GSLetterNeo Vol.50
2012年9月

構造化手法のススメ

オブジェクトモデリングスペシャリスト
土屋　正人
Masato Tsuchiya
m-tsuchi@sra.co.jp

◆ユースケースは役に立つか？

ソフトウエア開発に UML が使われるようになってずいぶん経ちます。UML を使って開発を行う場合、機能要求はユースケースモデルで表します。ユースケースモデルの要素であるユースケース図は、アクタとユースケースを使ってシステム境界を表現しますが、インタフェースの詳細は表現しません。そのため、何がやり取りされるか、どのようなイベントで起動されるかを知るには、ユースケース記述と呼ばれる文書を読む必要があります。この文書では、イベントフローと呼ばれる、システム外部とユースケース間のやりとりを記しますが、通常、自然言語で書かれるため、読むのも、整合性をチェックするのも大変です。

さらに、本来グレーボックス(What)で書くべきところを、ホワイトボックス(How)で書いてしまいがちで、要求に変更があった時には、結構な量の追加変更を行うことになります。保守性が悪いことこの上ありません。

◆構造化手法再び

オブジェクト指向手法が登場する前は、構造化手法がよく使われていました。構造化手法(構造化分析(SA: Structured Analysis)、構造化設計(SD: Structured Design))は、1970 年代後半から 80 年代にかけて広まったソフトウエア開発手法で、次のような成果物を作りながら開発を進めていきます。

■構造化分析
・データコンテキストダイアグラム(DCD)
・データフローダイアグラム(DFD)
・プロセス仕様書(PSPEC)
・データ辞書(DD)
・データ構造図(DSD)
・E-R 図(ERD)
■構造化設計
・構造化チャート(SC)
・モジュール仕様書(MSPEC)
■リアルタイム構造化分析
・制御コンテキストダイアグラム(CCD)
・制御フローダイアグラム(CFD)
・決定表(DT)
・状態遷移表(STT)
・状態遷移マトリクス(STM)
・状態遷移図(STD)

E-R 図は初期の構造化手法では使われていませんでしたが、データ表現を強化するものとして取り入れられ、現在でも DB 設計に使われています。

構造化手法では、システム境界を表現するために DCD(データコンテキストダイアグラム)という図を使います。DCD では「外部エンティティ」あるいは「ターミネータ」と呼ばれる矩形で開発するシステム外の存在を、「プロセス」と呼ばれる円で開発システムそのものを、それぞれ表現します。ターミネータとプロセスとのインタフェースはデータフローで表します。DFD(データフローダイアグラム)の特殊形(プロセスがひとつだけ)であり、Top Lebel DFD と呼ぶこともあります。

イベントドリブンシステムの場合には、リアルタイム構造化分析手法の CCD(コントロールコンテキストダイアグラム)を使います。書き方は DCD と同じですが、データフローの代わりに制御フロー(破線の矢印)を使います。

DCD と CCD を合成した図をユースケース毎に作ると、ユースケースに対するインタフェースが明確になります。

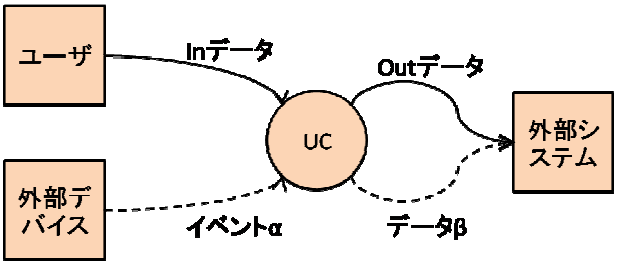

図 1　DCD+CCD

ユースケースはブラックボックスであることが望ましいのでこれで十分といえますが、ユースケース記述に相当するグレーボックスレベルの記述が必要であれば、DFD のルール（階層化ルール、平衡化ルール）に従って１つ下の階層の DFD を記述すればよく、整合性のチェックも容易に行うことができます。

ユースケース駆動アプローチでは、ユースケース図からバウンダリオブジェクト、コントロールオブジェクトを、ユースケース記述からエンティティオブジェクトをそれぞれ抽出して、それらのコラボレーションを検討します。この作業をユースケース分析と呼びます。ユースケース図とユースケース記述の代わりに DCD+CCD を使う場合、データフローとコントロールフローから「もの」「こと」が識別できるので、これらをオブジェクトとすることで、ユースケース分析と同様の成果を導き出すことができます。**インタフェース駆動**と言えるかも知れません。

◆構造化手法のススメ

オブジェクト指向、UML、アジャイル開発が認知され、広まるに従って、クラス図とシーケンス図だけあれば OK、最悪これらも作らなくてもいいから早く動くものを作ろうという風潮（誤解）があるような気がします。オブジェクト指向、UML、アジャイルを進める者として、これらを払拭するにはどうすればいいか、難しいところです。**シンプルさは、対象や行為が複雑であることを十分把握した上で、状況に応じて削ぎ落としていくことで生まれる**と思います。また、そのことを理解している人たちが、オブジェクト指向やアジャイルに至っていると思います。アジャイルソフトウエア開発宣言にある、「包括的なドキュメントよりも動くソフトウェア」により価値をおくというのは、ドキュメントは価値が低いということではなく、ドキュメントの価値を十分認識した上での宣言でしょう。

オブジェクト指向手法の多くは、構造化手法を提唱していた人たちによって作られました。その事実を思うと、オブジェクト指向分析設計やアジャイル開発を始める前に、基礎として構造化手法を理解しておくことが必要なのではないかと考えます。

構造化手法で使われるモジュール評価尺度のひとつに「凝集度（強度）と結合度」があります。構造化手法で開発する場合、凝集度が「情報的」「機能的」（最も強い凝集度）になるように、結合度が「データ結合」（最も弱い結合度）になるように、それぞれ注意を払いますが、オブジェクト指向の考え方によってこれらを達成しやすくなりました（オブジェクト指向設計原則として 10 数種類の原則に発展しています）。

ソフトウエアが複雑化している時代だからこそ構造化手法を見直す価値があるのではないかと思いますが、残念ながら構造化手法の多くの名著は絶版となっています。幸い以下の古典的名著は入手できるようです。

- **デマルコ著「構造化分析とシステム仕様」**
- **マイヤーズ著「ソフトウェアの複合/構造化設計」**

近年のものとしては、**セサミワーキンググループ**が記した**「組込みソフトウェア開発のための構造化モデリング」**があります。

GSLetterNeo Vol.50
2012 年 9 月 20 日発行
発行者●株式会社 SRA 産業第 1 事業部
編集者●土屋正人、柳田雅子、野島勇

バックナンバを公開しています●http://www.sra.co.jp/gsletter
ご感想・お問い合わせはこちらへお願いします●gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋２－３２－８