多言語対応のアプリケーション開発基盤
―言語別データディクショナリを活用した開発手法の紹介―

日本のITサービスやアプリケーションは、機能面やきめ細かな使い勝手の点で完成度が高いことは誇れるが、多言語対応に関する配慮が不足しており、世界になかなか通用しないのが現状である。本稿では、このような状況を変えるために、開発効率を犠牲にすることなく多言語化を行うシステム開発手法を紹介する。

野村総合研究所 基盤サービス本部
産業基盤サービス部 主席システムアナリスト
大原（おおはら） 聡一（そういち）
専門はアプリケーション開発、アプリケーション基盤構築

多言語対応は日本のITサービスの大きな課題

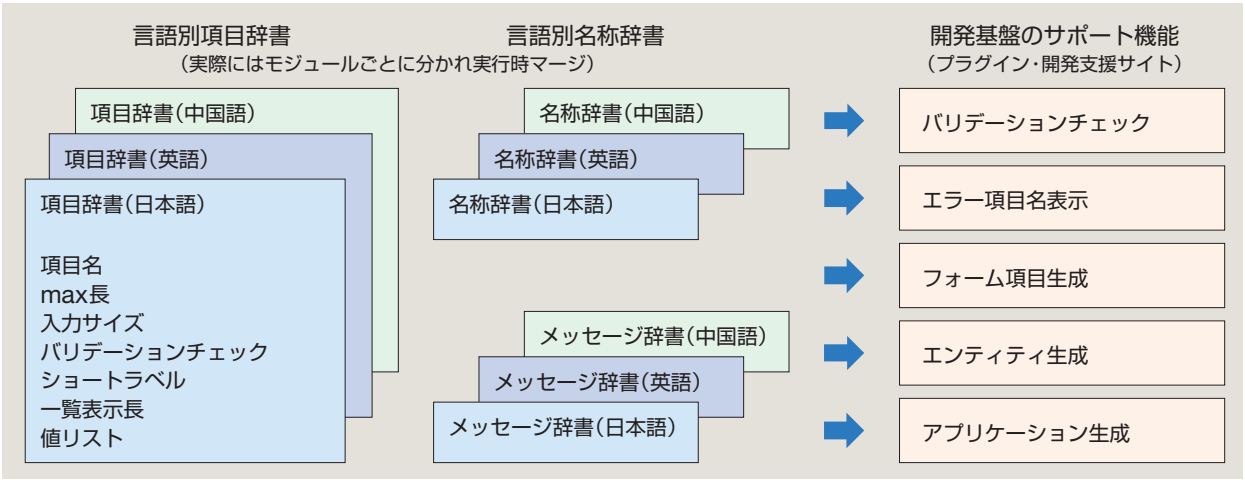
日本から世界をリードするようなITサービスがなかなか生まれない。国内で高いシェアを占めたサービスがやがてグローバルなサービスに取って代わられることも多い。しかし、それは日本のソフトウェア技術やサービスの内容が劣っているせいではないと思う。いろいろな意味で、言語の壁が大きいのではないだろうか。個々のソフトウェア開発の創造性が海外に引けを取るものでないとしても、それが日本語という枠にとどまっている限りグローバルな創造の連鎖につながらないのである。

最近、ドイツSAP社のERP（統合基幹業務システム）周辺のアプリケーションを開発するフレームワークを使う機会があった。多言語対応というのはチュートリアル（操作を覚えるための教材）の最初の「Hello World」から出てくるもので、サンプルのコードは特定の言語に依存していない。当然といえば当然だが、最初から英語・ドイツ語・フランス語などに対応しておかないと欧州市場では戦えない。多言語対応は彼らにとって当たり前のことなのだろう。

一方、日本の場合、既存の基幹系業務システムならともかく、海外展開を念頭に置いた新規事業のIT基盤ですら、将来的な多言語化への配慮が足りないのではないだろうか。ITを活用した新規事業は、ますますグローバルな展開スピードが重要になってきている。日本の中で完成度を高めているうちに、あるいは海外の実情に合わせた現地システムを構築している間に、海外の競合は完成度を犠牲にしてでも一斉に世界に展開し、圧倒的なシェアを押さえてしまう。そして数の力を背景に資本市場から大きな資金を調達し、あっという間にビジネスモデルとそれを支えるシステムを洗練させてしまう。

日本で一時的に大きなシェアを占めても、そういう彼らに勝ち続けるのは難しい。これからは、たとえ国内のユーザーを対象にしたシステムであっても、最初から多言語対応が可能なシステムをつくっていく必要があるのではないだろうか。

図1　多言語対応アプリケーション開発基盤の概要



多言語対応システムへの挑戦

アプリケーション開発を専門とする筆者は、この状況を何とかして打破しなければならないとずっと思ってきた。普通に生産性を高めながら開発を行った結果、自然に多言語対応のシステムになり、すぐにグローバルな横展開ができるといったことにはならないものだろうかということである。そして、いくつかのWebアプリケーション基盤開発の機会にこれにトライしてみた。

(1) データディクショナリによる多言語対応

業務系のシステムの場合、日本語と外国語では単に項目の名称が違うだけでなく、バリデーションチェック（データが正しく記述されているかを検証すること）の内容、項目長（入力桁数）、選択可能な値リストも変わってくる。そのため、アプリケーションのレベルで多言語化を図ると、単一言語で開発する時よりも工数がかかり、運用・保守も煩雑になってしまう。それは、言語によってバリデーションチェックの内容や項目長が異なると、それぞれの言語でソースコードを変える必要があるからである。

これを避けるには、項目長や値リストなど、項目の属性をソースコードから独立させるしかない。これはすなわちデータディクショナリの活用になるが、開発効率を落とさないようにするには、開発基盤のレベルでのサポートが必要となると考えた。

データディクショナリとは、システムで扱うデータの名称、意味、場所、タイプ（文字列、実数、整数など）などを列挙したファイルで、システム間でデータの一貫性や整合性を保つために使われるものである。このデータディクショナリを言語別に整備し、言語に応じて切り替えることにより、ソースコードを改変することなく多言語に対応したアプリケーションの開発が可能になる（図1参照）。

開発フレームワークのレベルで言語別のデータディクショナリを組み込む主な効果は次のようなものである。

①各言語に応じたバリデーションや画面上の項目編集がデータディクショナリから生成可能になり、コーディングの量が大きく削減可能になる。

図2　アプリケーションの自動生成の例

辞書をベースに、ドラッグ＆ドロップのエンティティ（アプリケーションが対象とするデータのまとまり）生成ツールや、より精度の高いひな形生成ツールを提供可能。初期開発の生産性向上が見込める

②Webアプリケーションのひな形を生成する機能にデータディクショナリを組み込むことで、登録・更新画面などのアプリケーションの基本機能を多言語対応可能な状態で自動生成することができる（図2参照）。

③アプリケーションの改変に当たっては、データディクショナリを直すだけで各言語のシステムに反映させることができるので保守効率が高まる。

④データディクショナリのオーバーライド機能（オブジェクト指向のプログラミングで、あるクラスが別のクラスを継承したときに、そのクラスが持つメソッドを上書きして再定義する機能）により、ソースコードを改変せずに汎用サービスの項目のカスタマイズができる。

⑤データディクショナリの蓄積でアプリケーション自動生成の精度が高まり、生産性がより向上する。

(2) メンテナンス性の高いデータディクショナリを実現

データディクショナリを装備したシステムは以前からあったが、そこにはデータディクショナリの管理・保守上の問題があった。データディクショナリをデータベースで管理すると、ソースコードのバージョン管理との整合性に問題が生じるのである。すなわち、ソースコードを複製したブランチを作成する場合、ブランチを切り替えてもデータディクショナリが切り替わらないという問題である。この問題に対処するために、あえてデータディクショナリをフラットファイル（単純な形式のテキストファイル）で管理し、ソースコードと同じレベルでバージョン管理が行えるようにした。

以上のコンセプトを取り込んだ開発基盤は、最終的に野村総合研究所（NRI）の製品「ZOOK+」に結実し、既にいくつかのプロジェクトで活用されている。データディクショナリによる多言語対応と、バリデーションやViewとの連動をフレームワークレベルで同時に実現する点は日本で特許を取得し、米国、英国で特許申請中である。

このコンセプトは、特定の開発言語（JavaやPHPなど）に依存したものではないので、

適用プロジェクトの開発環境に合わせて再構成しながら、より運用性を高める工夫を続けてきた。例えばJavaベースのオープンソースアプリケーションフレームワーク「Spring」向けには、データディクショナリのメンテナンスを容易にする以下の機能を持つ独自のメンテナンスツールを用意した。

①プロジェクト、サイト、モジュールといった複数階層のデータディクショナリの統合管理

②複数階層のデータディクショナリにまたがる項目IDや値情報の検索・更新

③言語間のデータディクショナリのマージ（先行開発の言語から他言語への展開を効率化。図3参照）

企業内SaaSの構築から

言語別のデータディクショナリを用意しておけば、グローバル展開の際には、データディクショナリを置き換えるだけで言語間の項目の違いに対応できる。また、この機能はSaaS（Software as a Service。ネットワーク経由で利用するソフトウェアまたはその仕組み）システム構築に応用できる。サービスを利用するプロジェクト側のデータディクショナリで、サービスのデータディクショナリをオーバーライドして改変することが可能なため、より柔軟なサービス化が可能になるのだ。汎用サービスは利用プロジェクトによって一部機能の差し替えが必要になることもあるが、データディクショナリと同時にconfig情報（設定ファイル）をオーバーライドする仕組みを装備することで対応できる。

図3　データディクショナリのメンテナンス

画面機能の汎用化は難しいものだが、企業内のSaaSをつくってみれば合理的な方法であることが分かるだろう。特に、同じアプリケーションで異なる商品を扱う場合などに効果が大きいと思われる。商品カテゴリによる違いをデータディクショナリによって吸収するのである。さらに、将来的に期待される人工知能による自動保守にも、データディクショナリを装備したシステムは相性がいいと思われる。

筆者は、データディクショナリの活用が今後の情報システム開発でキーテクノロジーになるのではないかと考えている。データディクショナリを装備したアプリケーション開発基盤をベースに、多言語に対応した企業内SaaSサービスの構築から始めてみることを提案したい。■