モデルを書くこと

ソフトウェアシンポジウム 2009　モデリング WG　ポジションペーパー

株式会社日本システムディベロップメント　　時本　永吉

1.　自己紹介

社内向け業務システムの受注開発がメインのプログラマ。他に、フレームワークのカスタマイズや、品質管理をしてきた。言語は Java が多いが、オブジェクト指向とは言い難い。最近、自分の考えの未熟さに気づかされ、自分が考えていることを整理している。

2.　考えていること

2.1.　モデルって何だろう

プログラマとしては UML、ER 図が先ず頭に浮かぶ。何かを表現して、みんなで共有するためのものなのだろうか。その為に、絵（記号、形式的表現）で表現しているのだろうか。確かに、形式的な書き方は余計なことを考えず、検証しやすく、意識違いを防げる。

自分と他人の考えは絶対に一致しない。なぜなら、それを証明する術がないからだ。「一致する」ことを証明するには、あらゆる事態において「一致する」ことが証明可能でなければならない。しかし、同一の対象でも事態によって意義（解釈）は変わりうる。「あらゆる事態において」を証明するには無限にある条件を満たさなければならず、完全に証明することはできない。これに対して、「ある問題領域においては必然的」であることが証明できる「問題領域」を定めれば、この問題を解決できる。「問題領域」の定義はシステム開発においては単純だ。ユーザーの事業をありのままに捉えればいい。そして、他に変な解釈ができないよう形式的な表現をすればよい。すなわち、UML を書いたり、プログラムを書いたりすることだ。文章（設計書）でも構わないが、形式的に書くか、変な解釈をしないよう、関係者を教育しておく必要がある。これにより、プログラマとプログラマ、プログラマとユーザー間で意思疎通することができるようになるのだ。

話が複雑になってしまったが、結論として、モデルはコミュニケーションのためのツール、または表現方法の一つとする。

2.2.　モデルを「書く」のか「描く」のか

クラス図や ER 図は「図」であるため、日経では「描く」で統一しているらしい、「書く」または「表現する」という方もいる。

私は「書く」を主張する。もともとは論理的に表現することを、わかりやすく図で書いたのに、それなのに、「描く」を使うのは上げ足をとられた感がする。これは一見、どうでもよさそうに思えるかもしれないが、実はこれが大きな問題であるのではないかと考える。問題は「描き手」ではなく、「描いたものを観る側」にある。綺麗に描かれた図、例えばクラス図なら関係が抽象的に表現された図を観ると、オブジェクト指向としてよく出来た気になってしまう。しかし、それは表現したいものを正確に表せて

いない。少なくとも、自分の感じたことを表現しただけで、コミュニケーションのために表現されたものとは言えない。表現されたものが正しいかどうか、書かれた論理を正確に読み取らなければない。これができていな人が私のまわりには私を含めて多い。そんな人たちが設計して（描いて）、検証して（観て）いるのだ。クリティカルなシステムなんて出来るわけがない。現代のシステム開発の問題として事業や技術の多様化、要求スピードなどが挙げられることがあるが、一番の問題はこれであると考えている。

3. TM（T 字形 ER）でモデルを書くこと

モデルは事実を正しく書き表されていなければならない。そして、正しく書き表されていることが論理的に検証できなければならない。

ER 図はすべてを一律に並べているだけでしかなく、事態を表すことは出来ない。事態はシステム設計書やプログラムといった別のモデルで表現される。これを正しく行えるのであれば、ER 図を使うことに何の問題もないが、たいてい、これらモデルの作成者は異なり、さらに意思の差異が生まれてしまう。また、同じものを表すモデルが複数に分割されるため、メンテナンスが大変になる。

また、プログラマはユーザーが正しく考えることができるように補助することが役割であるべきだ。自分たちの勝手な考えを押し付けることではない。そして、その考えが正しいことを保証できなければならない。それも出来ずに、すぐに「完全なものはつくれない」とか「運用で対応」とか言うプログラマは信用できない。

佐藤正美氏が考案した TM（T 字形 ER）は以上のことをやりやすくするためのモデリング手法の一つである。TM は現実の事態（ビジネス）を表し、現状分析と環境適応能力を見直すためのモデルである。このモデルはデータモデルにもシステム設計書としても使用できるため、複数のモデルを書く必要はない。また、TM は関係の制約束縛を重視しており、これについての一般手続きを定めている。TM によるモデリングが最適とまでは言わないが、事態をそのまま書き表すため、頭の中を整理し易く、検証もし易くなる。

4. 展望

TM で書いたモデルの検証方法に一般手続きはまだない。しかし、事実の中に埋もれた制約束縛関係をみると、クラス理論により一般手続きが出来そうだ。また、クラス理論を整理すれば、設計書の自動生成部分が増えるだろう。一般手続きで表せる工程が増えれば、新人が 10 年、20 年のベテラン・プログラマと同じものが作れるようになる。

プログラムもモデルであるので、分析・設計と一体化したい。現在の TM では論理関係をデータで管理する程度。それでも十分な恩恵を得られるのだが、もっと楽をしたい。欲を言えば、ユーザーに要件をヒアリング中、レビュー中に動くものが作れるくらい。候補としては Grails を考えているが、MDA も興味深い。

最終的には、ユーザーにシステムを開発してもらえるようにしたいと考えている。