Perspectives about paradigms in software engineering

Abstract. There is a broad use of the term "paradigm" in Software Engineering. Concepts such as structured paradigm, cascade paradigm or agent-oriented paradigm are very frequent in software engineering research proposals. In this essay we distinguish between functional and scientific paradigm and we show that the common use of paradigm in Software Engineering is about the functional or engineering paradigm rather than scientific paradigm. We distinguish among four possible perspectives and, in this context, we sustain that the scientific perspective is intrinsic and hence very difficult to properly identify and describe. We argue that a discussion about the scientific paradigm in Software Engineering could help us to evaluate and improve the research practice in the discipline.

1 Introduction

From the beginning of software engineering research, we can identify many proposals using the term "paradigm". We argue that the meanings of these references are engineering interpretations rather than scientific interpretations. Moreover, we think that there is not an obvious identification of what are the basics and philosophical assumptions in software engineering research which conform its scientific paradigm. In order to show these concepts we review in section 2 two common interpretations about the general concept of paradigm. In section 3 we review some interpretations of the paradigm concept in software engineering. In section 4 we distinguish between software engineering as a profession and software engineering as research discipline. Finally in section 5 we close the circle of the previous discussions arguing in favour of the necessary identification and description of the scientific paradigm in software engineering. The main conclusions are summarized in section 6.

2 The Concept of Paradigm

If we look up the word paradigm in some dictionaries, we will find out a definition built on words: "*model*", "*example*" or "*pattern*" [1-3]. For example in [3] paradigm is defined as "*an example that serves as pattern or model*". In [4] it is said that a paradigm is a "*a model of something which explains it or shows how it can be produced*". These common interpretations appear be included in computer topic, e.g. in [5] a paradigm is a pattern which constitutes a process or system model. Moreover, in [6] it is said that a paradigm is a technique, method or computer tool, which allows give a solutions to a specific problem.

On the other hand, there is a scientific use of the word. For example in [7] it is said that paradigm is "*a conceptual framework for a scientific discipline; a set of assumptions, methodologies, and objectives that determine a scientific investigation*". Moreover [8] refers to a "*set of fundamental assumptions that influence how people think and how they perceive the world*" and also as "*a framework of guiding assumptions, theories, and methods that define a particular approach to scientific problems*".

Consequently we have a common understanding of paradigm and, also, a more specific point of view with the scientific understanding of the word paradigm. One of the most influencing scholars on the scientific point of view is Kuhn [9]. He sustains that the scientific progress is done through paradigmatic shifts. He understands a paradigm as the total pattern of perceiving, conceptualizing, acting, validating, and valuing associated with a particular image of reality that prevails in a science or a branch of science. Kuhn formulated the cycled model of scientific progress with the first *pre-paradigmatic* stage, where a paradigm has not been yet broadly accepted; a *normal science* period, where the current paradigm is used; and a *revolutionary* stage, when the paradigm is changed; this process conform a *paradigmatic shift*. Although Feyerabend [10, 11] sustains that science does not precisely follows this pattern, the concept of paradigm imposed by Kuhn has gained acceptance in scientific discussions.

Kuhn also argues [9] that a scientific paradigm is a radical view, because when a paradigm changes the scientist works in a different world afterwards. He adds that there is a moment when different competing paradigms confront each other, generally by different schools. Moreover, these schools disagree what is a problem and what is a solution.

Summarizing we see two interpretations of the concept of paradigm: first, a common understanding related to a model, pattern or example of something and, on the other hand, the scientific approach, oriented to a set of assumptions related with a conceptual framework supporting these assumptions and influencing how scientists think and how science is carry out. In order to expose our points of view we distinguish these two approaches. We name the first *functional paradigm* and the second *scientific paradigm*. We think that both interpretations are different because a *functional paradigm* can be seen as an abstraction tool, something that we can change easily. We could follow a *model A* under some conditions, and under other conditions we could follow a *model B*. This does not mean that we have changed our basic assumptions or that we have modified our way of thinking.

On the other hand, we do not change easily our basic assumptions, because assumptions are beliefs. Therefore a scientific paradigm is constituted by a set of beliefs which influence the approach to define the research object and the ways to study it.

Finally, we also recognize other interpretations for the word "paradigm", especially from grammar, although these interpretations are not interesting for us at this moment.

3 The Concept of Paradigm in Software Engineering

In Software Engineering we have been using the word "paradigm" from many years ago: we have used the cascade paradigm, the structured paradigm, the object-oriented paradigm and some others.

Bosch [12], also from the software engineering point of view, said that "*paradigm refers to a set of related concepts which are used by a person to perceive the real world or a part of it*". But this sentence is from the modelling point of view.

When Korson and McGregor [13] argued in favour of object-orientation as a paradigm they said that this "*approach goes beyond the object-based technique…",* and that the "*… artefacts of the design process used in conjunction with a modelling-based decomposition approach yield a paradigm,…".*

Jennings [14] supports the idea of paradigm as a broadly conceptual framework. He argued that software paradigms generally go through three main phases: (1) early pioneers identify a new way of doing things, (2) individuals and organisations that are early adopters of leading-edge technologies recognise the potential, and (3) basic concepts become more widespread and enter in the mainstream.

In spite of these comments we have very few reviewing articles about the concept of paradigm from the software engineering discipline. One of the contributions on this topic is done by Göktürk [15]. Although he does not arrive at any final summary about the concept, his deep analysis includes many relevant points of view from Plato and Aristotle to the contemporaries Foucault and Kuhn. One of the explanations that Göktürk selects is the metaphor of the *darkness glass*. The paradigm would be this element that allows us a specific perception of the reality. There are two additional features of paradigms expressed by Göktürk: the existence of the imprecision in the conceptual framework and the idea of a broad application of it.

Moreover Göktürk has the perception that there is some mystic aura around the concept of paradigm. We think that this conception is supported by two elements. First the darkness glass metaphor, which reflects that an specific paradigm does not allow us to see the reality as is because there is a conceptual framework that acts as a filter and second, it is usual that there is not an agreement about the specific conceptual framework. On the other hand, we could speculate that this last feature allows an extensive use because many interpretations over the same conceptual framework are possible and therefore its use is not limited by the interpretation of this conceptual framework.

Additional support to sustain that a paradigm is constituted by a diffuse conceptual framework can be obtained mixing two results: first, the proposal of Jennings [14], who sustains that agent orientation is a software engineering paradigm and second the

work of Mao and Yu [16], who recently showed how the basic social conceptual framework of agent-oriented methodologies have many differences. Thus we have similar but not identical conceptual frameworks interpreting a specific paradigm (agent orientation).

Coming back to our proposal to distinguish between a functional paradigm and a scientific paradigm we can see that, the examples mentioned above refer to how to do software and not how to do science, i.e. the examples show a functional point of view of paradigms.

In the case of Göktürk's proposal, it is not clear how static is the inherent conceptual framework. i.e., can we easily change our *darkness glass*? Any answer (positive or negative) guides us to confirm our idea that dividing paradigm between functional and scientific. A positive answer implies that the perception can change easily and therefore we could use different paradigms under different conditions. On the other hand, a negative answer says that the vision is static, the conceptual framework is formed of solid beliefs and therefore they influence our thinking and hence our research.

4 The two faces of Software Engineering

In this section we briefly argue that Software Engineering has two faces, the professional face and the scientific one. We first review the concept of software engineering proposed by Sommerville [17] who briefly says that "*software engineering is an engineering discipline that is concerned with all aspects of software production*" and specifies that the basic activities are: software specification, development, validation and evolution. We think that there is no doubt about software engineering being an engineering discipline, in any case some arguments supporting this can be found in [18] where it is said that engineering principles have used successfully in order to build complex computer systems. Also in [19] this position is defended as a result of some answers about what is engineering.

The application of scientific knowledge always appears as one of the engineering principles. In this case, mathematics and computer science seems to be the most relevant sources of scientific knowledge provided to software engineering discipline. However we claim that software engineering is a research discipline too. We rely this belief on the work of Basili [20, 21] and Kitchenham [22, 23] among others, where software engineering is assumed a research discipline. In these cases the question is how to do research. In addition, according to the definition of software engineering, we can say that software engineering, as a research discipline, is concerned about the production of software and that the software process is the research object. Therefore, in software engineering as research discipline we have a relevant source of knowledge oriented to improve the software engineering practice. Thus, if the goal of a generic research area is to produce knowledge, then the goal of the software engineering scientific discipline is to produce knowledge about improving the software process.

When we distinguish software engineering as a research discipline we think that an additional differentiation from related disciplines is necessary. However this specific

differentiation could take us some additional space and it is not the focus of this essay. However we claim that computer science, software engineering and information systems research constitute different research disciplines with different research objects and different research approaches.

In Information Systems (IS) research the term paradigm in the scientific way is clearly acknowledged. For example Dobson presents [24], as part of its argumentation, a difference between the concept of scientific paradigm between Kuhn and Bhaskar and the implications for IS research. Fitzgerald and Howcroft [25] show paradigmatic dichotomies in IS research mentioning Interpretivism and Positivism. We think those research disciplines are different from Software Engineering in which the research object is the software process. Here there is a clear concentration about conceptual analysis and proof of concepts as its main research approach [26].

To sum up we argue in favour of differentiating between software engineering as a profession and software engineering as a research discipline. We also distinguish among computer science, information systems and software engineering research disciplines. This last distinction allows us to focus in software engineering as a different research discipline from computer science and information systems.

5 . The four perspectives

We have identified two types of paradigms and two facets of software engineering. Our proposal is that these two differentiations are orthogonal views i.e. that in practical aspects we can find the two types paradigms has been used by both, software engineering researches and software engineers. This cross product provides four different perspectives, (EE) engineering paradigms used by software engineers, (ES) engineering paradigms used by software engineering researchers, (SE) scientific paradigms used by software engineers, and (SS) scientific paradigms used by software engineering researchers. We illustrate these four perspectives in the figure 1.

On the EE perspective we observe that software engineering as a profession uses the different paradigms as tools. Maybe a simple add can be done in a structured way, a calculator could be implemented with a proper class and a data processing service could be implemented using an agent. But all these alternatives are not really competing. They are different choices to tackle a software development process. Hence the structured, object-oriented and agent-oriented paradigms coexist without problems and moreover, we sustain that this coexisting is positive and synergic. We claim that these functional paradigms are really engineering paradigms, i.e., model or patterns that guide us the modelling when we need to develop software.

On the ES perspective we observe that engineering paradigms constitute firstly research products and thus a way to focus the current solution approach to software development. We also sustain that engineering paradigms are not scientific paradigms, because they do not change our assumptions about software engineering is, they do not change our research object (the software process) and they do no change our way to do research. Moreover, we believe that the successful of software engineering as research discipline as precisely providing engineering paradigms with their related

components (for instance design tools, programming languages, developing techniques and testing methods). Therefore we see that the normal and historical behaviour of the software engineering research discipline has been to produce engineering paradigms about how to develop software.

	Engineering Paradigms (e.g. Object-Oriented)	Scientific Paradigms (e.g. Positivism)
Engineering Use (goal: to produce software)	**EE** Like a Design Metaphor	**ES** To understand the specific domain and developing assumptions
Scientific Use (goal: to produce knowledge)	**SE** Like a specific solution approach	**SS** To produce software engineering knowledge

Fig. 1. Perspectives about paradigms in software engineering

On the SE perspective we have found the use of scientific paradigms and some specific research methodologies into the software process. For example in [27] it is reviewed some scientific paradigms and its application to software development is analyzed. Other related proposals are [28, 29] where action research and focus groups research methodologies are proposed like requirements elicitation techniques. i.e. scientific paradigms and scientific approaches used into the software process as engineering techniques.

About the SS perspective is where we believe that a debate is necessary. We think that the behaviour of the discipline has been static. We have not found a paradigmatic SS discussion in software engineering. In the sense of Kuhn perhaps we are living a normal science period. But, as Dieguéz Lucena [30] has explained, this period has been critiqued because it has an inherent sense of mediocrity.

This point, should be very debatable, because, there are many proposals about the research methodologies that software engineering could follow [26, 31-34]. Furthermore, these proposals are oriented to change the research practice and, in this sense, we can say that there are initiatives to support a paradigmatic change in software engineering research. However, these proposals are based mainly on importing research methodologies, i.e. using somewhere formulated methodologies in Software Engineering. Thus research methodologies are visualized like technologies.

Indeed, our belief is that we need a broad and critique discussion about what is really the research scientific paradigm in software engineering, the SS perspective. We think that it is not clear, but, at the same time, we think that its identification and description is the first step to evaluate it, which could allow us seeing our set of inherent assumptions with their weak and strong points. We think that this step is foundational in the generation of a true paradigmatic-shift in software engineering research.

6 Conclusions

We have presented a review of the concept paradigm. We have argued that there exist at least two types of paradigms: functional paradigms and scientific paradigms. In a parallel way we have argued that software engineering has two faces, the professional and the scientific face. We have shown how the traditional concept of paradigm in software engineering corresponds to the functional type, i.e. that paradigm is broadly conceived as a modelling tool rather than a philosophical point of view. Thus we have identified four perspectives to understand the use of paradigms in software engineering. We argue that the scientific perspective of the current software engineering scientific paradigm is not evident and a broad discussion could be the first step to acknowledge our general assumptions which should be the start point of a real paradigmatic shift in software engineering, which has been the base of memorable research outcomes.

References

1. Cowie, A. P. (ed.): Oxford advanced learner's dictionary of current english. Fourth ed., Oxford university press, Oxford (1989)
2. O'Neill, M. (ed.): Concise dictionary & thesaurus. ed., Chambers Harrap Publishers Ltd, Edinburgh (2001)
3. Soukhanov, A. H. (ed.): The American heritage dictionary of the English language. Third ed., Houghton Mifflin Company, Boston - New York (1992)
4. Sinclair, J.: English dictionary for advanced learners. Chief editor John Sinclair, Third ed., Harper Collins publishers, Glasgow (2001)
5. Microsoft: Diccionario de Informática e Internet. Ediciones profesionales Microsoft, Mc Graw Hill Interamericana, Madrid (2001)
6. de-Alarcón, E.: Diccionario de Informática e Internet. Ediciones Anaya Multimedia, Anaya, Madrid (2002)
7. Archaeology Wordsmith. Visited at Feb 2006 Accesed at http://www.reference-wordsmith.com/archword/dict.html
8. Anthropology/Sociology dictionaries. Visited at Feb 2006 Accesed at http://www.webref.org/anthropology/ and http://www.webref.org/sociology
9. Kuhn, T. S.: The structure of scientific revolutions. Third ed., The University of Chicago Press, (1996)
10. Feyerabend, P.: Contra el método. Esquema de una teoría anarquista del conocimiento. 2nd ed., ARIEL S.A., Barcelona (1989)
11. Feyerabend, P.: Adios a la razón. 3th ed. 1996, Tecnos S.A., Madrid (1992)
12. Bosch, J.: Paradigm, language model and method. In Proc of the Workshop on Research Issues in the Intersection of Software Engineering and Programming Languages, ICSE-17, April 24-25 (1995)
13. Korson, T., McGregor, J. D.: Understanding object-oriented: a unifying paradigm. Communications of the ACM Vol. 33. (9) (1990) pp. 40-60
14. Jennings, N. R.: On agent-based software engineering. Artificial Intelligence. Vol. 117. (2) (2000) pp. 277-296
15. Göktürk, E.: What is “paradigm”? Visited at Dec 2005, Accesed at http://heim.ifi.uio.no/~erek/essays/paradigm.pdf

16. Mao, X. J.,Yu, E.: Organizational and social concepts in agent oriented software engineering. In Agent-Oriented Software Engineering. Lecture Notes In Computer Science Vol. 3382. (2005) pp. 1-15
17. Sommerville, I.: Software Engineering. 7th ed., Addison Wesley, (2004)
18. Wasserman, A. I.: Toward a discipline of software engineering. IEEE Software Vol. 13. (6) (1996) pp. 23-31
19. Shaw, M.: Prospects for an engineering discipline of software. IEEE Software Vol. 7. (6) (1990) pp. 15-24
20. Basili, V. R.: The role of experimentation in software engineering: past, current, and future. In Proc. of the 18th International Conference on Software Engineering, (1996) pp. 442-449
21. Basili, V. R., Selby, R. W.Hutchens, D. H.: Experimentation in software engineering. IEEE Transactions on Software Engineering, Vol. 12. (1986) pp. 733-743
22. Kitchenham, B. A., Dyba, T., Jorgensen, M.: Evidence-based Software Engineering. In Proc. of the 26th International Conference on Software Engineering (ICSE'04), (2004) pp. 273-281
23. Kitchenham, B. A., Pleeger, S. L., Pickard, L., Jones, P., Hoaglin, D., El-Emam, K., Rosenberg, J.: Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering, Vol. 28 (8). (2002) pp. 721-734
24. Dobson, P. J.: Critical realism and information systems research: why bother with philosophy. Information Research, Jan, Vol. 7 (2). (2002)
25. Fitzgerald, B., Howcroft, D.: Competing Dichotomies in IS research and possible strategies for resolutions. In Proc. of the Int. Conference on Information Systems (ICIS'98), Dec (1998) pp. 155-164
26. Glass, R. L., Vessey, I., Ramesh, V.: Research in software engineering: an analysis of the literature. Information and Software Technology, Jun Vol. 44. (8) (2002) pp. 491-506
27. Hirschheim, R., Klein, H.: Four Paradigms of Information Systems Development. Communications of the ACM Vol. 32. (10) (1989) pp. 1199-1216
28. Fowler, D. C., Swatman, P. A.: Building information systems development methods: synthesising from a basis in both theory and practice. In Australian Software Engineering Conference. ASWEC, Conference, Location (1998) pp. 110-117
29. Kontio, J., Lehtola, L., Bragge, J.: Using the focus group method in software engineering: obtaining practitioner and user experiences. In Proc of the International Symposium on Empirical Software Engineering, ISESE '04, Aug (2004) pp. 271-280
30. Diéguez-Lucena, A.: Filosofía de la Ciencia. Filosofía series, Biblioteca Nueva S.L., Madrid (2005)
31. Basili, V. R.: The Experimental Paradigm in Software Engineering. Lecture Notes in Computer Science. Vol. 706. (1993)
32. Basili, V. R.: The Role of Experimentation in Software Engineering: Past, Current, and Future. In Proc. of the 18th International Conference on Software Engineering, May (1996) pp. 442-449
33. Kitchenham, B. A., Dyba, T., Jorgensen, M.: Evidence-based Software Engineering. In Proc of the 26th International Conference on Software Engineering (ICSE'04), (2004) pp. 273-281
34. Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., Rosenberg, J.: Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering Vol. 28. (8) (2002) pp. 721-734