2009－SE－163 (33)
2009／3／19

プロジェクトの分析にBI機能を効果的に用いるための設計プラクティス

藤井 拓[†], 細川 亮二[†]

本論文では，プロジェクトから得られる様々な測定データを格納し，多面的に分析するための手段として市販のリレーショナルデータベースに付随するBI (Business Intelligence)機能を使う事を提案している．BI機能を使うことにより，開発終了時点だけではなく，開発途上での開発の進展を定量化し，どのような要因が開発生産性や品質に及ぼしたかを詳しく分析することが可能になる．著者らは，このような BI 機能に基づくプロジェクトの分析システムを開発し，プロジェクトの分析を行う過程で変更容易性や拡張性が重要であることに気づいた．本論文では，そのような変更容易性や拡張性を高めるための設計プラクティスを報告する．

Design Practices for Effective Use of BI Function on Project Analysis

Taku Fujii[†], Ryoji Hosokawa[†]

In this paper, application of BI (Business Intelligence) function, included in commercial relational database product, is proposed to store of various project data and analyze them over multiple aspects. In addition to analysis at project completion period, progress of software development can be quantified with BI function, and that enables detailed analysis on effectiveness of various development practices and tools used in the project. With the experience of developing project analysis system on BI function, authors have recognized to the importance of changeability and extendibility of the system. The design practices that are effective to enhance changeability and extendibility of the analysis system are reported in this paper.

1. はじめに

近年，ソフトウェア開発を行うための開発ライフサイクル，ツール，プラクティスなどの選択肢が広くなり，それらの組み合わせたプロジェクト毎の開発方式も多様化している．同時に DI (Dependency Injection) コンテナやリッチクライアントなどの新技術が登場し，普及しつつある．そのように多様化した開発方式や新技術の適用において，どんな因子が開発生産性や品質にどのような影響を及ぼすかを分析するための手段が求められている．

従来からソフトウェア開発プロセスの改善を行うために，以下のような測定が行われてきた．

A) 開発工程にレビュー作業を組み込み，そのレビュー回数，指摘件数とテスト段階で検出した欠陥数とを測定し，これらのデータから欠陥を効果的に予防できるように開発プロセスを改善する
B) 開発終了時点のコード行数，労力，欠陥数を測定し，それらから開発生産性や欠陥密度を求める

A)は，ウォーターフォール型の開発ライフサイクルに基づく標準的な開発プロセスが用いられ，ドメインや技術にも共通性があるような開発組織において用いられていることが多い．その一方で，開発途上で要求の追加や変更が多く発生したり，顧客によってドメイン，開発プロセス，技術がマチマチな受託開発では，A)のような測定や改善を行う事が難しい．

B)は，開発プロセスの違いによらず測定できる．しかし，開発終了時点でプロジェクト全体の開発生産性や欠陥密度などを測定しても，その結果から個別の開発作業が効果的に行われたかという点や，要求の追加/変更が開発生産性にどのような影響を及ぼしたかという点などを読み取ることは困難である．そのため，測定結果からプロセスを改善するためのヒントを得ることは困難である．

近年，ソフトウェア開発以外の一般的な業務分野では，プロセスをモニターし，その改善を定量的に行うための技術としてデータベースの BI (Business Intelligence) 機能[1]が用いられてきている．BI機能は，業務プロセスで発生する多様なデータを統合し，複数の異なる視点（次元）で分析をすることを可能にする．BI 機能を使うことで業務の実行状況を把握したり，業務プロセスの有効性を評価するための指標の値を求めたりすることが可能になる．

著者は，ソフトウェア開発で作成される様々な開発成果物の変化や開発作業の記録を自動計測するためのプロジェクトデータウェアハウス[2]というツールを提案した．この過去の研究では，成果物の改訂履歴をメタ要素に分解/格納し，成果物の変化量を出力するという機能の実現方法が中心であり，成果物変化以外の測定に基づく分析は行っていない．

本研究では，コード行数，労力，問題点などソフトウェア開発プロジェクトの開発過程で得られるデータを

† (株)オージス総研 技術部ソフトウェア工学センター

BI 機能により多面的に分析することを可能にするシステムの実現を目指している．このようなシステムを用いることで，ソフトウェア開発を効果的に行うためのプロセス(作業)やツールを明らかにできる可能性がある．本論文では，BI 機能を用いたプロジェクトの分析ツールを設計する際のプラクティスについて報告する．以降，2 節では BI 機能を構成する要素を説明し，それらの要素により多次元的データをどのように格納し，分析するかという概要を説明する．3 節では，著者らが提案する IDeAn というプロジェクトの分析ツールの初期の設計を説明し，さらに初期の設計の問題点とその解決策について説明する．4 節では，得られた知見をまとめるとともに，今後の課題を述べる．

2. Business Intelligence 機能とは

BI 機能は，一般的に以下のような基本要素で構成される．

- ETL(Extract Transformation Load)機能:外部のデータを読み，必要なデータ変換を行い，データベースに格納する機能
- データウェアハウス: 外部のデータをファクトとディメンジョンに分解して格納するためのデータベース
- OLAP(Online Analysis Processing):多次元的なデータを様々な軸に沿って高速に集約するためのデータベース
- レポート機能:OLAP のデータをグラフや表の形式に変換する機能

通常，ETL 機能でロードしたデータはスタースキーマという特殊なスキーマを持つリレーショナルデータベースに格納される．このデータベースをデータウェアハウスと呼ぶ．スタースキーマは，データを集約する軸となるディメンジョンと，集約の対象となるデータを格納するためのファクトの 2 種類のテーブルで構成される．各ファクトに注目すると，ファクトは放射状にディメンジョンを参照するために，このスキーマをスタースキーマと呼ぶ．図 1は，スタースキーマの例を示す．この例では，「売り上げ」というファクトに対して，その売り上げが発生した「日」や「月」という時間のディメンジョン，その売り上げが発生した「支店」や「地域」という「場所のディメンジョン，その売り上げの対象となった「製品」や「製品種別」の製品のディメンジョンの 3 つのディメンジョンが結びついている．1 つのディメンジョンを構成する

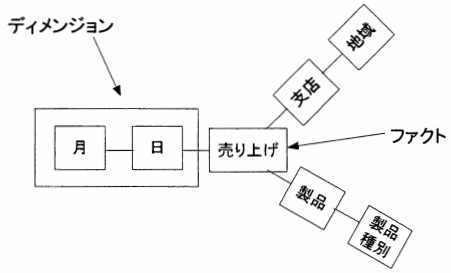


図 1 スタースキーマの例

「日」や「月」などの複数のノードはディメンジョンに対する粗さの異なる複数の目盛りを表している．

BI 機能では，このデータウェアハウス中のデータを OLAP キューブという構造に変換する機能を提供している．OLAP キューブは，概念的には各ディメンジョンに対応する次元軸を辺に持つ立方体(正確には多次元空間の立体)とイメージすればよい．この立方体は次元軸の目盛りで分割されており、各次元軸の目盛りで区切られた 1 つずつのセルにその位置のファクトの値が存在する。この各セルに位置するファクトのデータのことをメジャーと呼ぶ．このキューブ中のメジャーを特定の次元方向に集約したり，特定の次元の格子で切り出したりすることにより，様々な次元間のデータの傾向を得ることができる．図 2は，図 1のスタースキーマに格納されたデータを OLAP キューブに変換した例を示す．この例において，時間軸の「1月」で切り出し，支店別と製品別に集約すると，「1月の各支店の製品別売り上げ」が得られる．また，特定の製品「製品A」で切り出し，売り上げを月と支店ごとに集約すると「製品A の支店毎月毎の売り上げ推移」が得られる．

プロジェクトから得られる各種のデータは，表1のようにスタースキーマで表現することができる．このスタ

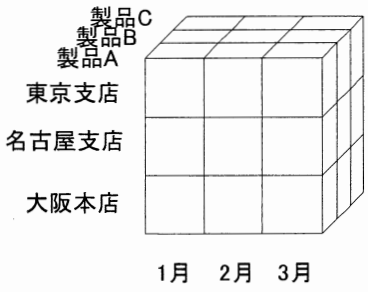


図 2 OLAP キューブの例

表 1 プロジェクトデータの
スタースキーマへのマッピング

ファクト	ディメンジョン
コード行数	時間，成果物，機能
労力	時間，作業種別，作業者，機能
問題点数	時間，機能
コードカバレッジ	時間，成果物，機能

ースキーマからOLAPキューブを作れば，開発過程や成果物構造に沿って多次元的にプロジェクトを分析することが可能になるのではないかと期待できる．

3. BI 機能を利用したプロジェクトの分析システムの設計

3.1. 3.1. IDeAn1.x の設計

前節で述べた着想に基づいて，著者らは 2006 年 12 月からマイクロソフト社の SQL Server の BI 機能を用いたプロジェクトの分析ツール IDeAn (Iterative Development Analyzer) の開発に着手し，2007 年 5 月に IDeAn ver1.0 をリリースした．

IDeAn 1.0 のアーキテクチャを図に示す．本アーキテクチャの特徴は，以下のとおりである．

A) ETL とデータ入力用ユーザインタフェース
- 入力ファイルを読み込み，対応するスタースキーマにデータを追加/更新する
- プロジェクトやプロジェクトの期間などの IDeAn 用に人間が作成するデータはユーザインタフェースから入力し，対応するスタースキーマにデータを追加/更新する

B) 複数の OLAP キューブ
- ファクトごとに OLAP キューブを作成する
- SQL Server の Analysis Service という機能によりキューブを生成している

C) レポート用データ作成処理とレポート用データベース
- C#のコードと MDX クエリでキューブ中のメジャーを集計したり，演算してレポートのグラフや表のデータを生成し，レポート用データベースに追加する
- レポート用データ作成処理の制御フローは SQL Server の Integration Service という ETL 用のビジュアル言語を用いて定義している

D) Web レポート
- レポート用データベース中のデータを Web ページでグラフや表として表示する
- SQL Server の Reporting Service という機能を利用している

ここで，MDX は OLAP キューブに対する問い合わせを行うために SQL を拡張した問い合わせ言語である．

ETL とデータ入力用ユーザインタフェースを採用したのは，プロジェクトのメンバーに直接測定データの入力をしてもらう必要があるだろうと考えたからであった．

複数の OLAP キューブという構成に至ったのは，プロジェクトからコード行数，労力，欠陥数等の測定データが得られるタイミングがバラバラだと当初想定していたからである．開発途上での開発生産性などの分析データは，特定の時点のコード行数と労力を演算することで求まるはずである．しかし，この両者のデータの

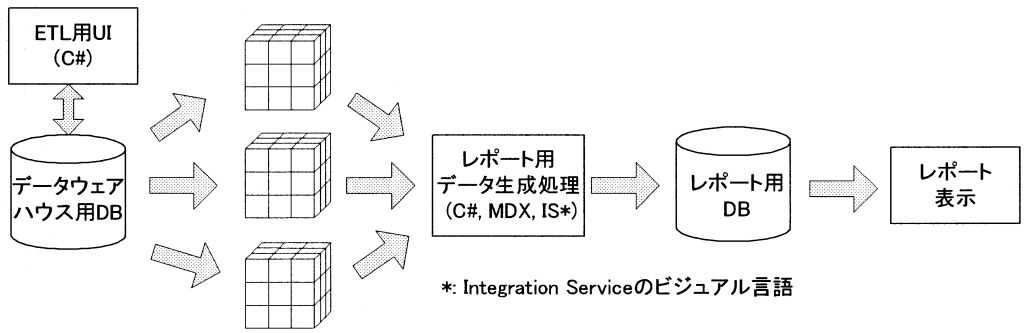


図 3 IDeAn 1.0 のアーキテクチャ

測定タイミングが異なるとそれらのデータを単純に時間軸に集約したり，得られた集約値を単純に演算しただけでは正確な測定値は求まらない．測定のタイミングがバラバラであると，得られたデータが時間次元を共有する 1 つの OLAP キューブに統合できないということである．これらのデータ間で機能次元も共有できる可能性があったが，この時点では異なるデータ間で共有できるような機能次元を定義できなかった．そのため，次元を共有できないこれらの複数のデータを 1 つの OLAP キューブに統合しなくてもよいのではないかと考えたのであった．結果として，レポートを生成するために複数のキューブにあるデータ間で複雑な演算を行う必要が生じて，その演算を行うためのレポート用データ作成処理を用意した．

Web レポートは Web 上でレポートが参照できればプロジェクトメンバーにフィードバックができると考えて，SQL Server の BI 機能に含まれるレポート機能を利用して作成した．

3.2. 3.2. IDeAn1.x の拡張と設計上の問題点

2007 年以降，IDeAn1.0 を用いたプロジェクトの分析を開始した．測定対象は、リッチクライアントと J2EE ベースのソフトウェアを社外で開発しているプロジェクトであった．このプロジェクトに対応するために，IDeAn に以下のような機能を追加した．

- Java のコード行数測定ツールへの対応
- Action Script のコード行数測定ツールへの対応
- 反復ごとの変更差分などの分析項目
- オフラインレポートの出力機能

これらの追加は各々異なるタイミングで行われたが，これらの機能を追加する度毎に以下のような状況に陥った．

- 1つの機能追加に1ヶ月以上要する
- レポート出力に計算間違いが多発

このような状況は，IDeAn のアーキテクチャに根本的な問題があるために発生するのではないかと考え，問題点を分析した．その結果，根本的な問題は以下の 2 点ではないかと思われた．

- 機能を追加する際に変更しなければならない範囲が分散している
  - 機能を追加する際にデータベーススキーマ，ETL とデータ入力用ユーザインタフェース，レポート用データ作成処理の複数部分を変更しなければならない
- レポート用データ作成処理が複雑すぎる
  - レポート用データ作成処理は，複数のキューブから MDX でデータを取得し，それらを C#で集計、演算していた．この処理が非常に複雑で変更が難しい

また同時に，機能追加を行う毎にスタースキーマが複雑化していくという問題や，レポート出力の数が増えて拡張や維持が困難になるなどの問題も抱えていた．

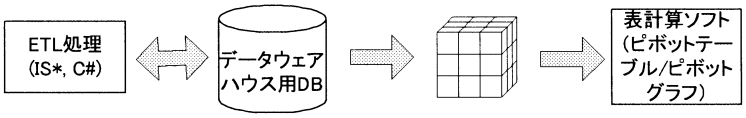
プロジェクトの分析を行う上では、今後もプロジェクト毎の特殊性やレポート出力する内容の変更はたびたび発生することが予想された．そのため，このような変更容易性や拡張性の低さは許容できないと考えた．

3.3. IDeAn 2.0 の設計

IDeAn の変更容易性や拡張性を高めるヒントを得るべく BI システムの設計に関する書籍[3]，[4]を読むと，以下のような設計プラクティスが推奨されていることが分かった．

A) データベースの数はなるべく少なくする
B) ETL 処理をディメンジョンやファクト単位でモジュール化する
C) データウェアハウスのデータを 1 つのキューブに統合する

A)は，データウェアハウス用のデータベースと OLAP キューブの 2 つだけにするということである．OLAP キューブはデータウェアハウス用のデータベースのスキーマに基づいて生成するので，スキーマはデータウェアハウス用のデータベースに一本化される．C)の方式にすることで，次元とメジャーの組み合わせを変えて様々な分析を行うためのデータを生成することが可能になる．この次元とメジャーを組み合わせての分析の実現方法については，前述した書籍ではあまり言及していない．その点については，ビジネスデータの分析に関する書籍[5]が参考になった．この書籍では，表計算ソフトのピボットテーブル/グラフ機能により次元とメジャーを組み合わせて様々な分析を行う方法が解説されている．実際には，これら以外にも細かいレベルで有用な設計プラクティスが提案されているが，



*: Integration Serviceのビジュアル言語

図 4 IDeAn 2.0 のアーキテクチャ

それらについて参考文献を参照して頂きたい.

IDeAn2.0 は, これらの設計プラクティスと表計算のピボットテーブル/グラフ機能を用いて作成された. 図4は, IDeAn2.0 のアーキテクチャを図示したものである.

データウェアハウスのデータを 1 つのキューブに統合するためには, 異なるメジャー間で次元を共有する必要がある. そのために, IDeAn 2.0 では以下の2つの次元を共有することにした.

- 時間
- 機能

時間は, 開発過程の分析を行うために不可欠な次元である. 時間次元は, 基本的に様々な測定データの測定タイミングを揃える事で共有可能になる. 時間次元をどのように揃えたかについては, 後述する. また, 機能はソフトウェアを以下の2階層に分解したものである.

- 機能ユニット
- サブユニット

多くのソフトウェアでは, ユースケースのようなアプリケーション固有の機能とアプリケーション間の共通機能で構成されている. それらの機能は, 最上位ではユーザがユースケースや画面単位のアプリケーション機能と共通機能と捉えることができる. 機能ユニットは, この最上位の機能を表す. さらに, アプリケーション機能や共通機能はビュー, コントロール, エンティティ, その他の内部機能としてさらに分解することができる. サブユニットは, これらの内部機能を表す. 機能ユニットは, 開発の進捗や問題点管理のようにプロジェクト管理において興味を持たれる機能の単位である. それに対して, サブユニットはソフトウェアのアーキテクチャ要素などの単位での開発生産性や開発の進捗などを把握するために役立つ機能の単位である.

IDeAn2.0 では, 時間次元を共有するために以下のようなデータベースの設計を行い, 測定のスケジュールに関する制約を設けている.

- プロジェクトを複数の「期間」に分割し, プロジェクトの各種測定を原則として「期間」の終了日に実行する
- データウェアハウスでは, 「期間」と「日付」で構成される時間ディメンジョンを設ける
- プロジェクトの測定結果は, 基本的に「期間」と結びつくファクトとしてデータベースに取り込むが, 問題点管理データなどのように「期間」より細かい粒度で生成や状態変化が発生するものは「日付」と結びつけてファクトとして格納する

このようにすることで, 時間軸を共有するメジャーとして測定データを 1 つの OLAP キューブに取り込むことが可能になる.

残る問題はレポートを出力するための仕組みだが, これには表計算ソフトのピボットテーブル/ピボットグラフ機能を用いている. ピボットテーブル/ピボットグラフを用いることで, 次元とメジャーの組み合わせを自由に選んで切り出しや集約を行い, その結果を表やグラフに表示することができる. IDeAn2.0 では, 表計算ソフトの OLAP キューブからデータの取り込み機能とピボットテーブル/ピボットグラフ機能を組み合わせることでレポート生成の変更容易性と拡張性を高めている.

本論文の執筆時点で, IDeAn2.0 の設計はほぼ完了しているが, 問題となる点は見つかっていない. 研究会の発表日までには, IDeAn2.0 の実装もほぼ完了している予定であり, 本論文で提案する設計方式に対する実装上の問題の有無も確認できる見込みである.

4. まとめ

本論文では, プロジェクトから得られる様々な測定データを格納し, 多面的に分析するための手段として市販のリレーショナルデータベースに付随するBI機能を使う事を提案した. 筆者らは, このようなBI機能に基づくプロジェクトの分析ツールである IDeAn1.0 を開発した. IDeAn1.0 を実際のプロジェクトに適用する過程で, プロジェクト毎の特殊性に対応し, 新たな分析方法を柔軟に追加できるような変更容易性や拡張性が不十分であることが判明した. このような不十分点を解

決するために，以下のような一般的な BI システムの設計プラクティスを取り入れて IDeAn2.0 を設計した.

- データベースの数はなるべく少なくする
- ETL 処理をディメンジョンやファクト単位でモジュール化する
- データウェアハウスのデータを 1 つのキューブに統合する
- レポート生成に表計算ソフトのピボットテーブル/ピボットグラフを利用する

現時点で，IDeAn2.0 の設計がほぼ完了しているが，これらの設計プラクティスを適用する上での問題は生じていない.

今後，IDeAn2.0 によるプロジェクトの分析を行い，必要なレポートが生成できるかという点を確認するとともに，変更容易性や拡張性が向上したかという点についても評価する予定である.

参考文献

[1] 平井明夫，BI システム構築実践入門，翔泳社，2005

[2] 藤井拓，上林弥彦，プロジェクトデータウェアハウスの概念，オブジェクト指向最前線 2003，近代科学社，2003, pp. 215-222

[3] John C. Hancock, Roger Toren, Practical Business Intelligence with SQL Server 2005, Addison-Wesley, 2007

[4] Joy Munday, Warren Thornthwaite, The Microsoft Data Warehouse Toolkit, Wiley, 2005

[5] 住中光夫，Excel でマスターするビジネスデータ分析実践の極意，アスキー，2003