David C. Yen, and T.M. Rajkumar. 2003. Enterprise Application Integration in the electronic commerce world. *Computer Standards Interfaces* 25, 2 (2003), 69–82. https://doi.org/10.1016/S0920-5489(02)00106-X
[5] Aachi Garg. 2022. Features to Predict Quality of Low-Code Integrations. *TScIT* 37 (2022).
[6] Philipp Haindl and Reinhold Plösch. 2022. Value-oriented quality metrics in software development: Practical relevance from a software engineering perspective. *IET Software* 16, 2 (2022), 167–184. https://doi.org/10.1049/sfw2.12051
[7] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS Quarterly* 28, 1 (Mar 2004), 75–105. https://doi.org/10.2307/25148625
[8] ISO/IEC 25010. 2011. ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.
[9] ISO/IEC 25023. 2016. ISO/IEC 25023:2016, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality.
[10] Markus Klems, David Bermbach, and René Weinert. 2012. A Runtime Quality Measurement Framework for Cloud Database Service Systems. In *2012 Eighth International Conference on the Quality of Information and Communications Technology*. 38–46. https://doi.org/10.1109/QUATIC.2012.17
[11] D.S. Linthicum. 2000. *Enterprise Application Integration*. Addison-Wesley.
[12] Ben Lutkevich. 2021. What is runtime?: Definition from TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/runtime
[13] Sara Mahdavi-Hezavehi, Matthias Galster, and Paris Avgeriou. 2013. Variability in quality attributes of service-based software systems: A systematic literature review. *Information and Software Technology* 55, 2 (2013), 320–343. https://doi.org/10.1016/j.infsof.2012.08.010 Special Section: Component-Based Software Engineering (CBSE), 2011.
[14] Andrew Meneely, Ben Smith, and Laurie Ann Williams. 2010. *Software metrics validation criteria: A systematic literature review*. http://www.lib.ncsu.edu/resolver/1840.4/4133
[15] Ahmad Ruzita. 2019. *Goal oriented software sustainability evaluation model for sustainable software development*. Ph. D. Dissertation. Universiti Utara Malaysia.
[16] Rini van Solingen and Egon Berghout. 1999. *The goal question metric method: A practical guide for quality improvement of software development*. McGraw-Hill.
[17] Stackpath. 2023. What is runtime? - stackpath. https://www.stackpath.com/edge-academy/what-is-runtime/
[18] Luis Emiliano Sánchez, J. Andrés Diaz-Pace, Alejandro Zunino, Sabine Moisan, and Jean-Paul Rigault. 2014. An Approach for Managing Quality Attributes at Runtime Using Feature Models. In *2014 Eighth Brazilian Symposium on Software Components, Architectures and Reuse*. 11–20. https://doi.org/10.1109/SBCARS.2014.13