intetics
Where software concepts come alive

Three W's of using prototype in the software development life cycle

www.intetics.com

What is a software prototype model? Why could it be useful in the software development projects? When should you prefer to use this approach and when it could be unnecessary?

Software development life cycle (SDLC) consists of several stages: requirements gathering, design, development, testing, maintenance. The overall success of the project appreciably depends on the first step of this process where the customer, product owner and the project team should formulate and agree the set of requirements to the software product. However, in reality these requirements are not clear and complete in many projects. It's not surprising as in software industry we often try to change the way we execute some processes, invent something new, make the predictions and assumptions. So, business owners and users shouldn't be expected to visualize the new software very clear. In these circumstances, the idea to produce prototype before the actual development looks very attractive.

WHAT is a software prototype model?

A prototype is a kind of a model or a simulation of a real thing. In software development, a software prototype model refers to building of a software application that displays the main functionality of the product under development. But at the same moment may not actually hold the exact logic of the original software and display all the features of the product. In different cases prototypes can take many forms. They can model the entire system with real data or just a few screens with sample data. The prototype could be created only for using on the requirements gathering and design stages or could become the part of delivered product.

There are several ways of how prototypes could be classified. One of the commonly used is the classification according to the fidelity of the prototype:

- A low-fidelity prototype is an illustrative one that could be quickly produced and has the main purpose to represent the screens mockups and demonstrate the main business scenarios of the future system.
- A middle-fidelity prototype has the main purpose to partially simulate the future system functionality, but usually doesn't use real data.
- A high-fidelity prototype is fully interactive, simulate much of the future system's functions. In some cases a high-fidelity prototype could use the real data (or database) or could produce model that become the part of the future real system.

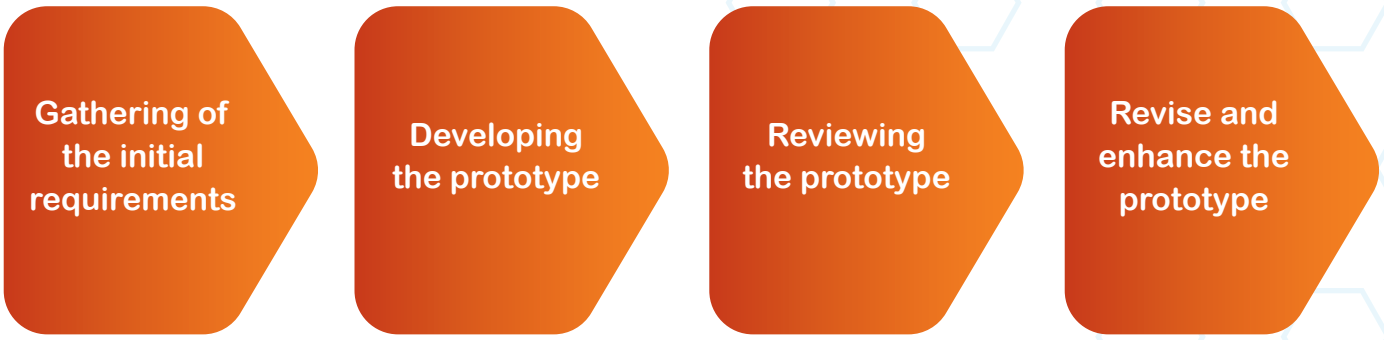
Another approach is to classify the prototypes by two major types:

- Throwaway prototypes that could be used on the early stages of the project and will be discarded and not used in the actual development of the product.
- Evolutionary prototypes that could become the part of the future product.

No matter what type of the prototype you are going to create there are some essential steps of its design.

intetics
Where software concepts come alive

The stepwise approach to design a software prototype:



- Gathering the initial requirements:

This step expects gathering and understanding the main product requirements including the requirements of the user interfaces. So, some details, accessory functions could be ignored at this stage. The basic idea of the software prototype model is that instead of freezing all the requirements before the design and development process the project team can make the prototype based on the currently known requirements.

- Developing the prototype:

The prototype is developed on this stage: the key features of the future product showcased and user interfaces are provided. It gives the customers and maybe end users a possibility to try the most valuable features and to follow the main business flow using the user interface of the product.

- Reviewing the prototype:

In this stage prototype should be demonstrated to the customer, product owner and other stakeholders of the project. It helps the customer to evaluate developer's proposal and try to use some functions of the future product before implementation.

It gives the possibility:

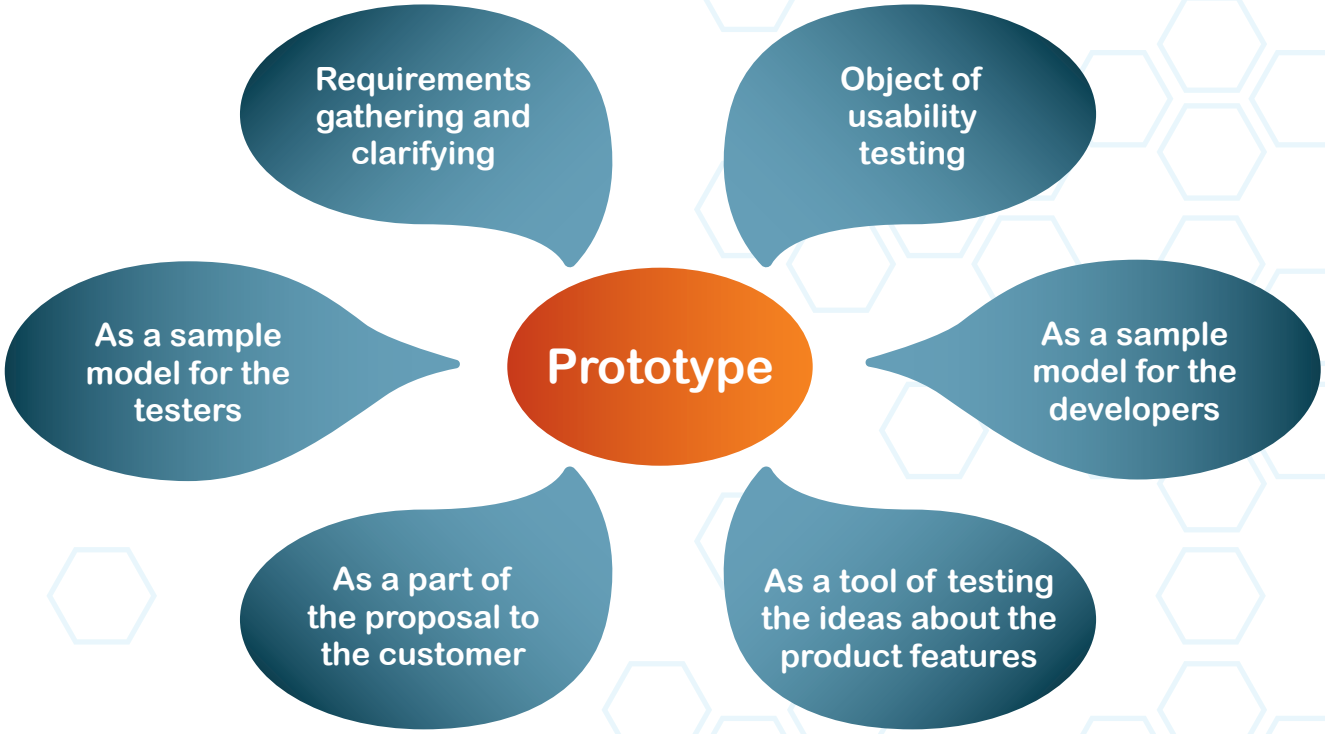
- to get the customers' or even end users' feedback;
- to test the ideas;
- to run usability tests (in some cases).

- Revise and enhance the prototype:

All the feedback, comments, propositions after the prototype's demonstration should be collected and analyzed. The result of the prototype analysis helps the development team to understand what exactly is expected from the product under development. The main idea is to better clarify and complement initial requirements. Also some changes could be accepted in this stage and some requirements could be added or eliminated from the product scope.

WHY the prototype model could be useful?

Prototype usually is created on the earliest stages of the development and could be used in different cases and on several stages of the SDLC from the first stage of the requirements gathering to the later stages of the development and testing. Also, software development companies could use the prototype as a part of the proposal to the customer. There are main use cases of the prototype in the SDLC.



Prototype model could be very useful because it gives many advantages comparing with traditional approach.

Advantages:

- Users' involvement is increased;
- Customers' feedback is available at the early stages;
- Missing or redundant functionality could be identified quicker;
- Confusing or implicit requirements can be clarified and agreed;
- It could give some benefits at the stage of preparing the proposal or tender documentation and contract signing;
- Time and money for actual development could be reduced because of agreed requirements and efficient communications with customer and development team;
- The product quality could be increased because of good understanding the requirements by all development team (analysts, developers, testers).

The list of advantages looks great, isn't it? But do not forget that this approach (as all the things in our real world) also has some disadvantages.

The list of advantages looks great, isn't it? But do not forget that this approach (as all the things in our real world) also has some disadvantages.

Disadvantages:

- The efforts of investing in prototype could be more than the values of the results of its analysis;
- The risks of expanding the first project scope is very high;
- Customers may be confused and disappointed because their expectation to the prototype are too high (for example, they expect that all features of the product will work correctly);
- Developers may try to reuse prototype to create the real product even when it is technically not optimal realization;
- Time and money for actual development could be increased because of inefficient way of building and analyzing the prototype.

So, this approach is typically used in cases when the decision should be taken very carefully so that the time, money and efforts spent in prototype development can increase the project's final values.

WHEN you should prefer to use this approach?

Different projects need different approaches to the software development. Choosing right approach could be the keystone of success. Therefore, the decision about using prototype model should be taken very carefully. Not all types of the projects need using this approach. Here you can find some cases of the projects in which using the prototypes could be useful or redundant.

Some cases when using prototype could be useful

Case	Reasons for using a prototype
You are going to create a new system with a lot of features and integrations with other systems.	The prototype allows to make experiments, simulating system behavior, making changes in the initial requirements at the early stages of the software life cycle.
Your product needs various interactions with the end users. End users are available.	In this case, the prototype could be very useful because it allows involving the end users for the development process, clarifying the requirements.
You project has very tight deadlines and there is no time to make researches. Requirements are gathered but there is no possibility to make their clarification.	The Prototype can help to demonstrate the key features quickly and make the decision about their implementation.
The project includes the automation of a complicated business process and has a very long development cycle.	In this case, the prototype can demonstrate the various stages of the process and gives users, business owners and development team something to work with.

Some cases when using a prototype could be redundant

Case	Reasons for choosing other development models
You are going to make some changes in the existing system.	The behavior of the system is well known and there is less uncertainty in the project requirements.
Your product needs various interactions with the end users… but…end users are unavailable.	The prototype could be very useful because you want to have the users' feedback. However, if you couldn't demonstrate your prototype and communicate with end users then this step in the SDLC would be waste of time and money.
You project is technical and has not a lot of interactions with users.	In such case the prototype will be useless.
You deal with short project. The requirements s pecification is available.	Start the development. Don't make unnecessary steps.

Conclusion.

Will your project benefit from using a prototype? We hope that our answers of this three W's questions will help you to make a right decision about choosing this approach. It could be extremely useful for some projects, or pointless for others.

Intetics team will be happy to contribute to your project. We have a lot of experience of running different types of the projects and will be able to help you to choose the most efficient way how to build your software product.

Intetics Inc. | 10001 Tamiami Trail N, Suit 114. Naples, FL 34108 | US: +1-239-217-4907 | Email: contact@intetics.com

About Intetics

Intetics is a leading global technology company focused on creation and operation of distributed professional teams for custom software development, software testing, systems integration, and data processing. Intetics is the pioneer of Offshore Dedicated Teams and the inventor of Remote In-Sourcing, which allows clients to create their ideal IT teams most efficiently. Intetics has broad industry experience, deep software engineering expertise, an outstanding quality management platform and an unparalleled methodology for talent recruitment, team building and talent retention that guarantee that clients receive exceptional results for their software applications and data processing projects. At Intetics, our outcomes do not just meet clients' expectations, they have been exceeding them for our two decades in business.

Intetics is ISO 9001 (quality) and ISO 27001 (security) certified and Microsoft and Oracle Gold Partner. The company's innovation and growth achievements are reflected in winning prestigious Inc 5000, Software 500, Chicago Innovation, CRN 100, Deloitte Technology Fast 50, European IT Excellence and Best European BPO awards, and inclusion into Top 100 Global Service Providers and Top 100 Outsourcing Companies lists.