Vol.2017-SE-195 No.32
2017/3/13

開発委託先変更に対する ソフトウェアの複雑さ・不具合修正回数の関係

阿部晃佑[†1]　津田直彦[†1]　鷲崎弘宜[†1]　深澤良彰[†1]
杉村俊輔[†2]　保田裕一朗[†2]　二上直将[†2]

概要：ソフトウェア開発において、開発を外部委託する場合に、委託先が変更になることがある。一般に、このような開発委託先の変更は開発中のソフトウェアの品質に悪影響を及ぼすことが知られている。開発委託先の変更がもたらす悪影響には様々なものがあるが、そのすべてが明らかになっているわけではない。そこで我々は、開発を外部委託する際の開発成果物の品質評価の手法を検討するために、複数のプロダクトのメトリクス測定値を調査した。具体的には、ベースとなるソフトウェアのメトリクス測定値と最終成果物のメトリクス測定値と各プロダクトの不具合修正回数を調査した。調査の結果、ベースとなるソフトウェアの開発から開発組織を変更したプロダクトと変更しないプロダクトで、開発前後のメトリクス測定値の変化に差があることがわかった。具体的には、開発組織変更を伴う開発では複雑度の大きな関数がさらに複雑となり、複雑度の小さな関数は複雑度の増加が小さいこと確認した。一方、開発組織変更がない開発では不必要には複雑になっていないことが確認できた。これにより、ソフトウェアの複雑度が大きい関数を減らすことで開発組織変更を伴う開発時に複雑な関数がより複雑になる事態を未然に防ぐことが可能となる。また、修正回数は開発組織変更の有無に関連がないことが確認された。

キーワード：開発委託、ソフトウェアの複雑さ、プロダクトメトリクス、Cyclomatic 複雑度、修正回数

Investigation of the relationship between Cyclomatic Complexity and outsourcing organizational changes

Kosuke Abe[†1]　Naohiko Tsuda[†1]　Hironori Washizaki[†1]　Yoshiaki Fukazawa[†1]
Shunsuke Sugimura[†2]　Yuichiro Yasuda[†2]　Naomasa Futakami[†2]

Abstract: Occasionally organizations taking on assignment of software development are changed while outsourcing the same software development. Generally, such changes adversely affect software quality. However, the extent of adverse effects on software quality has yet to be elucidated. Herein we investigate the product metrics of industry products to evaluate software quality due to changes of organizations contracting for outsourced software development. In particular, we researched product complexity and number of code modifications. We found that outsourcing organizational changes affect measurement values of the product metrics. Specifically, Cyclomatic Complexity (CC) becomes higher upon outsourcing organizational changes, but not for products without outsourcing organizational changes. Using this knowledge, developers can minimize the degradation of software quality by ensuring that the measurement value of CC is kept lower than the predefined threshold before outsourcing. In addition, we confirmed that an outsourcing change increases difference in the metric values.

Keywords: Outsourcing, Product metrics, Cyclomatic Complexity, Code modifications

1. はじめに

ソフトウェア開発企業が、開発対象プロダクトの一部を別の開発企業に委託することは度々起こりうる。このような開発委託の目的として、人件的コストや時間的コストの削減などが挙げられる。しかしながら、必ずしも開発委託がコスト削減を達成できるとは限らない。なぜならば、プロダクトの開発委託はソフトウェアの品質を低下させ、その後の開発工程において不必要な手間がかかる可能性を上げてしまうためである。

ソフトウェア品質低下を起こす原因の一つとして、開発の途中で開発組織が変更することが挙げられる。開発組織変更によるソフトウェアの品質低下は、様々な既存研究で言及されている。Conway's Law [1][2] によれば、ソフトウェア設計は開発組織の構造を反映している。そのため、開発組織が途中で変更されたならば、異なる構造を持つ組織が開発を行うこととなり矛盾を生む原因となる。また、Muckos ら[3]の研究では、はじめに開発を行った技術者は、プロダクトに暗黙の知識やギャップを入れ込むことを示している。すなわち、開発者の多くが置き換わる開発組織変更はソフトウェア品質の大幅な損失を引き起こす可能性がある。

†1 早稲田大学
Waseda University
†2 小松製作所　株式会社
Komatsu Ltd.

開発組織の変更によってソフトウェア品質が低下する可能性があるのならば、開発組織変更を行わないことが一番の解決策である。また、開発組織の変更は開発組織変更の一部であるといえるため、開発組織変更も行わないことが望ましい。しかし、開発途中における開発組織の変更は様々な避けられない事情から起こるものであり、開発組織の変更を行わざるを得ないことも多い。そこで我々は、開発組織変更を伴う開発を品質低下させずに行うため、品質低下を起こしやすい要因を特定する調査を行った。

本研究で我々が着目したのは開発組織変更前時点でのプロダクトメトリクス値（以下、元測定値）と開発組織変更後のプロダクトメトリクス値の変化量(以下、前後差分値)と不具合修正回数の関係についてである。元測定値は、開発組織変更前時点でのソフトウェア品質を定量的に評価するためのものである。我々は、元測定値が品質基準を満たしているならば、開発組織変更による技術者の暗黙の知識やギャップは発生しづらいと考え、開発組織変更が行われた場合でも品質が保たれると仮説をたてた。不具合修正回数と前後差分値は、開発組織変更後の品質を評価するための値である。既存研究において、プロダクトメトリクス[4][5][6][7][8]や不具合修正回数[9][10][11]はソフトウェア障害の指標として多くの既存研究において用いられているので、本研究の調査対象として用いた。

以下に本研究で明らかにしたい目的を示す。

RQ1　元測定値と前後差分値の関係は、開発組織変更の有無によって違いがあるか？

RQ2　元測定値、前後差分値と不具合修正回数の間にはどのような関係があるか？

RQ3　元測定値、前後差分値と不具合修正回数の間にある関係は、開発組織変更の有無によって違いがあるか？

以下に、これらに対する本研究の貢献を示す。

1）開発組織変更を伴うソフトウェア開発において、元測定値が品質基準を満たしていなかった場合に、Cyclomatic 複雑度の前後差分値が大きくなる危険性があることを確認した。

2）開発組織変更を伴わないソフトウェア開発において、元測定値が品質基準を満たしている場合でもCyclomatic 複雑度の前後差分値が大きな値を示す結果が得られた。ただし、最終成果物としての品質は保たれているため、開発組織変更を伴う場合に比べて品質低下のリスクが低いことを確認した。

3）開発組織変更の有無にかかわらず、前後差分値が大きい関数は不具合修正回数が多く、開発組織変更の有無は不具合修正回数と関連がないことを確認した。

2. 背景

2.1 既存研究

開発組織変更がソフトウェア品質に影響を与えるかどうかについては、既存研究でいくつか言及されている。本稿では、開発組織変更は開発組織変更の一部であるとしたうえで、既存研究をふまえて我々の研究の新規性について述べる。Seiji et al. [12] は独自に origin という開発組織変更の変遷を示すメトリクスを定義して、ファイルを分類した。そのうえで、origin がプロダクトメトリクスやファイルの修正回数、欠陥数との関連を調査している。調査の結果として、開発組織変更はプロダクトメトリクスをより複雑にしてしまうこと、コード修正回数を増加させる傾向にあることを述べている。我々の研究では、「開発組織変更」がソフトウェアに与える影響に加えて「元測定値」と「前後差分値」に着目している。

2.2 動機

共同研究先である企業 A は、複数の外部企業へ開発委託を行っているが、コストの削減率が想定通りとは言えないのが現状である。外部委託開発を行ったソフトウェアの一例として、初めは外部開発組織企業 B と企業 C にそれぞれ開発を委託していたソフトウェアを、開発がひと段落ついた後の追加工程はすべて企業 B に委託するケースがあった。我々は、この事例において「企業 B がはじめから開発を続けたソフトウェア」と「企業 C が開発を行った後に企業 B が追加開発を行ったソフトウェア」の２群に分けられることに注目し、最終的な開発元が同じ場合でも、はじめの開発元が異なるならば品質低下の危険性が大きくなるのではと仮説を立てた。「開発組織変更前の品質」と「開発組織変更後の変化量」の関係は既存の研究でも明らかになっていない。この関係を明らかにすることで、開発組織変更によるリスクを未然に防ぐ手立てをたてることができる。

3. 提案

開発組織変更の有無による元測定値と前後差分値の関係を調査するにあたって、同程度の規模のプロダクト群を開発組織変更の有無によって二軍に分け、比較調査することを提案する。開発組織変更の有無による影響を調査するのであれば、ある一つのプロダクトについて開発組織変更せずに開発が終了した場合と、同一プロダクトが開発途中で開発組織変更を行った場合のデータを比較調査することが望ましい。しかし、そのような開発工程を経たプロダクトを調達することは至極困難である。そのため、別プロダクトではあるが同規模程度のプロダクトを用いる。

本研究では、開発組織変更がソフトウェア品質に与える影響を調査するために元測定値と前後差分値という二つの値を定義した。元測定値は開発組織変更をする前に測定できるメトリクスであり、開発組織変更が行われなかった場合には測定することができない。同様に、前後差分値の測定も不可能である。以下の項では、開発組織変更の無かったプロダクトにおける元測定値と前後差分値の定義と、それらの関連性を明らかにする手法について述べる。

3.1 開発組織変更

本節にて、これまでの項で述べてきた開発組織変更を具体的に定義する。本研究における開発組織変更とは、あるプロダクトを開発する組織そのものが全く別の組織に変更することである。これは、あるプロダクトにおける二つの開発工程間で開発組織が異なることを意味する。開発工程とは、開発の開始から完了まで、つまりは納品までを一つのまとまりとしたものである。そのため、ある一つの組織内での開発期間に一部の開発のみ技術者 A が開発に関わり、途中で技術者 A は開発に関わらなくなったとしてもそれは開発組織の変更として扱わない。

3.2 プロダクトメトリクス

はじめにで述べたように、我々はソフトウェアの品質を定量的に評価するための値であるプロダクトメトリクスを測定する必要がある。プロダクトメトリクスは、プロダクトそのものや、クラス、ファイル、関数のような様々な単位で測定することができる。単位別に測定できるメトリクスの例として、プロダクト単位では「総コード行数」、クラス単位であれば「クラス結合度」などが挙げられる。本研究では前後差分値という具体的な変化量を用いるため、プロダクトメトリクスとして測定できる最小の単位である関数単位のメトリクスを調査対象とした。関数単位で測定できるメトリクスとしては、「関数単位のコード行数」や「Cyclomatic 複雑度」、「最大ネスト数」、「コメント率」などがある。プロダクトメトリクスの測定には Understand という測定ツールを用いた。

3.3 元測定値と前後差分値

元測定値は、開発組織変更前時点でのプロダクトメトリクス値を指す、前後差分値は、開発組織変更前後でのプロダクトメトリクスの変化量である。これらの値は、開発組織変更を伴うプロダクトにおいては、開発組織変更をおこなう直前のプロダクトからメトリクスの値を測定することで元測定値を求めることができる、前後差分値は開発組織変更後の最終リリース時のプロダクトからメトリクスを測定し、その値と元測定値の差分を取ることで求めることができる。

一方で、開発組織変更を伴わないプロダクトでは開発組織変更直前のメトリクスを測定することができない。そこで我々は、比較調査を行うことを前提に開発組織変更を伴わないプロダクトにおける元測定値と前後差分値を以下のように定義した。

1）比較調査を行う開発組織変更を伴わないプロダクトと開発組織変更を伴うプロダクトは、同様の開発工程を経ている。

2）開発組織変更を伴うプロダクトが、開発工程 A から開発工程 B に移る際に開発組織変更を伴う場合、元測定値は開発工程 A の終了時点でのプロダクトから測定されたメトリクスの値となる。

3）開発組織を伴わないプロダクトの開発工程は同様に開発工程 A⇒開発工程 B である。そのため、こちらのプロダクトにおける元測定値も開発組織変更を伴うプロダクトと同様のタイミングである開発工程 A 終了時点でのメトリクスの値とする。

このように定義することで、規模や開発工程がほぼ同じプロダクトにおける開発組織変更の有無における影響を調査する。

前後差分値を測定するには、元測定値の測定時点と開発最終時点において「同一の関数」が存在している必要がある。元測定値測定時点では存在していた関数が、最終開発時点では削除されていた場合、変化量を測定することはできない。同様に、元測定値測定時点では存在していなかった関数が開発最終時点までに新たに作られた場合も変化量は測定できない。そのため本研究で調査対象とするのは、元測定値測定時点と開発最終時点において同一関数が存在しているものとする。

3.4 関連性の調査方法

まず元測定値と前後差分値の関係を調査するため、3 次元グラフへのプロットを行う。X 軸に元測定値、Y 軸に前後差分値をプロットする。Z 軸には、{元測定値, 前後差分値}の組み合わせを満たす関数がいくつあるかをプロットする。3 次元グラフは 1 プロタクトごとにそれぞれ作成し、各プロダクトの開発組織変更の有無を考慮して傾向比較を行う。

また、傾向分析の結果として差があるように見られたメトリクスについては 2 要因による分散分析を用いて定量的に評価する。2 要因による分散分析を行うため、関数を 2 要因において分類する。一つ目の要因は開発組織変更の有無である。開発組織変更が行われたソフトウェアに含まれる関数ならば True とし、そうでなければ False とした。二つ目の要因は、元測定値が品質基準を満たしているかどうかである。この場合における品質基準は、メトリクスごとに異なるため対応する閾値を適宜決定する。品質基準となる閾値を決定した後に、閾値を満たしていれば True とし、満たしていなければ False に分類した。この 2 要因が前後差分値に影響を与えているかを調査することが目的であるため、前後差分値は数値データをそのまま扱った。

4. 事例

4.1 本研究において用いるプロダクト

本研究において調査対象は、企業間の外部委託によって開発されたソフトウェアである。開発工程の概要を図 1 に示す。開発企業 X は、プロダクト A、B を外部企業 Y、プロダクト C、D を外部企業 Z に委託する形で開発を行った。初めの外部委託開発は図 1 の第一開発工程の期間に行われた。第一開発工程は、初期リリースまでにおける開発期間と、その後に発生した不具合を修正する修正期間の二つに分けられる。初期リリースから不具合修正期間を経て、開発が完了した時点で企業 X は 4 つのプロダクトの納品を企業 Y と企業 Z から受け入れた。その後、それぞれのプロダクトに機能追加を行うため第二開発工程を外部に委託することになった。この第二開発工程の際には、企業 X は 4 つのプロダクトを全て外部企業 Y に委託する形で開発を行った。第二開発工程においても、第一開発工程を同じく開発期間と修正期間に分けら

表 1　事例にて用いるプロダクトの詳細

	第一工程LoC	第二工程LoC	第一工程関数の数	第二工程関数の数	共通名関数の数
ProductA	1437	2266	78	104	58
ProductB	1152	3378	62	483	52
ProductC	1766	4289	73	186	59
ProductD	322	606	7	15	6

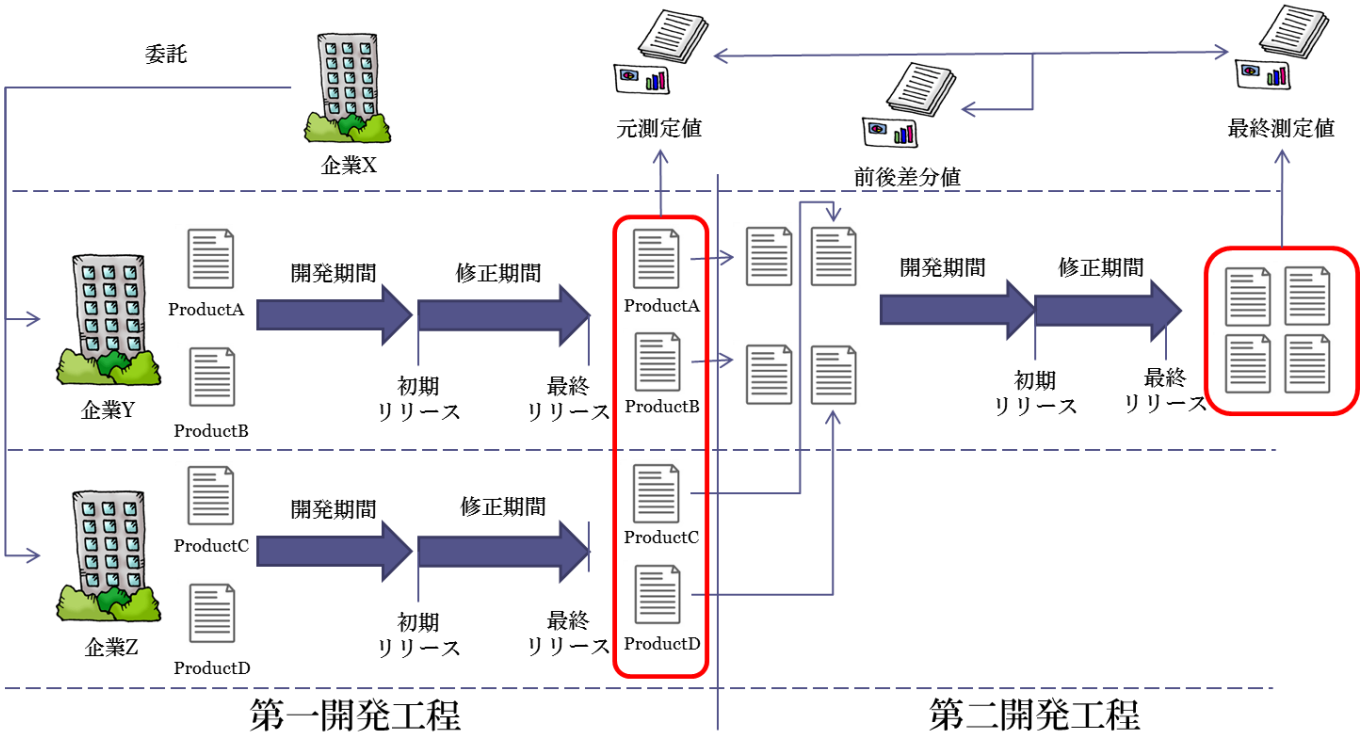

図 1　事例にて用いるプロダクト群の開発工程概要

れる。以上の開発工程において、組織変更を伴うプロダクトをそうでないプロダクトの 2 郡に分けられることに我々は着目した。4 つのプロダクトを以下の 2 郡に分けて比較調査することにより、組織変更の有無における影響を明らかにする。

・組織変更有（企業 C⇒企業 B）：プロダクト C, D
・組織変更無（企業 B のみ）：プロダクト A, B

4.2 プロダクトの詳細

本事例で用いたプロダクトは、1 つの大きなソフトウェアに内在する 4 つのプロダクトである。4 つのプロダクトはそれぞれ独立した機能を持っている。また、4 つのプロダクトはすべて C#で開発されている。表 1 に 4 つのプロダクトの規模を示す。LoC は Lines of Code を省略した形であり、総コード行数を意味する。それぞれの群における総コード行数を比較すると、この表に記載されている値は、修正期間を終えて納品がされたプロダクト、すなわちそれぞれの開発工程の最終時点での値。この表を見ると。第一開発工程においては ProductA, B の合計 LoC が 2589 行、それに対して ProductC, D の合計は 2088 行である。第二開発工程においては、それぞれ大きく値が増えて ProductA,B の合計が 5644 行、ProductC,D の合計が 4895 行となっている。これらの群は規模が全く同じといえるほどの値ではないが、大きく異なるわけではない。そのため、比較調査をするにあたって大きな影響は出ないと判断し、これらのプロダクトを用いて比較調査を行った。

2 つの工程における差分を測定するため、両方の工程に存在する関数やファイルからしか前後差分値は測定できないことを提案の項では述べた。そのような関数がいくつあるのかを明確にするため、表 1 に第一開発工程時点での関数の数、第二開発工程時点での関数の数、両方の工程において存在する同名関数の数を示した。この表を見ると、ProductA と ProductD においては第二開発工程において追加された関数は少ないことがわかる。一方で、ProductB と ProductC では第二開発工程において非常に多くの関数が追加されている。しかし、共通名関数の数と第一開発工程時点での関数の数を比較すると大きな差はない。これは、第二開発工程においてもほとんどの関数は削除やリネームをされることなく残っていることを示す。そのため、第一開発工程において記述されたコードが、開発組織変更の有無によって受ける影響を調

査することは可能であると考え、本プロダクト群を用いた。

4.3 不具合修正回数の測定方法

本研究によって得られた傾向が、不具合の修正回数と関連しているかを調べるため、外部開発組織との報告で用いていた文書（以下文書 A）を用いて修正回数の測定を行う。この文書に記述されているのは、図 1 における修正期間に修正された不具合の履歴である。そのため、開発期間における不具合修正回数は測定することができない。文書 A には、各リリースにおいてどのファイルを修正したかが記述されている。本研究では、各ファイルが何度のリリースにおいて修正されたかを測定することで「不具合修正回数」とした。

5. 評価

本稿では、前章で述べた統計解析結果を述べつつ、我々がたてた RQ の真偽について議論する。

・RQ1　元測定値と前後差分値の関係は、開発組織変更の有無によって違いがあるか？

「はじめに」の項において、「元測定値」が品質基準を満たしているならば、開発組織変更による技術者の暗黙の知識やギャップは発生しづらいと仮説を立てた。この仮説が正しいならば、開発組織変更を伴う開発でも、品質基準を満たすよう修正を施すことで、品質低下のリスクを低くすることができる。この仮説が正しいかを確かめるために、我々は測定した各メトリクスの元測定値と前後差分値の関係と開発組織変更の有無による違いを調査した。

図 2 は、測定したメトリクスの中でも特に顕著な傾向を示した Cyclomatic 複雑度における元測定値と前後差分値の関係である。本グラフでは X 軸が元測定値、Y 軸が前後差分値、Z 軸は元測定値と前後差分値の関係が同値の関数数を示している。ProductA と ProductB は、開発組織変更を伴わなかったプロダクトであり、ProductC と ProductD は開発組織変更を伴って開発されたプロダクトである。これら 4 つのグラフを開発組織変更の有無によって比較すると、ProductA と ProductC において前後差分値が大きな値を示す箇所の傾向

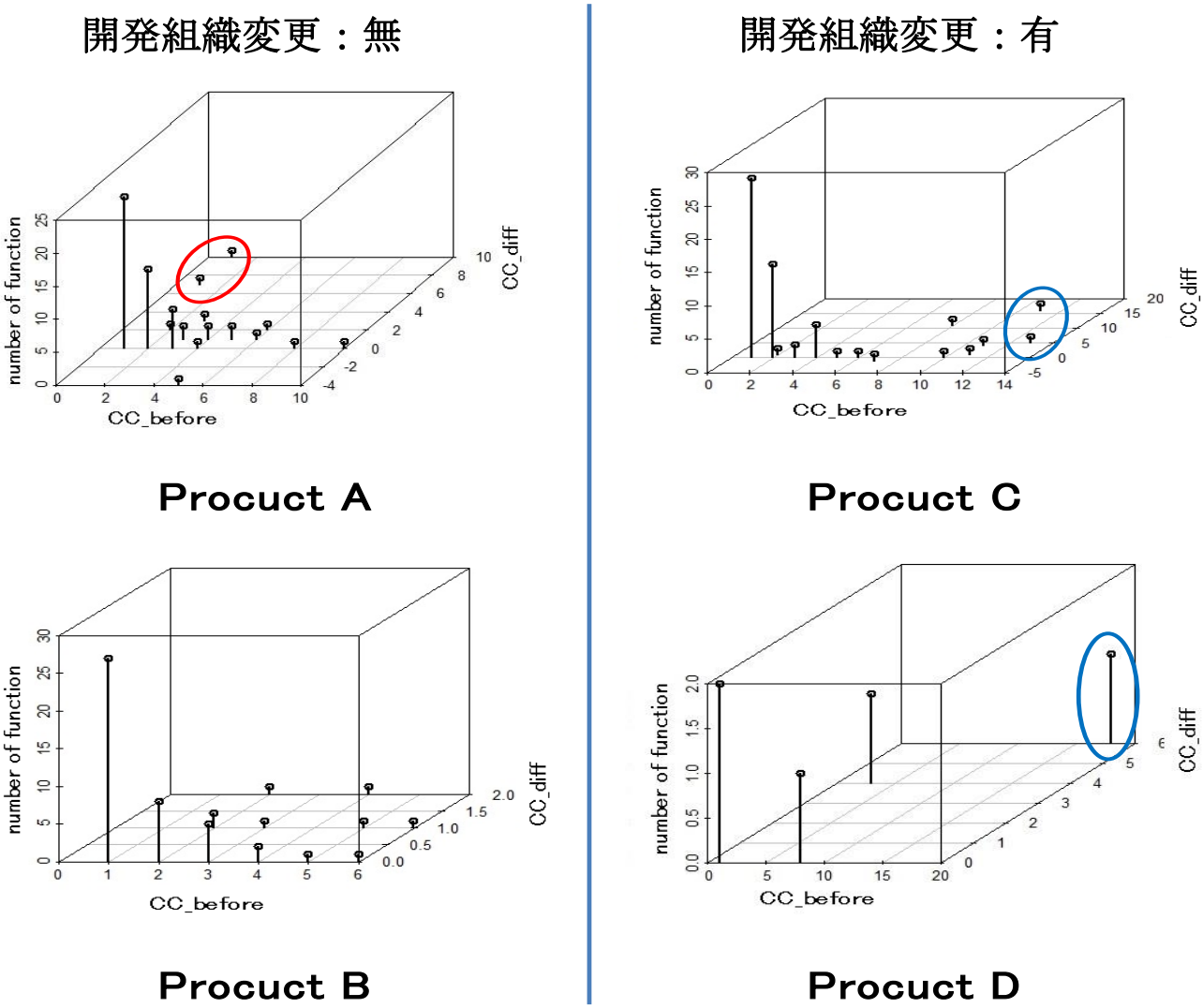

図 2 Cyclomatic 複雑度における 3Dplot 結果

が異なっていることがわかる。それぞれのグラフにおける丸で囲った部分を見ると、開発組織変更を伴った ProductA と ProductB は元測定値の大きな関数において前後差分値が大きな値を示しているのに対し、開発組織変更のなかった ProductC では元測定値の小さな関数において前後差分値が大きな値を示している。開発組織変更の無かったもう一つのプロダクトである ProductD は、そもそも元測定値が大きな値を示すものがなく、前後差分値も大きな値を示す関数が存在していない。

以上の結果から、元測定値と前後差分値の関係は開発組織変更の有無によって異なることがわかる。異なることを統計的に示すため、開発組織変更の有無と元測定値が品質基準を満たしているかどうかの２要因を用いて２要因による分散分析を行った。今回の分析は元測定値の Cyclomatic 複雑度が品質基準を満たしているかどうかの閾値を定める必要がある。この閾値は Cyclomatic 複雑度の望ましい値について電球している[13]を参考にして 10 とした。分散分析の結果を表１に示す。

表１では、一番左の列にあるそれぞれの要素と前後差分値の関係をそれぞれの行で示している。注目するべきは Pr の列である。Pr はそれぞれの要素が前後差分値と関連がないといえる割合を示す。まず「開発組織変更の有無」を示す Ent-changes の要素を見ると前後差分値の関連がないといえる割合は約 11.9%であることがわかる。この値は統計的に関連があるといえる数値ではない。次に「元測定値が品質基準を満たしているかどうか」を 10 という閾値で分類した要素であることを示す CCFactor10 について見ると、5.62×10^-10%であり関連がないといえる割合は非常に低いことが確認できた。最後に、この２つの要素を両方考慮したときの関連性は 0.114%でないといえる。これらの結果から、我々が提案した「元測定値」と「前後差分値」の関連があること、この関連が「開発組織変更の有無」によって異なる結果を導くという結果を統計的に示すことができた。

・RQ2　元測定値、前後差分値と不具合修正回数の間にはどのような関係があるか？

・RQ3　元測定値、前後差分値と不具合修正回数の間にある関係は、開発開発組織変更の有無によって違いがあるか？

次に、RQ１の調査結果と修正回数にどのような関係があるかを評価するため RQ2 と RQ3 に対して評価を行う。RQ1 への回答として、開発組織変更を伴う場合は元々複雑な関数がより複雑になりやすい傾向が存在し、開発組織変更を伴わない場合はこのような傾向はなく品質低下のリスクが低いという結果が得られた。ここでは、このような傾向を示した関数は何度の修正を経て前後差分値が増加する結果となったのかを調査することで、修正を引き起こす原因と修正が引き起こす前後差分値の変化を明確にする。

で述べたように、修正回数はファイル単位でしか測定できないため、前後差分値が大きな値を示した関数がどのファイルに記述されているかを考慮しつつ調査を行った。調査結果を以下の表２に示す。

この表における Product は、各 File と method がどのプロダクトに属しているかを示す。表に示した method はすべて”Cyclomatic 複雑度”の前後変化量が５以上のものであり、前後変化量と元測定値の列はそれぞれの method における”Cyclomatic 複雑度”の値を示している。「Ent-change」の列は開発組織変更の有無を示しており TRUE ならば開発組織変更有のプロダクト、FALSE ならば開発組織変更無のプロダクトに属していることを示す。「modifications」の列は各ファイルの不具合修正回数を示す。これは測定の都合上 method 単位に分けることができないため、ファイル単位で示してい

表 3　2 要因における分散分析の結果

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Org-change	1	5.6	5.56	2.444	0.11939
CCFactor 10	1	120.8	120.8	53.124	5.62E-12
Org-change : CCFactor 10	1	24.7	24.73	10.877	1.14E-03
Residuals	219	498	2.27		

表 2　前後差分値の大きい関数の不具合修正回数

product	File	Method	CC_before	CC_diff	Org-change	Modification	Ranking
A	1	a	1	10	FALSE	4	2nd
	2	b	1	7	FALSE	8	1st
C	1	a	11	16	TRUE	11	1st
		b	13	5	TRUE		
	2	c	8	11	TRUE	6	3rd
D	1	a	18	6	TRUE	5	1st

る。「Ranking」の列は、各プロダクトにおいて修正回数が多い順にファイルを並べた時、上位何番目のファイルであるかを示している。これらの結果を見ると、前後差分値が大きな関数は軒並み修正回数の多いファイルに含まれていることがわかる。

以上のことを考慮して考察すると、まず「元測定値の大きさは修正回数と関係がない」といえる。なぜならば、「前後差分値の大きな関数における元測定値」は RQ1 の結果から「開発組織変更を伴う場合は元測定値が大きく、伴わない場合は元測定値が小さい」のにもかかわらず、前後差分値が大きな関数は軒並み修正回数の多いファイルに含まれているためである。次にいえるのは、「修正回数の多いファイルは、複雑な関数を含んでいる可能性が高い」ということである。なぜならば、前後差分値が大きな関数は軒並み修正回数の多いファイルに含まれていることが調査結果から確認できるためである。修正回数を数えるタイミングは「Second development」の初版リリース時点からであるため、「Second development」の初版リリース時点でのメトリクス値を測定することでより詳細な分析ができるかもしれない。

これらのことから、RQ2 と RQ3 に対しては以下のように結論付けることができる・

・RQ2　元測定値と修正回数には関係がなく、修正回数の多いファイルには前後差分値の大きい関数が含まれている可能性が高い。

・RQ3　開発組織変更の有無に関わらず、前後差分値の大きな関数を含むファイルにおいてコードの修正が何度も行われていた。

6. おわりに

本研究では、元測定値と前後差分値という２つの値に着目し、産業プロダクトを用いて統計分析を実施することで開発組織変更の有無による２つの値の関連性を調査した。その結果、開発組織変更を伴う開発では Cyclomatic 複雑度の元測定値が大きな値である関数において前後差分値が大きな値を示した。一方、開発組織変更を伴わない開発では Cyclomatic 複雑度の前後差分値が大きな値を示したのは元測定値が小さい関数であった。これに加えてコードの修正回数という観点からも関連を調査した結果、開発組織変更の有無に関わらず前後差分値が大きな値を示した関数は多くのコード修正がなされていた。

これらの結果を踏まえると、現場で開発を行う技術者は開発組織変更を伴う開発において、開発組織変更を行う直前のプロダクト内で複雑度の大きな関数が開発組織変更後により大きな複雑度になる恐れがあるという考えのもとで開発を行うことができる。そのため、開発組織変更を行う直前のプロダクトにおけるメトリクス値を測定して、複雑度の大きな関数があったならばリファクタリングなどを行うことでより複雑になるリスクを防ぐことができる。

本研究で統計分析を行ったプロダクトは４つと非常に少ない。そのため、このような傾向が起きたのは本プロダクト特有のものである可能性があり、開発組織変更を伴うプロダクトすべてに適応できるとは現状決めつけることは難しい。

将来研究として、より多くのサンプルプロダクトを用いて同様の傾向が得られるかどうかを調査することが必要である。また、詳細なプロダクトメトリクス値の変化や、関数単位以外のプロダクトメトリクスに着目することで新たな傾向がないかを調査する予定である。

REFERENCES

[1] M. Conway, “How do committees invent?” *Datamation*, vol. 14, no. 4, pp. 28–31, 1968.
[2] J. O. Coplien, “A generative development-process pattern language,” in *Pattern languages of program design*, J. O. Coplien and D. C. Schmidt, Eds. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 183–237.
[3] A. Mockus, “Organizational volatility and its effects on software defects,” in *FSE ’10 Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, 2010, pp. 117–126.
[4] L. C. Briand, J. W¨ust, J. W. Daly, and D. V. Porter, “Exploring the relationship between design measures and software quality in objectoriented systems,” *J. Syst. Softw.*, vol. 51, no. 3, pp. 245–273, May 2000.
[5] T. Gyimothy, R. Ferenc, and I. Siket, “Empirical validation of objectoriented metrics on open source software for fault prediction,” *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 897–910, Oct. 2005.
[6] N. Nagappan, T. Ball, and A. Zeller, “Mining metrics to predict component failures,” in *Proceedings of the 28th international conference on Software engineering*, ser. ICSE ’06. ACM, 2006, pp. 452–461.
[7] Y. Shin, R. Bell, T. Ostrand, and E. Weyuker, “Does calling structure information improve the accuracy of fault prediction?” in *Mining Software Repositories, 2009. MSR ’09. 6th IEEE International Working Conference on*, May, pp. 61–70.
[8] R. Subramanyam and M. Krishnan, “Empirical analysis of ck metrics for object-oriented design complexity: implications for software defects,” *Software Engineering, IEEE Transactions on*, vol. 29, no. 4, pp. 297–310, April.
[9] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus, “Does code decay? assessing the evidence from change management data,” *IEEE Trans. Softw. Eng.*, vol. 27, no. 1, pp. 1–12, Jan. 2001.
[10] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy, “Predicting fault incidence using software change history,” *IEEE Transactions on Software Engineering*, vol. 26, no. 7, pp. 653–661, July 2000.
[11] A. Hassan and R. Holt, “The top ten list: dynamic fault prediction,” in *Software Maintenance, 2005. ICSM’05. Proceedings of the 21st IEEE International Conference on*, Sept. 2005, pp. 263–272.
[12] Seiji Sato, Hironori Washizaki, and Yoshiaki Fukazawa “Effects of Organizational Changes on Product Metrics and Defects
[13] H. Washizaki, R. Namiki, T. Fukuoka, Y. Harada, and H. Watanabe, “A framework for measuring and evaluating program source code quality,” in *Proceedings of the 8th international conference on Product-Focused Software Process Improvement*, ser. PROFES ’07. Springer-Verlag, 2007, pp. 284–299