「機能－作業マトリクス」を用いた、プロジェクトの簡潔な進捗管理手法の適用事例

Practical approach of project progress management with the use of “function-process matrix”.

株式会社 東芝 ソフトウェア技術センター
Software Engineering Center, Toshiba Corporation
○田中　裕大　　小森　真紀 [1)]　　黒川 久美子 [2)]　　大渕 一彦 [3)]
○Yuta Tanaka　　Maki Komori [1)]　　Kumiko Kurokawa [2)]　　Kazuhiko Ofuchi [3)]

Abstract Some departments in our company mainly perform small enhancement development from existing system. A general gantt chart is high-intensity for such small project. In this paper, we propose "function-process matrix" that is a progress management method suited for such departments. In this method, each phase in the project is broken down to a matrix that is made from functions developed and detail processes. Cells in the matrix are treated as tasks. We applied it to several projects in a department. The result shows that the cost of planning and collecting progress information are reduced to 1/6 and 1/10 respectively.

1. 背景と問題点

当社の社会インフラ事業では、新規事業の立ち上げも行なわれているが、成熟した製品の横展開が主なビジネスとなっているドメインも存在する。そのような組織では、既存製品からの派生開発が主流であり、開発規模も小さいプロジェクトが多い。しかし、国内向け／海外向けなど、多数のプロジェクトが同時に実行されており、作業の最適化のため分業化が進んでいる。例えば、システム設計グループ、ハードウェア設計グループ、ソフトウェア設計グループ、品質保証グループ、製造グループが分かれて、かつ分業化されている。工程ごとに担当するグループが分担されており、プロジェクトの責任が工程ごとに担当するグループに引き継がれていく仕組みとなっている。（図 1)。この仕組みにより、同時に多数の物件を処理することができる。

しかし、このような組織では、一般的なガントチャートによる計画／進捗管理が難しく、効率的な管理が行われていない現状が見受けられる。例えば、プロジェクト初期にシステム設計グループが、全体の工程表を作成しようとしても、各工程の作業内容が把握できず、詳細化することができない。各グループも、上流の設計結果により作業が詳細化されるため、プロジェクト開始時期にブレークダウンすることは難しい。その結果、大まかな工程表しか作成されない場合がある。大まかな工程表では正確な進捗を把握することが難しく、結果として、工程管理に多くの工数をかけているにも関わらず、正確な進捗が把握できず、プロジェクトの進捗の遅れに対し、適切な対策をタイムリーにとれないなどの問題が起こっている。さらに、進捗管理にかけている工数に見合った効果が出ていないことにより、進捗管理自体が定着せず、ますますプロジェクトの進捗がわからないという悪循環が形成されていた。

株式会社 東芝 ソフトウェア技術センター
Software Engineering Center, Toshiba Corporation

神奈川県川崎市幸区小向東芝町 1　Tel:044-549-2439　e-mail: yuta9.tanaka@toshiba.co.jp
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki-shi, Kanagawa, Japan

1）株式会社 東芝 ソフトウェア技術センター
2）株式会社 東芝 府中事業所 技術・情報システム部
3）株式会社 東芝 府中事業所 交通ドライブシステム部

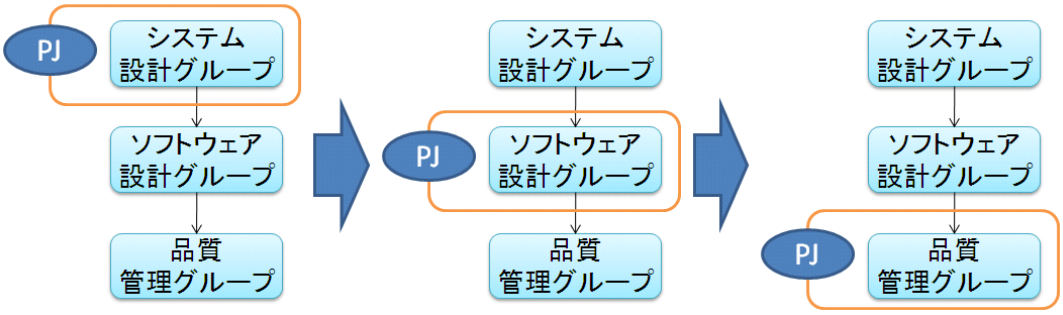


図 1 プロジェクトの流れ イメージ図

2. 問題点のブレークダウン

第 1 章であげた、「プロジェクトの進捗の遅れに対し、適切な対策をタイムリーにとれていない」「かけている工数に見合った効果が出ていない」などの問題を持つ社内のある部門に対し、改善を行った。まず、問題点のヒアリングを実施し、進捗管理に関する問題点を列挙しその因果関係を分析した。分析結果を図 2 に示す。

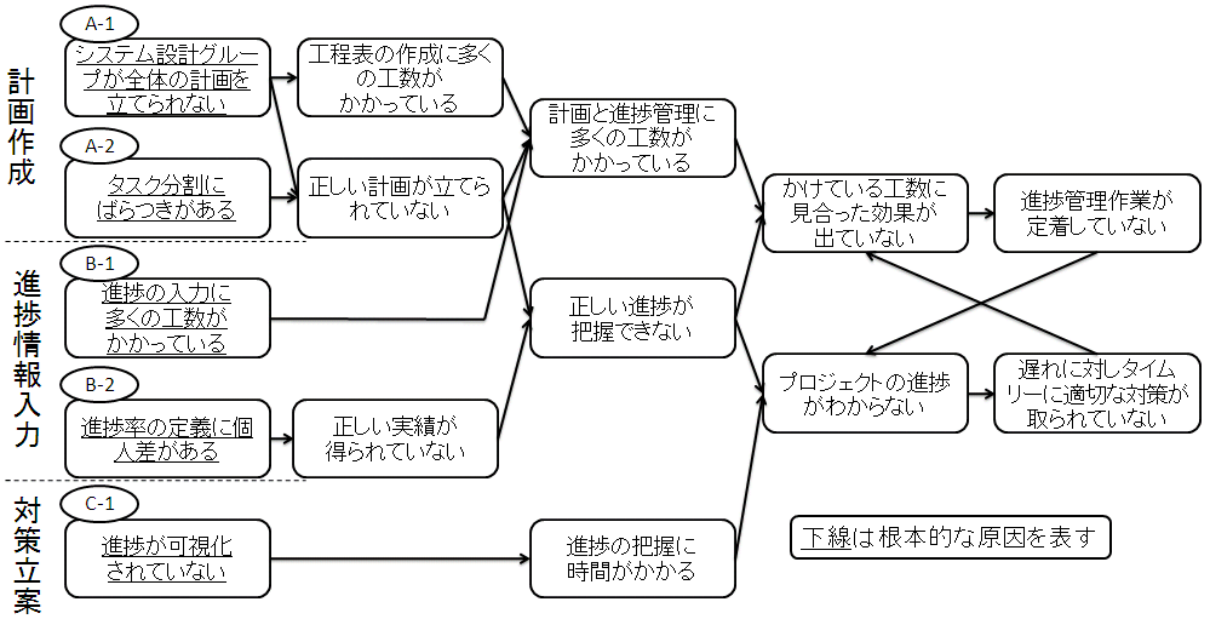


図 2 進捗管理に関する問題点とその因果関係

分析より、根本原因は、以下の 5 つに集約できることが分かる。

A. 計画作成上の問題

A-1. システム設計グループが全体の計画を立てられない

分業化により各グループの役割が定められているが、多くの場合、自身が作業を行わない部分まで含めたプロジェクト全体の計画策定をシステム設計グループが行なっている。これにより、計画の精度の低下や工数の増加が引き起こされている。

A-2. タスク分割にばらつきがある

プロジェクトの作業を WBS(Work Breakdown Structure)により詳細化を行う際、分割の手順や粒度などはあまり細かく規定されておらず作業者に任されている。その結果、作業者によってバラツキがある。

B. 進捗情報入力上の問題

B-1. 進捗の入力に多くの工数がかかっている

分割された各タスクの進捗率を考え、手作業で入力するのに多くの工数がかかっている。

B-2. 進捗率の定義に個人差がある

進捗率が明確に定義されていないため、各作業者が入力する進捗率に個人差がある。その結果、プロジェクトとしての正しい進捗を得ることができてない。

C. 対策立案上の問題

C-1. 進捗が可視化されていない

現状のガントチャートを用いた確認方法では、遅れているタスクや進んでいるタスクが混じっており、プロジェクトとしての遅れがどの程度なのかなど、必要な情報を手に入れるのが難しい。これにより、進捗の把握に時間がかかっている。

これらの根本原因により、第 1 章で挙げたような問題が顕在化していたと考えられる。

3. 既存手法の調査

第 2 章で挙げた根本原因を解決するために、学会誌や論文などを中心に、進捗管理手法についての調査を行った。その結果、論文[1]が本改善の参考になることが分かった。

論文[1]の進捗管理手法は、まず、プロジェクトを「工程」で分割する。次に、それぞれの工程を「準備・実行・レビュー」を基本とする「共通枠」へと分解を行う。例えば、「論理設計」工程は、「論理設計前の準備作業」「論理設計」「論理設計書のレビュー作業」の 3 つに分割される。このように共通枠を用いて工程を分割したものを「作業内容」と呼ぶ。表 1 は、論文[1]において、この分割を行った例である。プロジェクトでは、各工程で開発する機能ごとに、「作業内容」でブレークダウンしたものをタスクとして進捗管理を行なう。すなわち、この手法では、タスク分割手法の統一化を行なっている。これにより、第 2 章であげた問題のうち、問題点 A-2 が解消されることが期待される。

表 1 工程の作業要素の分割(論文[1]から引用)

	略号	作業内容
論理設計	SSP	論理設計前の準備作業
(SS)	SS	論理設計
	SSR	論理設計書のレビュー作業
プログラム開発	PS	詳細設計作成作業
(PG)	PG	プログラミング作業
	PTP	プログラム単体テストの準備作業
	PT	プログラム単体テスト
	PTR	プログラム単体テスト結果レビュー
結合テスト	ITP	結合テスト準備作業
(IT)	IT	結合テスト
	ITR	結合テスト結果レビュー
システムテスト	STP	システムテスト準備作業
(ST)	ST	システムテスト
	STR	システムテスト結果レビュー

4. 解決策の検討

第 2 章で挙げた問題を解決するために、論文[1]を参考として、「機能－作業マトリクス」を用いた進捗管理手法を提案した。

4.1 機能－作業マトリクス

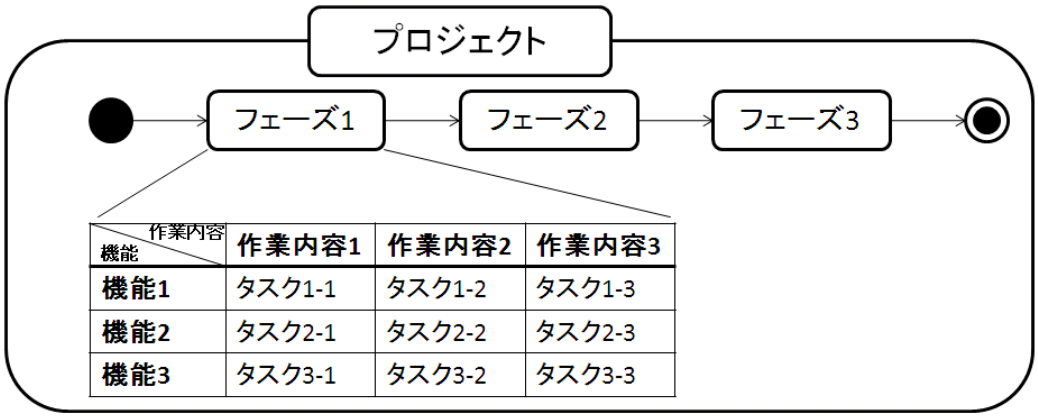
論文[1]を参考にして、機能と作業をマトリクス上に配置し、マトリクスの各マスをタスクとして管理する管理手法を考案した。これを「機能－作業マトリクス」と呼ぶ。

「機能－作業マトリクス」では、まず進捗管理のスコープ全体をいくつかの「フェーズ」に分割する。「フェーズ」は WBS の工程にあたる。次に、各「フェーズ」を具体的な作業を表す「作業内容」に分解を行う。「作業内容」は論文[1]の作業内容と似た概念である。例えば、ソフトウェア実装フェーズは「コーディング」「デバッグ」「単体テスト作成」などの「作業内容」に分解できる。また、各「フェーズ」内で実際に作成するモジュールや機能などを列挙する。これを「機

能」と呼ぶ。

各「フェーズ」について、この「機能」と「作業内容」のマトリクスを作成し、マトリクスの各マスを「タスク」として管理することで進捗管理を行う(図 3)。1 つの「フェーズ」のマトリクス内には、「機能」の数と「作業内容」の数の積と同数のタスクが存在することとなる。例えば図 3 の場合、フェーズ 1 には 9 個のタスクが存在している。この各タスクの予定と実績を管理することによりフェーズ内の進捗管理を行う。

各「フェーズ」をマトリクスで表すことで、従来のガントチャートと比較して、タスクの一覧性が向上する。これにより、問題点 C-1 の改善が期待できる。



機能＼作業内容	作業内容1	作業内容2	作業内容3
機能1	タスク1-1	タスク1-2	タスク1-3
機能2	タスク2-1	タスク2-2	タスク2-3
機能3	タスク3-1	タスク3-2	タスク3-3

図 3 「機能」と「作業内容」による「フェーズ」の分割例

4.2 「フェーズ」「作業内容」「機能」の定義

「機能－作業マトリクス」を用いた進捗管理を行うためには、「フェーズ」「作業内容」「機能」をあらかじめ決定しておく、もしくは決定方法を統一しておく必要がある。さらに、作業者の負荷を減らすためには、マトリクスの作成に極力人手をかけずに行えることが望ましい。

「フェーズ」は、プロジェクトの工程と定義した。プロジェクトはあらかじめ決められた工程を流れていくため、これに沿った形で作成した。結果として、以下の 3 フェーズへと分割された。

- システム設計フェーズ
  ソフトウェア・ハードウェアなどのシステム全体の基本設計を行い、既存システムからの変更点を抽出する
- CASE ツールフェーズ
  CASE ツールを用いて、設計・実装を行う
- ソフトウェア実装フェーズ
  ソフトウェアの詳細設計やコーディングを行う

「作業内容」は、「フェーズ」内で実施する定型的な作業と定義した。例えば、ソフトウェア実装フェーズでは、以下のような「作業内容」に分解される。

- 詳細設計
- コーディング
- デバッグ
- テスト結果記録

これらの「作業内容」は、すべてのプロジェクトで共通したものが用いられる。これにより、統一した基準に基づいたデータ収集が容易にできるようになり、進捗管理や見積りの精度向上につながることが期待できる。

「機能」は各「フェーズ」で作成・実装すべきモジュール・実現すべき機能と定義した。「機能」の洗出しには、システム設計フェーズと CASE ツールフェーズ、ソフトウェア設計フェーズで異なる手法を用いる。システム設計フェーズは、「機能」に相当する部分が製品ごとにほぼ固まっているため、製品ごとにパターンを用意することで定型化することができた。一方、CASE ツールフェ

ーズ、ソフトウェア実装フェーズの対象となる「機能」は膨大だが、今回のプロジェクトで変更が必要なものだけをピックアップすれば良い。必要な変更点はシステム設計のアウトプットである「差分開発点リスト」にて明確化される。そこで、CASE ツールフェーズ、ソフトウェア実装フェーズの「機能」はシステム設計フェーズ終了後に「差分開発点リスト」から展開することとした(図 4)。

以上のように、「フェーズ」および「作業内容」はあらかじめ用意されたものを利用し、「機能」はあらかじめ用意されたもの、もしくは前工程でのアウトプットから一意に決定できることから、作業者にはほとんど負荷をかけずに作業の洗出しを行うことができ、タスク分解手法も統一化することができる。これにより、問題点 A-1 および A-2 が解決されることが期待できる。

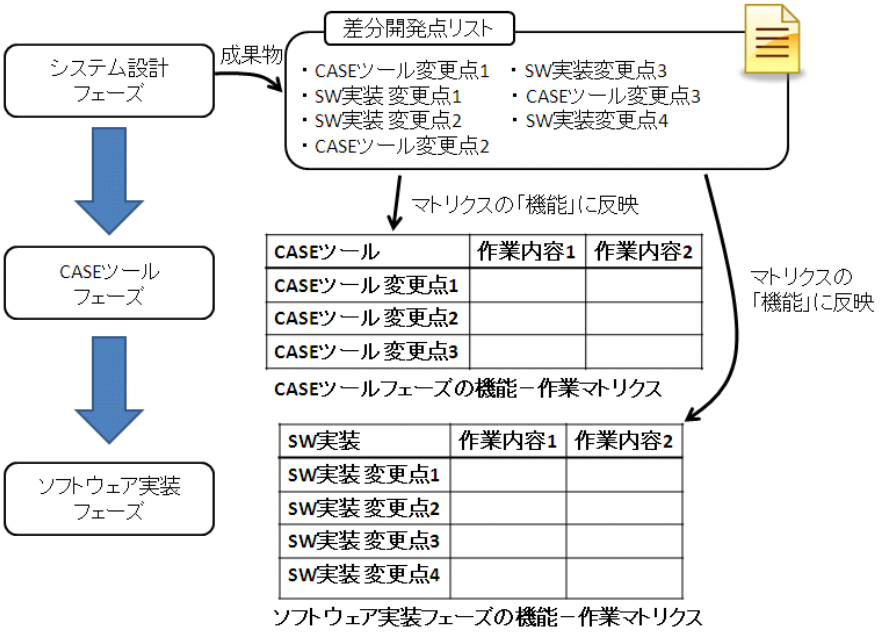


図 4 CASE ツール/ソフトウェア実装フェーズにおける、「機能」決定プロセス

4.3 重みを用いた開始・終了予定日の算出

プロジェクトの進捗管理を行うには、各タスクの予定を計画する必要がある。この作業は、通常、各タスクの開始予定日と終了予定日を入力することで行う。

しかし、マトリクス状にした各タスクに開始予定日、終了予定日を設定するのは作業者の負荷が高い。そこで、4.2 節で決定した「作業内容」が時系列に並んだ作業であることに着目し、「機能」単位で全作業内容の工数を見積もり、開始予定日と終了予定日を設定することとした。しかし、このままでは、各タスクの開始日を特定しがたい。また、「作業内容」によっても要する工数も異なる。そこで、各「作業内容」に重みを設定することで、各タスクの開始日を開始予定日、終了予定日を算出する工夫を行った。例えば、ある「機能」の開始予定日が 2013/8/1、終了予定日が 2013/8/10 だった場合、各タスクの開始予定日、終了予定日は表 2 のように自動的に算出できる。なお、重みは「作業内容」ごとにあたえられ、デフォルトの値があらかじめ入っている。なお、算出した各タスクの開始・終了予定日はマトリクスには表示していないが、タスク単位での遅れの表示などに使用する予定である。

表 2 開始予定日算出の例

「機能」の開始予定日が 2013/8/1、終了予定日が 2013/8/10 の場合				
	詳細設計	コーディング	デバッグ	テスト結果記録
重み	2	2	5	1
開始予定日	2013/8/1	2013/8/3	2013/8/5	2013/8/10
終了予定日	2013/8/2	2013/8/4	2013/8/9	2013/8/10

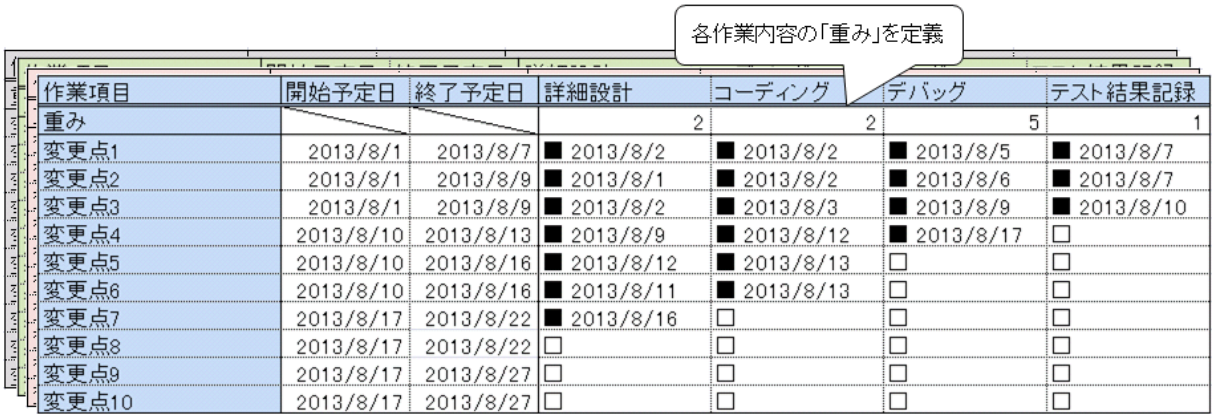
4.4 完了・未完了による各タスクの管理

通常、進捗管理では、各タスクの進捗率を入力する。しかし、各タスクに対し、進捗率を入力するためには、各作業の進捗率を割り出す作業が必要となり、作業者の負荷となる。そこで WBS の 0-100 法を参考に、各タスクの状態は完了・未完了の 2 値のみとすることにした。作業者は終わったタスクにのみチェックを入れ、未着手および進行中のタスクは未完了状態として扱う。この方法は各タスクの進捗率を管理する方法と比較して進捗率の精度が下がるという問題がある。しかし規模が小さいプロジェクトが多いため、機能－作業マトリクスにおける各タスクは十分に詳細化されていることから、作業者の負荷を減らすことを優先し採用した。これにより、問題点 B-1 が解消されることが期待できる。

4.5 重みを用いた進捗率の算出

プロジェクトの遅れに対し、適切な対策を行うには、「フェーズ」全体の進捗を把握する必要がある。4.3 節で使用した重みは、各「作業内容」に要する工数の比率を表している。この値を利用することで、進捗率の算出を行えるよう工夫した。図 5 にマトリクスを用いた進捗入力画面の例を示す。図 5 はソフトウェア実装フェーズにおけるものである。

ソフトウェア実装フェーズの各作業の重みは以下となっている。



作業項目	開始予定日	終了予定日	詳細設計	コーディング	デバッグ	テスト結果記録
重み			2	2	5	1
変更点1	2013/8/1	2013/8/7	■ 2013/8/2	■ 2013/8/2	■ 2013/8/5	■ 2013/8/7
変更点2	2013/8/1	2013/8/9	■ 2013/8/1	■ 2013/8/2	■ 2013/8/6	■ 2013/8/7
変更点3	2013/8/1	2013/8/9	■ 2013/8/2	■ 2013/8/3	■ 2013/8/9	■ 2013/8/10
変更点4	2013/8/10	2013/8/13	■ 2013/8/9	■ 2013/8/12	■ 2013/8/17	□
変更点5	2013/8/10	2013/8/16	■ 2013/8/12	■ 2013/8/13	□	□
変更点6	2013/8/10	2013/8/16	■ 2013/8/11	■ 2013/8/13	□	□
変更点7	2013/8/17	2013/8/22	■ 2013/8/16	□	□	□
変更点8	2013/8/17	2013/8/22	□	□	□	□
変更点9	2013/8/17	2013/8/27	□	□	□	□
変更点10	2013/8/17	2013/8/27	□	□	□	□

図 5 機能－作業マトリクスを用いた進捗入力画面の例

- 詳細設計(重み 2)
- コーディング(重み 2)
- デバッグ(重み 5)
- テスト結果記録(重み 1)

図 5 のように「機能」が 10 個あった場合、重み 2 のマスが 20 個、重み 5 のマスが 10 個、重み 1 のマスが 10 個ある計算となる。ここで、図 5 のように、重み 2 のマスが 13 個、重み 5 のマスが 4 個、重み 1 のマスが 4 個終了している(マスが黒四角で始まっているものは終了しているとする)場合の進捗率は以下の計算で算出できる。

$$\text{フェーズ進捗率} = \frac{(2\times13)+(5\times4)+(1\times4)}{(2\times20)+(5\times10)+(1\times10)} = \frac{1}{2} = 50\%$$

つまり、この時点でのソフトウェア実装フェーズの進捗率は 50%となる。このような工夫により、問題点 B-2 を解消し、開発者に負荷をかけずに現実に即した進捗率を求めることができる。

4.6 進捗の可視化

直感的に進捗を把握する手法として、バーンダウンチャートによる可視化を導入した。バーンダウンチャートでは、「フェーズ」ごとに進捗が可視化される。図 6 は、1 つの「フェーズ」の進捗の推移を表示した例であり、作業の未完了率(%)の 1 日ごとの推移が表示されている。グラフ内には計画線と実績線の 2 本の線が表示されている。計画線は、各「機能」の開始予定日・終了予定日から算出されたプロジェクトの進捗の目安となる線である。実績線は、作業者が進捗管理表の各項目のチェック履歴と各「作業内容」の重みを用いて算出された進捗率の実績値である。計画値と、実績値を見比べることにより、現在の計画からの遅れと傾向を直感的に把握することができる。例えば、図 6 では、序盤はほぼ計画通りの進捗だったが、2013 年 8 月 10 日頃からプロジェクトが遅れ始めてきたことが読み取れる。このようなグラフを用意することにより、問題点 C-1 が解消され、進捗の把握やスケジュール遅延に対する対策の必要性の判断などに利用することが期待できる。

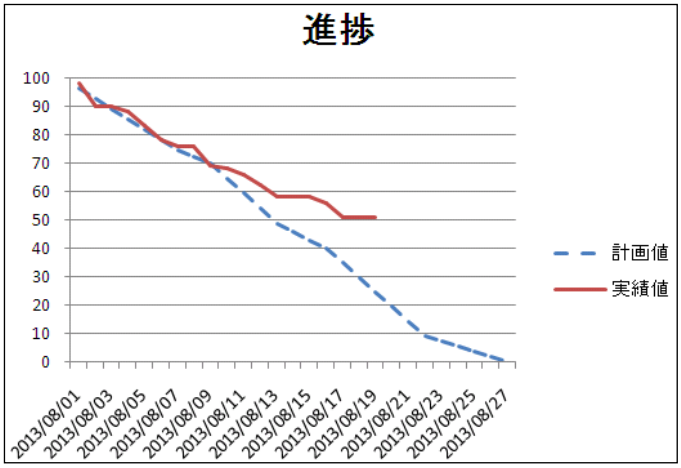


図 6 バーンダウンチャートによる可視化例

4.7 システムの実装

前述の重み付き「機能－作業マトリクス」と可視化手段としてのバーンダウンチャートを盛り込んだ進捗管理ツールを開発した。ダブルクリックによるチェック状態の切り替え、ユーザの入力情報から自動的にバーンダウンチャートを作成するなど、徹底的に作業者の負担を減らす工夫を行った。

5. 試行評価

本手法は、現在数プロジェクトにおいて試行を実施している。本手法を導入することの効果として、計画および進捗管理にかける工数の削減と可視化による進捗の簡単な把握があげられる。これまでのガントチャートベースの進捗管理では、タスクの詳細化が難しく、時間が掛かるにもかかわらず進捗の精度が低かった。これにより、進捗管理が定着していなかった。

あるプロジェクトでは、プロジェクトの工程表の策定にかかる工数が約 1/6 になったという報告が得られた。また、1 回の進捗記入に要する工数は、本手法適用前の時点では各タスクの進捗率の算出などで 10 分ほどかかっていたが、本手法では 1 分以内になったという結果が得られた。

システムの利用者からは、「簡潔な進捗入力により進捗管理工数が削減された」「進捗管理の仕組みが実情に合っている」などの感想があがった。また、バーンダウンチャートによる可視化は、従来使用していたガントチャートと比較して、進捗状況を直感的かつ簡潔に把握できるという評価を受けた。

6. まとめ

既存の進捗管理システムが合わない部門に対し、「機能－作業 マトリクス」を用いた進捗管理システムの提案と適用を行った。

まず、適用部門の進捗管理に関する問題点を洗い出し、5 つの根本原因を抽出した。その後、それらの問題の解決する手法として「機能－作業マトリクス」を提案し、部門への適用を行った。

本手法は、第 2 章で挙げた 5 つの根本原因に対し、下記のように解決を図っている。

A. 計画作成上の問題

A-1. システム設計グループが全体の計画を立てている

「機能－作業マトリクス」では、計画を「作業内容」と「機能」のマトリクスを用いて管理する。製品ごとに定型的にパターン化されたマトリクスを用意することで、従来起こっていた「進捗の計画作業に多くの工数がかかっている」「正しい計画が立てられていない」などの問題が解消されることが期待される。

A-2. タスク分割にばらつきがある

問題点 A-1 と同様の方法により、タスクの分割方法を定型化・統一化したことにより、タスク分割のばらつきは解消される。

B. 進捗情報入力上の問題

B-1. 進捗の入力に多くの工数がかかっている

プロジェクトが小規模であり、求められる精度もそれほど高くないことを利用し、各タスクの進捗を未完了状態(0%)か完了状態(100%)の 2 値とした。作業者は終わったタスクにチェックを入れるだけの操作で進捗管理を行える。これにより、進捗入力にかかる工数は大きく削減することが期待できる。

B-2. 進捗率の定義に個人差がある

問題点 B-1 で述べたように、各タスクは未完了か完了の 2 値であり、進捗率の定義に個人差は無い。

C. 対策立案上の問題

C-1. 進捗が可視化されていない

「機能－作業マトリクス」を用いて作成した予定および実績を、バーンダウンチャートを用いて可視化を行った。これにより予定と実績を直感的に把握することができ、プロジェクトの管理に利用することが期待される。

上記のような特徴を持つ進捗管理システムの実装を行い、数プロジェクトにて試行を行った。その結果、計画策定および進捗情報の入力にかかる工数が大きく削減されたという結果が得られた。また、実感と合う進捗率が直感的かつ簡潔に把握できるという評価を受けた。つまり、図 2 の「計画と進捗管理に多くの工数がかかっている」「正しい進捗が得られていない」「進捗の把握に時間がかかる」などの問題が解消されたと考えられる。これにより、今後、「かけた工数に見合った効果が出ておらず、進捗管理作業が定着していない」「プロジェクトの進捗がわからず、遅れに対しタイムリーに適切な対策が取られていない」などの問題が解決されることが期待できる。

現在このシステムは試行段階であるが、今後全てのプロジェクトに適用する予定である。また、現在このシステムの対象はプロジェクトの中のシステム設計およびソフトウェア設計・実装のみである。今後はハードウェア設計などプロジェクト内の他の部分への展開を検討していきたい。

参考文献

[1] 武智 英記、プロジェクト進捗管理における定量化及び可視化とそのコントロール方法について、Journal of the Society of Project Management Vol6, No.3, pp.50-53, 2004