第 1 章

研究の背景

オブジェクト指向によるソフトウェア開発は多くの企業で導入され始め，本格的なシステム開発に適用されようとしている．1980 年代の後期から 1990 年代前半にかけて，いくつかのオブジェクト指向分析／設計方法論が発表された．また，実装面では，ソフトウェア開発の長年に渡る課題であった再利用の分野でデザインパターンが提案され，再利用の対象にも多様性がでてきた．オブジェクト指向という比較的新しい技術，あるいは考え方を取り巻く「使い方」を，実システムへ適用するための期は熟していると言ってよいだろう．

ところで，ソフトウェアというものは，それがオブジェクト指向で開発されたか否かに関わらず，利用者の要求に対応するための仕様変更を避けて議論することはできない．一般にオブジェクト指向で開発されたソフトウェアは拡張性が高いと言われているが，この拡張性とは，あくまでも設計者が想定していた範囲の拡張であり，システムのフレームワークを変更しなければならない仕様変更に対しても柔軟に対応できるという意味ではない．それでは，分析段階で将来の予測を十分行うことは可能だろうか．システムを導入することによって，利用者の環境は変化する．利用者の環境が変化した結果，システムに要求される新たな仕様が求められることが多い．このように，将来の仕様変更を予測することは困難なだけでなく，現実世界の分析を行うときには次の問題がある．

- 分析にどの程度の時間を割くことができるのか
- 分析の対象を将来の不確定な状況まで広げることによって必要になる費用は，誰が負担するのか
- 将来の要求を実現するときのコンピュータ環境をどの程度まで予測すべきか

最近のシステムの開発方法は，システムの寿命を長くすることよりも，システムの寿命や世代を考慮して選択していく方向へと移りつつある [19].

利用者の満足度とシステムが提供する仕様の差について，A. M. Davis は「漸進的なプロトタイプ開発によって，利用者の満足度は高まる」と主張している [6]．多くのオブジェクト指向ソフトウェア開発ではこの漸進的プロトタイプ手法が採用されている [8]．つまり，オブジェクト指向が適用されているソフトウェアの多くは，十分分析を行い，将来の拡張に備えるというプロセスを経るのではなく，要求を確認しながら，利用者と共にシステムを育てていく形態が選択されている．このような開発手法を選択した場合でも，過去に開発されたライブラリクラスやフレームワーク，そしてデザインパターンの再利用は前提であるから，拡張しにくくなったソフトウェアを捨てて新たに作り直すとしても，全てをゼロから作り直すというわけではない．拡張しにくくなったソフトウェアに手を入れ続けるよりも，時期を見てソフトウェアを作り替えながらサービスを提供し続ける方が現実的である．

ところが，漸進型開発を採用する場合でも，次のような課題がある．

- システムの寿命とシステムを構成するオブジェクトの構造の変化との関係を，どのような手段で知り，その結果をどう判断すべきか

- システムの寿命を延ばす必要があるとしたら，いつ，システムのどの部分を再設計すべきか
- システムの寿命を延ばす手段を講じるより，システムを作り直した方が良いという意思決定は，何から判断すべきか
- システムを作り替えたとき，再利用されるオブジェクトと，再利用されないオブジェクトの差異を何から判断すべきか

これらの課題に対して，我々は，システムの進化，およびオブジェクトの進化という点に着目し，進化モデルという形で回答をだそうと考えた．システムは，システムを取り巻く環境が変化したとき，その仕様を変更させることによって環境に適合する．このとき，システムが環境適合するためには，内部のオブジェクトが個体として進化しなければならない．しかし，オブジェクトの進化はシステムのその後の進化に何らかの制約を与えているはずである．進化モデルでは，進化の時間軸をシステムが開発され，拡張され，さらに作りかえられる過程全体に広げ，その時間の中でオブジェクトの内部構造や，オブジェクトによって構成される組織がどのように成長・進化するかというプロセスをモデル化する．システムの世代交代まで進化モデルの時間軸を広げることで，再利用されるオブジェクトや再利用されないオブジェクトの進化にも言及できるようになるだろう．

システムの進化については，古くは Belady & Lehman が IBM360 の開発において調査研究を行った成果がある [11]．オブジェクト指向の研究分野では，初期開発，すなわちオブジェクトの組織化の方法に主な注目が集まっており，組織化後の進化の過程に注目した研究はあまり行われていないのが現状である．

1.1 目的

背景でも述べたように，漸進型の開発を進める場合，開発中のシステムを，全体的な開発過程の中に位置付けるための情報，システム内の老化したオブジェクトを発見するための情報，そして，次の開発において，現状のシステムを拡張していくのか，あるいは作りかえるのかを判断するための情報が必要である．

たとえば，オブジェクトの進化の状態には次のような状態が考えられる．

- オブジェクトの仕様が確定していない状態
- 大きく仕様を拡大させる状態
- 変更が起きない安定状態
- 新しいオブジェクトへ分割，統合，再設計など，再生されるべき状態

また，クラスは，そのクラスが再利用されるクラスと再利用されないクラスとで，進化の形態が異なると考える．そこで，クラスの進化の形態を調査することによって，クラスを分類する基準を発見することが可能になる．

以上のことから，本研究の目的は，

- システムの進化と関連づけて，オブジェクトの組織化から始まるオブジェクトの進化モデルを構築する
- クラスの進化の形態から，再利用されるクラスと再利用されないクラスとを分類する

1.2 研究方針

1.2.1 進化の捉え方

オブジェクトの進化の捉え方として，次に示すような定性的な分析方法と定量的な分析手法，および構造的な分析手法が考えられる．

- 定性的分析
  - オブジェクトの性質が進化によってどのように変化したか
  - オブジェクトの進化はなぜ起きたか
- 定量的分析
  - オブジェクトの規模はどの位変化したか
  - オブジェクトの複雑度はどの位変化したか
  - オブジェクト間の通信はどの位変化したか
- 構造的分析
  - オブジェクトを構成しているオブジェクトがどのように変化したか
  - オブジェクトのスーパークラス，サブクラスはどのように変化したか

本研究では，オブジェクトの進化の状態を客観的に判断するために，定量的分析を中心に行った．分析の過程では，必要に応じて進化の原因や結果を構造的に，あるいは定性的に分析した．

定量的分析を行うためには，進化を量で捉えるための定量尺度（メトリクス）が必要である．オブジェクト指向では，1990 年代に入ってメトリクスに関する研究が多く進められている [1, 4, 9, 13, 21]．そこで，既存のオブジェクト指向メトリクスに関する調査を行い，進化を定量化するための進化メトリクスを定義する．進化メトリクスは，計測に時間がかからず，設計者にとって理解しやすい尺度であり，かつオブジェクトの進化によって，敏感に値が変化する尺度を選択する．

1.3 作業手順

次に示す手順に従って，オブジェクトの進化を計測し，分析する．

1. 計測のための準備
   (a) データ収集方法の決定
   (b) データ解析方法の決定
   (c) 計測対象システムの条件の設定
   (d) 計測対象システムの選択
2. オブジェクト指向メトリクスの調査
3. 進化を定量化する視点の決定と進化メトリクスの定義
4. データ収集
5. 統計処理の結果に基づいた進化モデルの定義

1.3.1 データ収集方法

データの収集を行うために，システム顧客の要求変更による仕様の変化が大きいと思われる受託システムの中から，次の条件に合うシステムを選択する．

計測対象のシステムの条件は次の通りである．

- 開発過程に沿った 4 つ以上の版が入手可能なシステム

  開発過程とは，従来の開発工程の分類でいう初期開発工程から保守工程，そしてシステムの世代が変り最終的に破棄されるまでを含めたシステムのライフサイクル全体を指す．版とは，開発過程の，ある区切りで入手可能なソフトウェア成果物である．ただし，版は，それがプログラマによって管理された単位であるか，構成管理者によって管理された単位であるか，あるいは顧客へ納品された単位であるかは問わない．

- 開発者とその開発者が開発した箇所が明らかであるシステム

  オブジェクトの開発者に依存したオブジェクトの進化と一般的な進化とを区別するための条件である．

- 開発言語が VisualSmalltalk であること

  VisualSmalltalk は，計測を行うためのプログラムが VisualSmalltalk で開発されているものが既に手元にあったので，それを利用できた．また，計測を行うとき，必要に応じてプログラム稼働環境を入手できる保証があった．言語による計測結果の差異をなくためには，計測対象のシステム開発言語を統一する必要があったので，上記 2 つの理由から，開発言語を VisualSmalltalk に限定した．

- 開発者へのインタビューが可能であること

  進化したオブジェクトの設計内容について，開発者の意思決定の根拠を探るための手段，および，定量的データの収集や進化モデルから得られるデータが今後の設計に及ぼす影響についてインタビューを行う．

インタビュー方法については，できるだけ数値化できるようにアンケートを用意し，開発者に回答してもらう方法を選択する．実際に行ったアンケートは，クラスが削除された理由，クラスの規模が変化した理由について質問を設け，次の項目の中から該当する理由にマークをつけてもらい，後に集計を行った．

1. クラスの削除理由
   - 要求変更により，クラスが不要になった
   - 要求変更による再設計を行った
   - 設計上の理由による再設計を行った
   - クラス木の再設計に伴い，クラスの抽象化と具象化を行った
   - クラスを分割した
   - 新しいクラスが定義され，クラス名が混同するため名前を変更した
   - より適切なクラス名を発見したための変更した
   - その他：その他の場合の理由
2. クラスの規模が変化した理由
   - 要求変更によって当該クラスに直接影響があった
   - 開発途中の計測値である
   - クラスの設計上の理由による再設計を行った
   - クラスの抽象化や具象クラスの追加による再設計を行った
   - その他：その他の場合の理由

1.3.2　データ解析方法

収集した解析対象のデータには，定量的データと構造的データがある．それぞれのデータに対する解析方法は次のとおりである．

定量的データの解析方法

定量的データの解析には，次の統計手法を用いた．

- 基本統計量の計算
- 相関分析
- t 検定
- 回帰分析

計測結果が正規分布ではなく，右に尾を引く尖った形状をしているものが多い [13]．そのため，基本統計量（平均値，中央値，分散，最小値，最大値）と共に箱髭図で用いる四分位点と内堀値，および外れ値の検討を行う．

構造的データの解析方法

構造的なデータとして，クラスの継承構造に関するデータを収集する．継承構造に関するデータから，ある特定のクラスがどのようなクラスをスーパークラスとして定義していたか，どのようなクラスをサブクラスとして持っていたか，また，この構造が時系列でどのように変化したかを観察する．観察結果は，定量的データと関連づけて解析する．