ペアプログラミングでの対話における話題の分析

木村慎太郎 †， 羽山徹彩 †， 國藤 進 †

† 北陸先端科学技術大学院大学知識科学研究科

プログラマは，プログラミング中は同時に複数の抽象化レベルに対して関心を維持しなければならず，それがプログラミングの難しさの理由の 1 つであるとされている．ペアプログラミングは，それに対応して，2 人のプログラマが同時に異なる抽象化レベルに対して関心を持つため有効であるとされてきた．本研究では，ペアプログラミングにおける両者の関心の違いを，両者の対話の中に現れる問題提起に着目して分析を行った．結果として，両者の間では問題提起の内容には役割による抽象度レベルの差は認められず，両者の抽象度レベルによる関心の違いは実際には起こらないという結果が得られた．

Topic Analysis of Spoken Dialogue in Pair Programming

Shintaro KIMURA † Tessai HAYAMA † Susumu KUNIFUJI †

† School of Knowledge Science, Japan Advanced Institute of Science and Technology

The difficulty of computer programming is caused by maintaining programmer's concern to several levels of abstraction at the same time. Pair programming has been put forward as an programming style to conquer that difficulty because two programmer participated in pair programming concern to different levels of abstraction at the same time in programming. This study presents the result of a protocol analysis of ten pair programing sessions of graduate students. The analysis focuses on the deference of levels of abstraction of issues raised by pairs. In the result, such deference couldn't be observed. We suggest that the difference of pair's concern in pair programming is not occurs in actual.

1 序論

コンピュータプログラミングは，一般に高度に複雑な作業であるとされている．その理由として，Pennington ら [1] は，同時に複数のレベルの情報に気を使う必要があるためとし，“プログラマにとっての主な問題は，根本的に異なる問題空間を整理することである”としている．要するに，高度に構造化され論理的で制約のあるコンピュータプログラム領域と乱雑な現実世界の両方に関わる複雑さを強調している．また，そのプログラム領域のみを見ても，プログラマは，システム全体のプログラムの構造からプログラミング言語の文法に至るまで，複数の抽象化レベルに関わる必要がある．実際，Brooks[2] は，プログラミングを “問題領域から中間の諸領域を通り，プログラミング領域への関数を組み立てること”として記している．

ペアプログラミング（以下 PP）は，2 人のプログラマが横並びで 1 台のコンピュータに向かい，同一の設計，アルゴリズム，コード，テストについて継続的に協調して作業を行うプログラミングスタイルである．PP を取り入れることによる効果として，ソフトウェア開発事業への導入による生産生の向上効果 [3] や，プログラミング教育への導入による教育的効果 [4] が知られている．

PP が有効である理由の 1 つとして，ペアの 2 人のプログラマの関心が異なる抽象化レベルに分かれることにより，1 人のプログラマが 1 つの抽象化レベルに集中することができることにあるとされている．さらに言えば，キーボードを操作するプログラマが抽象化レベルの低い領域へ関心を持ち，もう一方のプログラマは抽象化レベルの高い領域へ関心を持つとされている．例えば，PP をエクストリーム・プログラミング (以下 XP) の 1 つの習慣として最初に一般的に広めた Beck[5] は，PP での 2 人のプログラマの役割について，次のように述べている．

> There are two roles in each pair. One partner, the one with the keyboard and the mouse, is thinking about the best way to implement this method right here. The

other partner is thinking more strategically:

- Is this whole approach going to work?
- What are some other test cases that might not work yet?
- Is there some way to simplify the whole system so the current problem just disappears?

このように，キーボードを操作するプログラマは実装方法を気遣い，もう一方のプログラマはより戦略的な問題について考えているとしている．また，Hazaan らによる XP に関する文献 [6] において，

> The one with the keyboard and the mouse thinks about the best way to implement a specific task; the other partner thinks more strategically. As the two individuals in the pair think at different levels of abstraction, the same task is thought about at two different levels of abstraction *at the same time*.

と，2 人のプログラマは同一のタスクについて異なる抽象度レベルで同時に思考を行うとしている．さらに，ペアプログラミングの教科書として最も広く引用されされている Williams と Kessler による文献 [7] では，

> One of the pair, called the *driver*, is typing at the computer or writing down a design. The other partner, called the *navigator*, has many jobs, one of which is to observe the work of the driver, looking for tactical and strategic defects. Tactical defects are syntax errors, typos, calling the wrong method, and so on. Strategic defects occur when the driver is headed down the wrong path — what is implemented just won't accomplish what needs to be accomplished. The navigator is the strategic, long-range thinker.

と記され，ここでは，コンピュータへの入力を担当するプログラマを “ドライバ”，もう一方のプログラマを “ナビゲータ” と呼び，ナビゲータはより戦略的に長期的視点から考えるとしている．

しかし，以上で述べたような PP における 2 人のプログラマの抽象化レベルでの関心の違いは，近年行われた実際の PP の現場を観察した諸研究 [8, 9, 10, 11] においては，観察されておらず，一般的に認知されている 2 者間の関心の違いに関する記述と，実態とは乖離していると指摘している．

本研究の目的も，この PP における 2 人のプログラマの関心の違いを検証することである．本研究では，ペア間で関心に違いが生じた場合，主体の着眼点が変わり，提起される問題も変わると考え，PP 内で双方が行う問題提起に着目してペア間の発話プロトコル分析を行い，双方から提起される問題の抽象化レベルに対する分布の比較を行った．

2 関連研究

本研究のように，PP 内で発生する発話を分析対象としている研究として，2 つの関連研究が挙げられる．

Chong と Hurlbut[10] は 2 つの会社の開発チームの PP に対して 4ヶ月に渡ってエスノグラフィック・スタディーを行っており，PP 内で発生する各議論での両者の発話内容を比較している．2 人のプログラマは議論の議題となっている問題について同じ戦略的な “range” や抽象化レベルで議論し，考えていると主張している．

Bryant ら [11] は，学生ではないプロのプログラマによる 24 回の PP セッションの発話プロトコル分析を行い，両者の発話内容の抽象化レベルに対する分布を調査している．その結果，ドライバとナビゲータではその分布に差はなく，同じ抽象度レベルで相互作用しているとして，ドライバ・ナビゲータではなく，それぞれが同じ役割を担うことから，“タッグチーム” という PP での新たな相互作用のモデルを提案している．

これら 2 つの研究に共通しているのは，PP での発生する議論の中での対話の全ての発話内容のドライバとナビゲータでの比較を行っているという点にある．議論中は，議論の議題に関心を奪われている可能性があるため，2 人の話者の抽象化レベルに差が出なかった可能性があり，また，議論が行われていない間のプログラミング中の主体の関心を反映したものとは必ずしも言えない．また，Bryant ら [11] の研究は，PP 内での発話を話者主体のプログラミング中の認知過程を反映したものとしているが，PP で起こる発話は対話が中心であり，対話での発話は主体の認知過程を見るための発話思考法のような内省的発話とは異なる．

本研究はこれらの研究の問題点を踏まて．PP 内で断続的に発生する各議論が開始される発話，すなわち問題提起の発話に着目した．議論の対象となっている議題，すなわちその議論で解決すべき問題の内容に着目し，ドライバ・ナビゲータのどちらがどのような問題を提起しているのかを比較している点でこれらの研究と異なる．

3　仮説

もし仮に，一般的に認知されているように，PP において，ドライバはより抽象化レベルが低い領域へ関心を持ち，ナビゲータはより抽象化レベルの高い領域へ関心を持つ場合，双方（ドライバ・ナビゲータ）のプログラミング中の着眼点も異なり，双方から提起される問題点も異なってくると考えられる．より具体的に言えば，ドライバは主に，実装中のソースコードの文法ミスや綴りミスなど，より抽象化レベルの低い問題を提起し，ナビゲータは現実世界や問題領域に関わる，より抽象化レベルの高い問題を提起することで，双方の提起する問題の分布が Fig.1 に示すようになると考えられる．

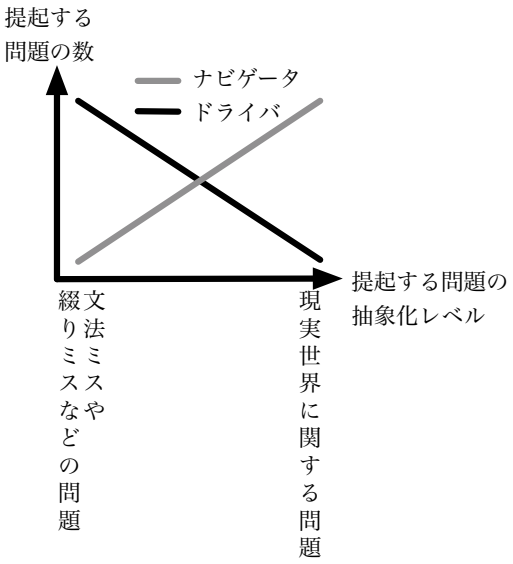

Fig. 1　ドライバ，ナビゲータが提起する問題の抽象化レベルに対する分布

4　方法論

4.1　データ収集

本研究では 1 セッション 90 分の PP を 10 セッションの収録を行い，被験者の発話内容と，各場面でドライバおよびナビゲータどちらの役を各被験者が担当しているのかを記録した．

4.1.1　被験者

Java でオブジェクト指向プログラミングができる大学院生を被験者とした．

本研究が重視するのはドライバとナビゲータという役割による差異が及ぼす影響であり，ペア間でのプログラミング経験や能力の差による影響を避けるため，プログラミング能力が同等レベルの学生のペアにより実験を行った．また，お互いにコミュニケーションを円滑に取り合いやすく，お互いにプログラミング能力を知り合っているペアの方がより効果的な PP が可能であることが分かっている [?]．そこで，お互いに同じ研究室に所属して，顔見知り同士であるペアを選んだ．

4.1.2　収録方法

音声とコンピュータ画面の収録は，音声と画面の両方を同時にキャプチャできる Microsoft 社の Expression Encoder 3 を用いた．また，各場面でキーボード・マウスをペアのどちらが操作しているのか，各発話がドライバとナビゲータのどちらからのものか，被験者が指示語使いながらどこを指さしているかを確認するため，被験者の顔面，キーボード，マウス，ディスプレイが画角に収まるようにカメラを固定し，図 2 に示すような映像を録画した．被験者の発話は，図 2 に示すように，コンピュータに接続されたヘッドセットのマイクを用いて録音した．実験者は被験者の背後から被験者の指さしを監視し，ディスプレイや問題用紙，メモ用紙の中でどこを指さしているのかをメモにより記録した．

Fig. 2　カメラで録画した映像の 1 場面

4.1.3 タスク

被験者には，実際に PP をする前に，Williams ら[7] による PP の方法と効果を教授した．その際，ドライバとナビゲータに関心の違いがあることも被験者に伝えた．

被験者には９０分間の PP セッションを３回行ってもらい，１回目のセッションは PP の練習と開発環境に慣れてもらうために実施し，後の２セッションを収録対象とした．

プログラミング課題は，入出力の形式を規定したのみの問題を提示し，内部設計はペアにまかせた．収録の対象となるセッションの内１回目のセッションでは○×ゲームを実装する課題，２回目のセッションでは自動販売機のコントロール部分のプログラムを実装する課題を課した．その時の課題内容は付録 A に示す．

課題プログラムの開発環境は，表１に示す．

Table 1　課題プログラムの開発環境

OS	Windows Vista Ultimate
プログラミング言語	Java (JDK 6)
統合開発環境 (IDE)	Eclipse 3.5 Galileo

ドライバとナビゲータの役は，一般的に行われている PP に習い，ペア間で固定せずに自由に交代することを許可した．

4.2 データ分析

データから抽出すべき情報は，問題提起の発話と，その提起された問題の内容ある．本研究では，問題提起の発話を１つの議論における最初の発話と定義し，問題の内容をその議論の中での話題と定義した上で，問題提起の発話の同定とそれにより提起された問題の分類を行い，どのように問題提起の内容の傾向が分布しているのかをドライバ・ナビゲータ間で比較する．

4.2.1 問題提起の発話の同定

本研究では，問題提起の発話を１つの議論における最初の発話と定義した．したがって．問題提起の発話の同定をするには，２者間の対話でのどこからどこまでが１つの議論であるのかを同定する必要がある．対話は、一般に複数の話題によって構成され，ある１つの話題について話されている連続した発話を対話セグメントという単位に分けられることが知られている[12]。本研究では，この対話セグメントを１つの議論の単位とし，１つの対話セグメントの最初の発話を問題提起の発話とする．

対話セグメントの同定の方法は，山下ら[13] によるタグ付けによる対話セグメント同定の方法を用いた．今回の実験では，プログラミング課題遂行に関してその内容が理解できるだけの動詞と必要な格要素を持っている発話を１つの文章として，これを発話単位とした．分析者は，また，タグ付けは３人の分析者にそれぞれ個別に独立した環境で作業を行わせ，３人のうち２人以上が開始タグを付与した発話を対話セグメント開始境界であるとした．分析者はソフトウェア工学を専攻する大学院生が担当した．

4.2.2 提起された問題の抽象度レベルによる分類

本研究では，提起された問題の内容をその議論の中での話題と定義した．したがって提起された問題の分類をするには，前節で示した方法により抽出された１つ１つの議論をその話題の内容により分類する必要がある．

抽象度レベルによる分類方法は，Table.2 に示すとおりである．Pennington[14] による詳細領域 (DE)，プログラム領域（そして現実世界領域の３つの領域に加えて，本研究では，さらに２つの領域（SY，BR）と，その他のプログラミングの抽象度レベルとは関連がない話題も加え，Table.2 中のタグを各話題セグメントの開始位置に付与した．

このタグ付けもソフトウェア工学を専攻する大学院生３人により個別に独立して行い，３人の内，１人の評価が分かれた場合は，一致している他の２人の評価を採用した．３人とも評価が分かれている場合は，その話題が曖昧なものであるとして，その他の話題を表す VA のタグを付与した．

実際に，話題の分類のためのタグ付けの結果を Table. 3 に示す．Table 中の話者の列は話者を識別するための記号（話者名の頭文字)，役割の列は役割を識別する記号（D：ドライバ，N：ナビゲータ)，話題タグは表２で定義した対話セグメントでの話題を表すタグ，開始タグは，対話セグメントの開始位置を示すタグである．発話内容の（）はフィラー，【】は相づちを示す．

以上の手順で PP の１つのセッションにおいてプログラマが提起する問題の抽象度レベルを分類

Table 2　問題の分類した場合の分類

タグ	話題	例
SY	言語の文法や綴り 意味論は含まない.	スペルミスや文法のミスへの言及
DE	詳細領域 （変数・演算子・識別子）	変数やメソッドの命名, 変数の型はどれを用いるか
PR	プログラム領域	プログラムの構造をどうするか. アルゴリズムやデータ構造はどれを用いるか.
BR	問題領域とプログラム領域間の橋渡し	要求をどのようにプログラムへの落とし込むか. 要求通りのプログラムになっているか
RW	現実世界や問題領域	ユーザが取り得る行為
VA	その他	進捗に関する話題, 開発ツールの操作に関する話題, 抽象度レベルが不確定な話題

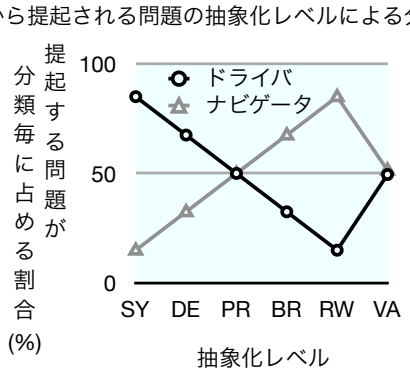

Fig. 3　ドライバとナビゲータの関心が異なる場合に提起される問題の抽象化レベルの分布を表すグラフ

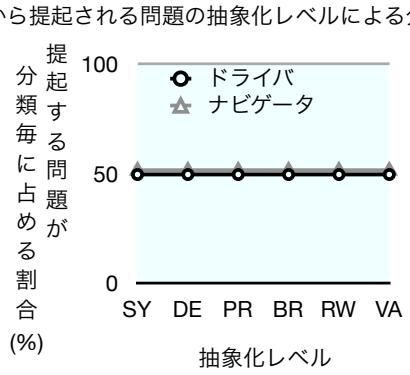

Fig. 4　ドライバとナビゲータの関心が同じ場合に提起される問題の抽象化レベルの分布を表すグラフ

し，それぞれの分類での提起される問題の分布をグラフ化すると，仮に 2 人のプログラマの関心が異なり，仮説通りにナビゲータが抽象度レベルの高い問題提起を主に行い，ドライバが抽象度レベルの低い問題提起を主に行っている場合は，Fig. 3 の通りの結果が得られ，ペア間の関心が同じ場合は，Fig. 4 の通りの結果が得られると考えられる.

5　結果

分析対象となった 10 セッションの 1 セッション当たりの平均発話数は 617，平均議論数は 98 であった．今回，複数の分析者により評価を行ったが，分析者間での評価の一致度を示すカッパ係数は 0.66 以上であり，十分信頼できる分析結果が得られた.

単純に 1 セッションでのドライバ，ナビゲータ双方からの問題提起の回数を比較した結果は Fig.5 に示す通りである．このように，ドライバ，ナビゲータ双方が提起する問題の数はそれぞれの抽象度のレベルにおいて，概ね同様に分布していることが分かる.

抽象度のレベル毎に分類された問題を提起した双方が占める割合で比較した結果は Fig.6 に示す．こちらも，前章で示したようなペア間で関心の違いがあった場合の分布の特徴は確認されなかった.

個々のプログラマに着目して，その個人が，役割によってどのように提起する問題が変化するかをみるため，1 個人が提起する問題を，分類当たりに占める提起時の役割別の割合を Fig.7 に示す．詳細領域の分類（DE）とその他の分類（VA）で有意差がわずかに見らた．書き起こし資料で DE

Table 3　話題の分類のためのタグが付与された書き起こし資料の一部

連番	話者	役割	話題タグ	開始タグ	発話内容
267	F	D	DE	S	ここは int でいいかな。
268	K	N			（うーん）もしくは。boolean か。
269	K	D			0 にしとくとあとあといいとおもうんだ。
270	F	N			（うーん）そうだね。
271	K	N	DE	S	初期化はさ、【うん】そこはマジックナンバ 使うのはよくないから、BLANK かな。
272	F	D			あそっか。
273	K	N	BR	S	プレーヤーはぜったい二人だよね。
274	F	D			そうだね。
275	K	N			だったら、結局、private の enum ですよね。
276	F	D			うん。
277	K	N	SY	S	ここにセミコロンが無いよ。
278	F	D			ほんまや
279	F	D	SY	S	あれ、なんでや。
280	K	N			ここ括弧でくくるんじゃないの。
281	F	D			まいいや 0 じゃなくても。
282	K	N			(まー）そうだね。
283	F	D	VA	S	で、これでとりあえず終わりじゃない。
284	K	N			うん。
285	F	D	PR	S	これ、配列作った方がよくね。
286	K	N			（んま、）for 文使うときにいいよね。
287	F	D			そうだね。

と VA に割り当てられた問題提起の発話を見てみると，DE ではドライバがナビゲータに,「このメソッド名はどうしようか」,「このへ運数の型は○○の方がいいかな」などの変数やメソッドの命名や型の選択に関する問いかけを頻繁に行っており，VA では,「このメソッドはこれで完成でいいかな」,「これで OK」など，作業が終わったことの確認をナビゲータに対して行っている発話が頻繁に見られた．

結果をまとめると，1 セッション当たりにドライバとナビゲータ双方から提起される問題の抽象度レベルでの分布に差は無かった．また，1 人が提起する問題には，その役割によってあまり変化することは無かった．これらことから，ドライバとナビゲータ間では抽象度レベルに対してその関心に違いはなく，ペアプログラミングでのプログラマの関心は，役割（ドライバ・ナビゲータ）によるというよりは，個人差によるものが大きいと考察できる．

6　本研究の制約と今後の課題

本研究の制約として，1 つ目に，被験者が PP の経験があまり無いプログラマであったことが挙げられる．PP に熟練したプログラマでは異なる結果が期待されるため，更なる検証が必要である．2 つ目に，プログラミング課題の規模が小さく，抽象度レベルの高い領域に関心が及ぶことが全体的に少なかったことが挙げられる．よって，複雑で大規模なプログラミング課題で検証してみる必要がある．3 つ目は，抽象化レベルの尺度として，今回は Pennington ら [14] によるものを用いたが，プログラム領域（PR）に関心が偏ってしまった．プログラム領域を更に細分化して分析する必要がある．また，プログラミングにおける抽象化レベルを尺度としてもちいたが，プログラミングではなく，作業の手順や計画の抽象度での分布を検証する必要がある．そして最後に，今回の実験では，プログラマには関心を分けることよりも，プログラミング課題の達成を優先させたことがあった．関心を分けることを優先させた場合，PP の効果が上がるかどうかを検証する必要があるといえる．

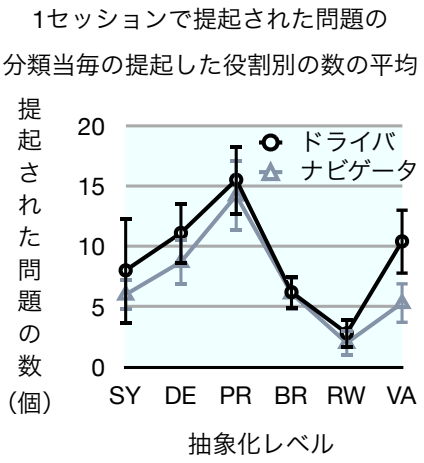

Fig. 5　実際にドライバとナビゲータそれぞれが提起した問題の抽象度レベルによる分布（エラーバーはデータが正規分布に分布したと仮定した場合の 90 %信頼区間）

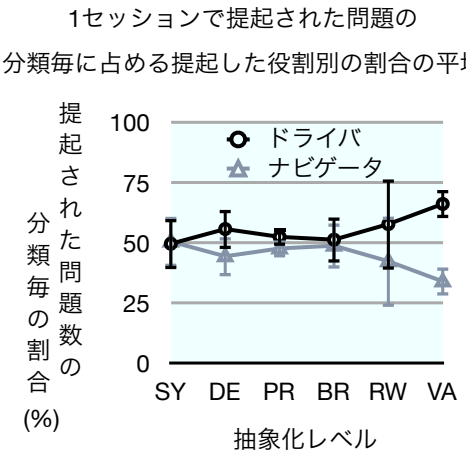

Fig. 6　実際にドライバとナビゲータそれぞれが提起した問題が 1 つの各抽象度レベル当たりに占める割合（エラーバーはデータが正規分布に分布したと仮定した場合の 90 %信頼区間）

7　結論

本研究では，PP における両者の関心の違いを，両者の対話の中に現れる問題提起に着目して分析を行った．結果として，ドライバ・ナビゲータ間では問題提起の内容には役割による抽象度レベルの差は認められず，ドライバとナビゲータ間の抽象度レベルによる関心の違いは実際には無いという，既存研究と同様の結果が得られた．しかし，得られた結果から，更に条件を変えた場合の検証が必要であることが分かった．

今後，我々が PP の効果を高めるためにどのように PP を管理・支援していけば良いのかを理解するためには，本研究のように PP における両者の相互作用の実際を解明していくことが更に求められる．

参考文献

1) Nancy Pennington, Adrienne Y. Lee, and Bob Rehder. Cognitive activities and levels of abstraction in procedural and object-oriented design. *Hum.-Comput. Interact.*, Vol. 10, No. 2, pp. 171–226, 1995.

2) R. Brooks. Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, Vol. 18, pp. 543–554, 1983.

3) A. Cockburn and Laurie Williams. The costs and benefits of pair programming. In G. Succi and M. Marchesi, editors, *Extreme programming Examined*, pp. 223–247. Addison-Wesley, Reeding, MA,, 2001.

4) Laurie Williams. Lessons learned from seven years of pair programming at north carolina state university. *Inroads: ACM SIGCSE Bulletin*, Vol. 39, No. 4, p. tbd, December 2007.

5) K. Beck. *Extreme programming explained: Embrace change*. Massachusetts, USA. Addison-Wesley, Reading, 1st 2000.

6) O. Hazzan and Y. Dubinsky. Bridging cognitive and social chasms in software development using extreme programming. In *Proceedings of the 4th International Conf. on Extreme Programming and Agile Processes in Software Engineering*, pp. 47–53, Genova, Italy, 2003.

7) L. Williams and R. Kessler. *Pair Programming Illuminated*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

8) E. A. Chaparro, A. Yuksel, P. Romero, and S. Bryant. Factors affecting the perceived effectiveness of pair programming in higher education. In *PPIG*, pp. 5–18, Brighton, UK, 2005.

9) S. Bryant, P. Romero, and B. du Boulay. The collaborative nature of pair programming. In *Extreme programming and agile processes in software engineering, volume 4044/2006 of Lecture Notes in Computer Science*, pp. 53–64. Springer, 2006.

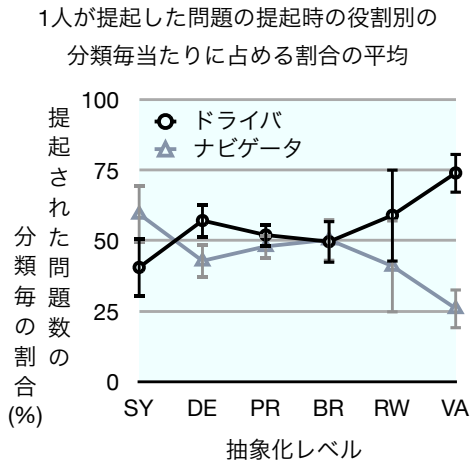

Fig. 7　実際に個人がドライバとナビゲータ時で提起した問題が 1 つの各抽象度レベル当たりに占める割合（エラーバーはデータが正規分布に分布したと仮定した場合の 90 %信頼区間）

10) J. Chong and T. Hurlbutt. The social dynamics of pair programming. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, Washington, DC, USA, 2007. IEEE Computer Society.

11) S. Bryant, P. Romero, and B. du Boulay. Pair programming and the mysterious role of the navigator. *Int. J. Hum.-Comput. Stud.*, Vol. 66, No. 7, pp. 519–529, 2008.

12) Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, Vol. 12, No. 3, pp. 175–204, 1986.

13) 山下洋一, 小磯花絵, 堀内靖雄. 音声対話に対する談話セグメントのタグ方式の検討. 人工知能学会誌, Vol. 14, No. 2, pp. 282–289, 1999.

14) Nancy Pennington. Comprehension strategies in programming. pp. 100–113, 1987.

A　付録

A.1　プログラミング課題１

Tick Tack Toe （○×ゲーム）ができるプログラムを作ってください。

引き分け、勝敗判定を行うこと。

☆実行例

```
---------------------------------------
S|S|S
-----
S|S|S
-----
S|S|S
A さん打ってください。
行番号＞ 2
列番号＞ 2
S|S|S
-----
S|A|S
-----
S|S|S
B さん打ってください。
行番号＞ 3
列番号＞ 3
S|S|S
-----
S|A|S
-----
S|S|B
A さん打ってください。
行番号＞ 1
列番号＞ 3
S|S|A
-----
S|A|S
-----
S|S|B
B さん打ってください。
行番号＞ 3
列番号＞ 2
S|S|A
-----
S|A|S
-----
S|B|B
A さん打ってください。
行番号＞ 3
列番号＞ 1
S|S|A
-----
S|A|S
-----
A|B|B
先手 A の勝ちです。
```

A.2　プログラミング課題２

下の実行例のように動く、自動販売機プログラムを作って下さい。

☆実行例

```
---------------------------------
＜＜取引開始＞＞
いらっしゃいませ。
商品リスト：
１．コーヒー（１２０円、在庫数：１０）
２．オレンジジュース（１３０円、在庫数：１０）
３．カルピス（１１０円、在庫数：１０）
４．コーラ（３０円、在庫数：１０）
５．サイダー（５０円、在庫数：１０）
６．メロンソーダ（１００円、在庫数：１０）
飲みたい飲料の番号を入力して下さい＞１
あなたが選んだ飲料は、
１．コーヒー（１２０円、在庫数：１０）
です。
投入する硬貨の枚数を入力してください。
５００円玉の枚数＞０
１００円玉の枚数＞１
５０円玉の枚数＞１
１０円玉の枚数＞０
あなたが入れた合計金額は、１５０円です。
おつりは、
１０円玉３枚です。
お買い上げ、ありがとうございました。
＜＜取引終了＞＞
＜＜取引開始＞＞
いらっしゃいませ。
商品リスト：
１．コーヒー（１２０円、在庫数：9）
２．オレンジジュース（１３０円、在庫数：１０）
３．カルピス（１１０円、在庫数：１０）
４．コーラ（３０円、在庫数：１０）
５．サイダー（５０円、在庫数：１０）
６．メロンソーダ（１００円、在庫数：１０）
飲みたい飲料の番号を入力して下さい＞
```