1 slide

Hi! In this video, I will introduce the prototyping model. I will present the idea behind this model and several approaches – how to implement it. Later we will discuss the advantages and disadvantages of the model, and finally we will see – when to use it. So – let’s start!

2 slide

To talk about software prototyping, we need to have a similar understanding – what is a software prototype. A software prototype is an incomplete software dedicated to demonstrating various aspects of the software such as design, functionality, and others.

The idea of prototyping is to validate software to be created by using a working prototype instead of using images and descriptions.

The prototyping software development model is an iterative/incremental model. The software is created iteratively as a whole or in small increments.

The prototyping practice appears in the famous book of Frederick Brooks,, “Them Mythical Man-Month “released 1975.

PrototypingPrototyping is a frequent technique in nowadays software development, but prototyping as a software development model is not used frequently. Anyway, let’s get familiar with the idea of prototyping.

3 slide

Let’s suppose that we face several uncertain factors such as not clear requirements or technologies that are not well understood. So, we gather the initial, abstract requirements from the customer and build an initial prototype. That prototype is provided to a customer for evaluation. The customer evaluates the prototype and provides feedback. Now, the development team discusses the feedback, and if the prototype is not adequate, the team refines the prototype and provides the improved prototype for further evaluation. The team receives feedback for other discussions. This cycle continues until a customer is happy with a prototype. And now the team proceeds. How!? It depends on a specific prototyping method. So, let’s investigate them.

4 slide

There are two major methods. The first one is “throwaway prototyping “or “rapid throwaway prototyping “. It’s logical. If you need to throw away your prototype, better you do it fast, so you don’t get attached to your prototype, and the work on the software to be thrown away is minimized. One might ask – why there is such a method at all. Well – this is balancing between the pros and cons of the prototyping method. We will get to it, but you should know that sometimes it works very well.

So, the start is the same as in the previous diagram. The development team gathers the initial requirements and builds the software prototype. The prototype is provided to the customer for evaluation. The customer gives us feedback, and our development team refines the prototype if it is not good enough. Until this moment, everything went the same as I presented in a previous slide. The difference is the outcome when the customer and the team find the prototype to be adequate. The useful things we discovered, such as refined requirements, designs, working algorithms, and others, are provided for a development team that uses another software process to create the actual software. And the software prototype is thrown away.

Still, it is a bit of pity to throw away the work you have done. Why not keep it and improve, so eventually, the prototype becomes the actual working software!? Yes, it is possible. I will present the second major method – evolutionary prototyping.

5 slide

Evolutionary prototyping is very similar to throwaway prototyping. The development team gathers the initial requirements and builds the software prototype. The prototype is provided to the customer for evaluation. The customer gives us feedback, and our development team refines the prototype. Take into account that in this model, the software system, as such, is evaluated. Therefore quality methods should be applied to verify and validate software in contrast to throwaway prototyping. When the created system becomes adequate, it is deployed.

Before discussing the advantages and disadvantages of the prototyping methods, let’s investigate two special cases.

6 slide

The first one is related to the deployment of the software using small increments. It means that the software is split into a number of subsystems, and each subsystem is being developed using evolutionary prototyping.

Again, the development team gathers the initial requirements and builds the software prototype. The prototype is provided to the customer for evaluation. The customer gives us feedback, and our development team refines the prototype of the increment if it is not good enough. The software increment is a part of the whole working software system. Therefore quality methods should be applied to verify and validate software as well. In the case the increment is good enough – it is deployed.

Now the team proceeds with the next subsystem, or the process finishes if all subsystems are implemented.

7 slide

The extreme prototyping method was initially related to web projects—the mockups of user interfaces were created in Html and provided for customers to validate it. Nowadays, we can achieve similar results for many other platforms. Therefore this method is not dedicated exclusively to web projects.

So, let’s start with gathering the initial requirements. Now, as you will see, the prototype creation is split into three parts. First, we create a user interface design. Next – we add the behavior, so we can simulate how the software will work. And finally, we create the required services that implement that behavior. Note that the customer should not wait until the development team creates the working prototype. The customer can evaluate the design and behavior before we start implementing the required functionality. It allows saving a lot of time.

After the software becomes adequate, we can deploy it in increments or as a whole.

8 slide

Now you will see a video that demonstrates an “Invision” prototyping tool. You will get an idea about some tools that are used in extreme prototyping.

9 slide

The model has many advantages to the iterative models. There is quite an intensive customer involvement. Even more intensive when compared to Agile models. The early determination of user needs sometimes allows us to build software faster. Because of frequent customer involvement, the missing functionalities can be easily figured out. Also, this model embraces the changes as the refinements are a part of a model.

10 slide

On the other hand – there are challenges as well. This model is a risky one, and it has difficulties in planning because of the uncertain number of iterations. Also, if the team focuses on prototyping, it can lead to overlooking better solutions. If we use rapid prototyping, developers may end up with sub-optimal solutions because of the rush. The important thing is that the system structure tends to degrade if the team uses evolutionary prototyping. For example, at some point, the team might find out that they have chosen the wrong architecture to accommodate newly discovered requirements. Prototypes are not documented very well because of its nature. And sometimes, there are issues with customers who do not understand prototyping and confuse a prototype with a finished system. Usually, there is a long way from the prototype to the complete software.

11 slide

In the beginning, I said that this model is not used frequently. Also, we have reviewed several drawbacks of this model. Still, it doesn’t mean that this model is not useful. Let’s see when to use this model. As the name implies, we should use this model when facing uncertain things such as obscure or unstable requirements, new technologies, or other. The prototyping allows removing such obstacles in the early phases of a project.

Furthermore, it is an excellent model to demonstrate the technical feasibility or proof of a product’s concept. It is good to prove that some products could be created or vice versa. For example, there are popular events called “hackathons“. Typically, there are given three days to develop the prototype and demonstrate an idea of the product. The best ideas and prototypes are rewarded, and the creators are encouraged to develop their idea. Many games are born in such a way.

As mentioned before, this model fits suitable for developing user interfaces, high technology-intensive systems, systems with big complexity.

This model is popular in scientific institutions such as research institutes or universities. The scientists create prototypes proving that some inventions can work. Later on, the prototypes are adopted by software companies through a technology transfer process. Larger companies themselves have R&D departments that have a similar purpose.

12 slide

To conclude, the prototyping model is an iterative model that can use incremental deployment. The main idea is to validate software to be created by using a working prototype. This model is good when we face uncertain factors such as unclear requirements or technologies. One of the best ways to solve it – is to use prototyping. On the other hand, this model is a risky one. It is challenging to plan the development of a project because of the uncertain number of iterations.

13 slide

Thank you for watching it. See you!