RUP®/XP ガイドライン: ペア・プログラミング

Robert C. Martin
Object Mentor, Inc.

Rational Software ホワイト・ペーパー

TP 158, 3/01

Rational®
the software development company

目次

概説

「ペア」について

ペア・プログラミングは、プロジェクトのソフトウェアをプログラマーのペアで作成する技術です。各ペアは、1 台のワークステーションで一緒に作業します。ペアの一方のメンバーがワークステーションで作業しているときに、もう一方のメンバーは生成されるコードを注意深く観察します。コード作成者は、現在作成しているコードを戦術的に考えます。観察者は構文を検証しながら、プログラム全体を戦略的に考えます。これらのロールを、お互いに何度も交代して行います。その結果、1 人で作業した場合よりもコードの作成時間は短くなり、障害も少なくなります。また、作成したコードについて少なくとも 2 人の開発者が熟知していることになります。

ペアの効果

一般的なコードのレビュー・セッションを考えます。1 人の開発者が作業に 8 時間かかったモジュールを、8 人で 1 時間かけてレビューするとします。結果として、このモジュールには 16 人時かかります。ただし、レビュー担当者がコードを詳しく理解するほどの時間はなく、実際には表面的なレビューになります。開発者 1 人で作業をする場合には、コードに関しては熟知していますが、そのためにかえって障害の大部分を見つけられない可能性があります。

この事例を、ペア・プログラミングの場合と比較します。モジュールの開発にペアで 8 時間かかる場合、合計で 16 人時が必要です。ただし、この場合は 2 人 の開発者がコードに対して詳細な知識を獲得します。一方の開発者が気付かない障害に、もう一方の開発者が気付く可能性があります。

ペア・プログラミング自体は単純な方法ですが、その効果は複雑で広範囲にわたります。ペア・プログラミングは、コードを作成してレビューする効果的な方法です。2 人の開発者がモジュールをよく理解し、コード中の障害もほとんどありません。コードはより適切な構造で作成され、コードに関する知識も 2 倍の頭脳に蓄積されます。これだけでもペア・プログラミングの利点として十分ですが、実際にはさらに多くの利点があります。

ペアを組むことで勇敢になります。プログラマー 1 人では躊躇するようなことでも、ペアを組むと、これを試みる勇気と評価する技術が得られます。

ペアを組むことでチームワークが育成されます。モジュールの作成は複数の開発者によって行われるため、コードは特定の開発者ではなくチームの資産となります。

ペアを組むことで知識が広がります。ペアを組む開発者が多くなるほど、システムに関するより多くの知識がチーム全体に浸透します。結果的に、チームのメンバーはシステムの特定の部分だけでなく、システム全体を理解するようになります。

ペアを組むことで生産性が向上します。1 人でプログラミングを行う開発者は、開発の速度が一定しません。ペアを組むと、お互いにペースを調整します。ペアの一方が疲れたら、ロールを交代します。1 人の人間が通常集中力を維持できる時間よりも長い時間、集中力を維持することができます。

ペアを組むことに楽しみがあります。ほかの開発者と作業することは、教育的で刺激的でもありますが、単純に楽しいことです。ペアを組むことで、仕事に対する満足度と総合的な士気が向上します。

実践原則

ペアを組む

ペアを組むことは、タスクを担当する開発者が、誰かに援助を求めたときから始まります。ルールは、依頼されたら必ず「はい」と答えることです。これは、自分の作業を直ちに中断しなければならないということではありません。自分が援助できる時間と、その埋め合わせに他の人から援助を受けられる時間との調整を行う必要があるということです。

ペア・パートナーが、タスクに関する責任を引き受けることはありません。タスクに関する責任は、タスクの所有者が保持します。また、ペア・パートナーは、タスクが完了するまでタスクの所有者ともに作業し続けなくてはならないということでもありません。ペア・パートナーは、援助するだけです。

ペアの一方のメンバーがコードの作成担当者となり、もう一方のメンバーがそれを観察します。コードの作成担当者は、コードの入力、コンパイラーの実行、単体テストの実行などを行います。観察担当者は、キー・ストローク、コマンド、テスト結果のそれぞれを確認し、援助と提案を行います。両者は常に一緒に作業します。

作成担当者が最適な方法を知っていて、観察担当者はただそれを見るだけという場合もあります。また、観察担当者が作成担当者に指示を与える場合もあります。作成担当者が行き詰って、観察担当者にキーボードを渡すこともあります。この場合は、ロールを交代します。また、観察担当者の方からキーボードを渡すように指示され、ロールを交代する場合もあります。これらのことは、ペアのセッション中に何度も起こります。

ペアの変更

パートナーとペアを組む期間は長くありません。ペアによる一般的なセッションの期間は、およそ半日です。どちらのパートナーも、理由を問わずペアを解消することができます。この場合、タスクの所有者は別のパートナーを探す必要があります。タスクの所有者は、ここで先週からペアを組んでいたパートナーに援助のお返しをする場合もあります。一方で、特に難しい問題に対して、適切な経験を持つ人物に援助を求めることになります。

このようなペアの変更により、開発チーム全体にシステムに関する知識が広まります。短期間に、チームのすべてのメンバーが、システムのほとんどすべての部分に対して作業を行います。これにより、プロジェクトの仕様変更に対して柔軟に対応できるようになり、すべてのプログラマーはシステム全体に対して自信を持って対処できるようになります。

共同所有

すべてのメンバーがシステム中のあらゆるモジュールに取り組むため、各メンバーが特定のモジュールを所有することはありません。つまり、システムに対する責任はモジュール単位で分割されません。システム全体をひとまとめにして、チーム全体で責任を持ちます。チームのメンバーは、システムの任意のモジュールをチェックアウトして変更できます。あるペアがモジュール X に対して変更を行い、この変更によってモジュール Y の単体テストが失敗した場合は、このペアがモジュール Y を修正します。

ペースと協調

ペア・プログラミングでは、コミュニケーションが非常に活発になります。会話は白熱した議論となることもよくあり、外部の人間がその意味を理解するのが困難な場合もあります。観察者となった場合には、ペアが「セミコロン」や「閉じ括弧」などの独特の単語を発するのが聞こえるかもしれません。また、プログラマーが画面に表示されたものに対する評価を小声で話すのが聞こえる場合もあります。表示されているコードに集中すると、コミュニケーションの大半が言葉以外で行われるようになります。この場合、身振りや手振りが重要な役割を果たすようになります。パートナーのコードに対して不満がある場合、言葉をしゃべらなくてもそれを伝えることはできます。しかめっ面、ため息、いらいらした態度、これらはすべてパートナー間のコミュニケーションを広げるものです。

ときには、パートナーの一方がマウスを、もう一方がキーボードを操作することもあります。マウスの担当者は、モジュール内の作業する場所を指示します。キーボードの担当者は、その場所に変更または追加する内容を入力します。また、一方がコードを作成している間に、もう一方のパートナーは使用される関数呼び出しを予測して、パートナーが必要とする仕様に関する API 文書の該当ページを開いておくこともあるでしょう。

一方のパートナーが疲れたら、もう一方のパートナーが主導権を握り、パートナーを観察者にして休憩を取らせることもできます。パートナーの両方が元気であれば、キーボードとマウスを頻繁に交換することもあります。

要約すると、ルールはほとんどなく、手順もほとんどありません。現実的な唯一の制約は、パートナーはどちらも積極的に作業に関与して、お互いのコミュニケーションを活発にする必要があることだけです。一方がキーボードから入力している間にもう一方が窓の外を眺めているようでは、本当のペアではありません。

ペアを組んでいないときの作業

常にペアを組んでいることはできません。*エクストリーム・プログラミング* (XP) (参考資料 [1])・プロセスを採用するプロジェクトなど、一部のプロジェクトは、すべての製品コードはペアで生成しなければならないというルールに従います。この場合、ペアを組んでいないときは、電子メールのチェック、新しい技術や API の調査、理解の浅いコードの確認、現在の反復や

将来的な計画についての利害関係者との話し合いを行うことができます。実際に、ペア・パートナーが見つからない数時間の間に、開発者は有益な作業に気付きます。

プロジェクトによっては、ペアを組むことにあまり厳格でない場合もあります。1 人の開発者にテストの作成を許可する場合もあります。また、抽象クラスやインターフェースの作成を 1 人の開発者に許可することもあります。また、ペアを組む最適な時期を開発者に判断させるプロジェクトもあります。ただし研究では、ペアを組んだ場合に障害の割合が劇的に減少することが知られています。この点は明らかです。

ペアを好まない開発者の場合

開発者によっては、ペアの概念を苦痛と感じる場合もあります。経験的には、実際に 1 週間程度のペア・プログラミングを試してみると、開発者の不安は消えるようです。ペアを組むことに楽しみを感じて、その有効性に気付きます。その後も実践を嫌がる開発者はほとんどありません。ほとんどの開発者の場合、単にペア・プログラミングを試してみて、慣れることが問題です。ペア・プログラミングを正しく実践してみて、それでも嫌がる開発者の場合は、チームで適切な対策を考案する必要があります。

備品、設備、ロジスティクス

モニターとキーボードの配置

ペア・プログラミングにおいて、備品の配置は非常に重要です。パートナーが隣同士に座ってすみやかにキーボードを交換できなければ、うまくペアは組めません。ルールとして、席を替わることなく、キーボードとマウスをお互いに渡すことができる必要があります。

最適な配置にするには、長い平らな机を使用します。モニターを中央に配置して、2 つの椅子をモニターに向けて置きます。モニターが 2 人の間にあるように座ります。キーボードとマウスをお互いに滑らせて交換できることを確認します。また、一方がキーボードを使用しているときに、もう一方は快適に座ったままモニターに集中できることを確認します。モニターを回転させなくても、両者がモニターを見ることができるようにします。

作業空間

ペア・パートナーが簡単に交代できるように、作業用に準備された部屋で作業すれば便利な場合がよくあります。1 つの部屋に、ペア用の作業場所をいくつか用意します。車輪付きの椅子を用意して、床はリノリウムまたはタイル張りにします。ワークステーションは、各ペアがお互いに向き合うように配置します。ここでの目標は、コミュニケーションを活発にすることです。最も重要なコミュニケーションは、思いがけず見つかる場合があります。このようなコミュニケーションが行われる機会を増やします。

部屋の角

今日の一般的な作業スペースでは、ワークステーションは角に配置されます。開発者は部屋の角に向かい、モニターのすぐ前に座ります。このような配置は個人作業には適していますが、ペア・プログラミングを効率よく行うことはほとんど不可能です。ワークステーションを部屋の角に配置している場合は、ペア用の作業場所を会議室などの別の場所に設置してください。会議室のテーブルでラップトップを使用してペアを組むと、非常に効果的な場合があります。

問題点と考察

生産性

同じタスクについて 2 人で一緒に作業する場合、1 人で作業する場合の 2 倍の人時となるのは当然です。これは道理に合っているので、問題ではありません。さまざまな研究 (参考資料 [2]) で、ペアによる作業で生産性が失われることはほとんどないことが示されています。仕事の楽しみが大きく増加しているのに対して、障害の割合やコード・サイズが実質的に減少することが、同じ研究によって示されています。

ペア・パートナー間での議論

設計上の議論に関する最終的な決定権はすべてタスクの所有者にありますが、論争を解決する最適な方法は、両方のアイデアを試してみて、うまく機能する方を選択することです。

専門家

従来の方法では、データベースや GUI などの特定の領域を専門とする開発者は、その領域に専心するように提案されています。ペア・プログラミング環境では、これらの専門家は専門領域の異なる開発者のメンターとなります。開発者は自分の専門外のタスクを申告しておき、後で専門家からの支援を得ることができます。これにより、専門分野に関する知識がプロジェクト・チーム全体に浸透し、プロジェクトの仕様変更にも柔軟に対応できるようになります。

騒音

プログラミング中のペアはうるさくなります。多数のペアが作業する作業空間では、常に低いざわめき声がします。この騒音を煩わしく、気が散ると感じる人もいます。しかし、これは必ずしも深刻な問題ではありません。騒音で気が散ると感じたら、しばらく会議室に移動するとよいでしょう。

カウボーイ

多くのチームは、「カウボーイ・プログラマー」を 1 人か 2 人抱えています。彼らは、ほかの誰よりも仕事が早く、ほかのプログラマーと共同で作業できず、自分のコードを他人が読むことを許さない (あるいは許していても、他人が読めない) ようなプログラマーです。このような開発者に対処する最善の方法は、プロジェクトから外すか、製品コードを作成するロールを与えないことです。彼らは、おそらく一時的なツールを作成したり、非常に厳密なテストを実行できます。

物理的な障害と作業スタイルに関する障害

QWERTY キーボードを使用する人がいます。ドボラク配列を好む人もいます。特殊なキーボード、マウス、ディスプレイ、フットスイッチなどが使用される場合もあります。ヘッドホンで大音量の音楽を聴かなければプログラムを作成できない開発者もいます。また、お菓子の空き箱で周囲をすぐに散らかしてしまう開発者もいます。Emacs を好む人もいれば、VI を好む人もいます。また、WordPad で作業したい人もいます。
実際に、名前を付けることのできる障害は無数にあります。しかし、いずれの障害もわずかな工夫と妥協により解決できます。これらのことが障害になるようなチームは、どんな努力をしても成功しません。

ペアの計画

タスクをペアに割り当てるべきではありません。各開発者にタスクの責任を持たせてください。ペアは形式にこだわらずに組むほうがよいでしょう。各開発者は、自分の責務を遂行する上で、ほかの開発者に短期間の援助を求めます。タスクの所有権とその遂行責任は、タスクの所有者が保持します。ペア・パートナーは、あくまで援助者です。

各開発者は、タスクの見積もりを提出するときに、ペアを組むための時間を考慮に入れる必要があります。

結論

ペア・プログラミングは、コード・レビューに代わる、十分にテストされ容認された方法です。それ以上に、基本的にはソフトウェアを開発するための方法です。利点は生産性と品質ばかりでなく、チームの堅固さや士気などにも影響します。

参考資料

[1] *eXtreme Programming eXplained*, Kent Beck, Addison Wesley, 2000.

[2] *Strengthening the Case for Pair Programming*, Laurie Williams, University of Utah, July/Aug 2000 IEEE Software.

Rational®
the software development company

Dual Headquarters:
Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail:info@rational.com
Web:www.rational.com
International Locations:www.rational.com/worldwide