A Framework to Manage Knowledge from Defect Resolution Process

Grégory Claude, Marc Boyer, Gaël Durand, Florence Sèdes

2011 IEEE Conference on Commerce and Enterprise Computing

A Framework to Manage Knowledge from Defect Resolution Process

Grégory Claude[A,B], Marc Boyer[C], Gaël Durand[B], Florence Sèdes[A]
[A] Université de Toulouse, Université Paul Sabatier, IRIT UMR 5505, France
{Gregory.Claude, Florence.Sedes}@irit.fr
[B] Intercim LLC, Eagan, USA / Intercim, Paris, France
{GClaude, GDurand}@intercim.com
[C] Université de Toulouse, Université Paul Sabatier, Inserm UMR 825, France
Marc.Boyer@iut-tlse3.fr

***Abstract*—This paper presents a framework for the management, the processing and the reuse, of information relative to defects. This framework is based on the fact that each defect triggers a resolution process in which information about the detected incident (i.e. the problem) and about the applied protocol to resolve it (i.e. the solution) is collected. These different types of information are the cornerstone of the optimization of corrective and preventive processes for new defects. Experimentations show that our prototype provides a very satisfactory quality of results with good performances.**

Keywords-maintenance; defect; knowledge management; information processing; information reuse.

I. INTRODUCTION

To ensure the quality of a final product, processing and tracking defects that occur during its manufacturing process have become essential activities. Indeed, information relative to defects may represent a large percentage of the final volume of product information. In maintenance activities, we can notice many benefits in leveraging information from defects [1], especially in aeronautical, aerospace and pharmaceutical industries where complex highly regulated and often manual production activities result in high rates of defects during the manufacturing process. A first benefit, in the short run and in an online mode, is to make corrective maintenance activities easier by assisting the maintainer in his task of finding a solution to solve a problem. A second benefit, in the long run and in an offline mode, is to prevent the emergence of recurrent defects by highlighting the reasons of their emergence.

Thus, creating knowledge by generalization or expansion [2] (also called probability estimates view and enacted salience view [3] respectively), storing it and making it available is necessary. To make it available and reusable, this knowledge has to be structured. For this, patterns are widely used. They link a problem, that we can often describe as recurrent, with a template of solution to be applied to solve it. Another kind of information can be associated to a pattern: the context in which the problem appears [4], [5], variations or minor changes that can be applied to the pattern to obtain slightly different effects [6] or indicators like the source (internal or external), the frequency, the severity [7].

The aim of this paper is to propose a framework that automatically takes advantage of information relative to defects in order to identify various defects groups that we put at the heart of corrective and preventive maintenance activities. To achieve this, this framework uses information about past defects not only to give assistance to solve a new defect but also to improve manufacturing process quality by feeding back this information from manufacturing teams to design teams. The identified groups are comparable to patterns. They describe a problem and one or more solutions are associated to them. They are not defined a priori, since past defects are analyzed to generate relevant groups. These groups can evolve with the addition of new defects (e.g. a new solution to solve a problem).

The paper is structured as follows. First, in section II, the defect resolution process is described. Section III presents the related works in our context of maintenance and defects processing systems. In section IV, the framework we propose to take advantage of information relative to defects is described. Section V is dedicated to the evaluation of the software prototype both in terms of quality and performance, scalability. Section VI concludes and discusses future work.

II. THE DEFECT RESOLUTION PROCESS

During a manufacturing process, when a variation is noticed between what is defined by the standard process and what is actually being executed, a document, which we call defect form, is instantiated to materialize this gap and to try to solve it. Whether paper or electronic, this form serves as a support for a defect. Based on industrial experience, we represent the resolution of a defect as a process initiated by a trigger (defect detection), during which data about the problem and the solution are filled in a form (Fig. 1). Once the solution has been put in place or if the gap is acceptable (no solution needed), the defect is closed. This form is composed of fields that store information about the defect.

DOI 10.1109/CEC.2011.11

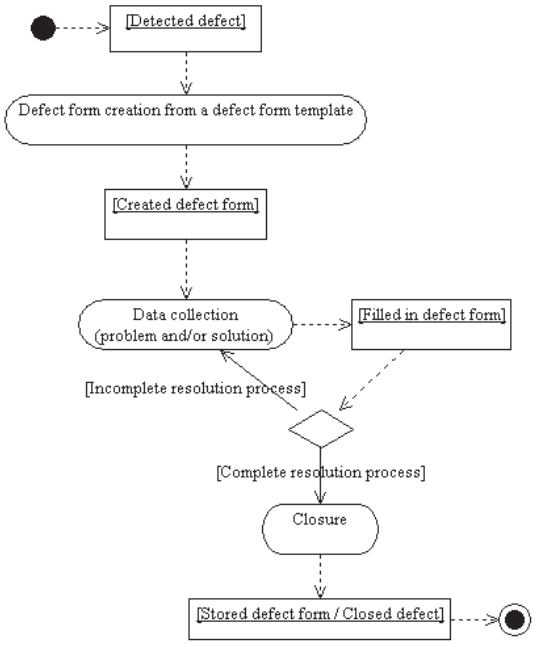

Figure 1. Our vision of the defect resolution process.

Here are some explanations about the objects involved in the activity diagram shown above:

- Detected defect: the user has to choose which defect form template should be used to solve the defect. We consider a set of templates because all defects are not solved in the same way according to their type. Several templates are defined to allow the user to choose the most appropriate one to solve the defect.
- Created defect form: it is an instance of a defect form template, data must be collected in its fields.
- Filled in defect form: defect description is filled in by several actors at different times. However, all actors have an overall vision about the defect. The defect form is not a set of fragmented elements that everyone has to fill in regardless of the other elements but a unique instance completed as the process evolves overtime.
- Stored defect form / Closed defect: defect resolution process is complete, the defect is closed, and the form that serves as a support for it is stored.

A defect resolution ends with the storage of its form. We would propose an approach that goes beyond this process, the latter being the base on which we can elaborate an effective defects processing. Without any use of the information contained in the defect forms, i.e. without processing extracting knowledge, this resolution process suffers from several issues:

- Slow process: filling in fields is a manual activity, no assistance is given to the user to guide him during the defect resolution, whether it comes to entering data about the problem or the solution. The time to propose a solution should be reduced by using experience-based knowledge from past defects. The less time-expensive or the less cost-expensive solution should be preferred to solve a problem.
- Costly process: following the previous point, the longer the defect resolution, the more expensive the process is, since it blocks the manufacturing process that depends on the resolution of the defect, and sometimes the whole production of the final product.
- No knowledge creation: no assistance to analyze past defect forms is carried out. Understanding recurrent defects should avoid their emergence by feeding back this knowledge from manufacturing teams to process design teams.

III. RELATED WORKS

A. Background

In the literature, maintenance is divided into several categories. In the industrial context, three major categories emerge: preventive (or proactive) maintenance, corrective (or reactive) maintenance and self-maintenance [8]. We can also notice the predictive and holistic (process oriented) categories [9] and the condition-based and intelligent categories [10]. In the software context, Swanson had already presented corrective, adaptive and perfective maintenances in the mid 70s [11]. The evolutionary and preventive maintenances have been added to these first categories and are still valid nowadays. We do not want to cover all these categories; we focus here on the corrective and preventive dimensions. The role of corrective maintenance is to solve a new defect's problem so that the element on which it is noted becomes again compliant with specifications. Preventive maintenance is intended to anticipate and prevent the emergence of known defects (recurrent defects management).

B. Defects Processing Systems

Regarding preventive maintenance, some analytical methods such as Failure, Mode, Effects and Criticality Analysis (FMECA) widely used in automobile [12], aerospace [13] and railway [14] sectors or Hazard Analysis and Critical Control Points (HACCP) derived from FMECA and used in chemical, pharmaceutical and food industries are used to prevent the emergence of defects. These methods, involving persons with various skills and experiences, aim at (1) searching and describing potential failures of a system from their origins (causes) to their consequences (effects), (2) quantifying the associated risks of these failures to the user thanks to a criticality indicator and (3) prioritizing corrective actions on the process definition to optimize it and maintain system reliability. Our goal is the same. We want to apply corrective actions on the manufacturing process template to prevent the emergence of defects. However, while these methods aim at manually identifying potential failures, a task that can be long and tedious, we prefer to use

the information contained in past defect forms. Grouping similar defects in an automatic way will allow defining corrections to put on the manufacturing process template.

Regarding corrective maintenance, two approaches are differentiated. The first one consists in searching relevant defect forms, similar to the new defect form in some parts, using keywords. For this, the user is prompted to enter one or more keywords. This starts a searching process of these keywords in the database containing the solved defect forms and thus selects and presents similar defect forms to the user. Using information from some of these defect forms, and in particular information about the solution used, the user can find a solution more easily and more quickly. However, this method is rather addressed to users familiar with the domain and able to provide significant and discriminatory keywords for the search. The second approach consists in organizing defect forms according to an a priori defined structure to allow the user to browse by selecting different categories. Used to process industrial defects management in aerospace, this approach was recently proposed [1] using a faceted classification [15], [16]. It allows grouping similar defects and setting up a feedback between the maintenance department and the design department that defines the templates of industrial manufacturing processes on which defects occur. Nevertheless, a major effort must be made upstream to design a fully usable classification scheme of the defect forms database. Analyzing and taking advantage of the defect forms structure would make this work easier.

To overcome the limits of these approaches (domain expert knowledge, considerable upstream work), we propose a framework that automatically classifies defect forms thanks to a preliminary analysis of defect forms structure. Thus, we can identify various defect forms groups that we put at the heart of corrective and preventive maintenance activities. In this way, we use past defects information to solve a new one but also to improve the quality of the manufacturing process by feeding back this information from manufacturing teams to design teams, feedback rarely existent within companies [17].

IV. Revealing Knowledge from Defects Forms

A. Defect Form Description

Considering nominal defects (i.e. those that can be foreseen), forms involve fields that can be qualified as regular. Indeed, all of them are distinctly named and the data a user fills in are clearly identifiable. Some of these fields define structured data, e.g. with a finite list of values and some others define unstructured data, such as free text. Considering non-nominal defects (i.e. those that cannot be foreseen), the form involves non-regular fields. These fields are, for example, comments or description of the first occurrence of a new kind of defect. They define unstructured data. Thus, a defect form is a semi-structured document that involves various fields containing structured and unstructured data. In the literature relative to the maintenance, we note a lack of such a formalization of a defect form because authors are more interested in the formalization of the maintenance process, whether in an industrial context [18], [19], [20] or in a software context [21], [22], [23], [24]. Although this process leads to make the defect resolution easier, it does not define how to process and reuse the information contained in defect forms.

The document that serves as a support for a defect contains data about the characteristics of the detected incident (i.e. the problem) and about the applied protocol to resolve it (i.e. the solution). It involves elements we call “attributes” in which data are contained. We define these attributes by two features: their structure and their descriptive quality (Table I).

The structure feature divides attributes between constrained and loose attributes. Constrained attributes contain accurate, structured data. The set of values these attributes can take is known, beforehand or not. Even if this set is not defined beforehand, the value of such an attribute is not unique among the set of defect forms and can frequently be found in other ones. Loose attributes contain unstructured data, in a textual form. Among them, we will particularly note the problem description of the defect and its solution description. The value of such an attribute is a free text. Constrained attributes can be compared to closed questions whereas loose attributes may be compared to open ones. We present this first classification in order to emphasize the need to apply a specific processing to loose attributes to extract information from them as easily as this could be done from constrained attributes.

Defect forms templates set up many attributes during the resolution process. However, all attributes do not provide the same kind of information. The descriptive quality feature divides them into three groups: the problem descriptive, the solution descriptive and the non-descriptive attributes. Thanks to this second categorization, we emphasize attributes that contain relevant data. Therefore, we give the possibility to partition attributes into:

- Problem descriptive attributes, which provide information about the encountered problem;
- Solution descriptive attributes, which explain how to solve the problem;
- Non-descriptive attributes, which do not fit into either of the two previous partitions.

An expert must decide the classification of attributes in the proposed partitions. Indeed, while the structure feature of an attribute can be automatically found, its descriptive quality feature is knowledge that must be provided by an expert who well knows the defect form template. Of course, the cost of this work increases with the number of attributes. To minimize human interaction, one can imagine that the expert is assisted in his choices by a tool giving a first classification using, for example, an ontology.

An attribute can contain data filled in by the user but also data the user did not directly fill in for the needs of

traceability. Data in these contextual attributes is related to the time the defect has been entered. An example of contextual attribute is the creation date of the defect or the person who entered it. By definition, such an attribute is constrained. But it can be problem descriptive, solution descriptive or non-descriptive. In [25], authors go even beyond our vision since they distinguish contextual defects from other ones. These defects are created in a specific context and their forms have contextual and behavioral attributes and have their own preventive processing. In our case, we do not make this distinction. We do not want to leave out potentially useful information contained in attributes, contextual or not.

B. Our Framework to Manage Defects Information

To assist the user in his solution proposal and to prevent the emergence of recurrent defects, we take advantage of information from past defect forms, whose problem has been already solved. For this, we organize the defect forms database in a scheme allowing us to meet quickly and precisely our searching and grouping objectives.

At first, as mentioned previously, all defects are not solved in the same way. Thus, it is meaningless to make comparisons between all defect forms in a database. What relevant similarity relationship may be established between a documentation defect and a defect that occurs during a drill activity? One can imagine that the drilling defect was caused by the documentation defect but it would be difficult to find similarity between their corresponding forms, between their problem and/or their solution. Therefore, we make a distinction between defect forms categories inside a unique defect forms database. These categories are defined according to various discriminatory elements.

The interest of creating defect forms categories is to have a first simple grouping in order to avoid comparing all defect forms whereas it is known that distinct populations exist. The difficulty is to identify discriminatory criteria leading to relevant categories creation. Because of the unstructured information they contain, these criteria cannot be loose attributes, it would require an important work of analysis by an expert or the use of a text processing algorithm that we want to apply only later. Therefore we have to consider only constrained attributes. As we want to facilitate the search for defect forms that are similar to a new defect form whose solution is obviously unknown, this classification must separate defect forms only according to their problem. Therefore, we are particularly interested in constrained and problem descriptive attributes. Some of them will define so characteristic values that searching similarity between defect forms that do not have the same value for one of these attributes does not seem to make sense.

Consequently, among the set of attributes, only some of them are used to define categories. Thanks to the defect description we propose, attributes that can serve as defect forms categories criteria can be filtered. A first filtering is performed through the structure feature, only constrained attributes are retained. Then, a second filtering is performed through the descriptive quality feature, only problem descriptive attributes are retained. Among the remaining attributes, whose number could be quite large in spite of the filtering step, an expert, who knows the domain well, identifies those that are really discriminatory. This work is necessarily specific to the domain, no attribute can be defined as a category criterion by default.

After that, within each category, defect forms groups, in which problems are similar, are identified. A clustering algorithm uses these comparisons to build problems groups. Then, within each problems group, defect forms groups, in which solutions are similar, are identified. In the same manner that problems groups are created, a clustering algorithm uses these comparisons to make solutions groups.

For each problems and solutions group, a prototype is defined, i.e. a defect form automatically constructed that is representative of the defect forms of the group. Thus, problem information of a problems group's prototype is used to calculate similarity in comparison with the problem information of a new defect form. Therefore, it is not necessary to compare the new defect form to all defects that belong to the problems group. With regards to solutions groups, on the one hand, solution information of the prototype is used to present a summary of a possible solution. On the other hand, to identify the solution of the problems group's that is best suited to a new problem, problem information of the prototype and of the defect form are compared.

C. The Framework Architecture

To implement this framework, two processes must be established, one offline to perform the classification of defect forms and another one online to search for similar defect forms (Fig. 2). This second activity relies on results obtained from the first one.

Regarding the classification of defect forms process, we call "defined attributes" the fact that the structure and descriptive quality features have been defined for each attribute and if the attribute serves as defect forms category criterion. Before the processing of defect forms, i.e. the similarity computation between them, keywords are extracted from attributes value. Keywords about the problem (extracted from problem descriptive attributes) and keywords about the solution (extracted from solution descriptive attributes) are retrieved. The result is a descriptor for each defect form; each defect form is represented as a couple of vectors of keywords, a problem vector and a solution vector. Clustering on problem information is performed by computing the similarity between problem vectors. In the same manner, clustering on solution information is performed by computing the similarity between solution vectors. As they have to be constructed defect forms, prototypes are also represented as a couple of problem/solution vectors.

Regarding the search for similar defect forms process, the first activities are identical to the process described above,

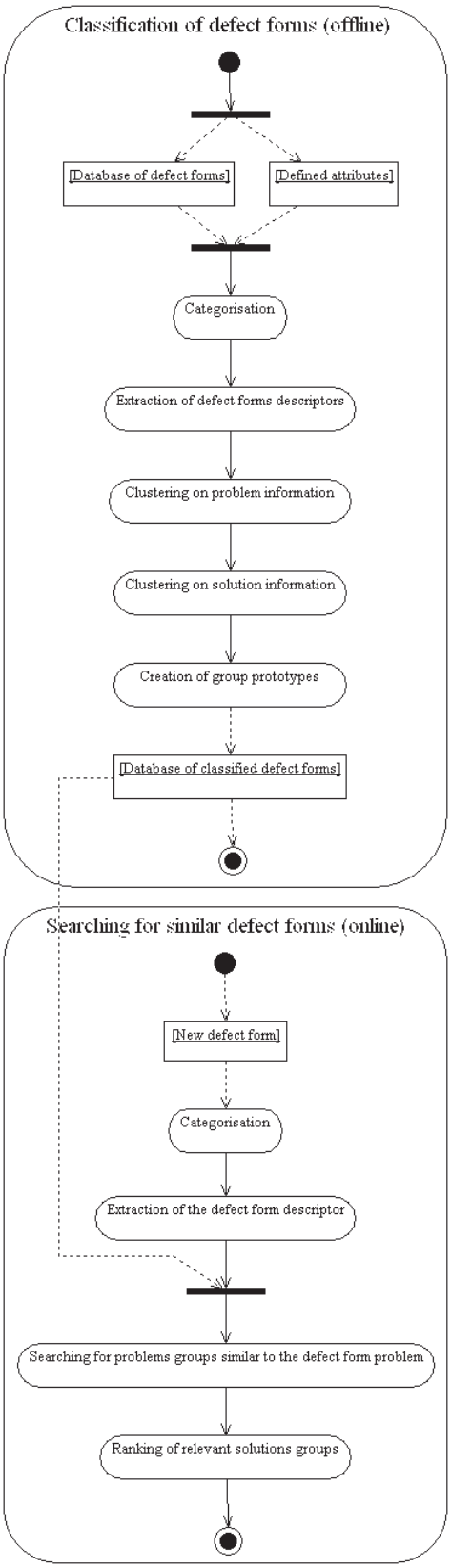

Figure 2. The framework activities.

namely the categorization of the new defect form and the extraction of its descriptor. With this information and the database of classified defect forms obtained with the process described above, the problem vector of the new defect form is compared with the problem vector of the prototype of all problems groups in the category. Those with sufficient similarity are retained. Each of these prototypes represents a problems group in which solutions groups are identified. The problem vector of each prototype of the solutions group is compared with the problem vector of the new defect form. Thus, solution groups are ranked according to their relevance between the problem they solved and the problem described in the new defect form.

The first activity enables improving preventive maintenance. Recurrent defects can easily be deduced using problems groups that have been identified. The analysis of large groups by an expert allows to understand the reasons of their emergence and to set up ways to prevent their recurrence. Production issues are capitalized and experience-based knowledge is fed back from manufacturing teams to process design teams. The second activity enables improving corrective maintenance. The framework automatically and quickly browses the problems groups to find the most similar one to the problem described in the new defect form. The solutions groups in the elected problems group that are presented to the user, are the most suited solutions to solve the new problem. Thus, the time spent by a user to process a defect is reduced. By referring to experienced solutions, every user can propose a better solution than if he had to find one using his own experience.

V. EXPERIMENTATION

This framework has been tested on a database containing more than 7000 real software defects from the bugs' database of the Intercim Company, involved in this research project. Intercim offers manufacturing execution and optimization systems for, especially in the aerospace and pharmaceutical industries. All defects relate to only one software but possibly various versions of it. They have been recorded by persons in charge of quality assurance (not final users) during more than 3 years, from Sep. 2005 to Dec. 2008.

All of the 7216 defects come from the same form template that involves 8 steps. These steps are not executed sequentially, the succession of steps is defined during the defect resolution, based on collected data. In this form template, 91 attributes have been identified. Partition according to the structure feature has revealed 20 loose attributes and 71 constrained attributes. One of the experts who defined this form template indicated the following partition according to the descriptive quality feature: 6 attributes are problem descriptive among which only one is loose. The major issue in this dataset is that there is not any attribute that describes the solution, in other words, no attribute can be defined as solution descriptive. After having used a list of stop words and applied a suffix stripping algorithm, 2680 keywords from problem descriptive

attributes have been extracted. Thanks to the definition of defect forms categories, these 2680 keywords are not used at the same time to perform the similarity computation. One of our 59 categories reaches 1098 keywords in the computation, the average being 213.

A. Implementation

Our software prototype, implemented in Java, is divided into several modules running sequentially. Once the categories are created and assigned to defect forms, keywords extraction is run. Regarding loose attributes, the text (their value) is split into words according to spaces and special characters. A list of general English stop words (containing 671 terms) and a list of field-specific stop words (poorer for the moment) are used to remove common words. Porter's suffix stripping algorithm [26], well-suited to Information Retrieval, is also applied to produce more relevant groups by considering only the root of the words. To improve similarity between defect forms, an algorithm of dimension reduction such as Singular Value Decomposition (SVD) [27], Principal Components Analysis (PCA) [28], Latent Semantic Analysis (LSA) [29] aka Latent Semantic Indexing (LSI), etc. could also be used.

Then, concerning the similarity computation, we chose to use the Vector Space Model with the TF-IDF measure (Term Frequency – Inverse Document Frequency), widespread in Information Retrieval [30], [31] and in Text Mining [32], [33]. This statistical measure evaluates the importance of a word with respect to a document from a corpus of documents and to the corpus itself. The obtained values are used to calculate the similarity between descriptors of defect forms using the cosine measure, according to the problem vector then the solution one. These similarity measures are then used in a hierarchical ascendant classification. Unlike most of the other algorithms, this clustering algorithm does not require a preliminary setting of the number of groups to obtain. It only requires the setting of a similarity threshold. In a category, all defect forms whose similarity between descriptor's problem vectors exceeds the threshold constitute a problems group. Solution groups are built in the same manner. Finally, a prototype is created for each group.

To evaluate the proposed framework both in terms of quality of the obtained clusters and in terms of performance of the hierarchical ascendant clustering algorithm, three methods of this algorithm have been implemented:

- The single-link method (aka MIN), which produces large groups potentially heterogeneous due to the chain effect it suffers (complexity: $O(n^2)$);
- The complete-link method (aka MAX), which solves the problem of the chain effect and produces smaller homogeneous groups (complexity: $O(n^2.\log(n))$);
- The incremental method which solves the problem of the quadratic complexity of the methods mentioned above but which is sensitive to the selection order of documents [34], [35], [36] (complexity: $O(n.\log(n))$).

B. Qualitative Evaluation

A study was conducted in a category containing 92 defect forms. This category possesses the advantage to have enough defect forms to obtain interesting results using our automatic approach and to have a sufficiently limited number of defect forms to compare the results with those we would accept using manual clustering. First, groupings have been manually identified, thus serving as a reference classification. This manual classification results in 70 groups.

Then, to identify the threshold value of similarity that produces the most relevant groups, different classifications have been obtained by varying the similarity threshold between 0.9 and 0.3 and compared with the reference classification. To assess the quality of classification, we present the precision-recall (Fig. 3), and the F-measure (Fig. 4) curves. The numbers shown besides the points of Fig. 3 indicate the best threshold value of similarity.

Fig. 3 shows a curve for each method of the algorithm. Each curve presents the proportion of correct groupings made by the algorithm among all the groups found (the precision) with respect to the proportion of correct groupings made by the algorithm among all the groups we would like to obtain (the recall). It shows that, whatever the method, the algorithm proposes a best classification for a similarity threshold between 0.43 and 0.5. Outside this range of values, the precision decreases, there is too much noise (i.e. too many incorrect groups), or it is the recall that decreases, there is too much silence (i.e. not enough correct groups).

In the precision-recall curve, the objective is to maximize the precision and the recall. However, it is very difficult to estimate a good similarity threshold because when the precision increases the recall decreases and inversely. The F-measure considers both the precision and the recall. Fig. 4 shows a curve for each method of the algorithm that represents a weighted average of the precision and the recall. The best similarity threshold can easily be identified at 0.45.

The three methods give very similar results. The chain effect does not seem to be very important on our dataset since the values obtained with the single-link method are close to the two other methods. Whatever the method, on our dataset, results point out relevant clusters of defect forms for a similarity threshold of 0.45. With this value, we have:

- For the single-link method, a precision and a recall of about 0.8;
- For the incremental method, a precision of 0.9 and a recall slightly lower than 0.8;
- For the complete-link method, a precision equal to 1 and a recall at 0.7;
- For all the methods, an F-measure score upper than 0.8.

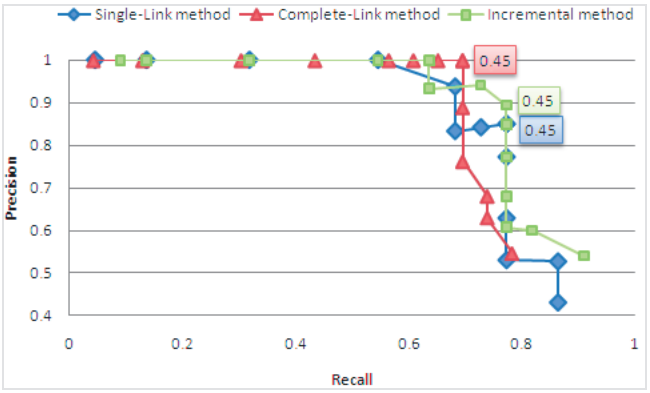

Figure 3. Precision-recall for the three methods.

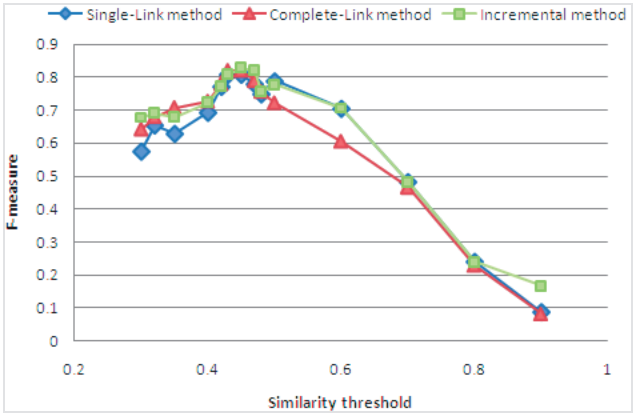

Figure 4. F-measure for the three methods.

C. *Performance and Scaling Assessment*

To evaluate the performance of the algorithm, classifications have been executed on defect forms categories of various sizes (from 17 to 1232 defect forms) to identify problems groups. The similarity threshold has been set to 0.45 since the best classification is obtained in every method with this value. The execution time represents the time spent by the methods of the algorithm to identify groups and the time to create each group prototype. Since the incremental method necessarily creates all prototypes for its own execution, this second time is included in the first one.

As expected, the incremental method is really faster than non-incremental ones even if the time to create group prototypes is not included (Fig. 5). Considering the category containing 1005 defect forms, the single-link method is 60% and the complete-link method is 85% more time consuming than the incremental method. When this time is included, the gap widens (Fig. 6). Considering the same category as previously, the single-link method is 5.5 times and the complete-link method is 9 times more time consuming than the incremental method. The execution time depends on the number of defect forms but also on the number of involved keywords, that explains the slight decrease of time around the category containing 950 defect forms.

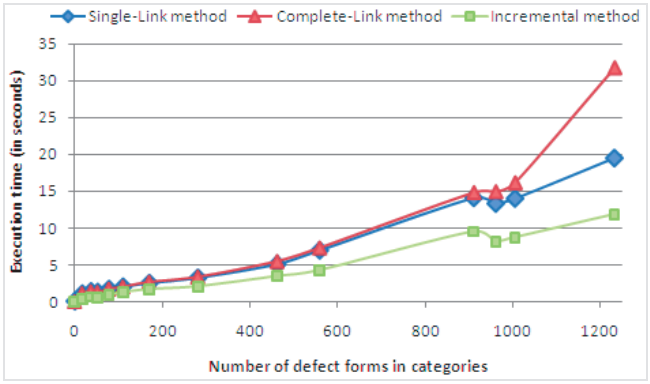

Figure 5. Execution time without prototypes creation.

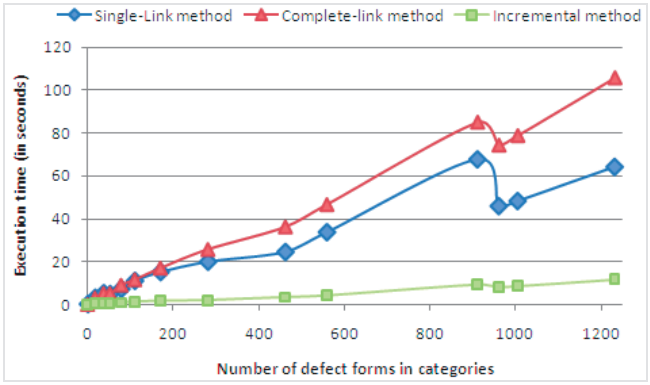

Figure 6. Execution time with prototypes creation.

Moreover, it is logical to see that the complete-link method is slower than the single-link one because it needs more comparisons between defect form descriptors during the potential addition of a defect form in a group (Fig. 5) and it creates more groups, which requires the creation of more group prototypes (Fig. 6).

Thus, the incremental method seems to be able to process a large volume of data within a reasonable processing time while preserving a good quality of clustering. The creation of group prototypes, required in the framework we propose, is inherent to this method, it does not add a new step contrary to the non-incremental methods.

VI. CONCLUSION AND PERSPECTIVES

We have presented a framework for the processing and the reuse of information contained in past defect forms. It reveals knowledge in order to make corrective maintenance activities easier by assisting the user in his search for a solution. It is suited to preventive maintenance since an analysis can be made on each problems group in order to identify their causes of emergence.

To assess the interest of our framework, we plan to do other tests on a larger database containing around 75000 industrial manufacturing defect forms. These defect forms should have problem descriptive attributes but also solution descriptive ones. Moreover, in this context, tasks are

repetitive (more than in software development). We should therefore find more recurrent defects since a kind of defect may be repeated on numerous products.

Eventually, we could take into account the user's profile in the solution ranking. By considering his skills on one hand, and the way he is used to executing a kind of solution to solve a problem on the other hand, we could re-rank the list of proposed solutions to be adapted to his profile. A solution he usually fails to execute could be depreciated in the list.

REFERENCES

[1] Y. M. Goh, M. D. Giess, C. A. McMahon, and Y. Liu, "From Faceted Classification to Knowledge Discovery of Semi-Structured Text Records," Foundations of Computational Intelligence Volume 6, Studies in Computational Intelligence, Springer, Heidelberg, vol. 206, pp. 151–169, 2009.

[2] J. W. Clark, "Exceptions and Other Rare and Irregular Events: Two Modes of Learning in Business Intelligence (research in progress)," Hawaii International Conference on System Sciences, pp. 1–10, 2011.

[3] J. Lampel, J. Shamsie, and Z. Shapira, "Experiencing the Improbable: Rare Events and Organizational Learning," Organization Science, vol. 20(5), pp. 835–845, 2009.

[4] A. Persson, and J. Stirna, "How to Transfer a Knowledge Management Approach to an Organization: A Set of Patterns and Anti-Patterns," International Conference on Practical Aspects of Knowledge Management, LNCS, vol. 4333, pp. 243–252, 2006.

[5] D. May, and P. Taylor, "Knowledge Management with Patterns," Communication of the ACM, vol. 46(7), pp. 94–99, 2003.

[6] B. S. Lerner, S. Christov, L. J. Osterweil, R. Bendraou, U. Kannengiesser, and A. Wise, "Exception Handling Pattern for Process Modeling," IEEE Transactions on Software Engineering, vol. 36(2), pp. 162–183, 2010.

[7] Y. Somekh, M. Peleg, and D. Dori, "Classifying and Modeling Exceptions through Object Process Methodology," Proceedings of the International Conference on Systems Engineering and Modeling, pp. 60–70, 2007.

[8] K. Komonen, "A Cost Model of Industrial Maintenance for Profitability Analysis and Benchmarking," International Journal of Production Economics, Elsevier, vol. 79(1), pp. 15–31, 2002.

[9] I. Alsyouf, "The Role of Maintenance in Improving Companies' Productivity and Profitability," International Journal of Production Economics, Elsevier, vol. 105(1), pp. 70–78, 2007.

[10] D. Dowlatshahi, "The Role of Industrial Maintenance in the Maquiladora Industry: an Empirical Analysis," International Journal of Production Economics, Elsevier, vol. 114(1), pp. 298–307, 2008.

[11] B. Swanson, "The Dimensions of Maintenance," IEEE Computer Society Press, International Conference on Software Engineering, pp. 492–497, 1976.

[12] J. B. Bowles, "An Assessment of RPN Prioritization in a Failure Modes Effects and Criticality Analysis," Reliability and Maintainability Symposium, pp. 380–386, 2003.

[13] E. Balaban, P. Bansal, P. Stoelting, A. Saxena, K. F. Goebel, and S. Curran, "A Diagnostic Approach for Electro-Mechanical Actuators in Aerospace Systems," IEEE Aerospace Conference, pp. 1–13, 2009.

[14] Y.-H. Li, Y.-D. Wang, and W.-Z. Zhao, "Bogie Failure Mode Analysis for Railway Freight Car based on FMECA," International Conference on Reliability, Maintainability and Safety, pp. 5–8, 2009.

[15] R. Prieto-Diaz, and P. Freeman, "Classifying Software for Reusability," IEEE Software, vol. 4(1), pp. 6–16, 1987.

[16] M. D. Giess, P. J. Wild, and C. A. McMahon, "The Generation of Faceted Classification Schemes for Use in the Organisation of Engineering Design Documents," International Journal of Information Management, Elsevier, vol. 28(5), pp. 379–390, 2008.

[17] E. Levner, D. Zuckerman, and G. Meirovich, "Total Quality Management of a Production-Maintenance System: A Network Approach," International Journal of Production Economics, Elsevier, vol. 56-57(1), pp. 407–421, 1998.

[18] G. Waeyenbergh, and L. Pintelon, "Maintenance Concept Development: a Case Study," International Journal of Production Economics, Elsevier, vol. 89(3), pp. 395–405, 2004.

[19] A. Despujols, "Approche Fonctionnelle de la Maintenance," Techniques de l'Ingénieur, AG4710, pp. 1–14, 2004.

[20] I. P. S. Ahuja, and J. S. Khamba, "Total Productive Maintenance: Literature Review and Directions," International Journal of Quality & Reliability Management, Emerald, vol. 25(7), pp. 709–756, 2008.

[21] M. Haziza, J. F. Voidrot, E. Minor, L. Pofelski, and S. Blazy, "Software Maintenance: an Analysis of Industrial Needs and Constraints," Conference on Software Maintenance, pp. 18–26, 1992.

[22] P.-Y. Lambolez, "Recherche d'Informations pour la Maintenance Logicielle," PhD thesis, Université Paul Sabatier, Toulouse 3, 1994.

[23] S. Barros, "Analyse a priori des Conséquences de la Modification de Systèmes Logiciels: de la Théorie à la Pratique," PhD thesis, Université Paul Sabatier, Toulouse 3, 1997.

[24] I. Alloui, "Conciliating Property Stability and System Evolution through Software Model Analysis," GDR Génie de la Programmation Logicielle 2009, pp. 224–231, 2009.

[25] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Computing Surveys, vol. 41(3), pp. 1–58, 2009.

[26] M. F. Porter, "An Algorithm for Suffix Stripping," Program: Electronic Library and Information Systems, vol. 40(3), pp. 211–218, 2006.

[27] G. W. Stewart, "On the Early History of Singular Value Decomposition," SIAM Review, Society for Industrial and Applied Mathematics, vol. 35(4), pp. 551–566, 1993.

[28] H. Abdi, and L. J. Williams, "Principal Components Analysis," Wiley Interdisciplinary Reviews: Computational Statistics, vol. 2(4), pp. 433–459, 2010.

[29] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," Journal of the American Society for Information Science, Wiley, vol. 41(6), pp. 391–407, 1990.

[30] G. Salton, and M. J. McGill, "Introduction to Modern Information Retrieval," McGraw-Hill, New York, 1983.

[31] G. Salton, and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing and Management, Elsevier, vol. 24(5), pp. 513–523, 1988.

[32] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman,Y. Schler, and O. Zamir, "Text Mining at the Term Level," PKDD, Second European Symposium. LNCS, Springer, Heidelberg vol. 1510, pp. 65–73, 1998.

[33] I. T. Fatudimu, A. G. Musa, C. K. Ayo, and A. B. Sofoluwe, "Knowledge Discovery in Online Repositories: A Text Mining Approach," European Journal of Scientific Research, EuroJournals, vol. 22(2), pp. 241–250, 2008.

[34] I. Gurrutxaga, O. Arbelaitz, J. I. Martín, J. Muguerza, J. M. Pérez, and I. Perona, "SIHC: A Stable Incremental Hierarchical Clustering Algorithm," International Conference on Enterprise Information Systems, pp. 300–304, 2009.

[35] B. Raskutti, and C. Leckie, "An Evaluation of Criteria for Measuring the Quality of Clusters," Joint Conference on Artificial Intelligence, pp. 905–910, 1999.

[36] Q. H. Nguyen, and V. J. Rayward-Smith, "Internal Quality Measures for Clustering in Metric Spaces," International Journal Business Intelligence and Data Mining, Inderscience, vol. 3(1), pp. 4–29, 2008.