psc_ TRANSITION TECHNOLOGIES

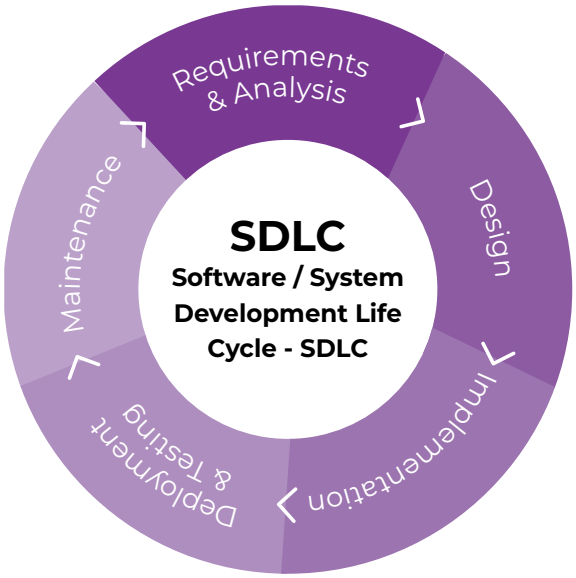
_Comment nous développons nos logiciels

Méthodologie SDLC

_Introduction au SDLC

Afin de vous permettre de comprendre comment nous développons des logiciels, commençons par une brève introduction au cycle de vie du développement logiciel (SDLC).

Le SDLC est le squelette d'un processus de développement de logiciels qui consiste en différentes phases qui devraient être impliquées. Vous en apprendrez davantage sur notre approche de chacune de ces phases, mais jetons un coup d'œil rapide sur les méthodologies SDLC qui complètent le processus de développement.



_Méthodologies SLDC

Introduction

On peut dire que les briques ci-dessus forment déjà un processus à suivre. Les phases pourraient être simplement exécutées l'une après l'autre, juste après l'achèvement de la phase précédente. C'est vrai. La méthodologie s'appelle Waterfall (chute d'eau).

Cependant, chez TTPSC, nous pensons que la méthode la plus simple n'est pas toujours la meilleure. Après avoir réalisé plus d'une centaine de projets, nous en avons découvert suffisamment pour dire qu'il existe d'autres méthodologies qui facilitent le développement de logiciels de manière bien plus efficace. Notre préférée est Scrum, issue d'une approche Agile.

Pourquoi choisir Agile ?

Dans le monde moderne, nous ne pouvons être sûrs que d'une chose : c'est le changement.

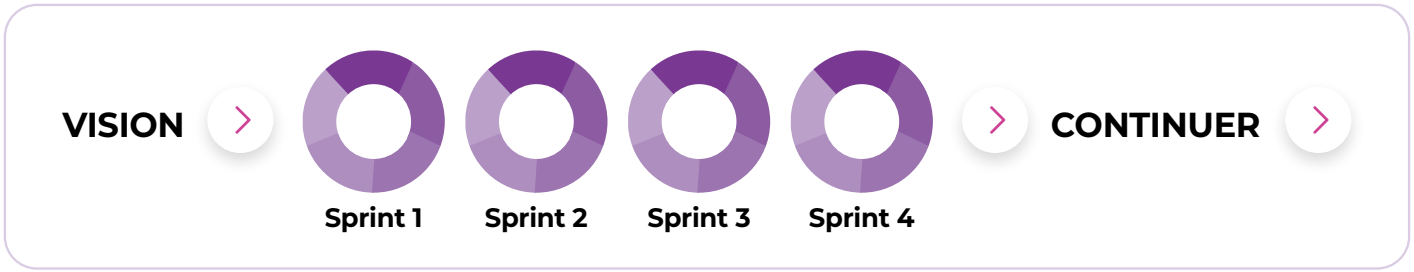
La technologie, les entreprises et même les besoins de nos clients évoluent rapidement. Très souvent, les projets prennent des mois, voire des années. Au cours de cette période, l'environnement peut changer radicalement - la situation récente liée au COVID-19 en est un excellent exemple. Dans l'approche en cascade, un projet qui a été conçu dans une phase très précoce peut manquer complètement sa destination. Ce n'est pas parce que la conception était mauvaise, mais simplement parce que la destination a changé entre-temps.

AGILE nous permet de changer de direction lorsque c'est nécessaire, très souvent sans impact, ou avec un impact très transparent, sur les délais de livraison.

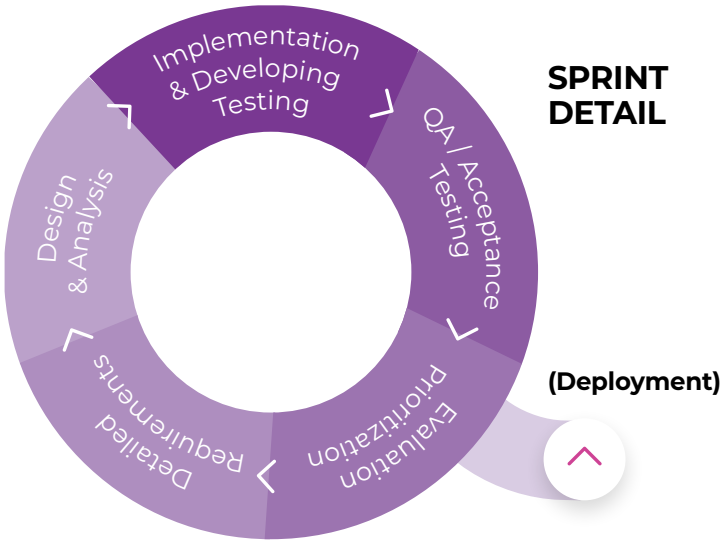
Scrum pour la satisfaction du client

Scrum, l'une des méthodologies agiles, est basée sur les besoins des clients. Le travail est réalisé en "sprints" (itérations).

Ces sprints sont précédés d'une cérémonie de planification qui débouche sur un carnet de commandes (sprint backlog), c'est-à-dire un plan de ce qui sera réalisé au cours du sprint. La séparation du travail en sprints autonomes permet d'atténuer l'un des risques, car les développeurs se concentrent sur le travail à venir et non sur l'ensemble du produit. L'ensemble de la vision du produit se trouve dans le carnet de commandes, qui est transparent pour toutes les parties prenantes du produit. Au cours d'un sprint, l'équipe travaille sur une partie spécifique de la fonctionnalité du produit, mais pour cette partie, toutes les phases sont impliquées. Cela signifie qu'après un sprint, l'incrément de produit livré a été conçu, mis en œuvre, testé et intégré aux fonctionnalités déjà existantes. Après le sprint, l'incrément est évalué au cours d'une réunion de révision spécifique. Le client a ainsi l'occasion d'inspecter ce qui a été fait jusqu'à présent et de donner son avis. Sur la base de ces commentaires, les prochaines itérations sont planifiées. Afin de rechercher en permanence des améliorations possibles dans le processus, une réunion rétrospective est également organisée, au cours de laquelle l'équipe décide des éléments qui pourraient être améliorés et planifie la manière d'y parvenir.



Pour revenir aux besoins du client, la durée du sprint, les cérémonies et les réunions sont alignées sur celles des autres équipes (s'il y en a), ainsi que sur toutes les étapes requises par le processus du client, de sorte que notre client n'a pas besoin d'effectuer un travail supplémentaire pour intégrer l'incrément du produit. Une bonne mise en œuvre de scrum permet aux équipes de changer de direction de manière fluide, sans perturber le rythme de livraison stable.



Cette approche profite également aux clients eux-mêmes : ils peuvent voir un incrément du produit qu'ils ont commandé dans des cycles constants, ce qui leur permet de prendre des décisions éclairées sur les prochaines étapes. Nous espérons que vous l'apprécierez également.

_Voici comment nous procédons à chaque phase

Exigences et analyse

L'approche dépend des besoins de nos clients.

Certains ont leurs Product Owners qui travailleront avec nos équipes et leur fourniront des éléments de backlog bien décrits, nous nous occupons du reste. Les autres ont leurs attentes concernant

le logiciel, décrites en détail, et ont juste besoin de notre (nos) Product Owner(s) pour traduire les attentes en éléments du carnet de commandes et conduire l'équipe à livrer conformément aux spécifications.

Enfin, nous avons des clients qui ont de grandes idées sur un produit qu'ils souhaitent avoir, mais qui ont besoin d'être guidés pour que leur vision devienne un produit réel. Avec de tels clients, nos Product Owners peuvent montrer toutes leurs compétences.

Après la conversation initiale avec le client, la première étape consiste à identifier les parties prenantes. En général, elles peuvent être divisées en deux groupes : les parties prenantes internes et les parties prenantes externes. Les parties prenantes internes sont généralement des personnes issues de différentes parties de l'entreprise qui peuvent fournir des informations sur le produit (par exemple, les services juridiques, la conformité, etc.) ou qui seront affectées par celui-ci (par exemple, les opérations, les utilisateurs internes). Les parties prenantes externes sont généralement les utilisateurs finaux du produit qui va être construit.

Lorsque le propriétaire du produit recueille les exigences relatives au produit à construire, il doit se rappeler que, bien que le produit soit destiné aux utilisateurs finaux et qu'il doive répondre à leurs besoins et à leurs attentes, il doit également atteindre les objectifs de l'organisation. C'est pourquoi il est si important d'identifier les parties prenantes, tant internes qu'externes, et de travailler avec elles. Après une première conversation, l'OP élabore une vision du produit et la partage avec les parties prenantes. Il est essentiel d'écouter et d'impliquer les clients dans cette phase, car sans leur contribution, l'OP fera des hypothèses, et non des suppositions éclairées.

L'une des techniques utilisées pour présenter le produit est la cartographie de l'histoire de l'utilisateur **(User Story Mapping).**

Il montre les itinéraires des différents types d'utilisateurs à travers le produit et ce qu'ils peuvent réaliser en suivant un itinéraire donné. Après cet exercice, le PO, les parties prenantes et l'équipe de développement ont une compréhension commune de ce qui doit être construit. À ce stade, les informations recueillies sont généralement stockées dans l'outil JIRA d'Atlassian, que Transition Technologies PSC utilise couramment pour suivre les exigences et l'état d'avancement du projet.

Bien entendu, le Product Owner n'est pas seul dans cette phase. Il arrive souvent que l'analyse approfondie de certaines exigences nécessite l'intervention d'un expert dans un domaine spécifique. Par exemple, certaines exigences non fonctionnelles font généralement appel à nos experts lorsqu'il s'agit de sécurité ou de performance. Celles-ci ne sont souvent pas évidentes, car lorsqu'on pense à un produit, c'est la fonctionnalité qui intéresse généralement le plus les clients. Nous sommes toujours heureux d'utiliser notre expérience pour vous aider à spécifier vos besoins. N'oubliez pas que, comme nous travaillons par cycles, cette phase et toutes les suivantes se répètent continuellement jusqu'à ce que le produit soit achevé.

Conception (Architecture & UX/UI)

Le fait que nous travaillions dans une approche agile ne signifie pas que nous procédons à la mise en œuvre sans une bonne conception.

Cela concerne à la fois la conception UX/UI, qui spécifie comment le logiciel doit se présenter et se comporter, et la conception de l'architecture du logiciel, qui décrit les éléments du logiciel et les relations qu'ils entretiennent entre eux. Lors de la conception de l'interface/expérience utilisateur, nous partons de zéro avec nos clients pour découvrir l'objectif principal du logiciel. Nous établissons les objectifs commerciaux essentiels et fixons les priorités qui définissent la portée du projet.

Au stade de la recherche, nous voulons connaître autant que possible nos clients et nos utilisateurs finaux. L'objectif de la phase de recherche est de rassembler toutes les données qualitatives et/ou quantitatives qui nous permettront de prendre conscience des besoins réels et du contexte qui nous permettra de développer un produit avantageux.

Nous passons ensuite à la partie stratégique. En comparaison avec d'autres produits similaires sur le marché, nous définissons comment nous voulons ajouter de la valeur au produit et dans quelle mesure il répond aux besoins, aux problèmes et aux objectifs de nos clients. Grâce à la modélisation et au prototypage, nous déterminons le contenu et la portée de l'application, ainsi que les principales caractéristiques et fonctionnalités nécessaires à la réussite du projet.

Nous créons des maquettes haute et basse fidélité qui seront proches du produit final. Quelle que soit l'étape du développement, nous devrions tester notre produit dès que possible, mais cette phase est cruciale pour recueillir des commentaires sur notre incrément et reconnaître les éventuels dysfonctionnements.

Dernier point, mais non des moindres, concernant l'expérience de l'utilisateur - **chez Transition Technologies PSC, nous pensons que les produits que nous développons doivent être accessibles également aux utilisateurs handicapés.**

Pourquoi votre produit devrait-il être inaccessible, par exemple, à une personne malvoyante ?

Nous ne voyons pas non plus de raison. C'est pourquoi nos produits sont conformes par défaut aux lignes directrices pour l'accessibilité des contenus web (WCAG). De plus, ils sont conformes aux réglementations légales en matière d'accessibilité.

En ce qui concerne l'architecture, nous sommes convaincus qu'une forte compétence architecturale au sein de l'équipe peut permettre d'économiser beaucoup de travail de développement et rendre le logiciel résultant plus facile à maintenir. C'est pourquoi, en plus de l'expérience naturellement attendue, nous nous assurons que nos architectes sont bien formés aux dernières tendances.

Lors de la planification de l'architecture, ils commencent par l'analyse. Ils ne s'en tiennent pas à une seule approche et sont ouverts à l'architecture monolithique comme à l'architecture microservices. Dans chaque approche, nos spécialistes n'oublient pas de répartir les responsabilités des modules (ou microservices), de manière à respecter le principe de la responsabilité unique. Ils ne créent

pas de modules étroitement couplés. Cela permet de remplacer des composants entiers et de les rendre interchangeables, ce qui offre une grande flexibilité en cas de changement de plan. Lors de la conception de l'architecture, nous suivons des règles telles que le DDD (Domain-Driven Design) qui nous permettent de comprendre en profondeur le problème auquel notre client est confronté et de l'aborder correctement. Lorsque nous décidons de la technologie à utiliser, nous évaluons également les besoins réels. Bien que TT PSC ait développé un ensemble éprouvé de technologies et de cadres de "premier choix" qui nous aident dans le processus de prise de décision, nous ne nous contentons pas de les réutiliser dans d'autres projets. Nous ne nous contentons pas de les réutiliser dans chaque cas, mais nous recherchons la meilleure solution pour un défi spécifique.

Mise en œuvre

C'est ici que la magie (le codage) opère. Au fil des ans, les développeurs de TT PSC ont acquis une grande expérience des différents langages, cadres et outils.

Cette expérience, combinée à des connaissances actualisées sur les dernières technologies et à la prise de conscience des aspects de sécurité, nous a permis d'introduire assez facilement une grande normalisation dans la phase d'écriture du code.

Nos ingénieurs accordent une grande importance à la qualité du code, c'est pourquoi nous avons des règles sur la manière dont le code doit être écrit (normes de codage) pour les différents langages de programmation. Afin de nous assurer que le code que nous produisons respecte ces règles, nous utilisons des outils tels que SonarQube pour l'analyse statique du code (il fait même plus, puisqu'il peut par exemple indiquer les vulnérabilités potentielles présentes dans le code).

En outre, lors de la mise en œuvre du code, nous le couvrons de tests unitaires automatiques. Ils sont utilisés pour vérifier qu'étant donné les conditions/paramètres attendus, l'exécution d'une unité du code (comme une méthode) produira le résultat souhaité. Cela est très utile lorsque nous modifions ultérieurement le code existant - un changement inattendu dans le comportement du code sera détecté automatiquement et le développeur en sera informé. Enfin, avant que le code ne soit intégré au produit, il fait l'objet d'une révision par un autre développeur, ce qui signifie qu'au moins deux ingénieurs confirment qu'il répond à nos normes.

Déploiement et tests

Nous profitons autant que possible des pratiques et des outils d'intégration et de déploiement continus pour introduire des tests automatisés à un stade précoce.

Grâce à cela, le code est fréquemment intégré et déployé, ce qui nous permet de découvrir rapidement tout problème (bug) et de fournir les corrections nécessaires lors du prochain cycle d'intégration. Cela va de pair avec la philosophie d'inspection et d'adaptation fréquentes.

Le fait que le déploiement soit effectué automatiquement et exécuté régulièrement rend le déploiement dans l'environnement de production beaucoup plus simple et moins difficile qu'il ne pourrait l'être.

Une fois le code déployé, nous pouvons en vérifier la qualité (élément indissociable des produits de valeur). Pour nous, le test n'est pas une phase distincte. Il fait partie du travail quotidien, intégré dans les fonctionnalités fournies. Pour y parvenir, le code achevé n'attend pas une phase de test pour que les bogues soient identifiés. Tous les contrôles de qualité ont lieu au cours d'une itération et sont effectués par des équipes interfonctionnelles.

Les tests sont une activité permanente, exécutée et prise en charge par tous les membres de l'équipe. C'est pourquoi nous incluons les meilleures pratiques telles que les tests unitaires, les tests automatisés, le TDD (Test Driven Development) ou le BDD (Behaviour Driven Development), etc. Lors des tests, nous ne pouvons pas oublier les aspects liés à la sécurité. Cela devient extrêmement important de nos jours, alors que le nombre de cyberattaques augmente très rapidement.

C'est pourquoi nous disposons d'un certain nombre de spécialistes qualifiés dans ce domaine, titulaires du titre de Certified Ethical Hacker. Ils sont à nos côtés pour découvrir les éventuelles vulnérabilités de l'application (et permettre aux développeurs de les supprimer) avant que le logiciel ne puisse être attaqué de l'extérieur.

Maintenance

Une fois le logiciel livré à nos clients, nous ne les laissons pas seuls.

TT PSC est prêt à vous aider à utiliser votre produit ou à le développer en y ajoutant de nouvelles fonctionnalités. Si vous n'avez pas besoin de plus, vous bénéficierez bien sûr d'une garantie et de notre assistance (même dans des fuseaux horaires différents).

_Vous souhaitez en savoir plus sur nos méthodes de travail ? N'hésitez pas à nous contacter !

_Entrons en contact

Transition Technologies PSC

contact@ttpsc.com
www.ttpsc.com