Viewpoints on Viewpoints

Julio Cesar Sampaio do Prado Leite*

Departamento de Informática, PUC-Rio
R. Marquês de S. Vicente 225
Rio de Janeiro 22453-900 Brasil
julio@inf.puc-rio.br

Abstract

This paper expresses the viewpoints, or opinions of the author on the use of viewpoints in the software development process. I will concentrate most of my attention on requirements engineering, but I also treat the topic in the broader environment of software development. I will state without much justification, besides the presentation itself, key concepts that I believe are the basics for the use of viewpoints. I classify viewpoint research in three areas: opinions, specifications and services, each of which is described in detail. I conclude listing topics in need for further research.

1 Introduction

In this brief position paper I will focus on general aspects of viewpoints, and as the title conveys I will give a personal view of the importance of *viewpoints* to the process of software construction. First of all, we have to understand that the process of software construction is in itself a social process and the resulting product will operate in an environment where social aspects will have to be considered.

I would say that in broader terms, the fact that the software engineering community is becoming aware of the concepts of *viewpoints* is a strong sign that the field is leaving its self centered technical superiority kind of discourse to be aware of aspects that are more and more soft, almost fuzzy. With this respect, the field of requirements engineering has been playing a major role. The first international requirements engineering symposium held in 1993 (RE93) at the Coronado Hotel in San Diego is a clear sign that the softer side of producing software finally started to get into the mainstream of computer science/software engineering research.

I believe that *viewpoints* have three main areas in which research and development will contribute to an effective improvement of software production. First of all, I believe that the ideia of *viewpoints* is an acknowledgement that several different opinions have to be considered before and during software construction. Second, I believe that *viewpoints* is an important representation concept for specifications, since it provides an important frame to propose and implement consistency analysis, thus incrementing the possibility of model verification. Third, the use of *viewpoints* as services is a structuring strategy for software definition as well as for software operation.

Below I will list the important aspects we already know and the pitfalls we should avoid in progressing with the work in *viewpoints*. I will treat each of the areas as a special section and will end, at the conclusion, with my view of future research.

2 Viewpoints as Opinions

It does not matter which kind of software we will produce, it is clear that we need a step in which the definition of the future product will take place. However, before the definition, we have to decide that a software will be necessary to perform a certain task in a composed system. This system can be an information system for an organization, an operating system for a certain type of hardware, an application for a certain software market, an embedded application for a certain appliance and so on. The point is that it does not matter if I will be doing software for controlling baggage or for taking care of cash flow or for a new word processor, beforehand somebody will make a decision to build it. The decision is normally a multi agent process, that is it comprises different people with an overall major goal, but with different personal experiences, tastes, and beliefs.

So, even before some organization starts to produce any given software, different agents had to negotiate on the necessity for producing that software. This happens before what we call software definition step and for sure will impact the definition step. Again, the software definition process does not happen in a vacuum, but in a social context where agents or actors already had to negotiate premises, priorities and resources before making a decision. The idea of negotiation, such that different stakeholders will be in place, is a simple observation that one dealing with software construction should not underestimate. We need to acknowledge the existence of different opinions or different ways of seeing the same thing.

Several artists and social scientists used different views in their work. Carroll's Alice, sculptures that take different shapes as we move, films like *Cinena Paradiso* where we can see the audience from the screen as well as the screen from the audience, are some of the examples of the awareness of viewpoints by artists. Viewpoints are essential to learning,

*This work is supported in part by CNPq grant n. 523894/95-3

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.
SIGSOFT 96 Workshop, San Francisco CA USA
© 1996 ACM 0-89791-867-3/96/10 ..$3.50

and fundamental in any balanced society. As an example of learning, I once more will use the film domain: in the motion picture *Groundhog Day* the main character is locked in time and has to live one day infinite number of times. No matter what he does he can not get rid of that day. After several tries to commit suicide, he quits and start to look at different ways of enjoying the day. The long process takes him to unexpected situations where he becomes a better person by doing the same thing differently each time. He finally gets to a point, where the circle breaks and he gets out of the trap. The important aspect is that he could experiment from different viewpoints for a better way to do something. In general social terms, the principle that more sources of information provide a better understanding of a subject has been used for centuries in court investigation. Different witnesses may have conflicting or complementary recollections.

Back to the software. If we understand that before the decision of producing a given software there exists a negotiation process, we are closer to understand the importance of taking several viewpoints in consideration before we start to produce the software. At this moment, we need the assistance of requirements engineers, whom should know the methods, techniques and tools of requirements engineering in order to produce the best possible requirements for the software to be built. As we are well aware, the definition part of the software construction process may take different shapes, schedules and levels of difficulty. So, the definition of a new application for directory management and display, under a certain operating system in a software company that markets its own products, is completely different from the definition of one part of the control software of a spacecraft being developed at Nasa. By the same token it is completely different defining an OLAP based decision system than it is to develop a software to help the interface of a VCR.

Our point is that, it does not matter which kind of software you will produce and which kind of social environment the definition process will occur, there is a need to take in consideration the fact that different viewpoints may exist regarding the development of that software.

Domain construction may also been seen as a process where *viewpoints* could be used as a learning mechanism. That is, several *viewpoints* will be considered, compared and negotiated in order to produce a domain.

The main justification for taking in consideration different viewpoints is that by doing so, the final product should better fit its purpose. It is important to have in mind that sometimes the fact of listening to different viewpoints may be counter productive. Take for instance an example of an information system being developed by the top management with a hidden agenda geared towards process reengineering. In this case it may be a waste of resources to take in consideration the viewpoints of those which will not be agents in the future environment where the software will be deployed.

3 Viewpoints as Specifications

The study of *viewpoints* in the context of requirements engineering have led to the proposal of several techniques. These techniques inherit from software engineering the tendency to systematization, so they are fundamentally based on languages at the specification level of abstraction. In this context, the highlight is on the use of viewpoints as a way of detecting problems of consistency and completeness.

Overall, the literature that uses this approach does not detail the process of producing such specifications. They assume that they are produced and concentrate on the task of detecting problems by comparing different specifications. Most of the work published so far, does comparison on a pairwise base, that is given two specifications: rules are fired to compare specification A with specification B. Given this general framework, it is important to stress the following points:

- Specification A and B are specifications of the same system.
- Specification A is a *viewpoint* if only if there is an agent responsible for its content.
- Specification B is a *viewpoint* if only if there is an agent responsible for its content and if only if the agent responsible for B is different from the agent responsible for A.
- The language of specification B may be different from that of specification A.
- If the language for specification A is different from the specification B but they are not different *viewpoints*, that is the agents for A and B are the same, then we have different *perspectives*.
- If the language of specification B is different from the language of specification A and they are different *viewpoints*, then we have different *viewpoints* expressed in different *perspectives*.
- A specification is usually written by a software engineer and not by the agent responsible for its content. This fact brings up the figure of the *specifier*, which himself/herself/themselves may have a *viewpoint* on the contents of the specification. For the sake of simplicity we will not discuss the role of the *specifier* any further.

I have used the word agent, in a very broad sense. It may represent just one particular person, a group of persons, or an important stakeholder. The *viewpoint* will be an expression of the agents' opinions.

Given the above and given that the specifications produced are amenable to be treated as data, it is possible to automate a comparison in a pairwise fashion. Assuming that there is a way of performing this comparison, we need to find commonalities between the specifications and then analyze these commonalities. It should be the case that some overlap does exist between the two specifications, since they are different expressions of the same system. So, any automation technique that deals with comparison of *viewpoints* must deal with detecting overlaps. Once overlaps are detected, it is then possible to detect inconsistency or incompleteness.

Overall the automation strategies for comparison lies in the following situation.

> If the specification A has an overlap with specification B, and the overlap is not an identity, then the two specifications are inconsistent with each other. A may be right and B wrong or vice-versa, or both may be wrong. If there are parts of both specifications that do not overlap then it is the case of incompleteness. So, A may be incomplete or B may be incomplete or both may be incomplete.

The important effect of using *viewpoints* for handling specifications is that we increase the power of inconsistency checking. If we take the frame – validation and verification – we will see that by using viewpoints we are blurring the distinction between verification and validation. If we consider validation as checking a formal model with reality and verification as the checking of two formal models, we will find that *viewpoint* comparison lies in between, since they represent fractions of reality but are formal and are on the same level of abstraction. Usually, when we perform verification we are treating with two formal descriptions in different levels of abstraction. We also have to consider intra verification, that is, given a specification in a given language representing one sole *viewpoint* it is also possible to analyze it by performing inconsistencies checks between its parts based on the expected built in redundancy of the given specification language.

The literature on specification has originally dealt with intra verification and with verification between different levels of abstraction, from programs to specifications. The idea of *viewpoints* brings to light an important distinction and an increase on the potentialities of consistency checkers. The possibility of dealing with inter verification, two specifications on the same level of abstraction written from different angles but focusing the same problem, increases the capability of consistency checking.

The availability of a machinery capable to help the detection of inconsistency and incompleteness between specifications is just half of the problem. Contrary to the approach taken in the usual verification process, we can no longer say that we found an error. It is also not sufficient to get back at reality and check where the error is. The fact that we are dealing with viewpoints implies that the feedback loop will require a negotiation process between the agents responsible for the *viewpoints*.

Automation for the negotiation process is much more complex than automation for inconsistency and incompleteness detection, but there are in the literature works in the area of automation of negotiation based on explicit establishment of goals and priorities for specifications parts. I see detection as a necessary step towards the production of an agenda for further negotiation. So, the feedback from the agents are an indispensable information in order to merge *viewpoints*

Merging *viewpoints* will depend on the management style of the software construction process. How long is it possibly to live with inconsistency? The delay on negotiation may imply a burden on the implementation process. On the other hand it will make the process more flexible and amenable to change. It may also be the case that some inconsistencies are duable to be fixed by negotiation patterns, but this is more a problem of *viewpoint* as services.

4 Viewpoints as Services

The study of *viewpoints* in the realm of the integration of the software with its environment as well as the study of database views gives rise to the third area where *viewpoints* have been studied. The main idea here is that future clients of the software system will interact with the system in different ways. The fact that a future software system will be seen in different ways by different agents, brings to the construction process the idea of services. These services will be provided by the system and will be linked to different *viewpoints*.

The major aspects being discussed in the literature is how to construct and use a system that will provide a set of services and that those services will be linked to different sets of agents. The database approach distinguishes between different views that a given database model will have to respond. Here the focus is on view integration and not on using *viewpoints* as a means for detecting problems in a specification. It is assumed that a given *viewpoint* related to a service is well established and that services do interact in a established way.

This approach is particularly well suited where there is an overall agreement on the functions and constraints of the software system, but there are more than one type of user/client community involved. Using the idea of services it is possible to tune the interface and the functions of a given software to attend particular interests of the community. The construction of these services linked to particular agents provides a system structure based on the agents. This structure not only makes it possible the construction of services that will fit the needs of the environment but also provides an opportunity for traceability towards the clients of the system. The idea of using *viewpoints* as a structuring mechanism is also been used for composing specifications.

Building a system by the combination of services also provides an excellent opportunity for the customization of the software interfaces as well as for a security policy based on responsibilities. The database community has been using this notion, which they call views, to differentiate between applications using a common database.

The idea of partitioning the system as a combination of services provides an outside-in view of the software and should be particularly important for systems built on top of component based architectures. The integration of the system is decided as part of the internal architecture and not from the outside, such that the designer will have more freedom to choose how to connect several components.

5 Conclusion

We expressed our viewpoint on *viewpoints* by providing an overall analysis of three main areas of *viewpoint* research. The division was made just to help the distinction of topics, but by no means implies that they are independent research areas, on the contrary they are all united by the common idea that it is necessary to have more than one vision or opinion in the process of producing software, specially in the process of software definition or the engineering of requirements.

I would say that the first area is more related to elicitation and validation, the second more related to issues of representation and verification and the third more related to issues of organization and presentation. All of them are intrinsically connected, but the problems are difficult enough, such that a more comprehensive approach will only be possible when we have enough results in each of these areas.

I believe that several research problems do exist that requires further investigation in each one of these areas. In particular I would like to point out some challenges we have ahead of us.

With respect to **Viewpoints as Opinions**, we did already had a reasonable progress in the understanding of the requirements problems and the different frames of thought that can be used to bridge the gap from reality to formal

models. The area has a much more broad view of requirements and talking about the relationship with social sciences is not a taboo anymore. I believe that it is just in this junction area that we will require more further work. I particularly favor the engineering approach, that is what we as requirements engineers could invent in terms of tools, techniques and methods to help the use of several of the available social strategies for handling viewpoints. On the other hand we may, as well, be more critic of the social oriented strategies and adapt them to the kind of work we perform, very much geared towards rationalization.

With respect to **Viewpoints as Specifications** we need to work on comparison, in order to advance the possibility of using more semantics and being less dependent on syntax, but that is the general aim of the software engineering community. I strongly believe that a reasonable combination of syntax and semantics still could give us more than we have now. On the other hand, we also have to progress and try to model the different situations that arises when dealing with *viewpoints*, in special I believe that the fact that a *specifier* is an agent needs to be further explored in the representations as well as in the comparisons heuristics. Another aspect that needs further research is the use of *viewpoints* and *perspectives* simultaneously in the process of *viewpoint* analysis. Yet another specific topic is related to the comparison of several *viewpoints* in parallel and not in a pairwise fashion.

With respect to **Viewpoints as Services** we need to improve the ability to consider non-functional requirements in particular those related to interfaces. I also believe that the link of this area with the issue of traceability should be better explored. Since I believe this area is more related to organization and presentation, I see here an opportunity for work trying to bridge these three areas. Another point where I believe further research is necessary is for how long in the process of software production we may delay the merging of partial specifications into a specification ready to be implemented.

6 Bibliography

This position paper does not include a reference list on purpose. We assume that most of the views presented above should be present in the Viewpoints Workshop, but not to let the readers without any reference I would recommend as a source for viewpoint bibliography two *IEEE Transactions on Software Engineering* articles on the theme. The first, "Requirements Validation Through Viewpoint Resolution" by Leite and Freeman on Vol. 12, N. 12 (Dec. 1991) and the second, "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification", by Nuseibeh, Kramer and Finkelstein on Vol. 20, N. 10 (Oct. 1994). Another reason that a reference was not used is that most of what was presented is a very free interpretation on the work of others. I tried the best I could to express most of the views in current discussion, but I certainly have left uncovered parts. I hope this position paper brings up this type of discussion.