IT分野で仕事をする人のために教育すべきこと

戸沢 義夫†

概要

システム開発の失敗が裁判になり高額な賠償金額の判決が出されたケースに対し，プロジェクトマネジメント義務違反と報じられた．システム開発の失敗はプロジェクトマネジメントがうまくいかないからと紋切り型に考えるのは少し短絡的だと思われる．システム開発の失敗，プロジェクトの失敗，誤謬など情報システム部門での失敗は何度も繰り返されており，そのような失敗をしないようにするための知見がベストプラクティスとしてまとめられている．PMBOK®はプロジェクトマネジメントのベストプラクティスとして有名だが，ITサービスマネジメントのベストプラクティスとしてITIL®がある．ITIL®にはアプリケーション管理機能が定義されており，そこにはシステム開発の失敗を避けることにつながるノウハウが書かれている．このようなベストプラクティスの知見はIT分野で仕事をする人であれば身に付けておくべきものであり，大学で教育可能なものである．J07-ISとは違った視点で，仕事で失敗しないための知識を大学で教える方法を議論する．

What to teach people who are to work in IT areas

Yoshio Tozawa †

Abstract

Several best practices are published as body of knowledge. PMBOK® is well accepted as a best practice of project management. ITIL® is also well known as a best practice of IT service management. Best practices can be used as a reference to avoid failures which many pioneers have experienced. A case of system development failure which became a dispute in a court is used to analyze the nature of causes. Those who are to work in IT areas had better learn knowledge which help to avoid failure. It is discussed that best practices should be taught in universities as well as the Japanese business practices in IT areas.

1. はじめに

システム開発の失敗による損害賠償（約74億円）の判決がニュースになり[1]，これについて日経コンピュータ誌はまとまった記事を掲載した[2]．その中に記者による次のような記述がある「プロジェクトマネジメント義務違反が認定されたとみられる」．

システム開発が失敗するのはプロジェクトマネジメントが悪いからだ，という思い込み（刷り込み）はかなり前から日本に浸透しており，それが日本の大学でのIT教育を変える原動力のひとつとなっていることは評価したい．プロジェクトマネジメント教育をカリキュラムに取り入れPMBOK®[3]を教える大学は増えてきている．これは好ましいことである．しかし，プロジェクトマネジメントをきちんと行っていたら，首記ケースのシステム開発の失敗を防げたかどうかについては疑問が残る．

IT分野では，いろいろな知見がベストプラクティスや標準としてまとめられている．PMBOK®はそれらの中のひとつである．他には，ベストプラクティスとして業界標準の地位を確立しつつあるものにITIL®[4]がある．ITIL®はITサービスマネジメントを志向したもので，プロジェクトを対象にしたPMBOK®とは全く違った視点でITを捉えている．システム開発（アプリケーション開発）についてもITIL®視点でのベストプラクティスが提示されており，その視点からシステム開発の失敗の原因を考察することが可能である．

IT産業で仕事をする人は，システム開発の失敗をしないようにすべきであり，そのために必要な知識やスキルは大学で教えておくべ

†産業技術大学院大学
Advanced Institute of Industrial Technology

きものと考える．PMBOK®もITIL®もマネジメントに焦点が当てられており，テクノロジーの話ではない．日本では，マネジメントというと，課長や部長の仕事だと思う人が多いが，そうではない．マネジメントはポジション（地位）で決まるものではなく，役割（Role）で決まるものである．

マネジメントをプロセスによって実施する（属人性を排除する）という考え方が大事であり PMBOK®にもITIL®にも共通している．PMBOK®はマネジメントの対象がプロジェクトであり，ITIL®はマネジメントの対象がITサービスである点が異なっている．

IT産業で仕事をする人はマネジメントを知っているべきだし，それは大学で教えるべきものと考える．システム開発に失敗した首記ケースでは，広い意味でマネジメントができていなかったと言える，プロジェクトマネジメントが悪かったという意見にはにわかには賛同しがたい．何をマネジメントすべきだったかという視点がぬけているからである．

2. 日本の商習慣とシステム開発

システム開発の失敗は極めて大きな損失をもたらす．首記ケースの損害賠償額は約74億円だが，これはキャッシュの部分だけで，新システムによる新業務から期待されたビジネス上のメリットや業務効率化などは含まれていない．全体の損失額はキャッシュ部分よりはるかに大きいはずである．キャッシュ部分の74億円自体が相当に大きな金額である．ひとりあたり年間500万円人件費がかかる社員がいたとすると1480人分である．

システム開発の失敗は100億円規模の損失をもたらし，しかもそれが頻発していた現実がある．経団連が「ソフトウェアの開発に携わる情報サービス産業」の深刻な問題として「高度ICT人材の不足」を挙げた[5]背景になっている．

図1は経団連が[5]の中で「企業ニーズとわが国大学における情報工学教育のギャップ」を表したものであるが，図をよく見ると，企業ニーズとは『ITサービス企業』を指していることがわかる．これは，明示的には述べられていないが，次の2つの意味で重要である．
①IT産業が製造業からサービスビジネスにシフトした
②ITサービス企業（システム開発を受注するベンダー）が（プロジェクトを失敗することが多く）困っている．

2.1 S銀行とI社のシステム開発失敗例

首記に挙げた裁判になったケースを紹介する．

地方銀行のS銀行と外資ベンダーのI社は，新勘定系システム開発について合意し2004年9月に要件定義を開始した．その後2005年9月に最終合意書が交わされ，開発にかかる総費用と稼働予定時期が明記された．しかし，要件定義は何度も失敗し稼働予定時期を遅らせることにしたが，結局開発できなかった．

このケースではパッケージ（Corebank）採用が前提になっていた．

システム開発に失敗したことはS銀行もI社も共通認識している．裁判になったのは，その責任がどちらにあるかである．当初判決内容や裁判の経緯は閲覧制限されており開示されていないので詳細はわからなかった．日経コンピュータ記者は「I社のプロジェクトマネジメントに不備があったのが失敗の原因」と推定している[2]．

その後判決書の閲覧制限が無くなり判決文を元に「東京地裁は『Corebank の機能や充足度、その適切な開発方法等についてあらかじめ十分に検証又は検討したものとはいえない』と判断、ITベンダーの責務であるPM義務に違反したと認定した」と書いている[6]．

2.2 プロジェクトマネジメント義務

「プロジェクトマネジメント義務」という考え方は，平成16年3月10日東京地裁の判決で明確化された．システム開発はユーザー企業とITベンダーの共同プロジェクトであり，請負契約であっても相互に一定の義務を負う．ITベンダーの義務は，プロジェクトマネジメント義務であり，ユーザー企業の義務は，システム開発のために必要な協力行う協力義務である．

S銀行とI社の判決文では，プロジェクトマネジメント義務が次のように定義された．
「システム開発業者として、自らが有する高度の専門的知識と経験に基づき、納入期限ま

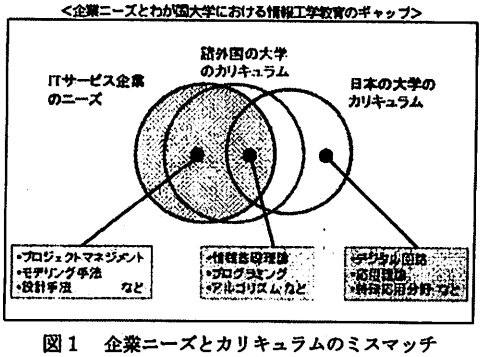

図1　企業ニーズとカリキュラムのミスマッチ
出典：日本経団連「産学官連携による高度な情報通信人材の育成強化に向けて」(2005)

でにシステムを完成させるようにユーザに提示し、ユーザとの間で合意された開発手順や開発手法、作業工程等に従って開発作業を進めるとともに、常に進捗状況を管理し、開発作業を阻害する要因の発見に努め、これに適切に対処すべき義務や、ユーザのシステム開発への関わりについても適切に管理するなどの行為をすべき義務」

判決文は次のように結論付けた.

(ア) Ｃｏｒｅｂａｎｋの機能や開発手法についての検証、検討等
(イ) ＦＩＳ社の関与による開発体制の整備等
(ウ) 被告の原告に対する説明
(エ) サービスインの時期の延期
(オ) 本件最終合意書の作成以降における被告の対応
(カ) ＴＣＢの提案
(キ) 以上によれば、被告には、本件システム開発のベンダとして適切にシステム開発を管理することなどを内容とするプロジェクト・マネジメント義務の違反があるものというべきである。

I 社にはプロジェクトマネジメント義務違反という不法行為があった.

2.3 日本のシステム開発ビジネス構造

かなり昔の話であるが，コンピュータが初めて企業に導入される場合は，コンピュータメーカーはその企業に対して「情報システム部門」を作るように働きかけた，新しい組織の誕生である．情報システム部門の主な役割は，業務システムの開発と運用であり，情報システム部の社員がプログラミングを行っていた．コンピュータ・メーカーの SE は情報システム部を支援するのが目的で，プログラミングをすることはあっても，委任であり請負うことはなかった．当初は，システム開発は情報システム部の責任で行うのが当たり前だった.

その後，システム開発要求が増えてくると，プログラミングを自社社員だけでは賄えなくなり，社外にプログラミングを委託せざるをえなくなった．プログラミングを他社に委託する場合，準委任と請負の２つの契約形態がある．準委任の場合は作業を委託するのに対し，請負の場合は成果物（結果）への責任（瑕疵担保責任）をとることが求められる．業務を発注する立場からは，準委任契約よりは請負契約の方が“楽”という意味があり，請負契約でシステム開発を受注する会社が増えてきた.

請負契約の場合，元請け会社はその業務の一部を下請けに出すことができる．発注者は請負業者の仕事の結果についてはいろいろ言えるが，仕事のやり方には口出しはできないので，子請け，孫請けなどの構造が発生する．子請け，孫請けが好ましいビジネス形態ではないと認識されているにもかかわらず，実態がそうなっているのは，システム開発業務委託が請負契約で行われていることと関係している.

請負契約には瑕疵担保責任があるが，旧来の契約では無限責任であることが多かった．国との請負契約が無限責任だったことに起因していると思われるが，運用上，無限責任を問われることがまずないので，国内企業はこの形態の契約を受け入れていた．しかし，契約を厳密に解釈する外資企業は無限責任を受け入れることができないため，責任範囲を明確にするように変わってきている.

請負契約の場合，納品物がどんなものであるかを発注時点できちんと示さなくではならない．建築物や土木工事では図面が示されるので，納品物が図面通りに作られているかどうかを検収時にチェックできる.

しかし，システム開発の場合，システム機能を発注者が明確に示すことはほとんどない．システム仕様があいまいなまま請負契約が締結され，納期，金額，システムの目的（システム名）だけが決められる．仕様があいまいなまま請負契約をし，仕様をきちんと決められずに失敗するシステム開発はかなり多い.

2.4 ウォーターフォール型開発のビジネス

仕様があいまいなまま請負契約するのは，システム開発を受注する企業にとって非常にリスクが多く，必要以上に受注額を高くしたり，契約金額内で収めるために開発機能を削ったりすることがしばしば行われていた．契約が１本になっているのが問題で，システム開発規模が定まらない時期に全体の金額を決めてしまうことから発生する問題である．これを回避するために，システム開発がウォーターフォール型の開発を行っていることから，ウォーターフォール型開発のステージごとに，個別の契約に分ける方法が考え出された．これを IT ベンダー側から提案した結果発注企業で受け入れられ採用されるようになった.

発注者側からは請負契約１本で済ませたいと思っているのに対し，受注側企業はステージごとの個別契約にしたいと思っているのでここに対立が生じる．首記ケースのＳ銀行とＩ社の場合は，金額と時期を明記した「最終合意書」を取り交わしているが，それをベースに個別契約を別途結んでいる．最終合意書が請負契約であるかどうかは解釈の問題になり，裁判で争われる争点のひとつになっている.

発注者側がステージごとの個別契約を嫌がるのは，ひとつの契約が終わり次のステージの契約を締結する際に，他ベンダーへの切り

替えが実質的に無理なのでベンダーの言い値の金額で契約せざるをえなくなるからである．

個別契約でも成果物（例えば要件定義書）は納品されるが，それが意味を持つのはそのベンダーが次のステージのインプットとして使用する場合だけであり，他ベンダーにとってはほとんど価値がない．他ベンダーへ切り替えるとそのベンダーのやり方に従った作業を2度手間でやり直すことになってしまう．S銀行とI社の裁判でも個別契約の成果物の（客観的）価値は認められなかった．

2.5 人月に基づく価格設定

発注者がシステム開発を業務委託する場合ベンダーに費用の内訳を出すように要求することが多い．この商慣習は製造業ではよく行われていることで，部品メーカーの部品の値段を，それを製造するコストを見て発注者が妥当かどうか判断する．費用の内訳を要求されたベンダーは，開発総工数を人月で表し，人月単価を掛けて内訳として提出するようになった．製造業の商習慣がシステム開発に入り込んでしまった結果，極めてまずいことを引き起こすようになった．これを是正するのは簡単ではなく現在もいろいろな努力が行われている．

ITスキル標準を設定し，スキルレベル（1～7）によりスキルの違いを強調するようになった背景に，総工数と単一の人月単価で価格が決まることへのベンダー側からの反抗があった．スキルの高い人が仕事をしたら当然コストは高くなることを発注者に納得してもらう手段として利用したかったのである．

そもそもシステムの仕様があいまいなのに，総工数を見積もることは無理なのだが，発注企業がベンダーから提示された総工数の妥当性をチェックすることはほとんどない．また，見積もり時に提示された総工数と，実際にかかった工数を比較することもない．従って，ベンダーが提示してくる見積金額が妥当かどうかを判断するのに，総工数を聞いたとしてもほとんど情報は増えてない．ベンダーが受注する意欲があれば，見積金額を下げるために，総工数を減らすか人月単価を下げるかしてつじつまを合わせることになる．本当に必要な総工数はあいまいだから見積もり時と実際の乖離が出るとシステム開発の失敗につながる．また，人月単価を下げると，より安く仕事をしてくれる下請けを探すことになるが，スキルや品質の心配がでてくる．また，プログラミングは人が行う作業なので，値切るとプログラマーのモラルが下がり生産性が期待できない．プログラマーは7K職場と言われるようになった原因のひとつが，人月単価にある．

発注者にとってほとんど情報が増えないにもかかわらず（製造業で行われている発注と同じ考え方で）システム開発委託の総工数と人月単価を求める商習慣は，システム開発を失敗に導く遠因になっている．

2.6 サービスビジネスへのシフトの影響

昔は，情報システム部門がその企業のためのシステム開発に責任を持っていた．コンピュータメーカーはSEによる支援を行ったり，準委任契約によるプログラミングの委託を受けたりしていた．当時でも，システム開発の総工数を見積もるのは困難であり，それはブルックスJrが“The Mythical Man-Month”で指摘している[7]．

昔はコンピュータの値段が極めて高かったため，ユーザー企業もコンピュータメーカーもSEサービス費用はコンピュータ経費の一部ととらえ，その割合はあまり大きくなかった．コンピュータメーカーにとってSEによるサービス部分を独立したビジネスと考え，ハードウェアやソフトウェアとは別に管理するようになったのはそれ程古いことではない．

コンピュータの技術進歩はハードウェアの値段を劇的に下げ，IT産業の主流は製造業からサービスビジネスへとシフトした．サービスビジネスとしての管理が必要になっているのだが，ユーザー企業もコンピュータメーカーもシステム開発を製造業的に扱い管理しようとする傾向があった．システム開発は，どちらかというと製造業よりは建築工事，土木工事に近いと思われるが，そのように扱い，管理しているケースはあまり見られない．その結果，サービスビジネスで重要視しなければならない部分が軽視されてしまっている．

2.7 人を投入すれば解決する？

システム開発がうまくいっていないことがプロジェクトの途中で判明した場合，受注者側が発注者側にお詫びする際に，とにかく人をたくさん投入して何とかします，と言うことが多い．うまくいかない原因を特定してそれに対処するというアプローチではなく，人を投入すれば何とかなるという幼稚な発想である．烏合の衆でシステム開発をやってうまくいかない場合，スキルのない人を投入しても烏合の衆が大きくなるだけで，結局はうまくいかない．経団連が高度ICT人材不足を指摘したのも，このような事例がいくつも見られたからである．

プロジェクトの遅れの原因がリソース不足であれば人を投入すれば解決できる．しかし，スキル不足が原因であれば単に人を投入するだけではダメである．どんなスキルが何のために必要なのか，どのスキルが不足していた

ために何ができていないのかを見極めて，プロジェクトを立て直さなければならない．実は，このような経験からシステム開発で失敗しないようにするための知見をまとめる努力が行われ，ベストプラクティスが作成されているのである．

3. システム開発失敗事例集

日経コンピュータ誌ではシリーズとして「動かないコンンピュータ」記事をほぼ毎号載せている．その中でシステム開発失敗事例もいろいろ報告されている[8]．また，経産省の「情報システムの信頼性向上のための取引慣行・契約に関する研究会」による「情報システム・ソフトウェア取引トラブル事例集」[9]も公開されている．それを防ぐための「情報システム・モデル取引・契約書」[10]が提示されている．

発注者が受注者に業務委託する内容が契約書にきちんと盛り込まれていなければならないのは当然である．しかし，契約したからといって，契約通りに受注者が仕事してくれるとは限らない．やれる Capability を持たないのに「やれます」と言って受注する業者もある．請負契約の場合，発注者が受注者の仕事のしかたに口出しできないのが原則だが，契約通りに納品物ができるかどうかを途中段階で確認することは重要である．

公共工事の場合，請負契約で受注業者に発注するが，法律や条例により発注者（例えば東京都）は監督員による工事監督業務を行うことが定められている．受注業者の現場代理人は監督員の同意を得て工事を行う．監督員は技術専門職である．請負契約であっても，受注者の仕事のしかたをチェックする仕組みになっている．残念ながら，システム開発ではこのような制度は一般化しておらず，個別に対応しているのが現実である．

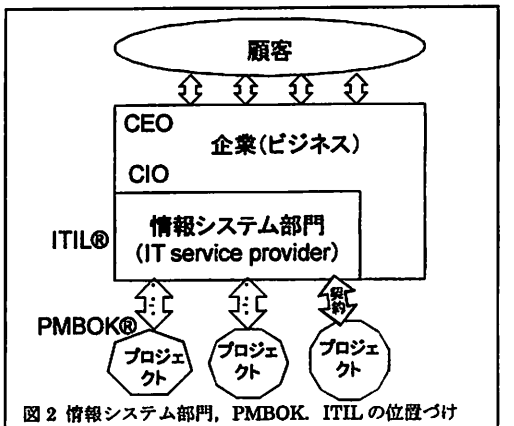

図 2 情報システム部門，PMBOK，ITIL の位置づけ

4. ベストプラクティス

IT の分野では，過去にいろいろな失敗を経験し苦い思いをしたことから，そのような失敗を繰り返さないようにするために知見をまとめる努力が行われてきた．ベストプラクティスは知見を集大成する方法のひとつで，有名なものに PMBOK®や ITIL®がある．どちらもそれらの知見を保有している人に対して国際資格が設定されている．

PMBOK®や ITIL®ではプロセスが定義されており，ベストプラクティスでは，それらのプロセスがどうなっているかを示している．ただし，ベストプラクティスは，そのように仕事をしなさいというものではない．もし何かうまくいかないことがあったらベストプラクティスと比較することにより，やるべきことがやられていなかったり，ぬけがあった部分を認識するために利用する．ベストプラクティスを知っていると，やるべきことのぬけを防ぐことができるので失敗を防ぐことが可能である．

本論文では，企業で使用する業務システムの開発に焦点を当てているので，その観点から PMBOK®と ITIL®の位置づけを図 2 に示す．図 2 では情報システム部門は企業の 1 部門であるように描かれているが，企業に IT サービスを提供する組織（IT service provider）であれば別会社でもよい．重要なポイントは，企業はその顧客に対してビジネスを行っており，IT サービスはそのビジネスを行う企業のために提供している点である．

システムに求められる機能は，企業がどのように顧客に対してビジネスを行っているか，ビジネスをどのように管理しているかから決まるものである．システムを利用する企業のビジネスを知らずに，言われたことだけに対応するシステム開発は，システムがビジネスに合わなくなる可能性が高く，非常に危険である．

システム開発はプロジェクトである．ほとんどの場合，システム開発は情報システム部門がベンダーへ委託し，請負契約により発注される．システム開発をプロジェクトと捉えた場合，PMBOK®の知見が活かされる．システム開発を（ベンダーへ委託したとしても）情報システム部門の仕事と捉えると ITIL®の知見が活かされる．

裁判所の見解では，S 銀行システム開発失敗の原因は I 社のプロジェクトマネジメント義務違反とされたが， PMBOK®を正しく理解していればシステム開発失敗は防げたかというとそうは思えない．ITIL®を含めてどこが悪かったかを見ていくことにする．

4.1 PMBOK®

PMBOK®はプロジェクトマネジメントのベストプラクティスである．PMBOK® 第 4 版では．5 個の基本的なプロセスグループと、9 個のナレッジエリアが定義され，その中に 42 のプロセスが規定されている．プロジェクトマネジメントがうまくいかないのは，42 のプロセスのうちのどこがうまくいかないかで説明できる．

S 銀行と I 社のケースは「5.1 要求事項の収集プロセス」が関係している．このプロセスは「プロジェクト目標を達成するためにステークホルダーのニーズを定義し、文書化するプロセス」と定義されている．

S 銀行と I 社のケースでは，要件定義を 3 回繰り返したと報告されている[11]ので要求事項の収集でつまづいたことは間違いない．しかし，要求事項の収集技法はいくつか書かれているが「要件定義」には触れていない．従って，PMBOK®を勉強していたとしてもこのシステム開発失敗を防げたとは言えない．

4.2 ITIL®

ITIL®は IT サービスマネジメントのベストプラクティスである．ITIL® version 3 は，(1)サービスストラテジ，(2)サービスデザイン，(3)サービストランジション，(4)サービスオペレーション，(5)継続的サービス改善の 5 つのステージに分かれており，26 のプロセスと 4 つの機能が定義されている．サービスオペレーションの中の機能のひとつに「アプリケーション管理」が定義されている．

4.2.1 ITIL®アプリケーション管理

ITIL®アプリケーション管理では，図 3 に示すアプリケーションライフサイクルを定義し，設計・構築だけでなく運用まで含めて捉えるようにしている．「設計」の前の「要件」には次の 6 種類があることを明記している．

- 機能要件
- 管理性要件
- ユーザビリティ要件
- アーキテクチャ要件
- インターフェース要件
- サービスレベル要件

これらの要件を，設計を始める前に，きちんと定義することが求められている．機能要件は，システムを使って業務を行うためにシステムが提供しなければならない機能を指し，通常「要件定義」で明確化・文書化していくものである．機能要件はユーザー企業のビジネスから決まるもので，本来なら，システム開発を委託する際に与件として提示すべきものである．

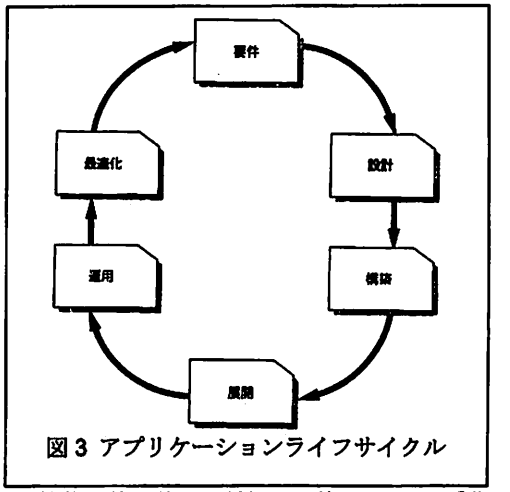

図 3 アプリケーションライフサイクル

機能要件以外の 5 種類の要件はまとめて「非機能要件」と呼ぶことがあるが，これらは，システムを展開したり運用したりする際に重要になる．これらの要件は受注するベンダーのノウハウとしてシステム開発時に活かすべきものである．

大企業の情報システム部門でシステム開発を何度も経験している場合は，6 種類の要件すべてを情報システム部門の責任で決定するケースもある，しかし，多くの場合はベンダーのノウハウを活用するのが望ましい．受注側のベンダーも与件にリアクティブに対応するだけでなく，プロアクティブにユーザー企業のメリットになることを考えてプロとして真摯に脾機能要件に対応すべきである．

旧システムから新システムへ移行する際，新システムの機能要件として「旧システムが持っている機能すべてを新システムで使えるようにする」と表現する場合がある．これは本来やるべきではないが，これがまかり通ってしまう現実がある．旧システムが持っている機能すべてについて，きちんとドキュメントがあればあまり困らない．しかし，ドキュメントはあっても動いているシステムを正しく表していない場合だと，動いているシステムを分析して旧システムの機能すべてを洗い出す作業（リバースエンジニアリング）を行わなければならなくなる．多大な作業が必要になるだけでなく，新システムでは必要ない旧システム機能を実装する無駄が発生する．機能要件は新しい業務のやり方にあった機能のことであって，取りあえず旧システムが持っている機能すべてとしておけば大丈夫だろうと発想するのはどこか間違っている．システム開発を委託する側（発注者）がいい加減だとアプリケーションライフサイクルの要件の部分でつまずくので失敗する．

4.2.2 アプリケーション管理での役割

ITIL®アプリケーション管理では，次の 2 つの役割（roles）が定められている

- Applications manager/team leader
- Applications analyst/architect

アプリケーション管理者（チームリーダー）はリーダーシップ，コントロール，意志決定の全般にわたって責任を持つ人である．アプリケーション分析者（アーキテクト）は，ユーザー，スポンサー，各種ステークホルダーのニーズを特定し，技術面を含めたシステムへの要件を定義する人である．管理者と分析者の役割は違うということを明確に述べている．誰がやるべきかには触れていないが，責任の所在を明確に示している．6 種類の要件を洗い出すのは分析者の仕事であるが，意志決定の責任は管理者にある．

ITIL®では，アプリケーションライフサイクルの説明の前に，アプリケーション管理の方針（principle）について書かれている．方針で最初に取り上げられているのが「Build or buy?」である．パッケージを使うか独自システムを開発するかを方針として先ず決定する必要があり，その後でアプリケーション管理がスタートすると位置づけられている．

4.2.3 Build or buy?

Build or buy?の意志決定の際にアプリケーション管理の立場から考慮すべきことが 9 項目挙げられている．その中のひとつが必要な機能とパッケージが提供する機能の差（フィットギャップ分析）およびカスタマイズの必要量である．少なくともこれらの事をきちんと調査した上で Build or buy?の意志決定すれば大きな間違いを防ぐことができる．このような知見が書かれているのがベストプラクティスを参照するメリットである．

4.2.4 ITIL®によるケース評価

S銀行とI社のケースをITIL®アプリケーション管理に当てはめてみる．要件をきちんと決められなかった部分については，アプリケーション管理者やアプリケーション分析者が何をしていたかが問題になる．このケースでは，そもそもアプリケーション管理者は誰か，アプリケーション分析者は誰かがはっきりしていただろうか．Role（役割）を誰が受け持つかは，業務委託や契約とは別次元の話である．Role についてS銀行とI社できちんと合意されていなかったと思われる．

次にパッケージの採用が適切であったかどうかである．Build or buy?の意志決定は最も重要なもののひとつで，どのような経緯でこの意志決定が行われたか，何か調査され，どんな分析が行われて，誰が何の責任を持っていたかがはっきりしているべきである．S 銀行とI社の間では，I社がパッケージを提案しそれをS銀行が受け容れたとされている．その後，パッケージに関係したいろいろな問題点が指摘されていることから，Build or buy?の意志決定はもっと慎重に行うべきだったのではないだろうか．ITIL®にはこの際に考慮すべき項目が挙げられているので今後このような失敗を犯さないようにベストプラクティスを活用することが望まれる．

5. 大学で教えるべきこと

5.1 J07-IS

情報処理学会では情報専門学科におけるカリキュラム標準を J07[12]としてとりまとめている．システム開発は J07-IS（情報システム領域）に含まれる．本論文に関係したラーニング・ユニットとしては次のものがある．

LU#0441 情報システム開発ビジネス
LU#0442 プロジェクト管理の基礎
LU#0443 見積もりとスケジューリング
LU#0444 プロジェクト計画書
LU#0445 プロジェクトファシリテーション
LU#0446 プロジェクトにおけるリスク管理
LU#0447 プロジェクトにおけるリスク対応

これらの中で PMBOK®は若干取り上げられているが，残念ながら，ITIL®は全く参照されていない．PMBOK®も ITIL®もベストプラクティスとして書籍としてまとめられている（日本語訳もある）ので，参考図書として挙げてあっても良いと思う．システムは運用されて初めてビジネス価値を生むようになるので，運用が視野に入っているのが望まれる．ITIL®のアプリケーション管理は「運用」が含まれている点が実際的である．

5.2 日本の IT 分野ビジネスの特徴

日本でシステム開発がうまくいかない理由のひとつに日本の商習慣がある．システム開発やシステム開発で必要になる役割と，業務委託のやり方，見積，契約，責任の所在との間でミスマッチを起こしている．日本人の仕事に対する考え方と，アメリカから導入された評価指標の考え方にもミスマッチがある．IT 分野で上手に仕事をしていくには，現実がどうなっているかを正しく認識することと，ミスマッチがあり是正が必要なポイントを知っていることである．

日本の商習慣が米国（グローバル）の商習慣と違っていることのひとつに IT 調達時の考え方がある．米国 Federal CIO Council は，CIO と CIO スタッフに求められるコア・コンピテンシー[13]をまとめている．第 8 章

Acquisition（調達）には調達の4つのステージが次のように規定されている．

1) Defining the business objective
2) Requirements definition and approval
3) Sourcing
4) Post-Award management

要件定義は発注者側（CIO）の責任でありその後にどのベンダーに発注するかを決めるとなっている．このようなCIOのコア・コンピテンシーがまとめられるきっかけは，米国政府でいろいろなプロジェクトでのIT投資の失敗にある．

IT分野のグローバル化に伴い，日本でのIT職種が7Kと言われるのは不幸である．これは社会のひずみであるから変えていくべきであるが，大学で日本の商習慣を正しく教え，是正ポイントをきちんと指摘しておけば世の中を変える手助けができると考える．

5.3 ベストプラクティス

考え方のフレームワークはいろいろ提案されているが，大学で教えるのであればグローバルに認知されたベストプラクティスが望ましい．また定期的に内容を見直し改訂していることも必要である．PMBOK®やITIL®はこれらの条件を満たしている．同じフレームワークに従っていれば，用語や概念を説明しなくても初めて会った人とでもコミュニケーションできるからである．

6. まとめ

大学で学生に何をどう教えるかは永遠のテーマである．学生が大学で学ぶことにより何ができるようになるか，という観点で教育内容を決めるのが良いと思われる．IT分野で仕事をする人は，ITをビジネス（社会）に役立てることに貢献できることが大事だと思われる．ITをビジネスに役立てようと努力したにもかかわらず，過去に多くの失敗を経験しているのが現実である．そのような経験から多くの知見が得られている．それらはいろいろな形態で発表されたり共有されたりしているが，そのひとつにベストプラクティスがある．

PMBOK®を教えている大学はいくつかあるが，ITIL®はほとんどない．大学での教育内容を先ずこの辺りから検討してはどうだろうか．IT分野がどんどんグローバル化していることを考えると，海外でまとめられたベストプラクティスは，日本の商習慣との違いを意識する上でも重要だと思われる．また過去の失敗に根付いた知見の知識があれば，システム開発に失敗して裁判沙汰になるようなことは防げたのではないかと思うからである．

参考文献

[1] IT Pro ニュース［速報］ http://itpro.nikkeibp.co.jp/article/NEWS/20120329/388219/

[2] 日経コンピュータ 2012年4月12日号 記事 http://itpro.nikkeibp.co.jp/article/COLUMN/20120409/390218/

[3] Project Management Institute,Inc.：プロジェクトマネジメント知識体系ガイド 第4版

[4] ITIL® 2011 Edition：Service Strategy
ITIL® 2011 Edition：Service Design
ITIL® 2011 Edition：Service Transition
ITIL® 2011 Edition：Service Operation
ITIL® 2011 Edition：Continual Service Improvement, TSO 発行

[5] 日本経団連：産学官連携による高度な情報通信人材の育成強化に向けて(2005) http://itpro.nikkeibp.co.jp/article/COLUMN/20120521/397641/

[6] 日経コンピュータ 2012年5月24日号 記事 http://itpro.nikkeibp.co.jp/article/COLUMN/20120521/397641/

[7] フレデリック・P・ブルックス Jr.：人月の神話 新組新装版,ピアソン桐原,2010

[8] 日経コンピュータ編：動かないコンピュータ 情報システムに見る失敗の研究,日経BP社,2002

[9] 情報システムの信頼性向上のための取引慣行・契約に関する研究会：情報システム・ソフトウェア取引トラブル事例集 http://www.meti.go.jp/policy/it_policy/softseibi/trouble%20cases.pdf

[10] 情報システムの信頼性向上のための取引慣行・契約に関する研究会：情報システム・モデル取引・契約書 http://www.meti.go.jp/policy/it_policy/softseibi/model_tuiho/model_tuiho.pdf

[11] 日経コンピュータ 2008年5月1日号 記事 http://itpro.nikkeibp.co.jp/article/NEWS/20080425/300145/

[12] 情報処理学会：情報専門学科におけるカリキュラム標準 J07http://www.ipsj.or.jp/12kyoiku/J07/J0720090407.html

[13] Federal CIO Council：2008 Clinger-Cohen Core Competencies and Learning Objectives http://www.cio.gov/documents_details.cfm/uid/1F437681-2170-9AD7-F20824C0DB3AF26A/structure/IT%20Workforce/category/IT%20Workforce