オープン環境におけるシステム開発成功のために

For a Success to a System Development at Open Computing Environments

丸 山　修

要　約　オープン環境でのシステム開発の特徴として，従来以上の潜在的顕在的リスクが内在することがあげられる．このリスクに効果的に対処し開発を成功させるためには，管理プロセスを体系化して計画に重点を置いたプロジェクト・マネジメントの導入を図ると同時に，リスク駆動型の開発プロセスを採用することが必要である．管理と開発のプロセスを統合することは，システム開発の成功確率を増すばかりでなく，生産性向上にも寄与する．最後に生産性の一般論に触れる．

Abstract　What the existence of potential and/or actual risk is inherent in more than ever as a characteristic of system development in open environment is raised. In order to cope with this risk effectively and to success in development, it is important to adopt a development process of risk driven type at the same time to organize a management process, and to plan introduction of project management that put an important point with a plan. To integrate a process of development with management contributes to productivity improvement in addition to increase success probability of system development. Finally, this paper mentions the general consideration of software productivity

1. はじめに

現在，情報技術は企業の経営を変革する重要な要素となっており，インターネットと e-Commerce はあらゆる企業のビジネス・インフラに影響を与えている．インターネット上での新たなビジネスモデルの開発が競われ，従来にないビジネス・スピードが要求されている．世界経済はますますソフトウェアに依存するようになってきている．

一方で現在のコンピューティング環境はそのアーキテクチャや技術，製品の多様性において格段に複雑化している．それはオープン化に伴う情報処理産業の水平分散型ビジネスモデルが齎した選択の多様性である．メインフレーム中心の時代の均質な環境では，アーキテクチャやプロダクトはメインフレーマが提供し，またミドルウェアやオペレーティング・システムに至るまでソースコード・レベルの当事者能力を持つことは容易であった．しかしオープン環境ではそのようなことは難しくなっている．幅広い選択肢の中で，最適なアーキテクチャを設計しプロダクトの組み合わせを決定してソリューションを構築する必要があるが，従来のシステムを新しい技術で進化させようとすると，技術や組織上の問題に直面する．ビジネスの世界が，ビジネス・スピードに追従できる開発スピードを求め，同時に生産性と品質の向上を弛むことなく要求することも，問題を複雑化する．従来以上のリスクの中で，ミッションクリティカルなシステムを，より速く，より安く，開発することを求められている……というのが今，我々の置かれている状況である．

昔も今も，システム開発を成功させるためには，システムを開発する「システム」

をうまくデザインし実装することが重要である．「開発チームの組織化」，「計画の策定」，「管理と統制の仕組み」，「開発方法と技術の選定」，「標準化」等々である．はじめにそれがうまく作られていれば，システム開発の失敗する確率は減少する．要求に応えるためには，経験や勘，リーダの個人的スキルや偶然集まった優秀なチームメンバに頼るだけでは済まない．プロジェクト・マネジメントの重要性は従来にも増し高まっている．プロセス・エンジニアリング的アプローチと開発方法の近代化，新世代技術の取り込みが必須である．

2. システム開発における管理と開発のプロセス

2.1 プロジェクト・マネジメントへの取組みの進展

プロジェクトは，「独自の成果物やサービスを創出するための有期活動」として定義される．明確なビジネスの目標と成果を持ち，スコープ（役務範囲と責任），スケジュール（開始日と終了日），リソース（人件費を含む費用と人的，物的資源）が明示された中で，それを達成する一連の作業である．欧米では早くから，建設・エンジニアリング業界を中心に体系的なプロジェクト・マネジメントの手法を整備してきた．即ち，要求の把握と具体的成果物の定義，計画策定とそれに基づくプロジェクトの実行，個々の作業の決定と役割分担の明確化，合理的な作業ネットワークとスケジュールの作成，リスク分析に基づく対策，予算管理などである．

米国では，全産業分野を対象にしたプロジェクト・マネジメントの職能団体であるPMI（Project Management Institute，1969 年設立）[1]を中心にプロジェクト・マネジメントの専門性の確立と発展に取り組んでおり，その基礎プロセスの明確化と知識体系を整備した PMBOK（Project Management Body of Knowledge）の公刊とそれに基づく PMP（Project Management Professional）認定を実施している．PMI は現在世界中に 5 万人以上の会員を擁しており，また PMBOK の内容は，1999 年に ANSI 規格として認定されている．

日本においても，プロジェクト・マネジメントの重要性が認識されるようになり，産学官での動きが活発化している．1998 年に(財)先端建設技術センターが事務局となり，PMI 日本支部を設立した．また同年，(財)エンジニアリング振興協会は日本プロジェクト・マネジメント・フォーラム（JPMF）を設立し，建設・エンジニアリング，IT，その他の三つのグループを中心に，情報共有などの活動を開始している．大学では，1997 年千葉工業大学にプロジェクト・マネジメント学科が設立され，また 1999 年にはプロジェクト・マネジメント学会が発足した（世界初）．一方，通産省でも，プロジェクト・マネジャの資格制度の検討に入っている．

2.2 システム開発におけるプロジェクト・マネジメント

システム開発の失敗の第一原因は，プロジェクト・マネジメントにあるとも言われている．このような状況を脱却し，従来以上に多くのリスクの中で，システム開発を成功させるためにも，プロジェクト・マネジメントの重要性を認識する必要がある．計画に重点をおき，管理プロセスと実行プロセスを統合する必要がある．

日本ユニシスのプロジェクト・マネジメント標準は，インフォメーション・サービスの業務標準である TEAMmethod の中で，TEAMprogram として体系化されてい

る．これは，PMI の制定した PMBOK に準拠し，加えて実運用上のノウハウを取り込んだものとなっている．さらに，日本ユニシスではこれを前提に，社内のプロジェクト・マネジャ資格認定制度を包括的な技術認定の一環として運用している．PMP 資格試験もしくは通産省情報処理技術者 PM 試験に合格することが前提条件で，加えてプロジェクトを二つ以上成功させること，面談による実績認定を受けることなどが認定を受ける条件になる．

プロジェクト・マネジメントの実際については，「統合的プロジェクトマネジメントのアプローチ事例」（井上隆）他本誌各論文に詳述されている．

2.3 開発プロセスの変革

短縮された工期，生産性と品質の向上，新技術や多岐に亙る技術コンポーネントの組み合わせの中で，システム開発を成功するためには，管理プロセスを体系化しそれに基づくプロジェクト実行と併せて，プロジェクトの進むべき道筋を明らかにし成功に導く開発方法，開発プロセスを持つことが重要である．また管理プロセスと開発プロセスは密接に連携したものでなければならない．

現在でも多くのプロジェクトで採用されている開発モデルであるウォータフォール・モデルには，理論面実践面から問題点がいくつか指摘されている．即ち，ウォータフォール・モデルは，

- ■開発の初期段階で正確な要求定義を実施し，固定する．開発ライフサイクルの間，要求事項は変わらないとの仮定のもとに逐次ドキュメンテーションと実装を実施する．しかし，現実世界でこのような条件が成立することは希で，プロジェクトは要求リスクにさらされ，初期からの紙ベースの成果物は打撃を受けることになる．
- ■初期に作られる紙ベースの成果物を重視する結果，リスクの解消が遅れ先送りされる．開発終盤の統合フェーズで初めて問題点が露呈し，設計が破綻するという最悪の事態を生じることもある．少なくとも，問題解決のために統合フェーズは長引く以上，参考文献[2]を参照）．

このような問題点を解消するリスク駆動型の開発プロセスに，Rational 社の開発プロセス・フレームワーク RUP（Rational Unified Process）[3]がある．RUP の特徴は，アーキテクチャ中心，追加的反復的開発，ユースケース主導開発により，リスクの早期発見と解消を目指す点である．RUP と共通する特徴を持つ開発プロセスをアーキテクチャ中心の開発プロセスと呼ぶ．

日本ユニシスのコンポーネント・ベースの開発方法である LUCINA は，アーキテクチャ中心の開発プロセスを特徴としている．加えて LUCINA では，アーキテクチャ・パターンとプロダクトセットを複数のソリューション・モデルに応じて実証評価した上で事前定義しており，実証済みアーキテクチャを再利用することにより，さらにリスクを軽減する．LUCINA は，事前定義アーキテクチャを再利用する時，最大の効率を得られるように設計されているが，アーキテクチャ構想から開発を始める場合にも十分効果的である．

開発プロセスと LUCINA については，本誌「青山学院バーチャル・キャンパス基盤システムの構築」（羽田昭裕）に詳述されている．

3. システム開発における生産性

管理と開発のプロセスを整備統合する目的は，システム開発を成功させる確率を高めることと同時に，生産性を向上することにある．最後に，本誌各論文であまり触れられていない生産性について若干述べたいと思う．

3.1 生産性マネジメント

システム開発における生産性を恒常的に向上させようとする時，生産性マネジメントが必須になる．生産性マネジメントは次の 5 項目で構成される．

① 生産性の定量的指標明確化
② 他社とのベンチマーキング
③ 生産性向上プログラムの策定と実施
④ 改善効果の定期的測定と評価
⑤ 生産性，見積もりナレッジ・データベースの構築

3.2 生産性に影響を与える因子（生産性因子）と生産性指標

システム開発における生産性を客観的に評価するためには，生産性を表わすエンジニアリング面での尺度が必要である．それによって我々は過去から現在に至る生産性の向上を客観的に測定でき,今後の生産性向上プログラムを進めることが可能となる．生産性指標としては，プロジェクト P/L のような財務メトリクスがある．これはシステム開発をビジネスの観点で捕らえるときの重要なメトリクスであるが，パラメタとして「売り値」を含むため，今我々が求めるものではない（高すぎる売り値，あるいは安すぎる売り値によって，生産性が左右される．）

生産性因子（図 1）には被開発システム（開発対象のシステム）と開発システム（開発対象システムを開発する「システム」）それぞれに起因するものがある．一般的には被開発システムに起因する生産性因子は開発の前提条件である場合が多く，我々にコントロール可能なのは開発システム側の生産性因子である．生産性因子は多岐にわたり，一つ一つの影響度合いが複雑に入り組む．これらは開発の結果を直接的，一意的に表現できないという理由で生産性指標としては不適切である．生産性指標は開発の結果である①「開発工数（どれだけの工数をかけて)」，②「開発期間（どれだけの期間で)」，③「生産物量（どれだけのものをつくったのか)」で表現するのが妥当と考える．(生産性因子を定量的に評価して開発工数と期間を見積もる手法に COCOMO II のソフトウェア・コストモデルがある．これについては後述する．)

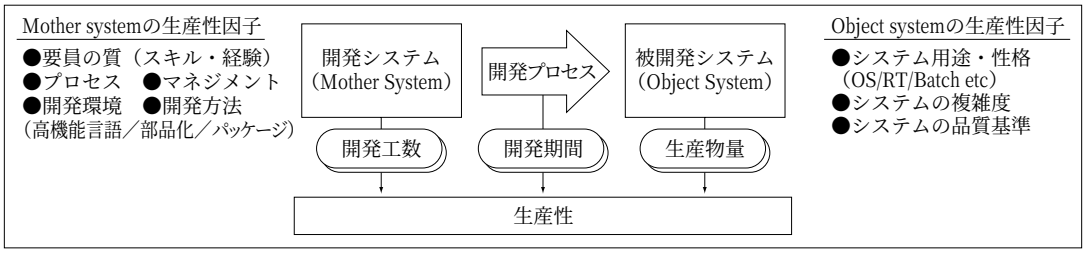

図 1　生産性因子

3.3　静的生産性指標

従来よく使われる生産性指標に，単位工数当りの生産物量（FP（Function Point）数/人月，SLOC（Source line of Code)/人月）があります．時間の要素が欠落しているところからこれを静的生産性（指標）と呼ぶ．静的生産性は，前節で述べた三つの開発結果パラメタ中の「開発期間」を考慮していないことに注意が必要である．

静的生産性は，開発規模や開発期間に影響される．一般的に次のことが知られている．

■開発規模が増大すると開発期間，開発工数ともに増大し，工数の増大率は期間の増大率より大きくまたぶれも大きい．静的生産性は減少する．

図 2 はこの様子を示している．日本ユニシスの手がけた過去 3 年間の 73 プロジェクトを，横軸に開発規模（KSLOC)，縦軸に開発工数（人月）と開発期間（月数）をそれぞれ対数目盛りでプロットしてある．

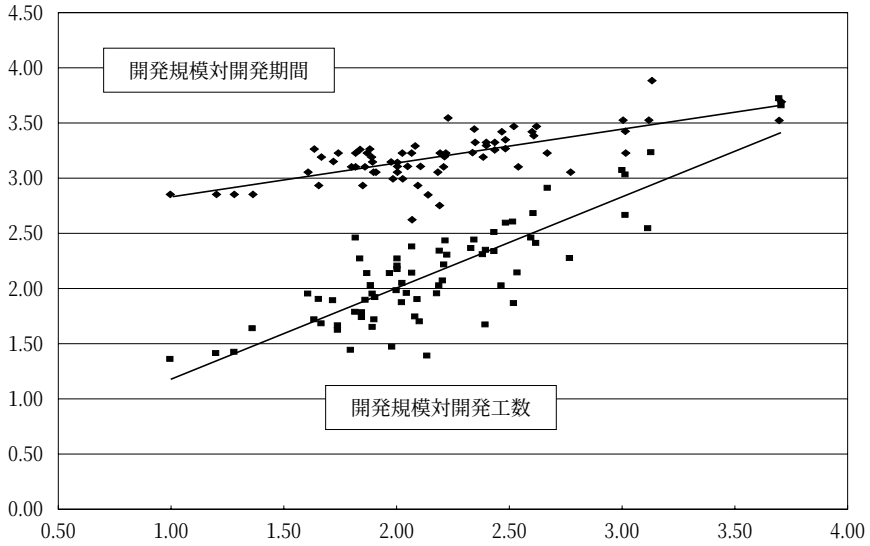

図 2　開発規模と開発期間，工数の関係

また，次のことも多くの研究（及び経験）から知られている．

■開発期間と開発工数の間に単純なトレードオフは存在しない（要員を倍にしても，開発期間は半分にならない)．開発期間を短縮すると開発工数は増大し，静的生産性は減少する．

我々は静的生産性指標について，次の理由で不十分と考える．

①開発規模や開発期間の異なる二つのプロジェクトの生産性を客観的に比較できない．

②開発期間を考慮していないため，開発スピードを評価できない．

新しい生産性指標が必要である．しかし，静的生産性指標は直感的に理解しやすいという理由から，我々はそれを，使用条件を明確にした上で併用する．

3.4　新たな生産性指標―動的生産性指標―

プロジェクトに対する要員投入のパターンは，初期の少数のキーメンバから始まり，問題の詳細化に伴って増加していき，システムが完成に近づくころ最大要員数となり以降徐々に減少していく．開発期間を短縮しようとするならば，要員投入のピークは前倒しされることになるが，むやみに要員を投入できるものではない．各時点での問題に応じて適切なスキルを持つ要員が投入される必要があり，要員が急激に増加してもそのオーバヘッドを受け止められるだけの組織の成熟度を開発チームが保有していることが条件になる．プログラミングの標準化もできていないところに大量のプログラマを投入しても非効率なことは容易に理解できる．

図 3 は，日本ユニシスの 73 プロジェクトについて開発期間と開発工数の関係から要員投入のパターンを，開発スピードにおいて 1（遅い）から 6（急速）までに 6 分類したものである．

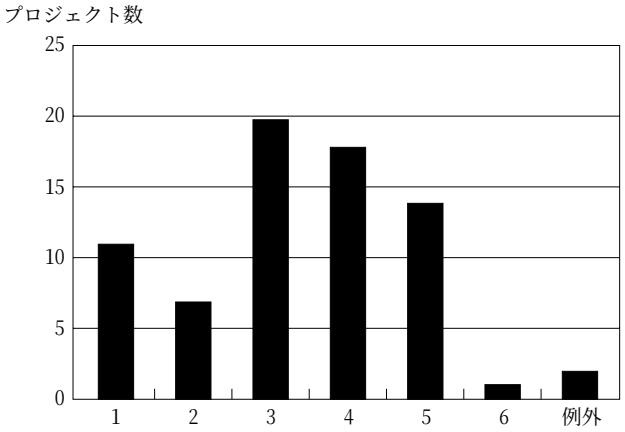

図 3　開発スピード別プロジェクト数

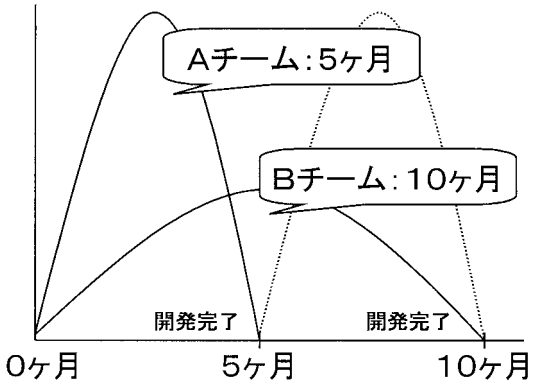

図 4　開発リソースの回転率

ビジネス・スピードは速くなっており，追従できる開発スピードがないと機会損失を生じる．開発リソースの回転率（図 4）をあげられる組織はより多くのビジネスを

獲得できるので，生産性は相対的によいと見ることができる．次の二つのプロジェクトを比較してみる．どちらも 50 人月で 10 万ステップの同一のシステムを開発したとする．A チームは 5 ヶ月で完了し，B チームは 10 ヶ月かかったとする．A チームは B チームに対して 2 倍の回転率である．A チームは 10 ヶ月の間にもう一つ 50 人月の仕事ができるのに対し，B チームは機会損失を起こす．

静的生産性はどちらも 2000 ステップ/人月で同じ値となっている．我々には，「A チームのほうが B チームより生産性が高い」と表現できる生産性指標が必要である．それが動的生産性指標である．動的生産性指標は，従来の静的生産性と開発スピードを総合的に評価する．開発スピードが同じならば，静的生産性が高いほうが動的生産性は高くなり，静的生産性が同じならば，開発スピードが速いほど動的生産性は高くなる．前途したように静的生産性指標には「開発期間」が欠落している．動的生産性指標は次の式で表わすことができる．

■動的生産性＝f（生産物量，開発工数，開発期間）

3.5 見 積 り

開発規模，開発工数，開発期間を，精度よく見積もることが要求される．今日，広く普及し,研究と実績データを持ち,明確にドキュメント化されているソフトウェア・コストモデルは，1981 年に Boehm が開発した構成的コストモデル COCOMO（COnstuctive COst MOdel）[4]である．COCOMO II は，COCOMO を，USC ソフトウェア工学研究センタを中心に業界関連企業（EDS，AT & T，Bellcore など 28 社）からなるプロジェクトが改定したもので，1997 年に第 1 版，以降毎年改定されており現在 1999 年版が制定されている．改定の目的は以下の 3 点である．

① 1990 年代から 2000 年代のソフトウェア開発に適用可能なソフトウェア・コストとスケジュールの見積もりモデルを開発する．

② ソフトウェア・コスト・データベースと，コストモデル改善のためのツールを開発する．

③ ソフトウェア技術とその経済効果を評価するための定量分析の枠組みを提供する．

COCOMO II では，ソフトウェア市場を五つのセクタ（エンドユーザ・プログラマ，コンポーネント開発者，コンポーネント統合作業者，システム・インテグレータ，インフラストラクチャ開発者）に分け，見積りの詳細化に応じて三つの見積りモデル（プロトタイピング，初期設計，ポストアーキテクチャ）を提供する．反復型（追加型）開発やコンポーネント利用開発，新しい管理プロセス，新世代の技術にも適合する．

ポストアーキテクチャ・モデルの基本式は次の通りである．

■$PM = 2.94 (Size)^{B} \prod_{i=1}^{17} EM_i$

PM：工数（人月）
Size : KSLOC
B：プロセス指数
EM：工数補正要素（17 種類）

ここで，17 種類の生産性因子が工数補正要素として使われる．また，B はプロセス指数と呼ばれ，B＞1 のとき開発規模が倍なら工数は倍以上となる度合いを表わし，B＝1 の場合は開発規模と工数の関係は線形となる．B＜1 の場合を COCOMO II で

は扱っていないが，コンポーネント再利用が大規模に進展することによりスケールメリットの達成が可能となるかも知れない．プロセス指数は開発期間の見積もり式にも現れ，プロセス指数の改善は工数と期間の双方に有効となる．尚，プロセス指数は五つのスケール・ファクタ（アプリケーション経験の有無，プロセスの柔軟性，アーキテクチャ上のリスクの解消度合い，チームのまとまり，SEI の成熟度モデルで定義されるプロセス成熟度）の組み合わせで定義される．開発規模については，ソフトウェア・ソリューションの構造が比較的不明な開発の初期フェーズでの見積りにはファンクションポイント法が，ポスト・アーキテクチャ・モデルではソースコード行数（SLOC）による見積りが推奨されている．

図 5 は日本ユニシスにおける一般的な見積りプロセスを表している．

COCOMO II は，これからの発展が期待できる注目すべき技術であると考える．

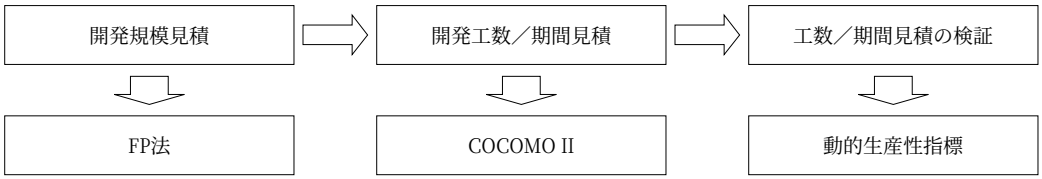

図 5　開発規模と開発工数/期間の見積り

3.6　生産性向上プログラム

我々は，今の開発プロジェクトを成功させ，かつ常に生産性向上にチャレンジしていく必要がある．そのためには生産性向上プログラムを策定し，組織的に実行していかなければならない．生産性向上プログラムには次のものが含まれる．

① 生産性阻害要因の除去

② プロジェクト・マネジメント力の強化と組織の SEI プロセス成熟度の向上

③ コンポーネント指向開発，反復型開発などの開発方法，環境，ツールの整備と新世代技術の採用

④ 要員の質（技術，スキル，経験）の向上

生産性向上プログラムでは，多元的多面的アクションが必要であり，個別のアクションの効果がすぐさま個別の測定結果として現れるわけではない．継続的に組織的にアプローチすることが重要になる．

4.　お　わ　り　に

日本ユニシスでは，TEAMmethod とコンポーネント・ベースの開発技法 LUCINA に基づいて管理と開発のプロセスを統合し，プロジェクト・マネジャとプロジェクトの実行を多方面から支援するプロジェクト支援オフィス（その機能として生産性マネジメント・オフィスも含まれる）を設置して，リスクの多いオープン環境でのシステム開発を成功させるよう取り組んでいる．その成果の一部を本技術報告としてご報告させていただく．皆様のお役に立てれば幸いである．

参考文献［1］ http : //www.pmi.org/
［2］ W. E. Royce,“ Software Project Management : A Unified Framework ”, Addison-Wesley, 1998［邦訳：日本ラショナルソフトウェア監訳, ソフトウェアプロジェクト管理, アジソン・ウェスレイ, 1999］
［3］ P. Kruchten,“ The Rational Unified Process ”, Addison-Wesley, 1998［邦訳：藤井拓監訳, ラショナル統一プロセス入門, ピアソン・エデュケーション, 1999］
［4］ http : //sunset.usc.edu/COCOMOII/cocomo.html

執筆者紹介 丸　山　　修（Osamu Maruyama）
1951 年生．1974 年京都大学理学部卒業．同年日本ユニシス(株)入社．大規模オンライントランザクション処理システムの開発，トランザクション処理ミドルウェア製品の企画開発，電力ユーザサービス等を経て，1999 年生産技術部長．現在，E ビジネス技術部長．