第１３２号
平成１２年９月

E-mail :
shimz@mb.infoweb.ne.jp
LDG04167@nifty.ne.jp

SCだより

編集発行人
清水吉男
（株）システムクリエイツ
横浜市緑区中山町 869-9
電話 045-933-0379
FAX 045-931-9202

ソフトウェア開発の原則

「ソフトウェア開発 201の鉄則」から ５０ かい

もし、要求仕様書に誤りがあれば、見つけるのが後になればなるほどとんでもなく高くつく。

・もし要求仕様書の誤りが設計まで残っていたら、それを見つけて修正するのに５倍のコストがかかる。
・もしコーディングまで残っていたら、１０倍かかる。
・もしテスティングまで残っていたら、２０倍かかる。
・もし納入時点まで残っていたら、２００倍かかる。

要求段階の期間中に要求仕様の誤りを修正することは当然だが、この数値は、このことの正当性を確信させる証拠以上のものだ。
（２０１の鉄則：原理４１＜要求分析の原理＝今すぐ要求仕様書の誤りを直せ＞）

― 解　説 ―

最近、この種のドキュメントの作成を省こうという動きがあるようです。その最大の動機は、仕様の変更に際して、ソースは修正するしかないので問題ないのですが、元の要求仕様も修正することに対して、合理性が認められないというものです。はたして原理４１は、「過去の呪文」となってしまったのでしょうか。

確かに、ある種の分野に於いては、今までのような要求仕様書ではなく、ソースを兼ねた手段で表現することで対応できるかもしれません。変更するより、毎回新しく作った方が早いという部分もあるかもしれません。でも、最初から作り直すことのリスクが発生します。特に、すでに運用している状態であれば、リスクの表面化によっては、運用を中断することにもなります。

要求の誤りって？

ソフトウェアの誤りの中で、影響の大きいものは「要求仕様の誤り」と「設計の誤り」です。後者は、そこで求められていた仕様を実現するためのデータ構造の選択ミスや、データ構造の不適切な構成によって仕様を満たせない状態、あるいは、処理モジュールの構造が適切でないため、思わぬ時間が掛かってしまったとか、選択されたデータ構造を処理しきれない状態を言います。

一方、要求仕様の誤りというのは、そこで考慮されるべき処理や対応が抜けていたり、仕様同士が矛盾していることに気付かなかったり、仕様そのものが的をはずしていたりする状態です。また、この中には、システム全体としての振る舞いを記述した要求仕様と、その中の個々のコンポーネントや処理モジュールにおける「What」を記述した要求仕様があります。後者の要求仕様は「設計」の結果として表現されるもので、原理４１で指摘しているのは、前者のシステム全体の方の要求仕様です。

このようなシステム全体の振る舞いを記述する要求仕様が間違っているというような場合、当人がそのことに気付いていないことが問題を大きくするのです。分かりやすく云うと、仕様が漏れていることを発見するのは、設計者ではなく「要求者」なのです。だから、レビューが必要なのです。問題は、このレビューを「何で」行うかです。ＵＩなどの場合には、文書以外でも可能かも知れませんが、プロトタイピング・ツールの場合、実際に操作を通じて「そこ」に到達しないと見えません。また、どこまで細かく見せることができるかという点でも不足が生じます。そこで最近は、インクリメンタルな対応で「本物」を早く作ってしまう方法によって、早期に本物の細やかさを見せる取り組みも出てきましたが、誤りを発見できる人は、今までと変わりません。

要求仕様書が修正されない理由

文書化すると云っても、必ずしも「文書」にこだわらなくてもよい部分はあるでしょう。ここでは、それらを含めて「文書」ということで話をすすめますが、要求仕様を文書にする理由としては、事後に要求仕様の変更が生じた時の対応を円滑にすることにあります。

要求仕様の変更が適切に行われない状況で目にすることは、もともと「仕様」が表現されていないことです。そこにあるのは「要求」であって「仕様」ではないのです。ところが開発途中の変更は、多くの場合、「仕様」のレベルで出てきます。もちろん「要求」レベルで変更されることもありますが、後の方の工程になるほど、「仕様」のレベルで変更が出されます。この時、もともと、仕様を表現したものがない場合、「要求仕様書」の中で当該箇所や影響範囲などを検討することが出来なかったり、不十分になってしまいます。“変更”も、いままで考慮の必要がないと思われていた領域で、仕様が追加されることもあり、影響範囲などの検討を難しくします

このような時に、事前に仕様のレベルで表現されていれば、依頼された変更内容が、仕様間の矛盾を引き起こしているような場合でも、結構、気が付くものです。実際に、私の周りにおいても、これは実現しています。

このように、要求仕様書が修正されないのは、修正できる状態で書かれていないからです。そのため、要求仕様書の修正に意義を見出せないのでしょう。

また、顧客が、ＵＩを操作しながら仕様の変更を申し出ることもあるでしょう。そのような場合は、まさにその「局面」のみで変更の依頼が出されますので、設計者は、その変更に関連して影響する仕様を探さなければなりませんが、ＵＩの操作だけで十分な検討を実現するのは難しいでしょう。

モジュールレベルの場合は？

一方、システム全体の要求仕様と違って、モジュール（関数）レベルの仕様については、必ずしも「文書」にこだわらなくても構わないでしょう。ソース上でのコメントや「ASERT文」などを上手に使うことで、その関数の「仕様」のほとんどが表現できるかも知れません。そのような状況では、関数仕様を別に文書にすることの合理性は薄いと云わざるを得ません。ただし、だからと云って、システムレベルの要求仕様書や本来の「設計書」まで省いてもよいと云うのは飛躍に過ぎます。

ソースと文書の役割分担を明確した上で、合理的な判断が求められます。

（次号に続く）

「みずほ」 の意味するもの

▲みずほフィナンシャルが、９月２９日に誕生した。総資産が１３４兆円という巨大バンクである。だが巨大なのは総資産だけではない。公表されている不良債権も４兆７千億円と巨額であるし、投入された公的資金も巨額である。
▲確かに、２１世紀を前にして世界的に銀行の統合が進んだが、そのほとんどはすでに終えている。そんな中で「みずほ」がようやく動きだしたわけだが、この巨艦銀行は、いったいどこに向かい、どのように戦うのだろうか。預金者に対して、１%の利払いが発生しただけでも相当な負担となる。
▲日本の銀行は、預金者から集めた資金を運用して、まずは自分達の欲求を満たし、余った資金を利息として預金者に配ってきた。金融再編にあたっての行動を見れば分かるし、この発想は、大量の公的資金を投入した後も変わっていない。日本の銀行は、低金利で選択肢を持たない国民の犠牲の上に成り立っているのである。
▲バンカーとしての能力をどの程度持っているか怪しいが、それよりも２１世紀を前に、なぜ世界で金融機関の統合が起きたのか。そのことにきちんと答えを出せなかったら、かって戦いの趨勢を見誤って巨艦主義の最後を象徴的に飾った「戦艦大和」の二の舞いになりかねない。