

ホスト系技術者がOO技術者に変身するための最短路

記： 上手裕(DTS) 2004/06/18

於： 情報処理学会 ソフトウェア工学研究会パターンWG 6回勉強会

<0. ホスト系技術者の定義(この勉強会での)>

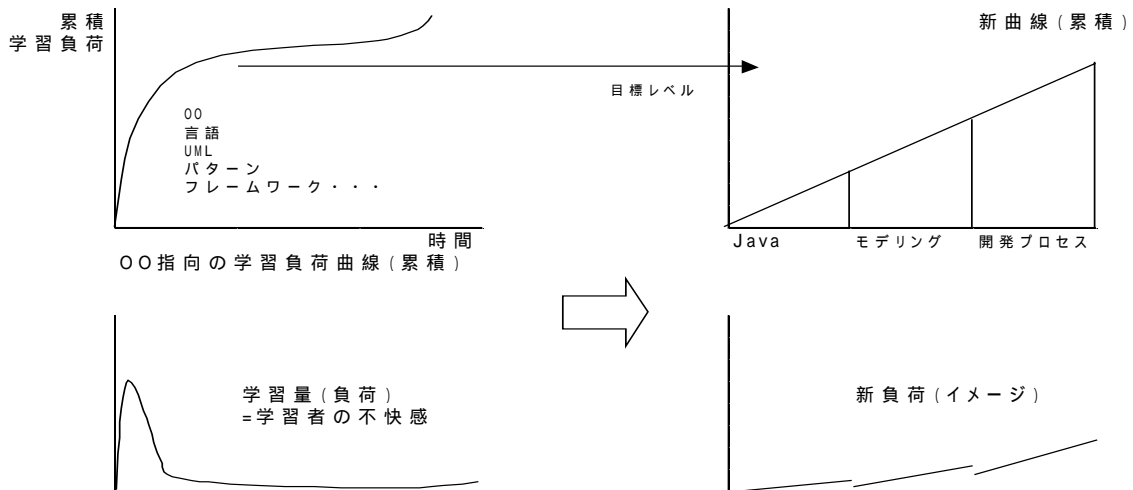
- ・ ウォーターフォールモデルで仕事をしている
- ・ ビジネス系
- ・ 上流の人はコードを書かない
- ・ コードを書く人は要件定義や分析をしない
- ・ 言語は、COBOL や VB+SQL が多い

<1. オブジェクト指向の学習曲線>

学習要素を分解して、学習曲線を直線に近いものに変更する

Java+UML -> モデリング -> 開発プロセスという順番で学ぶ

OO指向の学習曲線と、学習の直線化



<2. Javaを学ぶ>

【ポイント】

- 1 2~4のポイントを学んだら、即刻モデリングに移行する。Javaに長居をしない
- 2 クラスとインスタンスを理解する
 - ・ メソッド、変数における static と instance の違いを理解する
- 3 インターフェースによるポリモルフィズム(多態性)を理解する
 - ・ コレクションとイテレータ(反復)も理解
- 4 Eclipse(統合開発環境)のデバッガが使えるようにする
 - ・ JDKソースも含め、ソースにステップ・インできるようにする
 - ・ メソッドの定義元、メソッドの呼び出し元を自由に参照できるようにする
 - ・ インスタンスの中身を実際に見えるようになること
- 5 UMLのクラス図とシーケンス図の概要を知り、クラス間の関連がどう実装されるかを理解する
 - ・ 単方向関連、双方向関連、関連クラス

【補足】

ホスト系で上流の仕事をしている方は、「要件定義や分析・設計をするのにコーディングの技術なんて必要ないんじゃないの?」と思われるかもしれませんが、確かに手続き型の開発では、分析・設計とコーディングは別の人でも出来ました。しかし、OOではそうはいきません。OOでは、分析段階で抽出して「型」は設計書での「クラス」、ソース「class」としてシームレスに段階的に詳細化されていきます。実装段階で分析クラスの修正がはいるのも日常茶飯事です。ですから、分析・設計の実装イメージがわからないと、分析・設計自身が成立しないと言えます。

またプログラマの方にアドバイスしたいのは、「まず言語ありき、で Java の習得にのめりこまないこと」です。OO で本質的なのは、分析・設計であり、実装言語はなんでも良いのです。OO を実装するのに必要な Java のコア部分を習得したら、即刻モデリングに進みましょう。まず OO 開発の全体像をつかむのが先です。プログラミングはこれから、幾らでもできます。

<3. モデリングを覚える>

【ポイント】

- 1 ごく小さな要件（ユースケース）でエンティティクラス図を書き、シーケンス図でメソッド分析をして、実装してみることを目標とする【重要】
 - ・最初は業務で使う内容でモデリングしないこと。マスタや集計など、集約オブジェクトが出てきて、エンティティのモデリングが乱れてしまう
- 2 ユースケースごとに1つのクラス図という錯覚をなくす。システム全体で1つのクラス図になる
- 3 クラス図とオブジェクト図を自由に変換できるようにする
 - ・オブジェクト図が、クラス図のある瞬間のスナップショット（写真）であることを理解する
 - ・抽象クラスやインターフェースがオブジェクトを束ねる構造を記述する方法であることを理解する（これがわからないとデザインパターンは理解できない）
- 4 エンティティ（主体）がモデルの本質であり、ビジネスルールなどを構成することを理解する
- 5 MVC を理解する
 - ・エンティティクラス図にコントロール（入力や処理）、ビュー（見せ方）を追加する方法を理解する
- 6 ロバストネス分析（ICONIX 方式：参考文献1）を理解する
 - ・エンティティ、コントロール（処理）、バウンダリ（境界：入出力）を使いこなす
 - ・MVC とロバストネス分析の違いを理解する
- 7 モデルがうまく習得できない時は、1で作った動くソースをもらい、デバッガで動きをトレースして、モデルにさかのぼれば良い【重要】

【補足】

「オブジェクト指向とはモデリングである」と言っても間違いなくらい、モデリングは重要です。というのは、オブジェクト指向開発は、「システムをオブジェクト間のメッセージ交換として実現する」ことであり、この作業自体がモデリングだからです。OO 開発では、分析の最初から納入寸前までモデルをながめ、修正する作業が続きます。しかし率直に言ってモデリングは難しいです。抽象的な思考が要求されますし、分析段階ではアバウトに、設計、実装と段階的に詳細化していく階層的な感覚も要求されます。

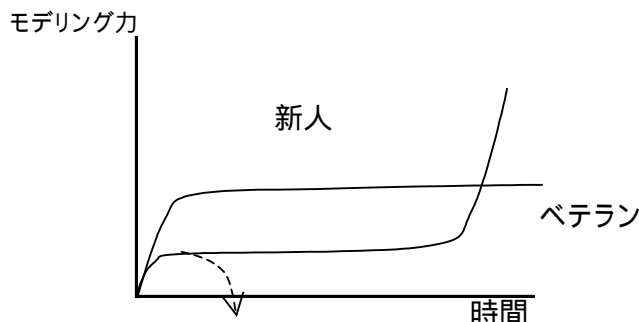
しかし、2点だけしっかり押さえればモデリングは乗り越えられます。

オブジェクト図とクラス図を自由に相互変換できるようになること

クラス図から MVC ではビューとコントロール、ロバストネス分析ならコントロールとバウンダリを自由に削除し、また追加できるようにすること

プログラマの人には、ちょっとしたツールを書く時でも必ずモデルを書くことをお勧めします。ベテランは、多くの良いモデルをみて良いセンスを身につけ、ひたすらモデルを書くしかないでしょう。

【図】モデリングの習熟曲線



【補足】

新人や1, 2年生は立ち上がりから急速に伸びます。しかし、ある所まで行く伸びは完全に止まり、フラットになってしまいます。これは、ビジネスや業務に対する知識が不足しているためです。モデリングの基礎が一通りわかっただら、開発プロセスやデザインパターンに進みましょう。

一方ベテランは一般的に、モデリングの最初は全くダメなケースが多いです。システムのメニュー構成をそのままクラスにしてみたり、処理がクラスになったりマスターDBが出てきたり、というモデルを書いてきます。この原因は、従来の開発での経験をモデリングに当てはめようとしているからです。多くの方が散々悩んだ挙句、「従来の経験が邪魔をしている。過去の知識を捨ててゼロからスタートしよう」と発想を切り替えた瞬間に壁を越えます。ベテランは業務知識が豊富ですし、分析・設計で抽象思考能力は高いですから、壁さえ越えれば、モデリング力はロケットのように上昇していきます。

今までで一番成長したメンバは、RDBモデリングの経験者でした。構造化設計、DBモデリング、DOA経験者は、わずかな実習をするだけでOOモデラーになることができます。

#ただし、エンティティクラスの中に、明確の処理を埋め込まないと、OOにはならないことに注意。

<4. OO開発プロセスを理解する>

【ポイント】

- 1 OOの開発工程の基本要素を理解する (WBS参照)
- 2 OOの開発工程は繰り返し型であり、各フェーズにさまざまな割合で1項の開発工程が含まれることを理解する
- 3 RUP (ラショナル統一プロセス)を理解する

【補足】

ウォーターフォールモデル (以下WF) を提唱したロイス (父) の論文には、WFに対する5つの改善策が同時に記述され、その改善さえ行えばWFの開発リスクはほとんど無くなると書かれています。(参考文献2)

その項目は、

- プログラム設計を最初に行うこと。ソフトウェア要求生成フェーズと分析フェーズの間に予備的なプログラム設計フェーズを挿入すること
- 設計を文書化する
- 同じことを2回行う
- テストの立案、管理、監視を行う
- 顧客を巻き込む

OO開発の事実上の標準開発プロセスであるRUPを体系化したのは、WFを提唱したロイスさんの息子さんです。6年間 (開発規模 (105万行) にわたるミサイル警告情報システム開発の経験をまとめ、オブジェクト指向の3大グルの方法論と統合して (参考文献3) ラショナル社でRUPとして体系化しました。

ロイス (子) 「新時代のソフトウェア管理の法則」として10項目を挙げていますが、上位5項目は以下です。

- アーキテクチャ先行アプローチ
- 反復型ライフサイクルプロセス
- コンポーネントベースの開発
- 変更管理環境
- ラウンドトリップエンジニアリング

お父さんの論文の は、プロトタイプ開発やアーキテクチャ評価を先にしなさいということで、 はこの改良形とも見えます。また は、繰り返し開発そのもので、 と同じように見えます。ロイス (子) は前出の本の中で、「この論文に示されている洞察にいつも圧倒されてきた。今日の技術の状況で適用される場合でも、理論面で大きな欠陥はない」と述べています。何のことはない、我々が思っていたWFはロイス (父) が提唱したWF理論と異なったものでした。本来のWFは非常に柔軟な開発プロセスだったのであり、その進化系がRUPであるとも言えるわけです。

RUPは概念的に非常にわかりにくいですが、上のようにWFの改良と考えると簡単に理解できます。方向付けフェーズはアイデアに焦点があり、推敲フェーズはアーキテクチャに焦点があります。2つ合わせてエンジニアリングステージで、プロトタイプを作って動かし、アーキテクチャ検討をして量産に移行するためのリスクをつぶします。OKとなれば製造ステージに移ります。作成フェーズは製造の中心であり、ベータリリースをします。移行フェーズでは、製品を出荷します。各フェーズでは、分析、設計、実装、テストの工程を組み合わせ、成果物や実際に動くものをつくって

いることに注意して下さい。

オブジェクト指向開発では、フェーズの繰り返し回数やそこでの処理内容をプロジェクトで定義することが必要になります。これが難しいですが、プロトタイプ作成、アーキテクチャ評価を1～2回やった後、繰り返し回数1回で動かすような方法で動かしているプロジェクトも沢山あります。要は、やりやすく、良い結果が出るようにプロセスを設計すれば良いわけで、原則論にとられる必要はありません。

応用編

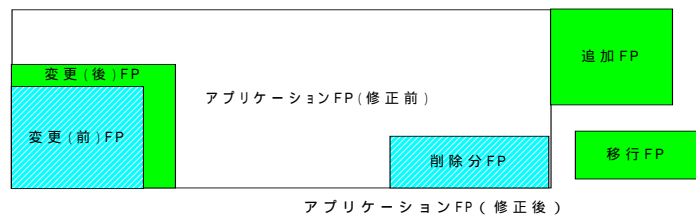
<5. 仕事の規模感をつかめるようにする> (3日で十分)

【ポイント】

- 1 機能規模、開発規模、開発工数の違いを理解する
- 2 WBS (業務分解構造) の作り方を理解する
- 3 FP 計測が出来るようにする。アプリケーション FP (機能規模)、プロジェクト FP (開発規模>工数)
#FP 計測の本質はデータモデリング

質問: あなたのやっている仕事で

- ・ お客様に提供する価値の総量 (機能規模) はどれ位?
- ・ 部品購入と再利用の方針は?
- ・ 自社開発分の開発規模は?
- ・ 生産性はどれ位?
- ・ 開発工数は?
- ・ その中であなたの分担は?



<6. 部品利用技術を学ぶ>

【ポイント】

- 1 パターン、フレームワークの理解
- 2 J2EE、OR マッピングの理解
- 3 マイ・フレームワーク構築利用方針の確認 (コーディング、モデル、アーキテクチャ)

(終わり)

参考文献

- 1 「ユースケース入門 ユーザマニュアルからプログラムを作る」 ダグ ローゼンバーグ、ケンドール スコット著
ピアソンエデュケーション; ISBN:4894713772(2001/11)
- 2 「ソフトウェアプロジェクト管理」 ウォーカー・ロイス著 日本ラショナルソフトウェア (株)監訳
アジソンウェスレイ ; ISBN : 4795297371(1999/5)
本書は、組めども尽きぬ洞察の塊で、OO 開発に携わる全技術者必読書と思います。
- 3 「UML による統一ソフトウェア開発プロセス オブジェクト指向開発方法論」
イヴァー ヤコブソン、ジェームズ ランボー、グラディ ブーチ著
翔泳社 ; ISBN: 4881358367 ; (2000/03)