

Empowering the Developer with a Personal Mentor

Gunnar Övergaard, Jaczone AB

Wouldn't it be nice to have as your personal assistant and mentor an expert that will help you when developing systems? An expert that can provide you with the information you need for the moment being, that can evaluate your work and provide solutions to modeling problems, that can perform routine work etc. An expert that is always available, never busy with something else. This expert will not replace existing tools, processes, or techniques but will enable you to use them in a more advantageous way. In short, how about having a personal expert that can facilitate your use of best practices and tools in an optimal way?

Lack of Knowledge

Developing software is becoming increasingly difficult, even though new languages, development processes and tools, platforms, standards, and the like keep being developed to support us in our work. The team members' lack of knowledge is one of the major reasons why development projects so often fail. Although the relevant information exists, it is difficult to find and apply it in a specific situation. The information is buried in books and manuals, in web pages, or in the heads of some experts, and how to find it and get access to it is a challenging task. Furthermore, even if the developer has the required knowledge, he or she does not always realize that it is applicable to the problem at hand.

To improve their knowledge team members are often given additional training. However, the result is often a basic level of understanding. Moreover, the classes seldom include substantial practical experiences. A project manager will usually try to add a few experienced people to the project, complement these with some skilled consultants who participate in the project as well as with a few mentors who support the team members, hoping that this will make up for the lack of experience and knowledge. However, these experts tend to be very busy and they seldom have time to answer questions, to suggest solutions to problems, to show how a task is to be performed, or to evaluate a proposal when a team member needs it. If no expert is available when a problem turns up, the team members tend to try to solve it on their own, with results of varying quality. Who will stop and wait for help in the short-time projects typical of today?

Personal Tutor and Assistant

When you have been hampered a few times by the absence of an expert who can help you, don't you wish you had a tutor that is always available to help you solve the problems of software developing? A tutor that isn't attending a separate meeting or engaged with something else when you have a problem; a tutor that can give you his or her full attention. Yet, for several reasons it is not feasible to have one person, who is an expert on system development, next to you attending to your needs, following each action you do, always evaluating your work, and suggesting what step to take next. However, a computerized tutor can do if not everything but a lot of what a human tutor can do for you. In addition, it is not restricted to office hours, it does not fall ill, and it is not busy – it is available when you need it. Such a tutor does not replace existing development techniques or tools. Instead, it enables

you to fully utilize them by suggesting what you should do and by teaching you what you need to know – when needed.

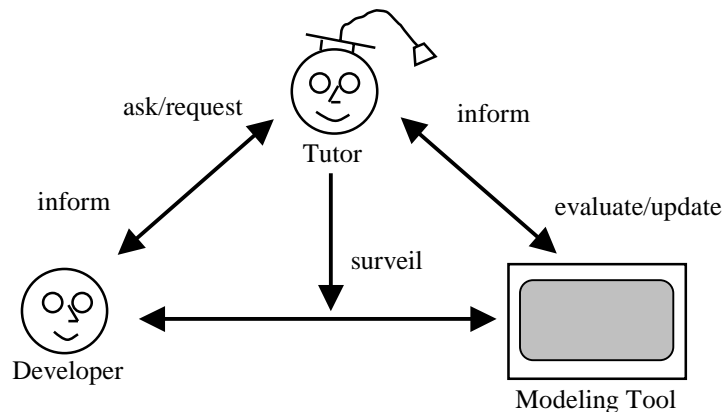


Figure 1. The tutor checks what you are doing and suggests the next step, evaluates your work, proposes solutions on problems, and informs you where you will find more information.

Assume that you are working in a development project using e.g. the Rational Unified Process (RUP) and the Rational Rose (Rose), and that you have completed the introduction classes, i.e. you are familiar with most of the basic concepts and you know how to create model elements and how to associate them with each other. However, to say that you know all, or at least most of RUP at a detailed level would be an exaggeration. Similarly, you know how to draw UML diagrams in Rose, but you do not feel too comfortable with version handling of different modeling elements, linking Microsoft Word documents to model elements, and similar operations.

Last week your manager introduced the new, computerized tutor to all the team members, and now that you have used it, you understand some more of RUP and you feel to some extent acquainted with it. This morning, the project manager asked you to talk to two persons representing the customer to identify what they want the system to do within a specific subject area. This should not pose any problems to you as you have worked within a similar subject area in another project, although you are a bit excited when you meet them. However, after a few minutes' friendly conversation you are quite calm again; this will not be a problem.

“Before we start, can you explain in more detail to us what we are going to do during this session?” one of the customers says.

Now you become a bit nervous again as you are quite aware that you have not used use cases before, but you say “We are going to identify the actors and the use cases.”

“Ok”, says the other customer, “Who are these guys?”

“An actor is someone or something outside the system interacting with it.” you quickly respond remembering what you were told in the class. “And a use case is a way to use the system.”

“Oh, I understand.” says the first customer. “Each menu alternative in the GUI is a way to use the system, so we should be able to identify the use cases very quickly.”

Now, you start sweating. This is not what you were told in the class, but you cannot put your finger on what is wrong.

This is a very common situation. You do not know exactly what a concept means, so you cannot really say why an argument is incorrect, but you have a gut feeling that it is wrong. This is where you turn to your tutor and ask for an explanation of what an actor is (a role

played by someone outside the system using the system, as perceived from the system's point of view), what a use case is (a complete usage of the system that gives a value to a stakeholder), and why each menu alternative in the GUI is not a use case (a use case can include a sequence of menu selections; a single menu selection does not necessarily cause the system to do something that has a value to someone).

After the clarification, the first customer asks you to take the lead in the identification of their system's use cases. Desperately you try to remember how the exercises were performed during the class. You recollect having seen a workflow graph in RUP describing how the work is to be performed, but you have forgotten the details of it. However, you remember that you found it strange that the graph did not mention any actors, but you do not remember the explanation why they were not mentioned. On the other hand, did not the teacher talk about a Find Actors and Use Cases activity? Come to think of it, how did that activity fit in with the workflow?

"We start by identifying those who will use the system, i.e. the actors." you stammer, vaguely remembering the class' exercises. "Based on these users, we can identify how they want to use the system."

We often remember the procedure principle for creating a model, but how the work is to be done in an efficient way and what different criteria are to be used are seldom rules and instructions that we remember by heart. Once again, you then turn to the tutor, who will present the steps in the Find Actors and Use Cases activity to you (and explain that the activity is performed in some of the boxes in the workflow of the Requirements discipline). What is even better, the tutor will ask you if it should present to you the Use-Case Workshop Guidelines that are described in RUP. This will really be of great help to you! Now both you and your customers will know what is to be done, and you will perform the workshop being sure that you will not miss any of the important actions.

During the process of identifying the actors and the use cases, the tutor pushes ahead by continuously updating and presenting a short list of relevant micro-activities to choose between. In this way, you and your customers will be able to focus on what is important for the moment, and you will not miss anything or diverge into what might be interesting but irrelevant matters. Moreover, since the tutor automates some of the work entering the model into the tool, you will move forward quickly without having to worry about how some of the operations are to be performed in Rose.

Suddenly, when you are identifying one of the use cases in the system, one of the two customers says: "The authentication of the user that is to be performed in this use case is the same as in this other use case that we have already identified. I am still not sure how the authentication is to be performed, but the procedure ought to be the same in both use cases. Can't we for the moment being just state that they are to be the same and that the details will be filled in later?" You then realize that the authentication procedure is a good candidate for an abstract use case referenced by the other two use cases, but you are not sure whether to use include or extend relationships between the abstract use case and the original use cases. What is worse, you do not know if you will be able to explain the two relationships to the customers.

Instead of blabbering about include and extend relationships, you ask the tutor what to do when use cases have commonalities. The tutor will give you and the customers a ten-minute tutorial explanation of how to model such a situation; moreover, it will state what these relationships are and when they are to be used. The tutor ends by saying that unless you have an explicit requirement that the flow must be the same in multiple use cases, you should not use any relationships between use cases at this stage. The customer answers that it is a firm

requirement that the authentication procedure be the same in both use cases, so the tutor helps you to extract the commonality into a separate, abstract use case and to define two include relationships from the original use cases to the abstract use case.

At the end of the day, when you close down your computer, you remember the comment from one of the customers when you met the project manager after the workshop: “This was great! I have never participated in a workshop like this that has run so smoothly.”

“Thanks, tutor. See you tomorrow,” you mumble as your computer comes to rest.

Constant Learning

Different persons will use the tutor in different ways: The more advanced modelers will use it to verify that the produced result is correct and to get help performing routine work. Other, less advanced developers will use it as a guide through the development process that will inform them what to do next, what different things mean, as well as identify errors and suggest solutions.

However, all kinds of developers will continuously learn from the tutor. Not only from mini-tutorials and help texts, but also from the actions performed by this expert. Each message from the tutor will not only explain what is to be done, and why, but it will also provide a motivation for it. If more information is needed, the mini-tutorials are very efficient. Additional information is found using the links provided by the tutor. This means that the project team will work according to the same (configuration of the) process and, at the same time the members will explicitly learn which things are relevant at a particular moment during the work, as their tutors are always present. In addition, the tutor will help an inexperienced developer to get over the threshold of starting to work with new tools and to apply a new process.

How this is Accomplished

Is this tutor a vision? No, it is not. Today, such a tutor can be implemented using a collection of intelligent agents. Each agent is specialized for a specific task: one knows how to identify actors and use cases, and another how to structure the use-case model using include, extend, and generalization relationships. Other agents handle other parts of the development process, such as the definition of the system architecture, the realization of a use case, the mapping of an analysis model onto a design model, the specification of a design element etc.

One such tutor is WayPointer. It currently includes a collection of agents supporting the development team in the process from capturing requirements down to J2EE design, but in the future it may also be able to support activities such as project management, managing business rules, and testing. WayPointer continuously receives information about modifications of the model which resides in a modeling tool, like Rational Rose or Rational XDE. Based on this information, together with its own knowledge, WayPointer interacts with the developer. It can, for example, suggest which next step is to be performed, suggest enhancements of the model, and present solutions on the modeling problem at hand. WayPointer is not limited only to describing the current problem; it can also explain why it is a problem, and why the suggested solution is preferable. If the developer does not have enough background information, WayPointer can also provide additional information about the subject.

WayPointer uses the APIs of Rational Rose and Rational XDE to read the relevant parts of the models as well as to add and to modify information expressed in these modeling tools. Furthermore, apart from interacting with these tools, WayPointer has its own GUI to interact directly with the developer, and it can also utilize links into RUP, to different web pages as well as to other sources of information.

It is important to note that not everyone needs the same assistance, so WayPointer is of course configurable. First, it can be adapted to an individual developer and the way he or she usually works. Second, each project has its own configuration of RUP (the development case), so WayPointer can be configured to follow the rules and recommendations of the project's development case. This implies that all instances of WayPointer do not behave identically, although they have the same origin. They will act differently for different developers in the same project as they will have different needs, as well as for the one developer working in different projects as they have different development cases. Hence, despite (or rather thanks to) the different behaviors of the personal configurations of it, all users of WayPointer have their own personal mentor guiding and teaching them how to use the existing process and modeling tools in an optimal way.

In short, the personal tutor is no longer just a vision – it does exist.

References

“A Multi-Agent System Assisting Software Developers” by Ivar Jacobson and Stefan Bylund.

WayPointer, an agent-based tutor. <http://www.jaczone.com/product/overview/>