# How Does Refactoring Affect Understandability of Business Process Models?

Ricardo Pérez-Castillo, Maria Fernández-Ropero, Mario Piattini

Instituto de Tecnologías y Sistemas de Información (ITSI),
University of Castilla-La Mancha,
Paseo de la Universidad 4, 13071, Ciudad Real, Spain
[ricardo.pdelcastillo, marias.fernandez, mario.piattini]@uclm.es

Danilo Caivano

Department of Informatics, University of Bari,
Via E. Orabona, 4, 70126 Bari, Italy
caivano@di.uniba.it

***Abstract*—Business process refactoring techniques have been often provided for business process manually modeled. Unfortunately, no many refactoring techniques lie in reversing business process models obtained from existing information systems, which need, even more, to be refactored. Hence, there is no strong empirical evidence on how the understandability of business process models is affected by this kind of refactoring techniques. This paper is aimed at providing a case study with two real-life information systems, from which 40 business process models were obtained by reverse engineering. The empirical study attempts to quantify the effect to the understandability of the order of refactoring operators as well as the previous refactoring actions. The main implication of the obtained results are a set of rules that may be used to optimize the understandability by means of the prioritization and configuration of refactoring techniques specially developed for business process models retrieved by reverse engineering.**

***Keywords-Business Process, Refactoring, Understandability***

## I. Introduction

Business process models depict the sequence of coordinated activities that an organization carried out to achieve their business goal [22]. Business processes models are considered one of the most important assets for organizations due to two main reasons. An appropriate management of business process models first helps companies to quickly adapt their business goals and structures to environmental changes while maintaining or even improving their competitiveness [10]. Secondly, from a software engineering viewpoint, business process models are the starting point for obtaining the requirements of new-development or maintenance projects [19].

Since business processes exist within organization in an intangible way, business process modeling provides tangible descriptions of them allowing their management. Unfortunately, not all business processes are modeled in the organization, or when business processes are modeled, these might be out of date and therefore could be misaligned regarding the enterprise information systems that give support to such processes [9]. Similarly to the chicken-and-egg dilemma, there is no way to truly know which came first, business process models or enterprise information systems. In fact, outdated and misaligned business process models (together with organizations that deal with business process modeling at the first time) are the key motivations for reverse engineering techniques devoted to retrieving the actual business process models supported by the existing information systems [17, 20].

Reverse engineering techniques for obtaining business process models are often less error-prone and time-consuming than manual (re-)modeling from scratch. However, reverse engineering techniques imply an inherent semantic loss due to the abstraction increase [2]. As a result, although outdated and misalignment problems are addressed, quality of the retrieved models is eroded. Reverse engineering techniques could retrieve, for example, incomplete or inaccurate business process models (i.e., with missing and wrong elements), or even modes with inadequate understandability and modifiability levels (e.g., with a vast amount of fine-grained and ambiguity elements) [7].

In order to cope with understandability and modifiability faults, refactoring of business process models has been widely used [7]. These techniques change the internal structure of business process models without altering or modifying their external behavior. There exist in literature several refactoring approaches to be applied with business process models [3, 11, 21]. Unfortunately, there are no refactoring techniques specially developed for those models obtained through reverse engineering and some of their peculiarities such as missing elements, mining of non-relevant elements, fine granularity, and so on. In addition to this drawback, the main problem is that current refactoring techniques often apply several refactoring operators to deal with different *bad smells*, i.e., refactoring opportunities (e.g., non-relevant elements, fine-grained elements, etc.). The application of different refactoring operators is commonly done in an arbitrary way [7]. Nevertheless, it has been demonstrated that the order and subset of refactoring operators lead to different results in terms of the understandability and modifiability gain [6].

This paper therefore focuses on the assessment and optimization of the understandability of business process models during refactoring. Hence, this paper tries to provide a set of arguments and insights through empirical validation so that the community can have a better answer to the question: *how affect refactoring to the business process model understandability?* In order to provide the mentioned insights for such answer this paper conducts a case study with two industrial information systems, from which 40 business process models were first obtained by reverse engineering. After that, those models were refactored by using IBUPROFEN [6], a refactoring approach, by setting up different orders and subsets of refactoring operators. IBUPROFEN is used in this study since this approach and its supporting tools were specially developed for refactoring business process models obtained by reverse engineering from existing source code. Finally, all the obtained business process models are inspected to evaluate the understandability gains and determine the best configurations.

The remainder of this paper is organized as follows. Section II briefly presents related work. Section III introduces IBUPROFEN, the approach used for refactoring. Section IV explains the case study in detail. Finally, Section V discusses conclusions and future work.

## II. Related Work

Business process management has become a valuable activity for managing organizations from an operational perspective. *Dijkman et al.* [4] provide various techniques for improving their management as merging, mining, refactoring, re-use, among other. Particularly, refactoring has been used for several authors in literature for improving the quality degree of business process models. For example, *Weber et al.* [21] collect a catalogue of process model *smells* for identifying refactoring opportunities and provide a set of behavior-preserving techniques for refactoring to avoid redundancies and increase in the complexity of the model. Similarly, *Dijkman et al.* [3] show a development of a technique based on metrics to detect refactoring opportunities and *La Rosa et al.* [11] identify patterns to reduce the model complexity through compacting, compositing, merging, amoung other. *Leopold et al.* [12], for their part, focus on refactoring of activity labels in a business process model following a verb-object style.

Concerning to the order of application of the refactoring operators or the selection of a sub-set of operators, previous approaches rely on the expert decision, or simply define an arbitrary sub-set and order. Although *Gambini et al.* [8] propose the automation of de business process models refactoring through a technique for automatically fixing the refactoring scenarios using Petri nets, the order of application is not mentioned. *Fernandez-Ropero et al.* [6] demonstrate that the order of application of refactoring operators affect the understandability and modifiability. However, that preliminary work does not assess the best sub-sets or application orders to achieve the highest understandability.

## III. IBUPROFEN

IBUPROFEN [6] (Improvement and BUsiness Process Refactoring OF Embedded Noise) is a framework with which to refactor business process models particularly retrieved by reverse engineering. IBUPROFEN allows applying different refactoring operators taking into account the assessment of various measures related to the modifiability and understandability of business process models [7] such as density, size, connectivity, separability, etc.

IBUPROFEN is supported by a tool specially designed for business process models represented according to the BPMN (Business Process Modeling Notation) [14]. The tool has being implemented as an Eclipse$^{TM}$ plug-in [1]. Hence, the supporting tool can be used in combination with other Eclipse™ plug-ins aimed, for example, at obtaining business process models from the source code of existing information systems.

IBUPROFEN provides a set of ten refactoring operators (see TABLE I) grouped into three categories in terms of the bad smells that the operators address: (i) relevant elements maximization; (ii) fine-grain granularity reduction; and finally, (iii) completeness maximization.

### *A. Relevant Elements maximization*

This category groups five refactoring operators (R1 to R5) responsible for removing non-relevant elements found in business process models as isolated tasks, sheet tasks and inconsistencies. Moreover, nested gateways can origin an increase in the complexity of business process models, thus these are replaced by equivalent, light-weight structures.

**R1** removes nodes (i.e., tasks, gateways or events) in the business process model that are not connected with any other node in the business process model. **R2** discards elements in the business process model that are considered sheet nodes. These nodes can be gateways or intermediate events that have no successor nodes. In turn, **R3** merges consecutive gateways of the same type when the first gateway has only one output and the second has only one input, i.e., nested gateways. **R4** removes sequence flows in the business process model that are considered as inconsistent. When two tasks are connected through a cut node, as an intermediate event or a gateway, and through a direct sequence flow this sequence flow are removed. Finally, **R5** removes gateways that connected only two nodes, i.e. with one input and one output. Such gateways are removed and a direct sequence flow is created between related nodes.

### *B. Fine-grained granularity reduction*

The different granularity of business tasks and callable units in existing information systems constitutes another important challenge [17]. According to the approach proposed by *Zou et al.* [24], each callable unit in an information systems is considered as a candidate business task. However, existing systems typically contain thousands of callable units, some of which are large ones supporting the main business functionalities of the system, while many are very small and do not directly support any business activity. In other situations a set of small callable units together supports a business activity. As a consequence, this category provides two refactoring operators (R6 and R7) to deal with large sets of fine-grained business tasks and data objects:

**R6** transforms each task in a compound task when the task *T* has several subsequent tasks which are in turn connected with a round-trip sequence flow to the task *T*. This scenario is due to each callable unit is transformed as a task during the reverse engineering stage when a certain callable unit can invoke another callable unit returning a value to the first one. In this case, the refactoring operator creates a compound task with a start and end event connected with each subsequent task through the respective split and join exclusive gateways. Additionally, **R7** combines data objects that are input and/or output of a task. The combination is possible when those data objects are exclusively used (written or read) for that task. The combination is done when the number of data objects is above a threshold. In order to mitigate the collateral semantic loss, all the names of the grouped data objects are saved in the documentation attribute defined by the BPMN specification.

## *C. Completeness Maximizatioin*

Any reverse engineering technique implies an increase of the abstraction degree, and therefore a semantic loss. For this reason, R8 to R10 operators are provided to deal with semantic loss by means of the incorporation of further elements. The refactoring operators are the following:

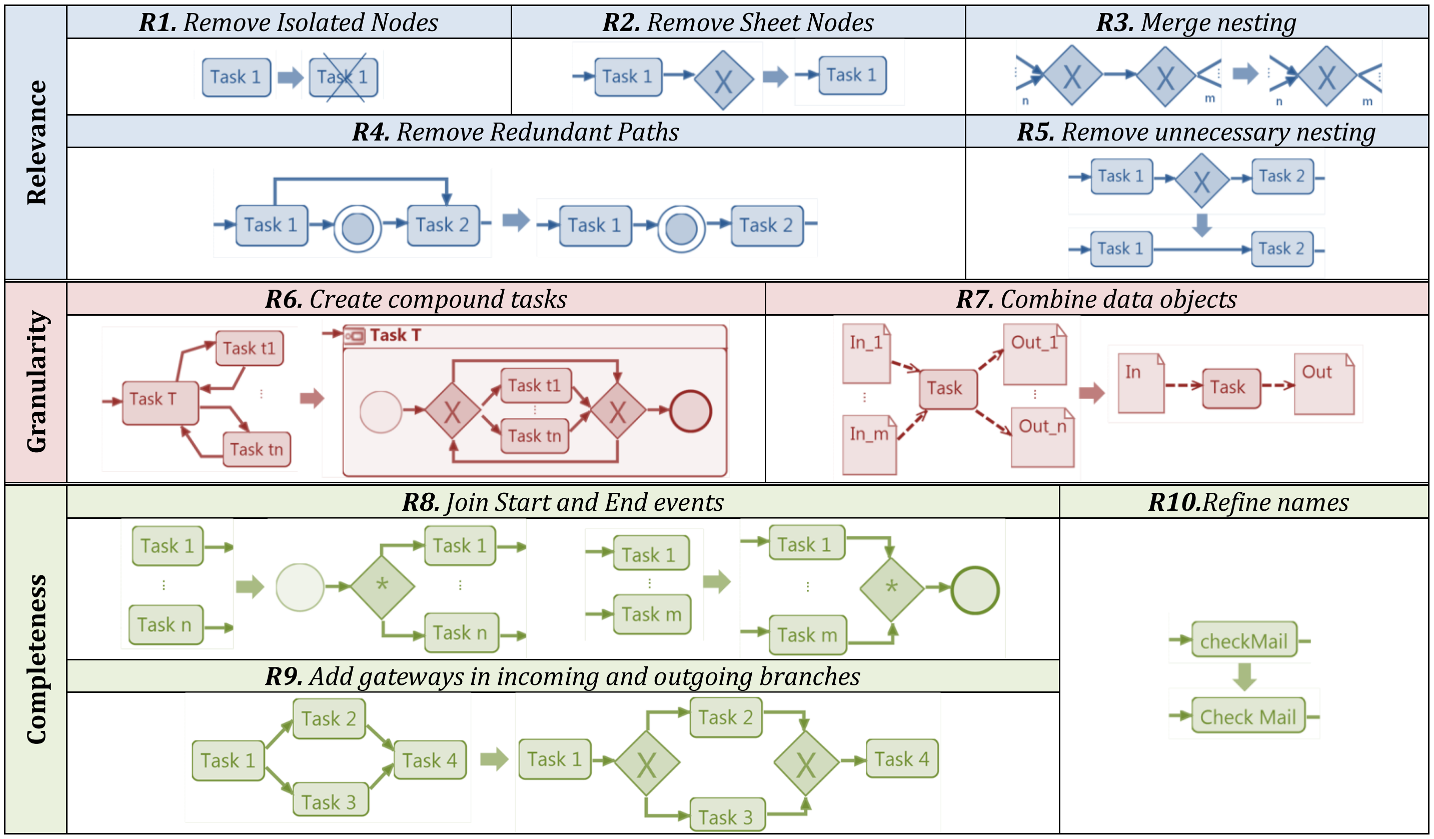
**R8** joins the start and end event with the starting and ending tasks, respectively. These events are created whether such events were not created by reverse engineering. When there are several starting tasks the refactoring operator adds a split complex gateway between the start event and starting tasks. Similarly, if there are several ending tasks, the refactoring operator adds a join complex gateway between ending tasks and the end event [13]. Furthermore, due to the usage of reverse engineering to retrieve business process models, it is possible to obtain models without following some of the modeling guidelines in accordance with the BPMN specification with regard to the gateways. **R9** therefore adds a join and split exclusive gateways when a certain task respectively has several precursor or subsequent tasks. Finally, **R10** improves names and labels of business tasks that were obtained almost directly from methods or functions of legacy source code through reverse engineering. These labels usually follow the camel case format (i.e., the concatenation of various capitalized words) in accordance with naming conventions present in most programming approaches. In an effort to have more understandable names, this refactoring operation split these labels into ones with various words.

## IV. CASE STUDY

This section provides a case study with two real-life information systems. The case study has been conducted by following the formal protocol developed by *Runeson et al*. [18] for conducting and reporting case studies in the software engineering field. Hence, the following sections show the stages proposed in the formal protocol: case study design, case selection procedure, execution procedure and data collection, analysis and interpretation, and finally, threats to the validity.

The *object* of this case study is the understandability of business process models after refactoring and the *purpose* of this case study is to evaluate how the execution order of the different refactoring operators and previous refactoring actions affect to the understandability. Taking into account the object and purpose of the study two main research questions are provided.

TABLE I. IBUPROFEN'S REFACTORING OPERATORS



**RQ1:** *How does the order of the application of refactoring operators affect to the understandability of business process models?*

**RQ2:** *How does previous refactoring affect to the understandability achieved with the application of certain refactoring operators?*

## A. *Case Study Design*

The case study follows the *embedded case study* design according to the classification proposed by *Yin* [23], whereby the case study consists of a multi case (i.e., it focuses on two information systems) but considers several analysis units as *independent variable* within the case, i.e., all the different business processes models retrieved from both information system. Therefore, the study consists of applying the three refactoring categories: relevance (R), granularity (G) and completeness (C) in different combinations and obtaining business process models. Such models are in turn analyzed to evaluate understandability in accordance with RQ1 and RQ2. In order to quantify understandability, size, connectivity, separability and density [5] measures are used as *dependent variables*.

**Size** is the number of nodes in a business process model (i.e., business tasks, gateways, data objects and events). This measure affects negatively to the understandability, i.e. a higher size difficult the understandability of a certain business process model [13]. **Connectivity** measures the ratio between the total number of arcs in a business process model (i.e., sequence flows and associations) and the total number of nodes. This measure negatively affects the understandability since a lower connectivity implies business process models more understandable due to a lower intricacy. **Separability** represents the ratio between the number of cut-vertices in a business process model (i.e. nodes that serve as bridges between otherwise strongly-connected components) and the total number of nodes. Separability positively affects to the understandability. **Density** is the ratio between the total number of arcs in a business process model and the theoretical maximum number of possible arcs regarding the number of nodes. The lower density, more understandable business process models.

## B. *Case Selection Procedure*

To select the case under study a set of selection criteria were formulated in order to rigorously select the source system: (1) the system should be a real-life information system currently in production; (2) and with a considerable size (to avoid toy programs) which ensure that the system supports a great number of business processes; (3) the system should be written in Java language to be able to use the MARBLE tool [15]. MARBLE is the tool used to recover business process models from existing Java code. This tool was selected because is released as an Eclipse plug-in and it therefore can be easily integrated with the IBUPROFEN tool.

After analyzing various information systems of partner companies, two cases were selected in accordance with the mentioned criteria: *Tabula* and *XCare*. *Tabula* is a web application of 33.3 KLOC (thousands of lines of code) devoted to create, manage and simulate decision tables for associating conditions with domain-specific actions. *XCare* is a mobile application of 9.9 KLOC intended for diabetes patients, which analyzes blood (through an external device) and suggests diet plans.

## C. *Execution Procedure and Data Collection*

The procedure to be performed to execute the case study consists of a set of steps. (i) A sample of 40 business process models are mined, by using MARBLE [16], from the source code from both information systems under study. (ii) After that, IBUPROFEN refactoring operators are executed in all the possible orders in terms of the three categories, so six different execution orders are considered (i.e., RGC, RCG, CRG, CGR, GCR and GRC). (iii) The mentioned measures are computed through IBUPROFEN tool after the execution of each category as well as before refactoring (i.e., four measurements for each execution order are taken). These semiautomatic steps are executed in a computer with a 2.66 GHz dual processor and 4.0 GB RAM.

Data collected during execution is used to compute the normalized gains after the execution of each category. TABLE II presents the normalized gains for each previous combination of refactoring category. This data represents the gain evolution for all the measures in accordance with the position in which a category is executed and regarding to the previous refactoring actions. Size, density, connectivity and separability cells are mean values computed for all the 40 business process models. The whole data, including base data directly obtained from the execution of the study is online available[1].

TABLE II. GAIN ON AVERAGE FOR EACH CATEGORY WITH DIFFERENT ORDERS

| Cat. | Pre-Act. | Size | Density | Connectivity | Separability |
|---|---|---|---|---|---|
| Relevance | - | 0.390 | -3.959 | -0.597 | 0.470 |
| | G | 0.500 | -7.798 | -0.954 | 0.548 |
| | C | 0.127 | -0.669 | -0.171 | 0.154 |
| | GC | 0.157 | -0.896 | -0.231 | 0.184 |
| | CG | 0.142 | -0.848 | -0.207 | 0.172 |
| Granularity | - | 0.269 | 0.051 | 0.218 | 0.064 |
| | R | 0.231 | -0.199 | 0.146 | 0.070 |
| | C | 0.072 | -0.067 | 0.022 | 0.068 |
| | RC | 0.107 | -0.114 | 0.028 | 0.059 |
| | CR | 0.085 | -0.103 | 0.009 | 0.078 |
| Correctness | - | -0.476 | -0.318 | -0.601 | -0.325 |
| | R | -0.341 | 0.163 | -0.082 | -0.152 |
| | G | -0.530 | -0.793 | -1.252 | -0.373 |
| | RG | -0.477 | 0.080 | -0.315 | -0.280 |
| | GR | -0.459 | 0.140 | -0.225 | -0.256 |

## D. *Analysis and Interpretation*

The inspection of data collected in TABLE II suggests that results highly vary with regards to the order in which each refactoring category is applied. These values also depend on the previous refactoring applied. However, in order to figure out whether these observations reflect a common pattern rather than the random effect, a statistical hypothesis testing were conducted for assessing the real effect of the application order.

For this purpose, the *Kruskal-Wallis* (KW) test was used. The KW test is a non-parametric method supporting a one-way analysis of variances by ranks. The KW test is used for comparing more than two non-related samples. Thus, the null

[1] http://alarcos.esi.uclm.es/per/mfernandez/

hypothesis is $H_0$: $\mu_1 = \mu_2 = \mu_n$, while the alternative hypothesis means that there is a significant difference between the means of sub-samples, i.e., H1: $\mu_1 \neq \mu_2 \neq \mu_n$. In this study, the different sub-samples were selected according to the five different configurations (order and previous actions). For example, the five samples of relevance (R) are in which R is applied at the beginning, is applied in second place (CR or GR), or is applied at the end (CGR or GCR). TABLE III provides the results of the KW test, whose inspection shows that the order (RQ1) and previous refactoring of all the categories (RQ2) affect the gain achieved at least for some of the measures. In case of relevance, the configuration affects to all the measures. In case of granularity, the order and previous refactoring affect to size, density and connectivity gain, but do not affect to separability. Finally, in case of completeness, the configuration only affects to density and connectivity. These results demonstrate that the application in an arbitrary order is not a good idea.

Having known there is a difference between different configurations, it is necessary (in order to complete the answer of research questions RQ1 and RQ2) to figure out which certain configuration is better than other in each category. Figure 1 graphically shows these variances. Regarding Relevance, the best choice was to apply it in the second place after granularity if the goal is to maximize size and separability. However, density and connectivity gains, which are always negative, are better if the relevance category is applied in second place after correctness refactoring. Concerning granularity, the best combination was to apply it at the beginning to achieve the greatest gain of size, density and connectivity. However, the best separability was achieved when granularity is applied at the end after correctness and relevance categories. Anyway, the differences of separability gains are negligible for every order (see Figure 1). Finally, with respect to completeness, most gains are unfortunately negative. Despite this fact, the best order in every case is to apply completeness in the second place after relevance.

After analyzing outgoing results, some rules to prioritize the application of refactoring categories can be derived so that research question can be fully answered. The first insight is that refactoring operators related to relevance should be applied in second place. Particularly, after granularity refactoring if the gain of size and separability are prioritized and after completeness if density and connectivity gain has to be maximized. The second rule is about granularity category, which should be applied in the first place. The third rule about completeness states that it should be applied in second place after relevance refactoring operators.

## E. *Validity Evaluation*

This section presents the threats to the validity of this case study and possible actions to mitigate them. There are mainly three types of validity: internal, construct and external. As far as the *internal* validity is concerned, a sample of 40 business process models was retrieved from a two information systems, and it is therefore possible to obtain statistically representative results. Nevertheless, the study may be replicated by using more information systems, to attain a larger sample of business process models. Anyway, there are two decisive threats. The first one is related to the way in which business process models were retrieved by reverse engineering, i.e., through MARBLE. This supporting tool was used to obtain the business process models, could be a factor that affects the initial sample of business process models. Secondly, the set of refactoring operators included in IBUPROFEN as well as their categories is a threat to the generalization of the results. The replication of the study by using different refactoring operators and techniques may be a mean for mitigating these threats.

TABLE III. KRUSKAL-WALLIS TEST RESULTS

| | Size | | Density | | Connectivity | | Separability | |
|---|---|---|---|---|---|---|---|---|
| | $\chi^2$ | Sig. | $\chi^2$ | Sig. | $\chi^2$ | Sig. | $\chi^2$ | Sig. |
| **Relevance** | 48.8 | 0.000 | 20.4 | 0.000 | 24.1 | 0.000 | 52.5 | 0.000 |
| **Granularity** | 24.6 | 0.000 | 21.7 | 0.000 | 25.8 | 0.000 | 1.6 | 0.801 |
| **Completeness** | 3.45 | 0.485 | 35.6 | 0.000 | 44.7 | 0.000 | 5.7 | 0.226 |

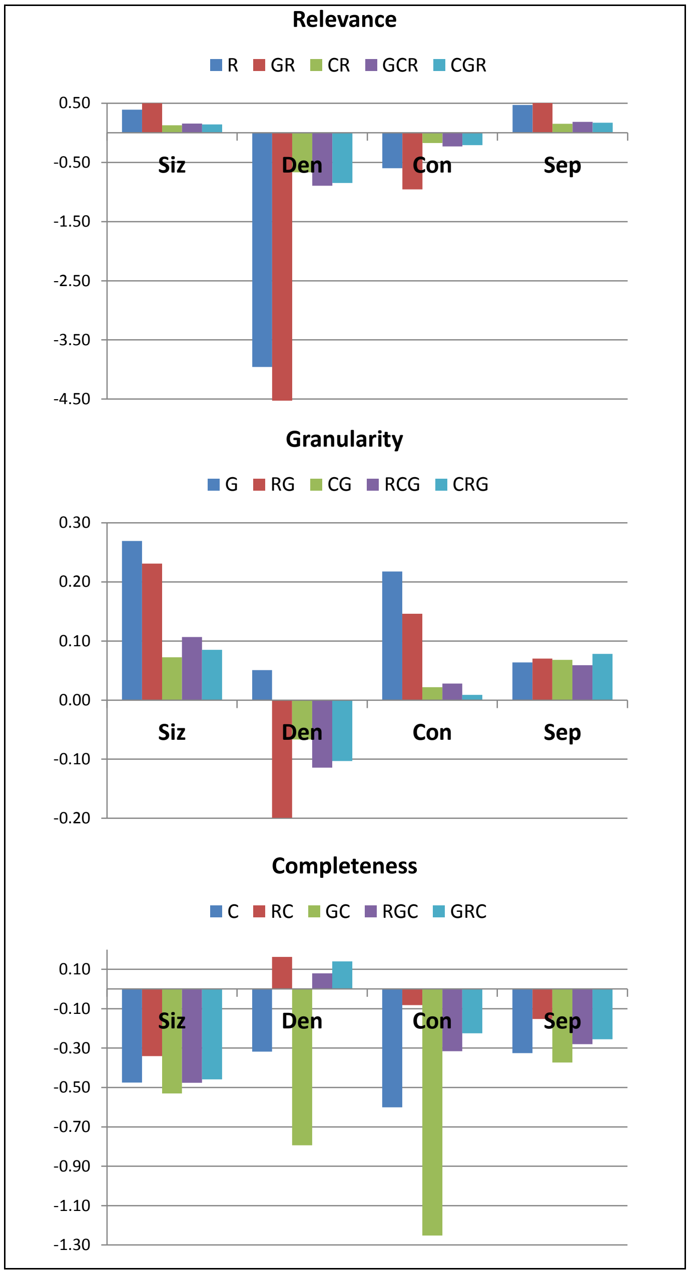


Figure 1. Behaviour of categories with different orders and previous actions

Moreover, with respect to the *construct* validity, the selected measures (size, density, connectivity and separability) were suitable for assessing the theoretical understandability of business process models. However, a more practical approach based on expert viewpoint could be used to assess the understandability of business process models. Finally, *external* validity is concerned with the generalization of the results. This study considers the whole population to be business process models retrieved by reverse engineering from legacy information systems. The results obtained can be strictly generalized to this population with the particularity that all the information systems under study are based on Java platform. This restriction is related to the mentioned supporting tools used in the study. This threat may be mitigated by replicating the study using systems implemented in different platforms.

## V. CONCLUSIONS AND FUTURE WORK

Business process model refactoring has proved to be a good mechanism for dealing with understandability problems and other faults. Unfortunately, most refactoring techniques only address business process models manually modeled and hardly ever consider reversing models semi-automatically retrieved from existing information systems. This paper precisely focuses on this kind of refactoring techniques by means of an empirical study that tries to assess how different configurations of refactoring affect the understandability gain. However, the understandability of business process model is difficult to be measured. On one hand, the understandability additionally depends on the people in charge of use, manage or evaluate such business process models, which is individually subjective. On the other hand, understandability of business process models that were previously refactored could vary due to the application of different refactoring operators. In fact, some operators might lead to a worse understandability. This study precisely attempts to establish links between different refactoring configurations (in terms of categories applied, i.e., relevance, completeness and granularity) and the understandability gain, which is measured with size, density, connectivity and separability of business process models. The study's results reflect that refactoring categories can be prioritized concerning the order in which to be applied as well as the previous refactoring actions so that the understandability gain can be optimized.

## ACKNOWLEDGMENTS

This work was supported by the FPU Spanish Program and the R&D projects PEGASO/MAGO (TIN2009-13718-C02-01) and GEODAS-BC (TIN2012-37493-C03-01).

## REFERENCES

[1] Alarcos Research Group. IBUPROFEN. 2012; Available from: http://marketplace.eclipse.org/node/423052.

[2] Canfora, G., M. Di Penta, and L. Cerulo, Achievements and challenges in software reverse engineering. Commun. ACM, 2011. 54(4): p. 142-151.

[3] Dijkman, R., B. Gfeller, J. Küster, and H. Völzer, Identifying refactoring opportunities in process model repositories. Information and Software Technology, 2011.

[4] Dijkman, R., M.L. Rosa, and H.A. Reijers, Managing large collections of business process models—Current techniques and challenges. Computers in Industry, 2012. 63(2): p. 91.

[5] Fernández-Ropero, M., R. Pérez-Castillo, I. Caballero, and M. Piattini, Quality-Driven Business Process Refactoring, International Conference on Business Information Systems (ICBIS 2012). 2012 p. 960-966.

[6] Fernández-Ropero, M., R. Pérez-Castillo, J.A. Cruz-Lemus, and M. Piattini, Assessing the Best-Order for Business Process Model Refactoring. 2013. p. 1400-1406.

[7] Fernández-Ropero, M., R. Pérez-Castillo, and M. Piattini, Refactoring Business Process Models - A Systematic Review, in 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012). 2012, INSTICC: Wroclaw, Poland. p. 140-145.

[8] Gambini, M., M. La Rosa, S. Migliorini, and A. Ter Hofstede, Automated error correction of business process models. Business Process Management, 2011: p. 148-165.

[9] Heuvel, W.-J.v.d., Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems). 2006: The MIT Press.

[10] Jeston, J., J. Nelis, and T. Davenport, Business Process Management: Practical Guidelines to Successful Implementations. 2nd ed. 2008, NV, USA: Butterworth-Heinemann (Elsevier Ltd.). 469.

[11] La Rosa, M., P. Wohed, J. Mendling, A.H.M. ter Hofstede, H.A. Reijers, and W. van der Aalst, Managing process model complexity via abstract syntax modifications. Industrial Informatics, IEEE Transactions on, 2011. 7(4): p. 614-629.

[12] Leopold, H., S. Smirnov, and J. Mendling, Refactoring of process model activity labels, in Proceedings of the Natural language processing and information systems, and 15th international conference on Applications of natural language to information systems. 2010, Springer-Verlag: Cardiff, UK. p. 268-276.

[13] Mendling, J., H.A. Reijers, and W.M.P. van der Aalst, Seven process modeling guidelines (7PMG). Information and Software Technology, 2010. 52(2): p. 127-136.

[14] OMG. Business Process Modeling Notation Specification 2.0. 2011; Available from: http://www.omg.org/spec/BPMN/2.0/PDF/.

[15] Pérez-Castillo, R., M. Fernández-Ropero, I. García Rodríguez de Guzmán, and M. Piattini, MARBLE. A Business Process Archeology Tool, in 27th IEEE International Conference on Software Maintenance (ICSM'11). 2011, IEEE Computer Society: Williamsburg, Virginia, USA. p. 578-581.

[16] Pérez-Castillo, R., M. Fernández-Ropero, I.G.-R.d. Guzmán, and M. Piattini, MARBLE. A Business Process Archeology Tool, in 27th IEEE International Conference on Software Maintenance (ICSM 2011). 2011: Williamsburg, VI. p. 578 - 581

[17] Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Generating Event Logs from Non-Process-Aware Systems Enabling Business Process Mining. Enterprise Information System Journal, 2011. 5(3): p. 301–335.

[18] Runeson, P. and M. Höst, Guidelines for Conducting and Reporting Case Study Research in Software Engineering. Empirical Softw. Eng., 2009. 14(2): p. 131-164.

[19] Sommerville, I., P. Sawyer, and S. Viller, Viewpoints for Requirements Elicitation: A Practical Approach, in Proceedings of the 3rd International Conference on Requirements Engineering: Putting Requirements Engineering to Practice. 1998, IEEE Computer Society. p. 74-81.

[20] van der Aalst, W., Process Mining: Overview and Opportunities. ACM Transactions on Management Information Systems (TMIS), 2012. 3(2): p. 7.

[21] Weber, B., M. Reichert, J. Mendling, and H.A. Reijers, Survey paper: Refactoring large process model repositories. Comput. Ind., 2011. 62(5): p. 467-486.

[22] Weske, M., Business Process Management: Concepts, Languages, Architectures. 2007, Leipzig, Germany: Springer-Verlag Berlin Heidelberg. 368.

[23] Yin, R.K., Case Study Research. Design and Methods. 3rd ed. 2003, London: Sage.

[24] Zou, Y. and M. Hung, An Approach for Extracting Workflows from E-Commerce Applications, in Proceedings of the Fourteenth International Conference on Program Comprehension. 2006, IEEE Computer Society. p. 127-136.