View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Open Archive Toulouse Archive Ouverte

Knowledge Formalization in Experience Feedback Processes : An Ontology-Based Approach

B. Kamsu Foguem[a], T.Coudert[a], C.Béler[a], L.Geneste[a]

[a] LGP Laboratoire Génie de Production - Ecole Nationale d'Ingénieurs de Tarbes
47, avenue Azereix - BP 1629, F-65016 Tarbes Cedex
Tel : + 33 (0)5 62 44 29 43 Fax : + 33 (0)5 62 44 27 08
E-mail: {Bernard.Kamsu-Foguem, Thierry.Coudert, cbeler, Laurent.Geneste }@enit.fr

Abstract

Because of the current trend of integration and interoperability of industrial systems, their size and complexity continue to grow making it more difficult to analyze, to understand and to solve the problems that happen in their organizations. Continuous improvement methodologies are powerful tools in order to understand and to solve problems, to control the effects of changes and finally to capitalize knowledge about changes and improvements. These tools involve suitably represent knowledge relating to the concerned system. Consequently, Knowledge Management (KM) is an increasingly important source of competitive advantage for organizations. Particularly, the capitalization and sharing of knowledge resulting from experience feedback are elements which play an essential role in the continuous improvement of industrial activities. In this paper, the contribution deals with semantic interoperability and relates to the structuring and the formalization of an Experience Feedback (EF) process aiming at transforming information or understanding gained by experience into explicit knowledge. The reuse of such knowledge has proved to have significant impact on achieving the missions of companies. However, the means of describing the knowledge objects of an experience generally remain informal. Based on an experience feedback process model and conceptual graphs, this paper takes domain ontology as a framework for the clarification of explicit knowledge and know-how, the aim of which is to get lessons learned descriptions that are significant, correct and applicable.

Keywords: Interoperability, Continuous improvement, Knowledge Management, Experience Feedback, Formal Ontology, Conceptual Graphs

1. Introduction

Because of the current trend of integration and interoperability of industrial systems, their size and complexity continue to grow making it more difficult to analyze, to understand and to solve the problems that happen in their organizations. Classical hierarchical and stable organizations are progressively replaced by distributed, networked and unstable ones, implying deep changes. Organizations have to adapt to this distributed and often ephemeral context. So, continuous improvement methodologies developed since many years in enterprises are still topical questions. They are powerful tools in order to understand problems, to solve them, to control the effects of changes and finally, to capitalize explicit knowledge about changes and improvements. These tools require to suitably represent knowledge relating to the concerned system, its environment, its missions and the situations in which this system evolves [1].

However, the ongoing distributed nature of enterprises leads to new requirements concerning interoperability (enterprises have to co-operate in order to reach global objectives - see section 1.2). On the other hand, continuous improvement and problem solving methodologies have to be adapted to these new configurations.

Considering this context, the new requirements about continuous improvement and enterprise

interoperability are successively described in the following sections 1.1 and 1.2.

1.1. *Continuous improvement requirements*

Continuous improvement constitutes a major aspect of the family of standards ISO 9000 maintained by ISO (International Organization for Standardization). It focuses on improving customer satisfaction through continuous and incremental improvements to products, services and processes. In order to meet requirements implied by continuous improvement, one key point is to optimize the problem solving process. This process is started when a negative event (i.e. with a negative impact on the client or on the organization) occurs. It aims at analyzing and solving the current problem then to avoid its reemergence.

Several methods have been set up in order to organize the problem solving processes. One of the most widely used is the plan-do-check-act (PDCA) cycle, also known as Deming Cycle. Several other frameworks are commonly used such as 8 Disciplines (8D) also called TOPS (Team-Oriented Problem Solving), Six Sigma DMAIC (Define, Measure, Analyze, Improve and Control), 7-step, etc.

The main actions in these processes are:

- to form a problem resolution team,
- to describe and evaluate the event criticality,
- to analyze the event, to search for root causes and to validate the analysis,
- to propose a solution to the problem and apply it (curative solution),
- to suggest actions to avoid another occurrence of the problem (lessons learned, preventive solution)

The proposed EF frame is a generic representation of industrial problem solving methods where five main information slots can be distinguished: event, context, analysis, solution and lessons learned. Figure 1 shows the structure of the main problem solving processes and a mapping between the corresponding activities and the information slots that compose an experience. This mapping enables to put the emphasis on the links between problem solving processes in industrial organizations and experience feedback capitalization. It can be considered as a high-level guideline for practical experience feedback implementation.

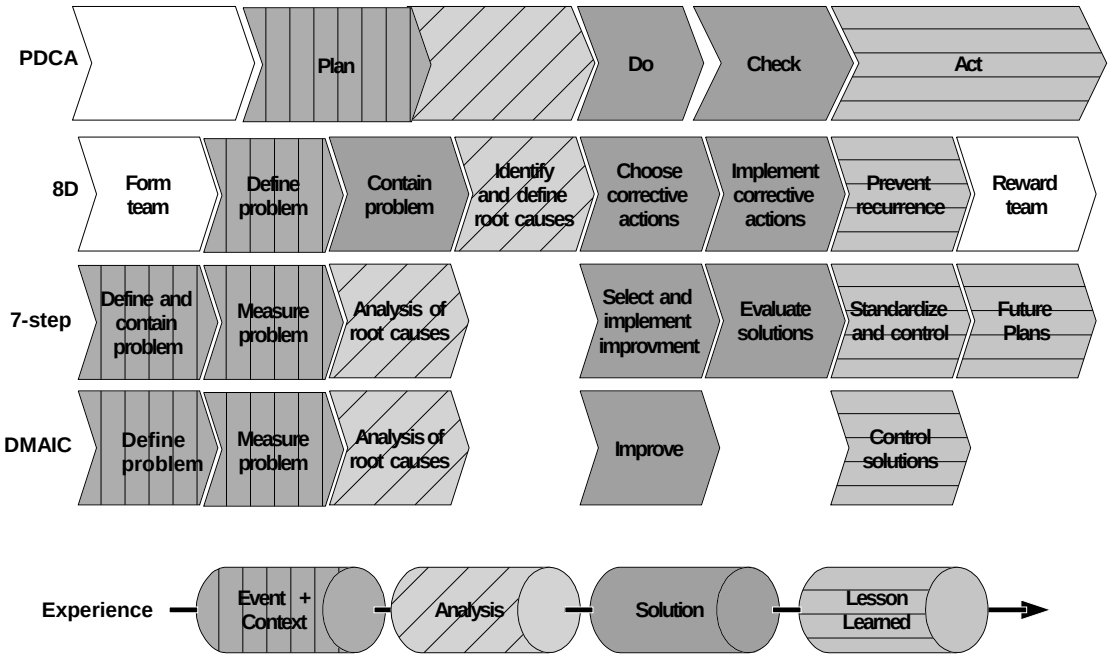

Fig. 1: Mapping between "Problem solving processes" and the proposed experience model

1.2. *Interoperability*

Many companies are getting away from tight application-to-application interfaces as well as traditional enterprise application integration because of the too monolithic resulting systems [2]. These approaches are more and more replaced by service oriented, loosely coupled, message-based, and asynchronous techniques within networked organizations. The main characteristics of these organisations are: "virtuality" (enterprises are gathered for ephemeral co-operation in order to respond to market challenges); distributed control; inter-organizational business processes (business processes cross the entire organization); various supply chains, shared information and knowledge. In this context, enterprise interoperability is a key

factor. IEEE [3] defines interoperability as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged".

Let us explain the links between interoperability and Experience Feedback developed in this work. Because of integration and interoperability requirements in industrial systems, the size and complexity of problems continue to grow. To be able to analyze, to understand and to solve the technical problems turn into a challenge. The quantity of information to model, to process, to store, to exchange and to analyze becomes more and more important.

- Let us consider a centralized industrial system without interoperability requirements. In such a system, the benefits of carrying out an EF process (see section 1.1) are usually well known and many enterprises have implemented it (at least partially in maintenance and quality services for instance – See examples of industrial applications in [4], [5]). In this situation, the most important is to have models based on ontology as well as problem solving processes as proposed in this article.
- Let us consider now a networked (or distributed) organization with strong interoperability requirements. The quantity of information becomes very important and applications are usually distributed. The occurrence of a technical problem at a node of the networked organization and its resolution generate information. This information has to be transformed into explicit knowledge in order to be reused by other entities when similar problems will arise in the future. Rather than solving the problem locally using problem solving techniques, this paper proposes an EF process that enables not only to solve problems and to capitalize knowledge, but also to share knowledge with the entire organisation. The proposed methodology is based on the idea that a better interoperability can be reached if different actors have guidelines for knowledge capitalization and exploitation using a common ontology. Analysis and solutions carried out can be reused each time a similar problem is detected. The interoperability for which the proposed EF process contributes is a semantic one. It concerns the ability to understand the content of exchanged messages by senders as well as by receivers [6]. Each of them must understand the knowledge capitalized by the others.

Therefore, the proposed EF process participates to the reduction of the complexity induced by the interoperability requirements. The corollary is that carrying out an EF process participates to the integration and then, favours a better interoperability because the system will be aided to share information and knowledge.

It is important to notice that this paper does not focus on "how to reach interoperability within an organization". It provides tools and methodological considerations aiming at supporting interoperability of networked organisations. The two following hypothesis are assumed:

i) To be able to capitalize and to share knowledge about problem solving participates to integration and to interoperability because this knowledge can be distributed among further actors or entities;

ii) Several distributed manufacturing units gathered with a common goal of performance can better interoperate if they are able to deploy a wide continuous improvement methodology using a predefined framework based on a common ontology of the domain and a common ontology of experience feedback.

The paper is structured as follows. Section 2 exposes the state of the art concerning Knowledge Management and Experience Feedback approaches. Some comparisons between common architectures like KADS/CommonKADS and the Experience Feedback ones are discussed. In section 3, the proposed experience feedback framework is presented. Section 4 explains how the vocabulary is structured defining an ontology. The definition of ontology and its importance in this work concerning interoperability and problem solving are described as well as the justification of the conceptual graphs paradigm as processing support for ontology. Section 5 describes the proposed experience feedback process. Knowledge formalization within it is described in Section 6. An illustrative example is exposed in section 7. Finally, section 8 concludes and discusses future challenges.

2. State of the art

Knowledge management (KM) may refer [7] to the ways organizations gather, manage, and use the knowledge that they acquire. The term also designates an approach to improving organizational outcomes and organizational learning by introducing into an organization a range of specific processes and practices for identifying and capturing knowledge, know-how, expertise and other intellectual capital, and for making such knowledge assets available for transfer and reuse across the organization [8]. Generally, the two major challenges in a knowledge management process of an organization are the capitalization and the exploitation [9]. The former, knowledge capitalization is the process which allows reusing, in a relevant way, a given domain knowledge previously stored and modeled, in order to perform new tasks [10]. The latter, knowledge exploitation, is the dissemination of knowledge to serve current practice and to train future practitioners.

Particularly, Experience Feedback (EF) approach [11] is a knowledge management initiative which objective is to convey experiential knowledge or lessons learned applicable to an operational, tactical,

or strategic level such that, when reused, this knowledge positively impacts on the results of the organization. Enterprises turn towards Experience Feedback processes to avoid reproducing past mistakes and to benefit from all the knowledge and the know-how used and produced within them. Experience feedback models are typically tied to specific organizational objectives and are intended to lead to the achievement of specific targeted results. In the literature, different approaches can be found: experiential learning [12], lessons learned systems [13], experience feedback loop [14]. There are two main limitations of these models: first, an imprecise description of the vocabulary related to knowledge models and second, a lack of formal tools allowing rigorous models analysis.
Section 2.1 exposes how EF is in line with Knowledge Based Systems, section 2.2 introduces languages and modeling for problem solving with the needed requirements and section 2.3 shows why conceptual graphs are a relevant paradigm to express and process experiences.

2.1. *Experience Feedback versus Knowledge Based Systems*

Differences with usual KM methodologies like CommonKADS [15] are interesting to consider in order to better understanding EF. CommonKADS was actually made from previous KBS methodologies, especially KADS (Knowledge Acquisition and Design Structuring). Indeed, it is often considered as a standard of KBS methodology. Its process consists in capturing and writing down the knowledge people are using to do a specific task. It actually regards the construction of a KBS as a modeling process. Once a kind of application is selected, a series of models are developed, which gradually transform real-world requirements and expertise into a system implementation. The main drawbacks of such approaches are:

- the high level of abstraction of models makes them difficult to be adopted,
- the time and human resources consumptions are important [16],
- the intervention of a knowledge engineer whose role is to help the experts to describe their knowledge is required,
- the extracted knowledge is weakly contextualized,
- the knowledge maintenance and update require regular knowledge acquisition sessions.

EF is another way to manage knowledge. It is a bottom-up approach, where knowledge is built gradually from useful cases. Generic knowledge can be extracted but above all, EF is a way to ensure partial knowledge preservation. Actually, given the classical hierarchy Data, Information, Knowledge, experience is halfway between information and explicit knowledge (knowledge that is formalized). The gradual transformation is done in three steps. First the event and its context are described (Information level) then analysis and solution are capitalized (Experience Level). The Knowledge level is reached when lessons learned, procedures, invariants, rules… are inferred from past experiences.
To summarize, one major EF advantage resides in the contextualization of knowledge which makes it useful for practical needs. Since information and fragments of knowledge are captured as they arise, it requires less manpower and time, and the experience base is gradually updated.
EF consists in capitalizing a particular problem resolution process which will be used again partially or totally whereas KADS capitalizes the generic knowledge necessary to solve a problem through models of expertise. Actually, the main objective of KADS is more general than solving an experience feedback problem.

One major interesting aspect taken from KADS in the EF framework realization approach is the formalization degree and the unambiguous modeling. Actually, there are industrial applications [17] [18] of EF consisting in the capitalization of event, context, analysis and solution but they do not have enough structuring capabilities and often remain only descriptive. Each container is described with free text, picture or video. It allows for instance to create experience booklets and, whereas it is a very good approach in order to give explanations, it becomes difficult when it comes to knowledge extraction and reuse. This is why the outlined EF approach insists in the formalization, the data structuring in order to be able to computerize the EF process and better extract knowledge.

2.2. *Languages and modeling for problem solving*

Knowledge-based systems generally include a complex knowledge base and an inference engine which uses this knowledge to solve a given problem. Languages for knowledge-based systems have to cover both aspects with means to describe knowledge about the domain and knowledge about how to use this domain knowledge in order to achieve the task assigned to the system [19]. There are several formal languages and modeling paradigms developed to support knowledge- based systems (KBSs), with three traditions:

- formal specification languages have spawned many purpose specification languages: operational languages (e.g. Prolog), algebraic specification techniques (e.g. TFL [20]) or model-based approaches (like B [21] and DESIRE [22]) which describe a system in terms of states and operations working on these states.

- reactive system modeling (e.g Statecharts [23]) that offers a framework in which changes between (complex) states can be specified.
- formal conceptual modeling (like Troll [24] and Conceptual Graphs [25]) which is concerned with capturing real-world knowledge and focuses on modeling domain entities, activities or agents (ontological knowledge) used to make assertions.

In accordance with the requirements presented by Baader in [26], the presented work is based upon Knowledge Representation languages which have the following characteristics:
- they have a declarative semantic: the knowledge represented should be defined independently to the programs processing the knowledge base.
- they are logically founded: the correctness of an inference mechanism should be defined relatively to logics, and more specifically to logical deduction.
- they support knowledge structuration: a Knowledge Representation language should provide a way to structure knowledge, such as the differentiation between ontological and assertional knowledge. Another aspect of knowledge structuring is that semantically related pieces of information should be gathered together. Hence, it is possible to really report as much as possible experiences and solution(s).

With respect to these requirements, the proposed work is concerned with formal conceptual modeling approach because it provides means to understand the application domain (modeling schemes capture key experiences concepts), hence it is possible to build models of humans' knowledge/beliefs about the world. Furthermore, formal conceptual modeling approach typically uses first order predicate logic as the underlying formalism and makes use of abstraction and refinement as structuring primitives. Particularly, for these reasons conceptual graphs are helpful in the context of experience feedback. Besides, on the basis of several criteria (expressive power, reusability, formal precision) of a method for language comparison given in [27], the ontology with conceptual graphs approach undertaken in this paper appears very interesting for problem solving. Indeed, the properties (e.g. formal semantics, separation of knowledge types and possible translations into other languages) of conceptual graphs make them suitable for modeling and specifying experiences feedback processes in which reasoning plays an essential role.

The main requirements for the experience modeling are: to enable a structured conceptual modeling and to support search to facilitate reuse. With respect to these requirements, two major approaches were studied: Case Based Reasoning and Conceptual Graphs. Conceptual Graphs were selected mainly because of their ability to include in a consistent framework modeling and search operations.

2.3. *Conceptual graphs to express and process experiences*

To provide efficiency and integration, EF processes must rely on solid theoretical foundations requiring an appropriate representation language, with clear and well-defined semantics. This enables the explicit and non-ambiguous modeling of relevant knowledge. The representation language must give means to rigorously represent the used vocabulary, to analyze the models or to reason about them directly [15].
Taking this into account, the suggested solution to the quality problem of EF representation relies on the utilization of a formal representation. EF can be carried out at operational level by several techniques. There are usual formalisms like the combination of frame or object oriented language with CBR (Case Based Reasoning) techniques or conceptual graphs for the representation of ontology and projection operation for the inference. Whereas both are quite similar in terms of computational objectives, the choice of Conceptual Graph (CG) was made because the integration of ontology is a basic feature of CG and therefore it constitutes an homogeneous formalism for interoperability requirements. Hence by the native ontology integration, a better interoperability is expected (see section 4 for details).

As a preliminary short definition, a conceptual graph [25] is a directed, finite, connected graph consisting of concepts and conceptual relations. Concepts and relations represent declarative knowledge. Procedural knowledge can be attached through graph operations. An essential point is that Conceptual Graphs support different representation formats (graphical representation, internal Prolog representation or first order predicate calculus). Accordingly, for editing and manipulation of a knowledge base of Conceptual Graphs, the user can choose the most preferable format or can combine the advantages of two different formats [28].
Conceptual graphs language, with its clear and well-defined semantics, can be helpful in both the definition of an adequate experience feedback representation and the development of techniques for the formal analysis of knowledge models. In addition to these advantages, a hypothesis can be made: conceptual graphs have an intuitive structure that can be understood easily. This paradigm is described in detail in section 6.
The idea of this paper is to take formal domain ontology as a foundation for the clarification of knowledge and know-how. Formal domain ontology provides a precise and consensual description of the basic terminology and concepts related to knowledge

capture and codification. With this formal domain ontology, people are better prepared to enhance knowledge sharing, which in turn fosters organizational learning and enriches innovation [29].

3. Proposed approach for experience feedback

The implementation of an experience feedback process relies on the development of a knowledge management framework. Generally, the task of a knowledge management framework is to capture and manage explicit and tacit knowledge of an organization in order to facilitate the access, sharing, and reuse of that information [30]. Knowledge management must be guided by a strategic vision to fulfill its primary organizational objectives: improving knowledge sharing and cooperative work inside the organization; disseminating best practices; preserving past knowledge of the company for reuse; improving the quality of projects and innovations; anticipating the evolution of the external environment; preparing for unexpected events and managing urgency and crisis situations [31].

In practice, a knowledge management framework can be materialized in various manners. A first solution consists in gathering and organizing a set of numerical documents by possibly associating it with information retrieval tools (for example, Documents Management Software). The advantage of this approach is its simplicity of implementation and thus its low cost. Its major drawback is its lack of clarification, which leads to difficulties in knowledge appropriation which, hidden in the documents, must be exhumed during the consultations. As a simple example, consider a company's employee database. Much explicit information can be retrieved about specific attributes of employees, but there is plenty of knowledge which remains implicit; for example, the technical or strategic knowledge used everyday inside the company. On the other hand, a second solution consists of capturing knowledge requirements in organizations through conceptual modeling which will be the heart of the Knowledge Management [15]. The disadvantage from this point of view is the generated cost: computational modeling is a heavy component in each problem. Its advantage is to allow the clarification and the structuring of knowledge and know-how, which facilitates at the same time the diffusion, the evolution and the relevance of knowledge.

In this work, the second point of view is advocated by considering that a framework of experience feedback should rely on the conceptualization of domain vocabulary and relevant knowledge relating to the activities of an organization. The objective is to explicitly represent experiential knowledge in an organization, while allowing its access and re-use by the organization members for their tasks. For doing this, conceptual graphs are used. They enable the users to visualize and understand the details of knowledge modeling.

Experience Feedback processes (capitalization and reuse of past solutions) can be analyzed from a KADS/CommonKADS perspective. In CommonKADS three layers or levels of knowledge are considered:

- the *domain layer* consists of the enumeration of concepts and their relationships. This level, often called ontology, materializes the domain knowledge;
- the *inference layer* can be seen as a library of problem solving methods and processes described in a declarative manner;
- the *task layer* provides a procedural interpretation of the inference layer.

More recently, a similar perspective was used to structure the Unified Problem-solving Methods Language (UPML) [32]. In UPML, three major component types are defined: task, domain model and problem solving method (PSM). Moreover, several bridges enable connections between components, an ontology being at the core of the framework to define a terminology and its properties.

For EF, the *Domain layer* consists of the domain ontology (domain knowledge) enriched by the ontology describing the experience feedback framework. The *Inference layer* consists of mechanisms aiming at capitalizing, retrieving and reusing experiences. Several techniques are explored for the search (based on similarity, adaptability) and for the adaptation phase (transformational, generative, compositional, hierarchical as explained in [11] – chapter 8). In this paper the projection operation of conceptual graph formalism [33] is used as a common technique for the search and the adaptation phase (see section 6 for details). Finally, the *Task Layer* consists of the actual process of experience feedback. Capitalization and Exploitation are the two main sub-processes. Capitalization is based on the industrial problem solving method as introduced in section 1.2. Each step is a capitalization sub-process (event description, context description, analysis and solution determination). Exploitation is based on the following sub-processes: retrieval, adaptation and generalization. These steps are the core techniques that support the EF problem solving cycle and have been inspired by the case-based reasoning cycle [34] [35]. Although CBR can be considered as very close to EF, it is originally not regarded as an organizational model for experience reuse, but mainly as a cognitive model and a technical architecture (see [11] – chapter 2).

The approach is structured in two steps: firstly, during knowledge capitalization, there is a formal modeling of knowledge, rules or heuristic associated to positive and negative events; afterward, there is the use of conceptual graphs operations founded on computing techniques to support the appropriation

and the dissemination of capitalized knowledge.
The activities through which the assessment of this approach has been carried out are the following (see figure 2):

- **Domain representation**: the domain ontology of the target enterprise or organization is formalized. This formalization is in itself a conceptual and terminological clarification activity during which more or less vague concepts must be brought to an expression devoid of any ambiguity. The ontology layer supports the evolution of vocabularies as it can define relations between the different concepts and expresses a community's consensus knowledge about a domain.
- **Capitalization**: capitalization involves all the activities making it possible to add new information into the Experience Feedback Base. These activities are organized for the creation of knowledge that will be the most effective in supporting the improvement of quality products and services delivered by the system. This knowledge describes fundamental facts and rules coming from the experience feedback and is generally issued by a pluridisciplinary committee.
- **Knowledge Formalization:** the knowledge analysis requires the translation of cases or lessons learned of the previous phase into a formal specification expressed in conceptual graphs formalism. Based on the formal reasoning of conceptual graphs, the analysis techniques should allow for the determination of consistency (no contradictions) and correctness (objectives are satisfied) of experiential knowledge. In section 5, the detailed process is presented.

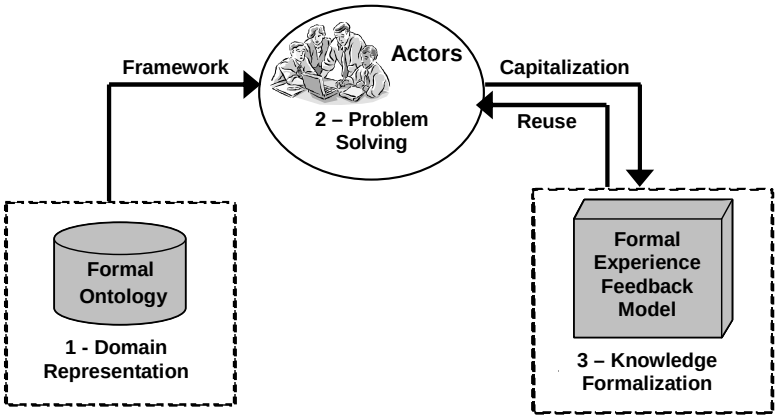

Fig. 2: Proposed Methodology for experience feedback process

4. Structuring the vocabulary with ontology

An experience feedback system is a knowledge base containing a set of experiential elements (characterizing knowledge resulting from the analysis of the events). The main problem thus consists in the organization of these elements in order to be able to enrich them and use them easily. These experiential elements could effectively be used only if the different actors share a common understanding of domain vocabulary. Consequently, it is necessary to structure the appropriate vocabulary which will be used to describe each experiential element.
Traditionally, the characterization of experiential terms is limited to simple taxonomies that lack constructs needed by analyst to reason over an instance representative of domain knowledge. Ontologies provide [36] powerful constructs with the ability to share a common understanding of experience thus enabling people to better reason over and analyze experiences.

4.1. *Definition of Ontology*

In philosophy, ontology is the study of the nature of Being and the essence of things. In the early 1990s computer scientists, particularly those in Artificial Intelligence, gave to the term a new, but related, meaning. It is possible to find in the literature several definitions of ontology. The most quoted one is proposed by Gruber [37]: *an ontology is a formal, explicit specification of a shared conceptualization*. This definition identifies four main concepts involved: an abstract model of a phenomenon termed "conceptualization", a precise mathematical description hints the word "formal", the precision of concepts and their relationships clearly defined are expressed by the term "explicit", and the existence of an agreement between ontology users is hinted by the term "shared" [38].

Some essential aspects of ontologies are:
- they are used to describe a model of a specific domain,
- their concepts and relations are unambiguously

and formally defined by axioms and definitions stated in a formal language, such as logic or some computer-oriented notation (e.g. conceptual graphs) that can be translated to logic,
- there is a mechanism to organize the concepts by means of relationships, which might be hierarchical or non-hierarchical,
- there is an agreement between users of an ontology in such a way that the meaning of the concepts is used consistently by all of them.

Intuitively, a conceptualization can be considered as given by a set of features constraining the structure of a piece of reality, which an agent uses in order to isolate and organize relevant objects and relevant relations [39]. A set of formal constraints, expressed in a suitable formal language, can therefore be used to (partially) characterize a conceptualization. An ontology is a rigorous representation of concepts and their allowed interactions, with the purpose of providing an explicit framework in which to elaborate the experience feedback modeling.

4.2. *Roles and Uses of Ontology*

Scientific ontologies are being developed and used in disciplines ranging from biology and medicine [40] to enterprise modeling [41] and knowledge management [42]. The role of ontologies as facilitators of knowledge management is being praised stronger than more today, especially since when an important place was attributed to them inside the vision of Semantic Web [43]. Indeed, knowledge management as well as e-business [44] or enterprise application integration [45] [46] are seen as the areas where the use of ontologies and other Semantic Web technologies will strongly contribute to the creation of pragmatic vocabulary allowing computers and/or humans to co-operate in sharing information and in solving problems.

Three main roles are devoted to ontologies in general:

- **Communication** (humans and organizations): In order to meaningfully share the information, an ontology provides a common understanding to reduce or eliminate conceptual and terminological confusion. Such an understanding can help in achieving better communication between people and organizations. Indeed, all ontology users agree on the meaning of involved concepts and their relationships represented in the model of a domain. Moreover, the ontologies are used in the communication with other (external) systems, such as, e.g. user interfaces or other (knowledge) systems in a distributed system [47]. Interfaces of a knowledge system can make use of ontologies to direct users in their responses by explaining reasoning behavior in terms of its ontology [48]. In information retrieval applications, ontologies serve to disambiguate user queries, to elaborate taxonomies of terms or thesaurus in order to enhance the quality of retrieved results [49] [50].
- **Interoperability** (machines and systems): interoperability among systems or machines [51] is achieved by translating between different models, paradigms, languages and software tools. The shared understanding of domain ontology is the basis for a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest. For instance, ontologies are relevant to provide semantic definitions on concepts and constructs, so they allow the semantic matching or mediation that achieves interoperability between enterprise models/tools [52]. The usefulness of ontologies in agent based systems can be pointed out as they enable knowledge interoperation. For example, Orgun and Vu propose an ontology-based multi-agent system that provides a framework for interoperability in heterogeneous medical information systems [53].
- **Reasoning and Problem Solving**: The basic role of ontology in this case is to represent the knowledge of the domain in order to be able to achieve reasoning, that is to say, to represent problems and generate solutions for these problems. This use is found in many expert systems (problem solvers) and decision support systems [54] [55] [56]. In using ontologies for this role, secondary goals are the creation of knowledge bases that are reusable, efficient, explainable, modular, etc. [57]. Indeed, the early use of ontologies in Artificial Intelligence research aimed at improving knowledge engineering by tackling these roles by creating “well structured” knowledge bases that would not only solve the problem at hand but be more maintainable, easier to extend, etc. In this sense, ontologies are a convenient engineering tool [58]. This role of ontologies implies the use of an inference engine that is used to achieve specific goals.

The main paradigms of languages currently used to represent ontologies are conceptual graphs [25], description logics [59] and frame logics (a deductive and object-oriented formalism [60]). These main paradigms are complemented by RDF (Resource Description Framework) [61] and its evolution, the OWL (Web Ontology Language) [62] mainly applied in connection with the “Semantic Web” scheme.

Frame logics are powerful in reasoning about and representing knowledge, but the flexible higher-

order syntax may lead to problems of stratification [63]. Description logics have been a successful attempt to combine well-defined logical semantics with efficient reasoning, but they suffer the problem of explaining specialization. Furthermore, conceptual graphs can be easily translated into the terminology of some other approaches in knowledge engineering, such as RDF [64], [65].

4.3. *Relevance of conceptual graphs for experience feedback ontology formalization*

In this work, ontology is represented in the conceptual graph formalism and used as a tool to support knowledge capitalization and reuse. Considering the three main roles of ontologies described above (communication, interoperability, reasoning and problem solving) a mapping can be done, describing how conceptual graphs are appropriate to the ontology role and, furthermore, to the addressed problematic.

From a **communication viewpoint**, two essential properties are seen in the conceptual graph formalism. The components of the knowledge base, simple graphs, are easily understandable by an end-user (a knowledge engineer, or even an expert). The graphical representation, the mapping to natural language [66], and the explanation mechanism help in expressing and understanding knowledge, which is beneficial for users to construct and manipulate knowledge. And reasoning mechanisms are easily understandable too (at least if the graphs are reasonably small), for two reasons: graph operations (join, projection, see section 6.3 for details) enable the end-user to follow reasoning step by step and the same language is used at interface and computing levels [67].

From an **interoperability viewpoint**, the existence of a standard (such as CoGXML) when the graphs themselves are exchanged facilitates the connection of different knowledge systems that are able to encode or decode conceptual graphs. Some researchers (for example [68]) suggested using conceptual graphs as a pivot language to allow the automatic translation of knowledge structures between different knowledge representation formalisms. Moreover, conceptual graph ontology provides semantic definitions on concepts and constructs that are important in order to match semantic interoperability [52]. This advantage has shown a practical usefulness to support the cooperative work in a network organization [69].

From a **reasoning and problem solving viewpoint**, conceptual graphs are provided with a logical semantics [25]. General problems associated with both kinds of conceptual graph based reasoning are NP-hard [70]. However, some polynomial cases [71] obtained by restricting the structure of the graphs are used in real-world knowledge. On the other hand, considering graphs instead of logical formulas gives another viewpoint (for instance, some notions like path, cycle or neighborhood are natural on graphs) and provides other algorithmic ideas [72]. Conceptual graph formalism provides both a controlled vocabulary of artifacts in the real world, captures the relations between them, and supports various reasoning mechanisms. This vocabulary is used to construct a formal knowledge representation of Experience Feedback cases (the experiences) and lessons learned, then to make some interesting reasoning tasks as explained in the continuation.

5. Modeling of Experience Feedback Process

5.1. *Definition*

Among many proposed definitions for "Experience Feedback", the one given in [5], [17], similar to the definition of "Lessons learned" given in [18], has been adopted.

Experience Feedback is a process of knowledge capitalization and exploitation mainly aimed at transforming understanding gained by experience into knowledge.

A lesson learned from experience must be **significant** in that it has a real or assumed impact on operations; **valid** in that it is factually and technically correct; and **applicable** in that it identifies a specific design, process, or decision that reduces or eliminates potential failures and mishaps, or reinforces a positive result.

In this paper, only the experience feedback capitalization with the aim to capitalize the results of the experiential knowledge and to learn from it is studied.

5.2. *Experience Feedback Capitalization Representation*

The capitalization stage of Experience Feedback (EF) process seeks to capture experiential element available in the organization (including project experiences, problem-solving expertise, and design rationale). The motivations is to exploit the experience acquired from past projects and to keep some lessons from past to avoid reproduction of some mistakes.

The input of the EF process is the occurrence of an unexpected (or “expected” – see section 6.2 – Definition of support) situation (an event) during the life cycle of a product, device or process in the organization. This study only concerns expected events. So, the event expresses an unexpected failure and a solving process is set up as soon as it is detected. For positive events, the rationale of successes is also considered as a source of lessons learned but this issue is not considered in this work.

This process corresponds to a sequence of activities defined according to a process that will lead, in the

best situation, to the resolution of the problem caused by the event occurrence. This study is based on a process defined in [17] and adapted to the addressed problematic. The proposed process (figure 3) is described as follow. First, the event's context and the event itself are described. This task is realized by the operational actors. They can be assisted by a knowledge engineer but the goal is that they would be able to capitalize themselves this information. This aspect can be achieved by the conceptual graphs formalism presented above (see section 4). This context will help later on to retrieve comparable problems in the experience base. Secondly, an expertise is realized by a committee of experts of the domain. The goal is to analyze the problem, to formalize experts analysis, to provide a solution and, if useful, to capitalize it. This whole information represents an experience. Thirdly, it may be judicious to build lessons that will be systematically used in future similar situations. To build them, a pluridisciplinary committee has to be defined in order to treat one or more particular themes (for instance, quantity of quality problems on a product). These lessons generalize and reinforce a set of previous experiences. In order to define a new experience, domain experts can search similar ones in the experience base.

Consequently, in the following, any experience will be described according to four elements: context, triggering event, analysis and solution. Some lesson learned can be derived from several experiences.

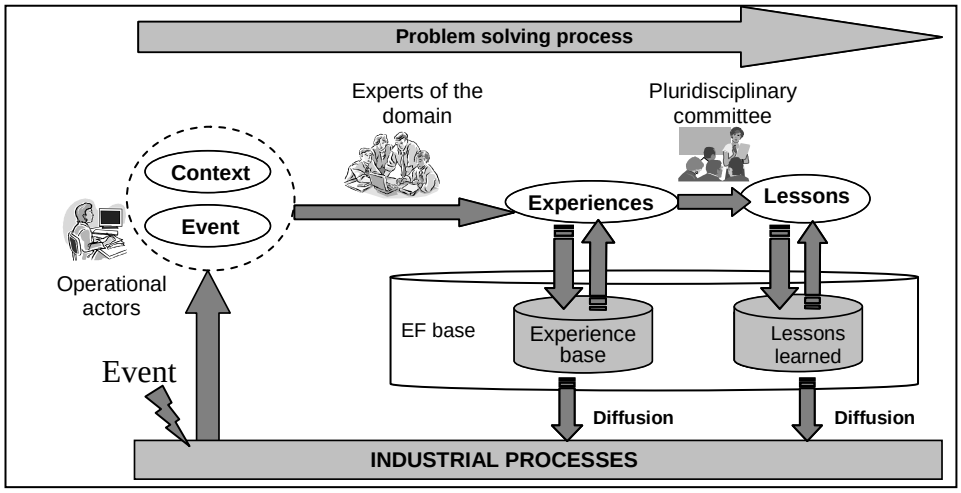

Fig. 3. Global process to capture experience feedback from an event based on [17]

Another more detailed point of view about this process is represented on figure 4 which highlights the information or knowledge reuse (drawn as dotted arrows). Four main tasks have been represented: domain representation, context and event capitalization, analysis and generalization. The domain representation task is achieved off-line by experts of the domain. Their role is to build the formal ontology by means of conceptual graphs. This task is described in the next section. Its outcome constitutes the support of the EF process: the Experience Feedback models. These models can be seen has *components* stored in a library as validated ready-to-use models facilitating reuse [73]. The second task represents context and event capitalization: it consists, for the operational actor, in using predefined EF models to represent the events as well as their context. Obviously, this capitalization requires that the event has occurred and more importantly has been detected. Then, the analysis of the problem has to be realized, aided by the previous capitalized experiences. This task is generally done by experts. The most important point is to capitalize this analysis within a model of experience. The last task concerns the generalization. It consists in building lessons (or rules) from the experiences previously capitalized but also from the previous lessons learned. This is realized by a pluridisciplinary committee and can not be automated. Obviously, tasks 2 and 3 are realized more frequently than the generalization one. Task 2 has to be done as early as it appears; task 3 is done immediately if the problem needs an immediate analysis and solution or it can be postponed to the next meeting of experts (for instance, a meeting can be planned each week). Task 4 generally requires several experiences, and is therefore realized less frequently. For instance, each month, a pluridisciplinary committee can be formed in order to treat particular problems (for instance, recurrent failures on a machine). Obviously, Experiences and lessons learned consist of conceptual graphs and they represent the capitalized knowledge. The next section gives some definitions about this paradigm and describes how the formal ontology is built.

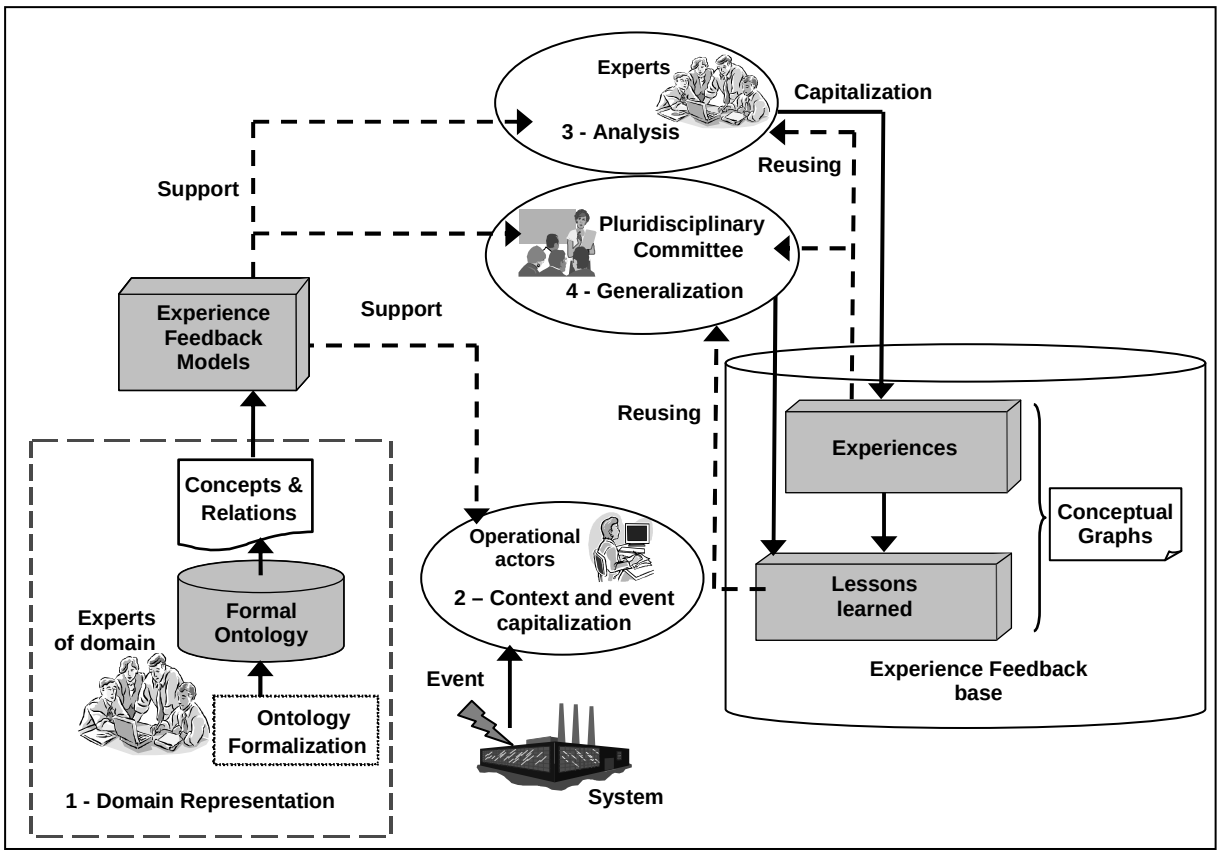

Fig. 4: Formalization of experience feedback process

6. Knowledge formalization with conceptual graphs

The conceptual graphs are a knowledge representation language, introduced by John Sowa in [25] and extended in [28]. Such language permits at the same time to define a vocabulary (i.e. ontology) and to use this vocabulary to conceptualize facts. Conceptual graphs enable to represent complete first-order, modal, and higher-order logics, but they were developed as a more intuitive notation for logic. Conceptual Graphs can be considered as a compromise representation between a formal language and a graphical language because it is visual and has a range of reasoning processes [33].

The attractive features of conceptual graphs have been noted previously by other knowledge engineering researchers who are using them in applications [49], [69], [74], [75], [76]. These features include the ability to represent complex relationships among entities; to express selection constraints for any given entity; to map conceptual graphs onto database representations; and to map onto other formal systems, such as first-order predicate calculus.

6.1. *Construction of the Formal Ontology*

The ontology is the heart of any knowledge description: knowledge is intimately related to the ontology, since it is necessarily expressed in terms of this ontology. The ontological objects are usually described as a set of concepts and a set of relations between concepts. These sets may be ordered to form a taxonomy of concepts types or relations types.

In the Conceptual Graph (CG) formalism, this knowledge is encoded in the support that includes the following sets:

- The Concept Type Lattice (T_C) describes all concept types that may be used in the concept tokens of conceptual graph representations. The Concept Type Lattice is a partially ordered finite set of concept types.
- The Relations Type Lattice (T_R) too describes the finite set of relations types that is structured by a partial order, forming a hierarchic structure.

Both type concepts and relations are ordered by a subsumption link showing their inheritance relationships. The interpretation of the subsumption link is that the extension (i.e. the set of objects characterized by the type) of a concept type (e.g. Machine) is a subset of the extension of another concept type (e.g. Ressource).

To a certain extent, the type used for concepts and relations must be precisely defined in the formal ontology where the terms may have associated constraints (e.g. signatures determining the link prerequisites for the relation types) and definitions

(e.g. definitions of necessary and/or sufficient conditions). In the rest of the paper, the terms "support" and "ontology" are used alternatively: the former is the Conceptual Graph implementation of the latter (see figure 4 for an example).

6.2. *Conceptual graphs*

Setting up the support is a preliminary task to build a knowledge based application using Conceptual Graphs [28], [77].

Definition 1: Support
A *support* is a four-tuple S = (T_C, T_R, I, τ).
T_C and T_R are two partially ordered finite sets, respectively of *concept types* and *relation types*. T_C possesses a greater element, called the universal type, and denoted by $\top$. Relation types may be of any arity greater or equal to 1. Only relation types with same arity are comparable. I is the set of *individual markers*. T_C, T_R and I are pairwise disjoint. τ is a mapping from I to T_C. The *generic marker* is denoted by *, where * $\notin$ I. The set $I \cup \{*\}$ is partially ordered in the following way: * is the greatest element and elements of I are pairwise non-comparable.
The partial orders on types are interpreted as specialization relations ($t \leq t'$ is read as *t is a specialization of t'*).

Definition 2: Simple Graph
A simple Conceptual Graph is a finite, directed, bipartite graph consisting of concept nodes (denoted as boxes), which are connected with conceptual relation nodes (denoted as circles). In the alternative linear notation, concept nodes are written within square brackets, while conceptual relation nodes are denoted within brackets.
A conceptual **relation** binds two or more concepts according to the following diagram $[C_1]\rightarrow$(*relation's name*)$\rightarrow$ $[C_2]$ (means 'C_1 is linked to C_2 by this *relation's name*'). For example, [Material]$\rightarrow$(Attr)$\rightarrow$[Hardness] means Material "has an" attribute "which is" Hardness. Each relation has a signature, which fixes its arity (the number of arguments it takes) and gives the maximum types of concept available, to which a relation of the type can relate.
The **nested** Conceptual Graphs [33] enables association of any concept node with a partial internal description. In addition nesting allows to create several representation levels, to organise these levels of detail into a hierarchy and thus to embed a conceptual graph in the marker of a concept by adding internal information to it. An important advantage of nested graph models is the option of partitioning the reasoning tasks into separate metalevel stages, each of which can be axiomatized in classical first-order logic. For that, a mathematical operator is defined that translates conceptual graphs into formulas in the first-order predicate calculus (relations become n-ary predicates, concepts become unary predicates, individual markers become constants and generic markers become existentially quantified variables). However, the underlying structure of the graph theory can support a broad variety of inferences that goes far beyond logical deduction. All properties on conceptual graphs depend on their own structure and on the ontology they share. The study of theirs manipulations by a knowledge base relies on conceptual graphs operations which are considered to be the backbone of the reasoning system.

6.3. *Conceptual graphs operations*

Reasoning relies on a standard operation of CG called ***projection*** [25]. Conceptual Graphs are logically founded, with projection being sound and complete with respect to deduction in first-order logic (FOL) [70]. In fact, the projection extracts a subgraph from a given graph by applying a sequence of specialization rules. More formally, a *projection* π from a graph *G1* to a graph *G2* is defined with the following properties:

- π is a mapping from the nodes of G_1 to the nodes of G_2 which preserves edges, i.e., if *xy* is an edge of G_1 then $\pi(x)\pi(y)$ is an edge of G_2.
- π may specialize the labels of concept and relation nodes. For each concept *c* in G_1, $\pi(c)$ is a concept in $\pi(G_1)$ such that type$(\pi(c)) \leq$ type(c) and if c is an individual concept, then marker($\pi(c)$) = marker(c). For each relation node *r* in G_1, $\pi(r)$ is a relation node in $\pi(G_1)$ such that type($\pi(r)) \leq$ type(r).

The existence of a projection from a CG G_1 to a CG G_2 means that the knowledge represented by G_1 (*request graph*) is deducible from the knowledge represented by G_2 (*context graph*), as shown in figure 5. In this picture a projection is feasible, because the concept "Machine" is a specialization of concept "Resource" and the "tuning activity" is a specific "activity". So, the only one projection ($\Pi(G_1)$) is encircled with a dot line. The context graph can be interpreted as "*There is machine which is doing an activity of turning on a metal which is in titanium*". The request can be interpreted as "*Is there a resource performing an activity?"*. The projection graph gives the response which can be interpreted as "*There is machine which is doing an activity of turning*". Conceptual Graph **projection** can be **extended** with an implementation of a depth-attenuated distance (between types in the ontology) or graph transformations allowing **approximate search** [49], [78].
The question of the existence of a projection of a graph into another graph is NP-complete [70]. However there are polynomial cases, for instance the

question of the existence of a projection of an acyclic graph into a general graph [71]. A practical interest of this result is that acyclic graphs seem to be very frequent in conceptual graph applications [49], [69], [76].

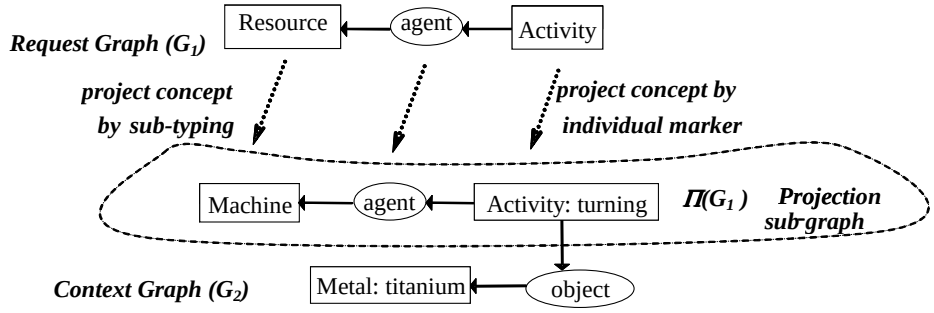

Figure 5. Application of a projection operation

The projection operation is a building block for more complex kinds of knowledge reasoning, like graph constraints and graph rules [33].

A graph rule has the following form: "*if condition then conclusion*" (where *condition* and *conclusion* are Conceptual graphs sharing the co-referents nodes). The graph rule is used in the following classical way: given a simple graph, if the condition of the rule projects to the graph, then the information contained in the conclusion is added to the graph. Each rule has a life duration that depends on the objective to be attained; when a context evolves, consistent knowledge that produce usable rules must be reviewed by competent actors.

7. Methodology for Experience Feedback using conceptual graphs

7.1. *Ontology*

The formal ontology used in this work for experience feedback is represented on figure 6. The main advantage of this ontology is to propose jointly a support for experience feedback and a support for the domain. In order to make enterprises interoperate, this particularity is important because the manner to model and capitalize knowledge is integrated to the ontology as well as the domain itself. In the proposed example, the ontology of the domain addresses a manufacturing environment. The concept type "feedback_object" can be specialized into "Activity", "Product", "Process", "Resource" or "Competency". The "Resource" concept type can be also specialized. The experience feedback domain is described by means of the "Experience_Element" concept type that can be specialized into "Event", "Context", "Analysis" and "Solution" concept types. These four concepts are the four pillars of an experience and they are used to build conceptual graphs (see next section).

The "Event" concept type can be specialized into "negative_event" or "positive_event" concepts type. In the rest of this paper, only negative events are considered. A negative event is generally unexpected and has a negative impact on the performance of the organization. In the proposed example, it can be specialized into "Breaking", "Start", "Late", etc. Such a negative event can be part of an experience if it is possible to capitalize this event with its concept type, to analyze the root causes and then, to formalize and capitalize the solution. Positive events correspond to situations where products, processes or services are good (or even better than expected). These situations are more difficult to detect and to formalize. For instance, a project scenario leading to good results with respect to one or more objectives in a given context can be considered as a positive event and can be capitalized with the proposed framework. Generally, these events are difficult to highlight because decision makers do not spend time to analyze standard and positive situations.

The second part of the support concerns relations types. It gathers some "high level" relation types like "Temporal", "Spatial", "Usual", "Logic" and "Experience_Relation" ones. The specialization of the relation type "Experience_Relation" ("Require", "Generate", "Concern", "Belong") enables to gather into conceptual graphs several "Experience_Element" concept types.

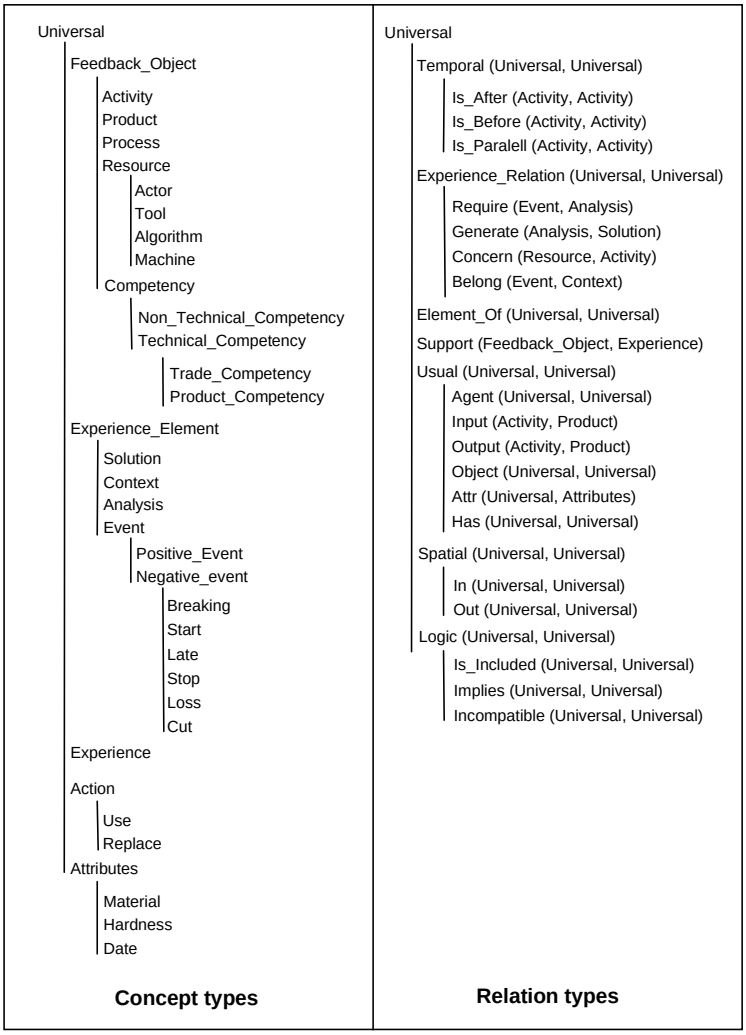

Figure 6. Formal ontology (lattice of concept and relation types) in conceptual graph

7.2. *Generic conceptual graph model of an experience*

Building a Conceptual Graph (CG) depends on the possible labels that can be used, i.e. on the sets of types, relations, and markers. These sets constitute the ontology of a Conceptual Graph-based representation. The framework introduced here therefore mainly consists of a set of canonical conceptual graphs for experience feedback representation. These graphs are considered to be the backbone of the experience feedback formalization.
Considering the general Experience Feedback process of figure 4, a framework describing an experience in a generic manner is required. When an event occurs, it is necessary to formalize this event and to precise the context in which the event has occurred. Clearly, in a continuous improvement context [1], it is not sufficient and the event has to be analyzed according to its context (search of causes, evaluation of effects on the system) and a solution has to be proposed. Most companies nowadays use this procedure. Each negative event (machine failure, quality problem on a product, etc.) has to be treated very quickly in order to avoid its propagation all along the process. In this context of reactivity, the analysis has to be quickly realized and can lead to a corrective solution. Decision makers need tools favoring rapid responses to problems, and in this respect, experience feedback is a good solution. This methodology enables to search in an experience base if similar experiences have been capitalized. From the capitalized experiences, analysis and solutions can be extracted, adapted to the current problem and, finally, also capitalized creating a new experience. Therefore, one contribution consists in a proposition of a conceptual graph based model for experiences, enabling the actors both to react more quickly and to capitalize more easily their knowledge. The generic conceptual graph of figure 7 uses nested CG for the representation of a generic experience. The **CoGUI** (Conceptual Graphs Graphical User Interface) tool [67] has been used in order to define the ontology and to build the graphs.

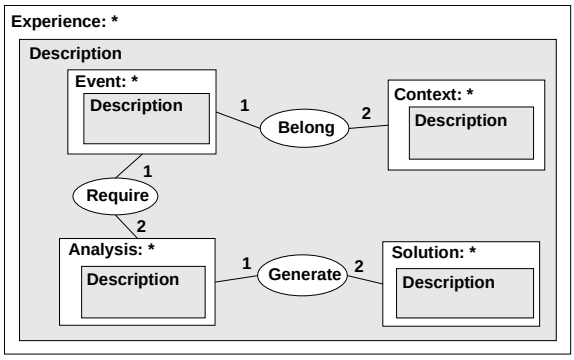

Figure 7. Generic experience GC

Five concepts coming from the formal ontology (figure 5) are used: *Event*, *Context*, *Analysis* and *Solution*. Three relations are used: *Belong*, *Require* and *Generate*. This generic graph can be interpreted in formal language as: *An experience has a description. This description is: there is an event belonging to a context, this event requires an analysis and the analysis generates a solution. The concepts Event, Context, Analysis and Solution can be described by means of nested CG*. This generic model has to be instantiated each time an event occurs on the system. The instantiation of the generic GC consists in defining the markers and the content of the nested graphs descriptions. The marker of an experience concept enables to differentiate experiences. For instance: "[Experience: *Exp1*]" means that there exists an experience *Exp1*. Implicitly, this experience has a description which has to be described by a CG. The markers of the Event and Context concepts have to be defined when the event and its context are modeled by means of CG. A marker different to the generic one ("*") for the event concept means that there is a CG describing this Event. In a similar way, the Analysis marker and the Solution marker are defined when the description of each one is provided.

The main difficulty during the process of knowledge capitalization is to know which concepts are needed for the different descriptions so as to have a comprehensive knowledge. A possibility consists in adding to certain descriptions of the generic CG (mainly the Event and Context descriptions) a nested predefined CG. This nested CG can be seen as the minimum set of required attributes with the appropriated relations and edges. Each attribute is represented by a concept type. For instance, to describe the context of an event in an industrial workshop environment, the following minimum set of concepts can be required: {Machine, Actor, Activity, Product, Tool}. Therefore, the generic model has to be enriched (specialized) by these concepts. An example of predefined CG is proposed on the figure 8. It concerns the contextual description in an industrial workshop environment.

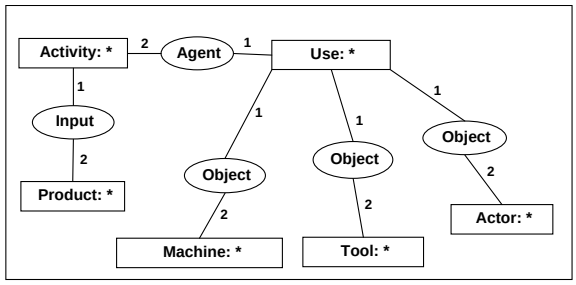

Figure 8. Example of generic CG for the context's description

The conceptual graph of figure 8 can be interpreted in formal language as: *a Product is the input of an activity; this activity uses a Machine, an Actor and a Tool*.

The main advantage of this generic graph is that experts in charge of the problem solving and the knowledge capitalization are guided by this framework. They are bound to use all the concepts of the minimum set. In return, this framework is very specialized to a particular domain. A flexible solution consists of further predefined nested CG (available on a library of generic conceptual graphs). Each one is adapted to a particular domain of the enterprise for which an experience feedback process is effective (production, quality, maintenance, project/process management, etc.)

For the description of events, a predefined CG is proposed as well (figure 9).

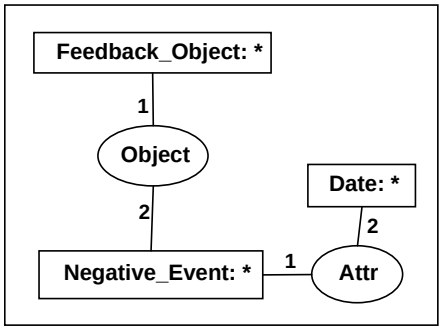

Figure 9. Example of generic CG for the event's description

The characteristic of the support to have a partial order between two concepts is used to obtain a sufficient generic description of the event. The translation into formal language is: *A feedback object is concerned by a negative event occurred at a date*. A feedback object is a generic concept that can be specialized (into Activity, Process, Product or Resource – see the ontology on figure 5). It gathers all the concepts for which an experience feedback is possible. A negative event can be also specialized (Breaking, Stop, Start, Loss, Cut, Late, etc.).

For the analysis description and the solution description, no generic nested CG is proposed. The reason is that it is impossible to know in advance what can be the problem analysis and its solution. On the other hand, this step could be aided by experience feedback. If similar events have already

occurred, experts can use the previous analysis as a framework, as well as the solution.

7.3. *Process of instantiation of a new experience*

The process of instantiation of a generic experience is as follow. After an event is detected, the actors in charge of the resolution of the problem have to instantiate a generic experience. Firstly, a marker is given to the Experience concept. Secondly, the Event has to be described by means of the associated predefined nested conceptual graph of figure 8, using the ontology. In parallel, the marker of the Event has to be defined. The Feedback_object concept and the Negative_event concept have to be specialized into the right ones. The markers of the two concepts have to be defined as well as the marker of the Date concept. Thirdly, the Context has to be described and its marker has to be defined. The next steps consist of making the analysis, finding a solution, capitalizing the experience and carrying out the solution. The actors have to search in the experience base if a similar event has already occurred in the past. Therefore, a request has to be realized in the experience base to retrieve the concerned experiences if they exist. In order to make this action, the projection operation is used. This operation is described in the next paragraph. If one or more experiences are found, they have to be integrated by the experts to build the analysis and to define a solution to the current problem. The analysis and the solution are inspired and adapted from the previously capitalized experiences. Clearly, if no experiences are found, the experts must perform the analysis from scratch and find a solution without any feedback.

7.4. *Similar experiences retrieval*

As presented in section 7.2, previous capitalized experiences are used to help the expert to solve the present problem. Reasoning mechanisms (projection, mappings or transformations [33]) of conceptual graphs can help to cover some problems solving methods closely or completely and to identify their relevance for the purpose of providing valuable results for a user (operator or manager) facing the problem. Particularly, the projection operation [71] defines a generalization/specialization relation over conceptual graphs and may slice the knowledge model to remove parts unrelated to the studied problem. Using the projection, the reasoning system is able to find not only descriptions of experiences that are annotated by some classes of normalized employed concepts and relationships but also those annotated by subtypes of these classes. Besides, to search with imprecise and/or incomplete experiences or to answer a vague query, approximate projections [49], [78], [79] can be used. Considering an event and its context modeled by means of GC, experts have to retrieve from the experience base experiences with similar events and/or context. To realize that search, the projection operation (see § 6.3) is used. Therefore, a request has to be defined, using the conceptual graph paradigm. The request content is directly defined by the experts. It can be a description of the event but also a description of its context (or a partial description of the context).
Therefore, a projection of the request graph on the capitalized experiences has to be realized. The generic CG of figure 6 is used to define the request. For the request, the descriptions of the event and its context have not to be very detailed. A set of concepts can be sufficient. In that case, the projection of the request on a CG modeling an experience enables to know if these concepts are present or not in the descriptions of the context or event. If all these concepts are present then there is at least one projection. Therefore, the concerned experience can be an interesting case and the analysis and solution can be helpful for the expert.

7.5. *Example of experience*

In order to illustrate the proposition, one suggests the example of a complete experience (figure 10). The conceptual graph is made according to the ontology proposed at paragraph 6.2. In formal language, this graph can be interpreted as follows. *The experience Exp1 has a description: The event Evt1 belongs to the context C1 and requires the analysis A1. This analysis generates the Solution S1. The event description is: the Tool Standard Milling Cutter Phi20 is broken on 2006/05/22. The context description is: the product Px_10 is the input of the activity Milling 010 which uses the actor Robert, the machine Huron_Kx_10 and the tool Standard Milling Cutter Phi20. The analysis description is: the product Px_10 is in Titanium with a hardness incompatible with the Tool Standard Milling Cutter Phi20. The solution description is: Replace the Tool Standard Milling Cutter Phi20 by Tool Carburized Milling Cutter Phi20.* This conceptual graph can be stored in the experience base in order to be reused.

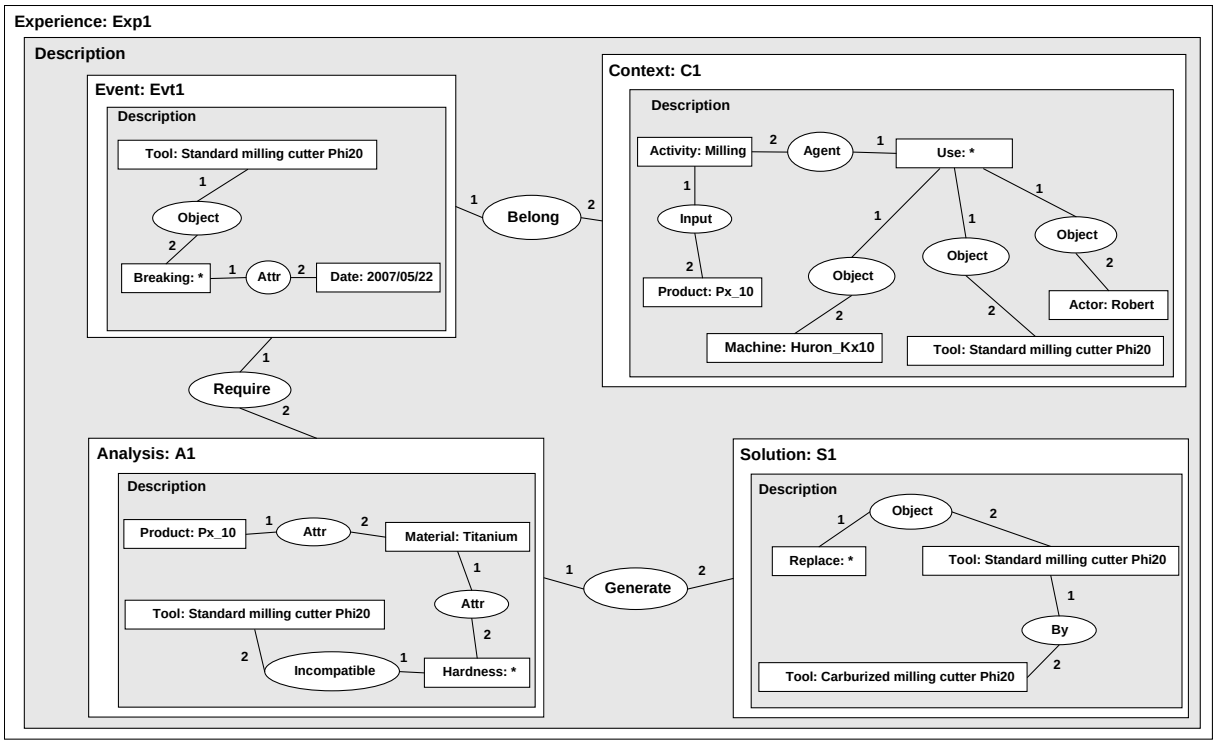

Figure 10. Example of a complete experience

7.6. *Example of experience feedback*

Considering the experience of figure 10 and a new event occurring in its context C025, the generic experience GC is instantiated with the marker *Exp2*. The event is described by the Event concept (marker *Evt025*) and its nested graph description (figure 11). The difference with the event Evt1 is the marker of the Tool concept (Turning versus Milling) and also the date of occurrence. The context C025 has the following description (figure 12).

In order to describe the analysis and the solution, the expert can make a projection operation on the experiences previously capitalized. There is only one experience (figure 10) capitalized in the experience base for this very simple example.

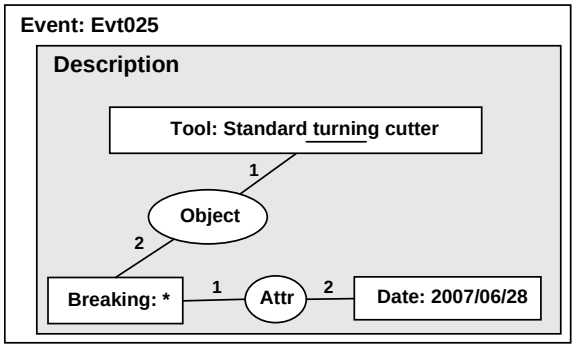

Figure 11. New event concept with its description

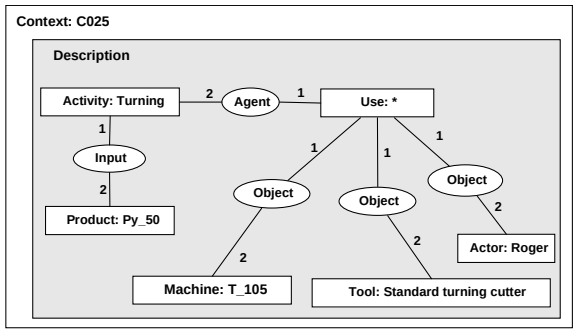

Figure 12. New context concept with its description

The request can be described by means of the following conceptual graph (figure 13).

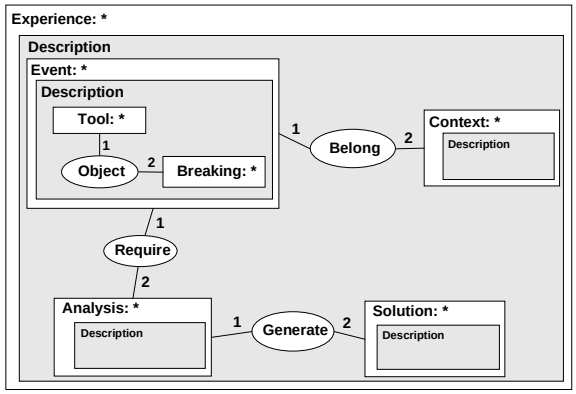

Figure 13. Request CG

This request is interpreted as: *Is there an experience where the event's description is about a Tool breaking and where the analysis and the solution have descriptions?* Therefore, the request CG is

projected on the experience Exp1.
The projection is represented on the figure 14.

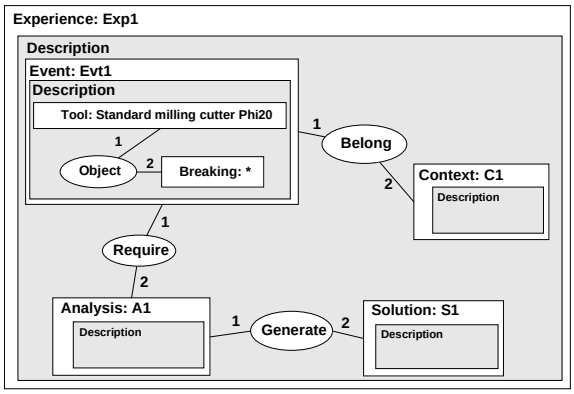

Figure 14. Projection of the request on the Experience Exp1

This response is interpreted as: *The experience Exp1 concerns the event Evt1, the analysis A1 and the solution S1. The event's description concerns the breaking of the tool "Standard milling Cutter Phi20"*. This experience previously capitalized does not concern the same tool (milling and not turning one), but the solution can be useful for the expert to find a solution. The response only points out that the experience Exp1 seems to be useful with respect to the request. The expert has to explore the experience Exp1 in order to find (if possible) a solution to his problem. Therefore, the experience Exp2 can be entirely defined, adapting its analysis and its solution concepts. Let us point out that an implementation in Prolog+CG is currently carried out. "*Prolog+CG is a conceptual and an object-oriented extension of Prolog, a standard programming language in Artificial Intelligence. Conceptual Graphs can be used to represent goals and can be used and manipulated as basic data structures, with operations like maximal join, projection (or more precisely subsumption), generalization and unification operations.*" [80]. The implementation of the example may be achieved as follows:

a) Description of the ontology

The concept type hierarchy is described using the specialization operator (>).

```
Universal >Feedback_Object, Experience_Element, Action, Attributes, Experience.
Experience_Element > Solution, Context, Analysis, Event.
Action > Use, Replace.
Attribute > Material, Hardness, Date
Competence > Technical_Competence, Non_Technical_Competence.
Feedback_Object > Resource, Activity, Product, Process.
Event > Positive_Event, Negative_Event.
Negative_Event > Cut, Late, Loss, Stop, Start, Breaking.
Technical_Competence > Product_Competence, Trade_Competence.
Resource > Tool, Actor, Machine, Algorithm.
```

b) Description of an experience

An experience is a conceptual graph made of four sub-graphs (Event, Context, Analysis and Solution). All sub-graphs are described separately and gathered thanks to the expression written in bold. Here, conceptual graphs are described with the linear notation in which concepts are represented by square brackets instead of boxes, and the conceptual relations are represented by parentheses instead of circles:

```
experience(_Exp, X) :-
        desc(_Exp,[Event : _E = E]),
        desc (_Exp,[Context : _C = C]),
        desc (_Exp,[Analysis : _A = A]),
        desc (_Exp,[Solution : _S = S]),
        eq(X , [Solution : _S = S]<-Generate-[Analysis : _A = A]<-Require-[Event : _E = E]-Belong->[Context : _C = C]).

desc ( exp1,[Event : Event1 =
        [Tool:Std_Phi20]-Obj->[Breaking]-Attr->[Date:zerotrois]]).

desc (exp1,[Context:Context1=
        [Product:Px10]<-Input-[Activity:Milling_10]-
                -Agnt->[Use]-
                -Obj->[Actor:Robert],
                -Obj->[Machine:Huron],
                -Obj->[Tool: Standard_milling_cutter]]).

desc (exp1,[Analysis:Analysis1=
        [Product:Px10]-Attr->[Material:Titanium]-Attr->[Hardness]-
        Incomp->[Tool:Standard_milling_cutter]]).

desc (exp1,[Solution:Solution1=
        [Replace]-Obj->[Tool:Standard_milling_cutter]-By-
        >[Tool:Carburized_milling_cutter]]).
```

c) Description of a query

A query is described by a name (query1) and an associated conceptual graph.

```
query(query1,[Event : _ =[Tool]-Obj->[Breaking]]).
```

d) Description of a search method based on projection (subsumption)

This method is based on the projection (subsumption) operation. The idea is to check for each experience of the experience base if it matches the query and to return the context of relevant experiences. The operation "subsume(A, B, C)" checks that A subsumes B and returns in C the image of A in B (the sub-graph of B that is isomorph to A).

```
find_context(Q,R) :-
    experience(E,_E),  // Get experiences
    query(Q,_Q),       // Get the CG corresponding to the query
    subsume(_Q,_E,_X), // Find experiences that match the query
    subsume([Context],_E,R). // Get context of relevant experiences
```

7.7. *Lesson learned*

The next step of the model concerns the experience generalization (lesson learned). Periodically, an expert committee analyzes the experience base in order to generalize the knowledge capitalized during

the experience feedback process described into the section 6. This process of generalization aims at preventing the occurrence of negative events. The process consists in searching into the experience base the experiences which contain knowledge about the theme treated by the committee.
Therefore, a request has to be instantiated and a projection operation carried out on the entire experience base. The set of retrieved experiences can contain sufficient knowledge for rule generalization.
It is important to notice that this process of generalization is not an automatic one. Its role is only to aid the experts to make appropriate decisions (to define the lessons). The advantage of this process is to capitalize the knowledge of experts in a structured way, enabling computer assistance to be more relevant.
A lesson can be described by means of two CG: hypothesis and conclusion. An example of rule is proposed on figure 15.

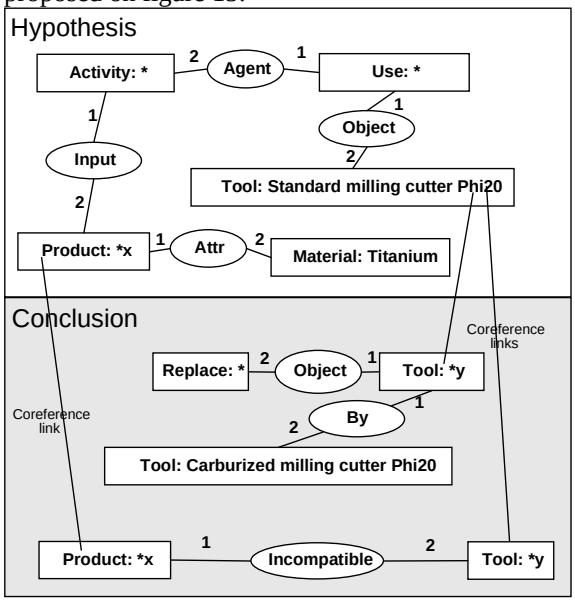

Figure 15. Example of CG of lesson learned

It can be interpreted as: *If a product *x is in titanium and it is the input of an activity using the Tool Standard milling cutter Phi20, then:*
*i) the product *x is incompatible with the tool Standard milling cutter Phi20;*
ii) this tool has to be replaced by the tool Carburized milling cutter Phi20.

The conclusion part of the CG lesson learned of this example is constituted of two CGs. The first one (i) corresponds to a corrective proposition; the second (ii) is rather a preventive one enabling to avoid the occurrence of the negative event (tool breaking).

8. Conclusion

Based on an experience feedback process model, this paper takes conceptual graph implementation of the domain ontology as a framework for experience feedback processes formalization and knowledge reasoning. This framework is defined considering requirements about interoperability of networked organizations within a global continuous improvement process.
The main contributions of this paper are:
i) A methodology (the Event-Context-Analysis-Solution-Lesson framework and its reification using conceptual graphs) suitable for effective description of experience feedback artifacts. Thus experiences acquired from past contexts and inducing lessons can be used to improve industrial activities within a continuous improvement process. This methodology is based on the idea that interoperability can be facilitated if different actors have guidelines for knowledge capitalization and exploitation using a common ontology. Therefore, these guidelines have five pillars:
- the library of reusable Experience Feedback models based on the conceptual graphs paradigm;
- the experience structure ;
- the methodology integration within a continuous improvement process well understood and accepted in enterprises;
- the strong theoretical background as the operator of projection in conceptual graphs very suitable for reasoning in a problem solving context;
- tools as Prolog+CG very suitable for artificial intelligence fields and particularly interesting as support in the proposed methodology for reasoning.
ii) A formal ontology that is “machine understandable”, in such a way that it enables making statements and asking queries about a particular domain due to the use of a precise conceptualization, which describes entities and their relationships. Consequently, domain ontology can be used in capitalization and exploitation of experience feedback processes.
iii) The reasoning system relies on a set of graph transformations; an **original feature** is that it enables graphical illustration of reasoning for the end-user, since lessons learned can be directly visualized on the conceptual graph. So the graph structure can be exploited to enhance the knowledge modeling and adapt a relevant experience given by the users.
The proposed methodology enables to capitalize and to share knowledge about problem solving. This aspect participates to integration and to interoperability because the knowledge is necessarily distributed among further actors or entities. So, two or more enterprises gathered with a common goal of performance can better interoperate if they are able to deploy a wide continuous improvement methodology using a predefined framework based on a common ontology about the domain and a common ontology about the knowledge engineering.
Obviously, this methodology is validated only on a very simple example and needs to be evaluated on

real and complex industrial contexts. For future work we wish to focus on an experience modeling closer to expert's natural expression, for instance relying on the explicit representation and management of fuzzy knowledge in conceptual graphs [79]. In order to help non-technical end user with practical guidelines as to how such graphs should be built, it is necessary to study the role and impact of domain knowledge [81]. Since many new applications have the same requirements as earlier ones, one possibility is to create generic domain properties as templates for requirements of certain classes of applications [82]. This would be very useful for the end-users in graphs manipulation by providing sets of predefined conceptual graphs for experience feedback processes.

The proposed approach aims at the same objective as case-based reasoning, but some standard vocabularies for case description are needed, which ensure the success of case interchange and distributed case-based reasoning. Comparing with traditional case representations (free-text format [83], object-oriented techniques [84], etc), this work has the advantages of enriched semantic representation and better integration with interoperability efforts. Meanwhile, several XML-based case representation languages such as CBML [85] and OML [11] have recently been introduced into the CBR community, in order to facilitate the storage and distribution of case data over a network and possible interoperability with non-CBR systems. Thus, the research direction aiming at interoperability between Experience Feedback formalization using conceptual graphs and CBR systems is worthy of continued investigation.

9. Bibliography

[1] Zaraté P., Munoz M., Soubie J.L., Houé R. Knowledge Management Systems: A process oriented view. Cybernetics and Systems Analysis, Springer Verlag, V. 41 N. 2, p. 274-277, 2005.

[2] Vernadat F. B., Interoperable enterprise systems: Principles, concepts and methods, Annual Reviews in Control, in press, doi:10.1016/j.arcontrol.2007.03.004, 2007.

[3] IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries. New York, NY: Institute of Electrical and Electronics Engineers, 1990.

[4] Viel S., Coudert T., Geneste L., Cherencq F. Proposition of a multi co-operating experience feedback processes architecture, IFAC MCPL'07, p. 79-84, September 27-30, 2007, Sibiu, Romania.

[5] Rakoto H. Intégration du Retour d'Expérience dans les processus industriels – Application à Alstom Transport (in French). PhD thesis, National Polytechnic Institute of Toulouse (France), October 2004.

[6] Whitman LE., Panetto H., The missing link: Culture and language barriers to interoperability, Annual Review in Control, Volume 30, p.233-241, 2006.

[7] Zorn T.E. and Taylor J.R. Knowledge management and/as organizational communication. In D. Tourish and O. Hargie (Eds.), Key Issues in Organizational Communication. London and New York: Routledge. ISBN 0415260930. Zorn and Taylor (pp 98-99) distinguish four uses of the term "knowledge management", 2004.

[8] Easterby-Smith M., Lyles M. A. (editors). The Blackwell Handbook of Organizational Learning and Knowledge Management, Oxford, Blackwell Publishing, 2003.

[9] Lebowitz J. Knowledge Management Handbook, CRC Press, 1999.

[10] Dalkir K. Knowledge Management In Theory and Practice. Amsterdam; Boston : Elsevier/Butterworth Heinemann, June 2005.

[11] Bergmann R. Experience Management: Foundations, Development Methodology, and Internet-Based Applications , volume 2432 of LNAI (Lecture Notes in Artificial Intelligence), Springer, 2002.

[12] Kolb D. Experiential learning: Experience as the source of learning and development. Englewood Cliffs, N.J.: Prentice Hall, 1984.

[13] Weber R., Aha D.W., Becerra-Fernandez I. Intelligent lessons learned systems. Expert Systems with Applications, Volume 20, Issue 1, p. 17-34, 2001.

[14] Faure A., Bisson G. Modeling the experience feedback loop to improve knowledge base reuse in industrial environment. Proceedings of KAW 99, Twelfth Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada, 1999.

[15] Schreiber G., Akkermans H., Anjewierden A., Hoog R.d., Shadbolt N., Velde W.v.d., Wielinga B. Knowledge engineering and management. The MIT Press, Cambridge, 2000.

[16] Duribreux M., Caulier P., Houriea B., Faroux DA. *Elicitation and Analysis of Expert Knowledge on the Operation of Gas Distribution Networks*. University of Kassel, Kassel, Germany, 1993.

[17] Hermosillo Worley J., Rakoto H., Grabot B., Geneste L. A competence approach in the experience feedback process. In Integrating Human Aspects in Production Management, IFIP International Federation for Information Processing series, Volume 160, pp 220-235, edited by Zulch G., Jagdev H.S and Stock P., Springer-Verlag, New York, USA, 2005.

[18] Weber RO., Aha DW. Intelligent delivery of military lessons learned. Decision Support Systems, Volume 34, Issue 3, February 2003, p. 287-304, 2003.

[19] Eck, Engelfriet, Fensel, van Harmelen, Venema and Willems. A survey of languages for specifying dynamics: a knowledge engineering perspective.

IEEE Transactions on Knowledge and Data Engineering, Vol.13, N°.3 May/June 2001.
[20] Pierret-Golbreich C., Talon X., TFL: An Algebraic Language to Specify the Dynamic Behaviour of Knowledge-Based Systems. The Knowledge Eng. Rev., vol. 11, no. 3, p. 253-280, 1996.
[21] Abrial JR. The B-Book: Assigning Programs to Meanings, Cambridge University Press: Cambridge,UK, 2005.
[22] Brazier FMT., Treur J., Wijngaards NJE., Willems M. Temporal Semantics of Compositional Task Models and Problem Solving Methods, Data and Knowledge Eng., vol. 29, no. 1, p. 17-42, 1999.
[23] David H. Statecharts: A visual formalism for complex systems. Science of Computer Programming, 8:231-274, 1987.
[24] Jungclaus R., Saake G., Hartmann T., Sernadas C. Troll: A Language for Object-Oriented Specification of Information Systems, ACM Trans. Information Systems, vol. 14, p. 175-211, 1996.
[25] Sowa JF. Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publishing Company, Reading, MA, 1984.
[26] Baader F. Logic-based knowledge representation. In MJ Wooldridge and M.Veloso editors, Artificial Intelligence Today, Recent Trends and Developments, number 1600, in Lecture Notes in Computer Science, p. 13-41. Springer Verlag, 1999.
[27] The REVISE project. A Purpose Driven Method for Language Comparison. Proceedings of the 8th European Knowledge Acquisition Workshop (EKAW'96) N. Shadbolt and K.O'Hara (eds.), LNAI 1076, p.66-81, Springer Verlag, 1996.
[28] Sowa JF. Knowledge Representation: Logical, Philosophical, and computational Foundations. Brooks Cole Publishing Co., 2000.
[29] Quinn KT. Review of Knowledge Management in Theory and Practice by Kimiz Dalkir, Elsevier Butterworth Heinemann, Interactions 13(2): 48-, 2006.
[30] Nonaka I., Takeuchi H. The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford and New York: Oxford University Press, 1995.
[31] Dieng-Kuntz R., Matta N. editors. Knowledge Management and Organizational Memories, Kluwer Academic Publishers, ISBN 0-7923-2659, July, 2002.
[32]. Fensel D., Motta E., van Harmelen F., Benjamins V. R., Crubezy M., Decker S., Gaspari M., Groenboom R., Grosso W., Musen M., Plaza E., Schreiber G., Studer R., Wielinga B. The Unified Problem-Solving Method Development Language UPML. Knowledge and Information Systems, Volume 5, Issue 1, P. 83-131.
[33] Baget JF., Mugnier M-L. Extensions of Simple Conceptual Graphs: the Complexity of Rules and Constraints, Journal of Artificial Intelligence Research (JAIR), vol. 16, p. 425-465, 2002.
[34] Aamodt A., Plaza E. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, **7 (1)**, p. 39-59, 1994.
[35] Kolodner J. Case-Based Reasoning, Morgan Kaufmann Publishers INC, 1993.
[36] Staab S., Studer R. (eds.). Handbook on Ontologies. International Handbooks on Information Systems, Springer Verlag, 2004.
[37] Gruber TR. A translation approach to portable ontology specifications. Knowledge Acquisition, n°2(5), p. 199-220, 1993.
[38] Fensel D. Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, ISBN: 3540416021, Berlin, 2001.
[39] Guarino N. Understanding, Building, and Using Ontologies: A Commentary to Using Explicit Ontologies in KBS Development, International Journal of Human and Computer Studies, 46, p. 293-310., 1997.
[40] Campbell KE, Oliver DE, Shortliffe EH. The Unified Medical Language System: toward a collaborative approach for solving terminologic problems. Journal of the American Medical Informatics Association 1998; 5(1), p. 12-16, 1998.
[41] Gruninger M., Atefi K., Fox MS., Ontologies to Support Process Integration in Enterprise Engineering, Computational and Mathematical Organization Theory, Vol. 6, n° 4, p. 381-394, 2000.
[42] Abecker A. and van Elst L. Ontologies for Knowledge Management, in Handbook of Ontologies, Springer, Berlin, 435-455, 2004.
[43] Berners-Lee T., Hendler J., Lassila O. The Semantic Web. Scientific American, 284(5), p. 34-43, 2001.
[44] Malucelli A., Palzer D., Oliveira E. Ontology-based Services to help solving the heterogeneity problem in e-commerce negotiations. Electronic Commerce Research and Applications, Volume 5, Spring Issue 1, p. 29-43, 2006.
[45] Berio G. Delivrable D3.1: Requirements analysis: initial core constructs and architecture. Unified Enterprise Modeling Language (UEML) Thematic Network, IST-2001-34229, (document available from the UEML portal at: http://www.ueml.org), 2003.
[46] Chen D., Vernadat F. Standards on enterprise integration and engineering – A state of the art. In International Journal of Computer Integrated Manufacturing (IJCIM), Volume 17, n°3, p. 235-253, April-May 2004.
[47] Shahar Y., Young O., Shalom E., Galperin M., Mayaffit A., Moskovitch R., Hessing A. A framework for a distributed, hybrid, multiple-ontology clinical-guideline library, and automated guideline-support tools. Journal of Biomedical Informatics, Volume 37, Issue 5, p. 325-344,

October 2004.
[48] Domingue J., Stutt A., Martins M. , Tan J. Petursson H., Motta E. Supporting online shopping through a combination of ontologies and interface metaphors. International Journal of Human-Computer Studies, Volume 59, Issue 5, p. 699-723, November 2003.
[49] Genest D., Chein M. A Content-search Information Retrieval Process Based on Conceptual Graphs. Knowledge And Information Systems, volume 8, no. 3, p. 292-309. Springer, 2005.
[50] Braga RMM., Werner CML., Mattoso M. Odyssey-Search: A multi-agent system for component information search and retrieval. Journal of Systems and Software, Volume 79, Issue 2, p. 204-215, 2006.
[51] Uschold M., Gruninger M. Ontologies: Principles, Methods and Applications, Knowledge Engineering Review, vol.11:2, p. 93-136, 1996.
[52] Ducq Y., Chen D., Vallespir B. Interoperability in enterprise modelling: requirements and roadmap. Advanced Engineering Informatics, Volume 18, Issue 4, p. 193-203, 2004.
[53] Orgun B., Vu J. HL7 ontology and mobile agents for interoperability in heterogeneous medical information systems. Computers in Biology and Medicine, Volume 36, Issues 7-8, p. 817-836, 2006.
[54] Tu SW., Eriksson H., Gennari JH., Shahar Y., Musen MA. Ontology-based configuration of problem-solving methods and generation of knowledge-acquisition tools: application of PROTÉGÉ-II to protocol-based decision support. Artificial Intelligence in Medicine, Volume 7, Issue 3, p. 257-289, 1995.
[55] Reynaud C., Tort F. Using explicit ontologies to create problem solving methods. International Journal of Human-Computer Studies, Volume 46, Issues 2-3, p. 339-364, 1997.
[56] Abel M., Silva LA., Campbell JA., De Ros LF. Knowledge acquisition and interpretation problem-solving methods for visual expertise: S study of petroleum-reservoir evaluation. Journal of Petroleum Science and Engineering, Volume 47, Issues 1-2, p. 51-69, 2005.
[57] Barthès JPA., Tacla CA. Agent-supported portals and knowledge management in complex R&D projects. Computers in Industry, Volume 48, Issue 1, p. 3-16., 2002.
[58] Wand Y. Ontology as a foundation for meta-modelling and method engineering. Information and Software Technology, Volume 38, Issue 4, p. 281-287, 1996.
[59] Baader F., Calvanese D., McGuinness D., Nardi D., P. Patel-Schneider. The description logic handbook. Theory, implementation and applications, Cambridge University Press, Cambridge, 2003.
[60] Angele J., Lausen G. Ontologies in f-logic. In: Staab S., Studer R. Editors, Handbook on ontologies, Springer-Verlag, Berlin, p. 29–50, 2004.
[61] Hayes P. Resource Description Framework (RDF) Semantics. W3C Recommendation, 2004.
[62] Antoniou G., Harmelen F.V. Web Ontology Language: OWL. In: Staab S. and Studer R., Editors, Handbook on ontologies, Springer-Verlag, Berlin, p. 67–92, 2004.
[63] Fensel D., Rousset MC, Decker S. Workshop on comparing description and frame logic. Data & Knowledge Engineering, p. 347–352, 1998.
[64] Dieng R., Corby O. Conceptual Graphs for Semantic Web Applications, In: Proc. of the 13th Int. Conference on Conceptual Structures (ICCS'2005), Dau F., Mugnier M-L., Stumme G. (editors), Kassel (Germany), July 17-23, 2005, Springer-Verlag, LNAI 3596, p. 19-50, 2005.
[65] Yao H., Etzkorn L., Automated conversion between different knowledge representation formats.Knowledge-Based Systems, Volume 19, Issue 6, p. 404-412, October 2006.
[66] Diallo D. Assistance to validation through paraphrasing of formal specification written in B. (in French). PhD thesis, University of Nantes, 2000.
[67] Gutierrez A. COGUI (Conceptual Graphs Graphical User Interface). Workshop "Tools" of the 13th International Conference on Conceptual Structures (ICCS'2005), July 18-22, 2005, Kassel, Germany. CoGui Project Web site: http://www.lirmm.fr/~gutierre/cogui/, 2005.
[68] Gerbé O., Mineau GW. The CG Formalism as an Ontolingua for Web-Oriented Representation Languages. ICCS'2002, Borovetz, July 2002, Springer, p. 205-219, 2002.
[69] Dieng-Kuntz R., Minier D., Ruzicka M., Corby F., Corby O., Alamarguy L. Building and Using a Medical Ontology for Knowledge Management and Cooperative Work in a Health Care Network. Computers in Biology and Medicine, Volume 36, Issues 7-8, p. 871-892, 2006.
[70] Chein M., Mugnier ML. Conceptual graphs: fundamental notions, Revue d'Intelligence Artificielle 1992 ; 6 (4), p. 365–406, 1992.
[71] Mugnier ML. On generalization/specialization for conceptual graphs, Journal of Experimental and Theoretical Artificial Intelligence, vol. 7, p. 325-344, 1995.
[72] Coulondre S. CG-SQL: a front-end language for conceptual graph knowledge bases. Knowledge-Based Systems, Volume 12, Issues 5-6, October 1999, p. 293-302.
[73] Kamigaki T., Nakamura N., An Object-Oriented Visual Model - Building and Simulation System for FMS Control, Simulation, vol. 67, n°6, 1996, p. 375-385.
[74] Volot F., Joubert M., Fieschi M. Review of biomedical knowledge and data representation with conceptual graphs. Methods of Information in Medicine, Volume 37, Issue 1, 1998, p. 86-96.
[75] Lee J., Lai LF. Verifying task-based specifications in conceptual graphs. Information and

Software Technology, Volume 39, Issues 14-15, 1998, P. 913-923.
[76] Kamsu-Foguem B., Chapurlat V. Requirements modelling and formal analysis using graph operations. International Journal of Production Research, Vol. 44, No. 17, 1 September 2006, p. 3451–3470.
[77] Mugnier ML., Chein M. (eds.) Conceptual Structures: Theory, Tools, and Applications, Lecture Notes in AI, vol. 1453, Springer-Verlag, Berlin, 1998.
[78] Corby O., Dieng-Kuntz R., Faron-Zucker C., Gandon F. Searching the Semantic Web: Approximate Query Processing based on Ontologies. IEEE Intelligent Systems Journal, Vol. 21, No.1, 2006.
[79] Thomopoulos R., Buche P., Hammerlé O. Representation of weakly structured imprecise data for fuzzy querying. Fuzzy Sets and Systems, vol. 140, p. 111-128, 2003.
[80] Kabbaj A., Petersen U. PROLOG+CG version 2.0 User's Manual, (Web site of PROLOG+CG: http://prologpluscg.sourceforge.net/)
[81] Jackson M. Problems, Methods and Specialisation. *Software Engineering Journal*, Volume 9, N° 6, p. 249-255, edited and abridged in IEEE Software, Vol. 11, N° 6, p. 57-62, November 1994.
[82] Maiden NAM, Hare M. Problem domain categories in requirements engineering. *International Journal of Human-Computer Studies*. 49(3), p. 281-304, September 1998.
[83] Aha DW., Breslow L., Muñoz-Avila H. Conversational Case-Based Reasoning. Applied Intelligence, 14(1): 9-32, 2001.
[84] Manago M., Bergmann R, Wess S., Traphöner R. CASUEL: A Common Case Representation Language - Version 2.0 ESPRIT-Project INRECA, Deliverable D1. Technical Report, University of Kaiserslautern, 1994. Document available from http://www.wi2.uni-trier.de/publications/CASUEL%20V2.03%20Language%20Def_w95.pdf, 1994.
[85] Hayes C., Cunningham P. Shaping a CBR view with XML, Lecture Notes in Computer Science, Vol. 1650, p. 468-482, 1999.