Vol. 36 No. 5　　情　報　処　理　　May 1995

解　説

ソフトウェアプロセス

9. ソフトウェアプロセス・リエンジニアリング†

青　山　幹　雄††

1. はじめに

ソフトウェア開発の仕組みを体系的に改善する**SPR(Software Process Re-engineering)**の概念と方法を筆者らの経験と事例に基づき紹介する.

プロセスプログラミングの提案[16]を契機として，ソフトウェア開発の手順を形式的に記述した**ソフトウェアプロセス**（以下，プロセスと呼ぶ）に関する研究，開発が始まった．プロセスは，プロダクトとともに，ソフトウェア開発という車の両輪を成す本質的な概念である．プロセスの研究，開発は，生産性と品質の向上，開発期間短縮を図る上で新しい技術をもたらした．しかし，プロセスの実行主体は「人」であるので，プロセスの改善には，技術面のみならず人間的要因などの多くの問題が潜んでいる.

一方，企業経営では**BPR(Business Process Re-engineering)**が注目されている[11]．BPRとSPRは，ともに，プロセスに着目して作業の仕組みを改善する方法であるから，共通点が多い．したがって，本稿では，他分野でのプロセス改善方法とも照し合わせて，SPRの概念と方法を示す.

2. SPRとは

2.1 SPRとは

SPRとは，プロセスに基づいて開発の仕組みを見直し，生産性や品質を向上する方法である．SPRは，形式的プロセス定義に基づきソフトウェアシステムの設計と同様な設計方法論にのっとり体系的にプロセスの再構築を図る点で，全社的品質管理（TQC）などの従来のプロセス改善方法を一歩進めたものと考えられる.

2.2 SPRのプロセスと方法

SPRでは，まず，現行のプロセスを明らかにし，次に，あるべき姿に再構築する．筆者らの経験に基づき他の事例も加味したSPRのプロセスとその方法を**図-1**に示す.

(1) プロセスのリバースエンジニアリング

まず，現行の作業を実態に即して目に見える形に表すことが出発点となる．リバースエンジニアリングとは，現行の作業内容からそのプロセスを明らかにして，ドキュメント化することである.

一般に，作業内容は全社，あるいはプロジェクトごとの作業標準としてドキュメント化されている．しかし，作業標準は，作業の詳細な構造（手順）が明確に規定されていなかったり，形式的でないなどの問題がある．このため，プロセスはソフトウェアの設計と同様，形式性と抽象化の枠組みを持った**プロセス記述言語**で記述する．筆者らは，**図-2**に示すように，木構造チャートを拡張したプロセス記述言語を用いてプロセスを定義している[3),4)]．この他，データフロー図やオブジェクト指向分析・設計方法論の表記法[1)]など多くのプロセス記述言語が提案されている.

このような人が見るためのプロセス定義を，コンピュータにより実行されるプロセスプログラムと区別して，**プロセススクリプト**と呼ぶ[5)].

(2) プロセスの再設計

現行のプロセスを企業戦略や開発プロジェクトの狙いに基づき再設計する．これは，**図-3**に示す二つのアプローチがある.

1) 改善：現行プロセスの問題点を小集団活動などを通して分析，改善する．**継続的プロセス改善(Continuous Process Improvement)**として知られている.

† Software Process Reengineering by Mikio AOYAMA (Department of Information and Electronics Engineering Niigata Institute of Technology).

†† 新潟工科大学情報電子工学科

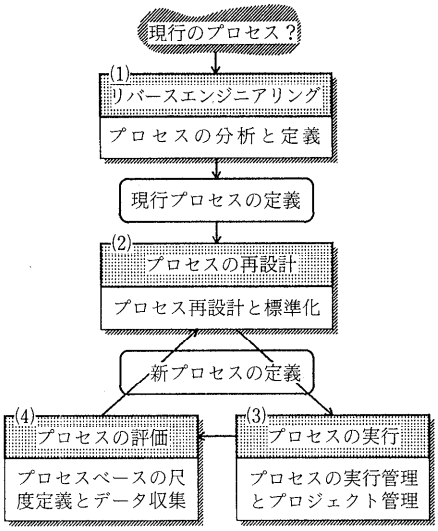


図-1 SPRのプロセス

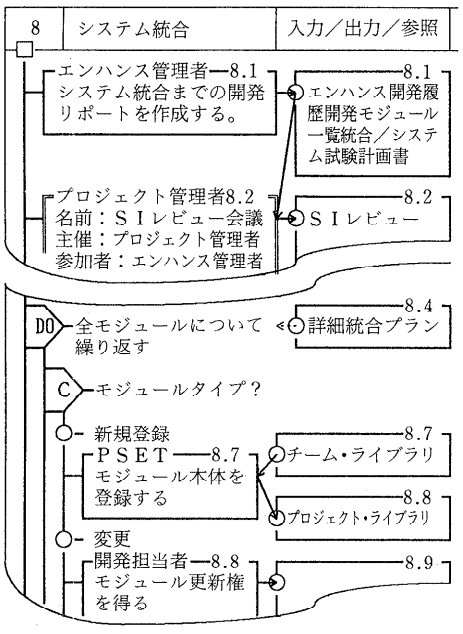


図-2 プロセススクリプトの記述例

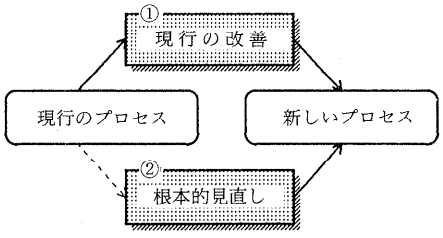


図-3 プロセス再設計の二つのアプローチ

2) 根本的見直し：プロセスのあるべき理想像から出発する**プロセス改革 (Process Innovation)**.

どのアプローチを採るかは開発プロジェクトの性質や狙いによる．たとえば，新規プロジェクトでは，現行の作業標準やプロセスを根本的に見直し，開発戦略に合致した新しいプロセスを設計することも可能である．一方，機能追加などの保守主体のプロジェクトでは，プロセス改善のアプローチがリスクの低減などの点から望ましい場合も多い．

プロセス改善の着眼点の一つは再作業(Rework)である．AT&T や HP での調査では，再作業の比率が20～33%を占めていた[10),17)]．Raythcon では，レビューなどの方法を改善し，再作業の比率を40%から11%に削減した結果，全体として30%のコストダウンを達成した[7)]．

(3) プロセスの実行

プロセスの実行主体が人であることがプロセスの実行と管理を難しくしている．一方，開発コストや納期を厳密かつ正確に管理する要求が高まっている．従来のプロジェクト管理方法は進捗や工数などを比較的マクロに管理していたため，このような要求には十分応えられない．しかし，プロセスに基づくアプローチは，次のような新しい解決方法をもたらす．

1) 細粒度：開発の最小単位である個人ごとに実行，管理を支援できる．

2) 一貫性：個人からチーム，プロジェクト全体にわたり一貫した共通の尺度となる．

3) 共通性：個人の背後にある共通の構造を表しているので，参照すべき基準となる．

さらに，個人ごとにプロセスの実行を支援するため，プロセスに基づきツールを統合するCASE環境が開発されている[14),18)]．また，グループウェアの中でも，ワークフロー管理としてプロセスの実行を支援する環境が提供されている．プロジェクト管理支援として，プロセスの実行に沿って各種の開発データを自動的に収集し，計画や管理を支援する環境も開発されている．

(4) プロセスの評価

プロセスの評価方法を分類して**表-1**に示す．

1) マクロ評価とミクロ評価

a) マクロ評価の方法：企業やプロジェクト

表-1　プロセス評価のアプローチ

	絶対評価	相対評価
マクロ評価	CMM	ベンチマーク 経年評価
ミクロ評価		

全体のプロセスの良さを評価する方法．たとえば，**プロセス成熟度モデルCMM(Capability Maturity Model)**は5段階の尺度で評価する[12]．**ROI(Return On Investment：投資対効果)**やプロジェクトの総コストは企業経営の観点から評価する定量的尺度である[2),7),13)]．

b)　ミクロ評価の方法：プロセスの実行データの分析や開発者の意見などから工程やチームごとの問題点を掘り起こす方法．たとえば，個人ごとの作業時間分析によって，全作業時間の60%は手待ちや作業の調整など非生産的な作業に費されていることが明らかになった[17]．

2)　絶対評価と相対評価

a)　絶対評価の方法：CMMや総コストなど絶対尺度に基づき評価する方法．

b)　相対評価の方法：他社や他プロジェクトや過去のデータと比較，評価する方法．**ベンチマーキング(Benchmarking)**は他社や他プロジェクトのプロセスと比較し，**ベストプラクティス**と呼ぶ実践で成果を上げている優れた技術を抽出する方法としてソフトウェア開発にも適用され始めている[8]．

絶対評価と相対評価は相補的な関係にあるので，組み合わせて用いるべきである．

2.3　プロセス・リエンジニアリングの系譜

プロセスに基づく改善のアプローチを**プロセス・リエンジニアリング**と呼ぼう．プロセス・リエンジニアリングは，対象領域により，**図-4**のように整理できる．ハードウェアの製造では，生産システム工学の中で，**プロセス設計(Process Design)**方法が開発されている．たとえば，リーン生産方式と呼ばれるトヨタ生産方式の核はプロセスの管理と改善にある[15]．トヨタ生産方式では，人が作業する上での問題をプロセスの構造問題として捉える．これは，ソフトウェア開発にも適用できると思われる．

また，開発着手後のSPRの効果には自ずと限界がある．開発以前のビジネスプロセスや出荷後の保守プロセスを含むトータルなプロセスのリエンジニアリングへ視点を広げる必要がある．

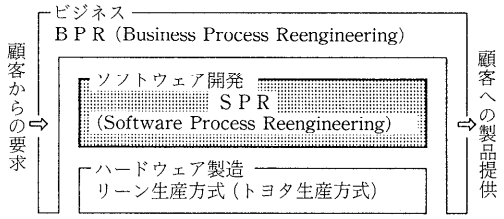


図-4　プロセス・リエンジニアリングの系譜

3.　SPRの実践

筆者らが従事している開発プロジェクトでは，**図-5**に示す3つのアプローチからSPRを進めてきた．

3.1　プロセス改善

(1)　プロセス改善の背景と狙い

プロセス改善の発端は，**分散並行開発**と呼ぶ新しいソフトウェアプロセスの導入にあった[1),3),4)]．地域分散した複数の開発拠点で単一ソフトウェアシステムの複数機能を並行して開発する．

分散並行開発では，プロセスの観点から次の問題がある．

1)　分散開発の問題：開発拠点間でプロセスが分断する．また，開発拠点ごとにプロセスや進捗基準などが乖離する．

2)　並行開発の問題：複数の並行開発チーム間のプロセスの相互作用（干渉）が，開発の円滑な実行を妨げる．あるチームの進捗遅れが他のチームの進捗を乱したり，システム統合段階ではプロジェクト全体の作業の同期を乱す．

分散並行開発を展開する上で，このような問題は現場の知恵やプロセス管理などの面から対応していた．しかし，プロセスプログラムの提案[16]とその後のプロセスに関する研究に触発され，プロジェクト共通のプロセススクリプト定義を行い，それに準拠して開発を進めることとした．これは，プロセスに基づく共通の尺度により進捗や品質を定量的に管理する方法と支援環境の開発，適用へと発展した．現在では，全工程にわたりプロセスを中心とする開発支援環境を適用している．

特筆すべきは，集中開発から分散並行開発への移行が，プロセスにとどまらず開発管理や支援環境など開発のあらゆる面で改善を促したことである．分散並行開発では，集中開発では分からなか

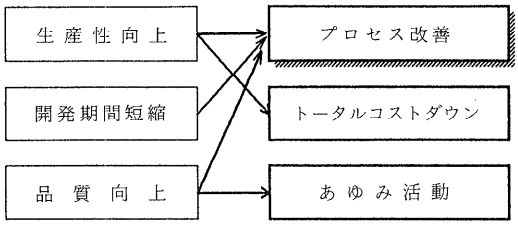


図-5 SPRのアプローチ

ったプロセスの実行と管理上の問題が露顕した．ドキュメントなどのプロダクトの電子化も促進した．これは，トヨタ生産方式の開発，導入過程で，従来のプロセスでは隠れていた問題点が露顕し，プロセス改善の契機となった経験[15]と相通ずる．

(2) プロセス記述言語YPLとプロセス定義

図-2に示すように，全工程にわたるプロセスを木構造チャートの一つであるYAC IIを拡張したプロセス記述言語YPL(YAC II-oriented Process Description Language)で記述した[1]．YAC IIを採用したのは，詳細設計言語としてすでにプロジェクトで全面的に使用していたためである．

(3) プロセスの実行と管理の支援：PRIME

プロセスに基づく開発支援環境PRIME (PRocess Infomation ManagEr: プライム)を開発，適用している．一人一人の開発者からチーム，プロジェクトに至る階層構造に応じてプロセスの実行と管理を一貫して支援する．PRIMEは，一人一台のワークステーションとWANを介したクライアント/サーバ型の広域分散処理環境を前提として次のような機能を提供する．

1) プロセス実行支援：一人一人の開発者に対するプロセススクリプトの提示．

2) プロセス管理支援：個人，チーム，プロジェクトの各階層でのプロセスの実行情報の収集と評価，共有による管理支援．

3) ツールや設計情報などの開発資源の統合

一方，PRIMEには，プロセスの実行主体が人である点を考慮して次のような工夫をした．

a) 「管理水準」の設定と「あそび」の許容：ある水準より詳細なプロセスの実行順序は任意とする．これは，機構設計における「あそび」の概念と対応している．

b) 非同期性の許容：プロセスの実行は一様に進まないし，繰返しもある．このような，複数の工程にわたる非同期な実行を支援する．

プロジェクト管理でしばしば問題なのは，このような管理水準が明確でないため，何が管理できて何が管理できていないかが分からないことである．

3.2 トータルコストダウン

トータルコストダウンとは，プロジェクトごとの総開発コストを削減する方法の体系である．プロセスに基づき算定された工程ごとの標準コストを基準として，従来よりも木目細かく開発コストを管理し，かつ低減する．詳細は文献2)を参照願いたい．

3.3 あゆみ活動

あゆみ活動とは，プロダクトの品質に関する定量的データに基づき，品質向上の観点からプロセスを改善する方法である．筆者らの開発プロジェクトもあゆみ活動に沿って開発している．詳細は文献9)を参照願いたい．

4. SPRの事例

SPRのいくつかの事例を**表-2**に示す．

5. SPRの評価

SPRの評価を下すことは現時点では難しいが，筆者らのプロジェクトでは次のような点からソフトウェア開発の基礎技術として捉えている．

(1) 分散並行開発の実現：分散並行開発により，開発サイクルが従来の1年から3カ月に短縮した．分散並行開発を実行する鍵はプロセスの正確な把握と実行の管理にある．実現の過程でプロセスの構造が重要であることが理解され，実現するための基盤技術となった．

(2) プロセス中心思考の定着：プロジェクト内で問題が発生したとき，常に，プロセスへのフィードバックが「自然に」議論されるようになった．これは，個人の努力やAd hocな取組みを越えた構造的な改善を促す．

(3) プロセスに基づく管理：従来のプロジェクト管理は丼勘定の感を否めない．プロセスに基づき，個人からチーム，プロジェクトに至る各レベルで一貫した管理が実現した．

さらに，**表-2**に示した事例では，生産性，総コスト，品質の顕著な向上が報告されている．特

表-2 SPRの事例

改善実施組織	主な改善成果	時期	改善方法
Hughes[13] 組込み型システム開発 (500人)	コスト削減 $200万/年 (予算超過率：6%－>3%) ROI>5 CMM Level 2－>3	88年～ 90年	・定量的プロセス管理 ・プロセス改善チーム ・仕様決定への参画・教育訓練 ・品質保証の強化・レビュー改善
Raytheon[7] 組込み型システム開発 (400人)	コスト削減 $1,580万/4年 (30%削減) ROI=7.7 生産性向上=2.3倍 CMM Level 1 ->3	88年～ 92年	・プロセス改善プロセスの確立 ・プロセスの標準化 ・再作業の削減 (Crosbyの方法) ・レビューの形式化 ・仕様の早期確定
Motorola[6] 携帯電話システム (1,000人以上)	CMM Level 1－>2	92年～ 93年	・組織的な改善への取組み ・部門内でのプロセス評価プロセス改善プロセスの定義と進捗評価
Hewlett-Packard[10] ソフトウェア部門 (3,500人)	コスト削減 $2,150万/年 (93年)	76年～ 93年	・インスペクションによる再作業の削減

に，米国では，プロセス改善の評価尺度として，総コストやROIなど企業経営の観点からの定量的尺度が導入されるようになった．これは，生産性向上などの成果を企業経営と関連づけて理解を促す点で有効である．一方，このような評価方法や評価の仕組みが十分確立していない点に留意する必要もある．

6. 今後の課題

SPRを実現するため，理論と実際の両面から研究，開発，適用，評価を行う必要がある．

理論面では，SPRの方法論を開発する必要がある．すなわち，図-1のSPRの各プロセスにおける方法の研究，開発，適用評価が必要である．

実際面では，CMMの普及とあいまって，米国で活動が活発である．BPRと同様，成功事例ばかりではないが，優れた成果も得られている．我が国でも，SPRの実践を推進する必要がある．

7. まとめ：ソフトウェアプロセス工学の提案

ソフトウェアプロセスを体系的に改善する枠組みをSPRの概念で整理し，その現状と動向を，筆者らの経験と事例に基づき紹介した．

SPRは本稿で初めて用いたが，その基礎であるプロセスの概念は，BPRやTQCとも共通しており，新しい概念ではない．しかし，SPRは次の2点でソフトウェア工学に新しい知見をもたらすと期待される．

(1) プロセスの形式的表現方法の提供：個人の創造的活動といわれている開発活動の背後にある構造をプロセスとして形式的に表現する．

(2) 形式的に表現されたプロセスに基づき工学的で体系的な方法論によって改善を可能とする．

また，SPRの研究，開発では，ソフトウェアプロセスにとどまらず，BPRやグループウェアなどのオフィス作業のプロセス技術，トヨタ生産方式に代表されるハードウェア製造のプロセス技術，さらにはハードウェアから始まりソフトウェア開発にも応用されつつあるコンカレント工学などの関連技術から学ぶ点も多い．

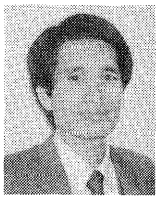
今後，このような基盤技術を活かし，ソフトウェアプロセスの設計，改善の方法論を**ソフトウェアプロセス工学(Software Process Engineering)**として体系化することを提唱したい．

参考文献

1) Aoyama, M.: Distributed Concurrent Development of Software Systems: An Object-Oriented Process Model, Proc. IEEE COMPSAC '90, pp. 330-337 (1990).
2) 青山幹雄ほか：ソフトウェア開発のコストダウンモデルとその適用，情報処理学会ソフトウェア工学研究会，No. 92-SE-84-5 (1992).
3) Aoyama, M.: Concurrent-Development Process Model, IEEE Software, Vol. 10, No. 4, pp. 46-55 (1993).
4) 青山幹雄：People Meet Process：開発プロセスとの出会いと対峙，ソフトウェア技術者協会ソフトウェアシンポジウム'94論文集，pp. 54-62 (1994).
5) Curtis, B. et al.: Process Modeling, CACM,

Vol. 35, No. 9, pp. 75-90 (1992).
6) Daskalantonakis, M. K.: Achieving Higher SEL Levels, IEEE Software, Vol. 11, No. 4, pp. 17-24 (1994).
7) Dion, R.: Process Improvement and the Corporate Balance Sheet, IEEE Software, Vol. 10, No. 4, pp. 28-35 (1993).
8) Fritsch, J.: The Motorola Software Engineering Benchmark Program: Organization, Directions, and Results, Proc. IEEE COMPSAC '93, pp. 284-290 (1993).
9) 富士通通信ソフトウェア開発部(編)：富士通における「あゆみ」活動，日科技連(1992).
10) Grady, R. B. and Slack, T. V.: Key Lessons in Achieving Widespread Inspection Use, IEEE Software, Vol. 11, No. 4, pp. 46-57 (1994).
11) Hammer, M. and Champy, J.: Reengineering the Coorporation, Harper Business (1993)[野中郁次郎(監訳)，リエンジニアリング革命，日本経済新聞社(1993)].
12) Humphrey, W. S.: Managing the Software Process, Addison Wesley (1989)[藤野喜一(監訳)，ソフトウェアプロセス成熟度の改善，日科技連(1991)].
13) Humphrey, W. S. et al.: Software Process Improvement at Hughes Aircraft, IEEE Software, Vol. 8, No. 4, pp. 11-23 (1991).
14) Mi, P. and Scacchi, W.: Process Integration in CASE Environments, IEEE Software, Vol. 9, No. 2, pp. 45-53 (1992).
15) 大野耐一：トヨタ生産方式，ダイヤモンド社(1978).
16) Osterweil, L.: Software Processes are Software Too, Proc. 9th ICSE, pp. 2-13 (1987).
17) Perry, D. E. et al.: People, Organizations, and Process Improvement, IEEE Software, Vol. 11, No. 4, pp. 36-45 (1994).
18) 山本里枝子他：ソフトウェアプロセスに基づくソフトウェア開発環境の検討，情報処理学会「ソフトウェアプロセス・シンポジウム」論文集，pp. 1-10 (1994).

(平成6年8月29日受付)



青山 幹雄(正会員)

1980年岡山大学大学院工学研究科修士課程修了．同年富士通(株)入社．分散処理ソフトウェアシステムの開発方法，開発支援環境，ソフトウェアプロセスなどの開発と適用に従事．この間，1986～88年米国イリノイ大学客員研究員．1995年4月より新潟工科大学情報電子工学科教授．ソフトウェア開発方法論，開発支援環境，ソフトウェアプロセスなどに興味を持つ．1993年より本会ソフトウェア工学研究幹事．1991～94年IEEE Softwareの Editor, IEEE COMPSACなどのプログラム委員．1993年情報処理学会研究賞受賞．編著書「オブジェクト指向分析・設計」(共著)，「Encyclopedia of Software Engineering」(共著)．電子情報通信学会，ソフトウェア科学会，ソフトウェア技術者協会，IEEE，ACMなど各会員．