**Trousse de déploiement**

 **Mise en oeuvre du logiciel**

**Profil d’entrée**

|  |
| --- |
| **Remarque:**Ce document est la propriété intellectuelle de l’organisation des auteurs. Cependant, l’information de ce document peut être utilisée de manière libre. La distribution du document complet ou des parties individuelles est autorisée pour l’utilisation non commerciale pourvue que la notification suivante légale soit mentionnée : © École de technologie supérieureL’utilisation commerciale de ce document est strictement interdite. Ce document est distribué afin d’augmenter l’échange d’information technique et scientifique. Ce matériel est fourni «tel quel». L'auteur n’offre aucune garantie d'aucune sorte, explicite ou implicite, quant à toute question, y compris, mais sans s'y limiter, à la garantie de performance, à l'usage ou de qualité marchande, l'exclusivité, ou les résultats obtenus par l'utilisation du matériel. Les processus décrits dans cette trousse de déploiement ne visent pas à empêcher ou à décourager l’utilisation de processus additionnels dont les très petits organismes pourraient trouver utiles.  |

|  |  |
| --- | --- |
| **Auteurs** | G. Hernandez École de technologie supérieure (ÉTS), (Canada)W. Gonzalez *-* École de technologie supérieure (ÉTS), (Canada) |
| **Éditeur** | C. Y. LAPORTE – École de technologie supérieure (ÉTS), (Canada) |
| **Date de création**  | 9 août 2010 |
| **Dernière mise à jour** | 22 septembre 2012 |
| **Statut** | Ébauche |
| **Version** | 0.4 |

Historique des versions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date**(jj-mm-aaaa) | **Version** | **Auteur** | **Modification** |
| 09/08/2010 | 0.1 | G. HernandezW. Gonzalez | Traduction de la version anglaise |
| 1/12/2010 | 0.2 | W. Gonzalez | Mise à jour |
| 25/02/2011 | 0.3 | V. Ribaud | Relecture et corrections mineures |
| 04/07/2012 | 0.4 | C Laporte | Mise à jour |
|  |  |  |  |
|  |  |  |  |

Abréviations/acronymes

|  |  |
| --- | --- |
| **Abré./Acro.** | **Définitions** |
| TD | Trousse de déploiement – Un ensemble d’artefacts développés pour faciliter l’implémentation dans un très petit organisme, un ensemble de pratiques provenant d’un cadre de référence sélectionné. |
| TPO | Très petit organisme – une entreprise, une organisation, un département ou un projet de 25 personnes ou moins. |
| IL | Implémentation logicielle |
| ISO | Organisation internationale de normalisation |
| IEC/CEI | International Electrotechnical Commission/Commission Electrotechnique Internationale |
| TR | Technical Report |
| SDP | Structure de découpage du projet |
| VSE | Very Small Entity |
| WBS | Work Breakdown Structure |

Table des matières

[1. Description Technique 4](#_Toc336070569)

[But de ce document 4](#_Toc336070570)

[Pourquoi la mise en œuvre du logiciel est-elle importante ? 4](#_Toc336070571)

[Définitions 6](#_Toc336070572)

[Termes génériques (en ordre alphabétique) 6](#_Toc336070573)

[Termes spécifiques (en ordre alphabétique) 6](#_Toc336070574)

[3. Liens avec la norme ISO/IEC 29110 8](#_Toc336070575)

[4. Description des processus, activités, tâches, étapes, rôles et produits 9](#_Toc336070576)

[4.1. Les activités du processus de mise en œuvre du logiciel (IL) 9](#_Toc336070577)

[4.1.1. Activité : SI.1 L'initiation de la mise en œuvre du logiciel 10](#_Toc336070578)

[4.1.2. Activité : SI.2 L'analyse des exigences du logiciel 11](#_Toc336070579)

[4.1.3. Activité : SI.3 Détermination des composants logiciels 13](#_Toc336070580)

[4.1.4. Activité : SI.4 Construction du logiciel 14](#_Toc336070581)

[4.1.5 Activité : SI.5 Intégration et tests du logiciel 17](#_Toc336070582)

[4.1.6. Activité : SI.6 Livraison du produit 20](#_Toc336070583)

[4.2 Description des rôles 21](#_Toc336070584)

[4.3. Description des produits 22](#_Toc336070585)

[4.4. Description des artefacts 29](#_Toc336070586)

[5. Gabarits et outils 31](#_Toc336070587)

[5.1. Modèle pour l’initiation de l’implémentation logicielle 31](#_Toc336070588)

[5.2. Modèle pour l’analyse des exigences logicielles 31](#_Toc336070589)

[5.3 Modèle pour l’identification des composants logiciels 32](#_Toc336070590)

[5.4 Modèle pour la construction du logiciel 32](#_Toc336070591)

[5.5 Modèle de spécifications des tests 33](#_Toc336070592)

[5.6 Modèle pour la livraison du produit final 35](#_Toc336070593)

[6. Exemple du cycle de vie de la mise en œuvre du logiciel 38](#_Toc336070594)

[Exemple du cycle de vie de la mise en œuvre du logiciel 38](#_Toc336070595)

[7. Listes de vérification 39](#_Toc336070596)

[7.1 Liste de vérification des exigences du logiciel 39](#_Toc336070597)

[7.2 Liste de vérification pour l’identification de composants logiciels 39](#_Toc336070598)

[7.3 Liste de vérification pour la construction du logiciel 39](#_Toc336070599)

[7.4 Liste de vérification pour la livraison du produit final. 40](#_Toc336070600)

[8. Références 42](#_Toc336070601)

[9. Formulaire d’évaluation 43](#_Toc336070602)

# 1. Description Technique

## But de ce document

Cette trousse de déploiement (TD) soutient le profil d’entrée (*Entry*) tel que définit dans le document ISO/IEC TR 29110-5-1-1:2012 - Guide d'ingénierie et de gestion: Groupe de profil générique: Profil d'entrée [ISO/IEC 29110]. Le profil d’entrée est un profil du groupe des profils génériques. Le groupe des profils génériques est applicable aux très petits organismes (TPO) qui ne développent pas des logiciels critiques. Le groupe des profils génériques est composé de 4 profils : entrée (*Entry*), basique (*Basic*), intermédiaire (*Intermediate*) et avancé (*Advanced*). Le groupe des profils génériques n'implique pas un domaine spécifique d'applications. Le profil d’entrée est destiné aux TPO qui travaillent sur de petits projets (par exemple, un projet dont l’effort est d’un maximum de six mois-personnes et pour les TPO en démarrage (*start-up*). Le profil d’entrée fournit une base permettant une migration vers les processus du profil basique (*Basic*).

Une trousse de déploiement est un ensemble d’artefacts qui facilitent l’implémentation d’un ensemble de pratiques dans un TPO. Une TD n’est pas un modèle de référence de processus (c.à.d. qu’elle n’est pas prescriptive). Les éléments d’une TD sont : la description du processus, les activités, les tâches, les rôles et les produits, les gabarits, les check-lists, les exemples et les outils.

Le contenu de ce document n’est pas normatif, ce document est informatif.

Ce document est destiné à être utilisé par un TPO afin d’établir des processus pour mettre en œuvre une approche de développement ou une méthodologie, par exemple, agile, évolutive, itérative, développement dirigé par les tests, etc. sur la base des besoins de l'organisme ou du projet d'un TPO.

Le document ISO/IEC TR 29110-5-1-1 :2012 est disponible sans frais sur le site de l'ISO suivant:

<http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>

## Pourquoi la mise en œuvre du logiciel est-elle importante ?

La mise en oeuvre c’est la réalisation ou l’exécution d'un plan, une méthode ou une conception (*design*) pour faire quelque chose.

Dans un contexte des technologies de l'information, la mise en oeuvre englobe tous les processus impliqués dans l'obtention d’un nouveau logiciel ou d’un matériel informatique opérant correctement dans son environnement, y compris le développement, l'installation, la configuration, la mise en exploitation, le test et l’exécution des changements nécessaires. Le mot déploiement est quelquefois utilisé comme synonyme d’implémentation [crm.com].

Ce processus est une démarche systématique de la façon d'effectuer les activités et les tâches. Chaque activité a pour but d’atteindre des objectifs stipulés dans le cahier des charges et dont le but ultime est de réaliser le produit final avec tous les attributs exigés par le client avec un haut niveau de qualité.

L'importance des activités de la mise en œuvre du logiciel est décrite comme suit :

* L'activité d’initiation de la mise en œuvre du logiciel permet de préparer les membres de l’équipe de travail pour les activités et aussi que les outils nécessaires pour accomplir le travail,
* L'activité d’analyse des exigences logicielles est importante pour bien définir la portée du projet et pour identifier les fonctionnalités clés du système avec le client afin d'éviter des problèmes comme des fonctionnalités oubliées ou des exigences mal comprises.
* L'activité d’identification des composants logiciels est une phase clé dans un projet logiciel. Une architecture inadéquatement conçue est une recette pour l’échec. Le client ne réalisera pas le paiement si la conception ne correspond pas à ses exigences.
* L’activité de construction logicielle est une phase clé pour les programmeurs qui produiront des composants d'une façon systématique, cette approche peut être utilisée aussi pour construire des composants complexes.
* L'activité d’intégration logicielle et de test permet d'exécuter différents types de tests et d’identifier les erreurs qui doivent être corrigées par l'équipe de développement logicielle. Les différents types de tests sont exécutés à des moments différents.
* L’activité de livraison du produit final comprend différentes tâches, il ne devrait pas y avoir de surprise, ni aucun retard pour obtenir l'acceptation du produit à livrer. Autrement, le client ne réalisera pas le paiement au TPO.

#  Définitions

Dans cette section, le lecteur trouvera deux groupes de définitions. Le premier groupe définit les termes utilisés dans toutes les trousses de déploiement, c'est-à-dire, les termes génériques. Le deuxième groupe définit les termes utilisés dans cette trousse de déploiement, c'est-à-dire, les termes spécifiques.

## Termes génériques (en ordre alphabétique)

***Activité :*** ensemble de tâches liées. [ISO/IEC 12207].

***Artefact :*** produit tangible résultant d’un processus de développement des logiciels. (p. ex. les cas d’utilisation, plan du projet, document des exigences).

***Étape :*** Dans une trousse de déploiement, une tâche est décomposée en une suite d’étapes.

***Processus :*** Un ensemble d’activités corrélées ou en interaction, qui transforme des éléments d’entrée en éléments de sorite. [ISO/IEC 12207].

***Produit :*** Un ensemble d’information ou un livrable qui peut être produit (de manière non-obligatoire) par une ou plusieurs tâches. (p. ex. le document de conception, le code source).

***Rôle***: Une fonction définie, destinée à être exécutée par un membre de l’équipe du projet. Par exemple testeur, inspecteur, codeur. [ISO/IEC 24765]

***Sous-Tâche:*** Lorsqu’une tâche est complexe, elle est divisée en sous-tâches.

***Tâche:*** Une action requise, recommandée oupermise destinée à contribuer à l’accomplissement d’un ou plusieurs résultats d’un processus.[ISO/IEC 12207].

## Termes spécifiques (en ordre alphabétique)

***Accord***: La définition des termes et des conditions sous lesquelles une relation de travail sera conduite. [ISO/IEC 12207]

Analyse des exigences : Le processus d’étudier les besoins de l’utilisateur final pour parvenir à la définition des exigences logicielles d’un système, d’un matériel ou d’un logiciel. [ISO/IEC 24765]

***Composant*:** Un ensemble de services fonctionnels dans le logiciel, qui une fois implémenté, représente un ensemble bien défini de fonctionnalités et est distinguable par un nom unique.[ISO/IEC 24765]

***Défaut*:** Un problème, s’il n’est pas corrigé, pourrait occasionner dans l’application une faillite ou des résultats incorrects. [ISO/IEC 24765].

Document des exigences : Un document contenant une combinaison des recommandations, des exigences ou des règlements qu’un logiciel doit satisfaire. [ISO/IEC 24765]

Environnement d’implémentation : Le matériel, le logiciel et des protocoles nécessaires pour effectuer l’implémentation.

***Exigence :*** 1. Une déclaration qui identifie ce qu'un produit ou un processus doit accomplir pour produire le comportement exigé et/ou des résultats. IEEE 1220-2005 Norme IEEE pour l'application et la gestion du processus d'ingénierie de systèmes. 3.1.16. 2. Une exigence d’un logiciel ou d’un système qui spécifie une fonction qu'un système/logiciel ou un composant d’un système/logiciel doit être capable d'exécuter. ISO/IEC 24765, vocabulaire en génie logiciel et des systèmes. 3. Une exigence qui spécifie une fonction qu'un système ou un composant de système doit pouvoir réaliser. [ISO/IEC 24765]

Exigences non fonctionnelles : Une exigence logicielle décrit comment le logiciel fonctionnera et non ce que le logiciel devra faire. ISO/IEC 24765, vocabulaire en génie logiciel et des systèmes. Syn. des contraintes de conception, l'exigence non fonctionnelle. Voir aussi : exigence fonctionnelle. NOTE  Par exemple, des exigences de performance logicielle, des exigences d'interfaces externes logicielles, des contraintes de conception logicielle et des attributs de qualité logicielle. Les exigences non-fonctionnelles sont parfois difficiles à tester, donc elles sont d'habitude évaluées subjectivement. [ISO/IEC24765]

Environnement des tests logiciels : L’environnement de tests logiciels qui satisfait les exigences d’exécution des tests et dans lequel les changements doivent être contrôlés pour permettre la régression si nécessaire. [Deployment Package - Software Testing - Basic Profile]

**Livraison *(delivery):*** *Mise à disponibilité d'un système ou d'un composant pour un client ou un utilisateur final.* *[ISO/IEC 24765]*

***Produit***: Le résultat d’un processus. [ISO/IEC 12207]

Référentiel (baseline) : Une spécification ou un produit qui a été formellement révisé et convenu par un accord, ce référentiel sert ensuite de base pour un nouveau développement et peut être changé seulement par des procédures de contrôle de changement formelles. [ISO/IEC 12207]. Le mot baseline a été traduit dans la partie 5 par 'version de référence'.

***Structure de découpage du projet (SDP) (Work Breakdown Structure, WBS) :*** Décomposition hiérarchique, axée sur les livrables, le travail que l'équipe du projet doit exécuter pour atteindre les objectifs du projet et produire les livrables voulus. La SDP organise et définit le contenu total du projet [PMI 2008]

# 3. Liens avec la norme ISO/IEC 29110

Cette trousse de déploiement présente les activités liées à la mise en œuvre du logiciel du rapport technique ISO/IEC 29110-5-1-1 pour les très petits organismes (TPO) - groupe des profils génériques : profil d’entrée [ISO/IEC 29110, 2012].

Le guide fournit des processus de gestion de projet et de mise en œuvre du logiciel. Ces processus intègrent des pratiques basées sur la sélection de processus de la norme ISO/IEC 12207:2008 Ingénierie des systèmes et du logiciel -- Processus du cycle de vie du logiciel  et de produits de documentation du standard ISO/IEC 15289:2011 Ingénierie des systèmes et du logiciel - Contenu des systèmes et produits d'information du processus de cycle de vie du logiciel (documentation).

Le but du processus de gestion de projet est d’établir et de réaliser, systématiquement, les Tâches inhérentes au projet de déploiement du logiciel, lesquelles permettent d’assurer la conformité avec les Objectifs du projet en matière de qualité, de durée et de budget.

Le but du processus de mise en œuvre du logiciel consiste à assurer la performance systématique des activités d'analyse, de conception, de développement, d'intégration et de tests de produits logiciels, nouveaux ou modifiés, selon les exigences établies.

Les deux processus sont en interaction (voir la figure 1).



Figure 1 — Guide de processus du profil d’entrée (ISO 29110-5-1-1)

# 4. Description des processus, activités, tâches, étapes, rôles et produits

Le diagramme suivant montre le flux d'information entre les activités du processus de mise en oeuvre du logiciel et montre les produits de travail les plus importants.



Figure 2 — Diagramme du processus de la mise en œuvre du logiciel

du profil d’entrée (ISO/IEC 29110)

## 4.1. Les activités du processus de mise en œuvre du logiciel (IL)

Le but du processus de mise en œuvre du logiciel (Software Implementation, SI) consiste à assurer la performance systématique des activités d'analyse, de conception, de développement, d'intégration et de tests de produits logiciels, nouveaux ou modifiés, selon les exigences établies.

Le processus de mise en œuvre du logiciel est composé des activités suivantes :

* SI.1 Initiation de la mise en œuvre du logiciel
* SI.2 Analyse des exigences du logiciel
* SI.3 Détermination des composants logiciels
* SI.4 Construction du logiciel
* SI.5 Intégration et tests du logiciel
* SI.6 Livraison du produit

### 4.1.1. Activité : SI.1 L'initiation de la mise en œuvre du logiciel

L'activité d'initiation de l’implémentation logicielle assure que le plan de projet établi pendant la planification du projet est présenté à l'équipe de travail afin d’obtenir son engagement. Cette activité fournit :

* La révision du plan de projet de l'équipe de travail pour déterminer l’assignation de tâches.
* Un environnement d’implémentation logicielle.

| **Rôle** | **Liste des tâches** | **Produits d’entrée** | **Produits de sortie** |
| --- | --- | --- | --- |
| PMWT | SI.1.1 Réviser le *Plan de projet* en vigueur avec les membres de l’équipe de travail afin d’arriver à une compréhension commune et d’obtenir leur engagement pour la réalisation du projet. | *Plan de projet* | *Plan de projet [révisé]* |
| WT | SI.1.2 Mettre en place ou à jour l’environnement de mise en œuvre. | *Plan de projet [révisé]* |  |

**L'initiation de la mise en œuvre du logiciel**

|  |
| --- |
|  |
| ***Objectifs :*** | Les membres de l'équipe de travail comprennent le plan du projet, identifient et mettent en place l'environnement d’implémentation logicielle. |
| ***Justification :***  | Cela permet de préparer l’équipe de travail pour les activités et d’acquérir et d’installer les outils nécessaires pour accomplir le travail. |
| ***Rôles :*** | Chef de projet  |
| Équipe de travail |
| ***Artefacts :*** | Plan du projet |
| ***Étapes :*** | Étape 1 : Réviser le plan de projet avec les membres de l'équipe. |
| Étape 2 : Mettre en place l’environnement d’implémentation. |
| ***Description des étapes :***  | ***Étape 1 : Réviser le plan de projet avec les membres de l'équipe.***Le chef de projet devrait organiser une rencontre avec tous les participants et avec l'équipe de travail pour assurer la compréhension du projet et de ses objectifs. Le chef de projet devrait s’assurer que chaque membre de l'équipe a été convoqué à la rencontre.À la fin de la réunion, le chef de projet devrait vérifier que toute l'équipe de travail comprend et accepte leur engagement au projet. |
| ***Étape 2: Mettre en place l’environnement d’implémentation.***Au cours de cette étape on prépare l'environnement sur lequel l’implémentation sera exécutée. L'objectif est de définir un environnement contrôlé et indépendant pour l’implémentation. |

### 4.1.2. Activité : SI.2 L'analyse des exigences du logiciel

L’activité *Analyse des exigences du logiciel* consiste à analyser les exigences établies en accord avec le client et à établir les exigences du projet validées. Cette activité permet d’obtenir :

* La revue du *Plan de projet* par les membres de l’équipe de travail dans le but d’assigner les tâches qui y sont énoncées.
* La sollicitation, l'analyse et la spécification des exigences du client.
* Une entente quant aux exigences du client.
* La vérification et la validation des exigences.

| **Rôle** | **Liste des tâches** | **Produits d’entrée** | **Produits de sortie** |
| --- | --- | --- | --- |
| TLWT | SI.2.1 Assigner les *Tâches* aux membres de l’équipe de travail conformément à leurs rôles selon *Plan de projet* en vigueur. | *Plan de projet [révisé]* |  |
| ANCUS | SI.2.2 Documenter ou mettre à jour la *Spécification des exigences.*Repérer et consulter les sources d’information (client, utilisateurs, systèmes précédents, documents, etc.) afin d'obtenir de nouvelles exigences.Analyser les exigences établies pour déterminer la *Portée* et la faisabilité. Générer ou mettre à jour la *Spécification des exigences.* | *Plan de projet** *Description du produit*
 | *Spécification des exigences[vérifiée]* |
| CUS | SI.2.3 Valider la *Spécification des exigences* et en obtenir l'approbation.Valider la *Spécification des exigences* afin de s'assurer qu'elle réponde aux besoins et aux attentes établis, dont la convivialité de l’interface utilisateur. | *Spécification des exigences [vérifiée]* | *Résultats de la validation**Spécification des exigences [validée]* |

**L'analyse des exigences du logiciel**

|  |
| --- |
|  |
| ***Objectifs :*** | L'objectif de cette activité est de définir clairement la portée du projet et identifier les exigences clés du système. |
| ***Justification :***  | Il est important de définir clairement la portée du projet (des frontières) et identifier les fonctionnalités clés du système avec le client pour éviter des problèmes comme des fonctionnalités clés oubliées ou les exigences mal comprises. |
| ***Rôles :*** | Équipe de travail |
| Client |
| ***Artefacts :*** | Document d’exigences logicielles |
| ***Étapes :*** | Étape 1 : Collecter l’information sur le domaine d'application (par exemple, les finances, la médecine) |
| Étape 2 : Identifier la portée du projet |
| Étape 3 : Identifier et saisir les besoins |
| Étape 4 : Structurer et prioriser les exigences |
| ***Description des étapes :*** | ***Étape 1. Collecter l’information sur le domaine d'application***Pendant cette étape, l’analyste saisie les concepts clés du domaine d'affaires du client. Le client aide l'analyste en lui donnant toutes les informations (la documentation existante ou l'explication) qui facilitera cette compréhension.Des concepts clés sont inscrits dans une section du glossaire dans l’ébauche du document des spécifications des exigences logicielles.***Étape 2. Identifier la portée du projet***L'analyste logiciel, aidé par la personne étant responsable des aspects contractuels du projet (le directeur commercial) identifie clairement les fonctionnalités principales qui sont incluses dans la portée du projet.Conseil: l'identification des fonctionnalités qui sont HORS de la portée du projet peut aider à clarifier les différences de compréhension avec vos clients.***Étape 3. Identifier et saisir les exigences***Ayant à l'esprit les concepts clés liés au domaine d'affaires du client, l'analyste peut commencer l'identification des exigences. Aucune des situations dans des projets informatiques n'est identique. Dans quelques cas, la plupart des exigences sont déjà identifiées dans un document (p.ex. l'appel d'offres). Cependant, dans la plupart des cas, des exigences clés sont juste mentionnées oralement par le client.L'analyste doit identifier et inscrire les exigences clés du système à être construit. Pendant cette étape, l’analyste ne devrait pas commencer à détailler les exigences identifiées. Le but principal est d'obtenir une vue complète des exigences du système à développer.***Étape 4. Structurer et prioriser les exigences***:En utilisant les exigences identifiées à l'étape précédente, l'analyste doit organiser et structurer les exigences identifiées en conséquence (par exemple, selon les processus commerciaux ou selon les fonctions du système).Les priorités doivent être identifiées par le client pour des fonctionnalités clés du système. Les priorités peuvent être définies comme suit : * 'Haute' - une fonctionnalité qui *devra* être implémentée.
* 'Moyenne' - une fonctionnalité qui *devrait* être implémentée.
* 'Basse' - une fonctionnalité qui *pourrait* être implémentée.

Le produit sortant de cette étape est une liste d'exigences qui sont organisées dans le document des exigences. |

### 4.1.3. Activité : SI.3 Détermination des composants logiciels

L’activité *Détermination des composants logiciels* transforme les exigences logicielles à l’architecture du logiciel. Cette activité permet d’obtenir :

* La revue, par les membres de l’équipe de travail, du *Plan de projet* pour assigner les tâches.
* La détermination des composants logiciels et des interfaces connexes.

| **Rôle** | **Liste des tâches** | **Produits d’entrée** | **Produits de sortie** |
| --- | --- | --- | --- |
| TLWT | SI.3.1 Assigner les *Tâches* aux membres de l’équipe de travail conformément à leurs rôles et selon *Plan de projet* en vigueur. | *Plan de projet* |  |
| ANDES | SI.3.2 Comprendre la *Spécification des exigences.* | *Spécification des exigences*  |  |
| ANDES | SI.3.3 Documenter ou mettre à jour la détermination des composants *logiciels.*Analyser la *Spécification des exigences* pour produire les *Composants logiciels,* sa disposition dans les sous-systèmes définissant les interfaces internes et externes. Fournir les détails des *Composants logiciels* et de leurs interfaces pour en permettre la construction de façon évidente.  | *Spécification des exigences*  |  |

**L'identification des composants logiciels**

|  |
| --- |
|  |
| ***Objectifs :*** | Identifier les composants logiciels qui répondront aux exigences demandées par le client, seront évalués avant la sortie en production. Vérifier que toutes les exigences soient satisfaites. |
| ***Justification :***  | L'identification des composants logiciels est une activité clé d'un projet logiciel. L’architecture inadéquatement conçue est une recette pour l’échec. Le client ne finalisera pas le paiement si la conception ne répond pas à ses exigences. |
| ***Rôles :*** | Équipe de travail |
| ***Artefacts :*** | Spécifications d'exigences |
| Configuration du logiciel |
| ***Produits :*** | Diagramme UML ou BPMN  |
| ***Étapes :*** | Étape 1 : Comprendre les spécifications des exigences logicielles |
| Étape 2 : Documenter ou mettre à jour les composants logiciels |
| ***Description des étapes :*** | ***Étape 1. Comprendre les spécifications des exigences logicielles*** * Examinez chacune des exigences et assurez-vous qu'elles soient comprises et groupées :
* Exigences fonctionnelles dans les groupes logiques.
* Exigences non-fonctionnelles groupées.
* Vérifiez les groupes et vérifiez si les exigences ont été comprises.
* Si nécessaire, mettez à jour le document des spécifications d’exigences logicielles afin d’y ajouter les clarifications nécessaires.
* Gardez la documentation à jour dans le dépôt d’information du projet.

***Étape 2. Documenter ou mettre à jour les composants logiciels**** Analysez les spécifications des exigences pour produire les composants, leurs dispositions dans les sous-systèmes et les composants qui définissent les interfaces internes et externes.
* Décrivez en détails l'apparence et le comportement de l'interface, basés sur les spécifications des exigences de manière à ce que les ressources pour son implémentation puissent être prévues.
* Fournissez les détails des composants et de leurs interfaces pour permettre leur construction de façon évidente.
 |

### 4.1.4. Activité : SI.4 Construction du logiciel

L’activité *Construction du logiciel* vise à développer le code et les données du logiciel à partir de la *Détermination des composants logiciels*. Cette activité permet d’obtenir :

* La revue, par les membres de l’équipe de travail, du *Plan de projet* pour assigner les tâches.
* L’examen, par les membres de l’équipe de travail des composants *logiciels déterminés*
* Des cas et des procédures de test pour les tests unitaires et les tests d'intégration.
* Des *Composants logiciels* codés et des tests unitaires exécutés.

| **Rôle** | **Liste des tâches** | **Produits d’entrée** | **Produits de sortie** |
| --- | --- | --- | --- |
| PMWT | SI.4.1 Assigner les *Tâches* aux membres de l’équipe de travail conformément à leurs rôles et selon le *Plan de projet* en vigueur. | *Plan de projet* |  |
| WT | SI.4.2 Comprendre la *Conception du logiciel.* | *Composants logiciels déterminés* |  |
| WT | SI.4.3 Construire ou mettre à jour les *Composants logiciels* | *Composants logiciels déterminés* | *Composants logiciels* |
| WT | SI.4.4 Concevoir ou mettre à jour des cas de test unitaire et des procédures de test pour la conduite de tests unitaires et de tests d'intégration qui reposent sur les spécifications du logiciel et les composants logiciels déterminés.Le client fourni les données de tests si nécessaire. | *Spécifications du logiciel [validées]**Composants logiciels déterminés* | *Cas et procédures de test* |
| WT | SI.4.5 Tester les composants logiciels. Corriger les défauts repérés jusqu'à ce que les tests unitaires soient accomplis avec succès. | *Cas et procédures de test**Composants logiciels* | *Composants logiciels [soumis aux tests unitaires]* |

**La construction du logiciel**

|  |
| --- |
|  |
| ***Objectifs :*** | Développer les composants logiciels identifiés. |
| ***Justification :***  | Certains développeurs font confiance à leurs connaissances pour développer des produits logiciels sans une approche systématique. Cependant, certains d'entre eux peuvent trouver cette approche utile pour construire des composants complexes. Il y a plusieurs approches pour produire des composants. Ici on utilise l'approche de pseudocode qui est répandue et acceptée.  |
| ***Rôles :*** | Équipe de travail |
| ***Artefacts :*** | Composants logiciels |
| ***Étapes :*** | Étape 1 : Dessiner les composants |
| Étape 2 : Codifier les composants  |
| Étape 3 : Vérifier les composants  |
| ***Description des étapes :*** | ***Étape 1. Dessiner*** ***les composants*** 1. **Vérifier la contribution des composants.**

Comprendre la contribution que les composants fournissent au produit final. 1. **Définissez le problème à résoudre par les composants.**

Établir le problème que le composant résoudra en détails pour faciliter son développement. Si l’information n’est pas suffisante, le développeur devrait la compléter avec le soutien de l’analyste. 1. **Recherchez des fonctionnalités disponibles dans des bibliothèques standards.**

Découvrez si les fonctionnalités des composants pourraient être déjà disponibles dans les bibliothèques du langage de programmation, la plateforme ou les outils que vous utiliserez.1. **Écrivez le pseudocode.**

Commencez par écrire le pseudocode de façon générale et ensuite, continuez vers quelque chose de plus spécifique. La partie la plus générale d'un composant est un commentaire d'en-tête décrivant ce que le composant est censé faire. Écrivez d'abord, une déclaration concise du but du composant, par la suite, décomposez les déclarations dans plusieurs déclarations de bas niveaux en visant toujours la perfection. Après la création de la tendance du composant en incluant le traitement d'erreurs et la gestion de toutes les choses qui pourraient mal tourner dans le composant, tels que : les valeurs des entrées incorrectes, des valeurs non invalides retournées de d'autres routines, et ainsi de suite.1. **Concevez les données des composants.**

Définissez les types de données clés. 1. **Vérifiez le pseudocode.**

Passez en revue de pseudocode, si vous n'êtes pas très confiant, expliquez-le à quelqu'un d'autre (par exemple, l’analyste), cette explication aidera à clarifier vos idées et peut améliorer le pseudocode.***Étape 2. Coder les composants***1. **Écrivez la déclaration du composant.**

Convertissez la déclaration en commentaires de langage de programmation.1. **Convertissez le pseudocode en commentaires de haut niveau.**
2. **Ajoutez les commentaires au code**

Ajoutez, au-dessous de chaque ligne de code, des commentaires sur le pseudocode. Chaque commentaire de pseudocode décrit un bloc ou un paragraphe de code.1. **Vérifiez si le code devrait inclure de nouveaux composants.**

Dans certains cas vous aurez beaucoup de codes en-dessous d'une des lignes de pseudocode. Dans ce cas, vous devriez créer un sous composant.***Étape 3. Vérifiez le composant***Cette étape consiste à une revue de code exécuté par le programmeur, une compilation et un débogage.1. **Faites une revue du code**
* Vérifiez le programme contre les exigences identifiées pour s'assurer que toutes les fonctions de programme exigées sont implémentées.
* Suivez la liste de contrôle (*check-list*) de la revue du code fournie à la section 7, trouvez les défauts dans le programme.
* Lors de la revue, marquez les défauts sur le code source.
* Après la revue du code, corrigez tous les défauts.
* Après l'achèvement des corrections, produisez une liste du programme source pour le programme corrigé.
* Passez en revue toutes les corrections pour s'assurer qu'elles ont été corrigées correctement.
* Corrigez les défauts restants.
1. **Compilez le code**

Laissez l’ordinateur vérifier les variables non déclarées, les conflits de noms, etc. Vous pouvez l'avoir fait avant la revue du code.1. **Passez à travers du code avec un débogueur**

Une fois que le logiciel est compilé, exécutez le programme débogueur et passez par chaque ligne de code. Assurez-vous que chaque ligne exécute comme vous vous y attendez. Vous pouvez le faire après les tests unitaires en vous concentrant sur la découverte de la source d'un défaut. |

###

### 4.1.5 Activité : SI.5 Intégration et tests du logiciel

L’activité *Intégration et tests du logiciel* vise à faire en sorte que les *Composants* logiciels intégrés répondent aux exigences logicielles. Cette activité permet d’obtenir :

* La revue, par les membres de l’équipe de travail, du *Plan de projet* pour assigner les tâches.
* La compréhension des *Jeux de tests* et des *procédures de tests*, ainsi que de l’environnement d’intégration.
* Les *Composants logiciels* intégrés, les défauts corrigés et les résultats documentés.

| **Rôle** | **Liste des tâches** | **Produits d’entrée** | **Produits de sortie** |
| --- | --- | --- | --- |
| PMWT | SI.5.1 Assigner les *Tâches* aux membres de l’équipe de travail conformément à leurs rôles et selon le *Plan de projet* en vigueur. | *Plan de projet* |  |
| WT | SI.5.2 Comprendre les *Cas et procédures de test*.Mettre en place ou à jour l'environnement de test. | *Cas et procédures de test*  |  |
| WT | SI.5.3 Intégrer le *Logiciel* en utilisant les *Composants logiciels* et mettre à jour les *Cas et procédures de test* d’intégration, s’il y a lieu. | *Composants logiciels* *Cas et procédures de test*  | *Logiciel**Cas et procédures de test [mis à jour]* |
| PRCUS | SI.5.4 Réaliser les tests en utilisant les *Cas et procédures de test* pour l’intégration et documenter les résultats dans le *Rapport de test*. | *Logiciel**Cas et procédures de test* | *Logiciel [soumis aux tests]**Rapport de test* |
| PR | SI.5.5 Corriger les défauts repérés et réaliser des tests jusqu’à ce que les tests sont complétés avec succès | *Logiciel* *Rapport de test**Cas et procédures de test* | *Logiciel [corrigé]**Rapport de test [défauts corrigés]* |
| PR | SI.5.6 Incorporer les *Spécifications logicielles* et le logiciel à la *Configuration du logiciel* | *Spécifications logicielles* *Logiciel*  | *Configuration du logiciel* *- Spécifications logicielles*  - *Logiciel* |

 **L'intégration et les tests du logiciel**

|  |
| --- |
|  |
| ***Objectifs :*** | Déterminer la stabilité du produit logiciel par les résultats obtenus en soumettant le produit aux tests. |
| ***Justification :***  | Permettre d'exécuter les différents types de tests et d'identifier les défauts qui doivent être corrigés par l'équipe de développement logiciel.  |
| ***Rôles :*** | Client  |
| Équipe de travail  |
| ***Artefacts :*** | Rapport de tests |
| ***Étapes :*** | Étape 1 : Préparez l'environnement de tests du logiciel  |
| Étape 2 : Exécutez des itérations de tests |
| Étape 3 : Exécutez des tests de régression  |
| Étape 4 : Complétez la procédure de tests |
| Étape 5 : Documentez les résultats dans le rapport de tests |
| ***Description des étapes :*** | ***Étape 1 : Préparez l'environnement de tests du logiciel*** Pendant cette étape, l'environnement de tests; la disponibilité des composants (corrigés et sous gestion de version) qui doivent être testés et les données qui seront utilisées pour cette exécution sont préparés. L'objectif est de définir un environnement contrôlé et indépendant pour les tests. La préparation de l'environnement de test inclut par exemple: Les spécifications de la configuration du matériel, le système d'exploitation, la configuration du fureteur; les spécifications logicielles du système, le moteur de base de données et le moteur de support des tests.***Étape 2: Exécutez des itérations de tests*** Avant l'exécution, il est nécessaire de garantir que l'analyste assigné comprenne les tests.Une itération de tests correspond à l'exécution de tests. L’itération de tests est faite une fois que l'équipe de développement a exécuté les tests d'unité. Selon la stabilité de l’application, l'analyste peut décider, à chaque itération, d’exécuter toutes les tests identifiés ou un sous-ensemble représentatif. Par l'exécution de tests, l'analyste peut trouver des écarts aux résultats attendus, il doit les cataloguer comme des « défauts » du produit. Les défauts du produit identifiés sont classifiés et enregistrés.Pour la classification des défauts, les définitions suivantes peuvent être prises en considération :* Bloqueur : Ce type de défaut arrête l'opération du programme ou du composant en faisant mentionner les résultats qui empêchent l'opération de continuer.
* Fonctionnel : Cela arrive lorsqu’un programme est exécuté et que ces résultats ne correspondent pas aux résultats attendus.
* Présentation : Relié à la présentation de produit. Celle-ci doit s'adapter aux normes définies et aux règles grammaticales et orthographiques de la langue dans laquelle le programme est montré.

***Étape 3 : Exécutez des tests de régression*** L'objectif est de s'assurer que le nouveau code ou celui modifié (pour fournir une solution aux défauts identifiés pendant les tests) accomplit les exigences indiquées et que le code non modifié n'est pas affecté par l'activité de maintenance.Les tests de régression commencent leur exécution aussitôt que le groupe technique livre la solution des défauts ou ajoute de nouvelles caractéristiques au logiciel et ce, afin d’appliquer le test dans un environnement contrôlé. Pour réaliser une procédure efficace, il est possible de combiner le test de régression avec une itération de test.***Étape 4 : Complétez*** ***la procédure de tests*** Une fois les itérations prédéfinies ont été achevées et que les défauts identifiés ont été résolus, une analyse pour démontrer qu'il y a une tendance continue de diminuer le nombre des défauts est effectué. La fermeture de la procédure de tests est faite en élaborant un rapport de fermeture, qui décrit la procédure exécutée et montre les conclusions avec des recommandations correspondantes pour l'amélioration du produit et les processus. Ces recommandations font partie des leçons apprises qui sont la base de la TPO. L'équipe de développement décide de sortir le produit logiciel à production, selon les résultats obtenus dans les tests.***Étape 5 : Documentez*** ***les résultats dans le rapport de tests*** L'analyste ou le programmeur doivent préparer un rapport de tests pour chaque itération de tests effectuée. Le rapport doit contenir la condition actuelle du produit. Le rapport doit être passé en revue par tous les membres de l'équipe du projet et ils doivent établir des engagements pour la nouvelle itération de tests. |

### 4.1.6. Activité : SI.6 Livraison du produit

L’activité Livraison du produit vise à fournir le produit logiciel intégré au gestionnaire du projet et du support pour la livraison. Cette activité permet d’obtenir :

* La revue, par les membres de l’équipe de travail, du *Plan de projet* pour assigner les tâches.
* La livraison du produit logiciel et la documentation pertinente conformément aux *Directives de livraison*.

| **Rôle** | **Liste des tâches** | **Produits d’entrée** | **Produits de sortie** |
| --- | --- | --- | --- |
| PMWT | SI.6.1 Assigner les *Tâches* aux membres de l’équipe de travail conformément à leurs rôles et selon le *Plan de projet* en vigueur. | *Plan de projet* |  |
| DES | SI.6.2 Comprendre la *Configuration du logiciel.* | *Configuration du logiciel* |  |
| PMWT | SI.6.3 Livrer le produit au gestionnaire du projet et fournir du support conformément au Plan de projet. | *Plan de projet**Configuration du logiciel* | *Configuration du logiciel (prête à être livrée)* |

**La livraison du produit**

|  |
| --- |
|  |
| ***Objectifs :*** | Conduire les activités de livraison en cours tel que, à la fin du projet, chaque élément à fournir est disponible et satisfait les critères d'acceptation définis dans les instructions de livraison. |
| ***Justification :***  | En conduisant les activités en cours, il ne devrait y avoir aucune surprise, aucun retard pour obtenir l'acceptation d'éléments à fournir. Autrement, le client ne finalisera pas les paiements au TPO. |
| ***Rôles :*** | Équipe de travail |
| ***Produits :*** | Configuration du logiciel |
| Registre d’acceptation  |
| ***Artefacts :*** | Plan du projet |
| Instructions de livraison approuvées |
| Registre d’acceptation |
| ***Étapes :*** | Étape 1 : Comprendre la configuration du logiciel. |
| Étape 2 : Effectuer la livraison selon les instructions de livraison. |
| ***Description des étapes :*** | ***Étape 1. Comprendre la configuration du logiciel***. Obtenez, du plan de projet, les conditions de livraison.* Construisez/obtenez la ligne des bases de produit
* L'équipe de travail prépare les éléments à fournir.
* L'équipe de travail établit les bases pour la configuration du produit incluant l'environnement, le manuel approprié, la documentation importante et le produit configuré.

***Étape 2. Effectuer la livraison* s*elon les instructions de livraison.*** Vérifiez que chaque composant logiciel satisfait les critères d'acceptation.* Mettez à jour le formulaire registre d'acceptation
* Planifiez une rencontre avec le client
* Obtenez l'approbation du client pour le formulaire registre d'acceptation
* Le client signera le formulaire.
* Donnez une copie du formulaire registre d'acceptation au client
* Stockez le formulaire d'acceptation dans le dépôt d’information du projet.
 |

## 4.2 Description des rôles

Ceci est une liste alphabétique des rôles, des abréviations et de la description des compétences suggérées.

|  | **Rôle** | **Abréviation** | **Compétences** |
| --- | --- | --- | --- |
| 1. | Client | CUS | Connaissance des processus du client et aptitude à expliquer les exigences du client.Le client (ou son représentant) doit avoir le pouvoir d’approuver les exigences et les changements à apporter. Le terme « client » comprend les personnes qui représentent les utilisateurs pour veiller à ce que l'environnement d’opération soit mis à l’étude.Connaissance du domaine du logiciel et expérience connexe. |
| 2. | Gestionnaire de projet | PM | Aptitude à diriger et expérience dans la prise de décisions, la planification, la gestion du personnel, la délégation et la supervision, les finances et le développement de logiciel. |
| 3. | Équipe de travail | WT | Connaissance et expérience selon les rôles du projet : TL, AN, DES ou PR.Connaissance des normes appliquées par le client et/ou le TPO. |

## 4.3. Description des produits

Voici une liste par ordre alphabétique des produits d'entrée, de sortie et des produits internes du processus, les descriptions, les états possibles ainsi que la source du produit. On entend par source soit un autre processus, soit une entité extérieure au projet, comme le client. Les éléments de cette liste sont énumérés, à des fins de présentation seulement, dans un tableau constitué de quatre colonnes. À quelques exceptions près, les éléments énumérés dans le tableau suivant sont fondés sur les éléments d’information de la norme ISO/CEI 15289.

Le statut d'un produit donne l'information à l'équipe du projet sur ​​le type de travail (tâches) déjà réalisé sur le produit (par exemple: évalué, vérifié, testé, établi à titre de référence). Cette information peut être utilisée pour lancer des tâches qui peuvent utiliser le produit comme entrée. Certains produits n'ont pas de statut attribué parce qu'ils sont seulement informatifs et ils ne changent pas de contenu (par exemple: l'enregistrement acceptation, le registre de correction, le dépôt de sauvegarde du projet, les résultats de la vérification ou de la validation)

Tableau 23 — Descriptions de produit

|  | **Nom** | **Description** | **Source** |
| --- | --- | --- | --- |
| *1.* | *Cas et procédures de test* | Éléments nécessaires pour le test du code. Un cas de test peut comprendre :* Détermination du cas de test
* Éléments du cas de test
* Spécifications d’entrée
* Spécifications de sortie
* Besoins en matière d’environnement
* Exigences spéciales en matière de procédure
* Dépendances de l’interface

 Les procédures de test peuvent comprendre :* Détermination du nom du test, description et date d’achèvement
* Détermination des problèmes potentiels en matière de mise en œuvre
* Détermination du nom de la personne qui a réalisé la procédure de test
* Détermination des préalables (*prerequisites*)
* Détermination des étapes de la procédure dont le numéro de l’étape, la mesure à être prise par le responsable du test et les résultats attendus

Les états possibles sont : vérifiés et établis à titre de référence. | Mise en œuvre du logiciel |
| *2.* | *Composants logiciels* | Un ensemble d’unités de codes liéesL'état possible est : tests unitaires réussis | Mise en œuvre du logiciel |
| *3.* | *Configuration du logiciel* | Un ensemble de produits logiciels uniques et cohérents dont :* *Spécification des exigences*
* *Logiciel*

Remarque: peut aussi inclure les caractéristiques additionnelles suivantes* *Conception du logiciel*
* *Enregistrement de la traçabilité*
* *Composants logiciels*
* *Cas et procédures de test*
* *Rapport de test*
* *Guide d’opération du produit*
* *Documentation de l’utilisateur du logiciel*
* *Documentation de la maintenance*

Les états possibles sont : prêt à êtrelivrée et acceptée. | Mise en œuvre du logiciel |
| *4.* | *Demande de changement* | Indique un problème relatif au *Logiciel* ou à la documentation, ou une amélioration requise et énonce la demande du changement. Elle peut être doté des caractéristiques suivantes :* Détermination de l'objectif du changement demandé
* Détermination de l'état de la demande
* Détermination des coordonnées du demandeur
* Détermination du système ou systèmes touchés
* Définition de l’incidence sur les opérations du ou des systèmes existants
* Définition de l'incidence sur la documentation connexe
* Degré d’importance de la demande, date à laquelle le changement est requis

Les états possibles sont : acceptée et suivie | Mise en œuvre du logicielClientGestion de projet |
| *5.* | *Dépôt d’information du projet* | Un média électronique dans lequel les produits et les livrables du projet sont stockés. Il peut être doté des caractéristiques suivantes :* Capacité de stockage des produits de travail du projet
* Capacité de stockage des produits livrables mis en circulation
* Capacités de stockage et de récupération (*retrieval*)
* Possibilité de naviguer dans son contenu
* Liste du contenu avec description des attributs
* Possibilité de partager et de transférer des produits de travail entre les groupes concernés
* Contrôles d'accès efficaces
* Maintien à jour des descriptions de produits de travail
* Possibilité de récupérer des versions archivées des produits de travail
* Capacité de communiquer l’état des produits de travail
* Changements apportés aux produits de travail sont tracés (*tracked*) avec les *Demandes de changement*

L'état possible est :mis à jour. | Gestion de projet |
| *6.* | *Détermination des composants logiciels* | Information textuelle et graphique sur la structure du logiciel. Il peut être doté des caractéristiques suivantes : Décrit la structure globale du logiciel:* Détermine les composants logiciels requis
* Détermine la relation entre les composants logiciels
 | Mise en œuvre du logiciel |
| *7.* | *Énoncé des travaux* | Description des travaux à effectuer dans le cadre de la *Construction du logiciel*. Il peut comprendre :* *Description de produit*
* But
* Exigences générales du client
* Description de la *Portée* de ce qui fait partie de l’entente et de ce qui n’en fait pas partie
* *Livrables*: la liste des produits à livrer au client

Remarque:peut aussi inclure la caractéristique additionnelle suivante* *Objectifs* du projet

L’état possible est :révisé. | Client |
| *8.* | *Enregistrement d’avancement* | Documente l'état d'avancement du projet par rapport au Plan de projet. Il peut être doté des caractéristiques suivantes :* L'état des tâches réelles par rapport aux tâches planifiées
* L'état des résultats réels par rapport aux objectifs et aux buts établis
* L'état des ressources réellement affectées au projet par rapport aux ressources planifiées
* L'état des coûts réels par rapport aux coûts prévus au budget
* L'état de la durée réelle par rapport à celle prévue
* L'état des risques réels par rapport aux risques précédemment identifiés
* L'enregistrement de tout écart par rapport aux tâches planifiées et la justification de cet écart

L’état possible est : évalué. | Gestion de projet |
| *9.* | *Enregistrement de réception* | Documente l’acceptation des livrables du projet par le client. Il peut être doté des caractéristiques suivantes :* Enregistrement de la réception du produit
* Mention de la date de réception
* Liste des éléments livrés
* Fait état de la vérification de tout critère d’acceptation défini par le client
* Enregistrement de tout problème non réglé ou de toute question restée sans réponse (s’il y a lieu)
* Signé par le client

Signature du client qui a réceptionné le produit | Gestion de projet |
| *10.* | *Enregistrement de réunion* | Documente les ententes conclues entre le client et l’équipe de travail. Il peut être doté des caractéristiques suivantes :* But de la réunion
* Participants
* Date, lieu de la rencontre
* Références à des comptes rendus précédents
* Accomplissements
* Problèmes ou questions soulevés
* Tout problème non réglé ou toute question restée sans réponse
* Ententes
* Date de la prochaine réunion, s’il y a lieu.

L’état possible est : mis à jour | Gestion de projet |
| *11.* | *Logiciel* | Un logiciel (source logicielle et code exécutable) destiné à un client, constitué d’un assemblage de *Composants logiciels* intégrés. L'état possible est : testé | Mise en œuvre du logiciel |
| *12.* | *Plan de projet* | Indique la façon dont les processus et les activités nécessaires à la réalisation du projet seront exécutés afin d'assurer l'achèvement réussi du projet et la qualité des produits livrés. Il comprend les éléments suivants qui peuvent faire état des points suivants :* *Description de produit*
* But
* Exigences générales du client
* Description de la *Portée*; ce qui fait partie de l’entente et ce qui n’en fait pas partie
* *Objectifs* du projet
* *Livrables :* une liste des produits à livrer au client
* *Tâches*, comprenant la vérification, la validation et les revues avec le client et l’équipe de travail pour veiller à la qualité des produits. Les *Tâches* peuvent être représentées dans une SDP (structure de découpage du projet).
* *Liens* entre les tâches et leurs interdépendances.
* *Durée prévue* des tâches
* *Ressources* (humaines, matérielles, en matière de normes, d'équipement et d'outils) dont la formation nécessaire et le calendrier d’utilisation des *ressources*
* *Composition de l’équipe de travail*
* *Calendrier des tâches du projet* indiquant la date prévue de commencement et d’achèvement de chaque tâche
* *Évaluation des efforts nécessaires et des coûts*
* *Détermination des risques associés au projet*

Remarque: peut aussi inclure les caractéristiques additionnelles suivantes* *Ressources* peut inclure la formation requise
* *Calendrier des tâches du projet* peut inclure les liens entre les tâches et leurs interdépendances.
* *Stratégie de contrôle des versions :*
* Outils ou mécanismes du dépôt d’information du produit
* Emplacement du dépôt et des mécanismes pour y accéder
* Méthodes d’identification et de contrôle des versions
* Mécanismes de sauvegarde et de récupération
* Mécanismes de stockage, de traitement et de livraison (comprenant l'archivage et la recherche)
* *Directives de livraison*
* Détermination des éléments nécessaires à la mise en circulation du produit (c.-à-d., matériel, logiciel, documentation, etc.)
* Exigences en matière de livraison
* Ordre séquentiel des *Tâches* à exécuter
* Détermination des versions applicables
* Détermination des *Composants logiciels* livrés avec les renseignements déterminant la version
* Détermination des procédures de sauvegarde ou de récupération nécessaires

L'état possible est : accepté. | Gestion de projet |
| *13.* | *Rapport de test* | Documente l’exécution des tests. Ce qui peut comprendre :* Résumé de chaque défaut
* Détermination du cas de test connexe
* Détermination des noms des personnes qui ont repéré chacun des défauts
* Détermination des fonctionnalités sur lesquelles influe chaque défaut
* Détermination de la date à laquelle s’est produit chaque défaut
* Détermination de la date à laquelle chaque défaut a été corrigé
* Détermination des noms des personnes qui ont corrigé chacun des défauts

Remarque: peut aussi inclure la caractéristique additionnelle suivante* Détermination du degré d’importance *(severity*) de chaque défaut
 | Mise en œuvre du logiciel |
| *14***.** | *Spécification des exigences* | Détermine les exigences logicielles. Elle peut être dotée des caractéristiques suivantes : * Introduction : une description générale du *Logiciel* et de son utilisation dans le cadre de la portée des activités d’affaires du client;
* Description des exigences :
* Fonctionnalité : les besoins établis qui doivent être satisfaits par le *Logiciel* lorsque ce dernier est utilisé dans des conditions particulières. La fonctionnalité doit être appropriée, précise et sûre.
* Interface utilisateur : définition des caractéristiques d’interface qui permettent de comprendre et de maîtriser le logiciel afin que l’utilisateur s’acquitte efficacement de ses tâches, dont la description du modèle de l’interface
* Interfaces externes : définition des interfaces avec d’autres logiciels ou matériels.

Remarque: peut aussi inclure les caractéristiques additionnelles suivantes* Fiabilité : spécification du mode d’exécution du logiciel en ce qui a trait à la maturité, à la résilience et à la récupération
* Efficience : indication du niveau d’exécution du logiciel en ce qui concerne le temps et l’utilisation des *Ressources*
* Maintenabilité : description des éléments facilitant la compréhension et l’exécution des modifications à venir à apporter au *Logiciel*
* Portabilité : description des caractéristiques du logiciel qui permettent de le transférer d’un emplacement à un autre
* Limites/contraintes en matière de conception et de construction : les exigences imposées par le client
* Interopérabilité : possibilité, pour deux systèmes ou *Composants logiciels* ou plus, de transférer des renseignements entre eux et de les utiliser
* Possibilité de réutilisation : caractéristique de tout produit/sous-produit ou d’une de leurs parties, qui permet à plusieurs utilisateurs de s’en servir comme produit final, dans le développement de leur logiciel ou pour exécuter d’autres produits logiciels
* Légales ou normatives : des exigences imposées par des lois, par des règlements, etc.

Chaque exigence est déterminée et unique, elle peut être vérifiée ou évaluée.Les états possibles sont : vérifiée, validée. | Mise en œuvre du logiciel |

## 4.4. Description des artefacts

|  |  |
| --- | --- |
| **Artefacts** | **Définition** |
| Lettre d’acceptation | Document qui établit l’acceptation du client des livrables établis dans le projet. |
| Description du Projet  | Description de haut niveau du projet, il inclut : la portée, les objectifs et les livrables majeurs. |
| Plan de Projet  | Une déclaration de comment et quand les objectifs d'un projet doivent être réalisés, en montrant les produits majeurs, les jalons, les activités et les ressources exigées pour le projet. |
| Document d’exigences logicielles  | Le document dans lequel on centralise toutes les exigences identifiées. |
| Logiciel | Un ensemble cohérent des produits logiciels qui incluent : * Spécification d'exigences
* Composants logiciels
* Logiciel (unité, produit, article)
* Test et rapports des incidents
* Manuel d’opération
* Manuel de l'utilisateur
 |
| *Composant Logiciel*  | Un ensemble d’unités de code. Les états applicables sont: unité testée, corrigée et de base. |
| *Configuration du logiciel*  | Ensemble de produits logiciels dans lequel doit inclure :* *Spécification d'exigences*
* *Architecture du logiciel*
* Composants logiciels
* *Logiciel*
* *Rapport de tests*
* *Guide d’opération du produit*
* *Guide de l’utilisateur*
* *Documentation de la maintenance*
 |
| *Rapport de tests* | Noter les résultats de l’exécution de tests dans un environnement spécifique.  |
| Diagrammes UML  | Ces diagrammes faciliteront la représentation graphique de la conception du logiciel en utilisant Unified Modelling Language. UML peut fournir une méthode facile et standard d'exprimer les données, les processus ou l'architecture. |

# 5. Gabarits et outils

Les modèles des gabarits fournis ici sont des exemples, ils devraient être personnalisés pour votre projet.

## 5.1. Modèle pour l’initiation de l’implémentation logicielle

Gabarit de la table des matières d’une notification de réunion

1. Nom de la réunion (incluant le nom de projet)

2. Priorité de la réunion

3. Destinataires

4. Objectif de la réunion

5. Contenu du message

6. La date, l’endroit et l’heure

7. D'autres informations (ex. repas inclus)

Gabarit d’une liste de contrôle d’assistance aux réunions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *No.* | *Nom du membre de l’équipe de travail* | *Rôle* | *Assistance**Oui Non*  | *Signature* |
|  |  |  |  |  |  |
| *Commentaires* |

## 5.2. Modèle pour l’analyse des exigences logicielles

Tableau des exigences logicielles

Pour être utilisé dans une feuille d'Excel structurée, par exemple :

|  |  |  |  |
| --- | --- | --- | --- |
| *ID* | *Exigence* | *Description* | *Priorité**Haute Moyenne Basse* |
|  |  |  |  |  |  |

Chacune des exigences peut être définie en format de table. La structure du tableau pourrait être:

|  |  |
| --- | --- |
| *Projet* | *<Identification dans le Projet >* |
| Date | <La date de la dernière modification> |
| Exigence | <Identification, Texte de l’exigence> |
| Description | < Description détaillée de l’exigence> |
| Justification | <Pourquoi est-elle importante> |
| Priorité | <Niveau de priorité de l’exigence> |
| Dépendance | <Rapport avec d’autres exigences> |
| Commentaire | <Remarques additionnelles>  |

## 5.3 Modèle pour l’identification des composants logiciels

Sites d’internet qui donnent des modèles de diagrammes standards.

|  |  |
| --- | --- |
| **Modèle** | **Source** |
| UML | <http://www.uml.org> |
| BPMN | <http://www.bpmn.org> |

Pour être utilisé dans une feuille d'Excel structurée, par exemple :

Gabarit de la table des matières de l’identification des composants

Identification du projet (Nom et date)

1. Composant 1.
	1. Identification
	2. Description
	3. Sous-composants
	4. Exigences
2. Composant 2.
	1. Identification
	2. Description
	3. Sous-composant
	4. Exigence
3. …

## 5.4 Modèle pour la construction du logiciel

Sites Internet qui donnent des exemples de modèles pour le codage.

|  |  |
| --- | --- |
| **Modèle** | **Source** |
| Gabarit standard de codage | <http://www.construx.com>  |
| Gabarit du code source C / C++ | <http://www.construx.com> |
| Gabarit du code source Java | <http://www.construx.com>  |
| Standard du codage Java | <http://www.ontko.com/java/java_coding_standards.html> |
| Gabarit du code source Visual Basic | <http://www.construx.com>  |
| Gabarit du code source XML | <http://www.construx.com>  |

Sites Internet qui donnent des exemples de modèles de spécifications d’interface d’utilisateur

|  |  |
| --- | --- |
| **Modèle** | **Source** |
| Utilisation spécifique d’interface  | <http://www.hcirn.com/tutor/docs/uitempl.php> |

## 5.5 Modèle de spécifications des tests

Pour être utilisé dans une feuille d'Excel structurée, par exemple :

|  |  |
| --- | --- |
| **Type d'application** |   |
| **Mode d'opération** |   |
| **Sujet** |   |
| **Groupe visé** |   |
| **Types de média à être utilisés** |   |

|  |  |  |
| --- | --- | --- |
| **Machine d’un client** | **Caractéristiques du système logiciel (software)** | **Version** |
| Système d'exploitation |   |   |
| Navigateur |   |   |
| Exigences du navigateur |   |   |
| Protocole de communication |   |   |
|   | **Caractéristiques du système hardware** | **Capacité** |
| Processeur |   |   |
| Disque fixe  |   |   |
| Mémoire RAM |   |   |
| Résolution d'écran |   |   |
| Multimédia | Applicable seulement en Multimédia |   |
| Vitesse CD ROM |   |   |
| Carte vidéo |   |   |
| Carte de son |   |   |
| Accélérateur 3D |   |   |
|   | **Caractéristiques du système hardware additionnées par le client** | **Capacité** |
| Imprimante |   |   |
| Numérisateur |   |   |

|  |  |  |
| --- | --- | --- |
| **Machine d’un client** | **Caractéristiques du système logiciel du serveur Web (software)** | **Version** |
| Système d'exploitation |   |   |
| Serveur Web |   |   |
| Langage de programmation |   |   |
| Exigences du langage  |   |   |
|   | **Caractéristiques du système logiciel du serveur des données (software)** | **Version** |
| Système d'exploitation |   |   |
| Base de données |   |   |
|   | **Caractéristiques du système hardware du serveur Web** | **Capacité** |
| Processeur |   |   |
| Disque fixe |   |   |
| Mémoire RAM |   |   |

Gabarit de la table des matières du Rapport de tests

1. Objectif du test ………………….
2. Résultats des tests ………………..
	1. Exigences vérifiées ou échoués ………………..
3. Commentaires et recommandations ………………..
	1. Échecs critiques ………………..
	2. Échecs sérieux ………………..
	3. Échecs mineurs ………………..
4. Intégration de composants
	1. Registre des tests
	2. Documentation des tests
	3. Fiches techniques des tests
	4. Analyse et conclusions ………………..
5. Fichier des données initiales ………………..
6. Conclusions ………….…….

##

## 5.6 Modèle pour la livraison du produit final

Exemple de formulaire d’instructions de livraison

|  |
| --- |
| **Formulaire d’instructions de livraison** |
| **Identification du projet ou nom du client : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Préparé par (Nom Initial) : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Date** (aaaa-mm-jj) : **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Identification des livrables** (par exemple : hardware, software, documentation etc.) **:** |
| **Exigences de livraison :**  |
| **Séquence de tâches à être exécutées :** |
| **Versions applicables :** |
| **Critères d’acceptation :**

|  |  |
| --- | --- |
| **Critères d’acceptation**  | **Date où le critère a été satisfait****(aaaa-mm-jj)** |
| **1** |  |
| **2** |  |
| **3** |  |

 |
|

|  |  |
| --- | --- |
| **Composants logiciels** | **Information de version**  |
|  |  |
|  |  |
|  |  |

 |
| **Procédures de récupération et sauvegarde:** |
| **Approuvé par :****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **Chef de projet Client ou personne représentante de la compagnie****Date** (aaaa-mm-jj) **:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

Exemple de formulaire d’acceptation de la livraison

|  |
| --- |
| **Formulaire d’acceptation**  |
| **Identification du projet :** Préparé par (nom initial) : **Date** (aaaa-mm-jj) : |
| **Critères d’acceptation :**

|  |  |
| --- | --- |
| **Critères d’acceptation** | **Date où le critère a été satisfait** **(aaaa-mm-jj)** |
| **1** |  |
| **2** |  |
| **3** |  |
| **4** |  |
| **5** |  |

 |
| **Signatures d’acceptation :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Composants logiciels** |  **Information de versions** | **Date d’acceptation du client** | **Signature du client** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

 |
| **Signatures :****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **Chef de project Client ou personne représentante de la compagnie** **Date** (aaaa-mm-jj) **:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

#### Outils

Cette section donne des informations sur les sources des outils qui pourraient aider à supporter certaines des activités du processus de l’implémentation logicielle.

**Intégration et tests du logiciel**

Pour assurer la gestion des défauts du produit, il est nécessaire d’implémenter un système, de préférence un système Web, qui permet de faire un suivi des problèmes et défauts identifiés d'un produit et les demandes associées à ceux-ci.

Comme des caractéristiques clés, ce système doit permettre :

* Identification automatique d’incidents
* Attachement de documents
* Produire des publications via courrier électronique
* Maîtriser les changements
* Produire des requêtes simples et interactives
* Classification d’incidents par : Type, Cause, Sévérité.

Il existe plusieurs outils de tests “open source” sur Internet, voici un site Web qui en offre quelques-uns :

<http://www.opensourcetesting.org/functional.php>

**Livraison du produit**

* Outils de contrôle de version comme CVS (Concurrent Versions System) ou SVN (version management).
* Dépôts disponibles en source libre : par exemple Google Doc.
* Outils de gestion de configuration.

# 6. Exemple du cycle de vie de la mise en œuvre du logiciel

***Avertissement***: Cette section fournit quelques représentations graphiques des exemples du cycle de vie de développement des logiciels. Ces exemples sont fournis pour aider le lecteur à l’implémentation de son propre cycle de vie de développement logiciel en l’adaptant au contexte de son projet informatique et ses contraintes.

###

### Exemple du cycle de vie de la mise en œuvre du logiciel

Celui-ci est un exemple qui utilise le patronSPEM de Microsoft Visio (http:// www.pa.icar.cnr.it/cossentino/FIPAmeth/docs/SPEM.vss) pour produire ce diagramme.



Figure 3 Exemple du cycle de vie de la mise en œuvre du logiciel

# 7. Listes de vérification

## 7.1 Liste de vérification des exigences du logiciel

Cette liste pour les exigences logicielles est basée sur [MCC 04]

|  |  |
| --- | --- |
| **RS 1 Testable** | Toutes les exigences sont vérifiables (objectivement) |
| **RS 2 Complet** | Est-ce que toutes les exigences sont complètes?  |
| **RS 3 Correct** | Les exigences doivent être correctes (par exemple, exprimer exactement les exigences du client) |
| **RS 4 Unique** | Les exigences doivent être établies seulement une fois. |
| **RS 5 Simple** | Les exigences doivent être détaillées de façon élémentaire |
| **RS 6 Scope** | Toutes les exigences font parties de la portée |
| **RS 7 Haut Niveau** | Les exigences doivent être établies en termes des besoins finaux, sans concevoir les moyens (solutions)  |
| **RS 8 Sans équivoque** | Les exigences doivent être formulées en termes qui ne peuvent être interprétées que dans un seul sens. |

## 7.2 Liste de vérification pour l’identification de composants logiciels

|  |  |
| --- | --- |
| **RS 1 Testable** | Tous les composants sont vérifiables (objectivement) |
| **RS 2 Complet** | Est-ce que tous les composants sont complets?  |
| **RS 3 Correct** | Les composants doivent être corrects (par exemple, exprimer exactement les exigences du client) |
| **RS 4 Unique** | Les composants doivent être établis seulement une fois. |
| **RS 5 Simple** | Les composants doivent être détaillés de façon élémentaire |
| **RS 6 Lié (*linked*)** | Les composants sont liés aux sous-composants  |
| **RS 7 Associé** | Les composants doivent être associés à d’autres composants. |
| **RS 8 Sans équivoque** | Doit être énoncés dans des termes qui ne peuvent être interprétés que dans un sens seulement. |

## 7.3 Liste de vérification pour la construction du logiciel

Liste de vérification de révision du code

Note: Cette liste de vérification peut être adaptée au langage de programmation que vous utilisez.

|  |  |
| --- | --- |
| **Matière** | **Description** |
| **Complète** | * Vérifier que toutes les fonctions dans la conception ont été codées et que toutes les fonctions nécessaires et des procédures ont été implémentées.
 |
| **Logique** | * Vérifier que le flux de programme, les procédures et la logique de fonctionnement sont cohérents la conception détaillée.
 |
| **Boucles** | * Faire une simulation manuelle de toutes les boucles et les procédures récursives qui n'ont pas été simulées dans la revue détaillée du dessin ou l'inspection.
* Assurer que chaque boucle est correctement amorcée et terminée.
* Vérifier que chaque boucle exécute le nombre de fois correct.
 |
| **Appels** | * Vérifier chaque fonction et l'appel de procédure pour assurer qu'il corresponde exactement à la définition des formats et des types.
 |
| **Déclarations** | Vérifiez que chaque variable et paramètre : - Ont exactement une déclaration.- Sont seulement utilisé dans sa portée déclarée.- Sont orthographié correctement partout où ils sont  utilisés. |
| **Initialisation** | Vérifier que chaque variable est initialisée. |
| **Limites** | * Vérifier toutes les variables, les tableaux et les index pour assurer que leur utilisation n'excède pas les limites déclarées.
 |
| **Début-Fin** | * Vérifier toutes les paires de début-fin ou leurs équivalents, y compris les cas où ifs emboîtés pourraient être mal interprétés.
 |
| **Booléen** | * Vérifier les conditions booléennes.
 |
| **Format** | * Vérifier le format d'instruction, l'orthographe et la ponctuation de chaque ligne de programme.
 |
| **Indicateurs** | * Vérifier que tous les indicateurs sont correctement utilisés.
 |
| **Input-output** | * Vérifier tous les formats input-output.
 |
| **Orthographe** | * Vérifier que chaque variable, paramètre et index est correctement orthographié
 |
| **Commentaires** | * Assurer que les commentaires sont précis et selon la norme.
 |

Cette information provient du site de la société Construx:

<http://www.construx.com/Page.aspx?nid=208>

## 7.4 Liste de vérification pour la livraison du produit final.

Caractéristiques des critères d’acceptation d’une TPO

* Les produits sont étiquetés correctement, montrant le nom, la date de sortie et le numéro de version correct.
* L'étiquetage du produit, y compris l'emplacement de livraison et le personnel d'acceptation de produit (si applicable), a été établi auparavant.
* Le produit logiciel du dépôt d’information du projet est conforme aux instructions de livraison.
* Le logiciel à être livré est la dernière version du logiciel qui se trouve au dépôt d’information du projet.
* Le document de description de version a été inspecté.
* Le document de description de version est inclus avec les médias de soutien.
* Tous les tests ont été exécutés avec succès.
* Toutes les erreurs ont été corrigées.
* Toute la documentation a été mise à jour.
* Le manuel de l'utilisateur a été inspecté.
* Le manuel de l'utilisateur est inclus avec les médias de soutien.
* Tous les sujets dans les instructions de livraison approuvées ont été couverts.
* Le formulaire de registre d'acceptation a été mise à jour et est prêt pour la signature.
* Les informations nécessaires pour la livraison (par exemple, l'adresse, le nom du client) ont été vérifiées avant la livraison.
* Votre client a été informé quand une livraison sera exécutée.
* Le client vous a informé que ses préparatifs de la livraison ont été complétés.

Caractéristiques des critères d’acceptation du client

* Vérifier que les composants logiciels livrés suivent les instructions de livraison.
* Vérifier que les composants logiciels livrés accomplissent les exigences initiales.
* Vérifier que la documentation attendue, les rapports de test, l’information de dessin et la conception et le matériel de formation ont été livrés.
* Vérifier que les composants logiciels livrés opèrent dans son environnement destiné (tant technique qu'organisationnel) et sont acceptables pour les utilisateurs.
* Vérifier que le formulaire d’acceptation a été révisé et signé.

# 8. Références

|  |  |
| --- | --- |
| **Nom** | **Référence** |
| [MCC 04] | Steve McConnell, Code Complete, Second Edition, Redmond, Wa.: Microsoft Press, 2004. |
| [IEEE 1012-2012] | [IEEE 1012-2012 IEEE Standard for System and Software Verification and Validation](http://shop.ieee.org/ieeestore/) |
| [ISO/IEC 12207]  | ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes. |
| [ISO/IEC 15289] | ISO/IEC 15289:2011 Systems and software engineering - Content of systems and software life cycle process information products (Documentation) |
| [ISO/IEC 29110-5-1-1] | ISO/IEC TR 29110-5-1-1:2012, Ingénierie du logiciel - Profils de cycle de vie pour très petits organismes (TPO) - Partie 5-1-1: Guide de gestion et d'ingénierie:: Groupe de profil générique: Profil d'entréeCe document est disponible gratuitement sur le suite suivant:<http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html> |
| [ISO/IEC/IEEE 24765] | ISO/IEC/IEEE 24765:2010 Systems and software engineering vocabulary.Une version électronique du vocabulaire est disponible à: <http://pascal.computer.org/sev_display/index.action> |
| [IEEE 1233-1998] | IEEE 1233-1998 Guide for Developing System Requirements Specifications. |

# 9. Formulaire d’évaluation

|  |
| --- |
| **Trousse de déploiement - Implémentation Logicielle – Profil d’entrée V 0.4**Vos commentaires nous permettront d'améliorer cette trousse de déploiement; vos suggestions sont les bienvenues. |
| 1. **Est-ce que le contenu de cette trousse de déploiement vous satisfait?**

 Très satisfait *Satisfait*  *Ni Satisfait ni insatisfait*  In*satisfait*  *Très* In*satisfait* |
| 1. **Par rapport à la SÉQUENCE dans laquelle les sujets sont discutés, vous êtes ?**

 Très satisfait  Satisfait  Ni Satisfait ni insatisfait  Insatisfait  Très Insatisfait |
|  **3. Par rapport à l’APPARENCE/FORMAT de cette trousse de déploiement, vous êtes?**  Très satisfait *Satisfait*  *Ni Satisfait ni insatisfait*  In*satisfait*  *Très* In*satisfait* |
|  **4. Considérez-vous que cette trousse contient des sujets inutiles? (décrivez-les S.V.P.)** |
|  **5. Considérez-vous que cette trousse manque de sujets importants? (décrivez-les S.V.P.)*** Sujet proposé :
* Justification du nouveau sujet:
 |
|  **6. Considérez-vous que cette trousse a des erreurs?*** + S’il vous plaît indiquez:
		- * Description de l’erreur:
			* Position de l’erreur dans le document (section #, figure #, table #) :
 |
|  **7. Avez-vous d’autres informations ou commentaires?** |
|  **8. Est-ce que vous recommanderez cette trousse de déploiement à un collègue d’une autre TPO?** *Certainement*  *Probablement*  *Peut-être*  *Probablement non*  *Certainement non* |

**Optionnel**

* Nom:
* courriel : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Remplissez le formulaire et envoyez-le au courriel**: claude.y.laporte@etsmtl.ca