効果的なモデリング開発方法

－MDA 開発方法の確立－

アブストラクト

1. 背景と課題

近年の情報システム構築において、短期開発、オープン化、分散化、再構築、新技術、機能追加や仕様変更への迅速化など、様々な課題がある。課題を解決する為には生産性の効率化が必要である。それを妨げる要因として以下３項目をメンバー共通の課題として取り上げ、解決へアプローチした。

１）要件から実装へつなげる技術不足
　要件定義する上流作業者から設計・実装する下流作業者へ情報欠落による工程手戻り発生。

２）マルチプラットフォーム環境対応
　同一機能を異なる OS／言語での再開発、新ハード／ミドルウェア対応に要件仕様再洗い出し。

３）機能追加・仕様変更に対する保守性
　ソースコードとドキュメントの不一致（反映漏れ）による保守生産性の低下。

2. 研究の目的と手順

課題を解決する一つの方法として、MDA（Model Driven Architecture）に注目した。

MDA は次の３つのレイヤーから構成される。CIM（Computation-Independent Model:システムでの実現性を意識しないで、業務を表したモデル)、PIM（Platform-Independent Model:プラットフォーム非依存モデル)、PSM（Platform-Specific Model:プラットフォーム依存モデル）である。しかし概念として MDA は普及しつつあるが、CIM/PIM/PSM の記述方法と変換ルールが決まっておらず、ツールに関しては十分な機能を満たす製品はない。特にソースコード生成に関しては明確な方法論が無い。

現在の MDA 技術要素での不足事項を補い、以下を目的として研究した。

１）システム開発の要件定義から実装まで、要求仕様を漏れなく正しく反映。

２）マルチプラットフォームに対応。プラットフォーム間の可搬性を実現。

３）要件と実装の対応関係（トレーサビリティ）を明確にし、機能追加や仕様変更の保守性を向上。

目的を実現するための研究手順として、MDA の基礎技術で OMG ™(Object Management Group)が提唱する方法論を用いることにした。しかし、その方法論でも PIM/PSM という概念は存在するが，上流からの連続的な変換方法、及びソースコード生成については何も言及していない。そこで、この方法論をベースにより効率的且つ上流から下流までのモデル変換を目指すための MDA 技術を考案した。実際にサンプルシステムに適用しガイドラインとしてまとめ、別サンプルシステムに適用して検証した。

3. 方法論の確立

モデルの記述方法と各モデル間の変換方法を決めて、それをガイドラインとしてまとめた。モデルの関係は、右図（図１）のように考案した。それを検証するための対象言語は、オブジェクト指向言語の C#／JAVA、非オブジェクト指向言語の COBOL とした。

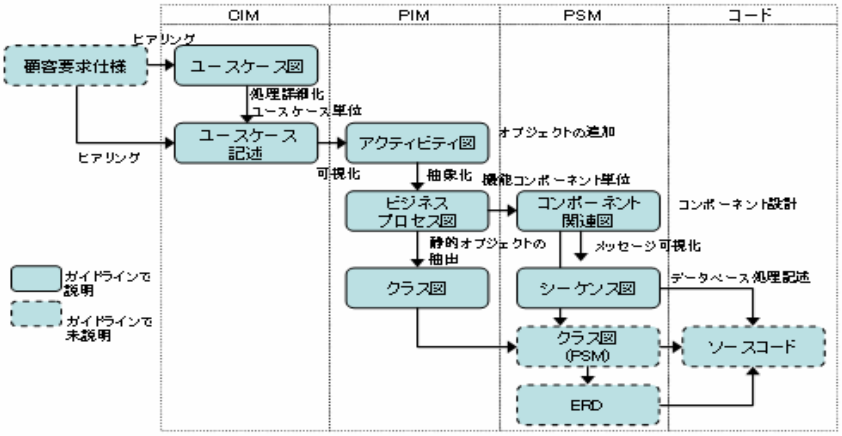

図１　ドキュメント変換図

1）ユースケース図／ユースケース記述　【CIM】
システムの要件定義を取りまとめ、ユーザからの業務ヒアリング結果から業務全体に関係する事項を分析し、開発の範囲を明確にする。ユースケース記述は自然言語で業務の流れを詳細に記述する。記述方法には一定のルールを設けた。今後の設計のベースとなる。

2）アクティビティ図　【PIM】
ユースケース記述を図にしてビジネスプロセスモデルで表現することで、処理の流れを可視化する。またシステム化する業務プロセスの分析を行い、関係するオブジェクトを明確にする。システム化対象を明確にする為、ステレオタイプに「人」「システム」にて識別するようにした。

3）ビジネスプロセス図　【PIM】
アクティビティ図で明確になった業務プロセスを、システム化する部分の実装単位を明確にし、コンポーネント（部品）を洗い出す。またオブジェクトの入出力関係欠落を防ぐ為、方向性をステレオタイプで表現するようにした。

4）クラス図　【PIM】
ビジネスプロセス図より、属性定義する対象クラスを特定し、プラットフォームに依存しない不変要素のみ属性定義する。

5）コンポーネント関連図　【PSM】
ビジネスプロセス図の機能コンポーネント、データコンポーネントをプラットフォーム固有のコンポーネント（部品）に変換し、コンポーネント間のインターフェースを洗い出す。COBOL ではフラットなプログラム構造上、データコンポーネントを直接データテーブルとして扱った。

6）シーケンス図　【PSM】
ソースコード生成する為の情報、すなわち、データテーブルの SQL 呼び出し、事前・事後条件、条件分岐/繰り返しの詳細を、漏れなく記述する。ソースコード実装への詳細設計書となる。

7）クラス図【PSM】
PIM のクラス図を元に実装レベルのクラス図を作成する。ERD 作成の元となる。

8）ソースコード変換
シーケンス図とクラス図を元に、C#,JAVA,COBOL それぞれのソースコードへの変換方法を考案した。その変換方法に基づきそれぞれのソースコードを実装した。

4. ガイドラインに沿った開発方法論の評価

MDA 概念から開発方法論全体を『自動変換可能か』を基準に、各モデル間の変換方法についてサンプルシステムにより検証した。人が補助入力をする必要があるが、ユースケース図からソースコードまでの変換ルールが明確になり、MDA 的概念に沿った開発方法が確立できたと評価する。

5. 結論

1）要件から実装までの連続性
CIM モデルから PSM モデルへのドキュメント連続性が保持され、要求仕様を PSM に反映する際の漏れを削減し、要求定義を実装に正確に反映することができた。

2）マルチプラットフォーム対応
１つの PIM を JAVA/C#/COBOL で実装したことにより、全体モデルを PIM で作成し、それぞれの言語で PSM に変換できた。特に、非オブジェクト指向言語の COBOL に適用し、プラットフォーム独立のモデルを検証できた。

3）機能追加、仕様変更に対する保守性向上
機能追加・仕様変更の場合、モデルを修正する為、変更点・追加点さらにその影響範囲を把握し易い。また、モデルで仕様を表現する為、修正内容を開発メンバーやユーザと共有し易い。

今回考案した MDA 開発方法論により、課題を解決する為の目的を達成したと考える。作成したガイドラインを有効活用して頂き、ぜひ、実プロジェクトに導入して頂きたい。

現在はモデル間を変換するツールが存在しない為、人による変換で作成するしかない。しかし、近い将来自動変換ツールが実現すればより効率的な開発が期待でき、効果的なモデリング開発方法が確立されることを望む。