社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－SE－152 (6)
2006／5／17

グラフ手法によるJavaプログラムの構造と構造変化の分析

木下喜幸 (東京大学総合文化研究科)
玉井哲雄 (東京大学総合文化研究科)

概 要

近年活発に研究/調査が進められている複雑ネットワーク (complex network) の研究の手法や視点を援用し，オープンソースのJavaプログラムの構造を測定した．その結果，ソフトウェアのグラフ構造はポワソングラフと比較して高いクラスタリング係数を持つ事が分かった．次数については，出次数は入次数と比較して小さな最大値を持つ事を実際に計測できた．また，出次数の次数分布は，係数が-4.5 近い指数分布に近いことを分析できた．

Analysing the structure of Java programs using graph metrics

Yoshiyuki Kinoshita(Graduate School of Arts and Sciences, University of Tokyo)
Tetsuo Tamai(Graduate School of Arts and Sciences, University of Tokyo)

Abstract

In this paper, some structure of Java programs are measured by the metrics used in the field of complex network study. As a result, it is shown that the clustring coefficient of real software graph is larger than relative poisson graph. And property of the maximum out-degree and the maximum in-degree is investigated. Finally, the analytical result that out-degree distribution of real software graph is alike exponential distribution with factor around -4.5 is shown.

1 はじめに

Javaで書かれたプログラムの各クラス/インターフェースの間の関係をグラフとして捉えるとどのような構造が見られるか，これが本研究の主題である．

グラフとして捉えて測定する際に，近年活発に研究/調査が進められている複雑ネットワーク (complex network) の研究 [1][2][3] のメトリクスを援用した．複雑ネットワークの研究は，俳優の共演関係，論文の共著関係，WWWのリンク構造，生態系などにおける関係をネットワークとして捉え，その構造を理論的解析やシミュレーション，実測により明らかにしていこうという試みである．これらの研究では，上記のような実際のネットワークや，それを模した理論的なネットワークの次数分布が，ポワソングラフ (次数分布がポワソン分布であるネットワーク) と比較して平均してより短い2点間の最短距離を持つこと (small world) などが調べられている．また，その次数分布モデルの候補としてべき分布や指数分布が挙げられ，その性質や実測したデータとの類似が調べられている．

ソフトウェアを対象としてこれらの視点や手法を用いた研究に [4] や [5] がある．[4] では，C++言語で書かれたプログラムのクラス間関係を無向グラフとして捉え，グラフの2点間距離がポワソングラフと比して非常に小さくなることを示した．無向グラフとしての分析なので，依存関係の深さ [6] などの以前から知られているオブジェクト指向メトリクスに対する直接的な結論は得られにくいが，ソフトウェアのクラスが「ランダム」に結合しあっているわけではない事を実測し興味深い結果を得ている．また，[5] では，C++言語で書かれたプログラムの次数分布のモデルを考察し，べき分布に近しいという結論を得ている．

Java言語によるプログラムの実装上の構造を見る研究としては，[7] がある．ここでは，少数のクラス間の依存関係を表したグラフ構造のうち，プログラム内に頻繁に現れるものを抽出し，それを実装上のパターンとして収集している．そして収集したパターンを実装者への手引きとし，優れたソフトウェアデザインの議論への土台となる事を期待している．

本研究では，プログラム全体をクラス間の依存関係を表すグラフとして自然に把握しやすいJava言語によるプログラムを対象として選ぶ．クラス間の依存関係を有向グラフとして把握し，複雑ネットワークの研究で用いられるメトリクスを利用して測定することで，プログラムの全体の構造に広く見られる特徴を捉える事を試みる．また，ひとつのプログラムの複数バージョンを対象とし，バージョン間に見られる共通性や変化の特徴についても調査する．

2 研究のアプローチ

2.1 基本的方針

本研究では，複雑ネットワークの研究のメトリクスを通してJavaプログラムのクラスとクラス間の関係をグラフ構造と捉えたときの特徴を測定する事を目的にする．ここでは，あるクラス (インターフェース) は

1. フィールドの型，メソッドの返り値，引数の型など，そのクラス (インターフェース) 内部で明示的に記述されるクラス (インターフェース)
2. 継承する親クラス
3. 実装するインターフェース

となるクラス (インターフェース) に関係がある，参照しているとする．明示的に記述されないクラスは考慮しない．グラフとして把握する際には，頂点をクラス (インターフェース) とし，上述の定義により関係があるクラス (インターフェース) 間には辺があるものとする．ただし，有向グラフとして捉えるので，クラス (インターフェース)i がクラス (インターフェース)j を参照している場合は $i \to j$ の関係があるとし，逆の関係 $(j \to i)$ は

無くても構わない．この際，対象がクラスであるかインターフェースであるかの区別はつけない．また，上述の関係の間の差 (関係が継承であるか実装であるかの差など) もつけない．実装者の立場から見て，あるクラス (インターフェース) を実装する際に把握するべきプログラム内の他のクラス (インターフェース)，という視点から見た構造を把握するためには，このような差異は無視しても構わないと考えた．対象の中には Java 言語の標準クラスライブラリは含めなかった．

2.2 対象プログラム

対象としたプログラムはオープンソースの Java プログラム開発プロジェクトである Apache Jakarta Project[10] に関連する以下の 3 つのプログラムである．

Tapestry Web アプリケーションのためのフレームワーク．2005 年 5 月 5 日リリースの alpha2 版から 2006 年 1 月 6 日リリースの最新版までの 19 バージョンを入手して対象とした．

Jmeter パフォーマンス解析ツール．1999 年 2 月リリースの 1.0.2 から 2003 年 8 月 17 日リリースの 1.9.1 までの 12 バージョンを入手して対象とした．

Ant ビルドツール．現在では Apache Jakarta Project からは独立して開発されている．2000 年 7 月 18 日リリースの 1.1 から 2005 年 7 月 2 日までの 11 バージョンを入手して対象とした．

これらは，複数のバージョンのソースコードをまとまって容易に入手しやすいので測定対象した．Apache Jakarta Project に関連して開発されたという点では共通しているが，測定結果の偏向をなるべく小さくするため，プログラムの対象ドメインがなるべく異なるプログラムを選択した．また，これらは開発陣容，開発日数，規模，リリースの頻度などの性質もそれぞれ異なるプログラムである．

また，これらのプログラムがバージョンの最初期を除いて数百〜千以上のクラス (インターフェース) 数を持っている事も選択の基準とした．後述するように，プログラムのグラフ構造をポワソングラフと素朴に仮定し比較する際には一定以上に大規模なプログラムである事が自然な前提となるが，これらのプログラムはそれを満たしていると考えることができる．

2.3 関係の抽出

実際の方法は以下である．まず，Java の StreamTokenizer クラスを用いてプログラム内の各 java ファイルを字句に分割する．その後，一字句毎に読み込み，字句「class」または「interface」に続く字句をクラス名，インターフェース名として抽出する．ただし，.class 表現やコメントなどを誤って抽出しないように工夫をする．このようにしてプログラム内部のすべてのクラス名/インターフェース名をすべて抽出する．再度一字句毎に読み込み，クラス (インターフェース) 記述の内部の字句にクラス名/インターフェース名が用いられていないか検証する．用いられている場合は，関係がある (参照している) とする．後に，厳密さを期すために Eclipse プラグイン [9] を用いて構文解析し，文字列が型名である保証をとった．得られた関係を隣接行列 A=(a_{ij}) の有向グラフとして表現し，クラス i がクラス j を参照しているならば，$a_{ij}=1$ とする．参照していないならば $a_{ij}=0$ とする．関係が継承，実装，その他である場合のグラフ構造上の区別はつけなかった．また，それがクラスであるかインターフェースであるかの区別もつけなかった．

2.4 測定する性質

次数 あるクラスの持つ他のクラスとの間の参照/被参照数はグラフ上では次数として表される．クラス i の次数 $deg(i)$, 出次数 (参照次数)$outdeg(i)$, 入次数 (被参照次数)$indeg(i)$ はそれぞれ

$$deg(i) = \sum_j a_{ij} + \sum_j a_{ji}$$
$$outdeg(i) = \sum_j a_{ij}$$
$$indeg(i) = \sum_j a_{ji}$$

として求められる．最大次数，最大出次数，最大入次数はそれぞれの最大値を，平均 (出，入) 次数はそれぞれの平均値を表す．

クラスタリング係数 クラス i が他の k_i 個のクラスとの間に参照/被参照関係を持つとする．もし，クラス i と関係のある k_i 個のクラス同士がまた，全てお互いに関係しているならば，これら k_i 個のクラスの間には $\frac{k_i(k_i-1)}{2}$ 個の関係が存在している事になる．もし，これら k_i 個の間の結合数の実測値が E_i 個とすると，その比 C_i

$$C_i = \frac{2E_i}{k_i(k_i-1)}$$

をクラスタリング係数と呼ぶ．また，グラフのクラスタリング係数は，グラフの全頂点のクラスタリング係数の平均値の事を指す．本研究では，クラスタリング係数をグラフのクラスタリング係数という意味で用いる．

2.5 ポワソングラフ

ポワソングラフは次数分布がポワソン分布にしたがうグラフである．ポワソン分布は以下のようなグラフである．まず，2 項分布を考える．2 項分布とは，2 種類の可能な結果をもたらす試行 A,B があり，それぞれの発生確率が p,$1-p$ である時に，A が x 回，B が $n-x$ 回生じるとした時の確率 $f(x)$ の確率分布である．

$$f(x) = C(n,x)p^x(1-p)^{n-x}$$

(ただし，$C(n,x)$ は n 個のものから x 個のものを選ぶ組み合わせ数とする)

2 項分布において，n の数が大きく，p が小さい場合，$np \to \lambda$ となるように $n \to \infty, p \to 0$ となるような極限では

$$C(n,x)p^x(1-p)^{n-x} \to e^{-\lambda}\frac{\lambda^x}{x!}$$

となる．この時

$$f(x) = e^{-\lambda}\frac{\lambda^x}{x!}$$

に従う確率分布をポワソン分布と呼ぶ．

ポワソングラフはクラスの数が大きく，クラス数に比してクラス間の関係の数が小さいような大規模なソフトウェアのグラフの構造の素朴なモデルとして用いる事ができる．本研究では，測定結果をポワソングラフと仮定した場合の数値と比較する事で分析を行う．

また，ポワソングラフのクラスタリング係数は

$$C_{poisson} = \frac{< k >}{N}$$

となる事が知られている．ただし，$< k >$ は平均次数，N は頂点数表す．

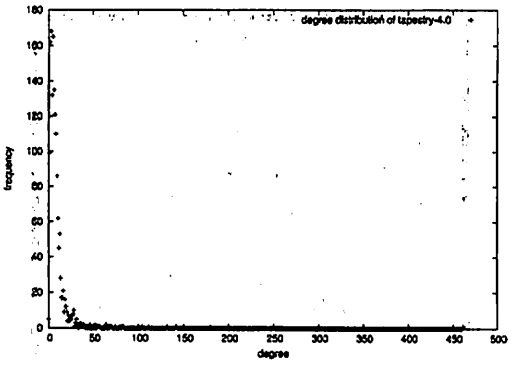

図 1: Tapestry(最新版) の degree 分布

3 測定結果

3.1 クラスタリング係数

各プログラムの平均次数，サイズ，クラスタリング係数を求め，同じ平均次数とサイズを持つポワソングラフのクラスタリング係数の理論値と比較した．Tapestry, Jmeter, Ant それぞれについての結果は表 1,2,3 の通りである．ただし，表中のサイズとはそのプログラムに含まれるクラスとインターフェースの総数，C_{real} は実測のクラスタリング係数，C_{rand} は同じサイズと平均次数を持つポワソン分布のクラスタリング係数の理論値を表す．

これらを見るとどのプログラムについても実測したクラスタリング係数はポワソン分布の場合と比べて 2 桁ほど大きなクラスタリング係数を持っていることが分かる．これは，実測したグラフに結合が密な部分が存在する事を推察させると同時に，プログラムのグラフの構造がポワソングラフと似かよっていない事を明確に示している．JMeter の最初期のものは，サイズが数十と大きく，理論値との差が大きくないが，サイズが 100 を超すにつれ，その差がはっきりしてくる．JMeter については，この間に Tapestry や Ant で見られるような構造が出来てきているとも推察できる．

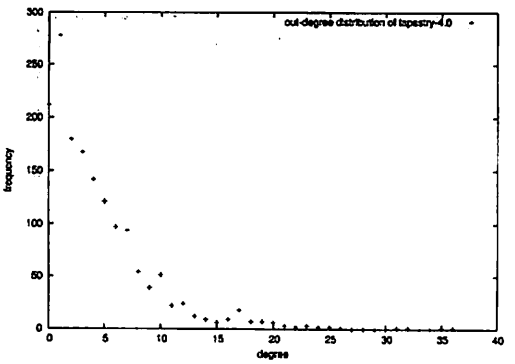

図 2: Tapestry(最新版) の出次数分布

3.2 次数分布の特徴

Tapestry の最新版の次数，出次数，入次数分布を見るとそれぞれ図 1,2,3 のようになる．

グラフから，出次数の最大値と入次数の最大値の違いが分かる．ここで出次数の最大値は 36，入次数の最大値は 451 となっている．Tapestry について，入手できたソースの最も古いものである alpha2 から最新版までの各バージョンについて，最大出次数と最大入次数の推移を見たところ，図 4，5 のようになる．

Jmeter，Ant について，出次数と入次数の最大値のバージョン間の推移を示したのがそれぞれ図 6,7 と図 8,9 である．

現実のネットワークでは，あるノードの持つ次数はネットワークのサイズ (グラフ内のノードの数) に比例すると素朴に仮定できる．実際，Tapestry について最

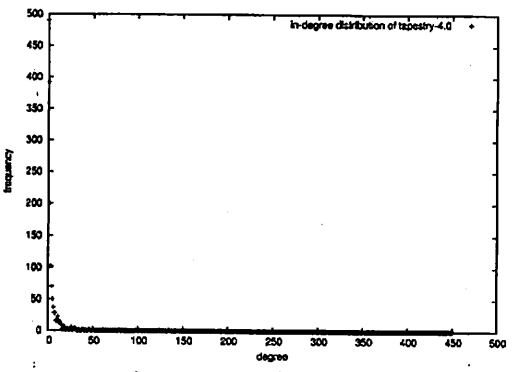

図 3: Tapestry(最新版) の入次数分布

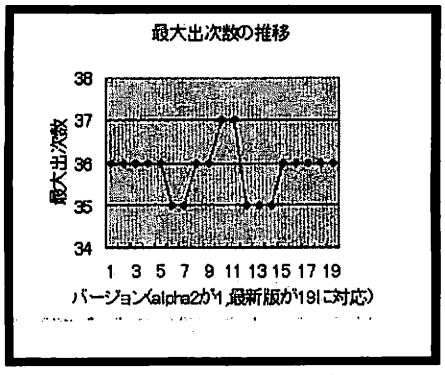

図 4: 最大出次数の推移 (Tapestry)

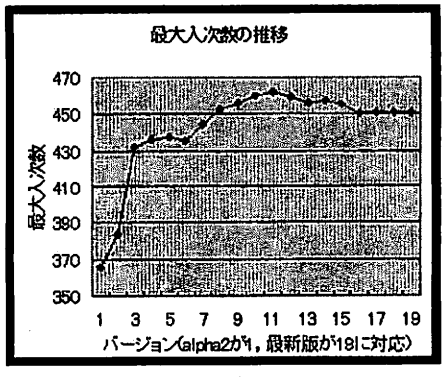

図 5: 最大入次数の推移 (Tapestry)

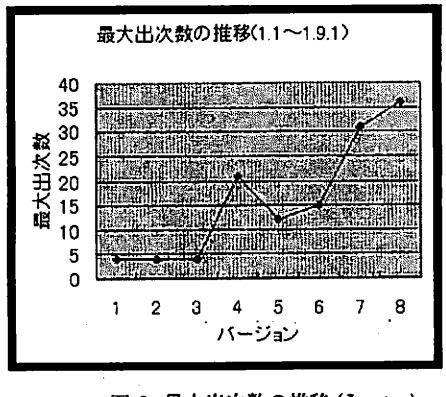

図 6: 最大出次数の推移 (Jmeter)

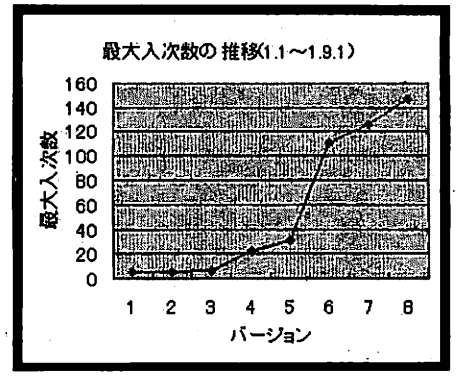

図 7: 最大入次数の推移 (Jmeter)

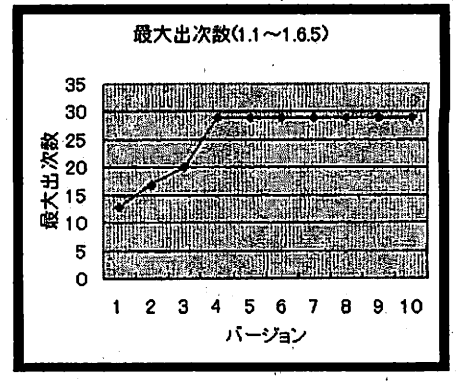

図 8: 最大出次数の推移 (Ant)

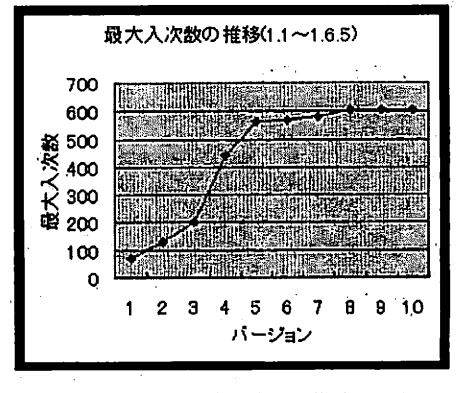

図 9: 最大入次数の推移 (Ant)

Ver.	サイズ	平均次数	C_{real}	C_{rand}
1	1302	8.26	0.210	0.006
2	1341	8.26	0.210	0.006
3	1486	8.55	0.203	0.006
4	1500	8.56	0.202	0.006
5	1505	8.54	0.202	0.006
6	1471	8.70	0.198	0.006
7	1485	8.74	0.198	0.006
8	1500	8.78	0.199	0.006
9	1508	8.82	0.199	0.006
10	1518	8.82	0.200	0.006
11	1525	8.84	0.200	0.006
12	1537	8.87	0.200	0.006
13	1545	8.88	0.200	0.006
14	1554	8.88	0.200	0.006
15	1565	8.92	0.200	0.006
16	1572	8.92	0.200	0.006
17	1573	8.92	0.200	0.006
18	1573	8.92	0.200	0.006
19	1573	8.91	0.200	0.006

表 1: Tapestry4.0 のクラスタリング係数 (Ver.1 が alpha2,Ver.19 が最新版を表す)

Ver.	サイズ	平均次数	C_{real}	C_{rand}
1	15	3.00	0.129	0.2
2	17	2.94	0.104	0.173
3	22	3.0	0.157	0.136
4	109	5.43	0.176	0.0498
5	176	6.34	0.153	0.0360
6	287	7.89	0.189	0.0275
7	383	8.10	0.186	0.0211
8	457	8.9	0.177	0.0194

表 2: JMeter のクラスタリング係数 (Ver.1 が Apache JMeter1.1,8 が jakarta-jmeter1.9.1 を表す)

Ver.	サイズ	平均次数	C_{real}	C_{rand}
1.1	108	6.78	0.287	0.0628
1.2	214	8.41	0.289	0.0393
1.3	502	8.10	0.268	0.0161
1.4	569	8.87	0.296	0.0156
1.5	915	8.55	0.255	0.00934
1.6.0	1226	9.22	0.262	0.00752
1.6.1	1232	9.23	0.263	0.00749
1.6.2	1262	9.31	0.266	0.00738
1.6.3	1304	9.38	0.266	0.00719
1.6.4	1304	9.37	0.266	0.00719
1.6.5	1304	9.37	0.266	0.00719

表 3: Ant のクラスタリング係数

Ver.	サイズ	最大入次数	係数
1	1302	366	0.281
2	1341	384	0.286
3	1486	432	0.291
4	1500	436	0.291
5	1505	437	0.290
6	1471	435	0.295
7	1485	444	0.299
8	1500	452	0.301
9	1508	456	0.302
10	1518	460	0.303
11	1525	462	0.303
12	1537	459	0.299
13	1545	456	0.295
14	1554	457	0.294
15	1565	455	0.291
16	1572	450	0.286
17	1573	451	0.287
18	1573	451	0.287
19	1573	451	0.287

表 4: Tapestry の最大入次数とグラフのサイズの関係

大入次数とクラス (インターフェース) 総数で測ったソフトウェアサイズの関係を見たものが表 4 である．ここで係数は最大入次数をサイズで割った値である．サイズや次数の変動率に対して，係数の変動率はより安定である事から，最大入次数はソフトウェアのサイズに依存するだろう事が分かる．表 4 の値について回帰分析をすると，決定係数は 0.87 になる．

しかしながら，出次数は必ずしもネットワークのサイズに比例した最大値を持っていない．Tapestry については，図 4 から分かる通り，出次数の最大値はほとんど変動していないが，この間，プログラムのサイズは表 5 のように推移している．

これは必ずしも最大出次数はグラフのサイズに関わらず同じ値を取る事を意味しない．実際，Jmeter の例 (図 6),Ant の例 (図 8) に見られるように，バージョン毎に値は大きく変化している．しかし，それでも次数に関わらずシステムサイズに比して大きくなっているわけではなく，30 前後で頭打ち傾向にある．この結果は，出次数は実装者が一度に把握しなければならない外部のクラス数と考えられるので，実装者の認知的な限界に基づいた制限があると捉える事ができる．認知的な限界に達する以前には，最大入次数と同様にシステムサイズが大きくなるにつれ大きくなっていくが，ある値に達すると最大値の増加が止まる．逆に，入次数は参照される数であり，実装者の認知能力に基づくそのような制限がないので，測定結果のようにシステムサイズによっては非常に大きな値が出ると考えられる．

3.3 次数分布のモデル

complex network の研究では，実際のネットワークの次数分布 $p(k)$ が多く測定され，それらがべき分布 $p(k) \sim k^{-\alpha}$ や指数分布 $p(k) \sim exp(-\alpha * k)$ のような形を取る事が報告されている ([1],[2])．そこで，測定したグラフの出次数，入次数分布を，べき分布や指数分布と比較する．

Tapestry の最新版の出次数分布を両対数グラフ，片対数グラフとして表したものを図 10, 図 11 に示す．関数の性質により，べき分布は両対数グラフが直線に指数分布は片対数グラフが直線になる．

両対数グラフのデータを線形回帰分析した場合，決定係数は 0.869 であり，係数は-0.19 であった．一方で，片対数グラフのデータを線形回帰分析した場合，決定係数は 0.936 であり，係数は-5.27 であった．図 11 を見ると，片対数グラフは次数が大きくなるにつれて，直線から離れた値を取る．そこで，データの 75 パーセンタイ

Ver.	サイズ	最大出次数
1	1302	36
2	1341	36
3	1486	36
4	1500	36
5	1505	36
6	1471	35
7	1485	35
8	1500	36
9	1508	36
10	1518	37
11	1525	37
12	1537	35
13	1545	35
14	1554	35
15	1565	36
16	1572	36
17	1573	36
18	1573	36
19	1573	36

表 5: Tapestry の最大出次数とグラフのサイズの関係

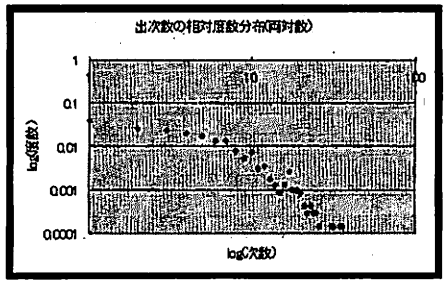

図 10: 出次数の相対度数分布 (両対数)-Tapestry 最新版

ル (第三四分位点) である次数 25 までのデータについて線形回帰分析をした場合，決定係数は 0.956 であり，係数は-4.55 であった．

入次数分布では出次数分布ほどよいフィッティングは見られなかった．両対数グラフは図 12, 片対数グラフは図 13, 次数 100 以下のものについての片対数グラフは図 14 に示してある．回帰分析の結果は両対数グラフのものについては次数を 20 以下に抑えても決定係数は 0.86 程度であった．片対数の場合は次数 20 以下に抑えた場合，0.89 程度であった．

出次数分布を，Tapestry の最も古いバージョン (alpha2) について両対数グラフ，片対数グラフで表すと図 15,16 のようになる．片対数グラフのデータを線形回帰分析したところ，決定係数が 0.899(係数:-5.21), 次数 25 まででは 0.951(係数:-4.49) と高い値を取った．また，Jmeter で出次数分布を指数分布との比較をしたところ，最新版について全データの場合決定係数 0.83, 第三四分位点のデータまでの場合，決定係数が 0.91(係数:-4.39) という値をとった．JMeter 最新版の出次数分布についての片対数グラフは図 17 に示す．同様に Ant 最新版について片対数グラフ (図 18) を線形回帰分析したところ，全データでは決定係数 0.83, 第三四分位点のデー

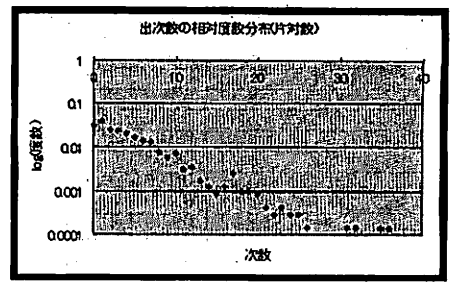

図 11: 出次数の相対度数分布 (片対数)-Tapestry 最新版

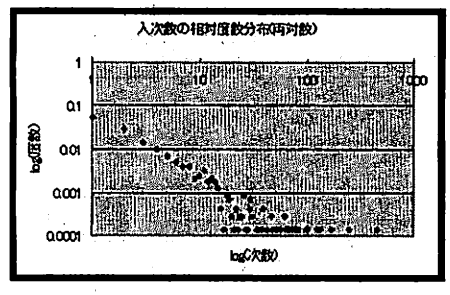

図 12: 入次数の相対度数分布 (両対数)-Tapestry(最新版)

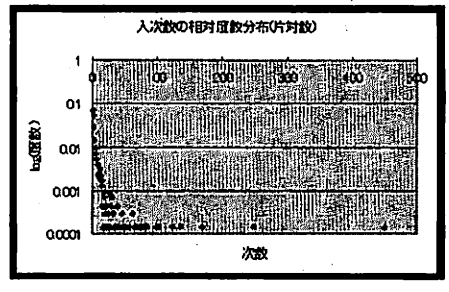

図 13: 入次数の相対度数分布 (片対数)-Tapestry(最新版)

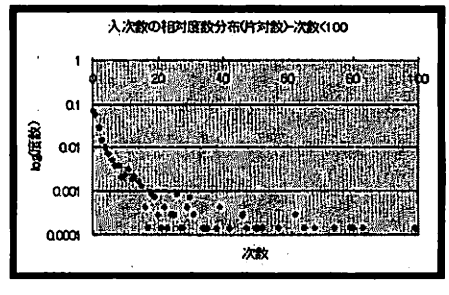

図 14: 入次数の相対度数分布 (片対数)-Tapestry(最新版):次数 100 以下について

タまでの場合は決定係数が 0.96(係数:-4.56) という値が出た．これらから，第三四分位点のデータまでの出次数分布は係数が-4.5 に近しい値を持つ指数分布と非常に類似している事が分かった．

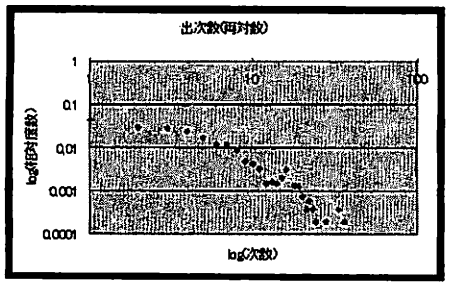

図 15: 出次数の相対度数分布 (両対数)-Tapestry(alpha2)

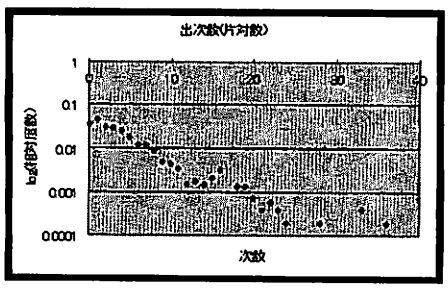

図 16: 出次数の相対度数分布 (片対数)-Tapestry(alpha2)

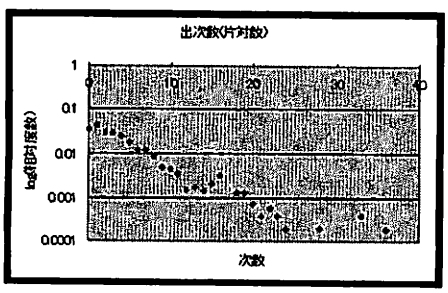

図 17: 出次数の相対度数分布 (片対数)-JMeter 最新版

4 まとめ

以上が，本研究で測定したグラフの特徴である．グラフの頂点間の最短距離，最大距離は現在測定中であるが，クラスタリング係数の値などから，ポワソングラフと比較して小さくなる事が予想される．頂点間の距離が小さ

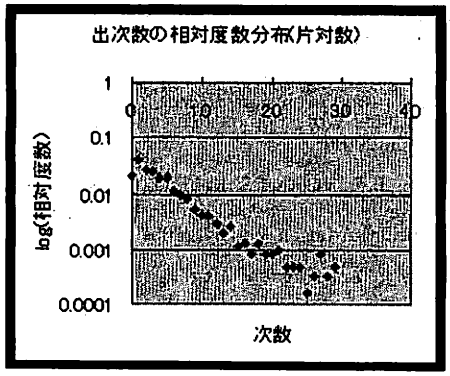

図 18: 出次数の相対度数分布 (片対数)-Ant 最新版

くなると，テストの際に追うべきクラス数を小さくできるので，ソフトウェア開発への直接的なメリットとなりうる．

また，特に出次数に対して，特に第三四分位点のデータまでの分布が係数-4.5 付近の指数分布に類似する事や，プログラムのサイズによらず，最大出次数が約 30〜40 で頭打ちになること，また，最大入次数はプログラムのサイズに比例して大きくなる傾向を持つ事は，実装前にソフトウェアの規模や構造を見積もるのに有効に利用しうる．ソフトウェアの変化をソフトウェア進化という言葉でとらえて，その法則について考察する研究がなされているが [8] 本研究の結果もソフトウェア進化の見通しを得るのに有効に使うことができる．

このようなグラフ把握により，実装前に出来上がるプログラムの構造に対して見通しが得られ，開発の各段階においてこの見通しを利用し，開発をより有効にする事が出来るようになることが期待される．

参考文献

[1] Albert,R.,BaraBasi,A.L.,Statistical mechanics of complex network,Review of Modern Physics,74,47-97(2002)

[2] Newman,M.E.J.,The structure and function of complex network,SIAM Review,45,167-256(2003)

[3] Watts,J.Duncan.,Small Worllds,Princeton University Press(1999)

[4] Valverde,S.,Sole,R.,Hierarchical Small World in Software Architecture, Santa Fe Institute working paper SFI/03-07-044(2003)

[5] Myers,C.R.,Software systems as complex networks: Structure,function,and evolvability of software collaboration graphs,Physical Review E, vol68,4,0461116(2003)

[6] Bellin,D.,Tyagi,M.,Tyler,M.,Object-Oriented Metrics:An Overview, Proceedings of the 1994 OOPSLA(1994)

[7] Gil,J.,Maman,I.,Micro Patterns in Java Code,OOPSLA 2005(2005)

[8] Lehman,M.M.,Ramil,J.F.,An approach to a theory of software evolution, Proceedings of the 4th International Workshop on Principles of Software Evolution(2001)

[9] D'Anjou,J.,Fairbrother,S.,Kehn,D.,Kellerman,J.,McCarthy,P., The Java™ Developer's Guide to ECLIPSE second edition, Addison Wesley(2005)

[10] http://jakarta.apache.org/