継承木進化における統計的特性

中谷 多哉子[†] 玉 井 哲 雄[††]

オブジェクト指向設計では，プログラムの変更を局所化する目的で抽象クラスのメソッドに実装を集中させたり，プログラムに柔軟性を持たせる目的で具象クラスにメソッドを実装したりして拡張性を考慮した設計を行う．設計するときには，継続的に発生する要求変更の特性によっていずれかの継承を選択しなければならない．設計された継承構造を評価する研究は，これまで，データ型や再利用といった観点からなされてきたが，定量的に継承の妥当性を評価する手法についてはあまり研究されていない．我々は，システム開発事例から得られた時系列の成果物に対して，メトリクスを適用した計測結果を収集し，統計的手法を用いて解析を行った．その結果，要求変更に伴って変更される継承木の定量的特徴を明らかにすることができた．解析結果によると，継承木に属するクラスから求めた 1 メソッド当たりの行数の平均値には，継承木間で有意な差があり，継承木に属するクラスが変更されても保持され続ける性質をもつことが明らかとなった．本稿では，これらの継承木進化の定量的な特性を示し，設計上の意味を議論する．

A Study on Statistic Characteristics of Inheritance Tree Evolution

Takako NAKATANI [†] and Tetsuo TAMAI[††]

Inheritance hierarchies in object-oriented systems are designed in consideration of reusability and extendibility. For example, localizing methods in an abstract class is a way of design to cope with the program changes and implementing method in a concrete class is another way of design for flexibility. The designers select the design alternatives according to the directions of the system requirement changes. We cannot say many studies have been done for evaluating the suitability of the design of inheritance hierarchies from quantitative view points, but from qualitative view points as the data type theory or reusability evaluation. We observed the characteristics of inheritance hierarchies from quantitative viewpoints. Firstly, we collected the products measurements by applying evolutionary metrics and secondly, analyzed the measurements statistically. Finally we could reveal the quantitative characteristics of evolution process of inheritance hierarchies. As the result, each inheritance hierarchy has its own value, the mean number of lines of code per method. There is a significant difference among the values of hierarchies. Furthermore, there is not significant difference between the values of the neighboring versions of a hierarchy. In this paper, we show the evolutional characteristics of inheritance hierarchies quantitatively and discuss their meanings from the design activities.

1. はじめに

漸進的開発が求められるシステム[5)]とは，利用者の要求変更に対応するために継続的に開発されるシステムである[11)]．オブジェクト指向では，当初より漸進的開発を取り入れることが重要であると言われてきた[7)]．1990 年中期以降，オブジェクト指向は急速に企業で導入され始め[9)]，少なからぬシステムがエンドユーザへ提供されるようになった．これらのシステムが，今後，多くの要求変更を受け入れなければならないことを考えると，漸進的開発を意図した設計手法の研究はますます重要になっていくと予想される．

しかし，OMT(Object Modeling Technique) や OOSE(Object Oriented Software Engineering) を代表とするオブジェクト指向開発方法論は，新規開発における問題抽出，問題理解，モデル化を主な対象としており，漸進的開発を対象とした提案を行って来たとはいえない[10),18)]．

漸進的開発では，機能を追加する設計だけではなく，既に開発された部分の再利用性向上や拡張性向上に留

† SLagoon
e-mail: tina@slagoon.to
†† 東京大学大学院総合文化研究科広域科学専攻
Graduate School of Arts and Sciences,
University of Tokyo
e-mail: tamai@graco.c.u-tokyo.ac.jp

意した設計変更も行わなければならない．オブジェクト指向の継承は，分類，特殊化，一般化，類似に着目してクラスがグループ化され，システム進化，すなわち再利用性向上や拡張性向上を目的として使われる[19]．クラスを既存の継承構造の中に追加するときには，クラスの追加によって型誤りを発生させない設計が必要となる．これには，B. Liskov らによる型継承に基づく設計上の制約が参考になる[12]．データベースの分野では，クラスの設計変更はスキーマ進化として研究されている．それによると，クラスの設計変更作業は，クラス内部の変更と継承構造の変更，クラスの追加／削除に分類でき[1]，さらに，クラス内部の変更はインスタンス変数，メソッド，およびクラス間の関連の変更に分類することができる[6]．これらの設計変更に関する研究は，開発方法論を補足する重要な研究ではあるが，いずれも定性的な設計指針の提示である．

定量的な設計の評価に対しては，メトリクスの研究者によって行われてきた．S. Chidamber と C. Kemerer が提案したオブジェクト指向メトリクスは CK メトリクスとも呼ばれ，メトリクスの特性を解明する研究が進められている[2),4]．しかし，設計変更のための手法とは結びつけられていない．

実際に行われたシステムの開発過程を，プログラムコードの分析や開発者へのインタビューによって調査すると，様々な設計変更の軌跡を得ることができる．漸進的開発のための設計手法の研究は，このような設計変更によって得られた軌跡を定性的，定量的に観測し，どのような設計がどのような状況において行われるのかを識別することから始めなければならないと考える．我々は，このような設計変更の軌跡を，開発状況や利用者の要求変更といった，環境変化にオブジェクトが適応する進化と捉えて，定性的，定量的に観測を続けてきた[14),16),17]．観測対象はシステム，クラス，メソッド，継承木といったレベルを選択している．

本稿では，漸進的開発のための設計手法を提案する第一歩として，設計変更の定量的観測を行う進化メトリクスを提案し，進化メトリクスを適用したオブジェクトの進化過程の観測結果を示すとともに，進化の定性的な特徴が定量的にどのように観測されるかを示すことにする．また，定量的な観測結果が示す定性的な意味を解釈する．本稿で議論する進化の観測対象には，継承木を選択した．継承木は，再利用性向上や拡張性向上，機能追加の各場面で開発者の手が入る対象であると予測したからである．

本稿の構成は次のとおりである．2で観測で用いる用語の定義を行い，3ではシステムレベルで観測された定量的な進化過程を紹介し，続く4では，4つのシステムを用いて行った観測結果とその解析結果を報告する．5では継承木の定量的な性質について考察し，最後に関連研究を紹介する．

2. 定　　義

2.1 進化メトリクス

進化メトリクスには二つの提案が含まれている．第一の提案は，その計測と解析の方法である．メトリクスを適用した計測は時系列に行い，定性的な調査結果との関連づけを行いながら解析を進める[15]．第二の提案は，進化を定量化するために使う定量尺度である．

進化メトリクスでは，S. Chidamber と C. Kemerer によるクラスの複雑度を定量化するメトリクス WMC(Weighted Methods per Class) や，可読性を定量化するメトリクス DIT(Deapth of Inheritance Tree)[3]，そして，M. Lorenz と J. Kidd によって提案されたクラスに定義されているメソッドの行数やメソッド数，クラスにおけるメソッドの行数の総和（クラスの行数と定義する）[13] を用いる．ただし，進化メトリクスで用いる WMC には，メソッドの重みを 1 と考え，クラスに定義されているメソッド数と同じ値を用いる．また，DIT には，直近のライブラリクラスから数えた継承の深さを用いる．これは，進化の観測をアプリケーションのクラスに限定するためである．これらのメトリクスは計測が容易であるだけでなく，設計変更による影響をよく表す．たとえば，メソッドの行数は，可読性の改善のための改行の挿入なども計測値の変化として観測することができる．

2.2 継　承　木

継承木を観測対象とするにあたり，継承構造に基づく次の基準を設けてクラスを分類する．

- ライブラリクラスを除くすべてのクラスにつき，そのクラスを根とする継承部分木に属するクラスの数が 5 以上のとき，その部分木を観測対象とする．
- システム進化の過程で，クラス数が増加し，その数が 5 以上となる継承木は，システム進化をさかのぼって継承木の成長過程を観測する．

2.3 版

継承木の進化過程は，システムの版ごとに観測する．システムの版とは，ある要求に対して開発された時系列で得られる成果物である．本研究で観測したシステムのうち，実際に顧客向けに開発されたシステムでは，システムの版が定期的に顧客へリリースされている．そこで，リリースされた成果物を「版」として識別し

た．社内開発やプロトタイプのように，リリースという明確な基準がないものについては，定期的に保存されていた成果物をそれぞれ「版」として識別した．

3. システムレベルの進化

オブジェクトの進化過程の調査を行うと，開発者が将来発生すると予想する要求変更に対して，どの程度の労力が必要となるかを予測しながら，意思決定を行う[16)]．開発現場では，設計の再検討はリリース時期を守りながら行わなければならないという制約がある．したがって，開発の善し悪しを，スナップショットから得られた計測結果だけに基づいて評価するのは適切でない．個々の開発状況を考慮するためには，時系列で観測を行う必要がある．システムレベルで進化の過程を定量的に観測すると，開発者の意思決定の成果を得ることができる．次に例を示そう．

例 1：第二版の開発で，まだシステムに定義されていたクラスの数が 20 未満である時点では，フレームワークを変更した形跡があった．しかし，システムの開発が進み，クラス数が 40 を越えた版以降では，フレームワークの変更は観測されず，特定のクラスの規模が増大し続けるという現象を観測した[14)]．

フレームワークはシステム全体の構造を決定しているため，フレームワークを変更すると多くのクラスに影響が及ぶ．開発者へインタビューしたところ，開発の初期に定義したフレームワークが想定していた以外の要求変更があり，フレームワークを変更する必要が明らかとなったが，少数のクラスに変更を集中させることで，定められた期日内に顧客へリリースしたことが明らかとなった．

例 2：継承木内に限定したクラスの継承構造の変更や抽象化／具象化の作業が，システムのリリース毎に行われていた．継承木の設計変更後，スーパークラスがメソッド数を順次増加させ，要求変更が抽象クラスへのメソッドの追加で対応できている様子を観測した[17)]．

クラスの変更は，情報隠蔽されたカプセル内の変更であり安価である．抽象化，具象化といった変更の影響は継承関係にあるクラスに止めることが可能である．しかも，クラスの抽象化やそれに伴う新たな具象化によって，将来の要求変更に対するシステムの柔軟性を高めることができる．開発者へのインタビューから，この開発では，新たな要求の追加や変更によってフレームワークの設計変更が必要になることはなく，定められた開発期間内に，再利用クラスの開発や拡張性を向上させるための設計変更を行えたことがわかった．

前者の例は機能追加重視型の開発例であり，後者はクラス構造の洗練化重視型開発の例である．これらの開発過程は，定量的な観測結果に統計モデルを適用することによって，視覚的に把握できる．たとえば，メソッドの行数や１クラスに定義されているメソッドの数の頻度分布は，継承木ごとに層別すると負の二項分布へ適合させることができる[17)]．

負の二項分布の確率密度関数 $f(x)$ は，成功確率 p，失敗確率 qのもとで，r 回目の成功が出現するまでのベルヌーイ試行回数 Xを確率変数として，

$$f(X) = {}_{X-1}C_{r-1} \cdot p^r \cdot q^{X-r},$$

で与えられる．ただし，$X = r, r+1, r+2, \ldots$である．また，その期待値と分散は，それぞれ $E(X) = rq/p$，$V(X) = rq/p^2$となる．

メソッドの行数の頻度分布が負の二項分布に従うということは，コーディングを行うプログラマの作業と関連づけて，次のように解釈することが可能である．

「プログラマのコーディング作業は，第三者にはプログラム行の集合から無作為に行を取り出して付加するというプロセスとして観測される．一つのメソッドが完結するためには，プログラム行のうち，ある特定の性質をもったものがちょうど r個選ばれなければならないと仮定し，プログラム行の集合の任意の一つの要素がその性質をもつ確率が p であると仮定すると，メソッドの行数は負の二項分布に従う.」

システムの進化に従って得られる計測値の分布の変化を，負の二項分布の 2 つの係数 p および rの変化として観測すると，

- 機能追加を重視する開発では，p と rがともに減少する傾向があり，
- クラスの分割や継承木内のクラスの抽象化／具象化といったクラスの洗練を伴う開発では，p と r がともに増加する傾向がある．

負の二項分布のプログラミング作業に対応させた解釈から，p が大きいほど，メソッドの完結に必要な特定の性質を持ったプログラム行が，メソッド全体に占める割合が高くなるので，プログラミングの自由度は減り，定められたプログラム構造やプログラミングスタイルの強制力が増すと解釈できる．また，p が大きい場合に，全体のプログラム行数が変わらないとすると，必然的にメソッドの完結に必要な性質をもったプログラム行の数 r も大きくならざるをえない．したがって，pr平面の右上方向への変化は，プログラム記述の制約が強まり，左下方向への変化は制約が弱まること

表 1 システムの概要

System	SE	NK	SK	DK
版数	4	4	14	1
開発期間 (月)	7	7	3	-
開発人数	1	1	4	4
開発プロセス	漸進型	漸進型	落水型	落水型
クラス数	52	62	133	39
メソッド数	927	2644	1487	380
行数	8677	20470	14934	5371
開発言語	Smalltalk	同左	同左	Java

を意味する．このような解釈は，B. Liscov が議論した継承によるサブクラスの設計制約や，クラス構造の洗練によってコーディングの規約が多くなるという経験則とも一致する．

適合度検定によると，システム全体で得られた頻度分布は負の二項分布へ適合しないことがわかった．しかし，計測値を継承木に層別化することによって，計測値は負の二項分布へ適合した．このことは，継承関係に着目した層別化が，計測値を同様の性質を持つデータへ層別化する手段として妥当であることを示唆している．このことから，継承木を用いたクラスの層別化は，統計的にも意味を持つと考えられる[17)]．

4. 観　　測

4.1 観測対象の概要

継承木の進化過程を調査するために選択したシステムの開発概要を表 1に示す．計測対象としたシステムは，SE: シミュレーションエディタ，NK: 入金消し込み，SK: 証券管理，DK: 書類管理の各システムである．

各システムを構成する継承木は次のとおり．

(1) SE システム
- 編集木：シミュレーションに必要な初期値を与える利用者インタフェースを構成するクラス群
- 表示木：シミュレーションの設備を表すアイコンなど，設備を視覚的に組み立てる利用者インタフェースを提供するクラス群
- 計算木：シミュレーションの計算エンジンを提供するクラス群

(2) NK システム
- 仲介木： 利用者インタフェースとデータベース上のオブジェクトとの仲介を行うクラス群．
- 表操作木：一覧表に表示されたオブジェクトの操作を行うクラス群．
- 永続化木：データベース上の永続オブジェクトに対応するクラス群．
- 表抽出木：データベース上のオブジェクトを一覧表に表示するために，データベースからオブジェクトを抽出するクラス群．

(3) SK システム
- 証券領域木：証券問題領域のクラス群で，5 つの部分継承木，およびその他のクラスからなる．5 つの部分継承木のうち 3 つを開発者αが開発し，残りの 2 つの継承木を開発者βが開発した．
- 仲介木：データベース上のオブジェクトと証券領域クラスとの仲介を行うクラス群で，開発者γが開発した 2 つの部分継承木を持つ．
- 定数表木：税率などの定数表を保持するクラス群で，開発者δが開発した．

(4) DK システム
- 表示木：書類管理情報の入力と参照のための GUI を担うクラス群．識別可能な継承木である表示部分木を含む．
- 書類管理領域木：書類の管理情報の構造に従って定義された問題領域のクラス群．ただし，書類管理領域部分木を含む．

4.2 観 測 結 果

横軸にクラスのメソッド数（CNOM），縦軸にクラスの行数（CLOC）をとった平面上に，個々のクラスをプロットして描いた散布図を図 1〜図 4に示す．それぞれの図は各システムに対応し，システム全体で求めた散布図と各継承木毎に求めた散布図を示してある．いずれのシステムでも CNOM と CLOC との間に強い正の相関関係があることを統計的に確認した．

4.3 観測のための視点

ここで，継承木の進化を定量化するために，次の用語を定義する．

C_c値　：クラスの 1 メソッド当たりの平均行数
C_t値　：継承木を構成するクラスの C_c値の平均値

図中の直線は，各継承木の C_t値を傾きとする直線である．また，図中の矢印は各クラスの版と版の間で変化した軌跡を表している．システム全体で求めた散布図には，いくつかの直線を識別することができるが，同一継承木に属するクラスを集めた散布図では線が一本しか見えていない点に着目して欲しい[14)]．C_c値と C_t値を用いて次の 3 つの仮説を提起し，検証することによって，継承木の進化による定量的な性質を捉えることを試みる．

(1) C_t値一致仮説：継承木の C_t値は継承木間で有意な差はない．

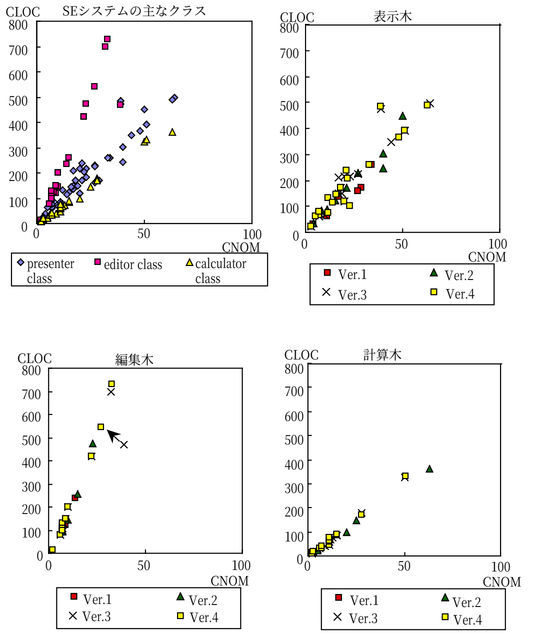

図 1　SE システムのクラスのメソッド数 (CNOM) に対するクラス行数 (CLOC) の散布図

Fig. 1　Scatter Diagram of CLOC against CNOM on SimulationEditor

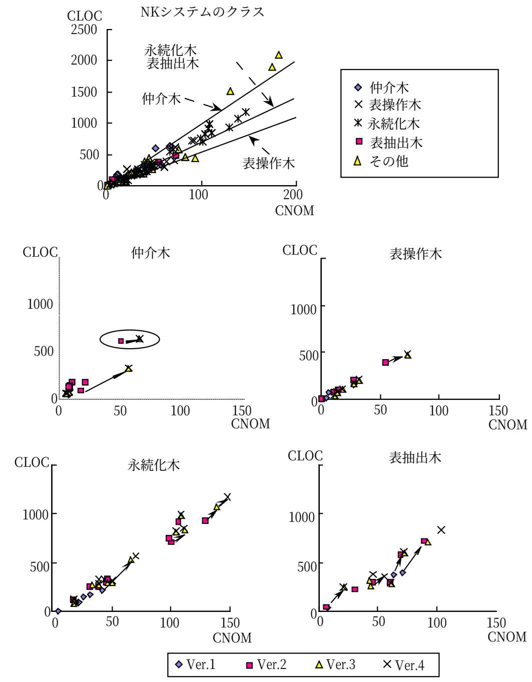

図 2　NK システムの CNOM に対する CLOC の散布図

Fig. 2　Scatter Diagram of CLOC against CNOM on Cash Receipts Transaction Management System

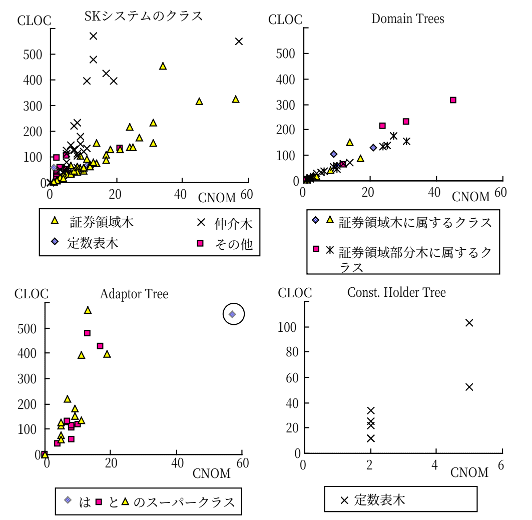

図 3　SK システムの CNOM に対する CLOC の散布図

Fig. 3　Scatter Diagram of CLOC against CNOM on Securities Management System

継承木間でその C_t 値に有意な差が認められる場合は，この仮説は棄却される．

(2)　C_c 値分布安定仮説：継承木に属するクラスで求められる C_c 値の平均値，および分散は，隣接する開発版の間で有意な差はない．

(3)　C_t 値開発者非依存仮説：継承木が同じならば，その C_t 値は，開発者間で有意な差はない．

4.4　継承木進化の定量的性質

4 つのシステムに対して行った仮説の検証結果から，以下の結果を得ることができた．

- 「同一システム内に定義された異なる継承木の C_t 値の間には有意な差が認められる」
  同一システム内の異なる継承木間で，クラスの C_c 値の分布を対象とした有意水準 5%の両側 t 検定を行った．その結果，異なる継承木の C_c 値の平均値（$= C_t$ 値）に有意な差が認められた．したがって C_t 値一致仮説を棄却する．
- 「隣接する版の間では，継承木内のクラスの C_c 値の平均値および分散に有意な差はない」
  SE システムと NK システムを調査対象として，隣接する版の間で計測した継承木に属するクラスの C_c 値に対して有意水準 5%の両側 t 検定および P 検定を行った．検定の結果から，いくつかの例外を除いて，隣接する継承木内のクラスの C_c 値の平均値および分散には有意な差があるとは言えないことを統計的に検証できた．参考として，表 2と表 3に 2 つのシステムの C_c 値に関する平均値と標準偏差の一覧表を示す．

検定で有意な差があったのは，NK システムで観

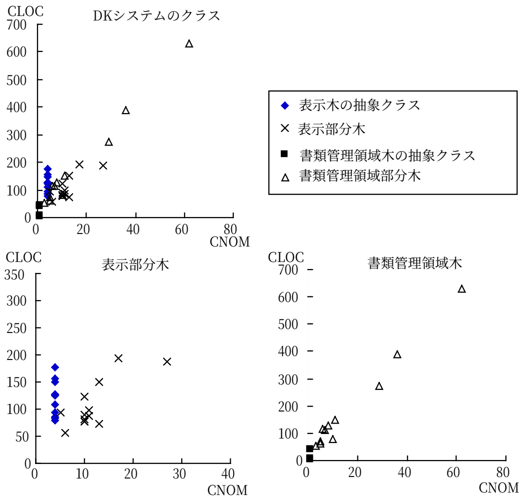

図 4 DK システムの CNOM に対する CLOC の散布図
Fig. 4 Scatter Diagram of CLOC against CNOM on Documents Management System

表 2 SE システムの継承木ごとに求めた C_c 値の平均値と標準偏差
Table 2 Average and standard deviation of C_c of each tree defined in SimulationEditor

表示木	Ver.1	Ver.2	Ver.3	Ver.4
平均 (C_t)	7.63	8.44	8.63	8.93
標準偏差	1.89	1.27	1.62	1.78
クラス数	6	8	16	18
編集木	Ver.1	Ver.2	Ver.3	Ver.4
平均 (C_t)	15.10	16.34	16.32	16.24
標準偏差	2.16	2.63	3.11	4.07
クラス数	4	6	11	12
計算木	Ver.1	Ver.2	Ver.3	Ver.4
平均 (C_t)	-	5.43	5.68	5.74
標準偏差	-	0.70	0.90	0.85
クラス数	-	6	15	15

測された仲介クラスの継承木における Ver.2 と Ver.3 の間である．この有意な差は，他の組織で開発されたデータベース管理クラスに対応するスタブに相当する 3 つのクラスの追加と削除が原因であった．3 クラスという規模は，継承木として観測するに十分な規模ではないため観測対象として独立させなかったが，その仕様から，明らかにそれらが属する継承木の他のクラスとは異なる性質を持つクラス群であると判断できた．3 クラスを除いた継承木では，版の間に有意な差を検出できなかった．

- 「クラスは，そのクラスが属する継承木の傾き C_t 値を持つ直線に沿って進化する」
  図 1，図 2の矢印から，
  – クラスの進化の方向は，継承木の C_t 値を表す直線の傾きに沿っているか，または，
  – クラスの進化の方向は，直線に向っている

  様子を観察できる．特に SE システムの編集木で観測されたあるクラスは，図 1の矢印で示すように，Ver.3 から Ver.4 に至る期間で，その C_c 値を継承木の C_t 値に向けて変化していた．開発者へのインタビューから，ここで観測された軌跡は，開発者が Ver.3 における設計に違和感を持ち，Ver.4 を開発する際に修正したクラスの進化の軌跡を表していることが明らかとなった．開発者の違和感は経験的な直観に基づくものであるが，Ver.3 における当該クラスの異常は，このクラスの C_c 値が，同じ継承木に属する他のクラスとは異なる C_c 値を持っていたという現象として，定量的にも観測できた．

表 3 NK システムの継承木ごとに求めた C_c 値の平均値と標準偏差
Table 3 Average and standard deviation of C_c of each tree defined in Cash Receipts Transaction Management System

仲介木	Ver.1	Ver.2	Ver.3	Ver.4
平均 (C_t)	8.04	12.29	9.06	9.07
標準偏差	1.80	4.53	1.64	1.64
クラス数	3	7	6	6
表操作木	Ver.1	Ver.2	Ver.3	Ver.4
平均 (C_t)	8.21	5.28	5.90	5.91
標準偏差	3.64	2.08	0.76	0.76
クラス数	4	11	7	7
永続化木	Ver.1	Ver.2	Ver.3	Ver.4
平均 (C_t)	4.97	7.08	7.37	7.49
標準偏差	1.26	0.78	1.10	1.13
クラス数	8	10	11	11
表抽出木	Ver.1	Ver.2	Ver.3	Ver.4
平均 (C_t)	5.77	6.80	7.67	7.77
標準偏差	0.30	1.17	2.37	2.38
クラス数	4	7	7	7

- 「継承木の C_t 値は開発者に依存せず，一定の値を持つ」
  SK システムでは，二人の開発者が証券領域木に属するクラスを分担して開発していた．属人的な差が継承木の C_t 値へ及ぼす影響について調査し，C_t 値開発者非依存仮説の検証を行った．開発者 α は証券領域 1，2 および 3 を開発し，開発者 β は証券領域 4 と 5 を開発している．そこで，それぞれの継承木の C_t 値に対して，有意水準 5%の両側 t 検定を行って，開発者による有意な差の有無を検定した．その結果求められた有意水準 P 値は，0.34 から 0.96 となり，いずれも 0.05 より大きく，継承木の C_t 値には開発者による有意な差を認められなかった．したがって，C_t 値開発者非依存仮

説を棄却することはできない．

クラスの C_c 値を観測すると，継承木には，他のクラスとは異なる性質を持つクラスを発見することもできた．それらのクラスを以下に説明する．

- 抽象クラス
  図3の左下の図で丸印をつけたクラスは他のクラスの C_c 値によって構成される分布から外れた値を持っている．このクラスは，2つの仲介木の共通のスーパークラスである．継承木を分類するとき，ライブラリクラス直下のクラスを継承木の頂点の候補としたが，2つの部分継承木の C_t 値には統計的な優位の差がなかったことから，このスーパークラスは部分継承木の頂点であるというよりも，2つの部分継承木が同様の定量的な性質を持つために定義された継承木の束ね役である．
- 若い継承木
  DKシステムの図4の表示木にはメソッド数4の複数のクラスが，また書類管理領域木には，メソッド数1のクラスが縦に並んでいる．これらのクラスはクラスライブラリの直下に定義されたクラスであり，将来の機能拡張の影響を受けないように具象クラスの機能を抽象化して定義されたクラスである．各スーパークラスは1つのサブクラスしか持たず，その継承構造は直線的である．将来は，個々の継承構造が継承木として発展することを想定して開発されていた．継承木の観測で，クラス数が5以上のクラスを選択するという基準を設けたのは，このような若い継承木を排除するためであった．しかし，若い継承構造が生育していく過程で，どのように継承木の定量的な性質が変化するかを観察するのは興味深い．今後の研究対象である．

5. 考　　察

以下に仮説の検証で得られた結果をまとめる．

(1) 継承木の C_t 値は継承木間で有意な差がある．

(2) 継承木に属するクラスで求められる C_c 値の平均値，および分散は，隣接する版の間で有意な差はない．

(3) 継承木が同じならば，その C_t 値は開発者間で有意な差はない．

継承木の性質を定量的に示す手段として C_c 値と C_t 値を検討し，それらの値によって表される継承木進化の定量的な特性を明らかにした．

仕様上，あるいは実装上，類似のクラスを分類することが，継承設計の主な作業である．したがって，異なる継承木に属するクラスは，仕様上，あるいは実装上，異なる性質を持つはずである．継承木の間で観測された C_t 値の有意な差は，継承木の分類基準の差異を表していると考えられる．

継承木内のクラスの C_c 値の分布が隣接する版の間で有意な差がないという性質は，クラスが，継承木の C_t 値の傾きを持つ直線に沿って進化していた帰結であった．これは，継承木内のクラスの設計が変更されても，設計変更の前後でも継承木自体のクラス分類基準には変化が起きないという継承木の進化的特性を表していると考えられる．クラスが継承木の C_t 値の傾きを持つ直線に沿って変化するという現象も，継承木自体のクラス分類の基準に変化がなかったことが原因であろう．開発者に依存せず継承木の C_t 値が一致していた例も，開発者が行った継承木の設計方針が同じであったためと考えられる．

以上の観測と解釈から，継承木の C_t 値は，継承木の設計方針を表す定量化尺度となり得ると期待される．また，クラスの C_c 値とそのクラスが属する継承木の C_t 値との差を示して，設計上の問題を指摘できる可能性もある．これらの考察を検証するためには，継承木の組み替えなどが発生するシステムの事例について，観測を進める必要がある．これは今後の研究課題である．

6. 関 連 研 究

オブジェクト指向システムを対象とした規模や複雑度を定量化する研究として，S. Chidamber と C. Kemerer の6種類のメトリクス[3)]や，Henderson-Selleres らによるオブジェクト指向システムのための複雑度メトリクスに関する研究[8)]を挙げることができる．

Lorenz と Kidd らの設計メトリクスに関する研究の特徴は，メソッド規模を計測するためのメトリクスを示しただけでなく，設計見直しの閾値を示した点にある[13)]．たとえば，メソッドの平均メッセージ送信数の閾値は言語にかかわらず9を，また，メソッドの平均行数の閾値として Smalltalk には6行，C++には18行，利用者インタフェースではそれらよりも多少大き目の数値になるとしている．しかし，Lorenz と Kidd の閾値の提案には次のような疑問がある．まず第一に，継承の種類によるメソッドの規模の影響を考慮する必要はないのか．第二に，一般的な閾値を用いるのでは，ソフトウェアの開発状況を考慮した評価ができないのではないか．平均的に示す閾値が，常に個々のクラス設計の定量的評価基準となり得るとは考えにくい．Lorentz らも，閾値を大きく外れる設計を不可

とするのではなく，計測値を問題抽出の補助手段として利用し，抽出された箇所については上級者に設計のレビューを依頼するように提案している．我々は，漸進的な開発過程に沿った時系列の計測を行うことで，徐々に規模が拡大され，複雑度が増していく過程を客観的に発見でき，そのような観測によって開発状況を考慮した設計評価が可能であると考える．

7. 結　　論

漸進的開発のための設計手法を提起するために，本稿では，従来から定性的に研究されてきた継承の設計に対して，継承木進化の計測方法と解析方法を示し，継承木進化における定量的な性質を明らかにした．本研究で取り扱った事例は数が少ないため，この研究だけで仮説が完全に検証されたとは言えない．しかし，オブジェクト指向における継承の性質から，継承木のC_t値が，開発者の直観的な設計方針を定量化する定量尺度となり，その安定性から，設計変更による継承木の設計のゆらぎを観測できる可能性は高い．今後は，C_t値の有効性を確認するために，設計者支援による有効性を評価していきたい．

謝　　辞

本研究を進めるにあたり，計測値の収集，解析にご協力いただいた友枝敦氏，松田晴美氏，システムの開発に携わり，多忙のなか，インタビューの時間を割いてくださった（株）SRA の技術者の方々に感謝いたします．

参 考 文 献

1) Banerjee, J., Kim, W., Kim, H. J. and Korth, H. F. : "Semantics and Implementation of Schema Evolution in Object-Oriented Databases," *Proc. of the ACM SIGMOD Annual Conf. on Management of Data*, pp. 311-322 (May, 1987).
2) Basili, V. R. and Melo, W. L. : "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, Vol. 22, No. 10, pp. 751–761 (1996).
3) Chidamber, S. R. and Kemerer, C. F. : "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, Vol.20, No.6, pp. 476–493 (1994).
4) Chidamber, S. R., Darcy, D. P. and Kemerer, C. F. : "Managing Use of Metrics for Object-Oriented Software: An Exploratory Analysis," *IEEE Transactions on Software Engineering*, Vol. 24, No. 8, pp. 629–639 (1998).
5) Davis, A. M. : A Strategy for Comparing Alternative Software Development Life Cycle Models, *IEEE Transaction on Software Engineering*, Vol. 14, No. 10, pp. 1453–1461 (1988).
6) Erradi, M., Bochmann, G. V., and Dssouli, R. : "A Framework for Dymanic Evolution of Object-Oriented Specifications," *Proc. of the 11th Conference on Software Maintenance*, IEEE, pp. 96–104 (1992).
7) Henderson-Selleres, B. and Edwards, J. M. : The Object-Oriented System Life Cycle, *Communications of ACM*, Vol. 33, No. 9, pp. 142-159 (1990).
8) Henderson-Sellers, B. : *Object-Oriented Metrics –Measures of Complexity–*, Prentice Hall, 1996.
9) 本位田真一, 青山幹雄, 深澤良彰, 中谷多哉子編著: オブジェクト指向システム分析・設計, 共立出版, 1995.
10) Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. : *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley (1992).
11) Lehman, M. M. and Belady, L. A. : *Program Evolution*. Academic Press, 1985.
12) Liskov, B. and Wing, J. M. : Specifications and Their Use in Defining Subtypes, *Proc. of Object-Oriented Programming Systems, Languages, and Applications*, ACM, pp. 16–28, 1993.
13) Lorenz, M. and Kidd, J. : *Object-Oriented Software Metrics*, Prentice Hall, 1994.
14) 中谷多哉子, 友枝敦, 酒匂寛, 玉井哲雄: オブジェクト指向システムの進化プロセスの定量的分析日本ソフトウェア科学会第 *13* 回大会論文集, pp. 389–392, 1996.
15) 中谷多哉子, 玉井哲雄, 友枝敦, 酒匂寛. : オブジェクト指向によるシステムの進化を表わすメトリクスの検討ソフトウェアシンポジウム *'96* 予稿集, SEA, pp. 52–62, 1996.
16) Nakatani,T., Tamai, T., Tomoeda, A. and Matsuda, H. Towards Constructing an Object Evolution Model *Proc. of Asia-Pacific Software Engineering Conference '97*, IEEE, Hong Kong, pp. 131–138, 1997.
17) 中谷多哉子, 玉井哲雄: オブジェクト進化の定量的観測値からの分布モデルの推定日本ソフトウェア科学会第 15 回大会論文集, pp. 293–296, 1998.
18) Rumbaugh, J., et al. : *Object-Oriented Modeling and Design*, Prentice Hall, 1991.
19) Wegner, P. : Concepts and Paradigms of Object-Oriented Programming, *OOPS Messenger*, Vol. 1, No. 1, ACM Press (1990).