The Knowledge Engineering Review, Vol. 26:1, 19–24.
doi:10.1017/S0269888910000366

From expert systems to context-based intelligent assistant systems: a testimony

PATRICK BRÉZILLON

LIP6, University Pierre et Marie Curie - UPMC, 4 Place Jussieu, 75005, Paris, France
e-mail: Patrick.Brezillon@lip6.fr

Abstract

This paper presents a personal interpretation of the evolution of artificial intelligence (AI) systems during these last 25 years. This evolution is presented along five generations of AI systems, namely expert systems, joint cognitive systems, intelligent systems, intelligent assistant systems, and the coming generation of context-based intelligent assistant systems. Our testimony relies on different real-world applications in different domains, especially for the French national power company, the subway companies in Paris and in Rio de Janeiro, in medicine, a platform for e-maintenance, road safety, and open sources. Our main claim is to underline that the next generation of AI systems (context-based intelligent assistant systems) requires a radically different consideration on context and its relations with the users, the task at hand, the situation, and the environment in which the task is accomplish by the user; the observation of users through their behaviors and not a profile library; a robust conceptual framework for modeling and managing context; and a computational tool for representing in a uniform way pieces of knowledge, of reasoning, and of contexts.

1 Introduction

This paper is a testimony of 20 years of research in artificial intelligence (AI). My initial interest for AI presented several facets. First, coming from the domain of mathematical modeling and having faced the limits of this approach in real-world applications, I was curious to study how AI tools (especially expert systems) could be a modeling tool for some problems in which human reasoning played a key role. Second, AI belongs to cognitive sciences and has to deal with different disciplines such as cognitive psychology or neurosciences far from the engineering viewpoint. Third, the initial paradigm of separation of the representation of the knowledge from its use was a powerful alternative to the usual approach in software engineering, at least in the design stage. Fourth, AI has been (and is always) a battlefield where 'hard' science (i.e. theory-based science) was opposed to 'soft' science (i.e. cognition-based science). For example, the mathematical logic bloomed in a large variety of logics attempting each to deal with a specific aspect of cognition. Fifth, AI is a domain where the success of an AI system is measured by its disappearance in classical software.

The first concept that plays an important role in AI is knowledge, and more specifically its nature. Education provides students with theoretical knowledge, when enterprises ask for their employees to have operational knowledge. For example, the Ohm's law is learned through instantiations where R has an exact value (say, 3.0 Ω) when real resistances have a value with a limited accuracy that may play a crucial role in an electronic circuit. The transformation of theoretical knowledge into operational knowledge is a process of contextualization during which we adapt our knowledge to the task, the situation, and the local environment. Expertise is contextualized knowledge.

A second point is the difference between data, information, and knowledge. Information is data with meaning that results from a process of interpretation. For example, a temperature of 24°C (the data) will be interpreted differently by a person living in the Polar circle (24°C is hot) and a person living near the equator (24°C is cold). This is due to the fact that their mental models to interpret the data are different. Relevant information enriches the person's knowledge (the person can establishes a link between the information and his mental model).

A third point concerns procedures and practices. Organizations develop procedures that are collections of secure action sequences to address a given focus in any case. As a consequence, actors prefer to plan their action in real time rather than to rely on procedures for two main reasons. First, the procedure is never perfectly adapted to any specific situation and can lead to improper actions or sub-optimal incident resolution strategies. Second, a practice captures advantages and disadvantages of the situation at hand when a procedure tries to capture only the abstracted solution. Given a practical unit of reasoning:

$$\text{if } P1, P2, \ldots \text{ then } C1$$

In a logical and theoretical reasoning (a procedure), the action leads to the conclusion C1 that is added to the base alone. A practical reasoning is an inductive probabilistic reasoning, the conclusion cannot be detached from the premises, that is, take meaning out of premises: C1 is added to the base linked to P1, P2, etc. This means that knowledge is defined in a context of use that is lost in procedures because procedures are supposed to be applicable to a large class of problems when a practice is a contextualized version of the procedure in a specific context. As a side effect, this implicitly shows that there is not always a unique and optimal solution, but a set of best solutions in given contexts.

It is a problem clearly identified in different disciplines: prescribed task versus effective task, task versus activity, logic of functioning versus logic of use. The difference between procedure and practice is a question of knowledge organization: parallel structures for procedures and sequential structures (and simplification of the domain knowledge from the user's viewpoint) for practices.

The role played by context between procedure and practice in real-world applications is, in my view, at the origin of successive mutations in AI that occurred during these past 20 years.

2 The time of expert systems

The paradigm of separation of the representation of knowledge from its use, is born from the attack of ill-defined problems (expertise, heuristics, combinatorial, etc.) with no algorithmic solutions. This corresponds to a shift of the interest from procedural knowledge (algorithm, software, etc.) to declarative knowledge (fact base, knowledge base, rule engine), although expert systems face both types of knowledge. Knowledge was represented in flat knowledge bases, easy to update without a need to recompile the software because an inference engine made the use of the knowledge removed from the representation. The rule base frame is employed to represent rules constraining potential courses of action, which are treated as non-negotiable ('facts of nature', prescriptions from authorities). However, this was a source of problem because part of the knowledge comes often from technical domains where knowledge organization and structure were not represented explicitly. This leads to the introduction of some implicit knowledge on the use of the knowledge.

A first option concerned the introduction of screening clauses. Figure 1 presents an example of a rule in MYCIN. Buchanan and Shortliffe (1984) proposed to add screening clauses to the condition part of rules, which were then activated only in some kind of context. For example, the 4th clause in the rule above (taken from MYCIN) constrains the activation of the rule, but is not a clause for identifying the infection. Clause 4 acts to restrict the rule application to the context of adults (the implicit reason in the rule is that tetracycline gives a violet color to teeth that is a problem for children).

Another option was the introduction of rule packets as a structuring factor of rule bases. The reason was, say in equipment diagnosis as in Figure 2, that one does not need to consider the knowledge on a protective relay when one diagnoses a circuit breaker.

https://doi.org/10.1017/S0269888910000366

IF
1. The infection which requires therapy is meningitis,
2. Only circumstantial evidence is available for this case,
3. The type of meningitis is bacterial,
4. *The age of the patient is greater than 17 years old,* and
5. The patient is an alcoholic,

THEN
There is evidence that the organisms, which might be causing the infection, are diplococcus-pneumoniae (.3) or e.coli (.2)

Figure 1 Example of screening clause

```
! check_cb
"Test of the circuit breaker in the control system $name"
> if failure freeze check_pw, check_teac
IF
        equipment_piece ($cos) := ($cb) ,
        nature ($cb)  := circuit_breaker .
THEN
        call the rule packet 'Circuit-Breaker_Diagnosis($cos, $cb)'
```

Figure 2 A rule calling a rule packet

Here, it is the whole IF part that acts as the context of a circuit breaker. The explicit use of context in such a rule offers also two advantages: (1) the rule packet, although valid for all the circuit-breakers (about 20 in this application for the French national power company, Electricité de France), is triggered for the unique circuit breaker '$cb' in the cut-off system '$cos'; and (2) the triggering of this rule allows to inhibit the test of other equipment pieces (Freeze 'check_pw' and 'check_teac') of the cut-off system.

Three lessons learned at the end of this phase are: (1) the need of a user-centered approach, (2) need to deal explicitly with context, and (3) need of new methodologies different from classical engineering.

3 The time of joint cognitive systems

The expression 'joint cognitive system' was coined to stress the fact that neither the system nor the user is able to solve the problem at hand alone (Woods, 1985). The emphasis was on the word 'cognitive'. In a broad sense, this implies that the system and the user are able to understand each other (or more realistically make compatible their interpretations (Karsenty and Brézillon, 1995)) during problem solving. This entails strong requirements about the design of the system as: the sharing of the cognitive representations, an agreement about the contextual knowledge of the problem solving, the ability to follow each others reasoning, the capability of exchanging explanations, the capability to acquire incrementally new knowledge and learn new practices from the other, and a sharing of the interaction control. A key point here is the user and system complementarity to exploit their agent asymmetry. The complementarity implies that the system may contain either some knowledge (model or data) or functions (e.g. database exploitation) that really complements the user's skills. However, joint cognitive systems focused on the task at hand, when some tasks such as knowledge acquisition, practice learning and explanation are parts of the task at hand and managed by the system as well as the user, because you cannot anticipate or 'design away' all the problems that might arise during user–system interaction. Thus, the system must know how to acquire knowledge incrementally when needed. This was not possible as long as context was not made explicit.

The interest here is not on the probability of a branch but about the possibility to determine, as soon as possible, on which path the operator is, to determine what is the next action to undertake. In other words, each state of nature (a sequence of events is a state of nature) describes a state of the world, or using our words, a *context for action*. As many contextual elements may intervene in several scenarios (e.g. traffic activity, position of the next train), operators prefer to take them into account as soon as possible to get a general picture of the best path to choose. At this step, contextual knowledge is proceduralized and in the meantime, operators postpone actions in sequences like macro-actions. The proceduralized context corresponds to the needed knowledge for addressing the focus of the operators. The main objective is to eliminate event nodes and to use contextual information for the choice of the actions.

Previously, users were considered as data gatherers with a total lack of understanding of the system behavior (i.e. not on the expert knowledge in the machine, but the way in which the system used this knowledge). The system relies on a logic of functioning often far from the logic of use of users. Thus, for completing or modifying knowledge in the system, users prefer to intervene superficially by adding new rules to control the firing of existing rules. As a consequence, the system rapidly becomes intractable.

Humans usually function in an anticipative mode; operators explicitly check hypotheses to eliminate discovery or surprise. An anticipatory system would use knowledge (as a model itself and the relevant part of its environment) about future states to decide what action to take at the moment. An intelligent system should predict what will probably happen and preadapt itself for the occurrence of a crucial or time-critical event. An anticipatory capability supposes that the system has a simulation component.

4 The time of intelligent systems

In Frederik Pohl's (1969) science fiction novel, *The Age of Pussyfoot,* each person needs a mobile device—a 'satisfactor'—to buy products, communicate with others, watch television, control the children, and even receive medications. In this future world, computing-centered humans cannot function in society without their satisfactors. This type of computing is now everywhere: mobile phones, intelligent houses, and soon communicating objects. However, such (new) systems propose services, integrate few contextual cues (GPS, weather, etc.) and a user model often coming from statistical consideration. However, it is not the right way because the line of reasoning of the user does not deal with the line of reasoning of the system.

An intelligent system must present different properties like:

- Provide users with a first approximation of environmental trends and events,
- Point out useful information implicit in large volumes of data to alert users to sudden changes,
- Develop multiple scenarios and perspectives on a given line of action,
- Attract user attention to existing and emerging strategic issues,
- Support users in sharing and communicating their views and perspectives,
- Guide user attention to specific data and its interpretation in relation to particular issues.

With such qualities, an intelligent system must be more considered as an intelligent assistant system, like a secretary with her boss, solving tactical and operational problems and preparing elements to her boss for political and strategic problems. For example, context-aware applications can be assimilated to intelligent systems. Note that the role of the system with respect to the user is more precise than in joint cognitive systems. There is an evolution from the problem-solver view to the information-processing view. In the e-business arena, one speaks of the push model.

5 The time of intelligent assistant systems

An intelligent assistant system (IAS) has two sides: the process side and the human side. An IAS has to understand: (a) the real-world process, (b) the tasks at hand, and (c) the operator's

behavior. The IAS may learn (i.e. acquire knowledge) by observing the behavior of the operators and the process. This is an incremental knowledge acquisition process where knowledge is acquired when needed and, above all, in its context of use. Only a kernel of knowledge has to be elicited directly, mainly to select the right formal representation for the knowledge.

Depending on the operator's experience, this cooperation can take one of two forms: a waking state if there is no discrepancy between the operator's actions and those of the system, and a cooperative state if a difference is noted relative to either the operator's action or its possible consequences in the future. In the wake state, the IAS must follow the operators' actions on the process, observe the corresponding effect on the process' behavior and compare it to a simulated behavior. The IAS does not intervene, if there is no discrepancy between the observed and simulated behaviors of the process. When a problem is detected, the IAS first waits to allow time for the operator to react and, only after a while, alert the operator about the problem. The main reason is to not disturb the operator in a critical situation. If the operator does not see the problem, then, the system enters a cooperative state. However, more frequently, it is the operator that triggers the cooperative state.

As the operator's assistant, the system must automatically solve minor problems and prepare elements for complex problems that the operator will have to solve. It must also benefit from operator interaction to learn new practices and acquire new knowledge when it faces failure. For example, when the user overrides a decision for various reasons, the assistant will not understand this action if they have incompatible internal representations. If the same decision must be made in altered circumstances (i.e. in other contexts), the assistant must know when it is out of its domain and again ask the user to acquire the current context of use for the decision. In the e-business arena, one speaks of the pull model.

Generally, there are different methods for executing a task, and the choice of a method relies heavily on contextual cues and operators' experience. Clearly, an IAS must be designed and developed in a formalism providing a uniform representation of knowledge, reasoning, and context elements. Moreover, the formalism must allow incremental knowledge acquisition and practice learning. Thus, the human side begins to be addressed, but not yet the context side.

6 The time of context-based intelligent assistant systems

As discussed above, the importance of context was clear since a long time. However, it is at the beginning of the 90s that a community appeared in AI with McCarthy (1993) as a leader. He formalized context as first class objects and introduced the basic relation $ist(c, p)$ (the proposition p is true in the context c), the formula being always asserted in a context. As a consequence, McCarthy showed that:

- context is always relative to another context,
- contexts have an infinite dimension,
- contexts cannot be described completely,
- when several contexts occur in a discussion, there is a common context above all of them in which terms and predicates can be lifted.

Now, a number of disciplines have a community of context, about context-aware applications, ubiquitous computing, pervasive computing, context-driven testing, etc. For our purpose in this paper, we retain that context is the key factor of intelligent assistant systems. Making context explicit allows us to use knowledge in its context of use, to capture variants in the reasoning (e.g. recording practices effectively developed by operators), to generate relevant explanations.

After 15 years, there is a conceptual framework for context modeling and management and its implementation in a piece of software called Contextual Graphs. The working definition of context is that 'context constrains a focus without intervening in it explicitly'. As a consequence, (1) context is relative to the focus, (2) as the focus evolves, its context evolves too, and (3) context is highly domain-dependent. Linking context to a focus, one can deduce that the focus divides

https://doi.org/10.1017/S0269888910000366

context into external knowledge and contextual knowledge. The latter constitutes a kind of tank where contextual elements are related to the focus in a flat way, whereas the former has nothing to do with the focus at the moment. The focus evolves because a new event occurs (e.g. an unpredicted event) or as a result of a decision made at the previous stage of the focus. For addressing the focus, the actor selects a subset of contextual knowledge that is organized, assembled, and structured in a proceduralized context that is used at the current focus. A following step concerns how to represent knowledge in context with the notion of 'focus dressing', and a three-layer model of context (a meta-model, a domain model, and instantiation).

A practice representing the development of an actor's reasoning as a kind of contextualization of a procedure, there is a formalism that allows us to represent in a uniform way, the elements of knowledge and the context in which they are to be used in a reasoning. The formalism called Contextual Graphs has been used in several real-world applications in different domains. Contextual graphs are the building blocks of 'experience bases' on which an IAS can reason and accompany a user in the realization of his task.

7 Conclusion

Evolution of AI systems comes from a move of the focus on the task alone (Expert System time), through tasks and user (Joint Cognitive System time), after to the user accomplish a task in a situation (Intelligent System time) to a holistic view on the user, the task, the situation, and the immediate environment with an emphasis on context. Thus, this view of AI systems is one that travel smoothly from an engineering viewpoint to a cognitive science viewpoint in which context plays a central role in the modeling and the representation of the knowledge and the reasoning. A first consequence is that it is not possible to reason in an abstract way: context is deeply embedded in the actor, the task at hand, the situation and the environment, and context is intimately related to a focus of attention. A second consequence is that most of the communities interested in context management now prefer to study context by a bottom-up approach instead of the previous top–down approach. However, it is always important to work in a coherent conceptual framework, with an operational definition of context, a modeling of context linked to other elements (focus, task, user, etc.), and to use this solid framework for developing relevant tools for analysis knowledge and reasoning in context. We do this with contextual graphs for representing users' behaviors (i.e. practices instead of procedures), but there are other aspects to study, like a description of a focus or a situation in terms of contextual elements. Finally, the current approaches try to catch what humans are doing, but one will soon have to face some challenge like representing time in context and develop AI systems that are able to predict new behaviors.

References

Buchanan, B. G. & Shortliffe, E. H. 1984. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley.

Karsenty, L. & Brézillon, P. 1995. Cooperative problem solving and explanation. *International Journal of Expert Systems With Applications* **8**(4), 445–462.

McCarthy, J. 1993. Notes on formalizing context. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence* **1**, 555–560.

Pohl, F. 1969. *The Age of Pussyfoot*. Ballantine.

Woods, D. D. 1985. Cognitive technologies: the design of joint human-machine cognitive systems. *AI Magazine* **6**(4), 86–92.