Linux SEのお仕事

言語と人と運用と
～プロジェクトに立ちはだかる壁

文：株式会社ビーブレイクシステムズ　鹿取裕樹

Linuxを始めとして、Apache、Tomcat、PostgreSQL、Perlなど、多くのオープンソースソフトウェアが業務システムにも浸透している。読者の皆さんの中にも、「お仕事」としてオープンソースソフトウェアに取り組んだ経験のある方がいるだろう。

今回も筆者が経験したプロジェクトの中で、オープンソースを活用した業務システム構築の実例を紹介する。

このプロジェクトでは、オープンソースならではの問題、オープンソースとは関係ない問題ともに発生した。本稿では、これらの問題の内容と対応策を紹介する。読者の皆さんの参考になれば幸いである。

オープンソース採用の経緯

SE no Oshigoto

今回紹介するシステム（以下、本システム）の顧客は、従業員200名未満のサービス業である。独立したシステム部門を社内には持っておらず、IT関連の業務は総務部門が兼務している。よって、技術についてのスキルは高くないため、システムの運用は自社では行っていなかった。

本システムについても運用を筆者の所属する会社が請け負うことになった。そのため必要な機能が適切に実現されれば、使用する技術には特別な要件はなかった。予算、筆者の会社がノウハウを有していたこと、安定稼動が可能なことなどを考慮し、オープンソースソフトウェアの利用を提案し、採用されることになった。

システム概要

SE no Oshigoto

簡単に今回のシステム「経営管理システム」の概要を説明しておこう。まず、システムの構成図は**図1**のようになる。

そして、システム導入の目的は以下のようなものだ。

- **従業員がサービスを提供した実時間と各種マスタを元に請求書を自動生成・発行する**
- **計画値と実績値とを元に各種分析レポートを出力する**
- **顧客情報を管理する**
- **承認フローをシステム化する**

データの流れと機能との関連は**図2**のようになる。具体的な業務の流れを、実績データと計画データに分けて少し詳しく説明しよう。

実績データの流れは、以下のようになる。

- **従業員は毎日の作業内容／作業時間／顧客を時間入力機能を用いて登録する。また、顧客名／請求方法／請求時期などが登録された顧客マスタを別途登録しておく**
- **入力された時間データを、関連する顧客ごとに集計し、顧客マスタに登録されたタイミングで請求書を生成し、出力する**
- **時間入力やマスタ登録には承認のワークフロー機能がある**

これに対して計画データの流れは、以下のようになる。

- **プロジェクトマネージャが担当者の稼働時間などプロジェクトごとの計**

画を入力する

- **計画には、変更をしない当初計画と状況の変化に合わせて変更していく現在計画の2つのバージョンを持たせる**
- **これらの情報により、次のような分析を行う。**
  - **各担当者の稼動計画**
  - **プロジェクトの収益性分析**
  - **当初計画と現在計画の比較**
  - **顧客ごとの収益性分析**

使用するソフトウェアは、**図1**のシステム構成図にあるとおりだ。お馴染みのものが並ぶ中で、「POI」というソフトウェアはご存じない方もいるかもしれないので、簡単に言及しておきたい。

POIとは、Apache Jakartaプロジェクトにおいて開発が進められているオープンソフトのライブラリで（http://jakarta.apache.org/poi/）、ExcelやWordといったマイクロソフトのドキュメントをJavaで扱うためのAPIを提供する。

今回の案件では、このPOIライブラリを帳票作成機能で利用している。

■プロジェクトの流れ

開発期間は約4カ月の予定で、以下の流れで行われた。

要件定義→実装→結合テスト→ユーザーテスト→データ移行→カットオーバー

結合テストまでを3カ月で終え、残りの1カ月でユーザーテストとデータ移行を行うという計画であった。

しかし、（詳しくはこのあと紹介す

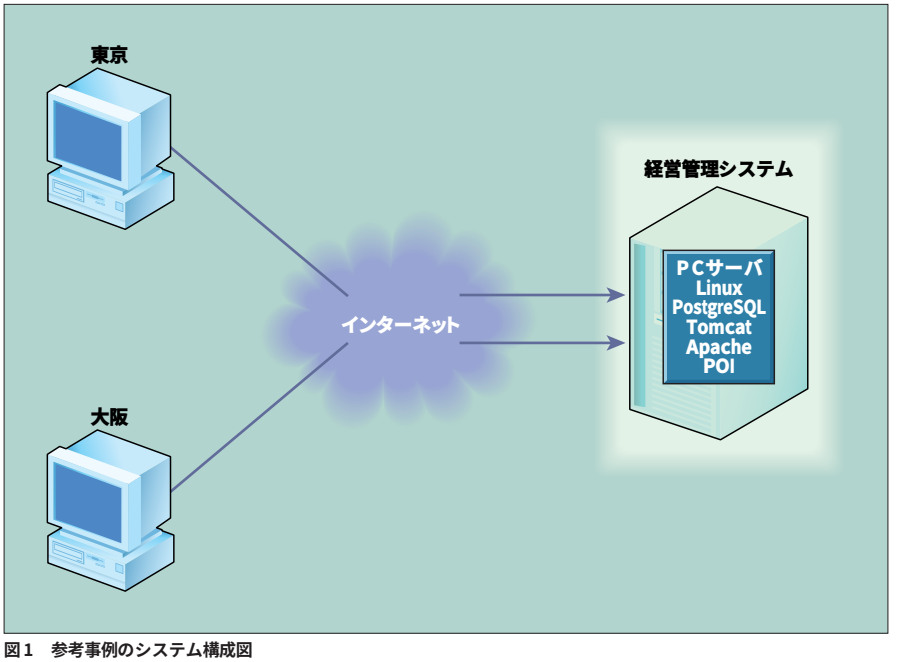

図1　参考事例のシステム構成図

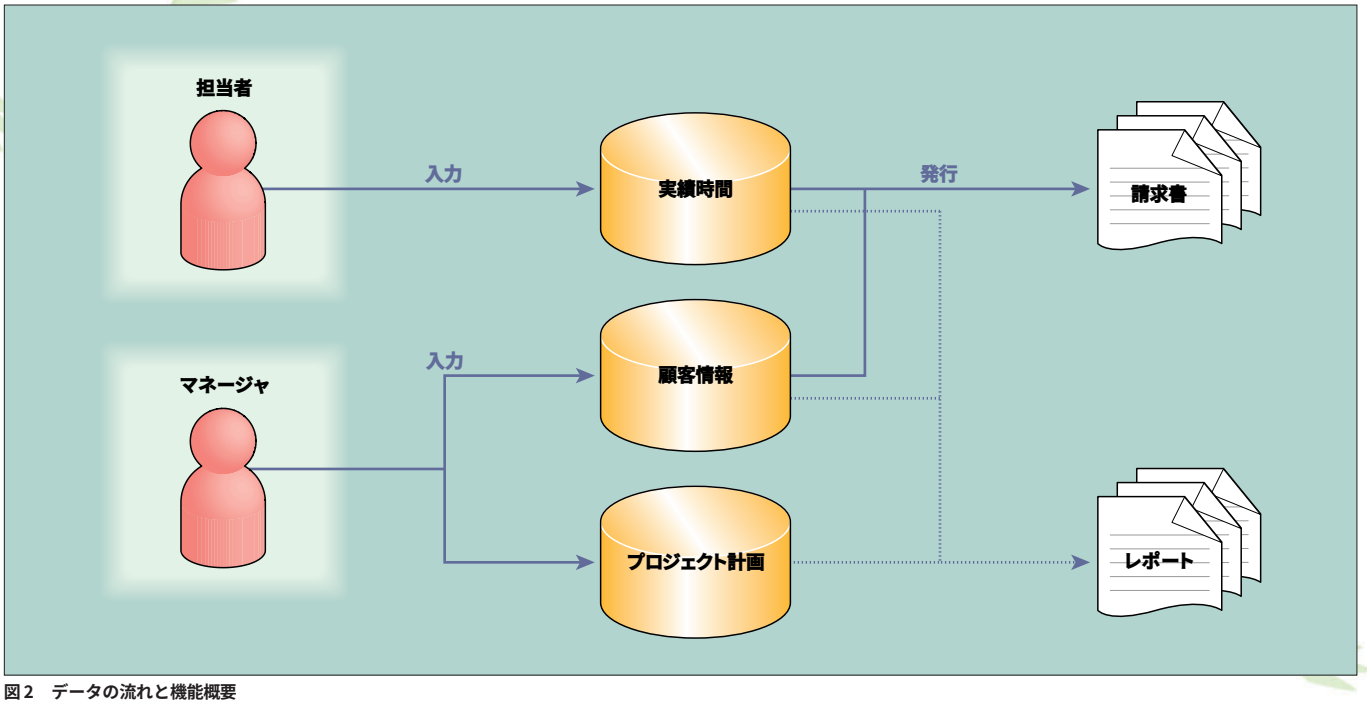

図2　データの流れと機能概要

るが）顧客の社内状況をよく把握できていなかったためにユーザーテストやデータ移行が遅れ、開発終了からカットオーバーまで予想以上の時間がかかることになった。

言語の壁

SE no Oshigoto

開発開始から2週間ほど経過し、POIのライブラリを使用したExcelの帳票出力機能の開発を始めた頃、あるエラーに悩まされることになった。

それは、POIによって生成したExcelファイルを開くと表示される「印刷できるものがありません」というエラーである。

この取りつくしまもないエラーが起きると、Excelは起動するもののファイルが開かれない。生成されたファイルのサイズを見ると空ファイルということではなさそうだ。さらに、生成したExcelファイルのシートの数が1つであればエラーが発生せず、2つ以上になった場合にエラーが発生することがわかった。

「ネットで検索してみて」筆者はインターネットで解決に必要な情報を検索できるだろうと思い、メンバーに指示した。しかし返ってきた答えは、「見つかりませんでした」であった。

オープンソースソフトウェアのメリットのひとつとして、公開されている情報の多さがある。インターネット上にコミュニティがあり、メーリングリストを通してQ＆Aなどが行われているため、それらから多くの情報を得ることができる。

だが、ネット上の豊富な情報を享受するには条件がある。それは「英語を読める」ことである。

メジャーなソフトウェアや、日本国内で開発されたソフトウェアであれば、多くの日本語のドキュメントがあるだろう。しかし、今回使用したPOIはそれには該当せず、ネット上で得られる日本語の情報が限られていた。

そこで筆者は再度、日本語以外のドキュメントも対象に検索することを指示した。すると、さっそくメンバーから質問を受けた。「“印刷できるものがありません”って英語で何と言うのですか？」。それくらい、と思われるかもしれないが、そのメンバーは英語がまったく駄目だったのである。

筆者も加わり調べた結果、生成したExcelのシートが選択されていない状態の場合、そのファイルを開こうとすると問題のエラーが発生するという情報を見つけることができた。

インターネットが普及し、膨大な情報を簡単に手に入れられるようになった。しかし、日本語の情報は全体のほんの一部にすぎないのである。英語の能力があれば得られる情報量はぐんと増加する。以前の回にも書いたが、これからはソフトウェア技術者も英語を身につけていくべきだと痛感したしだいである。

進まないユーザー作業

SE no Oshigoto

POIにより完全ではないにしろ、APIを使ってJavaからExcelファイルを生成できるというのは、開発工数を減らすのに大きく役立った。

プロジェクト開始から3カ月ほどで開発／単体テスト／結合テストを終え、ユーザーテストの段階となった。カットオーバーまで残り1カ月となった頃である。

システムの説明を行い、システムをユーザーに開放し、ユーザーテストをしてもらうように依頼した。テスト開始日は顧客の事務所にてテストのサポートを行ったが、それ以降は自社に待機し電話にてサポートすることにした。

「テストの進捗はいかがでしょうか？」。テスト開始から1週間がたった

画面1　POIの日本語ページ
このページはPOIプロジェクトのトップページを翻訳したもので、Jakartaプロジェクトの一環として提供されている。ただし、POIの細かな技術的なドキュメントやニュースなどについては、日本語の情報は用意されていない。

頃、ユーザーに電話をかけてみた。テスト開始以降、ユーザーからの問い合わせがなく不安に思ったためである。「すみません。業務が忙しくてできませんでした。今日からやります」。うすうす予想していた答えであった。

このとき、「確かに月末で業務が忙しくてまだテストどころではないのだろうな。まだスケジュールに余裕もあるし……」と思い、手を打つこともなく「わかりました。では、よろしくお願いします。わからないところがあればご連絡ください」と答えてしまった。

これが失敗であった

さらに1週間たって問い合わせてみたところ、全20機能のうちテスト済みは2機能のみという状態だった。しかもそのうち1機能はテスト開始日に顧客先で筆者らがサポートしながら行ったものであった。つまり、それ以降、1つの機能しかテストが行われていなかったのである。

前述したとおり、この顧客は大企業ではなく、システム専門の担当者がいるわけではない。システム担当を兼任している総務部の方は経理も行っており、月末になると月次処理で忙しくなる。こういったケースでは、顧客の業務や忙しい時期はいつかなどを考慮してユーザー作業を依頼しなくてはならないであろう。

また、顧客はシステムのテストを初めて行うということも少なくない。そのため、開発側はどのような手順とスケジュールでテストを進めるかというガイドを示し、顧客先につめてサポートするといったこともしていくべきだと実感した。

運用の視点が欠けた設計

SE no Oshigoto

ようやくユーザーテストが進み出し、半分ほどの機能のテストが終わった頃、顧客から打ち合わせをしたいとの申し出を受けた。ある機能が使いづらく運用できそうにないため、変更したいとのことだった。

最初にこの話を聞いたとき、正直なところ「要件定義のときに確認して、承認したではないか！」と思ってしまった。

しかし、システムは使ってもらうために作るものである。プロとしての我々技術者は顧客が運用できるように設計しなければならない。こちらにも非があるのは認めざるを得ない。

ただ、変更には当然コストがかかり、すべてに対応することはできない。そのため、できるだけ少ないコストで使いやすさを大きく改善できるような変更を考えなければならない。

幸い、内部設計はオブジェクト指向による設計で変更に対応できる形にしていたため、画面プログラムと画面制御プログラムの変更にとどまった。

筆者らは、どうしても機能を納期内に実現することに注力してしまい、システム運用への考慮が不足していた。システムは開発すれば必ず運用されていくものであり、運用についてはいくら考慮しても考慮しすぎということはないであろう。

特にマスタメンテナンスや業務フローは、ある程度以上のシステムでは十分検討しなくてはならない。

顧客は業務のことはよく知っていても、システム構築についてほとんど知らないことが多い。そのため、システムにさまざまな機能を盛り込もうとする。それをそのまま受け入れていくとどんどん複雑になり、最終的に入力できる項目や機能は多いが結局使われない、ということになりかねない。

要件定義の際に顧客から求められるままに作っていくのではなく、顧客の業務の本質を考え、それをできるだけシンプルに実現するシステムを設計することが必要であろう。

Column

翻訳サイトは使える？

SEにも英語力が求められるのは、本文で触れられているとおり。特に、海外発のオープンソースソフトウェアの場合は日本語の情報が限られてくる。英語をマスターするのがベストなのだろうが、現実的に考えて、忙しい業務の隙間を縫って習得するのは難しい。そこで解決策として浮かんでくるのが、翻訳ソフトやネット上の翻訳サービスだ。

専門用語や独特の言い回しの多いコンピュータ技術の文章は、こうした機械翻訳の苦手分野とされている。ただ、技術文書の英語の読解で問題になるのは、単語の訳よりも構文の理解だろう。翻訳ソフト／サービスで文書の概略が掴めれば、原文と照らし合わせて内容を理解できるはず。英語の苦手な人は、利用する価値はアリだろう（編集部）。

POIの英語ページを翻訳すると……