みずほのトラブルから何を学ぶか

工学院大学国際基礎工学科／畑村創造工学研究所
畑村 洋太郎
hatamura@shippai.org

みずほ銀行のシステムトラブルが起きてから約 2 年になる．あれだけの大きなシステムトラブルをやったのだから「せっかくやった失敗だ，しゃぶり尽くさなければもったいない」と考えて，このトラブルから学ぶべきことを筆者の提唱する失敗学の立場から瞥見（べっけん）してみよう．

システムを作る人（設計者）が学んだことは，異なる起源を持ったシステムを後から統合するのは，とうてい無理だということである．次々に生じる要求を満たすべく行われた「付加設計」は決して最良なものとはなり得ず，無駄と見えない欠陥（不具合）を内包し，思わぬときにそれらが顕在化する．このようなことを避けることはどんなに注意深く点検しても不可能であり，はじめから要求機能と制約条件を明らかにし，それらを満たすべく「全体設計」をやる以外に方策はない．このような考え方は，すべての設計に当てはまることで，筆者が永く親しんできた機械設計では当然視されているし，情報システムの設計者にも，たとえば流用設計を行うときの問題点として広く認識されている（図 -1）．

システム設計者が学んだもう 1 つの大事なことは，システムの不具合は，全体を動かしてみなければ分からないということである．別な言い方をすれば，部分確認の積み重ねでは全体適正の確認はできない，ということである．でき上がった全体システムの一部分を取り出し，その周囲状況を仮に設定し，その部分の動的な挙動を確認しても，部分を繋ぎ合わせて全体を再構築したときの全体としての挙動は確認できない．なぜなら部分と部分の接合部の状態がはじめに仮定しなかった状況になることはいくらでもあり得ることだからである．

システムを動かす人（経営者）が学んだことは，いったんシステムにトラブルが生じると，それぞれの作業の本質的な意味を知り，全体における部分の果たすべき役割を十分に知覚している人間の存在こそが重要であるということだ．システムが動かなくなったときに最後に頼れるのは人力（じんりょく）である．手書きの伝票に記入し，電話で伝達する．紙面に作表し，それをファックスで伝送する．ひとりずつがパソコンのキーを叩き，相手に伝送する．筆者はその現場に立ち会ったわけではなく，推測するだけであるが，多分この推測は合っているだろう．そして手作業ができたからこそ 1 カ月余の緊急処置で曲りなりにも銀行の基本機能を回復することができたのである．今回は従来の手書き作業を経験した従業員が健在だったからこそ対処できたが，コンピュータに依存した仕事しかしたことのない人たちだけで銀行を動かすようになる 10 年後に今回と同じトラブルが起こったら対処不能になり，応急処置不能のまま新しいシステムができるまで

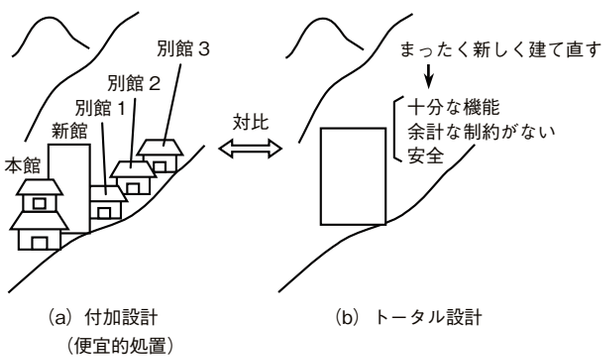

図 -1　付加設計とトータル設計の例－山の中の温泉宿－

2～3年待たなければならない，などという事態が起こり得る．

システムを使う人（利用者）が学んだことは，巨大システムは当てにならないことと，それでもそれに頼らざるを得ないことである．利用者から見ればシステムはうまく動いていて当たり前の“空気のようなもの”なのである．何の努力もせず，何の危険も想定せずとも，いつも便利さが取り囲んでくれていると思っていたものが突然動かなくなり，送金が届かなかったり，自動引き落としの料金が引き落とされずに督促がきたりしたのだから，起こった途端は何が何やら分からず，事態が明らかになるにつれてあきれ果て，怒り始めたのである．そして時間が経ち全容が分かってくると，身の回りが巨大システムに囲まれ，それの当てにならないことを考え，怖さを感じ始めている．利用者は情報システムだけでなく，今は空気のように思っている食料やエネルギー（電力を含む）にも同じことが起こり得ることに考えを広げなければならない．

最後にシステムを考える人（研究者）はいったい何を学んだのであろうか．自分には関係のないことだと何も学ばなかったのかもしれない．しかし，あのトラブルを真剣に考え，今までとは違った見方の必要性を感じた人もいるかもしれない．情報システムについてはまったくの門外漢である筆者の考えを披露し，システムを作る人へのヒントとしたい．

(1) システムの適正さの検証に「逆演算」の考えが使われているのだろうか（図-2）．ここでいう逆演算とは通常の設計の思考過程で行われる「これをやるとこうなるはず」という順演算のまったく逆で，まずくなる結果を仮定し「それになるには，これが要る」と逆に論理をたぐってゆくやり方である．失敗学において，この逆演算は最も重要視されており，昨今我が国で起こるさまざまな失敗は，この逆演算をせずにシステム構築し，しかも逆演算の必要性にすら気づかずにいることに根本原因があると筆者は考えている．なお，失敗分析では，失敗の樹木構造で示すことが広く行われており，「逆演算」に近い考え方をしている．

(2) システム設計に「全体設計」の考えはあるのか，また実用化されているのか．要求機能と制約条件を満たす解を，後から生じた要求を満たすべく変更し，それが積み重ねられてでき上がったのが現実のシステムであるとすれば，それはいつも付加設計の固まりということになる．しかしシステムをまったく新たに構築するには費用も時間もかかりすぎると考えると，躊躇が起こる．これをどう乗り越えればよいのか．なお，主に機械設計ではSuhの「設計公理」[1]や筆者らの「創造設計原理」[2]がこの課題を取り扱っている．

(3) システム構築の途中で求められる設計変更への対処理論はあるのか，また実用化されているのか．実際のシステム構築でははじめに定められた仕様がそのまま最後まで固定されることは稀で，作業途中でいろいろな変更が求められている．場合によっては作業途中ではじめから全部やり直さなければならないこともある．作業途中で加えられる変更の程度をあらかじめ仮定し，そのような変更があっても対処できるようにはじめに定める構造を構築することと，途中での変更のやり方を考える理論と方策が求められる．他分野の例では，さまざまなプラント設計でこのような考え方が採用されている．

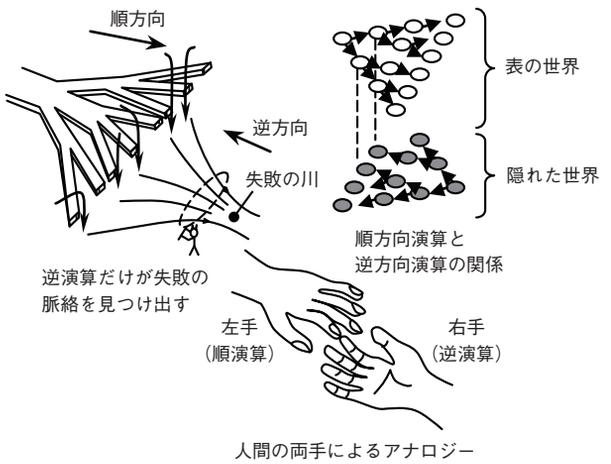

図-2　逆演算思考の必要性

約2年前に起こったみずほ銀行のシステムトラブルについて筆者の考えをいくつか紹介した．これをヒントにせっかく起こしたトラブルから将来のシステム構築の種をぜひ引き出していただきたい．

参考文献
1）Suh, N. P.:The Principles of Design, OXFORD UNIVERSITY PRESS（1990），畑村洋太郎（訳）：設計の原理，朝倉書店（1992）．
2）畑村洋太郎，小野耕三，中尾政之：機械創造学，丸善（2001）．

（平成15年2月8日受付）