# Evaluating Non-Functional Aspects of Business Process Management Systems

Andrea Delgado, Daniel Calegari
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, 11300
{adelgado, dcalegar}@fing.edu.uy

***Abstract*—Selecting a Business Process Management Systems (BPMS) for an organization is not a trivial task. It requires a thorough evaluation of its capabilities considering the whole support of the business process lifecycle and the organizational environment in which the BPMS will be used. In this context, in a previous work we have proposed a methodology for the systematic evaluation of BPMS, which was mostly focused on required functional and non-technical aspects. In this paper, we present the extension of our methodology with a detailed definition of non-functional aspects to be evaluated, and a set of test cases for their evaluation. We also performed a fine tuning of the methodology based on a comprehensive comparison with other existent methodologies and the provision of tool support. As a case study, we present an evaluation of open source and proprietary BPMS following our proposal.**

**Keywords: Business Process Management Systems (BPMS), evaluation methodology, systematic approach, non-functional requirements.**

## I. Introduction

Business Process Management (BPM) [1], [2] offers a framework to support the definition, control and continuous improvement of business operation. The business process lifecycle [1] can be described as an iterative process involving the modeling of business processes, the software development for their support, their execution and the evaluation of their execution. Business Process Management System (BPMS, [1], [3]) arise as a technology for supporting such lifecycle.

There is a wide variety of BPMS, both open source and proprietary, with different support levels for the defined solution. The selection of a BPMS for an organization is not a trivial task since to be able to compare features within different BPMS, it is necessary to provide an objective evaluation regarding the fulfillment of key technical features that should be provided, as defined in academia [4], [5] and industry [6], [7] studies. Moreover, since the business process vision is the identification of the set of activities that are performed in coordination within an organizational and technical environment to achieve defined business goals [1], the selection of the most adequate BPMS for an organization depends not only on the technological support it provides, but also on the characteristics of the organization itself. Finally, the evaluation should also be guided by a systematic procedure to ensure the quality of the results and its repeatability.

978-1-5386-3057-0/17/$31.00 

In a previous work [8] we have defined a methodology for the systematic evaluation of BPMS considering the specific needs of each organization. Our approach includes the definition of key activities to guide the evaluation and a list of key features that are relevant to this kind of systems. Besides the methodology provides a wide and detailed framework, we have identified some improvement opportunities, e.g., the consideration of non-functional aspects as performance and security (the former methodology was mostly focused on functional and non-technical aspects), and the development of supporting tools, among other aspects.

In this paper, we present an extension of such methodology. We provide a detailed description of non-functional aspects of interest to be evaluated within our methodology, and a set of test cases which provide a benchmark for the standardization and systematization of the evaluation process. We also performed a fine tuning of the methodology based on a comprehensive comparison with other existent methodologies and the provision of tool support. To illustrate the approach, we present results from the evaluation of open source and proprietary BPMS which constitute both a validation and assessment of our proposal and a contribution to knowledge regarding the capacities of selected BPMS technologies.

The rest of this paper is organized as follows. In Section II we provide the results of a comprehensive comparison with related work. Then, in Section III we present an update of the methodology for evaluating BPMS, and in Section IV we provide details on its extension with respect to the evaluation of non-functional requirements. In Section V we present case studies regarding the evaluation of open source and proprietary BPMS, and the tool we have built for supporting the evaluation process. Finally, in Section VI we present some conclusions and future work.

## II. Related Work

In [8] we already provided a brief comparison of our original methodology with respect to related work. These works were considered for the definition of our methodology and the list of characteristics we provide for the evaluation.

Many of these works are not specific about BPMS (e.g., ISO/IEC 9126, superseded by SQuaRE [9]), however the characteristics they consider can be applied to software of any kind and are very important for evaluating the quality

of software from different points of view. We also considered industry reports, e.g., Gartner [10], [11], TEC [7] and Forrester [12], which also consider commercial characteristics, such as: price, customer experience, market understanding and strategy, business model, among others. Unlike these works, our approach does not include any view from the vendors themselves, but from a specific evaluation carried out by the organization with respect to their own prioritization of characteristics.

We performed a complimentary comparison of methodologies with the purpose of detecting improvement opportunities.

In [13] the authors focus on the selection criteria for tools supporting business processes, but not on an evaluation methodology. Despite the fact that the tools are used in the context of electronic commerce, most of the characteristics they propose are already included as part of our methodology. The authors also refer to non-functional requirements.

In [14] the authors propose a generic methodology for Commercial off-the-shelf (COTS) tools combining the DESMET methodology and the Analytic Hierarchy Process (AHP). Their purpose is to define a method in which the evaluation can be performed less manually and with more reliability. It has little connection with our methodology since they focus on the planning of the evaluation and not on their concrete aspects.

In [15] the authors also propose a methodology for the evaluation of COTS. This work is based on considering requirements defined by every stakeholder of the tool to be evaluated. The methodology also proposes the direct participation of such stakeholders in the evaluation process, which is driven by a uniform set of scenarios, configurations and data. We also consider stakeholders requirements and potentially their participation in the evaluation process, but we do not force the evaluation of the characteristic by different people.

In [16] the authors define a framework for the evaluation of open source systems (OSS) which can be seen as a specific methodology, but not in the context of BPMS. The framework uses the OpenBQR method which consider both stakeholder requirements and a list of characteristics defined in the software quality ISO 9126 standard; we include many of them as part of this work. Their proposal tends to resolve common problems found in other methodologies for the evaluation of OSS, e.g., they do not consider support or costs related to proprietary modules that need to be integrated with the OSS. OpenBQR also proposes a filtering method and the use of qualitative data based on a previous prioritization of characteristics, together with some evaluation metrics associated to each evaluation criterion.

Finally, in [17] the authors discuss the main problems related to existent evaluation methods for COTS, basically the lack of a systematic, repetitive and well-defined method. They also refer to the importance of non-functional requirements for the evaluation, the consideration of user requirements, the filtering of characteristics in order to reduce time and cost of an evaluation, the need of a measurement method, and the possibility of using historical data.

In Table I there is a summary of interesting aspects found in the methodologies, and for each methodology, if the corresponding aspect is considered or not.

As can be seen, there were many desirable aspects not supported in our methodology (considered in the first column of Table I). In the case of the filtering step, we made some minor changes in our base methodology, which will be explained in Section III. We also extend the methodology with the inclusion of non-functional aspects. This extension, explained in Section IV, is based on well-known classifications [18], [9]. With respect to the use of historical data, we build a tool which is capable of using existent information in order to perform a comparative evaluation, as will be explained in Section V, and also allows the definition of roles participating in the evaluation process, as well as it simplifies the filtering of characteristics. Finally, the definition of evaluation metrics is subject of future work.

## III. A Systematic Approach for BPMS Evaluation

In what follows we briefly present the methodology we have defined for the systematic evaluation of BPMS in [8] and the changes we have introduced to cope with some of the non-supported aspects described in the last section. We recommend to refer to [8] for a deeper explanation of the methodology.

The methodology is based on a comprehensive list of characteristics regarding the fulfillment of key technical and non-technical features on a BPMS. Moreover, we have defined a systematic process to ensure the quality of the results and its repeatability. The process considers the concrete needs of an organization so the results are the most adequate for the organization, and can be performed reusing previous evaluations reducing evaluation costs.

### A. List of Characteristics

The list is organized into two modules: (1) Technical, which involves everything related to software itself, and (2) Non-technical, which encompasses other characteristics such as community support. Modules are composed of categories grouping cohesive characteristics (a hundred of them). Table II shows the defined structure including both modules and its categories. For more information on categories and their concrete characteristics, please refer to [8].

The technical module depicted in Table II shows only functional aspects. In this work we have divided the technical module into two, expressing functional and non-functional aspects separately. The non-functional aspects are presented in Section IV.

### B. Evaluation Methodology

In Figure 1 we express the evaluation methodology using Business Process Model and Notation v2.0 (BPMN, [19]). The model shows the different activities to be carried out within each organization, including the sub-process of actually evaluating the tools.

First, the list of characteristics is updated if needed and the tools to be evaluated are selected. Then, the organization determines the most important characteristics to be evaluated and rank them for a posterior quantitative evaluation.

TABLE I
COMPARISON OF DIFFERENT METHODOLOGIES

| Aspect | Delgado et al., 2015 [8] | Tsalgatidou et al., 1998 [13] | Morera, 2002 [14] | Lawlis et al., 2001 [15] | Taibi et al., 2007 [16] | Tarawneh et al., 2011 [17] | Description |
|---|---|---|---|---|---|---|---|
| Specific methodology | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | It is defined for the evaluation of specific software (e.g., BMPS) and not generic software (e.g., COTS). |
| Evaluation process | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | It defines a concrete process detailing the steps to follow for performing the evaluation. |
| Roles | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | It defines roles and the knowledge they have to have for the different steps within the evaluation process. |
| List of characteristics | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | It defines a list of characteristics to be considered for the evaluation. |
| Non-functional aspects | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | It considers non-functional aspects for the evaluation. |
| Filtering | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | It describes a filtering step and the corresponding filtering conditions for characteristics within the process. |
| Case study | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | It considers the development of a case study and its analysis to adjust the results of the evaluation process. |
| Qualitative evaluation | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | It defines how the characteristics can be evaluates. |
| Quantitative evaluation | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | It defines a quantitative method for evaluation. |
| Evaluation metrics | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | It defines evaluation metrics for each evaluation criterion. |
| User requirements | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | It considers user requirements for defining evaluation criteria. |
| Historical data | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | It considers historical data of past evaluations in each new evaluation. |

TABLE II
FUNCTIONAL ASPECTS

| Module | Category |
|---|---|
| Technical | Technology, Architecture and Interoperability |
| | Process Design and Modeling |
| | Form Management |
| | Workflow Engine |
| | Management, Monitoring and Audit |
| | Document Management System |
| | Portal |
| Non-Technical | Installation and Support |
| | Maturity |
| | Commercial |

Each organization classify the characteristics using a scale which depends on their needs for each evaluation and therefore allows to instantiate the evaluation to the organizational context. The scale determines different levels of importance: (1) Mandatory; (2) Medium priority and (3) Low priority. After that, test cases and a case study are defined (or adapted if needed, as we provide many of them). These elements are used within the evaluation sub process which involves valuating each characteristic within each tool in another scale we provide for results. This scale of support determines if the characteristic is: (1) Totally supported, the tool has the characteristic; (2) Partially supported, the tool does not cover the entire specification of the characteristic; (3) Not supported, the tool does not provide it. Additionally, three levels of compliance are defined for the support scale: (1) Native, the feature is part of the tool; (2) Particularization, specific software can be developed to achieve such compliance; (3) Integration, it is necessary to include a third component to support it.

Two ways for evaluating the characteristics are defined: theoretical and practical. The theoretical evaluation does not require executing the tool, but is mainly based on the tool documentation, e.g. when non-full versions are available or when characteristics are not a priority for the organization. The practical evaluation does requires executing the tool, with a specific test case to evaluate the level of support it provides. Test cases are defined to cover a selected set of characteristics within each one, that when executed allow us to assess the support the BPMS provides for them.

Finally, a total score (quantitative evaluation) for each tool is calculated regarding the importance defined, and the results

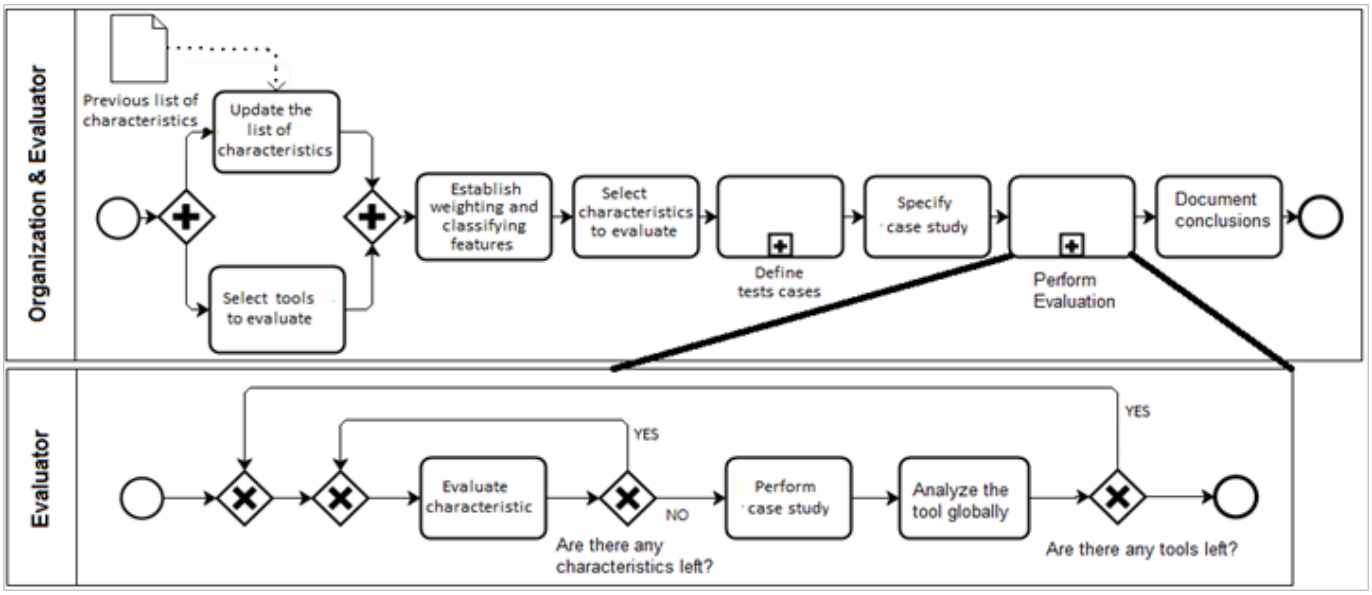


Fig. 1. Evaluation methodology process modeled in BPMN

level, and the conclusions are documented. A fair evaluation requires that practical and theoretical evaluated characteristics be weighed differently.

Besides the consideration of non-functional aspects, we made a couple of improvements with respect to this methodology. First, we use some basic set of characteristics for narrowing (Filtering aspect described in the last section) the selection of tools. We start with a comprehensive list of existent BPMS, and we filter them with respect to the existent (or expected) technological infrastructure of the organization (e.g., database and application servers, and development technologies) and non-technical aspects (e.g., open-source vs commercial, and local support). Second, we noticed that the scale of support is not an adequate scale for classifying non-functional results. Thus, we define a different evaluation method, as is explained in Section IV.

## IV. Evaluating Non-Functional Requirements

In this section, we provide a detailed description of the non-functional aspects of interest for BPMS, their evaluation method, and a brief introduction to the test cases used for their evaluation.

### A. Non-Functional Aspects

The non-functional aspects are part of the Technical module of our list of characteristics and it is basically a result of mixing the quality attributes taxonomy [18] and the Software product Quality Requirements and Evaluation (SQuaRE) standard [9]. This module comprehends the following set of categories and corresponding characteristics.

*1) Support:* In this category, we include those characteristics related to documentation and support mechanisms offered by the tools, specifically multi-language support.

*2) Reliability:* It refers to the capacity of the software for staying operative, within a defined time period and under a set of conditions defined. This category is focused on aspects related to the availability of software components. It also includes the capacity for being repaired offered by the product, taking into account modularity, reusability, analyzability and the ability to be modified and tested.

*3) Compatibility:* In this category, we group characteristics related to compatibility of products, i.e. the capacity of two or more systems or components to interchange information or to carry out its required functions when they share the same hardware or software environment. Compatibility takes into account several elements such as: i) the possibility of a product to coexist with other independent products, in a common environment or sharing resources, and ii) the capacity of the system to interchange information and use it.

*4) Performance:* It refers to the capacity of response of the software, either the required time to answer to specific events or the total number of events processed in a defined period of time.

*5) Portability:* It refers to the capacity of the product or component to be transfer in an effective and efficient way from one hardware, software, operational or using environment to another. It includes several related elements such as: i) the capacity of the software to provide support for different operating systems and/or browsers, ii) difficulties to install or uninstall the software in a successful way, and iii) the possibility to export the process and/or install the tool in another environment.

*6) Usability:* This category groups characteristics related to the ability of the tool for being used, focused on how easy it is for a user to learn how to use the product. Specifically, it takes into account the aesthetics of the user interface, the accessibility of manuals, error messages and suggestions.

*7) Security:* It refers to the capacity of the software to compliant to the levels of risk allowed, both for possible physical damages and for possible data risks.

In Table III there is a summary of the non-functional categories and their corresponding characteristics.

TABLE III
NON-FUNCTIONAL ASPECTS

| Category | Characteristic |
|---|---|
| Support | multi-language support |
| Reliability | Fault tolerance |
| | Engine availability |
| | Modeler availability |
| | Portal availability |
| | Maintainability |
| Compatibility | Co-existence |
| | Interoperability |
| Performance | Response time |
| | Throughput |
| | Capacity |
| Portability | Adaptability |
| | Installability |
| | Replaceability |
| | Separate environments |
| | External products |
| Usability | Learnability |
| | Operability |
| | User error protection |
| | User interface aesthetics |
| | Accessibility |
| Security | Confidentiality |
| | Integrity |
| | Non-repudiation |
| | Accountability |
| | Authenticity |

It is worth mentioning that each characteristic can contain sub-characteristics that allow a finer grained evaluation in each case. An example of this is:

- Engine, modeler and portal availability characteristics consider: mean time to failure, mean time to recover and support for clustering.
- Adaptability considers: support for different operating systems and support for different browsers.
- Learnability considers: existence of tooltips, descriptive errors, suggestion messages, user manual.

### B. Evaluation Method

In order to be able to determine the level of compliance of each characteristic, the evaluator must define the values for the labels: High, Medium and Low. These values will be used to evaluate all characteristics and to calculate thresholds following this procedure: taking the values defined for High, Medium and Low, we calculate the limit between Low and Medium as the mean between the values of Low and Medium, we call this value "minimum threshold", and the limit between High and Medium as the mean between the values of High and Medium, calling this value as "maximum threshold".

Also, as each characteristic can have several sub-characteristics, each one must be evaluated on its own. To calculate the level of support of the sub-characteristic we must define the values for L1 and L2 (limits for the defined ranks) which allow instantiating the support ranks from the given definitions for each sub-characteristic. The final value for the sub-characteristic (vsub) is obtained by means of the following defined ranks:

- vsub $<$ L1 $\rightarrow$ Level of support is Low.
- L1 $\leq$ vsub $<$ L2 $\rightarrow$ Level of support is Medium.
- L2 $\leq$ vsub $\rightarrow$ Level of support is High.

For each sub-characteristic, a value between 0 and 1 must be defined to provide a weight for each one, which will be multiplied for the level of support of each sub-characteristic. Adding up the results for each sub-characteristic and dividing between the sum of the weight of the sub-characteristics we obtain the final value for the level of support of the corresponding characteristic. Finally, based on the final value of the characteristic we can classify its level of support by taking into account the previously calculated thresholds:

- vf $<$ minimum threshold $\rightarrow$ Level of support is Low.
- minimum threshold $\leq$ vf $<$ maximum threshold $\rightarrow$ Level of support is Medium.
- maximum threshold $\leq$ vf $\rightarrow$ Level of support is High.

In the following, we present examples of the evaluation method definitions for selected characteristics from the Usability, Security and Performance categories. The definitions for the rest of the characteristics are similar following the general approach that we have presented above.

*1) Evaluating Usability:* In the first place, we provide a question for each characteristic or sub-characteristic to be answered by the evaluation. In Table IV we present the sub-characteristics and questions for the Learnability characteristic from the Usability category, showing one question for each sub-characteristic with answers in the scale Yes/No.

TABLE IV
EXAMPLE OF EVALUATION METHOD DEFINITIONS FOR LEARNABILITY FROM USABILITY

| Sub-characteristics | Question |
|---|---|
| Sub1 Tooltips | Does it provide tooltips? |
| Sub2 Descriptive errors | Does it have descriptive error messages? |
| Sub3 Suggestions | Does it provide suggestions of a possible solution when an error occurs? |
| Sub4 User manuals | Does it provide user manuals? |

Then, the level of support for each sub-characteristic is obtained answering each question:

- $R_n$ = 0 $\rightarrow$ Level of support is Low.
- $R_n$ = 1 $\rightarrow$ Level of support is High.

Being $R_n$ the response for each sub-characteristic, n $\in$ [1,4] for this characteristic.

*2) Evaluating Security:* In Table V we present the sub-characteristics and questions for the Integrity characteristic from the Security category. It also defines one question for each sub-characteristic, along with a way of evaluating each question.

TABLE V
EXAMPLE OF EVALUATION METHOD DEFINITIONS FOR INTEGRITY FROM SECURITY

| Sub-characteristics | Question |
|---|---|
| Sub1 Role definitions | Does it allow to define roles? |
| Sub2 Permissions on objects definitions | Does it provide differentiated access on documents by role? |
| Sub3 Restrictions based on roles | Does it allow visualizing different functions by role? |
| Sub4 User permissions mechanisms | Does it allow permissions administration? |
| Sub5 Document security and integrity | Does it store documents encrypted? |
| Sub6 Limited session time | Does it provide limited time for active sessions? |

Then, the level of support for each sub-characteristic is obtained answering each question:

- $R_n$ = 0 $\rightarrow$ Level of support is Low.
- $R_n$ = 1 $\rightarrow$ Level of support is High.

Being $R_n$ the response for each sub-characteristic, n $\in$ [1,6] for this characteristic.

Evaluation of sub-characteristics:

- Role definition: check that new roles can be defined, and define two of them with different levels: Admin and User.
- Permissions on objects definitions: upload and visualize documents with different levels of permissions for the roles Admin and User defined.
- Restrictions based on roles: verify that certain actions are only visible to the user with the Admin role.
- User permissions mechanisms: verify that a user with the Admin role can assign permissions to other roles defined in the hierarchy.
- Document security and integrity: upload a document and check that it is stored encrypted.
- Limited session time: verify that the tool provides an option to time-out the active session.

*3) Evaluating Performance:* In Table VI we present the sub-characteristics and questions for the Response Time characteristic from the Performance category. It defines one sub-characteristic and one question this time also with definition of parameters, along with a way of evaluating the characteristic.

TABLE VI
EXAMPLE OF EVALUATION METHOD DEFINITIONS FOR RESPONSE TIME, THROUGHPUT AND CAPACITY OF PERFORMANCE

| Sub-characteristics | Question |
|---|---|
| Response Time | What is the Response Time? |
| no sub-characteristics | What is the Throughput? |
| no sub-characteristics | What is the Capacity? |

*Response time:* we consider [20] for the definition of response time, i.e. the time from the start of a process until the start of its first task. We consider the following parameters:

- L1 = desired response time, in seconds.
- L2 = maximum acceptable response time, in seconds.
- vsub = response time

The level of support for the characteristic is obtained considering:

- vsub $<$ L1 $\rightarrow$ level of support is High.
- L1 $\leq$ vsub $<$ L2 $\rightarrow$ level of support is Medium.
- L2 $<=$ vsub $\rightarrow$ level of support is Low.

Taking into account the benchmark defined in [21] , the execution of a defined test case is automatized, at least a hundred times each of the following scenarios: one user, two users, four users, six users, eight users and ten users, sequentially. Based on the results of the test case executions, the average time will be considered as the value for the response time under evaluation.

*Throughput time:* we consider the following parameters:

- L1 = minimum percentage of processes that must execute in the period ExecutionPeriod.
- L2 = desired percentage of process that must execute in the period ExecutionPeriod.

By defining the value for the ExecutionPeriod and the quantity of services TotalServicesQuantity that will execute in the defined period, we calculate the Throughput by the formula:

$$vsub = \frac{StartedServicesQuantity \times 100}{TotalServicesQuantity} \quad (1)$$

being StartedServicesQuantity the quantity of services that started in the defined period.

Then the level of support for Throughput is calculated

- vsub $<$ L1 $\rightarrow$ level of support is Low.
- L1 $\leq$ vsub $<$ L2 $\rightarrow$ level of support is Medium.
- L2 $\leq$ vsub $\rightarrow$ level of support is High.

We defined a test case whose execution will be automatized for a period of thirty minutes. Taking into account the benchmark defined in [21], we defined the following cases: one user, two users, four users, six users, eight users and ten users, sequentially. The total quantity of services will be 1000.

*Capacity:* we consider the following parameters:

- L1 = minimum percentage of processes completed in the period ExecutionTime.
- L2 = desired percentage of process completed in the period ExecutionTime.

Then a execution period (ExecutionPeriod) is defined and the number of services that will execute in that period. The final value of the capacity is calculated by the following formula:

$$vsub = \frac{CompletedServicesQuantity \times 100}{TotalServicesQuantity} \quad (2)$$

being CompletedServicesQuantity the number of services that were completed in an execution time less or equal than the ExecutionTime defined by the evaluator (desired time of finishing process execution) and TotalServicesQuantity the

number of services that were completed in the execution period.

Then, the level of support for Capacity is calculated

- vsub $<$ L1 $\rightarrow$ level of support is Low.
- L1 $\leq$ vsub $<$ L2 $\rightarrow$ level of support is Medium.
- L2 $\leq$ vsub $\rightarrow$ level of support is High.

As with the Throughput characteristic, we defined a test case whose execution is automatized for a period of thirty minutes. Taking into account the benchmark defined in [21], we defined the following cases: one user, two users, four users, six users, eight users and ten users, sequentially. The total quantity of services will be 1000.

### C. Test Cases

We have extended our definition of test cases to provide support for the execution of the cases for the non-functional characteristics. As mentioned before, test cases are used to evaluate the support the BPMS provide for one or a set of characteristics (and/or sub-characteristics). We present here as an example the test cases for the characteristics performance and security.

*1) Security test cases:* The security test is shown in Figure 2. In this test case, we defined different roles to be able to test the sub-characteristics of security. It defines tasks that are performed by each role, where documents are uploaded, stored and visualized by different users, depending on the defined roles.

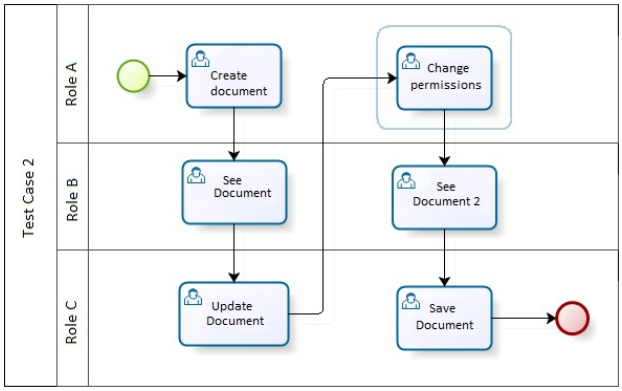


Fig. 2. Security characteristic test case

Detailed execution flow:

1) Start: the user starts a process instance.
2) Create document: this task is performed by users with role A where a document is created with reading permissions for the roles A and C.
3) Visualize document: this task is performed by the role B to try to visualize the document created by role A.
4) Update document: opens the document created by the role A and updates it.
5) Change permissions: this task is performed by the role A to update write/read permissions to allow users with role B to manipulate the documents.
6) Visualize document 2: a user with role B opens the document that was updated by role A.

Security sub-characteristics covered: Confidentiality, Integrity, Non-repudiation, Responsibility and Authentication.

*2) Performance test cases:* The performance test is shown in Figure 3. This test case is defined to evaluate characteristics from the performance category. Based on the benchmark [21], it presents a simple case, with a user task, without forms or external calls, to be able to easily obtain the defined times for the process execution.


Fig. 3. Performance characteristic test case

Detailed execution flow:

1) Start: the user starts a process instance.
2) The user that started the process instances completes the tasks and the process ends.

Performance sub-characteristics covered by the test: Response Time, Throughput and Capacity.

## V. Case Study

We carried out an additional evaluation of several BPMS platforms with the new defined non-functional characteristics, their evaluation method and the corresponding test cases. We have evaluated the following tools: jBoss BPMS, Bonita BPM, Intalio BPMS, Activiti, Bizagi BPMS, Camunda, Orchestra and Process Maker, along with Aris (but only the Aris Express module so we will not include it here).

We also defined as a new element of the methodology, a web tool to support the registration of different evaluations than can be carry out by different organizations, tailoring the selection of characteristics and their importance to their needs.To automate the test cases for the Performance category, we used a trial of the Microsoft visual studio tool, which allows recording the execution of the test cases within the web portal of the tools (only with IExplorer), and then running several executions of the test, presenting the data both in numerical and graphical forms.

In the following we present as examples, the results of executing the performance test cases for the tools.

### A. Performance test cases execution

In this section, we present the results of the executions of the test cases for the Response time characteristic, defining the values for the labels: L1 = 3 seconds and L2 = 10 seconds. Each test case was executed several times as defined, to obtain the average values for the characteristics for each tool. In each results graphic, the blue graph represents the values of the execution of the test case, and the orange the trend line.

*1) Bonita BPM:* In Figure 4 we show the results of the Response Time test case execution for Bonita BPM. The figure shows how the response time grows as the number of users increases, so the trend corresponds to a lineal function, implying that as the number of users increases, the response time grows in a directly proportional way. Taking the average of the values obtained in the test case execution, the response time for Bonita is 0,044 seconds.

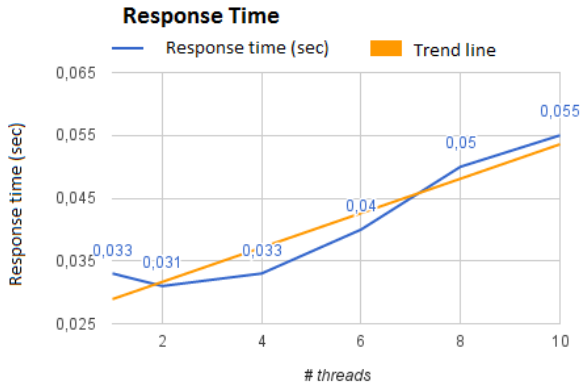


Fig. 4. Response time characteristic test case for Bonita

*2) Intalio BPMS:* In Figure 5 we show the results of the Response Time test case execution for Intalio BPMS, which is similar to Bonita. Taking the average of the obtained values, the response time for Intalio is 0,0395 seconds.

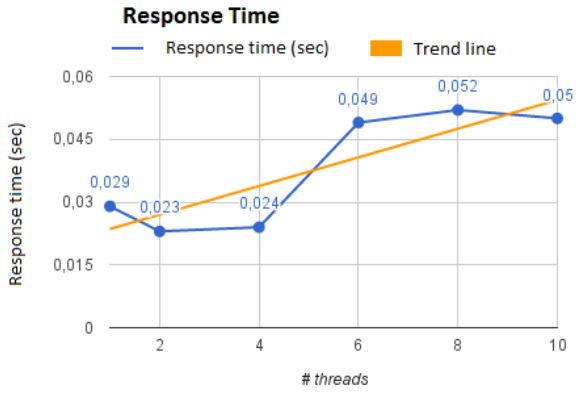


Fig. 5. Response time characteristic test case for Intalio

*3) Activiti BPMS:* In Figure 6 we show the results of the Response Time test case execution for Activiti BPMS. In this case, the trend is a second-degree polynomial function, meaning that the response time grows in quadratic proportion regarding the number of users added. Taking the average of the obtained values, the response time for Activiti is 0,455 seconds.

*4) Camunda BPMS:* In Figure 7 we show the results of the Response Time test case execution for Camunda BPMS, which is similar to Bonita. Taking the average of the obtained values, the response time for Camunda is 0,04 seconds.

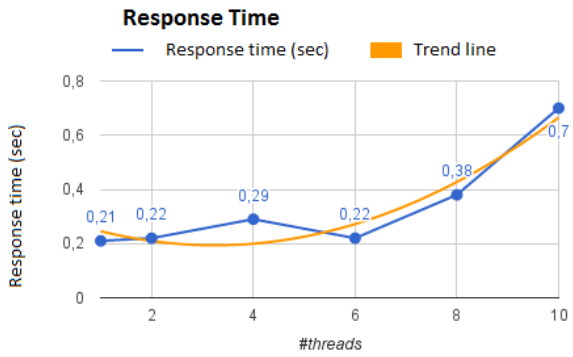


Fig. 6. Response time characteristic test case for Activiti

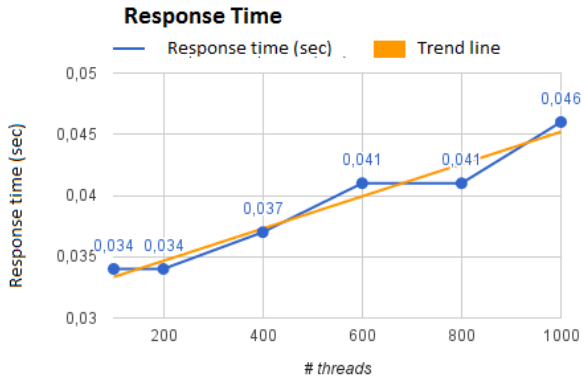


Fig. 7. Response time characteristic test case for Camunda

*5) ProcessMaker BPMS:* In Figure 8 we show the results of the Response Time test case execution for Process Maker. Similar to Activiti, the trend is a second-degree polynomial function as Activiti. Taking the average of the obtained values, the response time for Process Maker is 2,78 seconds, being the highest time, but below the L1 label value as defined.

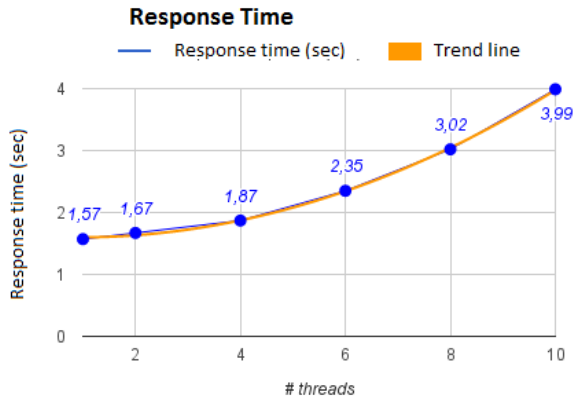


Fig. 8. Response time characteristic test case for Process Maker

Orchestra performed similar to Bonita, Intalio and Camunda, presenting the same trend line function, on the other

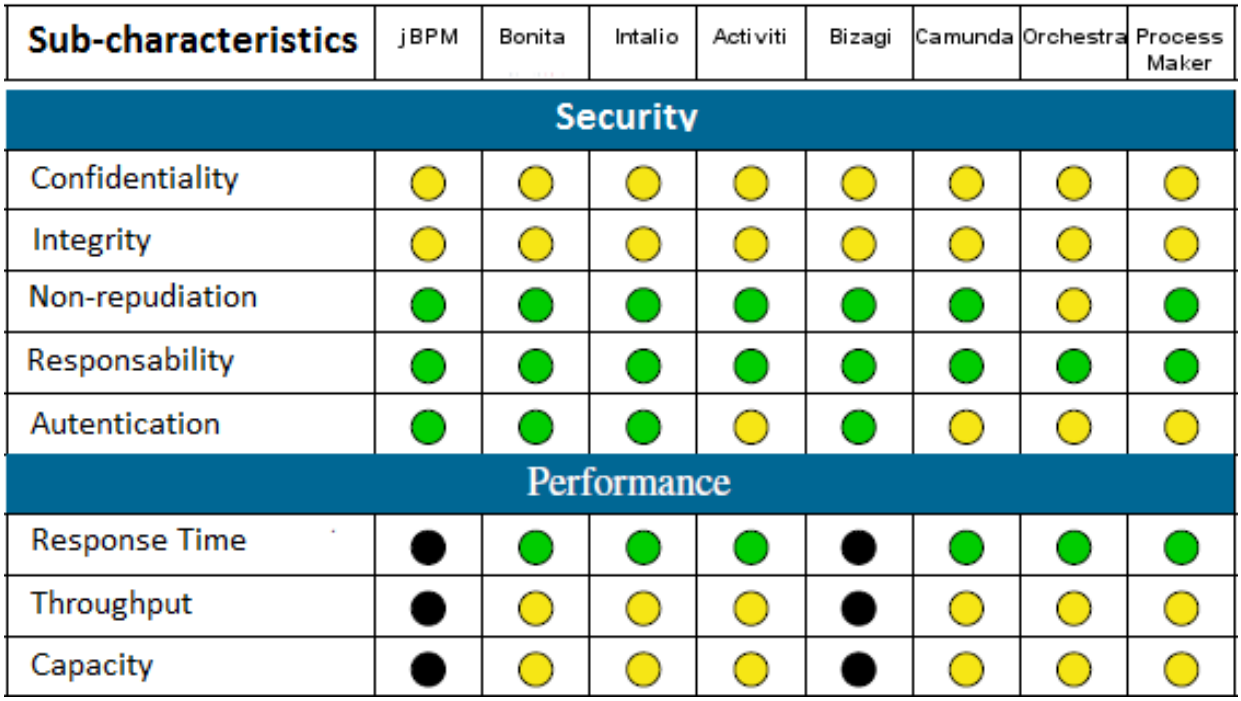


Fig. 9. Example of results for selected non-functional categories and sub-characteristics

hand Activity and Process Maker showed a similar trend line function but very different result values. Bizagi and jBPM presented some issues for the execution of the automated test cases, which prevented us to show their results.

In Figure 9 we present a summary of the evaluation of the selected characteristics and corresponding sub-characteristics. We use a semaphore-like notation where: green, yellow and red means high, medium and low levels of support of each characteristic, respectively. A black dot means that the characteristic was not evaluated.

It can be seen that regarding the non-functional characteristics performance and security, there is not a tool which stands out from the rest. We do not show here the evaluation results for others such as usability, which are also similar for most of the tools. Regarding selected functional characteristics, we saw in the evaluation, that some of them are still not supported by the tools, mainly process monitoring, process versioning, business rules, draft tasks, and document management.

As a result of the project we obtained a detailed evaluation of eight tools (and partially another one), using the characteristic list updated from previous applications. We defined a case study covering usual constructions in a business process, and defined test cases and detailed execution procedures to evaluate the newly non-functional aspects of BPMS we have included in the approach. Finally, a quantitative consolidated result can be delivered for each tool, along with the specific results for each sub-characteristic.

### *B. Web tool to support the evaluation of BPMS platforms*

We have developed a tool to support the evaluation of BPMS, which allows the management of: categories, characteristics, sub-characteristics, tools and corresponding versions, roles and users. It allows registering the evaluation results for each sub-characteristic and tool, generate evaluations by weighting selected characteristics (using a reference normalized evaluation we provide), and compare evaluation results from two tools.

In Figure 10 we present an example of the comparison functionality of the EvalBPMS tool (interface in Spanish only). We used the same semaphore-like notation than in Figure 9.

## VI. Conclusions and Future Work

In this paper, we have presented the extension of a systematic approach for evaluating BPMS tools [8]. The approach is based on a list of key characteristics for this kind of software which was extended for considering non-functional aspects of BPMS. We also define how these aspects can be evaluated by providing a set of technology-independent test cases. Finally, we show the practical application of our approach by evaluating several open-source and commercial BPMS.

As can be seen, there were many desirable aspects not supported in our methodology. In the case of the definition of roles participating in the evaluation process, and the filtering step, we made some minor changes to include them. We also extended the methodology with the inclusion of non-functional aspects, which is the focus of this article, based on well-known classifications [18], [9]. With respect to the use of historical data, we built a tool to support the approach, which is capable of using existing information in order to perform a comparative evaluation, and also simplifies the filtering of characteristics.

| | Activiti v.6.Beta2 (1,00) | JBoss BPM Suite v.6.1.0 (2,50) |
|---|---|---|
| Módulo **Técnico - Funcional** | | |
| Portal | 0,00 | 2,50 |
| Presentación de las tareas ⊞ | 🔴 (0,00) | 🟢 (2,50) |
| Módulo **Técnico - No funcional** | | |
| Usabilidad | 1,00 | 0,00 |
| Capacidad de aprendizaje ⊞ | 🟡 (1,00) | 🔴 (0,00) |

Fig. 10. Example of the evaluation results in the EvalBPMS tool

Finally, the definition of evaluation metrics is subject of future work.

We believe that our approach allows different organizations to tailor the evaluations to their needs, providing different results for each scenario defined, mainly regarding aspects of infrastructure hardware and software, language and architecture. Since the market for BPMS platforms is growing and each year several new tools emerge, we believe our approach can be a key element to consider and compare them.

## Acknowledgment

We would like to thank the undergraduate students who worked in the BPMS evaluation project: Alexandra Castelli, Germán Lagrega and Bettina Neira.

## References

[1] M. Weske, *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.

[2] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, "Business process management: a survey," in *Business Process Management, International Conference, BPM 2003, Proceedings*, ser. LNCS, vol. 2678. Springer, 2003, pp. 1–12.

[3] J. Chang, *Business Process Management Systems: Strategy and Implementation*. CRC Press, 2016.

[4] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[5] R. Garcês, T. Jesus, J. Cardoso, and P. Valente, *Open Source Workflow Management Systems: A Concise Survey*. Future Strategies Inc., 2009, pp. 179–190.

[6] "Gartner Group," http://www.gartner.com/technology.

[7] "Technology Evaluation Centers (TEC)," http://www.technologyevaluation.com/.

[8] A. Delgado, D. Calegari, P. Milanese, R. Falcon, and E. Garcia, "A systematic approach for evaluating BPM systems: Case studies on open source and proprietary tools," in *Open Source Systems: Adoption and Impact - 11th IFIP WG 2.13 Intl.Conf., OSS 2015, Proceedings*, ser. IFIP, vol. 451. Springer, 2015, pp. 81–90.

[9] ISO/IEC, "ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models," Tech. Rep., 2010.

[10] J. Sinur and J. Hill, "Magic quadrant for business process management suites." Gartner Inc., Tech. Rep., 2010.

[11] J. Sinur, W. Schulte, J. Hill, and T. Jones, "Magic quadrant for intelligent business process management suites," Gartner Inc., Tech. Rep., 2012.

[12] C. Richardson, D. Miers, A. Cullen, and J. Keenan, "BPM suites, q1 2013, how the top 10 vendors stack up for next-generation bpm suites," The Forrester Wave, Tech. Rep., 2013.

[13] A. Tsalgatidou, "Selection criteria for tools supporting business process transformation for electronic commerce," in *Proceedings of EURO-MED NET 98 Conference*, 1998, pp. 244–253.

[14] D. Morera, "Cots evaluation using desmet methodology & analytic hierarchy process (ahp)," in *Product Focused Software Process Improvement: 4th Intl. Conf., PROFES 2002, Proceedings*. Springer, 2002, pp. 485–493.

[15] P. K. Lawlis, K. E. Mark, D. A. Thomas, and T. Courtheyn, "A formal process for evaluating COTS software products," *IEEE Computer*, vol. 34, no. 5, pp. 58–63, 2001.

[16] D. Taibi, L. Lavazza, and S. Morasca, "OpenBQR: a framework for the assessment of OSS," in *Open Source Development, Adoption and Innovation, IFIP Working Group 2.13 on Open Source Software*, ser. IFIP, vol. 234. Springer, 2007, pp. 173–186.

[17] F. Tarawneh, F. Baharom, J. Yahaya, and F. Ahmad, "Evaluation and selection cots software process: the state of the art," *International Journal on New Computer Architectures and Their Applications (IJNCAA)*, vol. 1, no. 2, pp. 344–357, 2011.

[18] M. Barbacci, M. H. Klein, T. A. Longstaff, and C. B. Weinstock, "Quality attributes," Software Engineering Institute, Technical report CMU/SEI-95-TR-021, 1995.

[19] OMG, "Business process model and notation (BPMN) version 2.0," OMG, Tech. Rep., 2011.

[20] R. Aiello, "Workflow performance evaluation. PhD. Thesis," University of Salerno, Tech. Rep., 2004.

[21] J. Barrez, "The activiti performance showdown 2015," http://www.jorambarrez.be/blog/tag/performance/.