デジタルプラクティス　Vol.10 No.3(July 2019)

一般投稿論文

DevOpsを活用した協調型データ準備プロセス ―Data Wranglingツールおよび ETLツールを併用した複数ロールにおける反復的データ準備―

樫山 俊彦[1]　保田 弘武[1]　角井 健太郎[1]　北脇 淳[1]　鹿野 裕明[1]

[1]（株）日立製作所

データ利活用ニーズが高まる中，ビジネス環境変化のスピードに追随するため，ソリューション開発工数の8割を占めるデータ準備の迅速化が求められている．そのため，ドメイン知識・ITスキルの異なる複数ロール間において，データ準備要件を決定するとともに，検証済み加工データを提供する際の手戻り工数軽減が課題であった．本稿では，インタラクティブなデータ理解・加工を通じた早期の要件決定，およびDevOps活用による迅速な環境構築・開発・検証によるデータ準備プロセスの実践を述べる．過去のデータ分析案件に本プロセスを適用した結果，過去実績工数の1／3に低減できた．

1. はじめに

近年，センシング技術やコンピューティング性能の向上により，製品，環境，作業者等に関するさまざまなデータを収集し，利活用することで，売り上げ向上や仕損費削減等の経営指標改善を得る事例が増えている[1]．またさらには，改善を超えて新ビジネス創生を目指す，ディジタルソリューションの取り組みも各企業で加速している．

経営課題の改善や新しいビジネスを創生するディジタルソリューション実現では，データを利活用した分析アプリケーションの迅速な構築が必要となっている．しかしながら，大量のディジタルデータが企業内のさまざまなシステムや現場に散在している中，分析アプリケーション開発以前のデータ準備に要する手間がアプリケーション構築におけるペインポイントとなっている．

データ準備に要する工数は，データ準備を含むアプリケーション開発の全工数の8割を占めると言われており[2]，データ提供の迅速化のためには，データ準備工数の低減が大きな課題となる．データ準備の手間がかかる理由の一例として，目的とするデータを作るために，システム個別が持つデータの関連性を理解した上で，それぞれの個別データを統合する必要がある．

たとえば，空調装置の設置場所や資産番号等を管理するシステムと，空調の電力値を管理するシステムが個別にあった場合，特定の空調を識別する番号が前者のシステムでは資産番号，後者のシステムでは空調が繋がる分電盤IDというように管理されているとする．データ準備においては，各システムが管理している情報およびその意味を理解する必要があるが，情報の意味を示す

カラム名やデータ内容を理解するためには，業務の詳細を理解する必要があり，大きな手間がかかる．また，空調装置の設置場所と電力値の関係を分析したい場合，空調が接続される分電盤の情報を新たに得る必要があり，担当の部署に入手を依頼するといった事象も生じる．以下では，このように本来の想定に反して，作業のやり直しとなってしまう事象を手戻りと呼ぶ．

一般的に，業務を深く理解している担当者（以下，本ロールをドメインエキスパートと呼ぶ）は，ドメイン知識が高いものの，ITスキルは必ずしも高くない．一方で，データ準備を担当する開発者（以下，本ロールをデータエンジニアと呼ぶ）は，ITスキルが高いものの，ドメイン知識は必ずしも高くない．ドメインエキスパートとデータエンジニアが協力して，データ準備作業を進めていくことがポイントとなる．上記の例で示すように，データ準備を進める上で，ドメイン知識・ITスキルの異なる複数ロール間においてデータ準備要件を決定し，外部データを抽出した上で目的に応じた形式に加工・変換して格納するETL（Extract/Transform/Load）プログラムを介して検証済み加工データを提供する際に，手戻り工数を軽減することが課題であった．

本稿では，複数ロール間におけるデータ理解・加工・検証をする際の課題を解決する協調型データ準備プロセスを実践する．Data Wrangling[3]ツールと呼ばれるインタラクティブにデータを加工できるツールを用いて，ドメインエキスパートがデータ分析対象のサンプルデータに対して暫定加工する過程を可視化し，データエンジニアと情報共有することで，データ準備要件の早期合意，仕様伝達を可能にするプロセスを実践した結果を示す．また，DevOpsで利用されるコンテナ技術等や継続的ソフトウェア開発を活用することで，プロジェクト個別にデータを隔離した環境を迅速に配備するとともに，ETLツールにおいてチーム開発・検証を容易化するプロセスを実践した結果を示す．

2. データ準備の課題およびアプローチ

2.1 データ準備の課題

過去に実施した分析アプリケーション開発案件に基づいたデータ準備業務プロセスと問題点を**図1**に示す．データエンジニアが分析対象のデータを入手後，データ理解，ETL設計，ETL開発，ETL検証を実施し，分析目的ごとに生成されるデータベースであるデータマートの形で所望のデータを提供する．これらのプロセスを実施する際に，2つの問題点がある．

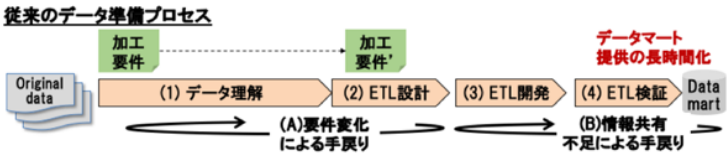

図1　従来のデータ準備プロセスにおける問題

データ理解，およびETL設計にかかわる1つ目の問題として，データ準備要件変化による手戻りが挙げられる．過去に実施したデータ準備では，アプリケーションが利用するデータの要件（分析に必要なデータの所在や加工方法等）を決め切らずにプロジェクトがスタートすることが

あった．また，データ理解が進むと要件が変化し，ETL設計後の変更対応といった手戻りが発生することも多くあった．第1章で述べた空調装置の例では，空調装置の設置場所と電力値の関係を分析するプロセスにおいて，データ理解が進んだ段階で，空調が接続される分電盤の情報を新たに得る必要があるというようにデータ準備要件が変化し，担当の部署に入手を依頼するといった手戻りが生じた．ドメインエキスパートとデータエンジニアの間において，迅速に対象データを選定し，データマートを作成するためのデータ加工・統合要件を合意すること，および合意した要件をデータエンジニアが迅速に理解し，検証されたETLを素早く提供することが課題となる．

ETL開発，およびETL検証にかかわる2つ目の問題として，チーム内での情報共有不足・テスト不十分による手戻りが挙げられる．データ準備では，一般的に1人以上のドメインエキスパート，およびデータエンジニアでチームを組む．一方で，データ準備プロジェクトは短期間になるため，過去に実施した案件では，開発者ローカル環境で作業することが多く，専用の環境を用意することや，一般的にソフトウェア開発で利用されることが多くなってきたDevOpsツール（ソースコードバージョン管理ツール，チケット管理ツール，コミュニケーションツール等）を利用できていなかった．上述の空調装置の例では，ドメインエキスパートが新たに分電盤の情報も用いる形でデータ準備要件が変化したことを把握したにもかかわらず，データエンジニアへの伝達が遅れ，ETL設計・開発を再度やり直すことが該当する．また，ETL開発では，ビジュアルプログラミングツールで開発することがある一方，これらのツールにおいて，DevOpsツールへの対応が不十分であった．そのため，ドメインエキスパートとデータエンジニア，もしくは複数のデータエンジニア間において，情報共有不足による手戻りが発生していた．

図1に示す従来のデータ準備プロセスに対して，実践プロセスにより見込まれる改善を**図2**に示す．要件変化による手戻りの問題に対しては，ドメインエキスパートとデータエンジニア間のインタラクティブなデータ理解・加工を通じた反復的な要件決定により，データ理解における理解時間短縮，およびETL設計における仕様書作成時間短縮が見込まれる．情報共有不足による手戻りの問題に対しては，DevOpsツール活用により，ETL開発における環境構築迅速化やチーム内情報共有による重複作業防止，およびETL検証時のテスト自動化による検証時間短縮が見込まれる．

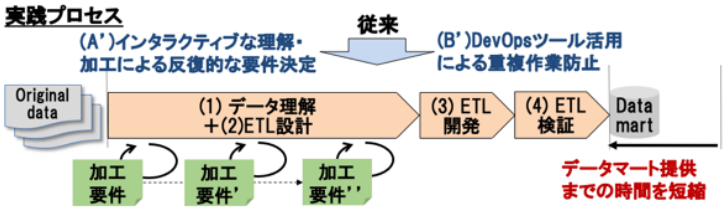

図2　従来のデータ準備プロセスと実践プロセスの比較

2.2　課題解決のアプローチ

課題に対するアプローチ全体像を**表1**に示す．要件変化による手戻りの問題に対応する課題（A'）に関しては，Data Wranglingツールと呼ばれるインタラクティブにデータを加工できるツールを用いて，早期にデータ準備の要件を合意し，テストフレームワークを持つETLツールへ

対応表を用いて移植するデータ準備プロセスを実践する．詳細は，第3.1節，および第3.2節で説明する．

表1　データ準備の課題およびアプローチ

分類	作業工程	課題	アプローチ	選定理由・他方法案
(A')	(1)データ理解＋(2)ETL設計	必要な加工要件を把握できない場合があり，インタラクティブなデータ理解・加工が必要	Data Wranglingツールを用いたデータ理解・加工過程の見える化	他のデータ理解・加工ツールは学習コストが高く，Data Wranglingツールを選定
(A')		人手・目視での確認を避けるため，テストフレームワークを持つデータ変換ツールを利用	Data WranglingツールからETLツールへの対応表を用いた移植支援	Data Wranglingツール統一はデータエンジニア工数増大となるためツールを併用
(B')	(3)ETL開発	データ加工環境を瞬時に用意	クラウドでのETLツールデプロイおよびブラウザでのデータ加工	ローカル環境では柔軟なデータエンジニア追加・変更に対応することが困難
(B')		データ理解・加工を再度繰り返さないために，チームメンバと共有	ETLツールのGitプラグインを用いたGit操作および差分可視化	打合せ等のコミュニケーションでは伝達が曖昧になる，かつ遅れる可能性あり
(B')	(4)ETL検証	テストフレームワークを用いてデータ加工を検証	ETLツールのテストフレームワークを用いたユニットテスト	目視による検証では，不具合発見漏れの可能性あり

情報共有不足による手戻りの問題に対応する課題（B'）に関しては，クラウド環境を用いた迅速なETLツールデプロイ，ETLツールのGit（プログラムのソースコードなどの変更履歴を記録・追跡するためのバージョン管理システム）プラグインを用いたGit操作および差分可視化，ETLツールのテストフレームワークを用いたユニットテストによるETL開発・検証を実践する．詳細は，第3.3節～第3.5節で説明する．

過去に実施した分析アプリケーション開発案件を見ると，データ準備工数肥大化により，当初見積もりと実際の開発コストが乖離している例が多い．空調装置の電力消費量の分析案件では，当初見積もりに対し，開発コストが倍となるケースもあり，データ準備を含む分析アプリケーション開発に掛かる工数を半減させることを目指す．データ準備が全体工数の8割とし，データ準備の工数低減で全体工数を半減させるために，本実践ではデータ準備工数の6割以上の削減を目標とする．

2.3　関連研究

データ工学分野におけるデータ準備では，生データが格納されるデータレイクから，情報を収集・整理・要約・公開するData Wrangling[3]に注目が集っている．Data Wranglingを提供するツールとしては，Wrangler[4]（Trifacta社[5]が提供）や，OpenRefine[6]等があり，スプレッドシート形式のインタフェースによりITスキルが高くない人でも扱えることが1つの特徴になっている．また，この概念は，ITスキルが高くない人でも，自身でデータ準備できるという観点からSelf-service Data Preparation[7]として扱われることもある．

上記を含むデータ変換ツールでは，さまざまな情報源に基づいてデータ変換を支援する．Web情報，知識基盤，クラウドソーシングの情報源を利用するDataXFormer[8]，過去の操作の機械学習結果を利用するWrangler[4]，およびセマンティックWeb技術におけるコンテキスト・オントロジーを利用するKarma[9][10]等が登場している．

しかしながら，高度のドメイン知識，およびITスキルの双方を要求されるデータ分析では，複数人による協業が現実的である．異なるロール間での情報共有支援が必要となるが，従来ツールではカバーできていない．また，データ品質におけるツール比較[11]においても，すべてに優れ

るツールは存在せず，状況に応じて複数のツールを使い分けることが重要になることもある．

ソフトウェア開発方法論の分野では，古くよりさまざまな方法論が議論されている[12]．近年では，継続的ソフトウェア開発[13]に注目が集まっており，アジャイル開発，DevOps，CI/CD（Continuous Integration / Continuous Deployment）等が議論されている．プロジェクト管理，バージョン管理，リリース管理等の各作業において，ビジネス（business / governance），開発（development），運用（operation / maintenance）の間で協力が必要であり，本実践との関連が深い．さらには，Data Science分野にもDevOpsの考え方を取り入れるDataOps[14]という概念も登場してきている．DataOpsの具体的なプロセスに関しては，まだ議論が不足していると考えられ，本稿は，データ理解を通じてデータ変換仕様を決定するデータ準備特有の作業における協力という観点で，DataOpsの実践例とも捉えることができる．

アプリケーション実行インフラの分野では，Docker[15]に代表されるコンテナ技術，およびKubernetes[16]に代表されるコンテナオーケストレーション技術が発展している．これらをもとに，常に処理リクエストを待ち受けるサーバを用意せず，実行時にコンテナをデプロイすることで処理を提供するServerless ArchitectureやFaaS（Function as a Service）という概念も登場してきている[17]．本実践では，これらの技術・ツールを積極的に利用することで，開発・検証作業短縮を狙う．

3. DevOpsを活用した協調型データ準備プロセスの実践

本章では，**図3**に示す簡単な例を用いて実践したプロセスを詳細に説明する．図左上の入力データ（CSVデータ）は，第1章で述べた空調装置のデータを簡単化したものである．カンマ区切りで1行目がヘッダとなっていることは理解できるものの，データの全体構造を理解することができない．

図3　データ理解・加工の例

データ理解フェーズでは，図右上にあるように，Excelデータにして，罫線を引くことで1装置あたり3行で構成されていること，月ごとに列が増えていることを理解することができる．さらには，カラム名の意味や，すべての装置に最大・最小・平均の値が入るか等，ドメインエキスパートのみが把握している内容があるため，データエンジニアと協調して作業する必要がある．

ETL開発フェーズでは，月ごとに列が増えていく構造は分析アプリケーションで処理しにくいため，データを変換・加工することで対応する．本章で説明するETLでは，装置名カラムにnullがなく，1装置で1月あたり1行のデータ構造となるように，入力データを加工する．

3.1　データ理解・加工過程の見える化

ドメインエキスパート自身が加工要件を把握するため，インタラクティブにデータを理解・加工できるツールが必要であり，第2.2節で述べたようにData Wranglingツールが該当する．一例として，Microsoft Excelにおいて利用可能なプラグインであるPower Query for Excel[18]（以下，Power Query）はExcel 2016より標準機能となり，プラグインをインストールすることなく利用できるため，ITスキルの高くないドメインエキスパートが利用するツールとして有力であると考えられる．**図4**にPower Queryにおけるデータ加工画面を示す．

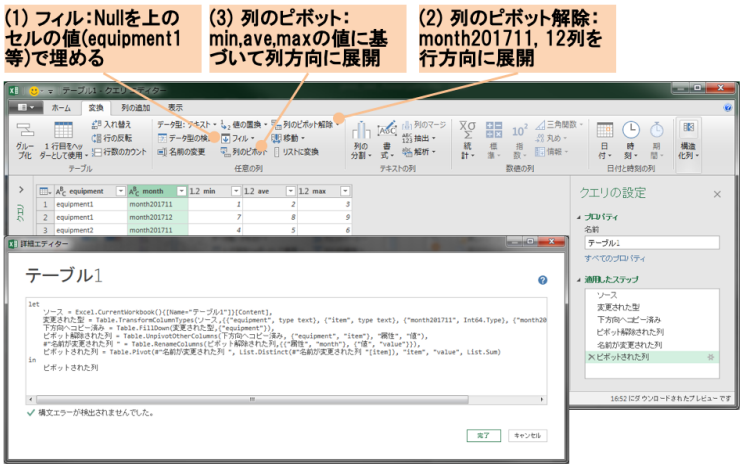

図4　Power Query for Excelを用いたデータ加工

メニューには，フィル，列のピボット等，頻繁に利用されるさまざまな加工メニューがそろっており，集計，テーブル結合，独自処理の列追加等もできる．画面中央に表示される加工後の結果，および画面右下に表示される加工履歴を確認しながら，加工操作を試行錯誤できる．図3に示した加工例の対応としては，フィルを用いることでNullを上のセルの値（equipment1等）で埋めることができる．また，列のピボット解除，および列のピボットを用いることで行と列の構造を変更することができる．

加工操作は画面下に表示されているM言語（加工操作を表現するプログラミング言語）としても記録される．Power Query画面を閉じると，加工後の結果がExcel別シートとして保存されるため，グラフ描画や統計出力等，Excel機能を利用して分析イメージを確認することも可能となる．データ理解・加工の過程で作成した処理の流れは，ほぼ同様のETLとして開発することとなる．したがって，処理の過程をETL開発の仕様書相当とできる．

以上のように，生データから生成される最終的な分析イメージ，および分析に必要なデータマートおよびデータ加工の要件を反復的に決定することにより，結果として早期に最終的な要件を合意することができる．これは，従来の自然文や口頭による要件表現・伝達と比較し，正確，かつ早期に要件を共有できる．

3.2　Data WranglingツールからETLツールへの移植

第3.1節で述べたように，Power Query等のData WranglingツールはITスキルの高くないドメインエキスパートがサンプルデータを理解・加工するには有用なツールである．一方で，データエンジニアが実施するETL運用を考えると，ドメインエキスパートが用いる100行程度のサンプルデータと異なり，数千行を超える大量データが対象となるため扱いにくい問題がある．さらには，APIが不十分でバッチ運用することができない，テストが人手・目視になってしまう，等の問題もある．このように，ドメインエキスパートにとって有用なツールが，必ずしもデータエンジニアにとって最適なツールとならない状況が発生する．

そこで，ドメインエキスパートがData Wranglingツールを用いて示したデータ理解結果やデータ加工要件に対して，テストフレームワークを持つETLツールを用いて，Power Queryで定義した処理を移植する．実践プロセスでは，ETLツールの一例として，Pentaho Data Integration（PDI）[19]を用い，**表2**に例を示すPower QueryのM言語とPDIの処理ステップの対応表を準備することで，この移植を支援する．

表2　Power Query M言語とPDI処理ステップの対応表例

PowerQuery M言語	PDIステップ	備考
Excel.CurrentWorkbook Table.PromoteHeaders Table.TransformColumnTypes	Microsoft Excel Input もしくはCSV file input	
Table.SelectRows	Filter rows	
Table.RenameColumns	Select values	
Table.AddColumn	Add constants	
Table.UnpivotOtherColumns	Row Normaliser	新カラム名，pivotした際の値等を指定する必要あり
Table.Pivot	Sort rows Row denormaliser	denormaliserの前にはSortが必要
Table.Transpose	Row Normaliser Sort rows Row denormaliser	後にpivot/unpivotする場合は，PDIで転置処理を実装する必要ない
Table.SplitColumn	複数ステップ組合せ (Split, Replace, Select values, Concat, Calc等)	文字の位置で分割，PDIではDelimiterによる分割のみ(「月」をDelimiterとして分割した後に整形，正規表現で該当項目を抽出等，工夫が必要)
Table.FillDown	Microsoft Excel Input	FieldsタブのRepeatカラムをYに設定

Power QueryのM言語とPDIの処理ステップは必ずしも1対1に対応しないため，注意が必要である．たとえば，列のピボット機能を提供しているPower QueryのTable.Pivot処理は，基本的にはPDIのRow Denormaliserステップが対応するが，列のピボットをする値ごとにソートする必要があり，Row Denormaliserステップの前にSort rowsステップを入れる必要がある．

このような表を参考にすることで，必ずしもすべての処理ステップを把握していないデータエンジニアであっても，適切にPDIの処理ステップを選択できる．さらには，この表の行ごとに詳細な説明を用意することで，さらにデータエンジニアを支援することも可能だと考えられる．

以上，第3.1節，および第3.2節で述べてきたように，Power Queryを用いて早期にデータ準備の要件を合意し，テストフレームワークを持つPDIへ対応表を用いて移植するデータ準備プロセスにより，要件変化に伴う手戻りを低減できる．

3.3　クラウド環境でのETLツールデプロイ

データ準備の要件を合意し，作業工程が開発フェーズになると，加工するための環境が必要となる．また，プロジェクト個々のセキュリティ要件等により，他環境と隔離した専用の環境を要求される場合もあるため，クラウド環境にプロジェクト専用のETL開発環境を用意することが適当である．そこで，PDIによるETL開発をブラウザ上で実行できるwebSpoon[20]，およびコンテナ技術を活用したクラウド環境を利用し，瞬時に専用の環境を用意する．

webSpoonは，**図5**に示すように，PDI開発環境でローカル環境にインストールして利用するSpoonを，Webブラウザ上で利用可能なようにしたものである．これにより，クラウド環境での利用，開発者間での設定共通化，タブレット／スマートフォンからのアクセス等を実現している．

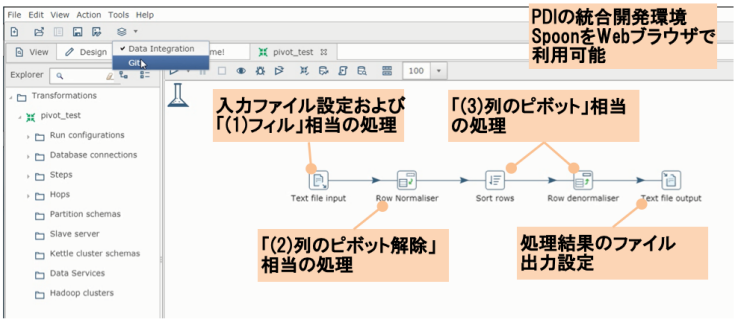

図5　webSpoonを用いたWebブラウザでのETL開発

図4に示した加工例では，表2の対応表を用いることで，対応する処理ステップを配置することで，ETLを開発することができる．

ここで，図5に示したwebSpoonを迅速にデプロイするためのクラウド環境は，**図6**に示すように，Dockerコンテナ管理プラットフォームのKubernetes[16]，Kubernetesのパッケージマネージャであるhelm[21]，HelmのWeb画面を提供するMonocular[22]から構成される．Kubernetesにデプロイされる各アプリケーションはChartと呼ばれ，Dockerイメージ名やインスタンス数等を記載した各種設定ファイルの集合となっている．実践プロセスでは，webSpoon用のHelm Chartを開発することで，Monocular画面にwebSpoon Helm Chartが表示されるようになり，数クリックで環境を構築することを可能にした．

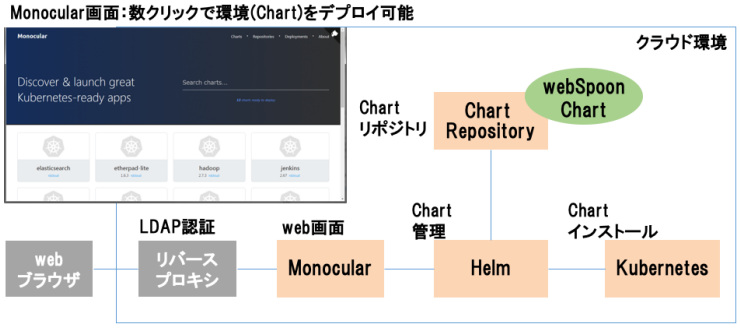

図6　クラウド環境でのwebSpoon Chartデプロイ

3.4　Gitプラグインを用いたGit操作および差分可視化

チームメンバ間での開発物共有に関しては，Git等のバージョン管理ツールを利用することが一般的になっている．たとえば，第3.1節で示したPower Queryを用いてサンプルデータを加工した結果が残っているExcelファイルを随時Gitリポジトリに格納することで，ドメインエキスパートが検討した内容や変更の意図を，データエンジニアが理解できる．

しかしながら，PDIのようなビジュアルプログラミングツールでは，Git操作をするためにターミナル等の別画面に移動する必要がある，PDIで実装したETLはXMLとして表現されるために差分が分かりにくい，といった課題があった．そこで，実践プロセスではSpoonGit[23]を利用する．

SpoonGitは，Spoon上で動作するPDIプラグインで，Spoon画面からのGit操作を容易化するとともに，VisualDiffという強力な独自機能を提供している．**図7**に示すように，画面左中央にコミット履歴，画面右中央に当該コミットにおける差分ファイル名，画面左下に差分ファイルにおける差分詳細が表示されている．差分ファイルに対して，VisualDiffメニューを選択すると，画面左下枠線内の画面が表示され，処理ステップ追加，削除，修正といった差分概要を容易に把握できる．

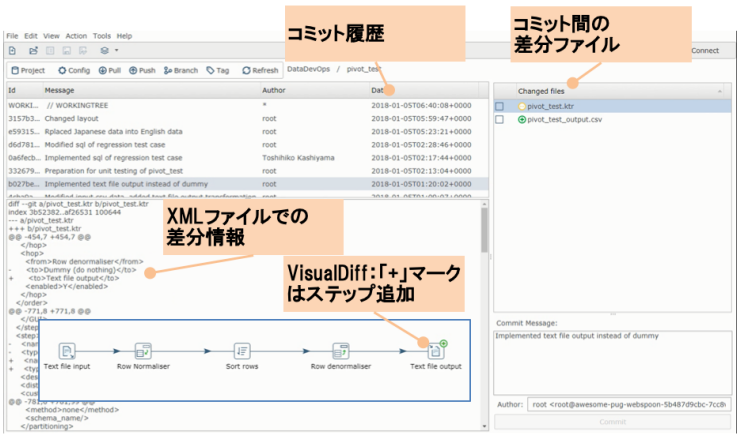

図7　SpoonGitを用いたPDI Git操作およびVisualDiff

3.5　テストフレームワークを用いたユニットテスト

PDIで実装したETLのテストに関しても，テストフレームワークが標準では付随していなかった．そこで，実践プロセスではpdi-dataset-plugin[24]を利用する．**図8**に示すように，各テストケースにおいて入力データと出力データ期待値のペアを指定し，入力データの実行結果が出力データ期待値と一致するか検証している．テストケースを指定した状態でETLを実行するとユニットテスト実行となり，テスト成功メッセージが表示される．複数テストケースをまとめて実行する機能も提供されている．

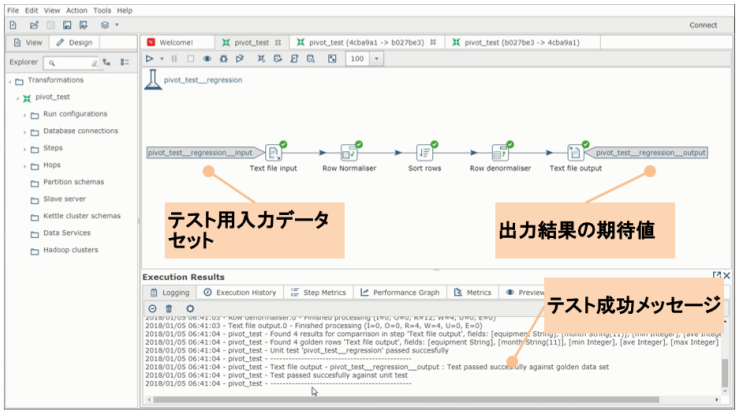

図8　pdi-dataset-pluginを用いたPDIユニットテスト

以上，第3.3節から第3.5節で述べてきたように， クラウド環境を用いた迅速なwebSpoonデプロイ，SpoonGitを用いたGit操作および差分把握の容易化，pdi-dataset-pluginを用いたユニットテストにより，情報共有不足・テスト不足に伴う手戻りを低減できる．

4. 実践結果

4.1　データ準備時間

実践プロセス評価のため，過去に実施した分析案件において，同一データおよび同一目的のデータ準備（ソースデータの理解から目的データを生成するETL開発まで）を実践プロセスにて実施し，データ準備にかかわる工数を比較した．

図9に実践プロセス評価で用いたユースケースを示す．空調装置の稼働状態を把握するために，空調装置と電源の配電にかかわる資産情報から空調装置と接続される配電盤を紐づけ，配電盤での消費電力情報から空調装置の電力，および他の装置に対する空調装置の電力比率を算出して分析することが目的である．入力データは5種類あり，合計で数千行のデータセットを用いている．

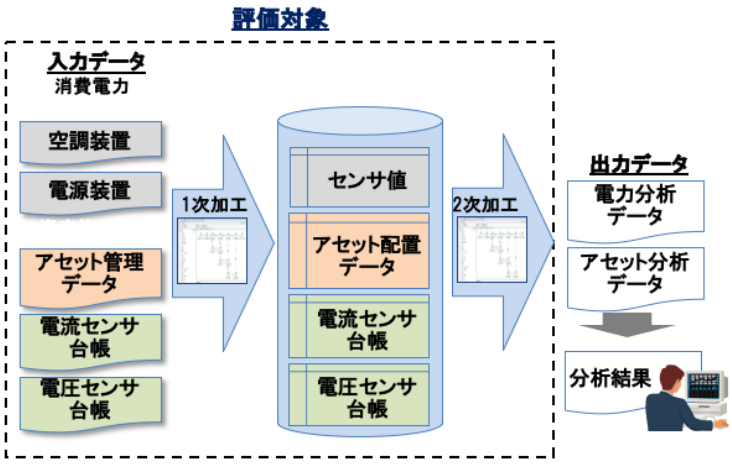

図9　実践プロセス評価用ユースケース

また，**図10**に過去の実装の工数と，実践プロセスによる実装の工数をグラフで示す．データ理解は図9に示す入力データの内容を理解する作業，ETL開発は図9に示す1次・2次加工プログラムを開発する作業，ETL検証は1次・2次加工プログラムを検証する作業，文書作成はデータ理解結果や1次・2次加工プログラムの設計書を作成する作業，調査他は開発・検証作業にかかわる調査などのその他の作業となる．なお，過去の作業工数はExcel線表による実績工数管理で作業時間を取得し，実践プロセスの作業工数はGitシステムに付随するチケット管理システムの実績工数から作業時間を取得している．

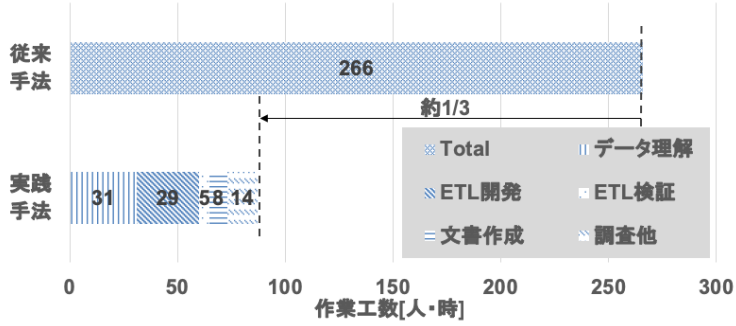

図10　データ準備時間の評価結果

比較対象としたのは，過去に同様のデータおよび目的で実施した分析案件での実績値であり，Excelによるデータ理解とPDIを用いたETL開発に，報告書作成や打合せを含む．過去の実装では，合計38人日（266人時）の工数を要していたのに対し，実践プロセスでは合計工数が87人時となり，同じ人数で同様のデータ準備を実施した場合，実践プロセスでは約1／3の時間でデータ準備が完了できる結果となった．データ準備の内訳をみると，データの理解とETL開発がそれぞれ全工数の約3割を占めている．

今回，比較対象とした従来手法での作業の内訳に関するデータを得ることができなかったため，総作業時間での比較しかできていない．また，プロジェクトマネジメントにかかわる各種作業等，完全に同一条件とはなっておらず，1回の測定のみであるために誤差が生じ得る．工数低減の効果の理由については，別途詳細な評価が必要なものの，従来手法ではETL設計時に，空調が接続される分電盤の情報を新たに得る必要があり，担当の部署に入手を依頼するといった手戻りが実践プロセスでは削減されている．また，結合対象ファイル誤認識による手戻りや，およびETL開発・検証中のテスト不足によるETL開発手戻りを削減できたためと考えられる．全体として，短時間の手戻り回数は多くなっている可能性もあるが，長時間の手戻り回数が減り，総合的に工数が短縮したとも考えられる．いずれにしても，同一のデータおよび同一の目的でのデータ準備，およびETL開発を実施しており，一定の工数削減効果を持つものと考える．

4.2　ETL実装差異評価

次に，従来手法と実践手法での開発したETLの実装差異を評価する．**図11**にETL実装差異の例を示す．本実装では，あるカラムの値として，「センサ値（平均）」，「センサ値（最大）」，「センサ値（最小）」という値を取る際に，平均／最大／最小を区分カラムとして抽出する加工処理を表している．

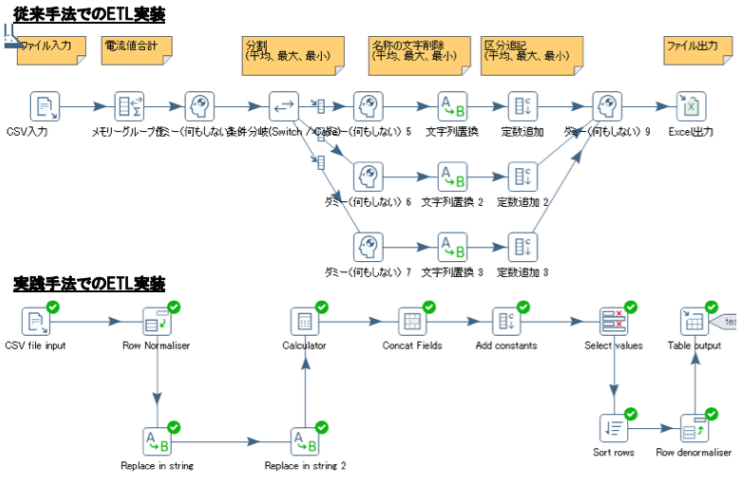

図11　ETL実装差異の例

従来手法と実践手法では大きく以下の3つの差異があった．まず，従来手法では別ツール（Excel等）で転置したCSVを入力したのに対して，実践手法ではオリジナルのCSVを入力していた（前者は追加データ受領の際に，再度手作業で転置をする必要がある）．また，名称の文字削除を，従来手法では条件分岐で実装していたのに対し，実践手法では正規表現で実装していた．さらには，従来ではダミーによるマージのため，レコード出力順が不定となるのに対し，実践手法ではソートの組合せにより決定的となる．

以上に示すように，ETLツールを用いる開発では複数の実装方法が存在し，開発者のスキル等によって差異が発生する．場合によっては，出力順が不定になる等の想定外の事象が発生する可能性もある．従来手法では，目視による検証を実施していたため出力順が不定でも対応可能で，作業時間に大きな影響はなかったと考えるが，実践手法でのテスト自動化実施の上では出力結果

の決定性は必須となる．さらには，開発者によって差異が発生するプログラムは，開発者の意図を把握するまでに時間がかかる，可読性が低くなる，再利用が難しくなる等の弊害が発生すると考えられる．表2に示したような対応表を用いてPower Query処理とPDIステップの移植を支援することにより，PDIの処理ステップに差異がなくなるため，ばらつきの少ないソフトウェアを開発できると考えられる．

5. 結論および今後の課題

データ利活用ニーズが高まる中，ビジネス環境変化スピードに追随するため，ソリューション開発工数の8割を占めるデータ準備の迅速化が求められている．そのため，ドメイン知識・ITスキルの異なる複数ロール間において，データ準備要件決定・検証済み加工データ提供時の手戻り最小化が課題であった．

本稿では，ドメインエキスパートとデータエンジニアにおけるインタラクティブなデータ理解・加工を通じた早期の要件決定，およびDevOpsツール活用による迅速な環境構築・開発・検証によるデータ準備プロセスを実践した．実践プロセスでは，Power Query for Excelを用いたデータ理解・加工，Power Query for ExcelとPentaho Data Integration処理ステップの対応表を用いた移植，クラウド環境でのwebSpoonデプロイによる環境構築，SpoonGitによるGit操作および差分把握，pdi-dataset-pluginを用いたユニットテストにより作業を効率化する．過去のデータ分析案件に本プロセスを適用した結果，過去実績工数の1／3に低減できた．

今後の課題として，まず提案プロセスのさらなる定量評価が挙げられる．本稿での評価は，比較対象の従来プロセスにおいて，作業内訳データを得ることができず，顧客折衝やプロジェクトマネージメントにかかわる各種作業も含んだ値のため，完全に同一条件となっていない．今後，被験者実験等を通じて，同一条件環境下にてプロセスを比較することで定量評価することが必要である．

また，本プロセスの分析方向への拡大が挙げられる．本稿では，データ準備を対象として検討を進めたが，分析結果に基づいてさらにデータを追加したり，さらなる加工により分析に用いる特徴量を追加したりすることが一般的である．本稿の成果をベースとして，分析にかかわるロールや分析ツールを加味して検討することが必要である．

参考文献

1) 総務省：平成27年版情報通信白書 特集テーマ「ICTの過去・現在・未来」第2部 ICTが拓く未来社会,
http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/html/nc254000.html
(2018年9月19日現在)
2) Gil Press : Cleaning Big Data : Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says, Forbes (Mar. 23, 2016) .
3) Terrizzano, I. et al.: Data Wrangling : The Challenging Journey from The Wild to The Lake, In CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA (2015) .
4) Kandel, S. et al. : Wrangler : Interactive Visual Specification of Data Transformation Scripts, CHI '11 Proceedings of The SIGCHI Conference on Human Factors in Computing Systems, pp.3363-3372 (2011) .
5) Trifacta, https://www.trifacta.com/ (2018年9月19日現在)
6) OpenRefine, http://openrefine.org/ (2018年9月19日現在)

7) Market Guide for Self-Service Data Preparation - Gartner, https://www.gartner.com/doc/reprints?id=1-3GQGPV9&ct=160901&st=sb (2018年9月19日現在)
8) Abedjan, Z. et al. : DataXFormer : A Robust Transformation Discovery System, IEEE 32nd International Conference on Data Engineering (ICDE) , Helsinki, pp.1134-1145 (2016) .
9) Knoblock, C. A. et al. : Semi-automatically Mapping Structured Sources into The Semantic Web, In Proc. of the 9th Extended Semantic Web Conf. (ESWC) , pp.375-390 (Springer, 2012) .
10) Taheriyan, M. et al. : Learning The Semantics of Structured Data Sources, Journal of Web Semantics (2016) .
11) Abedjan, Z. et al. : Detecting Data Errors : Where are We and What Needs to be Done?, PVLDB, 9（12) , pp.993-1004 (2016) .
12) Fuggetta, A. et al. : Software Process : A Roadmap, The Future of Software Engineering, A. Finkelstein (Editor) , ACM Press (2000) .
13) Fitzgerald, B. et al. : Continuous Software Engineering : A Roadmap and Agenda, Journal of Systems and Software, Vol.123, pp.176-189 (2017) .
14) From DevOps to DataOps, https://www.tamr.com/from-devops-to-dataops-by-andy-palmer/ (2018年9月19日現在)
15) Docker, https://www.docker.com/ (2018年9月19日現在)
16) Kubernetes, https://kubernetes.io/ (2018年9月19日現在)
17) Serverless Architecture, https://martinfowler.com/articles/serverless.html (2018年9月19日現在)
18) Microsoft Power Query for Excel, https://www.microsoft.com/en-us/download/details.aspx?id=39379 (2018年9月19日現在)
19) Pentaho Data Integration, https://www.hitachivantara.com/en-us/products/big-data-integration-analytics/pentaho-data-integration.html (2018年9月19日現在)
20) webSpoon - GitHub, https://github.com/HiromuHota/webspoon-docker (2018年9月19日現在)
21) Helm, https://helm.sh/ (2018年9月19日現在)
22) Monocular, https://github.com/kubernetes-helm/monocular (2018年9月19日現在)
23) SpoonGit - GitHub, https://github.com/HiromuHota/pdi-git-plugin (2018年9月19日現在)
24) pentaho-pdi-dataset - GitHub, https://github.com/mattcasters/pentaho-pdi-dataset (2018年9月19日現在)

樫山 俊彦（正会員）toshihiko.kashiyama.ez@hitachi.com
2003年東京工業大学工学部情報工学科卒業．2005年同大学大学院情報理工学研究科修士課程修了．同年（株）日立製作所入社．以来，ストリームデータ処理，データ準備向けデータ処理基盤の研究開発に従事．2004年FIT船井ベストペーパー賞受賞．

保田 弘武（非会員）hiromu.hota.co@hitachi.com
2012年名古屋大学大学院工学研究科博士課程修了．博士（工学）．同年（株）日立製作所入社．現在，スタンフォード大学客員研究員を兼務し，ダークデータ解析技術の研究に従事．

角井 健太郎（非会員）kentaro.kakui.eg@hitachi.com
2000年慶應義塾大学環境情報学部卒業．2002年同大学大学院政策・メディア研究科修士課程修了．同年（株）日立製作所入社．以来，ストレージソリューション，システム運用管理技術等の研究開発に従事．

北脇 淳（正会員）jun.kitawaki.yb@hitachi.com
1997年京都大学工学部情報工学科卒業．1999年同大学大学院情報学研究科修士課程修了．同年（株）日立製作所入社．以来，ネットワーク管理，システム運用管理，データ準備向けデータ処理基盤の研究開発に従事．2007～2009年電子情報通信学会TM研（現ICM研）功労賞受賞．

鹿野 裕明（正会員）hiroaki.shikano.gm@hitachi.com
2002年中央大学大学院理工学研究科情報工学専攻修士課程修了．同年（株）日立製作所入社．2009年早稲田大学大学院理工学研究科博士課程修了，博士（工学）．組込みシステム，並列処理，データセンタ運用管理，ネットワーク分析，データ処理基盤の研究開発に従事．2007年山下記念研究賞受賞．

投稿受付：2018年10月5日
採録決定：2019年2月13日
編集担当：串田高幸（日本アイ・ビー・エム（株））