第 44 章　PSP と TSP

チーム・ソフトウェア・プロセス策定までの経緯

ワッツ・ハンフリー（Watts S. Humphrey）博士は、カーネギー・メロン大学ソフトウェア工学研究所（SEI）の所長として能力成熟度モデル（CMM）[1]を策定するべく調査・研究をしている過程で、組織としてのソフトウェア・プロセスの成熟度を高めるためには、その組織を構成している個々のソフトウェア技術者が優れたソフトウェア・プロセスに基づいて作業をする必要があることを痛感した。そこで博士は CMM の策定に目処を付けた後、ソフトウェア技術者個人のソフトウェア・プロセス向上に関わる仕事、つまりパーソナル・ソフトウェア・プロセス（Personal Software Process：PSP）の策定を開始した。1989 年 4 月のことという[HUM00b]。

この時博士は、自分自身で Pascal、Object Pascal、C++などのプログラム言語で多くのプログラムを書き、その過程と結果を測定し、測定結果を分析して、1993 年頃には教科書のドラフトを作りあげるという形で、PSP を一応完成させた。これを使って、1993 年から 1994 年にかけて、カーネギー・メロン大学を含むいくつかの大学で実際に学生に教えてもらい、その結果を受けてドラフトに手を入れて、1995 年に PSP の最初の教科書を出版した[HUM95]。

その後すぐに、PSP がソフトウェアの品質向上とソフトウェア開発の生産性向上にたいへん効果があることが明らかになった。しかし同時に、PSP を習得した技術者がその習得した内容を実際のソフトウェア開発で生かして使うことが、主に組織としての仕事の仕方などの関係から容易ではないということもあわせて明らかになった。そこで、この PSP を生かして使うことができる状況を作るために、博士はチーム・ソフトウェア・プロセス（Team Software Process：TSP）の策定を開始した。1996 年頃のことである。

最初の TSP のモデルは簡単なものだったが、効果を確認しながらそれを順次改善し、2000 年には TSP も完成した。前述の通り当初 TSP は、「PSP を習得した技術者がそれを生かして使うための環境」という意味を持っていた。換言すれば PSP の方が重要で、それを生かすために TSP があるという形だった。

しかし実際に PSP と TSP の両方を習得した技術者の数が増えてくると、TSP の方が相対的により重視されるようになり、PSP は TSP を習得するための前提条件、というような位置づけに変わってきている。そして後述するように、両方を習得した技術者のチームの成果はたいへん素晴らしいものということが、明らかになってきている。

パーソナル・ソフトウェア・プロセスとは何か

PSP には、2 つの側面がある。1 つ目はソフトウェア技術者としての自分の生産性を把握しようとするもの、2 つ目は自分の弱点を認識しようとするものである。

生産性の把握は以下のようにして行い、さらにその後に記述するような効果を持つ。

ソフトウェア技術者としての必要な仕事をする時、その全ての活動の記録を取る。記録の対象はその活動に要した時間と、その結果作成した成果物の量である。端的にいえば、この成果物の量を作業に要した時間で割ると、それで生産性が求まる。もちろん実際はこんなに単純なものではなく、作業を中断しなければならなかった時間の取り扱いとか、プログラミングの作業でのコメント行の取り扱いとか、いくつかの実際的な事柄に対する配慮が必要になる。

---

[1] 能力成熟度モデル（CMM）については、第 40 章で記述した。

自分自身の生産性の数字を把握していると、何か作業を始めようとする時、作成するべき成果物の量を何らかの方法で見積もることができると、この量と生産性の数値から作業に必要な時間が求まる。さらに、毎週どの程度の時間をソフトウェア技術者として行うべき作業に割り当てているかを把握できていれば､それを使ってその作業の終了時点を推定することができる。これが、ソフトウェア技術者個人としてのスケジュールになる。

ソフトウェアの開発では、今でもスケジュール遅れが頻発している。これは当初のスケジューリングの段階で、このような実績に基づいたスケジュール、つまり「実際にできる」スケジュールを立てるのではなく、「やりたい」、あるいは「やらなければならない」ものをベースにしたスケジュールを立てているためである。一般に「やりたい」ことと「実際にできる」ことの間には、大きな開きがある。個々の技術者のスケジュールが実現の可能性が高いものに変わると、それを集めたチームのスケジュールの確度が高まり、さらにプロジェクトなどの開発組織全体のスケジュールの信憑性も高まる。これが、PSP を基にした生産性把握による効果である。

もう一つの弱点の認識も、同様の方法で行う。

プログラミングの作業を例に取ると、プログラミングが終わった直後に自分が作成したチェックリストを使用して、まず自分でプログラムのチェックを行う。そこで見つかった欠陥を除去した後、コンパイルを行う。コンパイルの結果、コンパイラがさらにいくつかの欠陥を見つけるかもしれない。当然それらを除去する。TSP を使用している場合などではここでコード・インスペクション、つまり公式のレビューを実施する。その後に単体テストを行う。インスペクションと単体テストでも、さらにいくつかの欠陥が見つかるだろう。当然それらは、除去しなければならない。

普通、一般のプログラマが行っている作業はここで終わりになっている。しかし PSP では、この後さらに 2 つのことを行う。

1 つ目は、それぞれの欠陥発見の過程で見つかった欠陥を分析して、自分はどういう間違いを犯しやすいのかを把握する。そしてその結果を前述のチェックリストに反映させて、自分の弱点はプログラミング終了直後というごく初期の段階で、自分自身で発見できるように準備する。チェックリストがあまり大きくなると実際にチェックを行うことがたいへんになるので、「もうその間違いを犯すことがなくなった」と確信できる欠陥については、この時チェックリストから削除する。

2 つ目は欠陥ごとに、どの作業の段階でどのようにしてそれがプログラムに入り込んだのかを把握し、そのような欠陥が今後入り込まないようにプログラミングなどの作業のやり方を変更する。PSP の訓練を受けた後、ソフトウェア技術者としてのそれぞれの作業は、スクリプトと呼ぶ作業手順書に則って行うことになる。端的にいえば、ここでこの作業手順書を修正することになる。これは端的にいえば、「PDCA サイクルを回す」形で行われている現在の品質管理の基本的な方法のプログラミングへの適用であり、ISO 9001 の精神のプログラマ個人のプログラミングへの応用でもある。

このようにして、ソフトウェア技術者として自分の成果物の品質に責任を持ち、高品質の製品を作り出すことが、PSP での自分自身の弱点把握の目的である。

なお PSP では、ソフトウェアを開発する時の方法論やプログラムを作成する時に使用するプログラム言語は、何であってもかまわない。

PSP の前提

これまでの説明から自明のことかもしれないが、PSP では以下の事項がその前提になっている[HUM00b]。

① ソフトウェア技術者はそれぞれ、場合によれば大きく異なる個性／適性を持っている。その自分の個性／適性を把握し、それに合わせて仕事を行う必要がある。
② 自分自身の生産性を常に向上させ続けるためには、ソフトウェア技術者は定量的な観点を踏まえてよく定義された、しっかりとしたプロセスに従う必要がある。
③ 良い品質の成果物を作成するために、ソフトウェア技術者は作成する成果物の品質に、個人的に充分な責任を持たなくてはならない。
④ 欠陥を早い段階で発見し、速やかに取り除くことは、開発費用を少なくする上でたいへん効果がある。
⑤ 欠陥を発見して取り除くよりも、もともとそのような欠陥を成果物に作り込まないようにする方が、はるかに効果が大きい。
⑥ 正しい方法は常に効率が良く、費用も少なくてすむ。

PSP での作業手順

若干繰り返しになるが、PSP を習得した技術者は、以下のような手順で作業を行う。ここで、ソフトウェア技術者として行うべき作業は、設計、プログラミングとテストであると仮定して話を進める。

最初に、プログラミングするべき「要求（仕様）」が渡される。その要求を基に、スケジュールを含む作業計画を立てる。この計画立案が、非常に重要な作業となる。この計画立案作業の結果を、作業計画書として作成する。この作業で量の見積もりが必要になるが、その方法については後述する。

計画立案が終了すると、それに基づいての実際の作業を始める。最初は設計作業、次はその設計書をレビューする作業、さらにプログラミングの作業、プログラムのレビュー作業、コンパイル、テストと作業が進む。

全ての作業で、作業に要した時間とその結果作成された成果物の量を「時間ログ」と呼ぶ様式に記録する。これが生産性把握のための基礎資料となる。

さらに設計レビュー、プログラム・レビュー、コンパイル、単体テスト、および場合によればコード・インスペクションの各作業で、設計とプログラミングの作業時に埋め込まれた欠陥を発見する。これらの欠陥を、「欠陥ログ」と呼ぶ様式に記録する。これが品質の認識と、それを通しての品質向上のための基礎資料となる。

成果物が完成すると、技術者はまとめの作業を行う。ここではまず、作業の結果を把握し、その作業実績を計画書に追記する。これによって、当初の計画の妥当性を見ることができる。

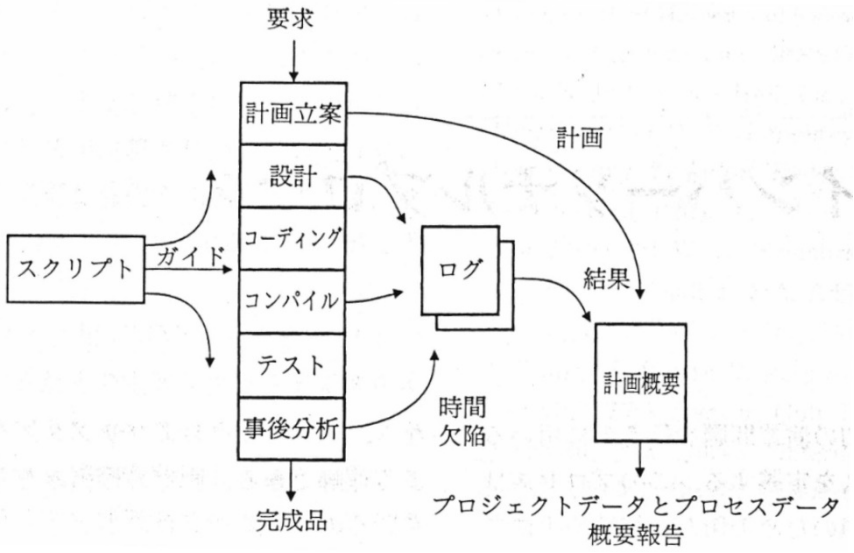
さらに必要に応じて、以下の作業を行う。

① 各作業の生産性を把握し直し、既に持っているこれらの数値を変更する。
② 埋め込んだ欠陥を分析し、チェックリストの修正を行う。
③ さらに上記の分析結果を使用して、スクリプトと呼んでいる作業手順書を修正することで、作業手順の変更を行う。

このまとめの作業を含めて、ソフトウェア技術者が行うべき作業が全て作業手順書に明記されている。逆のいい方をすると、全ての作業はこの作業手順書に準拠して行わなければならな

い。これが、PSP の最も大きな特徴の 1 つである。

これまでの話から明らかなように、PSP ではログを含む多くの様式と、作業手順書と標準を持っている。この PSP による作業の手順を、図表 44-1 に示す。

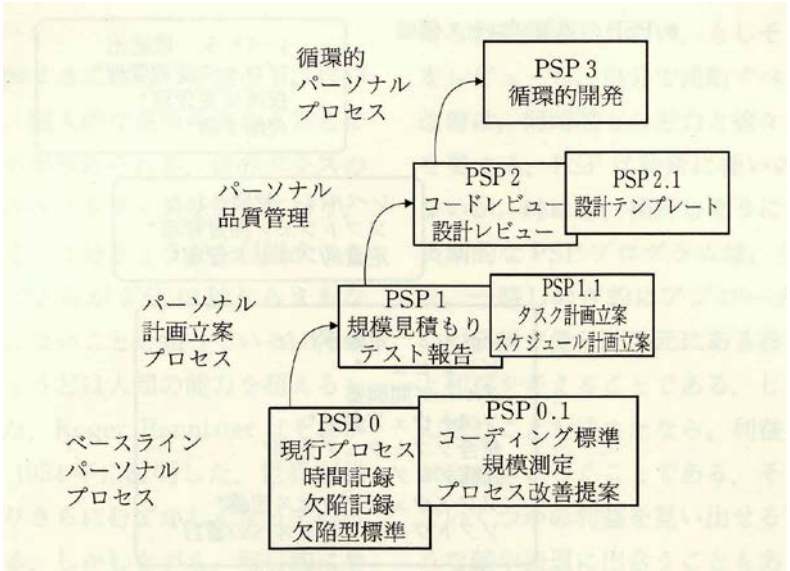


図表 44-1　PSP による作業手順（[HUM95]より）

PSP の訓練の方法

PSP には、現在のところ 3 通りの訓練の方法がある。

1 つ目は大学などで実施することを想定したもので、半期間の週 1 コマの授業の形で実施される。2 つ目は産業界のソフトウェア技術者を対象にしたもので、3 週間弱のフルタイムのセミナー形式で実施されている。そして 3 つ目は、独習する方法である。



図表 44-2　PSP の訓練の内容（[HUM95]より）

いずれの方法も、いくつかの講義を聴き、あるいは独習の場合は後述する本を読み、課題として指示された設計をし、さらにプログラムを作成して、その結果を記録し、それを分析し、レポートを作成という作業を繰り返して、PSP による作業手順の考え方と進め方を習得する。

ハンフリー博士はこれまで、PSP について 3 冊の書籍（[HUM95]、[HUM97]、[HUM05]）を出版している。いずれの本も教科書といえるが、この 3 冊目の本は、それを読みながら、SEI のホームページ（http://www.sei.cmu.edu/tsp/psp/download/student.html）から課題や指示をダウンロードして、独学方式でこの訓練を受けることができるように作成されたものである[2]。

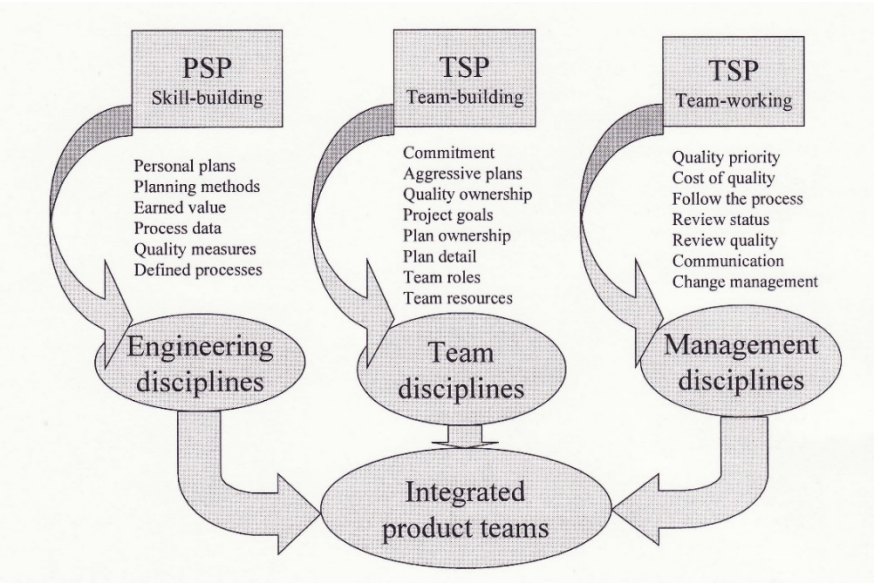
蛇足だがこの本はその目的のために使用する本であって、PSP とはどんなものかを知りたいということで読むには、必ずしも適していない。PSP がどんなものかを知りたいということなら、SEI が出している技術報告書（テクニカル・レポート）[HUM00b]か、2 冊目の本[HUM97]が適している。

なおこの訓練の過程で、博士が作成した作業手順書と、プログラムの見積もりで使用するプログラムの大きさについての基準値が示される。最初の段階はそれらをそのまま使用し、自分の実績を積み上げる過程でこの作業手順書と基準の数値を修正して、自分自身の作業手順書と基準値を作成して行くことになる。

この PSP の訓練の細かい内容を紹介することは、ここでは割愛する。ハンフリー博士はこの訓練の内容を、図表 44-2 の形で示している。

チーム・ソフトウェア・プロセスとは何か

チーム・ソフトウェア・プロセス（TSP）とは、PSP の訓練を終了したソフトウェア技術者を対象に、ソフトウェア開発を行うための「チーム」を構成し、協力してソフトウェアを開発するための手順を意味する。



図表 44-3　TSP の構成（[HUM00c]より）

[2] この URL からは、必要な資料がダウンロードできなくなってしまった（確認日：2017 年（平成 29 年）2 月 17 日）。

「チーム」は、単なる技術者の集合とは大きく異なる。チームの中では、ソフトウェア技術者たちはそれぞれ自分自身が責任を果たすべき領域を持ち、自分が担当している領域以外ではそれを担当している他の技術者の権限を尊重し、相互に情報を共有し、相互に協力し合い、できるだけ負荷を分散し、均等化し、全員がチームとしての目標達成に積極的に努める必要がある。TSP でのチームの大きさは、2 人から 20 人の間とハンフリー博士は記述している[HUM00c]。しかし実際のところ、2 人は少なすぎ、20 人は多すぎるというべきだろう。5 人から 10 人の間あたりが適切と、私は考える。

しっかりとしたチーム作りに成功すると、デマルコなどが「結束の強いチーム（jelled team）」と呼ぶ素晴らしい組織ができあがる[DEM87]。個人的な話で恐縮だが、私もソフトウェア技術者としてこのようなチームで仕事をした経験を 2 回持っている。今でもそのチームで仕事をしたことを、誇りを持って振り返ることができる。

TSP には、PSP のような特別の訓練は用意されていない。一般的な TSP の話は実践の前に必要だが、TSP は話を聞いただけですぐにそれを実践できるような簡単なものではない。TSP の訓練は、それに充分精通し、できれば SEI などが発行しているライセンスを持ったコーチの指導を受けながら、実際にチームの一員としてソフトウェア開発を経験することでしか習得することができない。

TSP の内容は、大きくチームの立ち上げ段階と、それ以降の実際の遂行の段階に分けられる。この関係を、図表 44-3 に示す。

なお TSP が開発するソフトウェアの種類、その稼働の形態などを一切拘束することはない。換言すれば、どのようなソフトウェアを作る時でも TSP は適用することができる。

チームの立ち上げ

チームの立ち上げは、TSP として非常に重要なステップである。これは後述するように 4 日間にわたる 9 つの作業で構成されるが、この目的は特定の仕事を遂行するための効果的で効率的なチームを構築することである。この立ち上げに成功すると、チームは結果として次のものを持つことができる。

- チームの目標に対する共通の認識
- その目標を達成することの重要性についての共通の認識
- チーム内での個々のメンバーの役割と、明確にされたそれぞれの責任領域
- 作業のスケジュール
- そのスケジュールについての経営者の承認

立ち上げでは、まず初日の最初のステップとして、チームを構成することになる全てのメンバーと、経営者やこれから開発しようとするソフトウェアのユーザなどがミーティングを持つ。これが、「1 つ目の作業」である。このミーティングで、経営者やユーザはそのソフトウェアを開発することの意義、重要性などをチーム・メンバーに伝え、併せて彼らが期待するカットオーバー時期、その理由などを明らかにする。チーム・メンバー側は必要な質問を行う以外、もっぱら聞き役に回る。

2 つ目から 8 つ目の作業までは、チーム・メンバーと、彼らに TSP を指導するコーチ以外の人は参加をしない。

2 つ目の作業は、「チーム・メンバーのそれぞれの責任領域を取り決め、チームのゴールを定

義する」ことである。この時に取り決められるべきチーム・メンバーの役割については、後述する。

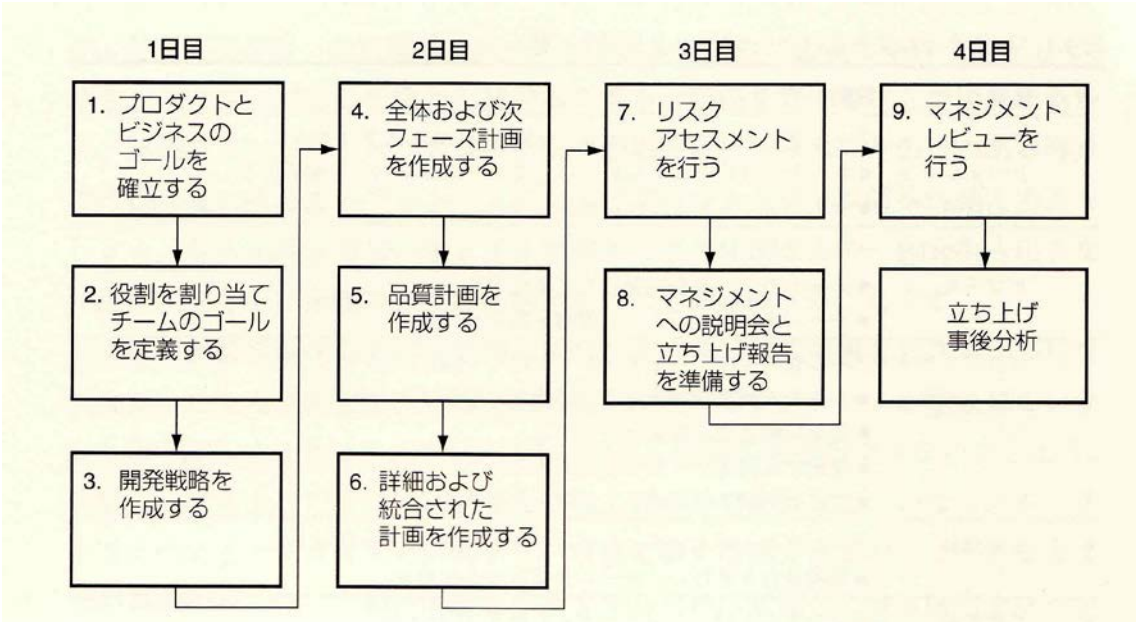
3 つ目の作業は、「ソフトウェアの開発戦略を作成する」ことである。最初の日の作業は、ここで終了する。

2 日目の 4 つ目の作業はトップダウンの方式で、「全体、および次のフェーズの計画を立案する」ことである。そして 5 つ目の作業で「品質計画を立て」、6 つ目の作業で、ボトムアップでこの「ソフトウェア開発についての詳細な計画を立てる」。もしこの時に、このままの規模では最初の段階で聞いた経営者やユーザが期待する時期までにカットオーバーできないことが明らかになると、何らかの代替案を策定する。例えば、チームをもっと大きくする／範囲を縮小する、あるいはチームを今のままにとどめて、開発の範囲も縮小しない場合は、新たな現実的なカットオーバーの時期を求める、などのことを行う。

3 日目の 7 つ目の作業で、このソフトウェアの開発での「リスクのアセスメントを行う」。そして 8 つ目の作業で、翌日の 9 つ目の作業である経営者などとの再度の「ミーティングに備えての準備」を行う。

そして 4 日目の 9 つ目の作業で経営者とユーザの代表などを交えた「再度のミーティングを持ち、この 3 日間のチームとしての検討結果を報告し、必要なら新たなチームの規模や情報システム化の範囲、あるいは現実的なカットオーバー時期の提案を行う」。経営者やユーザからいくつもの質問があるかもしれない。しかしここでプレゼンテーションしたものが、チームが 3 日間しっかりと考え、検討した結論であるなら、最終的に経営者の承認を受けられる可能性が高い。

経営者からの承認が得られるとチームは立ち上げ作業のまとめを行い、その後実際のチーム活動を開始する。



図表 44-4　チームの立ち上げ（[HUM06]より）

この立ち上げ作業は、ハンフリー博士が用意した作業手順書（スクリプト）に基づいて実施

する。そしてコーチが立ち上げ期間中ずっとチーム・メンバーと行動を共にして、必要な指導をそのチーム・メンバーに対して行うことになる。いうまでもないことだが、TSP にも多くの様式作業、手順書と標準がある。

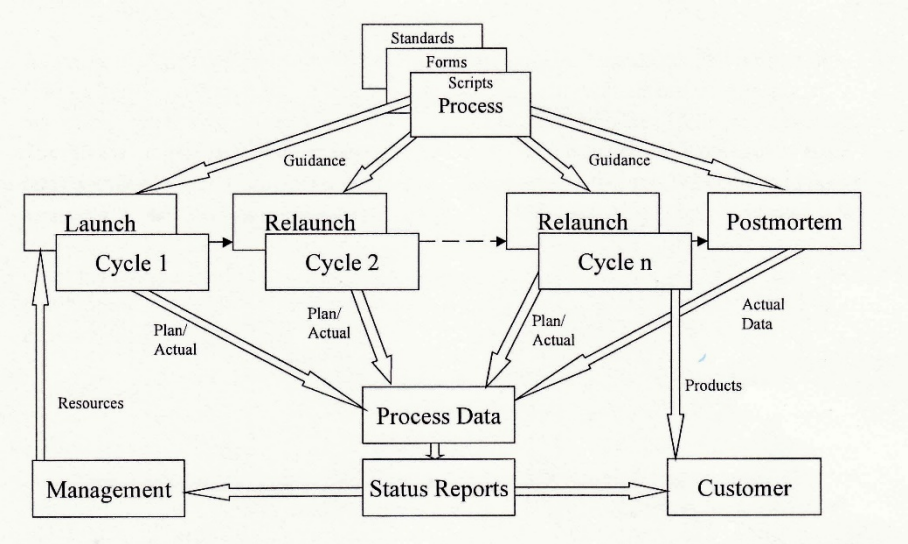
チームによるソフトウェア開発の活動

現実のソフトウェア開発のためのチームの活動では、全てのチーム・メンバーが以下に述べるような役割のいくつかを担って遂行することになる。この役割は、立ち上げ作業の 2 つ目のフェーズで取り決められるものである。

- チーム・リーダー
- 顧客とのインタフェースの責任者
- 計画立案の責任者
- 設計の責任者
- 開発の責任者
- テストの責任者
- 作業手順（プロセス）の責任者
- 品質に関する責任者
- サポートの責任者

それぞれの責任者が果たすべき役割は、やはり作業手順書の形で示され、実際の作業遂行に当たっては、コーチが指導／支援することになる。

TSP では、3～4 ヶ月間の詳細な計画を立てて仕事を進める。実際の開発期間がこの計画を立案した期間より長い場合、再計画、再々計画を立案して作業を継続する。この状況を、図表 44-5 に示す。この再計画、再々計画の際に、それぞれのチーム・メンバーが担っている役割を交換しても良い。



図表 44-5　TSP による再計画など（[HUM00c]より）

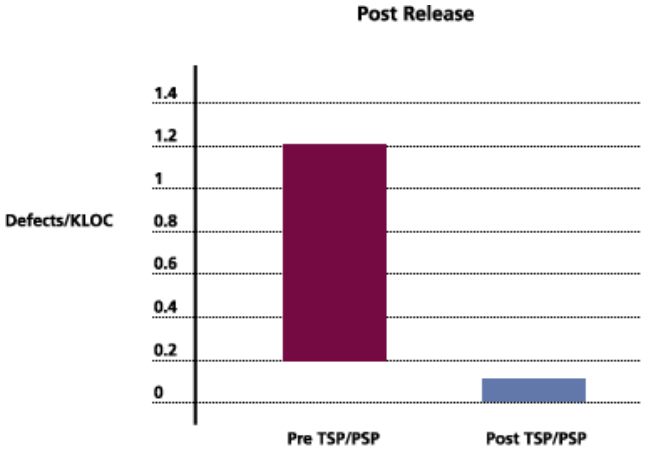
TSP についての本

ハンフリー博士はこれまでに、TSP についても 3 冊の本（[HUM00a], [HUM06], [HUM07]）を書いている。最初の本[HUM00a]は TSP の教科書ともいうべき本で、実際にコーチに指導を受けてソフトウェアの開発を行いながら TSP の訓練を受けている過程で、必要に応じて参照するのが適切と考える。

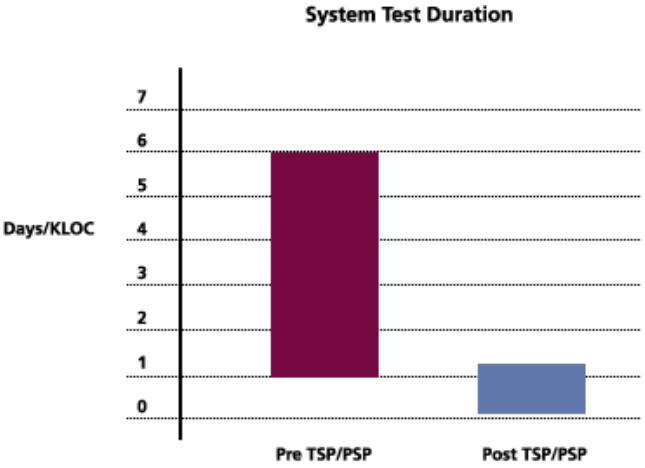
ただしこの本も、TSP とはどういうものかを把握したいという目的で読むのには適していない。その目的のためなら、残念ながら英文のものしかないが、SEI から出ているハンフリー博士が書いた技術報告書[HUM00c]が良いかもしれない。

2 冊目[HUM06]と 3 冊目[HUM07]の本は、TSP のチームでリーダになる人向けに書かれたものと、コーチを務める人向けに書かれたものである。しかし 2 冊目の本[HUM06]は、一般的なリーダ論としてもたいへん優れた本である。

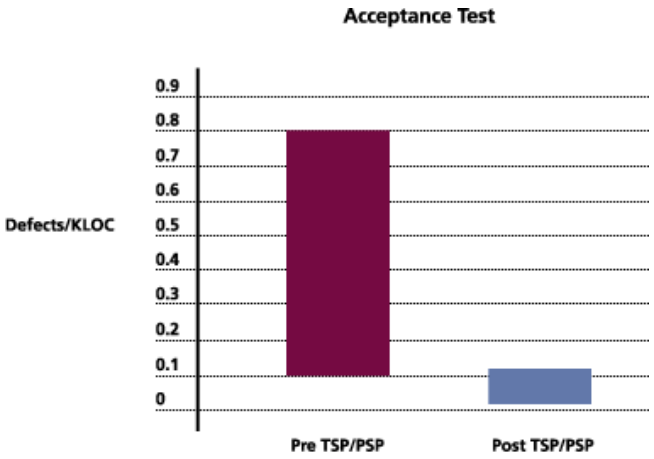
PSP と TSP による効果



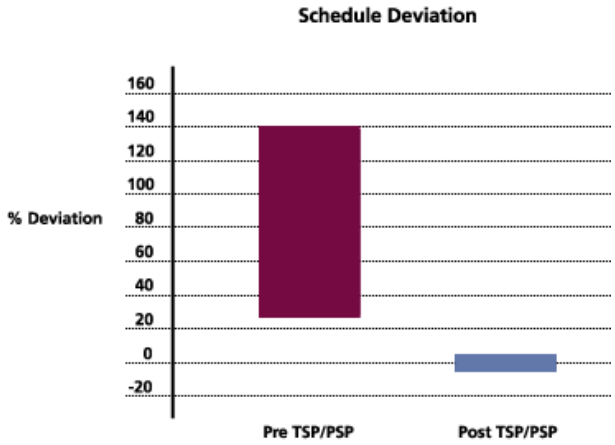
図表 44-6　カットオーバー後に発見された欠陥数（[SEI07b]より）



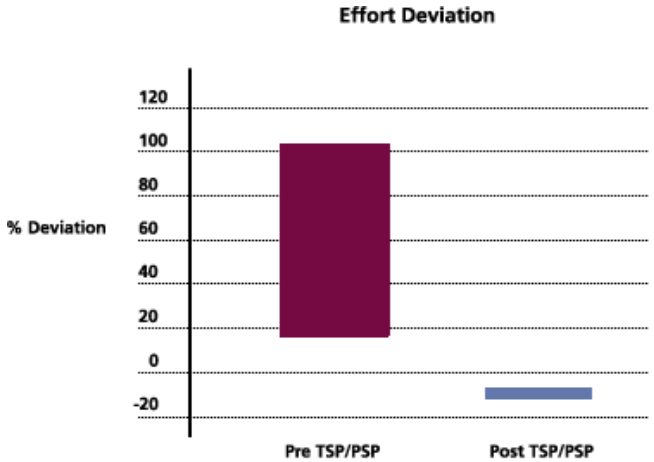
図表 44-7　システムテストに要した期間（[SEI07b]より）



図表 44-8　受け入れテストで発見された欠陥数（[SEI07b]より）



図表 44-9　スケジュールの予定と実勢の間の相違（[SEI07b]より）



図表 44-10　ワークロードの予定と実勢の間の相違（[SEI07b]より）

PSP と TSP を習得したチームが発揮した効果について、SEI は詳細な公式の技術報告書を発行している[DAV03]。しかしもっと簡便な報告[SEI07b]も発行しており、ここではその簡便

な報告を使用して、PSP と TSP の効果を記述してみたい。

ここで図示する結果は、ボーイングなど 4 社、18 個のプロジェクトから得た結果であって、PSP と TSP の訓練の前と後の結果を次の 5 つの領域で示している。

① カットオーバー後に発見された欠陥数（図表 44-6）
② システムテストに要した期間（図表 44-7）
③ 受け入れテストで発見された欠陥数（図表 44-8）
④ スケジュールの予定と実績の間の相違（図表 44-9）
⑤ ワークロードの予定と実績の間の相違（図表 44-10）

具体的には図表 44-6 以降のグラフを参照していただきたいが、いずれの場合も PSP と TSP の訓練の後では素晴らしい結果を示している。たまたまの偶然にしか過ぎないが、特にワークロードの実績（図表 44-10）では、全ての PSP と TSP の訓練を終了したプロジェクトは、当初の予定より少ないワークロードで作業を完了したという結果を示している。

PSP と TSP についての個人的なコメント

今のソフトウェアに関わる問題の多くは、フリードリッヒ・バウアー（Friedrich Bauer）教授が述べたように「ソフトウェア開発の方法が組織的でも論理的でもなく、まして工学的なアプローチもとられていなかった」ことによると、私も考えている[3]。この工学的なアプローチの中には、単に技術者の技術力の高さや新しい技術への対応力などだけでなく、技術者としての規範、態度、倫理、品質への対応方法など、技術者の生き方、考え方、仕事の仕方といった面をも広く含むものと私は受け止めている。

これまでソフトウェア技術者は、この後者の面について教育／訓練をほとんど受けてこなかった。つまりソフトウェア技術者の教育は、プログラム言語の仕様、その典型的な使い方、設計の方法など技術面だけで留まっており、それ以上には踏み出してきていないことが多かった。

日本では、良い職場には一般に仕事への取り組みの仕方などが既に「文化」として確立している。だからそのような職場に新入社員として配属されると、ごく初期の段階で管理職や先輩などからこのような面について OJT による厳しい訓練を受けるのが普通だった。

しかし私がソフトウェア技術者として仕事を始めた頃、私の上司であった管理者たちはプログラミングという作業が理解できず、結果としてソフトウェア技術者の扱い方が分からず、そのような訓練を施す機会を失してしまったように思える。

だからその時の新入社員は自分自身で好きなように仕事をし、それを誰からもとがめられず、仕事とはそういうものだ、それで良いのだと思いこんだまま年を取り、そのうち彼自身が管理者になり、経営者になり、そのまま今に至っているように見える。つまり自分が仕事の仕方などについての必要な訓練を受けていないものだから、自分が管理者になった時も自分の部下にそういう訓練を施していない。ソフトフェア技術者の世界ではこれが代々続いているように、私には思える。あるいはこれは日本だけのことではなく、世界的な現象なのかもしれない。

ハンフリー博士の PSP と TSP は、このソフトウェア技術者の仕事の仕方などについてのこれまでの態度を改める、1 つの大きな契機になるものである。

これからソフトウェア技術者を志すものは、単に Java でプログラミングできるとか、オブジェクト指向技法を駆使できるとかというだけのレベルに留まらず、PSP の訓練を通して個人としての仕事の仕方を身につけ、TSP によってチーム内で他の人と協力し合いながら作業を行

[3] バウアー教授の発言などについては、第 1 章を参照していただきたい。

う方法を習得して、これまでのソフトウェア技術者が必ずしも身につけていなかったものを率先して身につけ、大きく活躍してくれることを期待している。

ソフトウェア技術者に訓練を実施する立場の人たちもこのことを銘記し、そのカリキュラムに PSP と TSP を取り込んで欲しい。

キィワード

ソフトウェア工学研究所、能力成熟度モデル、パーソナル・ソフトウェア・プロセス、PSP、チーム・ソフトウェア・プロセス、TSP、チェックリスト（PSP）、PDCA サイクル、時間ログ、欠陥ログ、作業手順書、スクリプト、コーチ（TSP）、結束の強いチーム

略語

PSP：Personal Software Process

TSP：Team Software Process

人名

ワッツ・ハンフリー（Watts S. Humphrey）、フリードリッヒ・バウアー（Friedrich Bauer）

参考文献とリンク先

[DAV03] Noopur Davis, Julia Mullaney, “The Team Software Process℠ (TSP℠) in Practice: A Summary of Recent Results CMU/SEI-2003-014 ESC-TR-2003-014,” Carnegie Mellon University Software Engineering Institute, 2003.

[DEM87] トム・デマルコ／ティモシー・リスター著、松原友夫／山浦恒央訳、「ピープルウェア 第 2 版 ヤル気こそプロジェクト成功の鍵」、日経 BP 社、2001 年.

前記の書籍はこの本の第 2 版であるが、この第 2 版には初版の内容がそっくりそのままの形で収録されている。

この本の原書は、以下のものである。

Tom DeMarch, Timothy Lister, “Peopleware 2nd Edition Productive Projects and Teams,” Dorset House, 1999.

[HUM95] Watts S. Humphrey 著、松本 正雄他訳、「パーソナルソフトウェアプロセス技法―能力向上の決め手」、共立出版、1999 年.

この原書は、次のものである。

Watts S. Humphrey, “A Discipline for Software Engineering, “ Addison-Wesley, 1995.

[HUM97] Watts S. Humphrey 著、PSO ネットワーク訳、「パーソナルソフトウェアプロセス入門」、共立出版、2001 年.

この原書は、次のものである。

Watts S. Humphrey, “Introduction to Personal Software Process, “ Addison-Wesley, 1997.

[HUM00a] Watts S. Humphrey 著、秋山義博監訳、JASPIC TSP 研究会訳、「TSPi ガイドブック ソフトウェア開発の課題 10」、翔泳社、2008 年.

この原書は、次のものである。

Watts S. Humphrey, “Introduction to Team Software Process, “ Addison-Wesley, 2000.

[HUM00b] Watts S. Humphrey, “The Personal Software Process℠ (PSP℠) Technical Report

CMU/SEI-2000-TR-022 ESC-TR02000-022," Carnegie Mellon University Software Engineering institute, 2000.
この資料は、以下の URL からダウンロードすることができる（確認日：2017 年（平成 29 年）2 月 17 日）。
http://www.sei.cmu.edu/publications/documents/00.reports/00tr022.html

[HUM00c] Watts S. Humphrey, "The Team Software Process℠ (TSP℠) Technical Report CMU/SEI-2000-TR-023 ESC-TR02000-023," Carnegie Mellon University Software Engineering institute, 2000.
この資料は、以下の URL からダウンロードすることができる（確認日：2017 年（平成 29 年）2 月 17 日）。
http://www.sei.cmu.edu/publications/documents/00.reports/00tr023.html

[HUM05] ワッツ・S・ハンフリー著、秋山義博監訳、JASPIC TSP 研究会訳、「PSP ガイドブック ソフトウェア開発の課題 8 ソフトウェアエンジニア自己改善」、翔泳社、2007 年.
この原書は、次のものである。
Watts S. Humphrey, "PSPS℠：Self-Improvement Process for Software Engineers, " Pearson Educations, 2005.

[HUM06] ワッツ・S・ハンフリー著、秋山義博監訳、JASPIC TSP 研究会訳、「TSP ガイドブック：リーダー編 ソフトウェア開発の課題 6 」、翔泳社、2007 年.
この原書は、次のものである。
Watts S. Humphrey, "TSP：Leading a Development Team, " Pearson Educations, 2006.

[HUM07] ワッツ・S・ハンフリー著、JASPIC TSP研究会訳、「TSPガイドブック：コーチング編 (IT Architects' Archive ソフトウェア開発の課題 12)」、翔泳社、2009年.
この原書は、次のものである。
Watts S. Humphrey, "TSP：Coaching Development Teams (The SEI Series in Software Engineering)," Pearson Educations, 2007

[SEI07b] "Compilation Data for Projects Using TSP and PSP".
この資料は、以下の URL からダウンロードすることができる（確認日：2017 年（平成 29 年）2 月 17 日）。
http://www.sei.cmu.edu/tsp/results/compilation.html

（2007 年（平成 19 年）9 月 30　初稿作成）
（2016 年（平成 28 年）9 月 14 日　一部修正）