A PRACTICAL APPROACH TO IDENTIFYING AND PRIORITIZING MODERNIZATION OF LEGACY SYSTEMS

Mauricio Ortiz-Ochoa
Universidad Politécnica Salesiana
Cuenca, Azuay, Ecuador
email: mortizo@ups.edu.ec

Leandro Antonelli
Universidad Nacional de La Plata
La Plata, Buenos Aires, Argentina
email: leandro.antonelli@lifia.info.unlp.edu.ar

Roxana Giandini
Universidad Nacional de La Plata
La Plata, Buenos Aires, Argentina
email: roxana.giandini@lifia.info.unlp.edu.ar

ABSTRACT
Currently, there are information systems that were created decades ago but are still being used, thus, today such technology is considered obsolete. In this regard, new requirements, changes or improvements are more expensive because those systems are still playing a significant role in organizations and therefore need maintenance. This paper presents a basic process that enables us objectively identifying software assets in the organization, categorize different systems as legacy and finally prioritize modernization through four levels of action called: remove application, normal maintenance, conditional maintenance and modernization.

KEY WORDS
Software maintenance, Software Reusability, Legacy systems, Modernization, Prioritizing

1 Introduction

Legacy systems will always be an object of study in software engineering [6] [9] [14] [10] [19] [4], since, as it is proposed Zhang *"today's best solutions are tomorrow's legacy systems"* [26].

The modernization of legacy systems has become a complex and costly decision to be taken in every organization and that cannot be postponed indefinitely because *"currently, CIOs are struggling with replacing aging systems"* [20].

With regard to organizations, many of them, due to their lack of knowledge or bad experiences trying to reuse legacy architectures like SOA systems [3], prefer to modernize their applications either by rebuilding the system with new technology development or trying with implementations in the market that allow them to solve their requirements.

With this background and before opting for an upgrade, a planning must be first created [16] in order to provide updated data and detailed characteristics of each of the services or software systems of the organization.

Then a set of options that can be considered for the definition of a legacy system are detailed [25]:

- *Expert judgment*: These are opinions, reports and resolutions of professionals who have been consulted on the subject and who have extensive experience in identifying legacy systems [20]. These experts may be internal entities, or external entities related to the organization through audit engagements.
- *Political decisions*: The executive branch, operational plans, rules or regulations specifying the list of software applications that must modernize and therefore can be implicitly considered as legacies.
- *Demanding users*: Sometimes, modernization decisions can be affected by users who imposed their activities related to administrative processes or systems to have preferential treatment. In theory, this situation should not exist, but the lack of an adequate guidelines for the process of identification of legacy software, causes that the decisions based on demanding user has become a common practice.
- *Technical Criteria*: This possibility should have objective criteria that have to be related to inherent properties to legacy systems.

As shown in the above list, the definition of technical criteria can be considered the best possibility to identify legacy systems and therefore, how the principal goal of this paper, we define a basic process that enables us objectively identifying software assets in the organization, categorize different systems as legacy and finally prioritize modernization through four levels of action called: remove application, normal maintenance, conditional maintenance and modernization. Moreover, we can determinate what information systems can be considered as legacies from the functional or technological point of view [15]. Understanding by functional point of view the extent to which the functional requirements [13] of the business are fulfilled, while the technological point of view refers to the degree of obsolescence of the technology that was used to design and implement the information system [7] [17].

This proposal is presented in Figure 1, where a process consisting of three sequential activities and work product called ”Software systems catalog”.

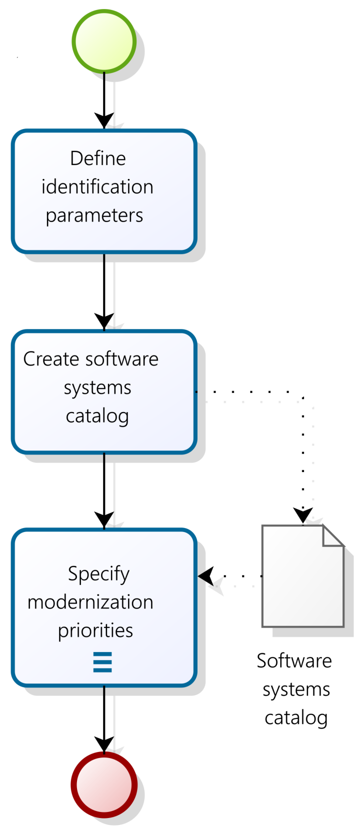
Define identification parameters
Create software systems catalog
Specify modernization priorities
Software systems catalog

Figure 1. Process for identification and specification of priorities for modernizing legacy systems.

With respect to related work, our proposal uses the term portfolio analysis. This approach is used by several authors[7][23][24]; these authors present specific parameters to assess the technical value and business value.

The advantage of our proposal is the identification and categorization of software assets of the organization through a set of generic parameters that can be modified according to the needs of the organization. Once identified and classified the software assets, a portfolio analysis is used.

The rest of the paper is structured as follows: Section 2 explains how defining identification parameters of software assets. Section 3 presents the creation of a Software system catalog. Section 4 specify modernization priorities. Section 5 presents a case study that implements the proposal and Section 6 presents the main conclusions and future work.

2 Defining Identification parameters

Within the first task to detail, a parameter is considered to be a variable value that can store a specific characteristic of an information system. The code assigned to a system, the detailed description, or version number of tables, may be regarded as identification parameters.

The main objective of this task is to define a set of parameters for the objective and uniform identification of the different information systems of an organization. In this sense, all the parameters that are defined must be related to a technology category, as well as to cross categories. Both categories are contributions trying to formally propose the empirical experiences of the author.

In Figure 2, the relationship between the parameters and categories presented is shown by a class diagram [18]; understanding that a parameter must be assigned to a single cross-category, but may be related to various technology categories.

2.1 Technological Categories

A technology category is the description of the main platform for implementing a group of systems or applications [12]. In this way, you can group quickly a set of applications that have been implemented with the same technology.

The modernization project manager must identify the technology platforms available to the organization and conduct a brief classification of the different applications. In case there is a single application with multiple development technologies, it is up to the person in charge to identify which would be the main development technology. An example of technology categories defined by the project manager for the modernization of an organization could be: COBOL aplications, Foxpro aplications, JSP aplications, Forms-Reports aplications, PHP aplications and APEX aplications.

While there may be other types of classifications such as a functional identification by coupling level [14] or responsible, it is preferred to use technological categories as first classification, as they can identify easily and quickly the character of obsolescence of information systems.[11]

2.2 Cross cutting categories

Cross categories refer to the set of parameters or attributes that a software system possess or can be associated with it. In this article, four cross categories are arisen:

- *Functional Identification*: They are independent parameters of the technological category, which together allow having an application overview. Within this category of parameters: code, name, version, date of production, physical location and responsible, etc. can be found.

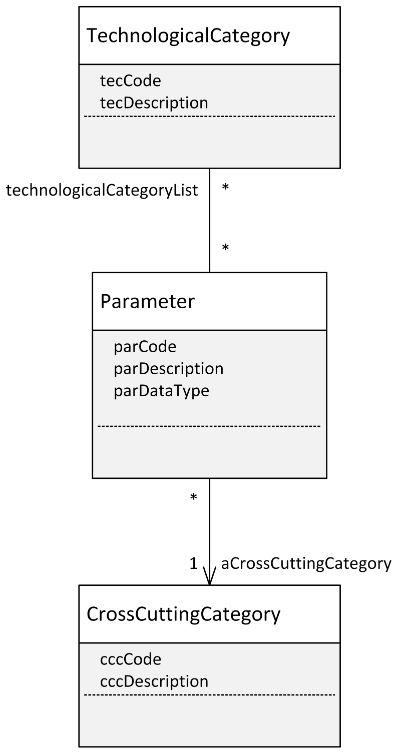

Figure 2. Relationship between categories and parameters.

- *Software Process*: This set of parameters is responsible for the process quality of the software process that the application has [5]. The software process is divided into five stages and a discrete value is associated with each one.

  Requirements: This item shows the grade of documentation for nonfunctional requirements, functional requirements and business rules.

  Design documentation: It can be considered the heart of the transactional systems as it defines storage structures of information in database, or displays the features of the objects in case it is required.

  Development practices: Actually, good development practices depend on the language or programming languages defined for application deployment. In this case, discrete scales can be set mainly with regard to compression or code readability.

  Tests: A suitable test process should consider three types of tests: unit, functional and non-functional. The quantitative capacity of managing this process can be reflected in this section.

  Maintenance: This feature quantitatively identifies the answer the development team has at an incidence (new requirement, improvement or error) generated by the system.

- *Technical characteristics*: At this point, the technological categories that have been defined become necessary, because the technical characteristics are different for each type of application. It means that, the manager of the modernization project must define the attributes or suitable technical parameters for each defined technological feature because it is not the same to identify applications in COBOL than applications in PHP. For example, in a Forms / Reports application, the path of production forms can be considered as a technical parameter, while for a Java application, the name of data source would be considered an adequate technical parameter.

- *Measuring artifacts*: In this category, the parameters associated with the various artifacts that can be measured quantitatively and which in turn depend on the defined technology category are defined. Within this category, parameters such as number of reports, number of tables, number of forms, number of packets or the number of classes may be found.

It is important to remember the need to define a proper and sufficient identification number of parameters, since they all together should allow representing each and every one of the existing applications.

3 Creation of Software systems catalog

Once the categories and identification parameters of different applications are defined, it is time to collect all information concerning and related to the applications under consideration.

This section is not intended to give details on how to collect information from each application, but to show what information we should collect and how to formalize it in a way that allows the creation of Software systems catalog. It should include assessments of each application around the parameters defined in the first activity of the primary process.

At this point of the proposal, the basic structure of the IEEE standards [1] [2] is used as reference for the creation of a document with four sections as shown in the template of Figure 3, and establishes a framework for registration and formalization of different software assets the organization possesses.

3.1 Introduction

The first chapter about the catalog should present the main information concerning the document and at the same time it must contain: purpose, scope, change control, definitions, acronyms, abbreviations, references, and overview of each chapter of the document.

3.2 Definition of identification parameters

This chapter about the catalog of software systems must record all identification parameters defined and organized

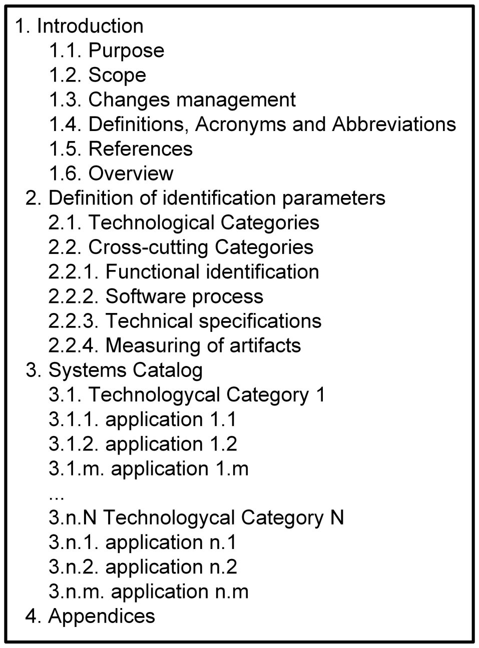
1. Introduction
    1.1. Purpose
    1.2. Scope
    1.3. Changes management
    1.4. Definitions, Acronyms and Abbreviations
    1.5. References
    1.6. Overview
2. Definition of identification parameters
    2.1. Technological Categories
    2.2. Cross-cutting Categories
    2.2.1. Functional identification
    2.2.2. Software process
    2.2.3. Technical specifications
    2.2.4. Measuring of artifacts
3. Systems Catalog
    3.1. Technologycal Category 1
    3.1.1. application 1.1
    3.1.2. application 1.2
    3.1.m. application 1.m
    ...
    3.n.N Technologycal Category N
    3.n.1. application n.1
    3.n.2. application n.2
    3.n.m. application n.m
4. Appendices

Figure 3. Template for the development of software systems catalog.

according to the technological categories and cross categories presented in section II of this article. Thus, this section should provide the reader all parameters that have been selected and if possible a justification of their choice.

3.3 Systems Catalog

This chapter is the most extensive since it is where the values of the different parameters in each and every one of the applications of the organization must be registered.

Depending on the category, a subsection of technology will be created. For example, 3.1 COBOL Applications, 3.2 Forms / Reports Applications, 3.3 APEX Applications, etc. Within each subsection, each application should be recorded with the values corresponding to the identification parameters associated with the definition provided in chapter two of the catalog.

In Table I, the data collected from a software application with JSF Application technology category + EJB +JPA is exemplified. In the first column there are the names of the different parameters grouped into transverse categories (functional identification, software process, technical characteristics and measuring devices) and on the right the corresponding values are shown.

3.4 Appendices

This is an optional section, in which you can record any information that may provide details for the document in full.

4 Specify modernization priorities

This is the final activity of the process and involves the evaluation of each of the applications from the Software systems catalog. This assessment is based on a quadrant analysis consisting of two axes: technical value and business value.

The quadrant used is initially proposed by Andrew Sage [23] and it is presented once again in Dedeke Adenekan's work [7]. In our proposal, this quadrant is extended to support hierarchies of values that allow a quantitative measure in setting the priorities for modernization.

4.1 Technical value

The technical value of a software asset can be quantitatively defined according to a scale of 1 to 5 for each of the following parameters:

- Quality characteristics (QC): A system with high quality features could be modified or functionally adequate to the requirements and business rules.
- Service Reliability (SR): Refers to the continuity of the service provided by the system. An application that is continuously in maintenance will have a lower reliability score.
- Maintenance costs (MC): It involves financial expenditure or effort required to keep the system running. The higher the expenditure, the lower the system value.
- Degradation factor (DF): This factor depends on the scale that is set in the organization. Within this scale we can find the years, licenses costs, additional users and extensions.

  While the degradation factor higher, the system value is lower.

For the calculation of the technical value, we used the equation (1):

$$vt = (QC * SR)/(5 * MC * DF) \tag{1}$$

4.2 Business value

The business value of a software asset can be quantitatively defined according to a scale of 1 to 5 for each of the following parameters:

- Competitive Advantage (CA): The first parameter defines what level of software assets can help in taking advantage of market opportunities.
- Profitability Impact (PI): This parameter indicates the maintenance costs within the overall budget.

- Interdependency with other systems (IS): It obeys the percentage of applications that rely with the system. This relationship can be functional or for data processing.

- Security (SC): It depends on the robustness of the system with respect to different security threats.

For the calculation of the business value, we used the equation (2)

$$vb = (CA * RI * IS * SC)/4 \tag{2}$$

4.3 Quadrant analysis

For each software asset, you must calculate the value in each axis and plot it in the quadrant of Figure 4 [7]. Then, if a software asset has a high business value and a low technical value, then you can consider the modernization of the software asset.

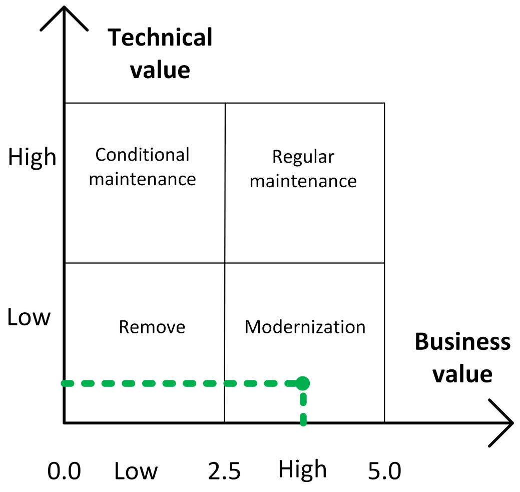

Figure 4. Quadrant analysis of modernization.

Both the technical value and the business value can generate a quantitative value that allows suggesting decisions regarding:

- Application Removal: These are applications that have been relegated and it makes no sense to modernize them. Instead, you should replace or otherwise dismiss them.

- Normal maintenance: It means that the application can continue to operate, but with a preventive maintenance if necessary.

- Conditional maintenance: These kinds of applications do not play a crucial activity in the business, but its high technical value causes them to support new features. Maintenance should be conditioned to the adjustment of business rules.

- Modernization: Applications in this range will be the target or priority of a modernization plan as they are necessary for business applications, but their technical value makes them to have an obsolescence character.

5 Case study

In this section, the information Systems of Salesian Polytechnic University are presented as a case study [8] to validate the process called identification and specification of priorities for modernizing legacy systems.

5.1 Research Methodology

In search of a methodology [22] that allows evidence of the scientific process in the design of the case study, active research is intended to influence or change any actual aspect of the object of the case study [21]. From this point of view, the purpose of the case study is the information systems from Salesian Polytechnic University and the real change is focused on how to identify legacy systems.

5.2 Case Study Objectives

- Identify the legacy systems of the organization in a uniform and objective manner.

- Identify legacy systems that must be part of a modernization plan.

5.3 Proposed Solutions

According to the various defined challenges, the proposed solutions go hand in hand with the implementation of the identification and specification of priorities for modernizing legacy systems; it will serve to understand and analyze the different information systems, since it is necessary to have a catalog of software systems to identify priorities for modernizing legacy systems. With this stage, we aim to fight against the lack of documentation and the complexity of the structure of the data, since parameters for a comprehensive view of the applications will be able to be defined.

5.4 Analysis and presentation of results

It is this section, we analyze the case study and a set of benefits achieved as a result of the implementation process and how this implementation has contributed directly to the management of the Coordination of software development and other areas of technology from Salesian Polytechnic University.

- Definition of information systems as legacies: The definition of a legacy system into the Coordination of development has been carried out thanks to expert judgment but this situation has generated biases due to explicit human intervention decisions. The implementation of the method in the first stage avoids these biases in the consideration of a legacy system because all applications are analyzed with the use of a set of similar parameters previously specified.

- Catalog software systems: When creating a catalog that organizes applications according to technological categories proposed in the case study, it was possible determine verifiable information which is subject to exchange controls of all existing applications within the Coordination of development.

- Specify priorities of modernization: this activity in the proposed process becomes the key objective to address modernization of legacy systems and can be the basis for the creation of the Plan for Modernization of information systems from Salesian Polytechnic University.

6 Conclusions

This article describes the process of identification and specification of priorities for modernizing legacy systems which defines three main tasks: the first is the definition of identification parameters, the second is the documentation of software systems and the third specifies modernization priorities for each previously listed systems.

With regard to the definition of parameters, it is possible to identify a common set of characteristics that can be used in gathering all information relating to the systems has to be analyzed. This collection is the second task of the overall process and aims to create a catalog of software systems generated from a technical and objective point of view. This catalog is linked to common parameters that avoid the use of other sets of possibilities for identification as expert judgment, policy decisions or demanding users.

Finally, the identification stage of legacy systems points to the use of technical value and business value as a quantitative measure for the prioritization of modernization, and suggest decisions regarding: application removal, normal maintenance performance, conditionally maintenance or modernization of the application.

Future work that can be generated from this article is the proposal of a process or structure that allows the creation of a modernization plan. The plan must be based on legacy systems identified through the implementation of the process of this article.

References

[1] Iso/iec/ieee 29148:2011 systems and software engineering -life cycle processes- requirements engineering, 2011.

[2] Iso/iec/ieee 42010:2011 systems and software engineering - architecture description, 2011.

[3] Patricia Bazan, Gabriela Perez, Roxana Giandini, Elisabet Estevez, and Javier Diaz. Services conceptualization within soa/bpm methodology. In *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, pages 1–10. IEEE, 2012.

[4] Fevzi Belli. Assuring dependability of software reuse: An industrial standard. In *Software Technologies*, pages 72–83. Springer, 2014.

[5] Pierre Bourque, Richard E Fairley, et al. *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0.* IEEE Computer Society Press, 2014.

[6] Michael L Brodie and Michael Stonebraker. *Migrating legacy systems: gateways, interfaces & the incremental approach.* Morgan Kaufmann Publishers Inc., 1995.

[7] Adenekan Dedeke. Improving legacy-system sustainability: A systematic approach. *IT Professional*, 14(1):38–43, 2012.

[8] Jessica Díaz, Jennifer Pérez, and Juan Garbajosa. A model for tracing variability from features to product-line architectures: a case study in smart grids. *Requirements Engineering*, pages 1–21, 2014.

[9] Gleison S do Nascimento, Cirano Iochpe, Lucineia Thom, and Manfred Reichert. A method for rewriting legacy systems using business process management technology. 2009.

[10] Hugo Gómez. Elaboración de una guía para la migración de sistemas legados oracle form6i hacía una arquitectura multi-capa. 2012.

[11] Arthur Langer. *Guide to Software Development: Designing and Managing the Life Cycle*. Springer Science & Business Media, 2012.

[12] Arthur Langer. Legacy systems and integration. In *Guide to Software Development*, pages 179–212. Springer London, 2012.

[13] Pericles Loucopoulos and Vassilios Karakostas. *System requirements engineering*. McGraw-Hill, Inc., 1995.

[14] Antonio Massari and Massimo Mecella. Il trattamento dei legacy system. *Capitolo 2 del libro Sistemi informativi*, 5, 2002.

[15] Carlos Matos and Reiko Heckel. Migrating legacy systems to service-oriented architectures. *Electronic Communications of the EASST*, 16, 2009.

[16] Mariano Méndez and Fernando Gustavo Tinetti. First steps towards a tool for legacy systems. In *XVII Congreso Argentino de Ciencias de la Computación*, 2011.

[17] T. Mens, Yann-Gaël Guehéneuc, Juan Fernández-Ramil, and Maja D'Hondt. Guest editors' introduction: Software evolution. *Software, IEEE*, 27(4):22–25, July 2010.

[18] Mauricio Ortiz and Andrea Plaza. *Programación orientada a objetos con Java y UML.* Ediciones Universitarias Universidad Politécnica Salesiana, 2013.

[19] Ricardo Pérez-Castillo, Ignacio García-Rodriguez de Guzmán, Mario Piattini, and Christof Ebert. Reengineering technologies. *IEEE software*, (6):13–17, 2011.

[20] Stacie Petter and Kerry Ward. Countdown to y2gray. *IT Professional*, 15(6):49–55, 2013.

[21] Colin Robson. *Real world research: A resource for social scientists and practitioner-researchers*, volume 2. Blackwell Oxford, 2002.

[22] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.

[23] Andrew Sage. *Systems management for information technology and software engineering*. Wiley-Interscience, 1995.

[24] Harry M Sneed. Planning the reengineering of legacy systems. *IEEE software*, (1):24–34, 1995.

[25] Steven R Walk. Projecting technology change to improve legacy system support strategies. *Naval Engineers Journal*, 122(3):113–120, 2010.

[26] Zhuo Zhang, Dong-Dai Zhou, Hong-Ji Yang, and Shao-Chun Zhong. A service composition approach based on sequence mining for migrating e-learning legacy system to soa. *International Journal of Automation and Computing*, 7(4):584–595, 2010.

PARAMETER	VALUE
Code	1.1
Name	Socioeconomic Profile
Version	Not Available
System	National Academic System
Short description	Socioeconomic Profile
Detailed description	It allows the entry of student's socioeconomic profile
Production Path	services.ups.edu.ec/SDJF
Production Date	Monday, January 3RD, 2011
Souerces Path	172.16.1.159/ fuentesaplicaciones/ sercices.ups.edu.ec/ FICHASOCIOECONOMICA
Responsible	Juan Pérez
Requirements (REQ)	1: There are documents related to functional requirements as cases of use or textual descriptions.
Design documentation (DES)	1: There are documents related to databases structure level such as an ER Diagram.
Development practice (DEV)	1: The code follows regular standars at a syntatic level but it does not record te code, it records the most important business rules.
Testing procedure (TEST)	2: There is a record through emails or user's acceptance letters saying that the system meets the required functional characteristics.
Maintenance process (MAI)	2: Official communication is carried out. It goes through an implicit process of responsability and priorities assignment.
Application server path	http://services.ups.edu.ec/ FICHASOCIOECONOMICA
Application server	Glassfish Community Edition 3.0.1 Build 22
Database	Oracle Standard Edition 11.2.0.4
DataSource	JDBC/SDJFPRO
Programming language	JAVA, JSF, XHTML, Javascript
JPA Implementation	Eclipselink
Packages	5
Pages	5
Entities	0
Beans	1
Controllers	2
Classes	30
Reports	6
Tables	14

Table 1. Identification of a software application