第 3 章

1995 年度の研究成果概要

3.1 定量化する対象オブジェクトの策定

オブジェクト指向で開発されたソフトウェアシステムおよびシステムを構成するクラスなどのオブジェクトの進化を捉えるために，観測する対象を定義した．静的な進化を観測する対象は，次の 4 レベルである．

- システム
- クラス
- メソッド
- メッセージ名

ここでメソッドとは，個々のクラスに定義されている操作手続きを指し，メッセージ名とは，オブジェクト間のメッセージ送受信で使われるメッセージの名前の種類を指す．

これら 4 つの対象は，システム全体の進化やクラスの進化に従って，個々のオブジェクトがどのように進化するかを捉えるために選択された．

3.2 進化メトリクスの定義

進化の様子を探るために，先に示した 4 つの対象について，時系列データに基づいて，オブジェクトが時間に従ってどのように変化するかを分析した．分析に用いたメトリクスは，大きく 2 つに分けられる．第一のメトリクスは，ソースコードから静的に計測値を求めることができる静的メトリクスで，第二のメトリクスは，プログラム実行時に計測値を求める動的メトリクスである．第二のメトリクスは，計測対象プログラムが，Smalltalk で開発されていたため，オブジェクト間の結合度を求めるために設定した．

各計測対象に対して定義した静的な進化メトリクスを次に列挙する．

1. システムの進化メトリクス

   オブジェクト指向システムの場合，システムが進化することによって変化する量は，システムに定義されているクラス数や行数などが考えられる．また，ここでメッセージ名の種類数とクラス内に定義されているメソッド数の総数を計測することによって，システムの進化とメッセージ名の多相性の変化との関係を知ることができる．

   進化の計測で選択したメトリクスは次のとおりである．

   - クラス数，行数，メッセージ名の種類数，クラスに定義されているメソッド数の全クラスの合計

- メッセージ名の種類数に対して，1 回しか定義されていないメッセージ名の種類数と，それが全メッセージ名の種類数に占める割合
- システムを構成する全クラスの，行数，メソッド数，インスタンス変数の数，継承の深さとサブクラスの数の分布
- 各クラスに定義されている全てのメソッド行数の分布

2. クラスの進化メトリクス
   - インスタンス変数の数 (NOV)
   - メソッド数 (CNOM)
   - 行数 (CLOC)
   - 継承の深さ (DIT)，および直下のサブクラスの数 (NOC)
   - 関連メソッド数
3. メッセージの進化メトリクス
   - 計測対象のメッセージを受信できるクラス数 (ACM)
   - メッセージを定義しているクラス数 (MDF)
4. メソッドの進化メトリクス
   - 行数 (MLOC)

クラス間の結合度を求めるために定義した動的メトリクスは次のとおりである．

- あるクラスのオブジェクトが送信したメッセージ毎の送信回数と送信先オブジェクトのクラス名
- あるクラスのオブジェクトが送信したメッセージの種類数

3.3 進化過程調査結果

3.3.1 調査対象システム概要

調査対象としたシステムは，温度調節シミュレーションシステムである．全工程 8ヵ月の開発を一人の開発者が担当し，その間，6 回のリリースが行われた．また，このシステムは，Smalltalk/V 上に構築されているコンポーネントウェアのひとつである PARTS Workbench を利用して開発された．PARTS Workbench を利用することで，登録されているパーツと呼ばれる部品を対話的に組み合わせ，様々な条件のシミュレーションシステムを視覚的に構築できるようになっている．

基本構成は，シミュレーション対象のシステムを組み立てる表示部分，シミュレーション用のデータを入力する編集部分，およびシミュレーションを実際に行う計算部分の 3 つである．処理は，3 種類のクラスの協調作業によって進む．初めの 2 つの部分が，PARTS Workbench のクラスを継承して構築した部分である（図 3.1）．

システムの規模は ver.0 でクラス数 10，最終の ver.5 で 52 であった．ver.0 から ver.1 までは 2ヵ月の間があり，それ以降の版はそれぞれ 1ヵ月毎にリリースされている．各版のリリース内容を次に示す．

1. ver.0

   PARTS Workbench のシミュレーションシステムへの適用可能性を検査するためのプロトタイプ．PARTS Workbench 上で定義したシミュレーション設備を動的に接続し，各設備の特性に従ってシミュレーションを実行できることが確認された．

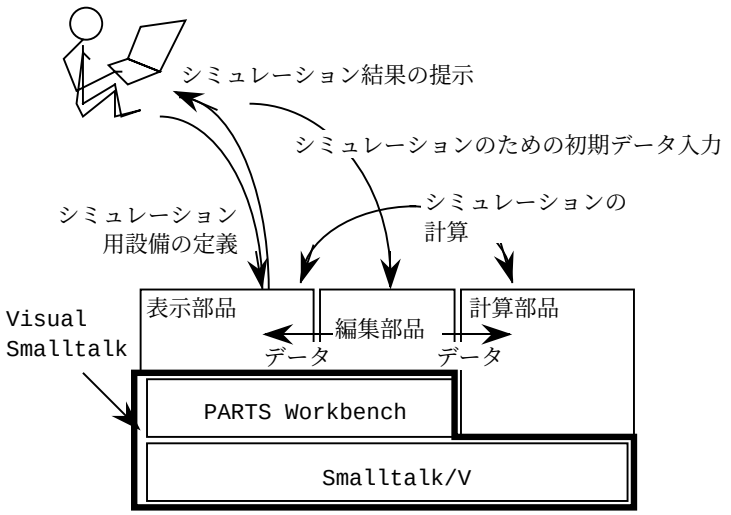

図 3.1: シミュレーションシステムの概要

2. ver.1

   シミュレーションを行う最小構成のシミュレーションシステムを開発した.

3. ver.2

   シミュレーション対象設備を多様化させ，さまざまな温度調節のための計算方法を導入した．そのため，システムの中心は GUI から，シミュレーションの計算が中心のアプリケーションに変化した．この版で定義されたシステムの構成を図 3.1に示した．シミュレーションシステムを組み立てるとき，実際に利用者が操作するのは表示用部品である．利用者がシミュレーション対象の設備の初期設定を行う時は，表示用部品を経由して編集部品をアクセスする．シミュレーションを実施しているときの計算結果は，計算部品から表示用部品を介して利用者に提示される.

4. ver.3

   さらに取り扱う設備の多様化，シミュレーションの複雑化の要求に応えると同時に，複雑化したシミュレーションを簡単な GUI で操作できるように GUI も変更された.

5. ver.4

   シミュレーションのための設備の追加と削除が行われた.

6. ver.5

   利用上の問題が解消された．特に設備の追加や削除，機能の追加は要求されなかった.

進化の検討では，ver.1 から ver.4 までを対象とし，ver.0 と ver.5 を除いた．これは，ver.0 が使い捨て型のプロトタイプであり，それ以降の進化型プロトタイプとは性質が違うため，また，ver.5 は，ver.4 とほとんどシステムに変化が見られなかったためである.

1995 年度の研究は，最初に熱交換シミュレーションシステムの 4 つの版について，進化メトリクスを用いた時系列の計測を収集し，基本統計量に基づいた分析を行い，進化メトリクスの評価を行った．また，計測した値がオブジェクトの進化をどのように表現しているかを明らかにするために，計測と分析を繰り返しながら，進化モデルの仮説の設定を行った.

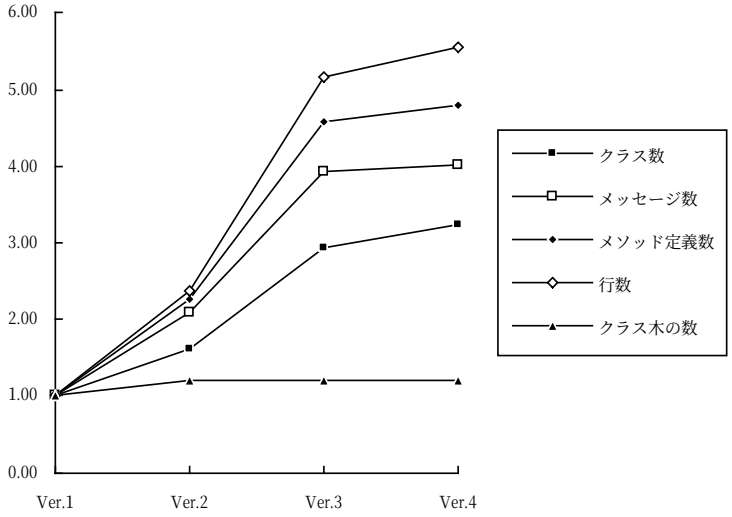

図 3.2: システムの S 字の成長曲線

3.3.2 オブジェクトの進化モデル（仮説）

1995 年度の研究では，このシステムに関し，次の進化の形態を観察した．

システムの進化形態

- システムの成長は時間軸に沿って一様ではなく，S 字の曲線を描く（図 3.2）
- 1 クラスに定義されるインスタンス変数の数，メソッドの数，行数，1 メソッドあたりの行数，1 メッセージ名に対するメソッドの定義回数の中央値は，システムの進化の状況に関わらず変化しないが，分散の値は大きくなっていく傾向を持つ
- これらの分布は，右に尾を引く，尖った分布を持つ．また，右に引かれた尾は，システムの進化に従ってさらに長くなる傾向を持つ
- これらの分布において，分散が大きくなっていくのは，大きな値を持っているいくつかのオブジェクトが，システムの進化に従って，さらにその値を増していく進化過程を持っていたためである
- 1 メッセージ名に応答できるクラスの数（メッセージの多相性）の平均値はシステムの進化に従って増加する傾向がある
- クラスのライブラリクラスから計測した継承の深さの中央値は，システムの進化の状況に関わらず，最大値は高だか 4 であった．新しいクラスが追加される位置は，クラス木の再下層というよりは，既に継承されているクラスに追加される傾向があった．

参考として，1 クラス当たりのメソッド数の基本統計量の変化を表 3.1に示した．

クラスの進化形態

- クラスの進化において，1 クラスに定義されているメソッド数，変数の数，行数には正の相関がある．

  これは，メソッド数や変数の数が多いクラスは，行数も多くなることを意味していると解釈できるので，メソッドの実装を行う前に，クラスの規模を見積もることが可能である．また，いずれかがシス

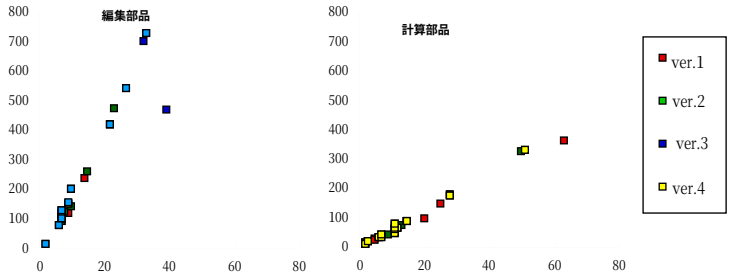

図 3.3: クラス木毎の行数／メソッド数の散布図

表 3.1: 1 クラス当たりのメソッド数の基本統計量

	ver.1	*ver.2*	*ver.3*	*ver.4*
平均	10.38	15.65	17.60	16.58
中央値	7	9	11	11
分散	123.58	279.76	373.16	386.88
尖度	0.01	1.50	6.56	9.46
歪度	1.03	1.40	2.26	2.67
最小	0	0	0	0
最大	34	63	100	110
データ数	16	26	47	52

テム全体の分布で外れ値を持っている場合は，他の計測値でも外れ値か，あるいはそれに近い値を持つ．規模が大きいクラスにはエラーの発生する可能性が高いので，このようなクラスを発見する場合，すべてのメトリクスについて計測を行わなくても，いずれか一つのメトリクスを用いて計測を行うだけで十分である．

- 行数／メソッド数の値は，クラス木毎に固有の値を持っており，クラスの進化によって変化しない（図 3.3）．

- あるクラスの行数／メソッド数の値が，同じクラス継承木に属する他のクラスの値と大きく異なる場合，そのクラスの設計に問題がある場合がある．

  図 3.3に示した矢印は，あるクラスの ver.3 から ver.4 への進化の様子を表わしたものである．ver.3 で，このクラスが持っていた値は，このクラスが所属するクラス継承木を構成する他のクラスの値から，明らかに外れていた．しかし，ver.4 では，このクラス継承木の固有の行数／メソッド数値に近い値を持つように進化していた．以上から，クラス継承木に固有な値から外れた値は，設計者が直観的に認識していた設計上の問題箇所を表わしていたと解釈できる．

- クラスの進化メトリクスの値が外れ値を持っている場合，クラスの進化によって，更に平均値から離れていく傾向がある

- クラスの進化メトリクスの値が平均値に近い場合，システムが進化してもクラスの進化は少ない

  以上の 2 つの形態は，システムの進化の影響が全クラスに平均に及んでいるのではなく，規模が大きいクラスに集中する傾向があることを意味していると解釈できる．また，図 3.4に見られるように，

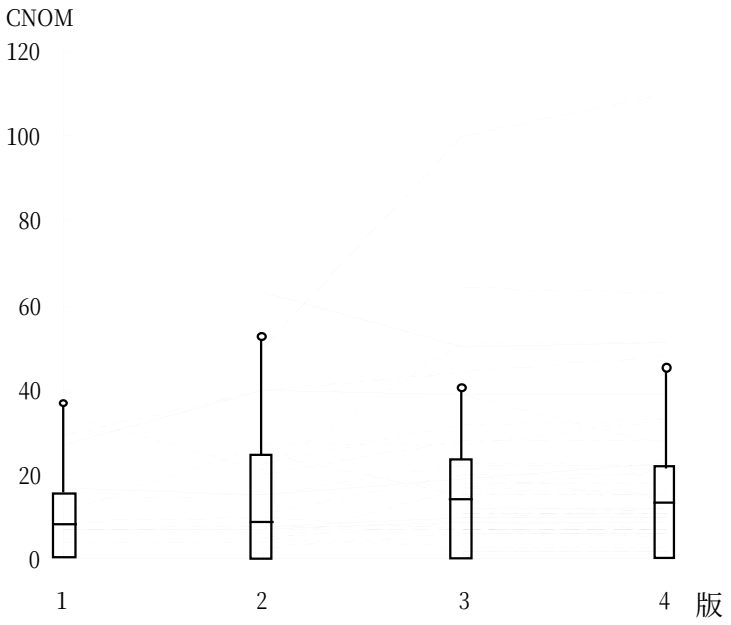

図 3.4: 1 クラスに定義されたメソッド数の変化

これらのクラスは，もともと外れ値を持つような規模の大きいクラスであったことから，開発当初から，システムの仕様がこれらのクラスに集中していたと考えられる．

ただし，クラスの進化の過程から，これらのクラスに設計上の対処がなされていなかったわけではない．ver.1 から ver.2，あるいは ver.2 から ver.3 へ至る過程で，最大規模を持っていたクラスの値が一時的に減少している．その後のクラスの進化の傾向を観測したが，一般的な傾向を述べることはできない．図 3.4のあるクラスは値を減少させた後，再び増加させているが，他のクラスは減少後，値を増加させていない．値を増加させていない期間は ver.3 から ver.4 へ至る過程である．この期間は，システムもほとんど進化していないので，値が安定している原因が，設計による対処の効果なのか，システムの進化の状態によるなのかを特定できない．

メソッド の進化形態

- あるメソッド の行数が，システム全体の分布で外れ値になっている場合，メソッド の進化に従ってさらに平均値から離れていく傾向がある
- メソッド の進化メトリクスの値が平均値に近い場合，システムが進化してもメソッド の進化は少ない

  メソッド の進化の計測結果に関する基本統計量は，第 4章に掲載してあるので参照されたい．ここでは，メソッド の進化に従ってさらに平均値から離れていく傾向があることを図 3.5に示す．規模の大きいメソッド に対しては，クラスで観察された再設計の効果を見ることはできなかった．

メッセージの進化形態

- 多相性を持つメッセージは，システムの進化に伴って，定義回数を増加させる傾向を持つ

  多相性を持つメッセージの進化は，図 4.16に示した．全般的に定義回数は増加する傾向がある．これは，新たに定義するメッセージの名前をつけるとき，同様の意味を表わす多相性を持つ既存のメッセージ名を選択していると解釈できる．いくつかのメッセージについては，定義回数が減少してい

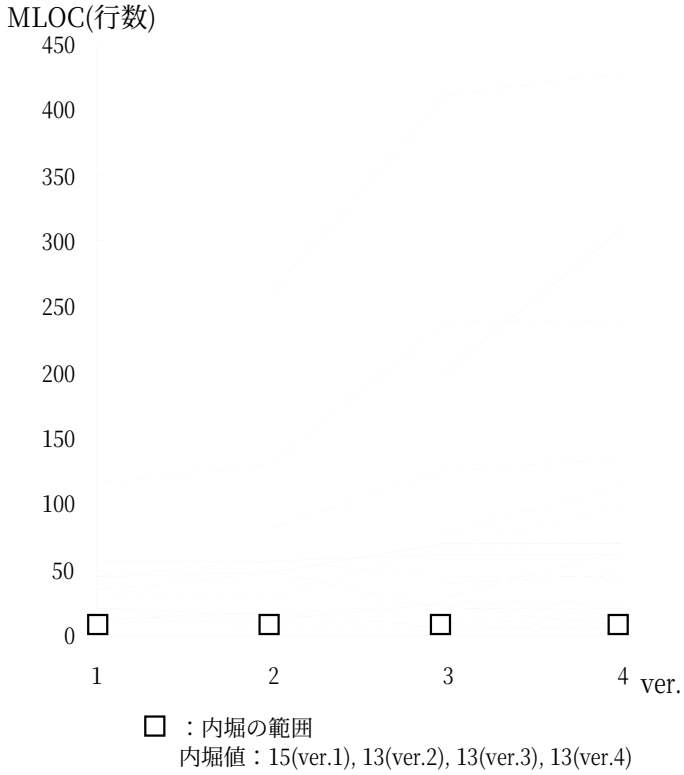

図 3.5: １メソッドの行数の変化

る．その原因として，スーパークラスへのメソッドの統合が考えられる．その他に，多相化による意味の繁雑さによる名前の再検討が起きていると思われる例が観測できた．

たとえば，reset というメッセージ名が多相性を持つメッセージとして定義されていた．reset は，ver.1 から ver.2 にかけて，定義回数を増加させるが，ver.3 では，応答可能なクラスの数を増加させていたにもかかわらず，定義回数は減少させていた．他のメッセージ名を調査した結果，この現象は，ver.3 で，reset という名前から，状態の初期化，計算値の初期化を表わす別の名前を分化させたものと解釈できた．

全体の進化の傾向

進化メトリクスを適用して得た計測結果の分布には，いずれの計測値にも，その形状と進化の傾向に同様の傾向を観測できた．計測値の分布は，いずれも右に尾を引く尖った形状を持ち，少数の外れ値を持つ．分布の中央値付近のオブジェクトは，システムが進化してもあまり進化をしないが，中央値から離れたところにあるオブジェクトは，システムが進化するに従って，その値を増加させる．

漸進型の開発を行ったとき，システムの規模は S 字の成長曲線を描く．この成長は，既存のすべてのクラスやメソッドに平均に影響を与えるのではなく，特定の規模の大きいクラスやメソッドに対して働いていた．

ところで，このようなシステムの成長の過程で，規模が大きいクラスを含めたいくつかのクラスに対して，規模を縮小させるような再設計が行われていた．規模が増大するということはクラスやメソッドのエラー発生可能性が増加することを意味する [11] ので，再設計は，エラー発生可能性を増加させないための開発であったと言うこともできる．しかし，増大する傾向を持つメソッドに対しては，再設計の効果は観測できなかった．

進化過程の一般化に向けて，再設計が行われる時期と対象について，他のシステムの調査を進めて明らかにする必要がある．また，規模が大きくなるクラスがシステムの中で果たしている役割を調査し，役割と進化の関係を明らかにしていく必要もあるだろう．