International Journal of u- and e- Service, Science and Technology
Vol. 2, No. 2, June 2009

Preliminary Remarks Regarding the Impact of the Modeling Languages towards the Quality Standards in the Software Industry

Dorin Bocu and Răzvan Bocu

Transilvania University of Braşov
d.bocu@unitbv.ro, razvan.bocu@unitbv.ro

Abstract

Both the official statistics and the dedicated literature acknowledge a problematic state-of-the-art for the software industry: the elevation of the quality standards is a problem that preoccupies both the industry and the academic environment, the results being obtained are still insufficient, and considering the percentage of the projects that fail at different stages is worrying large. The lack of quality or the unsatisfactory quality are definite reasons for a failure, at once or delayed for a while, of the IT projects. In this paper, we try to identify an optimal work frame, in which the modeling languages may bring an important contribution to the quality assurance in the software industry.

Keywords: *software quality, modeling languages, methodology, agility.*

1. Quality – a difficult-to-solve problem in the software industry

The quality is one of the essential levers for assuring the long-term successful market for the products, in any kind of industry. Just because it is an action lever with beneficial outcomes in the long term, the quality can be obtained only following a structured approach, initiated by the management and implemented at the operational level. The structuralism of the approach refers to the ensemble of the components that condition or produce the quality (the extensive side of the endeavour), as well as the dynamics of relations between components (the intensive side of the endeavour).

The viability of the systems, in general, is based on two contradictory features: the capacity of the components to deploy activities featured by a relative autonomy and the capacity of the ensemble of components to evolve according to some general principles of the whole community. The tensions and disharmonies that may appear in a system as a consequence of these features require the careful intervention of the management, having the goal to direct and fluidify the energies that are consumed in a natural manner, as a direct consequence of the intrinsic laws that coordinate the system and, also, as a direct consequence of the requests arisen at the community level. Actually, the problem of the optimization of the systems is a problem of a continuous approach of man to higher and higher levels of knowledge and reality modeling.

Most of the times, the idea of optimizing is associated to the idea of algorithm or procedures with algorithmic potentials, based on which one can bring a certain improvement to a system. The question that we try to deal with in this paper is: "Quality assurance in the software industry is an approach which can be algorithmised?". The answer that we'll provide is neither exhaustive nor definitive. Moreover, we'll bring into light the problematic nature of the approaches that try to turn the quality into a reliable ally in the software industry.

2. The general frame for assuring the quality in the software industry

If we were to discuss about the general background necessary to assure the quality in the software industry, it is mandatory to emphasize the fact that, in practice, the methodological approach was imposed and validated. It is beyond our scope to discuss about methodological nuances. Nevertheless, it is important to remark that the battle carried by the software companies against those who oppose, in a way or another, to producing quality software systems, is carried on according to perpetually moving rules.

In fact, according to the statistics realized by specialized institutions, change has become the most important enemy in the software industry. Only a while ago, the main issue in the software industry was represented by the need to make the complexity of the modeled system manageable. Presently, we can state that at the continuously growing complexity of the modeled system, a new important challenge is adding itself: the structural dynamics of these systems. In these conditions, the life cycle of a product that is realized by a software company is prone to become obsolete quickly. As a consequence, developers must find out means for shortening the time needed to realize software systems and, also, to attenuate the inherent effects of changes. Consequently, the general framework in which the problem of assuring the quality in the software industry is raised, nowadays, can be synthesized in Figure 1.

Studying Figure 1, we can deduce that the imperative of shortening the development cycle in the software industry, as a mean of increasing the resistance of the systems to changes, involve the utilization of some adequate tools. With their help, we can annihilate, at least partially, the effects of changes, truly ravishing for the software industry.

These tools are:

- Strategies and abstracting / modeling / representation techniques that are featured by flexibility and extensibility;
- Supporting technologies for template-acquisition oriented development. Using their help, the very good results that are obtained by developers during their abstracting / modeling / development of some problems' solutions, can be made available for any interested developers.

Considering the context, which is drafted in Figure 1, the triad Developer / Manager / Customer operates, with the declared goal to efficiently implement the requirements suggested in Figure 1.

The message that we try to transmit to the reader in Figure 2 has the following components:

- Any success project in IT is based on the three prototypical categories of actors: Customer, Developer and Manager;
- The customer is the actor that initiates the project and motivates it financially and, also, benefits from it;

- The developer is the actor that effects activities like these ones: conceptualization, specification, design and implementation of the customer's requirements, having a few main targets: the solution, the life span of the solution, the quality of the solution, the cost of the solution;
- The manager is the actor that is responsible for the efficiency of the activities conducted instances that belong to the three prototypical types of actors;
- The developers and the managers are the consumers of support technologies, which can be adapted to new requirements. They are also the promoters of some projects that aim to realize novel support technologies. This way, the spiral of the progress concerning the quality is bound to have a positive dynamics.

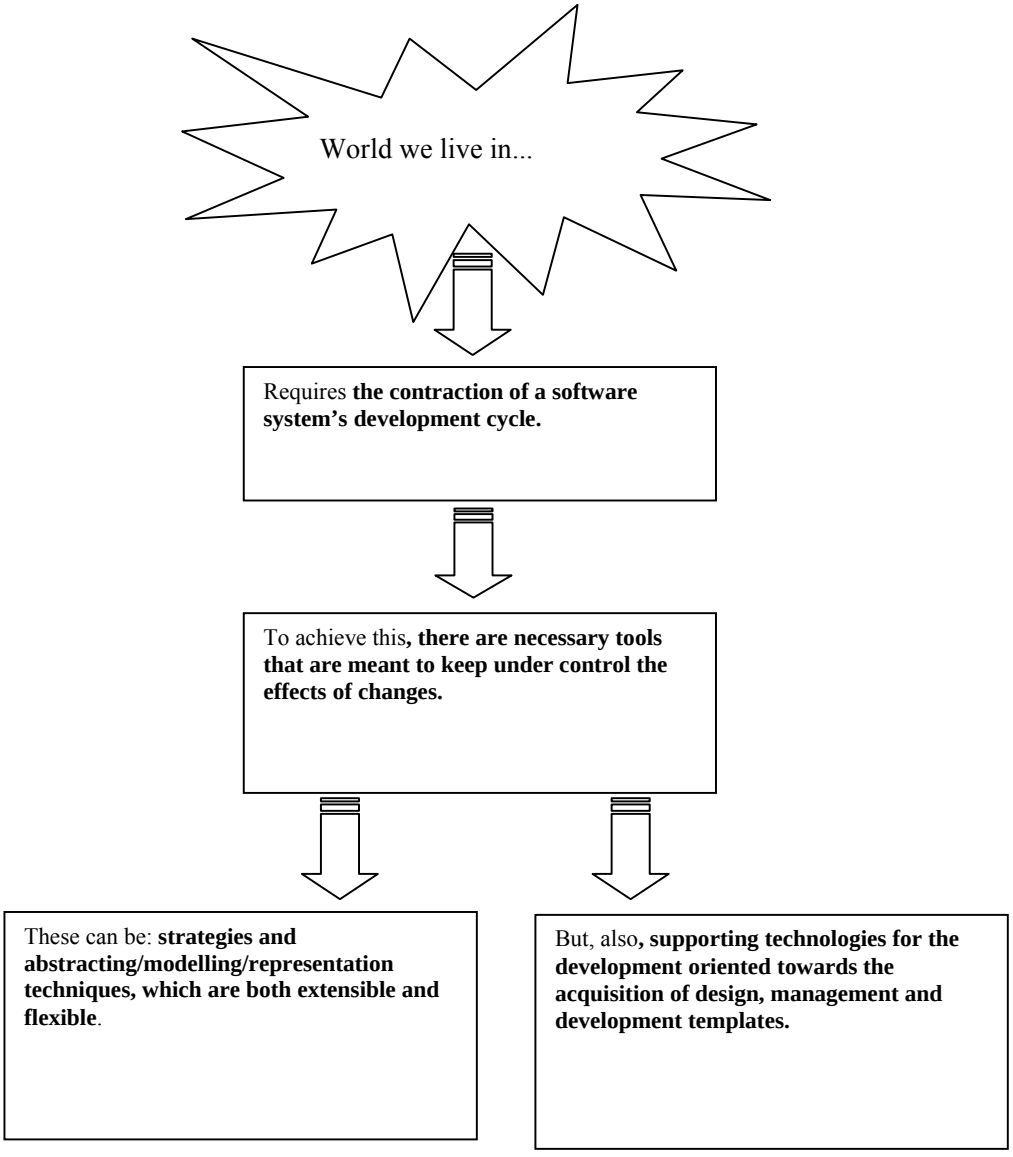

Figure 1. The present context in which we speak about assuring the quality in the software industry

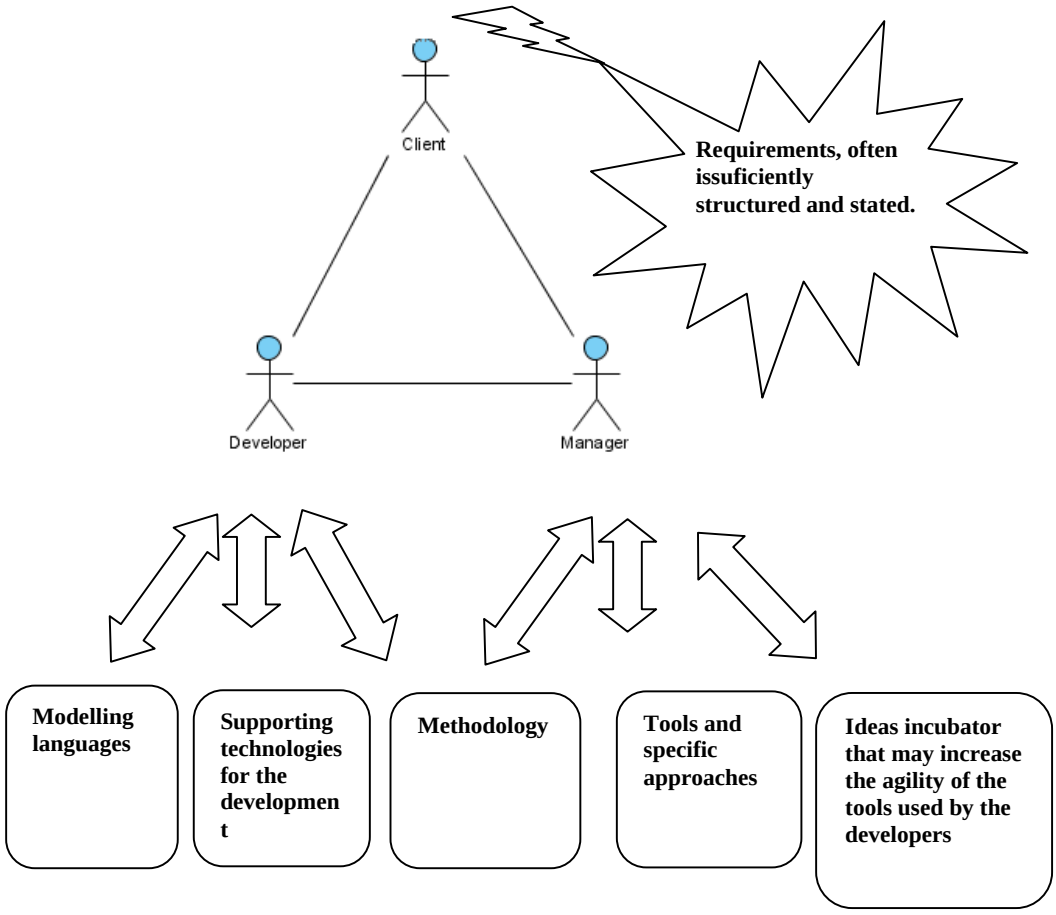

Figure 2. The fundamental triad for the success of a project in the software industry.

3. The impact of the modeling languages towards the software quality

3.1.Introduction

In paper [1], we presented a certain approach on the concept of quality in the software industry, as it is presented and understood in the dedicated literature. Obviously, the controversies on the concept of quality are still alive. The researcher's consensus on the concept of quality is missing as well as the discussion basis, which is permanently changing.

In this paper, we study, as we stated in the abstract, the impact of the modelling languages on the software quality.

The investigation and the comparison between the modeling languages used during the last decades in the software industry is, undoubtedly, an extensive and interesting research subject.

This paper, which has the goal to awake the interest for such a research topic, will try to highlight the following aspects:

- The utility of a modeling language;
- The specificity of the modeling languages in the software industry.;
- The relation routine-creativity in a development environment centered towards the systematic approach.

Based on what we stated above, we will conduct a series of exploratory considerations regarding the answer to the question: Focusing our attention towards the modeling language, the quality assurance in the software industry can be designed in an algorithmic manner?

3.2. The utility of a modeling language

Modeling languages are, both in the academic world and in the industry, more and more theoretical and practical instruments that help the researcher or the specialist in a certain field to approximate the solution of a problem, represent the solution of a problem and create the foundations for automatically generating the solution. The modeling activity comes, normally, before the effective implementation of a product that has the features required by the solution of a problem. Considering they are, by excellence, approximation tools for the complexity of the modeled systems, modeling languages wear twice as much the seal of imperfection:

1. The logic of their usage is based on protocols in which ambiguities are means of an efficient exploration of the extensive space of the alternative solutions;
2. Their structure is perishable in the relation with the semantics of the modeled systems, themselves featured by a continuous dynamics.

As a consequence, we have two key reasons that determine us to state that modeling languages approximate the complexity of the modeled systems. In theory, considering the way modeling activity goes on these days, in the short term we can speak about different degrees and manners for approximating the modeled systems. Also, in the long run, it is very encouraging to acknowledge a permanent competition that aims to optimize the approximation strategies.

At the foundation of a coherent strategy promoted by a modeling strategy, there is always a certain kind of abstractization.

As a consequence, the present paper and those that will follow, we insist on the tandem approximation-abstractization, as being the theoretical construct that can quantify both the strengths and the weaknesses of a modeling language.

Generally speaking, one of the universal functions of the languages is their availability to allow the storage of the creative experience of the intelligent systems. The presentation of goals that determine the creation of this knowledge treasure go beyond the scope of this paper but, the utility of knowledge storage for the progress of the science and of the means that allow us to relate to various forms of practical results of knowledge is out of question.

In order to assume, with practical results, the knowledge storage function, must be conceived to allow the compact, non-ambiguous and adaptive representation to new requirements of a solution of a problem.

Finally, an important reason(that is not often spoken about in the dedicated literature) for modeling, whatever the research field is, is the need to automatically generate the solution of a problem.

As soon as the solution is identified and properly represented, provided it is necessary, one can talk about the optimum condition for a large-scale industrial utilization of the respective solution.

3.3. The specificity of the modeling languages in the software industry

Without expecting all the readers to agree with us, we consider to be very important for the specificity of the modeling languages in the software industry the following:

- The semantics, which is the object of the modeling process;
- The syntax that is used during the modeling process.
- The modeling strategy being used.

The semantics: designates sets of data together with the operations they support.

The eternal reason that generates talks about semantics is represented by the tensed relation between data and the operations that can be applied to them.

Each of the two worlds aspires to conquer the position of information supplier, mandatory for organizing the other one. The relative agreement established between the two worlds, which is claimed by the object orientation, is already affected by the growing number of signals that say object orientation is just one of the multiple paths we can choose in order to develop software systems. Syntagms like “aspect orientation”, “component orientation”, “service orientation” etc,, are clear proofs of the insufficiency that affects the object orientation, insufficiency that is determined by the explosive diversification of the software systems, following the appearance of a new semantic contexts, difficult to model in relation to the old approaches.

This tensed relation between data and operations is the main cause for the perishability of the modeling languages in the software industry, more precisely, of the syntax that we use at a given moment in time to model the solution of a problem in the software industry.

The syntax: designates a set of concepts and principles that tailor their utilization, together with the required notation, specified having the goal to facilitate a certain kind of modeling but, also, with the goal to facilitate communication between the partners of a project in the software industry.

Considering change as one of the unchallenged achievements of the semantics in relation with which the syntax of a modeling language is, the longevity of the syntax proposed by a certain language depends on its agility, in other words on its capacity to adapt to new realities.

Some other reasons occasioned by the relation between software industry and the syntax of the modeling languages is related to the recipe used during the specification (the level of formalism, the readability of the models, the possibility to realize applications that turn modeling into an automated process, both as representation and as a generation process for some important components of the final solution. The syntax is tributary to a certain way of

understanding modeling and, as a consequence, it is a matter of taste. Thus, the break of the monopoly in relation to modeling, practiced with a natural adversity for decades, sees itself replaced progressively by the restoration of the monopoly, this time as a consequence of some practical reasons: globalization and the perpetual desire to make profit, needs a common language to solve the communication issues between developers, customers and managers all over the world.

The groups organized around an unrecognized as a standard syntax have their development possibilities restrained, as a consequence of the lack of communication efficiency, in the absence of a generally accepted standard.

The modeling strategy: a modeling language is, essentially, a sum of pledges, that may make us feel enthusiastic, or may mislead or provoke but, by themselves, these pledges do not solve, by no means, a modeling problem. In other words, having the tools available, how is one able to use them? In fact, there are more questions: what, how, when and, even, where?

Especially the projects featured by an important complexity require a special attention regarding the way in which the modeling language is used by an adequately-prepared and motivated human resource to make an IT project be successful.

The human resource but, also, the modeling languages, are two important examples of some systems featured by a natural entropy, if we want to leave an important expression space for the free will, in the long run.

In the short and medium run, the project management needs to make use of modeling strategies that value at its maximum the potential, which is offered by the human resource and by the modeling languages, methodically barring the entropic negative tendencies of these two types of resources.

It is difficult to say which of the above-discussed elements is the most important. Rather, it is correct to say the three elements constitute a system featured by a consistent dynamics, which still keep the failure rate in the software industry at a high level. Thus, it is natural to ask: “What can we do in order to reduce the number of mistakes throughout the development cycle of an IT project?”

Researchers passionately try to find an as appropriate as possible answer to this question.

Also, the important producers invest a lot of resources for finding novel training pathways towards making the process of obtaining the quality in the software industry be automatic. The target of these efforts is: the transfer to the computer of all the activities that can be made automatic, the human has to deal only with the authentic creation activities.

3.4. The relation between routine and creativity in a development environment centered towards systematic modeling

Provided the automation of the development of the software systems will go on according to the same trend, we can raise the following question: human capacity to create will still have enough space to express itself?

Not only the answer is yes, but it is important to emphasize that without stimulating the creativity of the developers, the software industry may enter into a deadlock state.

The stimulation and the organizing process of the creative activities inside IT companies must reach notable expression and productivity levels, if one wish to avoid the slow down process of the positive evolutions in the field of the IT technologies.

The dilemma that the management that understands the important role of creativity has to face is simple: while the activities based on routine can be completely designed in an algorithmic way, the activities that are based on creative contributions are difficult to plan, organize and coordinate. While the departures from the algorithmic line in a routine-based activity are generated, most of the times, by accidental weaknesses of the human resource, the creative activities, by excellence, have a dynamics that is mainly generated by the intrinsic entropy. They have the role of a positive feedback channel, which usually allows the issues that the development process of an IT project may encounter. Depending on the assumed goals, IT projects are more or less based on the creativity of the human resource.

Considering, for example, the clearly expressed goal by the triad in Figure 2 as the implementation of a system that satisfies some immediate requirements, then the routine is the main ingredient of the development process. If the triad in Figure 2 has the goal to implement a software system that can adapt itself to new requirements then, a proper routine development process may take into consideration a development scheme in which, for example, the following artifacts are useful:

- The model that involves immediate requirements of the customers related to the software system;
- The model that involves the long-term requirements of the customers related to the software system;
- Based on these two models, the model of the solution is created;
- It is easily to predict that in the case of some very complex projects, the management process himself may need a re-modeling that can contribute to its optimization.

Moreover, if the routine activity generates in the long run tiredness, boredom, which are important disruptive factors for the energy and motivation inside any generic project, then the creative activities are, at the same time, consumers and generators of energy and motivation.

The binomial routine-creativity is the main challenge for the management of a certain business that wants to be successful in a certain competitive context.

As if they were especially designed to allow the creativity burst of the developers, modeling languages require the managers to properly administer the way these modeling languages are used. Therefore, the unity of the contraries exhibits itself in the dynamics of learning and using with maximal efficiency a modeling language.

3.5. Is possible to turn quality assurance into an algorithmic process in the software industry?

An answer that may be able to cover all the perspectives suggested by this question is beyond the scope of this paper. In theory, a practical methodology does nothing else but proposes ”algorithms”, “networks”, ”schemes” for the software systems development, that values quality software system implementations as the guiding thread.

A quality product, provided it is promoted in an intelligent manner, essentially contributes to the survival and to the development of the system that realized the product. The financial resource comes back quicker to the one who invests it for creating quality products.

The software industry cannot be an exception, if we speak about the activities that financially sustain software companies. But, if we are preoccupied about the structure of the system that produces and assures the quality in the software industry, the problems become more complicated. It is known from the practical experience and from the dedicated literature that the creation of a software system is possible by maintaining functional at an appropriate level the following two activities:

- software systems engineering;
- The management of the software systems engineering.

The link between these two activities cannot be subject to improvisation without putting at risk the evolution of the projects.

Which of the activities is more important : engineering or management ? Obviously, they are both important, with the meaning that one without the other one implies a guaranteed failure of the project. As we stated before, this paper does not answer all the questions. Nevertheless, we shall focus our attention on a set of activities with technical content that influence the quality assurance in the software industry. The activity scopes in Figure 3 should be studied for a better understanding of the matter. The study of the possibility, even of the opportunity to make the quality assurance an algorithmic process may begin with the analysis of the content of these activities. The goal of this analysis should be the discovery of some strong enough invariants that are able to put into scheme the usage of modeling languages for ensuring these activities go on. This is because the lack of schematization is a constant source of entropy both for the specification and support activities.

The opinion of the authors of this paper, not yet founded theoretically and in practice, is that the abstractization techniques can be designed in an algorithmic manner in the software industry.

A few elements that support this opinion are the following:

- During the time, there existed numerous modeling languages, whose study may suggest the progressive approach to a theoretical and practical framework, which is suitable for designing the IT activity in an algorithmic manner;
- The performance increase of the computer-assisted development tools, increase expressed both by the accuracy and by the support, which is offered during the development, not only during the modeling, which is the main concern;
- The consolidation of the development patterns knowledge base, a big step forward in the modeling activity that systematically strives to value the qualitative and quantitative accumulations.
- The permanent work of the industry and the academic environment to increase the agility of the development models and, using a rebinding process, the agility of the modelling languages usage.

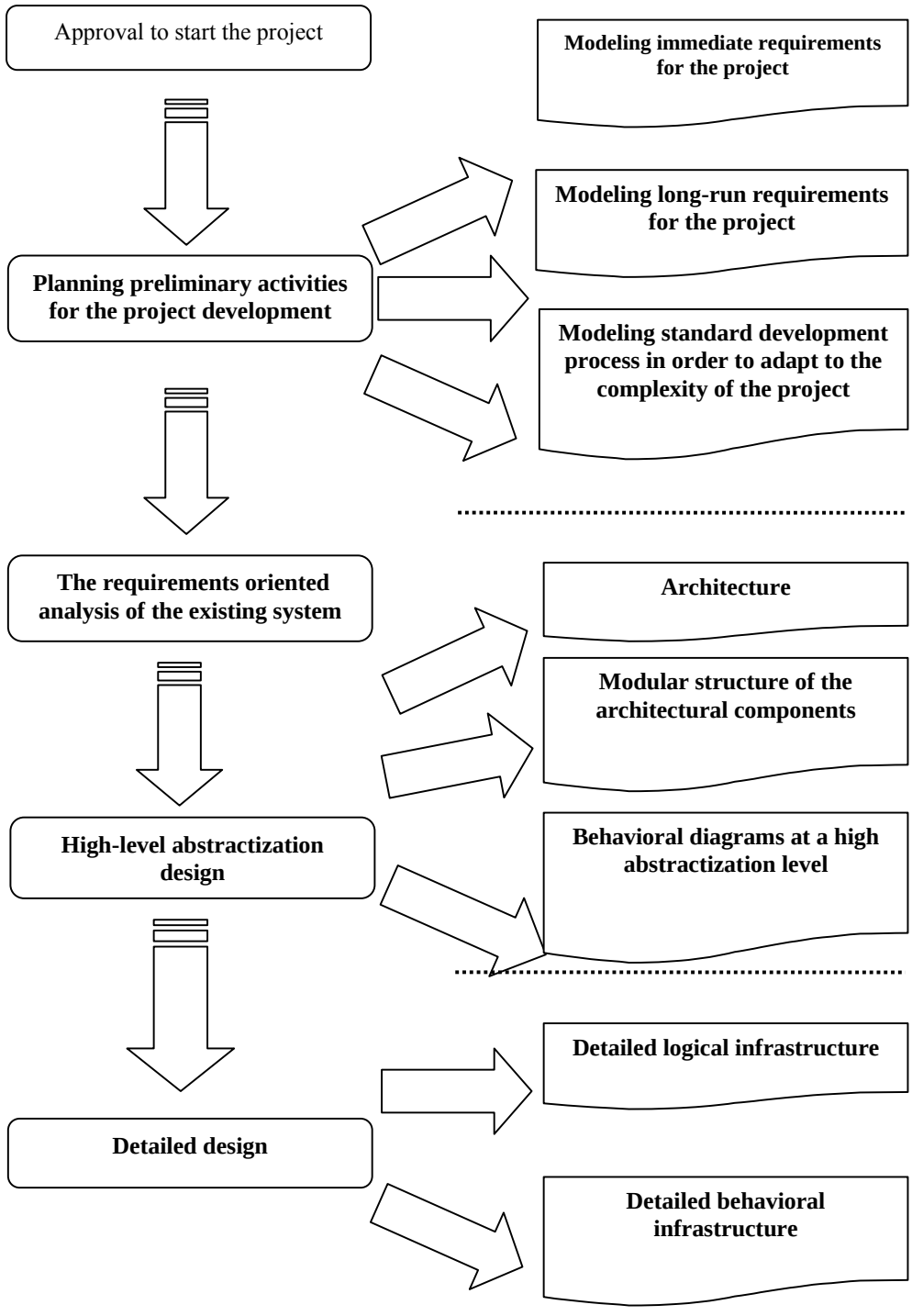

Figure 3. Activity scopes that may benefit from modeling languages usage

The human resource still remains a very problematic variable in the effort of turning the modelling activity into an algorithmic one. This is because the outstanding importance of the modelling activities, unfortunately for the managers in the software industry, is slowly assimilated and inappropriately in relation to the real conditions, in the teaching institutions at all the levels but, also, in the production where it was not found an optimal solution yet for the more general problem of instructing the human resource in order to use the new technologies.

4. Conclusions

The binomial modeling – quality will be studied in a future paper, starting from the activity scopes presented in Figure 3. The goal of this study can be synthesized as below.

It can be reached a high degree of algorithmization of the modeling activities in the software industry by:

- Simplifying the base syntax of modeling languages;
- Adding extension mechanisms to modeling languages that strongly support the acquisition of modeling patterns;
- Elaborating some novel paradigms and scenarios for gaining the ability to use modeling languages ; centering these paradigms and scenarios towards stimulating creativity;
- Redefining the content of the activities that are complementary to the modeling effort.

References

[1] Bocu, D., Bocu, R., "Modelarea obiect orientata cu UML. Fundamentele modelarii cu UML. Initiere in sabloane de proiectare utilizand sintaxa UML", Editura Albastra, Cluj-Napoca, 2006.
[2] Bocu, D., "About the Abstraction of a Software Systems Solution", Proceedings of the 25th International Conference on INFORMATION TEHNOLOGY INTERTFACES, June 16-19, 2003, Croatia, pp.621-625.
[3] Bocu, D., Neagu, M., "On the structural stabilization of a software system solution", The Proceedings of the 6th biennial International Economic Symposium, Vol. II, May, 20-21, 2006, Brasov, Romania, pp. 70-75.
[4] Bocu, D., Bocu, R., "The Dissolution of the Informational barriers in the Academic Management", The Proceedings of the 6th biennial International Economic Symposium, Vol. I, May, 20-21, 2006, Brasov, Romania.
[5] Hongxiu L., Reima S., "A Proposed Scale for Measuring E-service Quality", International Journal of u- and e-Service, Science and Technology, Vol. 2, No. 1, 2009.
[6] Muhammad U., Aamer N., "Automatic Generation of Java Code from UML Diagrams using UJECTOR", International Journal of Software Engineering and Its Applications, Vol.3, No.2, April, 2009.
[7] Weiqun Z., Gary B., "Contract-Based Software Component Testing with UML Models", International Journal of Software Engineering and its Application, Vol. 3, No. 1, January, 2009.

Authors

Dorin Bocu is a professor of Computer Science at Transilvania University of Brasov, Romania since 1990. He is a PhD in mathematics at the Transilvania University of Brasov, Romania. Currently he concentrates on topics around software engineering, IT project management and Intelligent computational systems.

Răzvan Bocu is assistant theacher of Computer Science at Transilvania University of Brasov, Romania since 2005. Hi is currently a PhD researcher at University College Cork, Ireland, in the area of Interactome Network Analysis.