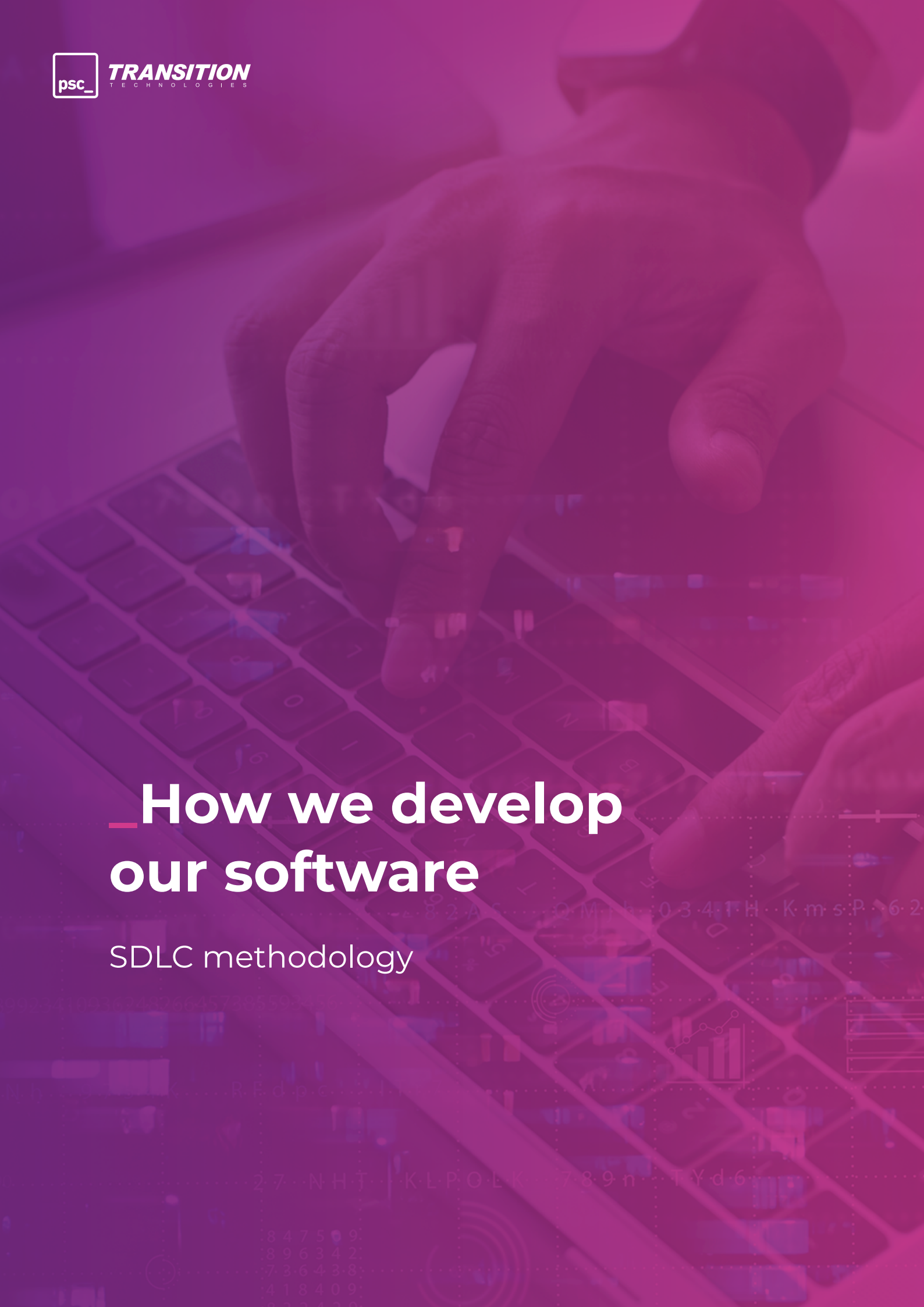


psc_ TRANSITION TECHNOLOGIES

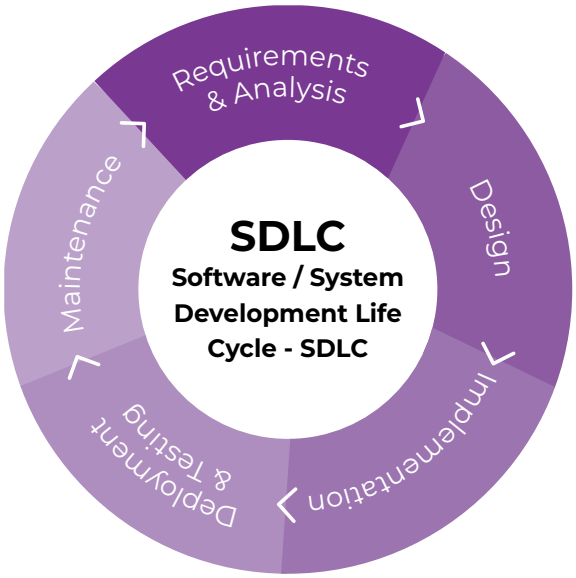
_How we develop our software

SDLC methodology

_SDLC introduction

In order to give you the opportunity to understand how we develop software let's start with a quick introduction on the Software Development Lifecycle (SDLC).

SDLC is a skeleton of a software development process that consists of different phases that should be involved. Further, you will learn more on our approach to each of those phases, but let's have a quick look on SDLC methodologies complementing the development process.



_SLDC methodologies

Introduction

Having the bricks above one could say that they already form a process to follow. Phases could be simply executed one after another, just after completion of the previous phase. That's true. The methodology is called Waterfall.

However in TTPSC we believe that the easiest is not always the best approach. Having delivered over hundred projects we've discovered enough to say that there are other methodologies that facilitate software development much better. Our favorite is Scrum coming from an Agile approach.

Why we choose Agile?

In the modern world, we can be sure of only one thing: it is change.

Technology, business and even our customer's needs change rapidly. Very often projects take months or even years. During this period the surroundings can change dramatically – recent COVID-19 related situation is a great example. In waterfall approach a project that was designed in a very early phase can miss its destination completely. Not because the design was bad – simply the destination changed in the meantime.

AGILE allows us to change the direction when needed, very often without, or with very transparent, impact on the delivery timelines.

Scrum for customer satisfaction

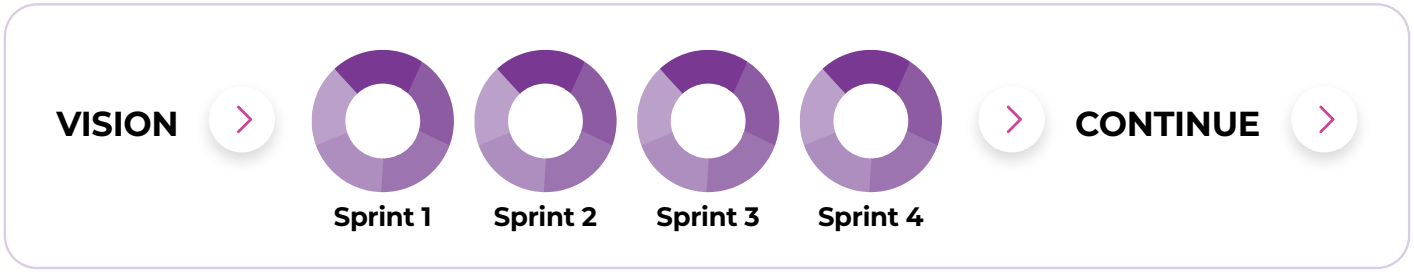
Scrum, one of agile methodologies, is based on customers' needs. Work is realized in so called sprints (iterations).

They are preceded by a planning ceremony that results in a sprint backlog being a plan on what will be realized during the sprint. Separating work into standalone sprints allows mitigation of

one of the risks as developers are focused on upcoming work and not the entire product. The entire product vision is held in Product Backlog, which is transparent to every stakeholder of the product. Inside the sprint team is working on a specific chunk of product's functionality, but for that chunk all phases are involved. That means that after a sprint, the delivered product increment has been designed, implemented, tested and integrated with already existing functionalities.

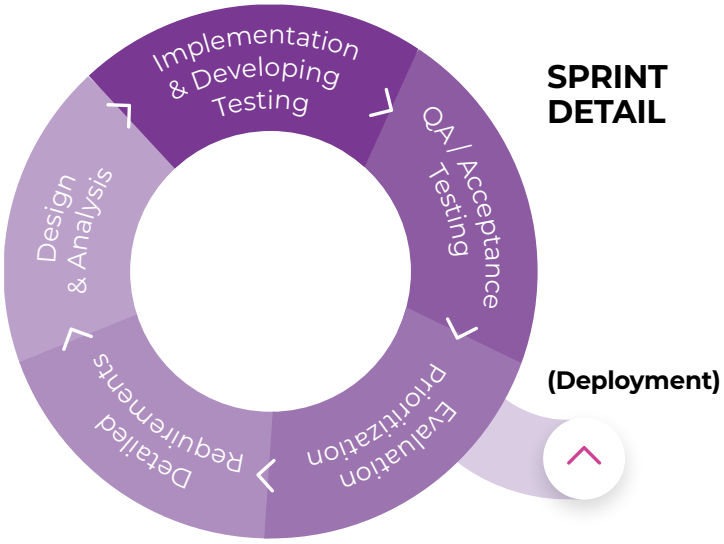
After the sprint the increment is evaluated during a dedicated review meeting. This gives the customer an opportunity to inspect what has been done so far and give feedback.

Based on this feedback next iterations are planned. In order to constantly look for possible improvements in the process a retrospective meeting is also held, on which the team decides which element could be improved and plans on how to do that.



Getting back to customer's needs, sprint length, ceremonies and meetings are aligned with other teams (if there are such), as well as any steps required by customer's process, so that our client does not need to perform any additional work to integrate the increment of the product.

Well implemented scrum allows teams to switch directions in a fluent way, without disturbing the stable delivery rhythm.



This approach also benefits the clients themselves: they can see an increment of their ordered product in constant cycles, making educated decisions on the next steps. We hope you also like it.

_This is how we do it in every phase

Requirements and analysis

The approach will depend on the needs of our customers.

Some have their Product Owners that will work with our teams and serve them well described backlog items, we take care about the rest. The others have their expectations about the software described in details and just need our Product Owner(s) to translate the expectations to the backlog items and drive the team to deliver according to the specification.

Finally, we have customers having great ideas about a product they want to have but, they need guidance on how to make their vision become a real product. With such customers our Product Owners can show all their skills.

After initial conversation with customer, the first step is to identify the stakeholders. Usually they can be divided into two groups: internal and external. Internal stakeholders are usually people from different parts of the company that can provide insights on the product (e.g. legal, compliance etc.) or will be impacted by it (e.g. operations, internal users). External stakeholders are usually the end users of the product that is going to be built.

When Product Owner gathers requirements for the product to be built, he needs to remember that although product is for the end users and it needs to fulfill their needs and expectations, it also needs to meet goals of the organization. That is why identifying and working with stakeholders, both internal and external, is so important. After initial conversation PO builds a vision of the product and shares it with the stakeholders. Listening and involving customers in this phase is crucial, because without their input PO will make assumptions, and not educated guesses.

One of the techniques used to show the product is **User Story Mapping.**

It shows the routes of different types of users through the product and what they can achieve by completing a given route. After such exercise, PO, stakeholders and the development team have a common understanding of what is to be built. At this point usually gathered information is stored in JIRA tool by Atlassian, that Transition Technologies PSC commonly uses for tracking requirements and further the complete status of the project.

Of course the Product Owner is not left alone in this phase. It is often the case that in order to analyze some of the requirements in depth, an expertise from a specific field will be required. For example some non-functional requirements usually engage our experts when it comes to security or performance. These often aren't obvious as when thinking about a product, functionality is what customers are usually interested in the most.

We are always happy to use our experience to support you in specifying your needs. Keep in mind that as we work in cycles, this and all following phases are continuously repeated until product is completed.

Design (Architecture & UX/UI)

The fact that we work in an agile approach does not mean we proceed with the implementation without a good design.

This relates to both UX/UI design, specifying how the software should look like and behave as well as the design of the software's architecture describing software's elements and relations between them. While creating user interface/experience design we start from the scratch with our clients discovering the main purpose of the software. We establish the essential business objectives and set priorities that defines project's scope.

At stage of research we want to get awareness of our customers and final users as much as possible. The goal of the research stage is to gather all qualitative and/or quantitative data that will make us aware of real needs and background that will allow us to develop beneficial product.

Then we go through strategy part. In comparison to other similar products on the market we define how we want to add value to product and how much it suits our customer's needs, problems and goals. By modeling and prototyping we determine application's content and scope, main features and functionalities required for the project to succeed. We create low and high fidelity mockups that will be close to final product. No matter on which stage we are with development, we should test our product as soon as possible, but this phase is crucial for collecting feedback about our increment and recognizing possible dysfunctionalities.

Last but not least about the user experience – **in Transition Technologies PSC we believe that products we develop should be accessible also by the users with disabilities.**

Why should your product be inaccessible for example by a visually impaired person?

We also see no reason. That is why our products are compliant with Web Content Accessibility Guidelines (WCAG) by default. As a nice addition they will meet legal regulations about the accessibility.

When it comes to architecture, we are confident that a strong architectural skill in the team can save a lot of development work, and make resulting software easier to maintain. That is why apart from the naturally expected experience we assure that our architects are well trained in recent trends.

While planning the architecture they start with the analysis. They do not stick to single approach and are open to both monolith as well as microservices architecture. In every approach our specialists remember to divide responsibilities of modules (or microservices), so that the single responsibility principle is followed. They do not create modules that are tightly coupled. This allows replacement of whole components and makes them interchangeable, that gives you a lot of flexibility in case the plan changes.

During architecture design we follow rules such as DDD (Domain-Driven Design) which allow us to deeply understand the problem our customer is facing and address it correctly. While making the decision on which technology to use we also asses the real needs. Although in TT PSC we have developed a proven set of “first choice” technologies and frameworks that helps us in a decision making process. We are not just re-using it in each case, instead we are looking for the best one for a specific challenge.

Implementation

This is where the magic (coding) happens. Over the years developers at TT PSC have gained a lot of experience with different languages, frameworks and tools.

That, combined with up to date knowledge on the latest technologies and the awareness of security aspects made it pretty easy for us to introduce a lot of standardization in the phase where code is being written.

Our engineers put a lot of emphasis on the code‘s quality, hence we have the rules on how the code should be written (coding standards) for different programming languages. In order to assure that the code we produce follows the rules, we are using tools like SonarQube for static code analysis (it does even more, as it can for example indicate potential vulnerabilities present in the code).

In addition, while implementing the code we cover it with automatic unit tests. They are used to verify that given expected conditions/parameters, execution of a unit of the code (like method) will produce the desired result. This helps a lot when we later modify the existing code – unexpected change in the code behavior will be detected automatically and developer will be notified about that. Finally, before the code becomes part of the product it, is a subject to code review by another developer – meaning that at least two engineers confirm it meets our standards.

Deployment & Testing

We benefit as much as possible from Continuous Integration/Continuous Deployment practices and tools to introduce automated testing at early stage.

Thank to that, code is frequently integrated and deployed, what allows us to discover any issue (bug) rapidly and provide necessary fixes in the next integration cycle. It goes hand in hand with philosophy of frequent inspection and adaptation.

Having deployment done automatically and regularly executed, makes the deployment to production environment much simpler and less challenging then it could be.

Once the code is deployed we can further verify the quality (being inseparable part of valuable products). Testing for us is not a separate phase. It's a part of everyday work, build into provided features.

To achieve it, completed code does not await a testing phase for bugs to be identified. All quality checks occur during an iteration and they are performed by cross-functional teams.

Testing is an ongoing activity, executed and owned by all team members. Therefore, we include best practices like unit tests, automated test, TDD (Test Driven Development) or BDD (Behaviour Driven Development), etc.

While testing we cannot forget about the security aspects. This becomes extremely important nowadays when number of cyberattacks is growing very fast.

In response to that, we have a number of qualified specialists in the area, holding the title of Certified Ethical Hacker. They are with us to discover possible vulnerabilities in the application (and allow developers to remove them) before the software can be attacked from the outside.

Maintenance

Once the software is delivered to our customer(s) we don't leave them on their own.

TT PSC is ready and willing to help you using your product or develop it further by adding new functionalities. When you don't need more, of course you will be given a warranty, and our support (even in different time zones).

_Would you like to know more on how we work? Just reach out to us!

_Contact us!

Transition Technologies PSC

contact@ttpsc.com
www.ttpsc.com