Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 48 (2020) 1079–1087

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

48th SME North American Manufacturing Research Conference, NAMRC 48 (Cancelled due to COVID-19)

Procedural knowledge and function blocks for smart process planning

Andrea de Giorgio[a]*, Magnus Lundgren[a], Lihui Wang[a]

[a]*KTH Royal Institute of Technology, Production Engineering, Sweden*

* Corresponding author. *E-mail address:* andreadg@kth.se

Abstract

In the age of digital manufacturing there is a need to elicit and transfer procedural knowledge between humans and machines. Having proper knowledge is essential in decision-making. The more the knowledge, the better decisions are made. To capture experiences and turn them into knowledge is fundamental in learning processes and knowledge development. Knowledge engineering and knowledge management have been subject for research for decades and several concepts about knowledge and knowledge transfer have been introduced, but a functional approach to exploit knowledge efficiently in manufacturing is still missing. In the era of Industry 4.0, humans and machines must be able to collaborate in such way that both can exploit the best abilities of each other in a manufacturing process. This paper introduces a procedural knowledge process (PKP) approach to capturing and defining unexpected events, while a process step is able to perform its required functions and transfer that information as machine-understandable knowledge about a failure mode. Function blocks (FBs), as per the IEC-61499 standard, have been proposed as a way to achieve distributed process planning in which the manufacturing process can adapt itself to runtime conditions, e.g. machine availability, etc. However, FBs are event-driven systems and the approach is limited to work under well-known runtime conditions, e.g. machine configurations and states, or deviations which are impossible to foresee in advance, for instance the outcome of a process failure mode effects analysis (PFMEA). The PKP introduced in this paper, aims at bridging this gap by integrating at runtime an expert operator's solution based on root cause analysis (RCA) in an FB architecture, making the FB knowledge-driven systems, for further executions of the same without redesigning it. Natural language representations of procedural knowledge blocks (PKBs) allow to transfer procedural knowledge to human operators, i.e. explain the process flow of a machine decision, while machine representations of PKBs allow to embed procedural knowledge that is elicited from expert operators upon unexpected events into the FBs process. The resulting PKP enhances the FBs for smart industrial applications, such as the process planning use case described in this paper.

Keywords: function blocks; procedural knowledge blocks; CAPP; FMEA; process planning; knowledge transfer; human-machine collaboration.

1. Introduction

Computer-aided process planning (CAPP) is the use of computer technology to support process planning in manufacturing. The first CAPP systems were conceived more than forty years ago. One of the earliest CAPP systems was the variant planning system CAM-I Automated Process Planning in 1976 [1]. A decade earlier, Niebel [2] discussed the use of computers for mechanized selection of optimum processes for manufacturing at design stage. Even though Niebel has been recognized as one of the first who discussed the use of computers in process planning [3], the impact of computers in manufacturing was addressed by Diebold [4] in 1952, thirteen years before Niebel, who in a "provocative study of the possibilities, limitations, and social and economic consequences of the revolutionary new machines of the electronic age" envisioned future computerized automatic factories. CAPP has been subject for research since the early 1970s and the first process planning system known for using artificial intelligence (AI) technology, GARI, was presented by Descotte *et al.* in 1981 [5]. The year after, Weill *et al.* published the first large survey of CAPP systems [6].

Several CAPP systems and CAPP research reviews, e.g. [2,3,6–13], have since been presented. Xu *et al.* [14] concludes that research has faced many still unsolved difficulties in attempting to implement AI techniques in CAPP. In a recent

10.1016/j.promfg.2020.05.148

review of the past 35 years 1981–2016 research on AI system applications in process planning and manufacturing, Leo Kumar [15] stated that a limited number of works has focused on knowledge acquisition methodology and key issues involved during knowledge acquisition.

In addition, lack of standardized definition of what process planning encompasses in terms of activities [16] puts up challenges in designing CAPP systems capable of supporting all kind of applications. Current knowledge-based methods are based upon expert knowledge codified in advance, e.g. as explained by Park [17]. This knowledge is "fixed" and is not extended or updated at runtime. Current CAPP approaches are also limited by the lack of ability to respond to unpredictable changes in daily industrial applications.

This paper addresses the question of capturing human expertise during common operations and not just at knowledge systems design stage, considering industrial operations as a dynamic procedural knowledge process (PKP). As emphasized by Gorecky et al. [18] and Thoben et al. [19] humans are the most flexible entity in the Industry 4.0 paradigm. In contrast to the technology oriented computer-integrated manufacturing (CIM) paradigm of the 1980s, humans are not to be simply replaced by artificial intelligence and automation on the shop floor but integrated into the cyber-physical structure in such a way that their individual skills and talents can be fully realized.

Human and machine collaboration is discussed by Zhong et al. [20] who suggested a "human-in-the-loop" machine learning approach as one future research direction under Industry 4.0. Advanced learning models for machines enable humans to interact efficiently and effectively in decision-making, for example traditional machine learning systems which are interjected with human knowledge to improve human-machine interactions and communications. However, a detailed outline of such collaboration is out of the scope for this paper and will be subject for further research.

Many researchers and philosophers have defined knowledge in various ways. Attempts to formalize knowledge started more than two thousand years ago. Aristotle's categories and his system of syllogisms for reasoning about the categories were the most highly developed systems of logic and ontology. Sowa among the latest ones, defined the *knowledge soup*, which explains how some knowledge may be represented in symbolic or propositional form, but much, if not most of it, is stored in image-like forms [21]. Making sense from an engineering perspective of the knowledge soup and unifying all the knowledge definitions might look challenging, nonetheless. Therefore, this paper proposes a functional redefinition of declarative and procedural knowledge, together with the concepts of procedural knowledge blocks and process as a framework to deal with knowledge processes in industry.

Function blocks (FBs), as per the IEC-61499 standard [22], can support intelligent and autonomous feature-based machining. FBs have been successfully used to manage process planning in a distributed manner [14] and allow for a better portability among resources, dynamism and adaptability [23]. This work points out a gap in dealing with unforeseen events at runtime. In fact, these systems are not specifically designed with an ability to define new states at runtime, a property that this paper points out is needed to be able to define and handle unforeseen events that might be happening during their execution, in accordance with the current state of the art.

This paper introduces in section 2 a framework for capturing procedural knowledge in a functional way through procedural knowledge blocks (PKBs). In section 3, FBs are introduced and it is explained how they can support distributed process planning. In section 4, it is briefly introduced how process failure mode effects analysis (PFMEA) and root cause analysis (RCA) contributes to the procedural knowledge process (PKP). Finally, the paper explains how PKBs can bridge a functional gap and enhance current FBs approaches by capturing human actions upon unexpected events and embed successful solutions as procedural knowledge into common FBs. A simple industrial manufacturing use case shows how an operator can stay in the loop and transfer their knowledge.

Abbreviations

AI	Artificial intelligence
CAPP	Computer-aided process planning
CIM	Computer-integrated manufacturing
CNC	Computer numerical control
DK	Declarative knowledge
ECC	Execution control chart
FB	Function block
FMEA	Failure mode effects analysis
FSM	Finite state machine
PFMEA	Process failure mode effects analysis
PK	Procedural knowledge
PKB	Procedural knowledge block
PKP	Procedural knowledge process
RCA	Root cause analysis

2. Procedural knowledge process

Engineering operates with models able to reproduce the functionality of real world objects or meta-objects. If knowledge is something that can be stored, computed, used and transferred, then there must be an engineering model that can do so. This paper aims at redefining declarative knowledge (DK) and procedural knowledge (PK) in order to introduce a framework called procedural knowledge process (PKP), which is based on the engineering use of the new definitions. A diagram called procedural knowledge block (PKB) is introduced to visualize DK and PK and their relationships. The concept of knowledge representation [24] with respect to *knowledge* is considered as part of the new definitions in order to be able to connect previous scientific work on the topic. A simple example, not pertaining the area of manufacturing, is provided in order to learn to reason intuitively in terms of DK and PK before applying them to the manufacturing use case presented later in this paper.

2.1. Declarative and procedural knowledge

Among the most recent definitions, Joachim Funke who is professor in theoretical and cognitive psychology defines knowledge as "a piece of subjectively acquired information about the world" that "can be construed as embodied

information" [25]. Nico Stehr who is a pioneer in the knowledge of societies defines knowledge as "the capacity to act, whereas information does not enable one to set anything in motion" [26]. Haapasalo et al. have done an extensively review on the matter of knowledge types and have showed that there is agreement on two kinds of knowledge: declarative (also called descriptive or conceptual) and procedural (also called practical) [27]. The most common distinction, originally from Skemp and Papert, is that declarative knowledge is identified as "knowing-that" or knowledge of facts, embodying concepts, principles, ideas, schemas and theories, while procedural knowledge is identified as "knowing-how", especially intended as how to perform some tasks [28–31].

The discovery that biology might have different structures to store declarative and procedural memories [32] might have encouraged the scientific community to separate the definitions of DK and PK, making it difficult for engineering to apply a unified type that can be used for practical purpose. Hence, a better definition of DK and PK needs to try to reconcile the differences as much as possible.

A more functional definition of declarative knowledge, starting from Funke's definition [25], needs to consider knowledge as organized information, subjectively acquired by a system, about another system, that is non-necessarily the world.

In order to make the definition applicable to any engineering systems, Funke's *subjective* or *world* knowledge has to be generalized to system knowledge. As by definition, a system is intended as a regularly interacting or interdependent group of items forming a unified whole [33].

In none of the studied literature there has been any reference to time within the knowledge definitions. Hence, a more functional definition of DK also needs to introduce time and characterize the system at an instant of time t in order to imply its stationary nature: a description of a system can only be valid if the system does not change or if it changes really slowly with respect to the time scale considered.

Due to the introduction of the time variable, there is a possibility to simplify the definition of PK. In fact, a better definition of PK can be considered as the variation of DK in a time interval. The advantage is that one definition is directly based on the other one and connected by a variation of time.

In light of these considerations, the following definitions are proposed:

Definition 1. Declarative knowledge is systemic information of a system A, related to information about a system B, at an instant of time t.

Definition 2. Procedural knowledge is systemic information of a system A, related to the variation of information about a system B, between two instants of time t_1 and t_2.

2.2. What is a knowledge representation?

An immediate advantage of the definitions introduced by this paper is that they remove the need to express knowledge and knowledge representations as two separate concepts. The reduction of ambiguity is vital in engineering applications.

In fact, imagine that a system A has knowledge about a system B. Then, there must be structural information instantiated in system A, i.e. a certain reorganization of its parts, in order to represent information about system B. Thereby follows that there is no knowledge without a representation.

This outcome helps to reconcile previous research presented in literature on knowledge and knowledge representations and can be postulated as the following definition:

Definition 3. A knowledge representation is the instantiation of information in a system through the reorganization of its parts.

This definition goes in line with a well-cited journal paper from Davis *et al.* [34] in which a knowledge representation is defined as:

- a surrogate;
- a set of ontological commitments;
- a fragmentary theory of intelligent reasoning;
- a medium for efficient computation;
- a medium of human expression.

In fact, if a knowledge representation is achieved in a system (surrogate) through the reorganization of its parts, i.e. a change of states, any systems can in theory represent knowledge. A system can be described by a set of ontologies. An intelligent system can act upon its own knowledge and become a medium for efficient computation. Moreover, humans, seen as systems, interact with other systems by means of language, images and every knowledgeable product of their being.

2.3. An intuitive view of DK and PK

The definitions of DK and PK can be interpreted in the following intuitive way: declarative knowledge is like having a snapshot of a system at a given time while procedural knowledge is like having a video sequence of a system within a time interval. Photos and videos are familiar systems to capture knowledge about systems, e.g. a kid playing in the sand, it is intuitive to grasp how they differ for the ability to capture knowledge. An image is static, as it does not express time, unless explicitly stated with tricks such as time-arrows or referring to actions taken by static figures. A video, on the other hand, is composed of successive frames (images). When they are seen in succession, they produce a dynamic experience that takes into account the flowing of time. Something that can be recognized as an event or an action.

As an intuitive example, imagine a man who is swimming in a calm lake. The man and the lake are two separate systems that can be described by means of language (a system). Reading this sentence changes the reader's system (brain) state and creates a mental image at time t that is declarative knowledge.

Bringing the example further, think about the man getting out of the lake at time $t+1$. The effect of this second declarative knowledge is an immediate interpolation of the two images as frames of a continuous video stream. Procedural

knowledge is generated by assuming that these two images are descriptions of a variation of states of the same systems (the man and the lake), happening between time t and time $t+1$.

This holds true even when the two states are equal but there is time running in between the states. For example, knowledge about a mobile system that instead maintains its spatial position during a period of time is also procedural knowledge, e.g. the man floating for a while at the same spot in the lake.

2.4. Procedural knowledge blocks

There is plenty of literature that ties actions with knowledge [35–37] in a superfluous way, however the practical application sought by this paper requires a formal connection between action, DK and PK.

It is a common habit to use an action (intended) or an event (unintended) to explain how time has progressed if a system varies in between two instants of time t_1 and t_2. Since this variation can be expressed by a PK, action/events and PKs are interchangeable ways to describe how a system evolves.

Going back to the example of the man swimming in the lake, the mental sequence of images (DKs) provided by the PK is a placeholder for an action/event label such as "a man is swimming in the lake".

The correspondence between PK and actions/events can be visualized as a procedural knowledge block (PKB), which is a diagram composed of two instances of DK at different times, separated by an action/event label that refers to a systemic transformation in the interval of time between them (see Fig 1).

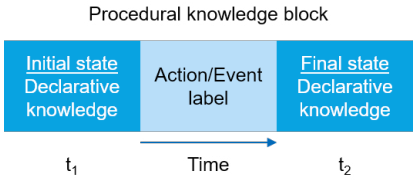

Fig 1. Procedural knowledge block. There are three components of a procedural knowledge block: The initial state of one or more systems at time t_1 is declarative knowledge; the action/event label that stands for what happens to the system(s) in the time interval between initial and final states; the final state of one or more systems at time t_2 is declarative knowledge.

The very correspondence between actions/events and PKs makes the PKB a conceptual module for knowledge-driven systems that can be integrated with or upgrade event-driven systems, such as the function blocks introduced in the next section.

2.5. Procedural knowledge transfer

Since both human language and machine language can quite precisely describe a system state, two corresponding PKBs can be used to refer to the same PK, under two different instantiations. One is used to communicate with humans, in natural language, the other is used to communicate among machines, through direct machine state representations.

An interesting observation related to automatic translations is that the approach of storing and using overall meaningful sentences in different languages has proven to be more efficient than word-by-word translations [38].

This intuitively leads to use a similar approach to transfer knowledge between operators and machines by means of PKBs which is conceptually demonstrated in this paper and will be explored further in future work.

The problem of transferring knowledge from humans to machines and the other way around can be thus reformulated into the problem of translating knowledge from one systemic instantiation to another.

2.6. Procedural knowledge process

A procedural knowledge process is a framework to structure any processes, especially the industrial ones, using the procedural knowledge blocks introduced before. In particular, there are two possible ways of connecting the blocks: in series or parallel. When the blocks are connected in series, the declarative knowledge part that ends the first block is the same that starts the second block. When two blocks are executed in parallel, the declarative knowledge at the beginning and at the end of both blocks coincides. A PKP is a diagram of a process that can be directly implemented as a system, exactly in the same way as a system can be described by a finite state machine (FSM) and then implemented.

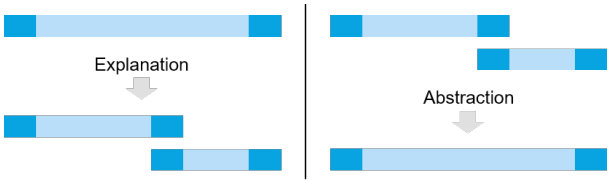

Fig 2. Explanation (left) and abstraction (right) of a process through procedural blocks. Explanation produces several procedural knowledge blocks starting from a single block. The initial state of the first produced block and the final state of the last produced block are the same of the original block. Abstraction merges several knowledge blocks into one, which is labelled after the most abstract action/event that leads the first block initial state into the last block final state.

By taking advantage of the holistic aspect of the PKBs, each of them can be replaced by several blocks. Such operation is called explanation, see Fig 2 (left). In reverse, several blocks can be aggregated into one. Such operation is called abstraction, see Fig 2 (right).

Thus, knowledge can be expressed at different detail levels or depths. Starting from the most abstract level of a given procedure, e.g. machining of a gear box housing, which can be represented by a single PKB, one can explain the process task by task in sequential or parallel operations represented each by one PKB. The resulting diagram is a PKP.

3. Function blocks

Function blocks are defined in the IEC-61499 standard [22] and can be immediately classified into two types: basic FBs and composite FBs. A basic FB is composed of two main parts that are the execution control chart (ECC), i.e. a finite state machine controller of the FB operations, and the core of the FB

containing the algorithms. Each basic FB, see Fig 3 (a), can encapsulate the algorithms needed to perform an operation that can be, for example, part of an assembly plan or a process plan for a specific assembly or process feature as shown by Wang *et al.* [39,40].

A composite FB, see Fig 3 (b), which is made of several basic FBs connected by events and data, relies on the internal states and embedded algorithms of the basic FBs. This allows to create a whole process made of several basic FBs that can be operated by one or more composite FBs, especially to handle a distributed process plan. In fact, the use of FBs solves the problem of handling the computer numerical control (CNC) language as machine neutral language. It adds portability in the form of FBs compatible to various machine tools with similar capability, where machine-specific data, e.g. cutting parameters or tool paths, are generated *ad hoc*. In case of a machine failure, the same function blocks can be redirected to another machine for local optimization and execution [40].

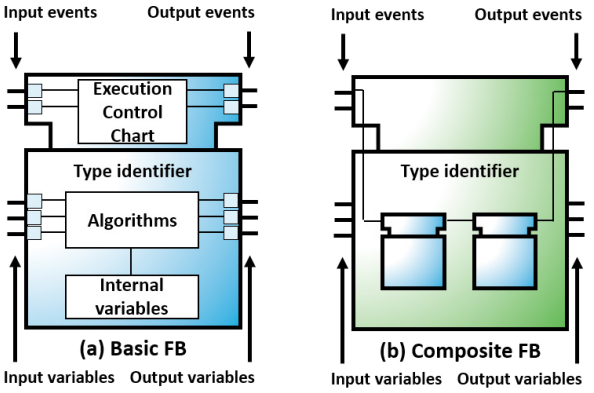

Fig 3. (a) Basic function blocks. Each FB is controlled by an execution control chart (ECC) that is event-driven. Input and internal variables influence the execution of the predefined algorithm embedded in the FB. (b) Composite function blocks. A composite FB contains several basic FBs and the ECC is replaced by interconnections between each of the embedded basic FB own ECC. A composite FB is capable of controlling an event-driven process flow.

However, it is important to note that a basic FB can produce different outputs even when the input is the same, depending on its internal state info. It is a characteristic dictated by the choice of defining an FB as an explicit event-driven model: when a trigger event is passed as an input, the FB controller starts the execution of its embedded algorithms, despite of the system conditions. It is the ECC that has the role to overview that the execution of the FB algorithm is completed in the best possible way given varying system conditions, i.e. external inputs and internal variables. Another way of seeing it is as a resource-driven model: once one or several resources are specified, the embedded algorithms of an FB will find the optimal parameters of the FB task for the given resources, e.g. machine. This, of course, allows for a great flexibility at runtime, but everything must be embedded in the FBs by design before execution, in order for it to handle all the varying but well-known runtime events. In fact, FBs are not structurally meant to define new and unexpected events at runtime, e.g. a product breaks during machining, and adjust the process to continue if the event was not foreseen at design stage. That is the part where a procedural knowledge process, introduced in section 2, can step in and bridge the gap, exploiting the pre-existing FB structure to instantiate the missing events. With this approach, in section 5, it is explained how FBs can be transformed from event-driven systems to knowledge-driven systems.

4. Handling faults with function blocks

4.1. Failure mode and effect analysis

Failure mode and effect analysis (FMEA), as defined in the ISO/IEC 31010 standard, is a technique to identify the ways in which components, systems or processes can fail to fulfill their design intent [41]. When talking about process planning, process failure mode and effect analysis (PFMEA) is often mentioned. PFMEA can be actively used to design and implement FBs architectures that are fault-resilient. However, there are no dynamic applications for it at runtime, when the FBs architecture is already designed and running a process. If and when a fault occurs at runtime, it is because of missing knowledge of such possibility at design stage, which was not spotted by the PFMEA. Major faults can trigger a redesign of the FBs architecture, which implies a stop of the operations that induces further losses.

4.2. Root cause analysis

Root cause analysis (RCA), as defined in the ISO/IEC 31010 standard, attempts to identify the root or original causes of a major loss instead of dealing only with the immediately obvious symptoms, in order to prevent its reoccurrence [41].

In order to include an RCA into an FBs architecture, the latter needs to be redesigned to include *ad hoc* FBs to respond to events generated by states that are discovered to be not desired, i.e. the root causes of successive losses. This is a complementary stage for the lean design of systems: RCA can spot missing knowledge that can be integrated in the design stage, when PFMEA is performed.

This opens up a gap for methods and subsequent systems that can elicit RCA knowledge from industrial actors and learn from it at runtime. Ideally, the outcome is a dynamical instantiation of new knowledge-driven FBs to handle the newly learned event, without the need to redesign the whole architecture.

5. Smart process planning via function blocks and procedural knowledge blocks

The demonstrative use of the framework proposed in this paper is to exploit skilled operators' knowledge to avoid critical faults in process planning by spotting and absorbing minor faults in the process at runtime before they become critical. The aid of PKBs in redesigning the FB process at runtime makes the FB process itself more adaptable and resilient to changing or new conditions. The research gap that is covered is about learning from unforeseen events at runtime through the operators' experience.

In this regards, the machining of a gear box housing is implemented as a use case to demonstrate how an unexpected event, as well as the successful actions of an experienced user upon the event, can be captured in one or several PKBs. The gear box housing has three bearing seats with a mutual functional relationship between the center-axes of the bearing seats. Most likely a four, or a five-axis milling machine would be a process planner's primary choice. However, as the intention with the case is to demonstrate how the PKP can capture unexpected events and successful actions upon them, the process planning scenario has been executed with a three-axis machine due to unavailability of a more versatile five-axis.

The choice of using the three-axis machine is also useful because it forces the involvement of an operator between each setup, which is essential for a human-machine knowledge transfer in form of PKBs between the operator and the automated FBs process.

5.1. Gear box housing machining process

The process plan for machining a gear box housing is made in seven setups. In the first setup the top face, and a step around the workpiece is machined. The step fulfills an important process function in serving as intermediate location surfaces in following setups. It is important to obtain an accurate surface when machining the top face (indicated in green in Fig 4) in setup 1, as this face is put against the solid jaw of the vise to achieve accurate location in setup 2 to 5. In fact, if the top face for some reason (which can be explained) becomes too rough in setup 1, that face might not fulfil its intended function as location surface roughness. An experienced operator may notice that the operation has not ended in an intended way. Secondly the operator may find the cause of the problem, eliminate and adjust the process.

5.2. Function blocks architecture

The FB architecture is introduced, as showed in Fig 5, to control the process from setup 1 to setup 7. The first function block in the simplified schema is not connected, assuming that the start signal will be given by other systems or manually by the operator. Every FB responsible for each setup operations is connected in cascade to the FB responsible for the next setup operations. These are to be considered composite FBs, as they coordinate the whole set of operations after a certain setup. Between setups, the operator is involved as they have to manually interact with the machine and the product to realize the setup. The FB simply waits for a confirmation of the operator, e.g. pushing a button, to proceed with the machining operations.

This simple process is repeatable, and the complexity of the FB structure can handle any foreseen event that can be added to it in design phase, but in case of unforeseen events there is no possibility to redesign the operations on the fly.

In order to see this, a finite state machine of the process can be drafted, see Fig 6. In fact, the ECC of an FB is nothing else than an FSM. The number of states is finite and consists of all the expected ones, by design.

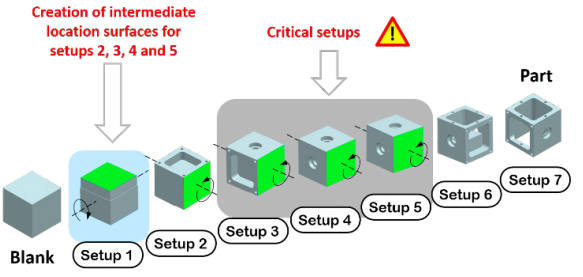

Fig 4. Process plan for a gear box housing produced with a 3-axis machine. Each setup corresponds to an operator involvement in the process. The smooth surface obtained after setup 1 is needed for good clamping in setups 2, 3, 4 and 5 and for the success of the relative FBs operations.

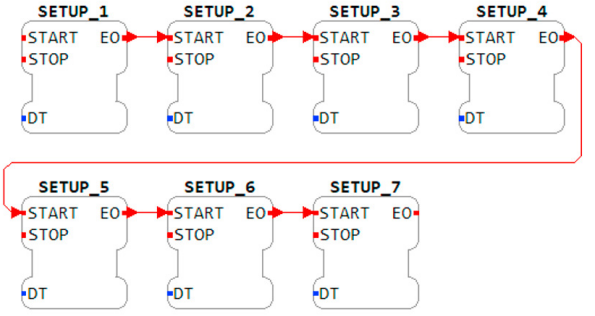

Fig 5. A simplified schema of composite FBs controlling the machining operation to produce a gear box with a 3-axis machine. Seven setups are required and manually executed by an operator, before that each FB starts controlling the CNC machine operations. In read, the event-driven flow of the process.

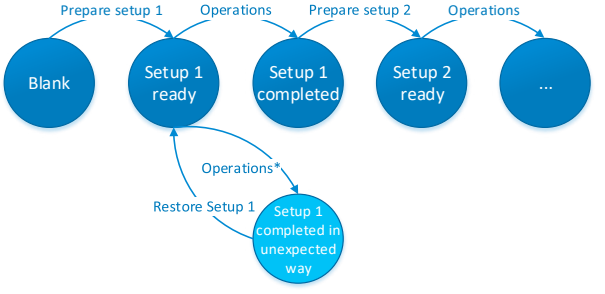

Fig 6. Finite State Machine. The first states of the machining process of the gear box housing are described through an FSM that is the equivalent of an ECC for an FB. In light blue the unexpected state that is reached with unrecognized operations* performed after setup 1. An expert actor can identify the unexpected state and restore setup 1 by expanding the FSM at runtime. This is possible with the joint use of procedural knowledge blocks and function blocks, as showed in this paper.

5.3. Elicitation of procedural knowledge blocks and instantiation of function blocks at runtime

Despite many good executions of the FB machining process, unwanted outcomes might still arise. For example, after the FB execution of the face milling operation after setup 1, the blank surface that is supposed to be smoothly machined as in Fig 7 appears instead as the rough surface showed in Fig 8.

In regular conditions, FBs are able to complete the process plan without errors. The flow of operations is continuous from setup 1 to setup 7. An actor intervenes at each setup and triggers the starting signal of the next function block (be that through a confirmation button or automated sensors). In case of failure, the only operation that is allowed by design is to abort the execution of the FB process plan or to restart it, placing a new blank and discarding the corrupted one.

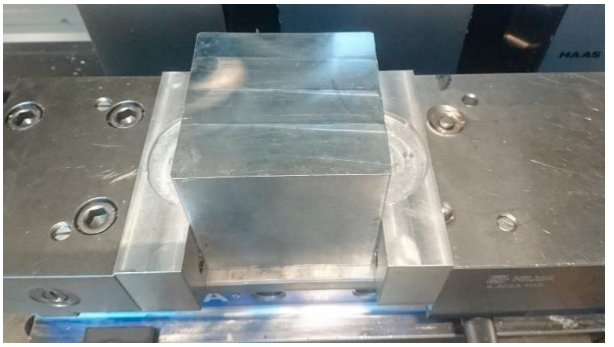

Fig 7. Outcome of face milling after setup 1: one of the blank surfaces is prepared for clamping.

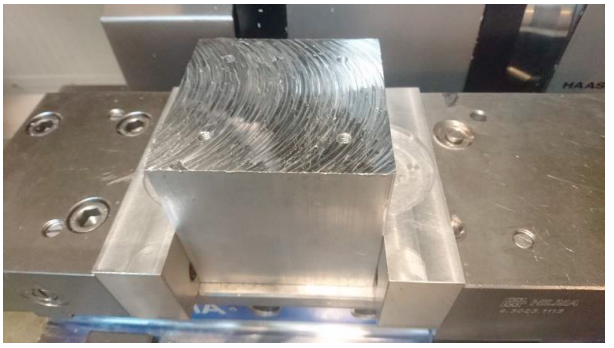

Fig 8. Unintended outcome of face milling in setup 1: Due to that cleaning of machine table has failed when the fixture was mounted; remaining swarfs contributes to create vibrations which causes the top surface of the blank unable to fulfill its location function. However, if not spotted by an operator, this failure will not prevent the process to advance where the rough surface of setup 1 will be the cause to a more critical failure in the critical setups 3, 4 or 5. Failure mode and effect analysis (FMEA) can be fixed at those stages, when it will be too late for the product to be saved but useful feedback can be learned from the production fault. On the other hand, anticipating the effect of an unexpected setup 1 would allow to save the product by restoring good conditions before the following critical setups.

Such a stop requires an actor to run an RCA and determine the cause of the error, then restore the process if possible or restart it. In either cases that the operator does or does not have such expertise, they can only restart the process and report the malfunction for someone else to analyze it and redesign the system. The risk is that the next process might fail again because the cause of fault has not yet been identified and might happen again. If this is the case, the production is stopped and the interruption might bring additional costs to those determined by the presence of a faulty product once and then, which in case of an aluminum gear box might be not expensive, but surely not affordable for a turbine component machined from a titanium block [42]. Both conditions can eventually be avoided by exploiting human expertise at runtime to prevent failure and improve the production process for the next executions.

Structuring the FB process as a PKP, allows to explain each FB operation by an associated PKB, as in Fig 9 (a). When the unexpected event occurs, an operator is requested to produce a PKB documenting the unexpected state with their own declarative knowledge. Furthermore, if an RCA can be successfully performed and a recovery procedure is known to the operator, it can be documented in another PKB for future executions, as in Fig 9 (b).

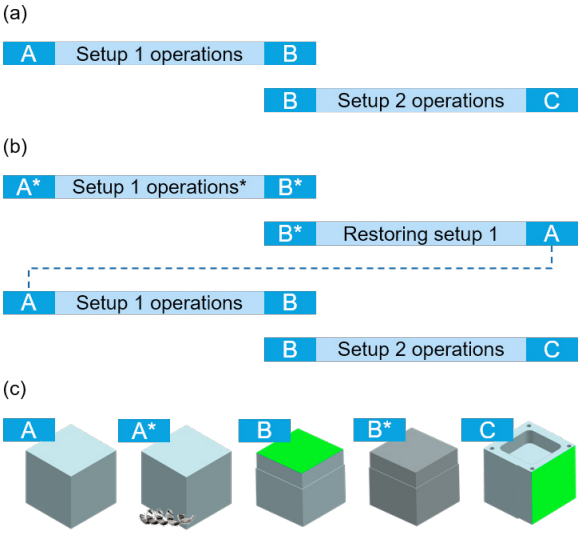

Fig 9. (a) Procedural knowledge process in regular conditions. (b) Procedural knowledge process with unforeseen event. An actor is called to identify the first block [A*]-[Setup 1 operations*]-[B*] and produce a corrective block [B*]-[Restoring setup 1]-[A] based on root cause analysis and own experience. (c) Descriptive knowledge of the procedural knowledge blocks. It can be represented by system states, determined by internal and input variables. A is the blank, A* is the blank and swarf in the machine, B is the correctly milled surface after setup 1, B* is the incorrectly milled surface after setup 1, C is the outcome of the machining operations after setup 2.

A simple scenario is introduced to explain the creation of new PKBs by the operators. The presence of swarf, state A* in Fig 9 (c), in the machine has led to a faulty execution of the machining operations after setup 1, see Fig 9 (b) and state B* in Fig 9 (c). The rough surface, showed in Fig 8, can be recognized as effect of something that went wrong, the swarf identified as the problem and setup 1 executed again to correct the smoothness of the surface, after cleaning the machine from any residuals of the previous operation, see state A in Fig 9 (c).

If setup 1 is not executed again in the correct conditions, then setups 3, 4 and 5 will create a product out of tolerance. For this, several PKBs are created, see also Fig 9, as follows:

- The creation of a PKB for a correct setup 1 can be done by an operator simply memorizing a natural language message of the kind "Please, operator, check that the machine is clean" in an available media support (text, video, audio, etc.).
- The PKB is directly translated in the instantiation at runtime of an FB that represents the event-driven manifestation of that PKB.

The FB needs to act as a placeholder for the presence of a PKB in the next executions of the FB architecture. For example, in case that the same event occurs during a process, the newly instantiated FB is called and this triggers the PKB known as "Please, operator, check that the machine is clean". Meanwhile, the FB architecture pauses on the instantiated FB while an operator needs to manually retrieve the PKB associated to the FB (it can be visualized on a screen, or a message on a speaker, etc.), interpret its instructions based on natural language, execute them and confirm the execution to the FB architecture so that the instantiated FB is considered executed.

This allows to keep the operators in the loop, while executing automated processes through FBs.

This structure is embedded in the FB architecture by adding at design phase an additional composite function block that runs the PKPs and can instantiate additional FBs representing each new PKB instantiated at runtime (PKB-FBs). The functionality to instantiate function blocks at runtime is present in the IEC-61499 standard. The PKB-FBs control the flow of the standard FBs through the additional events that they define. Such events are procedural knowledge-based and transform the FB architecture from an event-driven technology, to a truly knowledge-driven one.

An unexpected event is shaped as a PKB and this into a runtime-instantiated PKB-FB, that is set to be run in between regular FBs, thus redefining the flow of the FBs process, as described in the ECC or FSM (see Fig 6, state in light blue).

When similar conditions are assessed by the new PKB-FBs, the event takes place as expected, instead of unforeseen, and the process flow continues uninterrupted. Learning or knowledge transfer from the operator to the machine has happened.

6. Conclusions and future work

This paper redefines declarative and procedural knowledge and introduces the procedural knowledge blocks as useful diagrams to generate a procedural knowledge process, a framework to manage knowledge in industrial engineering processes.

The FBs are presented as a viable technology to implement smart CAPP systems in Industry 4.0, provided that they are enhanced with procedural knowledge capabilities as showed in this paper. It has been presented a way to use the procedural knowledge process framework to transform event-driven systems implemented with FBs to knowledge-driven systems implemented with PKBs and FBs. The clear advantage is the ability to represent and handle unforeseen events at runtime.

A simple process planning use case about the machining of a gear box housing is used as a proof of concept. The scenario requires unforeseen minor faults to manifest themselves before that critical faults can happen and that these are spotted by operators before irreparably damaging the product. The application allows to exploit the PKP framework to elicit knowledge from the operators and reshape the FB process at runtime, without interrupting the production or redesigning the FB architecture. The outcome is the preventive recovery from failure mode and knowledge transfer from humans to machines of the recovery operations performed by the operators on the shop floor.

Future work can be oriented in the following directions:

- Implementing a full demonstrator for the PKB instantiation at runtime that can be tested preferably on a real industrial case.
- Demonstrating the possibility to transfer knowledge between humans and machines by corresponding PKBs in different languages, i.e. respectively natural language and machine language, as proposed in section 2.5.
- Reshaping the whole process plan as a PKP in design phase, in order to make FB architectures with greater explainability and modularity. Research shall be conducted in such direction because procedural knowledge might have a determinant role in the way industrial processes are shaped and implemented in automated systems.
- Exploring further human-machine interaction and collaboration between operators and industrial systems with interfaces built with the aid of the PKP framework. Because efficient and explicit declarative and procedural knowledge transfer is needed to achieve meaningful collaborations.

References

[1] D.D. Bedworth, M.R. Henderson, P. Wolfe, Computer-integrated design and manufacturing, McGraw-Hill, 1991.

[2] B.W. Niebel, Mechanized process selection for planning new designs, ASME Pap. 737 (1965).

[3] L. Alting, H. Zhang, Computer Aided Process Planning: the state-of-the-art survey, Int. J. Prod. Res. 27 (1989) 553–585. https://doi.org/10.1080/00207548908942569.

[4] J. Diebold, The advent of the automatic factory, New York. (1952).

[5] Y. Descotte, J.-C. Latombe, GARI: A Problem Solver That Plans How to Machine Mechanical Parts., in: n.d.

[6] R. Weill, G. Spur, W. Eversheim, Survey of Computer-Aided Process Planning Systems, CIRP Ann. 31 (1982) 539–551. https://doi.org/10.1016/S0007-8506(07)60176-0.

[7] I. Ham, S.C.-Y. Lu, Computer-Aided Process Planning: The Present and the Future, CIRP Ann. 37 (1988) 591–601. https://doi.org/10.1016/S0007-8506(07)60756-2.

[8] A.H. Van'T Erve, Generative computer aided process planning for part manufacturing: An expert system approach, (1988).

[9] J.-P. Kruth, J. Detand, G. Van Zeir, J. Kempenaers, J. Pinte, Methods to improve the response time of a CAPP system that

generates non-linear process plans, Adv. Eng. Softw. 25 (1996) 9–17. https://doi.org/10.1016/0965-9978(95)00081-X.

[10] H.A. ElMaraghy, Evolution and Future Perspectives of CAPP, CIRP Ann. 42 (1993) 739–751. https://doi.org/10.1016/S0007-8506(07)62537-2.

[11] D. Kiritsis, A review of knowledge-based expert systems for process planning. Methods and problems, Int. J. Adv. Manuf. Technol. 10 (1995) 240–262. https://doi.org/10.1007/BF01186876.

[12] G. Halevi, R.D. Weill, Principles of Process Planning : a logical approach, Springer Netherlands, 1995.

[13] G. Halevi, G. Halevi, Process and operation planning, Kluwer Academic Publishers, 2003.

[14] X. Xu, L. Wang, S.T. Newman, Computer-aided process planning – A critical review of recent developments and future trends, Int. J. Comput. Integr. Manuf. 24 (2011) 1–31. https://doi.org/10.1080/0951192X.2010.518632.

[15] S.P. Leo Kumar, State of The Art-Intense Review on Artificial Intelligence Systems Application in Process Planning and Manufacturing, Eng. Appl. Artif. Intell. 65 (2017) 294–329. https://doi.org/10.1016/j.engappai.2017.08.005.

[16] M. Lundgren, M. Hedlind, G. Sivard, T. Kjellberg, Process Design as Fundament in Efficient Process Planning, Procedia Manuf. 25 (2018) 487–494. https://doi.org/10.1016/J.PROMFG.2018.06.126.

[17] S.C. Park, Knowledge capturing methodology in process planning, CAD Comput. Aided Des. 35 (2003) 1109–1117. https://doi.org/10.1016/S0010-4485(02)00182-3.

[18] D. Gorecky, M. Schmitt, M. Loskyll, D. Zühlke, Human-machine-interaction in the industry 4.0 era, in: Proc. - 2014 12th IEEE Int. Conf. Ind. Informatics, INDIN 2014, Institute of Electrical and Electronics Engineers Inc., 2014: pp. 289–294. https://doi.org/10.1109/INDIN.2014.6945523.

[19] K.-D. Thoben, S. Wiesner, T. Wuest, "Industrie 4.0" and Smart Manufacturing – A Review of Research Issues and Application Examples, Int. J. Autom. Technol. 11 (2017) 4–16. https://doi.org/10.20965/ijat.2017.p0004.

[20] R.Y. Zhong, X. Xu, E. Klotz, S.T. Newman, Intelligent Manufacturing in the Context of Industry 4.0: A Review, Engineering. 3 (2017) 616–630. https://doi.org/10.1016/J.ENG.2017.05.015.

[21] J.F. Sowa, The challenge of knowledge soup, in: Res. Trends Sci. Technol. Math. Educ., 2006: pp. 55–99.

[22] IEC61499 - international standard for distributed systems., (n.d.). http://www.iec61499.de/ (accessed November 30, 2019).

[23] X.W. Xu, Lihui Wang, Yiming Rong, STEP-NC and function blocks for interoperable manufacturing, IEEE Trans. Autom. Sci. Eng. 3 (2006) 297–308. https://doi.org/10.1109/TASE.2005.862147.

[24] R. Davis, H. Shrobe, P. Szolovits, What is a knowledge representation?, AI Mag. 14 (1993) 17.

[25] J. Funke, How much knowledge is necessary for action, Knowl. Action, 9th Ed.; Meusburger, P., Werlen, B., Suarsana, L., Eds. (2017) 99–111.

[26] N. Stehr, Knowing and Not Knowing, in: 2017: pp. 113–125. https://doi.org/10.1007/978-3-319-44588-5_7.

[27] L. Haapasalo, D. Kadijevich, Two Types of Mathematical Knowledge and Their Relation, J. Für Math. 21 (2000) 139–157. https://doi.org/10.1007/bf03338914.

[28] K.J. Holyoak, R.G. Morrison, The Cambridge Handbook of Thinking and Reasoning, n.d. www.cambridge.org (accessed November 24, 2019).

[29] R.R. Skemp, Intelligence, learning, and action : a foundation for theory and practice in education, Wiley, Chichester ;;New York, 1979.

[30] S. Papert, Mindstorms: Computers, children, and powerful ideas, NY Basic Books. (1980).

[31] E.D. Gagné, C.W. Yekovich, F.R. Yekovich, The cognitive psychology of school learning, 2nd ed., HarperCollins College Publishers, New York, 1993.

[32] M.T. Ullman, The Declarative/Procedural Model: A Neurobiological Model of Language Learning, Knowledge, and Use, in: Neurobiol. Lang., Elsevier Inc., 2015: pp. 953–968. https://doi.org/10.1016/B978-0-12-407794-2.00076-6.

[33] System | Definition of System by Merriam-Webster, (n.d.). https://www.merriam-webster.com/dictionary/system (accessed November 24, 2019).

[34] R. Davis, H. Shrobe, P. Szolovits, What Is a Knowledge Representation?, AI Mag. 14 (1993) 17–17. https://doi.org/10.1609/AIMAG.V14I1.1029.

[35] L. Morgenstern, Knowledge Preconditions for Actions and Plans, Readings Distrib. Artif. Intell. (1988) 192–199. https://doi.org/10.1016/B978-0-934613-63-7.50021-8.

[36] F. Stulp, M. Beetz, Combining declarative, procedural, and predictive knowledge to generate, execute, and optimize robot plans, Rob. Auton. Syst. 56 (2008) 967–979. https://doi.org/10.1016/j.robot.2008.08.011.

[37] R.J. Brachman, H.J. Levesque, M. Pagnucco, Knowledge representation and reasoning, Morgan Kaufmann Publishers Inc., 2004. https://dl.acm.org/citation.cfm?id=2821570 (accessed January 1, 2019).

[38] P. Resnik, O. Buzek, C. Hu, Y. Kronrod, A. Quinn, B.B. Bederson, Improving Translation via Targeted Paraphrasing, in: Proc. 2010 Conf. Empir. Methods Nat. Lang. Process., Association for Computational Linguistics, Stroudsburg, PA, USA, 2010: pp. 127–137. http://dl.acm.org/citation.cfm?id=1870658.1870671.

[39] L. Wang, S. Keshavarzmanesh, H.-Y. Feng, Design of adaptive function blocks for dynamic assembly planning and control, J. Manuf. Syst. 27 (2008) 45–51. https://doi.org/10.1016/J.JMSY.2008.06.003.

[40] L. Wang, Y. Song, Q. Gao, Designing function blocks for distributed process planning and adaptive control, Eng. Appl. Artif. Intell. 22 (2009) 1127–1138. https://doi.org/10.1016/J.ENGAPPAI.2008.11.008.

[41] ISO - IEC 31010:2009 - Risk management — Risk assessment techniques, (n.d.). https://www.iso.org/standard/51073.html (accessed November 29, 2019).

[42] G. Kappmeyer, C. Hubig, M. Hardy, M. Witty, M. Busch, Modern machining of advanced aerospace alloys-Enabler for quality and performance, in: Procedia CIRP, Elsevier B.V., 2012: pp. 28–43. https://doi.org/10.1016/j.procir.2012.04.005.