Method for Legacy Software Evaluation Based on the Balanced Scorecard and AHP

Frank den Heijer
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
denheijer.frank@gmail.com

ABSTRACT

Many businesses have adopted information systems in a certain way. As the IT industry rapidly moves forward, quickly a gap is formed between old technologies and current technologies. Therefore, legacy systems become inevitable to many businesses. Companies that have to deal with legacy systems encounter many setbacks due to the fact that legacy systems impede change and are poorly compatible with the latest technologies. Companies often struggle to make informed decisions regarding the identification legacy. Knowing which application are at risk and why they are at risk remain to be among the top concerns of IT decision makers. This paper provides a method for defining and identifying the degree of legacy in a software product taking into account the company's unique characteristics regarding their definition of legacy. First, several metrics for measuring legacy are identified based on existing quality measurements. Then the Balanced Scorecard will be considered as evaluation framework for structuring the found software metrics. Lastly, by using the Analytical Hierarchy Process (AHP) a method will be proposed for tailoring the software metric to reflect the companies interests and scoring the application portfolio.

Keywords

Legacy Scorecard, Legacy Systems, Software Metrics, Balanced Scorecard, AHP, Application Portfolio

1. INTRODUCTION

Since the entrance of information technology in business there has been an ongoing development. The IT industry has grown tremendously and new technologies have been emerging rapidly. These industry innovations provide new opportunities to companies, on the other hand they cause legacy. The definitions for legacy related to information technology vary greatly [10][14]. According to Rai et al. (2015) a typical legacy system aims to support the core IT processes of organizations, and are filled with maintainability and scalability issues [13]. Other definitions of legacy describe the relation of legacy to a company's organizational structure [18].

Besides the limited abilities to adapt, risks are a main concern when dealing with legacy systems. Bisbal et al. (1999) presents some of the risks that are associated with having legacy systems [5]. Risks can relate to the hardware that is needed to support the system. Most likely, the uncertainties surrounding the maintainability of a system belong to the riskiest aspects of a legacy software.

Although, the definition of legacy may be ambiguous, the problems it causes touches the entire business [8]. Therefore, it is no surprise companies would like to have insight in the degree of legacy their application portfolio contains. This information is important for managers to be able to derive appropriate steps for their roadmaps. To give actionable insights on the degree of legacy within an application portfolio each and every application within the portfolio has to be evaluated.

Currently, there is no clear methodology for measuring and visualizing the degree of legacy in information systems. Some research has been done to identify the legacy status of an information system from the perspective of its causes and effects [11]. This method only covers a limited number of indicators for measuring legacy, which makes it difficult to derive actionable results.

2. RESEARCH PROBLEM

The previous section introduced the desire of companies to have insight in the degree of legacy in their application portfolio. Subsequently, they want to know which aspects of their portfolio cause legacy, in order to take appropriate actions. Therefore, the main research question can be stated as:

- How to provide insight in the degree of legacy of an information system?

Answering this main question should provide companies answers to questions as: Do we have legacy in our application portfolio? Is the degree of legacy at a level that should make us worry? What aspects of the information systems cause the legacy? What prioritization should be given to systems to be improved?

In order to answer the main research-question several sub questions should be answered first. Related questions concern the method for measuring legacy in information systems and the ability to visualize the results insightfully.

- How is legacy being measured?
- What method for scoring, weighing and visualization are relevant for legacy indicators?
- How can these concepts be put together to provide insight in the degree of legacy of an application?

In this paper, the novel contribution to existing knowledge is to provide a methodology for the creation of a legacy scorecard using AHP. Based on the concept of the Balanced Scorecard and the use of AHP on existing software quality metrics a thorough method for legacy evaluation is provided.

3. RESEARCH METHOD

The paper is structured according to the Design Science Research Methodology as described by Peffers et al. (2007) [7].

This section, the first, introduces the problem and defines the objectives of this research. Section 2 will discuss the relevant literature to support the next sections. Covering the legacy indicators, ranking and visualization methods. Section 3 up to section 5 will apply the found measurement tools and concepts to design a coherent methodology. Section 6 will cover the validation of the methodology based on the case of a Dutch software supplier for the staffing industry. Subsequently, the case will be evaluated. Finally, Section 7 sums up this paper, draws conclusion, and points out some possible future research directions.

4. LITERATURE REVIEW

To be able to measure the degree of legacy in information systems the question arises how legacy should be defined. As mentioned in section 1 definitions of legacy vary greatly. This may indicate that software legacy has a wide impact on organizations. Among others, Althani et al. (2017) describes several concerns of information systems that indicate legacy.

1. Switching from a legacy system might have a high financial cost for the organization. In addition, the risk associated with legacy migration is high [14].
2. Legacy applications cannot utilize the latest developments in hardware, neither can they make use of the latest software paradigms [14].
3. Legacy systems maintenance is very expensive [7]. Replacing any legacy system could decrease costs by half [19].
4. Legacy systems may not be compliant with current standards, as they are often built in different timeframes. Often, they serve only a single purpose. Therefore, they rarely meet the requirements necessary to support an organization. Business processes and decisions are often integrated with predefined procedure flows, making intertwined software and business applications very inflexible and unfeasible [19].
5. The availability of skilled staff and appropriate documentation of legacy systems are often scarce [13].
6. Legacy systems are difficult to scale to the growth of the company and cannot easily be integrated with other systems [14].

Although, these concerns reflect the meaning of legacy quite well, finding an appropriate way of measuring all these aspects through a collection of KPIs remains to be difficult. O'Byrne &Wu (2002) established a definition of legacy by choosing 3 dimensions: *software quality*, *system suitability* and *underlying platform suitability* [7]. Another view on the definition of legacy is that of Sneed (1995), considering 2 dimensions: *business value* and *technical quality* [17]. Lastly, Ransom et al (1998) identified 3 dimensions: *business value*, *external environment* and *application*.

Apparently, all of the approaches recognize quality aspects on a technical dimension and a business value dimension. For measuring software quality, worldwide the most recognized standard is the ISO 25010 [1]. Consisting of 8 categories and 31 indicators in total this standard allows for a detailed evaluation of information systems.

As with other standards, ISO 25010 does not detail the thresholds for the evaluation metrics to be used, nor does it describe how to group these metrics in order to assign a quality value to a software product [12].

Identifying different dimensions to define legacy is a useful approach to structure the available legacy metrics, as mentioned earlier. Instead of using one of the existing approaches as defined by O'Byrne &Wu (2002) or Sneed (1995) it seems valuable to consider a non-software related scoring framework: The Balanced Scorecard by Kaplan & Norton (1996) [11][17][6].

The BSC provides a strategic framework to provide critical information on four perspectives: *financial*, *customer*, *internal processes* and *learning & growth*. It allows to have a broad view of the performance of an organization without focusing solely on the financials, the latter is common for senior management [6][5].

Now, the comparison between an organization versus an information system is not far off. Consider the four perspectives of the BSC: *financial*, *customer*, *internal processes* and *learning & growth*. Software has relations to all of these perspectives: Financial is relevant because of the costs of legacy [2]. The customer perspective can be compared to the user of an information system, which relates to functionality aspects. The internal processes perspective relates to the non-functional aspects of an information system and learning & growth can be compared to aspects related to the development of information systems, such as maintainability.

One of the main reason for the BSC being so powerful is the balanced insight it brings across the organization. At the same time, does the BSC allow for flexibility in the choice of KPIs [6].

5. SOFTWARE METRICS

As discussed in previous section, the degree of legacy in information systems is difficult to assess. However, the ISO 25010 standard for software quality does provide a detailed collection of metrics [1]. Financial aspects are not considered by the ISO standard, but are among the most relevant for decision-making in businesses. Costs are especially relevant for the identification of legacy within an application portfolio, because it often is the driver to start a legacy migration project [5]. Therefore, two financial metrics are added to the collection: *maintainability costs* and *developments costs*. Althani et al. (2017) incorporates maintainability costs as the only financial factor in his paper on legacy identification.

It is essential to deeply understand the metrics that are used to score information systems, because the quality of the final scorecard is fully dependent on the quality of the input. Below is a full list of the collection of metrics discussed. See appendix A for a full list of the ISO quality metrics including specific indicators per category.

5.1 Financial Metrics

5.1.1 Maintainability Costs

The costs related to keeping the system operational. Including recurring hardware costs, costs of bug fixing, costs of keeping the systems compliant and other costs related to keeping the system operational.

5.1.2 Customization Costs

Costs involved with building specific features for specific customers. According to Stamelos et al. (2002) are costs incurred for customization related to the quality of the software [17].

5.2 Quality Metrics

5.2.1 Functional Suitability

Indicators related to the extent to which a software application provides functionalities that meet the stated and assumed needs, when used under the intended conditions.

5.2.2 Performance Efficiency

Performance metrics determine the performance of an information system in relation to the amount of resources used under stated conditions.

5.2.3 Compatibility

The extent to which a software application can exchange information with other systems and perform desired functionalities while sharing the same hardware or software environment.

5.2.4 Usability

Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

5.2.5 Reliability

The extent to which a software application performs specific functionalities under specified conditions for a specified amount of time.

5.2.6 Security

The extent to which a software application protects information and data in a way that people or other systems have the right degree of data access appropriate to their type and level of authorization.

5.2.7 Maintainability

The extent to which a software application can be changed effectively and efficiently by the designated developers and administrators.

5.2.8 Portability Metrics

Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

6. MAPPING OF METRICS ONTO THE BALANCED SCORECARD

Because the total number of metrics discussed in previous section can be overwhelming and difficult to grasp, the next step is to create coherence by categorizing the metrics properly. The four perspectives of the BSC: *financial*, *customer*, *internal processes* and *learning & growth* are a great reference for identifying the dimensions of how legacy should be approached. Because of the similarities between organization and information systems discussed in section 2, a mapping of software metrics onto the BSC is possible. See figure 1 for a visual representation.

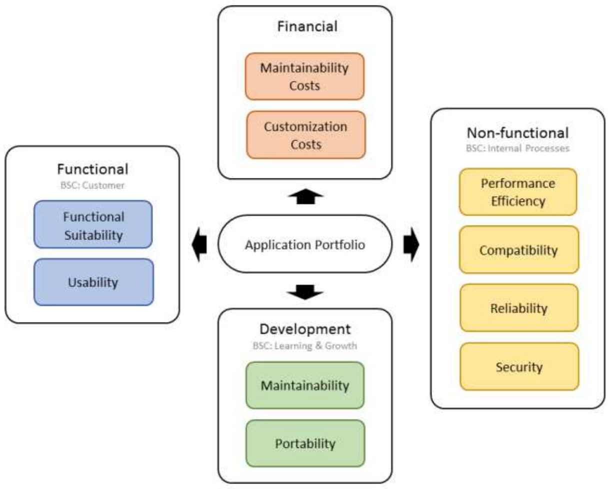


Figure 1. Mapping of legacy metrics on the BSC

The purpose of the BSC to focus on the non-financial indicators makes this framework highly useful for the evaluation of information systems [6]. This is because most of the identified metrics are non-financial.

Characteristics of the BSC, such as the ability to set objectives, are also applicable for the scorecard on software metrics [16]. For example, if for a particular application the *usability* scores exceptionally low, the related metrics can be used to determine the steps needed to improve.

7. WEIGHING AND SCORING METRICS USING AHP

Although, it has been tried to capture each aspect of legacy using the collection of metrics defined in section 3, in the end every company has its own characteristics. Which means the legacy indicators cannot be handled equally for all companies. For example, tax authorities will may find the *security* to be way more important than the *compatibility* of their software. In contrast to another company having a mobile app, which may value *compatibility* way more than *security*. Both of these organizations have information systems, but their perception of legacy is different. For the second company a security leak may cause an unpleasant situation, but they will get away with it. For tax authorities a security leak probably has far-reaching consequences. In short, what is legacy for one company may be no issue for.

To be able to deal with these differences between companies, weights are applied to each metric within the legacy scorecard. For determining weights several Multi Criteria Decision Analysis method are available, such as: ELECTRE III and AHP [15]. The ability of AHP to simplify a complex problem using a structural hierarchy and the principle of comparative judgment of criteria and alternatives make AHP the better choice [9].

The four perspectives of the BSC itself do not have to be weighted, doing so would make the BSC lose its *balanced* nature. Therefore, AHP will be applied only within the four perspectives of *financial*, *functional*, *non-functional* and *development*. See figure 2 for the related problem hierarchies.

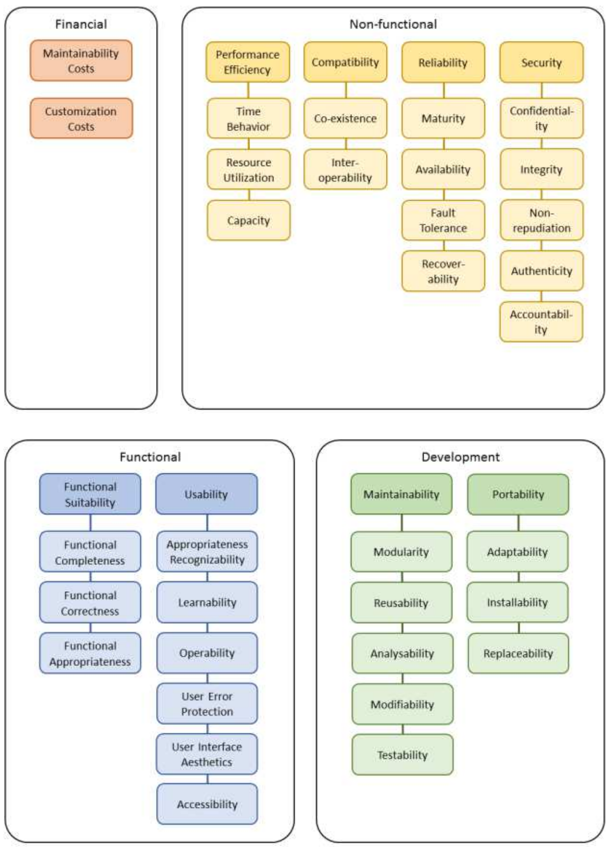


Figure 2. AHP problem hierarchies related to the four perspectives of the legacy scorecard

The process of weighing metrics using AHP consists of pairwise comparisons for each combination of metrics within the same layer. Take for example the *functional* perspective: here *functional suitability* needs to be compared to *usability*. Also, each metric within *functional suitability* needs to be pairwise compared to each other, the same goes for each metric within *usability*. In total there will be three sets of comparisons. From each set of comparisons a reciprocal matrix can be constructed, like figure 3 shows.

Functional Suitability	Functional Completeness	Functional Correctness	Functional Appropriat.	…
Functional Completeness	1	5	1/3	
Functional Correctness	1/5	1	1/7	
Functional Appropriateness	3	7	1	
…				1

Figure 3. Matrix example showing pairwise comparisons

Pairwise comparisons are done on a 9-points scale. A score of 1 stands for *neutral*, a score of 9 stands for *extreme preference* to one side. The scale should be interpreted in such way that when two metrics are compared a score of 9 should be given if the first metric is 9 times more preferable than the second one. If the second metric is 9 times more preferable than the first one, the score should be 1/9 [9].

Determining the final weights of the metrics is done by calculating the *priority vectors* for each reciprocal matrix. The vector is calculated by first summing each column in the matrix. Then divide each element of the matrix with the sum of its column, resulting in the normalized relative weight. The sum of each column should be equal to 1. The *priority vector* can be obtained by averaging across the rows.

The framework has been created and the weights have been determined. The very final step is to score the relevant applications using the specified collection of metrics. It is recommended to use a score ranging from 1 up to 10, since it is easy grasp for most people. Having a scoring range that is too large will lead to an exhausting scoring process, because of the level of detail needed to determine each score.

8. LEGACY CASE

Consider the case of a software supplier for the staffing industry. The delivered software provides full business process outsourcing covering the registration of new candidates that are looking for a job. Also, creating vacancies, receiving job applications, assigning candidates to a job, contracting and payrolling are supported by the software. To support all of the necessary business processes several applications have been developed over time. Currently, there are 4 main applications: 1 centralized administrating system that is around for about 15 years. A second system is positioned as a candidate portal, enabling candidates to enter timesheets and view salary slips. This second system has been operational for 5 to 10 years. Application 3 and 4 are both newly developed systems replacing the older applications: one centralized system used by staffing companies and one online portal for candidates.

This case is particularly interesting because of the 4 applications consisting of 2 pairs of similar application, one of which is suspected of legacy.

Because of privacy issue the name of the application cannot be specified, to summarize:

- Application 1 – Core administrative system (suspected legacy)
- Application 2 – Online portal (suspected legacy)
- Application 3 – New administrative system
- Application 4 – New online portal

8.1 Determining metric weights

The weights of the individual metrics and the overarching categories have been identified by pairwise comparisons according to the AHP method and can be seen in appendix C. By first determining the metric weights before scoring the application portfolio revealed the important aspects of the software considered in this case. For example, usability, security and maintainability are weighted relatively high, indicating that failures in these areas will contribute more to the degree of legacy in this particular portfolio.

8.2 Scoring the applications portfolio

Determining the score of an application on each metric is no easy task. There are 31 metrics in total, therefore it is not recommended to assess them all in one session. It would be even better to split the assessment of the application to different teams, for two reasons. Firstly, having to assess less metrics allows for more focus, hence a higher quality of the assessment. Secondly, by splitting the assessment and scoring of applications to multiple teams will allow more people with the right understanding to provide input. An example of how the assessment can be split is to let the financial perspective of application be assessed by the management team. The functional perspective can be assessed by the support staff or the requirement engineers, because they are the closest to the end-users. Even better would be to incorporate the view of the actual users of the system in the assessment. The non-functional perspective can be assessed by the product-owners. Lastly, the development perspective can be assessed by the entire development team. Since the development perspective is very detailed and technical the thorough understanding of specific topics is often split among the team, therefore no single person could asses this aspect properly.

The results of the assessment of applications related to this case can be found in appendix B. The identified scores are shown in the final scorecard. The lightly coloured cells represent the scores on each metric related to one application. The left columns show the weights calculated by the AHP, the related matrices can be found in appendix C.

8.3 Evaluation

The final results of the legacy scorecard can be found in appendix B, showing a final application score per application for each of the four perspectives: financial, functional, non-functional and development. In figure 4 these final scores are mapped onto a radar chart for easy comparison between application in the portfolio.

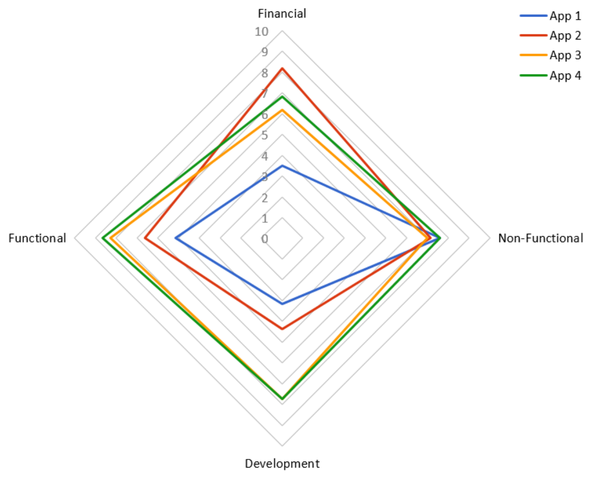


Figure 4. Final legacy scores per application related to the four perspectives of the legacy scorecard

The suspected legacy in application 1 and 2 are clearly visible in the results. Although, application 2 seems to hold up quite well. Its functional perspective contains some legacy, but not as bad as application 1. The financial perspective is even better than the newly developed applications. Another aspect that is reflected by the final results is that all applications score about 7 to 8 on the non-functional perspective of legacy. This is no surprise considering the mission critical tasks executed by the software, such as payrolling.

9. CONCLUSION

Information systems will always be prone to legacy. And the definition of legacy varies per company, which is not an issue. Having a proper definition for legacy is a must if a company wants to get insight in their degree of legacy. Approaching legacy as a quality deficiency is an effective and unified method for identifying legacy, but falls short in tailoring the unique characteristics of company. Combining existing quality metrics with AHP provides a structured method for assessing the degree of legacy within an information system as well as for entire application portfolio. Mapping software quality metrics combined with financial metrics onto the Balanced Scorecard allows for increased understanding of the origin of legacy and provides directions for improvements.

10. FUTURE WORK

Possible future work could relate to finding a way to transform the legacy scorecard into a software quality scorecard that can be assessed on a regular basis to keep track of the qualitative development of software. Also, possible actions and targets to improve the quality related to certain metrics can be further researched. This would bring more aspects of the BSC as it was intended by Kaplan & Norton (1996) to software development [6]. Another area of interest for further research is the combination of legacy scores to existing portfolio management concepts. This could allow application portfolio valuation techniques, such as proposed by Iacob et al. (2012) to incorporate the legacy score as factor to determine migration prioritization or resource allocation [4].

11. REFERENCES

[1] Anon. 2017. ISO - International Organization for Standardization. (August 2017). Retrieved January 24, 2018 from https://www.iso.org/standard/35733.html

[2] Bashair Althani and Souheil Khaddaj. 2017. Systematic Review of Legacy System Migration. 2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)(2017). DOI:http://dx.doi.org/10.1109/dcabes.2017.41

[3] Bisbal, J. et al. 1999. Legacy information systems: issues and directions. *IEEE Software*. 16, 5 (1999), 103-111. DOI: http://dx.doi.org/10.1109/52.795108

[4] Iacob, M.E., Quartel, D., and Jonkers. H., 2012. Capturing Business Strategy and Value in Enterprise Architecture to Support Portfolio Valuation. 2012 IEEE 16th International Enterprise Distributed Object Computing Conference(2012). DOI:http://dx.doi.org/10.1109/edoc.2012.12

[5] Jain, H., "Green ICT Organizational Implementations and Workplace Relationships," in Handbook of Research on Green ICT: Technology, Business and Social Perspectives, Hershey, IGI Global, 2011, pp. 156- 168.

[6] Kaplan, R. S., and Norton. D. P., 1996. The balanced scorecard: translating strategy into action, Boston, MA: Harvard business school Press.

[7] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. 2007. A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems24, 3 (January 2007), 45–77. DOI:http://dx.doi.org/10.2753/mis0742-1222240302

[8] Khadka, R., Saeidi, A., Jansen, S., Hage J. and Haas, G. P., "Migrating a large scale legacy application to SOA: Challenges and lessons learned," in Reverse Engineering (WCRE) 20th Working Conference , 2013, pp. 425-432.

[9] Monti, A., De Toro, P., Droste-Franke, B., Omann, I., and Stagl. S., Assessing the quality of different MCDA methods. Retrieved January 24, 2018 from http://people.unica.it/adm/files/2008/11/05_de_monti_et_al.pdf

[10] Ning, Egberts, J., A., W. Kozaczynski, 1994 "Automated support for Legacy Code Understanding" Communications of the ACM, Vol. 37, No. 5, pp. 50 - 57, Chicago.

[11] Obyrne, P., and Bing Wu. 2001. LACE frameworks and technique-identifying the legacy status of a business information system from the perspectives of its causes and effects. Proceedings International Symposium on Principles of Software Evolution(2001). DOI:http://dx.doi.org/10.1109/ispse.2000.913235

[12] Piattini. M., Software Product Quality Evaluation Using ISO/IEC 25000. Retrieved January 24, 2018 from https://ercim-news.ercim.eu/en99/special/software-product-quality-evaluation-using-iso-iec-25000

[13] Rai, R., Sahoo. G., and Mehfuz, S., "Exploring the factors influencing the cloud computing adoption: a systematic study on cloud migration," SpringerPlus, vol. 4, (1), pp. 1-12, 2015.

[14] Ransom, J., 1. Sommerville, I. Warren, 1998 "A method for assessing Legacy Systems for Evolution" February SEBPC workshop, Durham University.

[15] Saaty, T. L., "How to make a decision: the analytic hierarchy process," European journal of operational research, vol. 48, no. 1, pp. 9–26, 1990.

[16] Sim, K. L., and Koh, H. C., "Balanced Scorecard: A Rising Trend in Strategic Performance Measurement," Measuring Business Excellence, vol. 5, no. 2, pp. 18-26, 2001.

[17] SNEED, H., 1995 "Planning the reengineering of legacy systems" January IEEE Software.

[18] Xin Meng, Jingwei Shi, Xiaowei Liu, Huifeng Liu and Lian Wang, "Legacy application migration to cloud," in Cloud Computing (CLOUD), IEEE International Conference, 2011, pp. 750-751

[19] ZSL Inc, "Strategic approach to modernize your legacy systems and wreck the business bottlenecks," ZSL Inc, Edison, NJ, 2008

APPENDIX

A. ISO 25010 - Software Quality Metrics

A.1 Functionality Indicators

The extent to which a software application provides functionalities that meet the stated and assumed needs, when used under the intended conditions.

11.1.1 Functional appropriateness

The extent to which the functionalities of the application contribute to the achievement of specific tasks and goals. Also, the perception users have about the system being appropriate for their needs.

11.1.1.1 Performance

The extent to which a systems response and processing times, throughput and resource utilization meet the requirements.

A.2 Performance Efficiency

This characteristic represents the performance relative to the amount of resources used under stated conditions.

11.1.2 Time behavior

Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

11.1.3 Resource utilization

Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.

11.1.4 Capacity

Degree to which the maximum limits of a product or system parameter meet requirements.

A.3 Compatibility Indicators

The extent to which a software application can exchange information with other systems and perform desired functionalities while sharing the same hardware or software environment.

11.1.5 Interoperability

The extent to which two or more systems can exchange information and use the exchanged information.

11.1.6 Co-existence

The extent to which a software application can efficiently perform its desired functionalities while sharing a common environment and resources with other products, without adversely affecting any other product.

A.4 Usability

Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

11.1.7 Appropriateness recognizability

Degree to which users can recognize whether a product or system is appropriate for their needs.

11.1.8 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.

11.1.9 Operability

Degree to which a product or system has attributes that make it easy to operate and control.

11.1.10 User error protection

Degree to which a system protects users against making errors.

11.1.11 User interface aesthetics

Degree to which a user interface enables pleasing and satisfying interaction for the user.

11.1.12 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.

A.5 Reliability Indicators

The extent to which a software application performs specific functionalities under specified conditions for a specified amount of time.

11.1.13 Maturity

The extent to which a software application meets reliability needs under normal working conditions.

11.1.14 Availability

The extent to which a software application is operational and accessible when you want to use it.

11.1.15 Fault tolerance

The extent to which a software application works as intended despite the presence of hardware or software errors.

11.1.16 Recoverability

The extent to which a software application can, in case of an interruption or in case of a fault, restore the data directly involved and return the system to the desired state.

A.6 Security Indicator

The extent to which a software application protects information and data in a way that people or other systems have the right degree of data access appropriate to their type and level of authorization.

11.1.17 Confidentiality

The extent to which a software application ensures that data is only accessible to those who are authorized.

11.1.18 Integrity

The extent to which a software application prevents unauthorized access to or adaptation of computer programs or data.

11.1.19 Non-repudiation

The extent to which it can be proven that actions or events have taken place.

11.1.20 Authenticity

The extent to which it can be proven that the identity of a subject or source is as claimed.

11.1.21 Accountability

The extent to which actions of an entity can be traced to that specific entity.

A.7 Maintainability Indicators

The extent to which a software application can be changed effectively and efficiently by the designated developers and administrators.

11.1.22 Modularity

The extent to which a software application is built in separate components so that changes to one component have minimal impact on other components. In other words, the architecture of the software.

11.1.23 Reusability

The extent to which an existing part can be used in more than one system or can be reused when expanding the software.

11.1.24 Analyzability

The extent to which it is possible to effectively and efficiently assess the impact of a planned change on one or more components. To identify deviations or errors caused by a system or to identify components that have to be changed. The presence of documentation.

11.1.25 Modifiability

The extent to which a product or system can be changed effectively and efficiently without errors or quality reduction as a result.

11.1.26 Testability

The extent to which effective and efficient test criteria can be established for a system, product or component and in which tests can be performed to determine whether these criteria have been met.

A.8 Portability Indicators

Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

11.1.27 Adaptability

The extent to which a software application can be effectively and efficiently adapted for new hardware, software and other operational environments.

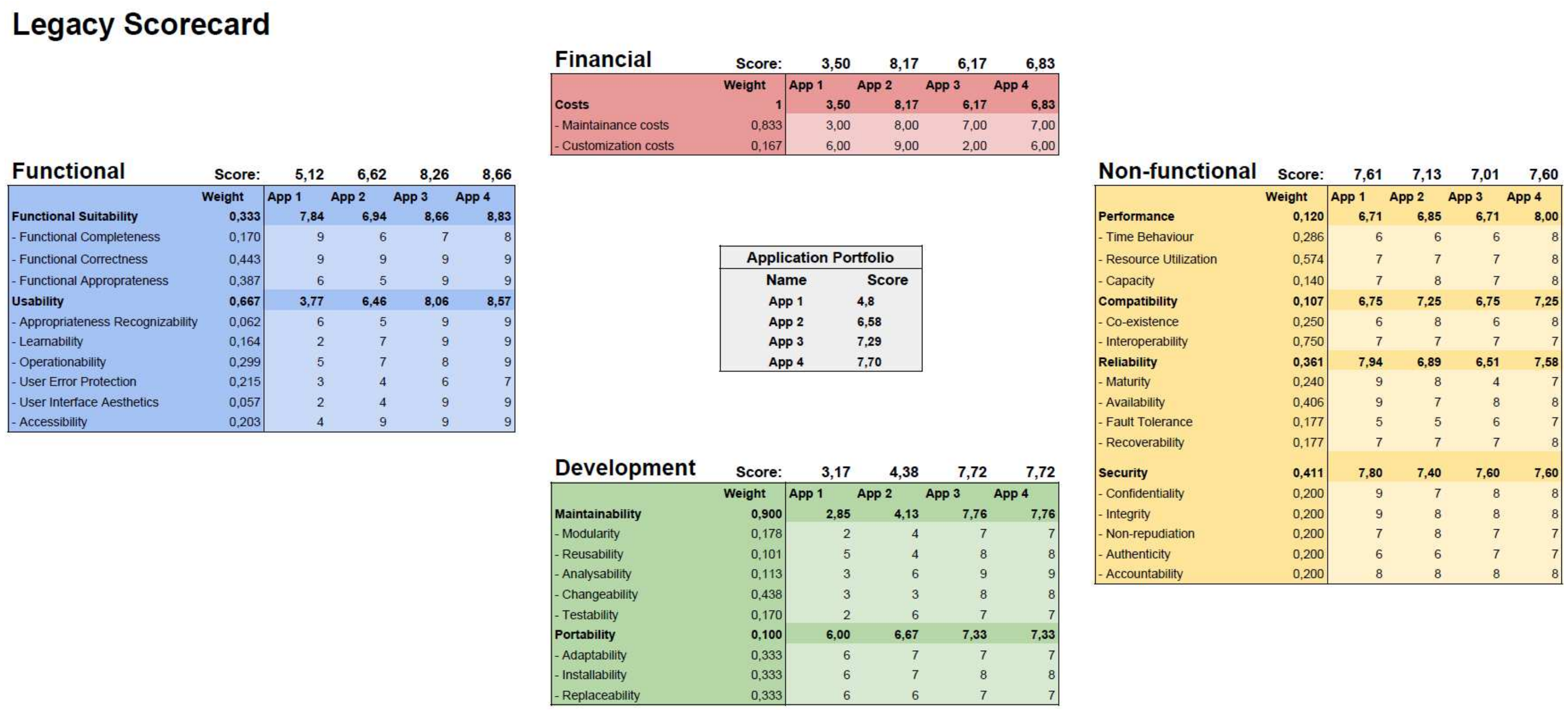
11.1.28 Installability

Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

11.1.29 Replaceability

The extent to which a software application can replace another specific application having the same purpose in the same environment.

B. LEGACY SCORECARD



Legacy Scorecard

Functional — Score: 5,12 | 6,62 | 8,26 | 8,66

	Weight	App 1	App 2	App 3	App 4
Functional Suitability	**0,333**	**7,84**	**6,94**	**8,66**	**8,83**
- Functional Completeness	0,170	9	6	7	8
- Functional Correctness	0,443	9	9	9	9
- Functional Appropprateness	0,387	6	5	9	9
Usability	**0,667**	**3,77**	**6,46**	**8,06**	**8,57**
- Appropriateness Recognizability	0,062	6	5	9	9
- Learnability	0,164	2	7	9	9
- Operationability	0,299	5	7	8	9
- User Error Protection	0,215	3	4	6	7
- User Interface Aesthetics	0,057	2	4	9	9
- Accessibility	0,203	4	9	9	9

Financial — Score: 3,50 | 8,17 | 6,17 | 6,83

	Weight	App 1	App 2	App 3	App 4
Costs	**1**	**3,50**	**8,17**	**6,17**	**6,83**
- Maintainance costs	0,833	3,00	8,00	7,00	7,00
- Customization costs	0,167	6,00	9,00	2,00	6,00

Application Portfolio

Name	Score
App 1	4,8
App 2	6,58
App 3	7,29
App 4	7,70

Development — Score: 3,17 | 4,38 | 7,72 | 7,72

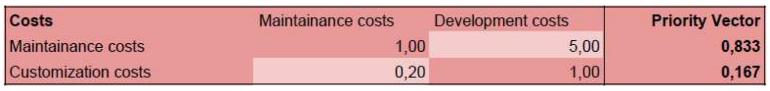
	Weight	App 1	App 2	App 3	App 4
Maintainability	**0,900**	**2,85**	**4,13**	**7,76**	**7,76**
- Modularity	0,178	2	4	7	7
- Reusability	0,101	5	4	8	8
- Analysability	0,113	3	6	9	9
- Changeability	0,438	3	3	8	8
- Testability	0,170	2	6	7	7
Portability	**0,100**	**6,00**	**6,67**	**7,33**	**7,33**
- Adaptability	0,333	6	7	7	7
- Installability	0,333	6	7	8	8
- Replaceability	0,333	6	6	7	7

Non-functional — Score: 7,61 | 7,13 | 7,01 | 7,60

	Weight	App 1	App 2	App 3	App 4
Performance	**0,120**	**6,71**	**6,85**	**6,71**	**8,00**
- Time Behaviour	0,286	6	6	6	8
- Resource Utilization	0,574	7	7	7	8
- Capacity	0,140	7	8	7	8
Compatibility	**0,107**	**6,75**	**7,25**	**6,75**	**7,25**
- Co-existence	0,250	6	8	6	8
- Interoperability	0,750	7	7	7	7
Reliability	**0,361**	**7,94**	**6,89**	**6,51**	**7,58**
- Maturity	0,240	9	8	4	7
- Availability	0,406	9	7	8	8
- Fault Tolerance	0,177	5	5	6	7
- Recoverability	0,177	7	7	7	8
Security	**0,411**	**7,80**	**7,40**	**7,60**	**7,60**
- Confidentiality	0,200	9	7	8	8
- Integrity	0,200	9	8	8	8
- Non-repudiation	0,200	7	8	7	7
- Authenticity	0,200	6	6	7	7
- Accountability	0,200	8	8	8	8

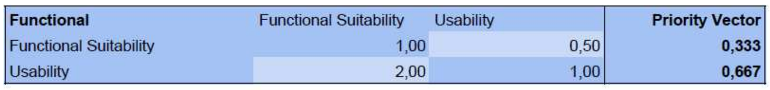
C. RECIPROCAL MATRICES

C.1 Financial

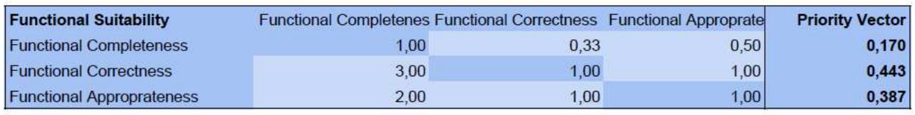


Costs	Maintainance costs	Development costs	Priority Vector
Maintainance costs	1,00	5,00	**0,833**
Customization costs	0,20	1,00	**0,167**

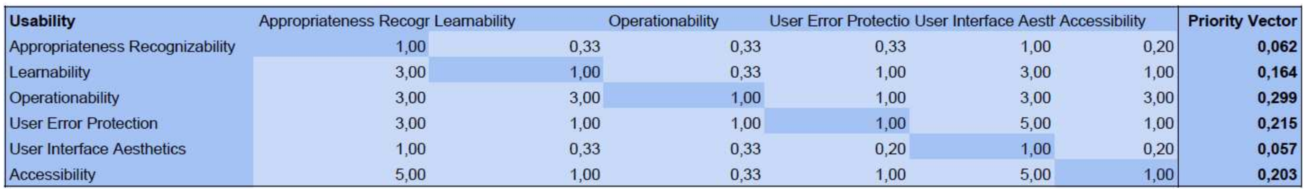
C.2 Functional



Functional	Functional Suitability	Usability	Priority Vector
Functional Suitability	1,00	0,50	**0,333**
Usability	2,00	1,00	**0,667**

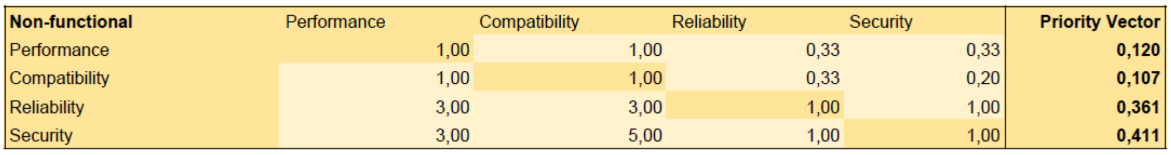


Functional Suitability	Functional Completenes	Functional Correctness	Functional Approprate	Priority Vector
Functional Completeness	1,00	0,33	0,50	**0,170**
Functional Correctness	3,00	1,00	1,00	**0,443**
Functional Approprateness	2,00	1,00	1,00	**0,387**

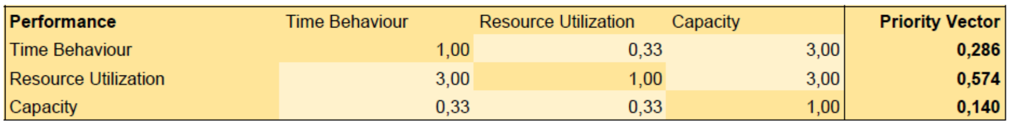


Usability	Appropriateness Recogr	Learnability	Operationability	User Error Protectio	User Interface Aesth	Accessibility	Priority Vector
Appropriateness Recognizability	1,00	0,33	0,33	0,33	1,00	0,20	**0,062**
Learnability	3,00	1,00	0,33	1,00	3,00	1,00	**0,164**
Operationability	3,00	3,00	1,00	1,00	3,00	3,00	**0,299**
User Error Protection	3,00	1,00	1,00	1,00	5,00	1,00	**0,215**
User Interface Aesthetics	1,00	0,33	0,33	0,20	1,00	0,20	**0,057**
Accessibility	5,00	1,00	0,33	1,00	5,00	1,00	**0,203**

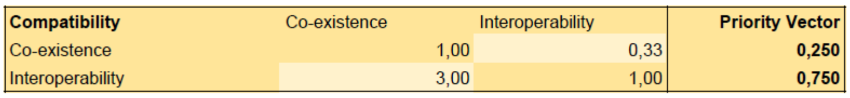
C.3 Non-functional



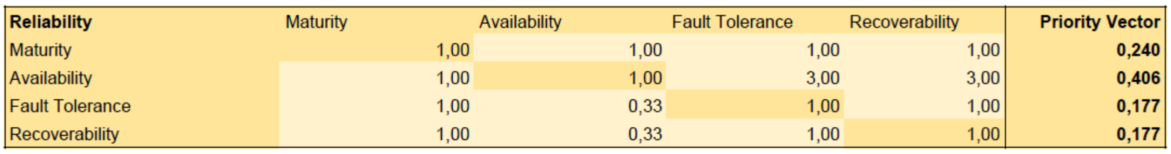
Non-functional	Performance	Compatibility	Reliability	Security	Priority Vector
Performance	1,00	1,00	0,33	0,33	**0,120**
Compatibility	1,00	1,00	0,33	0,20	**0,107**
Reliability	3,00	3,00	1,00	1,00	**0,361**
Security	3,00	5,00	1,00	1,00	**0,411**



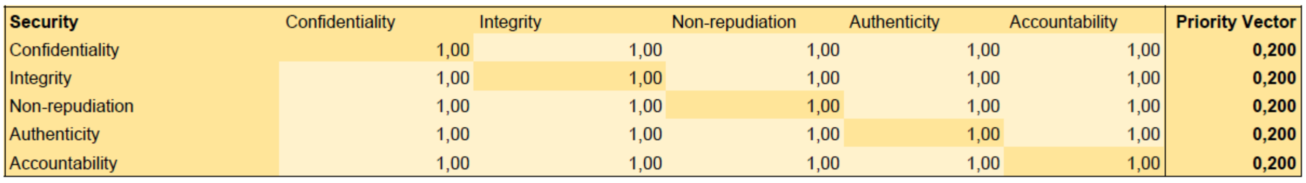
Performance	Time Behaviour	Resource Utilization	Capacity	Priority Vector
Time Behaviour	1,00	0,33	3,00	**0,286**
Resource Utilization	3,00	1,00	3,00	**0,574**
Capacity	0,33	0,33	1,00	**0,140**



Compatibility	Co-existence	Interoperability	Priority Vector
Co-existence	1,00	0,33	**0,250**
Interoperability	3,00	1,00	**0,750**



Reliability	Maturity	Availability	Fault Tolerance	Recoverability	Priority Vector
Maturity	1,00	1,00	1,00	1,00	**0,240**
Availability	1,00	1,00	3,00	3,00	**0,406**
Fault Tolerance	1,00	0,33	1,00	1,00	**0,177**
Recoverability	1,00	0,33	1,00	1,00	**0,177**



Security	Confidentiality	Integrity	Non-repudiation	Authenticity	Accountability	Priority Vector
Confidentiality	1,00	1,00	1,00	1,00	1,00	**0,200**
Integrity	1,00	1,00	1,00	1,00	1,00	**0,200**
Non-repudiation	1,00	1,00	1,00	1,00	1,00	**0,200**
Authenticity	1,00	1,00	1,00	1,00	1,00	**0,200**
Accountability	1,00	1,00	1,00	1,00	1,00	**0,200**

C.4 Development

Development	Maintainability	Portability	**Priority Vector**
Maintainability	1,00	9,00	**0,900**
Portability	0,11	1,00	**0,100**

Maintainability	Modularity	Reusability	Analysability	Changeability	Testability	**Priority Vector**
Modularity	1,00	1,00	3,00	0,33	1,00	**0,178**
Reusability	1,00	1,00	1,00	0,20	0,33	**0,101**
Analysability	0,33	1,00	1,00	0,33	1,00	**0,113**
Changeability	3,00	5,00	3,00	1,00	3,00	**0,438**
Testability	1,00	3,00	1,00	0,33	1,00	**0,170**

Portability	Adaptability	Installability	Replaceability	**Priority Vector**
Adaptability	1,00	1,00	1,00	**0,333**
Installability	1,00	1,00	1,00	**0,333**
Replaceability	1,00	1,00	1,00	**0,333**