# Developing Knowledge-Based Applications in Engineering: Quality, Productivity, and Technology Transfer Issues

P. Baroni*, G. Guida*, G. Lamperti* and S. Mussi°
* Dipartimento di Elettronica e Automazione
Università degli Studi di Brescia, Brescia
Italy
° CILEA (Consorzio Interuniversitario Lombardo per la Elaborazione Automatica),
Segrate (Milano)
Italy

**Abstract**. This paper investigates quality, productivity, and technology transfer issues in relation to the development of knowledge-based components of large and complex software systems. The concepts of quality and productivity are first introduced and their mutual relationships discussed. The role of technology transfer is then analyzed. Later, a novel proposal of an integrated approach to quality, productivity and technology transfer is illustrated. Finally, the main features of the proposed approach are discussed and the basic ideas concerning an original interpretation of the object-oriented paradigm are outlined.

# 1. Introduction

*Knowledge-based* (*KB*) technology has been rapidly growing over the last decade - for an introduction the reader may refer to: [34, 14, 9, 17, 23, 32, 11]. The impact of *knowledge-based systems* (*KBSs*) on industry and business has been impressive and convincing. Process supervision, diagnosis, production scheduling, troubleshooting, configuration, envisioning of economic and social scenarios, decision making, and cooperative human-computer interaction are just a few examples of tasks where KBSs proved appropriate and successful. Gradually, KB technology has established itself as a crucial factor for the construction of advanced software applications: integration of KB components into more traditional software systems has become the leading edge of innovative software applications.

However, the number of real operative applications that include KB components is still limited and in several fields KBSs are not yet accepted as definitely reliable components of complex systems. The main reasons of this apparent contradiction are:

- verification and validation of KBSs are to a large extent still open problems, generally faced by means of inadequate methods and tools; the issues of quality assurance and certification of KBSs are only rarely considered;
- the design of technically sound KBSs is a fairly complex task, that requires specialized competence and experience; development time and cost are generally high.

Moreover, even though KB technology is no more a novelty in almost all industry and business environments, in several cases it is only known at a theoretical level and only in its basic (and less useful) components. Moreover, KB technology is still rapidly growing and the non-specialists are often not aware of the most recent advances. Always, when a new project exploiting KB technology is started, there is a fundamental need of technology transfer.

Therefore, three aspects emerge as the fundamental issues for the establishment of KB technology on a more sound and accepted background, namely: *quality*, *productivity*, and *technology transfer*.

Before analyzing these aspects, it is worthwhile pointing out that KB technology has gradually turned out in last decade to be a valuable tool for developing sound and innovative engineering applications. Contrary to the pessimistic and not completely objective analysis developed in [29], KB technology offers today at least two useful things to engineers:

- a sound and general methodology for knowledge acquisition, analysis, and modeling (namely, *knowledge engineering*), that can help in a variety of tasks of primary importance in engineering, such as understanding and representing complex physical processes, modeling complex artifacts, eliciting the expertise and know-how of human experts, managing large knowledge assets, etc.;
- an integrated set of methodological and software tools (namely, knowledge-based technology) for building advanced engineering applications that exceed the capabilities of traditional approaches, such as task-oriented support systems, specialized problem-solvers, advanced human-computer interfaces, innovative control systems, etc.

Both knowledge engineering and KB technology share with the wider domain of engineering the ultimate objective of using all available knowledge and resource to build useful artifacts that can help achieve complex and challenging goals. Developing and using models belongs to the proper aim of engineering: knowledge engineering offers powerful tools for building, validating and refining models, and KB technology provides concrete means for exploiting models through powerful software applications. Of course, KB technology does not aim at replacing nothing existing - models, computer programs, or human experts; rather, it can help engineers build more effective and robust artifacts, that can better satisfy user needs and requirements. However, such benefits can only be foreseen if at least two conditions are met: high

and certified quality of the systems developed, and reasonable development time and cost.

This paper focuses on quality, productivity, and technology transfer issues in relation to KB components of large and complex software systems. Note that, although the analysis developed in this paper specifically refers to KB technology and is grounded on the experiences of the authors in this field, most of the conclusions reached apply in more general terms to the whole area of emerging software technologies, including, among others, neural networks, case-based systems, and genetic algorithms. In fact, abstracting from many specific details related to KB technology, it is possible to recognize that most of the considerations presented and of the reasoning lines developed do not depend on such details, but rather on some more general features that are common to any advanced software technology. Such features include the following facts:

- the technology has become available for industrial exploitation only relatively recently;
- it is still in rapid evolution, witnessed by the existence worldwide of a significant number of basic research projects related to its improvement;
- the basic know-how necessary to understand and effectively exploit the technology has a very limited diffusion in industrial environments and has been only recently included (or, even worse, has not been included yet) in standard academic teaching programs;
- the introduction of the technology allows the realization of products and services that were unfeasible or even inconceivable with previously existing technology.

As it will be clear in the sequel, many of the difficulties related to the critical issues of quality, productivity, and technology transfer in the field of KBSs, can be blamed more on these relatively general aspects rather than on more specific peculiarities of KB technology. Consequently, this paper may be useful not only to KBS practitioners, but also to anyone interested in the industrial spread of emerging software technologies.

Complex and sophisticated software applications in business and industry, such as financial advice giving, simulation of macro-economic scenarios, integrated building design, supervision of chemical plants, decision-support systems for power plants, environment monitoring, and intelligent cockpit aids, normally require the use of several, different software technologies, both traditional and advanced. The design of such complex software systems requires, therefore, at a very high-level of abstraction, a specific and deep analysis of the problem at hand in order to:

- develop a coarse-grained architecture of the system to be developed, appropriately organized according to sound decomposition and modularization criteria;
- select the most suitable implementation technologies, traditional or advanced (in particular KB technology), for each module of the architecture, in order to ensure the best matching between the features of the problem solved by the module and the specific characteristics of the adopted technology.

The above points implement the so called *technology-level design*, which is of primary importance for the correct, effective, and disciplined exploitation of available technologies in the construction of complex software systems. As a matter of fact, a skillful selection of implementation technologies is a key factor for the technical success of a project, with a primary impact on system design, project planning, resource allocation, scheduling, and costing. Disregarding the technology-level design issues is a serious error, which may entail negative effects on the whole project, thereby compromising its results.

The technological mosaic required by complex software applications offers evidence that the global quality of an heterogeneous system (namely the *quality in the large*) requires the quality of all the technological components the system is made up of (namely the *quality in the small*). Note that the provision of quality in the small for each technological component cannot guarantee quality in the large of the whole system. Therefore, quality in the small is only a necessary, but generally not sufficient, condition for quality in the large. Achieving quality in the large, through a cost- and time-effective process, is the ultimate objective of software production for advanced industry and business applications.

# 2. Quality and productivity issues

## 2.1 Basic issues

Quality and productivity are crucial issues not only in the KBS area [15], but in any software field, and, more generally, in any production environment. However, quality and productivity feature quite specific characteristics for KBSs. The quality model defined for traditional software systems does not apply to KB components, and also the productivity criteria that govern software development cannot be easily transferred to KBSs. KBSs are a very particular type of software, that differs substantially from traditional systems. In fact [16]:

- KBSs support explicit representation of domain knowledge, which is instead not explicit in the code of a traditional program;
- KBSs always feature a clear separation between the representation of the general problem-solving algorithms, coded in the procedures of the reasoning mechanism, and the representation of domain-specific knowledge, stored, in a mainly declarative form, in the knowledge base;
- KBS operation is basically non-deterministic, i.e. it essentially involves searching in a space of alternatives, while the behavior of traditional software is strictly deterministic;
- KBSs are designed and constructed according to a life cycle which is substantially different from that of traditional software;

- KBS development involves crucial activities devoted to knowledge elicitation, analysis, modeling, representation, integration, verification and validation, which are not relevant to the production of traditional software;
- in KBSs it is often impossible to fully specify the desired behavior and to identify the most suitable technical solutions before implementation begins; a typically iterative development allows changes to be made to specifications and experimentation with alternative technical approaches during prototyping.

## 2.2 The quality issue

The quality issue is increasingly becoming the issue of next years. The current decade is considered the quality era, in which software quality is quantified and brought to the center of the development process [5]. There is a vast range of critical applications (safety critical, production critical, result critical, time critical, etc.) in which the quality of software components is even a major requirement. In the past, poor quality software has been implicated in disasters in the worlds of finance, clinical care, air transport, and nuclear power - the reader may refer to [24] for an interesting yet dramatic overview of such disasters. In the domain of decision-aid systems, operators of nuclear power plants, train dispatchers and drivers, sea transportation controllers, and airline pilots, share a common responsibility for human lives that give an extra dimension to their decisions. These operators are required to carry out a number of tasks on a complex system within a quick-changing environment. Interestingly, there is evidence that KBSs may support such tasks appropriately. However, the critical aspect of the tasks invariably requires that the relevant software components are quality certified. In other terms, without certified quality, critical application domains cannot accept KB software components. Consequently, though the main demand of KB technology is for critical applications, the quality requirement is the main obstacle to the full exploitation of KB technology. To overcome this impediment, KB software components are expected, for the future, to be developed in such a way that their quality can be certified.

Certified quality means both quality as a substantial, real, tangible attribute and certification as a formal, official, authoritative statement. Note that quality and certification are to a large extent different, orthogonal concepts. Quality may exist without being certified and, conversely, formal certification may be only a piece of paper aimed at hiding the real poor quality of the software artifact. However, in the ideal perspective, a formal certification should always recognize real quality.

*Quality* may be defined as the totality of features and characteristics of a product that relate to its ability to satisfy stated or implied needs [20]. As part of the supporting life cycle processes, *quality assurance* defines the activities for objectively assuring that a software product and processes are in conformance with their specified requirements and adhere to their established plans [21]. According to [22] (currently under revision) software quality should be evaluated according to a specified set of quality characteristics, including: functionality, reliability, usability, efficiency, maintainability, and portability. Each characteristic should be individually

measured using appropriate metrics. The obtained measures should then be individually rated according to an established rating procedure, and, eventually, a global assessment should be derived using appropriate assessment criteria.

It is worth highlighting that the quality model may focus either on the *product*, namely the KB artifact, or on the *process*, that is the repertoire of methods, techniques and tools which are exploited for developing the KBS. Product and process quality models stem from different convictions and correspond to quite different approaches. Behind the product-oriented approach is the assumption that a product's tangible internal characteristics determine its external quality attributes. On the other hand, the process-oriented quality model is based on the claim that the quality of the development process can guarantee the quality of the final product. Both approaches present advantages and disadvantages. Among the main advantages of the product-oriented approach are the following:

- at least in principle, the approach is applicable to almost all KB products, independently of the development process;
- the approach is especially appropriated for small companies, that cannot afford the costs to establish a sophisticated, quality-oriented KBS development process, even though they can produce KB applications of a satisfactory quality.

The advantages of the process-oriented approach include, among others:

- the approach is applicable to a wide range of KB products, developed under the same process;
- the approach is especially appropriate for large companies, that produce KB applications on an industrial basis; in such cases, the costs of establishing and certifying a KBS development process are, to a large extent, lower than certifying any individual KBS (or any version of it) with respect to a product quality model.

In especially critical cases, both the process-oriented and the product-oriented approaches must be adopted in order to achieve the highest level of quality assurance.

*Quality certification* is the process through which a third independent party gives written assurance that a product or process conforms to specified characteristics. To this end, considering for example the product-oriented approach, it is essential to define particular quality characteristics of a KBS and assure their conformance with stated specifications. Conformance may be assured only if quality characteristics may be measured unambiguously.

## 2.3 The productivity issue

Quality is not the only requirement for a KBS. Quality is expected to come with a reasonable amount of resources, that is, quality must be cost-effective. So, beside quality, productivity is the additional major requirement for KBSs. *Productivity* may be defined as the capability of developing a KB product within reasonable time limits and costs. In practice, productivity entails the ability of a project team to accomplish

the various tasks it is in charge of in the shortest possible time, ensuring, however, that the products satisfy all stated specifications. The issue of productivity also includes, therefore, the ability to plan and control the whole KBS development process. Traditional software production suffers from inadequate productivity. This shortcoming is even emphasized in the production of KBSs, where highly specialized labor is required. Invariably, project schedule is unrealistically tight. Due to either lack of planning or unreasonable customer demand, many KBS projects start with insufficient time to do an appropriate job. Even worse, sometimes, the allocated resources are reduced while the project is underway.

Two main approaches have dominated, in past years, the field of KBS development, namely the craft-based and the technology-based approaches.

The *craft-based approach* is founded on the claim that the development of a KBS relies more on the individual skill and capability of the designer than on a structured and disciplined process. As such, the craft-based approach does not make a clear distinction between the different stages of KBS development, nor does it assume a well-defined life cycle concept. The designer is viewed as a craftsman and, at times, as an artist. The assumption that each KBS is a unique artifact leads to the conclusion that good development of KBSs can be carried out only through extensive experimentation and prototyping. Consequently, methodological issues are generally not considered: talent tends to replace methodology. However, if it must be acknowledged that very good and successful KBSs have been designed within the craft-based approach, it must also be recognized that this approach suffers from several limitations. The craft-based approach does not pursue productivity, but favors individual excellence. It does not support discipline, work sharing, and rational project management. Moreover, focusing on the creation of a unique artifact, it is largely inappropriate to face rapid changes in application requirements and in supporting environments. Finally, since specific skill in KBS design and development can only be achieved through application, talent and long experience are necessary conditions to become a specialist. Consequently, the cost for a KBS project of engaging real KBS specialists may become so great that small-scale, low-budget projects will not be able to justify their use. In such a scenario, a KB application can become a luxury for the few who can afford it. Therefore, approaches which rely less on talent and more on structure and procedure are likely to be more cost-effective for the future.

The *technology-based approach* assumes that the development of a KBS can be greatly improved only by resorting to sophisticated computer-based development tools. It is claimed that KBS designers are not productive because of a lack of automated tools during development, and technology is advocated to be the answer to the problem of productivity. Developing a KBS is not a trivial task. Normally, developers find themselves under intense commercial pressure to release the KBS. In such circumstances, the KBS is unlikely to get the analysis and design effort it requires and, once implemented, it is unlikely to meet user requirements. From the technology-based perspective, the solution to this problem is to provide automated development tools that will free developers from ordinary time-consuming tasks and leave them more time to spend on analysis and design. It is a fact that there is (not only in the KBS field) over-reliance on automated tools and that tools are often sold

as a remedy for all problems. These deformed approach has created a false sense of trust, which is likely to inflict untold damage on the KBS industry. The technology-based approach also encourages rapid prototyping as a means of involving users in the KBS life cycle. The availability of advanced support tools which support the rapid development of KBSs has certainly made prototyping more attractive and more viable for KBS developers. Without automated tool support, building of prototypes would be commercially impractical. However, the technology-based approach on its own is not sufficient to solve the problem of productivity. The design of a KBS begins long before the first prototype is developed. Consequently, unless effective assistance is provided also in the early analysis and design phases, automated tools will simply support the rapid and cost-effective development of low-quality KBSs.

The above discussion provides evidence for the claim that both the craft-based and the technology-based approaches to KBS development are to a large extent ineffective. A possible answer to this problem can be provided by a more recent approach centered on the concept of methodology. The *methodology-based approach* assumes that productivity can be substantially improved by providing KBS developers with a sound life cycle and an effective development methodology. The methodology-based approach is grounded on the issue of integrating different techniques and tools for KBS development within a structured and disciplined framework. As such, the methodological approach better support productivity, whilst retaining all the valuable features of automated tools and without sacrificing skill and talent. Still better, it provides the potential for the design of new and more powerful tools aimed at globally supporting the whole methodology or at providing specialized help for some specific phases or tasks. However, what distinguishes the methodology-based approach from the pure technology-based one is that in the former the focus is on methodological issues rather than on the supporting technology. A tool for a KBS designer plays the same role a word processor plays for an author. The ideal, mature KBS designer has realistic expectations on tools. He or she does not simply rely on tools without first understanding and establishing clear methodological issues, such as principles, processes, methods, and techniques. He or she is well aware that tools are, indeed, simply tools. Finally, the methodology-based approach also offers specific support for skill and talent: it provides a structured framework to accumulate and integrate individual and company know-how, to store and preserve it, to improve it with experience, and to make it available to all interested parties. The methodology-based approach appears to be the most pragmatic means of overcoming the current limitations of the craft-based and technology-based approaches.

## 2.4 Quality and productivity: conflict or synergy?

Any approach to KBS development has the basic objective of improving both quality and productivity. However, it is commonplace to consider quality and productivity as independent and, often, contrasting aspects. Most people do believe that if one wants

a quality KBS, fatally productivity will be low and, vice versa, if one pursues high productivity, the quality of the final product will be invariably poor. Perhaps, this prejudice is grounded on the belief that the craft-based approach, possibly supported by automated tools, is the only or the best way to building KBSs. In our experience, by contrast, quality and productivity are not independent and generally not conflicting issues. Low productivity is often the symptom of poor quality. Quality is most often the outcome of a simple, disciplined and highly productive development process. There are several good reasons to claim that a project which is well structured, well planned, and founded on a robust methodological basis, is likely to be highly productive and likely to yield a final quality product as well.

Interestingly, quality and productivity are synergetic rather than conflicting issues. Consequently, the dilemma is no longer whether giving priority to either the former or the latter. In the near future scenario, the real dilemma will be whether to pursue the objective of quality *and* productivity as a whole or to continue building KBSs relying only on craft and technology. Whatever idea or prejudice one may have, it is evident that denying one aspect of the quality-productivity conjunction, inevitably leads to inconsistent conclusions. Specifically, if productivity is the main objective and the requirements on quality are relaxed, then the final KB artifact is likely to be, at best, a good prototype, without real chances to be actually used in concrete applications. Conversely, if the objective is quality and productivity is not explicitly pursued, then invariably the development costs become so high that a real industrial business will be impractical in a commercial environment. Thus, quality and productivity represent a vital factor for the future of KB technology. Even though it is still possible nowadays to provide KB applications which are unnecessarily costly or (at times, and) of poor quality, quality and productivity are increasingly becoming the competitive factors for KB technology developers and providers.

Note that all the different approaches to KBS development (namely: craft-based, technology-based, and methodology-based) share the same objective of improving quality and productivity. What separates them, is their view about where the improvements in quality and productivity will come from.

For the craft-based approach the source is individual skill and talent. The general motto is: "Great designs come from great designers". As a matter of fact, it cannot be denied that talent can produce quality KBSs productively. After all, almost all of the successful KBSs in use today are the result of talented designers. However, talent is in short supply. Creativity should always be encouraged, but it cannot be relied upon to meet the needs of industrial KBSs development in the future. The design setting is rapidly changing; large development teams employing KBS specialists are increasingly becoming a luxury, while KBS development by small groups or individuals to meet local needs is a rapidly emerging trend. In this context, resorting to skilled and talented specialists (the KBScraftsmen) will simply be not possible.

For the technology-based approach, the source of quality and productivity is the technology or, better, the black-box process embedded in the automated tools aimed to support KBS development. Actually, such tools implicitly impose a specific process upon a number of steps in the KBS life cycle, as the developer is required to adhere to the programming paradigms they provide.

Finally, underlying the methodology-based approach is the assumption that the source of quality and productivity is the orchestration of techniques and automated tools through an explicit and transparent methodology. Of course, a methodology cannot replace talent but, at least, is likely to reduce dependence on it. Also a methodology should not limit creativity, but supplement it with structure and discipline. In the methodology-based approach, quality and productivity are determined by the nature of feedback applied during an iterative process of evaluation and refinement, not by the talent-based quality of the initial KBS design. Moreover, in order to ensure quality and productivity, a methodology supports better communication among designers, developers and users.

As a last remark, note that the technology-based approach does not exclude methodological issues. In fact, most tools do provide a sort of *methodology in the small*, that is, they are meant to support in a disciplined and structured way only a limited number of phases or tasks of the KBS life cycle (for example: technical design, coding, or verification). What distinguishes the methodology-based approach from the technology-based one is the scope of the methodological support they are expected to offer. Differently from a tool, a methodology is essentially a *methodology in the large*, which is aimed to support the complete KBS life cycle, in all its phases and tasks.

Indeed, a *methodology* can be viewed as composed of a collection of methodological *components*, each one aimed to support a subset of the life cycle. In turn, a component can be composed of other sub-components, and so on. This evidence leads to the notion of *methodological tree*, that is a tree representation where nodes represent components and the successors of a node represent its sub-components. Clearly, the more a component is placed down in the tree, the more the methods, techniques and tools it prescribes are focused and specific. Intuitively, a methodological tree parallels the notion of a top-down program structure, in which each module is defined in terms of other finer-grained modules. After all, a methodology is a kind of prescription or recipe (exactly like a software program) which, by means of a hierarchical representation (phases, tasks, activities, etc.) and of a set of control structures, such as IF-THEN, CASE, LOOP, PARALLEL [16], specifies the process to build a given artifact.

## 2.5 Standards for quality and productivity

When tackling the quality and productivity issues, it is commonplace calling for standardization. Generally speaking, standards concern prescribed ways of discussing, presenting or doing something, which seek to achieve consistency across products (and production processes) of the same type [19]. Software standards are, at least in theory, aimed to produce good software artifacts. The main advantages advocated for standards include, among others, a common terminology, portability, interoperability, improved maintainability, and reduction in training.

However, there is no general consensus on the effectiveness of software engineering standards in improving quality and productivity. Interestingly, in [33] the criterion substantiating effectiveness is the following: "a software standard is

effective if, when used properly, it improves the quality of the resulting software products cost-effectively". Surprisingly, on the basis of an empirical study conducted on more than 250 standards, [31] found no evidence that any of the existing standards are effective according to this criterion. Compared to traditional engineering standards, software engineering standards pose specific, domain-dependent problems, among which are the following:

- Software standards overemphasize process instead of the final product, thereby they implicitly call for the process-centered approach to quality. However, there is no general consensus on the quality of standardized processes, nor is there evidence that a good process necessarily entails good products.
- Many software standards are not standards in the traditional sense. Traditional standards state a set of requirements which are sufficiently precise to be assessed. When such precision cannot be provided, the terms "guidelines" or "code of practice" are used. The lack of such a distinction in software standards makes it possible the tacit proliferation of standards which are at best guidelines.
- It is extremely difficult, if not impossible, to check compliance of a software product with standards. Consequently, it is not possible to certify the application of the standard.
- Often, software standards prescribe the use of methods, techniques, and tools the effectiveness of which, for achieving high-quality products, is not proved.
- Often, the scope of standards is too large, such as, for example, addressing the complete software life cycle.

On the basis of the above remarks, it comes out that, in software technology, standards and quality are to a large extent independent issues. Standards are basically meant to overcome the *Babel Tower Syndrome*. However, the fact that all people speak the same language does not mean that that language is the best one, allowing for highest performance in communication. Undoubtedly, uniformity has its own advantages, but uniformity is not an absolute value and does not entail quality. Even worse, there is to some extent a conflict between standards and quality: a standard is inevitably general and abstract, while quality is concerned with artifacts, which are specific and concrete. Finally, quality and productivity do not come from existing standards, nor are they likely to come from future better standards. What can make the difference is the non-standardized (and non-standardizable) repertoire of methodologies, techniques, and tools which represents the specific, unique heritage of any organization involved in software production.

From this discussion it might seem that standards should not be considered at all or, even worse, that standards are in contrast with quality and productivity. On the contrary, the point we wish to advocate is not that standards should be avoided, thus leaving companies free to develop KB applications on the basis of their know-how and expertise only. Rather, we argue that standards should be always accompanied by guidelines for tailoring and customization, so as to be effectively integrated with the specific enterprise heritage. The potential for opening the way to quality and productivity may be achieved by coupling the general, non-competitive concepts

stated by international standards with specific and highly competitive corporate know-how. But the customization process necessary to merge standards with know-how is by no means straightforward. The quality of the generic and abstract standardized process is likely to deteriorate or even disappear in a poorly supported tailoring process. In this perspective, therefore, a standard can be viewed as a container (more formally, as a class) of methodologies. Specific methodologies may be distilled from standards. But, following the metaphor of the distillery, the final liqueur (the methodology) is not simply what is distilled from raw materials (the standards). In order to be tasty (of high quality) the distilled liquid (the tailored standard) must be mixed (integrated) with special secret aromas and fragrances (the enterprise know-how) which eventually make it successful.

# 3. Technology transfer issues

## 3.1 Basic concepts

Stating, in general, that a given technology (such as KB technology or any other emerging software technology) can be exploited in industrial environments in a cost-effective way, satisfying both quality and productivity requirements, is a necessary, but clearly not sufficient, condition in order to get any industrial company to actually exploit such technology. In fact, an advanced software technology is not just a plug-and-play component, that a company can simply buy on the market and then directly plug it into its productive organization. Acquiring and effectively inserting a new technology within a company organization, always require a specific technology transfer activity. In very general terms, *technology transfer* may be defined as the complex process of transferring from an organization to another three different levels of knowledge about a new technology, namely:

- the transfer of *know-how*, concerning the acquisition of specific working tools, techniques, and methodologies;
- the transfer of *know-what*, concerning the awareness of the global meaning of a technology and of the general rules that govern its correct technical application;
- the transfer of *know-why*, concerning the deep understanding of the potential of a technology and of the conditions for its effective exploitation from the economic and organizational points of view.

Only if this process is carried out successfully, effective exploitation of the transferred technology will be possible. Therefore technology transfer represents another crucial issue, as much important as quality and productivity, in the area of emerging software technologies. Moreover it should be stressed that this issue is not only present when a company decides to undertake the exploitation of a technology it has never used before.

Technologies are always evolving, driven by progress. In some fields, including KBS and other emerging software technologies, this evolution is still particularly rapid and tumultuous.

Therefore, also for technologies which already belong to the current production practice of a company, it is crucial to constantly watch over increasing opportunities provided by new scientific and technical results and to periodically set up technology transfer initiatives, in order to keep up to date the company technological asset. So, as well as quality and productivity, technology transfer is not just a crucial, but also a permanent issue.

The above general considerations, fit particularly well to the case of KB technology. In this field, technology transfer has always been a bottleneck. As mentioned in Section 1, if most managers and project leaders are nowadays aware of the existence of KBSs, only a few of them have a realistic vision of how KBSs could actually contribute to their business, and only a still smaller number of these have correctly and effectively adopted KB technology as a powerful innovation factor. Moreover, most of them are not up to date with the most recent and advances, that often offer real breakthroughs for the development of competitive applications.

KB technology transfer is a very complex task, whose management often eludes a rational justification. The reasons why some technology transfer initiatives succeed and others fail are not always clearly understood, and no general and safe recipe is available yet.

In the following subsection we review some approaches to KB technology transfer and on the basis of considerations similar to those developed for quality and productivity issues, we claim that a methodology-based approach to technology transfer should be adopted. We then identify some requirements, derived from previous experiences of the authors, a technology transfer approach should meet and analyze some common failure causes and success conditions of technology transfer initiatives.

## 3.2 Approaches to technology transfer in the field of knowledge-based systems

The very important role of technology transfer in supporting the diffusion and the institutionalization of KBSs has been early recognized [25, 16].

Several approaches to technology transfer in the field of KBSs have been reported in the literature; among others:

- naive approach or AI soup approach [18];
- strategic approach [18, 28, 1, 12];
- top-down approach or grand plan approach [7, 10];
- step-by-step approach [13, 27] or infusion approach [18] or incremental approach [12];
- bottom-up approach [30] or grassroots approach [7].

However, most of these approaches are strictly bound to the context where they have been firstly developed, are poorly general and hardly re-usable in practice. Attempts to apply them in new KBS initiatives often revealed problematic. A detailed analysis and review of these approaches is beyond the scope of this paper, however, in short, two main reasons for these problems can be identified:

- their definition is too abstract: information essential for a successful implementation is overlooked and several important aspects are not explained in detail;
- they are fragile: their success depends on several conditions, often not explicitly stated, and hard to verify and meet in practice.

In other words, the general specifications of these approaches are either too distant from the implementation practice or too dependent, in an implicit way, on the specific features, which do not reproduce spontaneously in other cases, of a successful implementation experience.

It is our opinion that these limitations can be overcome adopting a methodology-based approach, i.e. by defining also for technology transfer, a disciplined framework, including a sound life cycle for this activity and providing a well-defined structure, where concrete indications about the tasks to be carried out and about conditions to be verified are specified and clearly related one another.

## 3.3 Requirements for technology transfer

Starting from the considerations drawn in the previous section, we present here a set of requirements that a sound and effective approach to technology transfer should meet.

- It must be derived from practical experience. Approaches with a sound logical background, technically correct, incorporating organizational and psychological wisdom, but only partially tested in practice do not work in real cases. The most natural way to define a technology transfer approach is bottom-up, starting from concrete cases and trying to generalize from them.
- It must be applicable to a large variety of situations. It is not enough that it proved to work in one specific situation; there must be sound evidence that it might be successfully exploited in other contexts, under different conditions, with different goals.
- It must focus only on the few fundamental conditions that can determine success or failure, without getting lost in a number of immaterial details.
- It should be robust, i.e. its application should not be compromised by a number of parameters and conditions whose smallest variation may transform a potential success into a failure.
- It must be powerful enough to face in a balanced way all the three levels of technology transfer - namely, know-how, know-what, and know-why.

- Finally, since a technology transfer initiative always entails risk, a sound approach must be specifically oriented towards risk reduction [6].

Summing up, an ideal approach to technology transfer must be practical, general, simple, robust, balanced, and controllable. KB technology transfer is like pudding: a good recipe is easy to understand and simple to put into practice, and it must work in any kitchen and with any cook. The results of applying a good recipe are not always identical: there may be a variety of good puddings. All, however, must share a set of fundamental properties. A bit more sugar or a bit less milk do not make a great difference, but five eggs instead of six may compromise the result.

## 3.4 Failure causes and success conditions

KB technology transfer is a difficult issue. It was a challenging effort several years ago, in the pioneering age of KBS application, when the risk of a KBS initiative was quite high, and it is still a critical issue today. KBSs deeply interfere with the structure and organization of the target environment. The first steps of a KBS initiative may raise problems with the more traditional data processing and information system departments. Moreover, not only must KB technology find its own space and role in an organization, but it must also find out the best way to integrate with pre-existing technologies, both within the field of information technology and outside it. Individuals may have a negative attitude towards a KB technology transfer initiative [25]; they may fear that the new technology may replace them, may be used to exploit them or deprive them of their knowledge and professional know-how, may lessen their personal power, may create a new dependence. KB technology transfer is not simply a matter of buying a new software package.

A failure in KB technology transfer may be due to several, diverse reasons. Among the major causes that generally impede the success of a technology transfer initiative we mention [16]:

- lacking management support; the management must see the benefits of the project and provide all the support necessary for the project;
- setting objectives too high; a KBS project should never face issues beyond the concrete needs of the users, or exceed the potentials of knowledge-based technology or the capabilities of the available project team;
- underestimating user involvement; a KBS must first of all serve concrete user needs, this cannot be achieved without the appropriate involvement of the users in all relevant steps of KBS life cycle;
- underestimating operational and organizational issues, thus failing to produce a usable application;
- lacking competent, available, and cooperative domain experts: without domain experts a KBS can simply not be constructed;
- lacking the proper technical personnel: a weak project team always leads to a failure;

- using inappropriate tools, thus making design more complex and development less effective, and also spoiling the quality of the final result;
- involving too many domain experts, not really necessary for the development of the KBS; this may make the knowledge acquisition process longer and more costly, and may hinder the quality of the knowledge base;
- underestimating the maintenance and extension issues, thus failing to deliver an application which can be profitably exploited over time.

The success of a KBS initiative, instead, can be ascribed, in general, to just a couple of conditions, namely: having a good methodology and avoiding fatal errors. Both these conditions are, however, not easy to meet in practice. Sound, proven approaches to KB technology transfer are not many, and several of them are only applicable to a limited variety of contexts. Moreover, serious errors are difficult to anticipate, and often even to identify. In fact, some common and well-known errors, widely analyzed in the literature, are nevertheless continuously repeated in different projects by different people. Effective management of such an initiative requires to constantly keep attention focused on recognizing the possible occurrence of fatal errors as early as possible, before their effects become harmful, in such a way as to allow implementation of corrective actions in due time.

# 4. Viable approaches

## 4.1 Facing quality and productivity: a basic perspective

Considering the issues discussed in Section 3, the fundamental question is: what should a provider of advanced software applications concretely do for approaching appropriately the quality and productivity issues of complex systems, where KB technology represents one important component of the technological mosaic? As such, the question is not calling for a complete solution of the problem; rather, it focuses on a practical, yet technically correct and sound, approach towards a pragmatic, flexible, and evolutionary solution.

Here we introduce a novel and viable approach to the above stated demand. The approach we propose is grounded on three main points, as discussed below:

- *The reference*. First of all, it must be acknowledged that, if a standard cannot guarantee quality and productivity, nevertheless a clear, formally stated, and widely shared reference is necessary. It should constitute a general and abstract foundation on which particular methodologies specifically oriented towards quality and productivity can be grounded. As such, the reference should encompass all the top-level concepts which are specific and essential to the KBS life cycle. In other terms, it should provide a clear set of requirements for the

definition of a KBS life cycle or, equivalently, the definition of a very abstract and generic life cycle concept.

- *The tailoring mechanism*. In addition, a disciplined, possibly formal, mechanism for tailoring the reference is needed. This tailoring mechanism should allow the reference life cycle to be specialized to particular enterprise and project frameworks, while substantially preserving, however, the original rationale. Thus, the tailoring mechanism is required to be adequately flexible to allow for a continuous, progressive refinement of the reference. Additionally, the refinement should not be so undisciplined as to compromise the spirit of the reference, thus nullifying its primary raison d'être. Tailoring the reference life cycle yields a specific development methodology, including methods, techniques, and tools aimed at pursuing quality and productivity.
- *The enterprise know-how*. Specialization of the reference life cycle through a suitable tailoring mechanism requires a substantial addition of knowledge. It is indeed this knowledge that can turn a generic and abstract reference into a specific, powerful methodology. The knowledge necessary to build a development methodology has to be derived from enterprise know-how, which includes methods, techniques, tools, and practices used in implementing the various life cycle processes in a specific production environment. The power of a methodology is, therefore, in the know-how it actually contains. Typically, enterprise know-how is largely implicit and informal, and encompasses a variety of empirical and case-based knowledge derived from practical experience. Eliciting, formalizing, integrating, and embedding this knowledge in a well-structured and disciplined development methodology allows the effective preservation, improvement, and exploitation of such a valuable asset.

In our view, both the reference life cycle and the tailoring mechanism should be established as either formal or de facto international standards. They should therefore be of public domain, recognized by wide and competent scientific and technical community, and available to all interested parties. Enterprise know-how, instead, is a private, valuable, and highly competitive asset of each individual producer in the field of KB technology. It will not be shared and will represent the added-value of a specific development methodology with respect to the basic reference.

## 4.2 A practical methodology for technology transfer in the KBS field: the 9-ingredients approach

As stated in Section 4, in order for a technology transfer initiative to succeed, two things are necessary:

- having a good methodology;
- putting it in practice with competence, skill, and creativity.

The latter point can not be dealt with in a paper. So, we must limit ourselves to propose a methodology, including a lot of experiences gathered through several cases (both successful and failed) of technology transfer initiatives [16] and finally synthesized after the lessons learned in a particularly successful project [2, 3]. It is based on nine ingredients - and, therefore, it is called the 9-*ingredients approach* [4] - but, of course, it is not the number of ingredients or their list that is important. Every one of us knows that milk, eggs and sugar are needed to make a pudding, but only a few of us can actually cook a good pudding. Instead, it is the quality of the ingredients and the way they are used that make the difference between success and failure. In the following the nine ingredients on which our approach is grounded are illustrated.

## (1) Checking prerequisites

Three prerequisites must be satisfied for implementing the 9-ingredients approach, namely:

- *Problem*: having a real and important problem to solve. Technology transfer can not be carried out without the motivation of a concrete, short-term exigency. Technology transfer intended as a long-range investment, pushed by a generic need towards the improvement of the current technological level of company, is very risky since it generally lacks enough management attention and support.
- *Persons*: having the right persons to employ in the technology transfer initiative. Technology transfer, even if involving large organizations, is always a person-to-person activity; the individual skills and traits of the involved persons are of crucial importance for success.
- *Project*: having an appropriate context to set-up the technology transfer initiative. Technology transfer should always be framed into an appropriate context, namely a project, that provides the necessary justification for its existence.

If the "3-P rule", namely having the problem, the persons and the project, is not satisfied, a technology transfer initiative should not be initiated.

## (2) Setting goals

The specific goals of a technology transfer initiative should be stated first. The primary goal must always be technology transfer. This seems obvious, but is not easy to assert in practice. This goal entails as a result that:

- some managers of the target company understand and appreciate the main features of KB technology;
- some employees acquire the basic elements of the technology and are specifically and practically trained in some of the specific professional roles of KB technology;
- both of them are able and willing to exploit KB technology again in further applications, possibly without the support (or with a reduced support) of a technology provider.

The secondary (but mandatory) goal is the development of a full industrial KBS prototype that solves a real, important problem and can be rapidly developed, after the conclusion of the technology transfer initiative, into a complete target system or a product. Two aspects are vital for this goal: the problem faced must be real and important, the result must be nothing less than a full industrial prototype. Later, other additional goals may be added, that vary from case to case. These should not conflict with the primary or secondary goals, and their achievement should be regarded only as a complement to the main results expected from technology transfer.

### (3) Defining results

Once project goals have been stated, the expected results should be carefully defined. The most visible and concrete result will always be the industrial prototype. This must show at least the following properties:

- being complete and fully operational (with respect to the stated specifications);
- being usable and useful (for a defined class of users);
- being well documented (according to a stated standard);
- incorporating a significant knowledge asset (about a specific application domain).

In addition to the prototype, other results must be achieved, namely:

- the improvement of the professional skill of all the participants to the project (also including senior managers and domain experts);
- the creation of a competent and effective project team;
- the awareness of the management of the potentials and limits of KB technology and the availability to consider new application opportunities;
- the dissemination of knowledge about KB technology among potential users inside the company.

### (4) Stating evaluation criteria

The results stated above should be objectively evaluated and measured. The evaluation criteria and the methods to apply them must be stated before the project start. At best, evaluation should be assigned to an independent authority, not involved in the development of the project, and should not be carried out at specific milestones (not postponed to the end of the project).

### (5) Defining professional roles

The professional roles of the participants to the technology transfer project must be precisely defined. At least the following roles should always be considered [16]:

- *Project leader*, in charge of project definition, management and supervision; he or she leads the project team during all life-cycle phases, and is the sole responsible for the results of the project. The project leader must have a sound and specific technical background and deep and extensive experience in KBS project

management. He or she must be available to devote enough time and attention to the project to ensure effective management.

- *Knowledge engineer*, in charge of knowledge analysis, modeling and acquisition, and also of some aspects of system design and applied research.
- *Designer*, in charge of system design, of some aspects of applied research, and also of providing the necessary support to the developer.
- *Developers*, in charge of developing the empty system, the interfaces and all the software components of the KBS.
- *Domain experts*, providing the fundamental knowledge sources for the development of the KBS application.

## (6) Defining the team

The project team for a technology transfer project must be slim. One person for each professional role may be enough. Less than that may be insufficient and seriously compromises the result. The involvement of all members of the project team should be full time (except for the domain experts, who are generally involved only for a partial, even if substantial, percentage of their time). Sometimes, two knowledge engineers may be involved, a senior and a novice. Often, two developers may work well together. In several cases, more than one domain expert is necessary; one should however refrain from involving too many experts who can substantially increase the complexity of the project and the involved risk.

The project team should be structured rather hierarchically, in order to establish a disciplined work practice.

The project team should also include as external observers a couple of senior managers of the company which is the target of the technology transfer initiative. These should be regularly kept up to date with the progress of the project and should be concretely involved in some significant steps of the work plan. The involvement of senior managers is of primary importance in achieving correct and effective technology transfer.

Finally, also the intended users of the KBS application should be included in the project team. However, in a technology transfer project, their participation should be as far as possible focused. For example, it could be limited to the tasks of requirement analysis and validation, while for all other tasks the involved domain experts should be able to represent also the point of view of the users. An extensive involvement of the users may greatly increase the complexity of the project and make the risk level higher.

## (7) Project planning

Project planning does not allow for too many variations. The following life-cycle scheme may apply in the vast majority of cases (for each task, *dur*: denotes the duration in months - *par*: denotes that the task is supposed to run in parallel to other tasks).

0. basic training of project team members

phase A: *Background*

1. knowledge analysis and conceptual modeling, *dur*: 3;
2. definition of KBS specifications, *dur*: 1;
3. literature survey and analysis of potential technical approaches, *dur*: 4, *par*: 1-2;

phase B: *Design*

4. logical modeling and detailed technical design, *dur*: 4;
5. tool selection and acquisition, *dur*: 1, *par*: 4;
6. training on the tool, *dur*: 2, *par*: 4;

phase C: *Development*

7. implementation of the empty system, *dur*: 6;
8. knowledge acquisition and development of the knowledge base, *dur*: 8, *par*: 7;
9. experimentation and testing, *dur*: 2, *par*: 8;
10. applied research on key technical issues, *dur*: 12, *par*: 4-9;

phase D: *Improvement*

11. extension and refinement, *dur*: 8;
12. validation, *dur*: 4, *par*: 11.

Life-cycle tasks may be carried out sequentially (apart from the stated parallelisms), allowing however for extended experimentation and backtracking whenever appropriate. Discipline must however prevail on free exploration.

## (8) Implementing controls

Project control is a key issue in a technology transfer initiative. Implementing controls is the sole responsibility of the project leader. Controls should be frequent, rather formal, but not bureaucratic. They should concretely support the persons involved in the project, increase their self-confidence, enhance cooperation. Controls should focus at least on the following three issues: the progress of the project (with respect to the stated plan), the level of technology transfer, the quality of the results. They should allow the project leader to identify dangerous trends or harmful deviations early (before they become irreparable) and to define appropriate corrective actions.

## (9) Managing risks

Clearly, there is an enormous variety of risks that may compromise a technology transfer project. The most serious ones always concern persons rather than tools, methods, or techniques. Among the most frequent and harmful risks we mention:

- *Overlapping between professional roles*: if a limited flexibility among roles should always be allowed, confusion and interference can rapidly cause a project team to get stuck.

- *Separation between professional roles*: the precise definition of professional roles must not exclude a limited task sharing and should never bring to a rigid and bureaucratic attitude in interpersonal communication. Roles should not spoil cooperation.
- *Physical distribution*: the project team should be allocated in a couple of adjacent rooms. Distributing the project team in different places may dramatically affect communication and cooperation.
- *Replacing goals*: commercial or production issues that are often intermixed with a technology transfer initiative, should not prevail on the very purpose of the project, namely technology transfer.
- *Diverging goals*: if project goals have not been stated precisely and correctly, the project may fluctuate between different, contrasting goals, thus definitely spoiling the final results.
- *Critical resources*: while some members of the project team may be easily replaced, if necessary, during the development of the project, others must absolutely be guaranteed for the entire project duration. Unfortunately, in a technology transfer project all persons involved are generally critical.
- *Lack of leadership*: the project leader must actually conduct the project, keeping constantly and concretely connected to the project team. He or she must be personally and substantially involved. He or she should not launch the project and wait for the results, nor limit himself or herself to formal, scheduled check points.
- *Developing a negative attitude towards the technology*: a technology transfer initiative should not be a slaughtering race. Engagement in the project must be serious but not too hard, work load must not be excessive, overall project organization must be rigorous but light. At the end, should not only the expected results have been produced, but all the participants to the project should also be willing to work with KB technology again.

# 5. Conclusions and perspectives

From the issues discussed in the paper it clearly comes out that the approach we advocate for quality, productivity, and technology transfer is methodological. Ideally, it is possible to envisage an integrated methodological framework which incorporates both the development of KBSs and the transfer of the relevant technology as a whole. In this perspective, not only is the methodology a means for building a KB application, from requirements analysis to testing and maintenance. It is also appropriate to transfer the set of methods, techniques, and tools (including the methodology itself) to an organization which is going to develop KB applications. Eventually, the methodology can even support structured accumulation and sharing of enterprise know-how about KBSs design and construction, thus providing a platform for a continuous growth.

Within this integrated view, the first phase the methodology is expected to drive is precisely technology transfer. Roughly speaking, technology transfer is a kind of methodological bootstrapping which enables the set up of the preconditions necessary for the subsequent KB application development activities. As a component of the methodology, however, the technology transfer phase is subject to tailoring and refinement as well. Therefore, technology transfer should no longer be viewed as a crystallized process. Rather, starting from a generic, abstract technology transfer concept, it should be possible to derive, through a progressive refinement mechanism, a concrete, specific, organization-adapted technology transfer process.

Definitely, however, the success of the overall approach crucially depends on the tailoring mechanism. We claim that an object-oriented paradigm [8, 26] can provide an excellent conceptual framework to support a correct and effective implementation of the tailoring mechanism. In this perspective, starting from a very general and abstract reference life cycle, it is possible to define a hierarchy of methodological classes, where each class is meaningful and useful for a defined class of tasks and users [19]. Such a *methodological hierarchy* is expected to support the typical object-oriented notions of instantiation, refinement, and part-of, which allow a correct and clear implementation of the informal notion of tailoring introduced above.

The development of a complete object-oriented approach to definition of development methodologies is beyond the scope of this work. To substantiate our point of view, we introduce below some basic concepts of what might become an object-oriented approach to the definition of KBS development methodologies.

First of all, a top-level *methodology class* has to be defined as an abstract and generic specification of the concept of methodology. The methodology class characterizes the common, fundamental features of a generic methodology in terms of *attributes*, *methods*, and *constraints*. In other words, the methodology class may be viewed as a virtual container of any possible methodology that can be actually defined in the proposed approach. For example, a high-level standard for KBS life cycle may be modeled by means of a methodology class. Obviously, the methodology class is required to be instantiated before being actually used.

An *instantiation* of the methodology class is a *methodology*, and occurs by assigning appropriate values to a set of relevant attributes. Note that the concept of instantiation allows for parametric methodologies. In a parametric methodology, a number of parameters may be set, so as to allow for customization of the methodology for specific contexts. Parameters may represent, for example, the complexity of the project, the skill and experience of the project team, the budget, the available resources, and so on.

In addition to instantiation, a methodology class may be refined. A *refinement* of a methodology class $MC_1$ is the specification of a new methodology class $MC_2$ on the basis of $MC_1$. $MC_2$ is called a *methodology subclass* of $MC_1$. Refinement allows for addition of new attributes and methods, which are meant to express respectively the specific static and dynamic properties of $MC_2$. Intuitively, the refinement allows $MC_2$ to focus on a more specific context. For example, given a standard methodology class Standard-M, a refinement SW-Critical-M might be a methodology subclass of Standard-M which tailors the standard for safety-critical software, KBS-M another

methodology subclass which refines the standard for KBS software components, and Safe-KBS-M still another methodology subclass of both SW-Critical-M and KBS-M. We denote the refinement relationship with the symbol *refines*. So, $MC_2$ *refines* $MC_1$ means that $MC_2$ is a refinement of $MC_1$. As such, *refines* corresponds to the *inheritance* relationship of the object-oriented paradigm. We may then express the above relationships as follows: SW-Critical-M *refines* Standard-M, KBS-M *refines* Standard-M, Safe-KBS-M *refines* SW-Critical-M *and* KBS-M.

Observe that the latter is a multiple refinement (inheritance) relationship, as Safe-KBS-M is a methodology subclass of both SW-Critical-M and KBS-M. The refinement relationship gives rise to a *methodological hierarchy*, which should not be confused, however, with the notion of methodological tree introduced in Section 2.4. Indeed, the former specifies the refinement relationship among methodological classes, while the latter simply represents how a specific methodology (that is, an instance of a methodology class) is internally structured by means of sub-components.

The *methods* of a generic methodology class (or subclass) represent the dynamic part of the class specification, and prescribe what actions to perform, under which conditions, and in which specific way. Methods can be expressed by means of an appropriate procedural language with a specific syntax and semantics.

Finally, the *constraints* are logical predicates defined on attributes and methods. They are meant to safeguard the internal consistency of the a methodology class (or subclass), especially across the instantiation and refinement operations. Constraints may be expressed either through a simple first-order predicate, an higher-order formula, or even a formula involving temporal quantification.

# References


[1] J. Bachant and J. McDermott, R1 revisited: Four years in the trenches, *AI Magazine* 5(3), 1984, 21-32.

[2] P. Baroni, F. Cremonesi, G. Guida, S. Mussi, and S. Yakov, ASTRA: a knowledge based system for state assessment, preventive diagnosis, and intervention planning of power transformers, *Proc. ISAP 94 Int. Conf. on Intelligent System Application to Power System*, Montpellier, F, 1994.

[3] P. Baroni, G. Guida, and S. Mussi, Distributed and Uncertain Reasoning in a Knowledge-Based System for Preventive Diagnosis of Power Transformers, *Proceedings 1995 IEEE International Conference on Systems, Man, and Cybernetics*, Vancouver, BC, 1995.

[4] P. Baroni, G. Guida, and S. Mussi, A case study of technology transfer in the field of knowledge-based technology: the 9-ingredients approach, *Proceedings 1995 IEEE International Conference on Systems, Man, and Cybernetics*, Vancouver, BC, 1995, 3210-3215.

[5] V.R. Basili and J.D. Musa, The future of engineering software: A management perspective, *Computer* 24(9), IEEE Computer Society, 1991, 90-96.

[6] B.W. Boehm, Software risk management - Principles and practices, *IEEE Software* 8(1), 1991, 32-41.

[7] K.R. Bonnett, From grassroots to grand plan, *AI Expert* 4(8), 1989, 54-59.

[8] G. Booch, *Object-Oriented Analysis and Design*, Second Edition, Benjamin-Cummings, Redwood City, CA, 1994.

[9] D.N. Chorafas, *Knowledge Engineering*, Van Nostrand Reinhold, New York, NY, 1990.

[10] J. Cupello and D. Mishelevich, Managing prototype knowledge/expert system projects, *Communications of the ACM* 31(5), 534-541, 550.

[11] J.S. Edwards, *Building Knowledge-Based Systems - Towards a Methodology*, Pitman, London, UK, 1991.

[12] E. Feigenbaum, P. McCorduck, and H.P. Nii, *The Rise of the Expert Company*, Macmillan, London, UK, 1988.

[13] M. Freiling, J.H. Alexander, S.L. Messick, S. Rehfuss, and S.J. Shulman, Starting a knowledge engineering project: A step-by-step approach, *AI Magazine* 6(3), 1985, 150-164.

[14] M. Greenwell, *Knowledge Engineering for Expert Systems*, Ellis Horwood, Chichester, UK, 1988.

[15] G. Guida and G. Lamperti, Quality and productivity in the development of knowledge-based software components: methodological issues, *Atti Giornata di Studio "I Sistemi Basati sulla Conoscenza nei Processi Produttivi"*, ANIPLA, Milano, I, 1996, 1-14.

[16] G. Guida and C. Tasso, *Design and Development of Knowledge-Based Systems: From Life Cycle to Methodology*, John Wiley & Sons, Chichester, UK, 1994.

[17] P. Harmon and B. Sawyer, *Creating Expert Systems for Business and Industry*, John Wiley & Sons, New York, 1990.

[18] T. Helton, AI infusion: Getting your company involved, *AI Expert* 5(3), 1990, 54-59.

[19] W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, MA, 1990.

[20] ISO, International Organization for Standardization, *Quality Management and Quality Assurance - Vocabulary*, ISO, Geneva, Switzerland, 1995.

[21] ISO/IEC, International Organization for Standardization, International Electrotechnical Commission, *Information Technology - Software Life Cycle Process*, ISO/IEC Geneva, Switzerland, 1995.

[22] ISO, International Organization for Standardization, International Electrotechnical Commission, *Information Technology - Software Product Evaluation - Quality Characteristics and Guideline for their Use*, ISO/IEC, Geneva, Switzerland, 1991.

[23] P. Jackson, *Introduction to Expert Systems* (2nd edition), Addison-Wesley, Reading, MA, 1990.

[24] L. Lee, *The Day the Phones Stopped: How People Get Hurt When Computers Go Wrong*, Primus, Donald & Fine, New York, NY, 1992.

[25] J. Liebowitz, *Institutionalizing Expert Systems - A Handbook for Managers*, Prentice-Hall, Englewood Cliff, NJ, 1991.

[26] B. Mayer, *Object-Oriented Software Construction*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[27] T. McCullough, Six steps to selling AI, *AI Expert* 2(12), 1987, 55-60.

[28] J. McDermott, R1 - The formative years, *AI Magazine* 2(2), 1981, 21-29.

[29] Parnas, Why engineers should not use artificial intelligence, *INFOR* 26(4), 1988.

[30] T. Peters and R. Waterman, *In Search of Excellence*, Harper and Row, New York, NY, 1982.

[31] S.L. Pfleeger, N. Fenton, and S. Page, Evaluating software engineering standards, *Computer* 27(9), IEEE Computer Society, 1994, 71-79.

[32] D.S. Prerau, *Developing and Managing Expert Systems - Proven Techniques for Business and Industry*, Addison-Wesley, Reading, MA, 1990.

[33] N.F. Schneidewind and N. Fenton, Do standards improve quality?, *IEEE Software* 13(1), 1996, 22-24.

[34] D.A. Waterman, *A Guide to Expert Systems*, Addison-Wesley, Reading, MA, 1986, 234-246.