Features to Predict Quality of Low-Code Integrations

AACHI GARG, University of Twente, The Netherlands

The number of information systems in a business are growing rapidly as new technologies and software become available in the market, which increases the need for integrating these systems for a flow of data between them. These integrations have to be made quickly to keep up with the market developments and they have to be of excellent quality. To achieve this, the use of low code development using iPaaS (Integration Platform as a Service) is on the rise, however, currently, it is not possible to measure the quality of a low code integration specifically. Hence, this research aims to find features that can be used to predict the quality of low code integration in iPaaS. This will be done by conducting a systematic literature review, then the features found will be mapped to ISO, from that, the relevant features will be chosen through a certain criteria and expert guidance. Doing this would be beneficial for the company as it will save them multiple resources such as time and money. The result of this research would a list of features which will be found using a systematic literature review and guidance from experts, and the list could be used to determine the quality of a low code integration in an iPaaS.

Additional Key Words and Phrases: Information systems, integrations, low code development, quality of integration, integration platform as a service, iPaaS

1 INTRODUCTION

As businesses grow, their information systems become increasingly complex. They have different systems for different processes and while this is a feasible option when the business is small, it becomes increasingly impractical with growth. The same data is present in various systems and databases, however, the systems or databases can't communicate with each other, making the information extremely redundant. Integrating the systems together allows for the flow of data between them and interoperability between the systems. Hence, system integration allows the business to be more productive, cost-efficient, and have more reliable data[4]. It also has many other benefits for business processes. In fact, according to reports[9] by the consultancy firm Grand View Research, the market for system integrations is expected to grow to 530 billion dollars in 2025.

One way of creating the system integrations is using low code development. Low code development is a way to create applications with minimal use of coding. For example, one way of low code development is through a drag-and-drop interface as the blocks or modules are already built [10]. The interest in low code development is increasing rapidly; major tech companies like Google, Microsoft, Siemens and more have already created or bought Low Code Platforms or LCP [11]. A global research company Gartner has predicted that LCP will be used in over 65 percent of all development projects by 2024 [2]. There are multiple reasons behind this increase, the major one being that the market changes rapidly, and to keep up with it, companies also need to update and create their IT environment rapidly and LCPs allow them to do that. For this reason, LCPs are being used to create system integrations. Low code development is often done on iPaaS, that is, integration Platform as a Service. iPaaS is a platform that allows its users to build integration on the cloud, without any need for installing hardware. This, in combination with low code, is preferred by companies as it eliminates the need for any middleware or hardware, which also means that they don't have to maintain that [3]. Moreover, they need less resources, such as memory or CPU, as it is possible to buy that on the cloud [12].

1.1 Problem Statement and Research Questions

Since businesses rely on these integrations, their quality is of utmost importance. Integrations of bad quality could result in loss of data, inefficient functions, and many other problems. However, currently, there are either none or very limited ways to identify the quality of a low code integration on an iPaaS. If it turns out that the integration is of bad quality, the company will have to re-do the integration. This would waste a lot of their resources, time, and hence money. For the companies which work on creating integrations for other businesses, this might even mean that if their new developer makes a mistake, then the company would have to take the responsibility and fix the integration. By determining the quality of the integration before its deployment, the company will be able to take necessary actions to improve the integration if needed.

This problem leads us to the objective of this research and hence the research question:

RQ 1: *What features can be used to predict the quality of a low code integration?*

2 METHODOLOGY

To answer the research question, first, a systematic literature review will be conducted to find previous research on this topic from which the relevant papers will be chosen. Ideally, these papers will contain features for measuring the quality of an integration. Then, the features found in these papers will be mapped to ISO 25010 [7], and a final list of features will be made. Out of these, some features will be selected based on certain predefined criteria and advice from experts to find the most relevant features.

2.1 Literature Review

To do a systematic review of the research already done on the topic of the quality of an integration, a number of keywords are selected. These keywords are "integration", "information system", "success factors", "quality factors", "features", "quality", "application" and "integration". These keywords will be searched on the platforms Scopus and Web Of Science. The search term is made using a combination of these keywords and the connectors AND and OR. The

TScIT 37, July 8, 2022, Enschede, The Netherlands

resulting search term is:

("System" OR "Information system" OR "application") AND ("success factors" OR "quality factors" OR "quality features" OR "quality" OR "quality metrics") AND "integration")

At first, only the titles of the papers will be searched for these keywords. Then, the relevant articles will be selected and a further selection will be done by reading their abstracts to see which ones are relevant. In the end, only the papers which contain a list of features that can be used to measure the quality of an integration will be selected.

2.2 Mapping to ISO 25010

The features found in the papers from literature review will then be mapped to the sub-characteristics present in ISO 25010 [7] (Appendix A), which is a quality model for evaluating software. As this quality model has several well-defined characteristics and the similarities between general software and integrations have been noted multiple times in research before ([6] and [8]), it was decided that the characteristics in ISO 25010 will be used as a basis for selecting the features, after being mapped to the features from the selected research papers. Moreover, doing this ensures that the same factor is not repeated multiple times from the sources.

To do the mapping, first, the features will be mapped according to their names. That is, if a feature has the same name in a research paper and ISO, then they will be mapped to each other. It is important to note that if a feature is being mapped to a main characteristic instead of a sub-characteristic in ISO, then all the sub-characteristics will be mapped to that feature. The rest of the features will be mapped according to their definitions to see which ISO feature are they the most similar to. If the features from the papers are found to influence ISO features directly, then they will be mapped to that ISO feature as well.

In the end, a final list of relevant features will be made which will contain all the ISO features, excluding the ones which were not mapped to any feature in the papers. Also, any features from the papers which could not be mapped to ISO will also be added to this list to ensure that nothing is missed.

2.3 Selection of features

From the features found in the previous step, not all of them will be valid for this research. So, the selection of the features will be done based on certain criteria. To start with, the features will be divided into functional and non-functional requirements. As there is no globally accepted definition for non-functional requirements, some definitions as collected from various papers and summarized by [1] are:

(1) "The required overall attributes of the system, including portability, reliability, efficiency, human engineering, testability, understandability, and modifiability."
(2) "The term "non-functional requirement" is used to delineate requirements focusing on "how good" software does something as opposed to the functional requirements, which focus on "what" the software does."

As in this research, the focus is on the quality of the software, that is the non-functional requirements, and not on whether it meets the business requirements, the functional requirements will not be considered. Hence, only the features which are regarding the quality of the code will be kept.

Then, from the features that are an integral part of every low code integration platform will be removed. This is due to the fact that the selected papers are focusing on integrations in general and not low code integrations. So, for example, some features found in these papers will already be present in every low code integration and hence it will not make sense to measure them.

After this, experts in the field will be consulted to confirm whether the selected features are relevant and complete. If needed, features might also be removed or added based on their advice.

3 RESULTS

3.1 Literature Review

The first step in finding the factors to predict the quality of an integration was conducting a literature review on Scopus and Web of Science.

(1) Scopus: Using search term and restricting it to the title on Scopus resulted in 342 papers. The titles of these results were manually selected based on their relevance to the research. Most of the titles were regarding the quality of integrations in a specific industry or enterprise systems and hence were not selected. This led to 8 remaining papers, whose abstracts and conclusions were analysed to check which ones contain lists with features to measure the quality of integrations. Based on this the paper [5] was chosen.
(2) Web of Science: Following the same methodology on Web of Science resulted in 157 papers. From this, 3 papers were selected based on their titles. After checking the abstracts and conclusions, one paper [13] was finally selected from Web of Science.

So the two papers which will be used in this research are "Identifying quality factors of information systems integration design" by Zikra, Stirna and Zdravkovic (Paper 1) [13] and "Success Factors of Application Integration: An Exploratory Analysis" by Gericke, Klesse, Winter and Wortmann (Paper 2) [5]. Both these papers identify and list important factors which can be used to measure the quality of an integration, which meets the requirements set earlier. However, neither of them are written for the purpose of measuring the quality of low code integrations, which is the goal of this research paper.

3.2 Mapping to ISO 25010

Since integrations and software have similar characteristics and ISO 25010 [7] has well defined features, a matrix map was made for them following the methodology. For the first step, features which had the same names as ISO features were mapped together. For example, the factor “Performance, reliability and scalability” from paper 2 were mapped to the ISO features “Performance” and “Reliability”. In the cases where they did not have the same name, the definitions were compared and the feature was mapped according to what it meant. For example, in paper 1, the features “accuracy” and “freshness” were mapped to the ISO feature “functional correctness”. It was noticed that many features in paper 2 had to be mapped according to which ISO feature they influence, for example, “separation of software layers” would improve the ISO features of “reusability”, “analysability” and “modifiability”. The matrix map can be found in table 1.

While most of the factors present in the papers could be mapped to ISO, there were 13 factors in paper 2 which could not be mapped, so they were added to the final list of the features. Along with those, the features of ISO which were mapped to at least one paper were added, so only "Adaptability", "Installability" and "Replaceability" were removed from ISO. The final list of features can be found in table 2.

ISO 25010	Paper 1	Paper 2
Functional Suitability		
Functional Completeness	x	
Functional Correctness	x	
Functional Appropriateness	x	x
Performance Efficiency		
Time Behaviour	x	x
Resource Utilisation		x
Capacity		x
Compatibility		
Co-existence	x	x
Interoperability	x	x
Usability		
Appropriateness Recognisability	x	
Learnability	x	
Operability	x	
User Error Protection	x	
User Interface Aesthetics	x	x
Accessibility	x	
Reliability		
Maturity	x	x
Availability	x	x
Fault Tolerance	x	x
Recoverability	x	x
Security		
Confidentiality	x	
Integrity	x	
Non-repudiation	x	
Accountability	x	
Authenticity	x	
Maintainability		
Modularity		x
Reusability		x
Analysability	x	x
Modifiability	x	x
Testability	x	x
Portability		
Adaptability		
Installability		
Replaceability		

Table 1. Matrix Mapping

3.3 Selection of features

The final list from which the selection of the features will be done had 42 features which can be found in table 2. Following the methodology, the features were divided into function and non-functional requirements. As in this research, the concern was regarding the way the application is coded, that is, non-functional requirements, only those requirements will be kept. The removed features were mostly taken from paper 2, and they were concerning the quality of project management, which is not a non-functional requirement. Other than that, the features under the ISO “Usability” group were also removed since they are concerned with the users of the software and the front-end of the design, not with the quality of the code. Similarly, all the features under the ISO “Functional completeness” group were removed because they about the alignment of business and IT. Hence, this step led to the removal of 21 features and the remaining features can be found in table 3.

Then, the features which are inherently present in any low code integration on an iPaaS were removed that is, the two features “co-existence” and “interoperability” were not be considered as the purpose of an integration is to make systems compatible. Similarly, the features “Capacity”, “Reusability”, “Analysability”, “Availability” and “Testability” are already part of the functions of an iPaaS and hence will not be used.

For selecting the most relevant sub-features out of the rest, an expert in the field of low code integrations was consulted. Based on this discussion, all the factors in the ISO group “Reliability” were merged together. That is, reliability will be taken as the main feature where the metrics will be defined based on “Maturity”, “Fault Tolerance” and “Recoverability”. The same thing will be done for “Security” as well. The expert also noted that data is an essential part of an integration but has not been considered in this list, hence “Data Reliability” was added to the list with the definition “Degree to which the data in the system or product is available, reliable and complete under specified conditions”.

Based on the selection and advice from the expert, the features and their definitions from ISO 25010 [7] are as follows:

Functional Completeness	Maturity	Methods
Functional Correctness	Availability	Integration Strategy
Functional Appropriateness	Fault Tolerance	Packaged Applications
Time Behaviour	Recoverability	Number of Integration Tools
Resource Utilisation	Confidentiality	Number of Applications
Capacity	Integrity	Number of Platforms
Co-existence	Non-repudiation	Architecture Models/Modeling
Interoperability	Accountability	Scorecards
Appropriateness Recognisability	Authenticity	Principles and Guidelines
Learnability	Modularity	Business/IT Cooperation Capability
Operability	Reusability	Coordinated and Integrated Processes
User Error Protection	Analysability	Documentation of IT Processes
User Interface Aesthetics	Modifiability	Workflow Management
Accessibility	Testability	Clarity of Responsibilities

Table 2

Time Behaviour	Fault Tolerance	Modularity
Resource Utilisation	Recoverability	Reusability
Capacity	Confidentiality	Analysability
Co-existence	Integrity	Modifiability
Interoperability	Non-repudiation	Testability
Maturity	Accountability	Packaged Applications
Availability	Authenticity	Number of Integration Tools

Table 3

(1) Time Behaviour: "The degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements."
(2) Resource Utilisation: "The degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements."
(3) Reliability: "Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time."
(4) Security: "Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization."
(5) Modularity: "Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components."
(6) Modifiability: "Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality."
(7) Data Reliability: “Degree to which the data in the system or product is available, reliable and complete under specified conditions”

These features can act as guidelines for companies while developing a low code integration on an iPaaS and they can be used to measure its quality.

4 DISCUSSION

4.1 Limitations

Some possible limitations of this research could be that only two papers were found during literature review. It might be possible that other libraries would have had other relevant literature as well, however, considering the scope and time for this research project it was not possible to search in more libraries.

Another limitation could be that only one expert was consulted for this paper. If more experts were consulted, then it might be possible that other features would have been added or removed from the list.

4.2 Future work

This research can form a very good basis for further research in this field. Some ideas which can be taken forward is creating metrics for measuring the factors which were found in this research. An option to work out the metrics could be using Goal Question Metric, and table 4 shows some potential metrics which were found during this research however, they could not be validated because of the limited time. The next step after validating these metrics, or defining new ones, would be making thresholds which could be very useful for any company who wants to implement low code integrations on iPaaS.

If the thresholds have been defined, then the next step would be to find features which can predict the quality of the integration at the design stage itself. This could be done by building on this research, as qualities at the design stage which influence the list of features found in this research could be noted as guidelines and would help improve the quality of the integration from the start itself.

5 CONCLUSION

This research presents a list of features which can be used to predict the quality of a low code integration on an iPaaS. These features can be found in section 2.2. To make this list, first a systematic literature review was done. The research papers found through the literature review were then mapped to ISO, and a list of all the possible features was made using the mapping. Then, the relevant features were selected from this list based on criteria which had been defined in the methodology.

Goal	Purpose	Measure
	Issue	The quality of an
	Object	iPaaS integration
	Viewpoint	From a researcher's viewpoint
Question	Q1	Time Behaviour
Metrics	M1	Average throughput rate
Question	Q2	Resource Utilisation
Metrics	M2	Average memory usage
	M3	Average CPU usage
Question	Q3	Reliability
Metrics	M4	Is there alerting if something goes wrong?
	M5	How does the integration perform under a stress test?
Question	Q4	Security
Metrics	M6	Is there access control for making changes?
	M7	Is the data encrypted?
Question	Q5	Modularity
Metrics	M8	What is the coupling value?
	M9	What is the cohesion value?
Question	Q6	Modifiability
Metrics	M10	How many nodes are connected to gateway nodes?
	M11	What is the gateway heterogeneity?
Question	Q7	Data Reliability
Metrics	M12	What happens to the data in case of failure?
	M13	What percentage of data is transmitted?
	M14	Is the incoming data validated?

Table 4. Goal Question Metrics

This features can be extremely useful for a company who wants to measure the quality of its integration before it has been deployed. Moreover, this research can be used as a good base for future work such as defining metrics and thresholds for these features.

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor for this thesis, Lucas Meertens, for giving me the opportunity to work on this project. I would also like to thank Samet Kaya from the company eMagiz, who helped me validate my research and gave me insights into very important factors by taking out the time to meet me regularly.

Finally, I would like to acknowledge the work done by Jorn Boksem at eMagiz, as my research continues and makes use of his earlier but unpublished work into this topic.

REFERENCES

[1] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. 2000. Non-functional requirements in software engineering. (2000). https://doi.org/10.1007/978-1-4615-5269-7

[2] Shannon Duffy. 2021. Salesforce is named a leader in the 2019 gartner magic quadrant for low code application platforms. https://www.salesforce.com/blog/gartner-lcap/

[3] Nico Ebert, Kristin Weber, and Stefan Koruna. 2017. Integration platform as a service. *Business & Information Systems Engineering* 59, 5 (2017), 375–379.

[4] AltexSoft Editor. 2021. System integration: Types, approaches, and implementation steps. https://www.altexsoft.com/blog/system-integration/

[5] Anke Gericke, Mario Klesse, Robert Winter, and Felix Wortmann. 2010. Success Factors of Application Integration: An Exploratory Analysis. *http://www.alexandria.unisg.ch/Publikationen/72470* 27 (01 2010). https://doi.org/10.17705/1CAIS.02737

[6] Laura González, Felix Garcia, Francisco Ruiz, and Mario Piattini. 2010. Measurement in business processes: A systematic review. *Business Process Management Journal* 16 (02 2010), 114–134. https://doi.org/10.1108/14637151011017976

[7] 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.

[8] Jan Mendling. 1970. *Metrics for Business Process Models*. Vol. 6. 103–133. https://doi.org/10.1007/978-3-540-89224-3_4

[9] Grand View Research. 2022. System integration market share report, 2022-2030. https://www.grandviewresearch.com/industry-analysis/system-integration-market

[10] Raquel Sanchis, Óscar García-Perales, Francisco Fraile, and Raul Poler. 2019. Low-Code as Enabler of Digital Transformation in Manufacturing Industry. *Applied Sciences* 10, 1 (Dec 2019), 12. https://doi.org/10.3390/app10010012

[11] Marcus Woo. 2020. The Rise of No/Low Code Software Development—No Experience Needed? *Engineering* 6 (07 2020). https://doi.org/10.1016/j.eng.2020.07.007

[12] Kev Zettler. 2022. What is cloud computing? an overview of the cloud. https://www.atlassian.com/microservices/cloud-computing#:~:text=Cloud%20computing%20is%20the%20delivery%20of%20computing%20resources%20%E2%80%94%20including%20storage,the%20internet%20(the%20cloud)

[13] Iyad Zikra, Janis Stirna, and Jelena Zdravkovic. 2017. Identifying Quality Factors of Information Systems Integration Design. 45–60. https://doi.org/10.1007/978-3-319-64930-6_4

A APPENDIX A: ISO 25010

Characteristics Sub-Characteristics	Explanation
Functional Suitability	
Functional Completeness	Degree to which the set of functions covers all the specified tasks and user objectives.
Functional Correctness	Degree to which a product or system provides the correct results with the needed degree of precision.
Functional Appropriateness	Degree to which the functions facilitate the accomplishment of specified tasks and objectives.
Performance Efficiency	
Time Behaviour	Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
Resource Utilisation	Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
Capacity	Degree to which the maximum limits of a product or system parameter meet requirements.
Compatibility	
Co-existence	Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
Interoperability	Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
Usability	
Appropriateness Recognisability	Degree to which users can recognize whether a product or system is appropriate for their needs.
Learnability	Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
Operability	Degree to which a product or system has attributes that make it easy to operate and control.
User Error Protection	Degree to which a system protects users against making errors.
User Interface Aesthetics	Degree to which a user interface enables pleasing and satisfying interaction for the user.
Accessibility	Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
Reliability	
Maturity	Degree to which a system, product or component meets needs for reliability under normal operation.
Availability	Degree to which a system, product or component is operational and accessible when required for use.
Fault Tolerance	Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
Recoverability	Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.
Security	
Confidentiality	Degree to which a product or system ensures that data are accessible only to those authorized to have access.
Integrity	Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.
Non-repudiation	Degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later.
Accountability	Degree to which the actions of an entity can be traced uniquely to the entity.
Authenticity	Degree to which the identity of a subject or resource can be proved to be the one claimed.
Maintainability	
Modularity	Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
Reusability	Degree to which an asset can be used in more than one system, or in building other assets.
Analysability	Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
Modifiability	Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
Testability	Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.
Portability	
Adaptability	Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
Installability	Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.
Replaceability	Degree to which a product can replace another specified software product for the same purpose in the same environment.