組込みシステムのためのソフトウェア開発プロセス

―技法・記法・作法・方法・商法―

株式会社エクイティ・リサーチ　取締役　プリンシパルコンサルタント

大槻　繁

ZIPC に代表されるツールを開発の局面で有効に利用して行くためには、それを取り巻く諸々の思想面、人的・組織的な側面をも考慮して行く必要があります。ここでは、組込みシステムを開発する際に有効なソフトウェア開発技法も視野に入れつつ、その開発手順や考え方について論じます。さらに、開発プロセスやマネジメントに関する技術の基本的な動向についても紹介し、今後の、本分野での開発現場のあるべき姿を描き出して見ることにしましょう。

1．はじめに

ソフトウェアエンジニアリングの歴史は､人間の飽くなき欲望を満たす製品を、その時々の技術レベルに応じて、適切な方法で開発・供給するという戦いの歴史であると言えます。大まかな概要を図１に示します。1970 年代の構造化プログラミングや複合設計法の提唱に始り、1980 年代から普及・発展してきたオブジェクト指向開発技法、さらには､1990 年代に入り､コンポーネント技術や Java を中心としたネットワーク分野を見すえた開発技術が台頭してきていると言えるでしょう。この潮流は、ソフトウェアというものが、ますます社会的な意味で重要な位置づけになってきていて、開発技術に対する要請も、より広く、かつ、高度になってきていると言えるでしょう。

一方、日本の産業界の強みとして有望視されている組込みシステムの分野においても、ソフトウェアエンジニアリングの重要性がますます重要になってきています。今までは、数名の熟練したエンジニアで、職人芸的に開発してきた製品も、今や、規模も大きく、製品も多様化してきたために、系統的な開発技術なしでは対応しきれなくなってきています。組込みシステムの難しさは、ハードウェアとの複合体であること、実時間性（リアルタイム）が高いこと、要求品質が高いこと等に起因しています。

2．高品質化の要求

ソフトウェア技術そのものが、通信・並行プロセス・リアルタイム（実時間）

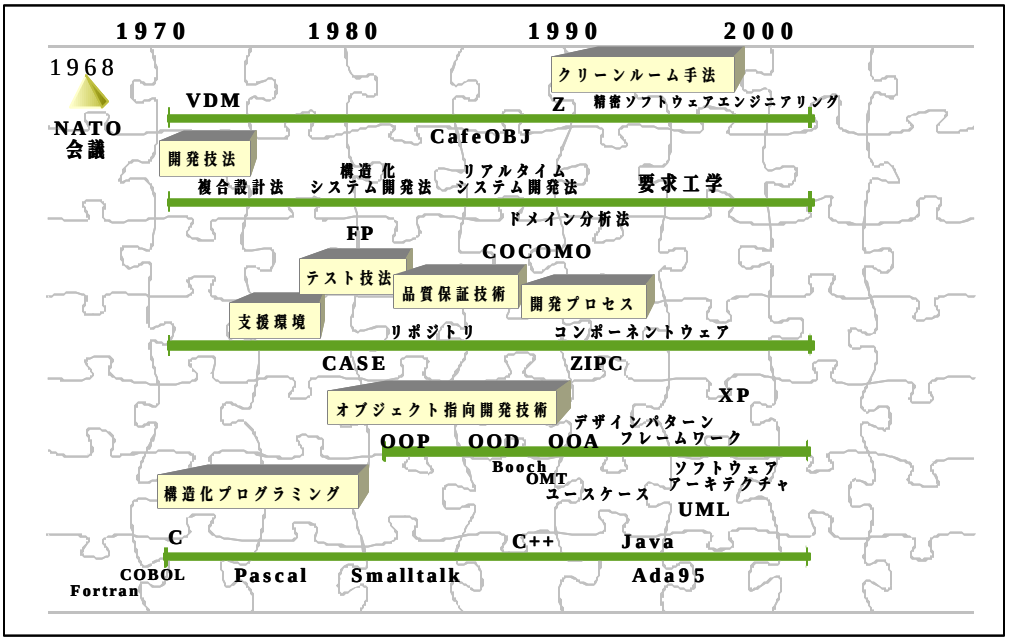

図１．ソフトウェアエンジニアリングの歴史

等を扱わなくてはならず、難易度を増してきていますが、それにも増して、ソフトウェアやシステムの社会的責任は、ますます大きくなってきています。組込みシステムの分野においてもミッションクリティカルやセーフティクリティカルと呼ばれるシステムが多くはなっています。アラバマに住む７歳の男の子は、シボレーのトラックの劣悪な燃料噴射制御のソフトのせいで命を奪われ、腫瘍治療用の低出力放射線制御ソフトの欠陥のせいで治癒可能な患者が死に至り、グァム行きの飛行機に乗っていたボーイング747の乗客200名以上の人々が、高度警報装置の劣悪なソフトのせいで亡くなるといった事例に見るように欠陥商品の蔓延は、かなり危機的な状況と言えるでしょう。欠陥が起きる社会的な要因は、専門家が欠陥システムを当たり前に認めていること、雑誌記者を始めとする業界をとりまく人々がバグを無視していること、そして、何よりも問題なのは、ソフトウェア開発会社が、製品が無欠陥であることが収益につながると思っていないということにあります。

技術的な要因は、要件定義、設計、プログラミングといったそれぞれの工程で存在します。前提条件の認識を誤り、想定されていない使い方がされたり、メモリ（リソース）の節約に起因する欠陥やアルゴリズムの漏れ抜け、さらには、ごく初歩的な構文エラーによる欠陥による

プログラミングエラーも散見されます。私たちは、ソフトウェアの本質的困難に正面から取り組むべきでありながらも、結局は、理論や原理の裏づけのない世界で、複雑で大きなものを、市場やユーザの心変わりに対応しながら、目に見えないソフトウェアを開発しなくてはならないのです。

3．開発技術の進展

開発技法の発展によって、次第に技術的な課題はある程度は解決されつつあると言えるでしょう。

データフロー型やオブジェクト指向型のモデルにリアルタイム性を付加する方法も多数提案されていますし、より理論的な側面でも、並行プロセスに関する体系も整備されてきています。

リアルタイム・組込みシステム向けの開発技術の典型的な発展の過程は、データフロー主体の方法論の譜系でみてとることができます。P. T. Ward 等によって提案された RTSA: Real-Time Structured Analysisは、従来のデータフローに加え、コントロールフローを付加したモデルを使っています。これをさらに発展させ、さまざまなプロセスの統合・分割規準や設計段階への詳細な変換方法を整備した技法がH. GomaaのCODARTS: Software Design Methods for Concurrent and Real-Time Systems です。

オブジェクト指向の分野でもさまざまな技法が提唱され、統一的な表記方法もUML: Unified Modeling Languageとして標準化も進められていますが、これにリアルタイム性を付加したものも B.P. Douglass 等によって提唱されています。H. Gomaa も、CODARTS を洗練化し、UML対応にした COMET: Concurrent Object Modeling and Architectural Design with UML を提案しています。

もう一つの重要な流れは、有限状態機械

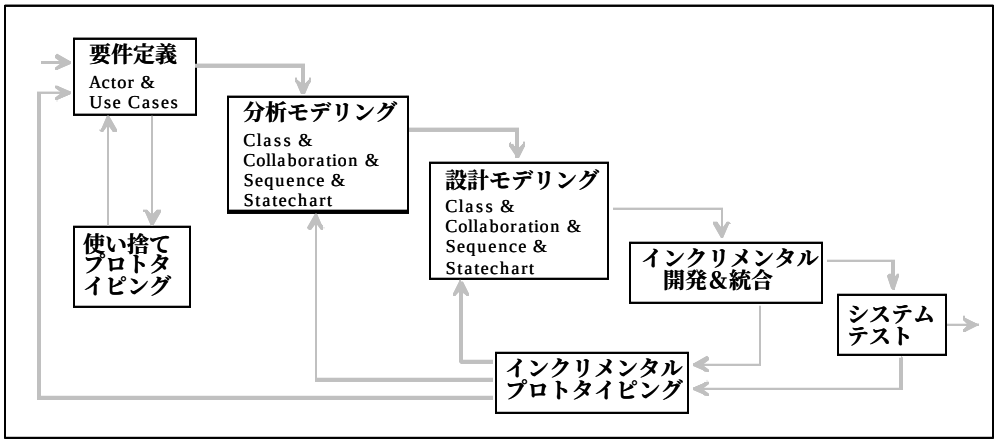

図２．COMETの開発プロセス

（状態遷移モデル）の活用の方法です。EHSTM: Enhanced Hierarchical State Transition Matrix は、従来の状態遷移モデルをソフトウェアエンジニアリング的な視点で拡張した有効な手法として位置づけることができます。

D. Harel の提唱した Statechart は、状態遷移モデルの状態数の爆発の問題を解決したばかりでなく、実用的な並行プロセスの概念も Broadcast 型の並行プロセスモデルという形で導入した有効な手法です。こういった、状態遷移モデルをベースにした並行プロセスの研究というのは、C.A.R Hore の CSP: Communicating Sequential Processes や、R. Milner の CCS: Communication and Conccurency System といった理論的な研究をベースとしつつも、実務領域との親和性も高いために、一つの主流となりつつあります。無論、技法というのは諸々のモデルやその表記方法を使った手順と、それぞれの工程での中間的な成果物の満たすべき規準とから構成されており、全体の開発プロセスの中で、どのように品質を作りこんで行くかがポイントとなります。図2に、COMET 技法の推奨する開発プロセスを示します。ここでは、特に、インクリメンタルな開発プロセスを採用して、近年の市場変動に迅速に対応できる方法になっています。

4. 品質と開発プロセス

システムやソフトウェアは、顧客やユーザの対象世界で稼動するものです。品質とは、顧客満足度という言葉に代表されるように、対象世界での価値を示しています。品質の体系については、図3に示されているような品質特性(quality characteristics)がまとめられています。ソフトウェアの品質向上については、技法や理論・原理的な側面のみならず、開発プロセスの視点が重要です。

信頼性の観点からは、従来の一般的な開発プロセスにおいて、テスト工程が全工程の工数の半分近くを占め、さらに、1KLOC 当りで数個～数十個の不良を含んだまま製品を出荷している場合もあると言

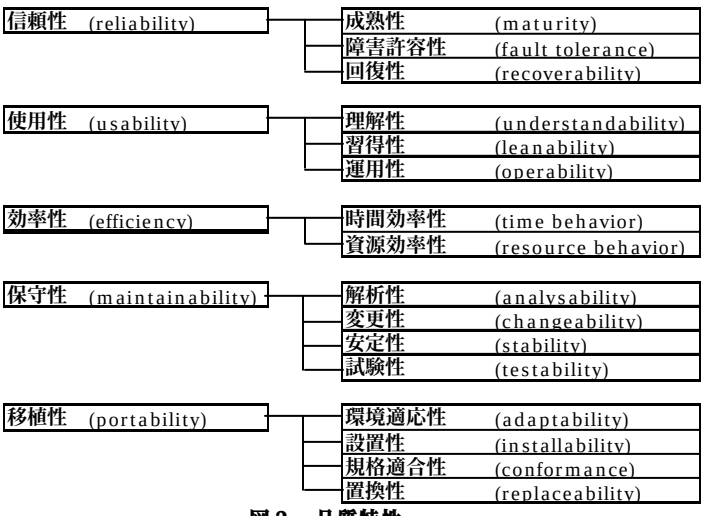

図3. 品質特性

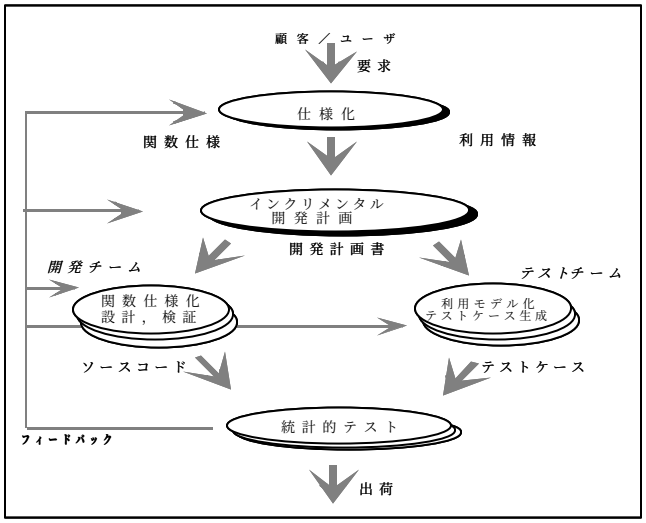

図４．クリーンルーム開発手法のプロセス

われています。また、初期段階で作り込まれた不良の発見が遅れる程、その修正にコストがかかるために、最初から不良を作り込まないように開発プロセスを管理・制御して行くことが求められています。

品質の作り込みのための効果的な方法は、系統的なレビュー技術を使うことです。ウォークスルーは、ソースコード等の中間的な成果物を実行させる前に、他人に説明しながら模擬的に実行させてメンバ間の合意を得ながらレビューを行なう方法です。この方法は、人間の心理や社会的な特性を旨く利用したもので不良の摘出に効果があります。模擬的な実行ではなく、コードの正当性を数学的な裏づけによってレビューを進めてゆく証明レビューを採用することもあります。インスペクションは、より公式な会合を利用した方法であり、会合の結果を報告書にまとめ、指摘事項の修正や、その後の修正結果の確認も行ないます。不良の結果を開発者にフィードバックし、不良の作り込みの予防も行なうことによって組織としての知識の蓄積を図って行く手法の一つと言えます。

テストとは、厳密に言えば、プログラムを実行させることによって、品質を評価することです。素朴な意味でのテストには、不良を見つけるためにプログラムを実行させることというものもありますが、これは開発者の論理であって、ソフトウェアそのものの特性としての品質とは異なります。テストは原理的にプログラムを対象としたものであって、ウォータフォール型の開発プロセスで言えば、要求定義、システム設計、プログラム設計、モジュール設計等の活動を通じて得られたプログラム実体を対象に行われます。それぞれの工程に対応して、その工程の正しさを確かめる詳細なテスト工程を対応させることもできます。最終的なプロダクトの特性としての信頼性を測定するために、ユーザの利用モデルを作成してテストを行ない、故障が起こる時間間隔を測定し、MTTF (Mean Time to Failure)を信頼性の値として採用する統計的テスト技術も提案されています。

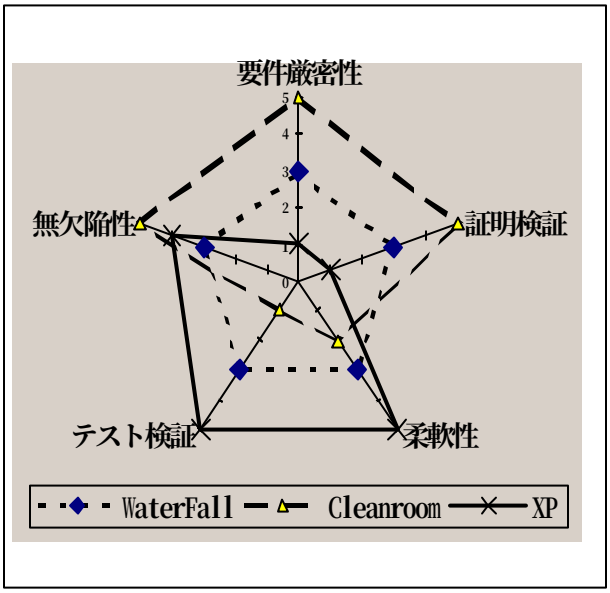

図５．開発プロセスの比較

最近では、テストをどのように開発プロセスの中で位置づけるかということが、一つのキーポイントになっています。例えば、ソフトウェアクリーンルーム手法では、図４に示すように、テストは、別のチームによって遂行され、かつ、そのテストも、ユニットテストや統合テストレベルのものではなく、あくまでもユーザの視点からのソフトウェアの品質特性を測定するという立場で行なうことが特徴となっています。この手法では、従来の内部テストに相当するものは、全て実現（プログラム）が仕様を満たすかどうかの証明によって行われます。

また、最近、話題になっている XP: extreme Programming（エクストリーム・プログラミング）では、逆に、仕様記述の代わりにテストを中心にプログラムの妥当性を保証する方法を採用しています。すなわち、インクリメンタルな開発プロセスを併用することにより、常に、実行可能なプログラムを保持し、そのプログラムの妥当性を、テストケースを流すことによって保証しています。

図５は、以上のような各種の開発プロセスの特徴を比較したものです。

５．マネジメントの必要性

以上、述べてきましたように、組込み製品を開発して行く際には、対象領域の問題を解決するための技術や実装上の意思決定だけではなく、より高レベルの、さらには、経営上の意思決定に結びつくような視点での諸々の知的活動が必要であると言えるでしょう。

図６に、ソフトウェアエンジニアリングを、製品を開発するという意味での社会的・経済的活動での捉え方（リファレンスモデル）を示します。マネジメントの一般的なサイクルと言われている PDCA (Plan- Do- Check- Action)は、いわゆる、ものづくり、工業製品での基本と言えます。建築業やプラントをはじめとする広範な分野でのマネジメントについては、米国 PMI (Project Management Institute)が、プロジェクト管理の知識体系を、統合(integration)、範囲(scope)、時間(time)、コスト(cost)、品質(quality)、組

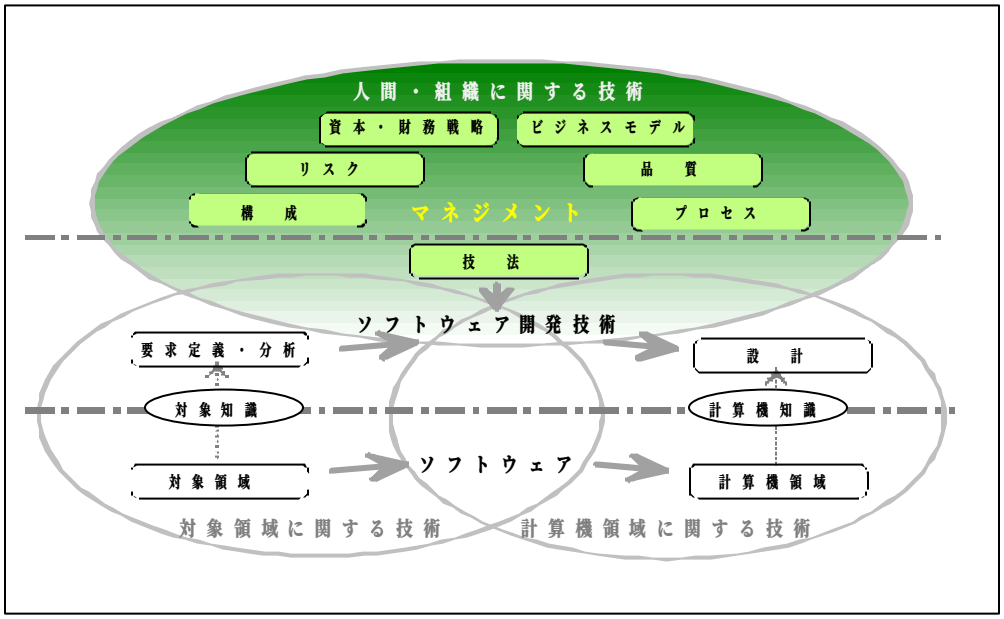

図6.ソフトウェアエンジニアリングのリファレンスモデル

織(organization)、コミュニケーション(communication)、リスク(risk)、調達(procurement)からなる枠組みでまとめています。こういった体系化された知識は、社会的な資格認定制度と連携して整備されて来ており、標準化もすすめられています。ISO9000の一環としてISO10006の品質管理の指針でも、PMIと同様の枠組みを採用しています。

ソフトウェアやシステムの開発においては、PMIのような一般的な枠組みだけではマネジメントすることはできません。ソフトウェア固有の本質的困難があるのです。すなわち、

・複雑性(Complexity)：大きいこと、複雑であること、それ自体が本質的な問題です。物理的な構成物とは異なり、単純な部品の組み合わせでは作ることができないのです。

・同調性(Conformity)：単純な原理（物理学で言う統一場理論のようなもの）は存在しません。(気まぐれな)人間の習慣や社会制度に順応させなくてはならないのです。

・可変性(Changeability)：使われれば、使われる程、機能拡張や新機能の要求が増えます。システムが社会に組み込まれるが故に、絶えず変化し続けなくてはならないのです。

・不可視性(Invisibility)：物理的な世界ではなく、概念の世界でしか捉えることができません。構造を抽象化して顕在化する手段がないのです。

プロジェクト管理で行なうべき事柄は、

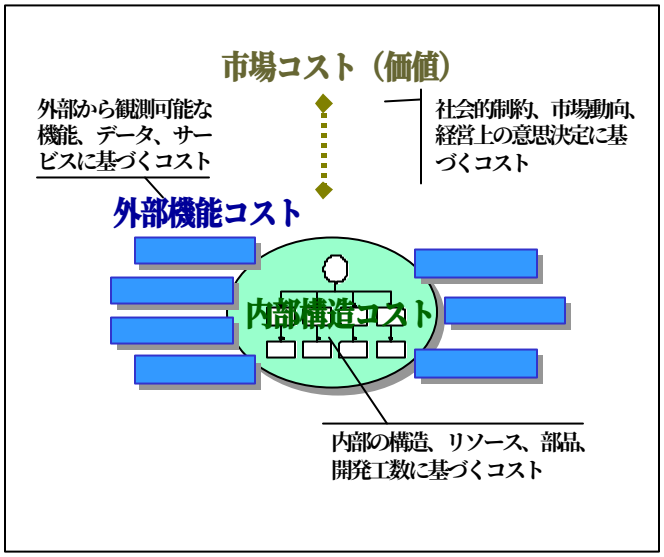

図7.コストの3つの側面

以下のようにまとめることができます。

(1) 計画：プロジェクトの目標設定やプロダクトの要求定義を行ないます。開発技法、開発環境、管理技術、構成管理手法、品質保証手法、文書体系、教育方法等を選択します。作業を抽出、分割し、スケジュールをたて、予算を割り当てます。リスクに関する評価を行ないます。

(2) 組織化：要求を分割し作業に分割します。組織やチーム構成を決定し、責務と権限を割り当てます。顧客、ベンダー、他の組織とのコミュニケーション方法を設定します。

(3) 人員割付け：組織の中に人員を配置します。メンバの導入、教育、トレーニングを行ないます。状況に応じて人員の再配置を行ないます。

(4) 指導・監督：リーダシップの発揮、メンバの監督、権威の利用、動機づけ、各作業の協調、コミュニケーションの円滑化、競合の解決等を行ないます。

(5) 制御：プロジェクトの予算、スケジュール、品質、生産性等の各種データを収集し、状況を把握し、必要に応じて制御を行ないます。

6．コスト予測

以上のように、プロジェクトマネジメントは、あたかも、ソフトウェアの開発を行なうように、開発プロセスそのものを計画し遂行して行くことに他なりません。このように考えると、ソフトウェア開発で最も重要な要件定義段階に相当する事項として、プロジェクトの計画段階が位置づけられます。

このプロジェクトの計画段階は、開発に関わる人々にとっては、経営上の意思決定にも直結しているわけですから、コストの問題が極めて重要なポイントになります。

しかしながら、現状で、明快な理論や原理に従っていて、かつ、実用的なコスト予測技術は、決定的に不足しています。この分野の研究・開発が遅れている一つの理由は、コスト予測という問題が、経営やビジネスの世界と、エンジニアリングに関する世界との両方に股がっていることに起因していると言っても過言ではありません。経営者は、システムや製品

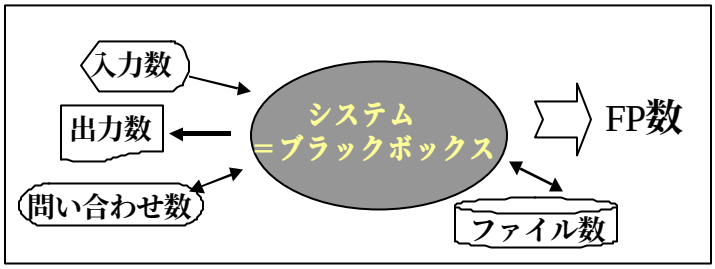

図８．ファンクションポイント法

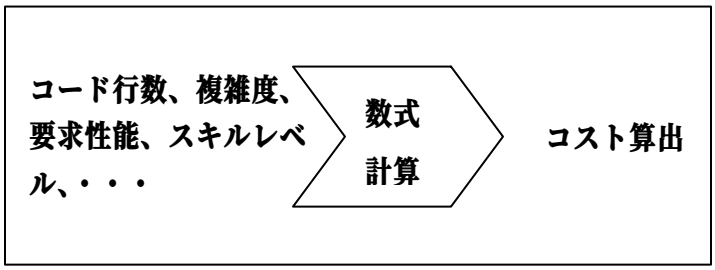

図９．回帰型構造的モデルCOCOMO

コストとは、製品を構成する部品、調達にかかる費用、ソフトウェアの開発費用等のいわゆるものの作りに関わるコストです。

本来は、コスト予測という時には、これ等を全て総合した視点でコストを決め、かつ、開発プロジェクトのプランをたてて行く必要があります。ソフトウェアを対象とした、上記のコスト予測に有効な開発に関する投資を、市場性や顧客との関係によって意思決定を行なうでしょうし、開発者（エンジニア）は、開発に関わる諸々の組織内部や環境、技術的な視点でのリソースに従って意思決定を行なっています。両者の間には、数多くのコミュニケーション上の課題が存在することになります。

コストの決定要因というのは、図７に示すように、全く独立な３つの側面があります。一つは、市場によって決まるコスト（価値）であり、これは、競合他社の製品価格や、経営上の戦略に基づく価格に相当します。外部機能コストというのは、製品の提供する外部観測可能な機能の評価コストです。そして、内部構造

いくつかのモデルが提唱されてきており、実績データも蓄積されつつあります。図８と図９にその代表的なモデルであるファンクションポイント法と回帰型構造的モデル(COCOMO: COnstructive COst Model)の概念を示します。COCOMO のようなモデルで、かなり正確な予測が可能になってきています。しかしながら、このようなモデルを使ったとしても、非常に多くのコスト要因がからんでいて、専門家の判断をあおがないと、なかなか正確な予測は難しいというのが現状です。よく言われるのは、プログラマの資質によって､生産性が 40 倍も異なる場合があるということです。これはソフトウェアの開発コストの内の大部分が人件費で占めら

Precedentedness	中	中	中	中	中	中
Dev. Flexibility	中	中	中	中	中	中
Arch./ Reis Resolution	中	中	中	中	中	中
Team Cohesion	中	中	中	中	高	高
Process Maturity	CMM2	CMM2	CMM2	CMM2	CMM5	CMM5
Analyst Capability	中	中	中	中	中	高
Application Experience	中	中	中	中	中	高
Programmer Capability	中	中	中	中	中	高
Platform Experience	中	中	中	中	中	高
Lang. And Tool Experience	中	中	中	中	中	高
Personnel Continuity	中	中	中	中	中	高
Use of Software Tools	中	中	高	高	高	高
Multisite Dev.	中	中	中	中	中	中
Dev. Schedule	中	中	中	短(18月→13.5月)	中	中
Execution Time Constraint	中	中	中	中	中	中
Main Storage Constraint	中	中	中	中	中	中
Platform Volatility	中	中	中	中	中	中
Required Reliability	中	高	高	高	高	高
Database Size	中	高	高	高	高	高
Product Complexity	中	高	高	高	高	高
Required Reusability	中	高	高	高	高	高
Documentation Match	中	高	高	高	高	高
<<COST>>	￥39,580,000	￥120,980,000	￥94,370,000	￥134,940,000	￥78,540,000	￥19,850,000
COST Ratio	100%	306%	238%	341%	198%	50%

図１０コスト変動の例（C言語、10KStep規模、再利用なし、1人月＝100万円）

れているという現状をみると、コスト予測が困難になっている要因の一つです。

ほんの一例ですが、図１０に、諸々のコスト要因を変化させた場合の、最終的な開発コストの比較を示しています。平均的な開発を想定して、それを１００として、システムの複雑度や、組織のスキルレベルが異なるとどの程度、最終的な開発コストの変動があるかを COCOMO モデルを使って計算したものです。

７．おわりに

成功するシステムやソフトウェア開発の要件は、適切な人員を技術や知識が高いレベルで揃え、適切な方法論やツールを揃え、作業の方法が正しく確立されていることです。プロセス改善モデルは、現状の組織のレベルを認識し、開発プロセスや組織の弱点を顕在化し、より上位のレベルに至るためのロードマップを提供しています。各方面でのプロセス成熟度モデルを統合化する試みもなされています。ISO-SPICE (Software Process Improvement and Capability dEtermination)でも SEI-CMM をベースに改訂を施し、ISO/IEC 12207 "Information Technology - Software life cycle processes"、さらに ISO 15504(SPICE)として詳細化され、まとめられています。

前節で述べたコスト予測といった重要な技術に関しても、プロジェクトやチーム内でのデータを蓄積し、それを次の

ステップにフィードバックして行くプロセスが有効です。図１１に示すようにプロジェクトマネジメントプロセスを捉えることによって、プロセス成熟度を向上したり、さまざまなスキルアップを図ることができます。

ここで述べた開発プロセスやマネジメントの側面は、組織知能の主要な部分を構成することになり、今後、組込みシステムの開発現場においても、その競争力の源泉となって行くことは必至です。

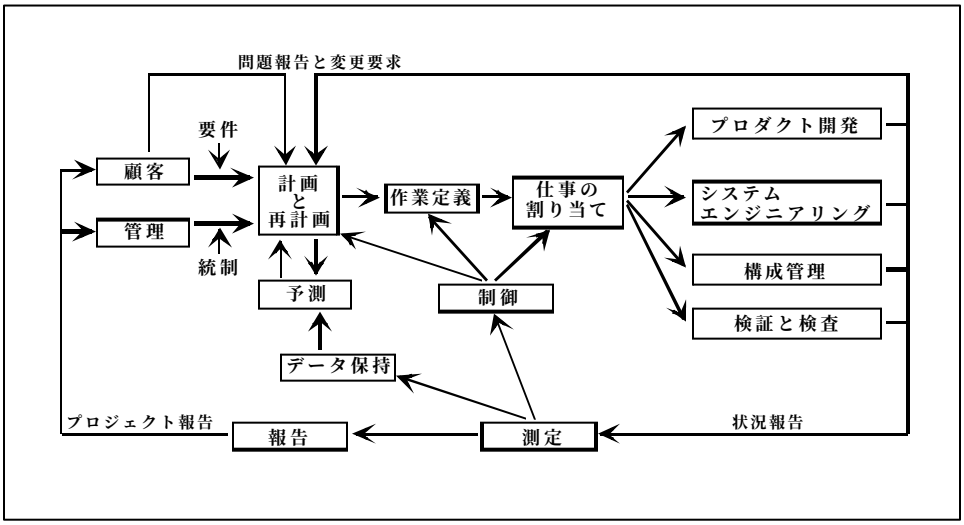

図11.プロジェクトマネジメントプロセス