ソフトウェア工学 108－1
（1996．3．22）

システム基盤開発を伴う業務システム開発のプロセス選定に関する一考察

菊地　伸治
NEC　生産システム開発本部
{kikuchi@psdl.tmg.nec.co.jp}

近年、ダウンサイジングが急速な勢いで進行しており、従来、汎用機でなければ不可能といわれた業務領域もその対象となりつつある。ダウンサイジングの進行に伴い、開発プロセスも”Waterfall Model”に基づくものから”Spiral Model”等に基づくものに変わり始めている。しかし新たな業務システムの為に、新たなミドルウエアや、パッケージコア部等のシステム基盤の開発を伴うプロジェクトでは、従来通り”Waterfall Model”に基づく開発プロセスを採用する方が一般的である。

本稿では、システムの基盤開発を伴う業務システム開発をする際に、リスク回避の為に、採用するべきプロセスについて考察する。

A Consideration on Selecting Process for Developing Application Program with Developing The Foundation of System.

Shinji Kikuchi

Production System Development Laboratory,
NEC Corporation
484 Tsukagoshi 3-Chome, Saiwai-ku, Kawasaki,
Kanagawa 210, Japan

Recently,it is expanding to be adopted into domains in the construction of systems with the ˝Open-Downsizing˝computers,but it could not be done without proper ones of each vender ,called ˝Host machine˝ so far.

With the progress to shift to ˝Open-Downsizing˝,the management of development is changing from the method in the ˝waterfall˝ model to one in the ˝spiral˝ model. But in the development of large scale systems,espacially developing application program with doing the new foundation of systems like a middle ware and the core of package softwares, useally it is progressed in the process of the ˝waterfall˝ model as before.

In this paper, it is discussed on selecting process for developing application program with doing the foundation of systems against risk.

1. はじめに

近年、ダウンサイジングが急速な勢いで進行しており、従来、汎用機でなければ不可能といわれた業務領域もその対象になりつつある。ダウンサイジングの進行に伴い、開発プロセスも" Waterfall Model" に基づくものから" Spiral Model" 等に基づくものに替わり始めている。しかし新規業務システムのために、新たなミドルウエアや、パッケージコア等のシステム基盤の開発を伴うプロジェクトでは、従来通り、" Waterfall Model" に基づく開発プロセスを採用する方が一般的である。本稿では、システム基盤ソフトウェアと、それを用いる応用ソフトウェアを段階的に開発する場合に採用するべき開発プロセスをリスク管理の観点から検討する。

システム基盤とは、その存在なくしてはシステムの実現が不可能な要素であり、例えばオペレーティングシステムと応用ソフトウェアの中間に位置するミドルウェアや、カスタマイズを事前に考慮したパッケージソフトウェア、小規模な例では共通ルーチン等が該当する。この様なソフトウェアは新規開発の傾向が強く、未知の部分も多い。それ故、機能面、性能面で仕様実現度等を向上させようとすると、版更新がなされていくのが一般的である。その様な事項を見積りに反映すると、その精度は変動の伴うものになる。それに対して、応用ソフトウェアは十分に検討されたシステム要素を用い、明確に定義され、既に計量化実績のある開発プロセスを用いて開発する傾向が強い為、既知の部分も多く、見積り精度は比較的高い傾向にある。以上から、変動要素が存在するシステム基盤ソフトウェアと、それを用いる応用ソフトウェアを段階的に開発する場合、そのリスクを事前に検討して、全体の開発プロセスを選定する必要がある。そのリスク対策の中心的課題は、システム基盤ソフトウェアの開発遅れがシステム全体の開発に影響を与えないことである。

その様なリスク事項に関して定量的な検討は十分になされて来ているとは言えない。B.W.Boehmは、その著書の中でリスク管理手法のフレームワークを定義しており、リスク査定を行う際の分析アクティビティにて、コスト分析並びにＰＥＲＴ図を始めとした工程分析について言及しているが、上記の様な開発に関する明確な分析方法については言及していない[1]。またP.R.Garvey、並びにF.D.Powellは、複数要素からなるシステムのコスト分析を、ＣＯＣＯＭＯを応用した３つの方法で比較検討しているが、そこではコスト分析に留まり、コスト故のリスクに関する評価が為されていない[2]。そこで本稿では、この問題を取り上げ、システム基盤とその応用ソフトウエアを段階的に開発する際のリスク評価、並びに開発プロセスの選定について検討する。尚、本稿で検討するリスク対象は、予算、開発期間上のリスクに留め、技術上のリスク、開発組織の成熟度に伴うリスクについては、直接的な検討対象とはしない。技術上のリスクについては、予算、並びに開発期間の査定をする際になされると見做す。また本稿の議論は、前述リスク管理フレームワークのうち、リスク査定に位置付けられる。

本稿の構成は、以下の通りである。２章ではリスク評価の指標として、工数、期間見積りに関して言及する。３章では、複数の開発プロセス案について定義する。ここでは比較検討を目的として、現実の開発で使用されるプロセスに留まらず、考えられ得る全プロセスも定義する。４章では、リスク回避の観点から見た開発プロセスの評価を行い、その選定方法に関する基本的な考え方について提案する。

2. 見積り式

前章の通り、本稿で検討するリスクは、予算、並びに開発期間に関するものに限定する。従って、検討の為に使用する見積り式について明記する必要がある。本稿では見積り方法に関して、以下の２点を仮定する。

(1) 見積りの実施は、多段階で為されることを仮定する。システムの基盤ソフトウェアを、新規に開発する必要があると判断出来るのは、通常、システム要求定義工程にて要件抽出を終え、システム基本設計段階でシステムの中心的な課題について合意出来た後と判断出来る。その段階では、システムの規模見積りはある程度の精度でなされていると推定出来る。

(2) ＣＯＣＯＭＯ等で定義されるコスト要因については、直接的には言及しない。理由は、ドメイン並びに利用する方法論が固定化されれば、殆どの要因が議論の中心的課題にはならないと判断されるためである。

従って見積りは、開発規模に対する開発工数、並びに開発期間を対象する。ＣＯＣＯＭＯに基づけば、開発工数kと開発規模klocは、式(1)で表現される相関関係でモデル化される[2][3]。

$$k = \hat{\alpha} \bullet (kloc)^{\hat{\beta}} \bullet \prod_{i=1}^{15} M_i \qquad \left(\hat{\alpha}, \hat{\beta} > 0\right) \tag{1}$$

ここで、$\prod_{i=1}^{15} M_i$は、コスト要因のパラメータセットである。しかし、現実には開発プロセスが完全に同一で無いことを考慮すると、統計値の基データとなる過去実例は、大数の法則を満足出来る程、多くはないと推定される。それ故、見積り根拠となる基データ数は有限と仮定し、信頼性を考慮した式(2)式(3)式(4)式(5)式(6)で検討する必要がある。その場合、信頼性係数100(1-α)%の予測域は、自由度(n-2)のt-分布で表現されることになる。

$$k = \hat{a} \bullet (kloc)^{\hat{b}} \bullet ten(\pm\theta(kloc)) \qquad (function : ten(x) \equiv 10^x) \tag{2}$$

$$\theta(kloc) = t_{n-2}(\alpha) \bullet \hat{\sigma}_k \bullet \sqrt{1 + \frac{1}{n_k} + \frac{\left(\log(kloc) - \log\left(\bar{L}\right)\right)^2}{S_{LL}}} \tag{3}$$

$$\hat{\sigma}_k^2 = \frac{1}{n_k-2} \bullet \sum_{i=1}^{n_k} \left(\log(k_i) - \log(\hat{a}) - \hat{b} \bullet \log(kloc_i) \right)^2 \tag{4}$$

$$S_{LL} = \sum_{i=1}^{n_k} \left(\log(kloc_i) - \log\left(\bar{L}\right) \right)^2 \tag{5}$$

$$\bar{L} = \left(\prod_{i=1}^{n_k} (kloc_i) \right)^{\frac{1}{n_k}} \tag{6}$$

尚、上記の式(2)式(3)式(4)式(5)式(6)に関しては、更に以下の３点を仮定している。仮定１） θ 関数は、正規変数として分布している。仮定２） 式(4)の各項は、互いに独立である。仮定３） 式(2)では、先の理由に基づきコスト要因を陽に記述しない。

開発工数kと開発期間tは、前述の関係同様、指数関数的相関関係でモデル化される[3]。しかし、現実には開発プロセスが完全に同一で無いことを考慮すると、この基データ数も大数の法則を満足出来る程、多くはないと推定される。それ故、これに関しても、見積り根拠となる基データ数は有限と仮定し、信頼性を考慮した式(7)式(8)式(9)式(10)式(11)で検討する必要がある。この場合も、信頼性係数100(1-α)%の予測域は、自由度(n-2)のt-分布で表現されることになる。

$$t = \hat{c} \bullet \hat{a}^{\hat{d}} \bullet (kloc)^{\hat{b}\hat{d}} \bullet ten\left(\pm \left(\theta(kloc) \bullet \hat{d} + \lambda(k) \right) \right) \tag{7}$$

$$\lambda(k) = t_{n-2}(\alpha) \bullet \hat{\sigma}_t \bullet \sqrt{1 + \frac{1}{n_t} + \frac{\left(\log(k) - \log\left(\bar{K}\right)\right)^2}{S_{KK}}} \tag{8}$$

$$\hat{\sigma}_t^2 = \frac{1}{n_t-2} \bullet \sum_{i=1}^{n_t} \left(\log(t_i) - \log(\hat{c}) - \hat{d} \bullet \log(k_i) \right)^2 \tag{9}$$

$$S_{KK} = \sum_{i=1}^{n_t} \left(\log(k_i) - \log\left(\bar{K}\right) \right)^2 \tag{10}$$

$$\bar{K} = \left(\prod_{i=1}^{n_t} k_i \right)^{\frac{1}{n_t}} \tag{11}$$

尚、上記の式(7)式(8)式(9)式(10)式(11)に関しても、以下の２点を仮定している。仮定１） λ 関数は、正規変数として分布している。仮定２） 式(8)の各項は、互いに独立である。　また、予想される納期の最大変動分は、式(12)にて計算される。

$$ten\left[\pm \left(t_{n-2}(\alpha) \bullet \hat{\sigma}_k \bullet \hat{d} \bullet \sqrt{1 + \frac{1}{n_k} + \frac{\left(\log(kloc) - \log\left(\bar{L}\right)\right)^2}{S_{LL}}} + t_{n-2}(\alpha) \bullet \hat{\sigma}_t \bullet \sqrt{1 + \frac{1}{n_t} + \frac{\left(\log(k) - \log\left(\bar{K}\right)\right)^2}{S_{KK}}} \right) \right] \tag{12}$$

3. プロセス定義

3.1　総括

図3.1に記す様に、"Waterfall Model"では開発工程は主に要求仕様定義、設計、製造、検査の４工程に分類出来る。通常、詳細設計とコーディングの工程は、モジュール毎に分割されて実施されるために、全体では時間的にオーバラップする工程も存在する。本稿では、マクロ的な評価を実施するために、図3.1に記した様に工程を再整理して、ＣＯＣＯＭＯ同様の定義を採用する。また本稿の検討では、先の仮定(1)に基づき、設計工程以後を検討の対象とする。従って、期間配分比、並びに工数配分比は、式(13)式(14)で定義される。

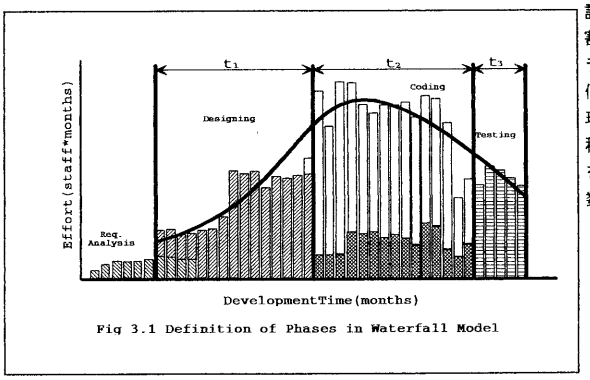

Fig 3.1 Definition of Phases in Waterfall Model

$$\sum_{i=1}^{3} t_{ij} = 1 \tag{13}$$

$$\sum_{i=1}^{3} k_{ij} = 1 \tag{14}$$

i=a,b　a:Application　b:Basic
j=1,2,3　1:Designing
2:Coding & Debug　3:Testing

更に以下の５点も検討上の前提とする。

(1)本稿で提案する手法は、プロジェクト実施前の査定段階で実施されるべきものなので、プロジェクト進行途中の仕様変更については、検討対象とはしない。従ってリスク評価は、開始～要求定義終了段階の前提条件、並びに規模見積り値のみでなされる。

(2)プロジェクトの基本方針は、コストよりもプロセス品質、および最終納期を優先することを仮定する。従って、Nordenの記したマンパワー曲線[4]が、外部要因によりEffort方向側に圧縮されることは考慮しない。

(3)システム基盤ソフトウェアの設計、並びに検査が終了する以前に、それに依存した応用ソフトウェアの一部でも設計、検査を着手することはしない。品質保証上の制約は厳格に仮定する。

(4)応用ソフトウエアについては、システム基盤ソフトウェアのApplication Program Interface（以下、API）に依存するモジュール群、並びにそれ以外の群（例えばバッチ系）の２部分から成ることを仮定する。ここで、前者は後者のａ倍である場合、各々の規模は以下の式(15)、式(16)で与えられる。

$$kloc_{ap1} = \frac{1}{(1+a)} kloc_{ap} \quad \text{（システム基盤ソフトウェアに依存しない応用ソフトウェア）} \tag{15}$$

$$kloc_{ap2} = \frac{a}{(1+a)} kloc_{ap} \quad \text{（システム基盤ソフトウェアに依存する応用ソフトウェア）} \tag{16}$$

(5)本稿では(2)に基づき、納期までの期間Ｔに対する実現可能性を評価することで、リスクを分析する。従って、期間Ｔの設定の仕方によって、リスクに関する観点が大きく異なる。それ故、明確に区別して議論する。表1.1は納期Ｔの捉え方についてまとめたものである。開発規模から推定される納期より、顧客要求納期の方が長いか、短いかに依ってリスクへの対処方法も変る。

Table1.1 The definition of the Relationship between LimitTime & DevelopmentTime

納期定義	特　徴	納期決定権限（可能性）	リスクに対する考慮点
安全納期	1)顧客要求納期が、過去実績から、ばらつきを考慮して推定される最大の開発納期よりも長い。 2)一部工程に遅延が発生しても、全体に影響を与えない。	ベンダの作業計画優先（殆ど有り得ない。）	1)どの様な場合でも、納期Ｔに対する遅延はないこと。 2)遅延発生時でも、作業者の遊休期間が発生しないこと。
標準納期	1)顧客要求納期が、過去実績から定量的に推定される納期よりも長い。但し納期遅延のリスクも存在する。 2)一部工程に遅延が発生しても、全体に影響を与えない。 3)プロセス全体で遅延が発生した場合、計画の調整実施が必要。	ベンダの作業計画、並びにベンダの作業計画に基づきユーザと機能削減の折衝。（一般的）	1)一部工程に遅延が出ても、開発期間Ｔ内で、対応出来ること。 2)遅延、並びに作業者遊休のリスクを減じるプロセスを選択すること。
限界納期	1)顧客要求納期が、過去実績から定量的に推定される納期よりも短い。 故に納期遅延のリスクが存在する。 2)工程間の順序関係は厳守 3)遅延が発生した時点で即、計画の調整実施が可能であることが前提。	ユーザの要求優先、並びにマーケット主導（一般的）	1)当初からリスクが存在する故、遅延並びに作業者遊休のリスクを減じるプロセスを選択すること。 2)工程間の順序関係は厳守

現実のプロジェクトでは、表1.1の安全納期で実施出来ることは殆どなく、標準納期もしくは限界納期で進めるのが一般的と推定される。そこで本稿では、後者の２点のみを検討の対象とする。しかし後者２点もリスクに対する対応が当然異なると推定される。標準納期の場合は、納期遅れ発生に対するリスクを考慮することが、検討するべき最大の問題であるが、限界納期の場合は、当初からリスクが存在している故、如何に解決を図るか、が最大の問題である。

3.2　２段階開発プロセス

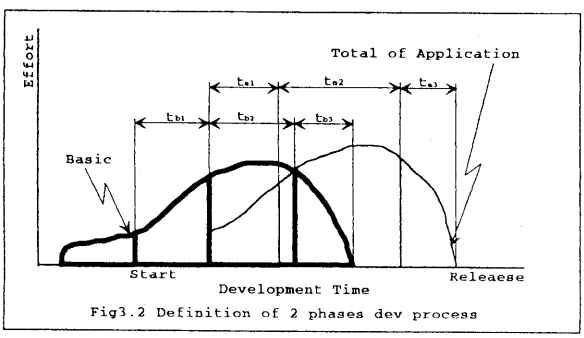

Fig3.2 Definition of 2 phases dev process

応用ソフトウェアを１つにまとめて開発する方法を２段階開発プロセスと定義する。図3.2は、このプロセスのモデル図である。現実のシステム開発プロセスはこれに近く、システム基盤ソフトウェアの設計の前段階で、システム基本設計を行い、基盤部分との切り分け、応用ソフトウェアとの界面の切り出しを行う。その後、応用ソフトウェアの開発に着手する。この方式では、システム基盤ソフトウェアに依存しない応用ソフトウェア機能についても、まとめて開発することになるので、システム基盤ソフトウェアの開発に遅れが発生した場合に、全機能がリリース出来なくなる欠点が存在する。２段階開発プロセスが採用される条件は３つ存在する。標準納期

の場合は、更に１条件が加わる。式(17)は、開発期間全体の制約条件である。ここで、分散補正分の考慮は限界納期に対してのみ適用する。また分散補正分に開発規模のものを含めていない理由は、それを含めると危険側での評価に偏り、結果として信頼性が高い評価が出来ないと推定されるためである。式(18)は、システム基盤ソフトウェア単独でも開発期間Ｔを満足する為の条件である。式(19)は検査段階に於いて、応用ソフトウェア側で、無駄工数が発生すること無く、効率的に作業を進めるための条件である。尚、システム基盤ソフトウェアに関連しない機能については、システム基盤ソフトウェアの検査中でも検査可能である。また式(17)に従うと、限界納期時に過去実績で最短の方法を採用した場合、開発期間Ｔ内で開発可能となることが示唆される。但しその場合、各工程で遅れが発生すれば、納期を厳守することは不可能となる。

$$t_{b1} \bullet \hat{c}_b \bullet (\hat{a}_b)^{\dot{d}_b} \bullet (kloc_b)^{\dot{b}_b\dot{d}_b} \bullet ten(-\lambda(k_b) \bullet \delta) + \hat{c}_a \bullet (\hat{a}_a)^{\dot{d}_a} \bullet (kloc_a)^{\dot{b}_a\dot{d}_a} \bullet ten(-\lambda(k_a) \bullet \delta) \leq T \quad (17)$$

（δ ：$\delta = 0$ when 標準納期、 $\delta = 1$ when 限界納期）

$$\hat{c}_b \bullet (\hat{a}_b)^{\dot{d}_b} \bullet (kloc_b)^{\dot{b}_b\dot{d}_b} \bullet ten(-\lambda(k_b) \bullet \delta) \leq T \quad (18)$$

$$(t_{b2} + t_{b3}) \bullet \hat{c}_b \bullet (\hat{a}_b)^{\dot{d}_b} \bullet (kloc_b)^{\dot{b}_b\dot{d}_b} \bullet ten\left(\lambda(k_b) \bullet \delta'\right) \leq \left(t_{a1} + t_{a2} + \frac{t_{a3}}{1+a^{\dot{b}_a\dot{d}_a}}\right) \bullet \hat{c}_a \bullet (\hat{a}_a)^{\dot{d}_a} \bullet (kloc_a)^{\dot{b}_a\dot{d}_a} \bullet ten\left(-\lambda(k_a) \bullet \delta'\right) \quad (19)$$

（δ' ：$\delta' = 0$ when 限界納期、 $\delta' = 1$ when 標準納期）

標準納期の場合、上記条件に加え、以下の条件も考慮する必要がある。それは、遅れに対する保証である。式(20)は、システム基盤ソフトウェアで最大の開発遅延が出た場合でも、全体が開発期間Ｔに完了することを保証する為の条件である。

$$\left[ten\left(\theta(kloc_b) \bullet \hat{d}_b + \lambda(k_b)\right) - 1\right] \bullet t_{b1} \bullet \hat{c}_b \bullet (\hat{a}_b)^{\dot{d}_b} \bullet (kloc_b)^{\dot{b}_b\dot{d}_b} \leq \left[1 - ten\left(-\left(\theta(kloc_a) \bullet \hat{d}_a + \lambda(k_a)\right)\right)\right] \bullet \hat{c}_a \bullet (\hat{a}_a)^{\dot{d}_a} \bullet (kloc_a)^{\dot{b}_a\dot{d}_a} \quad (20)$$

3.3 システム基盤ソフトウェア完了後３段階開発プロセス

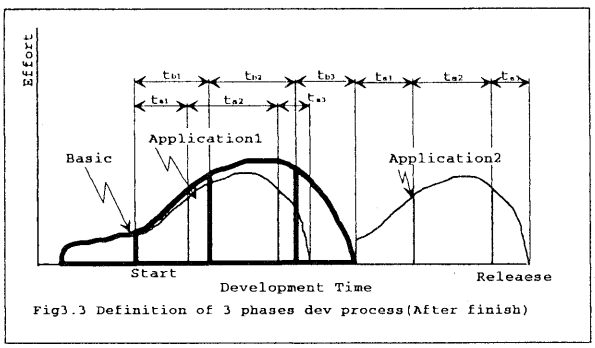

Fig3.3 Definition of 3 phases dev process(After finish)

システム基盤ソフトウェアに依存する応用ソフトウェアの開発着手時期を、システム基盤ソフトウェアの開発完了時点とする開発プロセスをシステム基盤ソフトウェア完了後３段階開発プロセス（略：完了後３段階開発プロセス）と定義する。図3.3は、このプロセスのモデル図である。著者の経験上、この開発プロセス実例も現実に幾つか確認している。この開発プロセスを採用すれば、２段階開発プロセスに見られた問題点は解決出来る。またシステム基盤ソフトウェアと応用ソフトウェアの界面であるAPIやｶｽﾀﾏｲｽﾞﾓｼﾞｭｰﾙが強固となる為、それに依存した応用ソフトウェア設計終了時点の設計後戻りは、殆ど皆無になる。しかし後述の開発プロセスと比較して、システム基盤ソフトウェアの開発遅れに対しては脆弱な点も存在する。本開発プロセスが採用される条件は、実質２つだけであり、標準納期では更に１条件が加わる。式(21)は開発期間全体の制約条件である。ここで、分散補正分に開発規模のものを含めていないのは、２段階開発プロセスと同様の理由による。式(22)は、システム基盤ソフトウェアに依存しない応用ソフトウェアの開発期間Ｔを満足するための条件である。

$$\hat{c}_b \bullet (\hat{a}_b)^{\dot{d}_b} \bullet (kloc_b)^{\dot{b}_b\dot{d}_b} \bullet ten(-\lambda(k_b) \bullet \delta) + \hat{c}_a \bullet (\hat{a}_a)^{\dot{d}_a} \bullet \left(\frac{a}{1+a}\right)^{\dot{b}_a\dot{d}_a} \bullet (kloc_a)^{\dot{b}_a\dot{d}_a} \bullet ten(-\lambda(k_a) \bullet \delta) \leq T \quad (21)$$

$$\hat{c}_a \bullet (\hat{a}_a)^{\dot{d}_a} \bullet \left(\frac{1}{1+a}\right)^{\dot{b}_a\dot{d}_a} \bullet (kloc_a)^{\dot{b}_a\dot{d}_a} \bullet ten(-\lambda(k_a) \bullet \delta) \leq T \quad (22)$$

標準納期の場合は上記２条件に加え、更にシステム基盤ソフトウェアの開発遅れに対して、開発期間Ｔを保証する条件を加える必要がある。式(23)は、その為の条件である。

$$\left[ten\left(\theta(kloc_b) \bullet \hat{d}_b + \lambda(k_b)\right) - 1\right] \bullet \hat{c}_b \bullet (\hat{a}_b)^{\dot{d}_b} \bullet (kloc_b)^{\dot{b}_b\dot{d}_b} \leq \left[1 - ten\left(-\left(\theta(kloc_a) \bullet \hat{d}_a + \lambda(k_a)\right)\right)\right] \bullet \hat{c}_a \bullet (\hat{a}_a)^{\dot{d}_a} \bullet \left(\frac{a}{1+a}\right)^{\dot{b}_a\dot{d}_a} \bullet (kloc_a)^{\dot{b}_a\dot{d}_a} \quad (23)$$

3.4　システム基盤ソフトウェア製造後３段階開発プロセス

システム基盤ソフトウェアに依存する応用ソフトウェアの開発着手時期を、システム基盤ソフトウェアのコーディング完了時点とする開発プロセスをシステム基盤ソフトウェア製造後３段階開発プロセス（略：製造後３段階開発プロセス）と定義する。図3.４は、このプロセスのモデル図である。著者の経験上、この開発プロセス例を確認したことはないが、前述の通り、開発プロセスとしては定義し得る故、評価の対象とする。この開発プロセスを採用すれば、２段階開発プロセスに見られた問題点は解決出来る。またシステム基盤ソフトウェアと応用ソフトウェアの界面であるAPIやｶｽﾀﾏｲｽﾞﾓｼﾞｭｰﾙは、完了後３段階開発プロセス程、強固ではなく、性能等の問題から作業の後戻りが発生する可能性も高いが、後述のシステム基盤ソフトウエア設計後３段階開発プロセスと比較した場合は安定している。しかし逆にシステム基盤ソフトウェアの開発遅れに対する脆弱性が大きくなる。

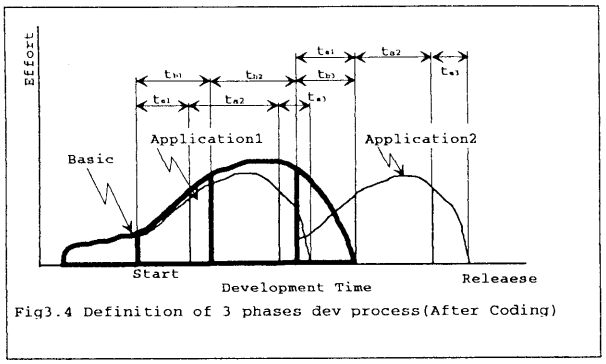

Fig3.4 Definition of 3 phases dev process(After Coding)

この開発プロセスが採用される条件は３つあり、標準納期では更に１条件が加わる。式(24)は、開発期間全体の制約条件である。ここで分散補正分に開発規模のものを含めていない理由は、２段階開発プロセスと同様の理由である。式(25)は、システム基盤ソフトウェアに依存しない応用ソフトウェアの開発期間Ｔを厳守するための条件である。式(26)は、検査段階に於いて応用ソフトウェア側の検査で無駄工数が発生すること無く、効率的に作業を進めるための条件である。

$$(t_{b1}+t_{b2})\bullet\hat{c}_b\bullet(\hat{a}_b)^{\hat{d}_b}\bullet(kloc_b)^{\hat{b}_b\hat{d}_b}\bullet ten(-\lambda(k_b)\bullet\delta)+\hat{c}_a\bullet(\hat{a}_a)^{\hat{d}_a}\bullet\left(\frac{a}{1+a}\right)^{\hat{b}_a\hat{d}_a}\bullet(kloc_a)^{\hat{b}_a\hat{d}_a}\bullet ten(-\lambda(k_a)\bullet\delta)\le T \quad (24)$$

$$\hat{c}_a\bullet(\hat{a}_a)^{\hat{d}_a}\bullet\left(\frac{1}{1+a}\right)^{\hat{b}_a\hat{d}_a}\bullet(kloc_a)^{\hat{b}_a\hat{d}_a}\bullet ten(-\lambda(k_a)\bullet\delta)\le T \quad (25)$$

$$t_{b3}\bullet\hat{c}_b\bullet(\hat{a}_b)^{\hat{d}_b}\bullet(kloc_b)^{\hat{b}_b\hat{d}_b}\bullet ten\left(\lambda(k_a)\bullet\delta'\right)\le (t_{a1}+t_{a2})\bullet\hat{c}_a\bullet(\hat{a}_a)^{\hat{d}_a}\bullet\left(\frac{a}{1+a}\right)^{\hat{b}_a\hat{d}_a}\bullet(kloc_a)^{\hat{b}_a\hat{d}_a}\bullet ten\left(-\lambda(k_a)\bullet\delta'\right) \quad (26)$$

（δ'：$\delta'=0$　when　限界納期、　$\delta'=1$　when　標準納期）

標準納期の場合は上記３条件に加えて、更にシステム基盤ソフトウェアの開発遅れに対して、全体開発期間Ｔを保証するための条件を加える必要がある。式(27)は、そのための条件である。

$$\left[ten\left(\theta(kloc_b)\bullet\hat{d}_b+\lambda(k_b)\right)-1\right]\bullet(t_{b1}+t_{b2})\bullet\hat{c}_b\bullet(\hat{a}_b)^{\hat{d}_b}\bullet(kloc_b)^{\hat{b}_b\hat{d}_b}\le\left[1-ten\left(-\left(\theta(kloc_a)\bullet\hat{d}_a+\lambda(k_a)\right)\right)\right]\bullet\hat{c}_a\bullet(\hat{a}_a)^{\hat{d}_a}\bullet\left(\frac{a}{1+a}\right)^{\hat{b}_a\hat{d}_a}\bullet(kloc_a)^{\hat{b}_a\hat{d}_a} \quad (27)$$

3.5　システム基盤ソフトウェア設計後３段階開発プロセス

システム基盤ソフトウェアに依存する応用ソフトウェアの開発着手時期を、システム基盤ソフトウェアの設計完了時点とする開発プロセスをシステム基盤ソフトウェア設計後３段階開発プロセス（略：設計後３段階開発プロセス）と定義する。図3.5は、このプロセスのモデル図である。このプロセスも２段階開発プロセスと同様、一般的な方法と推定出来る。この開発プロセスを採用すれば、２段階開発プロセスに見られた問題点は解決出来る。しかしシステム基盤ソフトウェアと応用ソフトウェアの界面であるAPIやｶｽﾀﾏｲｽﾞﾓｼﾞｭｰﾙが同様に脆弱である故、後戻りの発生する可能性は最も高い。しかしシステム基盤ソフトウェアの開発遅れに対しては、最も強固となる。本開発プロセスが採用される為の条件は３つあり、標準納期では更に１条件が加わる。式(28)は、開発期間全体の制約条件である。ここで、分散補正分に開発規模のものを含めていない理由は、２段階開発プロセスと同様の理由である。式(29)は、システム基盤

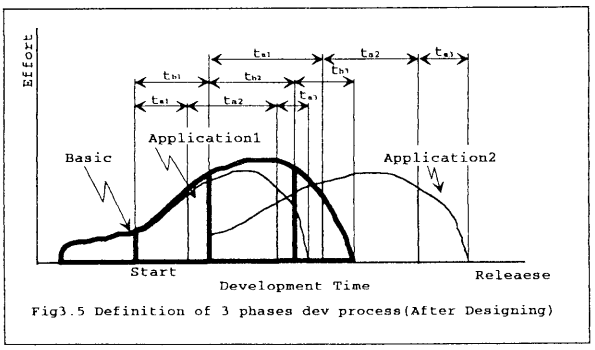

Fig3.5 Definition of 3 phases dev process(After Designing)

ソフトウェアに依存しない応用ソフトウェアの、開発期間Tを厳守するための条件である。式(30)は検査段階に於いて、応用ソフトウェア側の検査で、無駄工数が発生すること無く、効率的に作業を進めるための条件である。

$$t_{b1} \bullet \hat{c}_b \bullet (\hat{a}_b)^{\hat{d}_b} \bullet (kloc_b)^{\hat{b}_b\hat{d}_b} \bullet ten(-\lambda(k_b) \bullet \delta) + \hat{c}_a \bullet (\hat{a}_a)^{\hat{d}_a} \bullet \left(\frac{a}{1+a}\right)^{\hat{b}_a\hat{d}_a} \bullet (kloc_a)^{\hat{b}_a\hat{d}_a} \bullet ten(-\lambda(k_a) \bullet \delta) \leq T \quad (28)$$

$$\hat{c}_a \bullet (\hat{a}_a)^{\hat{d}_a} \bullet \left(\frac{1}{1+a}\right)^{\hat{b}_a\hat{d}_a} \bullet (kloc_a)^{\hat{b}_a\hat{d}_a} \bullet ten(-\lambda(k_a) \bullet \delta) \leq T \quad (29)$$

（δ ： $\delta = 0$ when 標準納期、 $\delta = 1$ when 限界納期）

$$(t_{b2} + t_{b3}) \bullet \hat{c}_b \bullet (\hat{a}_b)^{\hat{d}_b} \bullet (kloc_b)^{\hat{b}_b\hat{d}_b} \bullet ten\left(\lambda(k_a) \bullet \delta'\right) \leq (t_{a1} + t_{a2}) \bullet \hat{c}_a \bullet (\hat{a}_a)^{\hat{d}_a} \bullet \left(\frac{a}{1+a}\right)^{\hat{b}_a\hat{d}_a} \bullet (kloc_a)^{\hat{b}_a\hat{d}_a} \bullet ten\left(-\lambda(k_a) \bullet \delta'\right) \quad (30)$$

（δ' ： $\delta' = 0$ when 限界納期、 $\delta' = 1$ when 標準納期）

標準納期の場合は上記３条件に加えて、更にシステム基盤ソフトウェアの開発遅れに対して、全体の開発期間Tを保証するための条件を加える必要がある。式(31)は、そのための条件である。

$$\left[ten\left(\theta(kloc_b) \bullet \hat{d}_b + \lambda(k_b)\right) - 1\right] \bullet t_{b1} \bullet \hat{c}_b \bullet (\hat{a}_b)^{\hat{d}_b} \bullet (kloc_b)^{\hat{b}_b\hat{d}_b} \leq \left[1 - ten\left(-\left(\theta(kloc_a) \bullet \hat{d}_a + \lambda(k_a)\right)\right)\right] \bullet \hat{c}_a \bullet (\hat{a}_a)^{\hat{d}_a} \bullet \left(\frac{a}{1+a}\right)^{\hat{b}_a\hat{d}_a} \bullet (kloc_a)^{\hat{b}_a\hat{d}_a} \quad (31)$$

4. 評価結果、並びに考察

図4.1は式(32)～式(43)の条件下で求められる、標準納期時の各プロセスの適用範囲を記した図である。また図4.2は、式(32)～式(43)の条件下で求められる、限界納期時の各プロセスの適用範囲を記した図である。両図における横軸は、応用ソフトウエアの開発規模$kloc_a$であり、縦軸はシステム基盤ソフトウエアの開発規模$kloc_b$である。図4.1、並びに図4.2中のΦは式(44)により導出される。

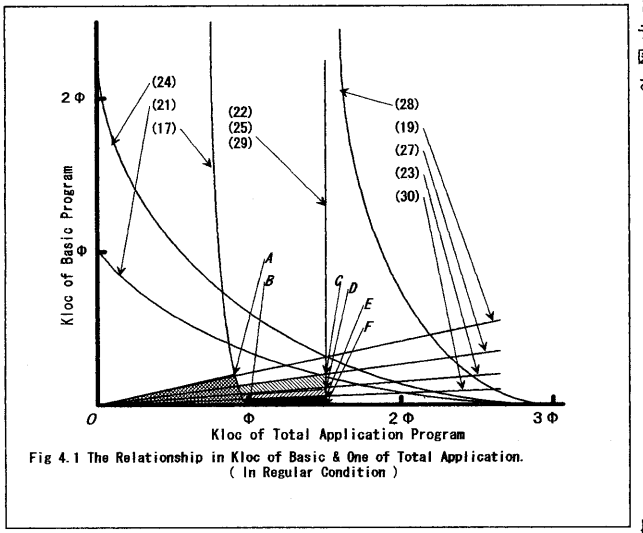

Fig 4.1 The Relationship in Kloc of Basic & One of Total Application.
(In Regular Condition)

$$t_{a1} = t_{b1} = 0.36 \quad (32)$$
$$t_{a2} = t_{b2} = 0.36 \quad (33)$$
$$t_{a3} = t_{b3} = 0.28 \quad (34)$$
$$\hat{a}_a = \hat{a}_b = 2.8 \quad (35)$$
$$\hat{b}_a = \hat{b}_b = 1.2 \quad (36)$$
$$\hat{c}_a = \hat{c}_b = 2.5 \quad (37)$$
$$\hat{d}_a = \hat{d}_b = 0.32 \quad (38)$$
$$a = 0.5 \quad (39)$$
$$n_a \cong n_b \cong \infty \quad (40)$$
$$\alpha = 0.95 \quad (41)$$
$$t_{n-2}(\alpha) = 1.96 \quad (42)$$
$$\hat{\sigma}_k = \hat{\sigma}_t = 0.1 \quad (43)$$
$$\Phi = \left[\frac{T}{\hat{c}_x \bullet (\hat{a}_x)^{\hat{d}_x}}\right]^{\frac{1}{\hat{b}_x\hat{d}_x}} (x = a, b) \quad (44)$$

図4.1中の図形$OABC$で囲まれた領域は、２段階開発プロセスが適用出来る範囲である。曲線(17)は、式(17)から導出される開発期間Tに対する限界曲線を示す。図4.1中の$\triangle OCF$で囲まれた領域は、システム基盤ソフトウエア製造後３段階開発プロセスが適用出来る範囲である。曲線(24)は、式(24)から導出される開発期間Tに対する限界曲線を示すが、適用範囲決定には寄与していない。図4.1中の$\triangle ODF$で囲まれた領域は、システム基盤ソフトウエア完了後３段階開発プロセスが適用出来る範囲である。曲線(21)は、式(21)から導出される開発期間Tに対する限界曲線を示すが、これも先と同様、適用範囲決定には殆ど寄与していない。図4.1中の$\triangle OEF$で囲まれた領域は、システム基盤ソフトウエア設計後３段階開発プロセスが適用出来る範囲である。曲線(28)は、式(28)から導出される開発期間Tに対する限界曲線を示すが、これも先と同様、適用範囲決定には寄与していない。

標準納期の場合、ある工程で遅れが生じても開発期間Tを厳守する為には、開発期間Tそのものに関係する関数が直接、適用範囲決定に寄与するとは推定し難い。それ故、標準納期の場合は、遅れ対応に関する条件、並びに工程順序を厳守する為の条件が適用範囲を決定することになる。より具体的に言えば、以下の様になる。システム基盤ソフトウエア設計後３段階開発プロセスは、開発期間Tに対する余裕度から見て、最も安全な開発プロセスと考えられる。しかし、実際にはリスクを最少にする開発プロセスとしては、適用範囲が極めて小さい。この開発プロセスは、システム基盤ソフトウエアの設計以後の工程で

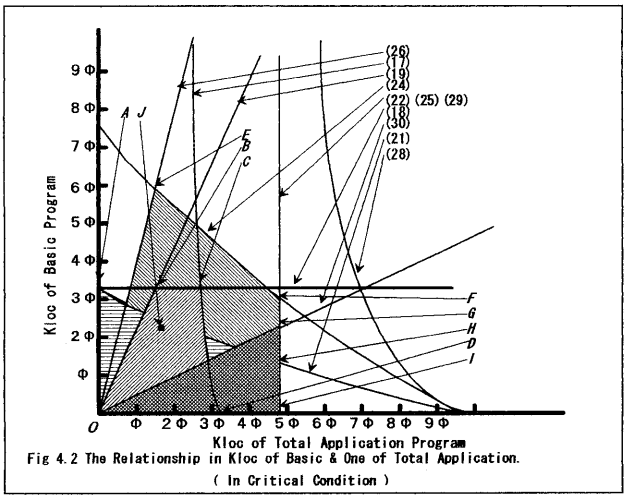

Fig 4.2 The Relationship in Kloc of Basic & One of Total Application.
(In Critical Condition)

大きく遅れが発生しても、それに依存する応用ソフトウエアの開発期間を大きく圧縮することが実質的に可能である。しかし実際には、依存する応用ソフトウエア開発での一時的な作業者遊休が発生する可能性も高くなる。従って後者を考慮すると、適用可能範囲は小さなものに留まってしまうことになる。以上から、条件(32)～(43)を満足する標準納期下では、図4.1に記された様に２段階開発プロセス、もしくはシステム基盤ソフトウエア製造後３段階開発プロセスを採用することが望ましいと考えられる。また式(23)、式(27)、式(30)から導出出来る様に応用ソフトウエアの内、システム基盤ソフトウエアに依存する部分が小さい程、各適用範囲は狭められ、かつ同領域に重なって来る。

図4.2中の図形ＯＢＣＤで囲まれる領域は、２段階開発プロセスが適用出来る範囲である。曲線(17)は、式(17)から導出される開発期間Ｔに対する限界曲線を示す。図4.２中の図形ＯＡＨＩで囲まれる領域は、システム基盤ソフトウエア完了後３段階開発プロセスが適用出来る範囲である。曲線(21)は、式(21)から導出される開発期間Ｔに対する限界曲線を示す。

図4.2中の図形ＯＥＦＩで囲まれる領域は、システム基盤ソフトウエア製造後３段階開発プロセスが適用出来る範囲である。曲線(24)は、式(24)から導出される開発期間Ｔに対する限界曲線を示す。図4.2中の△ＯＧＩで囲まれる領域は、システム基盤ソフトウエア設計後３段階開発プロセスが適用出来る範囲である。曲線(28)は、式(28)から導出される開発期間Ｔに対する限界曲線を示すが、適用範囲の決定には寄与していない。 限界納期時の場合は標準納期時とは異なり、適用範囲を決定する最大要因は、開発期間Ｔである。従って、図4.2から判断出来る様に式(17)、式(21)、式(24)、式(28)が大きな意味を持つ。限界納期の場合は、開発期間Ｔに対して、当初からリスクが存在している日程である故、如何にリスクを減少させるかが最大の問題となる。例えば、図4.2の点Ｊでプロットされる開発プロジェクトの場合、システム基盤ソフトウエア設計後開発プロセス以外の全プロセスが選択可能である。しかし開発期間Ｔに対する余裕を考慮すると、システム基盤ソフトウエア製造後３段階開発プロセスが最も余裕がある故、それを採用するべきと考えられる。より一般化すれば、式(45)に記されるプロセスリスク度というメトリクスを最少とする開発プロセスを選択すれば良いことになる。

$$Process\ Risk = (distance(Project Point(kloc_a, kloc_b), function))^{-1} \quad (45)$$

尚、式(45)中のfunctionは、式(17)式(21)式(24)式(28)である。

また、完了した実プロジェクトに本方法を適用した結果、成功実例に対しては特に不具合な点は確認されなかった。しかし適用実例数が十分とは言えない為、更に評価が必要である。

5. おわりに

本稿ではシステム基盤ソフトウェアと、その上位に位置する応用ソフトウェアを段階的に開発する場合に採用するべき開発プロセスをリスク管理の観点から検討し、評価メトリクスを提案した。本稿の方法は前述リスク管理フレームワークの内、リスク査定に位置付けられ、開発プロジェクトの計画段階で利用されるべきものである。尚、以下４点が今後の課題である。

課題1)適用実例数を増やし、実際の場における検証をすること。

課題2)同時期に投入出来る工数に制限がある場合のプロセス選択方法の検討。

課題3)規模見積りの曖昧さに対する保証、並びに開発期間の変動分の検討。

課題4)進行中のプロジェクトのリスク評価方法、並びにプロセス制御方法の検討。

謝辞

本発表の機会を頂きました職場上司の皆様、並びに査読等を御願いしましたＮＥＣﾏｲｺﾝｿﾌﾄ開発環境研究所砂塚課長殿に、紙面を御借りして御礼申し上げます。

[参考文献]

[1]B.W.Boehm :"Tutorial of Software risk management " ,pp1-16,115-147 IEEE Computer Society Press (1989).

[2]P.R.Garvey and F.D.Powell :"Three methods for quantifying software development effort uncertainty", Journal of Parametrics ,March 1988 ,pp76-92(1988).

[3]B.W.Boehm :"Software Engineering Economics" ,Prentice-Hall (1981).

[4]L.H.Putnam and W.Myers:"Measures for excellence:Reliable Software on Time within Budget",Prentice-Hall(1991), (研野和人訳："プロジェクトの見積りと管理のポイント",pp30-41 共立出版(1995)).