An ontology-based knowledge management approach supporting simulation-aided design for car crash simulation in the development phase

Naouress Fatfouta, Julie Stal Le-Cardinal

Title: An ontology-based knowledge management approach supporting simulation-aided design for car crash simulation in the development phase

Authors:

Naouress Fatfouta[1,2], Julie Stal le-Cardinal[1]

[1]Université Paris-Saclay, CentraleSupélec, Laboratoire Génie Industriel, 91190, Gif-sur-Yvette, France

[2]Renault SAS, 78280, Guyancourt, France

Corresponding author: Naouress Fatfouta,

e-mail address: naouress.fatfouta@centralesupelec.fr

Title: An application of an ontology-based knowledge management approach to support simulation-aided design to car crash simulation in the development phase

Abstract

In the automotive industry, the design process is both costly and time-consuming. This research focuses on improving the design process by mainly reducing time while producing more robust vehicles. Vehicle development is based on simulation; thus, the design process is referred to as simulation-aided design. Engineering design is highly collaborative and knowledge-intensive. Therefore, knowledge management plays a crucial role in today's global economy and is essential for the competitiveness of companies. However, current research on engineering knowledge management focuses on either the codification or the personalisation approaches of knowledge management. Thus, this paper addresses an integrated and collaborative approach. This paper aims to develop an ontology-based knowledge management system to support simulation-aided design, specifically car crash simulation. The knowledge management support system is designed to ensure the capture and retrieval of engineering knowledge and to enable collaboration between different stakeholders. An evaluation of the models and technologies used is also undertaken, based on use case scenarios.

Key words: knowledge management, ontology, knowledge model, knowledge retrieval, collaboration, engineering design, crash simulation

1. Introduction

Design involves people with the appropriate expertise undertaking a process to develop a product. This activity takes place within a specific organisation that provides the necessary infrastructure and resources [1]. Moreover, design tasks in all industries, including the automotive industry, are now more complex, with increasing timescales and geographically distributed teams [2]. This research focuses on the development phase of the vehicle design process. At the end of the development phase, the vehicle model is guaranteed to be at the right level of performance and the right manufacturing cost. Development is based both on modelling and simulation, and decision-making is mainly based on simulation. We are, therefore, in the context of simulation-aided design. This research focuses on improving the vehicle design process by mainly reducing time while producing more robust vehicles. The objective is, therefore, to support simulation-aided design. We focus on car crash simulation as it is an essential step in vehicle development [3]. Car safety is a crucial factor for car manufacturers since road accidents were the ninth leading cause of disease in the world in 2016 [4] and are expected to be the third by 2020 [5]. The car crash simulation is expensive, time-consuming and requires considerable effort.

Engineering design process is a knowledge-intensive and collaborative task [1], [6], [7]. It requires multiple designers and experts to conduct multiple tasks involving different areas of knowledge and expertise [8]. Cooperative communication among teams is complicated [7]. Collaborative engineering design is evolving towards a problem-solving task [7] that embodies a significant level of complexity; more than that which is required for a single engineer to work on such problems [6]. Engineering design is heavily informational [9], given that engineering designers spend considerable time locating information in human and nonhuman sources [1], [8], [9]; they engage 55.75% of their time in informational behaviour, such as processing, communicating and disseminating information [10]. They spend less time locating the source when it is a human source than when it is a non-human source, and so it is more efficient to use people as sources of information [10]. However, due to the current transient nature of modern industrial organisations, experienced designers and experts are not going to be available to consult in the future [1]; companies suffer significant setbacks from staff loss, such as the deterioration of their relationships with customers or suppliers and the loss of revenue [11].

The multidisciplinary, highly collaborative and contextual nature of engineering design has raised the need to support integrated and collaborative product development [9]. Successful collaborative engineering design depends on the ability to manage and share engineering knowledge [7]. Therefore, nowadays, knowledge management plays a crucial role in the global economy and it is important for the competitiveness of companies [12]; knowledge management is recognised in both academic and industrial studies [2].

In this paper, we propose an ontology-based approach for integrated and collaborative knowledge management. Specifically, we develop a knowledge management system to support simulation-aided design with an application to

car crash simulation. Current research focuses on either codification or personalisation of knowledge [9]. However, an integrated approach combines both the codification of knowledge objects and the transfer of contextual and tacit knowledge via communication. Collaboration also facilitates the sharing of knowledge between different actors. In this paper, the KM support system is based on an integrated knowledge model, an ontology, which captures formal, contextual and problem-solving knowledge. Knowledge retrieval models mimic the reasoning of simulation analysts and experts.

This paper is organised as follows. Section 2 presents a literature review on knowledge management in engineering design. Section 3 presents the industrial context describing car crash simulation in vehicle development. Section 4 explains the ontology-based approach to knowledge management and details the knowledge management support system by explaining its activities, blocks and sub-blocks. We propose an evaluation of the knowledge management support system using three use case scenarios that will be detailed in Section 5. A brief discussion is presented in Section 6 and conclusions are outlined in Section 7.

2. Literature review: knowledge management in engineering design

There are multiple definitions of knowledge management (KM) in literature [13], [14]. However, it is usually defined by its relationship with knowledge. Therefore, we start by defining knowledge.

Many ways of defining knowledge exist in literature. Knowledge is usually defined using a value chain or hierarchical relationship with data and information [2], [8], [9], [15]–[17]: where (1) data is in the form of symbols and words, (2) information is structured data that forms a pattern, finally (3) knowledge is processed information, facts and rules of thumb acquired through experience. Knowledge is difficult to assimilate and has a personal aspect that demonstrates the crucial difference between knowledge and information, a "notion of competence" and not a tangible object [9]. Knowledge can refer, from different perspectives, to an object, a cognitive state or a capability and it may reside in individuals, social groups, documents, processes, or computer applications and databases [11]. Knowledge can also refer to a state of knowing, the “know-about”, a capacity for action, the “know-how”, and finally to articulated and captured facts, methods and so on, the “body of knowledge” [18]. Another frequent dichotomy of knowledge is based on its stage of accessibility: explicit, implicit and tacit [2], [8], [9], [15]–[19]. Explicit knowledge, also identified as formal knowledge [9], is knowledge that can be expressed, codified and documented. Implicit knowledge is also knowledge that can be implied by or inferred from observable behaviour or performance. Tacit knowledge is subconscious and cannot be expressed.

In the context of engineering knowledge, formal knowledge can be codified in different sources, such as a 3D geometric model and a simulation model, and tacit knowledge refers to personal knowledge and experience, such as developing a problem-solving strategy and reasoning on possible decision. In [1], the authors explain that information is what is stored and transferred outside the human mind and knowledge only exists when information is interpreted. They classify engineering design knowledge into product knowledge and process knowledge. Thus, when it is stored externally, information can either describe the design process or the product itself. When it is sored un human memory, knowledge can be explicit, implicit and tacit. Explicit knowledge is an explanation of both the process and the product and can be articulated in the form of information. Implicit knowledge is the understanding the engineers have of both process and product and cannot be expressed by engineers themselves; it can be articulated through knowledge elicitation methods. Finally, tacit knowledge is more a matter of intuition and cannot be articulated.

Given the different definitions and perspectives of knowledge, knowledge management (KM) has multiples definitions. Knowledge management refers to identifying and leveraging collective knowledge [11], [12], [16] and it is usually defined as a process involving multiple sub-processes including knowledge capture, storing, retrieval and transfer [11], [14], [17], [20]. KM is defined as the ability of an organisation to manage, store, value and distribute knowledge [14]; an organisation is defined as a social unit of people structured and managed to meet a need or to achieve collective goals [12]. KM is an essential asset in modern institutions; it enables learning from corporate memory, growth, success and innovation [20]. In projects, KM enables communication improvement, best practice gathering while providing informal knowledge sharing, and productive collaboration [13]. Information systems plays a key role in the development of KM as it enables the automation, production and sharing of knowledge [20]. These systems are called knowledge management systems and they support KM processes [11]. The essential function of KMS is the sharing and reuse of knowledge. Therefore, to achieve this function, the acquisition of knowledge is fundamental. Thus, KMS has to fulfil three functions: acquiring knowledge, storing it and reusing it [21].

In engineering design, knowledge can be whether in form of tangible objects that can be modified, copied and transferred, or experience learnt and accumulated through a community of expertise; this constitute both approaches of KM, the codification and the personalisation view [9]. Most common approaches are technology-oriented, called codification approach; the explicit nature of knowledge is underlined, and knowledge need to be formalised and stored in knowledge repositories and transferred via Information and Communication Technologies [9]. On the other hand, people-oriented, also called personalisation approaches underlines the tacit and context-dependent nature of knowledge; it requires informal human communication to transfer knowledge [16], [22]. It is argued that a trade-off between the two approaches, called integrated approach, is more effective [2], [9], [22], [23]. An integrated approach to KM can improve the quality and efficiency of design through the capture and reuse of informal and contextual knowledge.

Knowledge representation and retrieval research areas aim to develop enabling technologies for knowledge management systems [9]. Ontology-based approaches to knowledge management have been developed for different application in design engineering. Sun et al. [24] propose a knowledge management approach to support knowledge intensive product design; they develop an ontology-based product knowledge model including different types of knowledge and propose methodologies for retrieving explicit and tacit knowledge. A prototype of the knowledge support system and an implementation at a paper currency binding' company are also carried out. Zhang et al. [21] construct a framework for an ontology-based knowledge management system. They also propose an implementation using a mechatronics case and develop part of the ontology-based knowledge retrieval modules. Premkumar et al. [25] develop an ontology-based semantic knowledge management system to support the design, analysis and manufacturing of laminated composite materials. The system includes an ontological framework for laminated composite materials and their design for manufacturing, and an integrated engineering design framework to facilitate collaboration among domain experts. To our knowledge, no ontology-based knowledge management approach has been developed to support simulation-aided design, and specifically car crash simulation process.

3. Industrial context: Car crash simulation in vehicle development

This research is conducted within a French multinational automotive company. Two empirical studies have been conducted with the aim of having a better understanding of the development phase of a vehicle development project [26], [27]. The results of these studies explain that simulation is expensive and time consuming, and that there is considerable effort required.

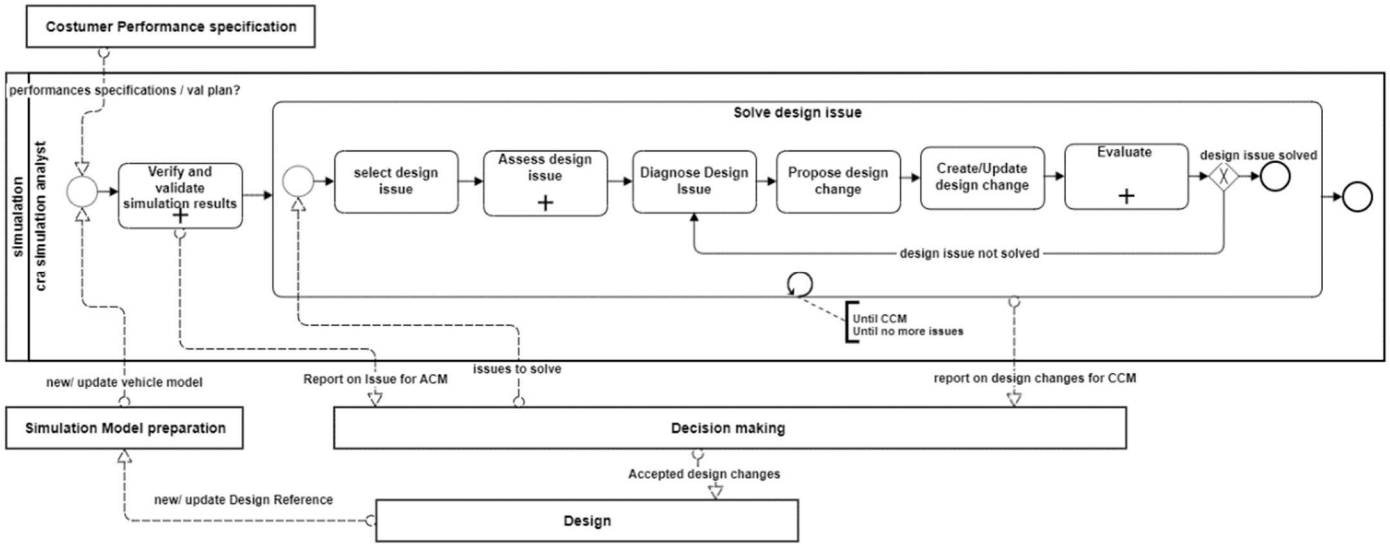

Figure 1. Simulation analyst's activities, focussing on the design issue resolution process, with a glance into the collaboration with other stakeholders

The development phase is a succession of digital loops, called design analysis loops, terminating in milestones. These loops involve design and simulation activities and are referred to as digital since the vehicle exists only in the form of numerical models, and both design and simulation activities are numerical. At the end of the development phase, the vehicle model must be at the right level of performance and the right manufacturing cost. The development phase consists of iteratively refining the design specifications, evaluating and validating the vehicle's performance and solving any design issues encountered. This phase is supported by modelling and simulation (M&S); the vehicle is

represented by a digital model and the engineering activities are digital. Several variations on a vehicle model exist at the same time, as there are several markets with different expectations. The design reference (DR) represents the knowledge gathered about the vehicle under development and its specifications. The design activity aims to define and adjust the DR. Simulation aims to evaluate the vehicle performance. If a design issue is revealed by the simulation, it must be solved. We focus on solving the design issues encountered during the car crash simulation.

Figure 1 shows the different processes in the development phase with a focus on the car crash simulation. The design process consists of creating and updating the DR. The simulation model generation process consists of creating and updating the digital vehicle models required for the simulation. The customer performance specification process provides the specification of the vehicle to be met. The simulation process evaluates the performance of the vehicle and solves the design issues; a design issue occurs when the performance does not meet the specification. Finally, the decision-making process determines which design changes to incorporate. A design change is a solution to the design issue. Different stakeholders are involved in the different processes, as explained in Table 1.

Table 1: Stakeholders of each process within the development phase

Process	*Stakeholders*
Simulation	Simulation analyst, synthesis engineer CAE
Design	Designer, synthesis architect
Simulation model generation	Model factory engineers
Costumer Performance specification	PPC, costumer performance leader
Decision making	Designer, synthesis architect, project manager

Our focus is on the crash simulation process and mainly on the analysts involved within this process. The analysts' missions are to evaluate and validate vehicle performances and to deliver a vehicle model compliant with multiple requirements. They rely on simulation to evaluate vehicle performance. During the development phase, for a design analysis loop, crash simulation analysts receive the specification and the vehicle model to be evaluated. Once they have obtained the simulation results, they proceed with the interpretation of these results. If the model does not meet one or more requirements, a design issue is identified. The analysts assess and diagnose the design issue, then propose a design change to fix the design issue. If the design change is not satisfactory (unresolved design issue), they consider a new path for another design change. If the design change satisfies the requirement, the project decision-makers decide whether to implement it in the vehicle model. This decision depends on project factors including cost and weight. The decision-making process is not considered in this research; only the resulting decision is considered.

Analysts are required to have knowledge of modelling and simulation. They are also expected to make decisions on paths relating to investigation of design issues and on proposals for design changes. At the end of each design analysis loop, all design changes that have been approved by the different disciplines, such as crash, stress and noise and vibration, are incorporated into the vehicle model, ready for the next loop.

Crash simulation analysts are geographically distributed, and there is only one crash simulation expert and one digital simulation expert in the company [28]. Car crash simulation analysts collaborate with different stakeholders. They collaborate with other crash simulation analysts from other work sites and with simulation analysts from different disciplines, such as noise and vibration simulation analysts. There is a strong collaboration between simulation analysts and simulation experts, as the latter have the most significant expertise within the company. Analysts, designers and customer performance specialists exchange information on the status of the resolution of design issues. Finally, decision-makers participate in the decision-making process regarding the design change to be incorporated into vehicle models.

According to the empirical study [27], analysts encounter several challenges within car crash simulation. They need to quickly locate all relevant data and knowledge, from different sources, in as little time as possible. Design-issue analysis and design-change proposal formulation need to be improved, in order to reduce the time spent on simulation and reduce bias of less experienced analysts when interpreting simulation results.

This paper addresses these issues by proposing an ontology-based knowledge management approach to support simulation-aided design. The knowledge management support system is based on a knowledge model that formalises formal knowledge and knowledge related to the resolution of design issues. Knowledge retrieval models will support

the analysis of design issues and design change proposals. Section 4 details the architecture of the KM support system with a focus on engineering knowledge capture and retrieval.

4. An ontology-based approach for an integrated and collaborative knowledge management

To improve the crash simulation analysis process, time should be reduced and decisions concerning investigation paths and design-change proposals should be substantiated. The crash simulation analysis process is knowledge intensive and highly dependent on expertise. Since there is only one crash simulation expert, it is important to capture the knowledge and expertise and share it with analysts. Collaboration between the different people involved is very important for the process to run smoothly. With these characteristics taken into consideration, we propose an integrated and collaborative knowledge management approach to support analysis and to ensure collaboration with the different stakeholders involved in the design issue resolution process.

In previous work [28], we proposed formalising the knowledge related to car crash simulation using an ontology. The interest in developing ontologies is growing in engineering design as it involves knowledge sharing and the development of a common standard language [29], usually used for the formalisation of domain knowledge [25]. The proposed ontology is called Crash Simulation Post-Processing CSPP ontology, and an overview of this ontology will be given in section 4.1. We also presented the first steps towards the development of the knowledge management support system in later work [30], based on an in-depth literature review of knowledge management frameworks and knowledge management systems architecture while taking into account the specificities of our industrial context, in light of the ontology.

Hence, in this paper, as an extension of the research presented in [30], we focus mainly on the development of the virtual KM support system. First, we define its capabilities, then we present the architecture and specify the blocks and sub-blocks.

4.1. The Crash Simulation Post-Processing (CSPP) Ontology

In this section, we briefly explain the Crash Simulation Post-Processing Ontology that we have developed in previous work [28]. The purpose of crash simulation is to evaluate the performance of a specific vehicle model. The ontology, therefore, formalises the knowledge related to design issue resolution, mainly the knowledge related to the post-processing phase of the crash simulation and the interpretation of the simulation results. The ontology is formalised according to three levels to better structure the engineering knowledge. Firstly, the context level contains information on the context in which the digital test takes place, such as the requirements to be met and the impact configuration, including frontal and rear crashes. Second, the project level formalises the engineering knowledge related to a specific vehicle project, including the vehicle model, the assessment of simulation results, and the resolution of design issues. Finally, the reasoning level formalises the reasoning behind the resolution of design issues. For brevity, we explain only the concepts, mainly at the level of reasoning, that are necessary to understand this paper.

A design issue occurs when requirements are not met by the vehicle. A vehicle model consists of vehicle parts, and each vehicle part has a role to play in the vehicle structure during the crash test and adopts an expected behaviour to ensure this role. Thus, we define a design issue as a behaviour gap between the expected behaviour and a defect behaviour of a vehicle part. A design issue has root causes. A root cause is also defined as a behaviour gap of a vehicle part. An elementary root cause is a root cause to which the analyst can propose a design change. A design change is one or more corrective actions applied to a vehicle part. This reasoning is based on the hypothesis that, first of all, crash simulation is based on physical and mechanical phenomena. Therefore, the behaviour of vehicle parts has limited possibilities of deformation and can be predictable. Secondly, vehicle parts sharing the same role and behaviour would be exposed to similar issues and could, therefore, belong to the same category. Then, issues with similar diagnosis and root causes would share similar corrective actions.

4.2. Capabilities of the knowledge management support system

Engineering knowledge related to design-issue resolution includes the design issue, its root causes, and associated design changes, as formalised in the CSPP ontology. We now refer to it as engineering knowledge.

The capabilities of the KM support system can be characterised into three overall capabilities:

- Collaboration: The support system must ensure collaboration between the various stakeholders within vehicle projects. Virtual collaboration is considered in this context.
- Capturing engineering knowledge: the support system must be able to capture engineering knowledge based on the CSPP ontology, classify it, and store it in knowledge repositories.
- Retrieval of engineering knowledge: the support system must be able to retrieve knowledge based on user queries and share it with users.

This paper aims to develop an integrated and collaborative knowledge management system. It is integrated since the CSPP ontology combines formal knowledge, such as requirements and models, and tacit knowledge, such as design issue resolution. The integrated aspect is also present in the combination of both methodologies for codification and personalisation of KM. The KM support system ensures the codification of knowledge and the capture of in-context and tacit knowledge through communication. The collaborative aspect is undertaken by ensuring stakeholder collaboration within vehicle projects and enabling communication.

4.3. A virtual KM support system for simulation-aided design

Based on pre-defined capabilities, we propose a virtual system to support simulation-aided design within vehicle projects. As shown in Figure 1, the support system consists of four main parts: three layers and a multi-user interface.

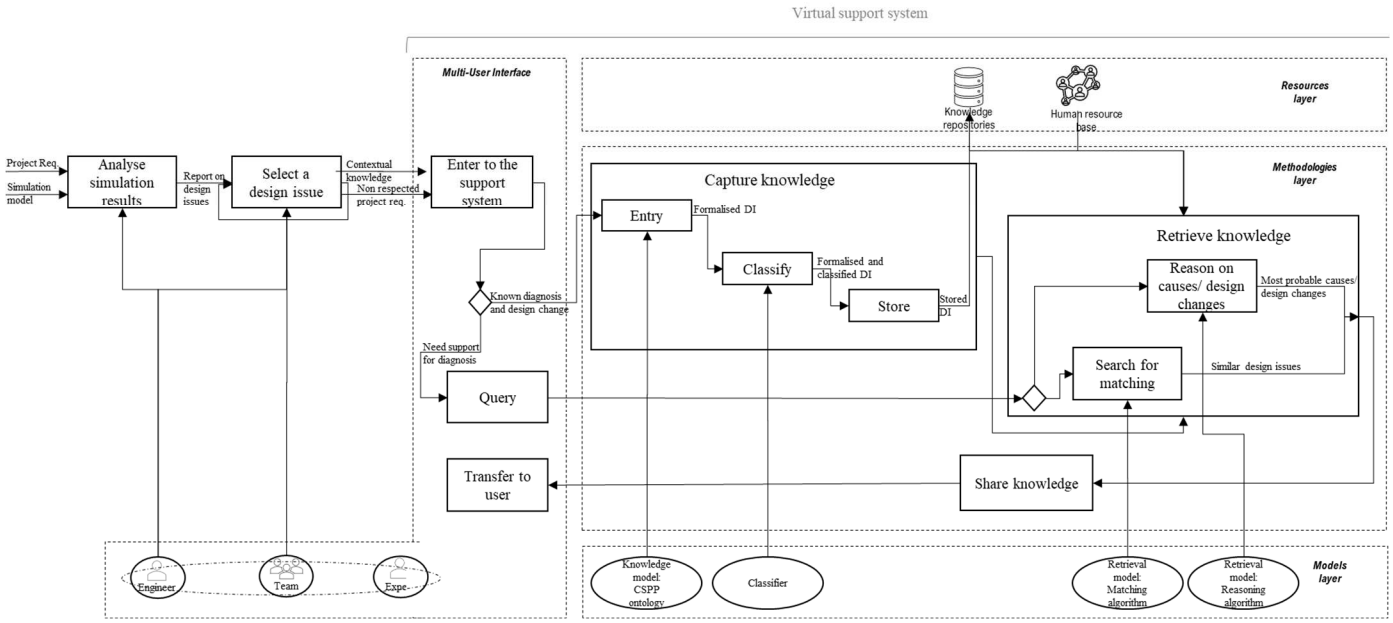

Figure 2: The description of the Virtual KM support system, with its three layers and multi-user interface. A representation of the activities performed before interacting with the KM support system is also given (on the right), to identify the inputs required.

The three layers constituting the support system are the resources, methodologies, and models layers:

- The resources layer refers to knowledge repositories in various data and knowledge formats. Repositories may include data and information such as simulation models and reports used and transferred as part of the process. Other explicit and implicit knowledge is stored in knowledge repositories. Implicit and tacit knowledge embedded in the human mind could be shared through communication. Therefore, the KM support system includes a human resource base of employees, detailing their roles, skills, and capabilities.
- The models layer consists of models developed to support the implementation of the system, including an integrated knowledge model, a classifier, and two knowledge retrieval models. The integrated knowledge model is the CSPP ontology. The classifier and the two retrieval models will be explained later in Section 4.4.1.

- The methodologies layer represents the methodologies used to conduct KM activities. Based on the capabilities detailed in section 4.2, two principal methodologies are required: "capture knowledge," leading from raw information to knowledge stored in repositories, and " retrieve knowledge," leading from knowledge in repositories to specific knowledge required by the user. We then added " share knowledge" to ensure the continuity of the information flow from the knowledge retrieval to the user. More details on "capture knowledge" and "retrieve knowledge" will be presented in section 4.4.

Finally, the multi-user interface (MUI) ensures interaction between users and the virtual system and collaboration between users within the integrated work environment. Users, including engineers, experts, and teams, can have distributed workspaces for individual tasks and project environments for collaboration if required. The support system must ensure collaboration between stakeholders working on the same project via a virtual environment where they can communicate, interact, and update the current project. The MUI provides multiple activities to ensure user interaction with the system, such as "enter support system," "query," and "transfer to the user." These activities ensure, firstly, the input of input data into the system, secondly, the choice of entering new engineering knowledge or retrieving knowledge, and thirdly, the transfer of the retrieved knowledge to the user. All MUI activities are accessible to the various users. For reasons of clarity, the links between users and MUI activities are not presented in the Figure 2.

4.4. The KM support system architecture

In this section, we focus on both the methodologies and models layers of the KM support system. As explained in Section 4.3, and based on the capabilities previously described, the KM support system must have two blocks realizing both "capture knowledge" and "retrieve knowledge" activities. Each block consists of sub-blocks and calls for a specific model to perform each function.

4.4.1. The knowledge capture block:

The KM support system must be capable of capturing engineering knowledge from different resources, such as extracting knowledge from formal knowledge records (such as simulation models, reports) and obtaining knowledge from the user. We focus on capturing the user's engineering knowledge.

The knowledge capture block carries out the "knowledge capture" activity presented in Figure 2. It should allow the user to document new design issues and their resolution. Accordingly, engineering knowledge is entered into the system based on the CSPP ontology. Each design issue is classified into its category via a classifier and then stored in knowledge repositories. Thus, three different sub-blocks are identified to perform three different functions.

Firstly, the *Entry* sub-block, shown in Figure 2, allows formalising the data and information entered into the system using the knowledge model. Information about the design issue and its resolution will be entered in the fields proposed by the CSPP ontology. The output of this sub-block is the formalised engineering knowledge.

Second, the *Classify* sub-block, shown in Figure 2, allows the system to classify formalised design issues. This sub-block relies on the classifier. The classifier can be based on a statistical or deterministic classification that identifies a set of categories to which the design issues belong. The result of this sub-block is a formalised design issue with its corresponding category.

In this paper, to exemplify, we propose a deterministic classifier based on the categories explained at the reasoning level of the CSPP ontology. Vehicle parts sharing similar roles and behaviours would be subject to similar design issues and then to similar design changes. Therefore, vehicle parts sharing similar roles and behaviours would belong to the same category. In this case, the classifier will sort the design issue according to the vehicle part affected. Based on this reasoning, we propose a classifier based on matrix multiplication, as explained in Figure 3. The first matrix P defines the vehicle parts (vp) by their roles (r) and their behaviours (b) using a pair (r, b). The second matrix C defines the categories (cat) according to the pair (r, b). Finally, the matrix PC classifies the vehicle parts into their respective categories by matrix multiplication of P and C, while maximising $pc_i > 0$ to 1.

In order to classify a new vehicle part, the designer must be involved. The designer must assign the pair (r,b). If (r,b) exists, the classifier will classify the new vehicle part. If not, the designer shall fill in the matrix C.

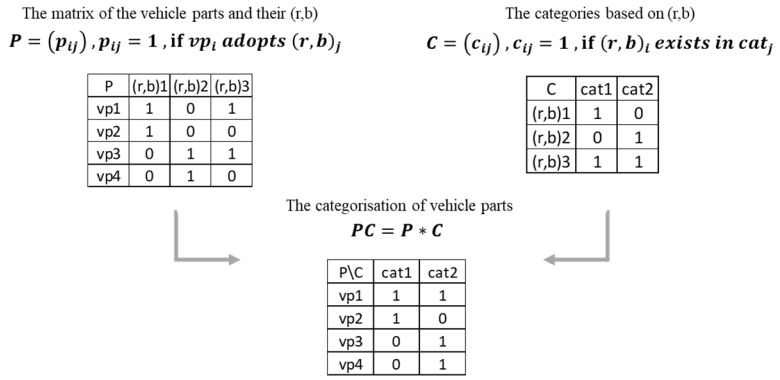

Figure 3: The classifier is described, which is based on the multiplication of matrices. An example is drawn for a better understanding. The equations representing each matrix are presented in detail. The matrix PC is the result of the product of the matrix and a normalization at 1 for all matrix elements >1.

Finally, the *Store* sub-block (see Figure 2) allows the system to store the formalised and classified design issue in knowledge repositories, ready to be retrieved later. The knowledge repository space would consist of several clusters representing the categories and containing the associated design issues and their resolution. It would facilitate later retrieval as it would limit the search space.

4.4.2. The knowledge retrieval block:

The knowledge retrieval block realises the "retrieve knowledge" activity presented in Figure 2. Based on the user's query, the knowledge retrieval block must call the appropriate retrieval model. Two queries are possible. First, the user may request a list of design issues that are similar to the design issue under study along with their resolution. For this purpose, the support system accesses the sub-block *Search for matching*, involving the matching algorithm. The user can also request the most probable design diagnosis (causes) and design change for the design issue under study. For this purpose, the support system accesses the *Reasoning on causes / design change* sub-block (referred to as Reasoning), involving the reasoning algorithm. The knowledge retrieval block automatically calls the knowledge capture block so that the support system captures the studied design issues. If the knowledge retrieval block does not find an answer to the query, it is supposed to search for the most apt stakeholder to answer the query. This option is not detailed in this paper.

To exemplify the two retrieval models, we propose a filter for the matching algorithm and a probabilistic reasoning for the reasoning algorithm.

For the matching algorithm, we propose a two-step filter, shown in Figure 4. As explained above, the knowledge repository space is divided into clusters of categories of design issues and their resolution. Therefore, similarity means that the design issues belong to the same categories. Thus, the first step will select the categories related to the vehicle part affected by the design issue, and thereby obtain all design issues from these categories. The second step is optional. If necessary, the user can define filter criteria in the query, according to which design issues resulting from the first step will be filtered. The filter criteria will address one or more elements of the definition of the design issue, such as the criterion to be respected and the impact configuration.

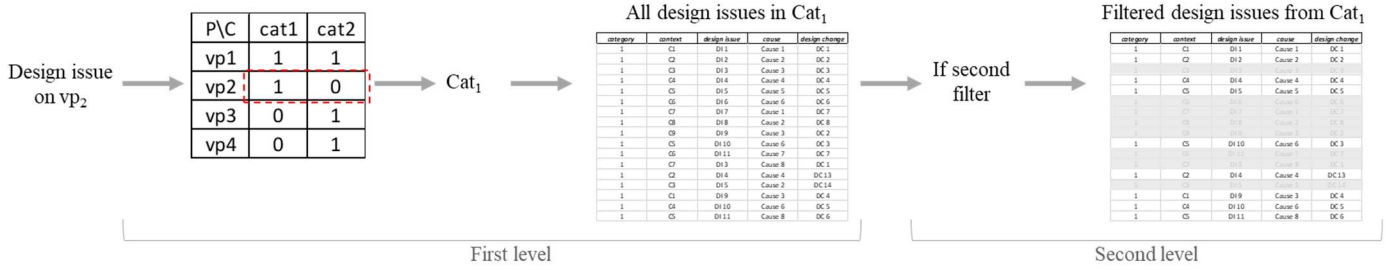

Figure 4: The two-step filter for the matching algorithm, with an example for explanation. Knowing the vehicle part affected by the design issue under study, the first step of the matching algorithm would identify the corresponding categories (Cat_1) to the vehicle part (vp_2) and then list all the respective design issues and their resolution. Then, for the second step, based on the filter criteria, it would retain only the design issues that meet those criteria. The hidden lines are the eliminated design issues.

More explicitly, for the first step, the matching algorithm will go into the PC matrix and extract the categories related to the vehicle part. Then, we will obtain a database containing all the design issues related to the selected categories. Then, for the second step, based on the filter criteria identified by the user, the matching algorithm will filter the new set of design issues, merely keeping the ones that satisfy these criteria.

I- Matching

category	context	design issue	cause	design change
1	C1	DI 1	Cause 1	DC 1
1	C2	DI 2	Cause 2	DC 2
1	C3	DI 3	Cause 3	DC 3
1	C4	DI 4	Cause 4	DC 4
1	C5	DI 5	Cause 5	DC 5
1	C6	DI 6	Cause 6	DC 6
1	C7	DI 7	Cause 1	DC 7
1	C8	DI 8	Cause 2	DC 8
1	C9	DI 9	Cause 3	DC 2
1	C5	DI 10	Cause 6	DC 3
1	C6	DI 11	Cause 7	DC 7
1	C7	DI 3	Cause 8	DC 1
1	C2	DI 4	Cause 4	DC 3
1	C3	DI 5	Cause 2	DC 6
1	C1	DI 9	Cause 3	DC 4
1	C4	DI 10	Cause 6	DC 5
1	C5	DI 11	Cause 8	DC 6

II- Construction of the network based on probabilistic reasoning

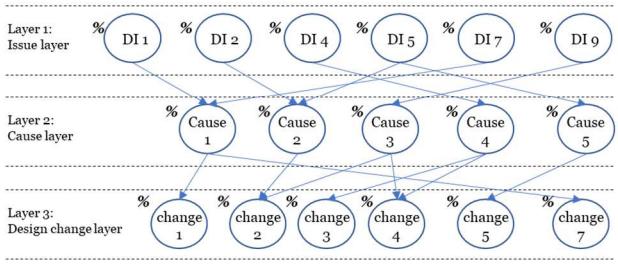

Figure 5: (I) presents the list of design issues and their resolution that is the result of the matching algorithm for the design issue under study. (II) explains the construction of the network, where some design issues (DIs) are presented with their respective causes and design changes. The % represents the probability of each node, which is calculated based on a statistical calculation of occurrence from part (I).

The reasoning algorithm must propose the most probable cause of the design issue and the most probable design change. As explained in Figure 5, it starts by calling the matching algorithm, which will identify the categories to be addressed, and thereby all design issues, from these categories, and their resolution. Then, based on probabilistic reasoning, it would establish a network of three layers, including design issues, causes, and design changes, while assigning their probability of occurrence. Hence, the reasoning algorithm will present the most probable causes and the most probable design changes for the design issue under study.

5. Knowledge management support system, application and evaluation

In this section, for the evaluation of our proposal, we present three use case scenarios with a proof of concept implementation for each model. Each use case scenario will take into account a methodology (capture and retrieval) and the models that will be applied. For the three scenarios, we have built a knowledge base of several design issues and their resolution. This base is built based on real information documented by simulation analysts, and we have duplicated some information in order to have more cases and to be able to calculate probabilities. As shown in Figure 6, the columns represent the fields proposed by the CSPP ontology; not all of them are presented, since we chose some of them to have a complete picture without getting into all details.

vehicule_seg	impact_config	performance	criterion	target	scope_crtierion	appreciation	category	cause_scope	expected_behaviour	behaviour_gap	elementary_cause	corrective_action
A	front	EuroNCAP	MPS	<2	Side_member	High	C1	CB	Compression	High compression	Low stiffness	Material change
A	front	EuroNCAP	MPS	<2	Floor_panel	High	C1	CB	Compression	High compression	Low stiffness	increase thickness
B	front	EuroNCAP	MPS	<2	Side_member	High	C1	CB	Compression	Break	Low stiffness	increase thickness
A	front	EuroNCAP	MPS	<2	Side_member	High	C1	CB	Compression	No compression	High stiffness	Material change
B	front	EuroNCAP	MPS	<2	Floor_panel	High	C1	CB	Compression	High compression	Low stiffness	increase thickness
A	rear	EuroNCAP	MPS	<2	Floor_panel	High	C1	Side_member	Compression	High compression	Wrong geometry	add rainure
A	rear	EuroNCAP	MPS	<2	Floor_panel	High	C1	CB	Compression	Low compression	High stiffness	decrease thickness
A	rear	EuroNCAP	MPS	<2	Side_member	High	C1	Side_member	Rotule	No rotule	Wrong geometry	design of new part
C	rear	EuroNCAP	MPS	<2	Side_member	High	C1	Side_member	Rotule	No rotule	High stiffness	decrease thickness
A	front	EuroNCAP	MPS	<2	Towing_bracket	High	C1	CB	Compression	Low compression	High stiffness	increase thickness
A	front	EuroNCAP	MPS	<2	Side_member	High	C1	Side_member	Compression	Low compression	Wrong geometry	design of new part
C	Pedestrian	NCAP	HIC	<1000	Hood	High	C2	Hood	contact force	High force	High stiffness	design change
B	front	EuroNCAP	MPS	<2	Side_member	High	The example of the use case scenario					

Figure 6: The knowledge base used for the use case scenarios. The last line represents the example that will be used for the use case scenarios 2 & 3.

For each use case scenario, we will document the interaction between the user and the system and present the inputs and outputs of each scenario. Figure 7 shows the main part of the algorithm that interacts with the user to obtain the query and then call the appropriate block or sub-block to answer the query. We used Python 3 for programming.

```
if __name__ == "__main__":

    txt = input("What do you want to do? (add design issue, find solutions, find causes,
find similar inputs)")

    if txt == "design issue" or "add design issue":
        capture_desing_issue('database.xlsx')

    else:
        db, levels_of_db, test_scenario = capture_test_scenario('database.xlsx')

        if txt == "solutions" or "find solutions":
            find_solutions(levels_of_db, test_scenario, "solutions")

        elif txt == "causes" or "find causes":
            find_solutions(levels_of_db, test_scenario, "causes")

        elif txt == "similar inputs" or "find similar inputs":
            category = test_scenario['category'][0]
            filter_similar(db, category)

        exit("closing app")
```

Query

If the query is to add a new design issue and its resolution, call for the knowledge capture block

If the query is to retrieve knowledge, start by calling the knowledge capture block

If the query is to search for most probable cause/ design change, call for the reasoning sub-block

If the query is to search for similar design issues, call for the matching sub-block

Figure 7: Presentation of the main part of the algorithm with a detailed description. The main part allows the user to enter the query. Then, depending on the query, it calls the appropriate functions.

5.1. Use case scenario 1: Capture engineering knowledge

The user must enter a new design issue with its resolution to the KM support system. The KM support system will then access the *Knowledge Capture* block and call the CSPP ontology and classifier as models.

Figure 8 shows the function “capture_design_issue” responsible for capturing engineering knowledge, classifying it and storing it.

```
def capture_desing_issue(name):

    table = pd.read_excel(name)
    print("input needed are {}".format(table.columns))

    input_line = {}
    for col in table.columns:
        if col != 'category':
            data = input('What is the value for {}?'.format(col))
            input_line[col] = data or None

    input_line = find_category(input_line)

    validation_and_add_to_db(input_line, name, table)

    return input_line
```

Propose the ontology fields for the user to fill

Entry of the knowledge documented by the user

Classification of the design issue

Storing

Figure 8: The knowledge capture block, starting with entry, through to classification and finally storing the design issue and its resolution.

Figure 9 describes the interaction between the user and the KM support system while capturing a new design issue and its resolution. The user must enter the various fields provided by the KM support system. The user is then requested to validate the new design issue and its resolution before storing it. As it is shown, the category is automatically generated by the classifier.

```
What do you want to do? (add design issue, find solutions, find causes, find similar inputs)
add design issue

input needed are ['vehicule_seg', 'impact_config', 'performance', 'criterion', 'target',
'scope_crtierion', 'appreciation', 'cause_scope', 'expected_behaviour', 'behaviour_gap',
'elementary_cause', 'corrective_action']


What is the value for vehicule_seg? A
What is the value for impact_config? rear
What is the value for performance? NCAP
What is the value for criterion? MPS
What is the value for target? <2
What is the value for scope_crtierion? Side_member
What is the value for appreciation? High
What is the value for cause_scope? CB
What is the value for expected_behaviour? Compression
What is the value for behaviour_gap? High compression
What is the value for elementary_cause? Low stiffness
What is the value for corrective_action? Material change


FYI, the part is in the category : C1
this line will be added to the database, do you validate ? (yes/no) : {'vehicule_seg': 'A',
'impact_config': 'rear', 'performance': 'NCAP', 'criterion': 'MPS', 'target': '<2',
'scope_crtierion': 'Side_member', 'appreciation': 'High', 'cause_scope': 'CB',
'expected_behaviour': 'Compression', 'behaviour_gap': 'High compression', 'elementary_cause':
'Low stiffness', 'corrective_action': 'Material change', 'category': 'C1'}yes
```

Figure 9: The knowledge capture block, with its inputs and outputs. Queries are in green. The result of this block is the creation and storage of a new design issue. The system does not ask the user to enter the category. However, it is assigned later when the system requests validation of the information entered.

5.2. Use case scenario 2: retrieval of similar design issues

The user will search for design issues similar to the one under study (last line of Figure 6). Therefore, during the query, the user will ask for "search for matching". The user will also use filter criteria to narrow the search result. The KM support system will then access the *Search for matching* sub-block and the matching algorithm as a model. Figure 10 presents the function “filter similar” in the algorithm, representing the *Search for matching* sub-block, with the two steps filter.

```
def filter_similar(table, category ):

    res = table.loc[category == table.category]

    other_filter = input("do you want to add another filter ?")
    while other_filter == "yes":
        print("the possible columns to filter are {}".format(list(table.columns)))
        precise_filter = input("write the filter like this :  'columns : value'").split(" : ")
        res = res.loc[precise_filter[1] == res[precise_filter[0]]]
        other_filter = input("do you want to add another filter ?")

    print(res)
    return res
```

First step filter based on the category

Second step filter based on filter criteria chosen by the user

Figure 10: The search for matching sub-block, with the matching algorithm represented by the function *loc [rows [column = value]]*. This function filters the rows of the knowledge base, based on a specified column. For the first step filter, the *[column=category]*, and the second step filter, the value of the column will be entered by the user.

As mentioned above, for use case scenario 2, the user requests design issues similar to the one under study (represented in the last line of the knowledge base). Figure 11 represents the interaction between the user and the support system and the results of the support system. First, the user selects "similar inputs" in the query and then requests a second step filter based on "impact_config=front." The support system then lists seven design issues and their resolution, similar to the one understudy.

```
What do you want to do? (add design issue, find solutions, find causes, find similar inputs)similar inputs
New study case found, we will try to find possible solutions
do you want to add another filter ?yes
the possible columns to filter are ['vehicule_seg', 'impact_config', 'performance', 'criterion', 'target', 'scope_crtierion',
'appreciation', 'category', 'cause_scope', 'expected_behaviour', 'behaviour_gap', 'elementary_cause', 'corrective_action']
write the filter like this :  'columns : value'impact_config : front
do you want to add another filter ?no

   vehicule_seg impact_config performance criterion target scope_crtierion  \
0             A         front    EuroNCAP       MPS     <2     Side_member
1             A         front    EuroNCAP       MPS     <2     Floor_panel
2             A         front    EuroNCAP       MPS     <2     Side_member
3             A         front    EuroNCAP       MPS     <2     Side_member
4             B         front    EuroNCAP       MPS     <2     Floor_panel
9             A         front    EuroNCAP       MPS     <2  Towing_bracket
10            A         front    EuroNCAP       MPS     <2     Side_member

   appreciation category  cause_scope expected_behaviour     behaviour_gap  \
0          High       C1           CB        Compression  High compression
1          High       C1           CB        Compression  High compression
2          High       C1           CB        Compression             Break
3          High       C1           CB        Compression    No compression
4          High       C1           CB        Compression  High compression
9          High       C1           CB        Compression   Low compression
10         High       C1  Side_member        Compression   Low compression

   elementary_cause   corrective_action
0     Low stiffness     Material change
1     Low stiffness  increase thickness
2     Low stiffness  increase thickness
3    High stiffness     Material change
4     Low stiffness  increase thickness
9    High stiffness  increase thickness
10   Wrong geometry  design of new part
```

Figure 11: The interaction between the user and the support system and the outputs. Queries are in green. The result is an extraction from the knowledge base. The user has requested design issues similar to the one he has documented and has requested a second step filter using the "impact_config: front" criterion. Seven design issues respond to the query.

5.3. Use case scenario 3: Retrieval of most probable causes and design changes

The user is looking for the most likely cause and design change for a design issue they are addressing. Therefore, when querying, the user will request "reason for the cause or design change". The KM support system then accesses the sub-block " *Reasoning on causes/ design change"* and the reasoning algorithm as a model whose representative function "find_solutions" is shown in Figure 12. To have a better understanding of the construction of the network, we present an example in Figure 13. The network layers in this use case scenario are consistent with the concepts used for the CSPP ontology. Starting by the design issue layer, the layers are: “cause scope”, “behaviour gap”, elementary cause” and “design change”.

```
def find_solutions(levels_of_db, test_case_scenario, last_layer):

    final_solutions = {}

    cause_scope_layer = search_children(levels_of_db["cause_scope"], test_case_scenario, "cause_scope")

    network_layers = ["behaviour_gap", "elementary_cause", "corrective_action"]
    if last_layer == "solutions" else ["behaviour_gap", "elementary_cause"]

    for studied_cause_scope, cause_scope_weight in cause_scope_layer.items():
        parent_layer = {studied_cause_scope: cause_scope_weight}

        for layer in network_layers:
            children_layer = {}
            for parent_node, parent_node_weight in parent_layer.items():
                children_nodes = search_children(levels_of_db[layer], parent_node, layer)
                children_layer = build_children_layer(children_nodes, children_layer, parent_node_weight)
            parent_layer = {k: v for k, v in sorted(children_layer.items(), key=lambda item: item[1],
             reverse=True)}

        final_layer = {key: round(value, 2) for key, value in parent_layer.items()}
        final_solutions[studied_cause_scope] = final_layer
        print("solutions for cause {} are : {}".format(studied_cause_scope, parent_layer))

    print(final_solutions)
    return final_solutions
```

Identify the children nodes (from the cause scope layer) of the design issue under study

Identify the layers involved, if the search focuses on causes or for design changes

for each node from the cause scope layer, construct the network

Construct the network: starting from a node of the cause scope layer, identify the children nodes (from the next layer) and calculate their probabilities

Print results for each cause scope node

Figure 12: The sub-block of the *Reasoning on causes/design change* represented by the "find solutions" function. In the "test_case _senario," the first step of the matching is performed. Starting from the cause scope layer, its nodes, and their probabilities, the construction of the network consists of identifying, for each layer, the children nodes, and their probabilities, up to the layer of elementary causes or design changes.

As shown in Figure 13, focusing on the first two layers, we begin by identifying the child nodes, from the cause scope layer, of each parent node, from the issue layer. Next, the probability of a child node is the probability of the parent node multiplied by the probability of the arrow (the probability of the cause scope knowing the design issue). If several arrows are pointing at a child node, then the probability is the sum of all the probabilities of the parent nodes multiplied by the probability of their arrows.

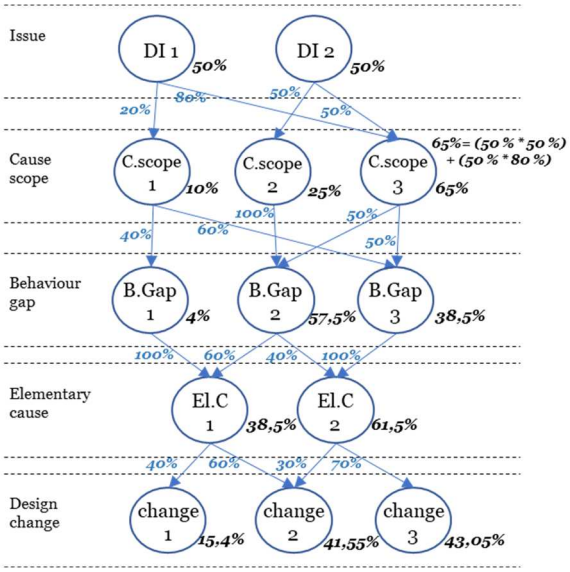

Figure 13: A detailed explanation of the proposed network, with its layers, and the calculation of probabilities for each node. The probability of each node is the probability of the parent node multiplied by the probability of the arrow.

For this use case, we present two scenarios: first, the user requests the causes, and second, the user requests the design changes. Figure 14 shows the interaction between the user and the support system when requesting causes and Figure 15 when requesting design changes. Both queries are for the same design issue with the same inputs.

```
What do you want to do? (add design issue, find solutions, find causes, find similar inputs)causes
New study case found, we will try to find possible solutions

solutions for cause CB are : {'Low stiffness': 29.55, 'High stiffness': 21.22, 'Wrong geometry': 12.88}

We need more info about the behaviour of the possible problematic piece : Side_member
Behaviours can be ['Compression', 'Rotule']
what value do you have as expected_behaviour ? Compression

solutions for cause Side_member are : {'Low stiffness': 13.63, 'High stiffness': 12.12, 'Wrong geometry': 10.6}

{'CB': {'Low stiffness': 29.55, 'High stiffness': 21.22, 'Wrong geometry': 12.88}, 'Side_member': {'Low stiffness': 13.63,
'High stiffness': 12.12, 'Wrong geometry': 10.6}}

closing app
```

Figure 14: The results of the KM support system for reasoning on causes. Queries are in green. In the end, we can read that two scopes of causes are possible, the CB and the Side_member, with their respective elementary causes and probabilities, in decreasing order.

For the design issue under study, the cause scope can be either the "Crash Box" or the "Side member." For the reasoning on causes (Figure 14), the algorithm stops at the elementary cause layer. It gives, as a result, each cause scope with its respective elementary causes and their probabilities, in descending order. Moreover, for the reasoning on design changes (Figure 15), the algorithm stops at the design change layer. It gives as output the same cause scope (since we are studying the same design issue), with their respective design changes and their probabilities, in descending order.

```
What do you want to do? (add design issue, find solutions, find causes, find similar inputs)solutions
New study case found, we will try to find possible solutions

solutions for cause CB are : {'increase thickness': 27.47, 'Material change': 12.69, 'decrease thickness': 10.61,
'design of new part': 8.59, 'add rainure': 4.29}

We need more info about the behaviour of the possible problematic piece : Side_member
Behaviours can be ['Compression', 'Rotule']
what value do you have as expected_behaviour ? Compression

solutions for cause Side_member are : {'increase thickness': 13.26, 'design of new part': 7.07, 'Material change': 6.44,
'decrease thickness': 6.06, 'add rainure': 3.53}

{'CB': {'increase thickness': 27.47, 'Material change': 12.69, 'decrease thickness': 10.61, 'design of new part': 8.59,
'add rainure': 4.29}, 'Side_member': {'increase thickness': 13.26, 'design of new part': 7.07, 'Material change': 6.44,
'decrease thickness': 6.06, 'add rainure': 3.53}}

closing app
```

Figure 15: The results of the KM support system for reasoning on design changes. Queries are in green. In the end, it indicates the same cause scopes, with their respective design changes and probabilities, in decreasing order.

6. Discussion

The integrated and collaborative knowledge management system aims to support simulation-aided design, based on the different models and technologies used. The KM support system captures engineering knowledge using the CSPP ontology, which is characterised as integrated since it combines formal and tacit knowledge. It also ensures the retrieval of engineering knowledge, based on user queries, using different retrieval models. The retrieval models are developed based on the reasoning already developed in the CSPP ontology, which formalises the resolution of design issues and is independent of the specificities of the vehicle project. This reasoning imitates the tacit knowledge of analysts and experts. The collaborative KM support system not only facilitates the resolution of design issues but also provides contextual and tacit knowledge captured during communication between the different stakeholders of vehicle projects.

The different models, which ensure the capture and retrieval of engineering knowledge, are developed and evaluated using use case scenarios of car crash simulation. The models and technologies used are effective in capturing and retrieving knowledge since the outcomes meet the expected results communicated by analysts and experts.

Time is one of the most important constraints in project development. The KM support system would help reduce crash simulation time by reducing the time spent solving design issues and proposing design changes. During the empirical study conducted within the company, we proved that simulation with more certainty about the results would allow us to gain at least one iteration per issue and so we save about nine hours of computation and about one hour of analysis [27]. Based on interviews with analysts, they encounter two or three significant design issues for about fifteen impact configurations, per project and design loop. Thus, by saving one iteration per design issue, we would save about thirty-five hours or one week of the design loop, out of the five weeks it usually lasts. Hence, we would save 20% of the analyst's time.

The KM support system would also contribute to more accurate proposals for more effective design changes since proposals are made based on in-depth knowledge of the company. It could lead to the development of more robust vehicle models. The KM support system is also considered a learning tool, where novice analysts would have access to the engineering knowledge retained.

The main functionalities of the system have been developed and applied, although it is still considered a prototype which can be further enhanced. First, there is a need to integrate the different simulation disciplines, such as noise and vibration and stress. Second, a web-based prototype must be developed to facilitate communication between users and the system. Third, users, with different roles, need to work in their virtual spaces and create knowledge as the project progresses, which will enrich the knowledge base and improve the models.

In this paper, the KM support system is applied to the car crash simulation. Although, it can be generalised to different simulation disciplines, such as strength and noise and vibration. The same laws of physics apply to the different disciplines. Based on the analysis of documented design issues from different simulation disciplines, the design issues are mainly described in the same way as for crash simulation as well as the design changes. Only the context level of the ontology should integrate the concepts formalising the requirements of the discipline. The project level of the ontology could also integrate some specificities of the discipline if needed. However, the reasoning level of the ontology would remain the same. Hence, the models implemented within the KMS support system would remain

unchanged. Future work on the generalisation of the ontology-based knowledge management approach will be undertaken.

7. Conclusion

In this paper, we present an ontology-based knowledge management system to support simulation-aided design, specifically car crash simulation. The KM support system includes an integrated ontology-based knowledge model that structures and formalises the formal knowledge and tacit knowledge related to design issue resolution. It also includes various knowledge retrieval methods to satisfy user needs. Moreover, a multi-user interface facilitates and enhances collaboration on vehicle projects.

A prototype and parts of knowledge retrieval methods are implemented. However, the next steps in this research need to focus on large-scale implementation of the support system and the generalisation to different simulation disciplines. The KM support system has been demonstrated to save simulation time. Further research could investigate the autonomy of such a system, whether we should continue to support analysts to analyse and solve design issues, or whether only expert validation would be sufficient.

References

[1] K. Wallace, S. Ahmed, and R. Bracewell, "Engineering knowledge management," in *Design process improvement*, London: Springer London, 2005, pp. 326–343.

[2] C. Mcmahon, A. Lowe, and S. Culley, "Knowledge management in engineering design: personalization and codification," *J. Eng. Des.*, vol. 15, no. 4, pp. 307–325, 2004.

[3] S. Yadav and S. K. Pradhan, "Investigations into Dynamic Response of Automobile Components during Crash Simulation," *Procedia Eng.*, vol. 97, pp. 1254–1264, Jan. 2014.

[4] H. Ritchie and M. Roser, "Causes of Death," *OurWorldInData.org*, 2018. [Online]. Available: https://ourworldindata.org/causes-of-death#causes-of-death-in-recent-decades.

[5] S. Patane, K. M. Narkar, and D. R. Panchagade, "Evaluation of Tools & Techniques used for Frontal Crash Analysis." 2015.

[6] X. F. Zha and H. Du, "Knowledge-intensive collaborative design modeling and support: Part I: Review, distributed models and framework," *Comput. Ind.*, vol. 57, no. 1, pp. 39–55, Jan. 2006.

[7] Y. J. Chen, Y. M. Chen, and H. C. Chu, "Enabling collaborative product design through distributed engineering knowledge management," *Comput. Ind.*, vol. 59, no. 4, pp. 395–409, Apr. 2008.

[8] W. Sun *et al.*, "Knowledge-intensive support for product design with an ontology-based approach," *Int J Adv Manuf Technol*, vol. 48, pp. 421–434, 2010.

[9] G. Peng, H. Wang, H. Zhang, Y. Zhao, and A. L. Johnson, "A collaborative system for capturing and reusing in-context design knowledge with an integrated representation model," *Adv. Eng. Informatics*, vol. 33, pp. 314–329, Aug. 2017.

[10] M. A. Robinson, "An empirical analysis of engineers' information behaviors," *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, no. 4, pp. 640–658, 2010.

[11] M. Alavi and D. E. Leidner, "Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," 2001.

[12] A. Barão, J. B. de Vasconcelos, Á. Rocha, and R. Pereira, "A knowledge management approach to capture organizational learning networks," *Int. J. Inf. Manage.*, vol. 37, no. 6, pp. 735–740, Dec. 2017.

[13] L. Kanapeckiene, A. Kaklauskas, E. K. Zavadskas, and M. Seniut, "Integrated knowledge management model and system for construction projects," *Eng. Appl. Artif. Intell.*, vol. 23, no. 7, pp. 1200–1215, Oct. 2010.

[14] J. Girard, M. Georgia, and S. College, "Defining knowledge management : Toward an applied compendium," *Online J. Appl. Knowl. Manag.*, vol. 3, no. 1, pp. 1–20, 2015.

[15] M. Shin, T. Holden, and R. A. Schmidt, "From knowledge theory to management practice: Towards an integrated approach," *Inf. Process. Manag.*, vol. 37, no. 2, pp. 335–355, 2001.

[16] J. Liebowitz, “Knowledge management and its link to artificial intelligence,” *Expert Syst. Appl.*, vol. 20, pp. 1–6, 2001.

[17] P. Heisig, “Harmonisation of knowledge management-comparing 160 KM frameworks around the globe,” *J. Knowl. Manag.*, vol. 13, no. 4, pp. 4–31, 2009.

[18] A. Apurva and S. M.D., “Understanding Knowledge Management: a literature review,” *Int. J. Eng. Sci. Technol.*, vol. 3, no. 2, pp. 926–939, 2011.

[19] L. G. A. Beesley and C. Cooper, “Defining knowledge management (KM) activities: towards consensus,” *J. Knowl. Manag.*, vol. 12, no. 3, pp. 48–62, 2008.

[20] M. Al-Emran, V. Mezhuyev, A. Kamaludin, and K. Shaalan, “The impact of knowledge management processes on information systems: A systematic review,” *International Journal of Information Management*, vol. 43. Elsevier Ltd, pp. 173–187, 01-Dec-2018.

[21] J. Zhang, W. Zhao, G. Xie, and H. Chen, “Ontology-based knowledge management system and application,” in *Procedia Engineering*, 2011, vol. 15, pp. 1021–1029.

[22] A. Saito, K. Umemoto, and M. Ikeda, “A strategy-based ontology of knowledge management technologies,” *J. Knowl. Manag.*, vol. 11, no. 1, pp. 97–114, 2007.

[23] A. H. H. Ng, M. W. Yip, S. binti Din, and N. A. Bakar, “Integrated Knowledge Management Strategy: A Preliminary Literature Review,” *Procedia - Soc. Behav. Sci.*, vol. 57, pp. 209–214, Oct. 2012.

[24] W. Sun *et al.*, “Knowledge-intensive support for product design with an ontology-based approach,” *Int. J. Adv. Manuf. Technol.*, vol. 48, pp. 421–434, 2010.

[25] V. Premkumar, S. Krishnamurty, J. C. Wileden, and I. R. Grosse, “A semantic knowledge management system for laminated composites,” *Adv. Eng. Informatics*, vol. 28, no. 1, pp. 91–101, 2014.

[26] T. Sissoko, M. Jankovic, C. J. J. Paredis, and E. Landel, “An empirical study of a dcision-making process supported by simulation in the automotive industry,” in *Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE*, 2018.

[27] N. Fatfouta, J. Stal-Le Cardinal, and C. Royer, “Empirical Study of Car Crash Simulation Analysis within the Development Phase,” in *22nd International Conference on Engineering Design ICED*, 2019, pp. 2843–2852.

[28] N. Fatfouta, A.-M. Hein, J. Stal Le-Cardinal, and E. Delacou, “An ontology towards a knowledge-based support of the post-processing phase of car crash simulation,” *Adv. Eng. Informatics*, vol. submitted, 2019.

[29] A. Saeema, K. Sanghee, and K. M. Wallace, “A Methodology for Creating Ontologies for Engineering Design,” *J. Comput. Inf. Sci. Eng.*, vol. 7, no. 2, pp. 132–140, 2007.

[30] N. Fatfouta and J. Stal-Le Cardinal, “Towards a framework for integrated and collaborative knowledge management for engineering design- A case study,” in *design 2020*, 2020.

APPENDIX

```
# -*- coding : utf-8 -*-
import os
import xlrd
import numpy as np
import pandas as pd


def search_children(data, raw_val, name):
    """    Compute the bayesian table and extract only the line matching our input case.
    The algorithm will do :
        - filter data to match input case
        - search all solutions  in the filtered data
        - compute statistical weight of this solutions to extract bayesian table
        - propose the table to the user
        - save user's choice into the table to have dynamic/learning beyesian table
    :param data: Table containing the data of the level
    :param raw_val: test scenario input or parent node name
    :param name: name of the children layer as stated in the database
    :return: solutions with wieghts for the level considered    """
    cols = list(data.columns)
    parameter_cols = get_parameters_cols(data, name)
    filter_cols = ['category'] if name == 'cause_scope' else parameter_cols
    tmp_data, input_scenario = filter_data_with_case(data, raw_val, cols, filter_cols, parameter_cols)

    # compute dummies to transform categorical values into binary columns that we will use to compute bayesian
table.
    tmp_data = pd.get_dummies(tmp_data, columns=[name])

    # Compute the occurence of each solution against all the time similar cases appear
    group_poids = tmp_data.groupby(filter_cols).sum()
    group_count = tmp_data.groupby(filter_cols).count()
    bayesian_table = group_poids.div(group_count, level=parameter_cols) * 100

    # technical line (groupby messes the index)
    bayesian_table = bayesian_table.reset_index()
    # post-treating to get the result in the desired format
    if bayesian_table.empty:
        # This means we have no data matching the case scenario
        result = input_scenario.to_dict()
        print("This is a new case, we have no data to help you, contact your team expert for support")
        return result, None
    # Combine bayesian table to the data to predict to add solutoins + weights in the scenario
    join = pd.merge(bayesian_table, input_scenario, on=filter_cols).drop([name], axis=1)
    # Technical : cleaning for easier manipulation and readability
    result = join.to_dict()

    childrens = {key[len(name)+1:]: round(value[0], 2) for key, value in result.items()
                 if key[0:len(name)] == name}
    return childrens


def add_new_test_scenario(name):
    table = pd.read_excel('database.xlsx')
    print("input needed are {}".format(table.columns))
    input_line = {}
    cols = get_parameters_cols(table, name)
    cols.remove('category')
    for col in cols:
        data = input('What is the value for {}?'.format(col))
        input_line[col] = data or None
    return input_line


def capture_desing_issue(name):
    """    ask for values to put in the Db    """
    table = pd.read_excel(name)
    print("input needed are {}".format(table.columns))
    input_line = {}
    for col in table.columns:
        if col != 'category':
            data = input('What is the value for {}?'.format(col))
            input_line[col] = data or None
    input_line = find_category(input_line)
    validation_and_add_to_db(input_line, name, table)
    return input_line


def cut_database_into_levels(database, category):
    database = database.loc[database.category == category]
    cause_scope = database[["impact_config","performance","criterion","target","scope_crtierion",
                            "appreciation","category","cause_scope"]]
    behaviour_gap = database[["cause_scope", "expected_behaviour", "behaviour_gap"]]
    elementary_cause = database[["behaviour_gap", "elementary_cause" ]]
    corrective_action = database[["elementary_cause", "corrective_action"]]
    return {"cause_scope": cause_scope, "behaviour_gap": behaviour_gap,"elementary_cause": elementary_cause,
```

```
            "corrective_action":corrective_action}


def get_parameters_cols(df, name):
    """ quickly filter the input columns based on list order. Can be specified for each table"""
    for i, val in enumerate(df.columns):
        if val == name:
            return list(df.columns[:i])


def filter_data_with_case(data, raw_val,cols, filter_cols, parameter_cols):
    # filter on cases with the same scenario (inputs = values in index columns)
    if type(raw_val) == str:
        # Technical for iterations over
        tmp_data = data[data[cols[0]] == raw_val]
        input_scenario = pd.DataFrame([[raw_val]+[None]*(len(cols)-1)], columns=cols)
        if 'expected_behaviour' in cols:
            input_scenario = find_behaviour(input_scenario)
        elif len(parameter_cols) > 1:
            # other case not possible in our databases but we ask the user if it happens
            print('it seems that we are missing context data')
            print(input_scenario)
            for col in parameter_cols[1:]:
                txt = input("what value do you want for {} ? ".format(col))
                input_scenario[col][0] = txt
    else:
        tmp_data = data.copy()
        # first table scenario
        input_scenario = raw_val
        for col in filter_cols:
            val = input_scenario[col][0]
            # Filter over all the columns that have the same parameters as the case scenario input
            tmp_data = tmp_data[tmp_data[col]==val]
    return tmp_data, input_scenario


def get_test_scenario(data):
    """     getter of the test scenario.
    we let 2 possibilities to the user :
        - one case is already open (so you can re-use it without manually entering values
        - manual insertion of the values of the case
    the open cased is modelled as the last line of the db with only the "issue" / input parameters filled and empty
out
    :param data: full database
    :return: database without the open case, case as df of one line    """
    if not type(data["cause_scope"][len(data)-1]) == str:
        print("New study case found, we will try to find possible solutions")
        data.loc[len(data)-1, "cause_scope"] = None
        input_scenario = data.iloc[len(data)-1].to_dict()
        data.drop(data.tail(1).index, inplace=True)
    else:
        print("No new scenario waiting in db adding new scenario")
        input_scenario = add_new_test_scenario("cause_scope")
    if "category" in input_scenario and type(input_scenario["category"]) not in [str, list]:
        input_scenario = find_category(input_scenario)
    input_scenario = pd.DataFrame(input_scenario, index=[0])
    return data, input_scenario


def find_category(input_scenario):
    """ use the classifier to add category to the input scenario"""
    classifier = pd.read_excel("parts" + ".xlsx")
    classes = list(classifier.loc[classifier.parts == input_scenario['scope_crtierion'], "category"])
    if classes :
        classes = classes[0]
    else :
        print("You are trying to insert a new part, caontact the expert to add this part in the parts datbase.")
    input_scenario["category"] = classes

    return input_scenario


def find_behaviour(input_scenario):
    """ use the classifier to add behaviour to the input scenario if not filled    let the user defined which one is
applied if there can be multiple behaviours for one part """
    classifier = pd.read_excel("parts" + ".xlsx")
    behaviours = list(classifier.loc[classifier.parts == input_scenario['cause_scope'][0], "behaviours"])
    behaviours = behaviours[0].split(', ')
    if len(behaviours) > 1:
        print('We need more info about the behaviour of the possible problematic piece :
{}'.format(input_scenario['cause_scope'][0]))
        print('Behaviours can be {}'.format(behaviours))
        txt = input("what value do you have as expected_behaviour ? ")
        input_scenario['expected_behaviour'][0] = txt
    else:
        input_scenario["expected_behaviour"][0] = behaviours[0]
    return input_scenario
```

```
def build_children_layer(children_nodes, children_layer, parent_node_weight):
    if children_nodes is not None:
        for key, val in children_nodes.items():
            try:
                children_layer[key] += val * parent_node_weight / 100
            except KeyError:
                children_layer[key] = val * parent_node_weight / 100
    return children_layer


def validation_and_add_to_db(result, name, data):
    validation = input("this line will be added to the database, do you validate ? (yes/no) : {}".format(result))
    if validation == "yes":
        final_data = pd.concat([data, pd.DataFrame(result, index=[0])]).reset_index()[list(data.columns)]
        final_data.to_excel(name, index=False)
        return final_data
    else:
        return None


def find_solutions(levels_of_db, test_case_scenario, last_layer):
    # get all possible cause_scope for level 0
    final_solutions = {}
    cause_scope_layer = search_children(levels_of_db["cause_scope"], test_case_scenario, "cause_scope")

    network_layers = ["behaviour_gap", "elementary_cause", "corrective_action"] if last_layer == "solutions" else
["behaviour_gap", "elementary_cause"]
    for studied_cause_scope, cause_scope_weight in cause_scope_layer.items():
        parent_layer = {studied_cause_scope: cause_scope_weight}
        for layer in network_layers:
            children_layer = {}
            for parent_node, parent_node_weight in parent_layer.items():
                children_nodes = search_children(levels_of_db[layer], parent_node, layer)
                children_layer = build_children_layer(children_nodes, children_layer, parent_node_weight)
            # sort to have most probable
            parent_layer = {k: v for k, v in sorted(children_layer.items(), key=lambda item: item[1], reverse=True)}
        final_layer = {key: round(value, 2) for key, value in parent_layer.items()}
        final_solutions[studied_cause_scope] = final_layer
        print("solutions for cause {} are : {}".format(studied_cause_scope, parent_layer))
    print(final_solutions)
    return final_solutions


def filter_similar(table, category):
    res = table.loc[category == table.category]
    other_filter = input("do you want to add another filter ?")
    while other_filter == "yes":
        print("the possible columns to filter are {}".format(list(table.columns)))
        precise_filter = input("write the filter like this :  'columns : value'").split(" : ")
        res = res.loc[precise_filter[1] == res[precise_filter[0]]]
        other_filter = input("do you want to add another filter ?")
    # res = pd.merge(table, test_case_scenario, on="category", how='inner')
    print(res)
    return res


def capture_test_scenario(database):
        data = pd.read_excel(database)
        db, test_case_scenario = get_test_scenario(data)
        levels_of_db = cut_database_into_levels(db, test_case_scenario['category'][0])
        return db, levels_of_db, test_case_scenario


if __name__ == "__main__":
    txt = input("What do you want to do? (add design issue, find solutions, find causes, find similar inputs)")
    if txt == "design issue" or "add design issue":
        capture_desing_issue('database.xlsx')
    else:
        db, levels_of_db, test_scenario = capture_test_scenario('database.xlsx')
        if txt == "solutions" or "find solutions":
            find_solutions(levels_of_db, test_scenario, "solutions")
        elif txt == "causes" or "find causes":
            find_solutions(levels_of_db, test_scenario, "causes")
        elif txt == "similar inputs" or "find similar inputs":
            category = test_scenario['category'][0]
            filter_similar(db, category)
        exit("closing app")
```