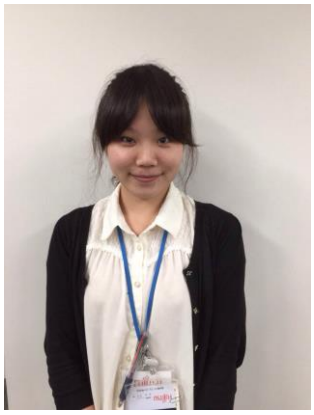
コーディング規約の運用を目指して

富士通エフ・アイ・ピー株式会社

■ 執筆者Ｐｒｏｆｉｌｅ ■



2014 年　富士通エフ・アイ・ピー（株）入社
EDI サービス事業部 EDI 基盤システム部所属
TradeFront システム担当

宗木　唯

■ 論文要旨 ■

筆者は配属と同時に EDI サービス事業部の TradeFront というサービスの運用業務に携わるチームに入った。TradeFront とは、小売業と卸業の間に入って、お客様間のデータのやりとりを支えるサービスである。
この一年間は Unix 系のシェルスクリプトの改修を担当する機会が多く、その中でも特に高負荷となったプロセスを検知する監視スクリプトの作成での経験が、本稿のテーマとなった改善活動を始めるきっかけになった。
シェルスクリプトでのトラブルを繰り返さない為に、EDI 事業部として「コーディング規約」を作成して、規約に沿って作成されていないシェルスクリプトがサービスに導入されないような体制をつくりたいと考えた。
本稿では改善活動の詳細とそこで工夫した点・苦労した点を述べる。

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　当社の概要

当社はお客様の大切なデータをしっかりと守り運用し、国内最高水準の安全性を誇るデータセンターを北海道から九州まで全国 16 か所に展開している。これらのデータセンター機能を基盤に LCM サービスとして、「アウトソーシング」、「クラウド」、「ソリューション」の 3 つのサービスを提供、システムの企画から設計、開発、保守、運用まで、ライフサイクル全般を支援（LCM サービス）し、お客様に安心・安全で、高品質かつ高コストパフォーマンスの ICT サービスを提供している。

筆者は 2014 年 4 月に入社し、TradeFront サービスを提供している EDI サービス事業部 EDI 基盤システム部へ配属となり、TradeFront の運用業務に携わるチームに入った。この一年間は Unix 系のシェルスクリプトの改修を担当する機会が多く、その中でも特に高負荷となったプロセスを検知する監視スクリプトの作成での経験が、本稿のテーマとなった改善活動を始めるきっかけになった。本稿では改善活動の詳細とそこで学んだこと、苦労した点等を述べる。

１．２　TradeFront の特徴

TradeFront とは、小売業と卸業の間に入って、以下のようにお客様間のデータのやりとりを支えるサービスである。当社は小売業様 250 社、お取引先様 30,000 社への EDI サービスを提供している。つまり業界のトップランナーとして EDI を提供しているといえる。

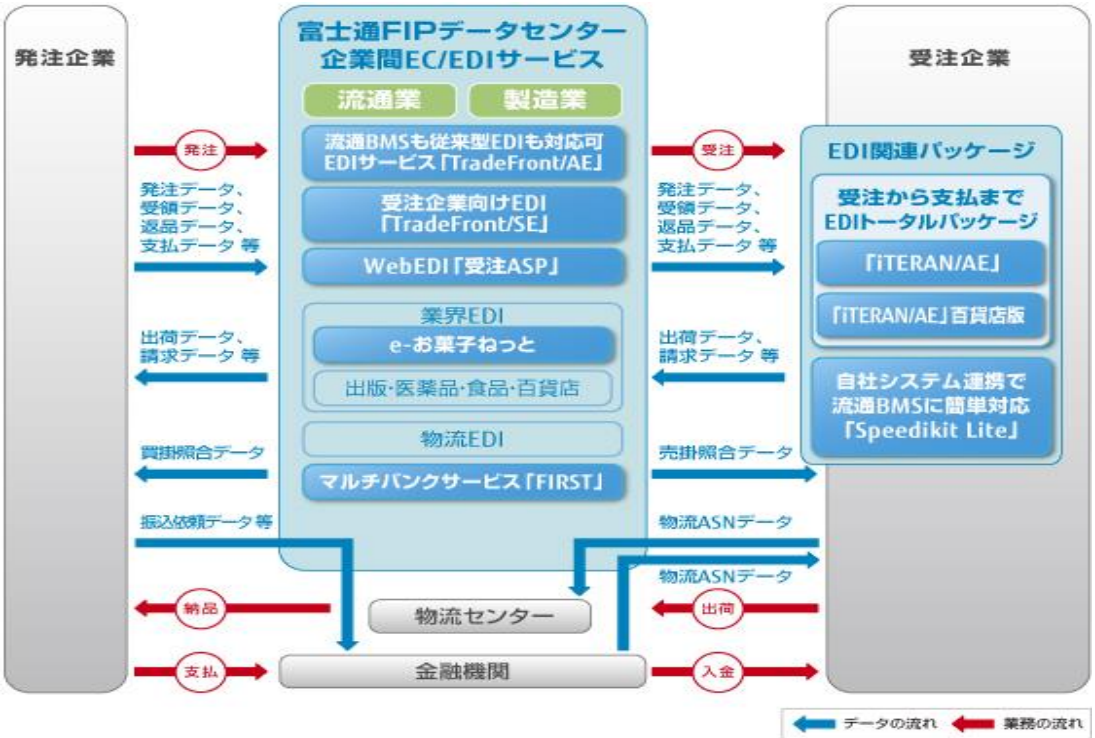


図 1. TradeFront サービスメニュー

- 受発注・データ交換業務をサポート
- クライアントパッケージ（iTERAN/AE）や Web-EDI 等、様々な接続環境を提供
- ディザスターリカバリーに対応し、万が一の事態からもデータを保護。
- 小売業様 260 社、お取引先様 30,000 社の豊富な実績

TradeFront のデータの流れをわかりやすく記載した絵を以下に示す。筆者は Tradefront サービスの運用・保守作業、サーバ構築関連作業を担当している。

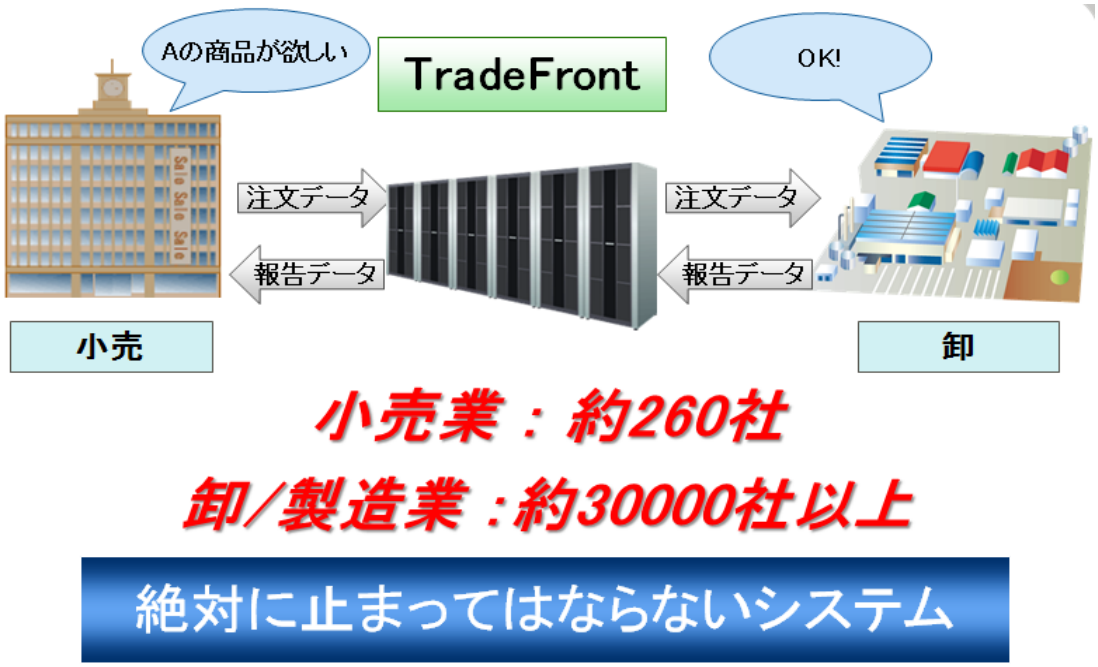


図 2. TradeFront データの流れ簡略図

2. シェルスクリプト導入で起きたトラブル

筆者は配属時からさまざまな業務に携わってきたが、特にシェルスクリプト（UNIX 系 OS のプログラム）の新規作成や改修を担当する機会が多かった。その経験のうち、後述するシェルスクリプトの作成において起きたトラブルが、本稿のテーマとなった改善活動を始めるきっかけとなった。

現在運用している CPU 監視ツールでは、全体の使用率のみで高負荷を判断しているため、単一のコアを占有し続けるようなプロセスを検知することができない。CPU を占有するプロセスを放置しておくとレスポンスの低下につながり、ミドルウェアの異常終了などを引き起こすリスクがあるため、CPU を占有する高負荷プロセスを検知する監視のスクリプトを新規で作成をした。

CPU 監視スクリプトは先輩に指導を受けて何とか作成できた。しかし、全環境に導入する際、システムテスト中に一部の環境で CPU 監視が動作しないという事態が発生した。調査には多くの工数を要し、周りの方々にフォローしていただくことになってしまった。

原因としては、これまでの RedHatOS バージョン 6.4 以前では、クーロン（ジョブ（スクリプト）を自動実行するための常駐プロセス）から起動されるプロセスの環境変数 LANG の初期値設定は「設定なし」だった。しかしバージョンが上がったことで、今まで読み込まなかった LANG に「英語表記の Locale（ユーザーがどの言語を使うか定義する環境変数）」を読み込むようになり、日本語を使った処理を実装しているスクリプトが機能しなくなっていた。つまり、その環境変数はスクリプト内で静的に日本語として設定しておかなければいけなかった。

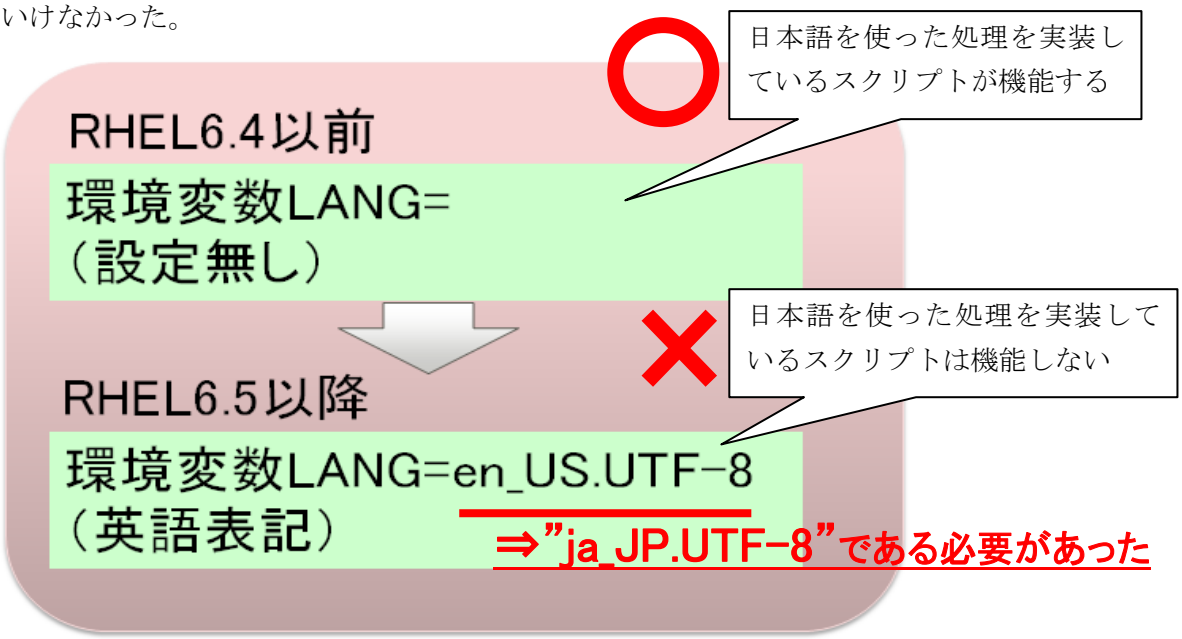


図 3. RHEL OS バージョン差によるトラブル原因

３．課題と改善策

３．１　現状と課題

監視のシェルスクリプトは既存のノウハウ集を元に作成していたが、LinuxOS バージョンの差異による環境変数（Locale）が固定できていないというポイントが記載されていなかった為、トラブルは起きてしまった。

このようなミスを未然に防ぐ方法はないか考え、ノウハウ集に環境変数について今回のポイントを追記する必要があると感じた筆者は、トレーナーに提案を行った。するとトレーナーから以下の指摘を頂いた。

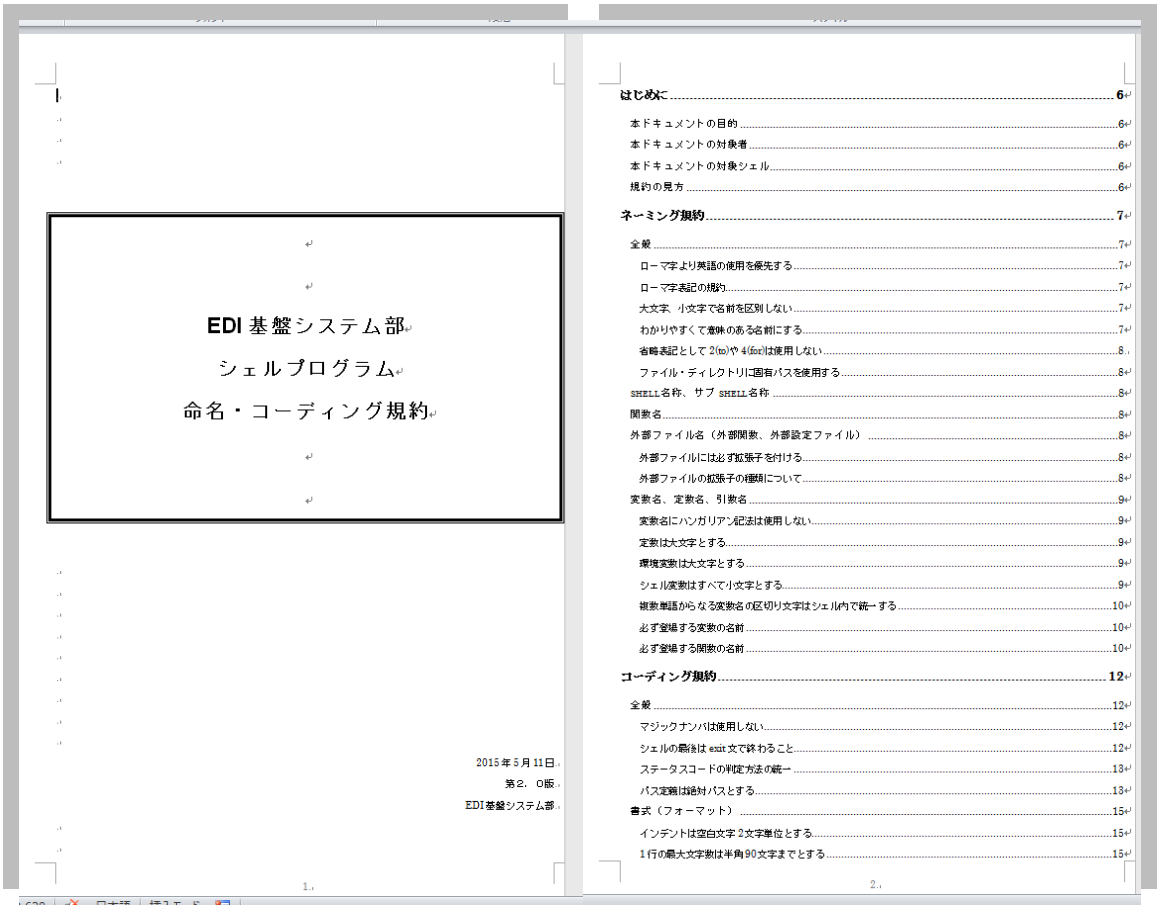
1. ノウハウ資料は最新化ができていないから、他にも同様のトラブルを繰り返す可能性がある
2. ノウハウ集に沿ったスクリプト作成が行われているかをチェックする体制が確立できていない、個人によっては資料を適用しない人が出てくる可能性がある

それを受けて筆者は、「コーディング規約」を作成して、規約に沿って作成されていないシェルスクリプトがサービスに導入されないような体制をつくりたい！という想いを強く持つようになり改善活動を始めた。

３．２　対応履歴と成果物

(1) 有識者に意見公募

まずは、現場の有識者にシェルスクリプト作成について守るべき規則や注意事項等の意見を公募することから始めた。今まで存在したが運用までいかなかったノウハウ集に収集した意見を追加、修正をし、コーディング規約と運用ルールの作成に取り組んだ。集めた意見をすべて理解するところから始めなければいかず、また疑問点がある項目に関しては意見してくださった方と筆者の意見とのすり合わせを行った。情報収集の為、自ら積極的にコミュニケーションを取りに行くようになったことと、コミュニケーションをスムーズに取るための知識や意見の準備を行っていく習慣が身についた。これまではチームの垣根を越えて仕事をする機会が少なかった為、今後に繋がる良い経験となった。今回作成した規約を図４に示す。


EDI 基盤システム部

シェルプログラム

命名・コーディング規約

2015年5月11日

第2．0版

EDI基盤システム部

はじめに 6

本ドキュメントの目的 6
本ドキュメントの対象者 6
本ドキュメントの対象シェル 6
規約の見方 6

ネーミング規約 7

全般 7
ローマ字より英語の使用を優先する 7
ローマ字表記の規約 7
大文字、小文字で名前を区別しない 7
わかりやすくて意味のある名前にする 7
省略表記として 2(to)や 4(for)は使用しない 8
ファイル・ディレクトリに固有パスを使用する 8
SHELL名称、サブ SHELL名称 8
関数名 8
外部ファイル名（外部関数、外部設定ファイル） 8
外部ファイルには必ず拡張子を付ける 8
外部ファイルの拡張子の種類について 8
変数名、定数名、引数名 9
変数名にハンガリアン記法は使用しない 9
定数は大文字とする 9
環境変数は大文字とする 9
シェル変数はすべて小文字とする 9
複数単語からなる変数名の区切り文字はシェル内で統一する 10
必ず登場する変数の名前 10
必ず登場する関数の名前 10

コーディング規約 12

全般 12
マジックナンバは使用しない 12
シェルの最後は exit 文で終わること 12
ステータスコードの判定方法の統一 13
パス定義は絶対パスとする 13
書式（フォーマット） 15
インデントは空白文字 2文字単位とする 15
1行の最大文字数は半角90文字までとする 15

図４．作成したシェルコーディング規約の表紙と見出し

(2) 報告会（各チームリーダーへ）

規約作成の進捗があった際は、毎日実施されているリーダー会に参加し、活動の進捗報告を行った。そこで意見を頂くことで、達成目標を明確に定めたり、規約内容の指摘を受けて修正を進めることができた。他チームのリーダーに筆者の意見を述べる時は大変緊張した。また最初は、技術的な話を行う際に理解が不足していたため上手く説明できなかった。しかし、報告会での反省を生かし、前向きに「自分が主担当で

行っている改善活動だ。」という責任感を強く持つことで自己啓発にも力が入り、報告会やレビューの準備を徹底したことで徐々に技術的な話や今後の具体的な話を進めることができた。また、周りの方の協力がとても大きく、活動のモチベーションになっていた。

(3) コーディング規約のレビュー

現場の有識者とコーディング規約のレビューを実施した。そこで内容の指摘と様々な意見を聞くことができた。レビューを行う前は、新しいルールを作成することへの周りの期待等でプレッシャーを感じていたが、有識者から多くの意見を頂いたり、レビューに大変多くの工数を割いてフォローして下さったことで、「自分は組織の一員」として仕事をしている、という当たり前の事実を再認識した。筆者一人ではコーディング規約の作成は業務経験や知識の不足からも、不可能であった。しかし周りの協力を仰いだことで規約は完成した、更に諸先輩方からは「よく網羅されていて内容は良い」という評価を頂けた。

4. 期待できる導入効果

期待できる導入効果として品質の向上とそれによるコスト削減がある。もし、規約を統一して運用していなければ、今後同様の作業を実施する状況が発生した際に、自グループはトラブルの再発防止はできるが、私たちのチームで起きたトラブルが他チームでは再発する可能性がある。

また、今回私たちのチームで起きたトラブルは、実際には５０h 以上対応工数がかかっていた。単純計算をすると、筆者のグループ以外に他に２チーム存在するとしたら、同様のトラブルの工数が１００ｈ発生する。しかし今回作成したコーディング規約を定期的にメンテナンスをして運用すれば、２件のトラブル発生とそれによる１００ｈのコストの発生を防止することになる。

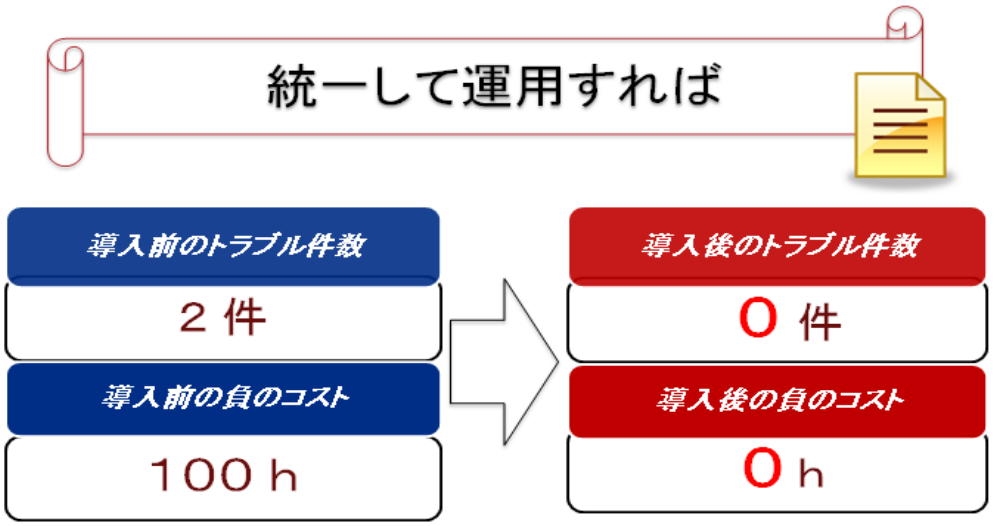


図 5. 期待できる導入の効果の一例

5. 今後の課題

現在、コーディング規約を確実に運用させるためのルール作成、チェック体制を整える活動を始めている。これについては既存のチェック体制を利用しようと考えている。具体的には、既にプロジェクトごとに工程判定という部として必ず承認を得なければいけないチェック体制がある。その中に今回作成した規約を遵守しているかというチェック項目を追加することで対応できると考えている。ただ、これだけでは規約を常に最新化させるための仕組みができていない。この仕組み作りも今後の課題として取り組んでいこうと考えている。

もちろん上記の課題達成は一人では不可能である。しかし、コーディング規約作成に取り組んだ時のように多くの方を巻き込んで改善活動を実施していきたい。そうすれば筆者の期待値以上のものが作成できると考える。

6. おわりに

今回「改善意識」を強く持つことにより、規約作成まで最後までやり遂げることができた。また、情報収集の為、積極的にコミュニケーションを取りに行けるようになったことと、コミュニケーションをスムーズに取るための知識や意見の準備を行っていく習慣が身についたことは成長点だった。今後は、規約を最新化する仕組みづくりがまだ作れていないので、引き続き改善意識を強く持って取り組んでいこうと思う。「5.今後の課題」でも述べたが、一人では不可能なことも、多くの方を巻き込んで改善活動を実施すれば期待値以上のものが作成できることを学んだ。今後もそのことを忘れずに仕事に取り組んでいきたい。

そして、部内にはまだまだ改善しなければいけないことが多数存在する。改善活動とスキルの更なる向上を目指して、「なぜ」「何のために」を考え、「作業一つ一つに疑問を持つこと」を大切にしてお客様により良いサービスを提供できるシステムエンジニアを目指していきたい。