修 士 論 文

ペアプログラミングでの対話における話題の分析

北陸先端科学技術大学院大学
知識科学研究科 知識科学専攻

木村慎太郎

2010 年 3 月

修 士 論 文

ペアプログラミングでの対話における話題の分析

指導教員　國藤 進 教授

審査委員主査　國藤 進 教授
審査委員　西本一志 教授
審査委員　藤波 努 准教授
審査委員　由井薗隆也 准教授

北陸先端科学技術大学院大学
知識科学研究科 知識科学専攻

0850802 木村慎太郎

提出年月: 2010 年 2 月

概 要

ペアプログラミングには．ソフトウェア開発の生産性向上やプログラミング教育への教育的効果があることが知られている．ペアプログラミングにおいて，キーボード・マウスの操作を担当するプログラマはドライバ，もう一方のプログラマはナビゲータと呼ばれる．その役割間では，プログラミング中の思考の抽象度レベルに違いがあるとされ，ナビゲータはより抽象度レベルが高い領域で思考でき，ドライバは抽象度レベルが低い領域で思考しているという役割分担のモデルが一般的に認知されている．しかし近年，実際のプログラマを観察した諸研究において，そのような役割分担の特徴が発見されないことから，この既存の役割分担のモデルと実態が乖離していることが指摘されている．本研究では，仮に既存のモデルが正しい場合，ドライバ，ナビゲータ間での対話において双方から提起される問題の抽象度レベルに差が生じていると考え，双方のプログラマによるペアプログラミング中の対話に現れる問題提起に着目して，大学院生による１０回のペアプログラミングセッションの発話プロトコル分析を行った．結果からは双方から提起される問題には抽象度で差異は確認できず，既存の役割分担のモデルとは異なる結果が得られた．

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

コンピュータプログラミングは，一般的に高度に複雑な作業であるとされ，その対策としてのプログラミング方法論の 1 つとしてペアプログラミング（以下 PP）[Bec00] というプログラミングスタイルが提案されている．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結果的な効果として，ソフトウェア開発事業への導入による生産性の向上 [CW01] や，プログラミング教育への導入による教育的有効性の存在 [Wil07] が知られており，それらは統計的に実証されている．しかし，PP に参加する２者が実際にどのように相互作用しながらプログラミングを進めているのかを理解しようとする研究は未だ初期段階にあり，PP をどのように習熟させ，管理し，支援して行けばいいかを考える上で問題である．

PP における２者の振る舞いを表現したものとしては，Williams ら [WK02] による車の運転のメタファが最も一般的に浸透している．その中では “ドライバ” と “ナビゲータ” が引き合いに出され，両者が異なった役割を割り当てられていることを表している．さらに，ドライバ役のプログラマはキーボードとマウスをコントロールしながら主に実装することに関与し，もう一方のナビゲータ役のプログラマは “戦略的 (strategically)” に長期的視点から思考し，ドライバの実装が要求を満たしているかどうかを監視する役割を担うと述べられている．これは，ドライバがよりプログラミング言語に関する問題の解決に専念し，ナビゲータはより問題領域に近い問題の解決に専念することで，プログラミングにおける抽象度レベルにおいて両者は異なるレベルで思考を行っていると捉えることが出来る．

しかし，近年，実際に PP を実践しているプログラマを観察した研究 [BRdB08, CH07, Hö8] において，先に述べたような PP における両者の役割分担のモデルを特徴付けるような振る舞いが観測されないことから，これまで広く浸透していた PP においける両者の役割分担のモデルと実態が乖離していると指摘されている．

1.2 本研究の目的

本研究の目的は，Beck[Bec00] や Williams ら [WK02] によって提唱され，広く認知されている，ナビゲータ・ドライバの役割分担のモデルのように，２人のプログラマ間でキーボード・マウスの操作以外の役割分担が存在することを実証することである．

PP の生産性向上効果や教育的効果の一因は両者間で起こる議論にあるとされている [FH91]．議論により，１人でのプログラミングよりもより合理的な選択がなされやすくなり，また，2 者間での知識の共有がなされるからである．そこで，本研究では PP 内で “議論を起こすこと”，すなわち問題提起を，キーボード・マウスの操作を除いた場合のプログラマの PP 中の仕事として捉えた．

本研究では，問題提起の内容が，それを提起したプログラマの思考の傾向を反映したものであるとして，ナビゲータとドライバでは提起する問題の抽象度レベルに差異があり，ナビゲータはよりプログラミングにおける抽象度レベルが高い問題領域に近い問題の提起を行い，ドライバはより抽象度レベルが低いプログラミング言語領域に近い問題の提起を行っているのではないかという仮説を立てた上で，その実証を試みた．

1.3 本論文の構成

本論文は 6 章構成になっている．第 2 章では関連研究と本研究の位置づけについて述べる．第 3 章では仮説について述べ，第 4 章では仮説を実証するめの方法論について述べる．第 5 章では結果についてのべ，最後に第 6 章で結論と今後の課題について述べる．

第2章　関連研究と本研究の位置づけ

本章では，まずペアプログラミング (PP) に関する研究の動向を概観した後，本研究の位置づけを行う．

2.1　ペアプログラミングに関する研究動向

PP に関する研究は，PP そのものの効果を実証した研究，ペアの組み合わせの相性に関する研究，PP での両者の役割分担に関する研究の３つに大別できる．

2.1.1　PP の効果を実証した研究

PP は，Beck による “エクストリーム・プログラミング” [Bec00] と呼ばれるソフトウェア開発手法の中の１つの習慣として紹介され，一般的に認知されるようになっている．しかし，*The Mythical Man Month* の著者として知られる Brooks は，２人が一緒にプログラミングをすることがペアプログラミングと呼ばれるようになる以前の 1950 年代から既に PP を実践しており，その効果について体験していたと述べている [WK02]．さらに，90 年代に入ってから Constantine[Con95] や Coplien[Cop95] によりその効果が報告されている．だが，これらの報告での PP の効果はどれも逸話的な主張であった．

2000 年代に入り，ソフトウェアの生産性の向上効果，プログラミング教育における効果を実証する研究が成されている．

ソフトウェアの生産性向上効果

Williams[Wil00] はソフトウェアの生産性の向上効果を統計的に実証した．この研究では 1 人でプログラミングを行うソロプログラミング (SP) よりも PP の方が納期を 9 割から 7 割の時間に短縮でき，さらに品質についても，SP よりも PP の方が高い傾向にあり，かつ，ばらつきが少ないことが統計的に確認されている．

プログラミング教育への導入による教育的効果

さらに，Williams ら [WK00] は，大学のプログラミング演習で PP を導入することを提案し，後の諸研究 [WK01, MWBF02, WYW+02, Cli03, DeC03, MWBF03, NWF+03, WWP+03, HMDK04, MAFLR05, XR05, Pre06] により様々な学年やコースで PP を実践した結果，学生の講義に対するモチベーション向上や学生同士の知識の伝達による学生のプログラミング技術の修習度の向上，ドロップアウト率の抑制という教育的効果があることが報告されている．

2.1.2 ペアの組み合わせの相性に関する研究

どのような性格や能力を備えたプログラマをペアにすればより効果を発揮できるかという研究もこれまで成されてきている．

Domino ら [DCHC03] は，PP における認知能力と対立処理スタイルの重要性を検査している．彼らは，Wonderlic Personal Test (WPT) により認知能力を，Rahim Organizational Conflict Inventory (ROCI-II) により対立処理能力を計測した 1 4 人のプログラミング経験のある社会人学生に PP を行わせたが，ペアの認知能力と対立処理スタイルには統計的に有意な相関は認められなかった．Chao ら [CA06] は，58 人の学部生に PP を行わせ，相性に影響を与える性格特性の同定を試みたが，こちらも統計的に有意な結果は得られていない．Katira[KWW+04] らは，564 人の学部 1 年生，学部生，そして大学院生で PP の適合性を検査している．学生の相手のプログラミング習熟度の認知度と相手との適合性に正の相関があること，さらに，学部 1 年生では，Mayer-Briggs 性格型が異なる相手との PP

がより適していることが発見されている．Sfetsos ら [SSAD06] は，Keirsey 気質の異なる学生からなる 22 組のペアと，Keirsey 気質が同じ学生からなる 20 組のペアのパフォーマンスの比較を行っている．この研究により，異なる気質同士のペアのほうがより多くのコミュニケーションを取り，作業完了までの時間も短いことが分かっている．

2.1.3 PP での両者の役割分担に関する研究

PP における両者の役割分担のモデルは，Beck や，Williams と Kessler によるものが一般的に広く認知されているが，しかし，近年，実際に PP を実践しているプログラマを観察した研究 [BRdB08, CH07, HÖ8] において，先に述べたような PP における両者の役割の違いを特徴付けるような振る舞いが観測されないことから，これまで広く浸透していた PP においける両者の相互作用のモデルと，実態との乖離が存在が指摘されている．

一般的に認知されている役割分担のモデル

PP が一般的に知られるきっかけになった Beck による “エクストリーム・プログラミング (XP)” の教科書 [Bec00] では，PP を XP での 12 個の習慣の中の 1 つとして提案しており，PP 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?

Beck による説明では “ドライバ”，“ナビゲータ” という言葉は用いられていないが，このように，キーボードを操作するプログラマは実装の詳細を気遣い，もう一方のプログラマ

はより広い視野で，戦略的な問題について考えているとされており，２人のプログラマが異なる抽象度レベルで考えていると確かに述べられている．この Beck によるペアプログラミングの定義の影響は他の文献にも見られ，Hazaan らによる XP に関する文献 [HD03] 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，同様に記されている．

最も広く引用されている Williams と Kessler による PP における両者の役割の定義 [WK02] では，“ドライバ” と “ナビゲータ” という用語を用いて，

> 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，この定義では，ナビゲータはいくらか範囲が広げられてはいるが，ナビゲータは戦略的に長期的視点から考えるというように，Beck による定義と概ね同様に記されている．

PP の現場の実態を観察した研究

近年まで，PP におけるペア間の自然に発生する相互作用に焦点をあてた研究はほとんどなされておらず，上にあげたナビゲータとドライバの役割分担のモデルは疑われてこなかった．しかし，実際に PP を実践しているプログラマを観察した諸研究 [BRdB06,

BRdB08, CH07] において，先に述べたような PP における両者の役割の違いを特徴付けるような振る舞いが観測されないことから，それまで一般的に認知されていた PP における両者の相互作用のモデルと，実態との乖離が存在が指摘されている．

Chaparro ら [CYRB05] は，オブジェクト指向プログラミングの講義に PP を取り入れる試みを行った中で，ドライバとナビゲータの役割分担が学生の PP では確認できなかったと述べている．この研究の著者の 1 人だった Bryant ら [BRdB06] は、学生ではないプロのプログラマの 3 6 回の PP セッションを観察し，両者の発話を協調度の指標とした上で，作業の種類（コーディング，テスト，デバッグなど）において，どの程度協調的に作業が進められているかを分析している．結果から，どの種類の作業においても高度に協調的に行われていることが分かり，ドライバとナビゲータの役割の違いによる各作業への寄与の程度の差は認められなかった．Bryant らによって続いて行われた研究 [BRdB08] では，同様に 24 回の PP のセッションを観察し，両者の発話内容の抽象度の分析が行われたが，役割による違いは認められていない．彼らは，両者とも同じ抽象度で相互作用しているとして，ドライバ，ナビゲータに替わる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．Chong と Hurlbut[CH07] もドライバとナビゲータの役割の存在に懐疑的で，４ヶ月間の２つの会社における開発チームを観察した結果から，ペアの両方のプログラマはほとんど同じ抽象度レベルで議論しているため，一般的なドライバとナビゲータの役割の記述と食い違っていると述べている．

2.2 本研究の位置づけ

本研究のように，ドライバ・ナビゲータの役割の違いに着目して実際の PP に加わったプログラマの発話内容を分析した研究として，前節で述べた Bryant らによる研究 [BRdB08] と Chong と Hurlbut による研究 [CH07] が挙げられる．

Bryant ら [BRdB08] は，PP 内で現れる両者の発話プロトコル分析を行い，それぞれの発話の抽象度レベルに違いが認められないことがドライバ・ナビゲータという既存モデルと実態との乖離の根拠としている．

Chong と Hurbut[CH07] は PP 内で現れる議論での両者の発話内容を比較し，２人のプ

ログラマは問題について同じ戦略的な “range”や抽象度レベルで議論し考えていると主張している．

これら２つの研究では，議論の中での全ての両者の発話内容の比較を行っている．しかし，議論での発話は発話思考法のような内省的発話とは異なり，主体の認知過程を反映したものであるとは言えない．また，一方の働きかけにたいして，その応答内容が働きかけの内容と同じ抽象化レベルになることは自然なことであるとも考えられる．

本研究はその議論の対象となる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は異なる．

第3章　仮説

2.1.3 節で一般的に認知されている役割分担のモデルとして挙げたた Beck[Bec00] と Williams・Kessler[WK02] によるドライバとナビゲータの役割のモデルでは，ナビゲータはドライバより抽象度が高いレベルで思考を行っているとしている．

プログラミングにおける “抽象度のレベル (levels of abstruction)” の概念の定義は文献によって異なるが，“領域 (domain)” の連なりであるとされているものもがある．例えば，Brooks[Bro83] は 5 つの領域（問題領域，アルゴリズム領域，プログラミング言語領域，識別子領域，実行領域）を提案し，Pennington は 詳細領域（演算子や変数），プログラム領域（ルーチンやファイル）そして現実世界領域を提案している．Bergantz と Hassel[BH91] は，プログラミングを機能の抽象レベルの階層モデルとして論じている．このように，抽象度のレベルの違いを領域の違いやプログラムの粒度の違いとして捉えられている．さらに，両方のレベルを組み合わせると，最も抽象度が低いレベルをプログラムの文法や綴りに関する領域，最も高いレベルを問題領域であるとすることができる．

そうした場合，ナビゲータは，より抽象度レベルの高い思考をするため，主に現実世界や問題領域に関わる問題を提起し，ドライバは主に，より抽象度の低い変数や演算子，プログラミング言語の文法や綴りに関する問題を提起することで，双方の提起する問題の分布は図 3.1 に示すようになると考えられ，これが実証された場合，キーボード・マウスの操作以外での双方の役割分担が存在することが確認できる．

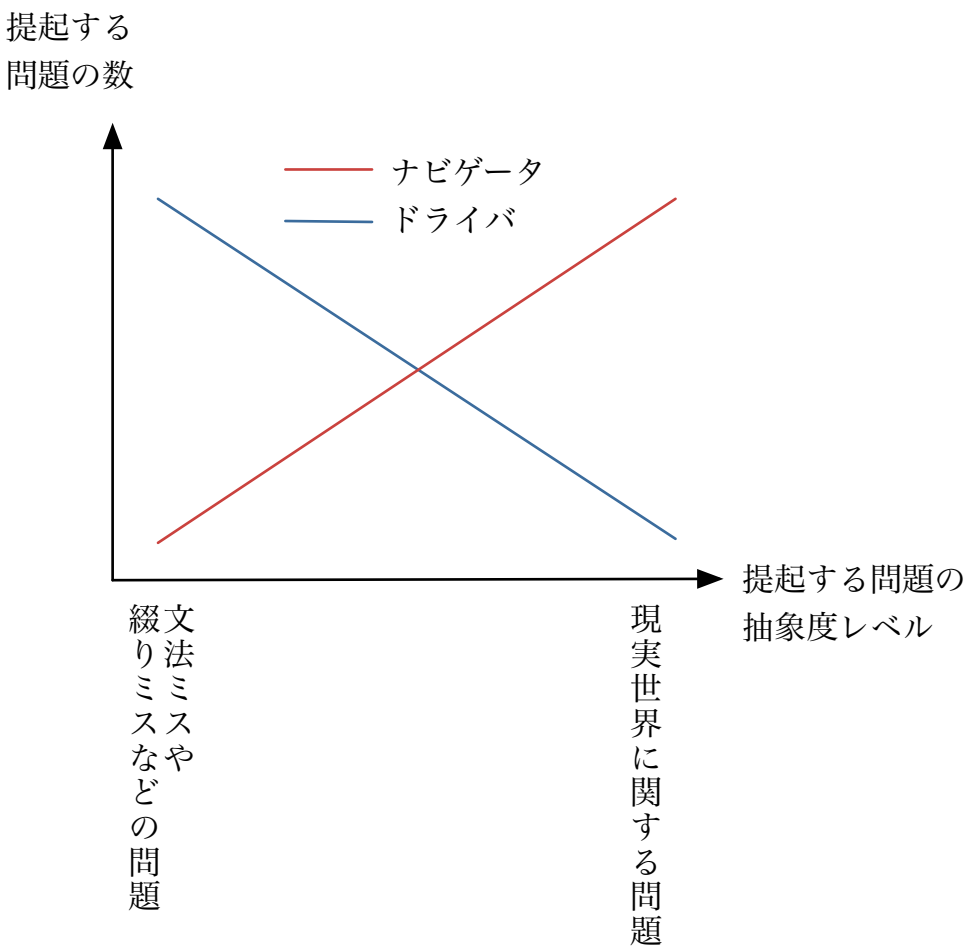

図 3.1: ドライバ，ナビゲータが提起した問題を抽象度レベルにより分類した場合の分布

第4章　方法論

4.1　データ収集方法

本研究では１セッション９０分の PP を，プログラミング能力が同等レベルの学生から成るペア５組に行ってもらい，２セッションづつ計１０セッションを分析対象として，両者の発話内容と，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するため，それぞれのセッションの収録を行った．

4.1.1　被験者

Java によるプログラミング経験が３年以上あ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．被験者にはあらかじめ Java プログラミング能力検定[1]を受験し，同等レベルであることを確認している．

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 [KWW+04]．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．

[1] Java に関する基本知識を有し、オブジェクト指向に基づくアプレットやアプリケーションプログラムを作成できる能力を認定することを目的とした検定試験で，サーティファイ情報処理能力認定委員会が主催・認定している．

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図4.1に示す位置にカメラを固定し，図4.2に示すような映像を録画した．被験者の発話は，図4.2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．

4.1.3 タスク

被験者には，実際にPPをする前に，PPのやり方・効果を教授した．その際，Williamsらによるドライバとナビゲータという役割分担のモデル[WK02]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4.1に示す．

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．

図 4.1: カメラの配置位置

4.2 データ分析方法

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．

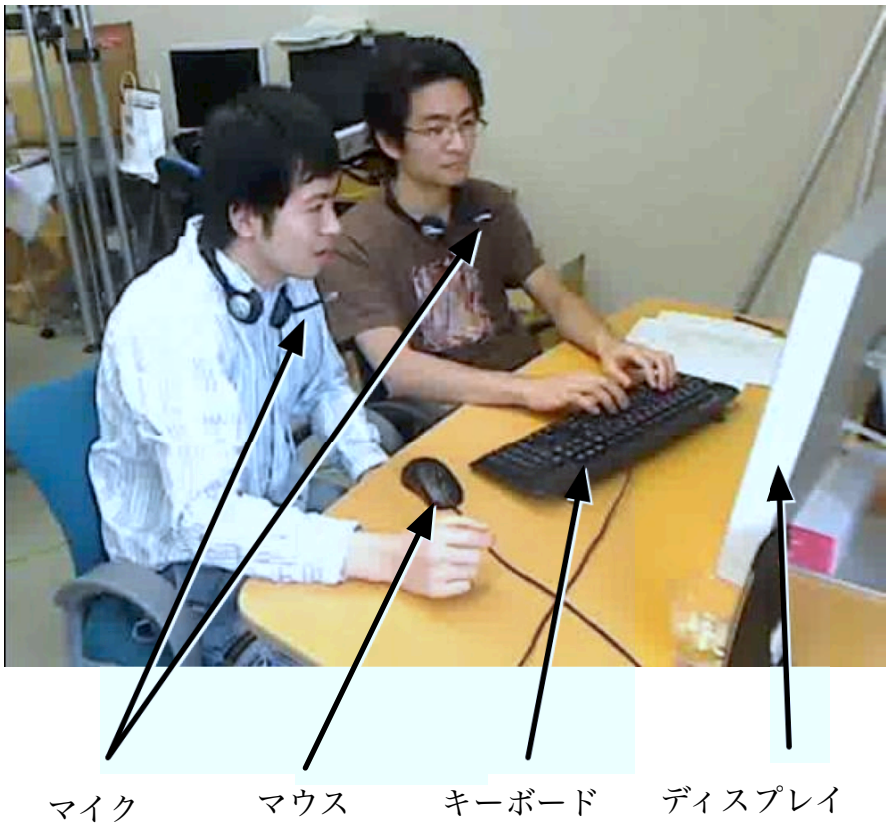

図 4.2: カメラで録画した映像の 1 場面

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 [GS86]。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 [山下 99] 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．その手順の要点を以下に示す．

表 4.1: 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

（1）書き起こし資料の作成

収録した音声を発話単位ごとに書き起こしていく．その際，その発話の話者と，映像を見ながらその話者がドライバ・ナビゲータどちらの役を担っているのかを記録する．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．

（2）やりとり構造の分析

対話に特有な「やりとり (exchange)」には，以下の基本構造を持つことが知られている [荒木 99]．

やりとり　→　働きかけ 応答＆働きかけ* 応答（了解）（了解）

ここで，*は 0 回以上の繰り返し，() は省略可能であることを示している．「働きかけ」はやりとりを開始する発話である．「応答」は，働きかけに対する行程／否定や受諾などの応答を行う発話である．「応答＆働きかけ」は応答の機能を持った働きかけの発話である．「了解」は相手の会うとうを理解した状況を示す発話である．手順（1）で作成した書き起こし資料の発話単位ごとに働きかけ，応答，応答＆働きかけ，了解にそれぞれ対応する [I]，[R]，[RI]，[F] のタグを付与する．

（3）対話セグメント開始境界の同定

分析者は，収録された音声とそれに連動する画面キャプチャ動画を見ながら，対話セグメントが開始されていると思われる発話に対して，対話セグメント開始タグ [S] を付与する．その際,「働きかけ」および「応答＆働きかけ」の発話で対話セグメント

が開始されるとして，対話セグメント開始タグは手順（2）で付与した [I] か [RI] のタグが付与されている発話のみに限る．今回の実験では 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．

以上の手順によりタグ付けされた書き起こし資料の一部を表 4.2 に示す．対話セグメントの開始を表す S が記されている発話を問題提起の発話とする．

4.3 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，Pennington[Pen87] による詳細領域，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，意味論を含まない，プログラミング言語の文法や識別子の綴りの領域を，最も抽象度レベルが低い領域として加えた．そして，問題領域とプログラム領域の結びつけを行う領域を現実世界領域とプログラム領域の間に加えた．さらに，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，合計 6 つの話題に分類し表 4.3 にしめす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，表 4.2 に示した書き起こし資料の部分に話題の分類のためのタグ付けの結果を 4.4 に示す．

表 4.2: タグ付けされた書き起こし資料の一部

連番	話[a]者	役[b]割	構造[c]タグ	開始[d]タグ	発話内容
267	F	D	I	S	ここは int でいいかな。
268	K	N	RI		（うーん）[e]もしくは。boolean か。
269	K	D	RI		0 にしとくとあとあといいとおもうんだ。
270	F	N	R		（うーん）そうだね。
271	K	N	I	S	初期化はさ、【うん】[f]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	R		あそっか。
273	K	N	I	S	プレーヤーはぜったい二人だよね。
274	F	D	R		そうだね。
275	K	N	I		だったら、結局、private の enum ですよね。
276	F	D	R		うん。
277	K	N	I	S	ここにセミコロンが無いよ。
278	F	D	R		ほんまや
279	F	D	I	S	あれ、なんでや。
280	K	N	RI		ここ括弧でくくるんじゃないの。
281	F	D	RI		まいいや 0 じゃなくても。
282	K	N	R		(まー）そうだね。
283	F	D	I	S	で、これでとりあえず終わりじゃない。
284	K	N	R		うん。
285	F	D	I	S	これ、配列作った方がよくね。
286	K	N	RI		（んま、）for 文使うときにいいよね。
287	F	D	R		そうだね。

[a]話者を識別する記号（名前の頭文字）
[b]ドライバの場合は D，ナビゲータの場合は N
[c]やりとり構造を示すタグ
[d]対話セグメント開始タグ
[e]（）はフィラーを表す.
[f]【】は相づちを表す.

表 4.3: 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	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	プログラムの構造	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

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，図 4.3 の通りの結果が得られると考えられる.

表 4.4: 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[a]タグ	開始[b]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。

[a]表 4.3 で定義した対話セグメントでの話題を表すタグ
[b]対話セグメント開始タグ

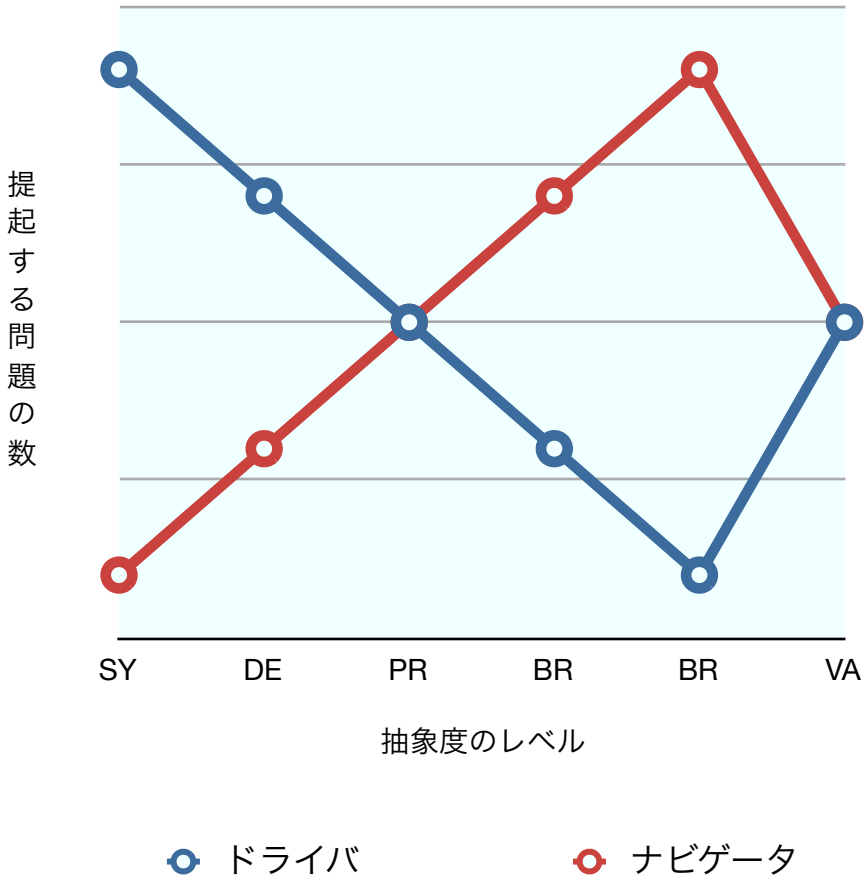

図 4.3: ドライバとナビゲータが異なる抽象度レベルで思考していた場合に提起される問題の抽象度レベルの分布

第5章　結果

分析対象となった10セッションの内総発話数は6168，議論数は984であり，1セッションあたりの両者の発話数は平均約617，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どのセッションも0.66以上あり，十分高い一致率が得られ，信頼できる分析結果が得られた．

単純にドライバ，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図5.1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図5.2に示す．こちらも，前章で示したような分布の特徴は確認されなかった．比較的差異が確認できる分類はDEとVAである．DEは詳細領域を表し，VAはプログラミングにおける抽象度レベルに属さない抽象度が曖昧な問題提起の分類を示している．

ペアの組ごとに比較したもの図5.3に示す．こちらも，5組中D組を除いた4組でDEとVAに統一的な傾向が見られた．

DEに分類された問題提起の内容を見てみると，ドライバによる「このメソッド名はどうしようか.」,「この変数名は○○でいいか.」,「この変数の型は○○の方がいいか.」などの，変数名，メソッド名，変数の型の選択をナビゲータに催促，あるいは確認している発話が目立った．Vに分類された問題提起の内容を見てみると，ドライバによる「じゃ，次はこのメソッド作るかな.」などの自らが今から何を取り組もうとしているのかを表したものや,「このメソッドはこれで完成でいいかな」という終了した作業を終了することをナビゲータへ確認する発話が多く散見された．ドライバはナビゲータに比べて変数名やメソッド名，変数の型やメソッドの返値の型など，型や名前に着目しながらプログラミングを行っていることが分かる．

図 5.1 を見ると，ドライバとナビゲータは双方とも同様に同じ抽象度のレベルで問題提起を行っているかのように見えるが，図 5.3 の割合の比較からは，組ごとの傾向にかなりばらつきがあることが確認でき，ドライバ，ナビゲータという役割により思考の抽象度のレベルが異なるというよりも，ペアによる差，あるいは個人による差が大きいと言える．

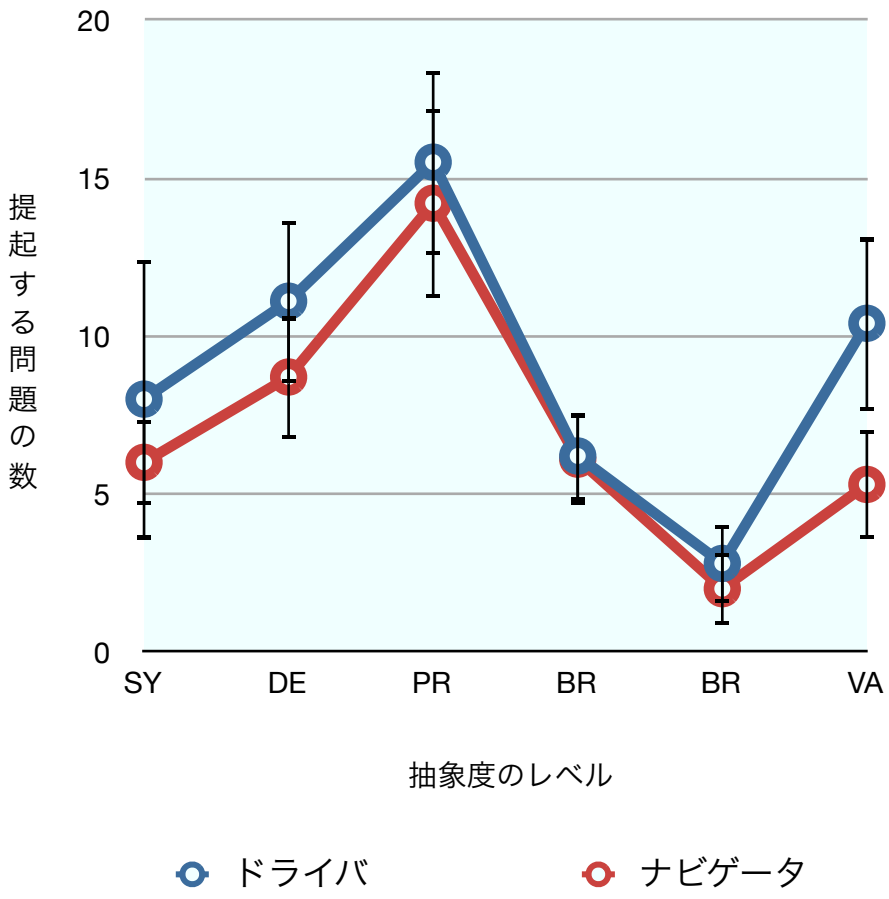

図 5.1: 実際にドライバとナビゲータそれぞれが提起した問題の抽象度レベルによる分布（エラーバーはデータが正規分布に分布したと仮定した場合の 90 %信頼区間を示す.）

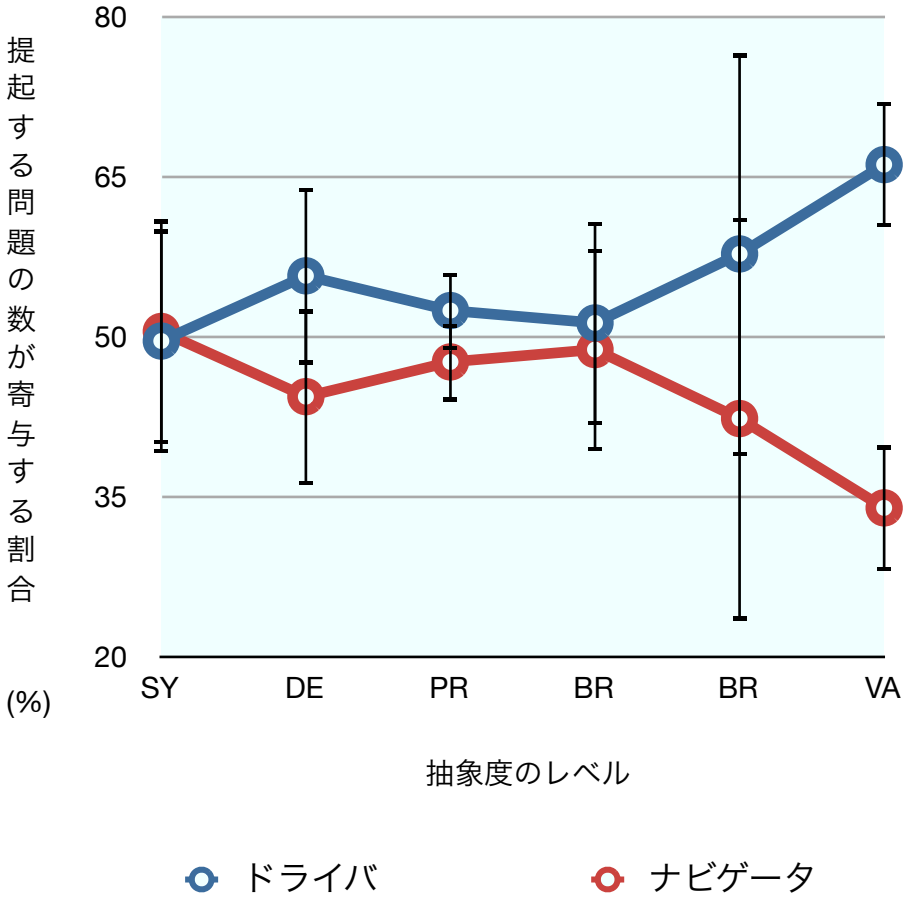

図 5.2: 実際にドライバとナビゲータそれぞれが提起した問題が 1 つの各抽象度レベル当たりに占める割合（エラーバーはデータが正規分布に分布したと仮定した場合の 90 %信頼区間を示す.）

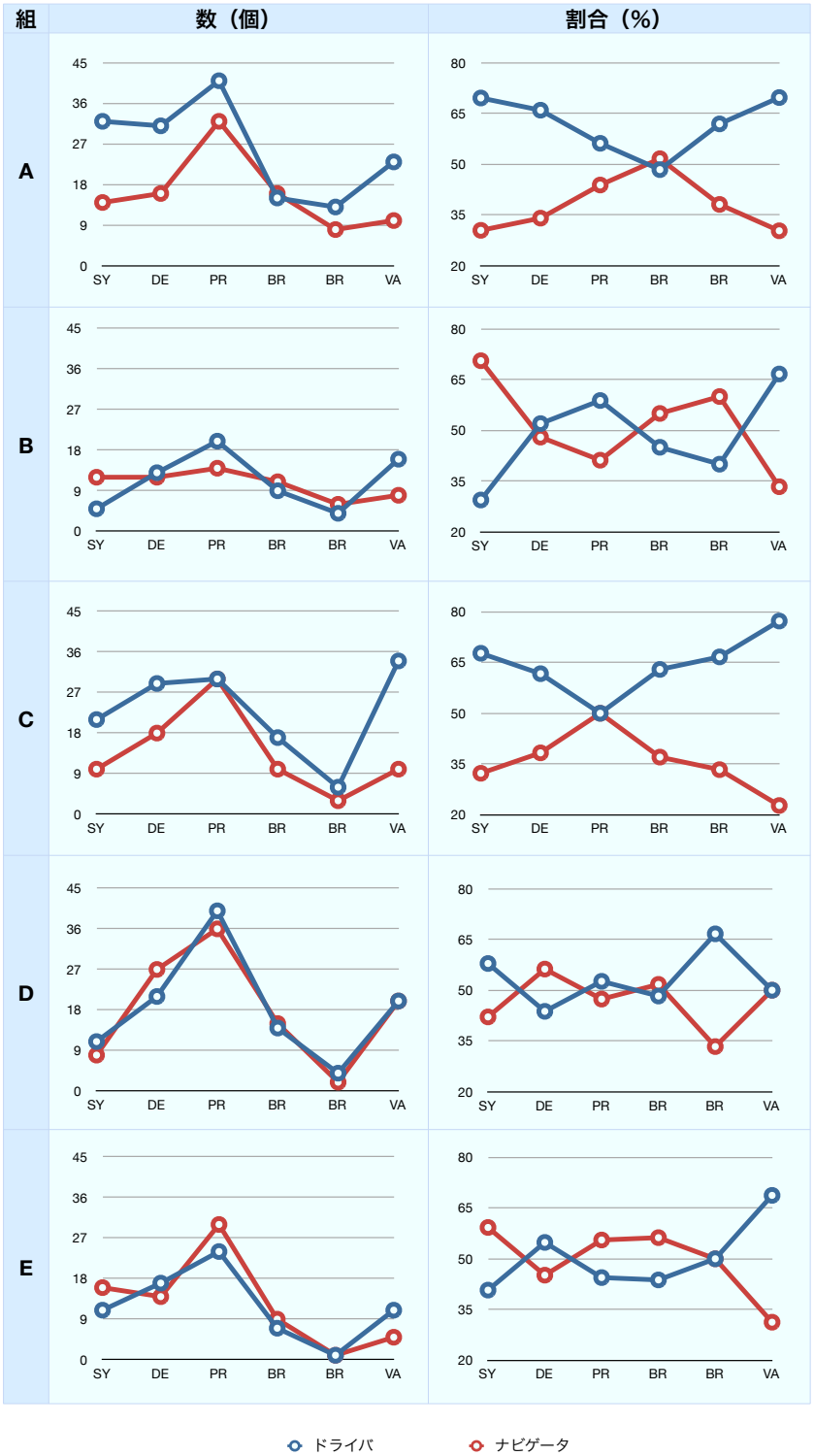

図 5.3: 個別の組ごとの比較．右の列は提起した問題が 1 つの各抽象度レベル当たりの数，左の列は 1 つの各抽象度レベル当たりに占める割合

第6章　おわりに

本研究では，PP における両者の役割分担がどのように行われているのかを，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異は認められず，Beck[Bec00] や Williams ら [WK02] によるドライバとナビゲータの役割分担のモデルは実証できなかった．

今回は，PP の経験があまり無い大学院生による PP セッションを分析したが，ソフトウェア開発の現場で日頃から実際に PP を実践しているプログラマらによる PP では異なる結果が得られる可能性がある．

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，更に 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求められる．

謝辞

本研究を遂行するにあたり，お世話になった方々へ感謝を述べさせていただきます．

指導教官である北陸先端科学技術大学院大学知識科学研究科の國藤進教授には研究に関するさまざまなご指導，ご鞭撻を賜りました．心より感謝いたします．また研究環境をはじめ，研究生活全般においてさまざまな支援をしていただいたことを深く感謝いたします．

羽山助教には研究面で多大なるご指導，ご鞭撻をいただいたことを心深く感謝いたします．

審査員である西本一志教授，藤波努准教授，由井薗隆也准教授には研究に関するさまざまな助言をいただき心より感謝いたします．

國藤研究室の皆様には常日頃から研究に関する助言，議論を頂きました．研究活動以外の面においても，大変お世話を頂いたことを心より感謝いたします．

被験者，ならびに分析者としてお忙しい中実験にお付き合い下さった皆様に心より感謝いたします．

最後に，金銭面，精神面において大きな支えとなってくれた両親に心より感謝します．本当にありがとうございました．

参考文献

[Bec00] K. Beck. *Extreme programming explained: Embrace change*. Massachusetts, USA. Addison-Wesley, Reading, 1st 2000.

[BH91] David Bergantz and Johnette Hassell. Information relationships in prolog programs: how do programmers comprehend functionality? *Int. J. Man-Mach. Stud.*, Vol. 35, No. 3, pp. 313–328, 1991.

[BRdB06] 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.

[BRdB08] 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.

[Bro83] R Brooks. Towards a theory of the comprehension of computer programs. *Int. J. Hum.-Comput. Stud.*, Vol. 18, No. 6, pp. 543–554, 1983.

[CA06] J. Chao and G. Atli. Critical personality traits in successful pair programming. In *Agile Conference, 2006*, pp. 5 pp.–93, July 2006.

[CH07] 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.

[Cli03] Daniel C. Cliburn. Experiences with pair programming at a small college. *Journal of Computing Sciences in Colleges*, Vol. 19, No. 1, pp. 20–29, 2003.

[Con95] L. L. Constantine. *Constantine on Peopleware*. Youdon Press, 1995.

[Cop95] J. O. Coplien. A development process generative pattern language. In James O. Coplien and Douglas C. Schmidt, editors, *Pattern Language of Program Design*, pp. 183–237. Addison-Wesley, 1995.

[CW01] 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.

[CYRB05] 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.

[DCHC03] Madeline Ann Domino, Rosann Webb Collins, Alan R. Hevner, and Cynthia F. Cohen. Conflict in collaborative software development. In *SIGMIS CPR '03: Proceedings of the 2003 SIGMIS conference on Computer personnel research*, pp. 44–51, New York, NY, USA, 2003. ACM.

[DeC03] Timothy H. DeClue. Pair programming and pair trading: effects on learning an motivation in a cs2 course. *Journal of Computer Sciences in Colleges*, Vol. 18, No. 5, pp. 49–56, 2003.

[FH91] N. V. Flor and E. L. Hutchins. Analyzing distributed cognition in software teams: a case study of team programming during perfective software maintenance. In *Empirical Studies of Programmers: Fourth Workshop*, pp. 36–63. 1991.

[GS86] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, Vol. 12, No. 3, pp. 175–204, 1986.

[HÖ8] A. Höfer. Video analysis of pair programming. In *APOS '08: Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral*, No. 37–41, New York, NY, USA, 2008. ACM.

[HD03] 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.

[HMDK04] Brian Hanks, Charlie McDowell, David Draper, and Milovan Krnjajic. Program quality with pair programming in cs1. *ACM SIGCSE Bulletin*, Vol. 36, No. 3, pp. 176–180, 2004.

[KWW+04] Neha Katira, Laurie Williams, Eric Wiebe, Carol Miller, Suzanne Balik, and Ed Gehringer. On understanding compatibility of student pair programmers. *SIGCSE Bull.*, Vol. 36, No. 1, pp. 7–11, 2004.

[MAFLR05] Emilia Mendes, Lubna Basil Al-Fakhri, and Andrew Luxton-Reilly. Investigating pair-programming in a 2nd-year software development and design computer science course. In *the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pp. 296–300, 2005.

[MWBF02] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. The effects of pair-programming on performance in an introductory programming course. In *the 33rd SIGCSE technical symposium on computer science education*, pp. 38–42, 2002.

[MWBF03] Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald. The impacto of pair-programming on student performance, perceptions and persistence. In *International Conference on Software Engineering (ICSE '03)*, pp. 602–607, 2003.

[NWF+03] Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, and Suzanne Balik. Improving the cs1 experience with pair programming. In *ACM Special Interest Group Computer Science Education (SIGCSE) 2003*, pp. 359–362, Reno, 2003.

[Pen87] Nancy Pennington. Comprehension strategies in programming. pp. 100–113, 1987.

[Pre06] David Preston. Using collaborative learning research to enhance pair programming pedagogy. *ACM SIGITE Newsletter*, Vol. 3, No. 1, pp. 16–21, January 2006.

[SSAD06] Panagiotis Sfetsos, Ioannis Stamelos, Lefteris Angelis, and Ignatios Deligiannis. Investigating the impact of personality types on communication and collaboration-viability in pair programming – an empirical study. In *Extreme Programming and Agile Processes in Software Engineering*, pp. 43–52. 2006.

[Wil00] L. A. Williams. *The Collaborative Software Pocess*. PhD thesis, University of Utah, 2000.

[Wil07] 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.

[WK00] Laurie A. Williams and Robert R. Kessler. The effects of "pair-pressure" and "pair-learning" on software engineering education. In *Conference of Software Engineering Education and Training 2000*, 2000.

[WK01] Laurie A. Williams and Robert R. Kessler. Experimenting with industry's "pair-programming" model in the compurter science classroom. *Journal on Computer Science Educations*, 2001.

[WK02] L. Williams and R. Kessler. *Pair Programming Illuminated*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[WWP+03] Eric N. Wiebe, Laurie Williams, Julie Petlick, Nagappan Nachiappan, Suzanne Balik, Carol Miller, and Miriam Ferzli. Pair programming in intoroductory programming labs. In *American Society for Engineering Education (ASEE) 2003*, 2003.

[WYW+02] Laurie Williams, Kai Yang, Eric Wiebe, Miriam Ferzli, and Carol miller. Pair programming in an introductory computer science course: Initial results and recommendations. In *Paper presented at the 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002). Seattle, WA, November408*, 2002.

[XR05] Shaochun Xu and Václav Rajlich. Pair programming in graduate software engineering course projects. In *Frontiers in Education (FIE 2005)*, pp. F1G 7–12, 2005.

[荒木 99] 荒木雅弘, 伊藤敏彦, 熊谷智子, 石崎雅人. 発話単位タグ標準化案の作成. 人工知能学会誌, Vol. 14, No. 2, pp. 251–260, 1999.

[山下 99] 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.

付 録A　プログラミング課題

A.1　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　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＞