開発業務とオペレーションズ・リサーチ

竹谷　信夫

オペレーションズ・リサーチの開発業務への適用例を示す．問題を発見し，解決に至るまでのプロセスに学習効果を考慮した進化ゲーム理論を盛り込んでつくって数理モデルを提案し，開発業務のメカニズム解明を行う．同時に，電子機器の製品開発における組み込みソフトウェアのバグ除去を低費用で期限内に行うことができる応用例について述べる．

キーワード：組み込みソフト，進化ゲーム理論，ソフトバグ

1.　はじめに

内閣府発表の「平成20年度 年次経済財政報告」を見ると，研究費を3年以内に回収すると回答した企業が2003年では120社だったものが，2008年では160社と増加しており，企業の開発研究が短時間化に向かっていることがわかる．この理由はいろいろ考えられるが，開発研究業務のグローバル競争が激化していることも一因と考えられる．

例えば，グローバル展開しているApple，Nokiaはマーケティングを本国の組織で実施し，主要部品開発やセット開発を日本・韓国で行い，中国などのアジアで製造を行っているが，このように「ものつくり」だけでなく，「開発研究」も水平分業され，グローバル競争下に置かれはじめたことが原因で，リードタイムの短縮が求められていると考えられる．

2.　本研究の目的

本研究では開発研究の中で，ソフトウェア作成にターゲットを絞っているが，その理由はソフト開発が日常的に業務の管理を数値化していることにある．通常，ソフト開発業務に携わっているメンバーは大勢存在し，かつ関係する職能も多様であるため，ソフト開発業務の進捗はWeb上で行っていることが多い．そのためソフト不具合（バグ）がいつ発見され，いつ対策されたのか情報が整備されている．またソフト開発のバグ件数だけなら，新薬開発などに比べて，機密性も低く，研究の対象として取り扱いやすいということもソフト開発にターゲットを絞っている理由である．

さて，本研究の目的に立ち返ってみると，電子機器の製品開発の中でハードに加えて，組み込みソフトウェア作成が重要かつ大きなコストを占めている．そこではシステム設計だけでなく，ソフト不具合（バグ）除去を中心に短時間に多くの人手が必要で，そのプロセスが製品の競争力に多大な影響を与えており，この開発業務の開発効率を行うことが本研究の目的である．そのためには開発業務のメカニズムを明確にし，数理モデル化し，様々なシミュレーションを実施できることが必要になってくるが，現状ではそのような数理モデルは存在しない．

3.　従来のソフト開発

3.1　ソフト進捗確認方法

図1の累積バグ曲線は現時点で行っているソフト進捗確認の一例を示しているが，横軸に時間，縦軸にバグ数をプロットしたものであり，累積バグと発見バグの2つの曲線がある．今までは累積バグ曲線の傾きが緩やかになったことや発見バグ曲線のピークの存在を確認することで，ソフト開発の収束が近いことを経験的に判断していた．

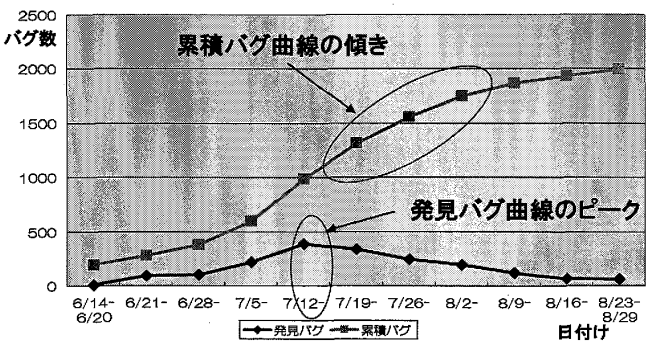


図1　累積バグ曲線の実例

たけたに　のぶお
パナソニック㈱
〒571-8504 門真市松生町1-5

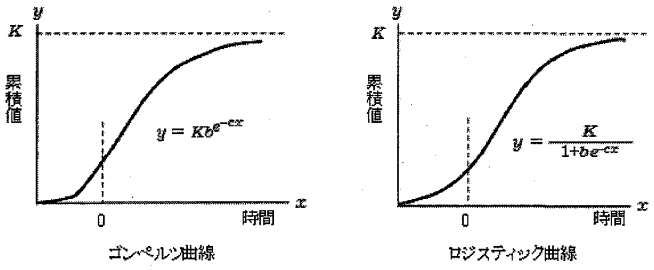


図2　従来のモデル曲線

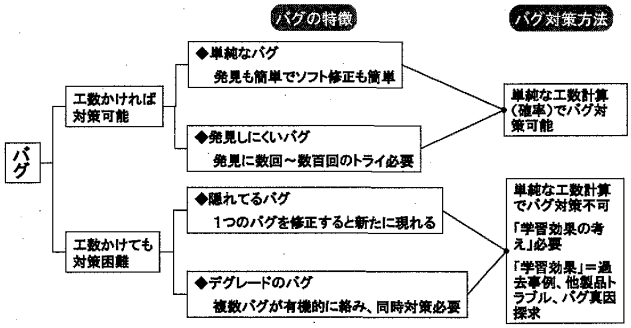


図3　バグの特徴と対策方法

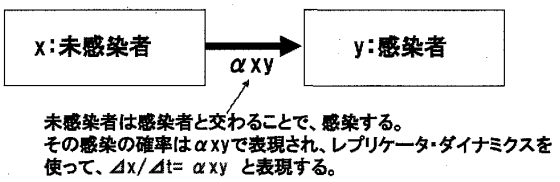


図4　感染モデル

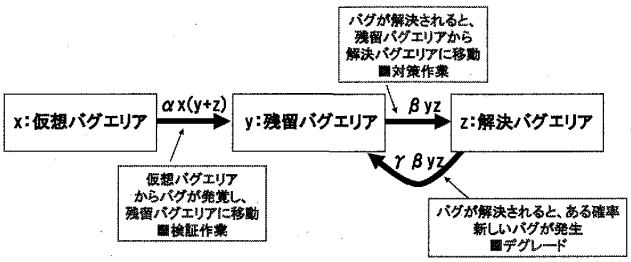


図5　開発業務の数理モデル

3.2　バグ曲線のモデル化

図2はバグ曲線のモデルとして従来から提案されているもので，1つはゴンペルツ曲線/成長曲線と呼ばれており，生物の個体数や新製品の販売数などを扱う曲線として知られている．初期段階と終焉段階で緩やかにカーブが立ち上がることが特徴的である．またもう1つはロジスティック曲線で，個体群増加のモデルとして有名であるが，これも形が累積バグ曲線に「似ている」という理由でモデル化されることもある．

しかしこれらの曲線と実際の累積バグ曲線との相関性は確かめられておらず，開発業務の効率を議論することには適さない．

4.　数理モデル化の作成

4.1　バグの特徴，対策方法について

ソフト開発の数理モデル化を行うには，バグの種類，特徴，対策方法などを分析する必要がある．図3はバグの種類を分類したもので，大きく分けて2つのバグに分類できる．時間と人をかければ対策できるバグと，複数のバグが有機的に絡んでいるため対策が困難なバグの2種類である．対策が困難なバグは一度対策しても，また現れる（デグード）ことがあり，これがソフト開発を遅らせる大きな原因の一つになっている．これを考慮するため，通常の「バグ発見⇒バグ対策」というプロセスだけでなく，「バグ発見⇒バグ対策⇒バグデグレード⇒バグ対策」というプロセスも考慮していく必要がある．

また，対策困難なバグは過去事例や他製品の事例を調査してそれを参考に対策を進めること，いわゆる学習効果を使うことが有効であり，この考えを数理モデルに入れ込む必要がある．

4.2　数理モデル立案の考え方

以上のようにソフト開発業務の数理モデル化にはバグ対策のプロセス全体を数理モデル化すること，学習効果を数理モデルに入れ込むことの2つが必要である．

図5はバグ対策のプロセス全体をモデル化したものを示しているが，ポイントは「x：仮想バグエリア」，「y：残留バグエリア」，「z：解決バグエリア」の3エリアを設け，バグ発見のプロセスである検証作業を実施することで，「x：仮想バグエリア」から「y：残留バグエリア」にバグが移動するという概念でバグを取り扱うことにある．同様にして，バグ対策作業により，「y：残留バグエリア」から「z：解決バグエリア」にバグが移動することでバグが解決されたと表現する．デグレードは「z：解決バグエリア」から，「y：残留バグエリア」にバグが移動することで，有機的に絡まっているバグが，再度バグとして現れることを示している．

学習効果を数理モデル化していくには，進化ゲーム理論で使われているレプリケータ・ダイナミクスを使うことで解決できる．図4の感染モデルはタカハトモデルとして有名であるが，「x：未感染者」と「y：感染者」が交じり合うことにより，「x：未感染者」が病気に感染するモデルである．単位時間当たりの

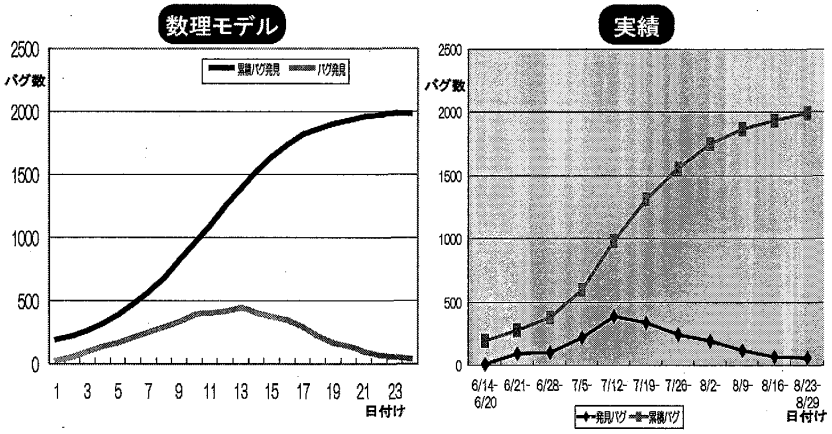


図6　累積バグ曲線

「x：未感染者」の変化量を

$$\Delta x/\Delta t=\alpha xy \tag{1}$$

と数式化し，式(1)を解くことで未感染者が感染していく状況を時系列に示すことができる．

この考え方を応用したものが図5の開発業務数理モデルである．

$$\alpha x(y+z) \tag{2}$$

式(2)の$x(y+z)$とは過去に発見されたバグ（y：残留バグエリア）と過去に対策されたバグ（z：解決バグエリア）の数が多くなればなるほど，発見数が増加することを示している．αはバグが発見される効率を表し，新規案件やバグ数が絶対的に多い場合は効率が悪いなど，案件ごとに値が変化すると考えられる．

$$\beta yz \tag{3}$$

式(3)のyzとは過去に対策されたバグ（z：解決バグエリア）の数が多くなると，バグ原因追求の速度がアップし，対策数が多くなることを示している．βはバグが対策される効率を表し，複雑のシステムなどでは対策効率が悪化すると考えられる．

$$\gamma\beta yz \tag{4}$$

式(4)は式(3)に対して，決まった確率で対策されたバグが再度バグとして表れることを示している．経験的には対策したバグの1～5%程度がデグレード（再度バグとして現れる）される．

次にレプリケータ・ダイナミクスを使って，それぞれの数式を解いていく．まず「x：仮想バグエリア」に注目すると，ある短時間Δtの間に$\alpha x(y+z)$のバグが流出していることになるため，

$$\Delta x/\Delta t=-\alpha x(y+z) \tag{5}$$

と表現できる．次に「y：残留バグエリア」に注目すると，短い時間Δtの間にβyzが流出し，$\alpha x(y+z)$，$\gamma\beta yz$が流入することになるため，

$$\Delta y/\Delta t=\alpha x(y+z)-\beta yz+\gamma\beta yz \tag{6}$$

と表現できる．「z：解決バグエリア」に注目すると，短い時間Δtの間に$\gamma\beta yz$が流出し，βyzが流入することになるため，

$$\Delta z/\Delta t=\beta yz-\gamma\beta yz \tag{7}$$

と表現できる．累積バグ曲線は「y：残留バグエリア」に流入するバグ数なので，式(8)のように表現できる．

$$\text{累積バグ数}=\Sigma(\alpha x(y+z))+\Sigma(\gamma\beta yz) \tag{8}$$

4.3　数理モデルと実績の比較

式(5)，(6)，(7)，(8)はエクセルなどの表計算ソフトを使うことで，Δtごとにx, y, zがどのように変化するかを，計算させることができるが，これをグラフ化したものが図6の累積バグ曲線（数理モデル）になる．図6の累積バグ曲線（実績）と比較すると，立ち上がり期間/収束に近い期間では累積バグ曲線の傾きが低くなることなどの特徴が良く似ていることが分かる．

5.　数理モデル化の活用

5.1　設定条件

トータルバグ数1,900件，収束期間（トータルバグ数の95%時点）が19週のプロジェクトが存在すると仮定する．その累積バグ曲線（当初予測）は図7のようになり，これをベースにしてプロジェクトを推進するための費用を計算できる．検証件数1人週＝100件/12万円，対策件数1人週＝100件/25万円と仮定すると，必要ピーク検証件数500件と予想されるので検証人員5名必要，ピーク対策件数は750件と予想されるため対策人員8名必要．よって，総費用/4,940万円かかると試算できる．

これを当初予測と設定してプロジェクトをスタートし，7週間後に累積バグ数が明確になったと仮定する．

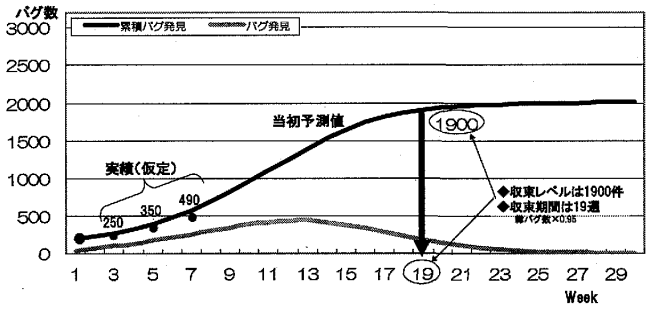


図7　当初予想と7週目実績

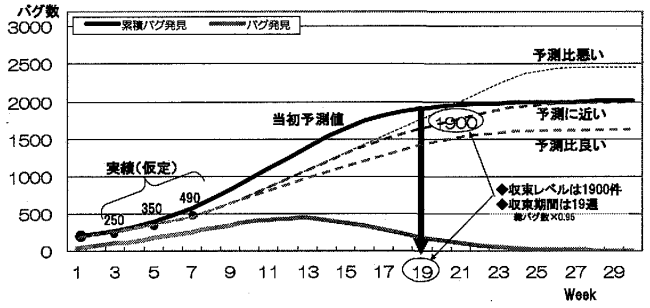


図8　3つの仮定バージョン

図7はそれを表しているが，実績（仮定）が当初予想よりも約4%程度少ないという場合を想定している．

5.2　3つの予想

7週間後の時点で，当初予測よりも約4%少ないとした場合，最終トータルバグ数は「当初予測よりも少ない（良い）」，「当初予測に近い」，「当初予測よりも多い」の3つが考えられるが，それを図に表したものが図8になる．今後，3つの予測累積バグ曲線を数理モデルでシミュレーションし，どういった対策が可能かを検討する．

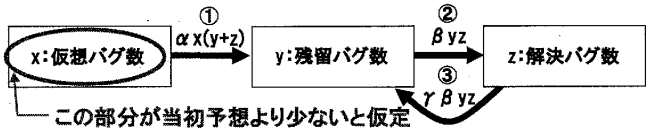
5.3　対策方法

● 予想より少ない場合

トータルバグ数が予想よりも少ない場合を考える．図9のブロック図にてトータルバグ数「x：仮想バグ」を変えることで，7週間後の累積バグ数がちょうど想定の490件になるように合わせ込む．結果的に当初予測1,900件の半分である850件になるが，その累積バグ曲線を図9に示す．

● 予想に近い場合

トータルバグ数が予想とほぼ同じ場合に，7週間後の累積バグ数が490件になるように，αの値を変化させる．αはバグが発見される効率を表しており，新規案件やバグ数が絶対的に多い場合は効率が悪くなると考えられる．結果的に当初予想の収束期間19週よりも3週多く，22週目にバグが収束できると予想される．その累積バグ曲線は図10に示してある．



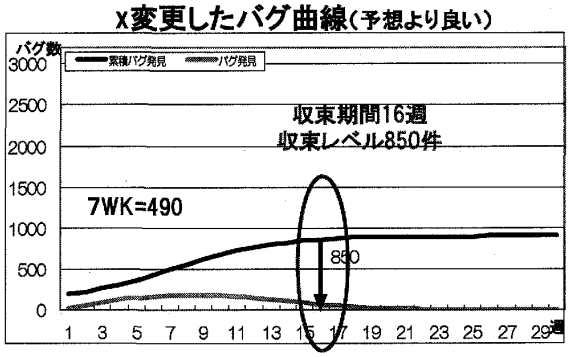
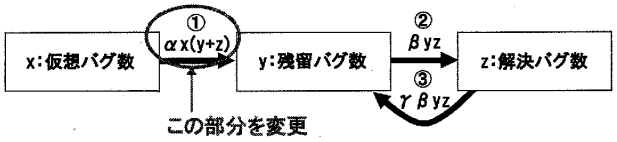


図9　予想より少ない想定



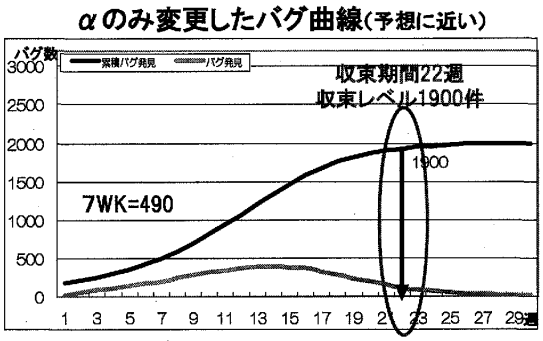


図10　予想に近い想定

● 予想よりも多い場合

トータルバグ数が予想よりも多い場合，7週間後の累積バグ数が490件になるように，αの値とトータルバグ数を変える．まず最初にトータルバグ数を2,930件と仮定し，次にαの値を変化させて，7週後の累積バグ数を490件に合わせこむ．結果的に当初予想の収束期間19週より6週多く，25週目にバグが収束できると予想され，トータルバグ数は2,930件となる．その累積バグ曲線は図11に示してある．

● 3バージョンの考察

図12は「トータルバグ数が予想よりも少ない，近い，多い」の3バージョンを一緒に描いたものである．従来のやり方だと発見バグ曲線のピークが判明するか，累積バグ曲線の傾きが低くならないと，収束時期が分からなかったが，図12ではそれが13～15週になるこ

とが分かる．一方，本研究のやり方だと，まず7週目にて予想よりも「トータルバグ数が少ない，近い，多い」の3バージョンを予測できる．このうち「トータルバグ数が少ないVer. A」は9週目において発見バグ曲線のピークを迎えるため，Ver. Aであるかどうか判別できる．次に「トータルバグ数が予想に近いVer. B」は14週目において発見バグ曲線のピークを迎えるため，Ver. Bかどうかを判別できる．

まとめると，7週目にて3つの予測累積バグ曲線を予想でき，9週目にVer. Aを判別でき，14週目にてVer. Bを判別できる．

5.4 挽回策と収束期間・開発費用

次に当初予測した収束期間内（19週）に開発を終えるには，どのような挽回策が考えられるか考察してみる．

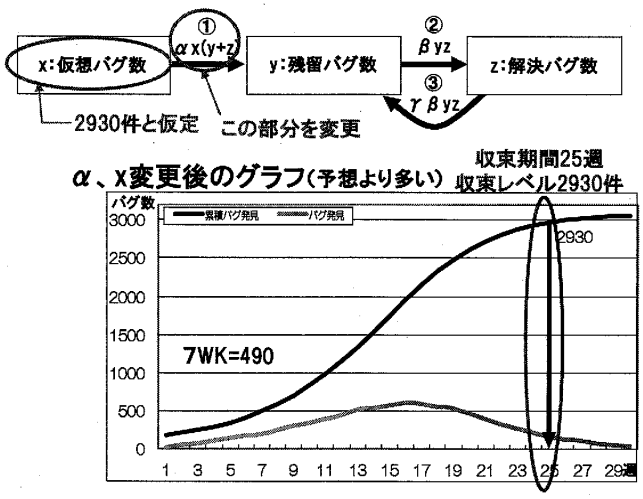


図11　予想より多い想定

まず9週目までは検証人員5名，対策人員8名で作業を行い，発見バグ曲線のピーク有無を見ることで，Ver. Aかどうかを判別するようにする．仮にVer. Aと判別できた場合はトータルバグ数が減っているため，数理モデルの発見バグ予想数，対策バグ予想数から判断して，検証人員を5名から3名に，対策人員を8名から4名に減らすことができる．これにより，開発費用を当初予定の4,940万円から3,292万円に減らすとともに，ソフト収束期間は当初の予定19週よりも短い16週で収束できる．また，9週目においてVER. A以外だと判別できた場合は，人員を20%増強させることで，Ver. Bの収束期間を19週にすることができる．

仮にVer. Bとなった場合は人員増強などを行う必要がなく，そのままソフト開発を続けることができ，総費用は当初予定の4,940万円から5,440万円で収束させることができる．もし数理モデルを使わずに，挽回策を講じなければ19週目で収束し，総費用は6,240万円かかることになり，数理モデルの効果としては6,240万円−5,440万円＝800万円ということになる．

一方14週目にてVer. Cと判別できた場合，人員をさらに40%増強し検証人員8名，対策人員14名体制でソフト開発を行う必要があると予想できる．これを実施することで，総費用は6,060万円と当初予測よりも1,120万円多くなるものの，19週でソフト開発を収束させることができる．ちなみに挽回策なしで開発すると，Ver. Cの場合は30週でバグが収束し，総費用は7,800万円かかることになり，数理モデルの効果

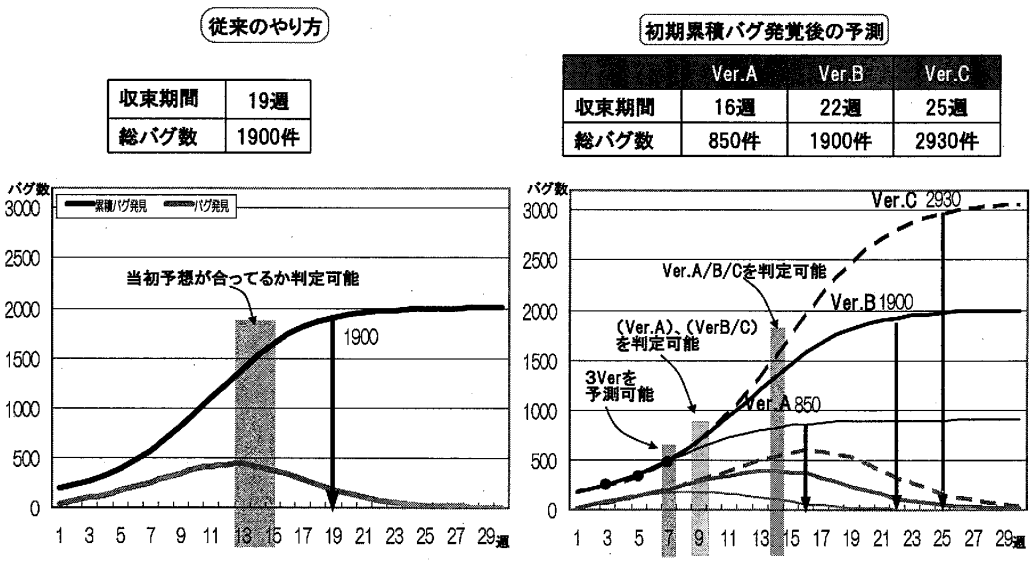


図12　予想累積バグ曲線の考察

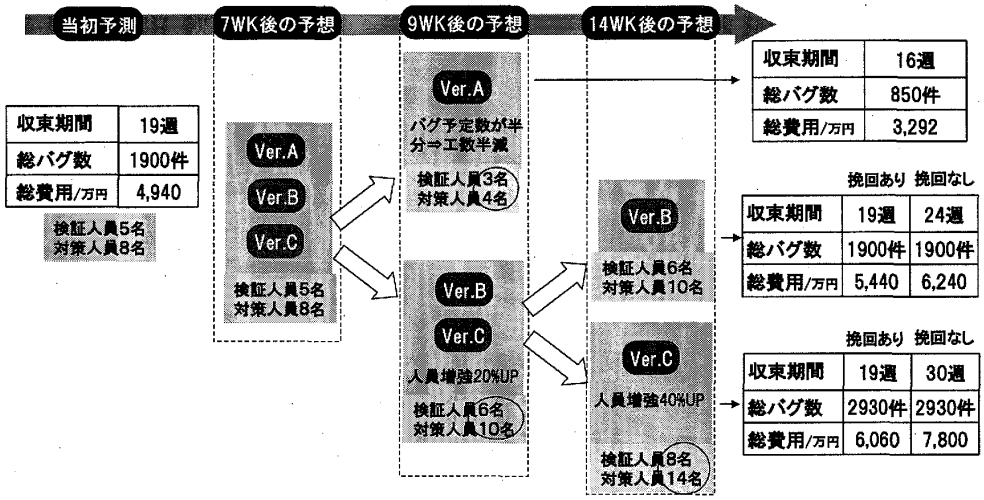


図13　挽回策と収束期間・開発費用

は7,800万円−6,060万円＝1,740万円となる．

6. まとめ

従来は累積バグ曲線の形が似ているということで，図2に示すようなゴンペルツ曲線，ロジスティック曲線を使うことがあったが，開発プロセスに沿ったモデルではないため，開発の収束時期を予想するまでは至らなかった．本研究では図5のような開発業務プロセス全体を包含し，かつ進化ゲーム理論をベースとした学習効果も盛り込んだ数理モデルを使って，実際の累積バグ曲線に近いモデルを構築できた．

最後に，ビジネススクールではファイナンス（事象をモデル化すること）を学んだが，事象をよく観察し，本質を見極めてモデル化することが重要．さらに現実とモデルと比較し，差異を究明してモデルをブラッシュアップしていくことが必要と学んだ．それを忘れることなく，研究を続けてきたことが，今回の成果につながったと考えられる．

参考文献

[1] J. W. Weibull, Evolutionary Game Theory (1995).
[2] 石原英樹, 金井雅之, 進化的意思決定 (2002).
[3] 大浦宏邦, 社会科学者のための進化ゲーム理論 (2008).
[4] 内閣府 平成20年度年次経済財政報告—リスクに立ち向かう日本経済—.