Usage of case-based reasoning in FMEA-driven software

Gabriela Cândeaa,b, Stefania Kifora, Carmen Constantinescuc

a Universitatea Lucian Blaga Sibiu, Bd-ul. Victoriei, Nr.10, Sibiu, 550024, România
b R&D Department, ROPARDO SRL, Reconstrucţiei 2A, 550129, Sibiu, România
c Fraunhofer Institute for Industrial Engineering (IAO), 70569 Stuttgart, Germany

* Corresponding author. Tel.: +40-269-231037; fax: +40-269-231037. E-mail address: gabriela.candea@ropardo.ro

Abstract

Failure Mode and Effect Analysis (FMEA) is among the most widely used safety analysis procedures in the various industries. The procedure is generally perceived as complex and time-consuming, hindering an effective reuse of previous knowledge. In this paper we present an innovative usage of knowledge system into FMEA process using the Case-based reasoning to reduce the time and effort associated with this analysis. Knowledge system is built to serve multi-projects work that nowadays are in place in any manufacturing or services provider, and knowledge must be retained and reused at the company level and not only at project level. Collaboration is assured through web-based GUI that supports multiple users access at any time. Initial results confirm the viability of this system for industrial application.

Keywords: FMEA; Case-based reasoning; knowledge reuse;

1. Introduction

Failure Mode and Effect Analysis (FMEA) is a powerful and documented method used to define, identify and eliminate known and/or potential failures, problems and errors from the system, design, process and/or service before they reach the customer, even before they reach the mass production.

Although the purpose, terminology and other details can vary according to type (e.g. Process FMEA - PFMEA, Design FMEA - DFMEA, System FMEA, Product FMEA, FMECA, etc.), the main objective of this methodology is to allow the analysts to identify and prevent known and potential problems before they occur. For years, FMEA/FMECA (failure mode, effects and criticality analysis) has been an integral part of engineering designs and respond at three main questions [11]: what might go wrong? What might cause it to wrong? And what effect would it have?

There are a number of published guidelines and standards for the requirements and recommended reporting format of failure mode and effects analyses. Some of the main published standards for this type of analysis include SAE J1739, AIAG FMEA-4 and MIL-STD-1629A. In addition, many industries and companies have developed their own procedures to meet the specific requirements of their products/processes.

A good FMEA can help analysts identify known and potential failure modes and their causes and effects, assess the risk associated with those failure modes, prioritize the identified failure modes and identify and carry out corrective actions. The priority of a failure mode is determined through the risk priority number (RPN), which is defined as the product of the occurrence (O), severity (S) and detection (D) (see Figure 1).

FMEA has been proven to be one of the most important early preventative initiatives during the design stage of a system, product, process or service. However, the big number of experienced people (from different departments) requested, the hardly reusable analyses because of the natural language, the unavailability (de-located team, overlap of membership between the teams) of team members represents the red line for the presented research.
The paper is organized as follows. In Section 2, we survey the FMEA process challenges, how were found in the industry (automotive sector). Section 3 describes the architecture of the FMEA and Experience Database software systems followed by in deep Experience Database description while the conclusions are presented in Section 4.

2. FMEA’s shortcomings

Today, FMEA is in widespread use by a multitude of industries, many of which have begun imposing FMEA standards. FMEA is a living document that facilitates the inter-departmental dialog and helps to prevent the error and not to react to them. The purpose of the FMEA is to take actions to eliminate or reduce failures, starting with the highest-priority ones [15]. But the effort to develop an FMEA is mainly considered as high or very high due to the number of involved persons [14]. In addition, the advantages that result out of failure prevention cannot be perceived immediately. To shorten the process of FMEA development and earning results, the knowledge included in already developed FMEA has to be reused and the first step is to capitalize the knowledge. The capitalization will not just shorten the FMEA process but will prevent that valuable job-related information will leave the company together with the employee.

The FMEA knowledge reuse endures from major drawbacks mentioned by Wirth et al. [16]: the FMEA-related information is acquired in natural language and is not much reusable because the systematized components, functions and failure modes are not made explicit. The meaning depends on the interpretation of the team/ a team member who performs the FMEA and can fluctuate when another team reuses this FMEA, or even if the same team tries to reuse it on a later occasion.

Although one person is in charge for coordinating the FMEA process, all FMEAs are team based. The scope for a FMEA team is to gather a range of perspectives and experiences in the project. Because each FMEA is unique in dealing with different aspects of the product or process (production, engineering, logistic, marketing, support), FMEA teams are formed and dispersed when is needed. Based on this variety of requested people another short-coming is sets by the unavailability (de-located team, overlap of membership between the teams) of team members to attend at FMEA meeting.

3. Software systems

In this chapter the FMEA-driven software system is presented from concept to architecture. The system is composed by two major subsystems – FMEA-driven software and Experience Database.

FMEA-driven software is respecting all FMEA requirements and processes of work and is web based software that allows team collaborative work on FMEA.

Second major system is Experience Database (Knowledge Repository System) that provides knowledge capitalization. In current implementation Experience Database uses for capitalization of knowledge a case base reasoning (CBR) approach.

3.1. FMEA-driven software

The purpose of FMEA-driven software is preventing process and production problems before they occur. It is used both in design and manufacturing processes and it substantially reduces costs by identifying product and process improvements early in the development process when changes are relatively inexpensive to implement.

The software (based on FMEA) processes are based on worksheets that contain important information about the device, such as the revision date, the team involved, the coordinator of FMEA process, the start date of the FMEA process, and the number of revisions. On these worksheets all the items/components or functions of the subject should be listed in a logical manner. For each piece / part / process / function / operation / feature are identified the possible failure modes, effect and causes and each of them are graded for their severity (S), frequency of occurrence (O), and detection rating (D). Afterwards, the Risk Priority Number (RPN) is calculated based on mathematical product of the three above factors S, O and D. Once this is done it is easy to determine the areas of greatest concern. This has to be done for the entire process and/or design and the items that have the highest RPN should be given the highest priority for corrective action. After these values are allocated, recommended actions with targets, responsibility and specifics deadlines of implementation are noted on the worksheets which actually consist in the output of this module.

3.2. Experience database

The experience database is proposed to provide an easy to use component by the knowledge engineer and by other software modules.

A knowledge management system it faces on few major challenges: 1) Acquisition – The main target here is to get hold of the information that is around, and turn it into knowledge by making it usable. This might involve, making
Tacit knowledge explicit, identifying gaps in the knowledge already held, acquiring and integrating knowledge from multiple sources. 2) Modelling – Knowledge model structures must be able to represent knowledge so that it can be used for problem-solving. One important knowledge modelling idea is that of ontology’s, which are specifications of the generic concepts, attributes, relations and axioms of a knowledge base or domain. Ontology’s can act as placeholders and organizing structures for acquired knowledge, while also providing a format for understanding how knowledge will be used. 3) Retrieval – When a knowledge repository gets very large, finding a particular piece of knowledge can become very difficult. 4) Reuse – On problem in using knowledge management systems is that often knowledge databases are rebuilt for each end user. 5) Publishing – can be described like: knowledge, in the right form, in the right place, to the right person, at the right time. 6) Maintenance – It may involve the regular updating of content as content changes. But it may also involve a deeper analysis of the knowledge content.

The Experience Database component that is the main architecture is described in Figure 2 is using Case-based Reasoning as computational engine. Actual design of Experience Database allows defining and storing different types of structures for knowledge representation. These structures can be defined using ontology editor that allow you to keep an organized and easy way to access and view of the database. In our case we are using Protégé as ontology editor and defined ontology is stored using Protégé internal storage. We built a mapping tool that allows exporting certain structure from ontology to the Experience Database as showed in Figure 2. Once that structure is exported to CBR engine, we will operate two atomic structures, one original that is defined inside of ontology and other one, inside of CBR system. In this way ontology can evolve and new case structures can be created any time, on the other side – CBR – once that case is created and populated with data, this structure is fixed and structure can be modified only manual – no automatic update process.

The communication system assures the independence of the module core processing model from the communication methods. The default implementation it is the direct Java calls: the client will get a communication object which expose the methods through a Java interface. The methods are invoked by direct calling, all the data types being passed without transformation (into/from XML or similar); other methods are exposed by Experience Database and can be used as well.

One important sub-system of Experience Database is the input/output (I/O) and validation that is responsible for the translation of the data from external sources into native data types which can be used by the controller.

The I/O system is split into two subsystems, one for input and one for output:

- The input subsystem does the translation of information from generic formats (XML structures) into Java formats (POJO – Plain Old Java Objects). This is done by validating and parsing the XML input into the corresponding POJO. The validation is done against the XSD and it is different from the validation done into the validation system – it consists only in checking the syntax of XML is correct.
- The output subsystem generates the XML answers from the Java objects (it is mainly a serialization of Java objects into the corresponding XML representation, but additional transformations may apply).

The validation system checks the incoming data for inconsistencies and rejects the wrong ones.

Input Data Parser – it is responsible with the parsing of the input (request) information. The input data requests are for similar cases (a search over the stored cases using some filtering parameters) or request for a single case (identified by its ID).

Feedback Data Parser – it is responsible with the parsing of the feedback data. The feedback consists in changes to a stored case (different solution, etc.).

Ontology/Mapping Parser – it is responsible with the parsing of the domain/case ontology and with the parsing of the mapping information that will be used by the controller to solve the problem. The mapping information is domain dependent and will be defined by the knowledge engineer. The default implementation will provide some default mappings but other will be needed to be defined.

Ontology Definition Sender – it is responsible for the formatting of the ontology definition from the internal format into the XML file. The sender is invoked by the controller upon a corresponding request is received. And the ontology is fetched from the database (please see relevant sequence charts).

Output Data Sender – it is responsible for the formatting of the retrieved cases/answers into the correct XML structures.

Input Validator – need to validate the parsed input data for inconsistencies.

Feedback Validator – needs to validate the feedback information for inconsistencies.

Ontology/Mapping Validator – needs to validate the ontology and domain mapping information for inconsistencies.

All the validation is done in order to lighten the controller processing (the controller receives only good information; the wrong input will be filtered before).

The standard invocation process starts a search in experience databases – search that is done after similarity
functions (detailed description on next chapter) that are defined for each data structure. Search is done separate for each data structure defined in separate spaces (case space) in its database for a better case management. To start a new search an XML containing the case pattern data is sent to Experience Database and, as response, an XML with the best ‘n’ cases is returned. For the feedback phase FMEA-driven software will send an XML with feedback data for the specific case pattern based on algorithms that Experience Database “learns”.

In our implementation of Experience Database, a Case Base Reasoning engine is the core computational engine that solves problems by adapting solutions to older ones.

A CBR system involves reasoning from prior examples, memorizing previous problems and associated solutions and solving new problems by referencing to that knowledge [10]. The problem-solving life cycle in CBR system consists essentially of the following four parts as in Figure 3.

- Retrieve most similar cases from previous experience (memory)
- Reuse the information and knowledge learned from past cases and solve the new problem
- Revise by evaluating the generated solution
- Retain the new found solution for future problem solving (optional step).

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing knowledge-base (case-base).

Fig. 3. CBR internal design.

**Case structure**

The development was started from a general case structure definition and contains the following information.

- ID. A case base unique identification number.
- Description. A brief description for the case.
- Meta-data. The case meta-data is maintained for each case.
- Creator. Name of person/organization/project that created the case.
- Creation date/time. Date and time the case was initially saved in the case base.
- Number of times accessed. Count of the number of times the case has been retrieved from the case base by a client.
- Date/time of last access. The date/time of the last time the case was retrieved.
- Features. A list of case features. A case feature is synonymous with a case index.
- Data or Subcases. This is also commonly referred to as the case solution. The case data (solution) contains the information that is returned to the client during case retrieval. If a case has child cases no data is associated with the parent case. For these aggregate cases a list of child cases is maintained.

![Fig. 4.a) FMEA case representation – the problem](image)

Fig. 4.a) FMEA case representation – the problem

![Fig. 4.b) FMEA case representation – the solution](image)

Fig. 4.b) FMEA case representation – the solution

Starting from this general definition we defined for FMEA-driven software a specific case schema and this one respect our scope for knowledge capitalization on multi project and multi user usage.

The starting point was FMEA domain specific representation (Figure 4) and the case structure was built with
information about process/product as well effects and measures that must be taken for each case.

For example we considered the next case structure:

FMEA Case Problem composed by:
PN – Project Name
P – Product /Process
O – Operation/Step
PD – Potential Defect
PE – Potential Effect
PC – Potential Cause
PM – Prevent measures
DM – Detect measures
G – The grade for gravity
F – The grade for frequency
D – The grade for frequency
C – The grade for risk priority number

FMEA Case Solution is composed by:
R – Remedy solution proposed
RE – Responsible for remedy
NG – The new grade for gravity
NF – The new grade for frequency
ND – The new grade for frequency
NC – The new grade for risk priority number

For each case from the database is calculated the degree of similarity equation-1, between qc and cc; i=1 to n; where n is the total number of cases in the database.

\[ SM(qc,cc) = \frac{common}{common+different} \]  

Where “common” represent the number of feature whose value is the same between qc and cc, and “different” represents the number of features whose value is different between qc and cc.

In current implementation we are implementing a similarity function that is based on the Euclidian weighted distance equation-2. The distance is calculated as the square root of the sum of the squares of the arithmetical differences between the corresponding coordinates of two objects [10].

\[ d_{pq}^w = \sqrt{\sum_{j=1}^{n} w_j \rho_{pq}^j (e_{pj} - e_{qj})^2} \]  

Where w is the weight of the associated j the feature to indicate the importance of that feature \( w_j \in [0,1] \).

For distance measure computation we used next formulas.

- \( \rho_{ij}(a,b) = |a-b| \) if a and b are real numbers
- \( \rho_{ij}(A,B) = \max_{a \in A, b \in B} |a-b| \) if A and B are intervals
- \( \rho_{ij}(a,b) = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases} \) if a and b are symbols

A most advanced solution was also implemented for similarity calculations. Latent semantic indexing is a method for automatic indexing and retrieval, is useful in situation where traditional lexical informational retrieval approach fail. This method consisting in taking advantage of implicit higher-order structure in the association of terms with documents in order to improve the detection of relevant documents on the basis of terms found in queries.

Latent Semantic Indexing (LSI) uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis, to model the associative relationships.

In our implementation the documents are represented by the attributes of the fields: potential defect, potential effect, potential causes, verification measures to prevent, verification measures to detect and suggested remedies as required. The steps of LSI algorithm applied in our FMEA driven software are:

- prepare the terms vector (extract the key words from the all documents)
- from documents eliminate the stop-words, stems terms and calculates statistics about the frequency of terms
- construct the term-document matrix
• normalize the term-document matrix

• singular value decomposition of the normalized term-document matrix

Our term-document matrix is built using the binary weighting:

\[
a_{ij} = \begin{cases} 1, & \text{if keyword } i \text{ occurs in document } j \\ 0, & \text{otherwise} \end{cases}
\]

where \(a_{ij}\) represent the degree of relationship between term \(i\) and document \(j\).

The binary weighting informs about the fact that a term is somehow related to a document but carries no information on the strength of the relationship.

Singular value decomposition of the length normalized term-document matrix means that the matrix \(A\) is factored into the product of 3 matrices (3)

\[
A = UV^T
\]

The SVD derives the latent semantic structure model from the orthogonal matrices \(U\) and \(V\) containing left and right singular vectors of \(A\), respectively, and the diagonal matrix, \(S\), of singular values of \(A\).

The truncated SVD, in one sense, captures most of the important underlying structure in the association of terms and documents, yet at the same time removes the noise or variability in word usage that plagues word-based retrieval methods.

4. Tests

The tests were executed in the COMPA factory and the length of the tests was 2 months. During these 2 months the tests consist in using the FMEA-driven software tool by all the teams involved in FMEA processes. The tests were focused to improve the time needed for FMEA process and also to improve the utilization of knowledge from past FMEA’s but also to reuse the know-how of the engineers, but also the know-how of the support team, marketing team, and purchase team. The know-how that is used in FMEA processes are not just the knowledge in materials and engineering area, but are also knowledge from logistics, from marketing and maintenance area.

The first results responded to our question: what is the best similarity function that can be used in FMEA-driven software. If the searches are focused on one word, the best results (in time) are obtained if we use the Euclidean distance for the similarity. But if the searches contains more than one word than is better to use the LSI algorithm (modified - we do not use the thesaurus).

In the next 4 months the tests will be concentrated on evaluate the quantity of knowledge that was capitalized in the FMEA-driven system, the quality of this knowledge stored, but also where and how is reused. Other tests will run in parallel with this because there is needed to check if the knowledge that was not used/reused is not relevant, or is incomplete or other causes. Of course there will be consider also the knowledge that is also partially reused (modified).

The completed tests results will be presented in the summer of 2014, after all the tests were finalized.

5. Conclusions

In current implementation we are proposing a method to mobilize the professional knowledge of those involved professionals into FMEA process. Now days in manufacturing sector decisions concerning processes and products must be anticipated by integrating the professional knowledge and know-how of experts from early stages to motorization and correction. Different aspects where investigated from artificial intelligence [3], Case-Base Reasoning [6] and knowledge management within knowledge capitalization.

In this paper is proposed an innovative method that allows knowledge capitalization for FMEA process. Moreover, presents the designed and built the software system that on one hand get a new approach for standard FMEA – collaborative on multi user, multi project using web GUI – FMEA-driven software; and on the other one we put together the Experience Database with the FMEA specific knowledge capitalization.

As a core computational engine for Experience Database it’s used Case Base Reasoning engine and for similarity function were implemented Euclidian weighted distance but also the Latent Semantic Indexing algorithm.

The software system presented in this article is launched in production to the biggest automotive spare parts supplier from Romania, starting from Q4 2013.

As future work we must investigate different similarity functions and we are looking to implement and evaluate fuzzy approach. Other task that must be carrying is the maintenance of the CBR system that over time current configuration may become sub-optimal, and therefore is critical to have the ability to optimize the configuration.

Acknowledgements

The research reported in this paper has been supported by project POSDRU/ CPP107/DM1.5/S/76851, co-financed by the European Social Found through Sectorial Operational Programme Human Resources Development 2007-2013.

References

[1] Barthes, J.P. (1996). ISMICK and Knowledge Management, Proceedings of the 4th International Symposium on the Management of Industrial and Corporate Knowledge (ISMICK’96), pp. 9-13, 1996.

[2] Cândea, C., Georgescu, A.V., Cândea, G. (2009). iPortal - Management Framework for Mobile Business. Proceedings of the International Conference on Manufacturing Science and Education MSE.

[3] Filip, F.G. (2007). Decision Support Systems. 2nd edition revised pp. 353 – 359. Ed. Tehnica, Bucuresti.

[4] Georgescu, A.V., Cândea, C., Constantin, Z.B. (2007). iGDSS - Software Framework for Group Decision Support Systems. In Proceedings of The Good, The Bad and The Unexpected.

[5] Goddard, P.L. (2000). Software FMEA techniques. Reliability and Maintainability Symposium. pp 118-123.

[6] Haque, B.U. et al. (2000). Towards the Application of Case Reasoning to Decision-making in Concurrent Product Development (Concurrent Engineering), School of Mechanical, Manufacturing Engineering and
Management, pp. 101–112, University of Nottingham, Nottingham, NG7 2RD, UK.

[7] Huang, G.Q., Shi, J., Mak K.L. (2000). Failure Mode and Effect Analysis (FMEA) Over the WWW. The International Journal of Advanced Manufacturing Technology, Vol. 16 (8), pp. 603-608.

[8] McDermott, R.E., Mikulak, R.J., Beauregard, M.R. (2008). The basics of FMEA. Productivity Press, USA

[9] Puente, J., Pino, R., Priore, P., Fuente, D. (2002). A decision support system for applying failure mode and effects analysis. International Journal of Quality & Reliability Management. Vol 19 (2), pp 137-150.

[10] Sankar K. Pal, Simon C.K. Shiu, (2004). Foundation of Soft Case-Based Reasoning

[11] Sankar NR, Prabhu B S. Modified approach for prioritization of failures in a system failure mode and effects analysis. International Journal of Quality & Reliability Management 2001; 18(3):324–35

[12] Schreinemakers, J.F. Knowledge Management: Organization, Competence and Methodology, Proceeding of ISMICK’96, pp. 9–13, 21–22 Oct, Rotterdam, Wuerzburg, Ergon Verlag.

[13] Stamatis, D.H. (2003). Failure mode and effect analysis: FMEA from theory to execution, William A. Tony, USA

[14] Stock, M., Stone, R. and Turner, I. Y. (2003). “Going Back in Time to Improve Design: The Function-Failure Design Method”, Submitted to Proceedings of the 2003 ASME Design Engineering Technical Conference, Design Theory and Methodology Conference, Chicago, IL.

[15] Tague, N.R. (2004). The Quality Toolbox, Edition ASQ Quality Press, pp 236-240.

[16] Wirth, R., & Berthold, B., & Krämer, A., & Peter, G. (1996). Knowledge-Based Support of System Analysis for Failure Mode and Effects Analysis. Engineering Applications of Artificial Intelligence, 9, 219-229.

[17] http://www.aiag.org/scriptcontent/index.cfm, Automotive Industry Action Group