変数名に対して開発支援を行うための調査と考察

松井亮介　蜂巣 吉成　吉田 敦　桑原 寛明

ソフトウェアにおいて識別子名は開発者がソースコードを理解するために重要な情報であり，名前のリファクタリングや命名支援に関する研究が行われている．先行研究では識別子名として関数名を扱うものが多く，変数名を使った研究は比較的少ない．ソースコードを理解するためには変数名も重要な情報である．本研究では，変数名に対する開発支援を行うために，オープンソースソフトウェアに対して変数の名前づけに関する規則の調査を行った．調査からユーザ定義の型から決まる変数名があること，変数名を区別する際にはナンバリングで区別している場合があるが，ナンバリングが適切でない例などが見つかった．ここで得られた知見を開発支援に応用する方法を考察する．

1　はじめに

ソフトウェアにおいて識別子名は開発者がソースコードを理解するために重要な情報である．識別子は，開発者が理解しやすい語彙で，値が表す内容を的確に表現する名前が望ましい．しかし，開発者は常に適切な識別子の命名を行えるとは限らず，ソースコードにはしばしば不適切な識別子名が含まれる．

本研究ではソフトウェア開発における識別子の命名を支援する．先行研究では識別子として関数名やクラス名を扱うものが多いが，変数名も重要な情報である．変数のうち大域変数の利用は避けるべきとされ，保守作業効率化のために変数のスコープを縮めるべきという主張[1] もある．それらを考慮し，本研究では識別子として変数に着目し，その中でも仮引数と局所変数の 2 つを扱う．なお，C 言語を対象言語とする．

本研究で目指す支援は，コーディング時における変数名命名支援，またはリファクタリング支援である．命名支援の一例として，変数宣言を含む関数の名前や変数の型の名前から変数名の候補となる語彙を提示する方法がある．リファクタリング支援では，型や関数に見られる命名傾向に変数名が合致するか判定し，修正が必要な変数の発見と修正候補を提示する方法がある．これらの支援を実現するためには，関数名や型名をはじめとした変数が出現する文脈と変数名の関係性を明らかにして規則として抽出する必要がある．

本稿では，変数名に対する開発支援に利用できる規則について，事例調査および考察を行う．命名という観点で変数に関する情報を調査し，開発支援に適用可能な規則となりうる関係性について議論する．

本稿の構成は次のとおりである．2 章で関連研究について述べる．3 章では事例調査を行うための予備調査を説明し，4 章でリサーチクエスチョン (以降 RQ) の設定を行う．5 章では事例調査の内容と結果の提示を行う．6 章では事例調査の結果からソフトウェア開発支援への適用可能性の議論と，調査のレビュー，妥当性に関する議論を行う．

Research and Discussions for Providing Support for Variable Names.

Ryosuke Matsui, 南山大学大学院理工学研究科ソフトウェア工学専攻, Software Engineering Major, Graduate School of Science and Technology, Nanzan University.

Yoshinari Hachisu, Atsushi Yoshida, 南山大学理工学部ソフトウェア工学科, Dept. of Software Engineering, Faculty of Science and Technology, Nanzan University.

Hiroaki Kuwabara, 南山大学理工学部電子情報工学科, Dept. of Electronics and Communication Technology, Faculty of Science and Technology, Nanzan University.

2 関連研究

柏原ら [2] は，メソッド名に用いる動詞の候補リストを提示する方法を提案している．メソッド名を動詞と目的語の組とみなし，相関ルールマイニングという手法を用いて，識別子やその型などメソッド本体とメソッド名に使われる動詞の関係を相関ルールとして抽出，適用することで命名の際に動詞の候補を提示する．

福田ら [3] は，識別子名の中でクラス名に着目し，既に内容が存在するクラス名の命名時，システムがふさわしいと考える名詞句または修飾語句といったフレーズを開発者に推薦することで，対話的にクラス名の命名を支援している．

Abebe ら [4] は，識別子名の提案を自動化する手法を提案している．識別子に含まれる自然言語情報を利用して，既存のコードから名詞や名詞句などの概念と，動詞など概念に対する依存関係を抽出し，それを利用して入力途中の識別子名の補完や置換を行う．term prefix，neighboring concepts，synonym という 3 つの語句候補から，文脈との関係に応じて優先順位付けを行い識別子を提案する．

識別子に関する支援はクラス名や関数名を中心にさまざまな方法が提案されているが，変数名に関する手段は少ない．

3 準備

3.1 予備調査について

変数名に対して事例調査を実施するにあたり，RQ を設定するための簡易的な調査を行った．以降これを予備調査と呼ぶ．オープンソース別に構文解析を行い，関数名と各関数に属する変数の型と名前を一覧化して目視で確認した．ここでは構文解析器 TEBA [5] を用いた．

今回対象としたオープンソースソフトウェア (以降 OSS) は表 1 の 3 件である．これは github で C 言語で検索をかけた際に star が多い順に上位にマッチした 3 件である．なお，検索日は 2022 年 5 月 24 日時点のものであり，その時点で最新のコードを調査した．

表 1　対象 OSS 一覧

OSS 名	C 言語ファイル数	総行数
darkforestgo[†1]	23	12915
hashcat-legacy[†2]	19	40962
openwebrtc[†3]	45	20024

3.2 予備調査から得られた知見

予備調査の結果，得られた知見を次に列挙する．

- 型名をそのまま変数名に使っているものがある
- 変数名のナンバリングが 2 から始まることが多い

型名をそのまま変数名に使っているものがあるとは，例えば Board という型で宣言された変数名が board であるといった，変数名と型名が同一である関係を指す．これらは特にユーザ定義の型で見られ，3 つの OSS で合計 247 個の変数がこれに該当した．

変数名のナンバリングとは，値を交換するような関数内で tmp1，tmp2 のように末尾に数字を振った名前のことを指す．末尾に数字がある変数は 3 つの OSS で合計 377 個あった．これらは常に 1 から順に数字を振るとは限らないことがわかった．

他にも一部で関数名に tree が含まれているとき，変数に parent という名前が使われているものや，変数名 db_key のように db と key といった命名が推測できうる語彙が使用されるものが見られた．

4 リサーチクエスチョン

調査にあたって設定した RQ を次に示す．

RQ1 型名と変数名にはどのような関係があるか

RQ2 関数名とその関数内の変数名には関係があるか

RQ3 変数名が複数の単語で構成されるとき，それぞれの単語間には関係があるか

RQ4 変数名の末尾にナンバリングをして区別する際，どのようにナンバリングされるか

5 事例調査

本研究での事例調査として次のような調査 1，2，3，4 を実施する．

†1 https://github.com/facebookresearch/darkforestGo
†2 https://github.com/hashcat/hashcat-legacy
†3 https://github.com/EricssonResearch/openwebrtc

調査 1　型名と変数名における関係調査

調査 2　関数名とその関数内の変数名における関係調査

調査 3　変数名の単語間における関係調査

調査 4　変数名のナンバリングに関する調査

これはそれぞれが RQ1，RQ2，RQ3，RQ4 に対応している．各調査の結果から対応する RQ に回答する．なお，ここでの調査対象は予備調査と同様の方法で選別しており，予備調査と同じ OSS を利用した．構文解析器には TEBA [5] を用いた．

5.1　調査 1 型名と変数名における関係調査

この調査の目的は，構造体などのユーザ定義型ではどのような変数名が使用されているか調べることである．そのために型名ごとに変数名を集計する．同時に変数名ごとの型名についても集計する．それぞれを調査 1-1，1-2 とする．

5.1.1　調査方法

調査 1-1 について

調査 1-1 では，型名ごとに宣言された変数名を集計する．ある型において特定の変数名が多く宣言されているとき，型と変数名に強い関連があると考えた．10 回以上変数宣言で使用されている型を集計対象とし，それぞれの型ごとで宣言されたすべての変数名を母集合として 80%が同一であった変数とその型を抽出した．例を図 1 に示す．なお，基本型は対象外として配列型とポインタ型は区別して扱う．またユーザ定義型はすべて独立した型とみなしており，元の型とは区別している．

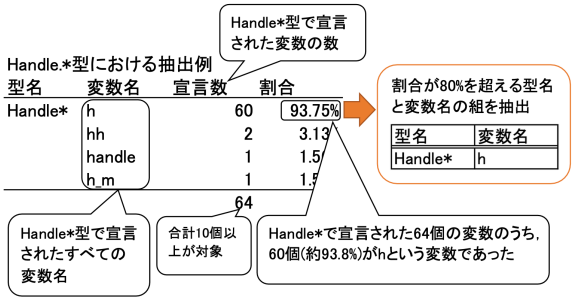

図 1　調査 1-1 の抽出方法

抽出したデータに対し，それぞれの型からの命名方法を目視確認し，命名方法の種類と該当数を調査する．

調査 1-2 について

調査 1-2 では変数名ごとにその変数の型をすべて集計する．調査 1-1 と同様の考え方で，ある変数名が特定の型で多く宣言されている場合にも型と変数名に強い関連があると考え，10 回以上宣言されている変数名を集計対象とし，それぞれの変数ごとで宣言に使用されたすべての型を母集合として 80%以上同一であった型とその変数を抽出した．例を図 2 に示す．

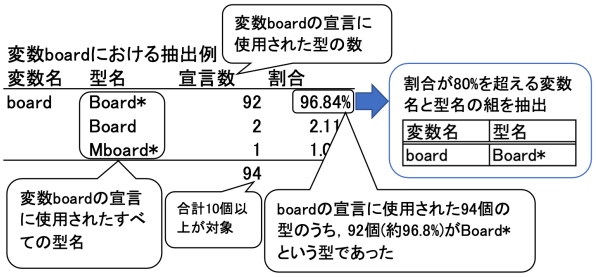

図 2　調査 1-2 の抽出方法

抽出したデータのうちユーザ定義型であるものに対し，こちらも同様に命名方法の種類と該当数を調査する．型は配列型とポインタ型を区別し，ユーザ定義型は独立したものとして扱う．

5.1.2　調査 1 結果

調査 1-1 の結果

OSS 別に抽出したデータに関する結果を表 2 示す．抽出型数とは，ある型で宣言されたすべての変数のうち，80%以上同一の変数であった型の数であり，割合は基本型を除いたすべての型のうちの何%に当たるかを示す．

表 2　調査 1-1 における型の抽出結果

OSS 名	抽出型数	全ユーザ定義型数	割合
darkforestgo	13	121	10.7%
hashcat-legacy	9	114	7.9%
openwebrtc	23	260	8.8%

表 3 は抽出結果の一例である．各 OSS のうち 5 件

ずつ例を示す.

表 3　調査 1-1 の抽出結果例

OSS 名	型名	変数名
darkforestgo	Handle*	h
	TreePool*	p
	GroupId4	ids
	ThreadInfo*	info
	Region*	r
hashcat-legacy	db_t*	db
	digest_t*	digest
	index_t*	index
	words_t*	words
	uint8_t*[ ]	buf
openwebrtc	GparamSpec*	pspec
	Gvalue*	value
	OwrSession*	session
	GObjectClass*	gobject_class
	GEnumValue[ ]	types

3 つの OSS における平均抽出型数は約 15 個，平均全ユーザ定義型数は約 165 個であり，おおよそ 8 から 11%程度の型が抽出された.

次に，OSS 別の命名方法調査結果を示す．抽出したデータの命名方法を調査した結果，次の 4 つの方法があると判断した.

方法 1　型名の頭文字を変数名にする

例：型名「Handle*」に対し，変数名が「h」

方法 2　型名をそのまま変数名にする

例：型名「Board」に対し，変数名が「board」

方法 3　型名を部分的に引用し変数名にする

例:型名「ThreadInfo*」に対し，変数名が「info」

方法 4　型名の略語を変数名にする

例：型名「GParamSpec*」に対し，変数名が「pspec」

抽出結果の変数名がそれぞれどの命名方法に該当するか集計した結果を表 4 に示す．なお，その他とは型名から命名されていないものを指す.

表 4　調査 1-1 の命名方法集計結果

OSS 名	方法 1	方法 2	方法 3	方法 4	その他	合計
darkforestgo	5	0	3	0	5	13
hashcat-legacy	0	0	8	0	1	9
openwebrtc	0	1	16	4	2	23

命名方法としては型名の一部を使うことが最も多いとわかる．型名からの命名ではないと考えられるものは平均約 17.8%存在した.

調査 1-2 の結果

調査 1-2 の抽出したデータに関する結果を表 5 に示す．抽出変数数とは，それぞれの変数を宣言したすべての型のうち，80%以上同一の型で宣言された変数の数であり，割合にてすべての変数のうちの何%に当たるかを示す． 表 6 は抽出結果の一例である.

表 5　調査 1-2 における変数の抽出結果

OSS 名	抽出変数数	全変数数	割合
darkforestgo	21	676	3.1%
hashcat-legacy	31	820	3.9%
openwebrtc	37	687	5.4%

表 6　調査 1-2 の抽出結果例

OSS 名	変数名	型名
darkforestgo	board	Board*
	player	Stone
	r	Region*
	bl	TreeBlock*
	PatternMove*	mv
hashcat-legacy	db	db_t*
	in	plain_t*
	dgst	digest_t[ ]
	index	index_*
	chain_buf	chain_t*
openwebrtc	object	GObject*
	pspec	GParamSpec*
	user_data	gpointer
	codec_type	OwrCodecType
	tag	gchar*

3 つの OSS における平均抽出変数数は約 30 個，変数の個数平均は約 728 個であった．抽出された変数は割合的に見て少ない．これはあらかじめ宣言回数で調査対象を選別したことと，型名は配列型やポインタ型を区別したことによる影響が大きい.

次に，命名方法集計結果を示す．命名方法は調査 1-1 で示した 4 つで集計する．抽出した変数のうちユーザ定義型のものは darkforestgo が 11 件，hashdcat-legacy が 18 件，openwebrtc が 37 件であった．それらの変数名がそれぞれどの命名方法に該当するか集

計した結果を表 7 に示す.

表 7　調査 1-2 の命名方法集計結果

OSS 名	方法 1	方法 2	方法 3	方法 4	それ以外	合計
darkforestgo	2	1	3	1	4	11
hashcat-legacy	2	0	9	1	6	18
openwebrtc	0	0	21	1	15	37

命名方法としては型名の一部をそのまま使うことが最も多いとわかる．型名の影響を受けていないと考えられる名前は平均約 37.9%ある．

また，調査 1-1 と 1-2 のどちらにも抽出された型名と変数名の組は darkforestgo は 3 件，hashcat-legacy は 3 件，openwebrtc は 12 件あった．これらは型名と変数名がほぼ 1 対 1 で決まっている可能性が高いものだといえる．

以上の結果から RQ1 に対する回答を示す．

RQ1 に対する回答

調査 1-1 から，型名と変数名の間に命名関係があると考えられるものは，OSS 中のすべてのユーザ定義型のうち 8 から 11%程度が該当し，そのうち 82.2%は実際に型名から命名されていた．一方，調査 1-2 から，すべての変数のうち型名との命名関係があると考えられるのは 3 から 5%程度が該当すると考えられ，そのうちのユーザ定義型の 62.1%が実際に型名から命名されていた．命名関係の種類には 4 種類の命名が考えられるが，そのうちの型名を部分的に引用した命名方法が最も多い命名方法である．

5.2　調査 2 関数名とその関数内の変数名における関係調査

この調査の目的は，変数名中で利用される単語が関数名から推測できるような関係があるかを調べることである．ここでは関数名や変数名を分解し，より小さな単位である単語に焦点を当てる．そのために関数名を構成する単語と，その関数内における変数名を構成する単語同士の組み合わせを集計し，関数名から変数名の単語利用の推測可否を調査する．

5.2.1　調査方法

はじめに関数名とその関数内の変数名を単語に分解し，関数の単語ごとに変数の単語を総当たりした組み合わせを集計する．

調査にはある程度の数出現した組み合わせが良いと考え，OSS 内で 10 回以上使用されている関数の単語から得られた組で，かつ関数の単語が作る各組の数がその関数の単語の使用回数に対して 10%以上の割合となるものを調査対象として抽出した．例を図 3 に示す．

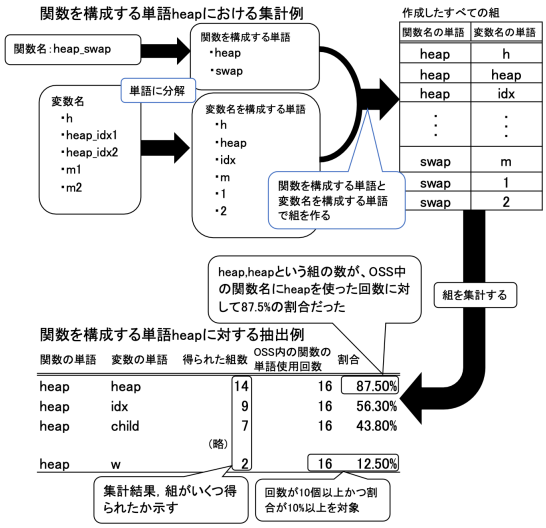

図 3　調査 2 における組の抽出方法

次に抽出した関数の単語と変数の単語の組を目視確認する．関数の単語と変数の単語が同名であったり，概念として近い関連があると判断できるものは関数名から変数名の利用が推測できると考え，そのような関係性を持った関数の単語と変数の単語の組を調査する．その後，これに該当する組を 1 つでも持った関数の単語を抽出する．ただし，接頭辞や接尾辞のような汎用的な単語だと判断されるものは，その単語同士の組に関連はないものとする．

5.2.2　調査結果

調査 2 の結果を表 8 に示す． 表 9 は抽出結果の一例である． 調査 2 の結果として，変数名の推測ができる関数の単語は 3 つの OSS で平均約 11.7 単語見つ

表 8　調査 2 の抽出結果

OSS 名	抽出した関数の単語数	関数の全単語数	割合
darkforestgo	7	423	1.7%
hashcat-legacy	5	368	1.4%
openwebrtc	23	342	6.7%

表 9　調査 2 における抽出結果の例

OSS 名	関数の単語	推測できる変数の単語
darkforestgo	Board	board
	heap	parent
	tree	parent
hashcat-legacy	digest	salt
	hashcat	digests
	hashing	salt
openwebrtc	renderer	renderer
	server	address
	video	fps

かった．1 語につき推測可能な変数名は，平均約 1.9 単語であった．

以上の結果から RQ2 に対する回答を示す．

RQ2 に対する回答

関数名とその関数内の変数名には，単語単位で見たとき関数名から変数名の命名が推測できるものはごくわずかにある．それらは関数名の命名に使用されたすべての単語のうち，およそ 1 から 7%程度，平均にして約 3.1%の関数の単語において確認され，各単語約 1.9 単語が推測できる．

5.3　調査 3 変数名の単語間における関係調査

この調査の目的は，変数名中においてある単語の利用が別の単語から推測できる関係はあるかを調べることである．そのために複数の単語で構成される変数を対象に，単語同士の組み合わせを集計し，単語利用の推測可否を調査する．

5.3.1　調査方法

はじめに複数の単語から構成される変数名を単語に分解し，変数別に組み合わせを総当たりで作成し集計する．調査 2 と同様の考えで，調査にはある程度の数出現した組み合わせが良いと考え，OSS 内で 10 回以上命名に使用された単語の作る組で，かつ各単語の作る組のうち 10%以上がどちらの単語も同一となる組であったものを調査対象として抽出した．例を図 4 に示す．

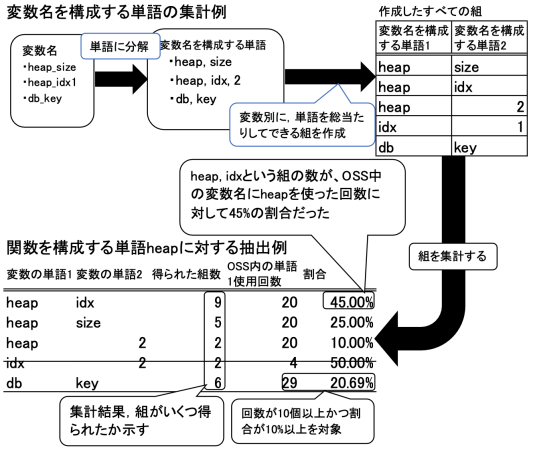

図 4　調査 3 における組の抽出方法

次に抽出した変数の単語の組を目視確認する．単語同士に概念として近い関連があると判断できるものは単語から単語の利用が推測できると考え，そのような関係性を持った変数の単語の組を調査する．その後，調査 2 同様にこれに該当する組を 1 つでも持った単語を抽出する．ここでも汎用的な単語だと判断できるものは，その組に関連がないものとする．目視確認の例を図 5 に示す．

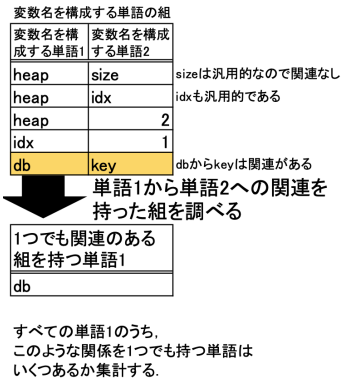

図 5　調査 3 における組の目視確認例

5.3.2 調査結果

調査 3 の結果を表 10 に示す.

表 10　調査 3 の抽出結果

OSS 名	抽出した変数の単語数	全単語数	割合
darkforestgo	2	222	0.9%
hashcat-legacy	6	255	2.4%
openwebrtc	7	249	2.9%

表 11 は抽出結果の一例である.

表 11　調査 3 における抽出結果の例

OSS 名	変数の単語	推測できる変数の単語
darkforestgo	black	win
	time	elapsed
	time	limit
hashcat-legacy	db	key
	digests	sha
	nonce	client
openwebrtc	transport	agent
	rtcp	port
	video	renderer

調査 3 の結果として，関連する単語の利用が推測できる単語は 3 つの OSS で平均約 5 単語見つかった．1 語につき推測可能な単語は，平均約 1.5 単語であった.

以上の結果から RQ3 に対する回答を示す.

RQ3 に対する回答

変数名が複数の単語で構成されるとき，単語から関連のある単語の利用が推測できるものはごくわずかにある．それらは変数名の命名に使用されたすべての単語のうち，およそ 1 から 3% 程度，平均にして約 2.1%の単語において確認され，各単語約 1.5 単語が推測できる.

5.4 調査 4 変数名のナンバリングに関する調査

この調査の目的は，変数名を区別する際にはどのようにナンバリングされているか調査することである．そのためにナンバリングされた単語を抽出し，番号の振られ方に関して集計する.

5.4.1 調査方法

調査 4 では，末尾にナンバリングされている変数をすべて抽出し，変数ごとに番号がどのように振られているかパターン別に集計する．数字が飛ばされず 0 か 1 から順に振られているものを正しいものとして，ナンバリングの分類についても集計する.

5.4.2 調査結果

調査 4 の結果を表 12，13，14 に示す．ナンバリングパターンとは変数名に割り当てられた数字の組み合わせを示す．例えば，a と a2 という変数名があったとき，これらのパターンは「なし,2」に該当する．他にも，a2 という変数のみがあったときは「2」というパターンに該当する．それらのパターンに該当する組み合わせの数を該当数にて示す.

表 12　darkforestgo の集計結果

ナンバリングパターン	該当数
なし,2	19
2	12
1,2	11
1	1
なし,2,3	1
なし,0,1,2	1
合計	45

表 13　hashcat-legacy の集計結果

ナンバリングパターン	該当数
1,2	50
0,1	24
0,1,2,3	15
1,2,3	7
なし,2	6
2	3
1	3
なし,0,1	2
3	1
0	1
なし,1	1
なし,1,2	1
合計	114

ナンバリングパターンには大きく分けて 3 種類の分類がある.

分類 1　0 か 1 から順番にナンバリングされている

例：「idx1，idx2」,「temp0，temp1」

表 14　openwebrtc の集計結果

ナンバリングパターン	該当数
1,2	2
1	1
なし,2	1
合計	4

分類 2　番号なしとナンバリングが混在している
　例：「ids，ids2」,「p，p1，p2」
分類 3　誤っている
　例：「id2」のみ,「b2」のみ

このうち，分類 1 は正しく番号が振られていると言えるが，分類 2 は番号によって正しいと扱えるとは限らないことがある．

表 15 で OSS 別に変数のナンバリングがそれぞれどの分類に該当するか集計した結果を示す．

表 15　ナンバリングの分類についての集計結果

OSS 名	分類 1	分類 2	分類 3	合計
darkforestgo	11	21	13	45
hashcat-legacy	96	10	8	114
openwebrtc	2	1	1	4

調査 4 の結果から，3 つの OSS におけるナンバリングされた変数の平均数は約 54.3 個だった．そのうち約 13.5%は誤ったナンバリングをされていた．ただし，OSS の内容によってあらかじめ想定される変数の数は異なるために，その数には大きな差が出た．

以上の結果から RQ4 に対する回答を示す．

RQ4 に対する回答

今回の OSS において，変数名の末尾にナンバリングをして区別する際は，1 からまたは 2 から番号が振られる場合が多い．ナンバリングをされている変数のうち，誤ったナンバリングをされているものは 3 つの OSS の平均で約 13.5%であった．数値計算のようなあらかじめ複数の変数の使用が想定される場合は正しいナンバリングをされることが多い．

6　考察

6.1　ソフトウェア開発支援への適用について

各調査結果から変数名に対するソフトウェア開発支援への適用可能性について議論する．

調査 1 について

調査結果から変数名に対する開発支援に対し適用できると判断した．OSS から命名関係があると期待できる結果が，すべてのユーザ型のうち 10%程度確認されたからである．命名方法をパターンとして定義することで規則へ発展できると考えられる．

調査 2 について

調査結果から変数名に対する開発支援に対し適用するのは難しいと判断した．関係が認められる調査結果がごくわずかであったからである．

調査 3 について

調査結果からは変数名に対する開発支援に対し適用するのは難しいと判断した．調査 2 同様，関係が認められる調査結果がごくわずかであったからである．

調査 4 について

調査結果からは変数名に対する開発支援に対し適用できると判断した．結果からナンバリングされる変数は番号を飛ばして数字を振ることがプロジェクトによっては多く，その分修正箇所となりうる箇所があるといえるからである．

次に調査結果から考えられる変数名命名支援とリファクタリング支援への適用について考察する．

命名支援はコードを作成する際に行う支援である．そこまでに記述したコードを入力し，型名に関する規則を用いて変数名の命名候補を出力する．ここではあらかじめ既存のコードから抽出した規則を利用する．

リファクタリング支援は一度作成したコードを修正する際に行う支援である．修正したいコードを入力し，型名に関する規則やナンバリングに関する規則を用いて修正が必要な変数や修正案を出力する．修正したいコードには既に何かの規則に従っている変数があると考えられるので，あらかじめ抽出した規則と修正したいコードの規則を比較しながら適用すべき規則を選択する必要がある．

現状の調査結果から，型名に関しては 4 種類の命名方法が規則化できる可能性がある．しかし，命名方法について語の略し方などさらなる詳細化の余地があり，その内容を今後議論し正しい規則を精査する必要がある．ナンバリングに関する規則には，変数名が

0 または 1 から番号順に振られている状況を正しい規則として採用することができる．

6.2 調査方法のレビュー

本研究で調査を実施したところ，調査 2，3 では開発支援に適用できる結果を得られなかった．考えられる要因としては次のものが挙げられる．

1. 関係性がそもそも希薄である
2. 関係性を適切に集計できていない

まず 1 つが関係性そのものが希薄である場合である．前提として本研究では仮引数と局所変数に主眼を置いて調査を実施しているが，それらは限定的な範囲で使用され，関数やクラスとは違いそれ自体が入出力を持たない．よって関係性そのものが希薄になりがちだと言える．調査の際には数で調査対象を選別したこともあり，関係性があったとしてもそれらを調査できていない可能性がある．

2 つ目が関係性はあるものの，それを適切に集計できていない場合である．この場合は集計方法を改善することで関係性の発見が期待できる．例えば変数名に対して行った調査 3 について，変数の構造には接頭辞や接尾辞といった単語の使用場所にも傾向があると考えられることから，変数内における単語の使用場所に関する集計を行うといった案が挙げられる．

6.3 妥当性への脅威

本研究で実施した調査について，妥当性の観点から懸念点となるものを次に示す．

- 調査方法が主観的であること
- 調査対象が例外的であるリスクがあること
- 構文解析が正確でない可能性があること

調査方法が主観的であること

本研究にて実施した調査にはいずれも目視確認による定性的な判断で実施している部分がある．今回の調査ではこの調査を 1 人の視点から実施しており主観が含まれている．

調査対象が例外的であるリスクがあること

本研究では妥当性を考慮し，調査対象の OSS は github で C 言語で検索したうち数件を扱っている．しかし命名に関する品質が保証されているわけではないので，本来考えられる規則から外れたものを調査している可能性がある．加えて，今回は予備調査にて調査した OSS を事例調査にも利用しており，調査結果が他の OSS においても同様のことが言えるということが担保されていないという点も懸念点である．

構文解析が正確でない可能性があること

本研究での調査の際は，プログラマが読むコードをそのまま構文解析している．つまり構文解析の前処理をしていない．しかし，本来は前処理を実施したコードでないと正確な構文解析はできないので，調査結果は構文解析結果が不正確である可能性がある．

7 まとめ

本稿では，変数名に対する開発支援に必要な規則に関する事例調査を実施し，結果から開発支援への適用について考察した．調査 1，4 については，開発支援へ適用可能性を持った関係性があるという結論に至ったが，調査 2，3 については，開発支援へ適用可能な結果は十分に得られなかった．今後の課題として，妥当性を担保するように調査方法を見直すこと，開発支援方法を提案することが挙げられる．

謝辞 本研究の一部は 2022 年度南山大学パッヘ奨励金 I-A-2 の助成を受けた．

参 考 文 献

[1] Boswell, D. and Foucher, T.: リーダブルコード: より良いコードを書くためのシンプルで実践的なテクニック, Theory in practice series, オライリー・ジャパン, 2012.

[2] Kashiwabara, Y., Onizuka, Y., Ishio, T., Hayase, Y., Yamamoto, T., and Inoue, K.: Recommending verbs for rename method using association rule mining, *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 2014, pp. 323–327.

[3] 福田宏樹, 早瀬康裕, 北川博之: クラス名命名を支援する段階的なフレーズ推薦手法, 研究報告ソフトウェア工学 17, 2016.

[4] Abebe, S. L. and Tonella, P.: Automated Identifier Completion and Replacement, *2013 17th European Conference on Software Maintenance and Reengineering*, 2013, pp. 263–272.

[5] 吉田敦, 蜂巣吉成, 沢田篤史, 張漢明, 野呂昌満: 属性付き字句系列に基づくソースコード書き換え支援環境, 情報処理学会論文誌, Vol. 53, No. 7(2012), pp. 1832–1849.