※コメント・編集歓迎。

まず「翻訳支援ツール」と「翻訳管理ツール」が混同されていることが多いので、ソコをはっきり分けたい。

1. クライアント側にとって管理作業が容易になるツール →開発支援ツール（翻訳アセット管理ツール）
2. LSPにとって管理作業が容易になるツール→LSP 支援ツール（翻訳管理ツール / 翻訳支援ツールのハイブリッド）
3. 翻訳者/実務担当者にとって実務が容易になるツール→翻訳支援ツール

これらの理解が一緒くたになっていることが多いので、各所で相談事をするときに、話が噛み合わないことがちょいちょい生じる。
開発元「うちも翻訳支援ツール導入したいんだけど」→実際には「1.（翻訳管理ツール）と、2 (翻訳管理ツールと翻訳支援ツールのハイブリッド)の橋渡しをするシステムを構築して、最終的に3 (翻訳支援ツール）とも連携したいんだけど」

3分類を解説

1クライアント側にとって管理作業が容易になるツール

- 開発者が、ローカライズされたアセット（テキストにかぎらず、音声や画像に対応しているものも）を容易にビルドに組み込めることに特化した機能を持つ
    - 各言語のオーディオファイルを指定した命名規則で管理してバージョン管理するとか
- 開発者とLSPがデータをやりとりする上で使用するコンテンツ管理システム
    - テキストが新規追加された場合や修正された場合に対象文字列にフラグが立つ機能などがある
- Localize Direct、XLOC などがこれに当たる。いずれも翻訳者が翻訳できる簡易編集環境を備えているが、全部の作業をそこで完結させるのは無理がある。

    - インポート/エクスポートは主にXLS形式で対応している（CSVではカンマが文中にある可能性が高すぎて使えない、タブ区切りテキストでは改行を含む文字列に対処できないため）。

留意点：LSP側は必ずしもITリテラシーが開発者ほど高いわけではないので、あまり高度な操作が求められると「システム上での操作」がボトルネックやケアレスミス発生の原因になる。アーティストにGitをCUIで使えと言っているくらいの間隔で話すこと。

2：LSPにとって管理作業が容易になるツール

- 発注されたファイル数・文字数・ワード数をすぐに確認できる環境が揃っている
- 複数人の翻訳者に作業文の割り当てをすることができるパッケージ機能などを備えている
- オリジナルファイルを、翻訳者が使うクライアントツール専用形式に変換するのもここ
    - たとえばひとつの巨大なファイルを10人で翻訳する場合であっても、ツール上で分割して、翻訳用ファイルを10個に分割して送付し、それぞれの進捗を個別に管理するなども可能
- 複数人が翻訳に従事していても、翻訳メモリを一元的に管理することも可能（MemoQ Serverなど）
- 複数人が翻訳に従事していても、用語集を一元的に管理することも可能（MemoQ Server、MultiTermなど）

各方面で色々と苦労/工夫している部分が1と2の「つなぎ方」。

- 1は開発者側でのバージョン管理ツール兼翻訳発注プラットフォームとして使うよう専念して
- 2はXLSで翻訳を管理したり、あるいはLSP側で勝手に持っているものを使ったりする（Trados StudioなりMemoQなり）

3：翻訳者/実務担当者にとって実務が容易になるツール

- 下に書いてある通り
- ここは別途スライド化したほうがええかな

1と2のつなぎ方のベストプラクティス

ゲームの規模とローカライズ開始タイミングによってベストプラクティスが大きく異なる点に注意。大規模開発をするのでなければ、要点だけ抑えたシンプルな自社システムで済ませるのがベスト。大規模な場合は血反吐を吐く覚悟が必要。

- テキストがフィックスしていてそれ以上変更がかからない場合
    - xls、csv、google spreadsheetなどの形式でファイルを吐き出し、それを翻訳者/翻訳会社に翻訳してもらう
    - Google Spreadsheet使用する場合、あまりにデータ量が多いと編集処理がおぼつかなくなるので大規模案件には向かない
    - 各言語の翻訳が完了したら、コピペでマスターデータに反映させる
        - Google Spreadsheetならマスターデータに直接入力する形になるので不要だが、そのぶんバージョン/履歴管理は細かく面倒になる可能性もある
- 頻繁に更新が入る場合
    - 言語ごとにファイルを分ける
        - バージョン管理が言語単位でできるようになる
            - 複数言語を同時進行させても、バージョン管理がクッソ容易になる

        - 原文更新時の差分取得はCATツールに任せる
            - MemoQほか、新しめのCATツールであれば差分を簡単に取得できる
                - その時、どのくらいの分量（文字数/単語数）が作業対象か判明するため、請求処理がクリーンになる。地味に大事なポイント
            - 新しい文字列や更新された文字列だけを渡されると前後関係がわかりにくくなるが、CATツールで差分取得をさせると前後の原文および確定済み訳文を見ながら翻訳できる

2と3のつなぎ方のベストプラクティス

文字列単位でのステータス管理（まだ翻訳してないよ/翻訳はしたけど見直しは終わってないよ/原文に更新があったよ）にどうアプローチするかと、更新があった場合のワード数/単語数レポートをどれだけシンプルにできるか（翻訳以外のところで労力を割かせない）。

- Unity Asset Store で売られているLocalization package（やDocsPin や I2 Localization）のようなGoogle Spreadsheet を使うプラグインを導入する
    - この場合差分取得がトリッキーになる（ローカルに保存しておいたバージョンとのDIFF取得などが必要になる）
    - 完全にゲームが完成してもう変更しない！という状態からローカライズを始めるのであれば問題ない
        - あるいはテキストが数千文字/単語程度なら力技でなんとかなる
    - 新規/更新あり/翻訳反映終了 などのフラグ管理をSpreadsheet内で行えばまだできないこともない
- 他にもUnity だと Asset Store に Smart Localization PRO とかもある
    - こちらは csv/xls インポート/エクスポートオプションを備えているのでセキュリティ的にGoogle Driveはちょっと...という場合はいいかも
- XLOCやLocalize Directなど3rd PT製システムを使う
- プレーンテキストの差分取得・比較・ファイルへの挿入を自動化する（継続的に同じ体勢で開発していくならコスパは良いはず、既存のサードパーティ製ソフトウェアを噛ませればなおさらに）
- 業界標準であるファイル形式 XLIFF へのコンバーターを使って3.翻訳支援ツールと連携する
    - 簡潔化できる部分も多
    - XLIFFコンバーターさえあれば1を自由に開発できるという利点ができるものの、2の時点でLSPがCATツールを使うのが必須になってしまう
    - 文脈情報の提供が難しくなる（原文テキストのみ渡すことになるため）
- Microsoft の Localization Studio のように専用のXMLを読み込むことで2と3をつなぐソフトウェアを作る
    - アプローチとしては「同じIDでテキストが変更されたならば、文意は同じはず」という原則のもと、「同一ID間でテキストが相違する場合は変更されたとフラグを立てる」という処理を行う。
    - 1と2のつなぎについてはXMLへのコンバーターと通常のバージョン管理ツールで実現されているため比較的技術的知識を持った人間（ORツールに習熟した人間）でないと扱えない
    - 小規模スタジオが作るには開発コストが非常に高い
    - すさまじい量のカスタムカラムを持つため話者情報から聞き手情報やら何でもかんでも突っ込めるのは素晴らしかった（その分可読性が低かった）

なんでXLIFF（あるいはCATツール専用形式ファイル）を使うのがいいのか

たとえばXMLを渡して中に入っている文字列を上書き翻訳してもらうとしよう

→ タグぶっ壊す可能性がある

→ 一回翻訳し終えてしまったら、原文がどんなだったか確認するのが手間（Diffツール使うとかになる）

→ そもそもXMLを人間が開いて編集するとか文明の敗北

上記のような事が、XLIFFではすべて解決される。なぜなら：

- XLIFFでは翻訳対象のテキストだけが編集可能になる
- XLIFFは「バイリンガルドキュメント」と呼ばれるもので、常時「原文」と「訳文」をペアにして格納している
- XLIFF自体はXMLであるが、専用クライアントで開いて編集するので文明は敗北しない。

トレードオフは何よ？

文脈情報が提供しにくくなる。

CATツールがゲーム翻訳でどのように活用できるか具体的に

翻訳メモリ

- 作中で数回しか出てこないキーフレーズが他でどう訳翻されていたのかボタンひとつでチェックできない
- 編集環境
- 上書き翻訳だと原文の訳し抜けが発生しうる

特定の内容（原語・訳語）を含む行だけをフィルタできない

- 進捗途中であとで見直す・もう大丈夫・まだ手を付けていないのステータタグや\\nなどを壊さないようにカプセル化できない
- 翻訳しちゃダメなところをロックできない
- クライアントによって作業環境が異なる（フォーマットとかが微妙に違ったりする）が、CATツールならば編集環境が完全に同一
- MemoQならリアルタイムプレビューしながら翻訳できるので、XLSとかで話者情報とか文脈情報とか書いてあっても見ながら編集環境を維持できる
- 意外と便利な自動QA機能
- XMLとかマークアップ言語のファイルもパーサーさえあれば翻訳が必要なところだけ抜けだして、他の部分を壊さずに済む
- 差分が取りにくい

用語集

- 用語集を自動でLookupしてくれない
- プラットフォーム用語とか覚えてなくても自動で引っ張ってくれる
    - 逆に原文のミスにも気づける（あれコレ用語なのにLook Upされてない→原文の表記が間違ってた）

進捗管理

- レポート形式で残しておけない（請求処理のとき先方が出してくる数字との突き合わせなど）
- 残りワード数管理がすぐできない

- 正確な更新数(文字数、ワード数、個数)を把握出来ない若しくは人間のカウントに依存せざるを得ないが故に変更から派生する作業の見積もりが立てにくい(上のレポート形式？)
- ヒストリーが追いにくい
- 同一言語またはクライアントとのチームワークが成立しない

メモ

このスライド誰向けにつくろうか。
「開発会社だけど自作ツールで対処しようと思ってる」人と「まだCAT使ってない翻訳者」。
より多くの人の幸せのためには前者に絞ったほうがいい気がする。恵まれない環境で作業する翻訳者を増やさないために。その後、まだCAT使ってない人向けに書くか。

前提条件として:「翻訳なんちゃらツール」は3分類できる

- 1:クライアント側にとって管理作業が容易になる
    - 翻訳後のデプロイが簡単とか
- 2:LSPにとって管理作業が用意になる
    - 複数人に仕事を割り振りしたり進捗管理したり
- 3:翻訳者/実務担当者にとって実務が用意になる
    - より生産的に仕事ができる

現在はこの3つがいっしょくたになっている製品が多い/スイート・サーバー製品という名前で、合体していることが多い。
そして自社で作るときは何を主眼に置くかを絞るのが大事。全部を網羅したComprehensiveな物を作ろうとするとおおごとになりすぎる。それならば、どこかに注力してしまって、あとは中間ファイルをインポート/エクスポートするしくみを取るだけにした方がいい気がする
SQENさんのMoomleはそうしてたはず
