Wearable Expert System Development: Definitions, Models and Challenges for the Future

Fabio Sartori and Riccardo Melen

Department of Computer Science, Systems and Communication (DISCo)
University of Milan - Bicocca
viale Sarca, 336
20126 - Milan (Italy)
tel +39 02 64487910 - fax +39 02 64487839
{sartori,melen}@disco.unimib.it

Structured abstract

Purpose A Wearable Expert System (WES) is an expert system designed and implemented to obtain input from and give outputs to wearable devices. Among its distinguishing features are the direct cooperation between domain experts and users, and the interaction with a knowledge maintenance system devoted to dynamically update the knowledge base taking care of the evolving scenario.

Design/Methodology approach The WES development method is based on the KAFKA framework. KAFKA employs multiple Knowledge Artifacts, each devoted to the acquisition and management of a specific kind of knowledge. The KAFKA framework is introduced from both the conceptual and computational points of view. An example is given which demonstrates the interaction, within this framework, of Taxonomies, Bayesian Networks and Rule Based Systems. An experimental assessment of the framework usability is also given.

Findings The most interesting characteristic of WESs is their capability to evolve over time, due both to the measurement of new values for input variables and to the detection of new input events, that can be used to modify, extend and maintain knowledge-bases and to represent domains characterized by variability over time.

Originality/value Wearable Expert System is a new and challenging concept, dealing with the possibility for a user to develop his/her own decision support systems and update them according to new events when they arise from the environment. The system fully supports domain experts and users with no particular skills in knowledge engineering methodologies, to create, maintain and exploit their expert systems, everywhere and when necessary.

Keywords: Wearable Expert System, Knowledge-Based Systems, Wearable devices, Knowledge Acquisition Bottleneck, Knowledge Artifact

1 Introduction

The terms *wearable technology*, *wearable devices*, and *wearables* all refer to electronic technologies or computers that are incorporated into items of clothing and accessories which can be worn comfortably on the body. Wearable technology usually provides sensory and scanning features not typically seen in mobile and laptop devices, such as biofeedback and tracking of physiological function.

Thus, they could be profitably used in a number of applications, being important sources of data for reasoning. In particular wearables could be exploited in the design and implementation of a new breed of expert systems. An *expert system* is a computer system that emulates the decision-making ability of a human expert (Jackson, 1998). Expert systems are designed to solve complex problems by reasoning about knowledge, represented primarily as if-then rules rather than through conventional procedural code. A generic rule-based system consists of an inference engine and a knowledge base made of rules and facts to be analyzed.

In the Knowledge Engineering (KE) field, expert systems development has been always conceived as an *offline* process, characterized by intense Knowledge Acquisition and Representation phases, conducted by the *knowledge engineer* in order to make explicit the knowledge involved in a decision making process, which is tacitly owned by human experts. This is not a simple task: as a matter of fact, the *knowledge acquisition bottleneck* (KAB) is a well-known and still unresolved problem (Gaines, 2013).

This paper deals with development of *wearable expert systems* (WES). A WES is an expert system designed and implemented to obtain input from and give outputs to wearable devices. To interact with wearables, an expert system should be able to recognize how new data coming from them should be interpreted, in dynamically evolving scenarios.

We assume that a WES features the following distinguishing characteristics with respect to traditional expert systems:

- agile development cycle;
- scalable and time-evolving knowledge bases;
- interaction with a centralized, possibly cloud-based, knowledge maintenance system based on the *knowledge artifact* notion.

Agility principles can be a starting point to solve the KAB problem, given that traditional expert systems development is not agile by definition, due to the need for three distinct roles: the user, the domain expert and the knowledge engineer. WES development, instead, will be centered on domain experts and users cooperating directly in the system development process, evaluating its effectiveness on the field and modifying and updating it quickly when necessary. Here we find the typical features of agile system development: the focus on incorporating user requirement changes during the project life-cycle (Lee and Xia, 2010) as well as rapidly creating and embracing changes, learning from them (Conboy, 2009).

As stated above, WESs will be able to update automatically their knowledge bases according to new observations on the field. To do so, knowledge bases will be able to handle *events*, refining the more general concept of *fact* typical of traditional rule-based systems. Informally, while a fact can be described simply as the value assumed by a variable, without making explicit reference to time, an event occurs or materializes at a specific point in time, and the semantics of an event-based system depends strictly on the (partial) time ordering of the events handled. The flow of new events not only will cause the firing of knowledge base rules, but will also feed a knowledge maintenance process, consisting in the updating of existing rules or in the generation of new rules to take care of the evolving situation.

This automatic knowledge base maintenance will be accomplished by a sophisticated back-office system, based on a model which must be flexible and complete enough to comprise all the knowledge kinds involved in the decision making process. The *Knowledge Acquisition Framework based on Knowledge Artifact* (KAFKA) has been designed and implemented to this scope. The main characteristic of KAKFA is its non-monolithic approach: it features the *definition* and *correlation* of several different conceptual and computational tools for the acquisition and management of specific kinds of knowledge, depending on the domain.

In addition to that, the knowledge engineer and domain expert roles are unified into the *Knowledge Artifact Developer* (KA-Developer) actor, while the user is substituted by the *Knowledge Artifact User* (KA-User). A main goal of KAKFA is then reducing knowledge complexity and potential inaccuracy by eliminating the knowledge engineer role.

The rest of the paper is organized as follows: section 2 will review the literature about knowledge artifacts and other important research topics concerning the development of WESs. The conceptual model of Knowledge Artifact in KAFKA will be presented in sections 3. Section 4 will discuss about the implementation of wearable expert systems, focusing both on their architectural characteristics and on the algorithm implemented to interact with wearables. Section 5 will present a small application example to show in practice how a wearable expert system can be created exploiting KAKFA. Section 6 will evaluate the practical applicability of KAFKA, showing the results of experiments conducted on different groups of students: in particular, the scalability and agility level of expert systems developed in KAFKA will be evaluated, to demonstrate that they can be considered WESs. Finally, the paper will conclude with some considerations about future work.

2 Related Work

In a fundamental paper about the crisis of rule-based systems Clancey (1983) Clancey studied the knowledge representation in MYCIN, pointing out the different nature and roles of the rules encoded in its knowledge base. Clancey's analysis on MYCIN was crucial to highlight how the development of expert systems is not a mere translation of expert reasoning processes into sequences of rules, but a complex expert knowledge modeling activity, aiming at making a computer able to reason as a human expert.

Different kinds of knowledge require different methods and tools to acquire them: Knowledge Acquisition is indeed the most crucial step of every knowledge engineering process. As widely demonstrated in the past, failures in acquiring knowledge are the main cause of unsatisfactory knowledge-based systems development. The possible causes of the KAB are many (Wagner, 2004), and *knowledge inaccuracy* is indeed the most difficult to detect and solve among them. It can be caused both by experts and knowledge engineers. In the first case, *mistakes made by experts* have the consequence to produce wrong knowledge bases; in the second case knowledge maintenance activities can turn previously correct knowledge bases into incorrect ones, or difficulties may arise in modeling correctly the acquired knowledge.

Given that traditional methodologies proposed in the past as possible solutions to the KAB problem, such as CommonKads (Schreiber et al., 1994) and MIKE (Angele et al., 1998), have substantially failed their goal (Hoppenbrouwers and Lucas, 2009), the KAB continues to be a very important research subject. Gaines (2013) points out the continuing need to consolidate and extend all that we know about knowledge acquisition processes and techniques. Many recent efforts (Aussenac-Gilles and Gandon, 2013) indicate in the social web a possible solution to that problem, as a natural evolution of past wiki-based methods (Wagner, 2006). The main purpose of these semantic methods is to make the knowledge acquisition phase slimmer, delegating to the expert/user some functions traditionally in charge of the knowledge engineer.

This paper advocates the applicability of the *Knowledge Artifact* notion to solve the KAB problem. According to Holsapple and Joshi (2001), a knowledge artifact is an object that conveys or holds usable representations of knowledge. Salazar-Torres et al. (2008) argued that, according to this definition, KAs are artifacts which represent executable-encoding of knowledge, which can be suitably embodied as computer programs, written in programming languages such as C, Java, or declarative modeling languages such as XML, OWL or SQL.

According to these definitions, Knowledge Artifacts can be meant as guides to the development of complete knowledge-based systems. A relevant case study addressing this direction is the *pKADS* project (Butler et al., 2008), that provided a web-based environment to store, share and use knowledge assets within enterprises or public administrations. Each knowledge asset is represented as an XML file and it can be browsed and analyzed by means of an ontological map. Although the reasoning process is not explicitly included into the knowledge asset structure, this can be considered an example of Knowledge Artifact, being machine readable and fully involved in a decision making process development. Salazar-Torres et al. (2008) proposed a tabular Knowledge Artifact, namely *T-Matrix* to implement knowledge-based systems according to the design-by-adaptation paradigm. A T-Matrix describes products as recipes where ingredients and their amount are correlated to the different performance requirements that the final product should satisfy, providing a proper grammar to define those correlations and implementing it as rule-based systems.

As presented in Cabitza and Locoro (2014), knowledge artifacts can be classified according to both the research field they refer to and the application domains they are used in. In particular, the authors grouped KA experiences into five conceptual clusters, where different KAs are used with different scopes, on the basis of *objectivity*[1] and *situativity* [2]dimensions: *Artificial Intelligence* (AI-KAs from now on), *Knowledge Management*, *Computer Supported Cooperative Learning*, *Information Systems* and *Computer Supported Cooperative Work*. Here, we are mainly interested in AI-KAs, devoted to design and implement decision support systems, expert systems and ontologies in many domains, like engineering design (Bandini and Sartori, 2010), furniture manufacturing (Cheatle and Jackson, 2015), healthcare (Eysenbach et al., 2007), emergency management (Fogli and Guida, 2013). AI-KAs seem to be characterized by a higher level of objectivity than situativity; this means that the knowledge engineer role remains fundamental, and the KA is a sort of passive tool helping him/her in the knowledge-bases definition, creation and maintenance.

With respect to existing AI-KAs, the challenge is defining new methodological tools to make them active entities in the knowledge engineering process. To do so, it is crucial to be able to distinguish the different kinds of knowledge involved in the decision making process and produce proper KAs for them. These KAs must be correlated, in order to recognize changes in the whole knowledge domain and propagate such changes from a KA to the others; doing so, KAs could manage *events*, with the consequence to become important tools for creating wearable expert systems.

According to Luckham (2002), an *event* is an object acting as a record of activity in a system. Events are related to other events by *time*, *causality* and *aggregation*. WESs can take advantage of solutions developed in other research fields oriented to modeling event-driven processes, like *Information Flow Processing* (IFP). Cugola and Margara (2012) use this term for indicating applications that require to process continuously data from geographically distributed sources at unpredictable rates, in order to obtain timely responses to complex queries. Moreover, they define *IFP engine* a tool capable of timely processing large amounts of information as they flow from the periphery to the center of the system.

Within the IFP domain, Complex Event Processing (CEP) is defined in Mehdiyev et al. (2015) as a novel and promising methodology that enables the real-time analysis of *stream event data*, with the main purpose to detect complex event patterns from the atomic and semantically low-level events, such as sensors, log or RFID data. CEP involves rules to aggregate, filter and match low-level events,

[1] Authors define objectivity as the *capability of the KA to represent true facts in an objective, crisp and context independent manner*.

[2] Authors define situativity as the *capability of the KA to adapt itself to the context and situation at hand*.

coupled with actions to generate from them new, higher level events (Robins, 2010).

A knowledge engineering methodology to develop executable knowledge-based systems, addressing issues quite similar to CEP system design and implementation, is fundamental to design and implement wearable experts systems. Conceptual tools for the acquisition of different kinds of knowledge should be used, to fully understand how an event is characterized, which relationships with other events should be considered and how these relationships change as the system evolves. The former kind of knowledge can be identified by *functional knowledge* (FK) (Kitamura et al., 2004), the latter by *procedural knowledge* (PK) (Surif et al., 2012). Finally *experiential knowledge* (EK)(Niedderer and Reilly, 2010) allows modeling events and relationships among them with a rule-based system variable over time, that is a wearable expert system.

In KAFKA, aggregation is provided by *ontologies* (Brewster and O'Hara, 2004), time and causality are tackled by *Bayesian Networks* (Korb and Nicholson, 2010). With respect to similar approaches (Margara et al., 2014; Teymourian and Paschke, 2009; Mehdiyev et al., 2015) that aim at automatically derive rule patterns by means of machine learning techniques, our approach is interested in the analysis of different kinds of knowledge involved in problem solving activities. The relationship between knowledge and data is then explored to understand how the variability in the data stream can drive the evolution of the related knowledge bases.

3 The KAFKA Conceptual Model

The KAFKA model represents several kinds of knowledge, employing a specific modeling methodology for each one, and defines the necessary correlations when the same system entities are handled within different knowledge models.

Figure 1 shows the multi-layered model of Knowledge Artifact in KAFKA (Sartori and Melen, 2015). Given a particular kind of knowledge k_i, $i \in [1, ..., n]$, a Knowledge Artifact for the acquisition of k_i is the pair

$$KA_{k_i} = \{E^{KA_{k_i}}, R^{KA_{k_i}}\}, \tag{1}$$

where

$$E^{KA_{k_i}} = \{e_j^{KA_{k_i}}\}, j \in [1, ..., m] \text{ and } R = \{r_k^{KA_{k_i}}\}, k \in [1, ..., s], \tag{2}$$

are the sets of *entities* and *relationships* among them respectively, with

$$r : e_1^{KA_{k_i}} \times ... \times e_{j-1}^{KA_{k_i}} \longrightarrow e_j, \forall r \in [r_1^{KA_{k_i}}, ..., r_s^{KA_{k_i}}]. \tag{3}$$

Note that in the following we will use the terms *entity* meaning a general category of "thing": within it, we will distinguish between *events* and *variables*,

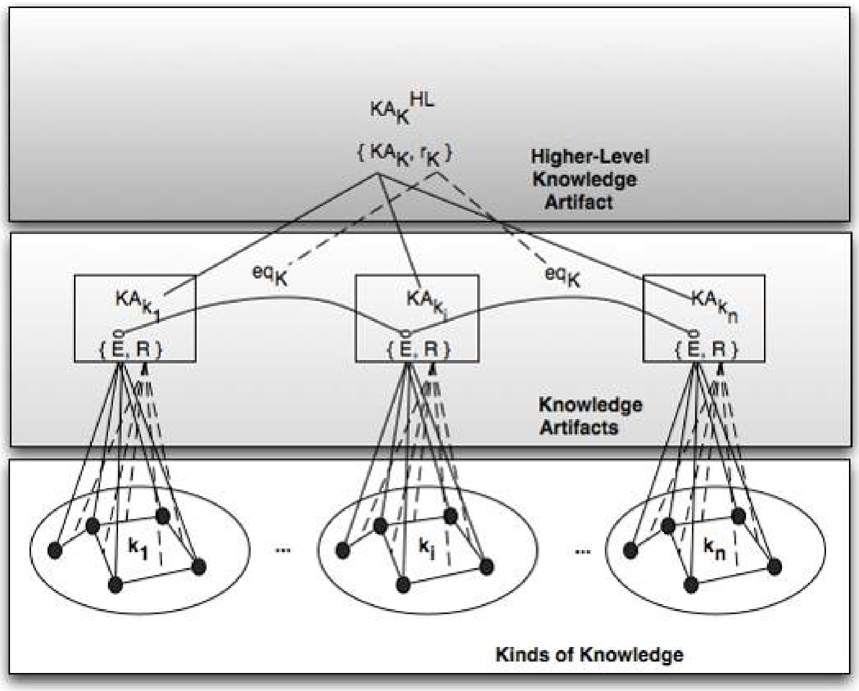


Fig. 1: The multi-layered model of Knowledge Artifact in KAFKA.

the former possessing the distinguishing feature of being related to a time of occurrence, the latter modeling the state of parts of the system and of its environment.

In order to correlate different KAs, it is necessary to extend the definition above: given a set of kinds of knowledge $K = \{k_1, ..., k_n\}$ and Knowledge Artifacts $KA_{k_i}, \forall i \in [1, ..., n]$ a *Higher-Level Knowledge Artifact* (HLKA) for the acquisition of K is the couple

$$KA_K^{HL} = \{KA_K, r_K\}, \tag{4}$$

where

$$KA_K = \{KA_{k_i}\} \text{ and } r_K : E^{KA_{k_i}} \longrightarrow E^{KA_{k_j}}, \text{ with } i, j \in [1, ..., n] \tag{5}$$

The way the correlation r_K is defined depends on the knowledge domain: at the current state of development, there exists only one correlation in KAFKA, namely eq_K (i.e. *equivalence on K*):

$$eq_K : E^{KA_{k_i}} \longrightarrow E^{KA_{k_j}} | e_l^{KA_{k_i}} \equiv e_p^{KA_{k_j}}, \tag{6}$$

with

$$e_l^{KA_{k_i}} \in E^{KA_{k_i}} \text{ and } e_p^{KA_{k_j}} \in E^{KA_{k_j}} \tag{7}$$

The equivalence notion allows managing the same entity differently according to the specific Knowledge Artifact, but preserving its main features moving from a kind of knowledge to another.

The general model presented here must then be configured on the basis of the knowledge involved: the HLKA acts as a library of Knowledge Artifacts, each of them chosen to model the related kind of knowledge in the best way. For example, Procedural Knowledge can be captured by many kinds of tools for modeling causal relationships, like e.g. Influence Nets (Rosen and Smith, 1996), Petri Nets, Bayesian Networks (Korb and Nicholson, 2010), Causal Nets (Van Der Aalst et al., 2011) or Superposed-Automata Nets (de Cindio et al., 1981).

The HLKA must then be configured in order to become a complete model for acquiring and relating all the kinds of knowledge involved in the development of knowledge-based systems.

We are employing the KAFKA modeling approach for a wide set of applications, including WESs, were:

- FK will be acquired by means of *taxonomies* (T), characterized by three kinds of entities and two relationships;
- PK will be modeled through *bayesian networks* (BN), characterized by three kinds of entities and two relationships;
- EK will be captured by *production rules* (PR), characterized by three kinds of entities and one relationship.

Thus, given the set $K = \{FK, PK, EK\}$, the KAFKA Higher-Level Knowledge Artifact to deal with K is $KA_K^{HL} = \{\{T_{FK}, BN_{PK}, PR_{EK}\}, eq_K\}$, where $eq_K : T_{FK} \longrightarrow BN_{PK}$ and $eq_K : BN_{PK} \longrightarrow PR_{EK}$ makes equivalent the event sets of T_{FK}, BN_{PK} and PR_{EK}.

The T_{FK} scope is the clear identification of the entities relevant to the system. As stated above, Wearable Expert Systems manage both variables, describing the environment and the system state, and timed events. Variables and events may have different roles in the system: they can be *inputs* (I), *outputs* (O) or *partial outputs* (PO). An input is a variable describing the environment in which the system operates or a possible domain restriction, or it can be an event, modeling an observation on the field, typically detected by a wearable device or manually introduced by the user. An output is a result of the computation, returned to the user as a command-type event in order to solve his/her problem, or as a generic alarm-type event if the management of the situation is the complete responsibility of the user. A partial output is the result on an internal computation, and often represents a state variable accumulating "partial" or "intermediate" knowledge about the history of the system and the future actions to be undertaken.

The T_{FK} classifies variables and events in I, PO and O, and describes how they are possibly *aggregated into* or *derived from* other entities:

$$KA_{FK} = T_{FK} = \{E^{T_{FK}}, R^{T_{FK}}\}, \tag{8}$$

with

$$E^{T_{FK}} = \{I \cup PO \cup O\}, \tag{9}$$

and

$$R^{T_{FK}} = \{is\text{-}a_{E^{T_{FK}}}, part\text{-}of_{E^{T_{FK}}}\} \tag{10}$$

where

$$is\text{-}a_{E^{T_{FK}}}, part\text{-}of_{E^{T_{FK}}} : E^{T_{FK}} \longrightarrow \emptyset \cup \mathcal{P}(E^{T_{FK}}) \tag{11}$$

with

$$\mathcal{P}(E^{T_{FK}}) \text{ the set of all the subsets of } E^{T_{FK}} \tag{12}$$

In addition to elementary entities, there are many situations in which more complex definitions are needed: the is-a relationship allows to extend the description of an existing input, partial output or output by means of a new value; the part-of relationship allows to aggregate two or more inputs, partial outputs or outputs into a new record-like element.

The Bayesian Network model is a structured process that allows to analyze complex problems of cause-effect type in order to infer hidden variables, assess expectations and determine an optimal strategy for the execution of certain actions. The *KAFKA Bayesian Network* (KBN) is the Knowledge Artifact adopted for procedural knowledge representation:

$$KA_{PK} = KBN_{PK} = \{E^{KBN_{PK}}, R^{KBN_{PK}}\}, \tag{13}$$

with

$$E^{KBN_{PK}} = \{I \cup PO \cup O\}, \tag{14}$$

and

$$R^{KBN_{PK}} = \{parent_{E^{KBN_{PK}}}, cpt_{E^{KBN_{PK}}}\}, \tag{15}$$

where

$$parent_{E^{KBN_{PK}}} : E^{KBN_{PK}} \longrightarrow \emptyset \cup \mathcal{P}(E^{KBN_{PK}}) \tag{16}$$

with

$$\mathcal{P}(E^{KBN_{PK}}) \text{ the set of all the subsets of } E^{KBN_{PK}} \tag{17}$$

and

$$cpt_{E^{KBN_{PK}}} : E^{KBN_{PK}} \longrightarrow P(E^{KBN_{PK}}|E^{KBN_{PK}}) \in [0,1] \tag{18}$$

The $parent_{E^{KBN_{PK}}}$ relationship returns, for each entity, the set of entities having a causal relation to it. Given an entity $e_j^{KBN_{PK}} \in E^{KBN_{PK}}$

$$parent_{e_j^{KBN_{PK}}} \begin{cases} = \emptyset, & \text{if } e_j^{KBN_{PK}} \in I, \\ \subset \mathcal{P}(E^{KBN_{PK}}), & \text{otherwise.} \end{cases}$$

The $cpt_{E^{KBN_{PK}}}$ stores, for each entity, the related *conditional probability table*, depending on the number of *entity outcomes* (i.e. the possible values assumed by the event or variable) and the number of *parents' outcomes*.

In particular, given an entity $e_j^{KBN_{PK}}$ we have

$$cpt_{e_j^{KBN_{PK}}} = \{P(e_j^{KBN_{PK}} \mid parent_{e_j^{KBN_{PK}}})\} \tag{19}$$

with

$$\sum_{j=1}^{\#e_j^{KBN_{PK}}} P(e_j^{KBN_{PK}} \mid parent_{e_j^{KBN_{PK}}}) = 1 \tag{20}$$

where

$$\#e_j^{KBN_{PK}} \text{ is the number of outcomes of } e_j^{KBN_{PK}} \tag{21}$$

Two observations are needed about the KBN. First, the *parent* relationship organizes the nodes of the model into a graph: this must be a DAG (Directed Acyclic Graph) to obtain a proper BN (this condition has not been introduced explicitly in the above formalism in order to keep it readable). A second, fundamental observation regards the use of causal relations as the guideline to build the KBN graph. It is well known that in some applications (e.g. in medical diagnosis) it may be difficult to determine the correct direction of causal relations, however in the practical cases that we have considered, clear causal (or logical precedence) relations were always available to guide us in the definition of the KBN.

Finally, Production Rules are used to describe how the causal process defined by a given KBN is modeled. Given that a KBN describes the steps necessary to reach outputs from inputs, it is also evident that the KBN allows identifying the different layers of the system to be developed, as well as possible parallelisms in sytem execution.

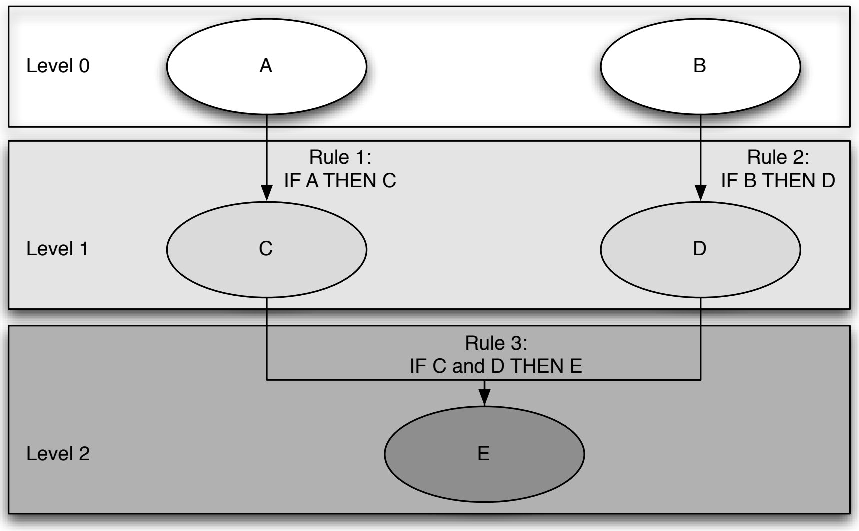


Fig. 2: The relationship between KBN and Production rules in KAFKA

Figure 2 shows a sketch of a KBN with two inputs (nodes A and B), two partial outputs (nodes C and D) and an output (node E). Rule 1, 2 and 3 allow specifying how the transitions described by KBN are implemented. Moreover, rules allow a clear identification of all the computational levels of the system: in particular, while Rule 1 and Rule 2 could be executed in parallel, it is evident that Rule 3 must wait that both Rule 1 and Rule 2 complete their execution, because its left hand side depends on their right hand sides.

This way, it is possible to identify easily the different computational levels of the system, where a computational level contains all the rules that can potentially be executed at the same time. There exist two particular computational levels in KAFKA: *Level 0*, where no rule is present, that is the level where inputs are collected, and *Level n* (n = 2 in Figure 2) where only one rule can be executed (for each output), that is the level where outputs are finally generated once all the necessary inputs and partial outputs have been provided.

According to the definition above, the third Knowledge Artifact in KAFKA is

$$PR_{EK} = \{E^{PR_{EK}}, R^{PR_{EK}}\}, \tag{22}$$

with

$$E^{PR_{EK}} = \{I \cup PO \cup O\}, \tag{23}$$

and

$$R^{PR_{EK}} = \{if\text{-}do_{LHS,RHS}\}, \tag{24}$$

where

$$if\text{-}do_{LHS,RHS} : \mathcal{P}(E^{KBN_{PK}}) \longrightarrow E^{KBN_{PK}} \tag{25}$$

The *if-do* relationship semantics is the definition of a left-hand side (LHS), specifying a set of evidences to be verified and a right-hand side (RHS), specifying one and only one action to execute in case of LHS verification.

4 Implementing WES in KAKFA

As previously stated, wearable expert systems are characterized by an agile development cycle, scalable and time-evolving knowledge bases and a cloud-based knowledge maintenance system. The adoption of KAFKA allows to consider all these aspects into a unique conceptual framework, the higher-level knowledge artifact: indeed, agility is incremented thanks to the direct interaction between domain expert and system user, while the time-evolving knowledge bases requirement is obtained trough the exploitation of bayesian networks in procedural knowledge modeling. The last point must be obtained from the implementation strategy.

The upper part of Figure 3 shows the computational model of WES development in KAKFA, according to the client-server paradigm: the three-tier architecture of the HLKA is implemented on the server, managed by the *KA-Developer*. The client side is managed by the *KA-User*, which is responsible for the definition of inputs to execute the expert system and gets outputs from it. In our implementation the KA-User sends data serialized into a JSON[3] object. JSON is an open standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs. These data are observations about the conditions of the problem domain. The GSON[4] library has been integrated to convert Java objects automatically into their JSON representation.

The bottom of Figure 3 shows the relationship between KA-Developer and KA-User as a UML class diagram. The KA-Developer is the server side of the framework. Its heart is the BayesNet class, which is responsible for the KBN creation and updating. When a new KBN is created, the KA-Developer allows specifying its nodes and parent relationships among them (i.e. the KAFKA taxonomy). The KBN is then stored into both XML and CSV files: the XML file is useful for parsing the KBN, the CSV file is necessary to allow managing the KBN by means of JAYES [5], our current *Bayesian network manager*. The knowledge engineer role is substituted by the KBN, which is responsible for the automatic generation of rules starting from the defined taxonomy. Doing so, knowledge inaccuracy is significantly decreased: the domain expert must only identify the inputs, partial outputs and outputs of the system, together with

[3] JavaScript Object Notation, see `http://json.org/`
[4] See `https://sites.google.com/site/gson/gson-user-guide#TOC-Goals-for-Gson`
[5] See `http://www.eclipse.org/recommenders/jayes/`.

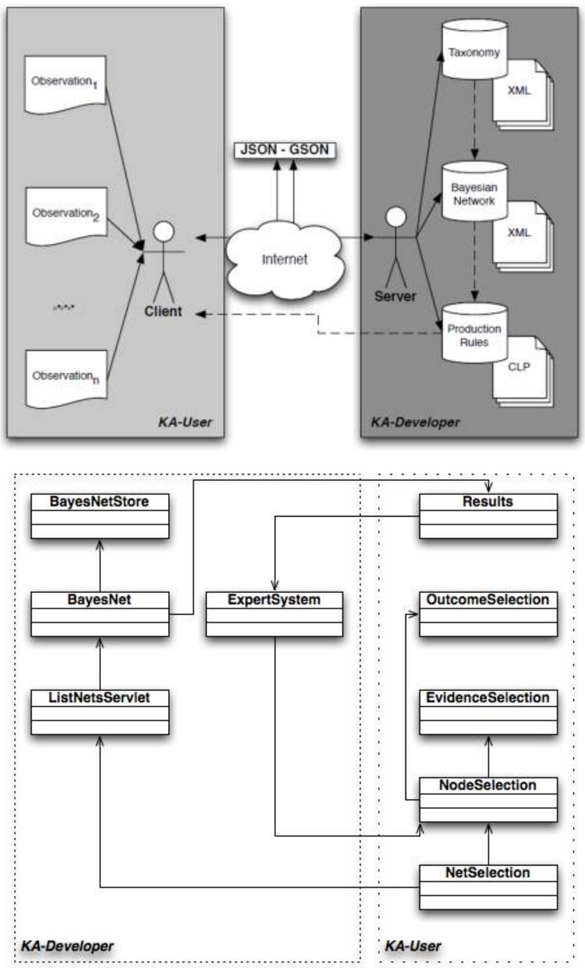


Fig. 3: The roles involved in the KAFKA knowledge engineering process and a sketch of the KAFKA UML class diagram.

their possible values and parent relationships. The KAFKA framework automatically generates a Bayesian network according to such specifications: at the beginning of the reasoning process, the related Conditional Probability Tables (CPT) are filled with uniform probability distributions. The CPTs are continuously updated according to the evidences and outcomes settings. The Bayesian network manager exploits the designed KBN to generate the rules of the expert system: these rules have inputs and partial outputs in the LHS and partial outputs and outputs in the RHS. Each rule is completed by a *reliability* value, computed on the basis of the CPT status. The rule-based system is exported as a Jess [6] clp file. The clp file is then sent to the KA-User serialized into a JSON-GSON object, and run on the KA-User, if Jess 8.0 (that is fully compatible with Android OS) is available on the related wearable; otherwise, it can be executed by the KA-Developer on the remote server, returning selected outputs to the KA-User by means of suitable JSON-GSON objects.

4.1 Initial configuration

For the purpose of computation, taxonomies and KBN elements of the HLKA have been represented in XML. Production rules are automatically generated starting from the KBN according to the Jess syntax.

The XML file structure for T (on the left) and KBN (on the right) is the following one:

```
<ontology>                                      |      <kafkaBayesianNetwork>
    <name> ... </name>                          |         <name> ... </name>
    <description> taxonomy </description>       |         <description> ... </description>
    <input>                                     |         <node>
        <name> ... </name>                      |            <name> ... </name>
        <is-a> ... </is-a>                      |            <value> ... </value>
        ...                                     |            ...
        <part-of> ... </part-of>                |            <parent> ... </parent>
        ...                                     |            ...
    </input>                                    |            <probabilities>
    <partialOutput>                             |            ...
        <name> ... </name>                      |            </probabilities>
        <is-a> ... </is-a>                      |         </node>
        ...                                     |         ...
        <part-of> ... </part-of>                |      </kafkaBayesianNetwork>
        ...                                     |
    </partialOutput>                            |
    <output>                                    |
        <name> ... </name>                      |
        <is-a> ... </is-a>                      |
        ...                                     |
        <part-of> ... </part-of>                |
        ...                                     |
   </output>                                    |
</ontology>                                     |
```

According to the conceptual model described so far, the taxonomy XML file is a collection of inputs, partial outputs and outputs, possibly linked by is-a and part-of relationships: given a specific node, an is-a tag will specify another node of the taxonomy that it extends; a part-of tag will specify a node of the taxonomy it contributes to describe. A node can have more than one is-a and part-of tags, or zero at all: this means that a set of isolated nodes may be a valid taxonomy in KAFKA. The is-a and part-of relationships are useful in the definition of possible values for evidences and outcomes in the KBN description step.

According to the HLKA definition, the eq_K relationship states that the input, partialOutput and output sets are the same as described in the taxonomy, given that different kinds of information are associated with them. In particular, one or more $< value > ... < /value >$ tags are used to specify the outcomes of

[6] See http://herzberg.ca.sandia.gov/.

each node and one or more $< parent > ... < /parent >$ tags are exploited to describe causal relationships among nodes. The KBN definition is completed by the specification of the Conditional Probability Table: given that the initial CPT is uniformly distributed, Alg. 1 allows to build up the initial CPT to start the rules generation.

Algorithm 1 Initial CPT Computation in KAFKA

```
for $j \in I \cup PO \cup O$ do
    $pOutcomes \leftarrow 1$;
    for $i \in parents(j)$ do
        $pOutcomes \leftarrow pOutcomes * outcomes(i)$
    end for
    $tConfigurations \leftarrow outcome(j) * pOutcomes$
    $probability_j = 1/outcome(j)$
    $CPT_j = [tConfigurations]$
    for $k \in [0, tConfigurations)$ do
        $CPT_j[k] \leftarrow probability_j$
    end for
end for
```

The KBN is browsed from the first node to the last one, and for every $<$ $parent > ... < /parent >$ tag of the current node, the total number of configurations (i.e. $tConfiguration$) is the product of the number of outcomes of the node (i.e. the result of the $outcome(j)$ function) and the number of outcomes of each parent (i.e. the value of the $pOutcomes$ variable). The tConfiguration variable is used to create CPT_j, namely the *Conditional Probability Table* for the node j, as an array of uniform $probability_j$ values.

Finally, all the possible production rules are generated starting from a KBN: each rule is made of LHS and RHS, subsets of evidences and outcomes respectively, and a reliability value is associated to it. Further details about the rule generation process are given below.

4.2 Rules Generation

Given the taxonomy T and the related KBN with its CPT, a new rule-based system can be finally generated. The mechanism through which this task is automatically made in KAFKA is the following one:

1. the KBN nodes acting as inputs of the system are selected;
2. all the possible configurations of inputs values previously identified are generated;
3. each possible input configuration is passed to the KBN, which infers the *a posteriori probability distributions* of output nodes;
4. such distributions are used to generate production rules: the right part of the conditional probability formula becomes the *left hand side* of the rule,

while the left part of the formula becomes the *right hand side* of the rule. The probability value associated to the output node in the CPT is taken as a measure of *rule reliability* that can be exploited to define different mechanisms of rule extraction/firing;

5. the procedure at point 4 is repeated for every possible configuration, originating a set of rules that represent an exhaustive expert system, that can be run on the observed input values.

The real-time adaptation of the rule-based system with respect to the user depends on the CPT updating algorithm, exploiting data collected by him/her. The rule reliability varies over time: when the KBN is asked to update the CPTs, the probability distributions are updated accordingly, making a rule less or more reliable than before. In this sense, an expert system generated in KAKFA dynamically changes its knowledge base according to the events occurring during its execution, given that the reliability of each rule varies. The user can select the desired reliability value as a threshold, in order to exclude from the execution step portions of the knowledge base. What is interesting in this scenario is that a rule discarded at the *i-th* execution step could be selected again at the *j-th* execution step, with $i < j$.

5 An Application Example

Our application example is related to the analysis of urban traffic in the town of Bergamo. The town is characterized by the division in two parts: the lower town, where most public services are located, and the upper town, more appealing from the tourist perspective. Nowadays, the need of having recommendations about mobility in the urban context is greater than ever, due to the ever-growing metropolitan areas, with higher population density. Such kinds of applications should be able to exploit real-time information about traffic congestion, physical-psychological state of the user, weather conditions, and so on. Each observation could be collected from personal devices, like smartphones and wearable devices, or hand inserted by the user.

A simple prototype composed by a KA-User and a KA-Developer has been implemented. The KA-User goal is to assist a person in reaching the upper town of Bergamo from the lower town using one of the three possibilities available (i.e. bus, funicular and stairs); the KA-Developer aim is to generate the expert system to suggest the best alternative.

The model comprises four entities:

- *Agent type* (AT), a parameter specifying the kind of user. Such entity can assume a value among *tourist*, *citizen* and *hinterland*;
- *Queue tolerance* (QT), a variable describing the capability of the user to maintain self-control in case of critical traffic conditions. Its value can be *high*, *moderate* or *low*, depending on *Agent type*;
- *Queue status* (QS), an input measurement sampling the traffic conditions. Possible values are *high*, *moderate*, *low* or *null*;

– *Best alternative* (CA), that is the suggested output. It can assume values *funicular*, *bus* or *stairs*, depending on *Queue tolerance* and *Queue status*.

The *Queue* input can be inserted by the user or automatically detected by the system through wearables, exploiting traffic conditions applications installed on them. The input of this value could change over time, generating different probability distributions and, consequently, inducing variations in the reliability of the WES rules. The related HLKA, according to the model described in section 3, is defined as follows. The first KA is the taxonomy *T*.

$$E = I \cup PO \cup O = \{AT, QS\} \cup \{QT\} \cup \{BA\}$$

$$T = \{E, \emptyset\}$$

Figure 4 is a complete taxonomy of the entities in the system, represented in graphical form; for simplicity, the *E* set as defined above comprises only the instantiated classes, that is the last level of the tree.

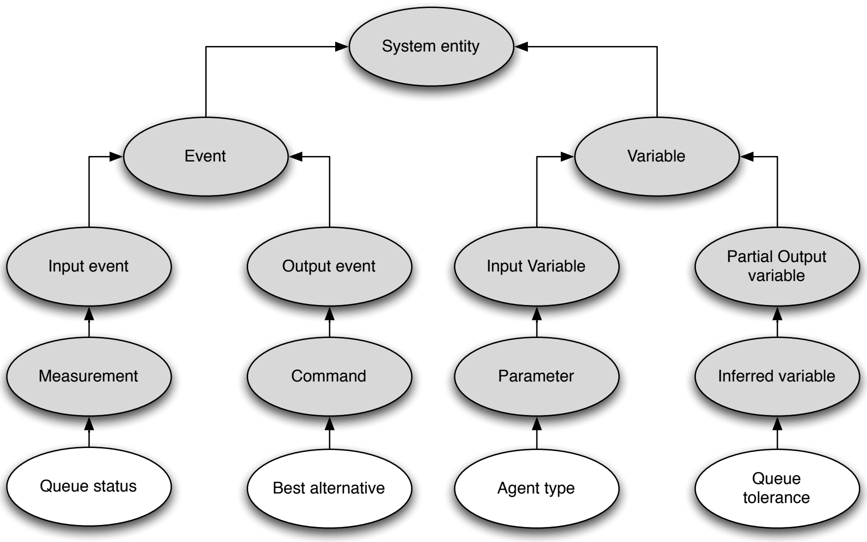


Fig. 4: A complete taxonomy relative to the application example

The second KA is the KBN, defined as follows:

$$KBN = \{E, A\},\ \text{with } A = \{\{\emptyset, P_{AT}\}, \{\emptyset, P_{QS}\}, \{\{AT\}, P(QT \mid AT)\}, \{\{QT, QS\}, P(CA \mid QT, QS)\}\}$$

and represented in figure 5

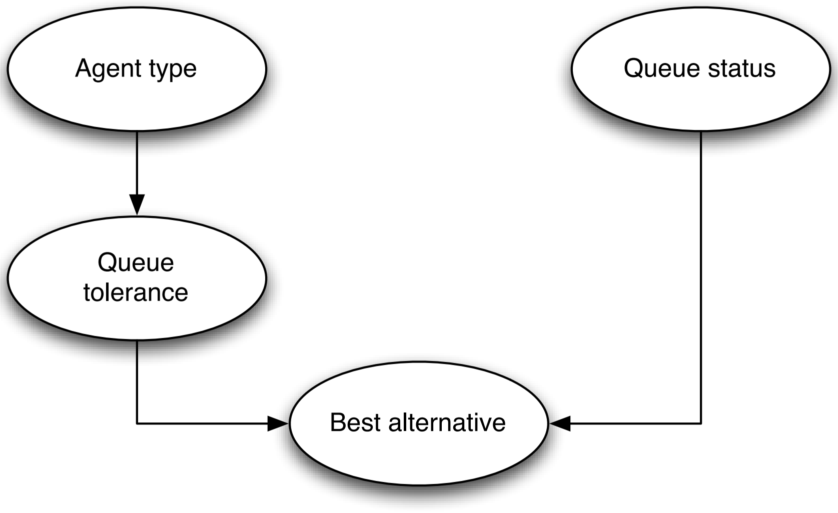


Fig. 5: The KBN relative to the application example

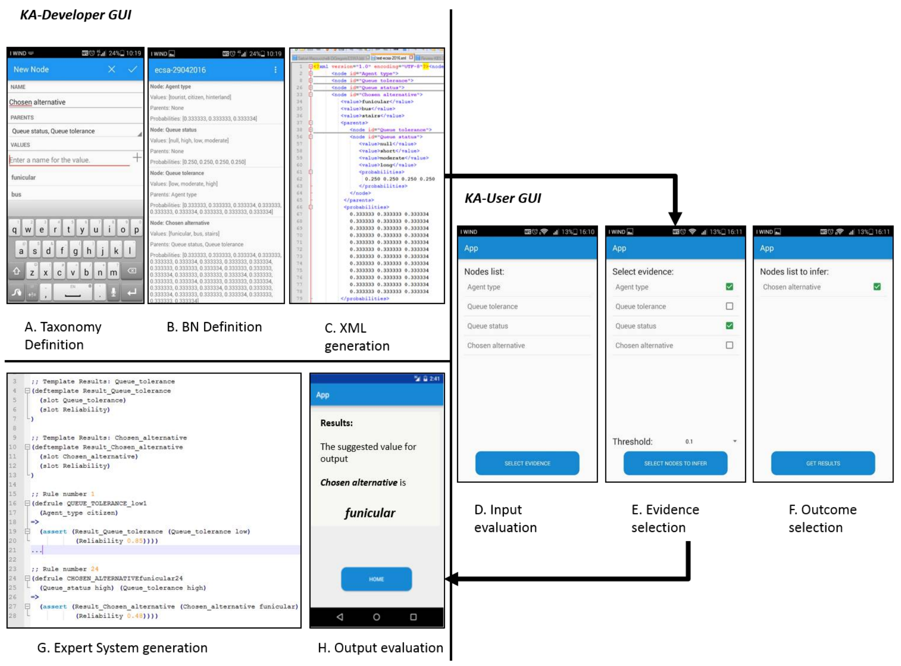
The third KA is a set of production rules, selected according to their level of reliability.

$$PR = \{E, \{\{if - do_{AT,QT}\}, \{if - do_{\{QT,QS\},CA}\}\}\}$$

Let us now consider the WES development process and the generation of the rules. In a first phase, the KA-Developer generates the taxonomy (see step A in Figure 6): each node is characterized by a name and a list of values. The KBN is then generated (step B) through the parents relationship for each node: this operation has the consequence to initialize the CPTs too. For each node, the number of probability values depends on its number of outcomes and the number of outcomes of each parent.

For instance, the CPT of *Best alternative* node will contain initially 36 uniform probabilities, given that it has three values and its parents *Queue status* and *Queue tolerance* have four and three outcomes respectively. Finally, an XML file (step C) is created to incorporate all the information about the current KBN, making it persistent and usable by KA-Users. Each line included in the $< probabilities >< /probabilities >$ tags represent a column in the node CPT: the sum of its values is equal to 1.

When the KBN is ready, a KA-User can exploit it to create an expert system, as shown in Figure 6: the nodes' list is shown for values selection (step D). This list is dynamically created from the KBN XML file and the KA-User must select the value for each input.



NAME
Chosen alternative
PARENTS
Queue status, Queue tolerance
VALUES
Enter a name for the value.
funicular
bus

Node: Agent type
Values: [tourist, citizen, hinterland]
Parents: None
Probabilities: [0.333333, 0.333333, 0.333334]
Node: Queue status
Values: [null, high, low, moderate]
Parents: None
Probabilities: [0.250, 0.250, 0.250, 0.250]
Node: Queue tolerance
Values: [low, moderate, high]
Parents: Agent type
Probabilities: [0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334]
Node: Chosen alternative
Values: [funicular, bus, stairs]
Parents: Queue status, Queue tolerance
Probabilities: [0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334, 0.333333, 0.333334, 0.333333, 0.333333, 0.333334]

```
<?xml version="1.0" encoding="UTF-8"?><node
    <node id="Agent type">
    <node id="Queue tolerance">
    <node id="Queue status">
    <node id="Chosen alternative">
      <value>funicular</value>
      <value>bus</value>
      <value>stairs</value>
      <parents>
        <node id="Queue tolerance">
        <node id="Queue status">
          <value>null</value>
          <value>short</value>
          <value>moderate</value>
          <value>long</value>
          <probabilities>
            0.250 0.250 0.250 0.250
          </probabilities>
        </node>
      </parents>
      <probabilities>
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
        0.333333 0.333333 0.333334
      </probabilities>
```

Nodes list:
Agent type
Queue tolerance
Queue status
Chosen alternative

Select evidence:
Agent type ✓
Queue tolerance ☐
Queue status ✓
Chosen alternative ☐
Threshold: 0.1

Nodes list to infer:
Chosen alternative ✓

```
;; Template Results: Queue_tolerance
(deftemplate Result_Queue_tolerance
  (slot Queue_tolerance)
  (slot Reliability)
)

;; Template Results: Chosen_alternative
(deftemplate Result_Chosen_alternative
  (slot Chosen_alternative)
  (slot Reliability)
)

;; Rule number 1
(defrule QUEUE_TOLERANCE_low1
  (Agent_type citizen)
->
  (assert (Result_Queue_tolerance (Queue_tolerance low)
            (Reliability 0.85))))
...

;; Rule number 24
(defrule CHOSEN_ALTERNATIVEfunicular24
  (Queue_status high) (Queue_tolerance high)
->
  (assert (Result_Chosen_alternative (Chosen_alternative funicular)
            (Reliability 0.48))))
```

Results:
The suggested value for output
Chosen alternative is
funicular

Fig. 6: The information flow processing in KAFKA, from KA-Developer (steps A., B. and C.) to the output (steps G. and H.), through the KA-User (steps D., E. and F.).

The second phase (step E) is the definition of evidences to browse the KBN properly. The evidences and their values will be transformed in the LHS of the rule to be created.

The user can also specify a threshold value on rule *reliability* for limiting the dimension of the rule-based system. The reliability value is a measure of how much the automatically generated rule can be considered applicable. The reliability of a rule is calculated by the framework, being modified at each iteration: it depends on the involved evidences and outcomes probabilities, on the basis of the principle that rules with higher probability values of their LHS and RHS will be more reliable. Since the CPTs are continuously updated by KAFKA, the reliability of the resulting rules will be dynamically modified.

Finally, the user selects the outcomes (step F), i.e. partial outputs and outputs of the system; they will be written in the RHS of each rule. The association between LHSs and RHSs depends on the parent relationships: following the chain of parent relationships in the KBN it will be possible to identify the computational levels of the rule-based system, with many benefits from the maintenance point of view. In our example, the *Queue tolerance* partial output must be obtained before the *Chosen alternative* output: a different behavior would be erroneous.

Figure 6 also shows a sketch of the rule-based system automatically created by the KA-User (step G) and the results of its execution (step H): the KBN has generated 24 rules and two templates for the *Chosen alternative* output and the *Queue tolerance* partial output . The rules' LHSs are combinations of parent nodes' outcomes: for example, rule number 1 calculates the Queue tolerance value according to the type of agent (i.e. the parent): if the parent value is *citizen*, then his/her tolerance to queue is low. Rule number 24 determines the chosen alternative *funicular* if the *Queue tolerance* and *Queue status* values are high. The rules RHSs are combinations of the current node outcomes together with the reliability of the rule.

The reliability value allows to prune the rules set to limit the combinatorial explosion of rules: in this way, the WES modifies its knowledge base according to the temporal evolution of its inputs and user preferences. The following code shows an excerpt of KA-Developer elaboration, where *none* outcome has been selected for the *Queue_status* node and *citizen* outcome has been selected for the *Agent_type* node:

```
Node: Queue_tolerance
Configuration exploited:
...... Node: Queue_status Outcome: none
...... Node: Agent_type Outcome: citizen
Posterior probability distribution:
------Queue_tolerance Outcome: low Probability: 0.85
------Queue_tolerance Outcome: moderate Probability: 0.13
------Queue_tolerance Outcome: high Probability: 0.02
```

Supposing that the user selects a reasonable value of 0.5 or greater as reliability threshold (step E), the rule chosen for the evaluation of the partial output *Queue tolerance* by the current WES will be:

```
(defrule Queue_tolerancelow1
  (Queue_status none)
(Agent_type citizen)
=>
  (assert
  (Result_Queue_tolerance
  (Queue_tolerance low)
(Reliability 0.85)
)
)
)
```

The partial output value will be then exploited to extract the rule for output *Best alternative* determination:

```
Node: Best_alternative
Configuration exploited:
...... Node: Queue_tolerance Outcome: low
```

```
Posterior probability distribution:
------Best_alternative Outcome: funicolar Probability: 0.7275
------Best_alternative Outcome: bus Probability: 0.2515
------Best_alternative Outcome: stairs Probability: 0.021000000000000005
```

It is evident how the *funicular* outcome is the most reliable among the three possibilities in the current instant of time, given that its probability value is *0.7275*. Thus the final rule to be included and executed in the WES will be:

```
(defrule Best_alternativefunicolar13
  (Queue_tolerance low)
=>
  (assert
  (Result_Best_alternative
  (Best_alternative funicolar)
(Reliability 0.7275)
)
)
)
```

6 Results and Discussion

In this paper we have presented wearable expert systems as an evolution of traditional expert systems (in particular rule-based systems) able to deal with scalable and time-dependent domains. The most interesting feature of WESs is the possibility to design, implement and use them on wearable devices, making them able to manage events dynamically according to the environment evolution. In our approach, WESs continuously adapt their knowledge bases to tackle such evolution, taking care of both new values for existing events and new event types.

Evaluating the performance of wearable expert systems is not simple, given that it strictly depends on the kinds of problem to be solved. Anyway, given that in our approach WES development is based on KAFKA adoption, we can evaluate the wearable expert systems development cycle from a methodological point of view. To this aim, we will use metrics existing in the literature.

The most similar framework found in the literature was developed by Ruiz-Mezcua et al. (2011), who designed and implemented a web server with the tools for knowledge-base construction and browsing, and two distinct interfaces for domain experts and users. The main difference between KAFKA and that approach is the target user addressed: while that work was devoted to support domain experts in developing their own, complete, expert systems, the KAFKA scope is to support any kind of user in developing knowledge based systems from scratch. For this reason, the main effort in KAFKA development was the characterization of the HLKA model to guide the user in the identification of knowledge kinds involved in his/her problem and (possibly) to extend them in case of need.

In order to test the usability of KAFKA, the framework has been submitted to different groups of students for the design and implementation of projects for their examinations: 4 students applying for a Bachelor Degree in Informatics (G1), 20 students from secondary schools applying for a stage at the Department of Computer Systems and Communication of the University of Milano-Bicocca (G2), 20 students attending the first year of first level degree in Mathematics (G3), and 20 students attending the IT-Knowledge course of the Master in Marketing Management at the University of Milano-Bicocca (G4).

Every group was divided into couples with a KA-User and a KA-Developer and the same problem: G1 had to configure the components of a *remote control car* to meet given performance levels, G2 and G3 had to find the best travel solution to go from a given source to a given destination, according to inputs like the distance between the places, the travel time and cost and the user needs, G4 had to analyze a subset of the balance indexes of an enterprise to define its ranking. Note that, although the objective of the experiment was the evaluation of the KAFKA efficacy in the design of a general expert system, the specific problem given to groups G2 and G3 is amenable to the implementation as a WES: for example, data about the remote control car could be derived from car sensors and travel solutions could depend on available services like weather conditions forecasting. Each couple had two weeks to complete its task, producing an executable Jess 7.0 file of rules.

Both couples in G1, seven couples in G2, eight in G3 and eight in G4 were able to complete their tasks successfully. On the other hand, three couples in G2, two in G3 and two in G4 were unable. Summarizing, 78% of couples involved in the experiment were able to complete the task of building an executable expert system: among them, only the couples from G1 had some knowledge on Jess and its usage, while the rest of them used it for the first time. They didn't need to learn the Jess syntax, but they spent most of the time in acquiring knowledge about the proposed problem domain. Of course, the obtained expert systems were heterogeneous in terms of quality and number of solutions found.

The couples were also asked to evaluate the KAFKA usability, exploiting the same indicators used by Ruiz-Mezcua et al. (2011), that are the perceived *power* of the framework in developing expert systems, the *ease of use* of the tool and the *adaptation* of the system to user's needs (i.e. *suitability* dimension); the results are shown in Figure 7.

As expected, the best overall evaluations were given by couples in groups G1 and G4, since they were composed of people with higher technical competencies than the other two. Anyway, it is important to highlight how also members of G2 and G3 recognized the capability of KAFKA in designing and implementing solutions to problems, independently by the application domain (power and suitability dimensions received very good votes). Summarizing, the KAFKA framework received an average evaluation of 8.9, 9.2 and 8.4 for ease of use, power and suitability, respectively. The total average evaluation of the framework usability was 8.8: these results are comparable with the previous

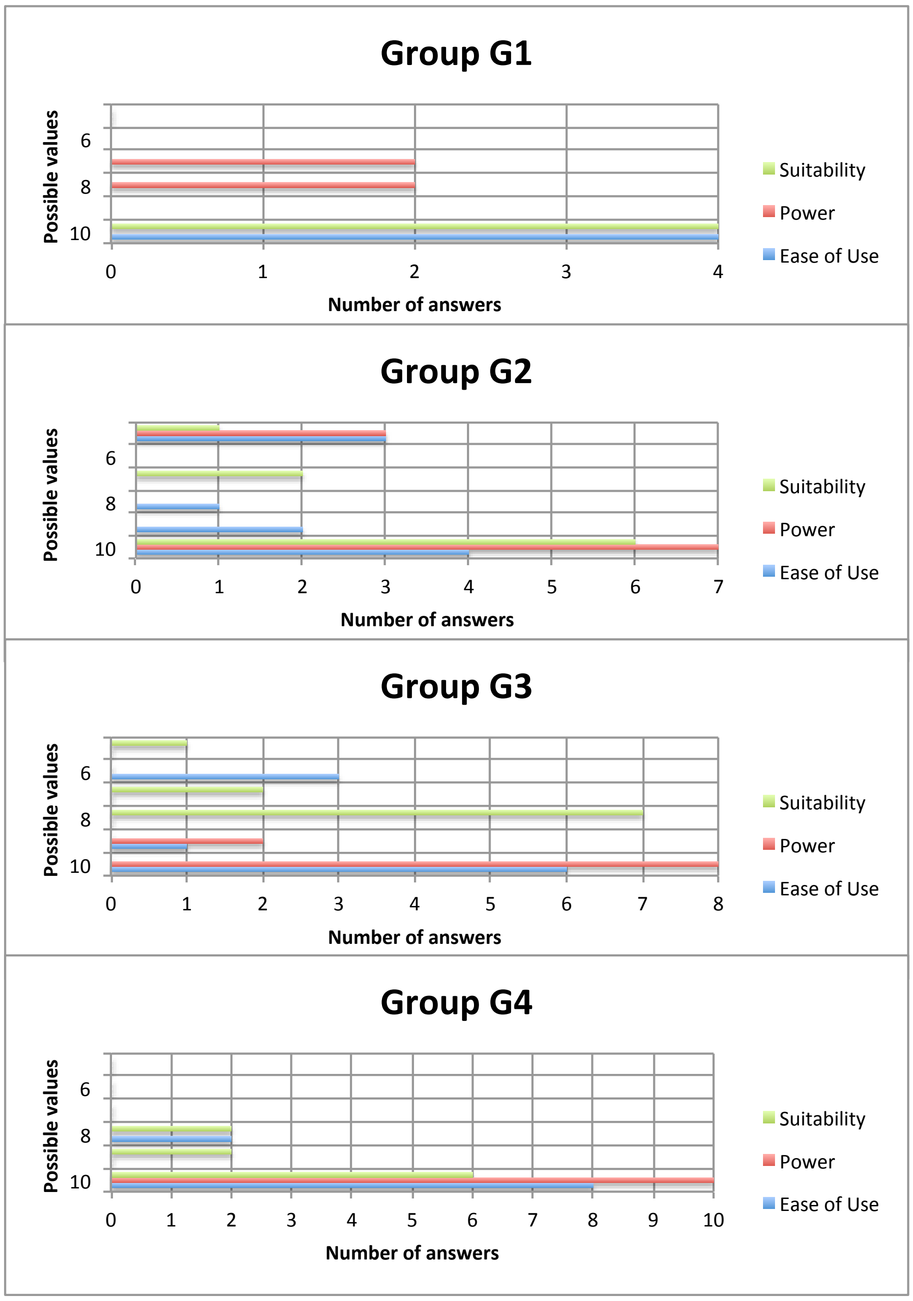


Fig. 7: Answers provided by each group to the KAFKA evaluation questionnaire.

work by Ruiz-Mezcua et al. (2011), although in that case the users were selected among domain experts, therefore had a much smaller competence gap to fill.

In order to project the evaluation of KAFKA to wearable expert systems developed through it, it is possible to observe that usability can be considered as a measure of the *scalability* feature of WESs. The expert systems designed and implemented by the students were continuously updated during the two weeks of development, according to new discoveries made by them, causing the addition of new events and variables. The implementation of the updating algorithm allowed them to maintain the knowledge base of their systems when needed, since the reliability calculus allowed to avoid the possible combinatorial explosion of rules generated by the KBN.

About the agility of the WES development cycle feature, a different kind of evaluation is necessary. To this aim, the *4-dimensional analytical tool* (4-DAT) proposed by Qumer and Henderson-Sellers (2008) has been applied to KAKFA. This method allows to evaluate and compare existing agile software development tools thanks to four qualitative and quantitative indexes: *method scope*, *agility characterization*, *agile value characterization* and *software process characterization*. Agility characterization is determined by valuing five key features, namely *flexibility* (FY), *speed* (SD), *leanness* (LS), *learning* (LG) and *responsiveness* (RS), of the phases the application life cycle is made up of as well as practices derived from them. Traditional models of experts systems development, like e.g. (Agarwal and Tanniru, 1990), are not intrinsically agile, being based on models for software systems development like the *waterfall* (Royce, 1987) or *spiral* (Boehm, 1988) approaches. With respect to other methodologies, KAFKA tries to reduce the knowledge engineering effort distributing it between users and experts, avoiding the knowledge engineer role. In this way, complex steps like knowledge acquisition, representation and coding can be made simpler.

Table 1 shows the results derived from the students' experiment above. The table is configured as prescribed by the 4-DAT model, with entries for both phases and practices characterization. Phases and practices have been derived from the ESDL model by Agarwal and Tanniru (1990): in particular, *functional, procedural and experiential knowledge* and *user interface* substitute *conceptualization, formalization, implementation and testing* practices in the *system development* phase of ESDL cycle of life, *connectivity* is the practice for *transfer to production* phase, while *KB maintenance* and *evaluation* are the practices for the *operation* phase.

The WES development cycle receives 0 characterization value on the dimensions related to *problem identification* and *evaluation*, since no tools are currently provided by KAFKA to tackle them. The evaluation of phases features is obtained from the related practices values, according to equation 26:

$$Phase_i = \bigwedge_{j \in [1 \ldots n]} Practice_i^j \tag{26}$$

where $i \in [FY, ..., RS]$ is the current feature to be valued and n is the number of practices related to the current phase.

Table 1: Agility characterization of WESs developed in KAFKA.

	Agility features					
	FY	SD	LS	LG	RS	Total
Phases						
Problem identification	0	0	0	0	0	0
System development	1	0	0	1	0	2
Transfer to production	1	1	1	1	1	5
Operation	1	1	1	1	1	5
Total	3	2	2	3	12	
Agility degree	3/4	2/4	2/4	3/4	2/4	12/20
Practices						
Functional Knowledge	1	0	0	1	1	3
Procedural Knowledge	1	1	1	1	1	5
Experiential Knowledge	1	1	0	1	0	3
User Interface	1	1	1	1	0	4
Connectivity	1	1	1	1	1	5
KB Maintenance	1	1	1	1	1	5
Evaluation	0	0	0	0	0	0
Total	6	5	4	6	4	
Agility degree	6/7	5/7	4/7	6/7	4/7	25/35

From the analysis of results, it emerges that the less agile practice is functional knowledge representation. Given that it is the initial step in the knowledge artifact definition, it is the most difficult one from the speed and leanness perspectives. In fact, users and domain experts must define inputs, partial outputs, outputs and relationships among them.

KAFKA fully supports the two roles in procedural knowledge, experiential knowledge, user interface, connectivity and KB maintenance steps. Only experiential knowledge and user interfaces received a zero value in the *responsiveness* characteristic, since the suggested set of rules and the standard GUI provided by the implemented WES are prototypes.

KB maintenance is automatically provided by the updating algorithm; moreover, mechanisms for the connectivity with wearables are automatically integrated into KAFKA-based wearable expert systems (Baretta et al.; Pinardi et al., 2016).

Finally, the total estimated agility degree is 60%, that is significantly higher than *waterfall* and *spiral* models (Qumer and Henderson-Sellers, 2008), although less than typical agile software engineering methodologies.

7 Conclusion and Future Works

This paper has discussed the design of *Wearable Expert Systems*. The proposed design methodology is KAFKA, a framework based on the Knowledge Artifact conceptual model, which is general enough to be adopted in different contexts and programming paradigms.

From the conceptual point of view, KAFKA aims at making the development of knowledge-based systems (in particular, rule-based systems) quicker and simpler through the reduction of knowledge engineer responsibilities. In this way, the knowledge engineering process is focused on the *kinds of knowledge involved* in the decision making activity rather than on *how to model it*, representing a radical change of perspective if compared with classical approaches like CommonKADS and MIKE. In this sense, KAFKA philosophy is closer to methodologies like MOKA (Stokes et al., 2001) and KNOMAD (Curran et al., 2010), proposed in the knowledge-based engineering field as product-oriented (Verhagen et al., 2012) rather than process-oriented tools for supporting users in the configuration of objects.

The WESs which can be developed employing this methodology bear some distinguishing features with respect to traditional expert systems. The WES knowledge base may change dynamically, following the long-term evolution of the monitored system and of its surrounding environment. Moreover, the presence of a centralized knowledge maintenance system, in principle common to a large number of WES instances, permits to exploit the massive amount of information coming from this large set of wearable devices: for instance, in the example of section 5, the *Queue status* may be monitored reliably by collecting data from all the application users in the city, tens or maybe hundreds of devices.

From these observations, it follows naturally that WESs are a kind of *time evolving expert systems* (Sartori and Melen, 2015), in which both the measurement of new values for input variables and the detection of new input events can be used to modify, extend and maintain knowledge-bases, to represent domains characterized by variability over time. Probabilistic event processing (Wang et al., 2013), exploiting graphical models like Dynamic Bayesian Networks (Murphy, 2002), could be useful to this scope. The introduction of KBNs in the HLKA model has been thought to build up a flexible, time-dependent and automatic mechanism to deal with this challenge.

Possible future developments are certainly ample and varied. In terms of implementation, the server is, without doubt, the component that needs more attention: some possible modifications range from the addition of new JESS features to the improvement of memory management and a more efficient management of concurrency. From the client side point of view, it should be increased the autonomy level of KA-User and KA-Developer in supporting the development of several kinds of applications as discussed in section 6.

Moreover, the framework must be quantitatively evaluated in concrete domains of application along the lines discussed in Section 6.

Bibliography

R. Agarwal and M. Tanniru. Systems development life-cycle for expert systems. *Knowledge-Based Systems*, 3(3):170–180, 1990.

J. Angele, D. Fensel, D. Landes, and R. Studer. Developing knowledge-based systems with mike. *Automated Software Engineering*, 5(4):389–418, 1998.

N. Aussenac-Gilles and F. Gandon. From the knowledge acquisition bottleneck to the knowledge acquisition overflow: A brief French history of knowledge acquisition. *International Journal of Human Computer Studies*, 71(2):157–165, 2013.

S. Bandini and F. Sartori. From handicraft prototypes to limited serial productions: Exploiting knowledge artifacts to support the industrial design of high quality products. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 24:17–34, 2 2010. ISSN 1469-1760. doi: 10.1017/S089006040999014X.

D. Baretta, F. Sartori, A. Greco, R. Melen, F. Stella, L. Bollini, M. Daddario, and P. Steca. Wearable devices and ai techniques integration to promote physical activity. doi: http://dx.doi.org/10.1145/2957265.2965011.

B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, May 1988. ISSN 0018-9162.

C. Brewster and K. O'Hara. Knowledge representation with ontologies: the present and future. *Intelligent Systems, IEEE*, 19(1):72–81, 2004.

T. Butler, J. Feller, A. Pope, B. Emerson, and C. Murphy. Designing a core it artefact for knowledge management systems using participatory action research in a government and a non-government organisation. *The Journal of Strategic Information Systems*, 17(4):249–267, 2008.

F. Cabitza and A. Locoro. "made with knowledge" - disentangling the it knowledge artifact by a qualitative literature review. In *Proceedings of the International Conference on Knowledge Management and Information Sharing (IC3K 2014)*, pages 64–75, 2014. ISBN 978-989-758-050-5. doi: 10.5220/0005086100640075.

A. Cheatle and S. J. Jackson. Digital entanglements: Craft, computation and collaboration in fine art furniture production. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 958–968. ACM, 2015.

W. J. Clancey. The epistemology of a rule-based expert system a framework for explanation. *Artificial Intelligence*, 20(3):215–251, 1983.

K. Conboy. Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, 20(3):329–354, 2009.

G. Cugola and A. Margara. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Computing Surveys (CSUR)*, 44(3):1–69, 2012. doi: 10.1145/2187671.2187677.

R. Curran, W. J. Verhagen, M. J. Van Tooren, and T. H. van der Laan. A multidisciplinary implementation methodology for knowledge based engineering: Knomad. *Expert Systems with Applications*, 37(11):7336–7350, 2010.

F. de Cindio, G. D. Michelis, L. Pomello, and C. Simone. Superposed automata nets. In *Application and Theory of Petri Nets, Selected Papers from the First and the Second European Workshop on Application and Theory of Petri Nets, Strasbourg 23.-26. September 1980, Bad Honnef 28.-30. September 1981*, pages 269–279, 1981.

G. Eysenbach et al. From intermediation to disintermediation and apomediation: new models for consumers to access and assess the credibility of health information in the age of web2. 0. *Studies in health technology and informatics*, 129(1):162, 2007.

D. Fogli and G. Guida. Knowledge-centered design of decision support systems for emergency management. *Decision Support Systems*, 55(1):336–347, 2013.

B. R. Gaines. Knowledge acquisition: Past, present and future. *International Journal of Human-Computer Studies*, 71:135–156, 2013. ISSN 10715819. doi: 10.1016/j.ijhcs.2012.10.010.

C. W. Holsapple and K. D. Joshi. Organizational knowledge resources. *Decision support systems*, 31(1):39–54, 2001.

S. Hoppenbrouwers and P. Lucas. Attacking the Knowledge Acquisition Bottleneck through Games-For-Modelling. In *Proceedings of the Symposium : AI and Games Symposium, A symposium at the AISB 2009 Convention (6-9 April 2009)Heriot-Watt University, Edinburgh, Scotland, p.81-86. ISBN 1902956818.*, pages 81–86. Falmer, Brighton : The Society for the Study of Artificial Intelligence and the Simulation of Behaviour, 2009.

P. Jackson. *Introduction to expert systems*. Addison-Wesley, Reading, MA, 1998. ISBN 978-0201876864.

Y. Kitamura, M. Kashiwase, M. Fuse, and R. Mizoguchi. Deployment of an ontological framework of functional design knowledge. *Advanced Engineering Informatics*, 18(2):115–127, 2004.

K. B. Korb and A. E. Nicholson. *Bayesian artificial intelligence*. CRC press, 2010.

G. Lee and W. Xia. Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility. *Mis Quarterly*, 34(1): 87–114, 2010.

D. Luckham. *The power of events*, volume 204. Addison-Wesley Reading, 2002.

A. Margara, G. Cugola, and G. Tamburrelli. Learning from the past: automated rule generation for complex event processing. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems - DEBS '14*, pages 47–58, 2014. ISBN 9781450327374. doi: 10.1145/2611286.2611289.

N. Mehdiyev, J. Krumeich, D. Enke, D. Werth, and P. Loos. Determination of Rule Patterns in Complex Event Processing Using Machine Learning Techniques. In *Procedia Computer Science*, volume 61, pages 395–401, 2015. doi: 10.1016/j.procs.2015.09.168.

K. P. Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

K. Niedderer and L. Reilly. Research practice in art and design: Experiential knowledge and organised inquiry. *Journal of Research Practice*, 6(2), 2010.

S. Pinardi, F. Sartori, and R. Melen. Integrating knowledge artifacts and inertial measurement unit sensors for decision support. In A. L. N. Fred, J. L. G. Dietz, D. Aveiro, K. Liu, J. Bernardino, and J. Filipe, editors, *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 3: KMIS, Porto - Portugal, November 9 - 11, 2016.*, pages 307–313. SciTePress, 2016. ISBN 978-989-758-203-5. doi: 10.5220/0006091203070313. URL `http://dx.doi.org/10.5220/0006091203070313`.

A. Qumer and B. Henderson-Sellers. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and software technology*, 50(4):280–295, 2008.

D. B. Robins. Complex Event Processing. *2010 Second International Workshop on Education Technology and Computer Science*, 2010. ISSN 1611-2776. doi: 10.1109/ETCS.2010.214.

J. A. Rosen and W. L. Smith. Influence net modeling with causal strengths: an evolutionary approach. In *Proceedings of the Command and Control Research and Technology Symposium*, pages 25–28, 1996.

W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press. ISBN 0-89791-216-0.

B. Ruiz-Mezcua, A. Garcia-Crespo, J. L. Lopez-Cuadrado, and I. Gonzalez-Carrasco. An expert system development tool for non ai experts. *Expert Systems with Applications*, 38(1):597–609, 2011.

G. Salazar-Torres, E. Colombo, F. C. Da Silva, C. Noriega, and S. Bandini. Design issues for knowledge artifacts. *Knowledge-based systems*, 21(8):856–867, 2008.

F. Sartori and R. Melen. Employing knowledge artifacts to develop time-depending expert systems. In A. L. N. Fred, J. L. G. Dietz, D. Aveiro, K. Liu, and J. Filipe, editors, *Knowledge Discovery, Knowledge Engineering and Knowledge Management - 7th International Joint Conference, IC3K 2015, Lisbon, Portugal, November 12-14, 2015, Revised Selected Papers*, volume 631 of *Communications in Computer and Information Science*, pages 281–301. Springer, 2015. ISBN 978-3-319-52757-4. doi: 10.1007/978-3-319-52758-1_16. URL `https://doi.org/10.1007/978-3-319-52758-1_16`.

G. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans, and W. Van de Velde. Commonkads: A comprehensive methodology for kbs development. *IEEE expert*, 9(6):28–37, 1994.

M. Stokes, M. Consortium, et al. *Managing engineering knowledge: MOKA: methodology for knowledge based engineering applications*. Professional Engineering Publ., 2001.

J. Surif, N. H. Ibrahim, and M. Mokhtar. Conceptual and procedural knowledge in problem solving. *Procedia - Social and Behavioral Sciences*, 56:416–425, 2012. URL `http://www.sciencedirect.com/science/article/pii/S187704281204133X`.

K. Teymourian and A. Paschke. Semantic rule-based complex event processing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artifi-*

cial Intelligence and Lecture Notes in Bioinformatics), volume 5858 LNCS, pages 82–92, 2009. ISBN 3642049842. doi: 10.1007/978-3-642-04985-9_10.

W. Van Der Aalst, A. Adriansyah, and B. Van Dongen. *Causal nets: A modeling language tailored towards process discovery*, volume 6901 LNCS, pages 28–42. 2011.

W. J. Verhagen, P. Bermell-Garcia, R. E. van Dijk, and R. Curran. A critical review of knowledge-based engineering: An identification of research challenges. *Advanced Engineering Informatics*, 26(1):5–15, 2012.

C. Wagner. Collaborative knowledge management: Breaking the knowledge acquisition bottleneck. In *Innovations Through Information Technology: 2004 Information Resources Management Association International Conference, New Orleans, Louisiana, USA, May 23-26, 2004*, volume 1, page 341. IGI Global, 2004.

C. Wagner. Breaking the knowledge acquisition bottleneck through conversational knowledge management. *Information Resources Management Journal (IRMJ)*, 19(1):70–83, 2006.

Y. H. Wang, K. Cao, and X. M. Zhang. Complex event processing over distributed probabilistic event streams. *Computers and Mathematics with Applications*, 66(10):1808–1821, 2013. ISSN 08981221. doi: 10.1016/j.camwa.2013.06.032.