ＵＭＬの活用による大規模アプリケーション設計

－ＵＭＬは魔法の杖なのか？－

アブストラクト

１．問題認識

昨今、インターネットの普及により、大規模システム開発においてもＷｅｂシステムを採用するケースが増えている。ユーザ企業は、他社との競争において優位性を確保するため、大規模Ｗｅｂシステムを用いた新規のサービスを、短期間で完成させることを要求している。開発現場では、この要求に応えようとするあまり、以下のような問題がますます顕在化してきている。

(1) 要求を漏れなく把握することができない。また、要求を効率良くドキュメント化することができない。
(2) 要求仕様を次工程へ確実に落とし込めず、実装レベルになって漏れが発覚している。
(3) 仕様変更時に影響範囲の見極めをドキュメントから行うことができない。
(4) オブジェクト指向開発において、設計者のスキルによって、ドキュメントの完成度が左右されている。

これらの問題を解決する手段の１つが、ＵＭＬだと一般的に考えられており、ＵＭＬに対して以下のように、あたかも「魔法の杖」であるかのような期待がされている。

(1) ＵＭＬのユースケースを利用すれば、ユーザの要求を正しく表現できる。
(2) ＵＭＬとＪＡＶＡは、親和性が高いので、要求を実装まで確実に繋げやすい。
(3) ＵＭＬは、要求から実装までをカバーしているので、影響範囲が把握できる。
(4) ＵＭＬは、オブジェクト指向開発をサポートする統一されたモデリング手法であり、誰でも一定の完成度のドキュメントを作成できる。

しかしながら、実際は、ＵＭＬを利用しても問題は起こり得る。なぜなら、ＵＭＬは、表記法のみを規定しているのであって、「開発のどの段階で、どのドキュメントを利用するのか」は規定していない。また、ＵＭＬは、実践に裏付けされたモデリング手法のノウハウが蓄積されていないため、これらの問題を解決できず、ＵＭＬを使用するメリットを感じ取れないのが実情である。

２．研究目的とアプローチ

当分科会では、以下の４つの事柄を解決しなければならないと考えた。

(1) ユーザの要求を漏れなくドキュメントに表現できる。
(2) 要求仕様が、漏れなく正確に実装に反映される。
(3) 仕様変更が起こったときにその影響範囲をドキュメントから把握できる。
(4) モデリングの習熟度に関係なく、ある一定のレベルのドキュメントを作成できる。

これらを実現するため、以下の研究を行った。

(1) ＵＭＬのみにこだわらず、要求から実装までが有機的に結びついたドキュメント体系を作成する。
(2) ＵＭＬを使用し、誰でも一定の完成度のドキュメントを作成できるガイドラインを導き出す。
(3) 実際の業務で使用された要求仕様書を基に、ドキュメント体系とガイドラインを検証する。

３．研究成果

工程を「要求」「分析」「設計」に分け、各ドキュメント間の関係を表したドキュメント体系図と各ドキュメントのガイドラインを策定した（表 1)。そのガイドラインの特徴は、以下の通りである。

(1) ユースケースを容易に抽出するため、アクティビティ図を業務概要フローとして採用
(2) 段階的詳細化手法によるユースケースの粒度の統一
(3) Ｔ字型ＥＲ図によるデータ指向とオブジェクト指向の融合

(4) クラス抽出するための思考過程をロバストネス図で表現

表１　ガイドラインとドキュメント体系

No	工程	ドキュメント名（＊は、ガイドラインを作成）	ドキュメント体系上の位置付け
1	要求	用語集	UML 以外だが必須
2		Ｔ字型ＥＲ図＊	UML 以外だが必須
3		アクティビティ図（業務概要フロー）＊	必須
4		ユースケース（図、記述、ビジネスルール）＊	必須
5	分析	画面遷移図・レイアウト	UML 以外だが必須
6		ロバストネス図＊	UML 以外だが必須
7		分析シーケンス図＊	必須
8		分析クラス図＊	必須
9	設計	画面一覧、画面遷移図、画面レイアウト、画面表示仕様、画面チェック仕様、データ操作仕様	UML 以外だが必須
10		設計シーケンス図	必須
11		設計クラス図＊	必須

これらの特徴を持ったガイドラインと各ドキュメント間を関係付けるドキュメント体系を規定した結果、以下の成果を確認できた。

(1) 効率的で漏れの無い要求仕様の表現

当ガイドラインで規定した要求工程のドキュメントにより、ユーザ役を務めたメンバに、自分の伝えたかった要求仕様を、お互いの共通認識の元、表現されたことを確認できた。

これを可能にしたのは、「業務概要フローとしてのアクティビティ図の採用」「段階的詳細化手法によるユースケースの導入」「Ｔ字型ＥＲ図の採用」である。

(2) 要求から設計まで繋がるドキュメント体系の策定

ロバストネス図の導入により、要求工程で作成したドキュメントから設計工程のクラス図まで、その要求仕様が漏れることなく、落とし込めることを確認できた。

(3) ユースケース駆動によるドキュメントへの保守性の確認

ドキュメント体系は、要求工程で作成したユースケース単位に、以降のドキュメントを作成していくようになっている。(ユースケース駆動のドキュメント体系である。)

分析工程を１サイクルまわした段階で、仕様変更を発生させたところ、影響範囲が見極められ、確実に関連ドキュメントに反映できることを検証できた。

(4) ガイドラインの導入容易性と設計作業の効率性

ドキュメント体系を策定するにあたっては、ＵＭＬに固執せず、要求仕様を確実に整理でき、かつ要求～設計工程まで確実に落とし込むために最低限必要なものが何であるかを導き出した。また、その中でＵＭＬそのものについても全てを採用せず、取捨選択を行った。さらに、ＵＭＬ以外のドキュメントも必要であることを明確にした上でガイドラインの作成を行っており、効率的な指針を示している。

各ガイドラインについては、作成手順を明確にしており、その表記方法の説明に加え、チュートリアルも充実しており、ＵＭＬの理解をサポートしている。

４．結論

ＵＭＬは、単なる表記法に過ぎない。また、ＵＭＬだけではシステム開発の全工程をカバーすることはできない。今回の分科会でドキュメント体系・ガイドラインを策定することによって、どの工程で何を行えばよいのか、明確にすることができた。ＵＭＬを知らない人は、ガイドラインの簡単な表記法の説明とチュートリアルによって、要求がどう設計に反映しているかが理解できる。これらを使うことで要求を実装まで漏れなく繋げることができ、要求変更による影響範囲を把握し、Ｗｅｂシステムの開発をスムーズ行うことができる。是非、システム開発で、このガイドラインを活用していただければ幸いである。