フレームワークを用いた
情報システムの継続的開発に関する研究

2008年1月

湯浦 克彦

フレームワークを用いた
情報システムの継続的開発に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2008年1月

湯浦　克彦

内容梗概

近年，企業経営の環境の変化が激化するなかで，企業のビジネスを支える情報システムにおいても，数年毎の全面再構築の時期を待つことなく，運用中のシステムに対して保守あるいは機能追加を行って，継続的に開発を進めることが求められている．

こうした継続的開発を効率的に行うためのシステム構築法として，フレームワークが注目される．フレームワークは，オブジェクト指向による組み立て型のソフトウェア構造の一つであり，固定部分と変化部分を区別してライブラリを提供するという考え方を持っている．情報システムの設計において，将来保守案件の発生が予想される部分をフレームワークの変化部分として構築し，継続的開発を効率化することが期待される．

本論文の研究は，フレームワークを用いた継続的開発を実用化することを目的とし，主に下記に述べる 2 つの課題に関して取り組まれた．

フレームワークはシステムの根幹に関わる構造であり，かつオブジェクト指向など新しい設計技術を含むため，導入を判断するには十分な評価が必要である．そこで，フレームワークの実用化に向けた第一の課題としては，導入効果を定量的に評価する体系の確立が必要となる．また，これまでのフレームワークの実績は，主にオペレーテイング・システムのライブラリ利用や，Web システム構築の基盤構築など技術的な領域に多い．これに対し，情報システムにおいては業務内容に関わる領域が大部分を占めており，継続的開発の上では，技術要素の構造化だけではなく，業務要素の構造化がより強く求められる．そこで，第二の課題としては，業務分野に特化して効果をもたらすフレームワークの機能や実現方式を明らかにする必要があると考えられた．

第一の課題であるフレームワークの導入効果については，計測方法の改良，および計測による効果の分析という 2 つのアプローチから研究を進めた．

情報システムの機能規模の計測に関しては，データ・ファイルとそれを処理するトランザクションを中心に計測するファンクションポイント法が，すでに主流となっている．そこで，これをクラス定義中心のオブジェクト指向型ソフトウェアに適用する方式が期待される．本研究では，クラス定義に加えてイベントトレース図に含まれるメソッド定義の情報を取り入れて，ファンクションポイントを計測する独自の方法を考案した．そして，熟練者による計測との比

較評価を行い，高い計測精度が得られることを検証した（第 2 章）．

計測によるフレームワーク導入効果の分析に関しては，機能規模および複雑度の両面から分析した．機能規模に関しては，クラス定義に対する既存のファンクションポイント計測法である OOFP（Object-Oriented Function Point）に基づき，あるフレームワークを用いた場合のシステム開発と用いない場合の開発の比較を行った．その結果，フレームワークの導入により機能規模が半分以下で済んでいることが計測された．ただし，フレームワークが開発する機能との整合性に乏しい場合には機能規模の削減効果が少ないことも同時に分析された．

複雑性に関しては，すでに C&K メトリクス（Chidamber & Kemerer のメトリクス）がよく使われているので，これを用いてフレームワークを用いた場合の開発と用いない場合とを比較した．フレームワークを用いた場合に複雑性が高くなることが計測されたが，分析の結果，フレームワーク内部の品質が十分高ければシステムの信頼性には影響が少ないことが示された（第 3 章）．

第二の課題である業務に特化したフレームワークに関しては，情報技術の知識を有したシステム担当者ではなく，ビジネスの専門業務担当者による継続的開発，および一般利用者に対する業務利用支援の立場からフレームワークを設計した．題材としては，財務報告記述の標準化言語である XBRL（Extended Business Reporting Language）で記述されたデータを処理するシステムを選んだ．

専門業務担当者向けには，XBRL に含まれる XLink, DOM など XML 関連の新技術を隠蔽した専用のデータモデルである SDX（Simple Data Model for XBRL Documents）を提案した．そして，SDX に基づくデータバインディング・ライブラリや専用言語を開発することにより，業務データ（XBRL 財務データ）へのアクセス記述を容易化するフレームワークの一方式を示した（第 4 章）．

一般利用者向けには，会計業務の専門家と共同した検討に基づいて，財務データを用いて企業への投資などを判断する手順に関する知識を形式化し，対話型ツール FIAT（A Financial Analysis Assistance Tool）として実装した．これにより，専門家が行うような財務分析と投資判断の手順やそこで用いる判断基準を，個人投資家などの一般利用者が参照しながら財務データを有効に利用するためのフレームワークの一方式を示した（第 5 章）．

主要論文

1. 柏本隆志，楠本真二，井上克郎，鈴木文音，湯浦克彦，津田道夫： “イベントトレース図に基づく要求仕様書からのファンクションポイント計測手法”，情報処理学会論文誌，Vol.41，No.6，pp.1895-1904，2000.1.

2. Hikaru Fujiwara, Shinji Kusumoto, Katsuro Inoue, Ayane Suzuki, Toshifusa Ootsubo, Katsuhiko Yuura: “Case Studies to Evaluate a Domain Specific Application Framework Based on Complexity and Functionality Metrics”, Information and Software Technology, Vol.45, Issue 1, pp.43-49, 2003.1.

3. 湯浦克彦，泉田聡介，松下 誠，井上克郎： “分析業務に関する知識を用いた財務分析支援方式”，情報処理学会論文誌，Vol.49，No.1，2008.1.（採録決定）

謝 辞

本研究の全般に関しご指導を賜りました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻　井上克郎教授に感謝を申し上げます．本研究の多くの部分は，報告者が大学院博士後期課程に入学する以前より開始されていた，井上研究室と報告者が所属する日立製作所との共同研究に端を発しており，その時代も含め 10 年余りにわたってご指導いただいたことに対し，あわせて感謝を申し上げます．

本論文の第 2 章および第 3 章の研究に関してご指導，ご協力をいただき，また本論文を執筆するにあたりご助言をいただきました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻　楠本真二教授に感謝を申し上げます．

本論文を執筆するにあたり，ご指導とご助言をいただきました，大阪大学産業科学研究所　八木康史教授に感謝を申し上げます．

本論文の第 4 章および第 5 章の研究に関してご指導，ご協力をいただきました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻　松下誠准教授に感謝を申し上げます．

井上研究室において共同で研究にあたった，柏本隆志氏，今川勝博氏，藤原晃氏，渡辺貴史氏，高尾祐治氏，泉田聡介氏に感謝いたします．

本論文の第 5 章の研究に関しご協力をいただきました，筑波大学大学院ビジネス科学研究科国際経営プロフェショナル専攻　白田佳子教授およびあずさ監査法人の筏井大祐公認会計士に感謝を申し上げます．

日立製作所において，第 2 章および第 3 章の研究に関して共同で研究にあたった鈴木文音氏，第 3 章の研究に関して共同で研究にあたった大坪稔房氏（現在は日立コンサルティングに所属）に感謝いたします．また，日立グループ各社において，本論文の研究に関連するプロダクトである REQUARIO，画面遷移フレームワークおよび XiRUTE（XBRL）に関する技術開発および普及利用をともにした多くの技術者に感謝いたします．

目 次

第1章　はじめに

1.1　情報システムの継続的開発

1.1.1 情報システムの開発と保守

ここ数十年，企業においては，企業内の業務プロセスを効率化することを主な目的として，次々と情報システムの導入が進められてきた．近年では，企業間にわたる取引や業務の計画に関しても導入が推進されている．企業における情報システムへの投資は年々増大する傾向にあり，企業経営のうえでは情報システムに関わるコストの削減が大きな関心事となっている．

情報システムを構築するためには，その企業の業務の特性を反映したソフトウェアを開発することが必要であり，一般に初期開発に大きな費用が発生する．ところが，情報システムが企業内に広く普及するにつれて，日々新たな業務上の要件が発生する頻度が高くなっている．そこで，現在稼動するシステムに対する保守という形で要件に対応する負荷が増大する傾向にある．

わが国における情報システムの開発と保守の費用について，図 1.1 を用いて説明する．

経済産業省の平成 18 年度情報処理実態調査 [KZ07] によれば，わが国の企業における情報投資における新規開発および保守は，費用にしておよそ 3：7 の割合である．ここで新規開発のうちの約半数は，既存のシステム群が存在し，それらを一部引き続き活用して新システムを構築する「再構築」のプロジェクトである．また，日本情報システム・ユーザ協会の企業ＩＴ動向調査 2007 [JU07] によれば，新規開発においては，ハードウェアの費用とソフトウェア開発／購入の費用の割合が約２：８である．保守においては，ソフトウェアの追加的開発と修正が約 4 分の 1 を占め，残りは運用に関わる人件費やハードウェアの維持等に割り当てられている．以上のデータによると，情報システムに関わる投資のうちの約 24%（＝　3/10 × 8/10）が初期のソフトウェア開発に費やされ，約 18%（＝　7/10 × 2.5/10）がその保守に費やされていることになる．また，再構築を保守の一部とみなすならば，開発が約 12%および保守が約 30%と分析することができる．

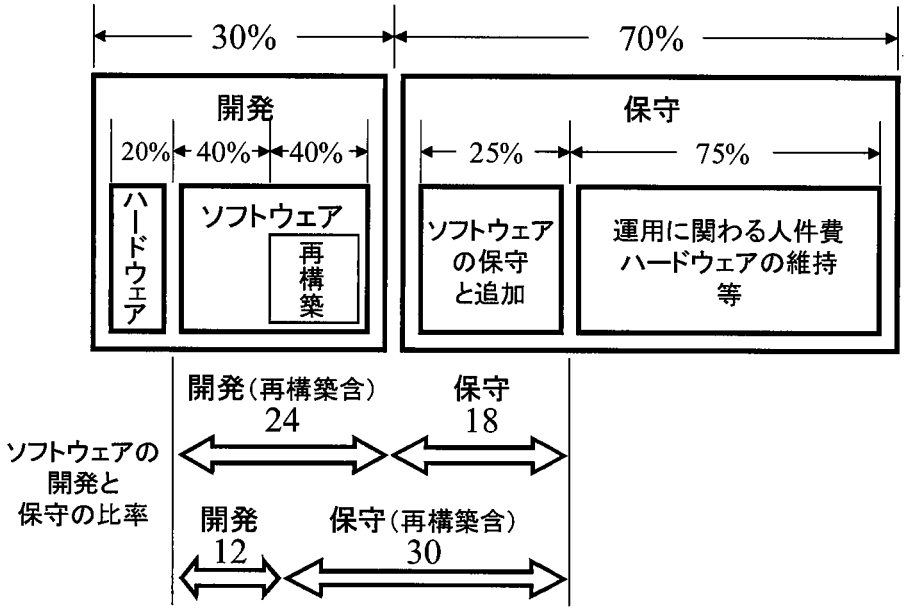


図 1.1 情報システムにおける開発と保守のコスト

(経済産業省[KZ07]および日本情報システム・ユーザ会[JU07]のアンケートに基づく)

保守の負荷が増大している背景としては，情報システムが広く行き渡ったことだけではなく，企業経営環境の変化が影響を及ぼしていると推察することができる．たとえば，インターネットや携帯機器に関係する業務では，新しい商品やサービスが計画されてから市場に投入されるまでに許される時間は短期化される傾向にある．そこで，再構築の時期を待つ猶予なく，発生した業務要件を直ちに実装することが多くなっていると考えられる．また企業の買収，統合，提携などにより業務組織が改編される頻度が高くなっている．そこで，新規開発は最小限に留めて，先ずは現行システムの保守によって業務の継続性を確保することが重視されていると考えられる．

このように，近年の情報システムにおいては，開発の負荷の軽減と並んで，あるいはそれ以上に，保守の負荷を軽減することが大きな課題となっている．

1.1.2 保守を基盤とする継続的開発

開発と保守は実際上同時に実施されることが多い．システムの仕様は開発の初期に設定されるが，そこから開発の完了までの期間にも新たな業務要件は発生する．そこで，一方で開発を実施しながら，もう一方で開発完了後の保守作

業を検討することになる．開発のなかでも再構築の場合には，現行システムのうちで活用できる部分の抽出や新たに開発する要素の統合方法を検討するため，作業の初期に現行システムを詳しく調査する．また逆に，保守の作業においては，しばしばまとまった単位でシステムへの機能追加が求められていく．

本論文では，こうした開発プロセスと保守プロセスが常に表裏一体で進行する開発・保守形態を**継続的開発**と名づけている．継続的開発の概要を図 1.2 に示す．

かつて情報システムの普及期においては，情報化されていない業務に対して新たにシステムを構築する，もしくは業務システムを全面的に構築し直すプロジェクトが主流であったため，情報システムの開発と保守のフェーズをはっきりと区別することが可能であった．しかし，昨今においては，前項の実態調査分析で述べたように，現行稼動しているシステムの保守を基盤として，次々と発生する業務案件に対して間断なく機能追加を実施していく傾向が強い．

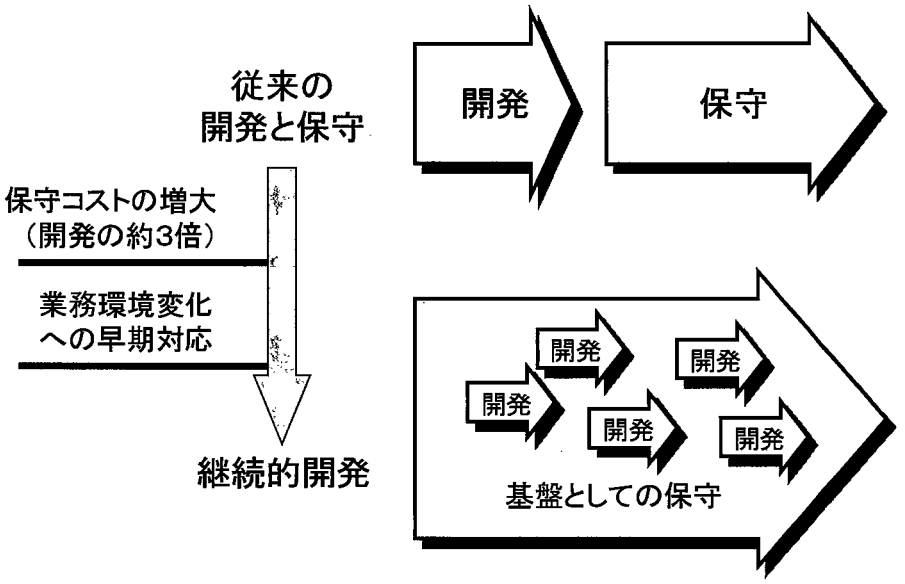


図 1.2 研究の背景：継続的開発の概要

1.2 継続的開発のためのフレームワーク

1.2.1 将来の保守に備えたソフトウェア構造としてのフレームワーク

従来における開発と保守のフェーズが区別されていた形態において，保守は

通常，開発時設計によるソフトウェア構造の上でそのまま行われていた．開発時設計によるソフトウェア構造とは，開発の初期に，その開発の期間内に実現する予定の機能群を想定して，それらの機能群の実現だけに焦点を絞って設計したソフトウェア構造である．開発時設計のソフトウェア構造に基づく保守方法は，あらゆる機能の保守に対応することができる．ただし，保守の担当者は開発時の設計モデルを理解する必要があり，保守担当者に技術的なスキルが求められることや，保守のコストを低く抑えにくいことが課題となっていた．

継続的開発の形態においては，頻繁に保守する可能性の高い機能のパターンを想定し，予めそれらの追加，修正に適したソフトウェア構造を整備しておくことが望ましい．この将来の保守に備えたソフトウェア構造を，本論文では**フレームワーク**と呼ぶことにする．フレームワーク上で保守は，想定した機能の範囲に限られるものの，専用の保守手順を用意して作業を効率化することができる．担当者は，開発時の設計モデルを理解しなくても継続的保守を担当することができる．

またフレームワークは，ソフトウェアの開発／保守の担当者だけではなく，利用者自身が業務への適用性や使い勝手を向上させる作業を進める上でも，基盤として整備しておくことが期待される．利用者における利用手順とそのなかで機能を追加する範囲を想定し，その利用や項目追加に適した利用者環境と専用手順をフレームワークとして実現し，利用者に提供する．こうした利用者の側での作業は，図 1.1 に示したソフトウェア開発／保守の体系に含まれるものではない．しかし，情報システムのライフサイクルのなかで，利用者の要求に沿って機能を追加・修正するという点では，開発／保守の担当者による継続的開発の作業と同様の位置付けを有している．

1.2.2 オブジェクト指向によるフレームワークの実現

一般に情報システムの機能には，商品や業務の動向に従って変化が激しい部分と，長期にわたって固定化している部分が存在する．フレームワークは，変化が激しい部分に着目し，これを固定化した部分と切り離して構成したものである．そして変化部分については，要素の組み合わせによって自在に機能を実現することを求めていく．

こうした独立性の高い要素の組み合わせによるソフトウェアの構成方法として，**オブジェクト指向**が注目される．オブジェクト指向 [GO83a] [GO83a] では，結合度の高いデータと手続きをカプセル化することを特徴の一つとしている．オブジェクト指向登場以前のアプローチ，たとえば手続き指向によるアプローチ [DI75] では，手続きは構造化されるが，手続きからのデータの参照が混

沌となる恐れがある．また一方データ指向によるアプローチ [CHE76] では，データは構造化されるが，データを参照する手続きが不統一となる恐れがある．オブジェクト指向においては，関連の深い手続きおよびデータをともにクラスという単位でカプセル化して定義していくため，独立性の高い要素を組み合わせてソフトウェアを構築していくことや，プログラムを再利用することに適している．

アプリケーションの基本構造をクラス群によって準備したものを，一般にアプリケーションフレームワークもしくは単にフレームワークと呼ぶ [BO94] ．実用化されたアプリケーションフレームワークとしては，MacOS のライブラリである MacAPP [AP91] や Windows のライブラリである Microsoft Foundation Class (MFC) [JEG03] など技術層において汎用性の高いもののほか，特定の設計方式を誘導するものがある．たとえば，Web アプリケーション構築用の Struts [CAV02] ，Eclipse [DA03]，本論文の第 3 章で評価対象として取り上げる画面遷移フレームワーク [YU02b] などである．

継続的開発のためのフレームワークは，これらオブジェクト指向によるアプリケーションフレームワークの技術をもとに実現されるものである．そして，対象とする情報システムの業務や利用者の特性に応じて，ソフトウェア構造と利用手順をより専用化したものと位置付けることができる．

オブジェクト指向技術をもとにした，継続的開発のためのフレームワークとそれを用いたシステム開発の概要を図 1.3 に示す．システムの固定化した部分は，クラスによるプログラムまたはそれによって包まれたソフトウェアとして提供される．変化する部分については，それを記述するためのクラス群とそれらの利用手順を提供し，継続的開発の担当者がそのクラス群を用いてオブジェクトの組み合わせを記述する．

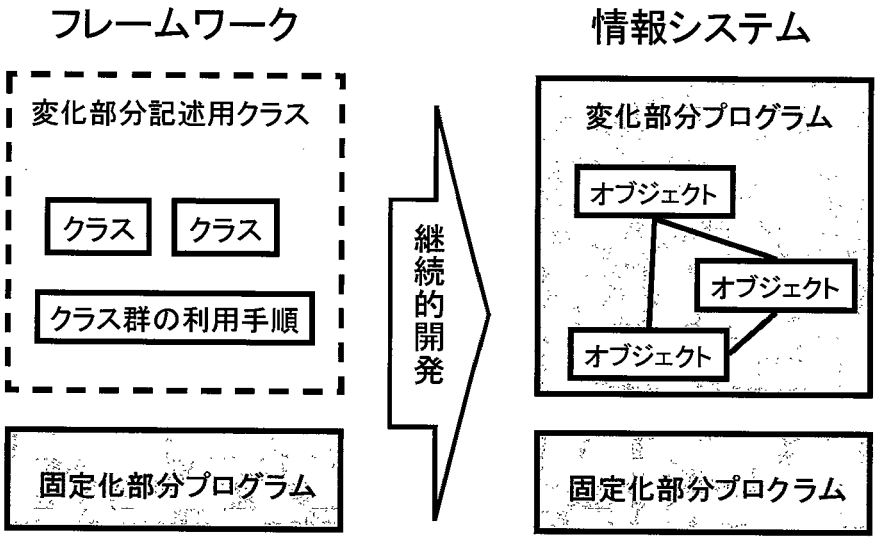


図 1.3 研究の目標：
フレームワークを用いた継続的開発の概要

1.3 フレームワークの技術課題

情報システムの継続的開発においてフレームワークを適用するうえでは，以下の技術課題が考えられる．関連技術との関係を含め，図 1.4 にフレームワークの技術課題と本論文の範囲を示す．

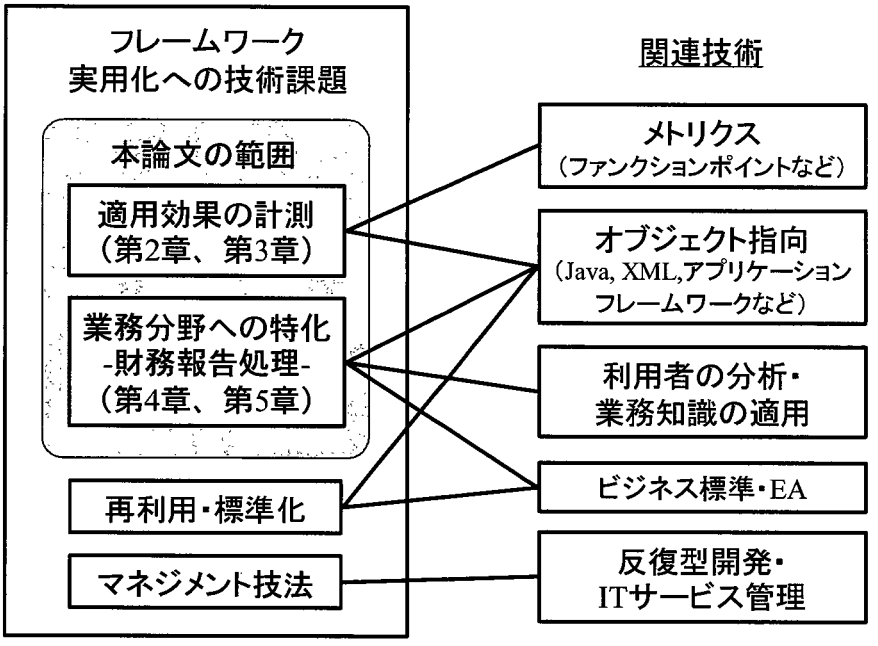


図 1.4 フレームワークの技術課題と関連技術

1.3.1 フレームワーク適用効果の明確化

オブジェクト指向による組み合わせプログラミングや再利用に関しては，ソフトウェアの生産性を向上させる技術として注目されているとはいえ，個々のシステム開発における効果については明確ではないことが多い．フレームワークという新しい開発技術を適用することにはリスクを伴うので，大規模な情報システム構築への導入を推進するためには，定量的な効果の測定と見積りの技術が必要となる．

情報システム開発の生産性評価に関しては，ファンクションポイント [AL79] が広く用いられている．ファンクションポイントは，ソフトウェアで扱うデータ・ファイルとデータ処理トランザクションに着目した尺度であり，プログラミング言語や開発環境に依存しない開発工数の評価に適している．しかし，オブジェクト指向ではデータ・ファイルやトランザクションの定義がプログラミングの中心概念ではなく，クラス定義が中心となる．そこで，クラス定義に対してファンクションポイントの計測方法を応用することが，フレームワーク適用の定量的評価の基盤となる．

クラス定義へのファンクションポイント計測方法としては OOFP [CA98] [AN99] [AN03] が提案されている．これを用いてフレームワーク適用の効果を測定することが期待される．また，OOFP ではクラス定義に含まれる情報の特性

に従って，データ・ファイルやトランザクションの計数方法を一部簡略化している．実際の仕様書ではクラス定義のほかにもメソッドに関する定義などが含まれていることがあるので，それらを利用してより精度の高い計測値を得る方法も期待されるところである．

ソフトウェアの生産性を述べるうえでは，ファンクションポイントによる機能規模の評価とともに，複雑性の評価が必要である．オブジェクト指向に基づくソフトウェア要素の組み合わせや再利用は，ソフトウェアの規模を圧縮することは間違いないが，その反面一つの要素を多様に用いることからソフトウェアの構造を複雑にする恐れもある．

オブジェクト指向プログラムの複雑性の評価に関しては，C&K メトリクス [CHI94] が知られている．C&K メトリクスでは，クラスに対して，内部構造，継承および結合／凝集欠如の 3 点から評価する．内部構造では，メソッドの数や内部に含まれるメッセージ送信の数を計測する．継承では，子クラスの数や継承の深さを計測する．結合（coupling）とは他クラスの参照（メッセージ送信や他クラスのインスタンス変数参照）が多いことを表し，凝集（cohesion）の欠如とは自クラスへの参照（同一クラスのメソッド対における同一インスタンス変数の参照）が少ないことを表す．再利用性の高いクラスを定義しようとすると，他のクラスとの関係を多数設定するために，これらの値は高くなる傾向がある．しかし，クラスの構造を単純化し，他のクラスからの独立性を維持するためにはある程度の値以下に抑えることが望ましい．

C&K メトリクスの適用に関しては，オープンソースソフトウェアへの適用 [CHI98] [FE04] [BL06] が報告されているほか，NASA における開発の経験から推奨値が提案されている [RO99]．また，C&K メトリクスと発見バグ数との関係 [BRI98] や他のメトリクスに比べた場合の予測精度の良さ [BA96] も報告されている．そこで，C&K メトリクスを用いてフレームワーク適用を評価することが待たれる．フレームワークは，通常のクラスライブラリとは異なり，まずクラスの利用方法を設定し，その用途の範囲でクラスを定義していく．そこで，ひたすらクラスの再利用性を高めるために複雑性を著しく高めてしまうようなことは避けられているはずであり，C&K メトリクスに関しても良好な値が計測されることが期待される．

1.3.2 業務分野に特化したフレームワークの実現

継続的開発のためのフレームワークは，情報システムが属する業務分野に特化することにより，より高い生産性を期待することができる．

インターネットを用いた情報システムにおいては，XML [WW96] をもとにし

たビジネスデータ交換が盛んとなっており，業務分野ごとにビジネスデータ記述言語が標準化されている．このようなXMLベースのビジネスデータへのアクセスを含むシステムに対しては，日々発生する業務要件が直接影響を与えるので，情報システムの開発・保守担当の部署だけではなく，業務担当の部署において継続的開発が実施される場合が多い．そこで，フレームワークの整備にあたっては，変化部分の機能規模や複雑性を改良するだけではなく，情報技術に関する知識の少ない業務担当者が理解しやすい構造を提供することが必要となる．

たとえば，財務報告データの標準化言語であるXBRL (eXtensible Business Reporting Language) [XB00] では，XML のほかに，XML の拡張仕様であるXML Schema [WW01a] およびXLink [WW01b] を導入しており，XMLデータのプログラミング・インタフェース言語であるDOM [WW97] も含めこれらの最新技術を習得しなければ，継続的開発を担当することができない．そこで XBRLデータのアクセスのためには，これらの技術要素を隠蔽し，変化部分に関しては財務的な意味の構造のみを記述すれば済むようなフレームワークが期待される．

ビジネスデータの参照と利用については，専門の業務担当者ばかりでなく，一般の利用者の環境においても適宜対話の方法を変更していく必要が生じる．一般利用者においては，技術知識だけではなく，業務知識に関しても多くは有していないと考えられる．

たとえば，上述のXBRL財務データは会計業務担当者や機関投資家などの専門家だけではなく，一般のビジネスマンや個人投資家においても利用されるデータである．ただし，一般利用者においては「流動資産」や「売上高利益率」など個々のデータ項目や指標の意味を知ってはいても，それらの値から企業の特性を考慮して投資や融資の判断を導く分析手順に関しては，知識を有しないことが多い．そこで，一般利用者の利用環境に対するフレームワークにおいては，理解されやすい形で財務データがアクセスされるだけではなく，専門家が持つ財務報告データの分析手順に関する知識を利用する機能が期待されることになる．

1.3.3 その他の課題

このほか，フレームワークを用いて継続的開発を実用化するうえでは，以下の課題が関連する．これらに関しては，本論文の次の章以降では言及しないが，すでに多くの研究開発と実務適用が進められている．

その課題の一つは，フレームワークの固定部分や変化部分に用いるソフトウ

ェア要素に関する標準化である．標準化によって，フレームワークに用いるソフトウェア要素の再利用と，フレームワークを用いて構築された情報システムの相互運用性の向上が推進される．前項で述べた XBRL は財務報告の領域における標準化であるが，このほか，たとえば人事情報の HR-XML（Human Resource XML）[HR01]，ニュース情報の NewsXML [IT03]，法律情報の LegalXML [LE98]，金融商品情報の FpML（Financial Products Markup Language）[FP99] など，業務分野ごとに数多くの標準化が推進されている．

また，業務分野を横断し企業全体としてソフトウェアの構造，要素やそれらの開発方法を標準化する活動として，EA（Enterprise Architecture，エンタープライズ・アーキテクチャ）[YU04] が注目される．EA は当初，公共機関を中心に IT 投資の透明性向上を主な目的として推進されたが，現在は民間企業にも普及し，特に昨今は IT 内部統制や個人情報保護などに関する IT ガバナンスの一環として，標準化およびその前提としての可視化が進められている [YU06a]．

もう一つ重要な課題として，継続的開発のマネジメント方法があげられる．保守のプロセスに関しては，従来のソフトウェア開発の体系，たとえば SLCP（Software Life Cycle Process, ISO/IEC12207, JIS X0160）[ISO02] [JIS07a] [IP07] においても保守プロセスが取り上げられており，さらにソフトウェア保守規格（ISO/IEC/IEEE14764, JIS X0161）[ISO06] [JIS07b] [SO07] において該当部分が定義されている．継続的開発においては，これに加えて，変化部分における追加や変更をより迅速に行うこと，および開発ではなく運用と保守を中心に全体の手順を組み立てることの 2 点で，従来とは異なる視点からの体系を取り入れていく必要がある．

変化部分の追加・変更に関しては，開発とその結果の評価を短いサイクルで繰り返していく反復型開発手法 [KR99] や，開発チームにエンドユーザを含めた体制でさらにサイクルを短期化するアジャイル型開発手法 [BE00] が注目される．反復型開発手法では初期のサイクルで開発対象のアーキテクチャを確定していくことから，本論文のフレームワークをもとにしたシステム構築法と整合性を持つ．これに対し，アジャイル型開発手法では最後まで顧客要求を受け止める（要求を抱きしめる）ことを主旨とするので，本論文とはいささか立場を異にする．

運用・保守を中心とする体系に関しては，ソフトウェアの変更管理を含む情報サービス・マネジメントの国際標準化体系である ITIL（IT Infrastructure Library）[OG00] [OG01] が注目される．ITIL では，システムの提供者と利用者の間で，システムの稼動や保守サービスの品質をあらかじめサービスレベルとして定義して契約し，システム運用を管理していく．保守サービスの品質を維持するためには，保守に適したソフトウェア構造と保守手順を用意する必要が

あり，フレームワークに基づく継続的開発に適した運用方法と言える．

1.4 本論文の概要と構成

本章に続く第 2 章では，1.3 で述べた課題のうち，1.3.1 に示した適用効果の明確化の基盤となるオブジェクト指向プログラムにおけるファンクションポイントの計測法について述べる．特に，オブジェクト指向型の要求仕様定義ツールである REQUARIO [SA95] [YU02a] を用いて記述された仕様書から計測する手段を明確にし，計測ツールを開発した．REQUARIO はかつて報告者が日立製作所において研究開発と適用推進にあたったツールであり，この章の研究は，報告者がツールの有効性検証という観点でソフトウェア生産性に関わりを持つ端緒となった研究である．

REQUARIO では，クラス定義に加えて，メソッドの動的特性を定義したイベントトレース図を作成するので，その情報を用いて精度の高いファンクションポイントを計測することができる．そのことを，熟練者による計測結果との比較によって分析した．

第 3 章では，1.3.1 で述べた適用効果の明確化の課題について，アプリケーションフレームワークを用いて開発したシステムに OOFP [CA98] および C&K メトリクス [CHI94] を適用し，開発規模と複雑性に関する分析結果を示した．ここで分析の題材とした画面遷移フレームワーク [YU02a] は，これも報告者が日立製作所において開発と普及に関わったものであり，この章の研究はその効果の検証という観点を含めて開始された．

アプリケーションフレームワークの利用に関しては，2 種類のケーススタディを用いて，利用した場合と利用しない場合とを比較した．つまり，小規模の機能を個々に開発するケースと，それらを連続して追加していくケースであり，両者の比較によりフレームワークが継続的開発に適することを計量化しようと試みた．

第 4 章では，1.3.2 で述べた業務領域に特化したフレームワークの一例として，インターネット上でのデータ交換のための財務情報記述言語である XBRL [XB00] を題材に取り上げ，必要な機能とその実現法を述べている．報告者は，XBRL 標準化の国内組織である XBRL Japan の副会長を務めた経緯があり，この章および次の章の研究は，XBRL 普及のための実践的な技術開発という位置付けも有する．

フレームワークとしては，XBRL で記述されたデータを会計業務担当者が扱うことを主な目的として，専用のデータモデルである SDX (Simple Data Model

for XBRL Documents）を開発した．SDXでは，XBRLに取り入れられているXML関連の新技術であるXML Schema [WW01a]，XLink [WW01b] およびDOM [WW97] の要素を隠蔽しており，それによって技術知識に乏しい会計業務担当者においても理解しやすいモデルとした．またSDXに基づいて，プログラミングに用いるためのデータバインディング・ライブラリと業務担当者向けの専用言語を開発した．これによって，財務データへのアクセスを記述する場合の生産性を，会計業務担当者が財務データを扱う場面を想定したシナリオに基づいて評価し，SDXの実用性を検証した．

第5章では，これも業務領域に特化したフレームワークの一例を構築した．この例では，第4章と同じくXBRLによって記述された財務データを扱うシステムを取り上げるが，今度は利用者の環境における機能拡張に対処するためのフレームワークをFIAT（A Financial Analysis Assistance Tool）と命名した対話型ツールとして実現した．財務報告データの利用者には，個人投資家など一般利用者が含まれる．一般利用者においては，個々の財務報告のデータ項目は知っていても，それらをいかに分析して融資や取引の判断を行なうかについては知識を有していないことが多い．そこで，こうした分析手順に関する知識を専門家が入力し，それを利用者環境において参照して財務分析を実施できるようにした．なお，財務分析知識の形式化に当たっては，会計学の研究者および実務に当たる公認会計士の協力を得て実施した．

第6章では，第2章から第5章での研究成果をもとに，フレームワークを用いた継続的開発の実用化に関するまとめを述べる．また，今後の研究方針に関して述べる．

第2章　イベントトレースに基づく要求仕様書からのファンクションポイント計測方法

2.1 まえがき

近年，コンピュータシステムへの依存度が高まるにつれて，その主要な構成要素であるソフトウェアはますます大規模化，複雑化，多様化してきている．さらに，ソフトウェアの需要も増大し，開発期間の短縮が求められるようになってきた．このような背景から，ソフトウェアを効率よく開発するために，明確な開発計画の下で開発プロセスの全工程を系統付けて管理する必要性が高まってきている．

明確な開発計画を立てるうえで，開発中に起こる様々な事象を予測し，あらかじめ必要な手段を講じておくことが重要である．ソフトウェア開発に関して予測の対象とすべきものに規模，投入する工数，開発期間，開発に使用される技術，品質などがある．なかでも重要なのは開発工数と開発期間である．

開発工数と開発期間を予測するには，通常，開発対象ソフトウェアの規模を見積もりに基づいて行われる．従来，ソフトウェアの規模を表すのにプログラム行数（SLOC）が使われていた．最近では，ソフトウェアの機能要件だけを抽出して定量的に計測するファンクションポイント法が世界的に普及しつつある．ファンクションポイント法は，Albrecht によって 1979 年に標準測定法が提案されたものである [AL79].

しかし，標準測定法ではファンクションポイントを計測するための一般的なルールが述べられているにすぎず，詳細な部分の計測には測定者の判断が必要になる．結果として，同一プロダクトに対してのファンクションポイントの計測であっても，計測する人間によって誤差が生じてしまうという問題点が指摘されている [LOW90]．たとえば，同じ組織内の人間が同じプロダクトに対して測定した場合には 12%，違う組織の人間が測定した場合は 30%以上の誤差が出るという報告もされている [KI97]．また，ファンクションポイントを実際に応用して見積もりを行うためには，そのソフトウェア開発組織で過去に開発され

たソフトウェアのファンクションポイント値が基礎データとして必要であり，その計測のためのコストが現場への導入の妨げになっている．したがって，新規開発のソフトウェアだけでなく，既存ソフトウェアに対するファンクションポイント計測を効率よく行うことが重要な課題となっている．

本章では，これらの問題点に対処するため一つの方法として，イベントトレース図 [RU91] に対してファンクションポイントを計測するための指針を提案する．

イベントトレース図はオブジェクト指向設計における標準的な設計仕様書の一つである．ファンクションポイントをオブジェクト指向ソフトウェアに適用する場合には，図 2.1 に示す通り，オブジェクト指向のクラスをデータ・ファイルに，メソッドをトランザクションに対応付ける考え方が一般的である．

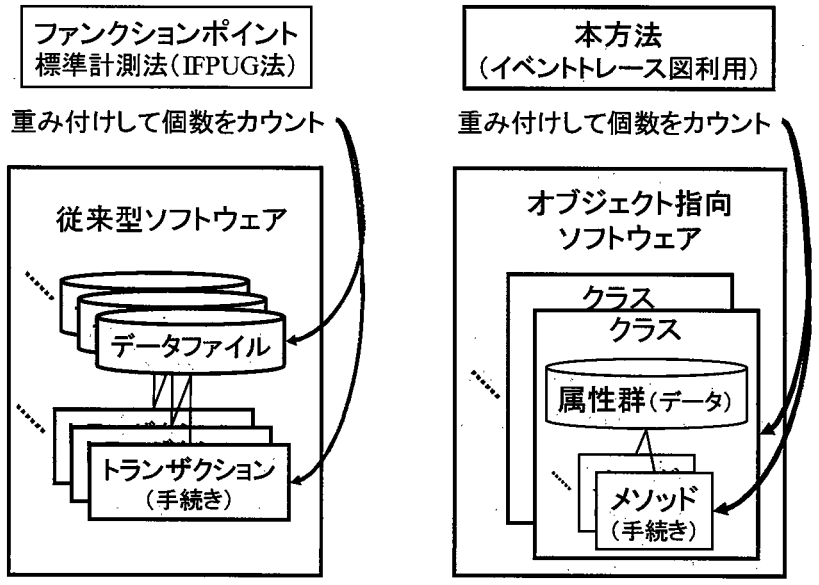


図 2.1 オブジェクト指向ソフトウェアへの
ファンクションポイントの適用の考え方

さらに，提案した方法の有効性を評価するために，CASE ツールで作成されたイベントトレース図に基づく要求仕様書に対して，ファンクションポイントを計測するためのツールの試作を行った．図 2.2 に示すとおり，ここで用いた CASE ツールの REQUARIO では，メソッドの動作の定義にデータの参照や更新の動作を関係付けて記述することができる．そこで，このデータに関する動作仕様記述をイベントトレース図の拡張と位置付け，ファンクションポイント計測の精緻化に利用した．

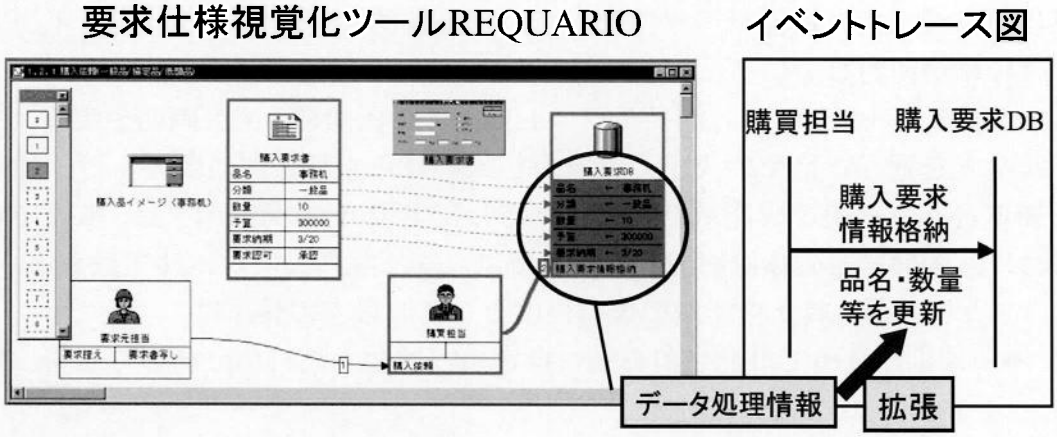


図 2.2 拡張されたイベントトレース図情報に基づく ファンクションポイント計測の精緻化

最後に，実際の要求仕様書に対してツールの計測したファンクションポイント値とファンクションポイント計測の熟練者が測定したファンクションポイント値の比較を行った．その結果，仕様書に記述されている機能については，熟練者が計測した値とツールが計測した値が一致した．

以降，2.2 節ではファンクションポイント法の概要を紹介する．2.3 節ではイベントトレース図に対するファンクションポイント計測ルールについて提案する．2.4 節では，提案したルールを評価するためのケーススタディについて述べ，最後に 2.5 節ではまとめと今後の課題について述べる．

2.2 ファンクションポイント法

2.2.1 概要

ファンクションポイント法は，Albrecht によって 1979 年に提案された [AL79]．その後，これをベースに様々な改良方法が提案されており，現在では，数十種類の計測方法がある．たとえば，IBM 法 [NI94]，IFPUG (International Function Point Users Group) 法 [IF94]，Function Points 法 [NA96]，Feature Points 法 [NA96]，MarkII 法 [SY91] などである．現在では IFPUG 法が主流技法となっている．IFPUG 法は，Albrecht 版の使い勝手やあいまいな部分を改良したバージョンであり，欧米で広く使用されている．

ファンクションポイント法は，利用者要求のうちの機能要求仕様の大きさを定量的に計測する方法であり，求められる計測値は，機能量または機能規模と呼ばれる．また，機能量の単位としては，伝統的にファンクションポイントという呼称が使われている．

機能量の計測では，計測対象ソフトウェアの機能のうち，画面や帳票，ファイルなどを通じた情報の入出力に着目し，それらを種類別に数え上げ，種類数を加重合計した値を機能量とする．ただし，実現方法の違いによる影響を除くために，画面などの数は物理設計結果ではなく，論理設計レベルで数える．

こうして得られた機能量の規模尺度としての最大の長所は，

- 規則に従って計測される値（推定値ではなくだれが計測しても同じ値が得られる）
- 機能仕様にだけ依存（開発環境へ開発言語などの技術要件に左右されない）

という 2 点である [IF94].

これに対して，従来からソフトウェアの規模尺度として使われてきた SLOC 尺度は，開発言語や開発環境が変わると値が変化する，プログラムが完成するまでは値が確定しない，などの点で機能量尺度より劣っている．

本章では，数多くのファンクションポイント法のなかでも主流技法となっている IFPUG 法を基にして，ファンクションポイントを計測する．

2.2.2 IFPUG 法

IFPUG 法は，Albrecht 版のファンクションポイント法に対して複雑さの評価の客観化やルールの精密化・適正化などの変更を行ったバージョンである．

IFPUG 法は次の Step1～Step7 でファンクションポイントを計測する[IF94].

Step1 （算出種類の選択）：算出種類を，アプリケーション FP（アプリケーションソフトウェアの大きさを表すファンクションポイント），新規開発プロジェクト FP（新規のアプリケーションを開発するプロジェクトの規模を知るために使用するファンクションポイント），機能改良プロジェクト FP（すでに存在するアプリケーションを拡張するプロジェクトの規模を知るために使用するファンクションポイント）の 3 種類のなかから選択する．

Step2 （計測境界の設定）：ファンクションポイントを計測する対象（範囲）を明確にするため，計測境界を設定する．通常は 1 つのアプリケーションを範囲とするが，そのアプリケーションが非常に大きい場合には，計測単位をサブシステムなどの機能単位に分割する．

Step3 （データファンクションの計測）：データファンクションとは，アプリケ

ーション中にあり，ユーザが認識できる論理的な意味でのデータのまとまりのことである．アプリケーションの中からデータファンクションを抽出し，ファンクションタイプを決定し，複雑さを測定する．

データファンクションには以下の 2 種類のファンクションタイプがある．

- 内部論理ファイル（Internal Logical File）：計測対象のアプリケーション内でデータが更新される論理的な関連をもったデータの集合．
- 外部インタフェースファイル（External Interface File）：計測対象のアプリケーションによって参照されるデータの集合（データは更新されない）．

次に，それぞれのデータファンクションをレコード種類数，データ項目数という 2 つのパラメータによって，低・中・高の 3 段階に重み付けする．

Step4 （トランザクションファンクションの計測）：アプリケーションに対するデータの出入りを伴う処理をトランザクションファンクションという．

アプリケーションの中からトランザクションファンクションを抽出し，ファンクションタイプを与え，複雑さを測定する．

トランザクションファンクションには以下の 3 種類のファンクションタイプがある．

- 外部入力（External Input）：計測境界外からデータ入力によってデータファンクションの更新を行う処理．
- 外部出力（External Output）：（オンライン処理における）計測境界外へのデータ出力を含む処理のうち，出力データに派生データ（計算や条件判断など何らかの加工を必要とするデータ項目）を含むもの．
- 外部照会（External Inquiry）：（オンライン処理における）計測境界外へのデータ出力を含む処理で，出力データに派生データを含まないもの．また，処理は内部論理ファイル（ILF）を更新しないもの．

次に，それぞれのトランザクションファンクションを関連ファイル数，データ項目数という 2 つのパラメータによって，低・中・高の 3 段階に重み付けする．

Step5 （未調整ファンクションポイント算出）：Step3，Step4 の結果を基に，種類別重み別に個数を数え，表 2.1 を用いて算出した結果を複雑さの評価値を加え併せる．この合計値が未調整ファンクションポイント（Unadjusted Function Points）となる．

Step6 （調整係数の算出）：未調整ファンクションポイントは，「データのまとまり」と「データの出入り」のみに着目した値で，性能，信頼性，ユーザインタフェースなどは考慮していない．そこでシステム特性をファンクションポイントに反映させるために，表 2.2 に示すシステム特性の 14 項目を 6 段

階で評価し，その結果から調整係数を算出する．

調整係数は，ファンクションポイント算出の際に未調整ファンクションポイントを補正する役割がある．

Step7 （最終ファンクションポイント算出）：未調整ファンクションポイントと調整係数を用いて，最終ファンクションポイントを算出する．

表 2.1 未調整ファンクションポイント算出表

	低	中	高	合計
ILF	□×7=□	□×10=□	□×15=□	
EIF	□×5=□	□×7=□	□×10=□	
EI	□×3=□	□×4=□	□×6=□	
EO	□×4=□	□×5=□	□×7=□	
EQ	□×3=□	□×4=□	□×6=□	
未調整ファンクションポイント				

表 2.2 システム特性の 14 項目

1	データ通信機能
2	分散データ処理
3	性能要件
4	高負荷構成
5	トランザクション率
6	オンラインデータ入力
7	エンドユーザの効率
8	オンライン更新
9	複雑な処理
10	再利用性
11	インストレーションの容易さ
12	運用の容易さ
13	複数サイト
14	変更の容易さ

調整係数 ＝ 0.01×影響度の合計 ＋ 0.65

2.2.3 ファンクションポイント計測ツール

これまでに多くのファンクションポイント計測ツールが開発され，市販されている☆.

たとえば，構造化設計を支援するためのCASEツールであるSAVERには，ファンクションポイント計測機能が実装されている [VE97]．これはデータフロー図上のプロセスをトランザクションファンクションに，ファイルをデータファンクションに対応させてファンクションポイントを計測している．しかし，ファンクションのタイプや複雑度をユーザがすべて指定しなければならない．同様に，ABT CorporationのFunction Point ManagerもIFPUG法に基づいて，ファンクションポイントを計測するが，計算の部分が自動化されているだけであり，トランザクションファンクション，データファンクション，複雑度などの基本データはすべてユーザが入力する必要がある．

オラクル社のDesigner/2000やVIASOFTのVisual Recap [VI97] にも，ファンクションポイント計測機能が存在する．これらは自動的に計算を行うが，たとえば，データファンクションにおける内部論理ファイルと外部インタフェースファイルの区別をせず，すべて内部論理ファイルとしてみてしまう．また，通常データファンクションとみなさない一時ファイルも内部論理ファイルとして計測してしまうため，実際の値に比べて数十倍もしくは数十倍以上の差が出る可能性もある．

2.3 イベントトレース図に対するファンクションポイント計測

2.3.1 イベントトレース図

通常，情報システム構築の上流工程では，システム分析者はユーザに文章，表，図などで作成した仕様書を提示し，ユーザはその仕様書でシステム構築の提案内容を確認している．その際，システムの静的な情報については，文章や図として表現することは可能であるが，動的な情報を記述することが難しい．

動的な情報を表現するための手段として，イベントトレース図 [RU91]（シ

☆ ここで述べたもの以外にも，コンサルティング会社，各種メーカーでは自社の計測ツールを所有していると言われている．しかし，情報が公開されていないため，その詳細は不明である．

ーケンス図）が広く用いられている．イベントトレース図は，システムに存在するオブジェクトの間におけるメッセージ送受信をもとにして，それから発生する事象の推移を定義した図である．図 2.3 にイベントトレース図の例を示す．矢印を用いてオブジェクト間のメッセージ送受信の様子を定義する．縦軸が時間である．

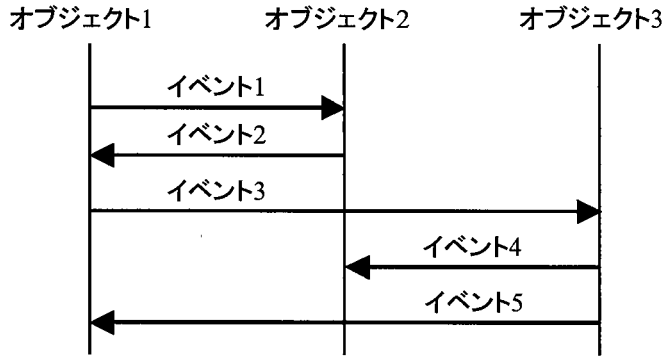


図 2.3 イベントトレース図の例

2.3.2 ファンクション識別方法

2.2 節で述べたように，ファンクションポイント計測で必要な情報は，計測境界を通じて，ユーザ，あるいは，外部アプリケーションとの間で行う処理と扱われるデータの情報である．イベントトレース図では，オブジェクトとして，計測対象アプリケーション内のデータベースやコンポーネント，あるいはユーザや外部アプリケーションが記述される．また，実行される処理もメッセージ送受信という形で記述される．したがって，データファンクションの候補にはイベントトレース図上の人間以外のオブジェクトを，トランザクションファンクションの候補にはメッセージのやり取りを行っている 2 つのオブジェクト対を，それぞれ選定すればよい．

次に，データファンクションとトランザクションファンクションのタイプ分けについて考える．トランザクションファンクションについては，メッセージのやり取りの前後で送信側のオブジェクトが受信側のオブジェクトの持つデー

タが変更されたかどうかを調べることで，外部入力，外部出力，外部照会の場合分けが可能である．具体的には，送信側オブジェクト A から受信側オブジェクト B へメッセージが送られる場合，オブジェクトがデータファンクションである場合とない場合で以下の 3 通りに分類し，以下に示す R1～R8 のルールを用いてタイプ分けを行う．

(1) オブジェクト A がデータファンクションの候補でなく，オブジェクト B がデータファンクションの候補の場合
R1：B のデータが変更されていれば外部入力．
R2：B のデータが変更されていなければ外部照会．
(2) オブジェクト A がデータファンクションの候補で，オブジェクト B がデータファンクションの候補でない場合．
R3：A の出力データが派生データであれば外部出力．
R4：A の出力データが派生データでなければ外部照会．
(3) オブジェクト A とオブジェクト B の両方がデータファンクションの候補の場合
R5：B のデータが変更され，A の出力データが派生データであれば，外部入力と外部出力．
R6：B のデータが変更され，A の出力データが派生データでなければ，外部入力と外部照会．
R7：B のデータが変更されず，A の出力データが派生データであれば，外部照会と外部出力．
R8：B のデータが変更されず，A の出力データが派生データでなければ，1 つの外部照会．

ただし，外部照会と判定されたオブジェクト対については，それと対になるオブジェクト対を調べ，複雑度の比較を行い，複雑度の高い方のみを外部照会とみなす必要がある（R8 ではその処理を同時に行う）．

一方，データファンクションについては，すべてのトランザクションファンクションの場合分けを行った後で，データファンクション候補のオブジェクトの中で，データが一度でも変更されているものを内部論理ファイル，一度も変更されていないものを外部インタフェースファイルと判定すればよい．

2.4 ケーススタディ

2.3.2 項で述べた手法の適用可能性を確認するために，ある CASE ツールで

作成されたイベントトレース図に基づく要求仕様書に対して実装を行う．具体的には，報告者が開発に携わった要求定義支援システム REQUARIO [SA95] [YU02a] で作成されたイベントトレース図（REQUARIO ではシナリオ図）を対象とする．

2.4.1 対象となる CASE ツール

REQUARIO （Requirement Scenario Designer）[SA95] [YU02a] は，開発するシステムへの要求仕様をアニメーションで表示するツールである．

REQUARIO では，ストーリー，シナリオ，シーン，キャラクタという概念で要求仕様を定義する．これらの概念構成を図 2.4 に示す．このなかのシナリオがイベントトレース図に対応し，さらにキャラクタがオブジェクトに対応する．

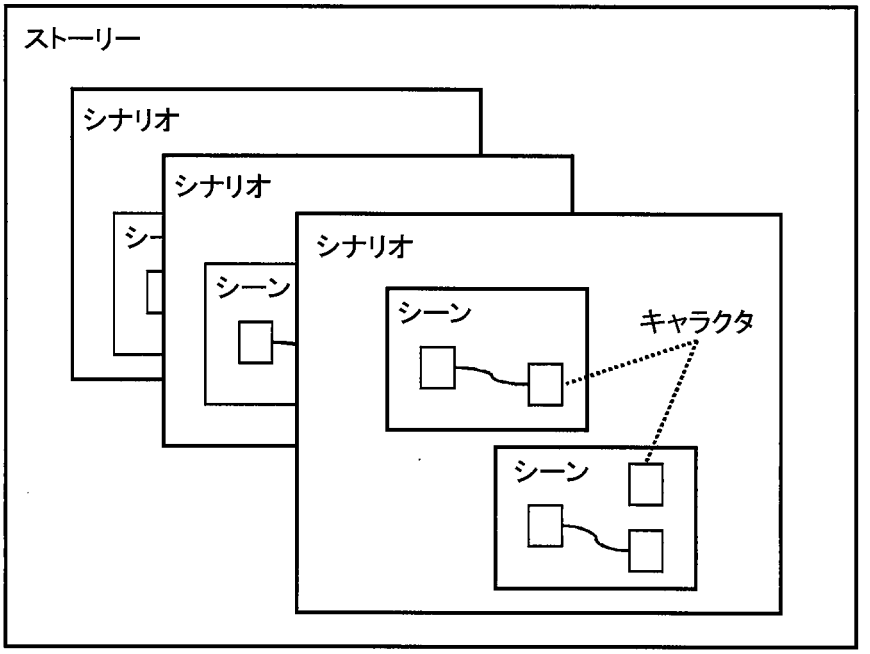


図 2.4　REQUARIO の概念構成

(1) ストーリー：要求仕様の断片的な具体例の集合である．REQUARIO の成果物はストーリー単位で管理されている．

(2) シナリオ：ストーリー中の「要求仕様の断片的な具体例」をシナリオという．シナリオは，ストーリーに登場する「ひと」あるいは「もの」と，その「ふるまい」（動作・処理）の流れから構成されている．つまり，「誰が（どこが)」，「誰に（どこに)」，「何を（どのようなイベントを)」発行するかを示したものである．

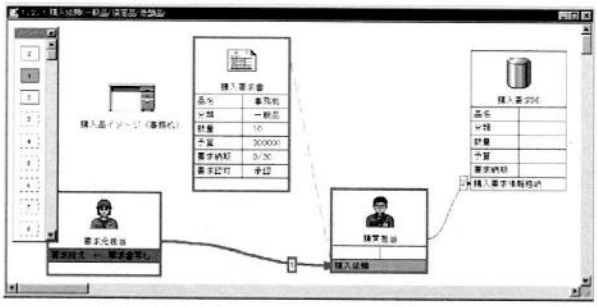
(3) シーン：ストーリーに登場する「ひと」あるいは「もの」をキャラクタといい，その「ふるまい」（動作・処理）の流れの一コマをシーンという．REQUARIO は，最小単位であるシーンとその連鎖を定義していくことでシナリオを作っていく．シーンには，次のような表現を加えることができる．

- キャラクタが持っているデータが，自分自身または他のキャラクタのデータ代入によって変化する様子．
- ふるまいに関係するキャラクタが，シーンに関与する様子（関与線と呼ぶ)．
- メッセージ呼び出しが発生する条件（起動条件と呼ぶ）を設定することで，実際の要求仕様のなかに出てくる「ある条件のときに，こういう処理を行う」というようなシナリオを表現することが可能となる．

(4) キャラクタ：ストーリーに登場する「ひと」あるいは「もの」をキャラクタという．REQUARIO ではシステムに登場する人，データの集まりなどの動作の主体となるキャラクタを“人”，“データベース”などのアイコンで表示する．キャラクタ同士のふるまいがシーンを構成していくことでシナリオとなり，ストーリーとなる．キャラクタは 2 種類の情報を持っている．1 つは，キャラクタの持つデータを表現した“属性”と，もう 1 つは，キャラクタの持つふるまいを表現した“メッセージ”である．

REQUARIO での要求仕様記述例を図 2.5 に示す．この例は購入業務というストーリーのなかの，購入依頼シナリオの一つを記述したものである．さらにそのなかで，図 2.5(a) は要求元から購買担当へ購入依頼を行うシーンであり，図 2.5(b) はそれに続く購入要求情報格納のシーンである．(a)購入依頼シーンにおいては，購入依頼書が用いられることが示されており，購入依頼書のデータ項目としては，品名，分類，数量，予算，要求納期，要求認可の各項目が示されている．(b)購入要求情報格納シーンにおいては，購入依頼書を参照して購入要求 DB への出力が行われていることが示されており，特に品名，分類，数量，予算，要求納期の各項目の値が更新されることが例示されている．

(a)
購入依頼
シーン



(b)
購入要求
情報格納
シーン

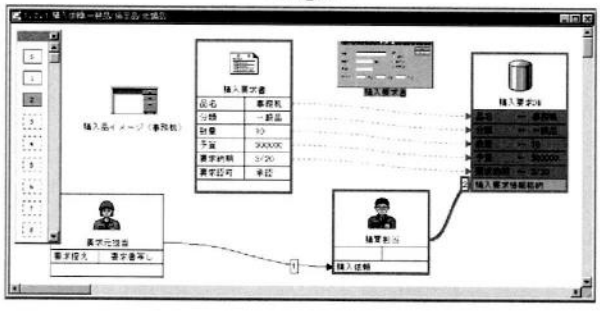


図 2.5 REQUARIO によるシーン記述の例

2.4.2 ファンクションポイント計測ツール

2.4.2.1 計測概要

ここではツールで実装するファンクションポイント計測の概要について説明する．2.4.1 項で述べたように，REQUARIO の記述はイベントトレース図よりも多くの情報を含んでいるので，Step3 と Step4 のデータファンクションとトランザクションファンクションの抽出については，2.2.2 項で述べたルールを拡張する必要がある．

Step1 （算出種類の選択）：算出種類はアプリケーションファンクションポイントに限定する．

Step2 （計測境界の設定）：算出種類をアプリケーションファンクションポイントに限定しているため，Step1 の入力として与えられた仕様書上で，ユーザとアプリケーションの境界が計測境界となる．

Step3 （データファンクションの計測）：指定したストーリーファイル中のデー

タファンクションを，2.4.2.2 で説明するルールに従って抽出する．

Step4 （トランザクションファンクションの計測）：指定したストーリーファイル中のトランザクションファンクションを，2.4.2.3 で説明するルールに従って抽出する．

Step5 （未調整ファンクションポイント算出）：Step3，Step4 の結果を基に，ファンクションタイプ別複雑さ別に個数をカウントし，それぞれに対して重み付けをし，加え合わせて未調整ファンクションポイントを算出する．

Step6 （調整係数の算出）：システム特性の影響度をユーザが入力し，調整係数を算出する．

Step7 （最終ファンクションポイント算出）：未調整ファンクションポイントと調整係数から最終ファンクションポイントを算出する．

2.4.2.2 データファンクション抽出ルールの調整

2.3.2 項で述べた識別方法に基づいて，データファンクションの抽出を以下のように行う．

(1) データベースのアイコンで表示されているキャラクタをデータファンクションとする．

(2) (1)で抽出されたデータファンクションのなかで，データが更新されるものを内部論理ファイル（ILF），更新されないものを外部インタフェースファイル（EIF）とする．

(3) レコード種類数（RET）は 1 種類☆，データ項目数（DET）はキャラクタの属性の数とし，予め設定した複雑さ決定表を用いて各データファンクションの複雑さを判定する．

なお，ツールの実装としては，ユーザは計測対象のシステムに出現するすべてのキャラクタに対して，データファンクションであるか否かの指定を行うこともできるようにする．

2.4.2.3 トランザクションファンクション抽出ルールの調整

2.3.2 項で述べたルールに基づいてトランザクションファンクション抽出ルールを図 2.6-1～図 2.6-3 のように拡張する．ルール 3 とルール 4 を，イベントトレース図に関与線という情報が付与されたことにより追加した．

☆ REQUARIO では複数のデータ項目をサブセットとして定義することができないため 1 となる．ただし，レコード種類数の値が確定している場合には，2.4.2.5 副項で述べるツールの機能を用いて設定することで計測は可能である．

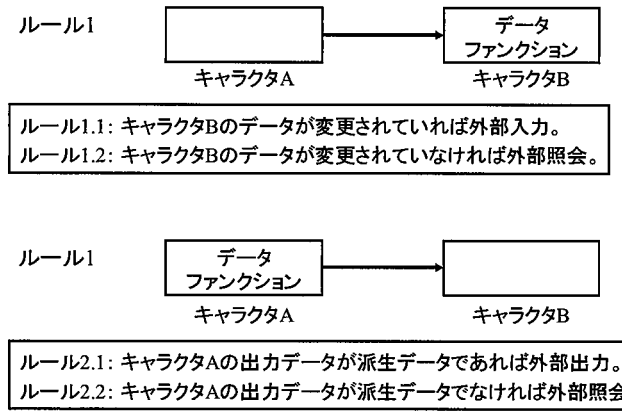


図 2.6-1 トランザクションファンクション判別ルール (1/3)

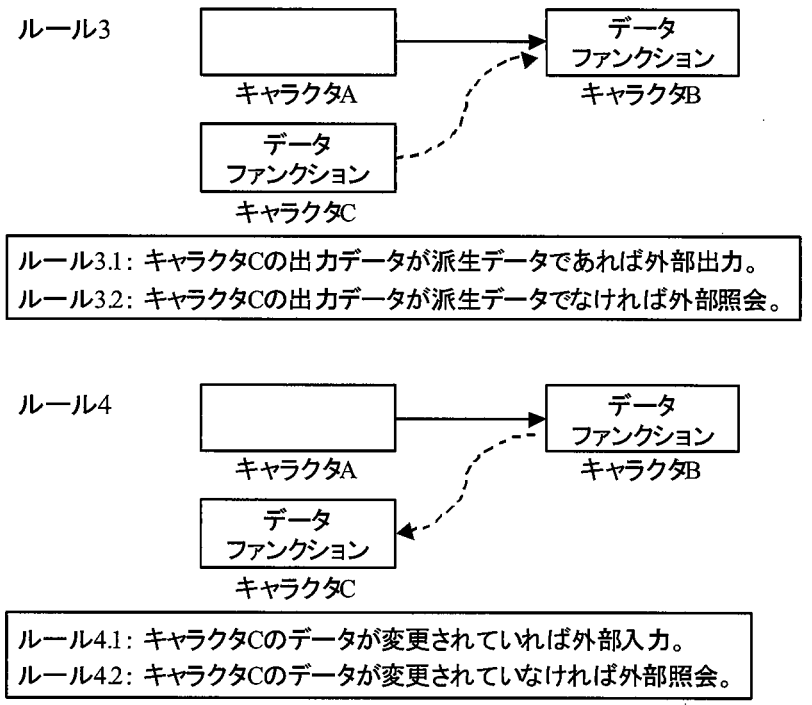
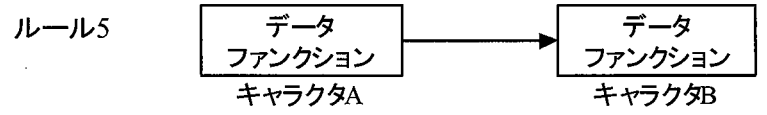


図 2.6-2 トランザクションファンクション判別ルール (2/3)



ルール5.1：キャラクタBのデータが変更され、キャラクタAの
出力データが派生データのとき、外部入力と外部出力。
ルール5.2：キャラクタBのデータが変更され、キャラクタAの
出力データが派生データでないとき、外部入力と外部照会。
ルール5.3：キャラクタBのデータが変更されず、キャラクタAの
出力データが派生データのとき、外部照会と外部出力。
ルール5.4：キャラクタBのデータが変更されず、キャラクタAの
出力データが派生データでないとき、外部照会。

図 2.6-3 トランザクションファンクション判別ルール (3/3)

外部照会と外部出力の決定は，出力データが派生データであるかどうかで行う．派生データの判別は，データ出力をしたキャラクタの属性とそのデータを受け取ったキャラクタの属性を比較し，異なる属性を持っていれば派生データと考えることにした．

抽出されるトランザクションファンクションが外部照会の場合には，それと対になる外部照会がないかを調べる．対となるシーンがあれば複雑度の比較を行い，複雑度の高い方を外部照会にする．各トランザクションファンクションの複雑さについては，関連ファイル数（対象となるトランザクションファンクションの処理中にデータが更新または参照されるデータファンクションの個数）とデータ項目数を数え上げることで判定する．

さらに，シーンに起動条件があるとき，その起動条件を含んでいるキャラクタがデータファンクションならば外部照会の機能を追加する．起動条件の判定には，なんらかの照会処理を行う必要があるからである．たとえば，図 2.3 で購入要求書がデータファンクションである場合を考える．まず，ルール 4 から外部入力が抽出できる．また，“分類＝一般品”という起動条件から，購入品が一般品であるかどうかを判定するための照会処理が必要となるので，外部照会も抽出する．

まとめると，計測ツールにおけるトランザクションファンクションの抽出手順は以下のようになる．

(1)　ルール（図 2.6 参照）に従い，トランザクションファンクションを抽

出し，その種類（外部入力，外部出力，外部照会）を決定する．

(2) 関連ファイル数（FTR）は，シーンに関与しているデータファンクションの数，データ項目数（DET）は入出力されたデータファクションの属性の数とし，予め定めた複雑さ決定表を用いて，各トランザクションファンクションの複雑さを判定する．

2.4.2.4 ファンクションポイント計測ツールの概要

Windouws95/98 上で動作する MFC（クラスライブラリ）を用いて，C++で実装した．プログラムサイズは約 7000 行である．ツールの入力は REQUARIO で作成された要求仕様書で，ファンクションポイントの計測結果とその計測のもとになる各ファンクションポイントの候補を出力する．

システム構成を図 2.7 に示す．

- 解析部：REQUARIO の出力ファイルを構文解析し，ファンクションポイント計測のために必要であるデータを抽出する．
- 解析 DB：解析部で抽出したデータを格納しておく．
- 計測部：ファンクションポイントを計算する．
- 計測 DB：計測結果（ファンクションポイント，データファンクション，トランザクションファンクション）を格納しておく．
- インタフェース部：計測結果（ファンクションポイント，データファンクションの候補，トランザクションファンクションの候補）を表示する．

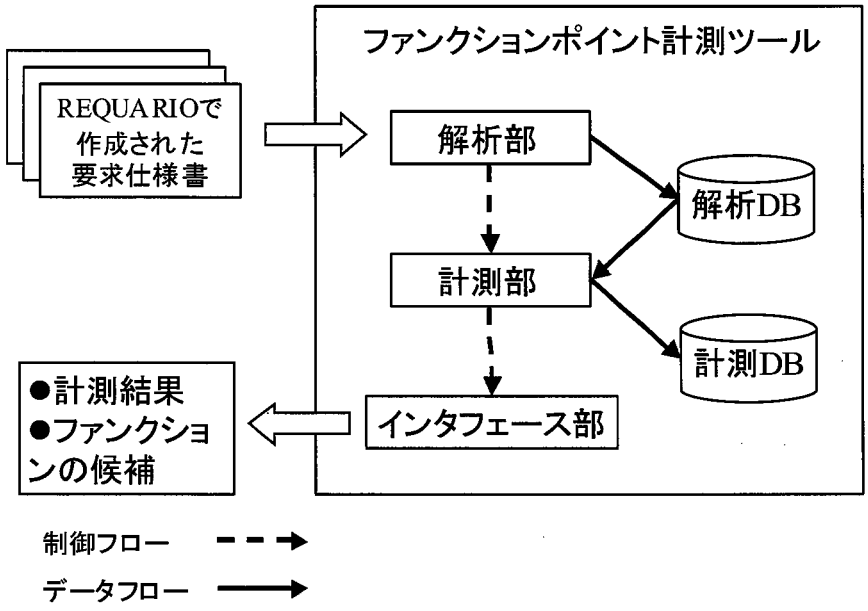


図 2.7 計測ツールのシステム構成

2.4.2.5 結果の調整

ツールで自動計測した後，各ファンクションの候補を画面表示により確認する．このとき本来ファンクションとして抽出されるべきだが漏れているものや，ファンクションの種類が間違っているものが存在すれば，キャラクタ一覧表示画面やシーン一覧表示画面から修正を行い，ファンクションポイントの再計算を行う．

この調整機能を用いて，自動的に判断できない部分の判断を計測者が行い，それによってより正確にファンクションポイントを求めることが可能となる．

2.4.3 適用対象

ツールによって計測されたファンクションポイント値が妥当であるものかを評価するために，計測ツールを実際の要求仕様書に対して適用した．具体的には，REQUARIO で作成された購入業務，発注業務，在庫管理業務の要求仕様書を評価対象とし，ツールによって計測されたファンクションポイント値と，ファンクションポイント計測の熟練者が測定したファンクションポイント値と

の比較を行った.

購入業務の要求仕様書には，備品を購入するときには様々な書類が作成されるが，それら書類の処理を含む業務の流れが記述されている．発注業務の要求仕様書には，備品を発注する際に行われる購入金額の見積りや発注先を決定する業務の流れが記述されている．在庫管理業務の要求仕様書には，商品を倉庫に管理する，あるいは商品の生産計画を立てるなどの業務の流れが記述されている.

これらのデータのシナリオ数，シーン数，キャラクタ数を表 2.3 に示す．なお，ツールを用いてファンクションポイントを計測する際には，データファンクションを指定しなければならないが，今回は熟練者がデータファンクションと判別したキャラクタを指定した．また，ツールによるファンクションポイント値としては，計測結果の調整をしていない値を用いることにした.

表 2.3 適用データ

	シナリオ数	シーン数	キャラクタ数
購入業務	7	15	6
発注業務	9	37	14
在庫管理業務	2	11	15

2.4.4 適用結果と考察

３つのシステムの要求仕様書に対して，ツールを用いて計測したファンクションポイント値と熟練者が計測したファンクションポイント値を表 2.4 に示す．比較の結果，ツールによって計測された値が，熟練者の測定した値に対して約80%の値となっていた.

表 2.4 適用結果

	熟練者の FP 値			ツールの FP 値		
	購入	発注	在庫	購入	発注	在庫
DF の FP 値	14	29	24	14	29	24
TF の FP 値	18	63	36	15	44	27
FP の合計	32	92	60	29	73	51

表 2.4 と各ファンクションの計測結果から，ツールでは抽出できていないトランザクションファンクションが存在するという結果が得られた．ツールと熟練者の計測結果をもとにその原因を分析した．

熟練者がファンクションポイントを計測する際に作成した資料とツールの出力結果を照合したところ，差が発生した原因は，熟練者が REQUARIO の記述にシステム化する際に必要と思われる処理を追加して計測しているためであることが判明した．そこで，差が発生した部分について，熟練者が追加した処理を REQUARIO の記述に追加して計測を行ったところ，今度は同じ値が計測された．

この結果は，計測ツールを用いてファンクションポイントを計測するときには，入力となる要求仕様書の詳細度をある程度保証する必要があることを示唆している．計測ツールに入力する前に，入力となる要求仕様書に対してレビューを実施して，必要な機能が正しく反映されていることを確認することが実用上必要となる．

2.5 まとめと今後の課題

イベントトレース図に対してファンクションポイントを計測するための指針を示した．そして，その指針に基づいて，ある CASE ツールで作成された要求仕様書に対して，ファンクションポイントを求めるための計測ツールを試作した．さらに，開発したツールを実際の開発現場での事例に適用し，ツールによって計算された値とファンクションポイント計測の熟練者が測定した値の比較を行った．

今後試作したツールを実用化していくためには，数多くの要求仕様書に対する適用とルールの改訂が必要と考えられる．今回の適用では，ツールの計測と

結果と熟練者の計測結果に大きな差は生じなかったが，業務仕様書には陽に記述されないシステム化機能の見積りに関して，ルールを補うべき点を発見することとなった．今後様々な要求仕様書からファンクションポイントを求めて，計測ルールを改善していく必要がある．

提案したファンクションポイントの計測指針は，イベントトレース図に基づく仕様書全般に適用可能である．たとえば，オブジェクト指向設計で作成された設計仕様書を対象として，ファンクションポイントを計測するツールを開発することも重要な課題である．REQUARIO 用のツールに続いて，Rational Rose [RA97]を用いて記述された設計書に対してファンクションポイントを計測するツールの開発も行われている [UE99].

第３章 複雑度と機能量に基づくアプリケーションフレームワークの実験的評価

3.1 はじめに

ソフトウェアの大規模化と複雑化に伴い，高品質なソフトウェアを一定期間内に効率よく開発することが重要になってきている．これを実現するために様々なソフトウェア工学技術が提案されてきている．再利用はそれらの中でも最も有効なものの一つである．

再利用は，既存のソフトウェア部品を同一システム内や他のシステムで用いることであると定義されている [BRA94]．一般にソフトウェアの再利用は生産性と品質を改善し，結果としてコストを改善するといわれている．実際の効果を報告した論文も多く発表されている [BA92] [IS92] [KE99] [MC02]．

一方，オブジェクト指向開発 [ZA99] では，凝集度が高く合成しやすいソフトウェア部品を再利用することでソフトウェアを効率よく開発することを目的としている．品質の高い部品を再利用することでソフトウェアの品質を向上することが可能であり，開発期間も短縮することができると指摘されている [JA97]．オブジェクト指向言語での開発において，開発者はフレームワークと呼ばれる特定のライブラリを再利用する [BO94]．フレームワークは特定のドメインに対するサービスを提供するクラスの集合である．フレームワークを用いた典型的な開発では，まず，開発者はプログラムの基本的な部品を取り出す．そして，アプリケーションを開発するのに必要な他の部分を作成し，それらを合成して一つのプログラムにする．例えば，The Microsoft Foundation Classes（MFC）は，Microsoft Windows のユーザインタフェース基準に準拠したアプリケーションを開発するためのフレームワークである [JEG01]．

ここに，ある情報システム開発を担当する部署が存在する．この部署では，特定ドメイン向けのビジネスアプリケーションフレームワークを開発し，システム開発に適用することを目標としていた．しかし，この部署では従来型のモジュール単位での再利用手法を長期にわたって使い続けており，新しいフレー

ムワークを開発現場に導入するには抵抗感も少なくなかった．コストの削減と生産性の向上の観点から，フレームワークを用いた再利用が従来の再利用よりも有効であることは定性的には理解されていたが，その効果を定量的に示して導入の動機付けをすることが必要となっていた．

そこで，本章では，ソフトウェアの品質と工数削減の観点から，フレームワークの有効性を実験的に評価することを目的とする．ここでは 2 つのケーススタディを行う．ケーススタディでは 4 種類のアプリケーションを Java で開発する．各々のアプリケーションは，フレームワークに基づく再利用と従来の再利用の 2 通りの方法で開発される．それらの間の差異を機能量と複雑さのメトリクスを用いて評価する．結果として，フレームワークを用いた再利用は従来型の再利用よりも効果的であることが確認された．

本章では，この節に続き，3.2 節では開発対象となるアプリケーションソフトウェアと導入されたフレームワークの特徴を紹介する．3.3 節ではケーススタディについて述べ，3.4 節では実験の結果を分析する．最後に 3.5 節で結論を述べ，今後の課題についてまとめる．

3.2 準 備

3.2.1 開発対象アプリケーション

ここに取り上げるある部署では，様々な地方自治体向けのオンラインアプリケーションシステムを開発している．ここでは，同一のアプリケーションがいくつかの地方自治体に対して開発されていく．アプリケーションシステムの機能には，住民票の登録，税金の支払い，健康保険の取り扱いなどがある．各地方自治体によって扱うデータが異なるので，各アプリケーションの細かい部分は異なっている．しかし，各機能について基本的な処理は同じである．簡単に言えば，いずれも典型的なデータベース操作を行う機能である．

従来，それらのアプリケーションの開発においては，モジュール単位の再利用が用いられてきた．図 3.1 はその典型的な例を示している．図 3.1(a) は地方自治体 A の健康保険のデータを更新するアプリケーションのモジュール構成図である．モジュール A_1 はメインモジュールであり，画面遷移の制御を行う．ユーザは入力データに応じて切り替わる画面を通してアプリケーションを操作する．各モジュール A_{11}，A_{12}，A_{13} は，データ照会，照会結果の表示，データ更新処理に対応する．従来のモジュール単位の再利用では，各モジュールは次の開発で再利用される．図 3.1(b) は地方自治体 B 向けの同じアプリケーションの構造を表している．ここで，モジュール B_{11}，B_{12}，B_{13} は，それぞれ A_{11}，A_{12}，

A_{13}に対応するモジュールをもとに，それらを変更して作成したものである．

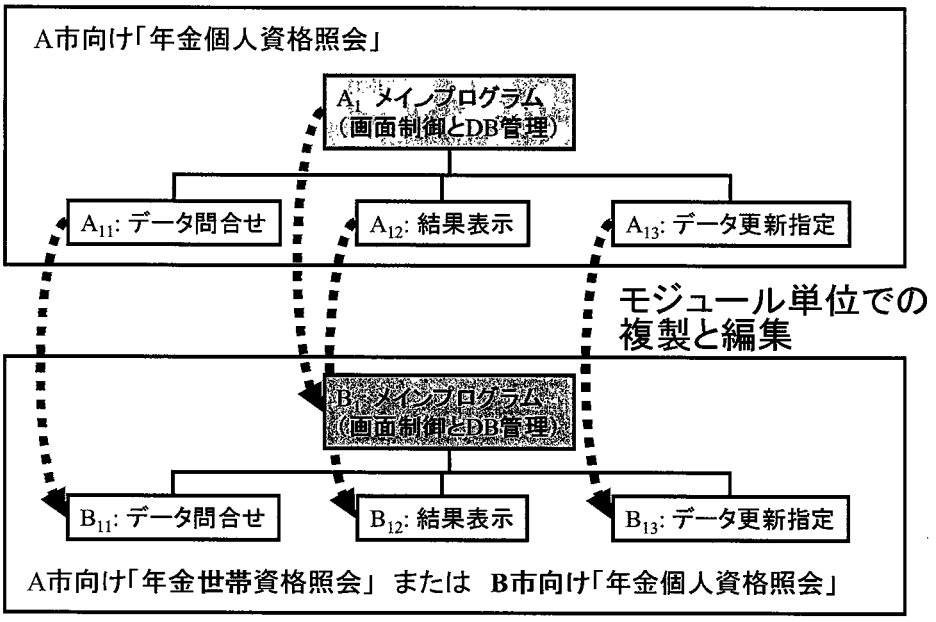


図 3.1 モジュール単位での類似プログラムの作成

3.2.2 導入するフレームワーク

近年，多くの企業でソフトウェア開発環境にオブジェクト指向技術が導入されている．前項で述べたアプリケーションもまたオブジェクト指向で設計し，Java で実装することが検討されており，新しいフレームワークが導入されようとしていた．このフレームワーク [YU02b] の概要を図 3.2 に示す．このフレームワークでは，モジュール型の再利用に加えて，画面遷移制御も再利用することを目的としている．また，データ照会，データ更新，データ追加，データ削除など，典型的な処理についての部品が用意されている．そこで，データ更新の部品は，データの照会，照会結果の表示，データ更新の 3 種類の画面の基本形式とそれらの間の画面遷移の制御方式を含んで構成されている．

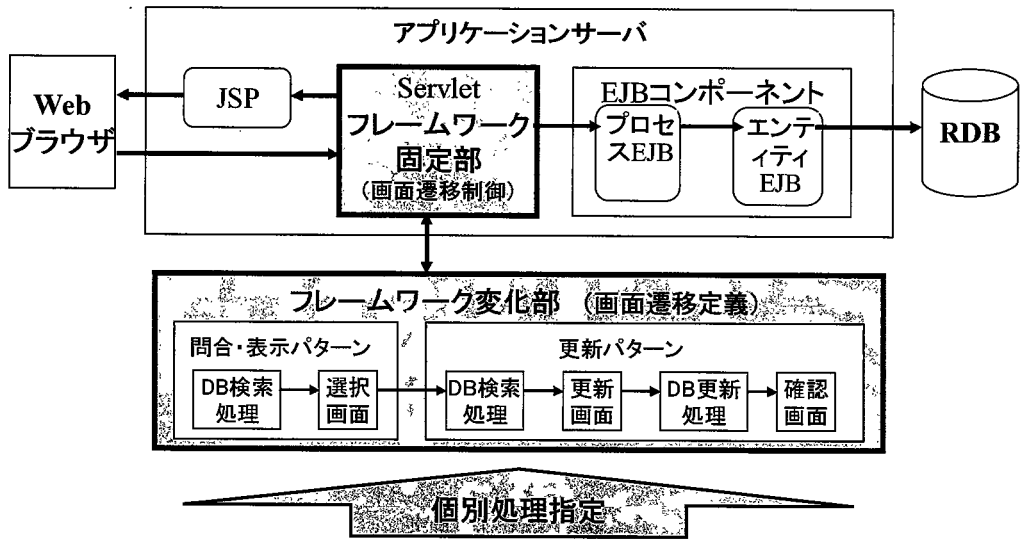


図 3.2 導入するフレームワークの概要

図 3.3 は, このフレームワークを用いた再利用の一例を示している. 図 3.3(a) は, 地方自治体 A の健康保険データを更新するアプリケーションの構造を示している. F_1 はデータ更新の画面遷移パターンをあわせ持っているフレームワークを表している. F_1 を用いることにより, データの照会, 照会結果の表示, データの更新は簡単に実装することができる. 地方自治体 A に固有の情報は F_1 へのパラメータとして与えられる. それゆえ同様のアプリケーションを地方自治体に対して開発する場合には, フレームワーク F_1 と地方自治体 B に固有の情報を用いて簡単に開発することができる (図 3.3(b) 参照).

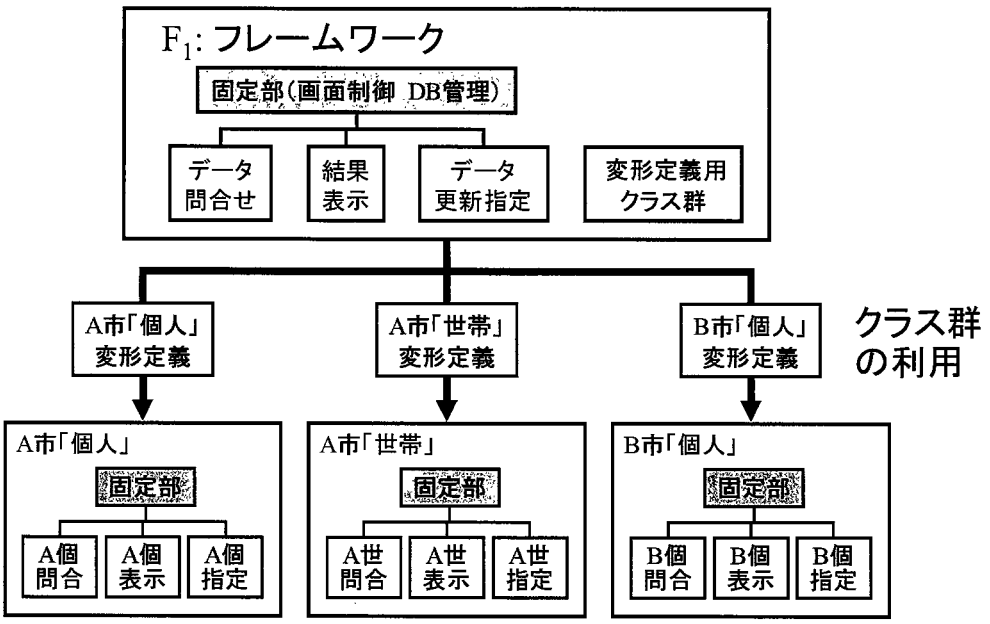


図 3.3 フレームワークを用いた再利用

3.3 ケーススタディ

ここでは，導入予定のフレームワークの有効性を評価するために行ったケーススタディについて述べる．

3.3.1 概要

ケーススタディ 1 とケーススタディ 2 の 2 つを取り上げた．

ケーススタディ 1 の概要を図 3.4 に示す．ケーススタディ 1 では，4 種類のプログラムを，従来のモジュール単位の再利用とフレームワークを用いた再利用の 2 通りの方法で開発した．C_a，C_b，C_c，C_d はフレームワークを用いた再利用で開発された 4 種類のプログラムを表す．一方，P_a，P_b，P_c，P_d は従来の再利用を用いたプログラムを表す．C_i と P_i，(i = a,b,c,d) はそれぞれ同じ機能である．

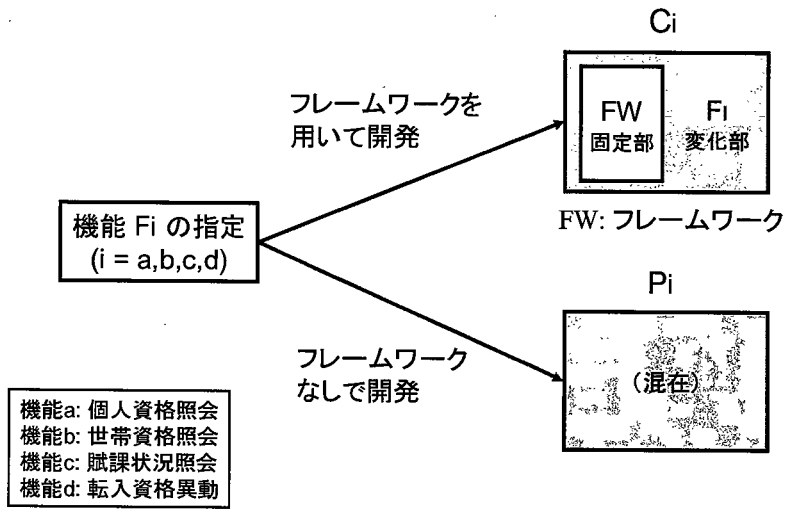


図 3.4 ケーススタディ 1 の概要

ケーススタディ 2 の概要を図 3.5 に示す．ケーススタディ 2 では，これも 1 つのプログラムを 2 通りの方法で開発した．ただし，4 つの機能 f_a，f_b，f_c，f_d は連続してプログラムに追加される．つまり，フレームワークを用いた場合は，まずプログラム f_a の機能を持つプログラム C_a を開発し，そして機能 f_b を C_a に追加することで C_{a+b} を開発する．同様に，C_{a+b} に f_c に追加することで C_{a+b+c} が開発され，最後に C_{a+b+c} に f_d に追加して $C_{a+b+c+d}$ が完成される．一方，従来の再利用手法を用いて P_a，P_{a+b}，P_{a+b+c}，$P_{a+b+c+d}$ が開発される．C_i と P_i はそれぞれ同じ機能である．

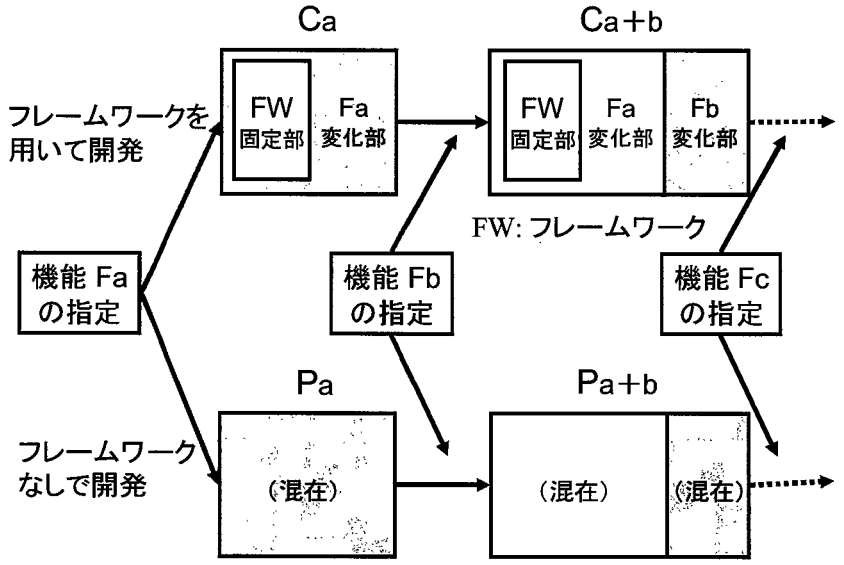


図 3.5 ケーススタディ 2 の概要

3.3.2 メトリクス

再利用による生産性と品質の向上を評価するため，OOFP(Object-Oriented Function Point)と C&K メトリクス(Chidamber と Kemerer の複雑さメトリクス)を用いた．なお，残念ながら，各ケーススタディにおいて実際の開発工数と開発中に摘出されたバグ数は収集できなかった．

ファンクションポイント（FP）はソフトウェアシステムの機能量を計測し，ソフトウェア開発の工数の見積り等に利用するために提案された [AL94] [IF94] [LO99]．しかし，元々の FP はオブジェクト指向プログラムに対して提案されたものではない．近年，OOFP が提案され [CA98] [AN99] [AN03]，これはオブジェクト指向言語によるプログラムから比較的容易に計測することが可能である．そこで，本章では OOFP を生産性に関するメトリクスとして用いることにした．

一方，C&K メトリクスはオブジェクト指向ソフトウェアの複雑さを計測する最も有名なメトリクスの一つである．C&K メトリクスと，発見されたバグ数との関係についての研究も報告されている [BRI98] [KA99a] [KA99b]．例えば Basili らは実験的に C&K メトリクスがクラスのバグや出やすさを，従来コードのメトリクスよりよく予測していることを評価した [BA96]．そこで，本章では C&K メトリクスを品質に関するメトリクスとして用いることにした．

3.3.3 OOFP

Caldiera らは, 従来の FP をオブジェクト指向で開発された要求/設計仕様書から計測できるようにカスタマイズし, OOFP として定義した [CA98].

従来, FP 計測における中心概念は, 論理ファイルとそれらのファイルを扱うトランザクションであった. オブジェクト指向ソフトウェアで中心となるのは, ファイルやデータベースではなく, オブジェクトが中心となる.

FP 法における論理ファイルは, オブジェクト指向ソフトウェアの上ではクラスに対応付けて考えられる. FP 法の論理ファイルは内部論理ファイル (EIF) と外部インタフェースファイル (EIF) に分類される. OOFP では, アプリケーションの境界内にあるクラスを ILF に対応させ, アプリケーションの境界外部にあるクラスを EIF に対応させている. ILF, EIF それぞれについてデータ項目数 (DET) とレコード種類数 (RET) を算出する必要がある. 整数やストリングなど単純な属性は DET の項目とみなされ, クラス型やクラスへの参照型の属性のように複雑な属性は RET の種類とみなされる.

FP 法におけるトランザクションは外部入力, 外部出力, 外部照会の 3 つに分類される. OOFP では, それらを単に一般的なサービスの要求 (SR : Service Request) として扱う. システム内のクラスのメソッドを SR として数えるが, 抽象メソッドは数えない. 具象メソッドについては, サブクラスに継承されていても, (宣言されているクラスないで) 1 度だけ数えられる. メソッドが数えられたら, その参照するデータの型を単純な型と複雑な型の 2 種類に分類する. 単純な型は引数やグローバル変数として参照している整数やストリング型を表し, 複雑な型はそのメソッドによって参照されている複雑な (クラス型やクラスへの参照型) の引数, 複雑なグローバル変数, オブジェクトを表す. 単純な型のデータは DET の項目として数え, 複雑な型のデータは関連ファイル数 (FTR) として数える.

最後に, 各 ILF, FIF, SR を DET, RET, FTR に基づく複雑さで重み付けして OOFP を算出する.

3.3.4 C&K メトリクス

Chidamber と Kemerer らのメトリクス [CHI94] は, オブジェクト指向ソフトウェアのクラス複雑度を, 内部, 継承, 結合の 3 つの視点から評価するものであり, 以下の 6 種類のメトリクスから構成される. C&K メトリクスの概要を図 3.6 に示す.

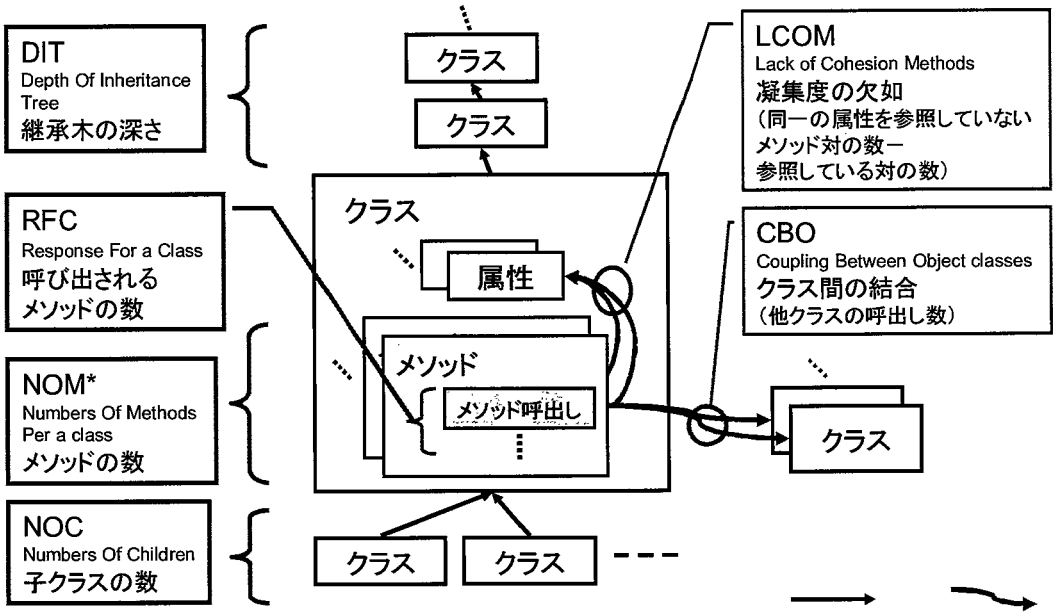


図 3.6 C&K メトリクスの概要

WMC（クラスの重み付きメソッド数；Weighted Methods per Class）：計測対象クラス C が，メソッド M_1，…，M_n を持つとする．これらのメソッドの複雑さをそれぞれ $c_1, \cdots, c_n$ とする．このとき $WMC(C)=\Sigma c_i$ である．適切な間隔尺度 f を選択して $c_i = f(M_i)$ によりメソッドを重み付けする．Basili ら [BA96] および Chidamber ら [CHI94] の研究においては，すべてのメソッドの複雑さが同じであるという仮定をおいて，WMC をメソッドの数としている．本章でも同じ仮定を用いる．

DIT（継承木における深さ；Depth of Inheritance Tree）：DIT は計測対象クラスの継承の深さである．多重継承が許される場合は，DIT は継承木におけるそのクラスを表す節点から根に至る最長パスの長さとなる．

NOC（子クラスの数；Number Of Children）：NOC は計測対象クラスから直接導出されているサブクラスの数である．

CBO（クラス間の結合；Coupling Between Object classes）：CBO は，計測対象クラスが結合しているクラスの数である．あるクラスが他のクラスのメソッドやインスタンス変数を参照しているとき，結合しているという．

RFC（クラスの反応；Response For a Class）：計測対象クラスのメソッドと，それらのメソッドから呼び出されるメソッドの数の和として定義さ

れる（すなわち，メッセージに反応して潜在的に実行されるメッセージの数である）．

LCOM（メソッドの凝集の欠如；Lack of Cohesion in Methods）：計測対象クラスCがn個のメソッドM_1，…，M_nを持つとする．I_i (i = 1，…，n) をそれぞれメソッドM_iによって用いられるインスタンス変数の集合とする．$P = \{(Ii, Ij) \mid Ii \cap Ij = \phi\}$と定義し，$Q = \{(Ii, Ij) \mid Ii \cap Ij \neq \phi\}$と定義する．もし，$I_i$，…，$I_n$がすべて$\phi$の時は，$P = \phi$する．このとき，$LCOM(C) = |P| - |Q|$，ただし，$|P| - |Q| < 0$の時は，$LCOM(C) = 0$と定義する．

3.3.5 計測方法

ここでは，上述したメトリクスをプログラムにどのように適用するかを説明する．フレームワークの効果を評価するために新規開発部分に注目する．図3.4は，ケーススタディ1におけるC_iとP_i，(i = a,b,c,d) の構造を示している．一方，図3.5は，ケーススタディ2におけるC_a，P_a，C_{a+b}，P_{a+b}の構造を示している．どちらのケースにおいても，メトリクスの値は新規にプログラムの開発された部分（影の付いた部分）から算出する．

表3.1と表3.2は，それぞれケーススタディ1とケーススタディ2の計測結果を表す．C&Kメトリクスの値は1クラスあたりの平均である．ここで，NOCとDITについては値が0になったため省略する．これは，これらのアプリケーションにおいてクラスの継承が全く用いられていないためである．

表3.1 ケーススタディ1の計測結果

	OOFP	CBO	RFC	WMC	LCOM*
C_a	176	3.8	14.4	7.4	21.4 (5.0)
C_b	180	5.8	18.2	7.6	23.2 (5.0)
C_c	418	5.4	33.1	8.1	28.7 (7.8)
C_d	252	4.1	17.8	6.4	13.8 (5.4)
P_a	526	2.1	5.8	3.3	2.1 (1.5)
P_b	526	2.3	5.9	3.3	2.1 (1.5)
P_c	671	3.0	7.4	3.4	2.0 (1.5)
P_d	672	2.4	8.6	4.3	15.7 (14.0)

*）括弧内はset/getメソッドを除いた場合

表 3.2 ケーススタディ 2 の計測結果

	OOFP	CBO	RFC	WMC	LCOM
C_a	176	3.8	14.4	7.4	21.4
C_{a+b}	251	5.0	16.7	7.6	20.1
C_{a+b+c}	578	5.9	30.1	8.3	28.6
$C_{a+b+c+d}$	743	5.8	28.3	7.9	25.6
P_a	526	2.1	5.8	3.3	2.1
P_{a+b}	615	2.8	6.9	3.3	2.0
P_{a+b+c}	849	3.7	8.6	3.3	1.8
$P_{a+b+c+d}$	1084	4.0	10.5	3.9	10.0

3.4 分 析

3.4.1 ケーススタディ 1

OOFP については，フレームワークを用いて開発したプログラムのほうが従来の再利用手法を用いたプログラムの値に比べて小さくなっている．(Pi は新規に開発されており，再利用部分はない．) 特に C_a，C_b，C_d の値は対応するプログラムの値より非常に小さくなっている．

次に C&K メトリクスの値を分析する．

CBO, RFC: C_i の値は P_i の値より高い複雑度を示している．これは C_i の新規開発クラスが多くのメソッド呼び出しを含んでいることを示している．フレームワーク内のメソッドに対する呼び出しを除いて CBO を計測すると，C_i のすべてのクラスで CBO は 0 となった．このことから，C_i のクラスのメソッド呼び出しは，すべてフレームワーク内のクラスのメソッドに対して行われていることがわかる．従って，フレームワーク内のクラスの品質が高ければ，結合による複雑さは一見高いがアプリケーションプログラム全体の品質に影響はないと考えられる．

WMC, LCOM: C_i の値は P_i の値よりも高くなっている．C_i のクラスのうち WMC, LCOM が平均より高いものについて調べたところ，クラスの属性を設定，参照するための set/get メソッドが約 45%を占めていた．set/get メソッドの処理はいずれも 1, 2 行であった．WMC の定義から，set/get メソッド数に比例して WMC は増加する．ある属性の set/get メソッドは，一般に他の属性の set/get メソッドと参照する属性に共通

するものがないため，LCOM の定義から，set/get メソッドの組み合わせ数に比例して LCOM は増加する．set/get メソッドが多いために WMC，LCOM が高くなっているが，set/get メソッドが単純であることから，アプリケーション全体の品質には影響しないと思われる．

3.4.2 ケーススタディ 2

表 3.2 を用いて，メトリクスの増加量を評価する．ここで，Δ_b，Δ_c，Δ_d の値はそれぞれ機能 f_b，f_c，f_d を追加した時の OOFP の増加量を表す．C_i（フレームワーク使用）では，Δ_b，Δ_c，Δ_d の値はそれぞれ 75，327，165 であった．一方，P_i（フレームワークなし）では，Δ_b，Δ_c，Δ_d の値はそれぞれ 89，234，235 であった．Δ_c では，C_i の方が P_i の値より高かった．これは機能 f_c を実装するために f_b，f_d よりも多くのデータ項目が処理されるためである．また機能 f_c とフレームワークとの適合性が良くないこともわかった．従って，f_c のような機能を合わせるために新たなコンポーネントをフレームワークに追加する必要がある．

次に C&K メトリクスについて分析する．ここで，δ_b，δ_c，δ_d はそれぞれ機能 f_b，f_c，f_d を追加した時の C&K メトリクスの増加量を示す．

CBO, WMC: C_i，P_i の間で大きな違いは見られない．

RFC: C_i において，δ_c は 13.4 (= 30.1 – 16.7)であり，P_i の値 (1.7 = 8.6 – 6.9) によりもかなり高い．フレームワークを用いると，メソッド呼び出しの数は増加する．呼ばれているメソッドはすべてフレームワークに含まれるクラスのものである．従って，フレームワークが高品質であれば，アプリケーション全体の品質に影響はない．

LCOM: C_i において特に差はない．一方，P_i では δ_d (8.2 = 10.0 – 1.8) が δ_b や δ_c に比べて大きくなっている．

最後に，$C_{a+b+c+d}$ と $P_{a+b+c+d}$ について比較する．OOFP はそれぞれ 743，1084 である．従って，全体ではフレームワークを用いることによって開発工数が削減されたと考えられる．一方，C&K メトリクスについては，$C_{a+b+c+d}$ の複雑度が $P_{a+b+c+d}$ よりも高くなっている．しかし，その複雑度はフレームワークのクラスと新規開発クラスとのメッセージの送受信によるものである．従って，フレームワークが高品質であれば，複雑度はアプリケーションシステム全体の品質に影響を与えないと考えられる．

3.5 おわりに

本章では，工数削減の効率とソフトウェアの品質という観点から，フレームワークの有効性を実験的に評価した．2つのケーススタディの結論として，フレームワーク型の再利用の方がモジュール型の再利用よりも効果的であることが示された．

実際にはフレームワークの開発のための工数が必要であり，OOFP は 1298 であった．しかし，ケーススタディ 1 の結果から，フレームワーク型の再利用での OOFP 値は従来の再利用より 2.5 倍ほど効果的であるといえる．従ってこの一連の開発プロジェクトを推進している部署では，3 つか 4 つのアプリケーションを開発すれば，フレームワークへの投資を回収できるものと考えられる．

今後の課題としては，フレームワークの有効性を示すために，多くのソフトウェア開発プロジェクトにフレームワークを適用していく．さらに，フレームワークの適用範囲を広げるために新しいコンポーネントを追加していく必要がある．

第4章 財務情報記述言語 XBRL で定義された財務データの会計ユーザ向け処理方式

4.1 まえがき

異なるプラットフォームの間で円滑にデータ交換を行うことを目的とし，XML [WW96] が開発された．XML は汎用的な言語であるため，特定業種の取引や情報を表現するために XML を利用した言語が多く策定されている．企業の財務情報は従来紙に記述されたものが授受されていたが，ネットワーク環境の発展に伴い，WEB 上で PDF 形式などを用いて迅速に公開されるようになった．しかし，財務情報の形式が統一されていないために，異なる企業間の財務報告書を比較しづらい，あるいはデータを計算機処理に供しにくいといった問題があった．

これらの問題を解決するため，XML の構文を用いた標準化言語として，2000 年 7 月に XBRL（eXtensible Business Reporting Language）[XB00] の第一版が XBRL International により策定された．わが国においては，2008 年度より金融庁において，XBRL 形式による財務報告の提出と EDINET（Electronic Disclosure for Investors' NETwork）による開示サービスの開始が発表されている [KI07b]．

XBRL 文書処理は，既存の XML 処理系を使って行うことができる．しかし，XML 処理系を用いた文書処理においては，XML に固有のデータ構造と処理系に固有のライブラリの仕様を理解し，データ構造を解析するプログラムを記述する必要がある．財務情報を記述または参照する利用者は，企業の財務部門や，税理士，会計士，投資家などであり，XBRL 文書を変換するために複雑な XML 関連技術の理解を要求することは現実的ではない．

そこで本章では，XBRL 文書処理を容易に行うためのデータモデル SDX を提案し，SDX に基づくデータバインディング・ライブラリを開発し，SDX の実用性を検証する．

以降，4.2 節では XBRL を，4.3 節では既存の XML 処理系による XBRL 文

書処理について概説する．4.4 節では SDX 設定の考え方と SDX の仕様を述べ，4.5 節では SDX に基づくデータバインディング・ライブラリの実現と利用について述べる．

4.2 財務情報記述言語 XBRL

4.2.1 財務情報と財務諸表

企業や団体は，定期的に自らの財務情報を貸借対照表，損益計算書，キャッシュフロー計算書といった財務諸表にまとめて公表する．投資機関や銀行は，財務諸表に記述された情報から企業の収益性や成長性を判断し投資を行う．取引先や顧客も，財務諸表の情報をもとにその企業の安定性を評価するなど，財務諸表は経済活動にとって極めて重要な文書として用いられている [YU06b]．

貸借対照表の例を図 4.1 に示す．貸借対照表とは，資産，負債および資本を対照表示することにより，企業の財政状態を明らかにする報告書であり，資産，負債，資本とそれらの内訳となる勘定項目とその値となる金額が列記される．この例では，＜資産の部＞が，「流動資産」および「固定資産」から構成されており，それぞれの和が＜資産の部＞および「資産合計」の値になっている．このように，各要素の間には，通常同じ表の中の要素と関係が存在し，あるいは他の表の要素と複雑な関係を持つ場合もある．

株式会社○○○　貸借対照表　(単位:百万円)

項目	金額	項目	金額
＜資産の部＞	8,592,822	＜負債の部＞	6,439,500
流動資産	3,620,881	流動負債	3,090,821
現預金及び有価証券	1,487,544	買掛債務	765,041
売掛債権	919,468	社債及び短期借入金	550,000
商品・製品	140,516	その他	1,775,780
その他	505,277	固定負債	3,348,679
固定資産	4,971,941	社債及び長期借入金	2,000,600
有形固定資産	1,269,042	その他	1,348,079
無形固定資産	1,242,883	＜資本の部＞	2,153,322
投資その他の資産	2,460,016	資本金	397,049
		資本剰余金	416,970
		利益剰余金	1,737,602
		その他	△398,299
資産合計	8,592,822	負債及び資本合計	8,592,822

図 4.1 貸借対照表の例

4.2.2 XBRL の目的

XBRL は，企業の財務諸表のデータを，企業の部署間や企業間で電子的に交換するための標準化記述言語であり，インターネット上での標準化データ記述言語である XML を利用することによって定義されている. XBRL が情報基盤として普及することによって実現が期待される，財務情報のサプライチェーンの概要を図 4.2 に示す.

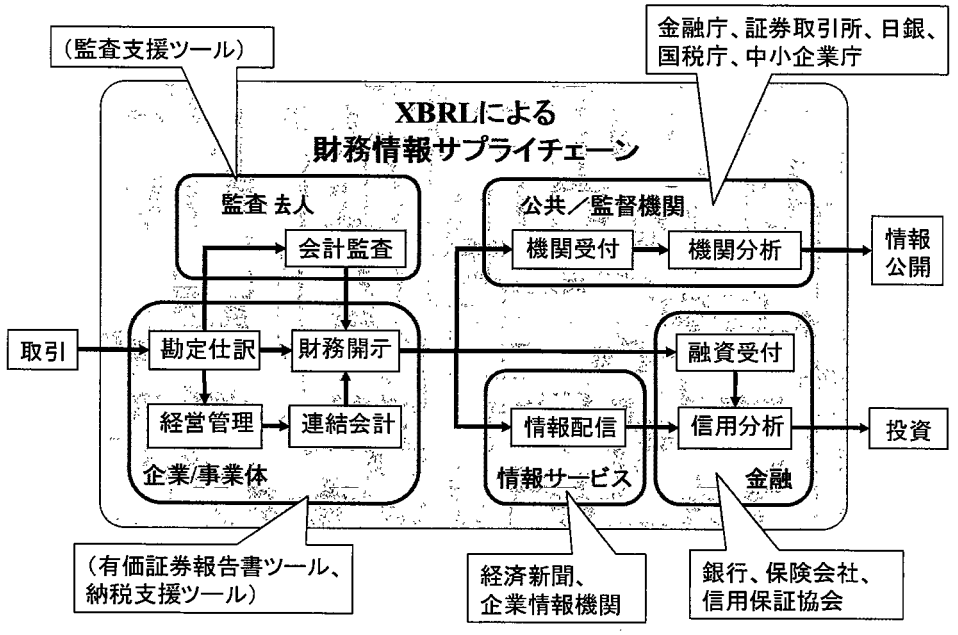


図 4.2 XBRL に基づく財務情報サプライチェーンの概要

個々のビジネスプロセスにおいて XBRL データを利用する局面では，次のような便益が期待される.

財務諸表を作成する立場においては，財務情報を XBRL で一元管理することによって，様々な形式の財務情報を自動的に変換するシステムを構築することが容易となり，各機関への提出文書の信頼性と作成効率を向上させることができる．財務諸表を，誤りを含んだままに開示してしまうと，その企業にとって社会的信用を失うことになり，誤りの程度によっては証券取引市場への上場廃止に追い込まれる場合もある．このため，財務諸表作成は，誤りを発生させないように極めて慎重に行う必要がある．また，企業単独での財務諸表のほかに

企業グループ全体で集計した連結決算での財務諸表の開示が必要となっており，財務諸表の作成過程は複雑化しているのに対し，開示日程は年々早期化しており，作業の効率化が大きな課題となっている [YU04].

また，利用する立場においては，XBRL 標準データを扱うソフトウェアが普及することによって，各社の財務情報を利用しやすい形式に変換することが容易となる．従来は，開示されている情報の画面表示をもとに自分の計算機環境のソフトウェアに対して転記入力を行っていたため，入力および入力誤りの修正，検証などに少なからず負荷を生じさせていた．株式投資業務においては，得られた企業財務情報を他の機関よりも早く活用することが差別化要因となる．また，転記や修正の事務作業に煩わされることがなくなれば，それだけ財務情報の専門的な分析作業に集中することが可能となる．

4.2.3 XBRL 文書の構成

本章で提案する処理方式は，2001 年 12 月に公開された XBRL 2.0 [XB00] を対象としている．以下では，XBRL 2.0 について簡単に説明する．

XBRL 文書はインスタンス文書とタクソノミ文書から構成される．タクソノミ文書はタクソノミ本体（以下，タクソノミ）と 5 種類のリンクベースから構成され，XBRL 文書の語彙と，財務報告書に記される項目間の関係を定義する．インスタンス文書にはタクソノミ文書によって定義された語彙で財務事実が記述される．図 4.3 に，XBRL 文書の構成を示す．

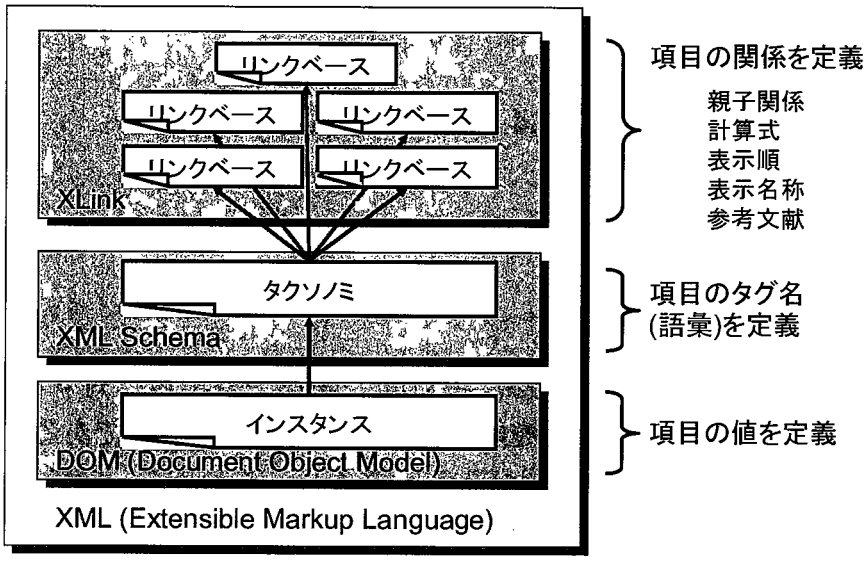


図 4.3 XBRL 文書の構成

4.2.3.1 タクソノミ

タクソノミ（Taxonomy）は，インスタンス文書の語彙（要素名，属性等）を定義するXMLスキーマ [WW01a] 文書である．タクソノミでは，インスタンス文書に記述される項目要素の要素名や型の定義を行う．XBRL 2.0では，金額型，10進数型，株数型，文字列型，URI型，日付型の6種類の型が定義されている．

タクソノミの例を図4.4に示す．この例では，XBRL 2.0を定義したURLを引用し（2行目），図4.1における「資産合計（Assets）」および「流動資産（CurrentAssets）」を金額型（xbrli:monetaryitemtype）のデータとして定義している（6-9行目）．

```
1 <schema
2    xmlns:xbrli="http://www.xbrl.org/2001/instance"
3    targetNamespace="http://www.Kaisha.com/bs"
4    ......      >
5    ......
6 <element id="assets" name="Assets"
7     type="xbrli:monetaryItemType"/>
8 <element id="currentassets" name="CurrentAssets"
9     type="xbrli:monetaryItemType"/>
10  ......
11 </schema>
```

図 4.4 タクソノミの例

4.2.3.2 リンクベース

リンクベース（Linkbase）は，項目間の関係や各項目に対する追加情報をXLink [WW01b] の外部リンク機能を利用して定義した文書である．

リンクベースには5つの種類がある（表4.1）．このうち，定義リンクは項目の親子関係を示すものである．親子間などで項目の値に関する計算式の関係を示す計算リンクや，兄弟にあたる項目のなかでの表示順序を示す表示リンクも定義リンクとは別に定義される．名称リンクは，一つの項目に対して日本語名，

英語名など複数の名称を対応付けるためのリンクである．参照リンクは，項目から項目の補足説明となる注記や参考資料などを参照するリンクである．

表 4.1 リンクベースの種類

種類	意味
定義リンク	項目間の親子関係を定義
計算リンク	項目の値の計算式を定義
表示リンク	項目の表示順序を定義
名称リンク	項目の名称を定義
参照リンク	項目の参考文献を定義

リンクベースの例を図 4.5 に示す．この例では，XBRL のリンクベース構文を定義した URL を引用し（5 行目），図 4.1 における「資産合計（Assets)」および「流動資産 (CurrentAssets)」の間の親子関係を定義している (8-17 行目)．

```
1  <linkbase
2    xmlns="http://www.xbrl.org/2001/XLink/xbrllinkbase"
3    xsi:schemaLocation=
4       " http://www.Kaisha.com/2003/bs.xsd"
5    xmlns:xlink= " http://www.w3.org/1999/xlink ..... >
6          .....
7  <definitionLink xlink:type="extended">
8   <loc xlink:type="locator" xlink:href="bs.xsd#Assets"
9    xlink:label="AssetsLocator"/>
10  <loc xlink:type="locator" xlink:href="bs.xsd#CurrentAssets"
11   xlink:label="CurrentAssetsLocator"/>
12  <definitionArc xlink:type="arc"
13   xlink:from="AssetsLocator" xlink:to="CurrentAssetsLocator"
14   xlink:arcrole="http://www.xbrl.org/linkprops/arc/parent-child"/>
15  <definitionArc xlink:type="arc"
16   xlink:from="CurrentAssetsLocator" xlink:to="AssetsLocator"
17   xlink:arcrole="http://www.xbrl.org/linkprops/arc/child-parent"/>
18         .....
19 </definitionLink>
20       .....
21 </linkbase>
```

図 4.5 リンクベースの例

4.2.3.3 インスタンス文書

インスタンス（Instance）文書は，タクソノミ文書で定義された項目によって財務事実を記述したXML文書である．主に，項目要素およびコンテキスト要素から構成される．

項目要素とは，財務事実を記述するための要素である．項目要素の要素名はタクソノミで定義される．財務事実は，次に説明するコンテキスト要素への参照を指定する．

コンテキスト要素は，数値コンテキスト要素（numericContext），非数値コンテキスト要素（nonNumericContext）の2種類がある．数値コンテキストは，数値として記述された財務事実に関するメタデータや属性（期間や単位など）を定義する．非数値コンテキストは，文字列として記述された財務事実に関して同様の情報を定義する．

図4.6に，簡単なインスタンス文書の例を示す．

この例では，図4.1に示す貸借対照表の項目のうち6つを記述している．1-3行目はインスタンス文書全体を示すトップレベルの項目要素（group要素）の開始行であり，22行目が対応する終端行である．この例では，トップレベル要素に含まれる項目要素として，6-17行目において6つの財務事実が記述されている．これらの項目要素では，c1というidによって，19行目から21行目に記述された数値コンテキストへの関係を定義している．

インスタンス文書では，トップレベル項目要素を除くと，項目要素が子要素を含むことを許しておらず，項目要素が単純に列挙されたXML文書となっている．そのために，group要素とトップレベルの項目要素の関係以外では，親子関係はすべてリンクベースを用いて定義することになる．

```
1 <xbrli:group
2   xmlns:xbrli="http://www.xbrl.org/2001/instance"
3   xmlns:jp-bs= " ..... "       ..... >
4   .....
5 <!-- Item-Elements -->
6 <jp-bs:Assets numericContext="c1">8592822000000
7   </jp-bs:Assets>
8 <jp-bs:CurrentAssets numericContext="c1">3620881000000
9   </jp-bs:CurrentAssets>
10 <jp-bs:FixedAssets numericContext="c1">4971941000000
11   </jp-bs:FixedAssets>
12 <jp-bs:LiabilitiesEquity numericContext="c1">8592822000000
13   </jp-bs:LiabilitiesEquity>
14 <jp-bs:Liabilities numericContext="c1">2889500000000
15   </jp-bs:Liabilities>
16 <jp-bs:Equity numericContext="c1">5703322000000
17   </jp-bs:Equity>
18 <!-- Context -->
19 <xbrli:numericContext id="c1" precision="18" cwa="true">
20   .....
21 </xbrli:numericContext>  .....
22 </xbrli:group>
```

図 4.6 インスタンス文書の例

4.3 DOM による XBRL 文書の表現

本節では，XML 文書を操作するための汎用的なデータモデルである DOM を用いた，従来の XBRL 文書の表現とその問題について述べる．

4.3.1 DOM による XBRL 文書の表現方法

DOM（Document Object Model）[WW97] とは，W3C（World Wide Web Consortium）が策定した XML 文書を表現するためのデータモデルである．DOM は，XML 文書に現れる要素，テキスト，属性などの文書構成要素をノードとして表し，ノードが相互に関係する木構造（DOM ツリー）として XML 文書を表現する．

DOM によって XBRL 文書を表現した場合，例えば図 4.6 のインスタンス文書は，図 4.7 のように表される．図中の四角がノードまたはノードに準じるものを表し，矢印で示す方向に関連付けられる．DOM ツリーは，XML 文書のトッ

プレベル要素である document ノードから始まる木構造として表現され，要素毎にノードが設けられるほか，要素の値としてのテキストデータや，要素に付与された属性，あるいは XML の記述にコメントが加えられた場合にはコメントにもノードが設けられる．XBRL の項目要素にはすべてコンテキスト属性が付与されるので，コンテキスト要素（この例では numericContext）もノードとして定義する．DOM ではこれらの属性に関するノードを項目要素のノードから名称管理表（NamedNodeMap）を経由して辿るように定義するので，名称管理表もノードに準じた木構造の要素として存在することになる．

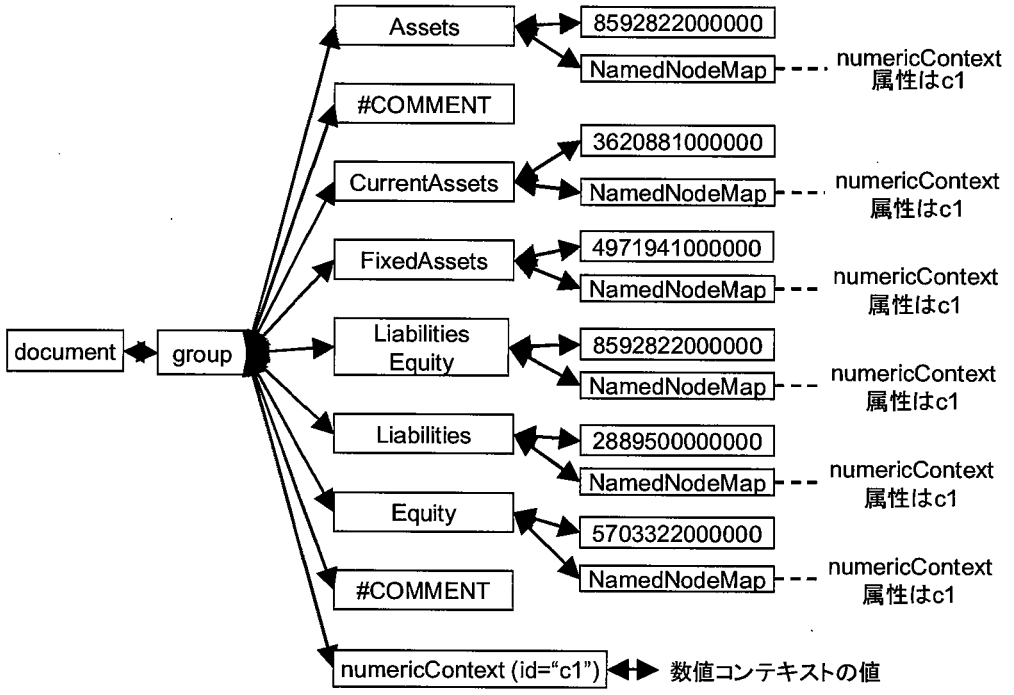


図 4.7 DOM ツリーによる XBRL 文書の表現

4.3.2 DOM による XBRL 文書表現の問題点

XML 文書においては，タグを区切り記号として要素が記述されるが，要素の内部にはさらにタグを用いて子要素を記述することが可能である．要素の値は，文字列や数値など線形のテキストデータだけではなく入れ子を深さの制限なく持つ木構造を含むことになるので，DOM においては，テキストデータも入

れ子を持つ要素も同じノードとして表現し，常に木構造を処理することを前提にしてXML文書を解析することになる．つまり，XBRLインスタンス文書における単純に列挙された項目要素に対してデータの読み取りや書き換えを行う際にも，木構造データ処理の枠組みで煩雑な処理を行う必要がある．例として，CurrentAssets要素の値を参照する手順を記す．

- DOMツリーから要素名をもとにCurrentAssets要素ノードを探す
- 見つかったCurrentAssets要素ノードの子ノードリストからテキストノードを選ぶ（コメントノードなどテキストノード以外のノードが含まれる場合がある）
- テキストノードの値を参照する

さらに動作の信頼性を重視する場合には，値を参照する前に，子ノードリストから名称管理表を介してコンテキスト属性のノードを探して CurrentAssets要素のデータ型などを確認する．

DOMは，JavaやC++などのプログラミング言語から利用するライブラリが普及しており，経験のあるプログラマにおいては汎用的に利用することができる．しかし，DOMを利用するためには，DOMにおけるノードの種類に関する知識やそれらを走査して識別するプログラミングの技法を習得する必要があり，ビジネスの現場において財務報告の情報を参照あるいは処理する会計業務担当者には利用が容易ではない．

4.3.3 関連する研究

企業間でXMLデータをやり取りする場合には，各社のスキーマの間でのスキーマ変換が多発する．Cupid [MA01] は，スキーマ間におけるデータ名称や構造の類似性を評価して自動変換を行うアルゴリズムとして提案されている．XBRL データにおいてもデータ変換は主要な処理の一つである．本章の研究では，XBRL のデータ変換だけではなく，個別のデータ加工，選別なども含めたプログラミングの場面を想定して，プログラミングにおけるデータアクセス記述の容易化を課題としている．

SMART [OO05] は，変換元と変換先のスキーマから概念モデルと呼ぶクラス図を抽出し，利用者にクラス間の対応関係を指定させ，その対応関係に基づいた変換を行うデータ変換支援システムである．SMART では，概念モデルを用いることにより，利用者がデータの内容を理解し，データ変換設計を容易に行うことを狙いの一つとしている．本章で提案するSDXは，いわばXBRLデータ

に対する概念モデルの一種であり，同様にデータの理解容易化を狙いとする．ただし，XBRL データにおけるクラス間の関係は，DOM や XLink の技術仕様に依存して定義されることになるので，SDX では，さらに財務諸表の仕様に特化したモデル化を行っている．

このほか，データ変換における DB への共通スキーマ参照の負荷を軽減する方式 [HA02] や，変換効率の良い変換ルールを作成する手法 [TO02] が報告されている．

また，財務報告の項目は多くの財務データを集約したものである．財務データのうち金融商品に関するデータは，財務報告の項目と集約される以前に，それ自身が金融市場において頻繁に取引されるものであり，データの交換や分析の方法に関して研究が進められている．たとえば，金融商品のためのカレンダー等時間情報の表現方法 [CHA93] [JES99] や，金融市況情報の流れのなかから値動きの類似を分析する方法 [WU04] [FO94] 等が報告されている．

4.4 XBRL アプリケーションのためのデータモデル SDX

本節では，XBRL インスタンス文書の特徴を考慮した専用のデータモデルである SDX に関して述べる．

4.4.1 XBRL インスタンス文書の特徴

XBRL タクソノミにおいて，項目要素は子要素を持たないものとして定義されるので，インスタンス文書の項目要素の値は，入れ子構造を持たない単一の値であると特定して処理することができる．

一方，インスタンス文書に記された項目要素間には，次に示す関係がある．

- 入れ子構造による親子関係
- リンクベースで定義された項目要素間の関係
- 項目要素からの属性参照関係

このうち入れ子構造は XML トップレベル要素および XBRL インスタンス文書としてのトップレベル項目要素 (group 項目要素) とその他の項目要素の間だけに用いられ，一般の項目要素同士の親子関係はすべてリンクベースで定義されている．

4.4.2 財務報告業務におけるデータ処理の分析とSDXモデル設定の考え方

SDXモデルの設定に当たっては，以下に示す二つの観点からXBRLデータ処理記述の容易化を図った．

一つは，XMLの技術的な知識を有しない会計業務担当の利用者においても，データモデルの理解を容易とすることである．業務担当者における理解が容易であれば，アプリケーションの要件定義やテストにおける正確性の向上が期待される．そこで，財務諸表の項目と属性の項目のみでノード（項目要素ノード）を構成することにした．

もう一つは，項目データの処理に関するプログラミングを単純化させ，生産性の向上を図ることである．そこで，プログラミングに用いる頻度の高い情報を，なるべく項目要素ノードおよび項目要素ノード間を連結するアーク（項目要素アーク）に集約させることにした．

XBRLデータ項目の利用頻度に関しては，財務報告業務におけるデータ処理のプロセスに基づいて分析した．

一般に，データ処理においては，定義，生成，参照，更新，削除のプロセスが考えられる．XBRLデータにおいては，これらのうちで，定義，生成，更新，削除は，おもに財務報告を作成する側で実施される．財務報告作成は，企業を代表して特定の担当者によって実施される．また，会計基準等により財務報告の形式が半ば規定されるので，アプリケーション・パッケージが開発されており，それが適用される可能性も高い．

これに対し，参照のプロセスは，金融機関，投資家，取引先など企業を取り巻く関係者全般にわたる財務データの利用者の側で実施され，それぞれの用途により参照方法も多様となる．そこで，個別にプログラミングが必要となる可能性が高い．つまり，XBRL データ処理の記述では，一般に「参照」を記述する頻度が高いことになる．

XBRLのデータ参照は，下記(1)から(3)の操作の繰り返しと考えられる．

(1) 項目要素を探す．
(2) 項目要素からその値を参照する．
(3) 参照した値を処理する．

さらに，XBRLデータにおいては，上記(1)の操作に対して下記の方法が提供

されている.

(a) 項目要素の名前から項目を探す.
(b) ある項目要素から親もしくは子にあたる項目要素を探す.
(c) ある項目要素から親子関係以外のリンクを辿って他の項目要素を探す.

このうち, 頻度が高いのは方法(a)および(b)である. 方法(a)に関しては, DOM においても要素名を指定するだけで簡便に記述される.

方法(b)は, 親子階層に属する項目全体にわたって処理を繰り返す場合などに発生する. また 4.3.2 節に述べたように, (2)の操作に付随して記述されることも多い. 従来はリンクベースを検索する専用関数を用意し, その専用関数を用いて親もしくは子の項目要素の所在を参照する必要があった. そこで, SDX モデルにおいては, 定義リンク情報を項目要素アークに格納し, 専用関数を用いることなく参照可能とすることにした.

方法(c)に関しては, 財務報告書類に含まれる注記から参照リンクを介して参照される場合が想定されるが, 2007 年 4 月現在, EDINET における XBRL 導入計画では注記への導入が除外されている [KI07c]. その他定義リンク以外のリンク情報は利用頻度が低いと考えられたので, リンクベース参照のままに残すことにした. これは, 項目要素アークの項目を少数に抑えて, データモデルの理解の容易さを維持するためである.

(2)の操作については, すでに DOM による操作手順を 4.3.2 節に記した. これによれば, 項目ノードのほかにテキストノードの探索が必要である. そこで, SDX モデルにおいては, これを項目要素ノードの内部だけで済ませることができるようにした. また, プログラムから値を参照する場合には, その場で改めてデータ型を検証することが多いので, タクソノミに記述されたデータ型の情報も項目要素ノードに含めることにした.

(3)の操作は, 参照した金額値等の加工や入出力の記述であるが, 会計業務担当者が利用する場合には, 数個の数値の合計や二つの比の計算など, 比較的単純で小規模の記述であることが多い. そこで, 加工や入出力の記述への対応は, SDX モデルでは直接取り上げないことにした.

4.4.3 SDX モデル

XBRL 文書で表された財務データを処理するアプリケーションのためのデータモデルとして, SDX (Simple Data Model for XBRL Documents) を提案する (図 4.8).

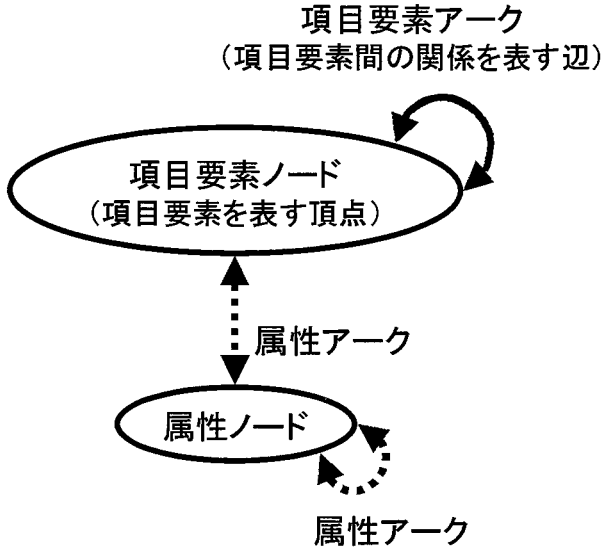


図 4.8 SDX モデルの概要

SDX は，以下の定義による有向グラフである．

- 頂点は，項目要素ノードおよび属性ノードの 2 種類である．
- 項目要素ノードは，項目要素に対応して設けられ，内部に項目のデータ型および値と，属性とその値への参照を保持する．
- 属性ノードは，属性値の表現に関して用いる．
- 辺は，項目要素アークおよび属性アークの 2 種類である．
- 項目要素アークは，項目要素間の親子関係に対応して設けられ，項目要素ノード間を連結する．
- 属性アークは，項目要素ノードと属性ノードの間，および属性ノード間の親子関係に関して設けられる．

SDX モデルの適用例を図 4.9 に記す．この例は，図 4.1 の貸借対照表の一部を表現している．例えば，図 4.4 のタクソノミにおける「資産合計（Assets)」および「流動資産（CurrentAssets)」に関する記述（6-9 行目）は，Assets および CurrentAssets の項目要素ノードとして表現される．図 4.5 のリンクベースにおける親子関係の定義（8-17 行目）は，Assets および CurrentAssets の項

目要素ノード間の項目要素アークとして表現される. 図 4.6 のインスタンス文書における値の記述（6-9 行目）は，Assets および CurrentAssets の項目要素ノードの内部情報として表現される．インスタンス文書で定義される group 要素は，これも項目要素として項目要素ノードで表現し，他の項目要素との親子関係を項目要素アークとして表現する.

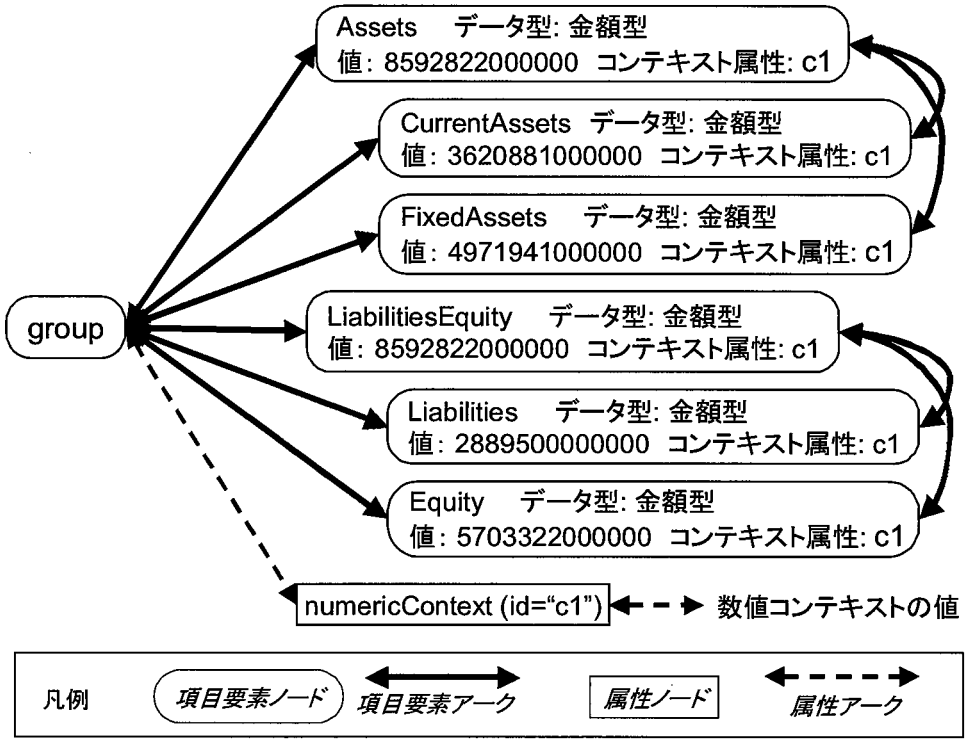


図 4.9 SDX データモデルの適用例

ただし，この例では簡単化のために numericContext の子要素の詳細（数値コンテキストの値）は省略してある．また，group 以外の項目要素ノードから数値コンテキストの値へは属性アークが存在するが，これも省略して表示している.

一方，定義リンク以外のリンクベース情報に関しては，項目要素ノードおよび項目要素アークに格納する情報を単純でわかりやすいものとする意図から，定義に含めていない．これらが用いられた場合には，従来通りリンクベースを解析することになる．また，図 4.6 のインスタンス文書の 5 行目および 18 行目

に記載されたコメントもSDXでは省略されている.

なお, 図4.5のリンクベースの記述と, それ以外とを含めて, 次の親子関係がリンクベースで定義されているとする.

- Assets（資産合計）は, CurrentAssets（流動資産）とFixedAssets（固定資産）の親である.
- LiabilitiesEquity（負債および資本合計）は, Liabilities（負債の部）とEquity（資本の部）の親である.

SDXモデルでは, DOMの各要素におけるテキストデータやコメントに関するノードや名称管理表が省略され, 項目要素ノードに集約されるので, 一般にノードの数がDOMツリーよりも少なくなり, 簡単なツリーとなると考えられる. たとえば, 図4.7および図4.9に示した同じ内容に関するDOMのみによる表現例およびSDXの適用例においては, 前者が, 数値コンテキストの値の表現部分を除いて23個のノードおよび名前管理表が用いられるのに対し, 後者では8個のノードで表現されている.

また従来定義リンクを介して参照する必要があった項目要素の親子関係を項目要素アークとして定義し, SDXモデルのみを用いて, 項目要素の走査と値の参照が可能となる. そこで, SDXモデルによれば, XBRLデータ操作に必要となるノード探索やリンクベース検索に関するプログラムの記述を短縮することが期待される.

4.5 SDXモデルの実用性の検証

SDXモデルをECMAScript言語 [EC99] において用いるためのSDXオブジェクトを定義し, SDXオブジェクトを生成するためのデータバインディング・ライブラリを試作した.

4.5.1 SDXオブジェクト

SDXオブジェクトは, ECMAScript上で定義される項目要素ノードおよびその項目要素ノードに連結する項目要素アークの情報を格納するオブジェクトである. ECMAScriptは, Webインタフェース構築に広く用いられるJavascriptの標準化仕様である. SDXオブジェクトは, 項目要素毎に生成され, 値や属性を読み書きするためのプロパティのほか, 親要素および子要素群のSDXオブジェクトを参照するプロパティを持つ（表4.2）.

表 4.2　SDX オブジェクトのプロパティ一覧

プロパティ名	意味	読み書き
name	要素名	read only
localname	ローカル名	read only
uri	名前空間 URI	read only
type	要素のデータ型	read only
value	要素の値	read/write
parent	要素の親	read/write
children	子要素の配列	read/write
context	コンテキスト属性	read/write
attribute	属性の配列	read/write

4.5.2　SDX データバインディング・ライブラリの実装

前項に示した SDX オブジェクトを事前に定義するための SDX データバインディング・ライブラリを実装した．ライブラリでは，以下のような手順で XBRL 文書を解析し，SDX オブジェクトによるツリー（SDX ツリー）を定義する．

1. インスタンス文書が参照しているタクソノミを解析する．
   ① 項目要素の定義が見つかれば，SDX オブジェクトを生成し，SDX ツリーに登録する．
   ② 項目要素の定義のなかでリンクベースへの参照が見つかれば，リンクベースの種類を判別し，ファイル名をリンクベースリストに登録する．
2. リンクベースリストのファイル名毎にリンクの解析を行う．
   ① リンク定義文が見つかれば，両端の項目要素の指定を取り出す．
   ② アークの根元側の SDX オブジェクトにおいて，先端側の SDX オブジェクトが求められるようにメソッド定義を登録する．

XBRL 関連技術と SDX データバインディング・ライブラリを含めたアプリケーション構築のアーキテクチャの概要を図 4.10 に示す．この図には，4.5.4 にて述べる専用言語も含めて記載している．

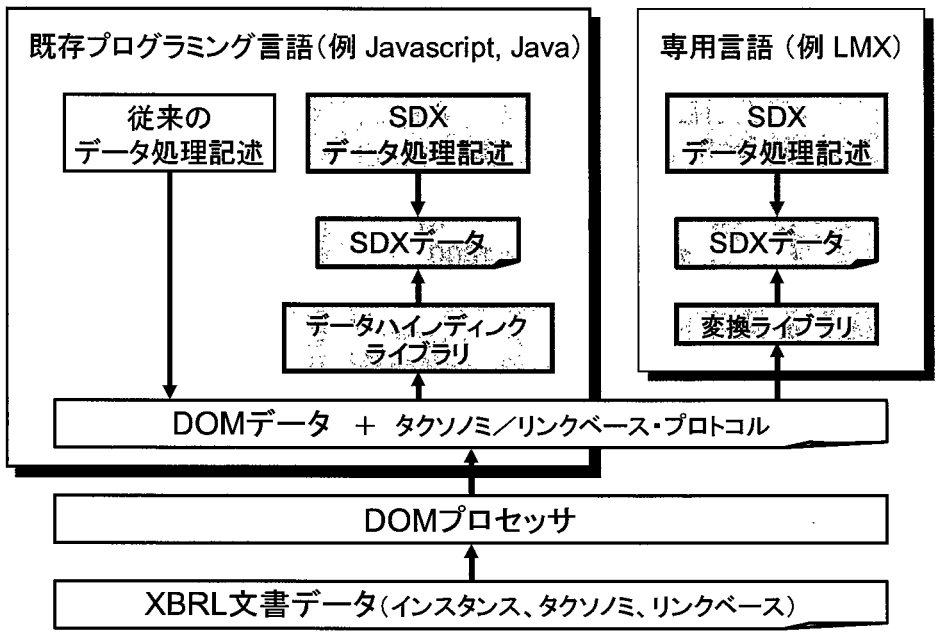


図 4.10　SDX 実現のアーキテクチャ

4.5.3 SDX オブジェクトの利用方法

SDX オブジェクトの模範的な利用例として, 金額単位の変換処理に関する記述の例を図 4.11 に示す．この例では，100 万円単位で記述された項目要素の値をすべて 1000 倍し，1000 円単位の値に変換している．実際に作成されるプログラムにおいては，特定の項目要素を選び，個々に必要とされる倍率に変換するなど多様性のある記述が想定されるが，ここでは記述形式の特徴を示すために単純な例を用いた.

```
1  // ルート要素の子を取得
2  nodes = balanceSheet.children;
3
4  // 項目要素 (groupの子ノード) の数だけループ
5  for (i=0; i<length(nodes); i++) {
6    // 処理対象の項目要素ならば
7   if (nodes[i].type == MONETARY_TYPE) {
8      nodes[i].value *= 1000;
9      // 値を1000倍に更新する
10   }
11 }
```

図 4.11　ECMAScript における SDX オブジェクトの利用例

言語記述は次のとおりである．予め，SDX ツリーの先頭の group 項目要素を表す SDX オブジェクトが変数 balanceSheet に設定されているとする．まず，group 要素の子要素列を取得する（2 行目）．そして，子要素それぞれについて繰り返し処理を行い（5 行目），要素のデータ型が金額型ならば（6 行目），要素の値を 1000 倍に書き換える（8 行目）．

この例は，4.4 節で述べた XBRL データ参照の操作手順(1)の方法(b)および操作手順(2)に当たる．DOM ツリーの場合には，アクセスしたノードが要素ノードであるか否かを判定しながら探索しなければならないが，ここではそうした記述は不要となっている．また要素の値を参照する際には項目のノードから別のノードを辿る必要があり，そのたびに値ノードであるかを判定しなければならないが，ここではそうした記述も不要となっている．

なお，ECMAScript は DOM インタフェースを有するので，同時に DOM を用いて XBRL データ参照の操作手順(1)の方法(a)を記述することも可能であり，また XLink インタフェースを用いて方法(c)を記述することも可能である．

4.5.4　SDX モデルの実用性

前項に示した通り，SDX データバインディング・ライブラリを用いることによって，XBRL データ処理のなかで中心となる参照処理のうち，頻度の高いと思われる 2 つの方法のなかで一つの記述を簡略化することができた．他の参照

方法や他のXBRLデータ処理に関しては，従来通りDOMやXLinkのインタフェースを用いることができる．

こうしたライブラリによってSDXデータモデルを提供する方式は，会計業務担当者のなかでも，一つの汎用プログラミング言語を知ってはいるがXML関連技術に関する知識を持たないような利用者に対して有効である．本研究ではECMAScript言語向きのライブラリを実現したが，オブジェクトの概念を持つ言語であれば，他の汎用プログラミング言語に対しても4.5.2に示した方式によってSDXオブジェクトを提供することが可能と考えられる．

なお，SDXデータバインディング・ライブラリは，データ参照の実行に先立って起動し，SDXツリーを生成しておくことが利用条件となる．そのため，いったん生成したSDXツリーを保存管理して利用するか，もしくはデータ参照プログラム起動時に毎回生成する必要がある．

同じ会計業務担当者のなかでも，プログラミングの経験がほとんどなく，財務データ以外ではデータ処理を記述することがないような利用者においては，SDXデータモデルに特化した簡易言語を提供することも選択肢となる．LMX [TA03b] [TA04] は，SDXオブジェクトの機能のほか，XBRLに関するDOMの利用やリンクベースの参照に関する機能を含めた簡易言語として提案されている．

他方，DOMやXLinkに熟練し，かつそれらの記述性に満足しているXML専門のプログラマにおいては，SDXモデルの有効性は比較的小さくなると思われる．DOM APIを用いて生成，更新，削除などを記述する場合には，両者を矛盾なく動作させることに注意すべきである．そのため，SDXおよびDOMのノード間でデータの更新を相互に交信するなどデータモデル間の連携を強化して利用者に提供する必要がある．

4.6 まとめ

本研究では，XBRL文書に記述する勘定項目に着目したデータモデルとしてSDXを提案した．また，SDXモデルに基づくデータアクセス方法を汎用プログラミング言語に提供するためのSDXデータバインディング・ライブラリを開発し，実用性を検証した．

SDXモデルに基づくXBRLデータ参照の記述においては，DOMデータモデルなどXMLに関連する技術的な知識が不要となり，またノード探索を大幅に省略できるので記述を簡略化することができる．

財務報告情報は，企業を取り巻く様々な機関，関係者によって利用されるが，その利用者のほとんどは，XML関連の技術知識を有していないと考えられる．

SDX モデルによって，これらの利用者にわかりやすいデータ参照方法を提供することにより，財務報告情報を用いた財務分析や投資活動に関する情報環境の整備を促進することが期待される.

第 5 章 分析業務に関する知識を用いた財務分析支援方式

5.1 はじめに

企業に対する投資や取引の実施を判断するうえで，財務指標に基づく財務分析は最も基本的な手段の 1 つである．財務分析は，企業における会計や金融の専門業務担当者ばかりでなく，近年は個人投資家など一般市民にとっても必要な作業となっている．

財務指標は，企業が発行する財務諸表に記載される項目値を用いた計算式で定義されており，機械的に計算することができる [YU07b]．しかし，指標値をもとに財務状態を判断するには，業種における事業特性や財務戦略などに対する専門知識をもとにした多面的な検証が必要であり，専門知識を持たない個人投資家などにおいては有効な財務分析が困難となっている．そこで，本研究では財務分析の専門家から手順や判定方法を取得し，個人投資家などにおいても専門家と同様のプロセスで財務分析を可能とすることを目指して，支援ツールを開発した．図 5.1 に，第 4 章で扱った業務担当者向けフレームワークと本章で扱う一般利用者向けフレームワークの関係を示す．

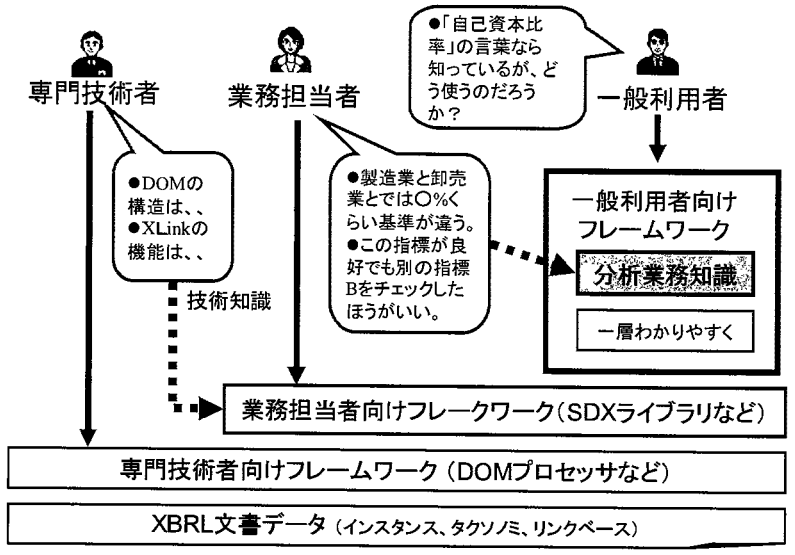


図 5.1　業務担当者向けフレームワークと一般利用者向けフレームワーク

以降，5.2 節では財務分析業務と従来の支援技術について概説する．5.3 節では財務分析の活動プロセスに基づく支援ツール FIAT について述べる．

5.2　財務分析の業務と支援

5.2.1 財務諸表と財務指標

企業は，定期的に自らの財務情報を貸借対照表，損益計算書などの財務諸表にまとめて公表する [YU07b]．貸借対照表の例を図 5.2 に示す．財務諸表の金額をもとに企業特性を分析する指標を財務指標と呼ぶ．たとえば，「自己資本比率」は安定性を評価する指標の一つであり，図 5.2 にある「＜資本の部＞」および「＜資産の部＞」という項目値から，「＜資本の部＞÷＜資産の部＞×100%」という計算式で求めることができる．

株式会社○○○　貸借対照表　(単位:百万円)

項目	金額	項目	金額
＜資産の部＞	8,592,822	＜負債の部＞	6,439,500
流動資産	3,620,881	流動負債	3,090,821
現預金及び有価証券	1,487,544	買掛債務	765,041
売掛債権	919,468	社債及び短期借入金	550,000
商品・製品	140,516	その他	1,775,780
その他	505,277	固定負債	3,348,679
固定資産	4,971,941	社債及び長期借入金	2,000,600
有形固定資産	1,269,042	その他	1,348,079
無形固定資産	1,242,883	＜資本の部＞	2,153,322
投資その他の資産	2,460,016	資本金	397,049
		資本剰余金	416,970
		利益剰余金	1,737,602
		その他	△398,299
資産合計	8,592,822	負債及び資本合計	8,592,822

図 5.2　貸借対照表の例

5.2.2 財務分析業務の課題と従来の支援技術

機関投資家，企業の財務担当者など会計専門家における財務分析業務では，Web などから企業の財務諸表を入手し，財務指標を計算しながら企業の財務状態を分析していく．従来，Web 上での財務諸表データは企業毎に異なる形式となっていたが，現在 XBRL [XB00] による標準化が提案されており，XBRL に基づくデータ取得の自動化が図られている [YU04]．わが国の金融庁においては，XBRL を採用した財務諸表の開示システムを平成 20 年 4 月に稼動予定であり [KI07a]，上場企業においては XBRL による財務諸表が作成されることになる．

ただし，XBRL は XML [WW96] を拡張した技術者向け言語であり，会計専門家が財務指標の計算式を記述するには必ずしも適していない．これに対しては，会計専門家に理解容易な言語で XBRL を内部に隠蔽するアプローチが考えられる．その一例として，LMX [TA03b] が提案されており，計算式記述の容易化が期待される．

一方，個人投資家など一般利用者には，代表的な財務指標について表やグラフで表示するサービスが，株式投資情報サイト [EK07] やオンライントレード・サイトなどで提供されている．これによれば，利用者は財務諸表の項目値の取得や計算に煩わされることなく企業の財務状態の概要を知ることができる．

しかし，個々の指標値をいかに評価し投資等の最終的な判断につなげていくかについては十分に支援されず，利用者の知識に依存していた．たとえば，株

式投資の検討に際しては一般に株価と資産および利益の比率が注目されるが，より確実なリターンを期待する個人投資家においては，資産や利益の増減の根拠についても関心を払うものも少なくない．これに対し，財務指標値から企業の特性に応じた分析を経て投資等の判断の根拠を確立する手順については，会計監査人や機関投資家など専門家の知識としてのみ存在し，一般にはほとんど知られていなかった．

5.2.3 財務指標評価の多様性

財務指標として計算された値は一般的な意味での企業の財務特性を示しているが，投資や取引などビジネスの判断のためには，その値だけでは必ずしも十分な情報とは言えない．まず，下記のような企業の特性や評価目的の違いによって水準が異なることを考慮する必要がある．

(1) **業種による違い**

たとえば，製造業は比較的設備投資の費用やリスクが高いので，安定性指標に関して卸売業や金融業のそれと比べて高い水準が期待される．

(2) **評価目的の違い**

企業の株式購入や融資を判断するうえでは，中長期的な安定性を重視するので，個別の取引を判断する場合よりも高い水準で判定する．

また，財務指標はあくまで一面的な測定である．そこで，財務特性を判定するためには，関連した財務指標をあわせて測定し，多面的に分析することが望ましい．たとえば，自己資本比率の値が「安定性が高い」ことを示した場合にも，それが本業における良好な収益に基づくものであり，財務的に取り繕われたものではないことを検証すれば，評価をさらに確信することができる．そこで，本業以外の利益の比重を評価する「特別利益比率」や，短期的な負債状況を評価する「流動負債比率」をあわせて測定することがある．

5.3 財務分析支援ツール FIAT

5.3.1 財務分析の活動モデル

ここでは，前節で述べた財務分析の課題に対処するために，図 5.3 に示す一般利用者向けの活動プロセスを設定した．活動プロセスにおいては，会計専門家の知識（財務分析知識），特に投資などの判断に至るまでの分析手順に関する知識を参照し，業種と評価目的の多様性に対応することを方針とした．また，財務分析知識の表現と対話方法をできるだけ単純化し，知識を拡充すれば一般利

用者がそれを受け容れて効果をあげやすいように配慮した.

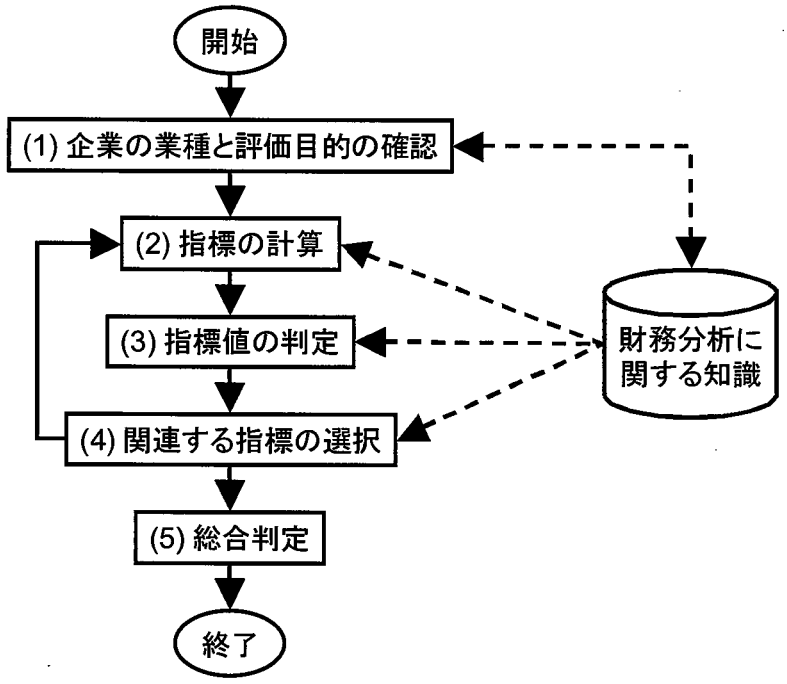


図 5.3 財務分析の活動プロセスの概要

活動プロセス（以下，プロセスと略）(1)では，対象企業の業種と評価の目的を確認する．業種と目的の違いによって，評価基準などの知識を使い分ける．

プロセス(2)では指標を計算し，プロセス(3)では計算された値を判定基準と比較して，その指標に関する財務特性を判定する．さらにプロセス(4)では財務分析知識を参照して関連する指標を選択し，プロセス(2)に戻って計算と評価を繰り返す．この繰り返しによって，基本的な指標の測定から業種や評価目的に即した分析を深めていく．最後にプロセス(5)では，それまでの指標判定に基づいて最終的な判定を行う．

5.3.2 FIAT の構成

上記の活動プロセスに基づく知識利用型の財務分析支援ツール FIAT （A Financial Analysis Assistance Tool）[IZ05a] [IZ05b] を開発した．

FIAT では，さまざまな企業の財務諸表のデータを統一した形式で扱うため，

財務情報の標準化言語である XBRL を採用している．また財務指標の計算式および基準に基づく判定などを記述するために，会計専門家向け XBRL 処理言語である LMX を利用する．このほか，FIAT は図 5.4 に示す要素から構成される．

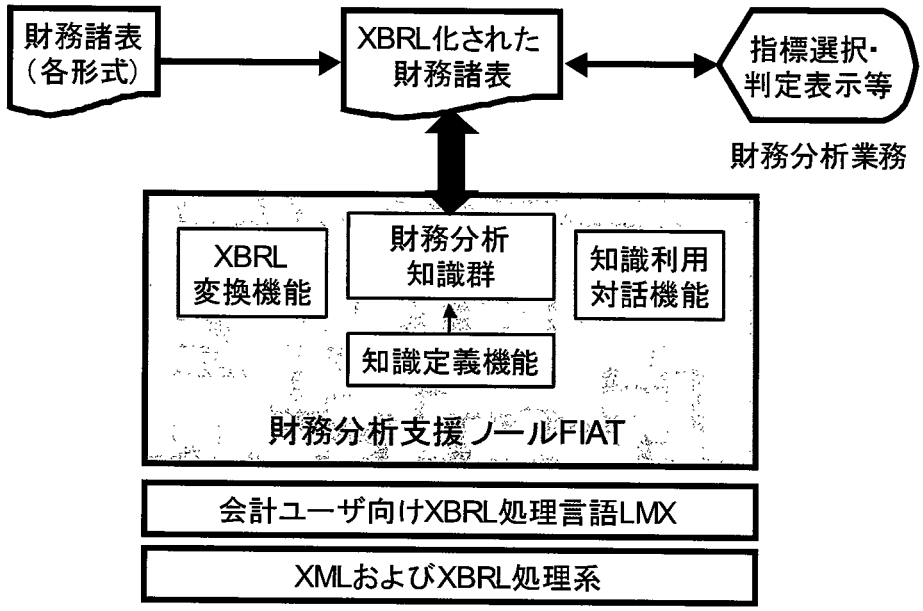


図 5.4　財務分析支援ツール FIAT の概要

(1) **XBRL 入力変換機能**

XBRLによって記述された財務諸表を読み込む機能，もしくはCSV(Comma Separated Value) などによる財務諸表を XBRL に変換する機能である．これにより，XBRL による財務諸表だけではなく，CSV 等で蓄積されている過去の財務諸表も入力可能となる．

(2) **財務分析知識群**

財務分析知識は，財務指標を単位として記述される．記述される要素には，指標の計算式定義のほか，業種と分析目的に対応する判定基準，判定結果に関する説明文，および判定に関連する他の財務指標を含む．

判定基準は，企業が属する業種や評価の目的による違いを含めて記述される．関連指標は，財務分析の専門家が実際に融資等のための財務分析を進める手順を反映させる．判定結果は基準を満たすか（真），もしくは不足であるか（偽）で表すので，説明文および関連指標は，真の場合と偽の場合の両方について記述される．なお，財務分析知識の形式の定義と例の作成にあたっては，会計研

究者および公認会計士など会計専門家へのヒアリングをもとに行った.

(3) 知識定義機能

会計専門家が, 計算機画面から財務分析知識を入力し, 保存する機能である. 図 5.5 に入力画面を示す.

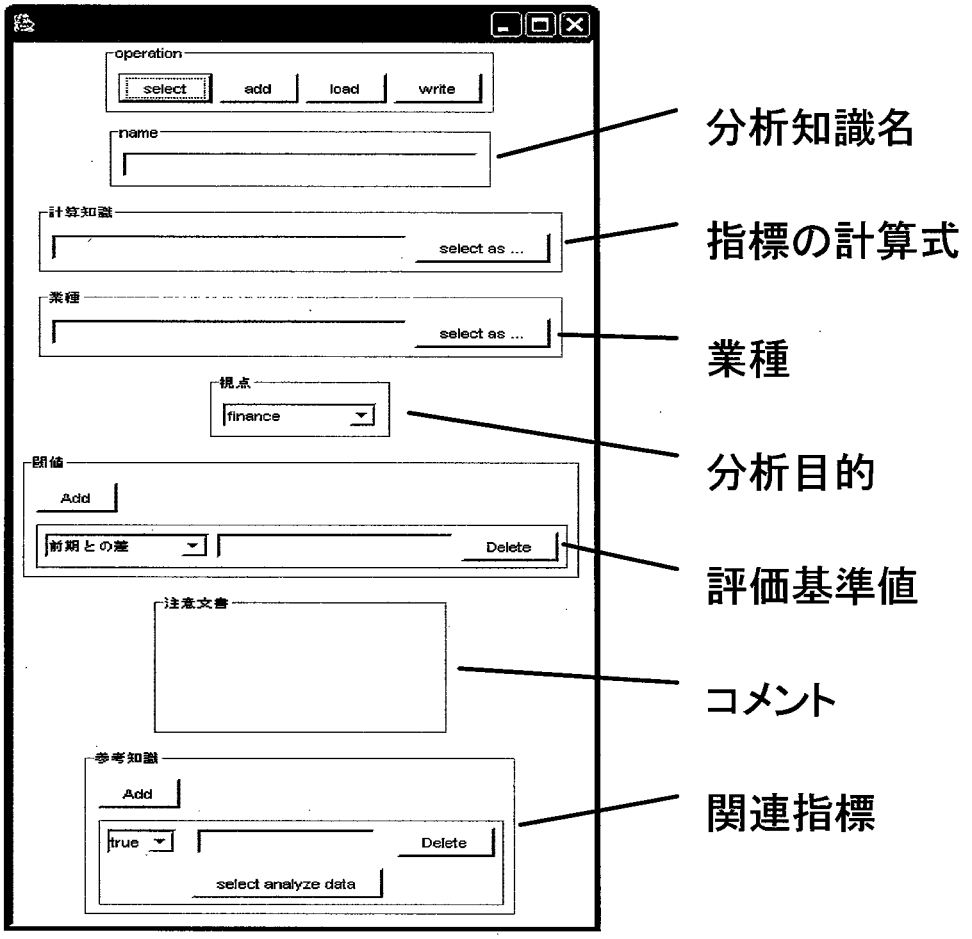


図 5.5　FIAT の知識入力画面

(4) 知識利用対話機能

利用者が, 財務分析知識を呼び出して対話的に財務分析を行う機能である.

5.3.3　FIAT の動作

ある企業 A 社との取引開始に際しての安定性評価を行う場面を例として,

FIAT の動作を説明する．

一般利用者が FIAT を使用する前に，会計専門家が図 5.5 の入力画面から財務分析知識群を入力する．ここでは，財務指標「自己資本比率」に関して，図 5.6 に示す内容の財務分析知識を入力したとする．また XBRL で記述された A 社の財務諸表を入力しておく．

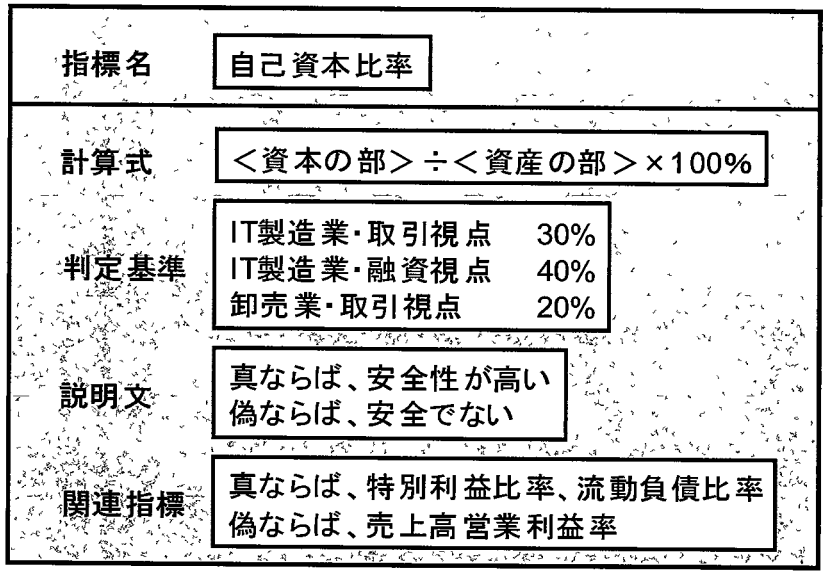


図 5.6　FIAT における財務分析知識の例

利用者は，図 5.3 の手順に従い財務分析を進める．まず，A 社が IT 製造業に属し，取引視点での評価を行うということを入力し，財務指標として自己資本比率を選択する（プロセス(1)，以下同様に図 5.3 参照）．すると，FIAT が財務分析知識の計算式を用いて自己資本比率を算出する（プロセス(2)）．算出値は 25%であり，財務分析知識に記されている判定基準 30%に達していないので，図 5.7 の画面例にあるように，「偽」である場合の説明文である「安全でない」を出力する（プロセス(3)）．

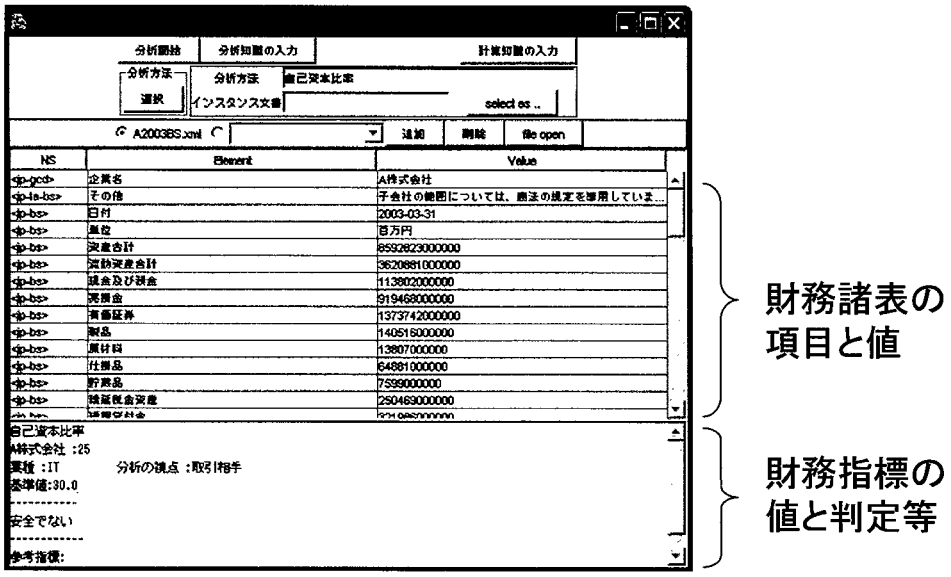


図 5.7　FIAT の出力例

そこで，利用者は取引開始に向けて否定的な根拠を一つ得たことになるが，さらに調査すべき関連指標が示されているのでそれらを評価していく（プロセス(4)）．図 5.7 の画面の下部をスクロールすると「売上高営業利益率」が表示されることになる．自己資本比率が基準に達しない場合には，これを評価し，こちらが基準に達すれば自己資本比率に関する不足を補うと解釈することが可能な場合もある．利用者における当の場面においては，関連指標として提供される会計専門家の経験的知識の背景とは異なる条件が存在することも想定される．そこで，偽である場合の関連指標に関しては，利用者自身が適用を判断し，総合判定を行うようにしている（プロセス(5)）．

上記の実行例とは逆に「安全性が高い」と判定された場合には，「真」の場合の関連指標である「特別利益比率」および「流動負債比率」を参照することになる．

これら関連指標に対する基準を上回るごとに取引開始への確信度を高め，関連指標の知識に記載されているその先の関連指標も含めてすべての基準を上回ると，取引開始判断の根拠を確立したことになる．

5.4　おわりに

財務指標の計算と判定を組み合わせて企業の財務分析を行うプロセスを考案し，そのプロセスに基づく支援ツール FIAT を開発した．

従来財務分析作業の支援に用いられていたツール，例えば表計算ソフトやオンライントレード・サイトのグラフ表示においては，所定の財務指標値を一括して表示することができる．これに対し FIAT では，複数の財務指標値を関連する順序に従って対話的に表示し，一般利用者に対して，対象企業の特性や評価目的との関連性について理解を促しながら，投資や取引等の判断に至るプロセスを支援することを可能とした．

また FIAT では，XBRL および LMX を基盤として導入した．これにより，財務諸表データの入力と処理を統一した方式で実現し，少なくとも平成 20 年 4 月以降のわが国の上場会社における財務諸表データをすべて適用可能とし，過去のデータや海外企業のデータに対しても適用性を高めた．

現在 FIAT は，基本機能の実装を完了し，試用を開始している．FIAT の操作は簡潔であり，順々に関連指標の調査が誘導される分析作業の形態についても違和感はないという評価を得ている．しかし，対話内容に関しては，利用者によって求める知識の範囲や詳細化レベルに異なり，満足が得られない場合も見受けられた．財務分析知識の拡充が今後の課題となっている．

第6章　むすび

6.1 まとめ

第 1 章にて提起したフレームワークに基づく継続的開発の実用化課題に関して，第 2 章から第 5 章にかけて具体的な研究成果を述べた．この節ではそれらの成果から得られる結論をまとめ，本題の実用化に向けて前進した事項を確認する．

第 2 章では，従来のクラス定義情報からだけではなく，イベントトレース図に含まれる情報も用いて，オブジェクト指向型ソフトウェアのファンクションポイントを詳細に計測する手順とツールを開発した．ある要求仕様定義ツールで作成された 3 つの事例について熟練者による計測と比較したところ，誤差は 20%程度に収まっていた．イベントトレース図はオブジェクト指向型ソフトウェアの設計仕様書において広く利用される書式であるので，本研究で提案する方式は，フレームワーク適用の情報システムをはじめとするオブジェクト指向型ソフトウェアの開発規模の早期見積りに有効であると考えられる．

第 3 章では，オブジェクト指向向けメトリクスとして参照されることが多い OOFP および C&K メトリクスを用いて，あるアプリケーションフレームワークを利用した場合と利用しなかった場合について，システム開発の機能量と複雑性を計測した．4 つの機能の開発に関して計測を行ったところ，フレームワークを利用した場合における機能量の削減効果は明らかで，3 つ目の機能の開発を終えたところでフレームワークの準備に必要な機能量の投資を回収するほどの効果をもたらすことが計測された．ただし，フレームワークと開発する機能との相性が十分ではない場合には削減効果が少なく，理想的なフレームワークを準備することの難しさが同時に分析された．複雑性に関してはフレームワークを利用した場合の方が高い数値が計測される部分があったが，それらはフレームワークの固定部分と変化部分の境界において発生する値に起因しており，固定部分の品質が十分高ければシステム全体の複雑性には問題が少ないと分析された．

以上の研究により，情報システムの継続的開発においてフレームワークを適用する場合の効果の予測や評価に関して，計測精度の向上を図る方法の一つと，

計測の結果および分析の例を示すことができた.

第 4 章では, 財務報告データを参照するシステムという分野に限定したシステム開発に対して, 専用のフレームワークを実現した. 財務報告情報記述の標準化言語である XBRL においては, DOM や XLink など XML に関連する新技術が導入されているため, 実際に財務報告データ処理を記述する会計業務担当者には理解が容易ではない. そこで, これら技術要素を隠蔽したデータモデル SDX (Simple Data Model for XBRL Documents) を設定し, SDX に基づく会計業務担当者用のフレームワークを開発した. このフレームワークには, データバインディング, ライブラリならびに専用言語を含んでおり, これらによって財務報告データアクセスのプログラム記述を簡単化することができた. 本研究のフレームワークおよびデータバイディングの考え方は日立グループの製品 [HS04] にも適用され, これを通じて海外を含む先進的な会計業務担当者における財務情報サプライチェーンの構築に貢献することができた.

第 5 章では, 同じく XBRL データをアクセスするシステムにおいて, 一般利用者向け対話環境のフレームワークを FIAT (A Financial Analysis Assistance Tool) というツールとして実現した. 個人投資家などの一般利用者においては, 技術知識のみならず業務知識も乏しい場合が多い. そこで, 会計学の研究者および公認会計士との共同作業により, 財務報告データを用いて企業への投資などを判断していく分析手順や判断基準をモデル化した. そして FIAT においては, 機関投資家など専門家が財務分析業務の知識を入力し, 一般利用者がそれらを参照して財務分析を行えるようにした.

以上の研究により, 業務分野を限定したフレームワークにおいて, 業務専門家および一般利用者に対して求められる機能とその実現に関して, XBRL 財務報告データ処理を題材として方式の一例を示すことができた.

6.2 今後の研究方針

情報システムのライフサイクルにわたる計画, 管理をより合理的に進めるためには, 保守および開発の生産性に関する尺度を明確にするとともに, 生産性に関する尺度と, 情報システムが目的とする業務プロセスの改善, 顧客への貢献あるいは財務的業績などの目標, 実績と関係付けたマネジメントが期待される. 1.3.3 項において言及した EA においては, アーキテクチャの一要素としてパフォーマンス評価の体系を掲げて, 業務等への貢献も含めたシステム評価の重要性を主張しているが, いまだ十分に具体化と普及が進められている状況にあるとはいえない. 本研究の機能規模および複雑性評価を一つの基盤として,

情報システム・マネジメントのための尺度の体系に関する研究を継続していきたいと考えている.

企業をとりまく経営環境は年々変化を加速しており，情報システムを常時改編／改善していく継続的開発への期待は一層高まっている．本研究で一例として示したXBRLに関するフレームワークの考え方や，XML関連技術の取り扱いを含めたソフトウェア構成法および設計経験を，他の業務分野向けのフレームワーク構築にも生かしていきたいと考えている．特に，1.3.3項において述べた通り XML をベースとするビジネスデータの標準化は多くの業務分野で進行しており，XBRL において先行して明らかにした成果や経験が生かされる機会は多いと期待している.

企業の情報システム部門や情報システムベンダにおいては，継続的開発のためのフレームワークの整備は重要な課題と認識されているものの，現実には十分な取り組みがなされていない場合が少なくない．フレームワークの整備には，設計能力と企業としての戦略的な取り組みが必要とされる．そこで，現状を打破するポイントの一つは，フレームワークを計画・普及させていくリーダーシップを有する技術者の育成にあると感じている．報告者はこれまで自社および他社の技術者への社会人教育に関わってきたが，この春からは大学に職を得て情報システム教育の実践に当たる．ビジネスの要求と技術の特徴の両面に対して柔軟性の高いフレームワークを設計する次世代の技術者を育成することを目標とし，その育成方法の確立に向けて研鑽を積んでいきたいと考えている.

参考文献

[AL79] A. J. Albrecht, "Measuring Application Development Productivity", Proc. Joint SHARE, GUIDE, and IBM Application Development Symposium, pp. 83-92 (1979).

[AL94] A. J. Albrecht, "Function point analysis", in J. J. Marciniak, editor, Encyclopedia of Software Engineering, Vol. 1, John Wiley & Sons, pp. 518-524 (1994).

[AN99] G. Antoniol, C. Loken, G. Caldiera, R. Fiutem: "A function point-like measure for object-oriented software", Empirical Software Engineering, Vol. 4, No.3, pp.263-287 (1999).

[AN03] G. Antoniol, R. Fiutem, C. Loken: "Object-Oriented Function Points: An Empirical Validation", Empirical Software Engineering, Vol.8, No.3, pp. 225-254 (2003).

[AP91] Apple Computer Inc.: Inside Macintosh Technical Library (1991).

[BA92] V. R. Basili, G. Caldiera, F. McGarry, R. Pajersky, G. Page and S. Waligora: "The software engineering laboratory – an operational software experience", Proc. of ICSE14, pp.370-381 (1992).

[BA96] V. R. Basili, L. C. Briand and W. L. Melo: "A validation of object-oriented design metrics as quality indicators", IEEE Trans. on Software Eng. , Vol.20, No.22, pp.751-761 (1996).

[BE00] K. Beck（長瀬他訳）： XP エクストリーム プログラミング入門, ピアソン・エデュケーション (2000).

[BL06] S. Blackburn, et al.: "The Dacapo benchmarks: Java Bencharking Development and Analysis", ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), pp.169-190 (2006).

[BRA94] C. Braun: Reuse, in J. J. Marciniak, editor, Encyclopedia of Software Engineering, Vol.2, John Wiley & Sons, pp.1055-1069 (1994).

[BRI98] L. C. Briand, J. Daly, V. Porter, J. Wust: "Predicting fault-prone classes with design measures in object-oriented systems", Proc. of the Ninth International Symposium on Software Reliability Engineering, pp.334-343 (1998).

[BO94] G. Booch: Object-Oriented Analysis and Design with Applications, The Benjamin/Cummings Publishing (1994).

[CA98] G. Caldiera, G. Antoniol, R. Fiutem and C. Lokan: "Definition and experimental evaluation of function points for object-oriented systems", IEEE, Proc. of METRICS98, pp.167-178 (1998).

[CAV02] C. Cavaness: Programming Jakarta Struts, Oreilly & Associates (2002).

[CHE76] P. Chen: "The Entity-Relationship Model - Toward a Unified View of Data", ACM Transactions on Database Systems (TODS), Vol.1, No.1, S. 9-36 (1976).

[CHA93] R Chandra, A Segev: "Managing Temporal Financial Data in an Extensible Database", Proc. of VLDB 1993, pp.302-313 (1993).

[CHI94] S. R. Chidamber and C. F. Kemerer: "A metrics suite for object-oriented design", IEEE Trans. on Software Eng., Vol.20, No.6, pp.476-493 (1994).

[DA03] B. Daum : Professional Eclipse 3 for Java Developers, Wrox Pr Inc (2003).

[DI75] E. ダイクストラ（野下訳）： 構造化プログラミング，サイエンス社(1975).

[EC99] Ecma International: "ECMA Script Language Specification", http://www.ecma-international.org/publications/standards/Ecma-262.htm (1999).

[EK07] e 株！： http://quants.interactive-tokyo.co.jp, (参照 2007/10/12).

[FE04] R. Ferec, I. Siket, T. Gyimothy: "Extracting Facts from Open Software", ICSM2004: Proc. of 20th IEEE International Conference on Software Maintenance, pp.60-69 (2004).

[FO94] C. Foloutsos, M. Ranganathan, Y. Manolopoulos: "Fast Subsequence Matching in Time-Series Databases", Proc. of SIGMOD 1994, pp.419-429 (1994).

[FP99] International Swaps and Derivatives Association FpML Working Groups: FpML, http://www.fpml.org/ (1999).

[GO83a] A. Goldberg: Smalltalk-80: The Interactive Programming Environment, Addison-Wesley (1983).

[GO83b] A. Goldberg, D. Robson: Smalltalk-80: The Interactive Language, Addison-Wesley (1983).

[HA02] 春日史朗, 鳥海幸輝, 坂田哲夫, 小林伸幸, 芳西 崇： “電子商取引における XML 検索・変換の最適化方式”, データ工学ワークショップ 2002 論文集 A3-4 (2002).

[HR01] The HR-XML Consortium: HR-XML, http://www.hr-xml.org/ (2001).

[HS04] 日立システムアンドサービス XiRUTE, http://www.hitachi-system.co.jp/xbrl/.

[IF94] IFPUG, Function Point Counting Practices Manual, International Function Points Users Group (1994).

[IP07] 情報処理推進機構 ソフトウェアエンジニア： 共通フレーム 2007 (2007).

[IS92] S. Isola: “Experience report on a software reuse project: Its structure, activities, and statistical results”, Proc. of ICSE14, pp.320-326 (1992).

[ISO02] International Organization for Standardization: ISO/IEC 12207 (2002).

[ISO06] International Organization for Standardization: ISO/IEC 14764 (2006).

[IT03] International Press Telecommunications Council: NewsXML, http://iptc.org/ (2003).

[IZ05a] 泉田聡介, 松下 誠, 井上克郎, 湯浦克彦： “記述言語 XBRL で書かれた財務諸表を対象とした分析支援ツールの試作”, 電子情報通信学会技術研究報告 SS-2004-65, Vol.104, No.723, pp.7-11 (2005).

[IZ05b] 泉田聡介： “言語 XBRL で書かれた財務諸表の分析支援ツールの試作”, 大阪大学大学院情報科学研究科修士論文 (2005).

[JA97] I. Jacobson, M. Griss and P. Jacobson: Software Reuse – Architecture Process and Organization for Business Success, Addison-Wesley (1997).

[JEG01] J. J. Jeng: “An approach to designing reusable service frameworks via virtual service machine”, Proc. of SSR, pp.58-66 (2001).

[JES99] C. S. Jensen, R. T. Snodgrass: "Temporal Data Management", IEEE Trans. on Knowledge and Data Engineering, Vol.11, No.1, pp.36-44 (1999).

[JIS07a] 日本規格協会： JIS X0160:2007 ソフトウェア・ライフサイクル・プロセス (2007).

[JIS07b] 日本規格協会： JIS X0161:2007 ソフトウェア保守 (2007).

[JU07] 日本情報システム・ユーザ協会 (JUAS)： 企業ＩＴ動向調査 2007 (2007).

[KA99a] T. Kamiya, S. Kusumoto, K. Inoue, Y. Mohri: "Empirical evaluation of reuse sentiveness of complexity metrics", Information and Software Technology, Vol. 41, No.5, pp.297-305 (1999).

[KA99b] T. Kamiya, S. Kusumoto, K. Inoue: "Prediction of fault-proneness at early phase in object-oriented development", Proc. of Second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 253-258 (1999).

[KE99] B. Keepence and M. Mannon: "Using patterns to model variability in product families", IEEE Software, Vol.16, No.4, pp.102-108 (1999).

[KR99] P. Kruchten （藤井他訳）： ラショナル統一プロセス入門, ピアソン・エデュケーション (1999).

[KIN07a] 金融庁： "EDINET 再構築に伴うパイロットプログラム実施のご案内", http://www.fsa.go.jp/singi/edinet/20070403.html (2007).

[KIN07b] 金融庁総務企画局企業化開示課： "EDINET タクソノミの概要", http://www.fsa.go.jp/singi/edinet/20070403/01.pdf (2007).

[KIN07c] 金融庁： "EDINET の高度化に関する協議会 実務者検討会 公開資料", http://www.fsa.go.jp/singi/edinet, (参照 2007/07/12) (2007).

[KI97] B. A. Kitchenham: "The Problem with Function Points", IEEE Software, Vol. 14, No.2, pp.29-31 (1997).

[KZ07] 経済産業省： 平成 18 年度情報処理実態調査 (2007).

[LE98] OASIS (Organization for the Advancement of Structured Information Standards) LegalXML Member Section: LegalXML, http://www.legalxml.org/ (1998).

[LO99] C. J. Lokan: "An empirical study of the correlations between function point elements", Proc. of the Sixth International Symposium on Software Metrics, pp. 200-206 (1999).

[LOW90] G. C. Low and D. R. Jeffery: "Function Points in Estimation and Evaluation of the Software Process", IEEE Trans. Software Engineering, Vol. 16, No. 1, pp. 64-71 (1990).

[NA96] 中村 永： "科学技術計算とリアルタイム制御に向くソフト計測手法"，日経エレクトロニクス，No. 658, pp. 175-185 (1996).

[NI94] 西山 茂： "ソフトウェア規模見積り技術の最近の流れ－行数による評価から機能量による評価へ"，情報処理, Vol. 35, No.4, pp.289-298 (1994).

[MA01] J. Madhavan, P. Bernstein, E. Rahm: "Generic Schema Matching with Cupid", Proc. of VLDB 2001, pp.49-58 (2001).

[MC02] K. McArthur, H. Saiedien, M. Zand: "An evaluation of the impact of component-based architectures on software reusability", information and technology, Vol.44, No.6, pp.331-393 (2002).

[OG00] Office of Government (UK): IT Infrastructure Library –Service Support (2000).

[OG01] Office of Government (UK): IT Infrastructure Library –Service Delivery (2001).

[OO05] 大河原俊明, 田中淳一, 森嶋厚行, 杉本重雄: "データ変換のためのスキーママッチング支援機構"，データ工学ワークショップ DEWS2005, 2C-i3 (2005).

[RA97] Rational Software Corporation: UML Notation Guide, version 1.1 (1997).

[RO99] L. Rosenberg: "Applying and Interpreting Object Oriented Metrics", Sixth International Symposium on Software Metrics, Measurement for Object-Oriented Software Projects Workshop (1999).

[RU91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen: Object Oriented Modeling and Design, Prentice Hall (1991).

[SA95] 斎藤美帆, 湯浦克彦, 大成宣行, 亀田達也："オブジェクト指向による要求仕様書視覚化ツール REQUARIO"，日立評論, Vol.77, No.12, pp. 15-18 (1995).

[SO07] ソフトウェア・メインテナンス研究会：ISO14764 によるソフトウェア保守開発, ソフトリサーチセンター (2007).

[SY91] C. Symons: Software Sizing and Estimating, John Wiley & Sons (1991).

[TA03a] 高尾祐治, 渡辺貴史, 松下 誠, 井上克郎, 湯浦克彦： “項目間の対応関係を用いた XBRL 財務報告書自動変換手法の提案”， ソフトウェア・シンポジウム 2003 論文集, pp.149-158, (2003).

[TA03b] 高尾祐治, 松下 誠, 井上克郎, 湯浦克彦： “XBRL で記述された財務データを扱う言語処理系の提案”， 電子情報通信学会技術研究報告, SS-2003-21, KBSE-2003-24, Vol.103, No.481, pp.31-36 (2003).

[TA04] 高尾祐治: “記述言語 XBRL で定義された財務諸表を計算・書式変換する言語処理系の提案と実現”, 大阪大学大学院情報科学研究科修士論文 (2004).

[TO02] 鳥海幸輝, 春日史朗, 坂田哲夫, 小林伸幸, 芳西 崇：“情報流通分野における XML 変換方式の研究”, データ工学ワークショップ 2002 論文集 A6-3 (2002).

[UE99] 上村拓也, 柏本隆志, 楠本真二, 井上克郎：“UML で記述された設計仕様書からのファンクションポイント計測手法”, 1999 年電子情報通信学会総合大会 情報・システム講演論文集 1, D-3-9 (1999).

[VE97] ヴェストソフトウェア：SAVER を利用したファンクションポイント法の実現 (1997).

[VI97] VIASOFT: Visual Recap Quality Management (1997).

[WW96] The World Wide Web Consortium: “Extensible Markup Language (XML)”, http://www.w3.org/XML (1996).

[WW97] The World Wide Web Consortium: “Document Object Model (DOM)”, http://www.w3.org/DOM (1997).

[WW01a] The World Wide Web Consortium: “XML Schema”, http:// www.w3.org/XML/Schema (2001).

[WW01b] The World Wide Web Consortium: “XML Linkink Language (XLink)”, http://www.w3.org/TR/xlink (2001).

[WU04] H Wu, B Salzberg, D Zhang: “Online Event-driven Subsequence Matching over Financial Data Streams”, Proc. of SIGMOD 2004, pp.23-34 (2004).

[XB00] XBRL International: XBRL Specifications,

http://www.xbrl.org/Specifications, (参照 2007/07/12) (2000).

[YU02a] 湯浦克彦, 大坪稔房, 団野博文, 石井義明, 古澤憲一, 桐越信一, 鈴木文音：「EJB コンポーネントによる Web システム構築技法」の 3.2.3 項「要求仕様視覚化ツール REQUARIO」, ソフトリサーチセンター (2002).

[YU02b] 湯浦克彦, 大坪稔房, 団野博文, 石井義明, 古澤憲一, 桐越信一, 鈴木文音：「EJB コンポーネントによる Web システム構築技法」の第 4 章「画面遷移フレームワーク」, ソフトリサーチセンター (2002).

[YU04] 湯浦克彦： インターネット財務情報システム, ソフトリサーチセンター, (2004).

[YU05] 湯浦克彦: 実践エンタープライズ・アーキテクチャ, ソフトリサーチセンター, (2005).

[YU06a] 湯浦克彦： IT ガバナンスの構造, エスアイビー・アクセス (2006).

[YU06b] 湯浦克彦, 染谷英雄, 河辺亮二： IT コンサルタントのための会計知識, ソフトリサーチセンター (2006).

[ZA99] S. Zamir: Handbook of Object Technology, CRC Press, Boca Raton (1999).