Vol.2017-SE-196 No.29
2017/7/21

Git 開発履歴情報に基づく不確かさの可視化

村岡 北斗[1,a)]　村本 大起[1,b)]　鵜林 尚靖[1,c)]　亀井 靖高[1,d)]　佐藤 亮介[1,e)]

概要：ソフトウェア工学において「不確かさ」をマネジメントした開発を行うことは重要である．そのためには，実際の不確かさについて明らかにする必要がある．本論文では，Git で管理しているリポジトリを対象に，不確かさがどのようなものであるか，不確かさはいつどのようなプロセスで発生するのか，不確かさにはどのような内容のものがあるのかといった，不確かさに関する様々な項目について，更新履歴情報を元に分析を行い，可視化するツールを提案する．

キーワード：不確かさ，ソフトウェア開発，Git

1. はじめに

ソフトウェア開発において，「不確かさ」とは避けられないものである [6]．ソフトウェア開発における不確かさとは，要求，設計，実装方法やテストなど様々な開発工程に現れる不確実性についての問題である．このような不確かさは，開発者にとって扱いにくいものであり，一時的な措置をとることも多く，バグやソースコードの煩雑化の原因となりがちである．近年，ソフトウェア工学において，不確かさを包容したソフトウェア開発は重要な研究課題の 1 つとされている [3–5, 8, 9, 12, 13]．

ソフトウェア開発における不確かさは大きく 3 つに分類される（*Known Knowns, Known Unknowns, Unknown Unknowns*）[5]．*Known Knowns* は，不確かさが存在しない開発である．*Known Unknowns* は，ソフトウェア開発のプロセスの中で不確かな問題が存在する開発である．不確かな問題が開発者によって認識されている状態である．*Unknown Unknowns* は，何が不確かさかを理解できていない開発を指している．従来，この *Known Unknowns* である不確かさの問題は，開発者にリスク管理の 1 つとして扱われ，Issue Tracking System やソースコード，仕様書にメモなどとして記載されるような対処が行われてきた．しかし，このような管理方法では，管理されている不確かさがソースコードやモデルのどの部分に対応しているかわからなくなることが多く，また自然言語で記述されていると読む人によって不確かさの認識が変わってしまう恐れもある．

このような問題から，不確かさをうまく管理した状態で開発を行うことは重要であると言える．不確かさを管理していく上で，実際にソフトウェア開発で発生している不確かさについて明らかにすることが必要である．しかし，実際の不確かさについて調査した論文は少ない．そこで本論文では，不確かさについて明らかにすることを目的に，既存のプロジェクトを対象に，不確かさに関する様々な情報を分析し可視化するツールを提案する．

今回提案するツールでは，*Known Unknowns* に該当する不確かさだけを扱う．*Unknown Unknowns* タイプの不確かさは，メタ的な不確かさで Git の履歴だけでは分析を行うことが困難である．また，*Known Unknowns* を特定する手段として，本稿では，ソフトウェア更新時のコミットコメントに不確かさに関する語句を含むものを不確かさと近似して扱う．

以降，第 2 章では，今回扱う不確かさが既存の不確かさの分類方法ではどのような不確かさに対応するか簡潔に述べたのち，本研究のアプローチを説明する．第 3 章では本ツールについてどのような機能が存在するか説明する．第 4 章では，ツールの実装方法について述べ，実装上の問題点を明らかにする．第 5 章で今後の課題について説明する．最後に，第 6 章で本研究のまとめを述べる．

2. 不確かさについて

本章では，不確かさの分類についての既存研究を説明し，本研究での対象がどのようなものであるかを明らかと

[1] 九州大学
Kyushu University
[a)] muraoka@posl.ait.kyushu-u.ac.jp
[b)] muramoto@posl.ait.kyushu-u.ac.jp
[c)] ubayashi@posl.ait.kyushu-u.ac.jp
[d)] kamei@posl.ait.kyushu-u.ac.jp
[e)] sato@posl.ait.kyushu-u.ac.jp

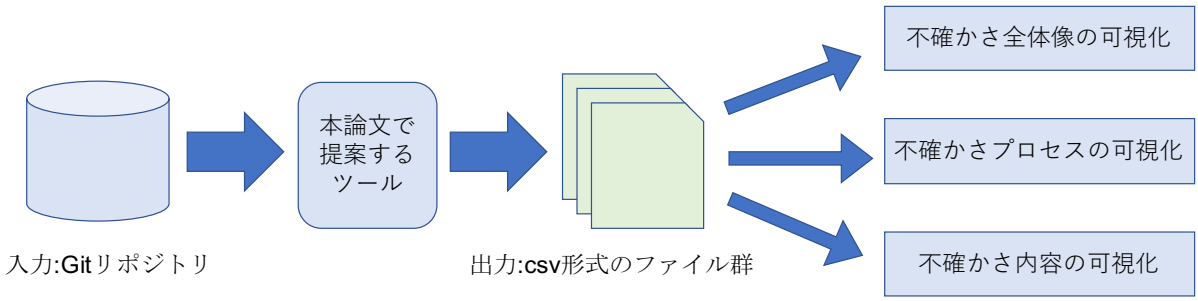

図 1　ツール適用範囲説明図

表 1　Perez-Palacin らによる不確かさの分類

観点	性質	性質の説明
場所	コンテキスト	環境に関する不確かさ
	モデル構造	モデル自体の構造に現れる不確かさ
	入力パラメータ	モデルへの入力に関する不確かさ
レベル	レベル 0	確定している知識
	レベル 1	知識の不足を認知している状態. 既知の不確かさ.
	レベル 2	知識の不足を認知できていない状態. 未知の不確かさ.
	レベル 3	不確かさを認知するプロセス自体が 不足している状態.
	レベル 4	メタな不確かさ. 不確かさのレベル自体が不確か.
性質	認識的	十分なデータや知識が無いために 発生する不確かさ
	偶発的	物理現象等の確率的不確かさ

する．その後，不確かさの研究に対する問題点と，それを解決することを目的とした我々の研究手法を述べる．

2.1　関連研究

Perez-Palacin らの研究 [10] では，不確かさを，場所，レベル，性質の 3 つの観点から分類している（表 1）．不確かさの「場所」とは，不確かさがどこに現れたのかを指している．不確かさの「レベル」とは，どの程度知識が確定したものであるのかを段階的に分類したものである．また，不確かさの「性質」とは，不確かさの発生した状況についての分類である．本研究では，表 1 にある分類のうち，以下の不確かさを扱う．

- 場所 — コンテキスト，モデル構造，入力パラメータ
- レベル — レベル 1
- 性質 — 認識的

本研究での対象となる不確かさは，コミットコメントを用いて認識されている不確かさ（*Known Unknowns*）を検出し分析していくものである．そのため，今回扱う不確かさについて，レベルと性質は上記のような分類になる．場所については，指定していないため全てを対象とする．

2.2　提案するツールのアプローチ

不確かさを扱っている研究は存在するものの，不確かさに関する実証研究は少ない．不確かさをうまく管理した開発を提案するには，実際の不確かさについて理解を深めることが必要である．そこで，本論文では，不確かさの問題を可視化することを目的として，既存の Git で管理されているリポジトリを対象に，不確かさについて分析を行うツールを提案する．

3.　ツールの概要

不確かさの問題について，適切な管理を行いつつ開発を行うことは重要である．不確かさを適切に扱いつつ開発をするためのマネジメント手法を提案したい．そのためには，まず，実際の不確かさがいつ，どこで，どのように発生しているか理解することが必要になる．そこで，我々は不確かさが持つ様々な要素に対して分析を行うツールを提案する．

本ツールは，大きく分類して 3 種類の分析を行う．図 1 は，ツールの入出力と各分析内容の範囲を表しているものである．

1）不確かさ全体像の可視化

まず，初めに不確かさを理解するにあたって，そもそも不確かさがどのようなものであるか理解する必要がある．本ツールでは，不確かさがプロジェクトにどの程度存在しているか，また，コミッタごとの不確かさ関与率を分析し，不確かさをプロジェクト単位で扱うことでアプローチする．また，不確かさがいかに問題であるかを調査するために，不確かさとバグの関連性についても分析を行う．

2）不確かさプロセスの可視化

次に，不確かさはどのようにして発生しているのか，不確かさに至るまでのプロセスについて分析することで，不確かさのマネジメントの足がかりになるのではないかと考える．そこで，発生過程の更新内容や更新間隔，更新回数，不確かさに関わったコミッタなどの要素を分析することで不確かさに関するプロセスの傾向を分析し提示する．

3）不確かさ内容の可視化

最後に，不確かさにはどのような種類のものが存在するかを知ることによって，不確かさを分類し，各内容に対す

表 2　ツールの各機能

カテゴリ	機能	使用する情報		
		コミットコメント	コードの履歴情報	コミッタ
全体像の可視化	全コミット数に対する不確かさの数	○		
	不確かさコミットのバグの割合	○		
	コミッタごとの不確かさ関与率	○		○
プロセスの可視化	コード片の更新回数		○	
	コード片に関わったコミッタ数		○	○
	コード片に含まれる関数		○	
内容の可視化	不確かさに関する特徴単語の抽出	○		

表 3　データセット GIMP

プロジェクト名	期間	総コミット数	総コミッタ数
GIMP	1997/6/1 - 2015/6/9	35,226	375

るマネジメントを提案するために，キーワードやコミットの内容について分析を行う．

今回扱う不確かさは，Git リポジトリの持つコミットに関する情報から，コミット時のコメント内容に不確かさに関するキーワードが含まれているかを基準に特定した．本論文で提案するツールの各機能を表 2 に示す．

3.1　不確かさ全体像の可視化

不確かさの全体像を可視化する上で，本ツールで分析した項目は以下の 3 つである．

- 全コミット数に対する不確かさの数の割合
- 不確かさコミットに含まれるバグの割合
- コミッタ毎の不確かさ関与率

各項目について詳しく説明していく．

3.1.1　全コミット数に対する不確かさの数の割合

この分析は，不確かさとコミット数の間にどのような関係が見られるのかを調査することを目的としている．

出力の csv に存在する項目はプロジェクトの全コミット数と含まれていた不確かさの数である．この項目から，全体に対する不確かさの割合を算出することができる．

全コミットの数と不確かさの数についての分析を応用し，多くのプロジェクトを対象として使用することで，分布図として出力することが可能である．応用例を図 2 に示す．横軸が，プロジェクトの全コミット数，縦軸がプロジェクトに存在していた不確かさの数である．データセットは，GitHub で公開されている 22,171 件のプロジェクトである．

この出力を利用することで，一般的に不確かさがどの程度の割合で含まれているか見ることができる．また，コミット数の数が多いものと少ないもので割合の傾向が異なるかを比べることも可能である．他にも，対象とするプロジェクト群をソフトウェアの種類によって分別することで，ソフトウェアの種類によって不確かさの出現頻度に差が現れるのかについても可視化することができる．

3.1.2　不確かさコミットに含まれるバグ修正コミットの検出

この項目は，不確かさがどの程度重要視されるべき問題であるのかを明らかにする目的があり，バグとの関連性を調査することで不確かさの問題の重要性を示す．ここでは，バグを持っているコミットかどうかをバグ修正に関連するキーワードを用いて特定している．

出力は，プロジェクト全コミット数に対するバグを含んだコミット数とその割合，プロジェクト内の不確かさを持つコミット群に対するバグを含んだコミット数とその割合である．

次に，出力の例を示す．ここで使用したデータセットは，GitHub で公開されている OSS の GIMP [1] プロジェクトである．データセットの概要を表 3 に示す．さらに，ツールを適用した結果を表 4 に示す．この出力例では，単一のプロジェクトでの傾向を表している結果でしかないが，適用するプロジェクト数を増やすことで，不確かさが持つ一般的な性質について言及できる結果が得られる．

3.1.3　コミッタごとの不確かさ関与率

この項目を調査することで，コミッタによって不確かさの出現頻度に差があるのかを明らかにする目的がある．

出力は，プロジェクトの全コミットに対するコミッタ別のコミット回数，とその割合，また不確かさを持つコミットに限った場合のコミット回数とその割合である．

出力例として，先ほどと同じ GIMP プロジェクトに対してツールを適用した結果の一部を表 5 に示す．プロジェクト全体でのコミット数上位 7 人分の分析結果である．

このデータを利用することで，コミット割合の高さに応じて不確かさが生まれやすいのかどうかを調査することができる．また，プロジェクト内の役職について調べこのデータと比較することで，コミッタの立場と不確かさ関与の割合などについても調べることができる．

例えば，今回の出力例を用いると，このプロジェクトでコミット数上位 2 名は開発者の中でも特別な役職であり，全体の半数以上のコミットをこの 2 名で行っている．

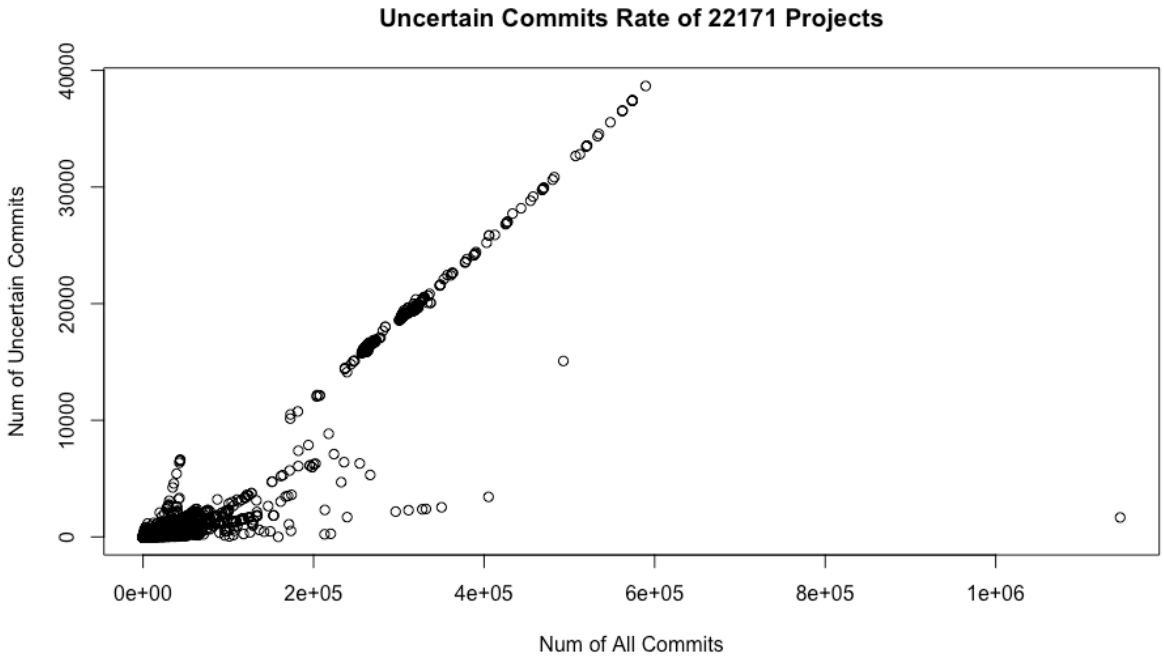

図 2 コミット数と不確かさの数の分布図

表 4 GIMP の不確かさとバグのコミット数

不確かさのキーワード	全コミット数	バグ修正コミット数	バグ修正コミットが占める割合 [%]
あり	108	30	28%
なし	35,118	4,660	13%

表 5 GIMP プロジェクトにおけるコミッタごとの不確かさ関与率 (上位 7 人)

コミッタ	コミット数	プロジェクト全体のコミット率 [%]	不確かさコミット内でのコミット率 [%]
Michael Natterer	11,869	31.62	67.86
Sven Neumann	9,341	24.89	8.19
Martin Nordholts	1,411	3.75	0.25
Manish Singh	1,276	3.4	3.27
Mukund Sivaraman	623	1.66	0.10
Simon Budig	486	1.29	0.40
Jehan	468	1.24	0.09
合計	37,713	100.0	100.0

しかし，不確かさを持つコミットに限定すると，Michael Natterer は不確かさの 6 割以上に関与しているのに対し，もう一人の Sven Neumann は 8.19%と不確かさへの関与率が低くなっていた．この項目では，このようにしてプロジェクト内での不確かさの傾向を分析することができる．

3.2 不確かさプロセスの可視化

この項目では，不確かさがどのようにして発生しているのかを調べるために，不確かさを持つコミットで更新されていたコードを，行の固まりごとにコード片として扱い，各コード片について，初めてコード片部分が記述されたコミットまで更新履歴を遡って分析する．

分析結果は，各コード片ごとに以下の項目について出力する．

- 不確かさを持つコミットのハッシュ ID
- 不確かさのコード片が過去に更新されたコミットのハッシュ ID
- コード片が含まれるファイル名
- コード片の更新回数
- コード片が初めて書かれてから不確かさが現れるまでの期間
- コード片に含まれている関数の名前
- コード片に含まれている関数の数

ハッシュ ID とファイル名以外の項目について，分析する目的と出力の使用例について詳しく述べていく．

3.2.1 コード片の更新回数

コード片の更新回数を調査する目的は，特定箇所を繰り返し更新することで発生しやすくなるのではないかという

表 6　データセット ead

プロジェクト名	期間	総コミット数	不確かさを持つコミット数
ead	2013/10/28 - 2015/10/10	4,611	12

表 7　各不確かさのコード片単位での分析機能出力例

不確かさコミットの ハッシュ ID	1 つ前の変更の ハッシュ ID	不確かさが存在する ファイル名	コード片の 更新回数	更新履歴の ハッシュ ID
7e5495a9...	20b00836...	VarsContext.java	2	20b00836... 684d0e1c...
7e5495a9...	4f4b96e4...	VarsContext.java	2	4f4b96e4... 59dacae4...
f37bcde3...	08fc4c0a...	menu.json	2	08fc4c0af... e020b2d6...
f37bcde3...	adec5ea6...	SceneElementActor.java	5	adec5ea6... c26d56b9... ...
f37bcde3...	db1bd8da...	SceneElementActor.java	6	db1bd8da... adec5ea6... ...
不確かさコミットの ハッシュ ID	1 つ前の変更の ハッシュ ID	不確かさの存在期間 [日]	関数更新回数	関数の変更
7e5495a9...	20b00836...	82.97266203703704	2	Variable(Object value) value.getClass()
7e5495a9...	4f4b96e4...	104.15609953703704	2	tvariables.put(v.getName() Variable(v.getInitialValue()
f37bcde3...	08fc4c0a...	8.873530092592592	0	
f37bcde3...	adec5ea6...	66.03050925925926	3	EAdEngine.factory.getElement(element.getRenderer() ...
f37bcde3...	db1bd8da...	66.03050925925926	3	EAdEngine.factory.getElement(element.getRenderer() ...

仮説を検証することである．この仮説は，我々が不確かさについて調査している段階で，「更新を重ねているうちに内容がわからなくなってしまった」という内容の不確かさに出会ったことから推測されたものである．出力は，不確かさに関わったコード片が不確かさとして更新される以前に何度更新を重ねたものであるかをカウントし回数を表示する．

3.2.2　コード片の履歴に関わったコミッタ数

複数人が関わることで内容がわからなくなってしまったという不確かさが発生することが多いのではないかという推測を検証することを主な目的とした項目である．出力は，不確かさに関わったコード片の更新履歴の中に何人のコミッタが含まれているかをカウントし表示する機能である．

例えば，この項目を利用することで，一人のコミッタが更新を繰り返し行ってきたコード片と複数人で更新を繰り返してきたコード片での不確かさの間に差が生まれるのか調べることができる．

3.2.3　コード片で変更された関数

この機能は，不確かさを持つコード片がどのような変更が行われたものであるかを判断するための材料として追加した機能である．現在，関数名しか抽出することができないが，将来的には関数の引数がどのように変更されてきたかなども抽出できるようにすることを考えている．

3.2.4　出力例

最後に，実際の OSS プロジェクトにツールを適用し，不確かさのプロセスを分析した場合の出力例を示す．この分析結果は csv 形式で出力される．ここでの例で用いる OSS プロジェクトは ead で，概要は表 6 に示す．この OSS，ead にツールを適用した出力例を表 7 に示す．各項目についての詳しい内容は，この csv ファイルを用いて集計し，表とすることで可視化することができる．

表 8　不確かさ内容の可視化機能 出力例

順位	may	might	probably	unknown	ambiguous
1	mayb	want	will	error	avoid
2	will	well	need	type	name
3	caus	sinc	sinc	reason	less
4	sinc	will	just	return	error
5	still	just	want	handl	also
6	differ	need	still	will	renam
7	want	caus	can	instead	case
8	need	also	way	messag	explicit
9	case	like	also	also	like
10	may	get	like	case	resolv

3.3　不確かさ内容の可視化

不確かさがどのような更新内容を持っているのかを可視化するために，不確かさを持つコミットを対象に，コミットコメントのテキスト分析を行う．今回提案するツールの機能では，更新時に更新内容について記述されているコミットコメントについて，特徴量を機械的に抽出し表示することで不確かさの内容を可視化する．

具体的な出力は，不確かさのコミットをキーワードごとに，コメントで特徴的とされる語句を特徴度が高い順に表示する．

不確かさ内容の可視化の機能を GIMP プロジェクトに適用した出力の一部を，出力例として表 8 に示す．出力の最上段にある英単語は，不確かさに関するキーワードである．各キーワードごとに，一緒にコミットコメントに出現していた特徴的な単語を，特徴度が高い順に表示している．この出力を見ることで，含まれていた特徴的な語句から各不確かさのキーワードを含んでいたコミットがどのような内容であるのか推測することができる．

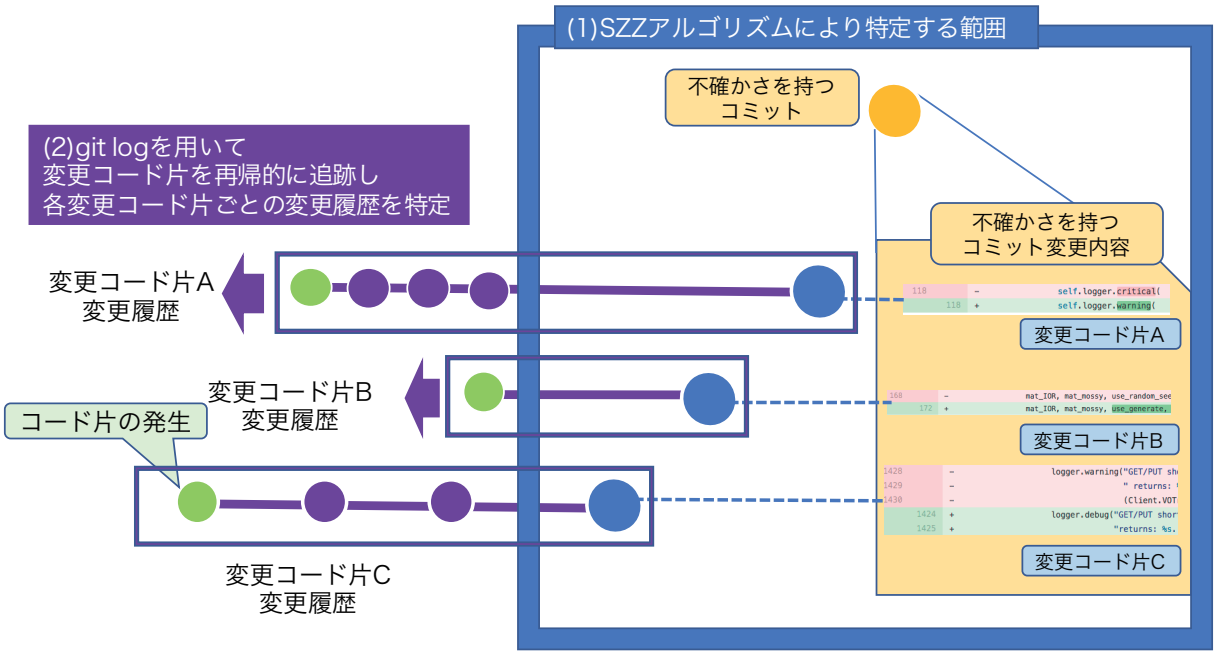

図 3　不確かさプロセスの可視化機能の実装概略図

表 9　不確かさ特定に用いたキーワード一覧

debatable, undetermined, unsure, unpredictable, unforeseeable, incalculable, risky, chancy, dicey, informal
iffy, vague, ambiguous, unknown, unascertainable, obscure, arcane, changeable, irregular, unreliable
unsettled, erratic, fluctuating, doubtful, dubious, undecided, irresolute, vacillating, unclear, ambivalent
hesitant, tentative, faltering, unconfident, may, might, probably, fuzzy

4. ツールの実装方法

この章では，ツールがどのような方法で不確かさを解析しているかについて述べる．

4.1 ツールの実装概要

我々が提案するツールは，現在，コマンドラインで実行するプログラムの形をとっている．ツールの概要を以下に示す．

- ツールの実装言語：シェルスクリプト，Perl，Python，R
- 入力：対象プロジェクトの.git ディレクトリの絶対パス
- 出力：不確かさを解析した結果（csv ファイル）

まず，動作のためにはツールが使用している言語の動作環境構築と，Git で管理されているプロジェクトを用意する必要がある．本ツールの出力は csv 形式のファイルであり，解析の段階において分けて複数出力される．実行時には，プロジェクトの.git ディレクトリが存在するパスを指定する必要がある．

4.2 不確かさの特定方法

最初に，本ツールで扱う不確かさを特定する方法を説明する．バージョン管理システムの履歴から不確かさに関連するキーワードが更新時コメントに存在しているコミットを，今回は不確かさが発見されたコミットと見なす．今回，不確かさの判定に用いたキーワード一覧を表 9 に示す．これらのキーワードは Oxford American Writer's Thesaurus から参照した uncertainty に関する類義語である．

4.3 不確かさの更新内容追跡手法

次に，不確かさのプロセスについて特定するまでの流れを図 3 に示す．この節は，ツールの機能のうち，refsec:process 節で説明した不確かさのプロセスを可視化する機能に対応している．

1）特定された不確かさを持つコミットに対して SZZ アルゴリズム [11] を適用することで，不確かさに関わったコード片の特定とそのコード片がどのコミットで記述されたものであるかを特定する．

2）発見された変更コード片を対象に，git log コマンドを用いて，過去にその変更コード片が何度更新されているかをコード片の発生まで追跡し，変更コード片の更新履歴とした．

1 つの不確かさを持つコミットについて，変更コード片が複数存在した場合は，各コード片ごとに独立して履歴を追跡した．

4.4 不確かさのコミットコメント分析手法

最後に，主に 3.3 節で説明した，不確かさの内容を可視化

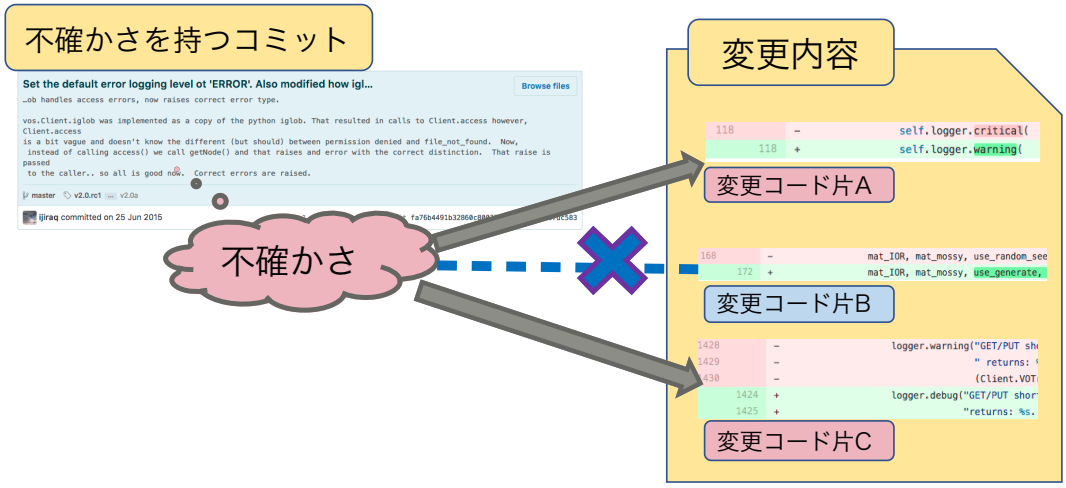

図 4　不確かさを持つコミットとコード片の対応

する機能に対応している部分を説明する．不確かさに関連するキーワードに対しての特徴量を分析するための，テキスト分析手法を述べる．この分析は，以下の 2 つのステップで行う．

ステップ 1：特徴値付き単語リストの作成

統計解析システム R の tm パッケージを利用して，すべてのコミットについてコミットメッセージを単語別に分解し，全コミットに存在する各単語の tf-idf の値を計算することで，全コミットでの特徴値付き単語リストを作成する．同様にして，不確かさに関連するコミットに限定した場合の特徴値付き単語リストを作成する．

ステップ 2：不確かさに関する特徴単語の抽出

ステップ 1 で作成した 2 つのリストに共通する単語について，不確かさのコミットでの特徴値から，全コミットでの特徴値を引くことで，各単語の不確かさにおいてどの程度特徴的かを算出する．数値が大きいものほど不確かさコミットの特徴を表した単語と見ることができる．

5. 今後の課題

5.1 実装上の課題

今回のツールでは不確かさに対して発見されたコミットから過去の更新へ遡って分析していくものだけだったが，今後，不確かさが発見された後にどのように更新が行われていったかも追跡して分析できるようにしたいと考えている．不確かさ後の様子を分析することで，不確かさはどのように対処されているのか，また，不確かさを持つ部分は将来的にバグになりやすいのかなどについても調査していきたい．

また，SZZ アルゴリズムで見つけたコード片には不確かさと関係のないものも混ざっている可能性がある．具体的には，SZZ アルゴリズムがコミットで行われた変更全てを原因のコード片としてしまうので，コミットの内容が不確かさだけでなく他の更新も同時に行われているコミットであれば，図 4 のように不確かさとは関係のないコミットも対象のコード片として扱ってしまう可能性がある．そこで，不確かさに関係するコード片だけを見つけ出すような仕組みも必要であると考えている．

最後に，不確かさを分析するにあたって，今回のツールで機械的に出力された結果だけでは分析するには不十分な部分が存在する．分析を行っていく上でさらに自動化できる部分を増やして実証分析をサポートしていきたいと考えている．

5.2 *iArch* との連動

最後に，将来的に *iArch* [2] と連動させることも考えている．[2] とは，不確かさを形式的に記述できるインターフェース機構 *Archface-U* を持つ統合開発環境である．*iArch* は，Git を用いて不確かさを記録し，不確かさに関する更新履歴を可視化することで不確かさの問題をマネジメントすることができる．現在，*iArch* はプロジェクトの初めから *iArch* を用いて管理されていたプロジェクトにしか対応していない．今回提案したツールを用いることで，*iArch* で管理していなかった既存のリポジトリも *iArch* で対応させるようにしたい．また，本ツールで可視化した様々な項目も提示することで，さらに不確かさのマネジメントをサポートすることができるのではないかと考えている．

6. まとめ

本論文では，Git で管理しているプロジェクトを対象とした不確かさの分析用ツールを提案した．また，不確かさの全体像の分析，不確かさのプロセスの分析，不確かさの内容に関する分析について，それぞれの目的と機能，実装方法について示した．また，今回提案したツールの使用例と出力例を示すことで，ツールを利用しての不確かさに関する様々な傾向を分析する方針を提案した．

今後の研究として，ツールの改良を重ねつつ，不確かさ

の分析を続けることで，不確かさをサポートする開発手法の提案の足がかりとするための知識を得ていきたい．

謝辞 本研究は，文部科学省科学研究補助費基盤研究（A）（課題番号 26240007）による助成を受けた．

参考文献

[1] GIMP. `https://www.gimp.org/`.
[2] iArch, `https://posl.github.io/iArch/`.
[3] Cheng, S.-W. and Garlan, D.: Handling uncertainty in autonomic systems, *International Workshop on Living with Uncertainties (IWLU'07)*, p. 2007 (2007).
[4] Devaraj, A., Mishra, K. and Trivedi, K. S.: Uncertainty propagation in analytic availability models, *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pp. 121–130 (2010).
[5] Elbaum, S. and Rosenblum, D. S.: Known Unknowns: Testing in the Presence of Uncertainty, *Proceedings of the 22nd International Symposium on Foundations of Software Engineering*, pp. 833–836 (2014).
[6] Garlan, D.: Software engineering in an uncertain world, *Proceedings of the FSE/SDP workshop on Future of software engineering research*, ACM, pp. 125–128 (2010).
[7] Magee, J. and Kramer, J.: *State Models and Java Programs* (1999).
[8] Meedeniya, I., Moser, I., Aleti, A. and Grunske, L.: Architecture-based reliability evaluation under uncertainty, *Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS*, pp. 85–94 (2011).
[9] Perez-Palacin, D. and Mirandola, R.: Uncertainties in the Modeling of Self-adaptive Systems: A Taxonomy and an Example of Availability Evaluation, *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, pp. 3–14 (2014).
[10] Perez-Palacin, D. and Mirandola, R.: Uncertainties in the Modeling of Self-adaptive Systems: A Taxonomy and an Example of Availability Evaluation, *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, pp. 3–14 (2014).
[11] Śliwerski, J., Zimmermann, T. and Zeller, A.: When do changes induce fixes?, *ACM sigsoft software engineering notes*, Vol. 30, No. 4, ACM, pp. 1–5 (2005).
[12] Sommerville, I.: Integrated requirements engineering: A tutorial, *Software, IEEE*, Vol. 22, No. 1, pp. 16–23 (2005).
[13] Ziv, H., Richardson, D. and Klösch, R.The uncertainty principle in software engineering, submitted to Proceedings of the 19th International Conference on Software Engineering (ICSE'97) (1997).
[14] 深町拓也，鵜林尚靖，細合晋太郎，亀井靖高：不確かさを包容する Java プログラミング環境，研究報告ソフトウェア工学 (SE)， Vol. 2015, No. 21, pp. 1–8 (2015).
[15] 深町拓也，鵜林尚靖，細合晋太郎，亀井靖高：Git 連携による不確かさマネジメントシステム，ソフトウェアエンジニアリングシンポジウム 2016 論文集， Vol. 2016, pp. 70–77 (2016).