Legacy information systems, can they be agile?

A framework for assessing agility.

Maik Verbaan
HU University of Applied Sciences Utrecht
Nijenoord 1, 3552 AS Utrecht, Netherlands, maikel.verbaan@student.hu.nl
Angarde Informatica B.V.
Burgemeester Verderlaan 15e, 3544AD Utrecht, Netherlands

ABSTRACT

Information systems should contribute to enterprise effectiveness, and usually do so during the operational phase of their lifecycle. From the experience of practitioners, the duration of this lifecycle is often not preset, therefore resulting in information systems with a relatively long lifecycle and information systems with a relatively short lifecyle. One of the elements in application management, is managing the application lifecycle. In the experience of practitioners, deciding the moment to end the lifecycle, refactor it, or leave it be are often not thoroughly researched. The decision to move on to a newer information system is therefore not always sufficiently justified and relies more on a gut feeling. What if the older information system is still able to perform and comply with the changes the enterprise desires? Prolonging the length of an application lifecycle could result in cost reduction in an application portfolio. In this paper, we create a method of assessment of the ability to change of a legacy information system and identifying areas in which a legacy information system would need improvement in order to increase this ability to change.

INTRODUCTION

Organizations are frequently confronted with the dynamics of the environment in which they operate (Baker, 1996; Levine, 2005; Wadwha & Rao, 2003). Such dynamics can consist of organizational changes, changes mandatory by law, changes in the customer group, and more. The organizational processes that are influenced because of such changes need to be able to cope. Supporting these processes, information systems are often employed (Hill, Kraemer, & Gurbaxani, 2004; Ives & Learmonth, 1984; Wankadir, 2004). For instance, organizational and environmental changes may imply change in the supporting information systems. As found in available literature, this issue is fairly well known and has persisted over the years (Lehman, 1969; Lehman, 1997).

Organizations are often confronted with information systems that have been operational for years, have gone through many changes, and are based on older programming languages and paradigms (Staudenmayer et al., 1998). Such information systems are highly important for an organizations day-to-day business, but grow more difficult to maintain as time progresses. In this regard, Berghout and Nijland speak of the IT management paradox. Risk and uncertainty are the highest at the beginning of a project. The same goes for flexibility. These risks and uncertainties decrease during the project as the project approaches competion. Unfortunately, the closer a project is to completion, the more difficult it is to adapt it to changing needs (Berghout & Nijland, 2002). This problem prolongs itself to the operational phase of information systems (after project completion). Operational information systems can be considered as complete, but are difficult to adapt to new circumstances.

This problem of necessity for change in information systems has been identified from many different perspectives, such as development methods (Datta, 2006), programming paradigms (Van-Roy & Haridi, 2004), design principles (Martin, 2000), etc… Especially regarding the context of legacy information systems, this problem has been around for a long time and is one that mostly older organizations are confronted with (Bisbal, Lawless, & Grimson, 1999). This is largely due to their age (Parnas, 1994) and absence of knowledge within organizations regarding the information systems original design, the processes it supports (Bennett & Rajlich, 2000), and more. Aging information systems are often referred to as legacy (Chen & Rajlich, 2001). Therefore we adopt the following definition for legacy information systems: "any information system that significantly resists modification and evolution" (Brodie & Stonebraker, 1998).

In order to effectively manage and thus maintain information systems, Looijen constructed a IT management framework. In this framework, three aspects of IT management are distinguished: *functional*, *application* and *technical* management (Looijen, 2004). The aspect of application management handles the aspect of information system management. A commonly applied framework in this regard, ASL2, speaks of the lifecycle of an application (van Der Pols, 2009a). The main phases in this lifecycle are: planning, developing and exploiting. These phases tend to differ in length, although the exploiting phase generally generates the largest amount of costs (Berghout & Nijland, 2002). At the end of the exploiting phase is the end-of-life moment of an application. How does one decide when an information system has reached its end-of-life? In this paper we aim to achieve a better understanding of how to decide when an information system has reached its end-of-life. We propose that when an information system is no longer able to adequately respond to desired change, it should reach its end-of-life.

Researchers have identified the ability to cope with change in different terms. Judging from their work, the different terms are highly context-dependent, and not for all of their definitions has a consensus been reached (Baker, 1996; Candea, 2008; Levine, 2005; Wadwha & Rao, 2003). Researchers and practitioners have applied (among others) the following terms:

Term	Definition
Adaptability	'The extent to which a software system adapts to change in its environment' (Subramanian & Chung, 2000).
Agility	'The ability to apply knowledge effectively, so that an organization has the potential to thrive in a continuous changing and unpredicted business environment' (Dove, 2001)
Flexibility	'The ability of a system to cope with change (certain or uncertain), in an effective an efficient manner, with minimum penalty regarding time, effort, cost, performance and quality' (Wadwha & Rao, 2003).
Maintainability	'A set of attributes that bear on the effort needed to make specified modifications: stability, analyzability, changeability, and testability' (ISO/IEC, 2003).
Manageability	'The level of human effort required to keep a system operating on a satisfactory level' (Candea, 2008).

Table 1: synonyms found in the literature for information systems ability to change.

There is clearly overlap in the respective terms. The context of legacy information systems offers an interesting research area. By reviewing the literature, several problems regarding the operation of legacy information systems can be identified:

1. Maintenance of information systems is largely affected by the complexity of a system. Complexity implies more difficulty when adapting an information system to a new environment (Candea, 2008).

2. Information systems suffer from the concept of aging. Older information systems have gone through many changes, not always well documented, not always following the original design, resulting in bulky, possibly unreliable programming code (Parnas, 1994).

3. Software complexity is greatly influenced by applied programming paradigms (Henry & Humphrey, 1993; Henry, Lattanzi, & Blacksburg, 1994; Van-Roy & Haridi, 2004).

Research questions and methodology outline

In this paper we do not want to find out what measures to take when dealing with systems in need of change, since this has already been extensively researched (Fowler, Beck, Brant, Opdyke, & Roberts, 2007; Wake, 2004). The statement that an information system is reluctant to changes is often a subjective statement. We propose to use a method to validate such a statement, therefore creating an understanding in the extent of which an information systems is really able to cope with change. This in order to perform application management (Looijen, 2004; van Der Pols, 2009) more efficiently.

Therefore we agree with the concept that one needs to understand that, what one wants to manage (Abbott, 2005). We see a potential means of understanding this concept in terms of quantifying the ability to change of an information system. This will also enable us to make better and well-informed decisions regarding the length of the lifecycle of an information system as well as its economic life-expectancy. In order to do so, the following research question is posed:

Q1 How can the ability to change of legacy information systems be quantified?

In order to successfully answer this question, we separate the different aspects of the question into several sub questions. First, in order to quantify the ability to change, we search for a proper denominator. This denominator is not necessarily limited to the context of legacy information systems, but can be used for information systems in general.

Q1.1 What denominator is suitable for identifying the ability to change in information systems?

In order to be able to quantify the ability to change in legacy information systems, we need to identify its characteristics. This will provide us with a basis for the next subquestion 1.3. After researching the characteristics of legacy information systems, we gain a better understanding on what metrics are appropriate for quantifying them.

Q1.2 What aspects define the ability to change in legacy information systems?

In order to enable us to quantify the characteristic aspects of the procedural programming paradigm, we would need a means. A common method in this case is the use of software metrics (El Emam, 2000; Henry, Lattanzi, & Blacksburg, 1994; Li & Henry, 1993; Subramanian & Chung, 2000).

Q1.3 What metrics are available for measuring the aspects that empower an information systems ability to change?

There is much research available on this topic. The body of knowledge regarding software metrics has increased, also supported by the introduction of object oriented programming (Basili, Briand, & Melo, 1996; Candea, 2008; Eden & Mens, 2006; El Emam, 2000; Henry, Lattanzi, & Blacksburg, 1994; Li & Henry, 1993).

Research design
This research will be conducted through the use of literature research and explorative interviews with professionals in applications management. In order to construct a framework for quantification, we need to narrow down our scope by selecting a denominator. Using this denominator, characteristics empowering the ability to change in legacy information systems will be analyzed. After construction of the framework, we intend to fill in the gaps by performing a broad survey in available literature on software metrics, which will be filtered using the selected denominator. These metrics will be matched to the found characteristics of legacy information systems empowering the ability to change, resulting in a framework of characteristics of ability to change, matched to software metrics. In order to validate the framework, we intend to apply it to an information system within an organization specialized in applications management.

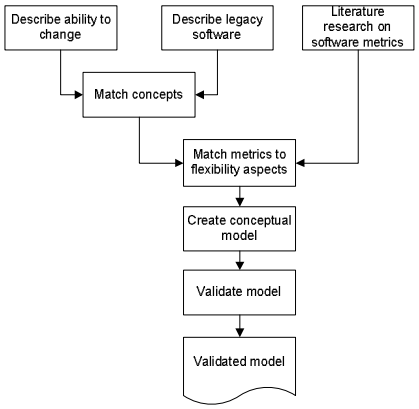

Figure 1: Research design

Organization of the paper
In the following section we provide more detail on the scope of our research and identification of legacy information systems. We also select our common denominator as a basis for our framework. In section 3 we describe the framework's operationalization and explain what metrics we apply in order to assess the ability to adapt. Section 4 describes a case study, in which the framework is applied. We will close this paper by drawing conclusions and present some discussions which were initiated from the case study.

CONCEPTUAL FRAMEWORK

Scoping legacy information systems
Starting with the end in mind, we have found software metrics as a commonly applied method of quantifying aspects of an information system. These metrics are commonly classified on the basis of a programming paradigm. From discussions with experts on the field of application management, we have found the concept of programming paradigms a suitable method of scoping legacy information systems. Researchers have frequently applied a programming paradigm as a means of scoping their research (Henry & Humphrey, 1993; Henry, Lattanzi, & Blacksburg, 1994; Li & Henry, 1993; Van-Roy & Haridi, 2004). Programming paradigms define characteristics of program flow and design patterns (Van-Roy & Haridi, 2004). In this regard, they offer a means of classifying program code in order to isolate characteristics. We pose to do so for legacy information systems. The object oriented paradigm is relatively new, however, novelty does not rule out the possibility of being regarded as legacy (Etzkorn, Hughes Jr, & Davis, 2001). The procedural paradigm is regarded as the predecessor

of the object oriented paradigm (White & Sivitanides, 2005). There is a large amount of procedural information systems being used in enterprises worldwide (Chen & Rajlich, 2001; Henry & Humphrey, 1993; Martin, 2000; White & Sivitanides, 2005). This complements our own experiences within organizations. Information systems build according to the object oriented paradigm form a part of the legacy as well, but seeing as object oriented programming is perceived as a partial answer to the main problem in this paper (Henry, Lattanzi, & Blacksburg, 1994; White & Sivitanides, 2005), we regard the procedural paradigm as a better way to scope our research. Therefore we use this paradigm as a basis for our research. By adopting this fact, we identify legacy software as procedural software.

Referring to the fourth research question, software metrics regarding the procedural paradigm seem to be plentiful (Li & Henry, 1993), possibly offering a very suitable basis for selecting appropriate metrics in order to quantify aspects which enable an information system to change. Available research on software metrics for the procedural paradigm seems to have focused on maintainability of information systems (Henry & Humphrey, 1993; Rombach, 1987; Rombach, 1990; Wake & Henry, 1988). Based on their research and previously stated definitions, we argue the equality of maintainability and ability to change, however we do see some similarities.

Ability to change

In order to select a denominator for the ability to change, we review the previously stated terms which were found in available literature. Naturally we have found other synonyms as well, but these terms were the ones that were most frequently used. They are: a*daptability, agility, flexibility, maintainability* and *manageability.*

Agility seems to be a term that has become rather popular in information systems development methods (Arteta & Giachetti, 2004; Conboy & Fitzgerald, 2004; Datta, 2006; David, McCarthy, & Sommer, 2003). Based on the given definition, it can be approached from a business perspective as well. Agility and flexibility are closely related (Baker, 1996), however Wadhwa and Rao propose a distinction between both concepts. Flexibility is the ability to adapt to situations that are familiar and for which the necessary procedures are already in place, whereas agility is viewed as the ability to adapt to unpredictable changes (Wadwha & Rao, 2003). This complements the previously stated definitions.

Manageability seems to be too narrow for our research. We see a difference between the ability to change and keeping an information system operational to a satisfactory level. When the level is satisfactory for its intended use, it might not be satisfactory for its currently unintended use, but in the future intended use. With maintainability we see the same narrowness as manageability. Finally, adaptability is a term that seems to suit the definition of ability to change well, unfortunately, it is not often mentioned in literature (ISO/IEC, 2003) and therefore, from our perspective lacks the strength to be our denominator.

In this regard, we propose to use the term agility as the denominator for our research, because of its property of responding to unpredictable change. Organizational changes can be both predictable and unpredictable (David, McCarthy, & Sommer, 2003).

Having settled on the term agility, we have narrowed down our scope. In order to find a suitable basis for our method of measurement, we perform a short survey of literature on agility measurement research, resulting in several approaches of agility that have been addressed.

Approach	Research
Information systems development agility	(Conboy & Fitzgerald, 2004)
Organizational agility	(Erande & Verma, 2008; Lin, Chiu, & Tseng, 2006; Tsourveloudis & Valavanis, 2002)
Manufacturing systems agility	(Elkins, 2004; Llorens, Molina, & Verdu, 2005; Tsourveloudis, Valavanis, Gracanin, & Matijasevic, 1999)
Supply chain agility	(Swafford, Ghosh, & Murthy, 2008)
Information systems agility	(Zhang, 2005)

Table 2: Brief overview of approaches for research on agility in available literature.

An observation that can be made from the literature above is the difference in approaches regarding identifying aspects of agility to measure. A conceptual framework that caught our interest is the conceptual model for the agile enterprise (Lin, Chiu, & Tseng, 2006). This model shows factors, both company-internal – and external, related to the agility of an enterprise, as well as its enablers. Its perspective is companywide, thus providing a placeholder for our research, which covers part of the aspects of the subject 'technology' which in our opinion has a rightful place in the model.

In order to partially fill the void on the technology perspective, we aim to develop a generic framework for evaluating procedural information system agility. We plan to operationalize this framework as a benchmark. In creating a benchmark, an ideal situation would be to identify attributes of an information system that is perceived as an information system with ultimate agility. This would enable us to benchmark a random information system against the ultimately agile information system, providing the perfect benchmark. Unfortunately, literature indicates there is no consensus on the concept of the ultimately agile information systems (Dove, 2001). In respect to this statement, we apply a different approach.

Another observed evaluation method is the comparison of two states of a system at different points in time (Arteta & Giachetti, 2004). This implies however, an evaluation after implementing a change. From our point of view, this wouldn't coincide with our definition of agility, referring to the ability to adapt to unknown situations. Comparing two states gives an indication of how well the information system was able to cope from the perspective of its previous state. However, it doesn't tell us anything about its ability to change after the performing the change, the new status quo.

Framework

We want to define an objective and repeatable benchmark, delivering comparable results. A very common approach in this regard is the use of software metrics (Basili, Briand, & Melo, 1996; Chidamber, Darcy, & Kemerer, 1998; Henry, Lattanzi, & Blacksburg, 1994; Leffingwell, 2007). By leveraging specific metrics and mapping them to defining concepts of agility, we can achieve a certain understanding of agility in procedural information systems. Yusuf et al have defined three levels of agility. These are: agility for the individual, enterprise and inter-enterprise (Yusuf, Sarhadi, & Gunasekaran, 1999). When defining agility of an information system, the proper level to review would be on all levels, since increasing agility in an information system could possibly have an effect on all levels. Applying these aspects to the context of an information system, we review their applicability, resulting in a set of agility factors based on the work of Wadwha and Rao.

Aspect	Explanation	Applicability

Speed	Speed is this context would refer to speed in adapting to change. This aspect is covered by flexibility.	Potentially applicable, however, we find that this aspect is better covered by flexibility.
Flexibility	*As previously stated, the ability to cope with change.*	*Although not as broad as agility, we regard flexibility as an important part of agility. Therefore we select this aspect.*
Innovation	The degree to which an information system can be regarded as innovative.	The term legacy rules out the ability to innovate.
Pro-activity	*Pro-active thinking, which has possibly been implemented in an information system, referring to the concept of designing for change (Parnas, 1994).*	*Earlier in this paper, we have established that there is a relationship between design for change and agility of an information system. For instance, messy code is surely a factor that decreases the ability to adapt to changes, whereas good, clean code would imply the opposite (Fowler, Beck, Brant, Opdyke, & Roberts, 2007). Therefore, this aspect is selected.*
Quality	*Regarded as functional quality of the system. The degree to which an information system complies with user demands and the extent to which an information system is error-prone.*	*Low quality in this regard will lead to unsatisfied customers. It is not possible for a highly error-prone information system to be agile. When changing the information system, these errors are processed in the changes as well, providing a very unstable basis to work.*
Profitability	The ability to earn profits from an information system.	Profitability does not indicate agility, for instance, when comparing profit and non-profit organizations. Therefore we do not select this aspect.

Table 3: Aspects of agility.

By applying appropriate metrics to the different factors of agility, we aim to achieve an understanding of the agility of a procedural information system. The results of applying the software metrics will most likely be very diverse in nature (Chidamber, Darcy, & Kemerer, 1998; Henry, Lattanzi, & Blacksburg, 1994; Li & Henry, 1993; Rombach, 1987; Subramanian & Chung, 2000), making it difficult to produce a result which is easily comparable and interpretable. In order to have the benchmark produce such results, we employ the principles of the balanced scorecard (Kaplan & Norton, 1996). A difference in our approach, is that the different factors that are measured, influence the center aspect (agility factor), opposite to the balanced scorecard, where the factors that are measured are influenced by the center aspects (vision and strategy). The understanding of agility in a legacy information system will help us control the lifecycle of such an information system, possibly identifying areas in need of attention in order to make a legacy information system more agile. A next step in our research would be to apply a generic scale to the results of such an assessment, in order to create more intuitive and better comparable results.

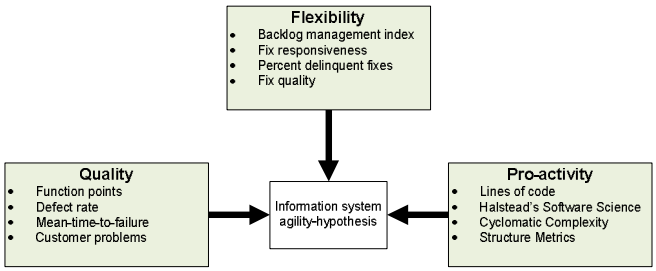

Figure 2: Conceptual framework for Quantifying Agility of Legacy Information Systems (QUALIS)

FRAMEWORK OPERATIONALIZATION

Identifying metrics

Based on the defined agility factors in information systems, we now aim to find appropriate metrics in order to properly gauge these factors. There is a large body of knowledge on software metrics, from which a large part consists of metrics designed for the object-oriented paradigm (Kitchenham, 2010). The focus on object oriented developing is probably caused by the fact that object oriented software projects are currently very common (Gomez, Oktaba, Piattini, & Garcia, 2006). Kitchenham states that researchers often assume a fundamental difference between object oriented and procedural metrics. She regards the difference between statistical en conceptual metrics as more important. Kitchenham bases her research on existing evaluation studies, aimed at the existing body of knowledge of software metrics. In our search for appropriate software metrics however, we set a criterion that a metric must be suitable for use in procedural software. Another criterion which we add to our scope is objectiveness, thus resulting in repeatable analysis and comparable results.

Kan(2002) has made a large contribution to the body of knowledge of software metrics. His work has been extensively cited and provides us with a categorized selection of software metrics, both procedural and object oriented (Kan, 2002) . The concept of software quality is explicitly defined and addressed by Kan. The aspect of customer satisfaction is mentioned as well, but since the proposed methods for measuring customer satisfaction are subjective, we do not integrate this measurement in our framework.

After reviewing Kan's categorization of metrics we select his metrics for software quality. These metrics are selected because of their relation and similarity to our found agility factors. Using these metrics, we perform a preliminary mapping of these metrics, their categories and our agility factors, resulting in the following table:

Agility factor	Kan's classification	Metrics	
Flexibility	*Software maintenance*	Fix backlog and backlog management index	Percent delinquent fixes
		Fix response time and fix responsiveness	Fix quality
Pro-activity	*Product quality*	Lines of code	Cyclomatic Complexity
		Halstead's Software Science	Structure Metrics
Quality	*In-process quality metrics*	Defect density	Mean time to failure
		Function points	Customer problems
		Defect rate	

Table 4: Software metrics applied to agility factors.

The mapping of flexibility and software maintenance can be explained because the metrics for software maintenance are mainly focused on changes in the system, therefore demanding a certain degree of flexibility. Pro-activity is assessed through intrinsic quality of the information system, the degree in which the programming code is coded cleanly and whether the code is simple or complex, as well as the extent to which an information system is designed for change (Parnas, 1994). This assessment provides us with insight to what extent the developers have pro-actively handled the programming code and kept it clean in order to make maintenance easier. Quality is referred to as functional quality of the information system. Well-performing information systems are more likely to be kept operational than occasionally failing information systems.

Application of the QUALIS framework
Based on the agility factors, we can make a statement regarding what aspects of an assessed information system are insufficient in providing a certain level of agility. This would provide us with the insight of what to improve in order to gain agility, or perhaps increasing agility of the information system would prove to be too costly. These insights will help us in managing applications throughout their lifecycles. Another possibility would be to automate a set of these metrics in order to have a continuously updated understanding of the levels of agility factors within information systems, potentially even in its development stage.

An important issue to note are limitations of the framework. Several criteria have to be met in order to successfully apply the framework: 1) in order to apply intrinsic software metrics, the source code must be available, 2) in order to successfully assess the flexibility, errors and corresponding fixes have to be recorded, 3) in order to make a statement regarding the flexibility of an information system, it should at least have an operational time of two years. Seeing as we are handling legacy systems, this should not be an issue.

Validation setup
To theoretically validate the concept of agility assessment in procedural information systems, we performed an empirical validation. This validation was conducted through the use of a survey. The survey was set-up in such a way, that fundamental decisions made in this paper were re-assessed. The denominator was reselected, the decisive factors of the ability to adapt were reselected, as well as the software metrics. The questions were composed in such a way that respondents were unable to discover which answers were in fact our own answers to the questions. Regarding the denominator, the questions following the deciding on a denominator were applied to all selectable denominators, in random order. Respondents completed the survey through the use of a digital form.

FRAMEWORK VALIDATION

Results
For the validation 5 people were interviewed by means of a survey: 1) two applications managers; 2) a technical consultant; 3) an applications consultant and 4) a senior system engineer. All interviewees are familiar with the ASL framework (van Der Pols, 2009b) and in possession of an ASL certificate. The survey followed a structured approach, enabling the interviewees to assess the decisions made in this paper. Each respondent was introduced to the core topics of each decision. The assessment consisted of:

1. Classifying legacy software;
2. Recognition of the problem;
3. Selecting a fitting denominator
4. Assessing core aspects of the denominator
5. Assessing the fit of software metrics and core aspects of the denominator

For the majority of the questions, the respondents were asked to score a statement on a scale of 1 to 5. All answers to the questions received scores. These scores were not interdependent, providing us with an objective score, enabling us to compare the scores. Because of this objectivity, we have set a validation threshold for a score of 2,5. Scoring 2,5 points or higher means the statement was successfully validated. In the next table, the results per topic are reviewed.

Topic #	Result
1	No common agreement on the term legacy information system among respondents. Keyphrases are: older information systems; no longer supported information systems; supporting core processes; mission critical.
2	A common statement in this topic was complexity of the structure in legacy information systems. Often information systems are too complex that IT management is too scared to change the information system. In many other cases the legacy information systems have greatly lost alignment with its supported processes and are therefore subject to change.
3	All available denominators in this paper (adaptability, agility, flexibility, maintainability and manageability) were assessed by the respondents. All denominators received just about the same score, however adaptability scored the highest. The score for agility passed the threshold of 2,5, therefore successfully validated.
4	Regarding adaptability and agility, the assessment resulted in a successful validation of our selected aspects. However, the aspect scores for agility were much higher than the scores for adaptability.
5	All software metrics for the aspect quality were successfully validated. Pro-activity resulted in a mostly successful validation as well, with high scores for quality and flexibility as well, but well above the threshold for pro-activity. The metric 'lines of code' however was classified as a better metric for flexibility. Flexibility was mostly successfully validated as well, with the exception of the metric 'fix quality'.

Table 5: Validation results

Framework implications

Based on the validation results, we applied a change to the framework, regarding the aspects flexibility and pro-activity. The delta consists of moving the metric 'lines of code' from pro-activity to flexibility. Information systems with a smaller amount of program code are not necessarily pro-active. Information systems with large amounts of program code tend to be more complex and therefore more inflexible. The metric 'fix quality' is removed from the framework, based on the validation and the fact that it is only meaningful in conjunction with other metrics.

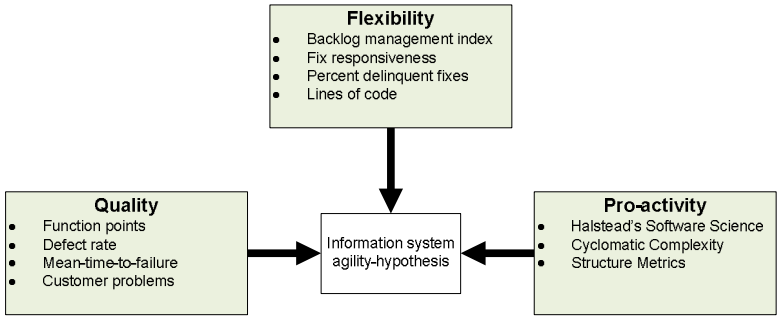

Figure 3: Validated framework for Quantifying Agility of Legacy Information Systems (QUALIS)

CONCLUSION

In our research, we constructed a framework for the assessment of agility within legacy information systems. We hope to achieve greater understanding of this concept in order to enable IT managers to better control the lifecycles of their legacy information systems. For further validation, a practical case study would prove to be very useful. This would provide us with more insight to create a more general scale on which to plot the results of the QUALIS framework, which would potentially make the results better comparable and more intuitive.

It is important to note that the research was focused on the intrinsic agility of procedural information systems. Availability of resources as technical knowledge of the system or technical skilled personnel has not been a part of this research. However, expanding the research scope to not only the intrinsic agility of the information system, but also its surrounding context, would from our perspective prove to be a valuable addition to the framework.

REFERENCES

1. Abbott, J. (2005). Understanding and Managing the Unknown: The Nature of Uncertainty in Planning. *Journal of Planning Education and Research*, *24*(3), 237-251. doi: 10.1177/0739456X04267710.

2. Arteta, B., & Giachetti, R. (2004). A measure of agility as the complexity of the enterprise system. *Robotics and Computer-Integrated Manufacturing*, *20*(6), 495-503.

3. Baker, J. (1996). *Agility and flexibility: what's the difference?. Dimension Contemporary German Arts And Letters*. Cranfield School of Management.

4. Basili, V., Briand, L., & Melo, W. (1996). A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, *22*(10), 751–761. Citeseer.

5. Bennett, K., & Rajlich, V. (2000). Software maintenance and evolution: a roadmap. In *Proceedings of the conference on The future of Software engineering* (p. 73–87). ACM New York, NY, USA.

6. Berghout, E., & Nijland, M. (2002). From Make or Break Issues in IT Management Full life cycle management and the IT management paradox. In *Make or Break Issues in IT Management* (pp. 77-107). Butterworth-Heinemann.

7. Bisbal, J., Lawless, D., & Grimson, J. (1999). Legacy information systems: issues and directions. *IEEE Software*, *16*(5), 103-111.

8. Brodie, M. L., & Stonebraker, M. (1998). *Migrating legacy systems: gateways, interfaces & the incremental approach*. *Morgan Kaufmann* (p. 210).

9. Candea, G. (2008). Toward quantifying system manageability. *usenix.org*.

10. Chen, K., & Rajlich, V. (2001). RIPPLES : Tool for Change in Legacy Software. In *Proceedings of IEEE Computer Society Press* (pp. 230-239). Los Alamitos.

11. Chidamber, S., Darcy, D., & Kemerer, C. (1998). Managerial use of metrics for object-oriented software: an exploratory analysis. *IEEE Transactions on Software Engineering*, *24*(8), 629-639. Citeseer.

12. Conboy, K., & Fitzgerald, B. (2004). *Toward a Conceptual Framework of Agile Methods: A Study of Agility in Different Disciplines*. *Management* (pp. 37-44). Newport Beach, CA, USA.

13. Datta, S. (2006). Agility measurement index: a metric for the crossroads of software development methodologies. In *Proceedings of the 44th annual Southeast regional conference* (p. 273). ACM.

14. David, J., McCarthy, W., & Sommer, B. (2003). Agility—: the key to survival of the fittest in the software market. *Communications of the ACM*, *46*(5), 69. ACM.

15. Dove, R. (2001). *Response ability: the language, structure, and culture of the agile enterprise*. New York: Wiley.

16. Eden, A., & Mens, T. (2006). Measuring software flexibility. *IEEE Proceedings Software*, *153*(3), 113–125. Citeseer.

17. El Emam, K. (2000). A methodology for validating software product metrics. *National Research Council of Canada, Ottawa, Ontario, Canada NCR/ERC-1076*, (June). Citeseer.

18. Elkins, D. (2004). Agile manufacturing systems in the automotive industry. *International Journal of Production Economics*, *91*(3), 201-214.

19. Erande, A. S., & Verma, A. K. (2008). Measuring Agility of Organizations – A Comprehensive Agility Measurement Tool (CAMT). In *Proceedings of the 2008 IAJC-IJME International Conference*.

20. Etzkorn, L. H., Hughes Jr, W. E., & Davis, C. G. (2001). Automated reusability quality analysis of OO legacy software. *Information and Software Technology*, *43*, 295-308.

21. Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (2007). *Refactoring: Improving the design of existing code*. Westford: Addison-Wesley.

22. Gomez, O., Oktaba, H., Piattini, M., & Garcia, F. (2006). A systematic review measurement in software engineering: state-of-the-art in measures. *Setubal, Portugal*, 8. Springer.

23. Henry, S., & Humphrey, M. (1993). Comparison of an Object Oriented Programming Language to a Procedural Programming Language for Effectiveness in Program Maintenance. *Journal of Object Oriented Programming*, *6*(3), 41-49.

24. Henry, S., Lattanzi, M., & Blacksburg, V. (1994). Measurement of Software Maintainability and Reusability in the Object Oriented Paradigm. *Current*, 1-7. Citeseer.

25. Hill, C., Kraemer, K., & Gurbaxani, V. (2004). Review: Information technology and organizational performance: An integrative model of IT Business Value. *MIS Quarterly*, *28*(2), 283-322.

26. ISO/IEC. (2003). *ISO/IEC TR 9126: Software engineering – product quality.*. International Organization for Standardization.

27. Ives, B., & Learmonth, G. (1984). The information system as a competitive weapon. *Communications of the ACM*, *27*(12), 1193-1201. ACM New York, NY, USA.

28. Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering*. *ACM SIGSOFT Software Engineering Notes* (Vol. 21). Addison Wesley.

29. Kaplan, R. S., & Norton, D. P. (1996). *The balanced scorecard: translating strategy into action* (pp. 21-246).

30. Kitchenham, B. (2010). What's up with software metrics? – A preliminary mapping study. *Journal of Systems and Software*, *83*(1), 37-51. doi: 10.1016/j.jss.2009.06.041.

31. Leffingwell, D. (2007). *Scaling software agility: best practices for large enterprises*. Addison-Wesley.

32. Lehman, M. M. (1969). The Programming Process. New York: Citeseer.

33. Lehman, M. M. (1997). Laws of Software Evolution Revisited. *Computing*, 1-11.

34. Levine, L. (2005). Reflections on software agility and agile methods: challenges, dilemmas and the way ahead. *Business Agility and Information Technology Diffusion*, (1999), 353–365. Springer.

35. Li, W., & Henry, S. (1993). Object-oriented metrics which predict maintainability. *Journal of systems and software*, *23*(2), 111–122.

36. Lin, C., Chiu, H., & Tseng, Y. (2006). Agility evaluation using fuzzy logic. *International Journal of Production Economics*, *101*, 353-368.

37. Llorens, F., Molina, L., & Verdu, A. (2005). Flexibility of manufacturing systems, strategic change and performance. *International Journal of Production Economics*, *98*(3), 273-289.

38. Looijen, M. (2004). *Beheer van informatiesystemen* (6de., pp. 155-188, 247-260). Den Haag: Hagen & Stam Uitgevers.

39. Martin, R. (2000). Design principles and design patterns. *Object Mentor*, (c), 1-34. Citeseer.

40. Parnas, D. (1994). Software aging. In *Proceedings of the 16th international conference on Software engineering* (p. 279–287). IEEE Computer Society Press Los Alamitos, CA, USA.

41. Rombach, H. (1987). A controlled experiment on the impact of software structure on maintainability. *IEEE Transactions on Software Engineering*, *13*(3), 89-94.

42. Rombach, H. (1990). Design Measurement: Some lessons learned. *IEEE Software*, *7*(2), 17-25.

43. Staudenmayer, N., Graves, T., Marron, J., Mockus, A., Siy, H., Votta, L., et al. (1998). Adapting to a new environment: how a legacy software organization copes with volatility and change. In *5th International Product Development Conference* (pp. 1-20).

44. Subramanian, N., & Chung, L. (2000). Metrics for software adaptability. *Applied Technology Division, Anritsu Company, Richardson, TX, USA*.

45. Swafford, P., Ghosh, S., & Murthy, N. (2008). Achieving supply chain agility through IT integration and flexibility. *International Journal of Production Economics*, *116*(2), 288-297.

46. Tsourveloudis, N., & Valavanis, K. (2002). On the measurement of enterprise agility. *Journal of Intelligent and Robotic Systems*, *33*(3), 329–342. Springer.

47. Tsourveloudis, N., Valavanis, K., Gracanin, D., & Matijasevic, M. (1999). On the Measurement of Agility in Manufacturing Systems. In *Proceedings of the 2-nd European Symposium on Intelligent Techniques. June. Chania, Greece*. Citeseer.

48. Van-Roy, P., & Haridi, S. (2004). *Concepts, techniques, and models of computer programming*. *Science*.

49. Wadwha, S., & Rao, K. S. (2003). Flexibility and Agility For Enterprise Synchronization: Knowledge and Innovation Management Towards Flexagility. *Studies in Informatics and Control*, *12*(2), 111-128.

50. Wake, S., & Henry, S. (1988). A model based on software quality factors which predicts maintainability. In *Proceedings: Conference on Software Maintenance* (pp. 382-387).

51. Wake, W. C. (2004). *Refactoring Workbook*. Addison-Wesley.

52. Wankadir, W. (2004). Relating evolving business rules to software design. *Journal of Systems Architecture*, *50*(7), 367-382.

53. White, G., & Sivitanides, M. (2005). differences between procedural programming and object oriented programming. *Information Technology and Management*, *6*(4).

54. Yusuf, Y., Sarhadi, M., & Gunasekaran, A. (1999). Agile manufacturing: the drivers, concepts and attributes. *International Journal of Production Economics*, *62*(1-2), 33–43. Elsevier.

55. Zhang, M. (2005). Information systems, strategic flexibility and firm performance: An empirical investigation. *Journal of Engineering and Technology Management*, *22*(3), 163-184.

56. van Der Pols, R. (2009). *ASL2 - een framework voor applicatie management* (1 ed., pp. 215-219). Zaltbommel: Van Haren Publishing.

57. van Der Pols, R. (2009). *ASL: een framework voor applicatiebeheer* (8ste oplag., pp. 1-203). Den Haag: SDU uitgevers BV.