A Modern Epistemological Reading of Agent Orientation

Yves Wautelet, Université catholique de Louvain, Belgium
Christophe Schinckus, Facultés Universitaires St-Louis, Belgium
Manuel Kolp, Université catholique de Louvain, Belgium

ABSTRACT

This article presents a modern epistemological validation of the process of agent oriented software development. Agent orientation has been widely presented in recent years as a novel modeling, design and programming paradigm for building systems using features such as openness, dynamics, sociality and intentionality. These will be put into perspective through a Lakatosian epistemological approach. The contribution of this article is to get researchers acquainted with the epistemological basis of the agent research domain and the context of the emergence of object and agent-orientation. This article advocates the use of the Lakatosian research programme concept as an epistemological basis for object and agent orientation. This is done on the basis of how these frameworks operationalize relevant theoretical concepts of the Kuhnian and Lakatosian theories.

Keywords: agent-oriented software engineering; epistemology; Kuhn; Lakatos; paradigm; research programme

INTRODUCTION

Information systems are deeply linked to human activities. Unfortunately, development methodologies have been traditionally inspired by programming concepts and not by organizational and human ones. This leads to ontological and semantic gaps between the systems and their environments. The adoption of agent orientation and Multi-Agent Systems (MAS) helps to reduce these gaps by offering modeling tools based on organizational concepts (actors, agents, goals, objectives, responsibilities, social dependencies, etc.) as fundamentals to conceive systems through all the development processes. Moreover, software development is becoming increasingly complex. Stakeholders’ expectations are growing higher while the development agendas have to be as short as possible. Project managers, business analysts and software developers need adequate processes and models to specify the organizational context, capture requirements and build efficient and flexible systems.

We propose, in this article, a modern epistemological validation of the emergence of Object-Orientation (OO) and Agent-Orientation
(AO). The latter will be put into perspective through the Lakatosian approach. Related work and contributions to the epistemological position of OO and AO will first be explicated. The emerging context of the conceptual frameworks of OO and AO, the software crisis, is then briefly described. The validation of our epistemological reading will be done on the basis of OO and AO operationalization of some critical theoretical concepts derived from the Kuhnian and Lakatosian theories. We finally discuss the adoption of the Lakatosian research programme concept to characterize both OO and AO. Implications of this epistemological position on everyday work have been distinguished both for software engineering researchers and practitioners. For researchers, it mostly has an implication on how agent ontologies are built and for practitioners it has an implication on how software problems are envisaged.

This article is organized as follows. Section two (State of the Art) presents the contributions as well as the research context. We point out the emergence of OO and AO as a consequence of the software crisis. That is why the software crisis is briefly reviewed. Section three (Epistemological Approach) focuses on our epistemological approach: AO is successively considered as a paradigm and a research programme. On the basis of how some relevant concepts of the Kuhnian and Lakatosian frameworks are operationalized by OO and AO, we provide a Lakatosian reading of those modeling concepts. Conclusions are summarized in Section four.

STATE OF THE ART
This section presents the contributions of an epistemological reading for the computer science researcher as well as the software crisis and AO; we consider this framework and many others as consequences of the crisis.

Related Work and Contributions
Basili (1992) defines Software Engineering (SE) as, “The disciplined development and evolution of software systems based upon a set of principles, technologies and processes.” These theoretical frameworks are expected to solve practical problems by proposing software solutions. SE is a practice-oriented field (where empiricism often plays an important role) and is constantly evolving; however, one must dispose of a framework to build common (and preferably best) practices improvement. Kaisler (2005) points out that, “We develop more experience, we not only continue to learn new practices, but we refine and hone the practices that we have already learned.” SE is the genuine discipline that emerged from this interconnection between practices and software solutions. Today’s software development has become a very complex task and no one has the required skills or time to resolve a sophisticated problem on his or her own. Software development phases need the input from lots of people having to use concepts and ideas for which they share a common understanding. This can be referred as SE’s key role: providing some common theoretical entities to allow specialists to develop software solutions.

Few papers in specialized literature point to an in depth questioning of SE knowledge evolution. As Kaisler (2005) emphasizes, the literature is mainly technical or practical and focused on the software design processes. Research methodologies, however, need to be conscientiously built to favour the development and improvement of software solutions. To this end, an epistemological analysis is of primary importance as pointed out by Basili (1992):

The goal is to develop the conceptual scientific foundations of SE upon which future researchers can build.

In this article, we mostly focus on a specific aspect of SE: OO and AO, which are modeling and programming ontologies rather than development processes. A few papers have discussed the evolution of knowledge in SE but encompass a broader range of aspects of this discipline than the current article.

In the book titled Software Paradigms (Kaisler, 2005) published in 2005, Kaisler uses the notion of a paradigm to characterize the
way of solving problems in software development. Though Kaisler explicitly quotes Kuhn’s work to define the concept of paradigm, Kaisler explains that included in this definition are the, “Concepts of law, theory, application and instrumentation: in, effect, both the theoretical and the practical.” The practical dimension being a key issue in computer science, it must be integrated into the paradigm when this concept is used in engineering software. Kaisler’s work is based on an empirical process: “We are going to apply the notion of paradigm to the investigation of programming languages and software architectures to determine how well we can solve different types of problems.” On the basis of this study, Kaisler proposes a problem typology which would determine the programming approach: “For a given problem class, we’d like to be able to create software that solves the problem or parts of it efficiently. There are two aspects to creating such software: the software’s architecture and the choice of programming language.” Kaisler uses the term paradigm in a very large sense since applying the concept to the whole “top-down analysis.” The problem typology defines a software typology, which finally determines a specific implementation.

Even if we do not assign the same meaning as Kaisler to the “paradigm” concept, our analysis can be related to his work. By proposing an epistemological reading of the evolution toward AO, we only focus on the last step of Kaisler’s view. Indeed, the emergence of AO can be seen as a broadening of the unit of implementation and consequently to the amount of problems that can be solved. This is in line with the vision developed in Jennings and Wooldridge (2001). AO is the best suited to solve complex problems because it refers to a higher abstraction level and it provides some advantages for solution programming. Jennings and Wooldridge (2001) use the term “paradigm” to characterize the evolution from OO to AO. They paradoxically emphasize the continuity between these two paradigms. Indeed, as far as the Kuhnian discussion is concerned, if two theoretical frameworks can be compared they cannot be quoted as paradigms; this will be explained in the following sections.

Finally, Göktürk and Akkok (2004) points out that, “One of the most recent (and widely accepted) examples to a ‘rescuer new paradigm’ in software engineering is the object-orientation paradigm…” and recalls that the evolution from OO toward AO is often presented in terms of paradigm shift (Basili, 1992; Jennings & Wooldridge, 2001; Woodridge & Jennings, 1995). A very important point discussed by Göktürk and Akkok (2004), is that the choice of a paradigm is similar to the choice of a conceptualization/communication language; consequently mixing two different paradigms could be counter productive. What Göktürk and Akkok (2004) emphasize indirectly is what epistemologists call the “incommensurability thesis.” Following this line of argument, two paradigms are totally incomparable since they represent two different set of knowledge.

Based on these related works, our contribution is multiple:

- We emphasize on the epistemological reasons why the paradigm concept is inappropriate to explain the differences between OO and AO. To be considered as paradigm, two theoretical frameworks need to be incomparable and “incommensurable;”
- We propose a new epistemological analysis of the evolution toward AO by using a Lakatosian framework, which can explain and justify the effectiveness of this framework. In this article, AO will be presented as a new research programme widening and improving the knowledge in SE. This improvement cannot be seen as discontinuity of knowledge but rather as continuity since AO could encapsulate object technology;
- By proposing an epistemological reading of the emergence of the AO, we try to reduce what Jennings and Wooldridge (2001) call the “gap” between knowledge and applications. We prove that this evolution can be explained in line with the conventional standards to justify the scientific status of
the SE discipline. The paradigm-concept cannot justify the scientific status whereas the research programme-concept can.

**The Software Crisis**

Although constant progress is being made in software development, complex information systems often imperfectly match users’ requirements. A “software engineering crisis” diagnosis was first made in the late sixties. Considerable work has been done in conceptual modeling and other engineering methodologies and processes so that the crisis can be mitigated. However, the diagnosis still remains partially to date. Indeed, structured methods are not systematically used in software projects and computers expect responsible users, but most of the time the problems encountered are the result of human decisions or methodological insufficiencies.

Software engineering can be defined as the methodological processes and the relevant artifacts used for the development of software on a large scale. The SE crisis has different reasons:

- The increasing complexity of modern software: continuous technological progress allows us to develop larger and larger applications while the increasing demand exceeds the productivity gains obtained by methodological, techniques and tool improvements;
- The common under-estimation of the difficulty and consequently the software development cost (the methods, widely empiric, of individual programming are not applicable to the development of large and complex systems), conflicting software development process and inadequate software products;
- Lack of reliability and maturity of software compared to hardware and its relative importance in the realization of the complex system functions;
- Substantial delay in the diffusion of best practices within the software industry (Davis, 1995; Meyer, 1997; Sommerville, 1992) due to the complexity and cost of education of software developers in a constantly evolving discipline;
- Inherent complexity:
- The problems that a software has to deal with can be arbitrarily complex; for example, the limits of system functionalities are often much less clear than those of tangible products;
- The sequential decomposition of the development phases is not so natural for SE than it is for other engineering disciplines (mechanical engineering for example).

We argue that advances in SE conceptual modeling such as OO and AO are part of the effort to address the crisis; this point becomes important in the context of the epistemological reading, as discussed in the section Epistemological Approach.

**Towards Agent Orientation**

The meteoric rise of the Internet and World-Wide Web technologies has created new application areas for enterprise software, including eBusiness, web services, ubiquitous computing, knowledge management and peer-to-peer networks. These areas demand software design that is robust, can operate within a wide range of environments, and can evolve over time to cope with changing requirements. Moreover, such software has to be highly customizable to meet the needs of a wide range of users, and sufficiently secure to protect personal data and other assets on behalf of its stakeholders.

Not surprisingly, researchers are looking for new software designs that can cope with such requirements. One promising source of ideas for designing such business software is the area of MAS. They appear to be more flexible, modular and robust than traditional systems including object-oriented ones. They tend to be open and dynamic in the sense that they exist in a changing organizational and operational environment where new components can be added, modified or removed at any time.

MAS are based on the concept of agent and are defined as, “A computer system, situated in
some environment that is capable of flexible autonomous action in order to meet its design objective” (Woodridge & Jennings, 1995). An agent exhibits the following characteristics:

- **Autonomy**: an agent has its own internal thread of execution, typically oriented to the achievement of a specific task, and it decides for itself what actions it should perform at what time.

- **Situatedness**: agents perform their actions in the context of being situated in a particular environment. This environment may be a computational one (e.g., a Web site) or a physical one (e.g., a manufacturing pipeline). The agent can sense and affect some portion of that environment.

- **Flexibility**: in order to accomplish its design objectives in a dynamic and unpredictable environment, the agent may need to act to ensure that its goals are achieved (by realizing alternative plan). This property is enabled by the fact that the agent is autonomous in its problem solving.

Agents can be useful as stand-alone entities that delegate particular tasks on behalf of a user (e.g., personal digital assistants and e-mail filters (Maes, 1994), or goal-driven office delivery mobile devices (Mataric, 1992). However, in the overwhelming majority of cases, agents exist in an environment that contains other agents. Such an environment is called a multi-agent system.

In MAS, the global behavior is derived from the interaction among the constituent agents: they cooperate, coordinate or negotiate with one another. A MAS is conceived as a society of autonomous, collaborative, and goal-driven software components (agents), much like a social organization. Each role an agent can play has a well-defined set of responsibilities (goals) achieved by means of an agent’s own abilities, as well as its interaction capabilities.

This sociality of MAS is well suited to tackle the complexity of an organization’s software systems for a number of reasons:

- It permits a better match between system architectures and their operational environment (e.g., a public organization, a corporation, a non-profit association, a local community, etc.);

- The autonomy of an agent (i.e., the ability an agent has to decide what actions it should take at what time (Woodridge & Jennings, 1995)) reflects the social and decentralized nature of modern enterprise systems (Tennenhouse, 2000) that are operated by different stakeholders (Parunak, 1997);

- The flexible way in which agents operate to accomplish their goals is suited to the dynamic and unpredictable situations in which business software is now expected to run (Zambonelli, Jennings, Omicini, & Wooldridge, 2000; Zambonelli, Jennings, & Wooldridge, 2000).

**EPISTEMOLOGICAL APPROACH**

We argue that a Lakatosian vision should be adapted for an epistemological reading of the emergence of AO. To develop our argumentation, we first consider its use in the literature dealing with the “paradigm” concept to characterize the evolution from OO to AO. We then explain why the Lakatosian “research programme” concept is more adequate than the Kuhnian “paradigm” concept to illustrate the evolution of modeling and programming concepts.

**The Traditional Kuhnian Perspective of Software Engineering**

**The Paradigm-Concept: A Definition**

The word “paradigm” comes directly from philosophy where its meaning remains surprisingly rather vague. Plato and Aristotle were the first authors to introduce this concept. According to them, the paradigm is a kind of explanatory model, which allows people to understand, in terms of causality, the changes imposed by...
Nature. However, the paradigm is not, strictly speaking, a logic. For Aristotle, the “paradigm” was, “different from both deduction, which goes from universal to particular, and induction, which goes from particular to universal, in the sense that the paradigm goes from particular to particular” (Göktürk & Akkok, 2004).

The term “paradigm” has not really been used before the 20th century when Thomas Kuhn developed a specific epistemology based on this concept. The paradigm is defined as, “A constellation of concepts, values, perceptions and practices shared by a community and which forms a particular vision of reality that is the basis of the way a community organizes itself” (Kuhn, 1996). Nevertheless, Kuhn himself admits that the use of the word remains rather vague: it is possible to identify twenty-two different meanings of the “paradigm” concept used in Kuhnian epistemology (Masterman, 1970). In the last edition of his book, Kuhn even recognized that the “paradigm” concept is vague but explained that it is close to what Kuhn calls a “disciplinary matrix.”

This leads us to consider in this article the “paradigm” as a way of representing the world, which necessarily includes conceptual tools and methods (the conjunction of these two elements forming what Kuhn called a disciplinary matrix), such that an observer can create models. Each paradigm refers to a particular ontology and represents a subset of “what is representable.” The representation abilities of a paradigm are basically related to the conceptual tools, to the modeling methodology and the use of these two elements by theoreticians.

**Paradigms and Software Engineering**

The first paradigm to be introduced in SE was the procedural paradigm (Göktürk & Akkok, 2004). It was based on the use of algorithms to execute particular tasks. The second paradigm was the data-hiding paradigm, which focused on the data’s organization and introduced the concept of modules (to hide the data’s). This paradigm was followed by the data-abstraction paradigm, which concentrated on the types and on the operations defined on these types. Next was the object-oriented paradigm, “built upon the data-abstraction” paradigm but introducing new concepts like inheritance and polymorphism. Finally, using the flexibility of the component-oriented logic, the agent-oriented paradigm has divided software into independent and communicating entities called “agents” (Woodridge & Jennings, 1995). This last paradigm has been described in detail in the section Towards Agent Orientation.

Programming languages and modeling paradigms are interdependent. The “chicken and egg” metaphor could be used to characterize their reciprocal relationship (Göktürk & Akkok, 2004); sometimes, specific needs for a programming language lead to a better implementation of a modeling paradigm and sometimes, the evolution of the modeling paradigm influences and improves the development of a specific programming language. However, even if programming languages and modeling paradigms are interdependent, the agent-oriented paradigm differs in the sense that it is not formally related to specific programming languages. The concepts used in AO have been inspired from the organizational structures found in the real world. In the beginning, agent-oriented models were implemented in object-oriented languages but further evolutions allow to support and directly implement MAS in terms of full-fledged agent concepts such as Beliefs, Desires and Intentions (BDI) (JACK Intelligent Agents, 2006; JADEX BDI Agent System, 2007).

**A Lakatosian Perspective of Software Engineering**

In the following sections, we will propose to review the Kuhnian vision of OO and AO to demonstrate that it is not best suited to describe the evolution in SE. With respect to the research programme concept developed by Lakatos, we will explain why a Lakatosian understanding of the evolution to AO is more appropriate than a Kuhnian one.
“Research programme” Concept: a Definition

In the continuity of the Popperian philosophy (which will be briefly presented in the following section), Imre Lakatos has developed in 1974 an original approach of science. Lakatos considers scientific theories as general structures called “research programmes.” A Lakatosian research programme is a kind of scientific construction, a theoretical framework, which guides future research (in a specific field) in a positive or negative way. Each research programme is constituted by a hard core, a protective belt of auxiliary hypotheses, and a positive and a negative heuristic.

The hard core is composed of general theoretical assumptions, which constitute the basic knowledge for the program development. In other words, these axioms are the assumptions the theorists will not challenge in their research. This hard core is surrounded with a protective belt composed of the auxiliary hypotheses, which complete the hard core and with assumptions related to the description of the initial conditions of a specific problem. These auxiliary hypotheses will be thoroughly studied again, widened and completed by theorists in their further studies within the program. This widening of the protective belt hypotheses contributes to the evolution of the research programme without calling into question the basic knowledge shared by a scientific community.

The positive heuristic represents the agreement among the theoreticians over the scientific evolution of the research programme. It is a kind of “problem solving machinery” composed by proposals and indications on the way to widen and enrich the research programme. The negative heuristic is the opposite of the positive one. Within each research programme, it is important to maintain the basic assumptions unchanged. It means that all the questions or methodologies that are not in accordance with the basic knowledge must be rejected. All doubts appearing about the basic knowledge of the main theoretical framework become a kind of negative heuristic of the research programme.

When the negative heuristic becomes more and more important, a research programme can become “degenerative” (i.e., it has more and more empirical anomalies). This means that theoreticians have to reconsider the basic knowledge of the program, which can lead to the creation of another research programme. Let us mention that this revision is always a very slow process.

According to Lakatos, we can characterize the evolution of knowledge as a series of “problems shifts” which allow the scientific theories to evolve without rejecting the basic axioms shared by theorists within a specific research programme. The concept of “research programme” represents a descriptive and minimal unit of knowledge, which allows for a rational reconstruction of the history of science.

At first glance, the “research programme” concept seems rather close to the “paradigm” concept. Indeed, it is, in both cases, a matter of “disciplinary matrix” used to describe a particular ontology of the external world. However, differences exist between these two concepts especially in the evolution of science and knowledge in a large sense.

According to Kuhn, the evolution of science does not follow a straight line and does not converge towards something, which would be the “truth.” In the Kuhnian vision, the evolution of science could be represented by a broken line where discontinuity would mark the passage from one paradigm to another. From this point of view, different paradigms cannot be compared. Moreover, Kuhn specifies that a paradigm always emerges within a discipline facing a methodological crisis (characterized by the absence of a dominating theoretical framework) (Kuhn, 1996). Following a crisis undergone by a previously dominating paradigm, a new paradigm emerges with a new language and a new rationality. This new way of thinking does not allow a comparison between the old and the new paradigm. Given that a new paradigm is a new way of thinking about the world, there is no basis for comparison. The “paradigms incommensurability” thesis has become a very
well known issue in the philosophy of sciences (Sankey, 1994).

Lakatos decomposes the evolution of science into successive methodological and epistemological steps. These steps form a kind of vertical structure built with a multitude of “layers of knowledge” and where each layer represents a particular research programme. In the Lakatosian vision, the emergence of a new research programme is induced by an empirical degeneration of a previously dominating research programme. The new research programme will constitute a superior layer of knowledge, which will integrate the same conceptual tools as the former but which would be able to solve its empirical anomalies through what Lakatos calls a “problem shift.” The latter is characterized by an extension or a redefinition of the protective belt of the preceding program. In this vision, research programmes remain comparable to each other (in both conceptual and empirical terms). The language and rationality of the new research programme result from a progressive evolution of knowledge and from the resolution of the empirical anomalies of the previous research programme. In contrast to the Kuhnian vision, Lakatos explains that there is no discontinuity between the different research programmes.

Agent-Orientation: Paradigm VS Research programme

In this section, we present three main arguments for the use of the Lakatosian research programme to understand the shift from OO to AO.

Kuhnian Crisis or Lakatosian Problem Shift?
As discussed in the first part of this article, a SE-crisis has been observed due to the fact that few software projects successfully manage to fully satisfy users’ requirements. In the Kuhnian vision, this crisis could be considered as a favourable argument for the emergence of a new paradigm. In this perspective, a crucial question raises: “Does the current crisis characterize the end of a dominating paradigm or is it simply the result of a pre-paradigmatic step specific to young sciences which have not found a dominating paradigm yet?” Using the Kuhnian rhetoric to analyse this crisis, we can consider the situation as a paradigm evolution. Indeed, the pre-paradigmatic step was rather characterized by the procedural framework (which was defined by an algorithmic and sequential i.e., a strictly computer/mathematical logic) as well as the data-hiding and the data-abstraction frameworks. This pre-paradigmatic period was essential to the evolution process towards OO. However, the crisis situation observed in SE must be carefully analysed. Even if the “SE-crisis” diagnosis has been noted for several years, we think that the context in which AO has emerged cannot be considered as a crisis in the Kuhnian sense. Indeed, most of the methodological rules existed before AO and the current software development process does not seem to be so chaotic: IT specialists dispose of analysis methods and methodological tools with a high level of abstraction (Castro, Kolp, & Mylopoulos, 2002; Kruchten, 2003). These elements tend to show that what looks like a crisis is rather an (animated but normal) evolution of knowledge in SE (Odell, 2002; Skarmeas, 1999), which could be interpreted as a “problem shift” in the Lakatosian vision.

Kuhnian Discontinuity or Lakatosian Continuity?
We consider that the Kuhnian discontinuity between paradigms is not appropriate to explain the emergence of AO because there is no real “fracture” between OO and AO. Indeed, in some software solutions, communicating objects are used and are completely relevant and sufficient. In software problems where no learning skills are valuable, the use of agents would not bring crucial advantages: they would just transfer messages and would not behave like learning and collaborating agents pursuing goals (Odell, 2002). In this special case, there is no contribution of the agent concept to the software solution in comparison to object technology. We can see that the cohabitation within the same application between modules exploiting object technology and others exploiting agent technology can thus
be an “optimal” solution (Odell, 2002). In a Lakatosian vision, this cohabitation represents a progressive evolution of knowledge in SE. Indeed, the Lakatosian epistemology implies that the transition between research programmes is not clear and depends on the specific aspects of the experiment conducted (the software solution is the experiment in our case). A hybrid solution between modules developed on the basis of objects and others on the basis of agents can thus be explained by the continuity between research programmes inherent to the Lakatosian vision of knowledge applied to SE.

Kuhnian Incommensurability or Lakatosian Commensurability?

Another drawback of the adoption of the Kuhnian epistemology in SE is the incommensurability between paradigms. OO and AO can be compared since collaborating agents can be used as communicating objects and, more important, agents can be implemented using object-oriented languages (see for example (Bigus & Bigus, 1997; Bellifemine, Poggi, & Rimassa, 2000). In this perspective, agents can be considered as “super objects” (i.e. objects possessing skills as collaboration, intentionality, learning, autonomy, reasoning, etc.) If we consider SE development as a history of “raising the level of abstraction,” AO can be seen as an evolution of OO because it raises that level a little higher (Skarmeas, 1999). In this perspective, research programmes preceding OO can also be considered as lower layered than the later. This vision perfectly matches with the Lakatosian concept of layers of knowledge introduced earlier. Indeed, considering SE evolution, each new research programme raises the abstraction level and constitutes a higher layer of knowledge. These layers are comparable so that OO and AO are said to be commensurable.

Lessons Learned

AO is based on the basic knowledge that existed before its emergence. We could say that agent-oriented modeling and programming has a hard core composed of the concepts defined in the previous research programmes (procedural, data hiding and data abstraction) on the one hand, and the artificial intelligence field (Woodridge & Jennings, 1995) on the other hand. The protective belt of the AO research programme would be characterized by the evolution towards a widely used SE methodology allowing the development of large projects.

Table 1 summarizes the contrast between the Kuhnian and Lakatosian epistemologies applied to the evolution from OO to AO. The lessons learned are:

- The context in which AO has emerged cannot be considered as a Kuhnian crisis because of the existence of strong methodological rules that emerged within OO preceding the emergence of AO. We claim there is a problem shift;
- AO seems to be based on the evolution of the previous methodological rules so that we point to continuity between OO and AO;
- OO and AO are directly commensurable since the latter can be conceptualized as a knowledge layer upon the first.

In the light of the arguments presented above, we contend that the Kuhnian epistemology often referred to in the literature is not appropriate to provide a correct epistemological analysis of the knowledge in SE. We rather propose to use the Lakatosian epistemology to characterize the emergence of AO.

We argue that the Lakatosian epistemology is directly in line with the idea of computer knowledge depicted as a “structure in layers.” We have represented this architecture in Figure 2.

Implications

In this section, we briefly discuss the implications of using the concept of research programme rather than paradigm to characterize OO and AO for both SE researchers and practitioners.

Our work provides SE researchers a complementary view and the specific nature of the
Moreover, the impact of this nuance is important due to the fact that, as we have pointed out, no fundamental departure was observed when evolving from OO to AO. Therefore, concepts developed in OO and other related research areas can be adapted to AO with some consequence on the process of building agent ontologies.

Managers and other SE practitioners will learn that AO should be envisaged as a natural evolution rather than as a complete revolution. AO can be seen as complementary to OO and leads to the fact that the modularity of software solutions resulting from several techniques can be utilized when developing new systems. Software modeling techniques can then be considered as problem oriented (i.e., modeling, design and implementation techniques are driven by the problem specificities), rather than solution oriented (where those should be driven by an ex ante methodological/technical choice). This conclusion can be taken at different levels:

- At the analysis stage where organizational modeling can be better performed using models representing collaborative agents (Yu, 1995) while purely functional requirements can be simply modeled using object modeling languages such as UML use cases;
- At the design stage, entities required to collaborate and learn can be designed as agents while others requiring less sophisticated behavior can be designed as objects;
- At the implementation level, modules can be implemented using agent technologies.
that facilitate communication with modules implemented in object technology.

Finally, our new conceptualization has a profound impact on the SE development life cycles. Indeed, the literature on the evolution of SE life cycles is rich and the adoption of mature development life cycles such as the spiral model, the Rational Unified Process, or agile methodologies traditionally operationalized by OO technologies can be adapted to AO development without revising the fundamental concepts.

CONCLUSION
User requirements that have been poorly taken into account as well as modeling and development languages inspired by programming concepts as opposed to organization and enterprise has led to a software crisis. Solutions can be found at different levels, and among those, OO at the first level and AO at the second level constituted definite progress. AO furnishes concepts to model the organization more precisely, thus enabling analysts to create more accurate models.

This article has presented an epistemological analysis of these improvements. AO is a promising innovation in tools allowing analysts to model the organization and user requirements. We have shown in this article that the emergence of this innovation can be better studied using Lakatosian ideas rather than Kuhnian ones. In this vision, AO would be considered as a research programme rather than a paradigm. Even if it seems to be just a matter of terminology, the difference between these two epistemological concepts is clear and profound so that they should not be confused. This article also advocates that the approaches taken by both researchers and practitioners to software problems and their possible solutions is influenced by the vision of OO and AO as a research programme.

More work should be carried out in SE fundamentals by studying other methodological frameworks and epistemological foundations to provide the computer science researcher a more accurate conscience of the research field that the researcher is working on. Software development life cycles such as waterfall or iterative development can also be studied from such a point of view. The latter is, for example, strongly inspired by Herbert Simon’s bounded rationality principle as well as Popper’s knowledge growth theory.

REFERENCES


Yves Wautela is an IT project manager and a postdoc fellow at the Université catholique de Louvain, Belgium. He completed a PhD thesis focusing on project and risk management issues in large enterprise software design. Dr. Wautela also holds a bachelor and master in management sciences as well as a master in information systems. His research interests include aspects of software engineering such as requirements engineering, software project management, software development life cycles and CASE-Tools development as well as information systems strategy.

Christophe Schinckus is a postdoc fellow at the Facultés universitaires Saint-Louis in Brussels, Belgium. He completed his PhD in financial economics at the University of Paris I Panthéon-Sorbonne. Economist and philosopher of science, Dr. Schinckus works on epistemological analysis of financial theory and philosophy of economic knowledge. He has also a BSc in computer science and is interested in software engineering processes.

Manuel Kolp is an associate professor in information systems at the Université catholique de Louvain, Belgium where he is also head of the Information Systems Research Unit and Academic Secretary of Research for the Louvain School of Management. Dr. Kolp is also invited professor with the University of Brussels and the University Faculties St. Louis of Brussels. His research work deals with agent-oriented and socio-technical architectures for e-business and ERP II systems. He was previously a Post Doctoral Fellow and an adjunct professor at the University of Toronto. He has been involved in the organization committee of international conferences and has chaired different workshops. His publications include more than 50 international refereed journals or periodicals and proceedings papers as well as three books.