第 5 章

進化モデルの検証

5.1 熱交換シミュレーションシステムの継続調査

今年度の調査では，クラス木の進化と定量データの変化の関係を調査し，設計者がの意思決定と定量データの関係を中心に調査した．

5.1.1 クラス木の進化と計測結果の関連調査結果

いくつかのクラスにおいて，その振る舞いの規模を表わす 1 クラス当たりに定義されているメソッドの数 ($CNOM$) が，システムの中でも際立って大きくなると，次の版で値が小さくなるという現象が見られた（図 5.1）．$CNOM$の値が小さくなる原因を探ったところ，クラス階層上のクラスの再設計が行われていることがわかった．

次に詳細を説明する．

- システムを移行させずに複数のクラス木に影響が及ぶ再設計は，進化の初期の段階であれば起こりやすい．

  表示部品，編集部品，計算部品の 3 クラス群が一つの単位になる構造は ver.2 で確立されたものである．計算部品は，ver.1 から ver.2 へ至るシステムの進化過程で，表示部品を分割して定義された．分割前のクラス群と分割されたクラス群を，図 5.2に group.1 として示した．図 5.1では，クラス p8 が分割されることによって$CNOM$の値を減少させている様子を表わした．クラス p8 や，ver.2 で新たに定義されたクラス p5 は，ver.2 以降のシステムの進化に従って，規模を増大させる方向に進化しているので，クラス p8 が ver.1 から ver.2 にその規模を減少させていたのは，再設計による効果であると言ってよいだろう．

  開発者へのインタビューの結果，ver.3 の開発を始めるとき，表示部品と計算部品の再構築を検討したが，変更による影響範囲が大きく，利用者へ配布する期日を守れないことが予想されたため，取り止めていたことがわかった．

  したがってシステムの広範囲に及ぶ再設計は，進化の初期に行われても，システムの規模が大きくなるにしたがって行われにくくなると考えられる．

- 抽象化を行い当該クラスのスーパークラスを新たに定義したり，兄弟クラスが追加されたときにスーパークラスの再設計を行った場合，当該クラスを含めた兄弟クラスの$CNOM$の値を減少させることができる．

  スーパークラスが新たに定義されると，当該クラスの仕様の一部をスーパークラスから継承できるようになるため，当該クラスの$CNOM$の値は減少する．この例は，調査を行った熱交換シミュレー

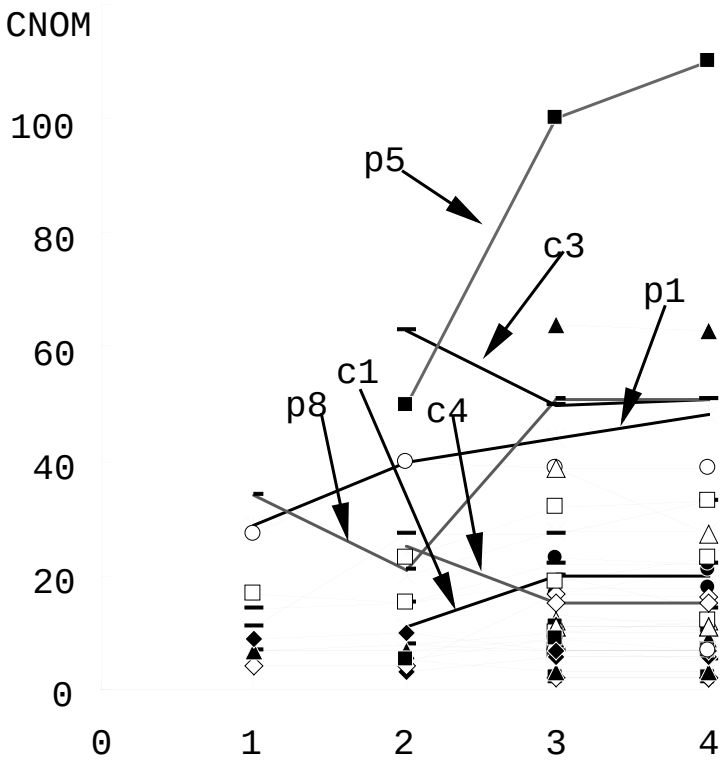

図 5.1: クラスの進化プロセス

ションシステムの表示部品及び計算部品で観測できた．このグループを group.2 として図 5.2に表示した．ver.1 から ver.2 へかけてと，ver.2 から ver.3 の間で，スーパークラスに変化があったクラスを抽出した．group.2 に属するクラスの $CNOM$ の値の変化の様子も（図 5.1）に c3，c4 として示した．これらのクラスの $CNOM$ の値はいずれも ver.2 から ver.3 にかけて減少している．

5.1.2 クラス木に固有な行数／メソッド 数の検定結果

- $CLOC$，$CNOM$，NIV には正の相関があり，例えば $CLOC/CNOM$ の値は，クラス木毎に固有の値を持ち，進化を通して一定である

  計測した結果から，相関分析を行った結果，$CLOC, CNOM, NIV$ に強い正の相関関係があり，クラス木毎に $CLOC/CNOM$ の値がほぼ一定であった．ここで，一方のクラス木の平均値を μ_0 とし，他方のクラス木の平均値 μ との間に有意差があるかどうかを，3 つの主なクラス木について両側 5%の t 検定を行った．検定の結果，P 値は 10^{-15} 以下で，互いのクラス木において，平均値が同じであるとは言えなかった．各クラス木における 4 版分のデータから求めた行数／メソッド 数の標本数，平均値，分散を表 5.1に示す．

5.1.3 動的メトリクスを用いて調査した結果と進化の関連調査結果

動的メトリクスを用いて，各クラスのオブジェクト間で送受信されたメッセージの回数を調査した．入手できたシステムのテストデータが 1 ケースであったため，あらゆるテストケースにおけるメッセージ送受信の状態を調査することはできなかった．しかし，オブジェクトがシステムの中で果たしている役割に基づいて，4 つのグループに分類したところ，オブジェクトの送受信の状態とクラスの進化とを関連づけることができた．

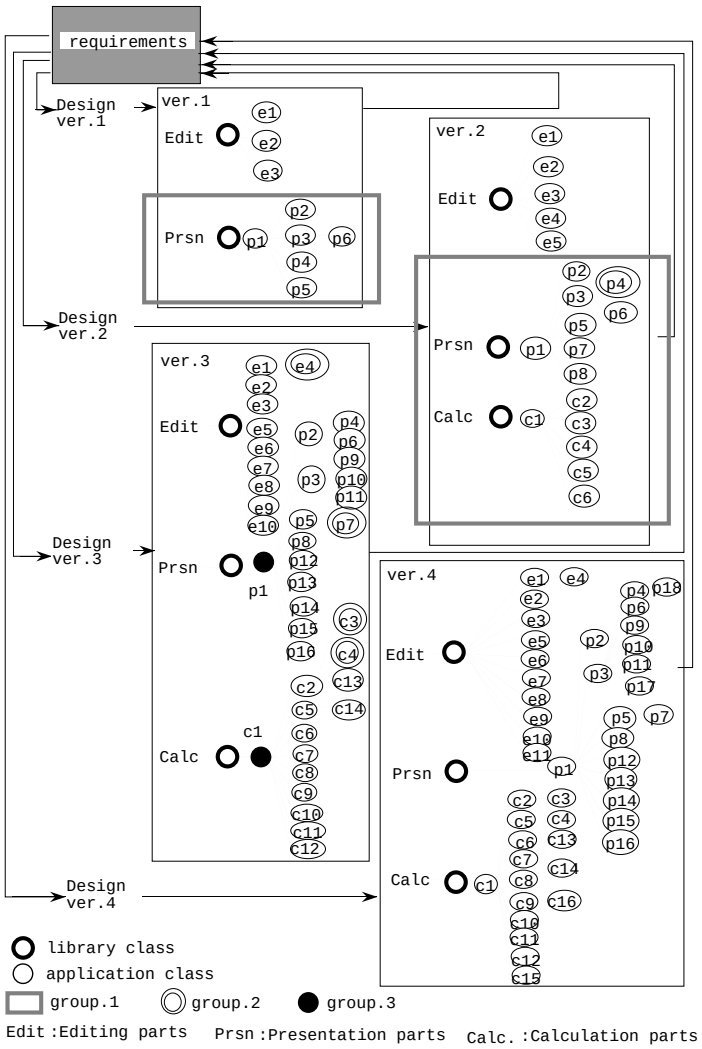
requirements
Design ver.1
ver.1
Edit
Prsn
Design ver.2
ver.2
Edit
Prsn
Calc
Design ver.3
ver.3
Edit
Prsn
Calc
Design ver.4
ver.4
Edit
Prsn
Calc
library class
application class
group.1
group.2
group.3
Edit :Editing parts
Prsn :Presentation parts
Calc. :Calculation parts

図 5.2: クラス木の進化

- ライブラリ用クラス

  ライブラリ用クラスは，他のシステムを構成するオブジェクトからメッセージを受けるが自分からは他のオブジェクトに対してメッセージを送らないクラスと定義する．シミュレーションシステムでは，物理量を提供するクラスなどがこの部類に入る．

- インターフェイスクラス

  インターフェイスクラスは，利用者インタフェースのように，利用者から直接メッセージを受けるが，システム内のオブジェクトからはメッセージを受けないクラスと定義する．このグループのクラスは，後で述べる他の制御クラスやエンティティクラスへ処理を起動させるメッセージを送るトリガーの役割を持っている．

- 制御クラス

  制御クラスは，システムを構成する他のオブジェクトの振る舞いを制御したり，管理する役割を持ったクラスと定義する．多くのクラスの動作を制御する機能を持っているので，規模も大きく，他のクラスとの結合度も大きい．

- エンティティクラス

  エンティティクラスは，制御クラスとメッセージを送受信して，状態遷移を起こしたり，状態を伝えるクラスと定義する．このクラスの規模は，全体の分布の中で，中央値付近に集中する．

図 5.3の (a) は，横軸に自分へのメッセージ送信を除いたメッセージ送信回数を，縦軸にそのクラスのメソッド数の最大変化量を示し，(b) は，横軸に自分からのメッセージ受信を除いたメッセージ受信回数を，縦軸にそのクラスのメソッド数の最大変化量を示して，各クラス毎にプロットして散布図として表わしたものである．

この図から，メッセージ送受信の回数が少ないクラスは，メソッド数を変化させない傾向があることがわかる．しかし，メッセージ送受信回数が多いクラスが，メソッド数を変化させる傾向を持っているとは言いきれない．

ここでメソッド数の最大変化量とは，クラス i の時刻 t におけるメソッド数 $CNOM_{i_t}$，時刻 $t+\Delta t$ 版のメソッド数 $CNOM_{i_{t+\Delta t}}$ としたとき，最大変化量$\Delta CNOM_i$を次の式で求めた．

$$\Delta CNOM_i = |(CNOM_{i_{t+\Delta t}} - CNOM_{i_t})|/\Delta t$$

図 5.3で，メッセージ受信量の軸上で横一列に並ぶいくつかのクラスがある．これらのクラスのメッセージ送信量は 0 である．この特徴から，これらのクラスをライブラリ用クラスへ分類する．ライブラリ用クラスには，物理量を提供しているクラスの他に，計算部品のクラスがあった．表 5.2は，各クラスのメッセージ送受信回数と $CNOM$の最大変化量に増減の±の記号をつけてまとめ，メッセージ送受信におけるクラス間の結合度（CBO) を示したものである．発信回数／受信回数の値によって，そのクラスがどの位他のクラスの仕様の影響を受けているかを表わした．ライブラリ用クラスの値は 0 であり，他のクラスの進化の影響を受けない設計になっている．

ここで，ライブラリ用クラスに分類できるクラス名に*をつけた．ライブラリ用クラスの $CNOM$の変化量は，たかだか 1 だから，ライブラリ用クラスは，徐々に進化してライブラリになるというよりは，発生当初からある決められた機能を持っており，システムの進化の状況に関わらず，安定した状態を保つ性質があると言ってよいだろう．また，これらのクラスの規模は小さく，$CNOM$の計測値の分布では，中央値付近の値を持つ．

典型的なインターフェイスクラスは，受信メッセージは 0 になる．表 5.2では，クラス名の後に**をつけて示した．いくつかのクラスで，メッセージ受信回数が 0 でないものがあるが，関連を持つ計算部品からのメッセージ受信回数を除いた値（括弧内の値）を見ると，表示部品の多くがインターフェイスクラスの

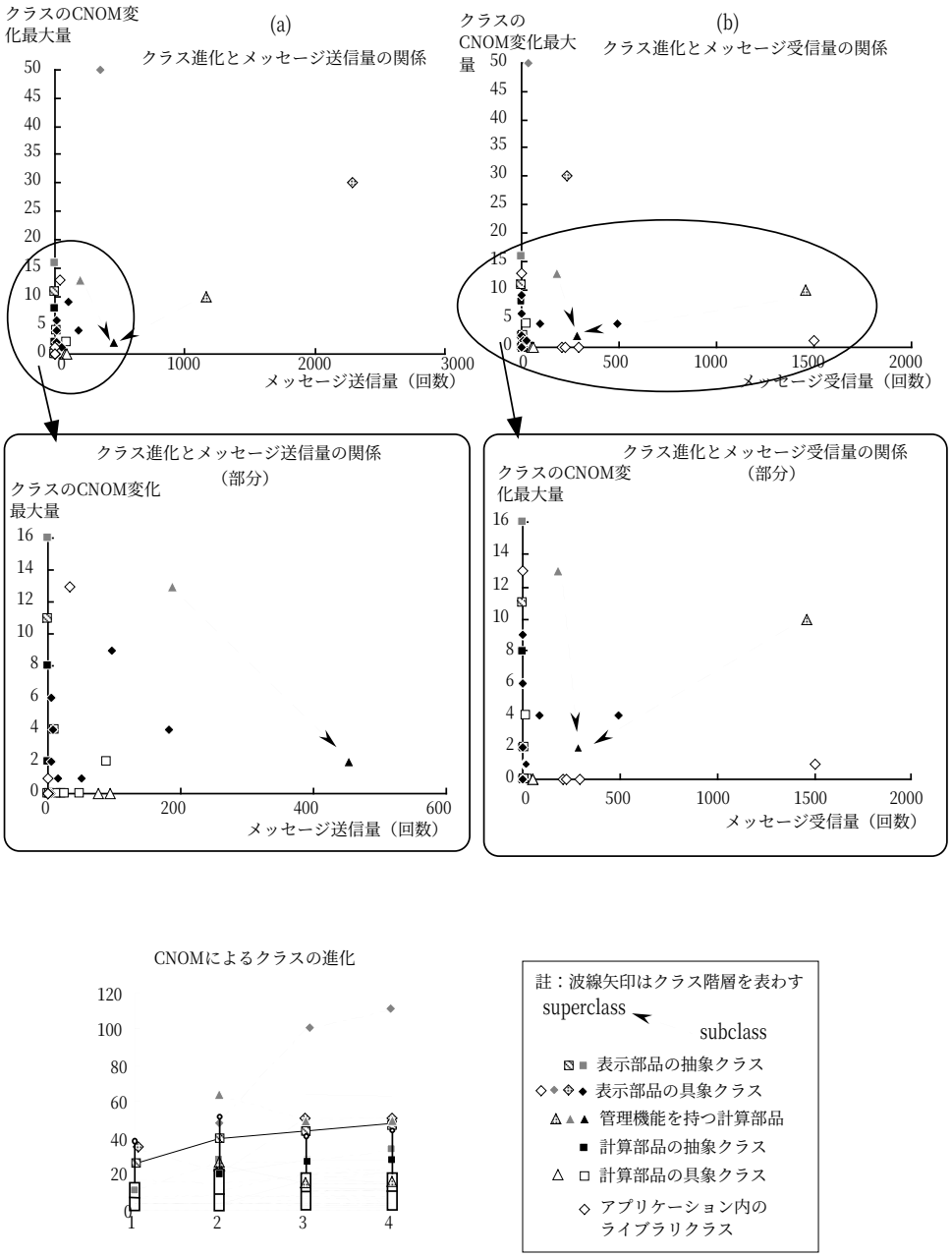

図 5.3: クラス進化とメッセージ送受信頻度の関係

特徴を持っていることがわかる．インターフェイスクラスは，他のクラスに比べて$CNOM$の値の変化量が大きいだけでなく，その変化の値は正である．この原因は，インターフェイスクラスは，各々固有の仕様を持っているため，クラスの抽象化によって仕様を整理することが困難であることが考えられる．

表示部品の中に，例外として受信回数の方が送信回数よりも多いクラスがいくつかある．たとえばP7では，受信回数490回に対して送信回数は9回となっている．これは，このクラスが表示部品としてのインターフェイスクラスとしての機能と共に，システム内の機能を制御する制御クラスとしての機能も持っていることを表わしている．P2，P6も同様である．

計算部品は，発信回数／受信回数の値にばらつきが見られる．そこで，1つのライブラリ用クラスを除いて，いくつかのグループに分類して，進化の様子を検討した．まず，計算部品は制御クラスとエンティティクラスの2種類に分類できる．この分類は，どのようなクラスとメッセージ送受信を行っているかによるもので，制御クラスを，2つ以上の計算部品とメッセージ送受信を行うクラスと定義し，エンティティクラスは，フレームワークを共に構成している表示部品以外に，ライブラリ用クラスと，ただ1つの計算部品とメッセージ送受信を行うクラスと定義した．さらに，制御クラスは管理クラスとトリガークラスに分類できる．管理クラスは，他の計算部品と双方向のメッセージ送受信を行うクラスであり，トリガークラスはメッセージの送信のみで，受信を行わないクラスである．表5.2では，制御クラス，トリガークラス，エンティティクラスについて，各々クラス名の後に+，++，#をつけた．また，CXXは計算部品，PXXは表示部品，LXXはシステムのライブラリ用クラスである．括弧内の値は，基本的な計算部品，表示部品間のメッセージ送受信を除いた回数を表わす．表5.2に見られるように，計算部品では，クラスの役割の違いによる進化の差異は明確ではない．ところで，利用者の要求変更は，利用者インターフェイスに40%以上を占めていると言われる[12]．表示部品は他の部品に比較して規模の変動が大きく，再設計による規模の縮小の機会もあまりないかもしれない．これは，表示部品が個々のインターフェイスに特化しており，抽象化による機能の整理があまりできないためであろう．これに対して，数百回のメッセージ送受信を行っている計算部品の場合は，一時的に規模が大きくなっても，システムの仕様変更の過程で再設計の機会があり，クラスの抽象化によって規模を縮小させることができていた．ライブラリ用クラスがあまり変化しないのは，その変更が，多くのクラスに影響を与えるためであろう．開発者にとっては，このようなクラスの変更はできるだけ避けたいはずである．ライブラリ用クラスに，システムの進化の状況に依存しない構造を持たせるためには，開発当初に詳細な設計を行うことと，機能を絞り込んで規模を小さくする必要がある．

5.2 入金消し込みシステムによる進化モデルの検証

5.2.1 システム概要

ここで調査対象としたシステムはあるサービス業者の入金消し込みシステムである．全行程9ヶ月の開発を一人の開発者が担当し，その間，顧客への提示は4回行われた．4回のリリースは，各々本稼働するものとは限らず，最初の1版はシステムの仕様を検討するためのプロトタイプであった．このプロトタイプは，使い捨てられたわけではなく，次の第2版へ引き継がれたクラスもある．

なお，このシステムは，VisualSmalltalkと関係データベースを用いて構築されたが，計測対象とした部分は，このうち，VisualSmalltalkを用いて開発された部分だけであり，関係データベースとのインタフェース部分は計測対象から外している．

各版の開発にかけられた日数と開発者が行った主な作業内容を次にまとめた．

ver.1 システムのフレームワーク構築

開発期間は約2ヵ月（全体工数の約30%）．顧客との設計に基づいてシステムのフレームワークを開発．データベースとは未接続で，システム単体のフレームワークの検証を行うためのプロトタイプ．

ver.2 使い勝手の修正と変更

開発期間は約 2ヵ月（全体工数の約 29%）．ver.1 の基本機能開発部分の継続．ほぼ全体の機能を開発し終わり，顧客へ試用を依頼したシステム評価版．

ver.3 仕様変更を受け付け，最終的な仕様を作り込み

開発期間は約 2ヵ月（全体工数の約 25%）1ヵ月の試用期間を経て得られた仕様変更と機能拡張を実装後に正式な稼働環境で動作するシステムを開発し，顧客へ引き渡した．仕様変更は 59 件寄せられ，そのうち GUI に関する変更が 70%で，機能に関する追加／削除／変更が 30%になっていた．これらの変更は，画面上の個々の部品の位置，実行中のデータベース更新のタイミング，メニューのマスキングなどの使い勝手に関する変更である．

ver.4 使い勝手のブラッシュアップ

開発期間は約 10 日（全体工数の約 16%）．2ヵ月間の試用期間を経て，新たな仕様変更と追加を実装．この開発は別途開発契約に基づいて開発された．仕様変更は 6 件寄せられ，そのうち GUI に関する変更と機能に関する変更はそれぞれ 50%ずつであった．

以上の様に，開発期間中の作業量は各期間で差がある．しかし，開発プロセス，開発組織が，1995 年度の調査対象に選択したシミュレーションシステムの開発プロセスと似ているため，人に依存する進化過程と人に依存しない進化過程を導くために適切な事例であると考えて，これを調査する対象として選択した．

5.2.2 進化モデルの検証

入金消し込みシステムについて，4 つの版にシステムの進化メトリクスを適用して計測を行った．計測結果は，第 4章にまとめてある．ここでは，入金消し込みシステムの計測結果に基づき，シミュレーションシステムで観測された進化の形態を検証したので説明する．検証は，シミュレーションシステムで確認した各々の進化の形態を順に示し，入金消し込みシステムでは，どのような進化の形態を観測できたかを述べながら進める．

システムの進化形態

- システムの進化は，時間軸に沿って一様ではなく，S 字の曲線を描く

  第 1 版を 1 として，各版でどのように変化したかを図 5.4に示した．システムの進化は，時間軸に沿って一様ではなく，明確な S 字の曲線を観察できたわけではない．これは，システムの開発過程で受けた仕様変更の内容，システムのアーキテクチャ，開発過程の進め方に依るもので，基本的には S 字曲線を描いていると考えられる．

  1. 立ち上がり部分の欠落について

     まず，システムの成長曲線が異なる理由を開発過程の違いから考察してみる．シミュレーションシステムは，1ヵ月毎に顧客にシステムを提示し，ver.3 まではシミュレーションを構成する設備の増加などの仕様変更があった．つまり，シミュレーションシステムは，開発前の仕様が明確ではなく，実現可能性を確認しながら徐々に仕様を膨らませた漸進型開発過程に基づいて開発された．これに対して入金消し込みシステムは，開発前にシステムのフレームワークの設計，機能の洗い出し，優先順位づけを行った後に開発を始めた．そのため，ver.2 の段階で，ほぼ最終システムの仕様が決定され，この後の ver.3 と ver.4 でのシステムの進化は，使い勝手の改善を中心とした開発になった．この開発過程の違いは，特に最初の S 字の立ち上がりの仕方に現われていると思われる．開発の当初から開発すべき内容が明らかであった入金消し込みシステムは，最初の緩やかなシステムの成長が必要なかったと考えられる．

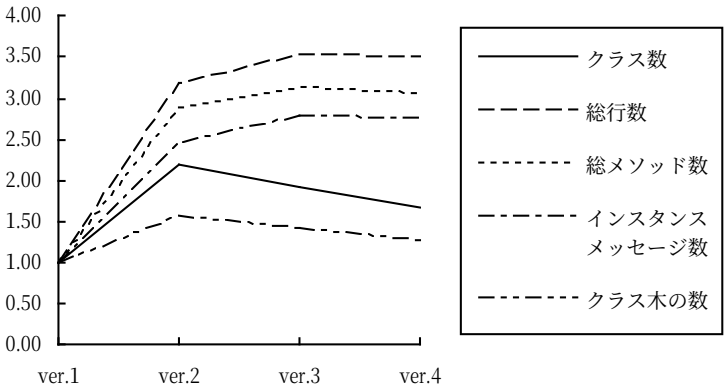

図 5.4: システムの S 字の成長曲線

2. クラス数の右下がりのカーブについて

   図 5.4で，特にクラス数が ver.2 以降で減少しているのは，S 字の形状には合致しない．クラス数の減少は，ver.2 から ver.3 へ至る過程で受けた仕様変更がきかっけとなって複数のクラスが統合されたりしたことと，暫定的なデータベース管理クラス群などが，別に開発されていたデータベース管理クラスの完成と共に不要になり削除されたことなど，再設計や設計の整理が行われた影響によるものである．したがって，計測した値が右下がりになっているのはシステムの成長の停止や退化によるものではないと言える．

3. クラス木の右下がりの傾向について

   シミュレーションシステムは，クラス木の数がシステムの進化過程で変化していたのはフレームワークの変更を意味していたが，入金消し込みシステムでのクラス木の変化は，暫定的なデータベースのモデル部分と再利用ライブラリクラス群の削除によるものであり，フレームワークの変更によるクラス木の追加／削除ではない．したがって，クラス木の数が減少しているのも，本質的なシステムの成長が止まったり，退化したということを意味しているのではない．

4. 総行数などのカーブの右上がりの傾向について

   さらに，ver.2 から ver.3 へ至る過程で，全体工数の 25%が開発に費やされている．この開発で，59 件分の仕様変更に対応するシステムの成長が観測されるはずであるが，図からは，そのようなシステムの成長を観察することができない．入金消し込みシステムのアーキテクチャでは，59 件の仕様変更の 70%を占める利用者インターフェイス部分が，VisualSmalltalk のコンポーネントである PARTS の組み合わせによって構成されている．PARTS を用いたプログラミング部分は，今回の計測範囲から外れているので，実際に開発した量は図に反映されていない．したがって，ver.2 から ver.3 へ至る過程で行数やメソッド数，メッセージ数が増えていないのは，システムの成長の量が正確にグラフに出ていないためと考えられる．それでも，緩やかな右上がりの傾向が観察できる．これは残りの 30%の仕様追加分による成長が現われていると思われる．

以上の議論から，図 5.4は，最初の立ち上がりを除いた S 字の成長曲線の一つを表わしていると考えることができる．

- 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，行数の分布は，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

  ver.3 で実稼働システムが完成しており，ver.4 で行われた変更項目は少なかった (6 項目)．3 つのメトリクスに基づいて計測された結果は，平均値と分散が増加した．クラスの総数は，ver.3 から ver.4

にかけて 71 から 62 へ減少したが，メソッド数の総数は変化していない．このことから，ver.3 から ver.4 へ至る平均値の増加は，6 項目の変更を行ったときに，クラスの統合などの再設計を行った結果だと思われる．

- 1 メソッドの行数の分布は，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

  入金消し込みシステムでは，クラスの再設計が行われても，1 メソッドの行数の平均値，中央値については，進化の過程でほぼ一定に保たれていた．しかし，分散，歪み度，尖り度は小さくなる傾向を見せていた．

- 1 メッセージ名に対するメソッドの定義回数の平均値，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，分散は大きくなっていく傾向を持つ

  ver.2 で，他に比べて大きい平均値，分散，尖り度が観察されたが，その後は ver.1 の値に戻っているので，ver.2 で観測された値は作り込み時期の一時的な現象であると思われる．したがって，入金消し込みシステムでは，1 メッセージ名に対するメソッドの定義回数の平均値，中央値，分散は，システムの進化の状況に関わらず変化していないと言える．

- 計測値の分布は，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

  分布の形状は，右に尾を引き，尖った形状を持っている．ただし，システムの進化に従ってさらに長くなるという傾向は必ずしも観察されるとは限らない．1 クラス当たりの行数，メソッド数ではその傾向を観察できたが，1 クラス当たりのインスタンス変数の数，1 メッセージ名に対するメソッドの定義回数，1 メソッドの行数では，この傾向は観察されなかった．

  シミュレーションシステムでこのような傾向を観察できたのは，顧客から要求された仕様変更を特定のクラスに取り込み続けた結果であると思われる．入金消し込みシステムでは，特定のクラスへの仕様の集中を観察することができなかった．

  インタビューを行ったところ，開発者は，

  「ver.1 のリリース時に確定した入金消し込みモデルの基本設計は自信作で，それ以降は機能拡張と調整が作業のほとんどを占めた」

  と，フレームワークの妥当性を評価していた．フレームワークがその後の仕様変更にどの程度頑健に対応できるかによって，ここで挙げた進化モデルの性質が発現するかどうかが決まるようである．

- 計測値の分布において分散が大きくなっていくのは，最大値を持っているオブジェクトが，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からである

  これは，分散が大きくなる現象を観察できなかったので，検討できない．

- 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

  入金消し込みシステムでは，この傾向は観察できなかった．入金消し込みシステムでは，開発初期にできるだけ多相性が大きくなるようにメッセージ名を選択し（ver.2 で最大），その後にメッセージ名に区別をつけて可読性を上げていく方法を選択したようである．

- クラスのライブラリから計測した継承の深さの平均値，中央値は，システムの進化の状況に関わらず変化せず，最大値は高だか 4 で，進化によって急激に深くなるということはない

  この進化は入金消し込みシステムでも観察できた．

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第 4章の表 4.14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

  全ての版について，各クラスのメソッド数，変数の数，行数を対象に，相関分析を行った．その結果，行数とメソッド数の間に強い正の相関が認められた．

- 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

  行数／メソッド数をクラス毎に計測した集合をクラス木毎に分類し，検定のための標本とした後，一方のクラス木の平均値をμ_0とし，他方のクラス木の平均値μとたとき，両者の間に有意差があるかどうかを検定した．検定に用いた方法は，それぞれのクラス木について分散が等しくないと仮定した 2 標本による有意水準 5%の両側 t 検定である．クラス木は全部で最大 11 あったが，標本数が 20 未満のクラス木については検定の対象から除いた．Smalltalk のライブラリクラスの直下に定義されたサブクラスを持たないクラスは，その他のグループに分類した．

  クラス木の行数／メソッド数の平均値，分散，標本数を表 5.3に示し，検定結果の P 値を表 5.4に示した．有意水準を 5%に設定したので，表 5.4で P 値が 0.05 以下の値を示している検定結果については，クラス木間のμ_0とμに有意差がある．また，P 値が 0.05 以上の値を示している検定結果については，クラス木間のμ_0とμとが異なるとは言えない場合である．

  検定結果から，ListModel 木と PickupModel 木の間，Adaptor 木，DBObject 木，および PARTS 木の間には有意差がないが，その他のクラス木の間では，有意差が認められた．

  以上の検定結果から，行数／メソッド数の値からクラス木を特定することはできないが，あるクラスが一つのクラス木に属していることがわかっている場合，そのクラスの行数／メソッド数の値とクラス木の平均値との偏差から，そのクラスの設計の妥当性を評価することができるだろう．

- 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

  この傾向は必ずしも観察されたとは言えない．

- 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

  クラスに関する計測値が小さいということは，クラスの役割も少なく，他のオブジェクトとの結合度も小さいクラスであり，他のオブジェクトに機能を提供するオブジェクトと考えることができる．他のオブジェクトを管理していない分，システムの進化の影響を受ける可能性が低いものと思われる．直観的に第 4章の図 4.6を見るとこの傾向を観察できるが，平均値からの離れと変化率の相関をさらに分析する必要がある．

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

  この進化も，開発するシステムを取り巻く状況によって，たとえメソッドの行数が外れ値になったとしても，再設計によるメソッドの分割は可能である．入金消し込みシステムでは，システムの進化によって，必ずしもクラス数が増えていたわけではなく，図 4.11に見られるように，メソッドの規模が大きくなっていく傾向も観察できなかった．むしろ，システムの進化によってクラス数が増加するのを防ぎ，1 メソッドの規模が大きくなるのを防ぐような再設計が行われたようである．これは，1 クラス当たりのメソッド数がシステムの進化に伴って増加していることから推測できる．

- 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

  直観的に第 4章の図 4.9を見るとこの傾向を観察できるが，平均値からの離れと変化率の相関をさらに分析する必要がある．

メッセージの進化形態

- 2 つ以上の応答可能なクラスを持つメッセージ名は，システムの進化に従ってその値は増加する

  入金消し込みシステムでは，第 4章の図 4.14に示すように，応答可能なクラスの数に波型の変化が観察された．必ずしも単調に増加するとは言えない．

5.2.3 オブジェクトの進化と進化の要因の関係

Belady & Lehman が提示したシステムの進化法則によると，システムは進化に従って規模が増大する性質を持っている．ところが，オブジェクト指向システムでは，たとえばクラスの数や 1 クラス当たりのメソッド数を例にとると，必ずしも増加し続けるとは言えないことが，このシステムの調査で明らかになった．そこで，どのような理由でクラスが削除されたか，あるいはメソッド数が変化したのか，その理由について開発者へアンケートを行った．

消滅したクラス

システムの進化過程で，定義された全 103 クラスのうち，削除されたクラスは 44 あった．理由別に集計した結果を表 5.5に示した．開発途中で不要になったクラスには，データベースとの接続前の暫定クラスで，別に開発されたデータベースとのインターフェースオブジェクトに置き換えられたものと，当初，他の開発者が開発したクラスを再利用していたが，過剰仕様であったため作り直した結果，そのクラスが不要になったものとがある．これらのクラスを「暫定版クラス」と「再利用後の再開発」に分類した．また，再設計で行われるのはクラス統合と分割である．クラスの統合と分割の方法には，2 つの手段がある．第一に継承関係を持たない他のクラスとの統合や分割で，第二に継承関係を持つクラスへの統合や分割である．前者を集約構造の変更，後者を継承構造の変更として表に示した．クラスの合計数が 44 になっていないのは，複数の理由による削除が重複しているためである．

オブジェクト指向のクラスの再利用には，継承や集約を用いた再利用が考えられる．しかし，再利用するクラスの仕様が過剰だった場合，仕様を軽くしたいときにクラスを作り替える手段がない．再利用したクラスが作り替えられたことによって削除された割合は，全体の 25.5%あり高い値を示していた．

メソッド数や行数が変化したクラス

メソッド数が変化したクラスについて，考えられる変更理由を我々が列挙し，その中から理由を選択してもらう形態でアンケート調査を行った．ここで再設計とは，クラスの抽象化によるクラス構造の見直しを指す．

変化が大きいと思われた全 17 クラスのうち，11 クラスが再設計によるもので全体の 60%以上を占め，要求変更の影響による変更を上回っている (表 5.6)．要求変更の影響を直接受けたクラスは，メソッド数や行数が増加する傾向があるが，開発者による再設計はこのような単調な増加傾向を抑える役割をもっている．したがって，クラスの規模の拡大の傾向を捉えて提示することで，システムの作りかえるべき時期を示唆できるだろう．

クラスの抽象化や具象クラスの追加による間接的な再設計によって行数やメソッド数に変化があった主なクラスを表 5.7にまとめた．表 5.7では，ClassD1*が ClassD11 から ClassD15 までのスーパークラスである．同様に，ClassD2*は ClassD21 と ClassD22 の，ClassD3*は ClassD31 の，それぞれスーパークラスである．スーパークラスの再設計にともなって，そのサブクラスのメソッド数や行数は減少し，スーパーク

ラスが持っているメソッド の数や行数の値は増加する傾向がある．ClassD2*の再設計の影響が，ClassD21 や ClassD22 に及んでいないように見えるが，これは ClassD2*の抽象度が高く，アプリケーションに特化したサブクラスの仕様を吸収できなかったものと思われる．

要求変更によって当該クラスに直接影響によるメソッド数に変化があったクラスを表 5.8にまとめた．進化による値の増加傾向が観察できる．

表 5.1: クラス木の行数／メソッド数の平均値，分散，標本数

クラス木名	標本数	平均値	分散
計算部品	36	5.66	0.70
表示部品	52	8.66	3.25
編集部品	33	16.15	10.38

表 5.2: CNOM の変化量とメッセージ受信量，送信量の計測値

クラス名	CNOM 変化量	送信回数	受信回数	発信／受信	受信 CBO	送信 CBO	ライブラリ CBO
C1#	-13	189(159)	180(26)	1.05(6.12)	2	2	3
C2+	-10	1164(1024)	1458(94)	0.80(10.89)	9	6	3
C3++	4	12(12)	21(12)	0.57(1.00)	2	2	0
C4#	-2	90(90)	10(4)	9.00(22.5)	2	1	1
C5+	-2	454(454)	287(287)	1.58(1.58)	6	5	2
C6++	0	21(19)	9(4)	2.33(4.75)	2	3	3
C7++	0	14(14)	30(24)	0.47(0.58)	2	2	0
C8*	0	0(0)	10(4)	0.00(0.00)	2	0	0
C9#	0	27(27)	11(8)	2.45(3.38)	2	1	0
C10+	0	76(76)	51(17)	1.49(4.47)	4	3	2
C11++	0	49(49)	8(3)	6.13(16.33)	2	2	2
C12+	0	94(74)	55(13)	1.71(5.69)	3	3	2
C13++	0	3(3)	7(4)	0.43(0.75)	2	1	0
L1*	0	0(0)	206(205)	0.00(0.00)	5	0	0
L2*	1	0(0)	1502(1502)	0.00(0.00)	10	0	0
L3*	0	0(0)	295(295)	0.00(0.00)	6	0	0
L4*	0	0(0)	225(225)	0.00(0.00)	2	0	0
P1**	50	344(190)	30(0)	11.47(−)	1	1	3
P2++	30	2284(790)	230(90)	9.93(8.78)	2	2	3
P3**	13	34(0)	0(0)	−(−)	0	1	0
P4**	9	96(96)	0(0)	−(−)	0	1	0
P5**	-6	6(0)	0(0)	−(−)	0	1	0
P6++	4	183(180)	90(90)	2.03(2.00)	1	3	0
P7++	4	9(0)	490(160)	0.02(0.00)	4	1	1
P8**	2	6(0)	0(0)	−(−)	0	1	0
P9**	1	52(10)	20(0)	2.60(−)	1	1	1
P10**	-1	16(0)	11(0)	1.45(−)	1	0	0
P11**	0	6(0)	0(0)	−(−)	0	1	0
P12**	0	5(0)	1(0)	5.00(−)	1	1	0

表 5.3: クラス木の行数／メソッド数の平均値，分散，標本数

クラス木名	標本数	平均値	分散
Adaptor	22	10.530	13.876
DBObject	39	12.267	18.376
ListMode	l 25	7.192	4.997
PARTS	36	10.528	21.000
Model	37	11.695	109.472
Pickup	25	7.602	3.958
その他	28	6.068	1.918

表 5.4: クラス木の行数／メソッド数の分散が等しくないと仮定したときの 2 標本による t 検定結果の P 値

	Adaptor	DBObject	ListModel	PARTS	Pickup	その他
Adaptor	-					
DBObject	0.105	-				
ListModel	0.001	0.000	-			
PARTS	0.999	0.095	0.000	-		
Pickup	0.002	0.000	0.497	0.001	-	
その他	0.000	0.000	0.036	0.000	0.002	-

表 5.5: 消滅したクラスの理由別の割合

	不要		クラスの統合と分割		名前変更	合計
	暫定版クラス	再利用後に再開発	集約構造	継承構造	適切名発見	
クラス数	6	12	12	2	15	47
割合 (%)	12.8	25.5	25.5	4.3	31.9	100

表 5.6: クラスの変更理由別の割合

理由	% (件数)
再設計	64.7 (11)
要求変更	29.4 (5)
機能拡張	5.9 (1)

表 5.7: クラスの抽象化や具象クラスの追加による間接的な再設計

		ver.1	*ver.2*	*ver.3*	*ver.4*
ClassD1*	行数	91	82	327	327
	メソッド数	9	17	55	55
ClassD11	行数	178	178	78	78
	メソッド数	10	10	8	8
ClassD12	行数	169	169	67	67
	メソッド数	9	9	7	7
ClassD13	行数	604	604	634	636
	メソッド数	46	46	58	58
ClassD14	行数	142	142	68	68
	メソッド数	9	9	7	7
ClassD15	行数	128	128	60	60
	メソッド数	8	8	6	6
ClassD2*	行数	90	300	310	318
	メソッド数	16	16	16	16
ClassD21	行数	102	255	282	282
	メソッド数	12	18	19	19
ClassD22	行数	9	111	125	125
	メソッド数	3	9	9	9
ClassD3*	行数		378	476	478
	メソッド数		46	64	64
ClassD31	行数	80	80	49	49
	メソッド数	10	10	9	9

表 5.8: 要求変更によって当該クラスに直接影響

		ver.1	*ver.2*	*ver.3*	*ver.4*
ClassR1	行数	45	45	258	258
	メソッド数	7	7	21	21
ClassR2	行数		1519	1902	2090
	メソッド数		99	139	145
ClassR3	行数		332	541	575
	メソッド数		22	40	42
ClassR4	行数	277	222	252	252
	メソッド数	44	27	32	32
ClassR5	行数	85	85	160	160
	メソッド数	11	11	11	11

5.3 証券システムによる進化モデルの検証

5.3.1 システム概要

ここで，調査対象としたシステムは一般企業の財務部等に向けた比較的小規模の有価証券管理システムである．開発には，４名の開発者と 12ヶ月の期間がかけられ，VisualSmalltalk が用いられた．開発初期にはプロトタイプが作成されたが，その後全体の詳細設計が行われており，実質的にはこのプロトタイプは破棄された．

このシステムは，他の２システムのように，顧客に提示しながら仕様を変更していく開発プロセスは採用されていない．したがって，この開発プロセスから得られた結果は進化過程と呼ばず，組織化過程と呼ぶことにする．

開発は，全体の分析を行い，設計を行った後，４名の開発者に設計仕様が配布されて進められた．また，この開発期間中，担当者は週一回の定例会議を行って，担当している部分についての進捗報告，およびクラスのインタフェースの再検討を行った．そのため，開発途中の成果物は，ソースコードでリポジトリに管理されており，多くの版が残されていたが，1 週間単位で区切ったものを 1 版分として計測対象とした．計測対象となった版は，プロトタイプ版 (1 版) と実装版 (14 版) 合わせて 15 版である．

さて，開発過程のうち最後の約 1ヵ月間が検証および妥当性確認の期間として割り当てられた．4 名の担当者は，システムを構成する 3 つの大きなクラス階層木と，その他の部分に分けて，担当部分をそれぞれ開発した．

このシステムが調査対象として選択された理由は，次の通りである．

- 開発プロセスの過程が残っている
- 開発者によって，開発箇所が明確に分割されており，人に依存した組織化過程を捉えることが可能である
- 仕様変更が外部から与えられていないため，開発中の変化はすべて開発者の設計上の意図によるものと判断できる

5.3.2 進化モデルの検証

証券管理システムについて，プロトタイプの版を除いた 14 の組織化過程にシステムの進化メトリクスを適用して，計測を行った結果は，第 4章に詳細をまとめて示してある．ここでは，主な計測結果と共に，計測結果に基づいて進化モデルを組織化の過程と比較して検討したので説明する．

システムの進化形態と組織化の形態

- システムの組織化過程

  図 5.5は，２番目の版以降は１週簡単の計測値を表わしている．この図に示されているように，組織化の過程では，S 字の成長曲線は観察されない．これは，入金消し込みシステムで，S 字の立ち上がりがない曲線とも異なった形状を持っている．最初の版は廃棄されたプロトタイプの規模を表わしており，それ以降の版は，プロトタイプの機能を大幅に拡張し，製品化を目指したシステムの仕様を設計した後，開発を進めた組織化の過程を表わしている．

  最初の版から 2 番目の版への過程で計測値が減少しているのは，プロトタイプを廃棄してシステムの作り直したためである．入金消し込みシステムで観測したようなクラス数の減少は，クラスの再設計と暫定版クラスの削除によるものであった．また，漸進型開発でプロトタイプを拡張していったシミュレーションシステムでは，システムの成長が S 字に進み，ここで見られるような一時的な成長の後退は観測されていない．

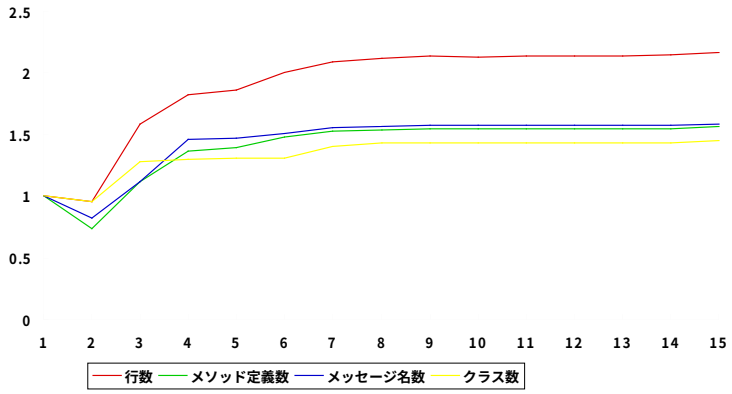

図 5.5: システムの組織化曲線

2 番目以降で観測された証券管理システムの組織化過程では，システム外の利用者からの要求変更が与えられなかった．だから，システムの成長は一定の割合で進み，やがて成長が止まった．成長が止まっている期間が最後の 2 か月というのは，長いと思う．これは，開発当初に決められた設計内容に大きな変更がなかったことを意味し，組織がほぼ完成してからシステムの設計内容を変化させるようなトリガーもなかったことを意味していると考えることができる．システムが進化するためには，要求変更や環境の変更などの外部からの働きかけが必要である．この働きかけによって，開発者はシステムの構造の再設計を行い，次の進化の段階へ向けて準備をする．このことは，シミュレーションシステムや入金消し込みシステムにおける再設計の起き方によって確認した．ところが，証券管理システムで観察される組織化の過程ではシステムの仕様を変化させる働きかけがないので，開発者による進化も起きていないと考えられる．

- 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，行数の分布の進化は，システムの進化の影響をほとんど受けず，変化しないが，分散は大きくなっていく傾向を持つ

  組織化の過程で選択された方法は，まず最初にメソッドが定義されていない開発すべきクラスを全て定義し，次に各担当者がクラス内の必要なメッセージ名だけを定義し，スケジュールに従ってメソッドのコーディングを進めていくものであった．したがって，開発が始まってから 1 か月未満の時期に 1 クラス当たりのメソッド数の平均値，分散は安定し，1 か月を過ぎた頃から 1 クラス当たりの行数，変数の数の平均値や分散は変化していない．

- 計測値の分布は，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

  分布は右に尾を引く，尖った分布を持っているが，組織化過程では，先に述べたように，設計の変更がなかったため，尾が伸びるような現象を観測することはなかった．

- クラスのライブラリから計測した継承の深さの平均値，中央値は，システムの進化の影響をほとんど受けず，変化せず，最大値は高だか 4 で，進化によって急激に深くなるということはない

  証券管理システムは，ライブラリクラスから数えた最大の深さは 7 であり，クラス階層は他の 2 システムに比べて深めである．証券管理システムの 7 階層は，金融商品の概念を整理したモデル化で定義されている．この最大深さは，組織化の過程で深くなることはなかった．継承が深くなったのは，証券管理システムが金融商品をモデル化したという特性によるものであり，一般的な傾向とは考えにくい．

  表 5.9にライブラリクラスから数えた最大の深さのヒストグラムのデータを示す．

表 5.9: ライブラリクラスから数えた継承の深さによるクラスのヒストグラム

継承の深さ	Proto.	ver.1.00	ver.1.01	ver.1.02	ver1.03	ver1.04	ver1.05	ver1.06
1	6	16	16	17	18	18	20	21
2	20	14	20	20	20	20	21	22
3	25	36	45	45	45	45	50	50
4	33	13	20	20	20	20	21	22
5	8	4	12	12	12	12	12	12
6	0	3	3	3	3	3	3	3
7	0	2	2	2	2	2	2	2

クラスの進化形態と組織化の形態

- 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

  表 5.10に相関分析の結果を示した．メソッド数，変数の数，クラス行数の間に正の相関が見られるが，特に，クラス行数とメソッド数の間に強い正の相関が見られる．

表 5.10: クラスの性質を表わす計測値の相関分析結果:証券管理システム

	メソッド数	クラス行数	変数の数
メソッド数	1.00		
クラス行数	0.79	1.00	
変数の数	0.68	0.51	1.00

- 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

  行数／メソッド数の平均値をクラス別に計測し，各版に定義されていたクラスの集合を検定のための標本とし，クラス木毎に分類した後，一方のクラス木の平均値をμ_0とし，他方のクラス木の平均値μとの間に有意差があるかどうかを，それぞれのクラス木について分散が等しくないと仮定した 2 標本による有意水準 5%の両側 t 検定を行った．

  各担当者が担当したクラス群の行数／メソッド数の平均値，分散，標本数を表 5.11に示し，検定結果の P 値を表 5.12に示した．ただし，検定では，標本数が少なかった開発者 4 のデータは除いてある．

  P 値の値を比較すると，開発者 2 と開発者 3 の平均値に差異は認められないという結果が得られる．130 余りのクラスは，4 名の開発者によって開発された．このうち，共通のスーパークラスを持つクラス群 2 つに分け，それぞれのクラス群を開発者 2 と開発者 3 が担当して別々に開発した．検定の結果，開発者 2 と開発者 3 の平均値に差異は認められなかったことから，同じクラス木に属するクラス同士で，共通の 1 メソッド当たりの行数の値を持つという性質は，開発者に依存する性質であると言うことはできない．

その他の組織化の形態

メソッドの組織化については，コードの意味を議論しなければならない．今年度の研究では，コードの内容まで立ち入った調査を行わなかった．また，メッセージの多相性に関する組織化についても，どのような手段を用いて議論すべきかも含めて，今後の課題としたい．

表 5.11: クラス木の行数／メソッド数の平均値，分散，標本数

開発者名	標本数	平均値	分散	担当したクラス木の数
開発者 1	42	17.71	128.247	10
開発者 2	31	5.96	6.528	6
開発者 3	37	5.59	7.532	2
開発者 4	12	12.95	145.955	3

表 5.12: クラス木の行数／メソッド数の分散が等しくないと仮定したときの 2 標本による t 検定結果の P 値: 証券管理システム

	開発者 1	開発者 2	開発者 3
開発者 1	—		
開発者 2	0.000	—	
開発者 3	0.000	0.568	—