第 6 章

まとめと今後の研究課題

6.1 本研究の成果

本研究では，オブジェクトの進化過程を調査し，組織化過程を検討することによって，次のことが明らかとなった．

- クラスの行数，クラスのメソッド数，メソッドの行数の分布は統計的にも安定しており，システムが漸進的に進化しても変わらない．これは，統計モデルを用いて漸進的開発の計画や見積りを行う情報として使えると期待できる．

- その分布の中で，特異な早い進化を示すオブジェクトは，開発者が設計上の問題を持つオブジェクトとして抽出していたオブジェクトと一致した．このことから，オブジェクトの進化過程を観測することによって，設計上の問題箇所を客観的に発見できる．

- クラスは，その C 値（１メソッド当たりの行数）がクラス継承木内のクラスの C 値の平均値に近い値を持つように設計されるか，あるいは設計変更される傾向があり，クラスの C 値とクラス継承木の C 値との差異は，クラスの設計上の問題を表わしている可能性が高い．

- クラス継承木の C 値は開発者によらず，またシステムの進化の影響も受けずに一定であり，各クラス継承木に固有の値として決定される．

- クラスを種に分類したとき，種によって異なる進化過程を観測した．この進化過程はシステムのソフトウェアアーキテクチャから予測できるものと一致する．もし，クラスが，予測した進化過程と異なる進化を起こしている場合，それはシステムのソフトウェアアーキテクチャが利用者環境の変化と対応していないか，あるいは開発者の技術的環境へクラスが適応した過程をあらわしているかのいずれかである．

- オブジェクトの進化過程を組織化過程に適用したところ，従来の手法よりも多面的な情報を持ったオブジェクトモデルを効率的に構築できることが明らかとなった．

6.2 関連研究

M. Lehman と L. Belady は，IBM360 の開発におけるシステムの進化過程を調査し，ソフトウェアの進化法則を定義した [33]．本研究では，Lehman らが指摘したように，クラスやメソッドの中には，利用者の新しい要求を取り入れることで，規模を増大させ続けるものが存在した．しかし，多くのクラスやメソッドは，その規模が増大しない方向に設計変更がなされていた．最近開発されるソフトウェアはサーバ／クライアントの三層モデルのように，階層構造を持たせることでシステムを取り巻く技術進歩の影響を局所化するアーキテクチャが一般的に使われている．これは，ソフトウェアの進化を従来のようにシステムレベルだけで観測するのではなく，システム内のオブジェクトの種に着目して，その進化の特徴を議論しなければならないことを意味していると考える．本研究では，オブジェクトの種という概念を導入し，オブジェクトの進化モデルの構築を試みた．この点で，Lehman らのソフトウェアの進化法則を前進させた議論を展開した．

本研究の調査でも明らかとなったように，オブジェクトは，特定のアプリケーションプログラム内で進化するだけでなく，新たなアプリケーションで再利用される場合にも進化する．ここで，問題となるのは，あるオブジェクトが進化したとき，そのオブジェクトを共有する他のプログラムにも影響が及ぶ場合の対処である．典型的な場合として，データベース上のオブジェクトにマイグレーションを起こさせる必要が発生する場合を指摘でき，この問題に対して

は，現在のオブジェクト指向プログラミングの枠組みでは，オブジェクトの進化を限定したプログラム内だけに収めるという対処ができない．この問題に関して，いくつかの研究が行われている．

W. Harrison らは，上記のオブジェクト指向プログラミングの課題を指摘し，主観的な視点に基づくモデル化の方法としてサブジェクト指向を提起した [19]．W. Harrison らが提案するプログラミング環境では，オブジェクトは識別子と他のオブジェクトとの合成規則によって管理される．合成規則は，オブジェクトの識別子をもとに，一方のオブジェクトに特定のメッセージが送られると，他方のオブジェクトに副作用として予め定義しておいたメッセージを送る役割をもっている．この機構をプログラミング環境に導入することによって，複数のプログラムが一貫性を保ちながら，物理的には別のオブジェクトでも理論的には同一のオブジェクトを取り扱うことができるようになる．

たとえば，ある主観で定義されたオブジェクトが他の主観で定義されたオブジェクトと現実世界の同一のオブジェクトを指示する場合，モデルの外部に定義された合成規則によってモデル間の一貫性が保証される．個々のオブジェクトはシステム内で互いに独立した継承構造，集約構造，振る舞いを持つことができるため，合成規則を書き換えることによって，関連するシステムのオブジェクトを変更することなく，新たなオブジェクトを定義したり変更することが可能である．また，データベース上のオブジェクトを共有しながら，既存のオブジェクトに影響を及ぼすことなく，新たな振る舞いを持つオブジェクトを漸進的に定義することが可能となる．

N. Minsky らが提唱する規則統制アーキテクチャは，各アプリケーションごとに代理オブジェクトを定義し，代理オブジェクトに共通な振る舞いや属性を基本オブジェクトへ定義するアーキテクチャである [40]．このアーキテクチャでは，個々の代理オブジェクトがそれぞれのアプリケーションプログラムの中でメッセージを受け取ると，必要に応じて基本オブジェクトにメッセージを送り，全体としての一貫性が保持されるようになっている．したがって，システムに新たなアプリケーションが追加され，それにともない，既存のオブジェクトに新たな振る舞いが必要となった場合，新たな振る舞いに対応する代理オブジェクトを基本オブジェクトに追加するだけで対応することが可能である．さらに，代理オブジェクトを導入したことによって，オブジェクト指向の継承に

よる可読性の低下を集約構造によって回避できたという利点もある．

プログラミング言語自体の拡張に関する研究も行われている．J. Silling らが提唱したプログラミング言語は，オブジェクト自身に三層構造を与え，代理オブジェクトと基本オブジェクトを一つのオブジェクトの中に実現した [54]．第一階層には，多重化されたインタフェースを定義する．インタフェースの多重化とは，一つのインタフェースに複数のメッセージの組みを定義するという意味である．このオブジェクトと対話を行うオブジェクトは，どのインタフェースを使用するかを予め指定しておく．第二階層は，インタフェースに定義されているメッセージと起動されるメソッドとの関係を管理し，第三階層は，メソッドとインスタンス変数との関係を管理する．この三層構造によって，オブジェクトを進化させるとき，第一階層の関連するインタフェースや，第三階層のメソッドとインスタンス変数の関係を変更するだけで他のインタフェースには影響を与えずにすむ．

いずれの研究も，個々のオブジェクトには多様な進化が許されるが，どのようにオブジェクトの同一性を感知し，一貫性を保つための手段を発見するかについて，議論されていない．本研究で示したオブジェクトの組織化過程は，多視点を用いてオブジェクトの進化過程を分析工程に導入した手法である．この手法ではオブジェクト辞書を用いて，オブジェクトの同一性を発見し，統合時にオブジェクト間に発生していた衝突を解消した．これらの手順は，上記の関連研究で説明した各手法や機構で，オブジェクトの同一性を感知し，一貫性を保つ手続きを定義するためにも有効である．

本研究の組織化過程では，オブジェクトモデルの統合を中心に議論した．オブジェクト指向では，他に機能モデル，動的モデルといった別の技術的視点で構築されるモデルがある．機能モデルはあるシナリオに沿って個々のオブジェクトのメッセージの流れを議論するものであるから，モデルの統合は意味をなさない．しかし，動的モデルについては，オブジェクトの状態遷移図の統合という課題がある．本位田らは，多視点を用いて定義された動的モデルを統合する手法について研究を行った [23]．動的モデルの統合では，主に，新たな状態と事象の発見と，状態の包含関係を発見することに重点がおかれる．分析モデルを構築する際には，本研究の組織化過程を補完する位置付けとなる．

また，本研究では，開発時の手法自体が持つ多視点については議論をしな

かったが．B. Nuseibeh らの多視点に関する研究では，動的，静的，機能的な側面を視点ととらえ，視点の異なる仕様書において発生するオブジェクト間の矛盾を発見し解消する方法を提案している [48, 49, 47, 14]．開発プロセス全体を網羅した手法と言われているが，本研究で示した組織化過程が対象とするオブジェクトの抽出，統合といった工程は対象とされていない．

同様の研究として，Nissen らによる M2 モデル（メタメタモデル）の提案がある [46]．M2 モデルとは，多視点による要求をシステムに取り入れるために，仕様書に書かれたモデルのメタモデルから，すべてのモデル化手法に共通な仕様化モデルのメタモデルとして定義されたモデルを指す．M2 モデルでは，システムの仕様書ごとに Agent，Activity，Data，Medium との関係を示すことができるため，各仕様書が記述の対象としている事物や事象の役割分担とシステムにおける立場を概観することが可能となる．したがって，M2 モデルを参照すれば，仕様書に記述された要求の矛盾や抜けを発見でき，そこから新たな要求を発見することも可能となる．

本研究で提案した組織化過程では，利用者モデルを用いて問題領域の専門家とのコミュニケーションを行おうと考えている．H. Nissen らの研究成果は，開発者が技術面からモデルの情報の不備を発見し，そこからさらに新たな要求を発見しようという点で新しい手法であると評価できる．このような手法は，組織化過程における矛盾を発見する手法としても導入する必要があるだろう．

6.3 今後の研究課題

オブジェクトの組織化と進化に関する今後の研究課題を次に挙げる．

- メッセージレベルにおける進化の分析

  メッセージレベルの進化では，メッセージの名前に着目できる [44]．システムの成長にともなう多相メッセージはどのように進化するのか．単語の種類と使用頻度などの解析によって，メッセージレベルの進化を観測することが可能である．

- 再利用ライブラリクラスの進化過程に関する研究

Smalltalk が提供しているクラスライブラリは Smalltak の改版がある都度，その構造やインタフェースに小さな変更がある．このような変更はどのような原因によって発生するものなのか．再利用クラスライブラリの進化過程を明らかにすることで，再利用クラスの開発手法も導くことが可能となるだろう．

- より長いシステム開発事例におけるオブジェクトの進化過程に関する研究

  Tamai らは，ソフトウェアライフサイクルについて，システムの移行の要因を事例から調査した [57]．彼等の研究では，いくつかの企業で，ソフトウェアのライフサイクルを管理し，目標を定めて寿命の予測をたてていることが報告されている．オブジェクトの寿命に関する調査を含めて，より多くの種の進化パターンを収集することは，オブジェクトの進化モデルを精緻化するために重要な研究である．

- 進化パターン抽出のための環境の開発

  本研究では，オブジェクトの進化メトリクスと，オブジェクトの進化過程を定量化するための計測方法および計測結果の解析方法を明らかにできた．今後の計測と分析の生産性を向上させるための環境が必要である．また，進化環境図における技術的環境からの距離に関する概念に関する議論，および観測手段について検討する余地は残されている．

- オブジェクトの組織化過程の充実に向けて

  本研究ではモデル間の矛盾の発見について，事例に基づいた解法を示した．さらに，一般的なモデル間の矛盾の発見と解消手段について検討する余地がある．