2007－IS－100 (5)
2007／6／5

SOAを実現するコンポーネントベースモデリング

桐越信一　浅井麻衣　浜口弘志　藤井啓詞
株式会社 日立製作所 ソフトウェア事業部

SOA(Service Oriented Architecture)による情報システムを実現していくためには、ビジネスプロセスの設計と、サービスを構成する単位のコンポーネント設計が鍵となる。我々のプロジェクトは、オブジェクト指向型アプリケーションの開発において、国際標準化団体OMG(Object Management Group)の推し進める参照アーキテクチャMDA(Model Driven Architecture)とUML(Unified Modeling Language)をベースに、業務分析の段階からコンポーネントを意識した分析モデルを作成し、これを各種の企業で適用してきた。さらに、その結果を設計・実装モデルにマッピングしてソースコードを生成するまでの開発スタイルを実現した。本論文ではSOAにおけるコンポーネントの位置付けとコンポーネント開発の背景、コンポーネント導出の分析モデル作成における開発プロセス、さらには分析モデルから設計・実装モデルに変換するまでツールの技術動向の実例について報告する。

Report of component base modeling for SOA

Shinichi Kirikoshi　Mai Asai　Hiroshi Hamaguchi　Keiji Fujii
Software Division ,Hitachi, Ltd.

Designing information systems with Service Oriented Architecture (SOA) requires appropriate design of business process and components constituting services. For the development of object-oriented application, analysis-model is created in the development process based on the reference architecture - Model Driven Architecture (MDA) and Unified Modeling Language (UML), which is promoted by Object Management Group (OMG). Our team has created analysis-models in taking notice of components even from business analysis phase, and applied this method to many companies. Moreover, we have built up the development style to generate design / implementation-models and finally the codes from the analysis-model by mapping patterns. This paper shows the meaning of components in SOA, the background of component development, the process to create an analysis-model to lead out components, and an example of the trend of development tools to translate from an analysis-model to the design / implementation-model.

1．SOAでのコンポーネントの位置付け

SOA による情報システムを一言で表現すると、ビジネスプロセスをアプリケーションプログラム(以下 AP と略す)とは別に外出しで定義し、そこから各種サービス単位を呼び出す形態のシステムと言える。

こうすることでビジネスプロセスの変化への即応やサービスの組替え、入替などが柔軟に実現できるというのが SOA で一般的に言われている特徴である。では、SOA システムのサービスとはいったいなんなのか。それは本来 SOA が目的とするサービスに該当する業務処理の単位が正解であろう。しかしながら企業の情報システムは過去の様々な歴史からそれほど都

合よく構築されていないのが現実であり、サービスの単位である業務処理単位がレガシーシステムのインタフェースであるかもしれないし、パッケージそのものかもしれない。または、データベースそのものを指している場合もありうる。しかしながら、それらに接続するための各種のアダプタがあれば、これも一つのサービスとして定義できる。一方、新規にAPを開発する場合、サービスの単位はコンポーネントとして置き換えることが可能となる。では、そのコンポーネントの粒度はどうすべきか。これを図1「SOAにおけるサービスとコンポーネントの関係」に示す。

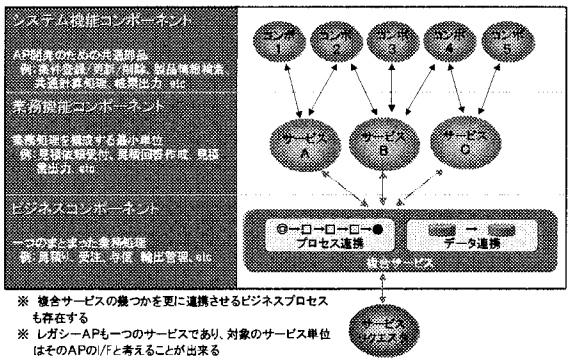


図1　SOAにおけるサービスとコンポーネントの関係

我々のプロジェクトではSOA以前にAPの開発効率向上を目的としてコンポーネントベース開発に取り組んできた。そこで得た結論としては、コンポーネントと一言で言っても各種の粒度があり、それらを事前に整理しておかないとコンポーネントの導出や実装がうまくいかないというものであった。そこで図1に示すように以下の三つの階層でコンポーネントの粒度を定義した。

(1) ビジネスコンポーネント
　受注、見積もりなどのように、一つのまとまった業務処理を指し、業務機能コンポーネントを複合することで成り立つ

(2) 業務機能コンポーネント
　受注受付、見積り回答など、業務処理の要素単位である

(3) システム機能コンポーネント
　情報の登録・更新・参照や認証など、システム寄りの部品群であり、業務機能コンポーネントはこれらを使用して作成される

これらのコンポーネントの階層別にSOAのサービスを当てはめると、(2)の業務機能コンポーネントがSOAでのサービスに該当し、(1)のビジネスコンポーネントはSOAのビジネスプロセスで結合された業務処理と定義できる。

それではこれらのコンポーネントをどうやって導出していくのか、そもそもコンポーネント開発がなぜ必要となったのかなどの背景について次に述べていく。

2．コンポーネントベース開発の背景

2．1　コンポーネント化の背景

前述したようにSOAに対してコンポーネント設計が必要になったわけではなく、それ以前から我々のプロジェクトはコンポーネントベース開発に取り組んできた。

その理由は失敗事例から来ている。

1999～2000年にかけて、筆者がある顧客プロジェクトでのオブジェクト指向開発に参画した際、予定通りシステムは出来上がったものの、開発ベンダーとしての立場からすると次の二つの反省材料があった。

(1) 出来上がったプログラム構造が開発担当者毎にバラバラであり、保守性の問題がある

(2) 開発効率が従来のCOBOLに比べて上がっていない

顧客プロジェクト参画後、自社内のシステム開発への参画機会があり、前述した問題解決に向け、次のような方針を立てて解決策を検証することとした。

(1) プログラム構造の問題についてはMVCフレームワークが出始めたこともあり、これを採用して開発する。

(2) 開発効率の向上についてはコンポーネントを意識し、部品はフレームワークが持っているものは極力使用、それ以外はチームを分けて共通コンポーネントを先行開発し、プロジェクト内で共有していく

この結果を図２「参画プロジェクトでの反省」に示す。

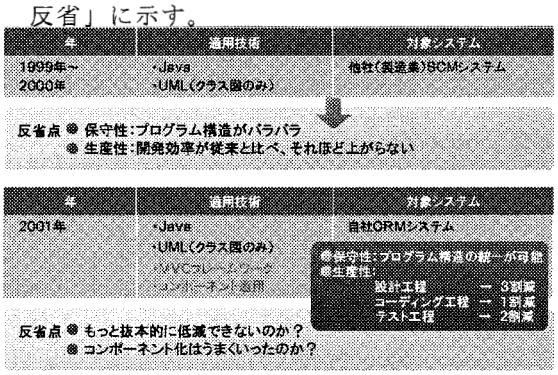


図２ 参画プロジェクトでの反省

図２に示すとおり、プログラム構造の統一については MVC フレームワークの適用により実現できた。一方、開発効率向上を目的とした生産性向上については、コンポーネント化の推進により多少の効果はあったものの、全体的には 1～2 割という程度に留まった。この原因として次のことが反省として挙げられた。

(1) 出来上がった共通コンポーネントは全てシステム寄り機能部品であった
(2) コンポーネントを導出するための開発プロセスを持っていなかった

プロジェクト内で共通コンポーネント化を行う専門チームを立ち上げたにも関わらず、実際にはそのチームがヒヤリングして必要コンポーネントを決定したり、あるいは自身で判断して開発するという、コンポーネント化を行なうチームのスキルに依存した結果となってしまった。

このことからコンポーネントを導出するための開発プロセスが必要であると強く認識し、以降、それらの開発プロセスを検討した。

２．２　開発プロセスの検討

これらを検討していた時期、そもそもコンポーネント設計はうまくいった試しがない、コンポーネント化はうまくいかない、コンポーネント化は永遠の課題だなどという、コンポーネントに対する批判的な意見を各種の雑誌などで目にする機会が多かった。筆者はメインフレーム全盛時期、データベースとネットワーク製品の開発のため、設計部署に一時在席していたことがある。いわゆるシステムプログラミングを経験してきたわけであるが、この当時、コンポーネントではなくモジュールと呼ばれる単位で設計し、自身の担当モジュールは他のモジュールから呼ばれたり、あるいは他のモジュールを呼んで処理したり、さらに共通部品(当時はマクロと呼ばれていた)は事業部内のある部署が開発して提供していた。

これが当たり前の開発スタイルと捉えていたが、次に AP 開発に携わると共通処理などを考えるよりも作ってしまった方が早い、そもそもコンポーネント導出の設計技法が無い、管理体制も無いなどの理由で、自身も共通化という意識から離れてしまった経験がある。これらの反省を踏まえ、コンポーネント化については次の三つの方針を立て検討することとした。

(1) コンポーネント導出の開発プロセスを策定する
(2) コンポーネント化の指針を作る
(3) コンポーネント管理体制を設ける

まず最初に、AP におけるコンポーネントとは、そもそもいったい何なのかを検討した。その結果を図３「業務とアプリケーションの構造」、図４「コンポーネントの粒度」に示す。

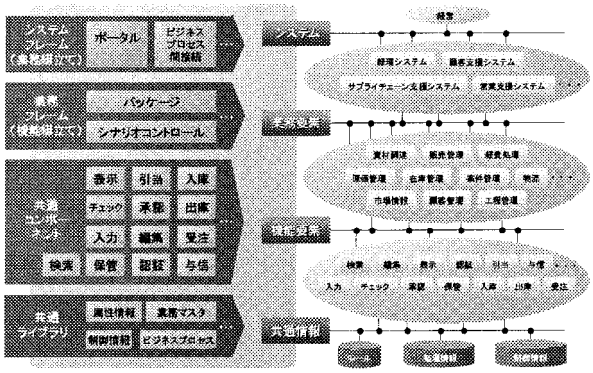


図３ 業務とアプリケーションの構造

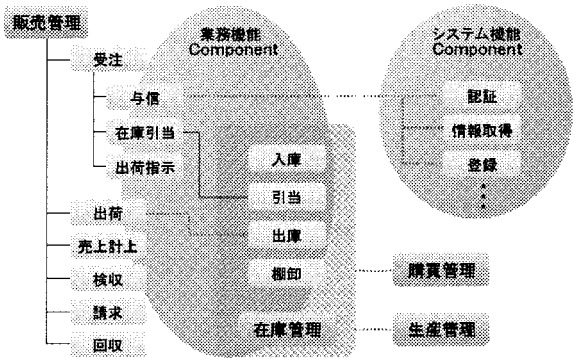


図４　コンポーネントの粒度

図３に示すようにAPはサプライチェーンマネージメントや在庫管理システムなどと呼ばれる場合があるが、それらは呼称であり、業務の実態を考えた場合、要求される機能はもっとプリミティブなものに分解できるはずである。これらの最小単位が機能要素であると捉え、ここをコンポーネント化することを目的とした。次に、その機能要素も引当、出庫などのような業務的な役割のものと、認証、登録などシステム的なものが混ざっているため、これを図４に示すように、業務機能コンポーネントとシステム機能コンポーネントという粒度に分けて考えることとした。さらにシステム機能コンポーネントは業務機能コンポーネントから呼び出されて共有される形態であると位置付けた。こうしてコンポーネント構造を分けて考えることで、それを導出するためのプロセスも分かりやすくなるとともに、結果的には前述したようにSOAでのサービスという位置付けも明確になった。

３．コンポーネントベースモデリング手法

３．１　要素技術の決定とコンポーネントベースモデリング手法

コンポーネントという単位を整理した後、これらのコンポーネントを導出していくための開発プロセスを決定した。この際に取り入れた要素技術がOMGが制定していたMDAとUMLである。

MDAの採用理由は、MDAツールによる生産性の向上よりも、むしろモデルの分離にあった。MDAでは業務分析の視点と設計・実装の視点を分離して、それぞれ独立したモデルを作成することができ、プラットフォームやアーキテクチャが変わっても業務分析モデルが流用可能である。さらに上流の分析フェーズでは業務主管部門の人達でも参画でき、要件定義がある程度行なえるということも採用理由であった。

MDAの分析・設計工程は次に示す３つのフェーズに分けられる。

(1) CIM（Computation-Independent Model、業務分析）
　システムの構造は提示せず、業務の構造やフローを明確にするフェーズ。

(2) PIM（Platform-Independent Model、要求分析・システム分析）
　プラットフォームに依存しないモデルを作成するフェーズ。要求分析ではシステムの外部仕様を、システム分析では内部仕様を定義する。

(3) PSM（Platform-Specific Model、システム設計）
　PIMを特定の技術にマッピングし、プラットフォームやアーキテクチャに依存したモデルを導出するフェーズ。

これらのモデルの大部分はUMLで表記される。次に開発プロセスの制定であるが、当時、各種の書籍などで取り上げられていたUMLを利用した一般的なオブジェクト指向開発のプロセスがあった。これを筆者も一度、経験したことがある。結果はというと、ユースケースからクラス図を導出し、論理設計工程でドメインクラスをまとめてパッケージ化していくものであり、APは出来たもののコンポーネント化という観点では失敗に終わった経験がある。この失敗理由は次の点に集約できる。

(1) 論理設計工程はAPの詳細設計工程であり、この段階で業務単位のコンポーネントは導出できない
(2) 論理設計工程ではそれ以前のモデルを導出した人と担当が異なることがほとんどである
(3) 論理設計工程では膨大なクラス図となっており、これを整理してコンポーネント化を図るのは至難の業である
(4) この工程になると開発工程が切迫

すると納期がちらつき、共通化などの意識が薄れてしまう

これらの問題を解決するため、開発プロセスの策定においては、図５「コンポーネントベースモデリングとは」に示すような手法を採用した。

この手法は Desmond D’Souza 氏、Alan Cameron Wills 氏が提唱したカタリシス手法であり、コンポーネントの切り出しを上流工程で実施し、以降はそのコンポーネント単位に内部分析モデルを見出していくというものである。

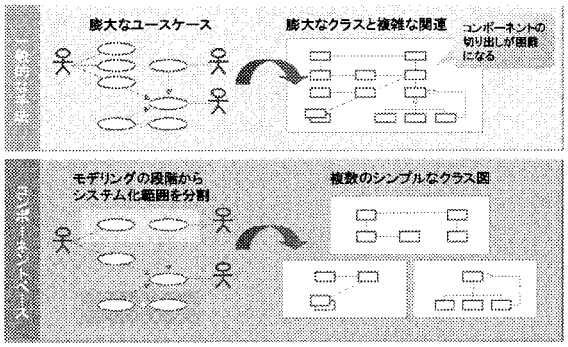


図５　コンポーネントベースモデリングとは

３．２　コンポーネントベースモデリングの開発プロセス

このように開発プロセスの策定においては、MDA ベースで、しかもコンポーネントの導出を前工程で行なうこととし、図６「コンポーネントベース開発プロセス」に示すような開発プロセスを策定した。

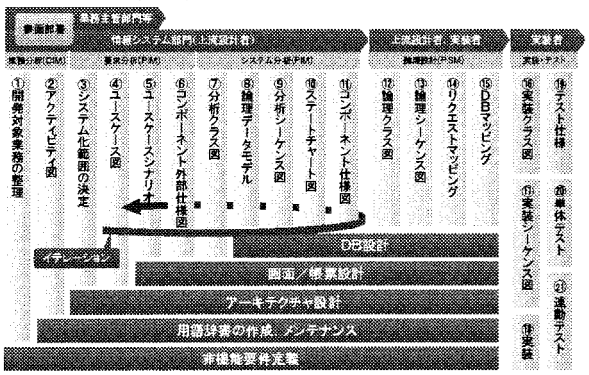


図６　コンポーネントベース開発プロセス

通常、オブジェクト指向開発では論理クラス図の工程でコンポーネント化を図ろうと努力するが、前述したような問題でコンポーネントの切り出しが困難になる。本開発プロセスではユースケース図とユースケースシナリオの段階でコンポーネントを導出し、以降の工程ではそれをリファインし、さらに内部分析モデルを作成して行く形態となる。なお、オブジェクト指向開発ではウォーターフォール型ではなく、イテレーティブ型の開発スタイルが一般的に唱えられている。図６の開発プロセスにもイテレーティブ型として記述しているが、現実は必ずしもそうではない。小規模開発ならイテレーティブ型も可能であるが、大規模開発ではいつまでたっても仕様が固まらないなどの弊害が発生する。本開発プロセスでは MDA の各フェーズである CIM,PIM,PSM 内でのイテレーションは認めるが、承認されたらウォーターフォールで次の工程に落とすというスタイルをとっている。ただし、コンポーネント化の比率がかなり高くなっている企業では、組み立て型の開発スタイルになるはずであり、そうなれば理想としてのイテレーティブ型が望ましい。

また、CIM,PIM の途中までは業務主管部門の参画を要している。基本的にはこの工程は要件定義フェーズに該当するため、本来は業務主管部門の責務である。さらに開発プロセスでは UML 表記のドキュメント導出工程は縦軸として表現しており、それ以外の画面設計やデータベース設計は横軸で表しているが、横軸の先頭は開始時期の目安を示している。

このように開発プロセスを制定したが、これを厳格に遵守するため、次のような指針を設けた。

(1) コンポーネントの使用の有無が分かるよう、ユースケース図でのコンポーネント流用の表記

(2) 最終的なコンポーネント関連が分かるドキュメントの作成

実際に開発プロセスが制定されていても、設計する人がそのコンポーネントを使用しなければ意味のないものになってし

まうため、図７「共通処理機能の事前見出し」に示すように、中間レビューでコンポーネント適用有無をチェックできるようドキュメント作成での指針を設定した。

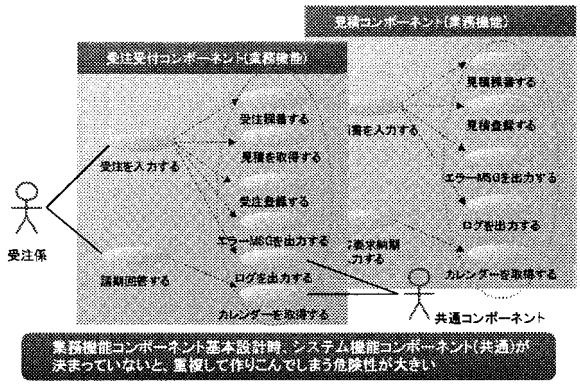


図７　共通処理機能の事前見出し

本指針は基本設計工程(PIM)時点でユースケース図を作成する際、コンポーネントの候補の見出しや既存コンポーネントを適用しているかをチェックするためのものである。この指針は実は我々の失敗から来ている。図７に示すシステム開発を実施したのであるが、受注処理と見積もり処理のプロジェクトチームが異なっていたため、まったく同じ処理をそれぞれに作りこまれてしまったことが元となっている。

そもそも、システム機能部品については、それが既存には無くても、ある程度共通的に使うことが判断しやすいため、前もって各チームに提示しておくなどの措置が必要となる。それをドキュメントの上で使用するとして作成してもらい、実装での重複を避けることが重要である。

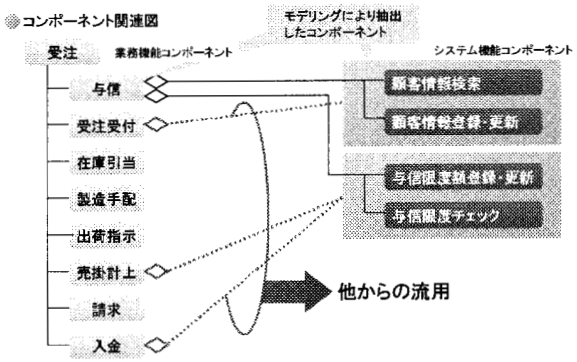
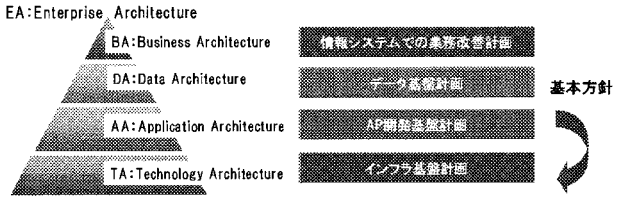


図８　コンポーネント仕様図

次に全体的なコンポーネント構成がどうなるのかを表すドキュメントの作成も指針として設けている。図８に「コンポーネント仕様図」を示す。

この図も基本設計工程(PIM)終了時での提出を指針としている。これによりコンポーネントの流用が把握できるとともに、全体の関連が明確になる。また、表形式での表現も可能としており、この場合は基本設計工程のシナリオステップ数を入れることで新規開発の全体シナリオステップ数、流用するコンポーネントのステップ数が明確になり、これらはシステム開発全体の規模を見積もる際に使用できる。

このようにして開発プロセスと指針を制定したが、これらがうまく運用できなければ意味が無い。コンポーネントは開発するよりも、むしろそれをうまく使う組織や管理体制の方が重要になってくる。図９に「プロジェクト推進体制の事例」を示す。



チーム名称	役　割
標準化チーム	標準開発プロセスや各種基準、ドキュメントの標準化、各種アーキテクチャ標準や共通処理方式方針などを決定する
共通コンポーネントチーム	特にシステム機能コンポーネントの提供と保守、管理、APチームへのデザインレビューへの参画
共通マスタチーム	共通マスタの設計とデータアクセスコンポーネントの提供
運用管理チーム	インフラ環境の構築と各種APの運用管理方式決定
AP開発チーム(複数)	各種APの開発

図９　プロジェクト推進体制の事例

図９は我々のプロジェクトが参画したある企業での推進体制の事例である。標準化チームは開発プロセスの制定も含め、アーキテクチャ全体の決定や処理方式の標準化を制定するチームである。共通コンポーネントチームはシステム機能コンポーネントの開発も行なうが、それを管理して使用させ、指針に沿って設計されているかなどのチェックの責務を負う。また、マスタチームは共通的に使われる DAO(Data Access Object)を開発するとともにデータ

ベース設計を行なう。企業においては標準化チームと共通コンポーネントチームを一緒にしている場合もある。

特にこの中で共通コンポーネントチームは各業務 AP 開発チームのデザインレビューに参画し、コンポーネント適用の有無の把握や指針通りに設計されているか否かの判断、チームとして開発すべきコンポーネントの見出しなども行なう。

この指針は企業により異なるが、前述したコンポーネント仕様書でのステップ数がコンポーネント流用率 50%以上でないと次の工程に進ませないという管理をとっている企業もある。50%という数字については驚くことは無く、システム機能コンポーネントの流用率がこの企業での実測値で 70～80%まで上がっている。これに業務機能コンポーネントを加味した全体の流用率が 50%以上というのは必ずしも無理な指針ではない。

4．コンポーネントベースモデリングの技術動向

このようにして開発プロセスに応じてそれぞれのモデルを策定していくことになるが、ここに一つの障壁がある。ＭＤＡやコンポーネントベース開発に特化したことではなく、一般的にオブジェクト指向開発を行なう上で散見される問題であるが、図 10「プロジェクトが抱える問題」に示すように、分析モデルと設計モデルの橋渡しを行なえるアーキテクト人材が世の中に不足していることである。

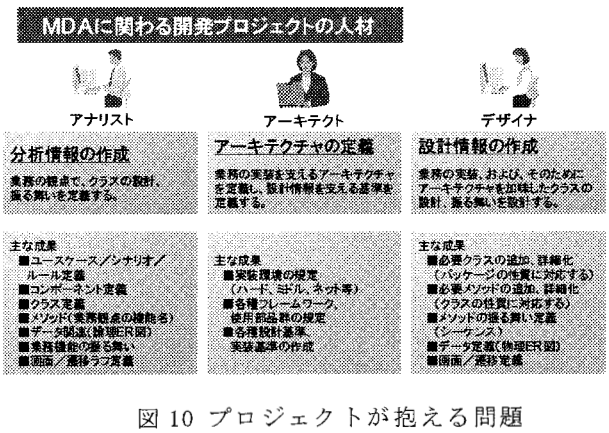


図 10 プロジェクトが抱える問題

これらを行なえる人は業務処理もさることながら、ＭＶＣフレームワークやプログラムデザインパターン、各種アーキテクチャなどのスキルを要している必要があり、なかなかこういう人材が揃わないのが実情である。プロジェクトとしては、こういった数少ないスキルを持った人がアーキテクチャ設計書や処理方式設計書などを作成し、上流のモデルを実装モデルへ落としていくためのノウハウを教えている形態が多い。しかしながら、全てのデザイナーが平準的に実装できるかというと、なかなかうまくいかないケースもある。そこで、この部分を支援するための開発支援について、我々のプロジェクトと製品開発部署、さらに弊社の研究所での共同研究で図 11「モデルのパターン適用」に示すようなＭＤＡツールを開発した。

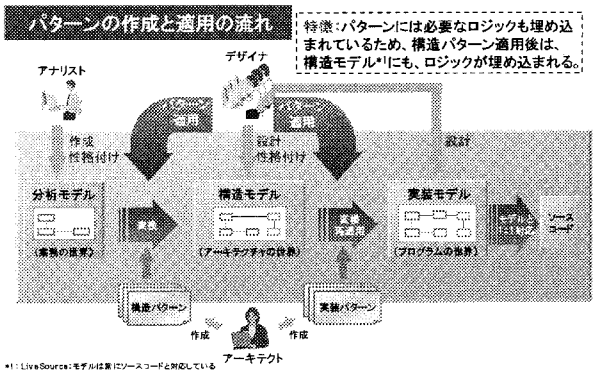


図 11 モデルのパターン適用

本ツールは、アーキテクトの人が頭の中で描いているパターンを外出しにし、それをテンプレートとして入力し、ナビゲーション方式で次工程のドキュメントを自動生成するＭＤＡツールである。

当初、分析モデルを一気に実装モデルまで変換することを考えたが、分析モデルをいきなり Java などの J2EE デザインパターンに変換するには無理があり、中間に構造モデルと呼ばれる業務処理パターンを設け、変換を二段階としているという特徴を持つ。この発想はプログラムにデザインパターンがあるのなら、業務処理にもパターンがあるのではということからであった。

本ツールの適用としては、アーキテクト

する人がパターンを準備して登録し、それをデザイナーが使用するというもので、これにより生産性向上と開発の平準化が図れることになる。このようなＭＤＡツールは既に世の中にいくつか出ているが、パターンを準備して変換するという特徴を持つものは他に無い。

また、これらのツールはさらに上流工程に波及し、現在、ユースケースシナリオを元にして、そこからクラス図やシーケンス図を生成するところまで既に研究成果を見出している。

５．まとめ

今回報告したコンポーネントベースでの開発プロセスと開発方法論の策定について、すでに 20 数社の実際のプロジェクトでこの手法を適用中であり、コンポーネントベースでの開発が行われている。実際の分析工程では、新規開発システムの中で導出したコンポーネントの 70%以上がプロジェクト内で流用できるという結果を見出している企業もある。

さらに、開発支援も実用化ができる段階まで到達してきており、今後、コンポーネントベース開発は、SOA などのシステム構築方法とあいまって、より一層拍車がかかるものと推察する。今後は今までの体系を元に、SOA でのビジネスプロセスモデリングなどとの連携や、インタフェース設計技法などを確立していく所存である。

参考文献

(1) David S.Frankel、”MDA モデル駆動型アーキテクチャ”、日本アイ・ビー・エム株式会社 TEC-JMDA 分科会、(株)エスアイビー・アクセス、2003.

(2) (株)テクノロジックアート、”独習 UML”、 (株)翔永社、2005.

(3)長瀬嘉秀、橋本大輔、”UML システム設計実践技大全”、(株)ナツメ社、2004.

(4)湯浦克彦、大坪稔房、団野博文、石井義明、古澤憲一、桐越信一、鈴木文音、”EJB コンポーネントによる Web システム構築技法”、(株)ソフト・リサーチ・センター、2002.

(8)Anneke Kleppe、Jos Warmer、Wim Bast、”MDA モデル駆動型アーキテクチャ導入ガイド”、(株)インプレス、2003.

(9)(株)テクノロジックアート、”パターンモデリングガイド”、(株)ピアソン・エデュケーション、2002.

(10)Ivar Jacobson、Grady Booch、James Rumbaugh、”UML による統一ソフトウェア開発プロセス”、(株)翔永社、2000.

(11)現場の UML　(株)ソーテック社
浅井麻衣他　2006.

商標