ソフトウェアパターンの現状と適用経験

平成15年5月23日

三井情報開発株式会社
佐枝 三郎

ソフトウェアパターンに関する現状調査

- この報告は平成13年度に(社)情報サービス産業協会が行なった「ソフトウェアパターンの実践的利用に関する調査研究」において、国内外のソフトウェアパターン専門家にヒアリングを行なった結果の要約である。

調査報告書
いかにソフトウエアパターンを使えばよいか,
―情報サービス企業マネージャーのための
ソフトウェアパターンガイドブック―, 2002年3月.

この調査の問題意識

- 問題認識：
  - ソフトウエアパターンは有用な概念であるが、わが国のソフトウエア開発現場に普及定着しているとは言いがたい。
- 背　景：
  - なぜか、ソフトウエア開発現場の決定権を持つマネージャ層に、ソフトウエアパターンへの理解が少なく、現場に実践する機会が与えられていない。
- 解決策：
  - マネージャ層が、ソフトウエア開発プロジェクトでソフトウエアパターンを利用するよう動機付ける方法を見つけ、それに沿ったガイドライン資料を作成する。

従来のソフトウエア開発環境でパターンが利用されなかった原因

- ソフトウエア開発の主流であるビジネス・システム開発では、オブジェクト指向設計適用の歴史が新しい。
  - ビジネスシステムの経験者(マネージャ層)は、構造化設計、データ中心設計で育っており、オブジェクト指向設計について一部を除いてなじみがない。
  - パターンはオブジェクト思考設計の成果であり、GoFの提案から5,6年の歴史という最新の概念である。
- パターン(特にGoF関連のパターン)は、デザイン上の議論であり、かつビジネスシステム経験者には難解である。
  - オブジェクト指向設計の素人には、取っ付きが悪く、概念自体が理解しにくい。
  - 事例がウィンドウシステムや、ミドルソフトウエア中心であり、ビジネスシステムとのつながりが分かりにくい。
  - 具体的にお金に結びつく、有効性を示すデータが公開されていない。

ソフトウエア開発環境の変化
―パターンへの追い風―

- ここ5年間でインターネット、WEBベースシステムが、ビジネスシステムの大きな割合を占めるよう変化した。
  - WEBシステムは、基本的にMVCのオブジェクト指向の概念に則っている。
  - ビジネスシステムでも、Java、C++、(C#)などのオブジェクト指向言語、UMLなどのオブジェクト指向設計環境を利用せざるを得ない状況。
  - 特にJavaをベースとした開発環境では、Java自身がパターンを利用して設計され、J2EEのアプリケーション開発にも、パターンの利用が推奨されている。

ヒアリング調査の対象専門家

- 海外の専門家
  - R. Johnson, L. Rising, M. Mans, M. Fowler, J. Vlissides
- 国内の専門家（研究者）
  - 青山 幹雄、中谷 多哉子、佐藤 英人、高木 浩光
- 国内の専門家（企業内の実践家）
  - 友野 晶夫、細谷 竜一、佐藤 啓太、山本 里枝子、
  - 内田 弘之

ヒアリング調査の項目

- 1 パターンを利用したシステム開発経験
- 2 利用したパターンの種類
- 3 パターン利用の動機
- 4 パターンの推進者
- 5 GoFパターンの難しさ
- 6 パターンの適用範囲
- 7 パターンの効果
- 8 パターンの有効性の評価
- 9　パターンとフレームワーク
- 10 パターンはSE全員に必要か
- 11 パターンの教育方法
- 12 パターンを書くことについて
- 13 パターンを浸透させるための管理者の役割
- 14 企業でパターンを浸透させるには

ソフトウェアパターンの適用範囲

海外の専門家のコメント

- パターンはビジネスの分野でも、COBOL、VBなどOOPLでなくても活用できる。
  ヘイのデータモデルパターンがよい例である。
- パターンはOOPLの世界だけでない。他言語でも、ビジネスの分野でも十分に適用可能。
- パターンは頭脳活動の食料のようなもの、様々なシステムの共通性を抽出するのが
  パターンであるから、原理的には無限にある。現在も様々な分野で作られている。
- どのようなシステムにも有効であるが、特にビジネスシステムに有効なパターンが多い。

国内の専門家(研究者）のコメント

- デザインパターンだけでなく、具体的な問題に対応して様々なパターンが有効、
  例えば性能評価パターン
- 色々な分野で利用可能だが、重要なのはフレームワークを構築するために利用すること。
- 基本はGoFのデザインパターン。それ以外のあたり前のノウハウをパターンというのは
  おかしい。

国内の専門家(実践家）のコメント

- デザインパターンだけでなく、分析パターンその他色々が利用可能である。
- フレームワーク作りに適用するのが適切であろう。但しその利用方法のためには
  分析パターンが必要だと考えている。
- 人間の知的活動全て

ソフトウェアパターンの効果

海外の専門家のコメント

- パターンは物事をカプセル化するので理解がしやすくなる利点がある。反対に最初に理解するコストがかかる。
- 全ての面で効果があった。パターンによるフレームワークを利用し、18ヶ月のプロジェクトが3ヶ月でできた。
- パターンは人間の経験を一般化、抽象化する知的活動であるから、当然効果がある。
- パターンはコンポーネント化、部品化に役立つ。同時に設計者の教育ツールとして有効である。

国内の専門家(研究者）のコメント

- 経験者のものであったし性能評価が初学者でも理解できるようになった。
- パターンのよいところは設計時に迷わなくてよい点にある。
- パターンを利用しない場合にくらべ、適用したら1/5の期間で設計を終了。実装も手戻りがなく2週間で完了
- OOPLでの設計の基本であり、設計者の最低レベルのボキャブラリ

国内の専門家(実践家）のコメント

- 設計時にモジュラリティを守らせること、DBのラッピング統一化などに利用。品質面、テスト工数削減などの効果がある。
- 生産性の効果もあるが、品質向上の効果が高い。同様なものを連続して開発、あるいは保守するのに効果が高い。
- パターンを利用してフレームワークを構築する結果として効果がある。実際のプログラムはフレームワークから自動生成する。
- 開発部分を限定したため、特にバグフィックス、テストの工数が削減した。

ソフトウェアパターンとフレームワーク

海外の専門家のコメント

- フレームワークを作る人間にパターンは重要だが、使う人間もある程度理解する必要あり。
- パターンは全ての設計者のものであり、フレームワークを利用する人間も、それを直すためにはパターンを知る必要あり。
- フレームワークは慎重に設計する必要があり、パターンが重要。アプリケーションはそれほどでもないが、保守、変化に対応するためにはパターンを理解する必要あり。
- デザインパターンは技術を高めるためのものであり、全ての設計者が理解すべきである。

国内の専門家(研究者)のコメント

- フレームワークとコンポーネントを作る人は詳細に理解が必要。フレームワークを利用する人は典型例だけ理解すればよい。
- フレームワーク化することは重要。このプロジェクトもパターンを利用したテスト環境のフレームワーク化を検討している。
- フレームワークは多人数で体制を作って開発する場合のもの。パターンは一人で設計する場合でも有効。

国内の専門家(実践家)のコメント

- フレームワークは作成にコストがかかるため利用していない。パターンは知識を伝え共有するもの少人数でやる場合はフレームワークは必要ない。
- パターンはフレームワーク構築の際の重要な手段であり、パターンはフレームワークを構築する人が理解をすればよい。
- パターンを理解している人は開発グループの10%程度でよい。利用者には部品を組み立てるだけである。

ソフトウェアパターンの教育方法

海外の専門家のコメント

•受講生の直面した問題に合わせて、対応するパターンを教えること。
•パターンは講義だけではだめ。実習と自分はロールプレイングの方式で教えた。
•パターンは強制された教育にはなじまない。パターンは人間の内在的なものに依存し、
改善の意欲を持たない人には教えられない。
•パターンは理解者が回りの人間に伝え、自分でプログラムを作って学ぶべきものである。
草の根的な教育が必要

国内の専門家(研究者）のコメント

•業務の実態、自分の直面している問題に対応する具体例に基づいて学習すること。
またプログラムを実際に書いて実践で覚えるべきである。
•パターンを利用してうまくいった事例が増え、その事例に基づいて教育するのがベスト
•設計者がモデリングを行い、自分でパターンを使ってみることが重要。
•GoFのパターンの本質を理解させること、OOPLにおける処理の共通化、抽象化の必然で
あることの教育が必要。

国内の専門家(実践家）のコメント

•基本的にOOPLの基礎とUMLを理解したメンバーを対象としている。
自分のパターンは部門で共有しているので、新入者は周りに聞きながら自然と覚える。
•パターンを利用しないで作ったプログラムをパターンを用いて作り直すことで教育。
•専門家とコンセプトを考え、レビューして、面白いと発見するところにパターンが生まれる。
現場の経験を抽象化できる能力のある人を見つけること。・

ソフトウェアパターンの作成

海外の専門家のコメント

- パターンを利用するにくらべ、パターンをまとめる事は能力がいるし、時間がかかる。
  コンサルタントや企業のテクニカルリーダーの仕事である。
- パターンを書く能力を持つ人は限られている。しかし書く人がいなければ、他の人間は使えない。
  それぞれの文化に適したパターンが書かれるのを望んでいる。

国内の専門家(研究者）のコメント

- 設計者にモデリングの時間を与える必要がある。設計者も自分のプロジェクトで問題を抽象化し
  パターンを抽出する試みをやる必要あり。

国内の専門家(実践家）のコメント

- パターンを書ける人は社内、協力会社を含め、10人に一人はいる。
- 新しいパターンを書く能力があるのは100人以上いる開発グループにうち、2, 3人である。

ソフトウェアパターン浸透のための管理者の役割

海外の専門家のコメント

- パターンが有効だと確信し、実践すること。専門組織を作るのではなく、日常に問題が起きればパターンを使って解決する、こういう風土を育てればよい。
- 設計者がよいものを作りたい、勉強したいという環境を支援すること
- 管理者は設計者を勇気付け、新しい技術を発見するように仕向ける役割がある。パターンもそのひとつである。
- 設計者に新しいテクニックを覚えさせ、勇気付けること。

国内の専門家(研究者）のコメント

- ソフトウエアの製造体制がフレームワーク提供と、カスタマイズ提供に別れる必要あり。そうすればフレームワーク提供者側はパターンをよい利用するであろう。
- 管理者はモデリングを行っている設計者に余計なことを言わない、させないことで専念させるべきである。

国内の専門家(研究者）のコメント

- アーキテクトをリソースだと認識し、継続的に育成すること。コミュニティーを作る環境整備。
- フレームワークやパターンが開発技術の維持に重要だと言う認識を持ってもらうこと。
- パターンを利用しフレームワークを作る部隊を確立すること。そしてその部隊にフレームワーク構築の分析力、構想力をつけさせること。
- 優秀なアーキテクトを見つけ、それを中心にパターンを広めること。また開発を丸投げしないで社内で行い、技術の向上へのインセンティブを持つこと

企業でソフトウェアパターンを浸透させるには

海外の専門家のコメント

- 技術的に先進的な設計者を刺激し、平均的な人間に広めること、またパターンのエキスパートをコンサルタントに雇い、具体的に説明してもらうこと。
- 献身的なリーダを作ること。
- 設計者、プログラマに本を読み、パターンを実践する時間を与えること、また理解を手助けする人を準備すること、こうした環境を管理者が用意することである。
- 設計者にパターンを勉強する時間と環境を与えること。パターンはよりよいデザインの知識であるから、必ず成果がある。スキルレベルが上がり、仕事の手本となる。

国内の専門家(研究者）のコメント

- オブジェクト指向設計の機会をもっと増加させる。また設計者がパターンをもっと使う努力をするべきである。
- 現在のような納期とコストだけでシステム開発を評価するのではなく、保守まで含め品質保証を行う契約も含めたビジネス形態をとっていく必要がある。
- 設計者にモデリングの時間を与える必要がある。パイロットプロジェクトでパターンを実際に使って、失敗をさせそれを通じてパターンのよさを理解させる必要あり。

国内の専門家(研究者）のコメント

- プロジェクトリーダの機能を分解してアーキテクトを役割化し、複数プロジェクトで活用すること。またトップダウンでなく、インフォーマルなアーキテクトのコミュニティを作り、皆が相談しあえる風土を作ること
- フレームワーク構築部隊と利用部隊に分けること。またフレームワークの効果を利用例などで明示すること
- 開発を実際に社内で行い、実践的にパターンを利用すること。実際に対象にあるパターンを見極め、プログラム、テストの自動化に結びつくパターン利用を行うこと。

調査結果のまとめ
ソフトウェアパターンの導入手順

- プロジェクトメンバにパターンを考える時間を与える。
- パターンを考え、他のメンバーの成果をレビュー、指導するメンタを育成する。
- 社外のパターン専門家による講演など、パターンに関する最新情報が常に社内に取り込まれる仕組みを作る。
- 作成されたパターンを、社内（部門内）に公開し、多くのメンバに利用してもらうための仕組みを作る。
- 優れたパターンを作成したエンジニアを勇気付け、奨励する仕組みを作る。
- 公開されたパターンの利用状況を追跡し、利用した成果と改善点をレビューできる仕組みを作る。
- ソフトウェアパターンのできること、できないことをしっかり認識する。

調査結果のまとめ(2)
そのために解決をすべき課題

- **ソフトウェア企業が解決すべき課題**
  - **ソフトウェアのライフサイクルを考えた、料金制度、プロジェクト運営の導入。 ⇒ SEにじっくり考える時間を与える。**
  - **プロジェクトマネージメントとアーキテクト（デザイン専門家）の役割を分離する。 ⇒ デザイン専門家の評価を高める。**
  - **SIベンダの場合に、設計・製造を大幅に協力会社に依存せず、内部で設計をしっかり行う仕組みとする。⇒ デザイン専門家の内部育成**
- **研究者・専門家が解決すべき課題**
  - **ソフトウェアパターンに関する分かりやすい教科書、マニュアルなどの整備**
  - **ビジネス分野における具体的な例題、事例集の整備**