Aide à la rédaction des artefacts

LOG3900

Automne 2021

Table des matières

Objectif du document

Ce document vous est fourni dans le but de vous aider dans la rédaction des différents artefacts propre à votre projet intégrateur. Les conseils et recommandations proférés à même ce document sont basés sur les erreurs les plus courantes pour chaque artefact. Ainsi, nul besoin de mentionner que la lecture attentive de ce document et l'application des conseils et recommandations qu'il contient vous permettront d'augmenter significativement la qualité de vos artefacts. Ces commentaires s'ajoutent aux instructions déjà reçues en lien avec la rédaction des artefacts. Ainsi, ils n'indiquent pas quoi faire, mais vous conseillent sur ce qui devrait se retrouver dans chaque section et sous-section de vos artefacts, en plus des éléments déjà demandés et implicites à chaque artefact.

La structure du présent document est fort simple. Pour chaque artefact, les commentaires et recommandations sont classés par section et sous-section, en se basant sur l'organisation du contenu propre à chaque gabarit. Sachez qu'il n'existe pas nécessairement de commentaire pour toutes les sections et sous-sections d'un même artefact. Seules les sections jugées problématiques ont été analysées afin de vous fournir des précisions. De plus, certains commentaires et recommandations concernent l'ensemble d'une section, tandis que d'autres précisent spécifiquement une sous-section. Les crochets [ ] indiquent la portée des commentaires et recommandations pour chaque artefact.

Attention! Le respect des éléments listés ci-dessous est fondamental au maintien de la qualité de vos artefacts. Il serait difficile d'insister davantage sur l'importance du respect des éléments de cette liste. Les éléments de cette liste concernent tous les artefacts de votre projet intégrateur. En d'autres mots, l'application de ces conseils et recommandations est directement proportionnelle à la qualité de vos artefacts.

- Le maintien d'une mise en forme impeccable et identique à travers tous les artefacts;
- Le maintien d'une numérotation et d'une sous-numérotation cohérente et incrémentale;
- Le maintien d'une cohérence dans les titres et sous-titres;
- Le maintien d'un ordre logique de présentation du contenu;
- La traduction des anglicismes, lorsque pertinent;
- L'utilisation d'un type de puce unique (ex.: - ou •);
- Un glossaire complet et en ordre alphabétique;
- Une structure d'écriture identique, constante et adéquate;
- L'utilisation du même temps de verbe approprié;
- L'emploi du même verbe dans des contextes similaires et/ou identiques;
- L'emploi d'une seule couleur de texte;
- L'emploi d'un niveau de langage soutenu.

SRS

[Section 1. Introduction]

Veillez à mettre à jour les définitions, acronymes et abréviations une fois la rédaction complétée.

[Section 2. Description globale]

[Sous-section 2.1. Caractéristique des utilisateurs]
Veillez à préciser les caractéristiques des utilisateurs, c'est-à-dire des éléments tels que leur tranche d'âge, ainsi que leur type d'expérience.
Par exemple, on fera la distinction entre un usager avec une expérience spécifique au domaine de l'application, un étudiant en génie, ou même un étudiant dans un domaine non technologique.

[Sous-section 2.2.1. Interface usagers]
Veiller à définir de manière sommaire les interfaces des clients et du serveur.

[Sous-section 2.2.2. Interfaces matérielles]
Veillez à considérer tout type d'interface matérielle, des écrans en passant par les claviers.

[Sous-section 2.2.3. Interfaces logicielles]
Veillez à considérer tout type d'interface logicielle. Les éléments tels que les librairies utilisées (que ce soit pour l'authentification, la gestion des transactions, etc.), les systèmes d'exploitation et leur version, ainsi que les cadriciels (*framework*) sont souvent oubliés.

[Sous-section 2.2.4. Interfaces de communication]
N'oubliez pas d'inclure les éléments liés au réseau, ainsi qu'à ses diverses couches (p. ex.: couche physique).

[Sous-section 2.3. Contraintes générales]
N'oubliez pas d'inclure des éléments de contraintes liés à des aspects matériels (mémoire, persistance, réseau, taille, mémoire vive, etc.) et logiciels (ports utilisables sur le réseau, etc.).

[Sous-section 2.4. Hypothèses et dépendances]
Veillez à préciser les facteurs assumés qui pourraient affecter les exigences mentionnées dans votre SRS. Pour les hypothèses, ça peut inclure les composantes externes qui seront utilisées. Par ailleurs, vous pouvez mentionner les dépendances de votre projet sur des facteurs externes, comme, par exemple, des composantes qui pourraient être réutilisées dans un autre projet.

[Section 3. Exigences fonctionnelles]

Les exigences fonctionnelles expliquent ce que le système doit pouvoir faire. De ce fait, plusieurs directives sont à appliquer afin de maintenir une cohérence, une uniformité et une clarté au travers de ces exigences. Ainsi, la forme est tout autant importante que le contenu.

Veuillez vous assurer que vos exigences soient assez détaillées pour qu'un développeur puisse prendre une exigence comme demande Redmine (JIRA, ou autre) et commencer l'implémentation sans trop se poser de question. Pour ce faire, explicitez les fonctionnalités de manière générale et faites attention de donner des détails lors de l'utilisation d'adjectifs.
Ex.: Qu'est-ce que dynamique veut dire concrètement? Sophistiqué? Adapté à l'utilisateur? Cohérent? Ces adjectifs ne veulent concrètement rien dire s'ils ne sont pas explicités.

Veillez à réduire les ambiguïtés. Par exemple, il est suggéré d'expliciter des éléments tels que: les détails de gestion des profils, les formats de fichier, les protocoles, leur version, les mécanismes de recherche, les temps d'actualisations des interfaces, les types de notification, le nombre d'utilisateurs, les types de visibilité, les paramètres, les actions, les modifications, les ajouts, etc. Bref, soyez le plus précis possible, et faites attention aux sous-entendus.

Veuillez vous assurer que votre design est indépendant d'une technologie en particulier. Autrement dit, vous ne devez pas mentionner les éléments d'interface utilisateur. Cela révèle un mauvais niveau d'abstraction. Une exigence fonctionnelle s'intéresse davantage à ce que le système doit pouvoir faire.

Veillez à maintenir l'atomicité des exigences. Autrement dit, une exigence ne devrait jamais être composée, c'est-à-dire qu'elle ne devrait pas pouvoir être scindée en plusieurs exigences atomiques.
Ex.: Il est possible de clavarder en mode fenêtré ou mode intégré et un bouton permet d'alterner entre ces deux modes.
devrait être
3.X.X.X Le système doit permettre à l'utilisateur de clavarder dans un mode fenêtré.
3.X.X.X Le système doit permettre à l'utilisateur de clavarder dans un mode intégré.
3.X.X.X Le système doit permettre à l'utilisateur d'alterner entre le mode fenêtré et le mode intégré.

Veillez à éviter les redondances.
Ex.: Une fois les deux entrées approuvées par le système, l'utilisateur est enregistré dans la base de données et pourra s'authentifier.
versus
Le système doit permettre à un nouvel utilisateur de s'enregistrer pour authentification future.
Cela revient à dire la même chose. Une seule de ces exigences devrait donc se retrouver dans votre document.

Veillez à maintenir une traçabilité pour chaque exigence. Pour ce faire, assurez-vous d'associer un identificateur unique à chaque exigence.
Ex.: 3.1.1.1 Le système doit permettre à un utilisateur de se créer un compte. (Ici, l'identificateur unique pour cette exigence est 3.1.1.1)

Veillez à utiliser une structure d'écriture adéquate. Il est recommandé que chaque exigence ait la forme suivante: « Le système doit permettre à l'utilisateur de ... »
Il est recommandé d'utiliser le verbe devoir.

Veillez à privilégier un style direct. Autrement dit, l'utilisation de la négation est à proscrire.
Ex. Le système vérifie si le nom d'utilisateur entré n'est pas déjà présent dans la base de données.
devrait être
3.X.X Le système doit vérifier l'unicité du nom d'utilisateur.

Veillez à privilégier le singulier à travers toutes vos exigences. Le pluriel est accepté lorsque c'est vraiment nécessaire. Si vous employez le pluriel, chiffrez.
Ex. Plusieurs canaux de discussion peuvent être créés par les utilisateurs.
devrait être
3.X.X.X Le système doit permettre à un utilisateur de créer un canal de discussion.

[Section 4. Exigences non-fonctionnelles]

Les commentaires énoncés précédemment pour les exigences fonctionnelles sont à appliquer aux exigences non-fonctionnelles. S'ajoutent à ces derniers les commentaires et recommandations spécifiques aux exigences non-fonctionnelles se trouvant ci-dessous.

Veillez à écrire des exigences vérifiables et objectives. Autrement dit, évitez les exigences non vérifiables, dures à vérifier, composées, conflictuelles ou contradictoires. Pour ce faire, utilisez des métriques quantitatives. Les métriques quantitatives inutiles, trop longues, non réalistes, non représentatives de la réalité ou non adaptées aux cas d'utilisations sont à proscrire. Ainsi, veillez à utiliser des métriques quantitatives vérifiables.
Ex.: adéquat, minimum, presque immédiatement, facile à suivre, standard, sont des termes subjectifs, et sont donc à éviter. Même chose pour conserver, garder intact, etc.
Ex.: Lors de l'utilisation d'un pourcentage, préciser l'intervalle de temps (50% de temps par année, par mois, par jour?).

Veillez à utiliser une structure similaire au format des exigences fonctionnelles. Utilisez une formule du type « Qui/Quoi doit… » et écrivez ces exigences comme si elles étaient finales, donc directement reliées au produit visé.

Document d'architecture logicielle

[Section 3. Vue des cas d'utilisation]

Veillez à ce que chaque cas d'utilisation (schématisé par une bulle dans un diagramme) représente en réalité une action (et non simplement un état) et soit composé de plusieurs étapes (qui ne sont pas indiquées dans le diagramme). Autrement dit, chacun de vos cas d'utilisation doit impliquer un verbe ou une action.
Ex.: « Fin de la partie après 5 minutes de joute » n'est pas un cas d'utilisation.

Veillez à ce que chaque cas d'utilisation comporte une frontière (entre ce qui fait partie du système et ce qui n'en fait pas partie). N'oubliez pas que deux cas d'utilisation ne peuvent pas être liés sans stéréotype.

Veillez à vérifier le sens de vos flèches « extends ».

[Section 4. Vue logique]

Veillez à présenter un diagramme de paquetages (ou diagramme de classes, selon le contexte), par paquetage. Vous devriez donc avoir autant de diagrammes de paquetages que d'éléments dans votre tableau.

Veillez à présenter, si nécessaire, un diagramme de paquetages de haut niveau afin d'augmenter la compréhension de votre architecture. Un diagramme de paquetages de haut niveau est parfois pertinent (notamment pour les relations entre ses paquetages). Le cas échéant, veillez à préciser les liens entre les paquetages.

Veuillez éviter des encapsulations portant à confusion.
Ex.: Un paquetage "Client lourd" dans un paquetage "Client lourd" va nécessairement finir par semer la confusion.

Veuillez libeller correctement vos diagrammes de classes et/ou vos diagrammes de paquetages, et référer à ces derniers de la bonne manière dans vos tableaux (paquets versus classes). Il est conseillé de revoir la distinction entre paquetage et classe.

Veillez à bien respecter UML, notamment concernant les relations entre les paquetages et les classes telles que association, agrégation, composition, généralisation, dépendance et abstraction.

[Section 5. Vue des processus]

Un diagramme de séquence doit présenter une séquence d'interactions entre les acteurs et le système. Assurez-vous de bien comprendre les concepts suivants, souvent moins bien compris: bande d'activation (aussi appelé *execution specification*), fragments combinés et messages de retour. Les fragments combinés impliquent un opérateur (les plus courants étant *alt*, *opt* et *loop*) et des opérandes.

Veillez à utiliser des messages informatifs, et à bien respecter UML, notamment, le type de trait pour une réponse (message de retour) et l'ordre chronologique des messages.

[Section 6. Vue de déploiement]

Veillez à maîtriser et revoir les notions d'UML lié aux diagrammes de déploiement.

[Section 7. Taille et performance]

Veillez à préciser et détailler au meilleur de votre capacité les éléments de cette section. Pour ce faire, il est conseillé de répondre à des questions du genre: En quoi votre architecture permet de satisfaire pleinement l'utilisateur? Comment l'évaluez-vous?

Protocole de communication

[Section 1. Introduction]

Veillez à présenter l'objectif du document, ainsi que les différentes sections, puisque ce document se doit d'être adapté selon vos besoins. Ceci implique qu'il est acceptable d'ajouter, en plus des sections requises, de nouvelles sections.

[Section 2. Communication client-serveur]

Veuillez justifier l'utilisation des technologies nécessaires à votre communication client-serveur. De plus, au besoin, il est parfois utile d'appuyer vos propos à l'aide d'un ou plusieurs diagrammes. Il n'est cependant pas nécessaire de discuter des détails de l'implémentation.

Veillez à mentionner pour quelles parties spécifiques et quelles fonctionnalités de votre application les technologies mentionnées dans cette section seront utilisées.
Ex.: Les fonctionnalités XX et YY utiliseront SocketIO, tandis que le REST API sera utilisé pour ZZ...

Veuillez vérifier la cohérence de votre architecture réseau, en rapport aux besoins de votre application, mais aussi aux protocoles utilisés et à la forme des paquets qui en découle.
Ex.: Si vous faites référence à l'utilisation de requêtes HTTP, la forme des paquets doit représenter cette architecture.

Veuillez faire mention aux types de données transférées, et ce pour chaque technologie et/ou protocole utilisé.
Ex.: JSON, strings, bytes

[Section 3. Descriptions des paquets]

Veillez à détailler le contenu des différents types de paquets utilisés au sein de **votre** protocole de communication. En d'autres mots, veuillez spécifier tous les paquets nécessaires pour que les clients et le serveur puissent communiquer. Le but de cette section est d'établir de manière réelle et détaillée la structure, le contenu et le type de tout message utilisé au sein de votre application. C'est donc à partir de cette section que vous pourrez directement implémenter vos paquets. Ainsi, il est important d'inculquer une logique, un fil conducteur, une structure récurrente à vos paquets, pour en faciliter l'implémentation et la compréhension.

Veillez à couvrir la description des paquets pour l'ensemble des technologies et protocoles mentionnés à la section 2. Ce faisant, veillez à spécifier les éléments techniques propres à chaque technologie/protocole. Ceci concerne des éléments tels que la spécification d'une modification, les paramètres de la modification et leur type, les réponses possibles du serveur et leur format, les différents paquets pour la gestion des canaux de communication. Évitez les exemples génériques et sans lien évident avec le projet.

Ex.: Pour chaque appel REST, veuillez spécifier l'URL d'accès ainsi que les paramètres et les réponses attendues.
Si vous décrivez une méthode getImages pour REST, assurez-vous d'être consistant avec cette méthodologie. Ainsi getImages devra être de type GET.

Veillez à décrire toutes les requêtes possibles et toutes les réponses possibles pour chacune de ces requêtes.

Veuillez vous assurer de définir clairement les paramètres de vos paquets. Ceci inclut des éléments tels que les structures et/ou objets faisant partie des paramètres de vos paquets.
Ex.: Si l'un des paramètres est une image, vous devez spécifier les propriétés de l'image.
Si vous utilisez une structure Stroke, spécifiez ce qu'elle contient.
Si vous référez à une structure Text et une structure String, spécifiez-en les différences et les spécificités.
Pour un outil XX, spécifiez la structure de l'objet JSON concerné.

Plan de projet

[Section 3. Gestion et suivi de l’avancement]

[Sous-section 3.1. Gestion des exigences]
Veillez à expliciter des mécanismes concrets permettant non seulement d’identifier les changements, mais aussi d’assurer que les changements aux exigences sont bien gérés avec les outils à votre disposition. En d’autres mots, il s’agit d’expliciter la manière dont vous gérez les changements aux exigences.

[Sous-section 3.2. Contrôle de la qualité]
Veillez à détailler vos processus pour entreprendre vos actions correctives. Ici, il s’agit d’expliciter quand et comment vous entreprendrez une action corrective.

[Sous-section 3.3. Gestion de risque]
Veillez à identifier des risques principalement technologiques de votre système en particulier, en plus des risques humains (communication, planification, mauvaise compréhension), qui eux sont moins importants.

Veillez à éviter les risques très généraux. En effet, plutôt qu’avoir des risques du type "Panne du serveur", "Plantage de l'application" et "Perte de connexion", il est plus pertinent d'indiquer les risques spécifiques à votre produit et aux technologies utilisées qui pourraient causer ces pannes.

Veillez à identifier des facteurs qui sont des **métriques du système** telles que la fiabilité, la disponibilité du système, la performance, la cohésion, le couplage, etc.
Ex.: Une "mauvaise connexion internet" peut être mesurée par la métrique de latence du système (faisant elle-même partie de la métrique “performance”).

Veillez à identifier des stratégies de gestion détaillées et des actions proactives pour gérer le risque, notamment pour atténuer l’ampleur/impact du risque. Il est recommandé de limiter les stratégies réactives.
Ex.: “Redémarrer vos clients et votre serveur” est une stratégie réactive présentant peu d’intérêt.

[Sous-section 3.4. Gestion de configuration]
Veillez à établir des stratégies permettant de revoir et disposer d’un problème. Il est suggéré d’expliciter comment vous allez gérer les problèmes et les changements, en plus du mécanisme de soumission de ce dernier.

[Section 4. Échéancier du projet]

Veillez à bien découper et identifier vos lots de travail en fonction du ratio de 45h/crédit/personne prévu (donc 900 heures-personnes pour une équipe de 5 et 1080 heures-personnes pour une équipe de 6), et à respecter l’effort estimé. En effet, un lot n'est pas

une semaine de travail. Chaque ligne de vos tableaux devrait représenter des lots de travail distincts.

Veillez à maintenir une fine granularité dans le descriptif de vos tâches et leur estimation en heure.
Ex.: La tâche "Commencer le dessin collaboratif" n'est pas assez granulaire. Qu'est-ce que cela représente? Quelles tâches précisément allez-vous accomplir pour commencer le dessin collaboratif?
Ex.: 65h pour SRS et Exigences n'est pas suffisamment granulaire.

Veillez à présenter un échéancier complet, soit un échéancier comportant les éléments suivants: un ordre chronologique, les principaux lots de travail avec leur effort estimé en heure-personne et non pas en heure/personne (voir exemple), les dates de début et de fin des lots, les itérations, les jalons et les dates de tombées des livrables.
Ex.: Si vous jugez que 2 développeurs passeront 4h chacun sur une tâche, alors l'effort sera de 8 heures-personnes.

Veillez à faire la différence entre un lot et un jalon, et à identifier correctement ces derniers. Les lots doivent avoir des dates précises de début et de fin.

Plan de tests

[Section 2. Exigences à tester]

Veillez à identifier dans cette section ce que vous avez réellement l'intention de tester (cas d'utilisation, exigences fonctionnelles, exigences non-fonctionnelles, etc.). Prêtez attention à la traçabilité des tests (soit le lien entre le SRS et les tests qui seront éventuellement exécutés). Notez qu'il n'est pas pertinent de copier-coller intégralement le SRS dans cette section, vous pouvez y faire référence de manière directe.
Ex.:

Exigences	**Tests associés**
3.1.1. Gestion des profils	Test de fonction, test d'intégrité des données, test de performance
4.2. Fiabilité	Test de sécurité et de contrôle d'accès
...	...

[Section 3. Stratégie de test]

Veillez à résumer vos stratégies de test et non définir vos cas de test. Il est important de ne pas confondre le document de cas de test (qui n'est pas demandé dans le cadre du cours) et plan de test. Il est recommandé d'étoffer les stratégies présentées.

Veillez à ne présenter que les types de tests que vous jugez pertinents. En d'autres mots, choisissez parmi les 11 types de tests proposés dans le gabarit ceux que vous allez réellement exécuter. Vous pouvez aussi en ajouter des supplémentaires (comme des tests unitaires, par exemple).

[Section 5. Jalons du projet]

Veillez à préciser l'effort relatif à chaque jalon de la **discipline de test** (et non du projet en général). L'effort de chaque jalon doit être exprimé en heure-personne.

Résultats de tests logiciels

[Section 2. Sommaire des résultats]

Veillez à être cohérent face au plan de test. Les dates d'exécution doivent être cohérentes avec les jalons prévus à la section 5 du plan de tests. En cas de disparités, de cas de tests ignorés ou qui échouent, fournissez des explications. Autrement dit, en plus des résultats eux-mêmes, expliquez ces derniers.