# ソフトウェア開発の失敗に関する会計処理案-ソフトウェアの仕損を会計ではどのように捕捉すればよいか-

# ソフトウェア開発の失敗に関する会計処理案

## —ソフトウェアの仕損を会計ではどのように捕捉すればよいか—

## An Accounting Idea for Failures of Software Development

## —How should we acquire spoilages of software in accounting？—

会計専門職研究科　会計専門職専攻

2011年3月修了

長田　芙悠子

Fuyuko OSADA

**【論文要旨】**

本稿は、ソフトウェア開発における仕損に関する会計処理案を提示するものである。プロジェクトの失敗は後を絶たないが、現行の財務報告において、プロジェクトの失敗が情報開示されることはほぼ無い。また、ソフトウェアの仕損は捕捉困難とされ、仕損費として処理せずに放置されてきた。事業リスクの大きな部分を占めるシステム・リスクが情報開示されていないということである。それに対して、ソフトウェア開発プロジェクトの失敗を会計的に捉えるために、ソフトウェア開発における仕損に関して、減損会計に準じた会計処理案を提示する。

会計処理案のポイントは、次の通りである。①プロジェクトの失敗を３大指標ＱＣＤで捉える。②仕損は、プロジェクトの中止・中断と、継続・完了した場合に大別する。③適用対象は、研究開発目的以外のソフトウェア開発のすべてとする。④仕損の区分は、異常仕損と仕損度合いが甚大な特別損失とする。⑤仕損の兆候の識別は、計画比の４要件で行う。⑥仕損の認識並びに測定は、個々のタスク並びにモジュールを各々生産性と品質の指標で行う。⑦認識規準は「過去平均実績緩衝規準」とし、認識と測定を連続的に行う１段階アプローチとする。

**【キーワード】**

・ソフトウェア（Software）

・仕損（Spoilage）

・原価計算（Cost Accounting）

・減損会計（Impairment of Assets）

・兆候の識別・認識・測定（Evidence of Identification ・Recognition・Measurement）

# 第１章　はじめに

ソフトウェア開発に関しては、ＰＭＢＯＫ（Project Management Body of Knowledge：プロジェクトマネジメント知識体系）を始めとするプロジェクト管理のモデルや方法、開発方式や開発技法が数多く提唱されてきており、実際のプロジェクトでも適用されている。それでも、プロジェクトの失敗は後を絶たない。プロジェクトの失敗はなくならないどころか、顕著に減少してもいない[1]。

ところが、プロジェクトの失敗は一般的に知られることはほとんどない。マスメディアで報じられるシステム・トラブルの多くは、情報漏洩やサイト攻撃、あるいは金融機関又は取引所などのシステム・ダウンであり、運用トラブルである。しかし、運用トラブルはあくまで問題が表面化した「氷山の一角」であり、且つ運用における直接的な不具合による場合もあるが、問題の大半はそれ以前のソフトウェア開発に基因しているのである。開発におけるプロジェクトの失敗が潜在的なリスクを作り込んでおり、それが顕在化すると運用トラブルとなって表面化すると言える。そして、経営戦略ないし事業活動を推進する上でのＩＴ（中核はソフトウェア）の重要性を考えれば、システム・リスクは事業リスクの大きな部分を占めている。そのリスクを知り得るために、ソフトウェア開発の成否に関わる情報が必要である。

事例を取り上げて、問題をもう少し具体的に浮き彫りにしてみたい。事例は、まだ記憶に新しいみずほ銀行のシステム・トラブルである。東日本大震災の３日後、2011 年 3 月 14 日に義援金の振り込みが集中したことをきっかけに発生し、収束までに 10 日間を要した。他のメガバンクなどが取り立ててトラブルもなく対応したことに比して際立っていたが、それだけではない。真因が、遡る 2002 年の３行合併に伴うシステム統合時にトラブルを起こし、抜本的な改善を求められたにも関わらず、問題を先送りし続けてきた歴代経営陣の「ＩＴ軽視、あるいはＩＴへの理解不足」[2]だからである[3]。そして、「九年前のシステム統合では、最終的に四千億円の費用を投じた。プロジェクトの失敗やシステム障害対応などによって、金額は当初の見積もりの二倍以上に膨らんだ」[4]が、それだ

---

[1] 日経コンピュータ(2008) P.38 における第２回システム開発プロジェクトの実態調査(2008 年実施)に拠れば、814 社の有効回答でのプロジェクト「成功」率は 31.1%であり、第１回調査(2003 年実施)より 4.4%の上昇とのことである。なお、同調査の「成功」は、「品質、コスト、納期の３基準すべてで「当初計画通りの成果」を収めたプロジェクトだけを「成功」と定義した」独特の規準による。その理由は、「確かにプロジェクトの成否をどこで判断するかは難しい問題だが、本誌ではこの３基準を満たしていなければ確実に成功とは言い切れないと考え」た、と説明している。同調査での「成功」の「逆数」が即「失敗」とは言えないが、それでも成功率からすれば、「失敗」が少なくないことは察せられよう。５年間の推移も、同誌も認めている通り、顕著な好転とは言えない。

[2] 日経コンピュータ編(2011)P.２等。なお、トラブルの経緯等は主として同書を参照した。

[3] 2012 年 6 月 29 日に金融庁が「主要行等向けの総合的な監督指針」等及び『金融検査マニュアル』等(http://www.fsa.go.jp/manual/manualj/yokin.pdf#seach='金融検査マニュアル')の改正を行ったが、これはみずほ銀行のトラブルを念頭に置いたものであり、個別の改善命令だけでは足りず、また同様のリスクは他行にもあり得るとの判断によるものである。「監督指針」(「別紙１」P.1)ではそれまで「経営者」としていたものを「代表取締役」とし(経営トップの責務であることを強調した意味に解せられる)、新設項目として「② 代表取締役は、システム障害の未然防止と発生時の迅速な復旧対応について、経営上の重大な課題と認識し、態勢を整備しているか」、「③ 取締役会は、システムリスクの重要性を十分に認識した上で、システムを統括管理する役員を定めているか。なお、システム統括役員は、システムに精通していることが望ましい」、といったことを挙げている。

[4] 日経コンピュータ編(2011)P.76

けの巨費を投じたにも関わらず、再び甚大な信用失墜や顧客離れなどを招来する潜在的なリスクを抱えたものだったのである。これが適切に情報開示されていれば、その時点で信用失墜や顧客離れなどが既に発生したであろうが、実態を結果的に糊塗・隠蔽することを容認する財務情報がミスリードしてきたと言わざるを得ない[5]。しかも、問題はそれだけではない。「はっきりいって、みずほ銀行を批判できる企業はそう多くはない。多くの企業の情報システムもいつ問題が起きるか分からない。トラブルの内容によっては、みずほ銀行の例など比較にならないほどの実害が発生するリスクがある」という意見があるが[6]、筆者も同感である[7]。こうしたことを開示しない財務情報は、今日的には明らかに情報の不備・不足と言わざるを得ないであろう。

繰り返しになるが、財務報告でソフトウェア開発プロジェクトの失敗が情報開示されることはほぼ無いと言える。原価計算においても、後述するように、ソフトウェア開発における仕損費の把握は困難として、的確に捕捉しないまま放置されてきた。それに加えて、現行の会計制度では、支出が伴えば、開発に掛かったコスト[8]は、費用あるいは資産として会計処理されている。つまり、仕損を仕損処理せずに資産ないし費用として処理することを容認してきたのである。これでは、利害関係者がプロジェクトの失敗を知り得る術はないと言わざるを得ない。それ故に、企業活動の写像である財務会計は、ソフトウェア開発プロジェクトの失敗を適正に捕捉し、財務報告で開示させるべきではないのか。これが、筆者の問題提起である[9]。

本稿では、ソフトウェア開発プロジェクトの失敗を会計的に捉えるために、ソフトウェア開発における仕損に関して、その兆候を識別し、続いて認識並びに測定を行う会計処理案を提示する。それにより、これまで「正常な事象」と区分されることなく資産ないし費用として処理されてきたソフトウェア開発の仕損費を適正に捕捉するべきことを提案したい。

本稿の構成は、次の通りとする。まず、ソフトウェア開発プロジェクトの失敗とは如何なる事態なのか、会計的な考察の前提として、必要な範囲で説明する。次に、予備的考察として、ソフトウェアの仕損に関する先行研究を瞥見し、それらの取り扱いではソフトウェアの仕損を捕捉すること

---

[5] 最近、新たなシステムの全面再構築の計画が報道された(日経コンピュータ(2012)P.10)。再び4,000億円を超える開発費が見込まれているが、計画内容の概要からは、新たな事象・事態への対応は限られていて、大半は2002年のシステム統合における課題・要件への対応と変わらない。前回投じた巨費がほぼ無駄金であったことを明かして余りある。また前回と同様、サプライヤへの配慮から4社(日本IBM、富士通、日立、NTTデータ)に委託する体制を取るようだが、みずほ自体が余程強力なＳＩ(システム・インテグレーション)能力を発揮しない限り、整合的且つ遺漏のない構築は難しく、「三度目の正直」となるか、「二度あることは三度ある」となるかは、まだ予断を許さない。

[6] 日経コンピュータ編(2011)P.3-4

[7] 具体的な根拠としては、今日の大規模システムは行数規模で１億行を超えるが(日経コンピュータ編(2011)P.183)、バグ残存率を0.02/KLOCと推定すると(一般的には良好な品質である)、2,000個(=0.02/KLOC×100,000KLOC)超のバグが潜在していることになる。ごく軽微なものからシステム・ダウン等に到る重大なバグまでが散在しているとして、開発時に作り込まれたトラブル原因に限っても、これだけのものを抱えて日々運用している。こういったリスクは周知されなければならない。

[8] 本稿で用いる「コスト(cost)」という用語はすべて、費用あるいは資産に認識される以前の、発生現象を意味する。

[9] みずほ銀行のような事態に関しては、包括的には内部統制全般による対処を必要とするが、個々の具体的な事象への対処として、本稿の主題とする仕損会計の意義は十分に有している。

が困難である点を明らかにする。続いて、それを踏まえ、減損会計に準じた会計的な構想を提示する。そして、本稿の主要部分である「ソフトウェアの仕損に係る適正な会計処理」では、仕損の兆候の識別、認識並びに測定をどのように行うかを説明する。併せて理解に供するために、試算例を掲示する。また、検証可能性並びに仕損会計処理を実施する際のコスト／ベネフィットに言及する。最後に、本稿の限界並びに今後の課題を述べることにする。

## 第２章　ソフトウェア開発プロジェクトの失敗の概観

まず、本稿が取り上げるソフトウェア開発プロジェクトの失敗という事態を概観することから始める。ソフトウェア業界で遍く共通認識となっているのは、プロジェクト管理の３大指標である「ＱＣＤ」（品質(Quality)、コスト(Cost)、納期又は開発期間(Delivery)）が計画より大幅に悪い結果となった場合である。つまり、品質が劣悪か、又はコストが大幅に超過したか、又は納期が遅延(開発期間が延長)したか、ということである。具体的な事象は、ケーパーズ・ジョーンズ(Capers Jones)の『ソフトウェア病理学』などに具に書かれている。また、エドワード・ヨードン(Edward Yourdon)の『デスマーチ　第２版』では、そうした悲惨な失敗プロジェクトという事態を「デスマーチ」（死の行進）という刺激的な名辞で形容し、それを冠した一書を物している。高名なソフトウェア工学者であり、1960年代の「ソフトウェア危機」に際して開発技法で対処するために考案された構造化技法の考案者の１人であり、ソフトウェア業界を知悉したエドワード・ヨードンは、次のように言明している。「私の結論は、デスマーチ・プロジェクトは、常態であって、例外ではないというものだ。ソフトウェア開発者もプロジェクト・マネジャーも十分賢明で、合理的な方法でプロジェクトを管理したいはずだし、ユーザーや上層部も一世代前よりはるかにコンピュータに明るく、限りある資源で開発できるソフトウェアがどの程度のものかもわかってきたと思う。しかし、開発側もユーザー側も賢明になっても、デスマーチ・プロジェクトの発生はとめられない。ビジネス上の競争激化による圧力や、新技術の登場による競争激化が駆り立てるからだ」（傍点邦訳）[10]。

### １　ケーパーズ・ジョーンズの『ソフトウェアの成功と失敗』

ケーパーズ・ジョーンズの『ソフトウェアの成功と失敗』では、プロジェクトの失敗を「絶対的失敗」と「相対的失敗」に大別し、「絶対的失敗とは、プロジェクトがその完成の前に中止される場合と定義」し、「相対的失敗とは、プロジェクトがその予定されたスケジュールあるいはコスト目標と比較して大きく超過した場合と定義」[11]している。そして「相対的失敗」の例として、「ソフトウェアプロジェクトの引渡しは行われたが、その予定コストおよびスケジュールにおいて50%以上の超過を生じた場合」と、「ソフトウェアプロジェクトを引き渡した後、品質レベルが悪く、顧客の全

---

[10] Yourdon, Edward(2004)邦訳 P. viii-ix
[11] Jones, Capers(1995)邦訳 P. 4

面的利用が６カ月以上も遅れた場合」を挙げている[12]。

「絶対的失敗」は、プロジェクトの中止という外見的にも明白な事態なので定義は明瞭であるが、「相対的な失敗」は「大きく超過した」と大まかな定義に留まっている。例示では、予定に対し「50%以上の超過を生じた場合」や、予定より「６カ月以上も遅れた場合」を挙げている。実際には「成功」と「失敗」の中間に、「成功」とは言えず、「失敗」とも言えないプロジェクトが多くあるが、敢えて対比的に取り上げて[13]、際立たせ、問題を浮き彫りにしていると言えようか。

なお、ジョーンズは「相対的失敗」に関して、定義にはスケジュールとコストしか内包させていないが、例示では図らずも品質がトリガーとなった事態を挙げている。これも定義に含めたほうがよいであろう。ソフトウェア開発のプロジェクト管理で必ず取り上げられる３大指標だからである。すなわち、ソフトウェア開発プロジェクトの失敗とは、差し当たり、予定に対する大幅な、①納期遅延、②品質不良、③コスト超過である、と確認しておく。

## 2　JUAS「企業ＩＴ動向調査」における３大指標の調査

JUAS （Japan Users Association of Information Systems：日本情報システム・ユーザー協会）の「企業ＩＴ動向調査」には、３大指標に関する調査が含まれている[14]。システム開発工期[15]に関しては、「予定通り完了する」、「ある程度は予定通り完了する」、「予定より遅延する」、「該当プロジェクトなし」の何れかをプロジェクト規模（工数[16]規模）毎に選択回答するように求めている。結果は、「予定より遅延する」が 100 人月未満のプロジェクト（回答数 n=649）では 22%、100～500 人月未満（n=341）では 32%、500 人月以上（n=199）では 42%である。規模が増大すると、遅延比率が増大する傾向を示している。

品質に関しては、「満足」、「ある程度は満足」、「不満」、「該当プロジェクトなし」の何れかをプロジェクト規模毎に選択回答するように求めている。回答結果は、「不満」が 100 人月未満（n=651）では 13%、100～500 人月未満（n=341）では 22%、500 人月以上（n=199）では 31%である。規模が増大すると、やはり不満の比率が増大する傾向であることを示している。なお、品質は同調査では広義の品質を意味し、「満足」「不満」という定性的な回答を求めているが、失敗や仕損を定量的に把握し、会計処理に繋げていくには、採用し難い指標である。また、工期やコストが予定対比での回答を求めているのに対して、品質はそれ自体での評価を求めているが、失敗や仕損を把握する際には統一的でなければならないだろう。

コストに関しては、「予定通り完了する」、「ある程度は予定通り完了する」、「予定より超過する」、

---

[12] 同書邦訳 P. 4
[13] 同書邦訳 P. 15-16 など
[14] JUAS(2011)P. 188-190
[15] 「納期」「工期」「開発期間」「スケジュール」を、引用との関連や文脈によって混用しているが、これらは同義的である。納期はどちらかと言えばソフトウェア企業が受託の立場で使うことが多く、工期はユーザ企業が自社の立場で使うことが多いが、ユーザ企業でもユーザ(部門)に対しては納期という場合もある。
[16] 工数は、１人の技術者の単位期間当たりの作業量を、人月、人日、人時の尺度で示すものである。

「該当プロジェクトなし」の何れかをプロジェクト規模ごとに選択する。回答結果は、「予定より超過する」が100人月未満（n=648）では15%、100～500人月未満（n=342）では29%、500人月以上（n=200）では40%である。コストの規模との対応傾向も、工期・品質と同様であり、何れも開発規模が大きいと、それだけ失敗する傾向が高くなることを示している。

なお、同調査では、工期の「予定通り」と「遅延」の間に「ある程度は予定通り」という選択肢があり、工期に関する傾向性を3つの離散的な事態に分別して捉えようとしている。つまりは、調査の趣旨は失敗か否かという指標で捉えるのではないが、「予定通り」と「ある程度は予定通り」という選択肢があることからすれば、それら以外の「予定より遅延する」は失敗と見做しても差し支えないであろう。だが、「予定より遅延する」に関してそれ以上の調査項目はないので、予定よりどの程度の遅延かは不明である。品質とコストに関しても、同様である。傾向を知る意味では得ることの多い調査であるが、定量的なアプローチに関しては別途考えざるを得ないであろう。

# 第3章　ソフトウェアの仕損に関する予備的考察

## 1　仕損の予備的考察

仕損は、原価計算基準の第一章五「非原価項目」及び第二章二七「仕損および減損の処理」並びに三五「仕損の計算および処理」に拠れば、正常仕損と異常仕損に分けられ、正常仕損は原価項目、異常仕損は非原価項目とされる。

太田昭和監査法人は、仕損費に関して、次のように扱うことを提唱している。「ソフトウェア開発がうまく行かないために支出した、再作成のための費用あるいは修正のための費用は仕損費となる。一般に正常な仕損費については原価に算入し、異常な仕損費は特別損失に計上するものとされている。ソフトウェア開発においては、この仕損費を把握することが難しく、仕損を正常と異常とに区分することは、さらに困難である。なぜならソフトウェア開発においては、つねに何んらかの修正・手戻りが発生するものである。これを仕損として認識してよいか問題があるからである。したがって、修正が大幅であって金額的にも多額になる場合を除いて、すべてソフトウェア開発費に含ませることが実務的である」[17]。補足すると、ソフトウェア開発では正規に必須の工程としてテスト工程を組み入れており、それは間違いなどの発生を当然視してのことであり、「修正・手戻りが発生する」ことは直ちに仕損ではないということである。この点に関しては、筆者も同感である。しかし、「修正が大幅」や「金額的にも多額になる場合」というだけで、その具体的な「規準」が示されていない曖昧な扱いに留まっており、事実上は仕損を扱わないに等しいものである。更に、大幅な修正に多額の費用が掛かる場合以外は、「すべてソフトウェア開発費に含ませる」という処理を容認していることと相俟って、プロジェクトの失敗は会計において捕捉されずにきたのであろう。

櫻井通晴は、「仕損費の計算および処理」に関して、次のように述べている。「仕損は、原則とし

---

[17] 太田昭和監査法人(1992)P.58(関連箇所P.57)

て、次の２つのケースから生じる。第１は、設計上のミス、コーディング・ミスなどで不完全なソフトウェアとして仕上がったが、手なおしによって完成品に回復できるケースである。ソフトウェアは工業生産物とは違って試行錯誤的に手なおしすることが多いため、この意味での仕損費の把握は困難である。第２は、手なおしをしようにも回復できないため、代品を製作するケースである」[18]。第１のケースに関しては「「原価計算基準」の 35 の(1)に準じて、当該ソフトウェアの仕損費（直接費扱）として処理し、製造原価に加算するのが妥当であろう」[19]とし、第２のケースに関しては「「原価計算基準」の 35 の(2)の１に準じて、旧製作品指図書に集計された原価を仕損費とし、新しい製作品指図書にたいして直接費として賦課すべきである」とする[20]。しかしながら、第１のケースは一方では「仕損費の把握は困難である」と言いながら、他方ではバグをすべて仕損と見做すに等しいものである。「試行錯誤的に手なおしすることが多い」ことに関連した、テスト工程を必須に組み入れていることを、十分に配慮してはいないように思える。第２のケースに関しては、事象としては明解であり、仕損の把握は容易であるから、捉え方は妥当と言えるが、原価計算ないし会計処理としては新旧の「製作品指図書」といった「工業生産物」に準じた扱いとするのは、そもそも開発プロジェクトで多くのコンポーネント[21]を開発する際に個々のコンポーネント単位に「製作品指図書」を発行するわけではないから、実務的な取り扱いとしては少々難点がある。

また、上記２つのケースは正常仕損に関することだが、異常仕損に関しては、「異常な仕損費は、非原価であるから、営業外費用か特別損失として処理すべきである。ただ、災害などのような質的な異常性が原因であれば判定が容易であるが、量的な異常性の判定は「重要性の原則」により、個々に判断されなければならない」[22]としながらも、「ソフトウェア開発においては工業生産物とは違い、異常工数の発生は通常の作業のなかで吸収されてしまうことが多い。また、どこからが生産活動でどこまでが研究開発であるかの設定もむずかしい[23]。そのため、現実には異常仕損費の把握がきわめて困難である」[24]として、結局は「困難」を理由にして具体的な処理方法などを提示していない。

そして、「しかし、効果的な原価管理のためには、異常工数の把握は必須である。これの効果的運用を図るためには、適正な標準工数を算定し、標準原価計算制度をもつことが望まれる」[25]と、異常仕損の把握を将来的な標準原価計算の実施に託すに留まっている。しかしながら、櫻井の構想す

---

[18] 櫻井通晴(1992)P. 75
[19] 同書 P. 75
[20] 同書 P. 75
[21] コンポーネントは、この文脈では、広義のプログラム(群)ないし機能(群)を意味している。
[22] 櫻井通晴(1992)P. 75
[23] 櫻井は、別の箇所では「ソフトウェアの製作は必ずしもすべてが研究開発活動のようにリスキーなものばかりではなく」(同書 P. 41)と言いながら、「どこまでが研究開発であるかの設定もむずかしい」という指摘に留まっている。また、この文脈で「研究開発」を持ち出すことは不適切ではないだろうか。そもそも研究開発であれば、「試行錯誤」の繰り返しと言える性格上、仕損は発生しない、あるいは仕損とは見做さないのではないだろうか。
[24] 櫻井通晴(1992) P. 76
[25] 同書 P. 76

る標準原価計算は工程別（フェーズ別）という粒度なので[26]、例え実施したとしても、粒度が粗過ぎて、櫻井の期待に反し、それによる異常仕損の把握は「困難である」。代替案を提示する際にも触れるが、ソフトウェア開発は余程小規模な開発でない限り、同一工程においても複数の要員が作業をするのである。その要員間にはスキルの差が少なからずあり、仕損の発生可能性は一律ではないのである。ところが、フェーズという粒度では、差異が平均化されてしまい、把握が困難になる。もう１つは、バグはテスト工程で表面化することが多いが、テストに問題があるわけではなく、プログラム製造工程ないし設計工程でバグを「作り込んだ」のである。バグの「作り込み」箇所を特定しなければ、的確な把握は困難である[27]。

これらの予備的考察で明らかになったことは、次の通りである。ソフトウェア開発はテスト工程を必須の工程として組み込んだものであり、逐一のバグを仕損と見做すことはソフトウェア開発の特性を無視しており、適合的ではない。また、把握が困難とされる由因は捕捉の粒度にあり、フェーズという粒度での捕捉では不十分で、更に精細な粒度で捉える必要がある、ということである。

## 2　会計的構想

これに対して、ソフトウェアの仕損を捕捉すべく会計処理案を提示したい。ソフトウェア開発プロジェクトの失敗が後を絶たず、ほぼ恒久的に今後もなくならないと考えており、仕損の処理が必要不可欠だからである。基本的な考え方を述べ、筆者案（以下、「仕損会計」と称す）のアウトラインを明らかにする。

第１に、仕損会計は、減損会計と類似的な性格のものとし、基準構成を同様にする。減損会計とは、ＩＡＳ（International Accounting Standards ）36 の「資産の減損（Impairment of Assets)」（以下、IAS36 と略記）を念頭に置いており、それを参考にして構想を具体化する。基本的な考え方は、減損会計と類似的に、資産の回収可能価額を帳簿価額に付することである。仕損会計は当初の認識時点における回収可能性の評価として、回収対象となる原価か否かを仕分け、回収不能部分を除くものであるのに対して､減損会計はその延長上の事後的な評価として位置付けるものである。回収可能性を評価し、不能分を帳簿価額から減じることは共通的であり、両者は連携関係を持つ一連の流れとなる。ただし、多少の相違点がある。１つは、両者は認識時点の違いから対象が多少異なる。減損会計の対象は既に資産認識された資産であるが、仕損会計の対象は、資産あるいは費用として認識される以前のコストとするので、費用として認識されるものを除外しない。費用処理される場合にも、開発に投じた費用を回収することに変わりはないからである。もう１つは、兆候は

---

[26] 同書 P. 83-92

[27] これに関して誉田直美(2010)では、ＮＥＣが考案し、実施しているソフトウェアの品質管理の手法である「ソフトウェア品質会計」を取り上げている。目標管理において、数値的な計測に留まらず、バグの「作り込み工程」と「摘出工程」の両面で捉えて管理している。摘出は一般的にも捕捉しているが、「作り込み工程」を捕捉する点はユニークな着眼である。仕損に即応して言えば、「摘出工程」で仕損が発覚し、その工程で改修などの対応をするわけであるが、仕損を真に発生させたのは「作り込み工程」である。それを特定することで、仕損の測定は的確になるのである。

それまでの企業努力（ソフトウェア開発を含む）の結果を含めて対象を当初（プロジェクト終了時点）ないし毎期末という時点で識別する点では同様だが、減損会計の場合には内外の諸要因が影響を及ぼすのに対して、仕損会計の場合には専ら内部要因（開発プロジェクトの適否）によることである。基準構成は、共通的に、第１に仕損の兆候を識別する、第２に仕損の認識を行う、第３に仕損の測定を行う、という３ステップを踏むものである。

第２に、考え方以外に、減損会計と異なる点の１つは、減損会計では回収可能価額（Recoverable amount）を正味売却価額（Fair value less costs to sell(売却費控除後の公正価値)）と使用価値（Value in use）のいずれかとするが、ソフトウェアの仕損会計では正味売却価額は選択肢としない。理由は、売却か継続的使用かの判断は、事後的な評価においては同時期に起こり得るのであるが、当初の認識時点では使用することを前提に認識するからである。当初は、そもそもの開発あるいは導入の意思決定がなされ、プロジェクトが遂行されるので、それは既に使用の意思決定が行われたことに他ならない。従って、回収可能性を示す価値は使用価値のみとする。

第３に、回収可能性を示す使用価値としては、基本的には、当該ソフトウェアの継続的使用により生ずるであろう将来キャッシュインフローの獲得のために、又は当該ソフトウェアの使用を可能とするために、必要なキャッシュアウトフローであり、当該ソフトウェアに直接帰属させることができ、又は合理的かつ首尾一貫した基礎により配分できる見積り[28]が妥当であると考える。ソフトウェアが資産認識されることを考慮すると、あくまで現行の取得原価の算定の修正であると位置付けているので、取得原価で認識される他の資産とも整合的であるからである。

第４に、減損会計では対象となる資産を、１つの資産全体あるいは資産グループ全体の単位で取り扱うが、ソフトウェアの仕損会計では、ソフトウェアの仕損の兆候の識別には１つのソフトウェア開発プロジェクト全体の単位で取り扱うのに対し、仕損の認識並びに測定には１つのソフトウェア（開発）を細分化した個々のタスク（プロセス）並びにモジュール（プロダクト）の単位で取り扱うことにする。そして、この単位を、規模、開発期間、品質、コストの項目で捉えるのである。

ソフトウェアを捉える場合には、プロセスとプロダクトの両面から総合的に捕捉しなければならないと考えている[29]。プロダクトとしてのソフトウェアは相当に認知されているが、プロセスとしてのソフトウェア、つまりソフトウェアを開発する側面・様相は、それほど知られていない。前段で、プロジェクトの失敗に関する調査を取り上げ、概観した所以でもあるが、それらに基づいてソフトウェアの仕損を捉える限り、プロセスの側面も捉えるのでなければならない。何よりも、ソフトウェアに係る費用の大半は開発作業に従事した開発要員の人件費であり、開発プロセスをどのように遂行したかに大きく左右されるからである。仮にプロダクトには問題がなく仕損がなくとも、その開発を非効率に行ったとすれば、ソフトウェア開発費は不必要に多額になってしまうのである。

---

[28] IAS36par.39「Composition of estimate of future cash flows」の(b)の規定に依拠している。

[29] 経営学のイノベーション研究では、イノベーションをプロダクトとプロセスのそれぞれの面で捉えることは共通認識とも言える(Tidd, Joe(1997)参照)。そうした捉え方を参考にしているが、それをソフトウェア会計に具体的に適用することは筆者の問題意識に基づいた独自の考案である。

そして、プロセスに関する仕損を捕捉可能とするためには、より精細な粒度であるタスク・ベースでの捕捉が必要であることを提唱したい。開発実務では、粒度や厳格度は様々であるが、プロジェクト管理において作業の定量的な測定はほぼ必ず行われている、と言える。また、開発の作業区分は様々あるが、一般的にはフェーズ＞アクティビティ＞タスクという粒度で区分することになっている。順に作業粒度が細かくなることを示し、タスク・ベースが最小の作業単位となる。開発原価の見積りや実績の測定において、フェーズ・ベースで区分して捕捉することはごく当たり前に行われており、櫻井もこの粒度での標準原価計算を提唱している。それに対し、ケーパーズ・ジョーンズは、「アクテビィティベース」という捉え方を、見積り、更には実績測定をも包含したものとして提起している[30]。これらを踏まえて、更に精細な粒度であるタスク・ベースでの捕捉が必要である、と筆者は考えている。開発原価の見積りや実績の測定で使われることは余りないが、開発プロセスにおける実際の作業はこの粒度で行われているので、開発実務では馴染みのあるものである。

なお、プロダクトに関するモジュールは既述のコンポーネントの最小単位であり、プロセスと同様に精細な粒度で捉える趣旨である。

# 第４章　ソフトウェアの仕損に係る適正な会計処理

## １　定義並びに適用対象

まず、ソフトウェアに対して仕損処理を適用するにあたって用いる用語に関する定義を行う。

（ａ）ソフトウェア開発プロジェクトの失敗とは、プロジェクト管理の３大指標である品質(Quality)、コスト(Cost)、納期又は開発期間(Delivery)（ＱＣＤ）が予定より「大きく超過した場合」である。これを大別すると、プロジェクトを中止・中断する場合と、ともかく継続して完了する場合とに分けられる。

（ｂ）ソフトウェアの仕損とは、１つはプロジェクトを中止・中断した場合であり、それまでに投じた費用のすべてが仕損費である。２つに、継続して完了した場合のプロダクト（成果物）が粗悪品、不良品であるか、又は、プロセス（作業）が非効率で余計な工数を費消し、引いては無駄なコストに帰結するような現象である。

（ｃ）計画比規準とは、ソフトウェアの仕損の兆候を識別する規準である。

（ｄ）過去平均実績緩衝規準とは、ソフトウェア仕損の認識並びに測定を行う規準である。

（ｅ）異常仕損とは、ソフトウェアの仕損のうち、非原価項目として費用処理するものである。

（ｆ）特別仕損とは、ソフトウェアの仕損のうち、異常仕損より更に仕損度合いが甚大な特別損失として取り扱うものである。

（ａ）と（ｂ）の関連性は、本稿では、ソフトウェアの仕損に関して、主としてソフトウェア開

---

[30] Jones, Capers(2007)邦訳 P.287-436 、Jones, Capers(2008)

発プロジェクトの失敗並びにその具体的な現象とするが、厳密には相即的なものではない。失敗プロジェクトにおいても、仕損と仕損でないことがあり、精査しなければならない。逆に、開発は失敗でなく、成功裡に完了したプロジェクトでも、仕損が発生していた可能性はある。従って、失敗プロジェクトのみならず、プロジェクト全般を対象として仕損を取り扱うことが本来的である。そうでありながら、何故プロジェクトの失敗に照準を定めるかと言えば、筆者は、２段階で仕損処理の適用を構想しているからである。第１段階では、プロジェクトの失敗に照準を定め、そこにおける仕損を適切に捕捉する処理を確立し、会計慣行としての定着化を図る。続いて第２段階では、仕損処理をソフトウェア開発一般にも適用拡大する、ということである。２段階構想とする理由は、ソフトウェア開発実務並びにその会計実務の現状を考えれば、仕損の認識や測定を実施可能とする管理粒度を装備し、処理コストを負担することはどの企業でも容易に対応できるとは限らないと考えるからである。それに対し、プロジェクトの失敗という識別は容易であり、失敗を招来した経営責任において、仕損測定の負担を負うことは当然の責務と考える。

（ｃ）と（ｄ）については、兆候の識別又は認識並びに測定の該当箇所で説明する。

（ｅ）と（ｆ）について、これらを設定した前提となる捉え方をまず説明しておきたい。原価計算基準における正常仕損は、ソフトウェアに当て嵌めるのは不適格だということである。ソフトウェア開発は、非定型的な「知識労働」であり、一般的な検査とは明らかに異なった相当程度の期間と工数を要するテスト工程を組み込んでいるのである。そして、成果物であるソフトウェアは完成品であってもバグ（不具合）がほぼ必ず潜在している。従って、軽微な正常仕損を捕捉することは難しいだけではなく、こうした特性から無用である、と考える。よって、後述するように一定の許容範囲を設定した範囲内にあるので仕損とは見做さず、ソフトウェア開発費に含めて会計処理するので、結果として原価計算基準に準じた取り扱いとなる。それ故、（ｅ）と（ｆ）を捕捉対象とする。（ｅ）と（ｆ）を区別するのは、甚大な仕損度合いである（ｆ）は開発時点の失敗に留まらず、運用に入っても障害などが発生する可能性が大きく、それが予め高い確率で予想できるので、それに見合った処置を施しておくことが相当である、と考える。

次に、適用対象は、研究開発目的以外のソフトウェア（開発）とする。日本の現行のソフトウェアに係る会計基準の区分で言えば、市場販売目的、自社利用、受注制作の何れにも一律に適用するものとする。何故ならば、制作目的によって、ソフトウェア開発の仕方が異なることは基本的にないからである。開発の進め方、開発プロジェクトの編成など、何れも大差ないのである。また、技術的な難易度は、個々には当然に差異はあるが、制作目的によって差異があるわけではない。更に、市場販売目的または自社利用で、自社開発といっても、自社の社員だけで開発することはほとんどなく、大なり小なり外部委託していることも変わりない。契約形式上は請負か準委任か派遣かという違いはあるとしても、あくまでソフトウェア開発の実態としては基本的に違いはないのである。むしろ、開発の工数や費用が増大した場合に、委託－受託の契約内容如何によって何れが負担するかは異なることが、一律に適用する際の留意事項である。追加発生費用などをすべて受託側が負担

する場合には、委託側では仕損費が発生しないこともあり得るからである。なお、制作目的によって個別原価計算あるいは総合原価計算と異なる適用をした場合にも、異常仕損と特別仕損の捉え方は一律であり、変わるものではない。

また、ＩＡＳ38「無形資産（Intangible Assets)」に照らして言えば、外部取得（購入）と自己創設（自社開発）という取得形態による区分のうち、自己創設に適用するものである。

研究開発目的を対象外とするのは、そもそも研究開発の試行錯誤を仕損と見做すことは、研究開発の性格に照らして不適切であると考えるからである[31]。

## 2　ソフトウェアの仕損の兆候の識別

ソフトウェア仕損の兆候を識別する規準は、「計画比規準」であり、ソフトウェア開発プロジェクトが失敗か否かという識別とし、規定は次の通りとする。なお、一般的にはプロジェクトの失敗は３大指標であるＱＣＤで捉えるが、筆者はそれにソフトウェアの規模を追加することを提言する。それによって、失敗が十全に捕捉可能となるからであり、理由は後述する。規定はOR条件であり、何れかが該当すれば、プロジェクトは失敗と見做し、仕損の兆候があると識別する。なお、プロジェクトを中止・中断した場合は、兆候の識別に留まらず、直ちにすべてを特別仕損とする。

①ソフトウェアの規模が、概ね計画比20%を超過した場合、プロジェクトは失敗と見做す。
②ソフトウェアの開発期間が、計画より延長した場合、プロジェクトは失敗と見做す。
③ソフトウェアの品質が、概ね計画比20%超悪化した場合、プロジェクトは失敗と見做す。
④ソフトウェアのコストが、概ね計画比20%を超過した場合、プロジェクトは失敗と見做す。

まず、すべてを計画との対比としているが、失敗か否かを実務的に捉える場合[32]のソフトウェア業界での共通認識であり、業界慣行と言えるものなので妥当であると考える。次に、規準の数値を「20%超」[33]としたが、これは明快に提示するための参考値であり、原則主義的な規定とし、幅を持たせてもよいし、企業が適宜設定可能とするのでもよいと考えている。ただし、失敗と見做されることを回避する操作をさせない限度規定は必要であろう。何れにせよ、選択理由を含めて開示す

---

[31] これに関連して、筆者はアメリカの会計基準も含めて現行のソフトウェア会計基準が研究開発目的のみならず、それ以外のソフトウェア開発をも研究開発と見做していることに、根本的な異議を持っており、これらのことを主題的に取り上げた論稿を準備している。

[32] 開発の事後評価では、計画対比以外に開発経験により習得できたことなど、または計画対比でもQCDより精細な事項に関して行うこともあるが、この文脈では関係のないことなので立ち入らない。

[33] 例えば、①人月単価の契約で150H-180Hまでは固定で、それを上下する場合には時間精算するといったこと(アローワンス20%程度)、②プロジェクトの進捗管理で20%までの遅延は様子見とし、それを超えた遅延は対策を講じることが義務付けられるといったルールが割合多いこと、③契約書上に明記されることはほとんどないが、請負において規模や工数の増加が20%程度を超えた場合には、追加契約交渉をする場合が慣行的に少なくないこと、④マイクロソフトでは開発期間に関して「予備期間」(想定外のことに対応する緩衝期間)を20%程度設定していること(Cusumano, Michael A.(1995)邦訳 P.283-284)、などを参考にしている。

ることで、企業の姿勢や効率性を窮知することができる。

個々の説明に入る。①の規模は、ソフトウェアの機能が増加し、利便性・有用性が増した場合、より良いソフトウェアになったのだから、失敗ではないという抗弁を封じるためである。無条件にソフトウェアの価値評価をしているわけではなく、開発計画を策定し、それが承認されて開発を実施するという事業活動であるのだから、計画と違えた結果は失敗なのである。また、規模を規定に加えたのは、②開発期間や④コストが同様の意味で、規模が増加すれば期間やコストが増加することは対応関係としては合理性を有するが、それにより②や④の規定を無効化させたり、擦り抜けさせないためである。更に、後述するように、規模は仕損の測定では必須の基本算出項目である。

②の開発期間は、他の規定と異なって許容のアローワンスを持たせていないが、プロジェクトで遵守すべきミッションの筆頭に挙げられることであり、そもそも運用に入れず、利用できないということにおいて、掛け値なしに失敗だからである。

③に関しては、１つは、品質には広狭様々な捉え方があるが、仕損の捕捉では狭義の品質、すなわちバグの有無とする。広義の品質は、一般的な評価としては妥当でも、失敗あるいは仕損に照準を定めている場合には余計である。２つには、バグは「実数」で測定可能だが、広義の品質は主観的な「指数」で測定せざるを得ないものがあり、測定の硬度が劣るからである。なお、バグの具体的な指標は後で詳述する。

④のコストは、開発費総額で計画対比を行う。

### 3　ソフトウェアの仕損の認識

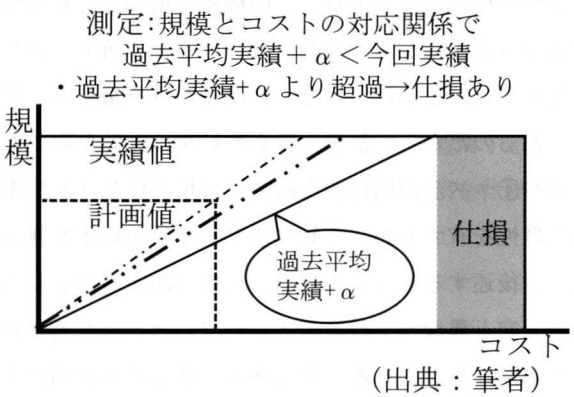
ソフトウェア仕損の兆候があると識別された場合には、次に仕損の認識を行うことになる。仕損の兆候の識別は計画比で、１つのソフトウェア開発プロジェクト全体の対応関係により行うのが妥当であるが、仕損の認識は今回実績を過去実績比で、ソフトウェアにおける個々のタスク（プロセス）並びにモジュール（プロダクト）の単位における対応関係により行うことにする。仕損の兆候が識別された以上は、更に踏込んだ判定をする必要があり、計画値の設定には当為や期待といった作為が混入しており、実績値と性格が全同ではなく、また情報として不確実性を呈している。それに対して、過去実績値は情報として確実なものであり、性格は全同で、近似的な諸条件で達成できた数値としては、今回実績の妥当性を比較・判定する尺度として最も適格性を有していると言える。また、認識と測定を同様の規準で連続的に行える、いわゆる１段階アプローチ[34]が適切だと考えているので、先述のように精細な粒度で、測定を主眼とした規準とするためである。

なお、プロジェクトを中止・中断した場合は、この認識の規準に関わらず、すべてを特別仕損とする。測定に関しても、同様である。

---

[34] 平松一夫(2012)P.104では、IAS36「資産の減損」の１段階アプローチに対して、日本の減損基準を２段階アプローチと対比的に取上げている。また、秋葉賢一(2011)P.232でも同様に、１段階方式と２段階方式として取上げている。

図１　ソフトウェア仕損の認識並びに測定



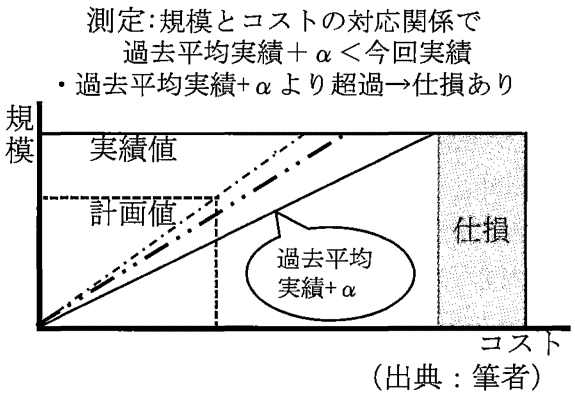

（出典：筆者）

では、仕損の認識並びに測定において用いる「過去平均実績緩衝規準」の考え方を説明する。仕損の認識は、今回実績を過去実績の平均値、例えば過去５ヶ年の開発実績の平均値、との比較で行う。その際に、単純に絶対値での過去実績との比較では意味がないから、単位規模当たりの値（規模との対応関係）で比較を行うことにする。同一尺度での比較なので有意だからである。更に、過去平均実績を超過していても、その超過を直ちにすべて仕損と見做すのではなく、予め許容範囲を設定し、その閾値を超えた場合にその分を仕損と見做すのである。図１に示したように、例えば図の+αに該当する、超過が20%以内は許容範囲とし、それを超えた場合に仕損と見做し、仕損を限定的に捉えるのである。正常仕損を捕捉対象とせず、品質に関して逐一のバグを捉えるのではなく、バグ全体の程度により判断するということである。

過去平均実績緩衝規準並びに個々のタスク（プロセス）並びにモジュール（プロダクト）の単位での捕捉を採用する理由の１つは、１つのソフトウェアを全体として取り扱う場合、個々のモジュールや作業では良悪のバラつきがあっても全体で平均化されてしまうが、個々の細かい単位で取り扱う場合、格段に精確となり、「良」は捨象されるので、一定の緩衝領域を設定するのが穏当と考えるからである。２つには、プロジェクトの失敗の発生率を鑑みると、第１段階で余りに厳格な規準とすることは実施可能性を難しくするからである。３つには、現段階では、過去平均実績値が十分に信頼できる確実な情報として企業が保有しているとは限らないので、それを集積する猶予を配慮することが必要と考えるからである。ソフトウェア開発実務が改善され、ソフトウェア会計実務としても仕損処理が定着した第２段階では、予め許容範囲を設定しない「過去平均実績規準」に変更することが望まれる。

仕損の認識並びに測定に用いる具体的な項目は、生産性並びに品質とする。兆候の識別で取り扱った項目との対応関係は、これまでは明示的に規模との対応関係でＱＣＤを捉えたのに対して、認識並びに測定ではそれとは一見異なるように見えても、取り扱う項目にそれが織り込み済みなので明示的でないだけで、同様なのである。何故ならば、コスト並びに開発期間と、生産性とは、事象として、生産性の悪化が、進捗の遅延による開発期間の延長あるいは工数の増加によるコストの増

図１　ソフトウェア仕損の認識並びに測定




（出典：筆者）

では、仕損の認識並びに測定において用いる「過去平均実績緩衝規準」の考え方を説明する。仕損の認識は、今回実績を過去実績の平均値、例えば過去５ヶ年の開発実績の平均値、との比較で行う。その際に、単純に絶対値での過去実績との比較では意味がないから、単位規模当たりの値（規模との対応関係）で比較を行うことにする。同一尺度での比較なので有意だからである。更に、過去平均実績を超過していても、その超過を直ちにすべて仕損と見做すのではなく、予め許容範囲を設定し、その閾値を超えた場合にその分を仕損と見做すのである。図１に示したように、例えば図の+α に該当する、超過が 20%以内は許容範囲とし、それを超えた場合に仕損と見做し、仕損を限定的に捉えるのである。正常仕損を捕捉対象とせず、品質に関して逐一のバグを捉えるのではなく、バグ全体の程度により判断するということである。

過去平均実績緩衝規準並びに個々のタスク（プロセス）並びにモジュール（プロダクト）の単位での捕捉を採用する理由の１つは、１つのソフトウェアを全体として取り扱う場合、個々のモジュールや作業では良悪のバラつきがあっても全体で平均化されてしまうが、個々の細かい単位で取り扱う場合、格段に精確となり、「良」は捨象されるので、一定の緩衝領域を設定するのが穏当と考えるからである。２つには、プロジェクトの失敗の発生率を鑑みると、第１段階で余りに厳格な規準とすることは実施可能性を難しくするからである。３つには、現段階では、過去平均実績値が十分に信頼できる確実な情報として企業が保有しているとは限らないので、それを集積する猶予を配慮することが必要と考えるからである。ソフトウェア開発実務が改善され、ソフトウェア会計実務としても仕損処理が定着した第２段階では、予め許容範囲を設定しない「過去平均実績規準」に変更することが望まれる。

仕損の認識並びに測定に用いる具体的な項目は、生産性並びに品質とする。兆候の識別で取り扱った項目との対応関係は、これまでは明示的に規模との対応関係でＱＣＤを捉えたのに対して、認識並びに測定ではそれとは一見異なるように見えても、取り扱う項目にそれが織り込み済みなので明示的でないだけで、同様なのである。何故ならば、コスト並びに開発期間と、生産性とは、事象として、生産性の悪化が、進捗の遅延による開発期間の延長あるいは工数の増加によるコストの増

加、に帰結するという意味で、生産性がコスト並びに開発期間を集約的に表現している項目なのである。また、ソフトウェアの定量化手法であるソフトウェア測定（Software Measurement）においても、生産性の実績値は、直接的に測定できず、実績生産性＝実績規模／実績工数という算定式により、実績規模と実績工数から導出するのである。それ故、生産性には規模と開発期間が集約され、それらの測定値を裏付けとしてコストを算出するわけである。先に、仕損測定で規模が必須の基本算出項目であると言及した所以である。

それに対して、品質はこれまでと全く同様である。具体的には、a.バグ発生率（規模単位当たりの発生件数）、b.バグ除去率（規模単位当たりの発生件数に対する除去件数）、c.バグ残存率（規模単位当たりの残存件数の推定値）の３項目とする。c.は最終的なプロダクトの品質を表す指標であり、a.とb.はプロダクトの中間的な品質指標であり、且つ品質に関するプロセスを表す指標でもあるので、この３項目で品質を捉えれば必要且つ十分であると考える。

なお、上記の図では規模とコストの対応関係を示したが、実際には下記の表１に示すように生産性並びに品質に関して比較を行い、過去平均実績＋αより超過している場合には、仕損と認識する。

## ４　ソフトウェアの仕損の測定

ソフトウェア仕損が認識された場合には、引き続き仕損の測定を行うことになる。実際には認識と測定は一連の流れで行うが、説明上は分けて行う。測定の規準は「過去平均実績緩衝規準」であり、測定の単位と項目は、タスク（プロセス）並びにモジュール（プロダクト）毎の生産性並びに品質である。認識で示したものと同様であり、認識の連続的プロセスとして、単位毎の実績値に対する、過去平均実績値により、仕損を測定する。

具体的な測定単位当たりの測定方法（計算方法）を提示する。各項目の定義は、次の通りである。

（ｇ）実績率＝今回実績値／過去平均実績値。百分率ではなく、小数点数とする。仕損費を円単位まで計算可能とするため、小数第６位までを有効とする。なお、仕損費は円未満切り捨てとする。

（ｈ）今回実績値（単位規模当たり）＝今回実績値／今回実績規模

（ｉ）過去平均実績値（単位規模当たり）＝｛（過去実績値 *1*／過去実績規模 *1*）＋（過去実績値 *2*／過去実績規模 *2*）＋…＋（過去実績値 *n*／過去実績規模 *n*）｝／ｎ

（ｊ）生産性実績率＝生産性今回実績値／生産性過去平均実績値

（ｋ）品質実績率＝品質今回実績値／品質過去平均実績値

上記の各実績率以外の項目は、以降の計算式で導出されるものである。なお、項目（変数）は、日本語表記のほうが解りやすいので、記号化表記せず、そのままとする。

判定式並びに計算式の前提は、４つの区分を設けたことである。１つ目は、生産性実績率並びに品質実績率の「過去平均実績範囲内」とする。２つ目は、悪化程度が 20%以内を「許容範囲内」の緩衝域とする。３つ目は、「異常仕損」は悪化程度が 20%超～50%以内とする。４つ目は、「特別仕

損」の悪化程度が 50%超とする。以下の斜字体の定数は、それに基づいたものである。区分範囲の境界値を異なる設定にする場合は、定数を変更すればよい。

仕損費を具体的にどのように計算するか、一覧的に挙示した表１に沿って説明する。

表１　生産性、品質の範囲区分

| | | 品質 | | | |
|---|---|---|---|---|---|
| | | 過去平均実績範囲内 | 許容範囲内 | 異常仕損 | 特別仕損 |
| 生産性 | 過去平均実績範囲内 | | | ⑤ 品質異常仕損 | ⑦ 品質異常仕損<br>品質特別仕損 |
| | 許容範囲内 | | | ⑥ 品質異常仕損 | ⑧ 品質異常仕損<br>品質特別仕損 |
| | 異常仕損 | ① 生産性異常仕損 | ② 生産性異常仕損 | ⑨ 生産性異常仕損<br>品質異常仕損 | ⑪ 生産性異常仕損<br>品質異常仕損<br>品質特別仕損 |
| | 特別仕損 | ③ 生産性異常仕損<br>生産性特別仕損 | ④ 生産性異常仕損<br>生産性特別仕損 | ⑩ 生産性異常仕損<br>生産性特別仕損<br>品質異常仕損 | ⑫ 生産性異常仕損<br>生産性特別仕損<br>品質異常仕損<br>品質特別仕損 |

（出典：筆者）

①生産性で異常仕損が発生し、品質は過去平均実績範囲内だった場合の判定式は、*0.2*＜1－生産性実績率≦*0.5*　＆　品質実績率－１≦0.0、である。異常仕損費の計算式は、生産性異常仕損率＝*0.8*－生産性実績率、異常仕損費＝実績コスト×生産性異常仕損率、である。

②生産性で異常仕損が発生し、品質は過去平均実績範囲外であるが、許容範囲内だった場合の判定式は、*0.2*＜1－生産性実績率≦*0.5*　＆　品質実績率－１≦*0.2*、である。異常仕損費の計算式は、生産性異常仕損率＝*0.8*－生産性実績率、異常仕損費＝実績コスト×生産性異常仕損率、である。

③生産性で特別仕損が発生し、品質は過去平均実績範囲内だった場合の判定式は、*0.5*＜1－生産性実績率　＆　品質実績率－１≦0.0、である。異常仕損費の計算式は、生産性異常仕損率＝*0.3*、異常仕損費＝実績コスト×*0.3*、である。特別仕損費の計算式は、生産性特別仕損率＝*0.5*－生産性実績率、特別仕損費＝実績コスト×生産性特別仕損率、である。

④生産性で特別仕損が発生し、品質は過去平均実績範囲外ではあるが、許容範囲内だった場合の判定式は、*0.5*＜1－生産性実績率　＆　品質実績率－１≦*0.2*、である。異常仕損費の計算式は、生産性異常仕損率＝*0.3*、異常仕損費＝実績コスト×*0.3*、である。特別仕損費の計算式は、生産性特別仕損率＝*0.5*－生産性実績率、特別仕損費＝実績コスト×生産性特別仕損率、である。

⑤生産性は過去平均実績範囲内だったが、品質で異常仕損が発生した場合の判定式は、1－生産性実績率≦0.0　＆　*0.2*＜品質実績率－１≦*0.5*、である。異常仕損費の計算式は、品質異常仕損率＝品質実績率－*1.2*、異常仕損費＝実績コスト×品質異常仕損率、である。

⑥生産性は過去平均実績範囲外であるが、許容範囲内だったが、品質で異常仕損が発生した場合

の判定式は、1－生産性実績率≦*0.2* ＆ *0.2*＜品質実績率－1≦*0.5*、である。異常仕損費の計算式は、品質異常仕損率＝品質実績率－*1.2*、である。異常仕損費＝実績コスト×品質異常仕損率、である。

⑦生産性は過去平均実績範囲内だったが、品質で特別仕損が発生した場合の判定式は、1－生産性実績率≦0.0 ＆ *0.5*＜品質実績率－1、である。異常仕損費の計算式は、品質異常仕損率＝*0.3*、異常仕損費＝実績コスト×*0.3*、である。特別仕損費の計算式は、品質特別仕損率＝品質実績率－*1.5*、特別仕損費＝実績コスト×品質特別仕損率、である。

⑧生産性は過去平均実績範囲外であるが、許容範囲内だったが、品質で特別仕損が発生した場合の判定式は、1－生産性実績率≦*0.2* ＆ *0.5*＜品質実績率－1、である。異常仕損費の計算式は、品質異常仕損率＝*0.3*、異常仕損費＝実績コスト×*0.3*、である。特別仕損費の計算式は、品質特別仕損率＝品質実績率－*1.5*、特別仕損費＝実績コスト×品質特別仕損率、である。

⑨生産性で異常仕損が発生し、品質でも異常仕損が発生した場合の判定式は、*0.2*＜1－生産性実績率≦*0.5* ＆ *0.2*＜品質実績率－1≦*0.5*、である。異常仕損費の計算式は、生産性異常仕損率＝*0.8*－生産性実績率、品質異常仕損率＝品質実績率－*1.2*、異常仕損費＝実績コスト×（生産性異常仕損率＋品質異常仕損率－生産性異常仕損率×品質異常仕損率）、である。

⑩生産性で特別仕損が発生し、品質では異常仕損が発生した場合の判定式は、*0.5*＜1－生産性実績率 ＆ *0.2*＜品質実績率－1≦*0.5*、である。異常仕損費の計算式は、生産性異常仕損率＝*0.3*、品質異常仕損率＝品質実績率－*1.2*、異常仕損費＝実績コスト×（*0.3*＋品質異常仕損率－*0.3*×品質異常仕損率）、である。特別仕損費の計算式は、生産性特別仕損率＝*0.5*－生産性実績率、特別仕損費＝実績コスト×生産性特別仕損率、である。

⑪生産性で異常仕損が発生し、品質では特別仕損が発生した場合の判定式は、*0.2*＜1－生産性実績率≦*0.5* ＆ *0.5*＜品質実績率－1、である。異常仕損費の計算式は、生産性異常仕損率＝*0.8*－生産性実績率、品質異常仕損率＝*0.3*、異常仕損費＝実績コスト×（生産性異常仕損率＋*0.3*－生産性異常仕損率×*0.3*）、である。特別仕損費の計算式は、品質特別仕損率＝品質実績率－*1.5*、特別仕損費＝実績コスト×品質特別仕損率、である。

⑫生産性で特別仕損が発生し、品質でも特別仕損が発生した場合の判定式は、*0.5*＜1－生産性実績率 ＆ *0.5*＜品質実績率－1、である。異常仕損費の計算式は、生産性異常仕損率＝*0.3*、品質異常仕損率＝*0.3*、異常仕損費＝実績コスト×（*0.3*＋*0.3*－*0.3*×*0.3*）＝実績コスト×*0.51*、である。特別仕損費の計算式は、生産性特別仕損率＝*0.5*－生産性実績率、品質特別仕損率＝品質実績率－*1.5*、特別仕損費＝実績コスト×（生産性特別仕損率＋品質特別仕損率－生産性特別仕損率×品質特別仕損率）、である。

仕損を上記の通りに計算した試算結果は次の通りである。

## 表２　仕損費試算表

| ケース | 生産性 | | | 品質 | | | 仕損費 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 実績値(FP/人月) | 異常仕損率 | 特別仕損率 | 実績値(件/FP) | 異常仕損率 | 特別仕損率 | 異常仕損費(円) | 特別仕損費(円) | 仕損費合計(円) |
| ① | 9.00 | 0.050 | | 0.006 | | | 50,000 | | 50,000 |
| ② | 9.00 | 0.050 | | 0.007 | | | 50,000 | | 50,000 |
| ③ | 5.00 | 0.300 | 0.083333 | 0.006 | | | 300,000 | 83,333 | 383,333 |
| ④ | 5.00 | 0.300 | 0.083333 | 0.007 | | | 300,000 | 83,333 | 383,333 |
| ⑤ | 12.00 | | | 0.008 | 0.133333 | | 133,333 | | 133,333 |
| ⑥ | 10.00 | | | 0.008 | 0.133333 | | 133,333 | | 133,333 |
| ⑦ | 12.00 | | | 0.010 | 0.300000 | 0.166667 | 300,000 | 166,667 | 466,667 |
| ⑧ | 10.00 | | | 0.010 | 0.300000 | 0.166667 | 300,000 | 166,667 | 466,667 |
| ⑨ | 9.00 | 0.050 | | 0.008 | 0.133333 | | 176,667 | | 176,667 |
| ⑩ | 5.00 | 0.300 | 0.083333 | 0.008 | 0.133333 | | 393,333 | | 393,333 |
| ⑪ | 9.00 | 0.050 | | 0.010 | 0.300000 | 0.166667 | 335,000 | 166,667 | 501,667 |
| ⑫ | 5.00 | 0.300 | 0.083333 | 0.010 | 0.300000 | 0.166667 | 510,000 | 236,111 | 746,111 |

（注）過去平均実績値：生産性12FP/人月、品質0.006件/FP　　（出典：筆者）
実績コスト1000,000円

なお計算上、生産性実績率は極限的にはゼロに収束するが、品質実績率は拡散し、収束しないために、仕損費が実績コストを上回ることがあり得る。しかし、仕損に関してペナルティを課すことが目的ではないので、その場合には、実績コスト全額を仕損費と見做す取り扱いとする。

さて、最後に、監査可能性ないし検証可能性に言及しておきたい。精細な粒度で精細な数値を捕捉するものであるが、コストに関しては給与明細など、開発が委託－受託であれば請求書並びに検収書などの経理書類、生産性並びに品質に関してはプロジェクト管理資料という証跡があり、算出・集計した処理手順が示されれば、追跡して検証することは、十分に可能である。

また、この会計処理のコスト制約であるが、コストに関しては、それまでの管理粒度が粗い場合には、精細化するためのコストは要する。しかし、開発実務の作業はタスク・ベースで行うことが一般的となっているので、管理の粒度を精細化し、記録するコストに留まり、実際の作業そのものに及ぶものではない。これに対するベネフィットに関しては、目的適合性を充足し、忠実な表現となる情報開示を行う企業であることを示し、利害関係者の信頼を得るという便益が得られ、その関係改善ないし良好な関係構築に寄与することになる。そういう意味では、コスト／ベネフィットが過大な負担ないし著しい不均衡となることはないと考える。

## 第５章　おわりに

「はじめに」で言及した問題提起に対しては、ほぼすべて応え得たものと思う。ソフトウェアの開発実務ではそれほど珍しいことではないにも関わらず、一般的には余り知られていないソフトウェア開発プロジェクトの失敗に対して、また、これまで捕捉が困難とされてきた仕損を、筆者は会計的に適正に捕捉可能とし、仕損費として取り扱う会計処理案を提示した。

会計処理案は、減損会計に準じた会計的な構想に基づき、具体化した。ポイントは、次の通りである。①ソフトウェア業界の慣行である３大指標ＱＣＤでプロジェクトの失敗を捉えることを基本認識とすること、②仕損はプロジェクトの中止・中断と、継続・完了した場合の品質不良と作業非効率に大別すること、③適用対象は研究開発目的以外のソフトウェア開発のすべてとすること、④仕損の区分は、非原価項目の費用処理する異常仕損と、仕損度合いが甚大で特別損失とする特別仕損とすること、⑤仕損の兆候の識別は計画比の４要件で行うこと、⑥仕損の認識はより精緻に行う必要性から、個々のタスク並びにモジュールを各々生産性と品質の指標で行うこと、⑦認識規準は「過去平均実績緩衝規準」とし、認識と測定を連続的に行う１段階アプローチとすること、である。

なお、兆候の識別や認識並びに測定の規準には、具体的な数値を示したが、あくまで例示で、原則主義規定であると断った上でのことだが、これらに関しては議論の余地があろう。また、会計処理案の適用を２段階構想とし、第１段階ではプロジェクトの失敗に限定し、第２段階で失敗か否かに関わらず、すべてのソフトウェア並びにソフトウェア開発に適用するという構想に関しても、実務の現状への配慮を理由としているが、その適否には議論の余地があろう。更に、異常仕損と特別仕損に区分すること、両者の境に挙げた数値により区分することの妥当性には議論の余地があるであろう。これらに関しては、更なる検討を深めたいと考えている。

また、本稿では規模と工数並びにコストの関係を正比例的と見做しているが、近似的には間違いではないし、実務における大方の扱いでもあるが、精密には非線形的と捉えるべきである。そうした認識に則った開発見積り手法もあるが、余り普及していない。過去平均実績との対比も、一律の扱いをしているが、精密にはソフトウェアの規模や難易度、技術者のスキルなどを考慮すべきであろう。これらに関しては、仕損の測定が複雑になること、いまだソフトウェア工学の分野でも懸案事項となっていることを理由に見送らざるを得なかった。更に、本稿は会計処理案の提示を目的としたため、情報開示をどのように行うか、具体的な提示を行うことはできなかった。これらに関しては考察を深めることと併せて、今後の課題である。

【引用文献】
・秋葉賢一(2011)『エッセンシャルＩＦＲＳ』中央経済社
・太田昭和監査法人、ビジネス・ブレイン太田昭和編(1992)『ソフトウェア開発の原価管理　改訂版』中央経済社
・企業会計審議会(1962)「原価計算基準」企業会計審議会
・金融庁(2012)「金融検査マニュアル」
・金融庁(2012)「主要行等向けの総合的な監督指針」
・誉田直美(2010)『ソフトウェア品質会計　ＮＥＣの高品質ソフトウェア開発を支える品質保証技術』日科技連
・櫻井通晴(1992)『ソフトウェア原価計算　増訂版』白桃書房
・日経コンピュータ(2008)「第２回システム開発プロジェクトの実態調査(2008年実施)」、『日経コンピュータ』2008年12月01日号、日経ＢＰ社
・日経コンピュータ(2012)「みずほの次期システムはマルチベンダー発注先決定、預金系を除きオープン化」、『日経コンピュータ』2012年11月22日号、日経ＢＰ社
・日経コンピュータ編(2011)『システム障害はなぜ二度起きたか　みずほ、12年の教訓』日経BP社

・平松一夫(2012)『ＩＦＲＳ国際会計基準の基礎　第２版』中央経済社
・森秀明(2003)『ＩＴ不良資産』ダイヤモンド社
・JUAS 編(2011)『2010 年度版　企業ＩＴ動向調査 2011』JUAS
・Cusumano, Michael A., Selby Richard W.(1995), *Microsoft Secrets*, The Free Press, A Division of Simon & Schuster Inc., New York City.(マイケル・Ａ・クスマノ、リチャード・Ｗ・セルビー(山岡洋一訳)(1996 年初版)『マイクロソフト・シークレット〈上〉』日本経済新聞社)
・ＩＡＳＢ(International Accounting Standards Board), *"The Conceptual Framework for Financial Reporting"*, 2010.
・ＩＡＳＣ(International Accounting Standards Committee), *International Accounting Standards No. 36 "Impairment of Assets"*, 1998.
・ＩＡＳＣ, *International Accounting Standards No. 38 "Intangible Assets"*, 1998.
・Jones, Capers(1995), *Patterns of Software Systems Failure and Success*, International Thomson Computer Press, Boston.(ケーパーズ・ジョーンズ(伊土誠一・富野壽監訳)(1999)『ソフトウェアの成功と失敗』構造計画研究所、共立出版)
・Jones, Capers(1993), *Assessment and Control of Software Risks*, Prentice Hall, New Jersey.(ケーパーズ・ジョーンズ(島崎恭一・富野寿監訳)(1995)『ソフトウェア病理学―システム開発・保守の手引』構造計画研究所 、共立出版)
・Jones, Capers(2007), *Estimating Software Costs: Bringing Realism to Estimating*, McGraw-Hill 2nd ed., New York.(ケーパーズ・ジョーンズ(富野壽、岩尾俊二監訳)(2009)『ソフトウェア見積りのすべて第 2 版』構造計画研究所発行、共立出版発売)
・Jones, Capers(2008), *Applied Software Measurement: Assuring Productivity and Quality*, McGraw-Hill 3rd ed., New York.(ケーパーズ・ジョーンズ(富野壽・小坂恭一監訳)(2010)『ソフトウェア開発の定量化手法　生産性と品質の向上をめざして第 3 版』構造計画研究所発行、共立出版発売)
・Tidd, Joe, Bessant, John, Pavitt, Keith(1997), *Managing Innovation: Integrating Technological, Market and Organizational Change*, John Wiley & Sons, Hoboken.(ジョー・ティッド、ジョン・ベサント、キース・パビット(後藤晃、鈴木潤訳)(2004)『イノベーションの経営学――市場・組織の統合的マネジメント』NTT 出版)
・Yourdon, Edward(2004), *Death March 2nd Edition*, Pearson Education, New Jersey.(エドワード・ヨードン(松原友夫・山浦恒央訳)(2006)『デスマーチ　第２版　ソフトウェア開発プロジェクトはなぜ混乱するのか』日経ＢＰ社)