

設計の本質とは何か ——設計者の思考過程を分析する

ソフトウェアコンサルタント
(ニルソフトウェア)
伊藤 昌夫
いとう まさお

よりよい設計をしたい。これが設計者の共通の望みです。この望みは、設計方法論やツールによってかなえられると宣伝されています。しかしそれらを使えば、何も考えなくてよい、というわけではありません。方法論やツールはしょせん、設計を支援する道具にすぎません。設計業務のなかの複雑な思考の過程を分析することは、方法論を導入したり、設計環境を構築するうえで、不可欠な作業です。

●問題を解決する典型的な3手法

さて、設計者は設計業務を行うために、どのような戦略をとるのでしょうか。たとえば次の三つの方法が知られています。

- (1) 問題分割と解合成
- (2) 事例の検索
- (3) 弛緩法

“問題分割と解合成”は、問題を簡単に取り扱える単位に分割していく方法です。分割して得た副問題がまだ大きい場合には、さらに分割します。結果として階層構造が得られます。細分化された副問題の解を合成することで、全体的な解を得るという方法です。

“事例の検索”は、類似の問題を見つけて、その解を参考にする方法です。数学者ポリアの本⁽¹⁾を見ると、この方法が数学的問題の解決策として重要であることがわかります。

“弛緩法”は、計画問題によく用いられる方法です。われわれが業務スケジュールを立てるとき、各作業単位にはさまざまな制約がついてきます。たとえば人や機械といった資源が制約になります。“弛緩法”はそれぞれの制約の程度に応じて条件を緩和させながら、最適な解を求める方法です。

たとえば、LSIの階層設計は、典型的な“問題分割と解合成”の方法です。また、設計資産を簡単に再利用できるしかけを作るというのは、“事例の検索”にあたります。ただ、これらの方法によって、設計の苦勞から完全に解放されるかというと、そうでもありません。たとえば“問題分割と再合成”の方法によって扱いやすい副問題を得ることができても、どのように分割するかという問題は依然とし

て残っているからです。

●一度に多くのことは考えられない

われわれが問題解決のために考えるときのことを振り返ってみましょう。もっとも典型的な思考の欠点(と思われるもの)は、「われわれは一度に多くのことを考えられない」、ということです。われわれはミスを犯します。最もありがちなのは、ある観点から見ると正しかったものが、それが満足すべき別の観点から見ると間違いだった、という場合です。

このとき、われわれはどうするのでしょうか。まず意識の中で、何かの観点によって問題の分割を図り、分割点に印をつけます。次に別の観点で見て、先の印と比較し、どちらが妥当かを考えます。これをすばやく、くり返し行うことで、その時点でもっとも妥当と思われる解に到達します。多様な複雑さをもつ問題の分割をこのように位置づけると、設計過程とば異なった観点に基づく、すばやい意識の移動のくり返し”ということになります。

システムの要求分析の研究で有名なGuindon氏は発話プロトコル分析(被験者が課題遂行中に考えたことを口に出し、その記録を分析することによって思考の過程を推測する方法)を通して、意識に直接現れない設計過程の三つの特徴を挙げています⁽²⁾。

- (a) 何度もくり返し行われる。
- (b) 過程が分岐する。
- (c) 定まった順序があるわけではない。

このように「われわれは一度に多くのことを考えられない」という欠点を補うために、一つのことをじっくり考えているつもりでも、高速にさまざまな観点到焦点をあてながら考えています。ちょうど、あるものをじっくり見つめているつもりでも、微視的には瞳は微妙に振れながらその輪郭をなぞっている、という現象に似ています。

さて、Guindon氏の2番目の解析結果である“過程が分岐する”という項目に着目してみます。これは観点的切り替えに相当します。それでは、観点的切り替え(過程の分岐)が成り立つ条件とはなんでしょう。AからBへの観点的切り替えが生じるとき、必ずしもAか

らBに論理的なつながりがあるとは限りません。むしろ、Aの観点で考えているときに、「あっ、そういえば」ということでBの観点に移行する場合のほうが多いように思われます。つまり不連続の流れです。したがって、3番目の解析結果である“定まった順序があるわけではない”ということも起こるわけです。

●ツールは思考を支援する方向へ

一般に、開発の初期においてわれわれがぶつかる問題はたいてい不斉構造(ill-defined problem)になっています。つまり、数学的にきっちり定義することができません。ユーザ要求が不明瞭であったり、問題は明確であっても、解決すべき技術という点から見ると、定義がむずかしくなります。

問題が不斉構造である場合、単純な解法を事前に定義することはできません。したがって、上記のようなさまざまな観点から問題を分析したり、評価したりといった、“くり返しと分岐”が必要になります。そうして、徐々に問題は明瞭になっていきます。

ここで、現在の設計方法論や設計支援ツールのほとんどが、主として問題が明瞭になって以降の工程を支援していることに気づくと思います。これは開発初期よりも開発の後半の方が重要だからというわけではありません。不斉構造問題の解決は、今のところコンピュータでは支援しづらい活動だからです。

したがって、今後の設計において重要な点は次のことになります。方法論や支援ツールは、今後、不斉構造問題を含む開発初期の段階から支援する必要があります。そのためには、人間の思考の特徴である“くり返しと分岐”というものを十分に考慮する必要があります。このような方法論やツールを生み出すことができれば、これらはよりよい設計を実現するための大きな助けになるはずです。

参考文献

- (1) G.ポリア(柿内賢信訳)；「いかにして問題をとくか」、丸善、1954年。
- (2) R.Guindon, B.Curtis；“Control of Cognitive Processes during Software Design: What tools are needed”, CHI 88, ACM, 1988。