Vol.2018-DBS-167 No.10
Vol.2018-IFAT-132 No.10
2018/9/12

DevOps を活用した協調型データ準備プロセスの提案

樫山俊彦[†1]　保田弘武[†2]　角井健太郎[†1]　北脇淳[†1]　鹿野裕明[†1]

概要：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し，検証済み加工データを提供する際の手戻り最小化が課題であった．本研究では，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を提案する．過去のデータ分析案件に本プロセスを適用した結果，過去実績工数の最大 1/3 に低減できた．

キーワード：データ準備，ETL，DevOps

Proposal of Collaborative Data Preparation Process Utilizing DevOps Method

TOSHIHIKO KASHIYAMA[†1]　HIROMU HOTA[†2]　KENTARO KAKUI[†1]
JUN KITAWAKI[†1]　HIROAKI SHIKANO[†1]

Abstract: Data-utilizing applications are being attracted to improve company's operational key performance indicators or to create new business. Due to faster changes in business environments, such applications need to be released more quickly. However, data preparation is very time-consuming and hindering quick deployment of such applications. Since requirements of data used for analytics applications cannot be fixed before a project starts, or they change even in development, collaboration between multiple persons who have different level of domain knowledge and IT skills is a challenge to determine specification of data transformation. In this research, collaborative data preparation process utilizing DevOps method is proposed. This process realizes quick agreement of concrete specification through interactive data understanding / transformation with data wrangling tools. In addition to that, the process enables to build development environment rapidly, and develop/test ETL efficiently by utilizing multiple plugins of an ETL tool and container technology. Initial evaluation with the existing data and the same goal of data preparation that was used for actual analytics application, data preparation time reduced to about up to 1/3.

Keywords: Data Preparation, ETL, DevOps

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，作業者等に関する様々なデータを収集し，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，大量のデジタルデータが企業内の様々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．

例えば，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例えば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手戻りも生じる．

一般的に，業務を深く理解している担当者は，ドメイン

†1 (株)日立製作所 研究開発グループ
Hitachi Ltd., Research & Development Group
†2 Hitachi America Ltd., Research & Development Group

Vol.2018-DBS-167 No.10
Vol.2018-IFAT-132 No.10
2018/9/12

知識が高いものの，IT スキルは必ずしも高くない．以下，本ロールをドメインエキスパートと呼ぶ．一方で，データ準備を担当する開発者は，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．以下，本ロールをデータエンジニアと呼ぶ．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，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，手戻りを最小化することが課題であった．

本報告では，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提案する．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手法を示す．また，DevOps で利用されるコンテナ技術等を活用することで，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し，データ加工ツールである Pentaho Data Integration (PDI) のプラグインを利用することで開発・検証を容易化するプロセスを示す．

2. データ利活用におけるデータ準備の課題

2.1 データ準備の課題およびアプローチ

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，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 つの問題点があることが分かった．

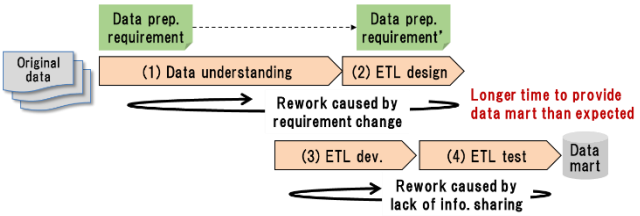

図 1　従来のデータ準備プロセスにおける問題
Figure 1　Problems of conventional data preparation process.

データ理解，および ETL 設計に関わる 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．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 検証に関わる 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．また，ETL 開発では，PDI のような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，提案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．

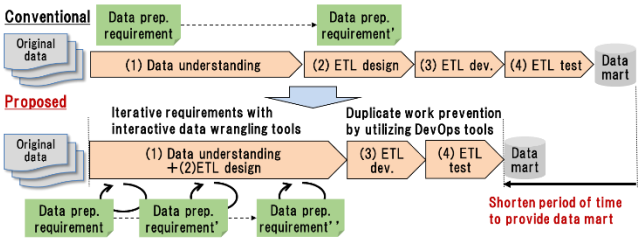

図 2　提案プロセスと従来プロセスとの比較
Figure 2　Comparison between proposed process and conventional process.

また，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#1, #3 に関しては，3.1 で詳細に説明する．情報共有不足による手戻りの問題に対応する課題#2, #4, #5 に関しては 3.2 で詳細に説明する．

Vol.2018-DBS-167 No.10
Vol.2018-IFAT-132 No.10
2018/9/12

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，当初の開発予算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，開発予算に対し，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，本研究ではデータ準備工数の 6 割以上の削減を目標とする．

表 1　データ準備の課題およびアプローチ
Table 1　Approaches against data preparation challenges.

#	Working process	Challenge	Approach
1	(1)Data understanding +(2)ETL design	Interactive data understanding and transformation to recognize non-trivial requirements of transformation	Visualization of sample data transformation processes with Excel Power Query (became as standard feature from Excel 2016)
2	(3)ETL development	Prepare data transform environment only for specific project instantly	PDI development in web browser with webSpoon and easy webSpoon deployment with cloud environment
3		Use data transform tool with test framework to avoid manual confirmation	Porting with mapping table between Power Query M Language and PDI processing steps
4		Information sharing within team members to avoid repeating data understanding and transformation	PDI git operations and VisualDiff with SpoonGit
5	(4)ETL test	Verify transformation with unit test framework	PDI unit testing with pdi-dataset-plugin

2.2 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]や，OpenRefine[6]等があり，スプレッドシート形式のインターフェースにより 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 に精通していない人でも，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，様々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（DBpedia 等），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，古くより様々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，本研究との関連が深い．一方で，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に関しては，議論が不十分である．

アプリケーション実行インフラの分野では，Docker[14]に代表されるコンテナ技術，および Kubernetes[15]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[16]．本研究では，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提案

3.1 処理過程見える化によるデータ準備要件の早期合意

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，Data Wrangling ツールが該当する．中でも，Microsoft Excel において利用可能なプラグインである Power Query for Excel[17]（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 3 に Power Query におけるデータ加工画面を示す．

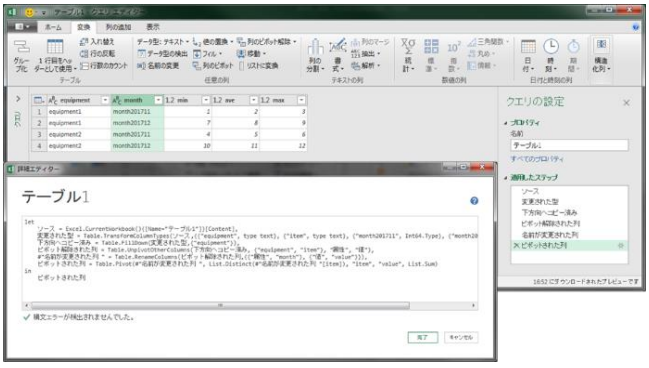

図 3　Power Query for Excel を用いたデータ加工
Figure 3　Data transformation with Power Query for Excel.

メニューには，フィル，列のピボット等，頻繁に利用さ

れる様々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．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．

さらには，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．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．従って，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．

今まで述べてきたように，Power Query 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，ETL 運用を考えると，大量データを扱いにくい，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，等の問題がある．

そこで，テストフレームワークを持つ PDI 等の ETL ツールを用いて，Power Query で定義した処理を移植することを提案する．提案プロセスでは，表 2 に例を示す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
Table 2　Example of mapping table between Power Query M Language and PDI processing steps.

PowerQuery M Language	PDI processing step	Comments
Excel.CurrentWorkbook Table.PromoteHeaders Table.TransformColumnTypes	Microsoft Excel Input / CSV file input	Choose either Excel or CSV based on input file type
Table.SelectRows	Filter rows	
Table.RenameColumns	Select values	
Table.AddColumn	Add constants	
Table.UnpivotOtherColumns	Row Normaliser	Need to set new column names created by normaliser
Table.Pivot	Sort rows Row denormaliser	Sort step required in front of denormaliser step
Table.SplitColumn	Combination of multiple steps (Split, Replace, Select values, Concat, Calc, etc.)	PDI only provides split function with delimiter. Need to realize similar function with replace or regular expression
Table.FillDown	Microsoft Excel Input	Set Yes at Repeat column value in Fields tab

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，注意が必要であるが，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．

上記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最小化を実現できる．

3.2 DevOps ツール活用による環境構築・開発・検証の迅速化

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[18]，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方法を提案する．

webSpoon は，図 4 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．
クラウド環境は，Docker コンテナ管理プラットフォームの Kubernetes[15]，Kubernetes のパッケージマネージャである Helm[19]，Helm の Web 画面を提供する Monocular[20]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．提案プロセスでは，開発した webSpoon Helm Chart を利用することにより，迅速な環境構築を可能にした．

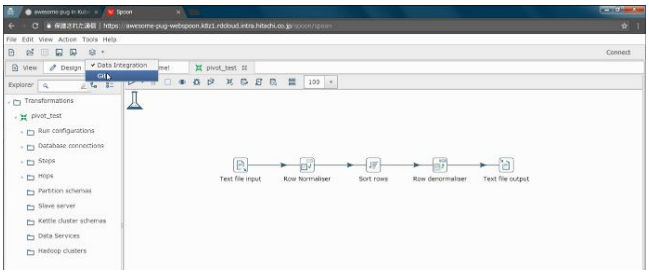

図 4　webSpoon を用いたウェブブラウザでの PDI 開発
Figure 4　PDI development in web browser with webSpoon.

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．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, 提案プロセスでは SpoonGit[21]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 5 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 (緑色の＋), 削除 (赤色の-), 修正（黄色の…）により，差分概要を容易に把握できる．

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，提案プロセスでは pdi-dataset-plugin[22]を利用する. 図 6 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, 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.

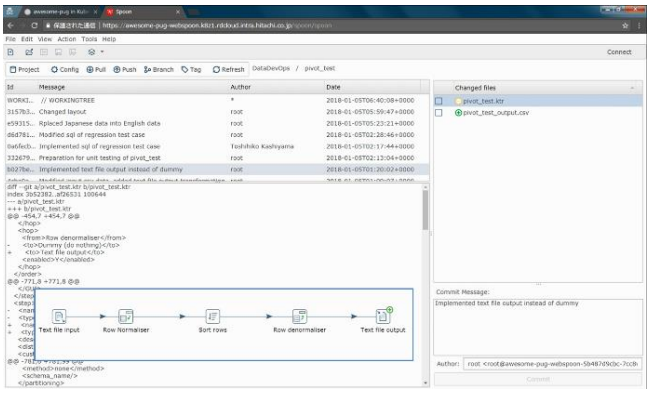

図 5　SpoonGit を用いた PDI Git 操作および VisualDiff
Figure 5　PDI git operations and VisualDiff with SpoonGit.

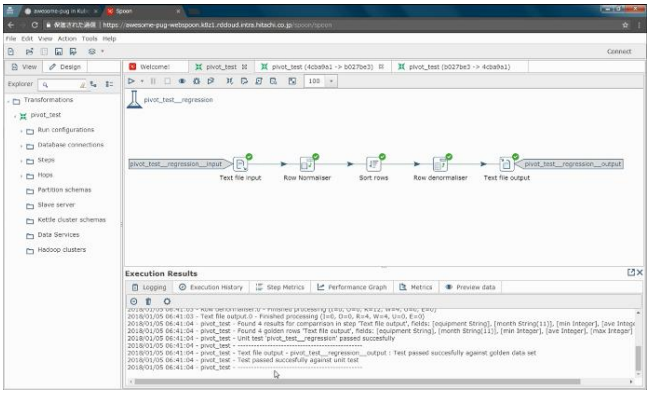

図 6　pdi-dataset-plugin を用いた PDI ユニットテスト
Figure 6　PDI unit testing with pdi-dataset-plugin.

上記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最小化を実現できる.

4.　初期評価

提案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）を提案プロセスにて実施し，データ準備に関わる工数を比較した.

図 7 に本案件での開発物の概要を示す．本案件では，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関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．また，図 8 に過去の実装の工数と，提案プロセスによる実装の工数をグラフで示す.

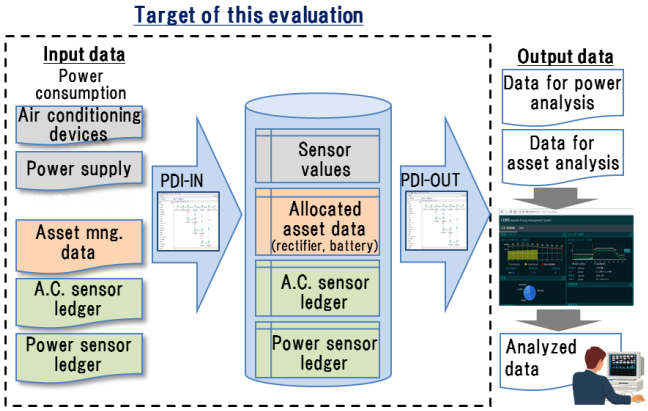

図 7　提案プロセス評価のためのターゲットユースケース
Figure 7　Target use case for evaluation of proposed process.

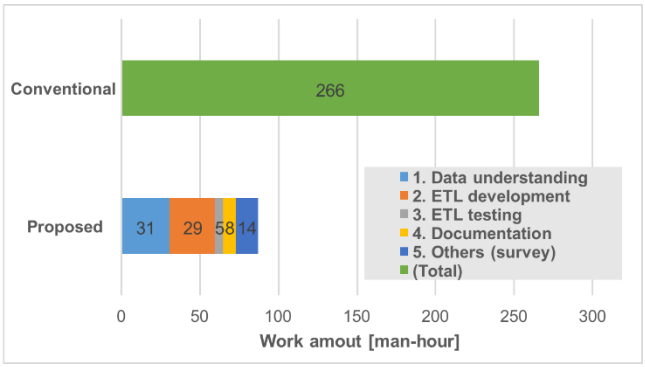

図 8　データ準備時間の評価結果
Figure 8　Evaluation result of data preparation work time.

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）の工数が掛かっていたのに対し, 提案プロセスでは合計工数は 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，提案プロセスでは約 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ず，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である．また，プロジェクトマネジメントに関わる各種作業等，完全に同一条件とはなっていない．しかしながら，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．

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．

本研究では，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提案した．提案プロセスでは，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，過去実績工数の最大 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．本報告での評価は，比較対象の従来手法において，作業内訳データを得ることができず，プロジェクトマネジメントに関わる各種作業等，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も課題である．本研究では，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．本研究の成果をベースとして，分析に関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」第二部 ICT が拓く未来社会", http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/html/nc254000.html (accessed on Jan. 22, 2018).
[2] Gil Press, “Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says,” Forbes, Mar. 23, 2016.
[3] Ignacio Terrizzano, 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] Sean Kandel, 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/ (accessed on Jan. 22, 2018).
[6] OpenRefine, http://openrefine.org/ (accessed on Jan. 22, 2018).
[7] Market Guide for Self-Service Data Preparation - Gartner, https://www.gartner.com/doc/reprints?id=1-3GQGPV9&ct=160901&st=sb (accessed on Jan. 22, 2018).
[8] Ziawasch Abedjan, et al., "DataXFormer: A robust transformation discovery system," IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, pp.1134-1145, 2016.
[9] Craig A. Knoblock, 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] Mohsen Taheriyan, et al., "Learning the Semantics of Structured Data Sources," Journal of Web Semantics, 2016.
[11] Ziawasch Abedjan, et al., "Detecting data errors: Where are we and what needs to be done?," PVLDB, 9(12) pp.993–1004, 2016.
[12] Alfonso Fuggetta, et al., "Software process: A roadmap," The future of software engineering, A. Finkelstein (Editor), ACM Press, 2000.
[13] Brian Fitzgerald, et al., “Continuous Software Engineering: A Roadmap and Agenda,” Journal of Systems and Software, vol. 123, pp.176–189, 2017.
[14] Docker, https://www.docker.com/ (accessed on Jan. 22, 2018).
[15] Kubernetes, https://kubernetes.io/ (accessed on Jan. 22, 2018).
[16] Serverless Architecture, https://martinfowler.com/articles/serverless.html, 2016 (accessed on Jan. 22, 2018).
[17] Microsoft Power Query for Excel, https://www.microsoft.com/en-us/download/details.aspx?id=39379 (accessed on Jan. 22, 2018).
[18] webSpoon - GitHub, https://github.com/HiromuHota/webspoon-docker (accessed on Jan. 22, 2018).
[19] Helm, https://helm.sh/ (accessed on Jan. 22, 2018).
[20] Monocular, https://github.com/kubernetes-helm/monocular (accessed on Jan. 22, 2018).
[21] SpoonGit - GitHub, https://github.com/HiromuHota/pdi-git-plugin (accessed on Jan. 22, 2018).
[22] pentaho-pdi-dataset - GitHub, https://github.com/mattcasters/pentaho-pdi-dataset (accessed on Jan. 22, 2018).