DocBook XML/XSL et des outils logiciels libres

pour produire une documentation technique de qualité à moindre coût

Documation, Paris, 16 mars 2004, Session C04 - Les logiciels libres : alternative aux progiciels ? Etude de cas

Sommaire

Contexte et résumé

Jaluna est une entreprise créée en août 2002 par d'anciens employés de Sun Microsystems inventeurs du système d'exploitation temps réel Chorus, pour développer, distribuer et assurer le support des composants logiciels de la gamme Jaluna (voir http://www.jaluna.com).

Cet article retrace l'évolution du système de production de la documentation des produits Jaluna mis sur le marché entre septembre 2002 et novembre 2003.

L'accent est mis :

1. sur les technologies et outils DocBook, et leur mise en oeuvre pour écrire et produire de la documentation technique à Jaluna.

2. sur les autres outils logiciels utilisés en équipe pour partager les tâches de documentation (rédaction, relecture, validation, production) entre des compétences complémentaires dans l'entreprise, tout en gardant la maîtrise de la qualité de résultats obtenus par un travail collaboratif.

Caractéristiques de la documentation technique Jaluna

La production de la documentation technique à Jaluna est très fortement intégrée au processus de développement et de qualification des produits logiciels. A Jaluna, on utilise le plus possible les mêmes outils de gestion de projet et de contrôle de la qualité pour le produit et pour sa documentation. C'est un choix de l'entreprise, stratégique avant même d'être économique (et en plus, il l'est !).

Depuis l'origine du système d'exploitation Chorus en 1988, jusqu'à celle du premier produit Jaluna en 2002, l'organisation du contenu de la documentation technique a bien sûr beaucoup évolué mais on a toujours gardé une structure globale sous forme de collections de manuels organisés suivant les tâches à réaliser par type d'utilisateur :

- description du produit par plateforme d'utilisation (références hardware, environnements logiciels, fonctionalités)

- guides d'installation par plateforme matérielle et logicielle

- guides de développement d'applications
- guides d'administration et de configuration du système
- manuels de référence (commandes, appels système, etc.)

On retrouve ce type d'organisation pour la documentation de la plupart des systèmes d'exploitation du marché (divers Unix, FreeBSD, Solaris, distributions Linux). Cette structure correspond actuellement encore aux attentes des utilisateurs qui y sont habitués.

La documentation Jaluna est fortement structurée. Dans une collection par produit, on trouve des parties communes s'appliquant quelle que soit la plateforme d'utilisation, et des parties spécifiques dépendantes de l'architecture matérielle et logicielle (processeur, système).

La documentation des produits Jaluna est susceptible d'être fréquemment rééditée pour introduire des fonctionalités nouvelles, ou documenter le portage du produit sur une nouvelle plateforme de développement. Le plus souvent, les parties communes, indépendantes de la plateforme, seront simplement réutilisées dans une nouvelle collection avec des documents spécifiques nouveaux ou modifiés.

Il est souvent nécessaire de faire des références croisées entre les documents d'une même collection.

La documentation des produits Jaluna est toujours livrée en même temps que le produit, dans un *package* qui s'installe sur le système de l'utilisateur à côté des sources du produit. Les documents de type *guide* sont disponibles en HTML et en PDF, les pages de type *manuel de référence* sont disponibles en HTML et consultables en-ligne avec la commande **man**.

A titre d'illustration, la documentation, en anglais, des produits Jaluna-2 sortis en novembre 2003 comporte 17 guides pour un total de 968 pages imprimables (le plus petit document comporte 18 pages, le plus gros en comporte 320). La collection de pages de type manuel de référence (*pages man*) comporte 805 fichiers. La collection de pages man Linux en comporte 2804.

Evolution du système DocBook de production de la documentation technique Jaluna

En bref (les grandes lignes)

Sun Microsystems ayant mis en open source la documentation de ChorusOS 5.0 au format SolBook SGML, l'idée de reprendre les documents ChorusOS en DocBook SGML pour les exploiter ensuite venait naturellement à l'esprit, puisque SolBook est une personnalisation de DocBook pour Sun.

Au démarrage de Jaluna, l'orientation des premiers développements autour de Linux et de la distribution Red Hat, ont conduit également très naturellement à essayer de re-produire la documentation ChorusOS avec les outils docbook OpenJade (db2*).

De cette façon (DTD DocBook, feuilles de style DSSL, outils de transformation en HTML et PDF OpenJade) on disposait des composants essentiels d'un système complet (et gratuit !) de production des documents ChorusOS.

En mettant en place le système pour produire la documentation du premier produit Jaluna (C5), quelques limitations sont apparues qui pouvaient être résolues en passant à un système "tout XML" (par exemple, il n'était pas possible d'établir des liens actifs entre les documents d'une même collection). De plus, il est apparu clairement que les développements DocBook SGML étaient sur le déclin à l'inverse de ceux de DocBook XML, des feuilles de style XSL, et des outils s'appuyant sur des transformations XSLT.

Dans une première étape transitoire, le système a gardé un pied côté JadeTex/DSSSL pour la génération de PDF, le temps d'essayer et de comparer plusieurs outils de production de PDF (PassiveTex a ainsi finalement été écarté au bénéfice de FOP).

Pour finir, le système installé pour la production des produits Jaluna-2 intègre complètement DocBook XML et DocBook XSL, avec les outils xsltproc et FOP, pour constituer un processus cohérent de production de la documentation produit Jaluna. Ce système est robuste, fiable, et par conséquent bien adapté à la documentation produit Jaluna qui est importante en nombre de documents et taille des documents.

L'évolution du système par étapes a permis de produire la documentation de plusieurs produits successivement dans un laps de temps réduit, en bénéficiant d'améliorations technologiques et de nouvelles fonctionalités du système de publication, à chaque étape, tout en conservant une présentation homogène des documents, commune aux différentes sorties de produits.

Parallèlement à la mise en place et à l'utilisation du système de rédaction et de production des documents, un mécanisme de *build* automatique quotidien et d'affichage des résultats, permet de valider en continu l'avancement des projets de documentation associés au développement des produits.

En détail (la chronologie)

Juillet – Septembre 2002

- Pendant l'été 2002, vérification de la faisabilité de la reprise de la documentation de ChorusOS 5.0 en SolBook SGML.
- Transformation par scripts des fichiers sources SolBook d'origine en DocBook SGML (déclarations, nom de produit, quelques tags).
- Mise en place d'un environnement de production de la documentation aux formats HTML et PDF, sous Linux. Utilisation de OpenJade, JadeTex (outils open source de James Clark) et des feuilles de style DSSSL (distribution libre standard de Norman Walsh).
- Personnalisation des feuilles de style DSSSL, pour un rendu au look Jaluna.
- Adaptations mineures du contenu des documents ChorusOS 5.0 pour C5 1.0 par les développeurs.

Octobre 2002

- Sortie du produit Jaluna-1 C5 1.0 et de la documentation associée.
- Formats HTML (voir http://www.jaluna.com) et PDF pour les guides. Format HTML seulement pour les pages man (pas PDF, pas man).
- Pas de liens entre documents C5 1.0 en HTML, comme cela existait pour la documentation ChorusOS chez Sun avec le format AnswerBook (voir http://docs.sun.com).
- Les guides C5 1.0 au format PDF ont une fenêtre de navigation dans Acroread (Bookmarks) et des liens actifs à l'intérieur du document. Ces qualités n'existaient pas chez Sun pour le format PDF des docs ChorusOS.
- Les figures ChorusOS 5.0 (EPS, non retraitable) n'ont pas été adaptées pour C5 1.0. Quelques problèmes de qualité du rendu des images en PDF.

Novembre – Décembre 2002

- Première formation interne sur DocBook pour la mise en route des développeurs à l'écriture de documents de type HOWTO (simples) pour la pre-release de Jaluna-2. Ce tutoriel est accompagné d'un guide publié en HTML et PDF sur le site interne de Jaluna (*Writing Documentation Using DocBook at Jaluna*).

  Les documents de type HOWTO sont rédigés par les développeurs Jaluna-2 et directement codés en DocBook SGML.
- Les documents au format HTML sont produits avec les feuilles de style DSSSL et les outils OpenJade (Linux), comme pour C5 1.0 (Jaluna-1).
- En parallèle, préparation de l'environnement DocBook XML et des feuilles de style XSL au look Jaluna, pour remplacer ultérieurement les outils OpenJade et les feuilles de style DSSSL.

Janvier 2003

- Deuxième formation interne et guide plus orientés sur la rédaction technique et le style (*Jaluna Style Guide for Writing Technical Documentation*).
- Sortie de la Pre Release Jaluna-2 et de la documentation associée.
- Format HTML seulement, pas de liens entre docs, nombre très limité de pages man (ordre de la dizaine) en HTML seulement.

Février – Juillet 2003

- Mise en place de l'environnement de production "tout XML" de la documentation. Utilisation de DocBook XML et des feuilles de style DocBook XSL (distribution standard de Norman Walsh). Utilisation de xsltproc pour générer HTML, et de FOP (Apache) pour générer PDF.

  Publication interne d'un mode d'emploi (*How to Write and Publish Jaluna-2 Release 1.0 Documents*).
- Personnalisation des feuilles de style DocBook XSL, pour un rendu au look Jaluna.
- Rédaction des guides Jaluna-2 1.0 en DocBook XML par les développeurs.
- Sorties successives de produits Jaluna-2 1.0 et des collections de documents associées.
- Formats HTML et PDF (avec Bookmarks et liens actifs) pour les guides.

  Les figures sont éditées à part (avec StarOffice,

GIMP ou XV) et intégrées aux documents en PNG ou GIF.

Formats HTML et man pour les pages man Jaluna et Linux.

- Liens actifs entre les documents en HTML d'une même collection (par produit/architecture processeur). Liens depuis les documents en HTML vers les pages man.

Août – Octobre 2003

- Un changement de la stratégie produits amène à réorganiser complètement l'espace des sources de la documentation.
- Le système de production, les outils, les formats générés, et la présentation des documents restent inchangés.

  Publication interne (*How to Write and Publish Jaluna-2 1.1 Documents*).
- Sorties successives de produits Jaluna-2/EL 1.1 et Jaluna-2/RT 1.1 et des collections de documents associées.

Des pistes pour la suite

- **Xincludes** (modularisation) : pour mieux gérer la généricité ou la spécificité des documents.
- **XML Catalogs** : pour optimiser l'utilisation et l'accès aux feuilles de style, et DTDs.
- **Makefiles** : pour améliorer le build automatique de la documentation.
- intégration avec un **CMS** pour avoir la possibilité de produire d'autres types de documents (par exemple : aide-en-ligne, supports de formation) à partir de la même source de contenus.
- **DocBook pour Eclipse** : pour intégrer la documentation en DocBook XML à la plateforme de développement produit.
- **RenderX XEP** : processeur XSL-FO (commercial) pour un meilleur rendu en PDF, si besoin.
- éditeur **WYSIWIG** pour écrire en DocBook XML, si besoin (suivre l'avancement de OpenOffice.org).
- éditeur **SVG** : pour des graphiques directement en XML.

Description du système DocBook actuel de production de la documentation technique Jaluna

Les composants du système

La DTD DocBook

C'est le composant standard de base du système, associé à des feuilles de style XSL, et à des processeurs XSLT pour générer les différents formats de document publiés.

Elle est installée dans `/usr/local/share/xml/docbook-xml`. La version courante est `DTD DocBook V4.2`.

La description complète (ô combien !) de tous les éléments (~ 400) et des balises associées, est dans l'incontournable *DocBook: The Definitive Guide*, de Norman Walsh.

Des outils d'édition, de gestion et de validation des fichiers natifs

- GNU `emacs+psgml` convient bien pour écrire des documents DocBook XML à Jaluna. L'expérience montre qu'après un certain temps d'adaptation, les rédacteurs-développeurs sont plus rapides et plus efficaces avec un éditeur XML textuel, qu'avec un éditeur WYSIWIG.

  En association avec un éditeur de texte, il est fortement recommandé d'utiliser, de faire utiliser, d'user et abuser d'outils de validation comme `aspell`, `xmllint`, `xsllint`.
- A Jaluna nous avons utilisé SCCS et TeamWare (commercial) pour le partage des tâches de rédaction technique.

  Ce système de contrôle de version de fichiers sources est parfaitement connu et maîtrisé par les développeurs à Jaluna, qui l'utilisent quotidiennement dans leur activité d'édition de code.

  CVS, Borges, sont des solutions libres et gratuites qui conviendraient aussi.
- Pour suivre le déroulement du plan de

production des documents Jaluna, enregistrer et tracer les résultats des revues et relectures de documents, nous utilisons le système de gestion de rapports d'anomalies : `BugZilla`, de Mozilla.

Comme pour SCCS, les développeurs Jaluna, utilisent BugZilla au jour le jour pour contrôler la qualité de leurs développements coopératifs. BugZilla est également utilisé à Jaluna pour les services de support et de maintenance des produits déjà commercialisés.

Les feuilles de style DocBook XSL

Les feuilles de style standard sont installées dans `/usr/local/share/xml/docbook-xsl`. La dernière version installée est `DocBook XSL V1.64.1`. Les nouvelles versions sont distribuées par Norman Walsh tous les deux à trois mois environ, elles sont décrites et annoncés dans *http://lists.oasis-open.org/archives/docbook-apps/* et sont disponibles sur *http://sourceforge.net/*.

Les feuilles de style `docbook-xsl` comportent un grand nombre de paramètres (> 200) qui permettent de contrôler finement le rendu final.

La personnalisation (*customization*) des feuilles de style standard est parfaitement expliquée en détail par Bob Stayton dans *DocBook XSL: The Complete Guide*.

Les feuilles de style personnalisées pour Jaluna sont archivées dans les arbres de sources des documents (répertoire `stylesheets`). Les feuilles de style adaptées ne sont pas autosuffisantes (*standalone*), elles font appel aux feuilles de style standard en utilisant leur modularité et les mécanismes XSL d'importation et d'inclusion.

Remarque: il est plus facile d'adapter les feuilles de style pour HTML que pour FO (PDF).

Les outils de transformation

Une fois que l'on a d'une part, un document écrit en DocBook XML, et de l'autre les feuilles de style pour le format HTML et pour PDF (par l'intermédiaire de FO), il ne manque plus que les processeurs qui traitent le tout pour restituer le document dans un format *lisible* :

- **xsltproc** (avec libxslt de Gnome, Daniel Veillard)

  Transforme directement DocBook XML en HTML.

  Transforme DocBook XML en FO. Ce langage définit l'apparence physique du texte et des images dans un document, et leur disposition sur une page.

- **FOP**

  Transforme les objets XSL-FO (*formatting objects* en PDF. C'est un outil Apache qui nécessite un runtime Java. Un outil en C++ est peut-être en cours de développement pour le concurrencer (à surveiller).

Chez Jaluna, la chaîne de production de documents fonctionne de la même façon pour Solaris et pour Linux.

Le cas particulier des pages man

Les pages man (*Reference Manual Pages*) des produits Jaluna-2 posaient un problème spécifique avec la juxtaposition de deux collections, les pages man Linux de la distribution Red Hat, et les pages man C5 (micro-noyau) de Jaluna.

Les pages Linux sont en source troff. Les pages Jaluna sont en source DocBook XML.

A partir des fichiers source des pages man Linux et Jaluna, on a voulu fournir, pour chacune des collections, le format HTML de chaque page, et le format troff pour utilisation de la commande **man**.

Il existe une transformation DocBook XML —> HTML pour les pages man avec les feuilles de style DocBook XSL standard. Mais cela ne suffisait pas pour faire toutes les transformations nécessaires :

- DocBook —> troff/man (pages Jaluna/C5)
- troff —> HTML (pages Linux)

On a choisi de :

1. Transformer les pages Jaluna/C5 (source DocBook XML) en troff/man avec la feuille de style standard `docbook-xsl/manpages/docbook.xsl`
2. Transformer les pages Jaluna/C5 et les pages Linux (source troff/man) en HTML avec le même outil pour un rendu similaire. On a adapté le programme

libre `vh-man2html` (Michael Hamilton) de façon à obtenir les résultats escomptés.

Le build automatique

Pourquoi ?

Parce que c'est le moyen d'assurer la qualité de résultats obtenus en répartissant les tâches de rédaction technique entre les développeurs.

Lorsque les tâches de rédaction et de relecture des documents sont réparties dans une équipe ou entre équipes, il est indispensable de faire des validations systématiques au fur et à mesure des modifications apportées dans les documents d'une collection. Cette pratique évitera de trouver au moment de la qualification du produit, des bugs de construction de la documentation d'un produit qui n'ont pas été détectés plus tôt, et qui seront plus difficiles à corriger que lorsqu'on les détecte au moment où ils interviennent.

La validation d'un document passe obligatoirement par sa lecture (inspection visuelle). Le build automatique de la documentation d'un projet (ou produit) au fur et à mesure de l'avancement permet cette vérification *à l'oeil nu*, en continu. Les résultats doivent être facilement accessibles, et souvent rafraîchis.

Les modifications faites dans les sources DocBook XML des documents, doivent d'abord être validées localement par leur auteur, avant qu'il les remonte dans l'arbre commun de sources des documents (sous TeamWare). Pour cela, il doit générer localement le document isolé dans un format *lisible*, HTML ou PDF. Cette première étape de validation par l'auteur ne fait pas partie de la procédure de build automatique, mais elle est absolument indispensable.

Indépendamment de cette vérification locale d'un document isolé, les modifications apportées à un document peuvent avoir des répercussions sur le build de la collection complète des documents pour un produit (par exemple, la rupture de liens entre les documents). Le build automatique global de la documentation technique d'un produit permet de détecter ce type d'accident.

Comment ?

La méthode que l'auteur du présent article a utilisée pendant la période de référence est grossière. J'ai simplement scripté l'enchaînement des commandes de génération des formats HTML et PDF de tous les documents d'une collection, lancé un traitement batch quotidien, et affiché les résultats sur une page web interne.

Chaque jour les auteurs, relecteurs, chef de projet, ou toute personne de l'entreprise interessée par l'avancement du travail de documentation d'un produit, peuvent consulter la version la plus à jour des documents en préparation.

L'examen des fichiers `log` des builds automatiques quotidiens apporte un premier niveau de renseignement sur la qualité de la présentation des documents, mais attention : cela ne remplacera jamais la lecture par l'homme !

En fonction de la gravité d'un défaut identifié lors d'une inspection visuelle, et de la difficulté à le résoudre dans la documentation avant la sortie du produit, on utilise le système BugZilla de Mozilla, pour enregistrer la description du problème, et suivre sa résolution.

TeamWare, pour la gestion des fichiers source en utilisation partagée, et BugZilla pour le suivi des erreurs dans la documentation (forme ou contenu), sont les principaux outils non-documentaires qui servent à assurer la qualité du travail en équipe sur la documentation à Jaluna. Ces mêmes outils sont utilisés pour le développement logiciel.

Pour améliorer la technique du build automatique de la documentation, on pourra sans peine être beaucoup plus *ingénieux*, et par exemple de mettre en place un mécanisme avec des `Makefiles`.

L'intégration avec Eclipse ouvrira certainement de nouvelles pistes pour l'intégration du processus de publication de la documentation avec celui du développement du produit.

Quelques leçons tirées de l'expérience

Apprendre à utiliser DocBook en l'utilisant

Cette recommandation est bien sûr valable pour beaucoup d'autres apprentissages que celui de DocBook XML. Chez Jaluna, l'auteur de cet article a aidé les développeurs-rédacteurs à gagner sur le délai d'appropriation avec :

- des formations internes collectives courtes,

- des guides d'utilisation simplifiés,
- des modèles commentés, des fichiers d'entités communes,
- une FAQ pour les problèmes les plus récurrents.

Lectures conseillées :

- l'article un peu ancien, mais toujours juste, de Michael Smith : *Take My Advice: Don't Learn XML*, dont l'une des conclusions est :

  "I am suggesting that you learn XML by actually using it, that is by using DocBook now, right away, to do structured authoring of your documents.",

- le *DocBook Demystification HOWTO* d'Eric Raymond.
- *DocBook, la quatrième dimension de la documentation technique* par Camille Bégnis.

- DocBook est une technologie robuste, souple, pérenne et facile à mettre en oeuvre pour le travail coopératif (contrôle de version, travail partagé).
- DocBook est "la" référence pour la documentation technique logicielle, même si il existe des DTDs et/ou schemas XML pour toutes sortes de besoins et de domaines.
- Les outils de transformation de DocBook XML libres et gratuits sont de plus en plus nombreux et performants.

  Il est facile de changer un outil du système pour un autre plus performant, ou qui présente des fonctionalités mieux adaptées, lorsque les outils que l'on compare implémentent la DTD DocBook standard.

  La vraie richesse du système de publication réside dans les données codées en DocBook XML, et donc, dans sa portabilité, pas dans les outils.
- En se généralisant dans le monde du logiciel libre et de Linux, DocBook apparaît de plus en plus comme un label de qualité de la documentation technique qu'il ne faut pas hésiter à utiliser, et à afficher comme ci-après (pardon, Monsieur Magritte [1] !).

Ceci n'est pas un pingouin

Ne pas perdre de vue les avantages de DocBook

- Unicité de la source, multiplicité des formats de publication.
- Séparation du contenu et du rendu pour une présentation homogène et des documents bien structurés.
- Réutilisation de l'information. On peut organiser l'information en modules qui peuvent être facilement placés dans des documents distincts. On pourra construire un document sur mesure à partir de briques élémentaires déjà prêtes. On pourra aussi facilement restructurer une collection de documents existante.

Ne pas perdre de vue les limites de DocBook

- Dans DocBook, il y a *book*. La DTD reste très marquée par la finalité de production sous forme imprimable, qui l'emportait sans doute largement à l'origine, il y a plus de dix ans. On regrette ainsi l'absence d'une unité d'infomation de base de type `sujet` (*topic*) dont on aura besoin pour publier facilement sous la forme aide-en-ligne (*online help*).

[1] Interprétation "libre et ouverte" de l'oeuvre de Magritte, *Ceci n'est pas une pomme*.

- La richesse de la DTD DocBook est aussi une faiblesse avec un très grand nombre d'éléments, qui intimide, ou pire effraie les utilisateurs. Pour pouvoir conserver les bénéfices de l'utilisation d'un standard largement reconnu, il faudra cependant résister le plus possible à la tentation de trop personnaliser la DTD, ou d'en concevoir une autre, différente.

  Certains utilisateurs réclament la modularisation de la DTD DocBook en sous-ensembles d'éléments, disjoints, spécifiques de classes d'utilisation documentaire prédéfinies. L'unique ensemble d'éléments communs ne comporterait plus qu'une centaine d'éléments.

- De l'avis même de Norman Walsh, il est grand temps de passer à *"DocBook V.next"* (dixit) et à une approche complètement repensée et modernisée. Même si le calendrier et la teneur de cette évolution n'ont pas encore été officiellement annoncés par le Comité Technique DocBook d'OASIS, des signes tangibles montrent que les travaux ont vraiment démarré le 1er janvier 2004 ! (voir http://norman.walsh.name/threads/refactorDocBook).

  Un grand principe de ce chantier de rénovation sera d'assurer que les documents existants, écrits avec DocBook *standard* et valides, puissent être facilement transformés en *"DocBook V.next"* avec XSLT.

Continuer de surveiller les progrès de la communauté DocBook

XML est une technologie qui évolue rapidement, ainsi que les outils qui en tirent parti. Ceci est vrai aussi pour la production de documentation technique avec DocBook XML. La chaîne de traitement dans son ensemble n'est pas encore arrivée à maturité complète, même si l'on peut considérer que le niveau de qualité atteint est déjà satisfaisant pour une utilisation professionnelle.

Il existe une large communauté d'experts dévoués à la cause DocBook, et une documentation pléthorique sur laquelle s'appuyer pour maintenir et faire évoluer l'environnement de production de la documentation produit Jaluna.

Le moyen préconisé pour suivre la progression de la technologie DocBook XML/XSL, est la surveillance de l'activité autour de listes de diffusion comme http://lists.oasis-open.org/archives/docbook-apps/.

En conclusion, un témoignage

“ La documentation des produits développés par Jaluna apporte une dimension essentielle dans l'acceptation de ceux-ci par les clients. Cette dimension inclut l'organisation générale de la documentation, son contenu, son accessiblité et sa présentation.

L'utilisation d'un système de production basé sur DocBook nous a permis, à partir d'une base de sources XML, de produire un ensemble de documentation cohérent disponible dans deux formats, le premier (HTML) adapté à l'utilisation au jour le jour des produits, incluant en particulier un mécanisme de navigation performant, le second (PDF) permettant d'imprimer des documents de référence.

DocBook, en structurant clairement la documentation, avec l'introduction de feuilles de style et d'outils de transformation paramétrables autorise une séparation des différentes fonctions liées à la production de la documentation : édition du contenu, présentation, création de références, organisation globale...

Le système DocBook nous a permis un travail collaboratif efficace entre la responsable du système de publication, de la qualité et de la cohérence globale de la documentation, et les ingénieurs en charge de la production du contenu.”

— Christian Jacquemot, Directeur Produits à Jaluna SA