第３０号
平成４年３月

（株）システムクリエイツ

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

プロセスレベルの改善 3

プロセス・レベル

前回、ハンフリーは成熟度を５段階に分けたことを紹介しました。則ち、
１）Initial level （初期のレベル）
２）Repeatable Level （反復可能なレベル）
３）Defined level （定義されたレベル）
４）Managed level （管理されたレベル）
５）Optimizing level （最適化されたレベル）
の５段階です。

こうした分類は、その企業のソフトウェア開発能力を効果的に向上させるために必要なものです。最新のＣＡＳＥツールを購入しても、そのツールは生産性を向上させる魔法を持っていないし、それを使う人の能力以上の能力を発揮してくれるわけではありません。実際、現在出回っている殆どのＣＡＳＥツールは『構造化手法』に基づいており、この手法に精通していなければ、直には効果が出てきません。さらに、開発作業がチームで行なわれている場合は、一人だけこの手法に精通していても、やはりツールを使う効果は少ないでしょう。
それではメンバー全員に『構造化手法』を教えればどうだろう？　当然ある程度の生産性や保守性といった品質の向上は見込めるでしょうが、開発プロジェクトそのものがチームで行なわれ、各々のエンジニアの分担している部分（モジュール）が独立して機能するわけではなく、互いに関連し合って作動するものであることを考えれば、各自が作成した仕様書類は、少なくとも隣接するモジュールを分担している者同士は、交換し合えるものでなければなりません。また『構造化手法』に基づいて書かれた仕様書といっても、手法自身はそれらの表現の仕方や問題解決へのアプローチに対して、一つの基準を与えるものであって、各々の作業の成果物は、各々のエンジニアの頭の中から作り出されたものであることには変わりはないのです。

従ってそのようにして書かれた仕様書と言えども、レビューやウォークスルーの習慣がなければ、それらの成果物は単に独善的に書かれたものという域を出ないことになります。
また、最近では『オブジェクト指向』のプログラミングや分析手法が“トレンディ”ですが、同じ様にオブジェクト指向というアプローチ方法を知ったところで、それが活かせる状況になければ、実際には効果を発揮しないし、その前に使ってみるチャンスすら掴めないでしょう。
このように、優れた手法やツールを駆使し、より効果を上げるには、それらに習熟することは勿論ですが、同時にそれらを駆使できる状態に持っていかなければなりません。プロセス・レベルは、どの様にすれば、それらを駆使できる状態に移行できるのかを考える際に、重要な指標となるのです。
以下、各段階のレベルの状態について簡単に説明します。

初期のレベル

この段階は場当たり的で、組織として作業に何の手順も与えられていない、つまりほとんど全面的に担当者の『善意』に委ねられている状態です。
見積の方法も持っていませんし、プロジェクト管理も行なわれていません。ただスケジュールを表す「ガン・チャート」が一枚あるだけです。ＰＥＲＴ図なら、作業項目を関連させて表したり、後になって作業が見えてきたときに、関連させて追加しやすいのですが、ガン・チャートでは一般にその様な気分が沸いてきません。そのために、そのチャートからは具体的な作業はよく見えないものです。ましてや、何処に問題が潜んでいるのか、どの作業の中にスケジュールを遅らせる原因が隠れているのか、全く分かりません。実際に遅れが現実のものとなって初めてそのことに気付くだけです。そのスケジュールが表しているのは、全てのことが期待通りに運べば、この様な予定で完了するはずですということだけです。しかしながら実際にはこの通りに進むことはありません。
ソフトウェアのプロジェクト管理は、どれくらいの作業が何処に発生するかを、出来るだけ早く知ることから始まります。それでも初期の段階で全てのことを予見しきれないでしょうから、途中で何度も手を加えることになります。それが単純なガン・チャートでは、その様な変化に対応することは難しいのです。にも関わらずこの段階の組織はそこから抜けてきません。
したがって予定の作業が予定どおりに終わるかどうかは担当者もやってみなければ分からないし、まして管理者がその様な事を外部に約束できるはずはないのです。そのため、ソフト会社の管理者は、実際に納期が迫ってくると、担当者が入院でもしてくれないかと願いたく（？）なることもあるのです。或いは客先から以前に提示された仕様書の欠陥を探し始める人もいるでしょう。

この段階はツールも旨く整備されていません。それは誰も本気でツールを使うための調査をしないし、第一どの様なツールがあれば、何が改善されるのか、といったことを考えている人は一人もいないのです。いや、一時期はその様な人も居たかも知れませんが、時が経つと共に、次第に考えなくなってしまうものです。実際に後からこの段階の組織に参加した人が、先輩達を出し抜いてこの様な提案が出来る状況にないのが、初期のレベルの組織の空気でもあるのです。
プログラムの変更制御も担当者に任されるだけで、組織的にコントロールされることはありません。バグが報告されたのを受けて、最も可能性のある人（どうやって判断するのか？）がその原因を追及し、一夜明けて見事にその問題を解決するのです。この時、その解決方法が適切なものかどうかは全く問題になりません。問題にするのは報告されたバグの現象が消えたかどうかだけなのです。バグの原因究明を任された人は、そのバグの現象を消すことが役目ですから、何とかしてその役目を果たそうとします。だがその時、同時に新しいバグの原因を埋め込んでしまうことがしばしばあるのです。これは初めのバグの解決方法が明確に管理されていないし、それを検討する手立ても持ち合わせていないのが最大の原因です。残念ながら、原因や解決方法が“分かった”という時、そのことの全てを分かったのではなく、“自分が分かったと思う範囲で分かった”だけであることに気付いていないのです。
そしてこの段階の組織は、どうしても早くコーディングやテストに入ろうとする傾向があります。プロジェクト管理や変更管理が行なわれていない分、少しでも前に進めておきたいと思うのか、或いは管理者の単なる安心感の為なのか。勿論本当にコーディングに入れる状態なら、早く入ればよいのですが、その検討が全く行なわれることなくコーディングに入ります。その判断が担当者個人に任されたままであるため、というよりも誰からもその判断を求められていない状況に於ては、このレベルの組織の担当者は、精神的安定を求めてさっさとコーディングに取り掛かるのは自然の成り行きでもあります。

（次号に続く）

暁鐘（かね）の音

13

カンニング

子供の時からカンニングはず～と禁止されてきた。もちろん今だって禁止されている（筈だ）。いや、今はあんまり聞かないところをみると、こんなことを言わなくても、誰もカンニングなどやらないのだろうか。だとすると随分と行儀の良いことである。

現実のテスト自体が覚えていることの量や質を問う以上、これを禁止しなければ確かに公正さを欠くことになる。そこでは、あくまでも自分でこつこつ努力して、いっぱい頭に詰め込んで、テストの問題に合わせて頭の中から引っ張りだしてくることを求めている。だから覚えてないことを問題に出されたらお手上げである。その時一メートル隣の机の上に開がっている答案用紙を見ることはルール違反なのである。

■■■■■■■■■■

だがこれは学校を卒業するまでの話である。にも関わらず卒業してエンジニアとして企業に就職しても、長年染み付いた【カンニング禁止】の習慣はおいそれとは抜けてくれない。もはや記憶を試すようなテストは出されることはないし、企業もエンジニアにそんなものを求めてはいないにも関わらずである。

■■■■■■■■■■

一般に学校で教えていることは、物事の基礎か、或いは過去の事である。それらを基にしてこれからどうするか、ある目的を実現するために、どれとどれを組み合わせれば良いのか、ということは教えていない（それは個人の努力の範疇に入っている）。

むしろ重要なのはこのような『未知』に対する対応の仕方である。それは今初めて彼の目の前に出された問題であるから、そのこと自体、彼は学校で教わっていない筈である。本当はその中の一部は、教わったことを応用できるものでも、問題を旨く分解し分割しなければ、そのことは見えてこない。そして悲しいことに、このように分割する方法も彼は教わっていない。

■■■■■■■■■■

今や日本中の教育機関は、「要点」を教えることに全力を傾注しているし、生徒の方も、手っ取り早くそれを求めているのである。学生時代を謳歌するには「思案」や「思索」に深入りするわけには行かない。こうして考えることを避け、また本来考えることであっても、数多くパターンを覚えることで代用してきた彼らは、覚えることに慣れ過ぎた。

そうして覚えた要点が彼にとって財産である以上、彼はそれを誰にも教えないだろうし、誰かに教えるという行為そのものを、彼は認めないだろう。そうして大事に仕舞っておいた知識が、年月と共に陳腐化しても、彼はそれを大事に持ち続けることになる。

■■■■■■■■■■

これは「カンニングをしない」事の裏返しでもある。自分は決してカンニングしないし、するつもりもない。正に『自力本願』なのであって、恐らく『他力本願』など認める余地は無いだろう。したがってその代わりに人にもカンニングさせないのである。カンニングすること、人に答えを聞くことは卑怯であり、悪いことである、と無意識の内に反応している。このことは、日本のエンジニア集団に『レビュー』や『ウォークルー』が根付かない原因の一つであると思われる。

■■■■■■■■■■

『自力本願』の権化と思われている研究者といえども、優れた人達は自分の知っていることを人に話し、代わりに自分の気付いていないことを得ようとする。大きな発明や発見はこうして生まれる。

エンジニアは多くの場合、チームによる共同作業が中心である。一人では出来ないことをチームで実現しようというのである。自分の知っていることを教えることによって、彼は一段高いところへ足を伸ばすことも出来るし、レパートリーを広げることも出来る。と同時に、彼の知識を分けてもらった人は、その分ステップアップ出来るのである。

自分を活かそうと思えば、先ず人を活かすことを考えた方が近道なことが以外と多い。自分で「分かった」と思ったことが、実は以外と狭い範囲であったということを、年配のエンジニアは何度も体験しているはずである。お互いの個性や人格を尊重し、ぎりぎりの所まで協力することは、むしろ「甘え」を許さない、厳しい状況である。

■■■■■■■■■■

企業がエンジニアに求めているのは、未知への挑戦である以上、お互いにカンニングし合わなければ、このような挑戦そのものが、まるで別の世界の話になって、日本の企業から何も産み出されなくなってしまう。■

今月の一言

「後生畏るべし。焉（いずく）んぞ来者の今に如（し）かざるを知らんや」

論語（子罕）

後生とは先生に対峙する言葉で、分かりやすく言えば「後に続いてくる者」で、「来者」とは「後生の将来」、「今」とは「孔子自身の今の状態」です。若いということは、それだけ多くの可能性を秘めていることでもあります。「後生畏るべし」は、その様な若者の未知の力に対する畏敬の念なのです。よく〈今の若いものは〉と言って、若者の不足を嘆きますが、そんな人に限って、本当にその人を越える様な若者が現われたら慌てることでしょう。（実際にはその様な人の前には現われません）

人が育つには二つの要因が必要です。一つは本人の意識、というよりも『生成化育』の考え方を受け入れることの出来る性格で、もう一つは陰に陽に支えてくれる人がいることです。何時も助け舟を出すのではなく、時には障害となるのも支えの一つの形でもあります。

受験勉強の形に慣れ切ってしまっている今日、「後生畏るべし」が実現するかどうかは、本人よりも周囲の力の方が大きいかも知れません。この句は、後に続く者に対して、自分を踏み台にして越えて行って欲しい、という孔子の悲痛な叫びに聞こえます。