ソフトウェア工学 130－9
（2001．3．22）

UMLベースの開発成果物の変更解析システム

藤井 拓†, 上林 弥彦††

本論文では，データウエアハウス技術を応用したUMLベースの開発成果物の変更解析システムであるプロジェクトデータウエアハウスの概念及び設計について報告する．本システムは，オブジェクト指向プログラミング言語間の文法の違いや様々な種類の計測に柔軟に対応するために拡張性の良いメタモデルに基づくアーキテクチャを採用している．さらに，多様な解析を可能とするために多次元データベースのテクニックを適用している．

A Change Analysis System for Development Artifacts Based on UML

Taku Fujii, Yahiko Kambayashi

A new system, Project Data Warehouse, for the measurements of changes in development artifacts based on Data Warehouse technology is reported. The new system is based on an architecture that is extendable to store various kinds of UML based development artifacts and flexible to apply the various types of measurements to quantify the changes in those artifacts. Also, it provides multi-dimensional analysis of the measurement results.

1. はじめに

近年のソフトウエア開発において，分析,設計段階でUML(Unified Modeling Language)[1]を用いる開発プロジェクトが徐々に増加してきている．

UMLは，ソフトウエアに対する要求内容や設計構造に対する表現形式を統一し，ソフトウエア開発チーム内外のコミュニケーションに起因する問題の発生を減少させるうえで有効である．また，UMLを用いて詳細設計を行った場合，設計内容と実装を直接的に対応させることができる．その結果，実装と設計間の相互変換を自動化することが可能になる．さらに，UMLによって表現されるオブジェクトモデルをオブジェクト指向モデリングのテクニック[2]を用いて適切に作成することにより，機能の変更や拡張に強い設計モデルを作成できる可能性がある．

このようなオブジェクトモデルの柔軟性や変更容易性に着目して，UMLを活用するためのソフトウエア開発プロセスとして反復型開発プロセスとオブジェクト指向モデリング[3]を適用することが提案されている．反復型開発プロセスは，Boehmらが提唱したSpiral開発モデル[4]を発展させたものであり，要求分析，設計，実装，テストの各工程を1サイクルとし，そのサイクルを複数回に渡って反復することにより開発を進める開発プロセスの方式である．要求分析，設計，実装，テストの各工程から成る1サイクルの開発過程を反復と呼ぶ．反復型開発プロセスは，従来一般的に用いられてきたWaterfall型開発プロセスと比較して開発に伴うリスクの早期低減や顧客の要求により適合するソフトウエアを開発することに優れていると言われている．その一方で，反復型開発プロセスを適用した場合，反復を重ねる度に以前の反復で作成した開発成果物に対する再作業が頻繁に発生して開発の生産性が低下する可能性も生ずる．

UMLに基づく開発成果物を反復型開発プロセスにより開発するアプローチの有効性や問題点を理解したり，プロセスの改善を行うためには，開発の進行状況を定量的に把握することが必要になる．例えば，開発成果物の発展過程を再作業の発生状況とともに計測することにより，反復間で生じる開発成果物の再作業の発生状況や生産性に対する影響を定量的に理解することが可能になる．

本論文では，UMLに基づく設計及び実装成果物を対象に，それらの成果物において開発途上でどのような変更が加えられるかということを定量的に計測，分析するためのシステムであるプロジェクトデータウエアハウスの概念と設計について報告する．

†(株)オージス総研ソフトウエア工学センター
††京都大学大学院情報学研究科

2. 関連研究

開発成果物の変更を計測することにより反復型開発プロセスにおける開発の進捗や成果物の品質を評価するというアイデアは，Royce[5]により提案された．この提案では，進捗や品質の評価に必要なデータは主としてCR(Change Request)を格納するデータベースから収集する方式を用いている．すなわち，CR に対して変更に伴う労力や変更規模の見積もり値や実績値をLOC(Lines Of Code)や作業時間の単位で入力する欄を設け，各開発者が入力した値をデータベース検索により集計してメトリックスを求めるのである．

Royce の提案内容については，計測内容と計測方法の両面で改良の余地があると考えられる．Royce の尺度は，もっぱらLOC や作業時間のように管理的な視点での計測値に基づいており，問題点を発見するうえでは有用であると考えられる．その反面，問題が発生した場合にその技術的な原因を究明するための手がかりとして利用するのは困難である．また，計測を行うために個別開発者の入力を必要としており，一般的なプロジェクトへの適用は困難である．

さらに，開発プロジェクトのステータスを判断する各種データを統合し，提示するための技術であるSPCP(Software Project Control Panel)がBrown[6]により提案されている．SPCP は，計測データの統合や提示という面では非常に有用な提案だと考えられるが，その反面計測過程の自動化や問題が発生した時の技術的な原因究明という点では不十分だと考えられる．

一方で，オブジェクト指向設計やオブジェクト指向プログラミングの成果物の品質評価を行うためのメトリックスの研究もChidamber ら[7]を始めとして過去多数報告されている．これらのメトリックスの研究は，成果物の静的な構造に基づく評価を中心として展開しており，成果物に加えられた変更に基づく計測手法はそれほど研究されていない．また，メトリックスの開発期間における推移を論じた論文[8]においても，成果物の変更過程を計測するためのシステムの提案はなされていない．

3. 開発成果物の計測方法

3.1. 開発成果物のメタモデル

UMLはオブジェクト指向モデルに基づいているため，成果物を構成する中心的な要素は分類子やオブジェクトである．UML の表現しうる概念範囲は，分類子を含むオブジェクトの静的な構造，オブジェクトの動的な振る舞い，その他実装ファイル，コンポーネントの構成やプロセスの構成と幅広い．しかし，このような多様な表現内容の中でも最も広範に用いられているのは，実装コードの宣言と直接的に対応させることが可能な静的な構造を示すためのクラス図である．

クラス図では，以下のような分類子の定義がグラフィカルに示される．

1) 分類子の名前
2) 分類子の種別
3) 分類子の属性
4) 分類子の操作
5) 分類子間の関係

ここで，3),4)については一般的に名前,可視性を始めとするより詳細な定義が与えられる．また，5)については，複数の関係の種類が存在し，関係の種類毎に詳細な定義内容が決まっている．これら3),4),5)の定義の詳細は実装言語によって異なるものが多い．

さらに，UML では分類子の集合を表すためにパッケージというモデル要素を用いることができる．このパッケージという記法を用いて，クラス図で設計モデルのハイレベルの構成を表現することができる．

UMLを用いた場合，このような分類子の定義がどのように変化したかという観点で設計成果物の変化を特徴づけし，計測することができる．例えば，クラスの操作の集合を変更前後で比較することにより，操作の数の増減を計測することができる．

設計成果物の構造を定義する上で用いられる分類子やパッケージ，操作等々のモデル要素は，計測を行うという観点で捕らえれば，図1に示されるような統一的なクラス構造からなる成果物メタ要素として抽象化できる．図1において，各成果物メタ要素はArtifactMetaObject, MetaValue, MetaAggregation, MetaReference の4種類のオブジェクトによりモデリングされている．ArtifactMetaObject は，成果物のメタ要素本体に対応し，MetaValue は成果物メタ要素の属性値，MetaAggregration は ArtifactMetaObject 間の集約関係，

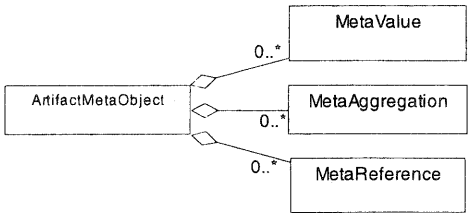

図 1. 成果物メタ要素のモデル

MetaReference は ArtifactMetaObject 間の参照関係に対応する. このモデルは, UML のメタモデルを大幅に簡略化したものになっている. このモデルを適用すると, 分類子, 操作, パッケージ等の UML モデル要素はモデル要素の種別毎に共通の構成を有するオブジェクトモデルとしてモデル化できる. さらに, 特定の種類の成果物メタ要素に対して, その種類の ArtifactMetaObject を作成するための雛形となるオブジェクトを定義することができる. このように特定の ArtifactMetaObject を作成するための雛形となるオブジェクトを ArtifactMetaClass と呼ぶ. このように成果物メタ要素の内部構造をメタオブジェクトによりモデリングすることにより, 分類子や関係の種類について存在する開発言語毎の個別性を柔軟に吸収することができる.

図2は, ArtifactMetaClass や ArtifactMetaObject を中心とした成果物メタモデルの主要なオブジェクトを示している. この図において, 成果物メタ要素を物理的に包含する物理成果物を表す PhysicalArtifact クラス, 物理成果物の改訂履歴を表す Revision クラス, PhysicalArtifact の種別毎に格納するための Aspect クラスが併せて示されている. PhysicalArtifact クラスは, 開発成果物の物理的な実体であるファイルを表現するのに対して, Aspect は分析, 設計, 実装など工程毎の開発成果物を種別毎に分類するために導入されている.

3.2. 開発成果物に加えられた変更の計測と計測結果の利用

3.1 で論じた成果物メタ要素のモデル適用することで, 図3に示されるように開発成果物のバージョン間の変更内容は対応する ArtifactMetaObject や MetaValue の比較を行い, 差違を計量化できる. しかし, 実際にこのような成果物の変更履歴から開発プロジェクトの開発過程を分析する上で有用な情報を抽出するためには3つの問題に対処しなければならない。

まず1点目の問題は, 計測の拡張性ということである. すなわち, 2.1 で述べた成果物メタ要素について計測できる変更内容は MetaValue, MetaAggregration, MetaReference の組み合わせで非常に多くのバリエーションを考えることができる. メトリックスの研究を行うためには, このようなバリエーションの多様さに対応できるように計測の拡張性が求められる.

2点目の問題は, データの利用者の関心に応じた計測結果の利用や提示を行える必要があるという点である. 開発プロジェクト内での役割の違いによって, 開発成果物の計測値に対する関心が異なる. 通常, 開発者は単一あるいは複数のパッケージを担当しており, 担当するパッケージ群全体に渡る変更の傾向が主たる関心対象となる. 他方, プロジェクト管理者の立場では, 設計成果物全体の変更傾向に関心を寄せる可能性高い. このような関心の違いに対応できることが計測システムに求められる.

3点目の問題は, 反復型開発プロセスにおける進捗や生産性の理解に役立つデータを計測しなければいけないという点である. 反復型開発プロセスにおける開発の進捗をモニターしたり, 生産性を把握するためには, 各反復内での進捗や生産性のみならず, 複数回の反復により進行する開発期間全体での進捗や生産性を把握するために役立つデータが必要になっている.

1点目の問題は, 計測システムの設計において解決できる問題であり, 3.3 においてこの問題に対する一つの解決策を示す. 2点目の問題の解決には, 計測したデータを関心レベルの違いに応じて集約したり, 検索したりすることが求められる. このような問題は, データベース技術の1分野である多次元データベースが解決しようとしている問題と同じである. 従って, 多次元データベースの実現技術により解決できる. 3点目の問題は, 開発の特定時点で作成された開発成果物から相対的な変更を計測する方法をいくつか導入することで解決できる. 以降, 多次元データベースによる関心対象の違いへの対応方法及び反復型開発プロセスにおける成果物の変更計測のいくつかの形式について論ずる.

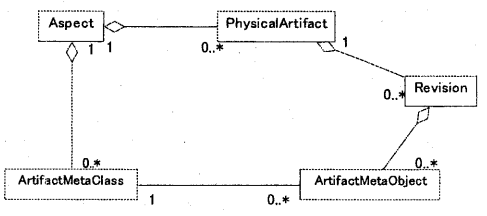

図 2. 成果物メタモデルの主要なオブジェクト

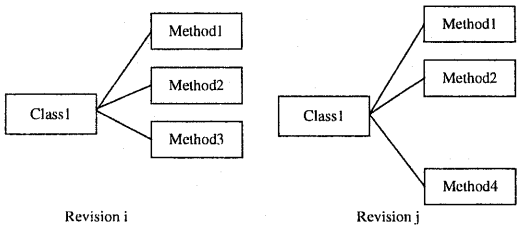

図3 バージョン間の成果物変更の計測

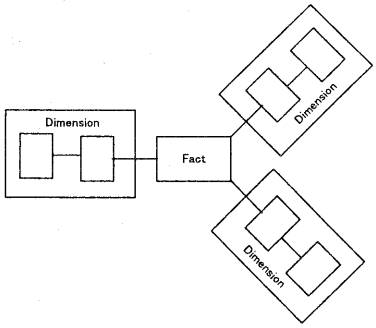

図4 Star Schemaの基本構造

多次元データベースは，データを特徴づけるための複数の次元からなる次元空間にデータを空間的に配置し，様々な観点，詳細度でデータを多面的に解析することを可能にする技術である．このような多次元データベースは，一般的には図4で示されるような Star Schema 構造[9]を用いて関係データベース上に実現される．Star Schemaは，1つのFactと複数のDimensionから構成され，Factと複数のDimensionとの間に関係が存在する構造からなっている．Fact は，複数の Dimension で特定される次元空間の1点で発生した事象である．また，各 Dimension には複数のノードがあり，各 Dimension に対する集約階層を表すことができる．このような Star Schema 構造を用いることにより，データを解析する人の興味に即してFactを複数のDimensionの様々なノードに対して集約したり，選択することが可能になる．

このような Star Schema の考え方を成果物の変更計測に適用した場合，Fact に相当するものは変更計測値群となる．さらに，そのような Fact を特徴づける Dimension としては，計測した日時，成果物の変更箇所，変更を行った開発組織部分の3つを考えることができる．図5は，成果物の変更計測結果に Star Schema を適用した例である．各 Dimension のノードは，以下のような考察に基づいて設定されている．

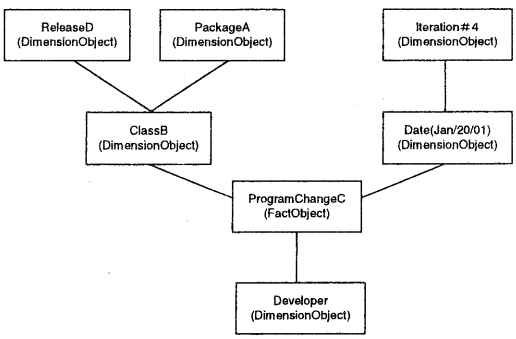

図5 変更計測結果への Star Schema の適用例

反復型開発プロセスを考えた場合，時間の Dimension においては，1日,1回の反復,複数の反復に渡って成果物の変更計測結果を求めたり，集約できることが求められる．そのため，日(Date)と反復(Iteration)が時間 Dimension のノードとして設定されている．この場合，変更計測は日単位に行われ，1回の反復に対する変更トレンドはそれら日毎の変更計測結果を集約することにより求めることができる．

成果物の Dimension においては，クラス,パッケージ,リリースなど開発成果物の異なる単位で変更計測結果を求めたり，集約できることが求められる．そのため，成果物 Dimension のノードとして クラス(Class),パッケージ(Package),リリース(Release)が設定されている．この場合，変更計測はクラス単位で行われ，パッケージ単位の変更トレンドはそれらの変更計測結果を集約することにより求めることができる．

開発組織の Dimension においては，様々な組織階層のレベルで変更計測結果を集約できる必要がある．そのため，開発組織の Dimension としては開発者(Developer)，開発グループ,開発チームのような組織構造を反映したノードが必要になる．そのようなノードを設定することにより，開発者単位の変更計測結果を集約することにより任意の開発組織レベルでの変更トレンドを求めることができる．

図5で示されるリリース(Release)というノードは，成果物の Dimension の1ノードとして集約の単位となるだけではなく，成果物の変更を特定のリリースに対する相対量として求めるために用いることができる．このようにリリースを表すノードを導入することでそのリリースを構成する成果物要素の集合やそのバージョンを特定することが可能になり，そのリリースに対する相対的な変更計測が可能になる．

さらに，反復型開発プロセスを考えた場合，1回の反復における正味の開発の進捗や生産性を把握することが有用である．このような1回の反復における正味の開発の進捗や生産性は，直前の反復で作成された最終リリースを基準として成果物の変更を計測することができる．直前の反復で作成された最終リリースを基準として成果物の変更計測を行う場合には，開発の進捗や開発生産性を把握するために役立つような変更の計測方式や変更計測結果の集約範囲を考える必要がある．

特定の基準リリースを起点とした成果物の変更計測

方式としては，変更差分量計測と累積変更量計測の2種類の計測方法を考えることができる．この場合，変更差分量とは特定の時点の成果物と基準リリースにおける成果物の差分を意味する．それに対して，累積変更量とは基準リリース以降，特定時点まで加えられた成果物の変更の累積量を表す．このような成果物の変更差分量は開発の正味の進捗を成果物レベルで示すのに対して，成果物の累積変更量は特定期間に渡る変更労力を示すと考えられる．そのように考えるならば，累積変更量と変更差分量を対比することにより特定期間に渡る成果物の変更に関する試行錯誤の発生状況を把握することができるはずである．このような試行錯誤に関する情報は，開発組織の様々なメンバーが自分自身の開発の進捗や生産性を把握する上で有用である．また，反復単位で求めた成果物の変更差分量や累積変更量に関する開発期間を通したトレンドを求めることにより，開発の進捗状況をより大局的に把握することが可能になる．

一方で，変更計測結果を集約する成果物範囲の分類としては，現在の反復の直前の反復の最終リリースから引き継いだ成果物部分とその最終リリース以降追加された成果物部分に分類することが有用である．このように変更計測結果の集約範囲を設定することにより，成果物に加えられた変更を過去の反復で設計,実装された成果物部分に対する変更と現在の反復で設計,実装された成果物部分に対する変更に分離することができる．過去の反復で設計,実装された成果物部分に対する変更は，設計の拡張性が低いための設計変更，実装コードの欠陥や問題，当初想定していなかった仕様変更などに起因する．非常に大雑把な目安としては，このような過去の反復で設計,実装された成果物部分に対する変更は直接的に開発生産性に貢献しない活動と見なすことができる．一方で，現在の反復で設計,実装された成果物部分に対する変更は，成果物の拡張や置き換えなどに起因する．非常に大雑把な目安としては，このような現在の反復で設計,実装された成果物部分に対する変更は開発生産性に貢献する活動を表していると見なすことができる．

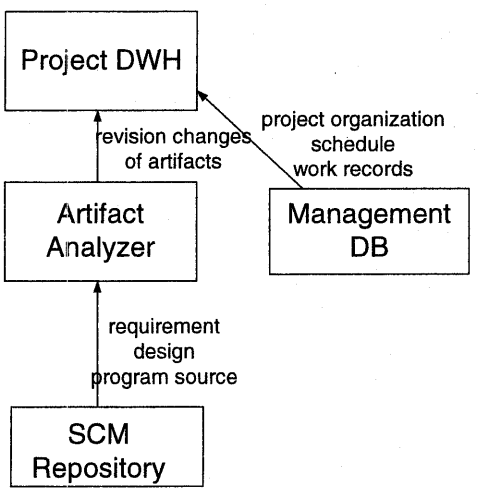

図6 Project Data Warehouse のシステム構成

以上論じたように，直前の反復の最終リリースを基準として変更差分量や累積変更量という形式で成果物の変更を計測したり，その計測結果を基準となる最終リリースから受け継いだ成果物部分に対する変更と最終リリース以降追加された成果物部分に対する変更に分解して集約することにより，反復型開発プロセスを実行しているプロジェクトの進捗や生産性へ影響する開発活動の様子を定量的に把握することが可能になる．また，そのような開発活動の定量的な把握は個別の開発者の開発能力の向上に役立つばかりではなく，開発プロセスを改良にも役立つと期待できる．

3.3. Project Data Warehouse の設計

3.2 で論じた3つの問題に対応するための成果物の変更解析システムを Project Data Warehouse と呼ぶ．図6に，Project Data Warehouse のシステム構成を示す．この図で，Artifact Analyzer と呼ばれる部分が 3.1 で論じた成果物をメタ要素に分解してその履歴とともに格納するレポジトリーである．Artifact Analyzer は成果物範囲の Dimension や成果物の変更に対応する Fact を作成するためのデータを提供している．一方，Management DB とは，反復期間や全体の開発スケジュールや開発チームの構成を保持するためのデータベースであり，Project Data Warehouse の時間や開発組織に対する Dimension を作成するためのデータを提供する．また、SCM Repository は市販の構成管理ツールの成果物データベースを表している．

図7は， Project Data Warehouse の Star Schema 構造を保持するために用いられる主要な永続クラスを示したものである．この図において Star Schema の Dimension の各集約レベルに対応するのが DimensionObject であり，Fact に対応するのが FactObject である．DimensionObject 間や DimensionObject,FactObject 間に定義された関係により，Star 構造が実現されている．

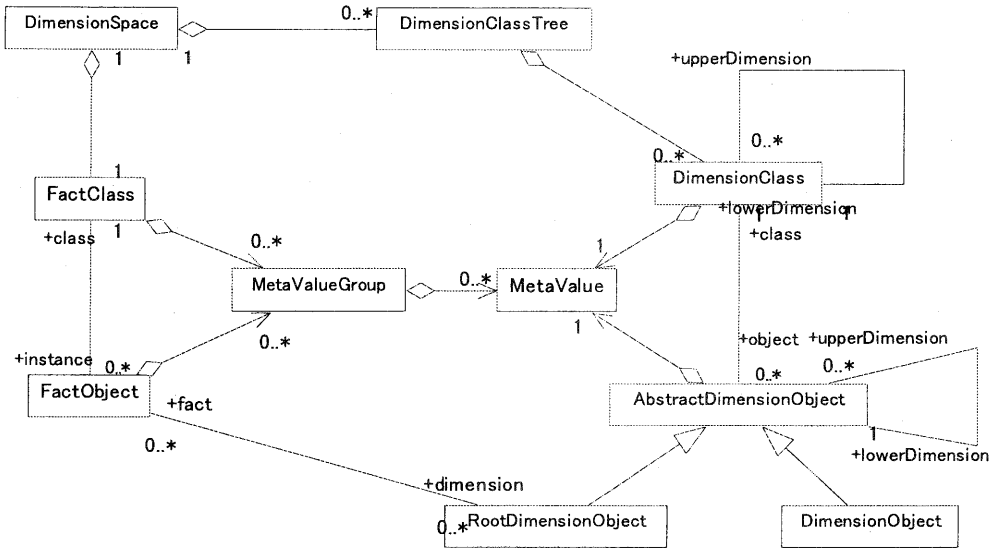

図7 Star Schemaを実現する設計モデル

FactObject は，複数の計測値を保持するMetaValueGroup を保有する構造となっており，DimensionObject は単一の値を MetaValue において保持する構造となっている．また，2.1 で論じた成果物メタオブジェクトと同様に同一種類の DimensionObject や FactObject を生成するための雛形としての役割を担うDimensionClass や FactClass が定義されている．さらに，一つの Dimension を構成する DimensionClass の木構造を定義するために，DimensionClassTree というクラスが導入されている．また，個別の計測結果を格納する単位として DimensionSpace というクラスが定義されている．クラス図から分かるように，DimensionSpace は単一の FactClass と複数の DimensionClassTree を単位に作成されている．

成果物の次元に対応する DimensionObject や成果物の変更を FactObject は，Artifact Analyzer に蓄積された ArtifactMetaObejct から作成することができる．これらの DimensionObject や FactObject を作成するために，Artifact Analyzer から図8に示すように時間方向と ArtifactMetaObejct 方向の2方向で成果物の改訂情報を取得できることが必要になる．

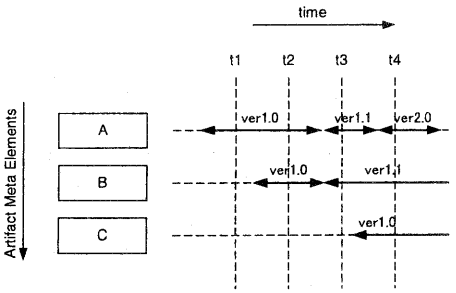

図8 改訂内容に対する2次元のアクセス

さに，Artifact Analyzer から得られた成果物メタオブジェクトや管理データベースのデータは，図9に示される Populator クラス群により Dimension や Fact に変換される．DimensionPopulator は Dimension のインスタンスを作成し，各 Dimension 軸に沿った木構造を形成する責務を担っている．また，これらの DimenionPopulator クラスは Dimension の集約階層と反映する形で構成されており，次元軸や集約レベルを柔軟に追加,変更することが可能になっている．一方で，Fact オブジェクトの Populator は MetaValueGroup 単位で計測を行う Measurement オブジェクトから構成されている．FactObjectPopulator を構成する Measurement オブジェクトとしては，ArtifactMetaObejct の差違を一次計測する Measurement オブジェクトに加えて，差違の累積を行うための Measurement オブジェクトのような2次データ生成用の Measurement オブジェクトを併用することも可能である．

このような DimensionObject や FactObject と対応する Populator の構造を形成することにより，Dimension の定義や Fact に格納する計測結果を柔軟に変更することが

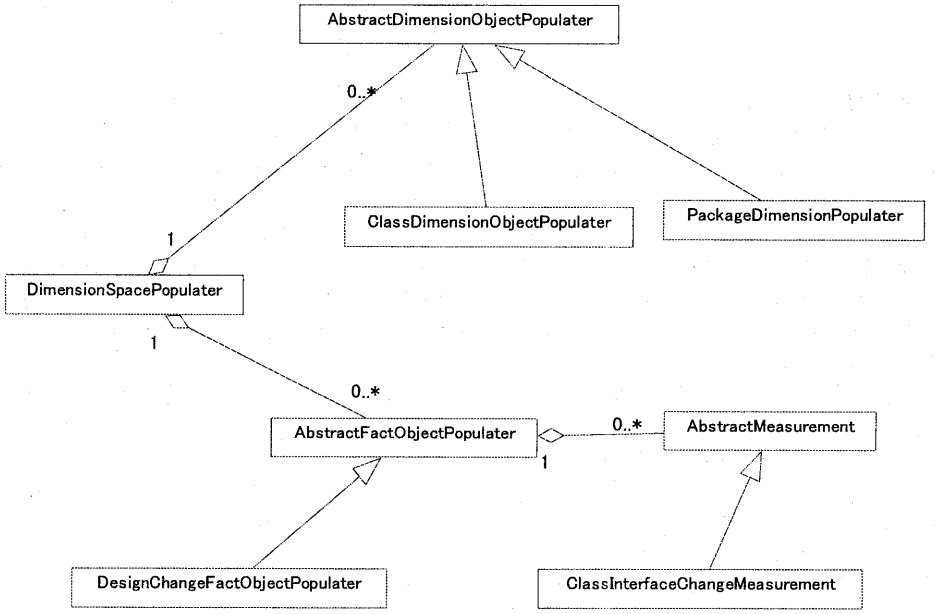

図9 DimensionObject と FactObject を生成するためのクラス

可能になった.

3.4. Project Data Warehouse のプロトタイプシステム

3.3 で論じた設計構造に基づいて Project Data Warehouse のプロトタイプの実装を行った. プロトタイプでは, Project Data Warehouse, Artifact Analyzer, Management DB はすべて OODBMS(Objectivity/DB for Java)を用いて実装された.

実装成果物の実装コードの変化はメソッド単位でパーサにより計測され, 成果物メタ要素に分解されて複数の改訂履歴が成果物アナライザーに蓄積された.それに対して, 設計成果物はビジュアルモデリングツール(Rational Rose)から COM の API を通じて取得され, 成果物メタ要素に分解されて複数の改訂履歴に渡り成果物アナライザーに蓄積された.

Project Data Warehouse 内の DimensionObject や FactObject は, Artifact Analyzer に蓄積されたデータを Java で実装された Populater オブジェクトや Measurement オブジェクトにより変換して作成することができる. 現在のプロトタイプでは, Dimension としては時間と開発成果物だけが生成できる. また, 計測内容としては, C++プログラムコードに対してメソッド単位の変更差分量及び特定リリースを基準とした累積変更量の計測が可能になっている. また, 設計成果物についてはクラス数, 操作数, 属性数, 集約関係数, 継承関係数の変更差分量や特定リリースを基準とした累積変更量の計測が可能になっている. また, 実装成果物や設計成果物の変更計測において, 変更量を追加的な成分と削除的な成分に分解して計測を行っている さらに, 成果物の Dimension 中のリリースノードを用いて, 計測対象の成果物を特定リリースに含まれる部分と特定リリース以降に追加された部分にグループ分けして成果物に加えられた変更の計測結果を集約することが可能になっている.

設計成果物や実装成果物の成果物メタ要素の変更を計測する場合には, 利用できる場合にはビジュアルモデリングツールで発生した ID に基づいてメタ要素のマッチングを行い, 利用できない場合にはオブジェクト名やシグニチャを用いてメタ要素のマッチングを行っている.

計測結果として得られた DimensionObject や FactObject は, FactObject 単位に結合して ASCII ファイルにダンプし, ISAM DB や RDBMS にインポートすることができる. Dimension 軸に対する選択や集約演算は ISAM や RDBMS 上で実行され, その結果が SpreadSheet プログラムに出力され, 各種トレンドグラフ

や分布図等を作成することができる.

4. 今後の課題

現在実装できているProject Data Warehouseを用いて, 実プロジェクトの開発成果物を計測し, 反復型開発プロセスにおける開発の進捗状況の定量化を試みる予定である.

また, Project Data Warehouseの現在のプロトタイプではデータを計測したり, DBMSに格納する過程の自動化が実現されているが, 今後さらにDBMSに格納された計測データ利用を支援する機能の整備が求められる. そのためには, 計測データの提示方法を研究するとともに, 計測データが開発プロジェクトの問題点の発見や分析にどのように役立つかという点について研究を行う必要がある. 前者については, Webによる計測結果の提示方法を今後研究しようと考えている. また, 後者については成果物に加えた変更の性質や成果物の品質を自動的に判定するための技術を今後研究する予定である.

5. まとめ

本論文では, UMLに基づく成果物の変更過程を解析するためのシステムであるProject Data Warehouseの概念および設計を紹介した.

このような成果物の変更過程を解析するためのシステムを設計する上では, 成果物を作成する際に使用される様々な開発言語間の文法の違いや成果物を計測するための計測方法のバリエーションに対応できることを目指す必要がある. また, 計測結果をプロジェクト管理者や開発者のように役割に応じた関心の違いに対応して提示できるようなシステムの実現が求められる. さらに, 反復型開発の開発過程を分析するためには, 特定の成果物リリースを基準とした成果物の変更内容の計測やグループ分けを行う必要がある.

Project Data Warehouseの設計において, 開発言語間の文法の違いや成果物を計測するための計測方法のバリエーションへの対応は, 成果物や計測結果に関するメタモデルを導入することにより解決されている. また, 関心の違いに対応して計測結果を柔軟に提示できるようなシステムを実現するという課題は, Star Schema構造という多次元データベースの一般的なテクニックを用いることで解決されている. さらに, 特定の成果物リリースを基準とした成果物の変更内容の計測やグループ分けを行うという課題については, Star Schemaの成果物Dimensionにおいて Releaseというノードを導入して解決することができた.

参考文献

1. The Unified Modeling Language Specification. Online-at http://www.omg.org/.
2. Booch, Grady: Object-Oriented Analysis and Design with Applicaitons, Benjamin/Cummings (1994).
3. Jacobson, Ivar, et.al.: The Unified Software Development Process, Addison Wesley Longman (1999).
4. Boehm, Barry W.; "A Spiral Model of Software Development and Enhancement", *Computer*, Volume 21, Number 5, pp.61-72 (1988).
5. Royce, Walker: The Software Project Management – A Unified Framework, Addison Wesley Longman (1998).
6. Brown, Norm: "Industrial-Strength Management Strategies", *IEEE Software*, Volume 13, Number 4, pp.94-103 (July 1996).
7. Chidamber, Shyan R., et al.: "A metrics Suite for Object Oriented Design", *IEEE Transaction on Software Engineering*, Vol.20, No.6, pp.476-493 (1994).
8. 中谷多哉子, 玉井哲雄:"オブジェクトの進化プロセスにおける統計的観測", 情報処理学会ソフトウエア工学研究会報告, Vol.120, No.55,pp.69-76 (1998)
9. Kimball, Ralph: The Data Warehouse Toolkit, John Wiley & Sons (1996).