ソフトウェア工学 118－24
（1998．3．11）

開発プロジェクトにおける段階的技術導入法：クリーンルーム手法とオブジェクト指向開発法の開発プラン

本吉 由紀夫
日立電子サービス（株）ASSIST 開発部
横浜市戸塚区川上町 87-4
moe@hitachi-densa.co.jp

要旨

本論文では，小規模安定型の開発チームに適したプロジェクト計画技法として技術依存関係という概念を用いた，段階的な開発プロセスの計画アプローチを紹介する。このアプローチにより，開発チームは従来の開発プロセスから新たなパラダイムへと段階的に移行することが可能となる。

本アプローチでは，プロジェクト計画は整合の取れた漸増的な改善を必要としており，前回のプロジェクトに対して新技術や新技法を導入することにより，次回のプロジェクトをよりよいものにしようと考える。

本アプローチの要点は，導入すべき新技術・技法を技術依存関係にしたがって分割し，導入手順を立案することにある。本論文では，このアイデアに基づくオブジェクト指向開発技法とクリーンルーム手法の実プロジェクトへの適用結果について紹介する。

An Incremental Development Process Planning Approach : Introducing Cleanroom Method and Object-Oriented Development Method

Yukio Motoyoshi
ASSIST Development Department
Hitachi Electronics Services Co.,Ltd.
87-4 Kawakami-cho, Totsuka-ku,
Yokohama, 244-0805 JAPAN
+81 45 822 1111
moe@hitachi-densa.co.jp

Abstract

In this paper, we present an incremental development process planning approach (IDPA) which uses the idea of technical-dependency-based assessment of the project planning for a small and stable development team shifting slowly but steady to a new software paradigm that fits the traditional development process. The project plan was a well-connected set of incremental fragments of improvements with the introduction of new technologies or methods to the previous plans. IDPA is a method of assessing each technology or method to be decomposed and scheduled according to the technical dependency at the time of introduction. We also present a case study applying this idea to a real development project, in which object-oriented technology and the cleanroom method were introduced, and present the results of its evaluation.

１．はじめに

いわゆるミッションクリティカルシステムと呼ばれるシステムの開発において，従来我々はウォーターフォール型のプロセスモデルと構造化分析・設計・開発手法を採用していた。このアプローチは確かに十分役に立つ開発手法であるが，今日ではそのような重要なシステムの開発においてもトータルの開発コストの削減が求められている。コストの削減は，開発・保守・および品質を維持したまま開発期間を短縮するなど，システムのライフサイクル全般にわたる。このような状況をかんがみ，システム開発コストの低減のために，プロジェクトの計画・管理・開発手法などについて新たなアプローチを検討する必要があった。いくつかの事例的なプロジェクトの結果，組織に適したプロセスの改善を段階的に実現するためのプロジェクト計画の手法について，「段階的プロジェクト計画手法」というアプローチに至った。

１．１ プロジェクトチームの状況

過去５年にわたり，我々は開発チームに対して，オブジェクト指向技術・クリーンルーム手法・インクリメンタル開発アプローチなどの新しい技術・技法・ツール・手法を導入させることにより開発プロセスの改善を図ってきた。開発チームの特徴は，メンバーの交代が少なく，定期的に教育を受けており，チーム間でのメンバーの交代も少ないことであり，このため，メンバーのスキルは通常開発を行っているドメインに特化していることである。

改善に取り組む前は，開発期間に占める分析・設計・製造工程の比率は，1:2:3 であった。これは，開発チームのメンバーがターゲットシステムの要求内容を熟知しており新システムにおける深い分析をおろそかにしがちなためであった。この点は，人員予算が潤沢な時代においては大きな問題ではなく，必要なシステムが確実に動作することが重要であった。つまりコストパフォーマンスよりも品質・信頼性重視のシステム開発アプローチであった。プロジェクト期間は通常1〜2 年と比較的長期であった。しかし今日では，情報システム関連の投資も削減を余儀なくされ，メインフレーム中心のシステムがクライアントサーバシステムへとシフトするに伴い，またプロジェクト期間が 3〜12 ヶ月へと短縮されるに伴い，従来の開発アプローチでは時代に追いつけなくなりつつある。そこで，開発アプローチの改善を進めることにより工程の比率も 3:2:1 とすることを課題とした。

１．２ プロジェクト計画

プロセスを改善するために，過去のプロジェクトをレビューした結果，まず計画プロセスの改善から取り組むことにした。

プロジェクト計画のプロセスは，製品の品質・機能性・コストを左右する点で，非常に重要である。そこで，開発プロセスをコントロール可能で繰り返し可能とするために，十分に練り上げた PDCA(Plan/Do/Check/Action)サイクルが必要であり，開発チームをより習熟させる必要がある。

このアプローチを成功させるために，IEEE 規格 1058.1 をプロジェクト計画の基本的なテンプレートとして採用し，ドキュメントで考慮すべきまたは記述すべき項目の参考とした。項目としては，組織的な課題に関する項目，管理プロセスに関する項目，技法・手法・ツールなどの技術的な課題に関する項目，ワークパッケージ，作成すべきドキュメント，スケジュール，予算などである。このテンプレートに基づき，新技術に関する項目のうち，何を・いつ・どのように追加するか，あるいは修正するかについて検討した。

１．３ オブジェクト指向技術とクリーンルーム手法の導入

最初のステップとして，オブジェクト指向技術を導入して生産性を向上させることと，インクリメンタルプロセスモデルや段階的詳細化とレビュー技法などのクリーンルーム手法のキーとなる要素をいくつか導入し品質を向上させることを検討した。

よい構造と汎用コンポーネントの蓄積により生産性を向上するというストラテジに基づき，オブジェクト指向技術を導入する一方，よい構造の開発プロセスが製品の品質を保証するというストラテジに基づき，クリーンルーム手法を適用することとした。

１．４ 段階的な開発プロジェクトの計画手法

理想的には，新しいパラダイムの導入はいくつかの段階に分けられるとよい。というのも，受け入れる側が同時に多くのことを受け入れることは困難となるからである。計画的に新しい技術を導入することにより，スムーズな移行を図ることが望ましい。ここで無理をすると，受け手に無用な混乱を引き起こし，結果として新しいパラダイムへと移行できなくなる。このような状況を避けるために，我々は新技術の導入を計画的に行うこととした。これを段階的プロジェクト計画アプローチと呼んでいる。プロジェクトリーダは，開発チームの各メンバーが１つ１つの技術的課題を確実にクリアするまで待ち，スキルを向上させる。これにより開発メンバーの自信を持ちモラールを向上するよう仕向けるものである。

２． 段階的プロジェクト計画とは

２．１ ２つの側面： プロセスとプロダクト

基本的に，プロジェクトはプロダクト（製品）とプロセスの積であると考える。これをプロジェクト空間と呼ぶ。（図１） このプロジェクト空間には，プロジェクトを達成するための道筋が投影される。ここで言う道筋とは理想的なプロジェクト状態へ至るまでの経路を指しており，いくつかのプロジェクトを経験することにより，理想的なプロジェクト状態へ近づくことが可能であるということと，同時に各プロジェクトは理想へ至る途上（ステージ）であることを意味する。

我々は，理想的なプロジェクト状態の軸として，プロセスとプロダクトを設定した。この軸にしたがって改善を進めることにより理想的なプロジェクト状態へ近づくことを目標とした。図１において，Pn は計画した道筋を示す。n が大きくなるほどプロジェクトの状態は理想状態に近づいている。換言すれば，n と n+1 の間で新技術などを導入しプロジェクトを改善することを意味する。

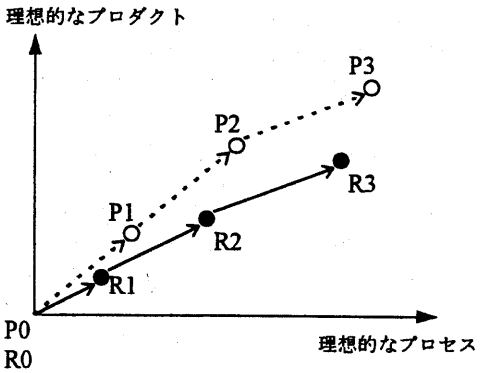

図 1. 理想的なプロジェクト状態への道筋

我々はプロジェクトを成功させるために，いつも最善の策を講じてプロジェクトを立案する(P1, P2, . . .)のだが，結果としては計画通りにいかないことが往々にしてある(R1, R2, . . .)。このギャップは，計画がつたないか管理の方法に問題があるなどの要因で発生すると考えられる。そこで，このギャップを排除するためには，ギャップの発生原因を突き止め，どうすれば排除できるかを明らかにしなければならない。ソフトウェアの開発においては従来からいくつもの手法が考案されてきているが，我々は主にプロセス面からはクリーンルーム手法による品質向上を，プロダクト面からはオブジェクト指向技術を用いることにより生産性向上を図り，プロジェクトを改善することとした。

２．２ 技術依存グラフによる技術的な課題への取り組み

プロセスとプロダクトの改善のためには各ステージにおいて何を改善するか，どんな新技術を導入するかを明確にしておく必要がある。例えば「計画ステージ１では継承を用いる」などの宣言である。しかしながら，このような宣言は，前提となる技術を習得していない場合受け入れられるものではない。すなわち，新技術の導入は各技術の依存関係にしたがって導入せざるを得ないことを意味する。また同時に適切な依存関係を設定しておかないと，いくつかの技術的なトピックを１つのステージで導入することは非常に困難になる。

そこで，個々の技術をプロジェクトに１度に導入できる程度の適切な塊に分割し，塊ごとの関係をきちんと分析しておく必要がある。例えば，クリーンルーム手法における機能検証は，プログラムの構造（連接，分岐，繰り返し）に依存する。そこで，構造化プログラミング技法が機能検証の前提習得技術となる。さらに手法間の関係も明確にする必要がある。例えば，クリーンルーム手法における機能検証は，オブジェクト指向開発手法におけるデータ変換ビューに基づかねば正しい検証ができない。

２．３ 段階的プロジェクト計画アプローチの必要性

新技術を導入するにあたり技術依存グラフとともに考慮すべき事柄として，人間にとっての差異がどれくらいかという問題がある。開発チームのメンバーにとって受け入れられる程度の違いにすることによって初めて，新技術・技法の導入が成功する。つまり一度に導入できる新技術・技法はメンバーのスキルにも依存する。ただし逆説的であるが，メンバーは受け入れ準備を常にスキルの向上を心がけねばならない。この点で，本論文の中心となる課題は，「プロセスを一番よい方法で改善するため，コスト・品質・生産性においてもっとも効果的な導入計画を，いかにして計画するか」という点である。

３． 段階的プロジェクト計画の事例

３．１ 対象領域

我々の部署では，コンピュータシステムの保守に関わるシステム開発を行っている。

従来はメインフレーム対象の保守業務が中心であったが，昨今の PC やワークステーションの普及により，保守員の活動範囲も多岐にわたるようになり，また 24 時間 365 日無停止で稼動するシステムが増えるにつれ，点検や障害対策を短時間で実施する手段が一層重要になってきている。

最初のプロジェクトは，保守員の点検作業などの作業スケジュールを行うシステムを，メインフレームとワークステーションによるシステムから PC をクライアントとするクライアントサーバシステムへ再構築するものである。このプロジェクトは開発チームにとって初めての PC を用いたクライアントサーバシステムの開発，またいわゆるモバイルコンピューティングへの対応を初めて考慮した開発である。また，C++とオブジェクト指向技術を初めて導入した。

２番目のプロジェクトは，問題管理システムと呼ぶ Q&A などの進捗管理システムの機能拡張である。この開発において，ユーザインタフェース部分の開発にクリーンルーム手法を初めて適用した。

３番目のプロジェクトは，次のプロジェクトに向けたプロトタイプを構築するものである。次のプロジェクトを非常に重要視しているため，プロトタイプの構築によって，WWW(World Wide Web) や JavaScript などの新技術がうまく動作するものか検証することとした。同時に，話題になりつつある CORBA/IIOP や Java などの新技術の評価も行った。

４番目のプロジェクトはテクニカルサポート用のシステムの再構築である。システムのユーザはオフィスや現場にいるエンジニアであり，機器の故障などの障害対策に用いる。従来のシステムは，メインフレームとワークステーションを用いて構築していたが，構築後すでに 10 年を経過し技術者のニーズの変化に伴い再構築が必要となっている。このプロジェクトでは，WWW などのインターネット技術に基づいたシステムの構築を目指しているが，現時点では基本設計が完了した段階である。

３．２ プロジェクトの規模とスケジュール

各プロジェクトの開発規模などを表１に示す。開発規模は見積りと実績をKLOC(Kilo Lines Of Code)で，工数を人月で，検出された欠陥の数，正規化した生産性と品質を示している。

各プロジェクトのスケジュールとインクリメントの数，プロジェクトに適用した技術を表２に示す。最初のプロジェクトではインクリメンタルアプローチを採用しなかったためインクリメントは１となっている。２番目と３番目のプロジェクトではインクリメンタルアプローチを採用したが，開発期間の都合でインクリメント数は２とした。

表 1. プロジェクトの規模

	見積 (KLOC)	実績 (KLOC)	工数 (人月)	欠陥の数	生産性	品質
スケジュール システム	120	146	370	92	1	1
問題管理システム	12	15	18	検出できず	2	-
プロトタイプ	18	12	12	6	2.5	1.26
テクニカルサポート システム	250	-	計画中	-	-	-

表 2. プロジェクトのスケジュール

	開始	終了	インクリメント	導入技術
スケジュール システム	Apr., 93	Oct., 96	1	OOT
問題管理システム	Oct., 95	Mar.,96	2	OOT+CRM
プロトタイプ	May, 97	Oct.,97	2	OOT
テクニカルサポート システム	Jan., 97	Oct.,99	計画中	OOT+CRM

３．３ 段階的計画と開発スケジュール

表 1,2 に示すように，インクリメンタルアプローチを２回実施した。２番目のプロジェクトではよい結果が得られたが，３番目のプロジェクトでは劇的な効果は見られなかった。

これは，３番目のプロジェクトで導入しようとした技術が新しすぎて上手な導入を図ることができなかったためであり，計画段階の不具合である。次のプロジェクトでよい結果を得るため，段階的計画アプローチの再検討を行った。

我々は初期の段階で，オブジェクト指向技術とクリーンルーム手法に関して，技術要素の依存関係を整理し，表３に示すような技術課題マトリクスを作成し，プロジェクトの段階においてどれだけの技術課題を導入するかを検討した。そして，図２に示すような技術依存グラフを作成し，オブジェクト指向技術とクリーンルーム手法の導入を進めてきた。図は，依存グラフの１部とそこに現れる計画していた道筋を示している。細い矢印の元側は前提となる技術であり，矢印の先が次に習得する技術課題である。矢印につけた名前は習得すべき技術概念を示している。太い矢

印は開発チームがたどるべき技術習得の道筋を示す。上と左に導入する技術の位置づけを与えている。

最初のプロジェクトで，オブジェクト指向プログラミングとその環境を導入した。C++言語の利用，クラスライブラリのユーザインタフェース要素への利用，オブジェクトダイアグラムの採用，構造化プログラミングの採用である。

2番目のプロジェクトにおいて，設計段階への技術導入とドキュメントの品質向上をねらってオブジェクト指向設計ツールの採用，状態ダイアグラムの採用，チームレビューの実施強化を行った。3番目のプロジェクトでは，対象領域が通信システムへシフトした。そこで，ミドルレベルの再利用，例えば，Java/JavaScript の利用，コンポーネントの利用，動的ビューダイアグラムの利用を導入した。4番目のプロジェクトでは，高レベルの再利用技術や厳密なアプローチの導入を計画している。例えば，デザインパターンやアーキテクチャパターンの利用や，関数的仕様記述や検証の手法である。

表1に示すように，プロジェクトの結果は生産性と品質の両面において期待通りに向上している。段階的計画の初期で生産性の向上の方向に向かっているのは，図2に示すように，よい構造のソフトウェアは厳密アプローチへの前提となるからである。現在のプロジェクトは品質面の改善を計画している。このように，段階的計画アプローチは開発プロジェクトを的を絞って段階的に改善してゆくよい手段であるといえる。

表3. オブジェクト指向関連技術の技術課題マトリクス(部分)

分野	技術課題	プロジェクト			
		スケジュール	問題管理	プロトタイプ	ﾃｸﾆｶﾙｻﾎﾟｰﾄ
言語・ツール	C++の使用	○	○	○	×
	JavaScript の使用	×	×	○	○
	Java の使用	×	×	×	○
	UI 要素へのクラスライブラリ使用	○	○	－	○
	コンポーネントの使用	×	○	○	○
	汎用データ型のクラスライブラリ使用	×	○	－	○
	継承ベースの再利用	○	○	○	○
	テンプレートベースの再利用	×	○	×	×
	コンポーネントベースの再利用	×	△	○	○
	オブジェクトの直列化	×	△	○	○
	オブジェクトの永続化	×	×	○	○
	CASE ツール(Rational Rose)の利用	△	○	－	○
オブジェクト指向技術	オブジェクト図の作成	○	○	○	○
	データフロー図の作成	×	○	○	○
	状態図の作成	×	○	○	○
	イベントダイアグラム，ユースケースの作成	×	×	×	○
設計技術	デザインパターンの利用	×	×	○	○
	アーキテクチャパターンの利用	×	×	×	○

○：適用，×：適用せず，△：一部，－：該当せず

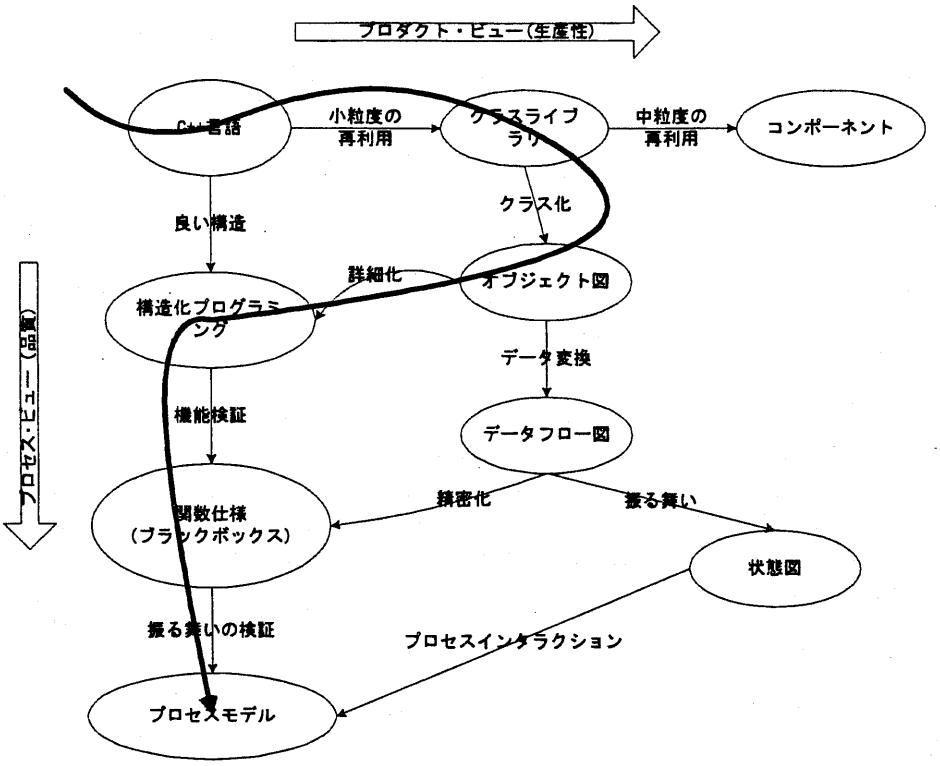

図2. 技術依存グラフ(部分)

4．考察

4．1 計画立案の主要ファクタについて

問題を単純化するため，プロセスとプロダクトを計画の主要なファクタとした。しかし，現実にはコストや期間などのビジネスファクタや組織的なファクタもある。プロジェクトの結果から技術面について計測制御可能なプロジェクトの基本的な属性は，ドキュメントを含むプロダクトの品質とチームならびにチームメンバーの生産性であると判断し，これら２面について計画を進めている。

4．2 技術依存グラフに基づく制約について

開発チームに対しては，今までに多くの技術的課題を与えてきたが，ここで示した技術依存グラフは初期のものであり，もっと洗練する必要がある。

問題としては，洗練した結果が，すでにその課題を過ぎてしまった開発チームにとっては無用という点である。洗練した技術依存グラフを企業全体に広げれば役に立つこともあろうが，他の開発チームは，異なったバックグラウンドを持ち，新しいパラダイムを受け入れるには別のアプローチが必要かもしれない。この点について，技術依存グラフをどんなプロジェクトでも適用できるように繰り返し可能とする手順が必要である。いろいろな方法論の技術的な依存関係を分析することは，ソフトウェアシステムにおいて要求仕様を分析することに似ている。技術依存グラフを作成するために，さまざまな方法論について，方法論そのものと個々の間の連続性を分析し，中間プロダクトを設け，プロダクト間の関係を抽出する必要がある。

4．3 段階的プロジェクト計画アプローチの必要性について

段階的プロジェクト計画アプローチを適用するもっとも重要な根拠は，たとえリスクがあってもプロジェクトを成功へ導きたいからである。プロジェクトが新しいドメインを目標としているとき，プロジェクト計画も何らかのよいリスク管理計画や最善の結果を引き出すための対応策を備えるべきである。

最善の結果を得るためには，観測点においてリスクを検知する手段とそのリスクを回復または回避する対応策を準備していなければならない。もし，開発チームのメンバーがスキル不足などの場合にそのリスクを回避できないかもしれない。そのために，常日頃，メンバーの教育を推進し，そのようなリスクの発生を避けようとしている。しかしながら，現在の悩みは，世の中の教育のカリキュラムがプロジェクト指向でないため，メンバーのスキルの判定が難しいことである。ここで提示した技術依存グラフに基づくような教育カリキュラムが必要である。

5． 結論

本論文では，段階的プロジェクト計画アプローチというアイデアを提案し，実プロジェクトでの適用の結果について議論した。今回の事例の範囲においては，このアイデアはうまく機能したが，技術依存グラフの内容については，繰り返し適用可能とするための方法論を欠いている。今後は企業レベルでの教育計画やメンバースキルを向上するための仕掛け，組織的な知識の蓄積などの企業全体にわたりスキルを共有できるようなシステムなどのトピックについてさらに研究を広げる所存である。

また，開発チームの教育の仕方については，企業ごとに異なる可能性もある。このアイデアを自分の組織に導入する場合は，技術依存グラフを十分に吟味する必要がある。

謝辞

本論文で取り上げたプロジェクトは，開発チームのメンバーの多大な努力と協力の賜物である。彼らの協力に大いに感謝する。

参考文献

1. Booch, G. Object-Oriented Analysis and Design with Applications, 2nd Edition. Benjamin-Cummings, 1994.
2. Shlaer, S. and S.J. Mellor. Object-Oriented Lifecycles: Modeling the World in States. Prentice-Hall, 1992.
3. Rumbaugh, J. Booch, G. Jacobson, I. Unified Modeling Language, version 0.91. Rational Software Corporation, Santa Clara, California, 1996.
4. Linger, C.R. Cleanroom Software Engineering for Zero-Defect Software. 15th Annual Conference on Software Engineering, IEEE, 1993.
5. Linger, C.R. Cleanroom Process Model. IEEE Software, March, 1994.
6. Motoyoshi,Y. Otsuki,S. An Incremental Project Plan: Introducing Cleanroom Method and Object-Oriented Development Method. 20th International Conference on Software Engineering, Learning and Status Report, IEEE, April, 1998.
7. 二木厚吉監修，「ソフトウェアクリーンルーム手法」，日科技連，1997年12月