Traduire
Revue française de la traduction

233 | 2015
Voyage en équipage

Du développement à la traduction : les projets en équipe d'OmegaT

Didier Briel

Édition électronique
URL : http://journals.openedition.org/traduire/739
DOI : 10.4000/traduire.739
ISSN : 2272-9992

Éditeur
Société française des traducteurs

Édition imprimée
Date de publication : 15 décembre 2015
Pagination : 26-32
ISSN : 0395-773X

Référence électronique
Didier Briel, « Du développement à la traduction : les projets en équipe d'OmegaT », *Traduire* [En ligne], 233 | 2015, mis en ligne le 15 décembre 2017, consulté le 05 mai 2019. URL : http://journals.openedition.org/traduire/739 ; DOI : 10.4000/traduire.739

Du développement à la traduction : les projets en équipe d'OmegaT

Didier Briel

Les outils de traduction permettant de collaborer en temps réel nécessitent habituellement des ressources conséquentes, aussi bien technologiques que financières. Avec une approche différente, issue des outils de développement, les projets en équipe d'OmegaT permettent à des traducteurs indépendants de s'approprier ces méthodes de travail.

Une brève histoire d'OmegaT

Contrairement à la traduction, le développement de logiciels relève rarement d'une activité solitaire. Afin d'élaborer un programme d'une certaine envergure et plus encore de le faire vivre, plusieurs intervenants sont généralement nécessaires.

Créé par Keith Godfrey, un Américain, et porté à bout de bras par Marc Prior, un Anglais vivant en Allemagne, OmegaT a réellement pris son envol technique sous la direction de Maxym Mykhalchuk, un Ukrainien étudiant en Italie, secondé pour la documentation par Jean-Christophe Helary, un Français vivant au Japon. L'équipe actuelle d'OmegaT est conforme à son histoire : si Vincent Bidaux, le responsable de la documentation, et moi-même sommes tous deux Français et vivons en France, Alex Buloichik, le développeur principal, est Biélorusse, Aaron Madlon-Kay, le responsable de l'intégration, est Américain et vit à Tokyo non loin de Yu Tang, qui est Japonais et travaille souvent sur les scripts (les « macros » d'OmegaT) avec Kos Ivantsov, qui est Ukrainien. Jean-Christophe vit encore au Japon et fait toujours partie de l'équipe, comme responsable de la localisation, c'est-à-dire de la traduction du logiciel (OmegaT est disponible en 35 langues). Il ne s'agit là que de quelques personnes parmi les près de quarante ayant contribué au programme durant son histoire.

Développer, un travail d'équipe

Plus encore que pour une équipe de développement « classique », l'éloignement géographique, la différence de fuseau horaire et la barrière de la langue (si l'anglais écrit est la langue de

travail, tous les contributeurs ne sont pas à l'aise à l'oral avec cette langue) rendent les réunions, même en ligne, impraticables. Les outils de communication et de collaboration se doivent donc d'être asynchrones : chacun répond ou intervient lorsqu'il est disponible. L'équipe d'OmegaT s'appuie principalement sur deux outils : les listes de diffusion, qui permettent de gérer et d'archiver les conversations par courriel, et les systèmes de gestion de versions.

Développer de nouvelles fonctionnalités nécessite souvent la modification de nombreuses parties du logiciel. Chacun des développeurs dispose donc chez lui d'une « copie de travail » sur laquelle il peut travailler durant des jours, des semaines ou des mois. Cela est d'autant plus vrai dans le cas d'un logiciel libre comme OmegaT, où l'essentiel du développement est une activité bénévole, et donc fragmentée. Durant ce temps, d'autres développeurs font chez eux d'autres modifications, qui touchent parfois les mêmes parties du logiciel. Comment faire alors pour que les modifications d'un développeur n'écrasent pas celles d'un autre ?

La solution adoptée presque unanimement par les équipes de développement, aussi bien commerciales que bénévoles, passe par un logiciel de gestion de versions. Le principe en est assez simple. Un référentiel, appelé parfois « dépôt », contient la version de référence. Lorsqu'un développeur veut effectuer des modifications, il commence par télécharger la version de référence, et obtient ainsi une copie de travail. Chaque modification d'un fichier de la copie de travail est comparée à la version de référence et produit un *diff*, c'est-à-dire une liste de différences entre l'original et la version modifiée. Lorsqu'un développeur est satisfait de son travail, il effectue un *commit*, qui valide l'ensemble de ses modifications, et crée ainsi une nouvelle révision du logiciel dans le référentiel. Qu'arrive-t-il alors aux copies de travail en cours de modification chez les autres développeurs ? Elles peuvent être mises à jour pour incorporer le nouveau développement. Durant ce processus, le *diff* est appliqué à chacun des fichiers. Cela consiste, de façon automatisée, à appliquer les différences comme on pourrait le faire avec un simple éditeur de texte : la ligne 25 a été ajoutée, la ligne 32 a été supprimée, la ligne 43 a été modifiée, etc. Si le *diff* doit être appliqué à un fichier qui a déjà été modifié localement, une fusion est effectuée, pour obtenir une version hybride incorporant les deux modifications. Si la même ligne a été modifiée différemment, le logiciel de gestion de versions ne peut pas décider laquelle conserver : c'est ce que l'on appelle un conflit. Le développeur doit alors le résoudre manuellement en décidant de la modification la plus adéquate.

À l'usage, ce fonctionnement s'avère pratique et facile au quotidien. Une fonction d'historique permet de connaître chacune des modifications effectuées depuis la création du référentiel en 2002. On dispose ainsi d'une « mémoire des modifications ». Des notifications par courriel permettent à chacun d'être tenu automatiquement au courant de ce que réalisent les autres développeurs, palliant ainsi en partie le manque de communication directe.

Les deux systèmes de gestion de versions les plus répandus sont libres et gratuits : Subversion et Git. Le principe d'utilisation est le même, Subversion étant d'une approche plus facile et Git offrant des fonctions plus avancées. Ces deux outils sont proposés par Sourceforge,

l'hébergeur du projet OmegaT. L'équipe utilise les deux systèmes : Subversion pour les versions officielles et Git pour les prototypes.

Clients et serveurs

Contrairement aux outils conçus pour le développement en équipe, les outils de traduction collaborative sont majoritairement propriétaires. Si l'on excepte les solutions utilisables à partir d'un simple navigateur, les outils commerciaux sont constitués d'un client (le logiciel fonctionnant sur l'ordinateur de l'utilisateur) et d'un serveur, c'est-à-dire un logiciel fonctionnant sur un ordinateur distant et chargé de communiquer avec les clients. Le coût de développement d'une telle architecture est relativement important. Le prix des versions serveur des outils de TAO courants est ainsi sans commune mesure avec celui des versions individuelles. *memoQ cloud server* coûte par exemple un minimum de 130 € par mois, tandis que *memoQ server* coûte au minimum 6 060 €. Selon toute vraisemblance, il s'agit là d'un tarif bien inférieur à celui de SDL Trados, ce dernier éditeur se contentant d'indiquer qu'il faut « le contacter ».

Jusqu'en 2012, OmegaT était strictement un outil local. Les possibilités de collaboration étaient réduites : outre le traditionnel envoi de projets par messagerie ou outil de transfert, plusieurs utilisateurs pouvaient faire référence à une même mémoire de traduction sur un réseau local ou au travers d'un logiciel de partage et de synchronisation de fichiers.

La création d'une version serveur d'OmegaT avait été évoquée, mais représentait une charge de travail importante. OmegaT ne possédant pas de base de données, mais uniquement des fichiers texte, cette méthode classique d'accès partagé ne pouvait pas être envisagée. Il fallait développer entièrement un nouveau logiciel serveur et gérer la communication avec les clients. L'installation sur le serveur nécessiterait des compétences et une documentation spécifiques.

Une approche différente

Une grande partie des améliorations d'OmegaT ont été réalisées pour répondre aux besoins des développeurs, eux-mêmes traducteurs. Alex Buloichik avait un besoin important de travail en équipe pour la localisation d'un logiciel. En pensant au fonctionnement d'une équipe de développement, il eut l'idée que ce qui semblait une faiblesse d'OmegaT, l'usage exclusif de fichiers texte plutôt que d'une base de données, pouvait être une force. Les fichiers produits par OmegaT, la mémoire de traduction et les glossaires, sont des fichiers texte. Les outils de gestion de versions sont donc parfaitement à même de les analyser, de trouver les différences et de synchroniser les copies de travail des différents traducteurs. Cette approche particulièrement innovante représente une grande économie de moyens. La partie serveur ne demande aucun développement, puisqu'elle est assurée par les outils standard existants : Subversion et Git. La documentation est par ailleurs abondante sur ces produits. La partie communication et synchronisation avec les clients est assurée par des bibliothèques libres déjà développées en

Java, le langage de développement d'OmegaT. Il ne restait « plus » à Alex qu'à développer la logique de synchronisation, donnant ainsi naissance aux projets en équipe d'OmegaT dans la version 2.6.0, publiée en 2012.

Fonctionnement des projets en équipe

Il faut bien distinguer le rôle du chef de projet et celui des traducteurs.

Pour les traducteurs, utiliser un projet en équipe n'est pas plus difficile que d'utiliser un projet local. Directement depuis OmegaT, le traducteur commence par télécharger le projet en équipe. Cela va créer un projet local habituel (une copie de travail) avec quelques particularités. À chaque ouverture, l'ensemble du projet sera mis à jour (il pourrait y avoir des documents source nouveaux ou modifiés, par exemple). Ensuite, toutes les trois minutes (par défaut, ce délai pouvant être modifié), le projet est synchronisé avec les modifications des autres traducteurs (segments modifiés ou nouvellement traduits et modifications du glossaire). Caractéristique héritée des systèmes de gestion de versions, il est possible de travailler en mode déconnecté (la bascule se fait automatiquement), ce qui est pratique dans le cas d'une connexion internet hasardeuse ou pour les traducteurs nomades. Lorsque la connexion revient, OmegaT se resynchronise.

Un conflit peut survenir lorsque deux traducteurs ont traduit le même segment différemment entre deux connexions. Dans ce cas, une fenêtre s'ouvre et permet de « résoudre le conflit », ce qui consiste simplement à sélectionner la traduction préférée entre celle proposée par le serveur et celle enregistrée localement.

À l'heure actuelle, le travail du chef de projet doit se faire partiellement en dehors d'OmegaT. Il commence par préparer un projet local habituel avec OmegaT : documents source, mémoires de traduction, glossaires, etc. Une fois le projet complet, il faut le créer dans le référentiel. C'est là l'essentiel du travail à réaliser en dehors d'OmegaT. Cela n'a pas non plus à être effectué avec des outils barbares. Sous Windows, avec le logiciel TortoiseSVN, un clic droit sur le projet, suivi de *Commit* (« Livrer », dans la version française) peut suffire à accomplir l'action nécessaire. Il ne reste plus qu'à transmettre l'URL du projet aux traducteurs.

Choisir l'hébergement

Une autre tâche préalable consiste à choisir un hébergement pour le référentiel du ou des projets. Le choix est très large, aussi bien du point de vue technologique que financier.

Les projets en équipe d'OmegaT peuvent être hébergés partout où l'on peut faire fonctionner un serveur Subversion ou Git. Concrètement, cela va d'un ordinateur du réseau local (qui n'est pas forcément un véritable serveur) à un hébergement en nuage en passant par un véritable serveur. Passons rapidement sur une utilisation locale : il existe plusieurs solutions gratuites, dont certaines visuelles et conviviales.

Il existe de nombreuses solutions en nuage, gratuites ou payantes, aussi bien pour Subversion que pour Git. Parmi les solutions gratuites, certaines nécessitent que les fichiers soient visibles par tous, ce qui n'est bien sûr pas envisageable pour des traductions commerciales, d'autres sont limitées en nombre d'utilisateurs (ex. : 2 utilisateurs), et d'autres encore ne seront limitées qu'en taille (ex. : 1 Go). Toutes les solutions en nuage permettent une administration simplifiée : la gestion des projets et la définition des utilisateurs s'effectuent graphiquement depuis un navigateur.

Il reste enfin l'hébergement sur un serveur identifié, géré pour vous de façon spécifique par un prestataire. Pour un professionnel gérant déjà un serveur Apache (la majorité des serveurs web au monde), installer et gérer un serveur Subversion est une tâche relativement aisée, Subversion n'étant qu'un module du serveur Apache. Cet hébergement ne sera certainement pas gratuit, mais pourrait offrir une confidentialité plus rassurante et une assistance plus personnalisée.

Révision synchronisée

L'aspect le plus attractif des projets en équipe n'est pas la traduction simultanée de différentes parties d'un document, mais plutôt les avantages d'une révision « synchronisée ». J'illustrerai ce propos par la description d'un projet réel réalisé par Jean-Christophe Helary en tant que traducteur et moi-même en tant que réviseur.

Il s'agissait d'un projet relativement important (environ 75 000 mots) avec un délai de livraison assez court. Après analyse, la durée de traduction restait acceptable. Le problème se situait plutôt au niveau du temps de révision, puis de la prise en compte des révisions par le traducteur pour enfin obtenir la version finale.

Dans un projet classique, le traducteur effectue d'abord sa traduction, puis la livre au réviseur. Si une même tournure ou un terme technique doivent être corrigés, et qu'ils reviennent 100 fois dans le texte, il faudra effectuer 100 fois la correction, qui devra ensuite être vérifiée 100 fois par le traducteur. Lors d'une révision synchronisée, la tournure ou le terme sont corrigés dès les premières occurrences. La qualité s'améliore ainsi « automatiquement » à mesure de la progression du projet et le réviseur peut se concentrer davantage sur les aspects spécifiques aux nouvelles parties du texte.

La révision étant nettement plus rapide que la traduction, Jean-Christophe a donc commencé par traduire seul pendant deux jours.

Nous avons ensuite mis en place une procédure tirant parti de nos fuseaux horaires respectifs (Jean-Christophe au Japon et moi en France).

Je commençais ma journée ou demi-journée de révision (selon l'avancée de la traduction) en appliquant un filtre pour n'afficher que les segments modifiés le jour précédent (sauf bien

évidemment lors de la première révision). Je travaillais ensuite de façon classique, mais en prenant soin d'ajouter une note explicitant la modification dans tous les segments dans lesquels le changement n'était pas « trivial » (faute de frappe ou d'inattention). Si nécessaire, je remodifiais bien sûr également les segments plus « anciens » pour refléter les nouvelles décisions, en tirant parti des fonctions de recherche. Le cas échéant, je modifiais également le glossaire pour actualiser la terminologie. Je créais également des notes pour suggérer des améliorations ou pour soulever un point particulier.

Mes modifications s'étant effectuées pendant la nuit, Jean-Christophe pouvait commencer dès le matin sa journée de traduction en appliquant un filtre pour n'afficher que les segments comportant des notes modifiées la veille. Après en avoir pris connaissance et décidé des actions à entreprendre, il pouvait effacer la note ou la compléter.

Avec ce système, la mémoire de traduction s'est améliorée à chaque itération. Les quelques répétitions étaient bien sûr corrigées immédiatement, grâce à l'auto-propagation d'OmegaT. Mais l'amélioration se faisait sentir dans toutes les nouvelles traductions. Le volume de modifications apportées a décru constamment tout au long du projet. Le dernier jour, les changements apportés ont été minimes, permettant ainsi de livrer le projet final une demi-journée seulement après la fin de la traduction.

Le bilan de cette méthode de travail a été très largement positif. Au prix d'un effort supplémentaire au début du projet, pour commenter pratiquement chaque modification effectuée (et, pour le traducteur, prendre en compte ces commentaires), l'effort de révision a été bien inférieur à celui nécessaire lors d'un projet classique. La révision, qui aurait nécessité environ une semaine avec une procédure traditionnelle, a pu s'effectuer pratiquement entièrement en temps masqué.

En guise de conclusion

Par leur côté abordable, aussi bien technologiquement que financièrement, les projets en équipe d'OmegaT donnent accès à des méthodes de travail jusque-là réservées aux entreprises d'une certaine taille. Si des agences de traduction aussi bien que des sociétés avec des besoins de traduction utilisent déjà ces projets, ils permettent également à des traducteurs indépendants de travailler en équipe en temps réel. La liberté offerte par un logiciel libre, sans la contrainte de l'achat et de la gestion des licences, autorise la création d'équipes multiformes et éphémères. Enfin, à l'opposé des outils de traduction en nuage, où tout est contrôlé par le donneur d'ordre, l'architecture d'OmegaT permet aux traducteurs de garder la maîtrise de leurs ressources locales, mémoires de traduction et glossaires.

didier@didierbriel.fr

Après une première carrière dans l'informatique, notamment en tant qu'architecte produit, directeur technique et directeur marketing et stratégie au sein d'éditeurs européens de logiciels pour entreprise, ***Didier Briel*** *est devenu traducteur et consultant en 2004, et est membre de la SFT. Il a commencé à contribuer à OmegaT en 2006 et en est devenu responsable du développement en 2007, puis responsable du projet en 2014.*

Il est l'auteur de l'article « Les outils libres du traducteur, un écosystème à apprivoiser » paru dans Traduire *numéro 224.*

Références

Wikipedia, *Apache Subversion*, https://fr.wikipedia.org/wiki/Apache_Subversion, consulté le 7 octobre 2015.

Wikipedia, *Git*, https://fr.wikipedia.org/wiki/Git, consulté le 7 octobre 2015.

Kilgray, 2015, *memoQ – Versions and pricing*, https://www.memoq.com/versions-and-pricing, consulté le 7 octobre 2015.

TortoiseSVN, 2015, *TortoiseSVN – Home*, http://tortoisesvn.net/, consulté le 7 octobre 2015.