オフショア開発向けＵＭＬ適用ガイドライン
Ver ３．０

2010 年 8 月

特定非営利活動法人　UMLモデリング推進協議会

オフショアソフトウェア開発部会

目次

１．目的

（１）（オフショア開発全体における）本ガイドラインの位置付け

オフショアでのシステム開発においても、オフショアではない国内開発と同様、要求/設計/構築/テストの開発サイクル各工程に関するテクノロジー視点での開発方法論、プロセスやコストに関するプロジェクト管理手法、成果物の品質管理手法などが利用される。

本ガイドラインは、その中でも”UML を用いたオフショアでのソフトウェア開発”に焦点をあて、オフショア開発特有の、かつ、UMLを用いた開発に共通に見られる課題点へのHints & Tipsを、実践経験に基づきまとめたものである。

オフショア開発全体における本ガイドラインの位置付けを次に説明する。

オフショア開発全体には、以下の要素が必要である。

①開発方法論（開発プロセス、開発手順、開発技法）

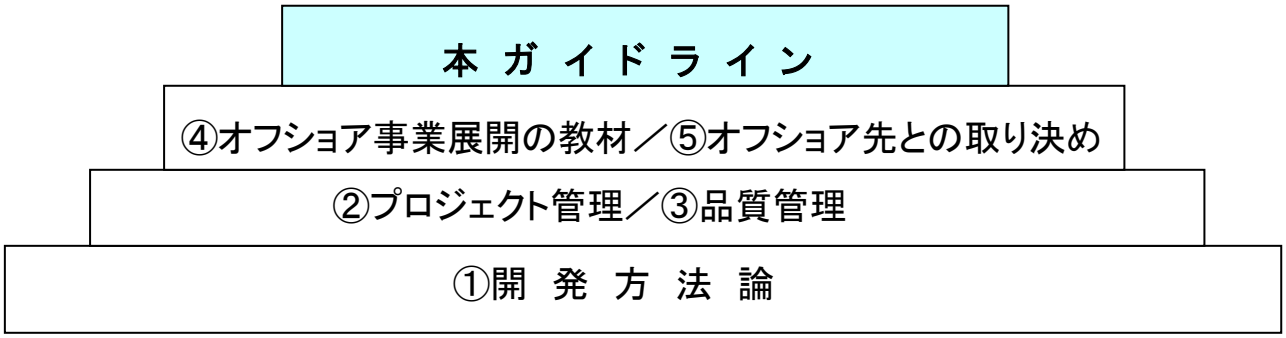
②プロジェクト管理（プロジェクトマネジメント計画書などに基づいた管理）

③品質管理（品質マネジメント計画書などに基づいた管理）

④オフショア事業展開の教材（オフショア事業展開事例集（市販本）、オフショア開発メルマガ、各社ノウハウなど）

⑤オフショア先との取り決め（オフショア先との作業範囲、スケジュール、納入物、価格などを決めたもの）

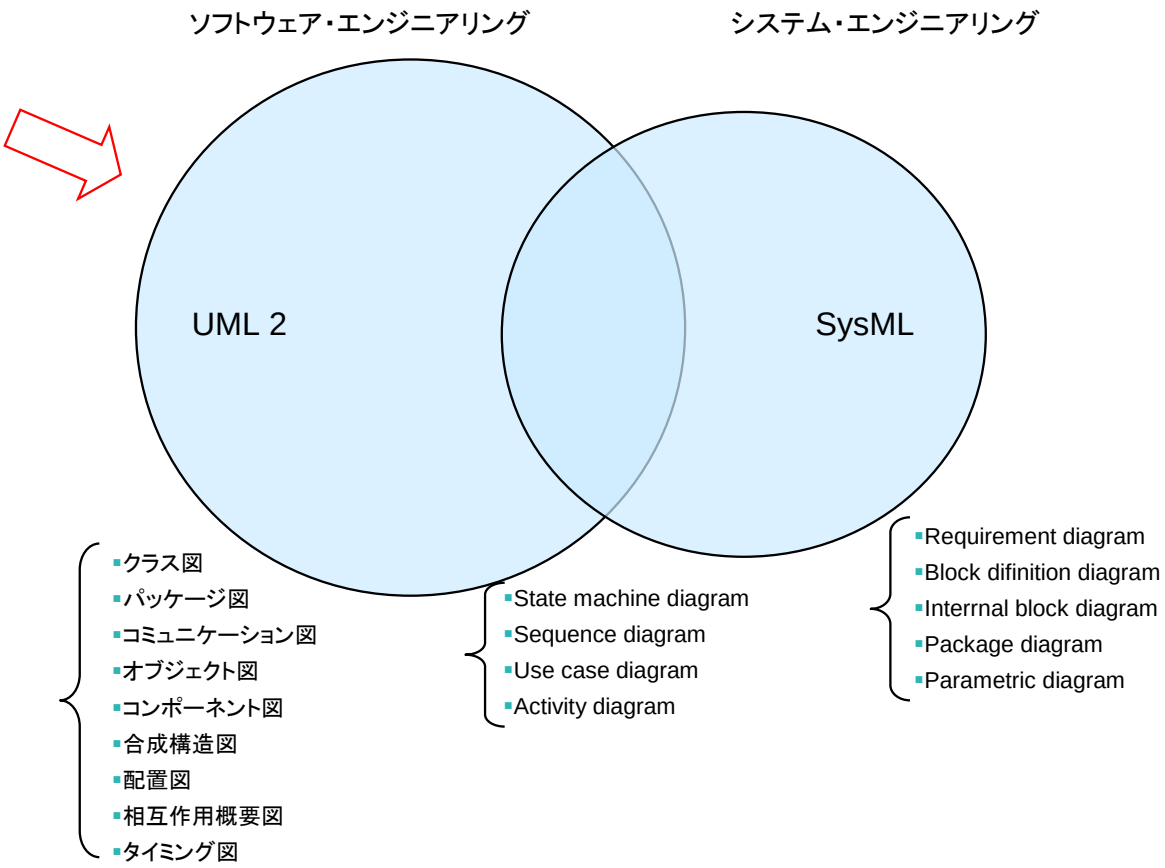
システム開発を行うには、上記の①～③を必要とする。オフショア開発では、さらに④⑤を必要とする。①～⑤は、オフショア開発に先立ち、各社が準備する必要がある。本ガイドラインは、モデリングを中心とした共通的な開発方法、開発手順、成果物を説明したものであり、①～⑤を補完するもの（ノウハウ）である。既に①～⑤が準備されていることを前提としており、それらに付加して使用することにより、オフショア開発の効率向上を目的としたものである。したがって、各社が決めるべき作業内容、作業手順などには言及していない。



（２）本ガイドラインでの対象

本ガイドラインでは、以下の範囲をガイド対象としている。

まず、「システム開発」という広義の開発を考えてみると、UML2 などを主に用いるソフトウェア・エンジニアリングと、SysML なども用いるシステム・エンジニアリングに分けられる。本ガイドラインでは、このうち、ソフトウェア・エンジニアリングに焦点をあてて検討した成果を記している。



次に、そのソフトウェア・エンジニアリングを詳細に見てみる。

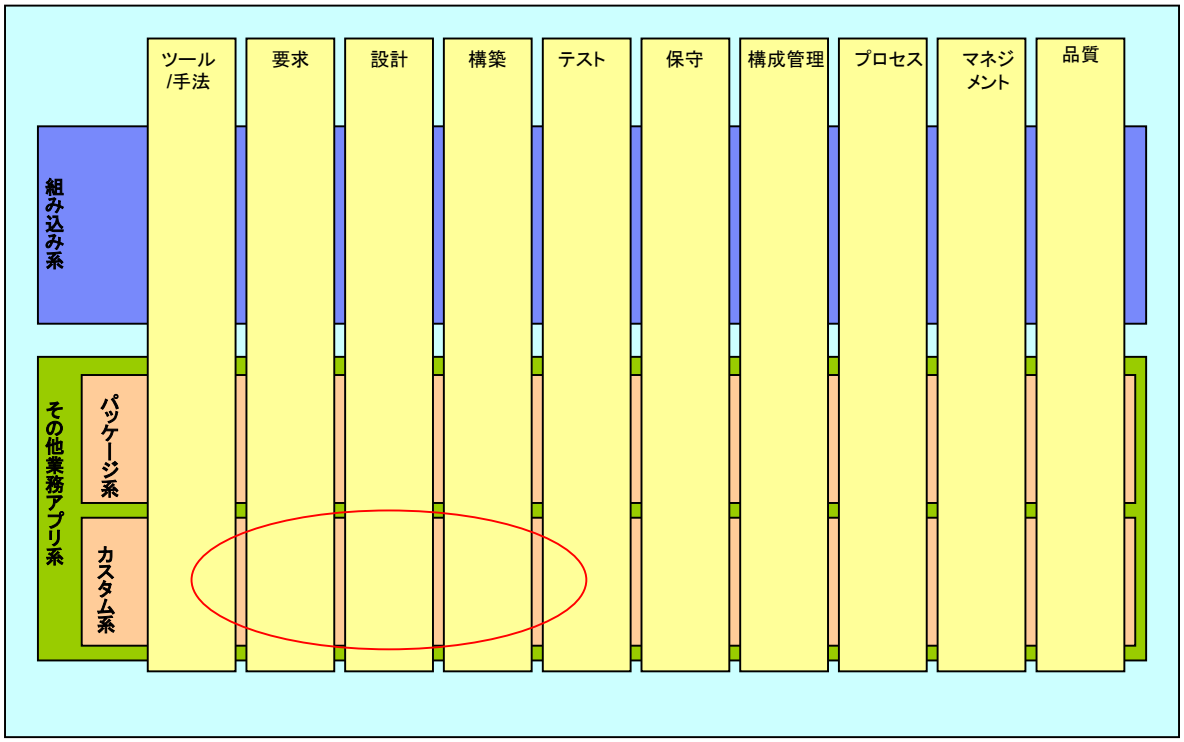
ソフトウェア・エンジニアリングは、大きく

- 要求～設計～構築～テスト～保守の、開発サイクルの各工程での開発技術の要素
- ツールや開発方法論、構成管理方法といった成果物作成支援の要素
- 開発のプロセスやコストに関するプロジェクト管理の要素
- 仕様書、プログラムなど成果物に関する品質管理の要素

に分けられる。

一方、そのエンジニアリング対象となるソフトウェアは、1) 組み込みを前提としたソフトウェアと、2)業務アプリケーション用ソフトウェアとに分けられる。 さらにその業務アプリケーションは、2a) ソフトウェア製品をパッケージとして利用し、その製品仕様のカスタマイズ範囲内で設計・構築を行うケースと、2b) 適用パッケージを考えにくく、自由度は大きいが、新規設計・構築部分が多くなるカスタム開発を行うケースとがある。

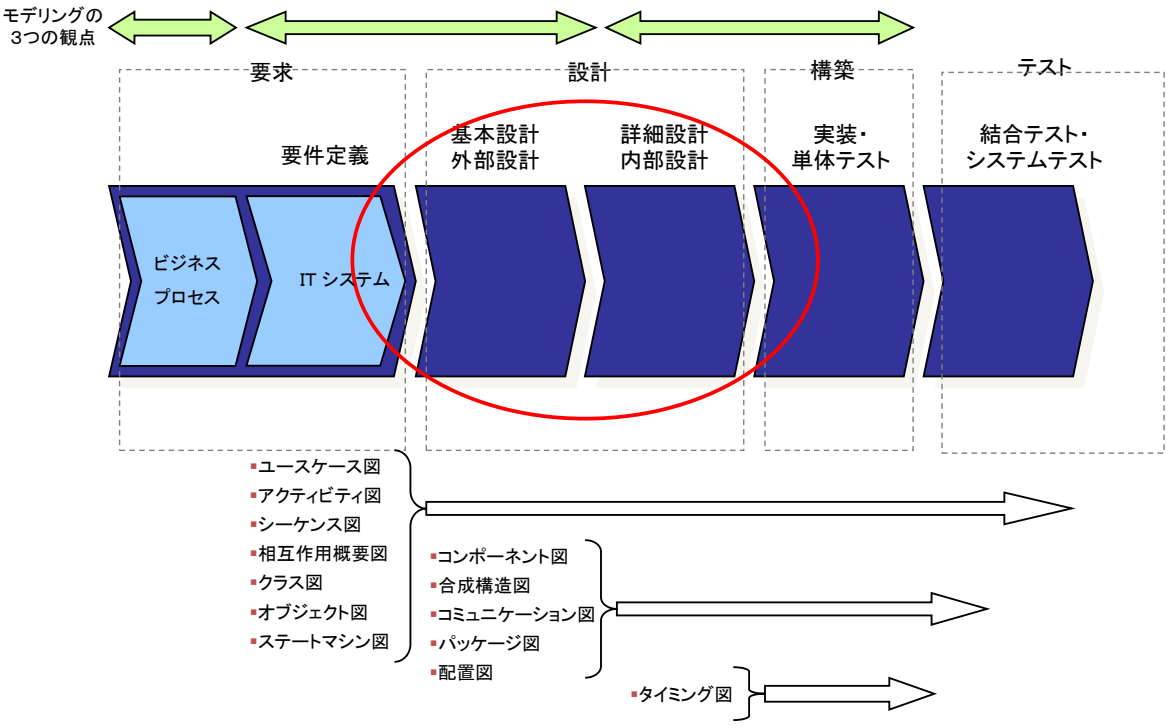
本ガイドラインは、この中で、開発サイクルの各工程での開発技術要素、それもパッケージではなく、カスタムの業務アプリケーション・ソフトウェア開発に焦点をあてて、Hints & tips とするものである。



さらに開発サイクルの各工程を、より詳細に見てみる。

開発サイクルは、要求工程の後半の要件定義から、設計を経て、構築、テストへと続いていく。 これを”UML”という切り口から見ると、ビジネスプロセスを実現する IT システムの要件定義を行う工程から、プログラミングを始める構築工程までが、UML を利用する局面の対象であると考えることができる。 具体的には、要件を UML により記述する、設計による仕様を UML により記述する、その仕様を UML によりオフショアメンバに伝え、UML に基づき実装を行う、といった場面になる。

本ガイドラインは、要件定義や外部、内部設計の各工程で UML により仕様記述を行う部分に着目することになるが、モデリングの観点からすると、仕様モデリングや実装モデリングが該当の範囲となる。



なお、ここでは要件定義、外部設計、内部設計、実装という名称を用いたものの、後述の本ガイドライン内では、

- 業務分析
- 要求分析
- システム分析
- アーキテクチャ設計
- 詳細設計
- 実装、単体テスト
- 結合テスト

の分け方・用語を採用しているので、注意されたい。

最後に、前述の UML 関連の工程において、オフショア開発の視点でどのように整理されるかについて触れる。

本ガイドラインにおいて、オフショアは、外部設計（アーキテクチャ設計）完了後、内部設計（詳細設計）から実装を担当することを想定してまとめている。 開発者の視点では、

「オフショアのメンバに渡す仕様は、UML を用いて何をどう作成すればよいだろうか？」

「オフショア開発特有に気をつけるべき点は何になるのだろうか？」

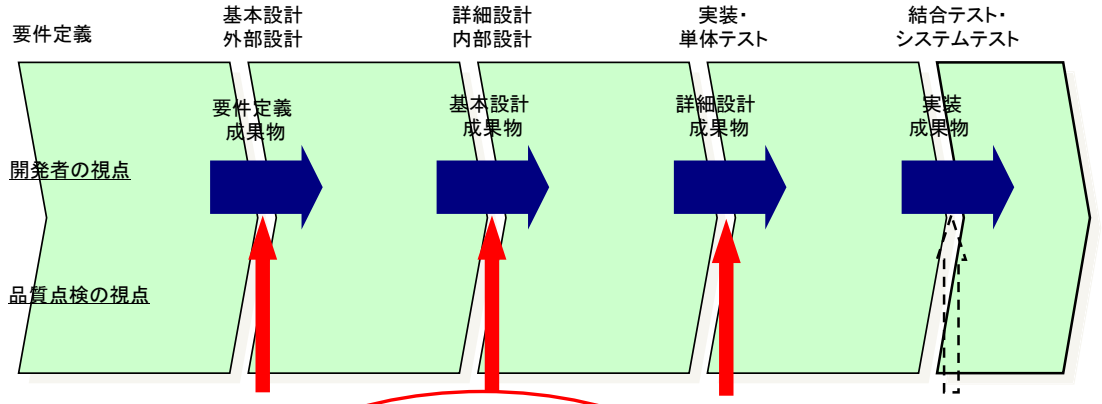
があるし、一方、そのオフショア成果の品質を点検する視点では、

「オフショア開発用に、どのような点に気をつけて点検するのがよいだろうか？」

「オフショア開発用には何が満たされていればよいと考えればよいだろうか？」

という関心事項が挙げられよう。

本ガイドラインは、特にこの点に着目したノウハウを示すものである。



		要件定義 → 基本設計	基本設計 → 詳細設計	詳細設計 → 実装	実装 → 結合テスト・システムテスト
開発者の視点	日本側	どのように利用・作成すればよいのか	どのようなものを渡せばよいのか	－	－
	オフショア側	－	どのようなものを受取りたいのか	－	－
品質点検の視点	日本側	どのような内容を点検すればよいのか	どのような内容・品質レベルを達成すればよいのか	どのような内容を点検すればよいのか	－
	オフショア側	－	どのようなものを受取りたいのか	どのような内容を点検すればよいのか	－

以上、本ガイドラインで、どの範囲を対象としているのかを示した。

オフショア開発において典型的なパターンである「オフショアでの内部設計～構築」を想定し、日本側での要件・設計のまとめと、それらのオフショアメンバへの伝達、オフショア側・日本側双方でのオフショア開発成果の点検、に関してが本ガイドラインの対象である。

補足として、やや異なる視点での対象範囲を示してみる。

要求を分析する、それに対応するシステムを設計する、仕様をオフショアに伝える、伝達結果に基づいて作成し、その成果の品質を点検する、という開発過程において、

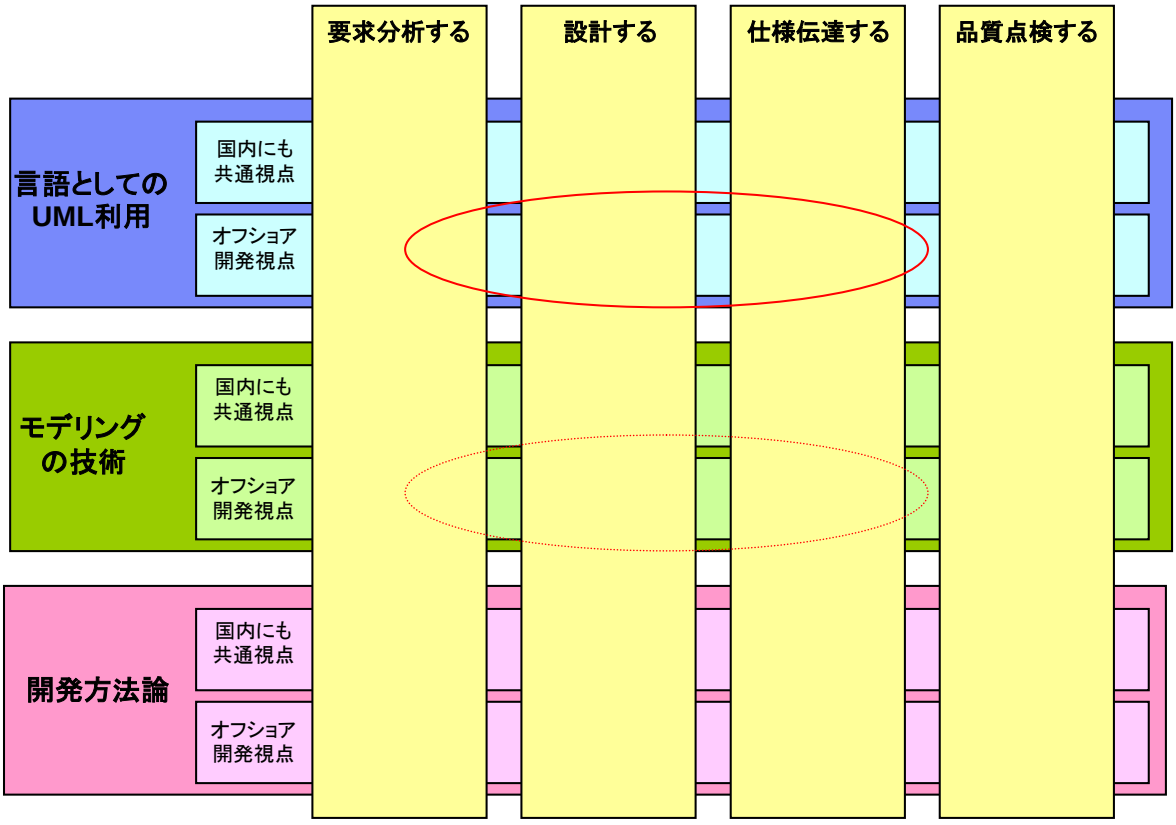
・　国内開発にも共通の点と、オフショア開発特有の点

があり、そこでオフショア開発特有の点に着目しても、開発に利用する要素として、

・　言語として UML 利用する際の技術とノウハウ
・　モデリングを行うにあたっての技術とノウハウ
・　経験に基づく知的資産となった開発方法論の技術とノウハウ

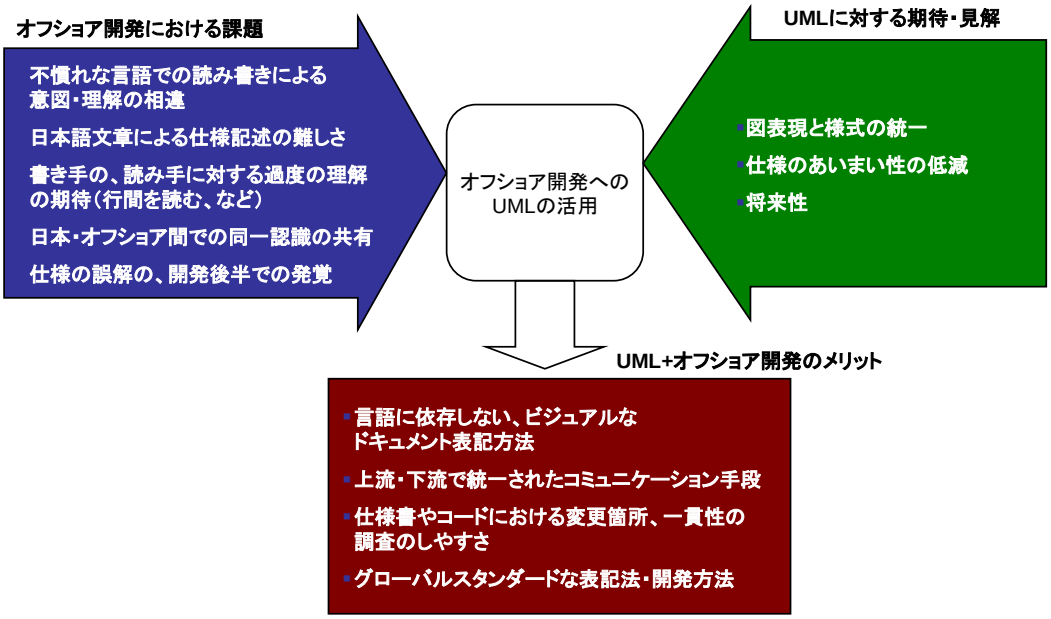
が関連事項として挙げられる。
しかし、本ガイドラインでは、「UML を利用するオフショア開発で、要求や仕様を記述し、仕様を伝えて、成果を点検する」という点に絞って記述している。すなわち、UML そのものの解説や使い方、モデリングの解説や具体例、開発方法論の解説や使い方については、本ガイドラインで示してはいない。

以上のことから、本ガイドラインの読者には、UML、モデリング、開発方法論、ソフトウェア開発技術要素それぞれに知識があることを前提とし、本ガイドラインをそれらの知識をベースに前提知識に紐付けながら読み進められることを期待したい。（本ガイドラインは、「オフショア開発に固有でない UML、モデリング、開発方法論、品質管理、プロジェクト管理のすべても解説し、かつ、オフショアソフトウェア開発のノウハウも合わせて詳述する」ことを目的としていない。）



２．オフショア開発における現状の問題点と対策

オフショア開発の発注側（日本側）・受注側（オフショア側）それぞれに対して、オフショア開発における UML 活用状況の現状を把握するべくアンケートおよびヒアリングを実施した。アンケート結果のポイントをまとめると下図のようになる。発注側・受注側共に、UMLに対する期待が大きく、UMLの活用で、課題がかなり解決されると認識している。



（１）アンケート調査概要

＜発注側（日本側）アンケート＞

調査対象： UMTP 参加企業および関連企業

調査時期： 2006年11月

有効回答数： 70 （主にPM/PLの回答　平均開発経験年数 16.2 年　オフショア開発経験年数　2.2 年）

＜受注側（オフショア側）アンケート＞

調査対象： UMTP 参加企業および関連企業のオフショア発注先の中国企業

調査時期： 2007年3月

有効回答数： 91 （主にPM/PLの回答　平均開発経験年数 4.4 年）

※今回は、日本からのオフショア先として現状では最も多いと考えられる中国を調査対象とした。

（２）アンケート結果内容（一部抜粋）

アンケート結果より、オフショア開発における課題、UMLに対する期待・見解とオフショア開発にUMLを活用した時のメリットについて下記に抜粋する。

オフショア開発における課題

表2.1.1 オフショア開発における課題と深刻度 － 発注側（日本側）

		極めて深刻	かなり深刻	深刻	やや深刻	全く深刻ではない	深刻度が高い割合(%)
1	言語や文化の差が原因で誤解が生じる	4	19	14	13	10	38
2	オフショア企業側に伝達内容を正確に伝えるのに時間がかかりすぎる	1	15	16	18	10	27
3	仕様書の曖昧さが原因で誤解が生じる	1	11	14	22	13	20
4	仕様書とオフショア企業側から納品されたソースコードに乖離が生じる	4	17	21	14	4	35
5	仕様書に漏れ・抜けが起こる	2	13	19	16	10	25
6	オフショア企業からくる大量の質問への対応で業務が圧迫される	6	19	19	13	2	**42**
7	テスト段階以降になって品質の問題が多く露呈する	3	14	11	26	6	28
8	仕様書を詳細に記述するための負担が大きい	3	14	22	14	8	28
9	オフショア企業側でのテスト内容に不備、不足がある	5	13	14	22	6	30
10	オフショア企業の担当者が離職することで、業務の継続に支障がある	12	22	13	9	3	**58**
11	オフショア企業が保守・メンテナンスを担当する場合、バグの改修におけるリードタイムが大きい	11	23	16	7	2	**58**
12	オフショア企業と同じイメージで効果的な情報共有ができない	7	27	17	6	2	**58**

表2.1.2 オフショア開発における課題と深刻度 － 受注側（オフショア側）

（注： 一部質問が異なるため、通し番号が飛んでいる）

		極めて深刻	かなり深刻	深刻	やや深刻	全く深刻ではない	深刻度が高い割合(%)
1	言語や文化の差が原因で誤解が生じる	6	7	11	31	36	14
2	発注元（日本）側に伝達内容を正確に伝えるのに時間がかかりすぎる	2	6	14	32	37	9
3	仕様書の曖昧さが原因で誤解が生じる	9	28	24	28	2	**41**
4	仕様書とオフショア企業側から納品されたソースコードに乖離が生じる	13	12	10	27	28	28
5	仕様書に漏れ・抜けが起こる	12	23	33	19	4	38
12	発注元（日本）側と同じイメージで効果的な情報共有ができない	7	27	17	6	2	**58**
13	プロジェクト内メンバ間のコミュニケーションができていない	4	15	16	23	32	21

発注側（日本側）・受注側（オフショア側）共に相手と「12. 同じイメージで効果的な情報共有ができない」点を深刻な問題点と捉えている。受注者側は、「3. 仕様書の曖昧さが原因で誤解が生じている」ことを問題視しているのに対し、仕様書の書き手である発注側はあまり問題としていない。しかし、実際には仕様書で伝えきれないために発注側は「6. オフショア企業からくる大量の質問への対応で業務が圧迫される」という構図が推測できる。

オフショア開発における課題に対する解決策

前記の「12. 同じイメージで効果的な情報共有ができない」という課題に対して、仕様書やレビューでUMLに限らず統一的に図表を利用している人は表 2.2 の通り全体の74%であり、大多数の人が図表の利用を実践し重要性は認識している。

表2.2 仕様書やレビューで図表やUMLを使う割合 － 発注側（日本側）

	ケース	平均割合(%)	
(a)	UML （そのダイアグラム1つでも） を利用する	19%	
(b)	UMLではないが、組織内で図表の描き方を統一して利用する	38%	
(c)	(a)(b)には該当しないが、個人的に図表の描き方を統一して利用する	17%	74%
(d)	統一されていない描き方で図表を使用する	14%	
(e)	文字は用いるが、その中に図表は使わない	5%	
(f)	口頭で行う	3%	
(g)	その他の方法	4%	

さらに、UML に対しての認識は下記の調査の通りであり、単なる図表に比べてより明確な効果が期待されていることが伺える。

表2.3.1 UML に対する期待・見解 － 発注側（日本側）

	意見	全く反対	どちらか言うと反対	どちらとも言えない	ある程度賛成	非常に賛成	賛成の割合(%)
1	UMLは上流工程に関わる人だけが知っていればよい	29	28	8	2	0	3
2	UMLを使っても目に見えるほどのコストは下がらない	3	9	28	21	6	40
3	周囲にUMLを使っている事例がほとんどない	10	17	9	22	9	46
4	UMLは教育が難しい、教育にコストがかかる	4	10	17	31	5	**54**
5	UMLによって生産性が向上する	1	4	38	19	5	36
6	UMLによって下流工程の手戻りが少なくなる	2	5	30	24	6	45
7	UMLによって品質が向上する	2	4	27	29	5	**51**
8	UMLは専門的すぎて難しい	9	22	24	10	2	18
9	UMLを使うと見積が難しくなる	5	20	32	8	2	15
10	UMLを使うと進捗管理が難しくなる	6	20	37	4	0	6
11	仕様にUMLを使うと曖昧性を低減することができる	1	2	18	37	9	**69**
12	仕様にUMLを使うと仕様書変更の影響範囲が把握しやすくなる	2	6	17	32	10	**63**
13	レビューでUMLを使うと、説明内容を効果的に伝えることができる	2	4	15	35	11	**69**
14	UMLを使うと開発に混乱が生じる	12	29	20	6	0	9
15	上流工程ではUMLよりも、それに特化したツールの方が優れている	5	15	36	9	2	16
16	UMLを使うと発注元や発注先との間のコミュニケーションが促進される	2	3	30	23	9	**48**
17	UMLを使うと維持管理のトータルコストを下げることができる	1	13	36	12	5	25
18	UMLは開発を自動化するのに有効である	3	15	26	18	5	34
19	UMLは国内開発よりオフショア開発でより沢山利用されている	3	6	42	12	3	23
20	UMLは国内開発よりオフショア開発の方が向いている	4	6	29	24	3	41

表2.3.2 UML に対する期待・見解 － 受注側（オフショア側）

（注： 一部質問が異なるため、通し番号が飛んでいる）

	意見	全く反対	どちらか言うと反対	どちらとも言えない	ある程度賛成	非常に賛成	賛成の割合(%)
1	UMLは発注元（日本）側だけが知っていればよい	51	22	4	4	1	6
2	UMLを使っても目に見えるほどのコストは下がらない	7	23	35	15	2	21
3	周囲にUMLを使っている事例がほとんどない	27	19	23	9	4	16
4	UMLは習得するのにコストがかかる	4	18	26	27	6	41
5	UMLによって生産性が向上する	0	0	14	47	21	**83**
6	UMLによって発注元（日本）からの手戻りが少なくなる	1	6	21	33	18	**65**
7	UMLによって品質が向上する	0	1	26	36	19	**67**
8	UMLは専門的すぎて難しい	10	21	20	23	8	38
9	UMLを使うと見積が難しくなる	4	32	39	7	0	9
10	UMLを使うと進捗管理が難しくなる	6	45	27	3	1	5
11	仕様にUMLを使うと曖昧性を低減することができる	1	3	12	45	21	**80**
12	仕様にUMLを使うと仕様書変更の影響範囲が把握しやすくなる	0	3	18	37	24	**74**
13	レビューでUMLを使うと、説明内容を効果的に伝えることができる	1	4	19	35	23	**71**
14	UMLを使うと開発に混乱が生じる	24	37	15	5	1	7
16	UMLを使うと発注元（日本）との間のコミュニケーションが促進される	0	3	21	40	17	**70**
18	UMLは開発を自動化するのに有効である	1	3	23	43	12	**67**
19	UMLは他の開発よりオフショア開発でより沢山利用されている	2	12	44	20	4	29
20	UMLはオフショア開発に向いている	4	4	43	27	6	39
21	UMLに関する本は沢山ある	0	13	16	30	23	**65**
22	UMLは知っていると昇進や転職のときに有利になる	1	4	25	28	24	**63**

発注側（日本側）・受注側（オフショア側）共に UML の「11. 仕様の曖昧性を低減」、「12. 仕様変更の影響範囲が把握しやすい」、「13. レビューで UML を使うと説明内容を効果的に伝えることができる」といった効果を認識し、双方間での「16. コミュニケーションが促進」される結果「7. 品質が向上する」と考えている。特に、受注側の UML に対する期待が高い。一方で、発注側の多くは「4. 教育が難しい、教育コストがかかる」といった課題をあげている。

ＵＭＬをオフショアに使った時のメリット

表2.4.1 UML をオフショア開発で使った時のメリット － 発注側（日本側）

	メリット	全く重要でない	ある程度重要	非常に重要	加重合計
1	グローバルスタンダードなドキュメント表記方法が使える	13	26	22	**131**
2	ビジュアルなドキュメント表記方法が使える	10	26	24	**134**
3	ユーザ企業側でも分かるようなドキュメント表記方法が使える	13	27	20	**127**
4	ソースコードを見なくても仕様書で修正箇所が調査できるので保守しやすい	14	31	15	**121**
5	開発の上流から下流までシームレスになってい	16	27	18	**124**
6	オフショア開発と国内開発で共通の開発支援ツールが使用できる	9	21	30	**141**
7	上流と下流で統一されたコミュニケーション手段がある	8	26	26	**138**
8	グローバルスタンダードな開発方法が使える	10	27	24	**136**
9	開発されたものが再利用できる	15	28	17	**122**
10	できるだけ自然言語に依存しない表記法が使え	13	30	17	**124**
11	仕様書とソースコードの一貫性が確認できる	8	31	21	**133**
12	発注先の国の言語（例：中国語）によるコミュニケーションできる	24	22	14	**110**
13	品質指標が確立されていて、品質に保障が取れる	13	21	26	**133**

表2.4.2 UML をオフショア開発で使った時のメリット － 受注側（オフショア側）

	メリット	全く重要でない	ある程度重要	非常に重要	加重合計
1	グローバルスタンダードなドキュメント表記方法が使える	7	41	43	**218**
2	ビジュアルなドキュメント表記方法が使える	5	44	42	**219**
3	ユーザ企業側でも分かるようなドキュメント表記方法が使える	3	43	45	**224**
4	ソースコードを見なくても仕様書で修正箇所が調査できるので保守しやすい	5	43	43	**220**
5	開発の上流から下流までシームレスになってい	2	48	41	**221**
6	オフショア開発と国内開発で共通の開発支援ツールが使用できる	17	42	32	**197**
7	上流と下流で統一されたコミュニケーション手段がある	7	39	58	**259**
8	グローバルスタンダードな開発方法が使える	5	46	40	**217**
9	開発されたものが再利用できる	5	36	50	**227**
10	できるだけ自然言語に依存しない表記法が使え	10	53	28	**200**
11	仕様書とソースコードの一貫性が確認できる	5	26	60	**237**
12	発注元の国の言語（日本語）によるコミュニケーションできる	5	35	51	**228**
13	品質指標が確立されていて、品質に保障が取れる	5	20	66	**243**

オフショア開発の場合、オフショア先とのコミュニケーションが重要になる。そのため、「12. 相手の国の言語によるコミュニケーション」というのは重要である。日中オフショアの場合には、中国側が日本語を理解してくれるケースが多いので、発注側の結果ではさほど UML 利用のメリットを感じていないが、受注側（オフショア側）はメリットと感じている。上流工程を発注側（日本側）で下流を受注側（オフショア側）が分担するケースが多いと思われるが、「7. 上流と下流で統一されたコミュニケーション手段がある」というメリットは、発注側・受注側共に認識している。

また、UMLの特徴である「2. ビジュアルな表記法」、「1. グローバルスタンダードな表記法」・「8. グローバルスタンダードな開発方法」などもオフショア開発においては重要なポイントと考える。

さらに保守性の観点では、「11. 仕様書とソースコードの一貫性が確認できる」などの支持も多数ある。

（3）ヒアリング

前記のアンケートによる調査結果を補完するために下記のヒアリングを実施し、本ガイドラインが受注側（オフショア側）からみても納得感のある内容となっていること、中国だけでなくインドにおいても適用可能な汎用性のあるものになっていることを確認している。

＜中国のPM／PLへのヒアリング（一部は記述式のアンケート）＞
目的： 定量的なアンケート調査では掴みきれない実態の把握のため
時期： 2007年11月

＜インドのアーキテクトへのヒアリング＞
目的： オフショア開発で日－中を先行するUS－インドにおいてのUML利用状況把握のため
時期： 2008年3月

３．UML モデリング

3.1 ＵＭＬの特徴

（１）ＵＭＬとは

UML(Unified Modeling Language)は、OMG(Object Management Group)により標準化された、ソフトウェアの成果物を仕様化、図式化するときに使用する表記法である。2005 年 4 月にUML Version 1.4.2 が国際規格 ISO/IEC 19501:2005 として、2009 年 5 月には 、JIS X 4170:2009 として成立した。

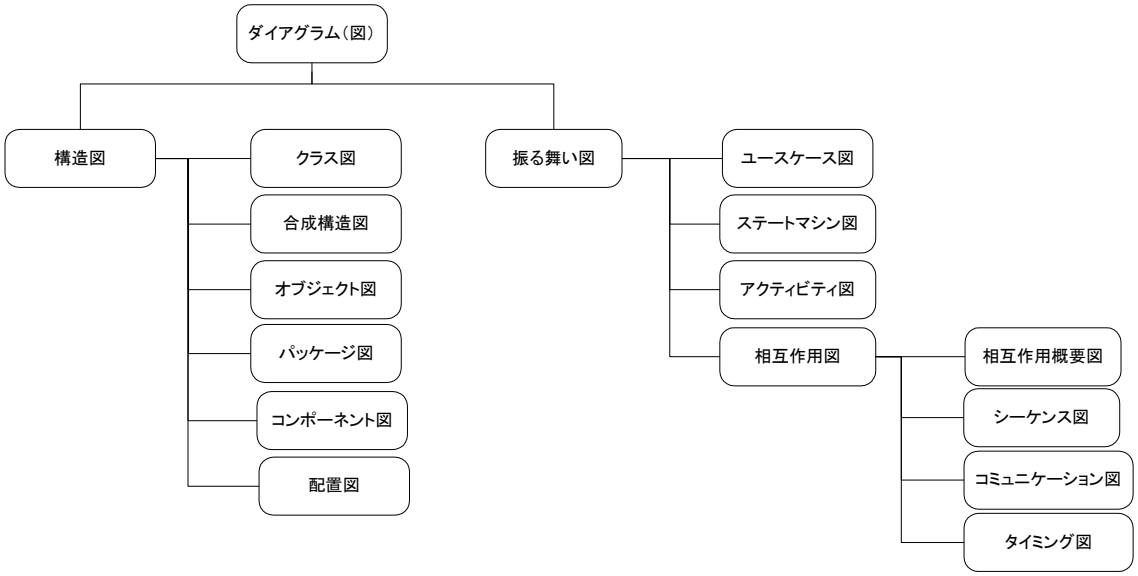
（２）ＵＭＬの特徴

UMLの特徴としては次のようなものが挙げられる。

- 世界標準である。
  - 世界中の異なる国間でも、異なるプロジェクト間においても、コミュニケーションが容易である。
- 表現力が高く、しかも理解が容易である。
  - UMLが提供する複数のダイアグラム（図）により、多様な視点で分析・設計対象を表現できる。しかし、どのダイアグラムも直感的で理解しやすい。
- 開発の全ての工程で一貫して使用できる。
  - 開発の各工程における成果物のトレーサビリティが確保されており、各工程間で開発者の間のコミュニケーションが容易である。

（３）ＵＭＬのダイアグラム（図）

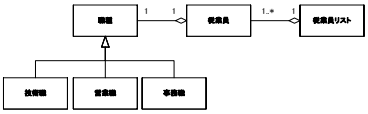
UMLのダイアグラム（図）には次のようなものがある。



・構造図

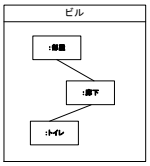
●クラス図

分析・設計領域の物や事を概念的に捉え、それをクラスおよびその関係により静的に表現する。



●合成構造図

クラスやコンポーネントの内部構造を階層的に表現する。

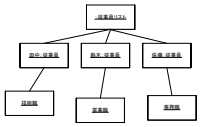


●パッケージ図

ＵＭＬ要素をまとめるパッケージ間の関係を表現する。

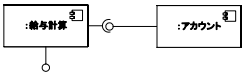
●オブジェクト図

システムのある瞬間の状態をオブジェクトおよびその関係により表現する。



●コンポーネント図

ソフトウェアコンポーネント（再利用可能な部品）の構成を表現する。



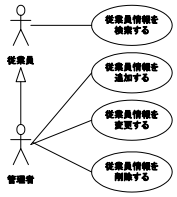
●配置図

システムの実行環境を表現する。

・振る舞い図

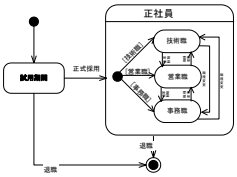
●ユースケース図

システムが提供する機能と外部との関係を表現する。



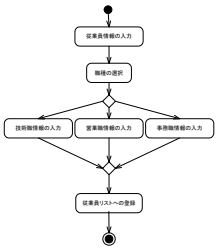
●ステートマシン図

一つのオブジェクトの時間経過に伴う状態の変化を表現する。



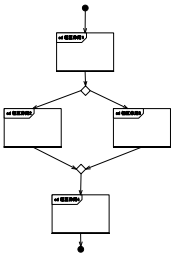
●アクティビティ図

業務フロー、イベントフロー、アルゴリズムの流れ（フロー）を表現する。



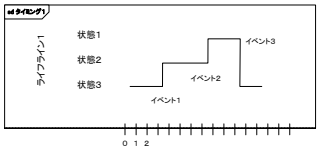
●相互作用概要図

複数の相互作用図の関係を概観し表現する。



●タイミング図

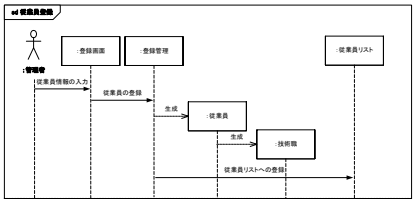
時間経過とともに変化するライフラインの状態の変化やメッセージのやり取りを表現する。



○

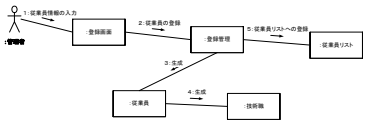
●シーケンス図

相互作用（分析・設計対象内に存在する“物”や“人”のやり取り）を時系列に表現する。



●コミュニケーション図

相互作用（分析・設計対象内に存在する“物”や“人”のやり取り）を“物”、“人”を中心にその接続関係に着目して表現する。



3.2 前提とするＵＭＬモデリングスキルレベル

モデルを利用したオフショア開発を効率的に行う為には、発注側、受注側ともある一定のUMLモデリングのスキルを保持している必要がある。これらのスキルを保持していない場合は、事前に学習、教育を行っておくべきである。また UML モデリングスキルレベルは、分析者、設計者、実装者、アーキテクト、プロジェクトリーダー、プロジェクトマネージャーなど立場によって、必要とされるレベルは異なることになる。

尚、ここで定義しているスキルレベルは、モデリングに関するもののみであり、分析者、設計者、実装者、アーキテクト、プロジェクトリーダー、プロジェクトマネージャーに一般的に必要とされるスキル、知識を前提としたものである。これら一般的に必要とされる、スキル、知識についてはここでは定義していない。

【目的】

オフショア開発に UML を使用した成果物のやりとりを、発注側と受注側で行うときに、双方が一定の UML モデリングスキルレベルを保持していれば、必要最小限の成果物のやり取りのみで、正解に意図が伝わる。開発要員の UML モデリングスキルが十分でないとき、成果物の意図が正確に伝わらないばかりか、UML やモデリングについての説明を開発担当者が行う必要が発生し、その分無駄な工数が発生してしまうことになる。

【詳細・補足】

〇レベルの測定（確認）方法

UMTP では、モデリングスキルレベルを次のように、L1～L4 まで定義している。

レベル	モデリングスキル	説明
L4	実践に基づいてモデリングを指導できる	L3のスキルを有し、開発プロジェクトでモデリングを一定数あるいは期間実践した経験を持つ
L3	実務でモデリングが実践できる	拡張性や変更容易性の点で高品質なモデルを定義できる
		ビジネスモデリング、分析、アーキテクチャ設計、組み込み開発を行うための専門的な知識を備えている
L2	UMLモデルの読み書きが普通にできる（モデリングリテラシーがある）	開発範囲の一部を担当し、モデリングができる
		他者のモデルの意味を理解できる
L1	簡単なUMLモデルの意味が分かる	UMLなどを使ってモデリングを行う最低限の知識を持っている

UMTPでは、L1～L2 まで、このスキルレベルに対応した試験を実施している。（L3 は 2008 年4月より実施）この試験に合格することで、スキルレベルの測定が可能である。

また L1 レベルについては、L1-T1 と L1-T2 の2つの試験を実施している。

L1-T1→UML の表記法を知っている。

L1-T2→UML で描かれたモデルを読み、良し悪しを判断できる。

※中国では、L1-T2 と L2 の試験について、実施中である。

実際の開発においては役割により、それぞれ次のようなレベルが最低限要求される。

実装者・・・UMLのモデルを見て、ソースコードを作成する。→L1-T2 レベル

分析者（一般）・・・分析者（上級）の指示のもとUMLの分析モデルを独力で作成する。→L2 レベル

設計者・・・アーキテクチャに基づきUMLの設計モデルを独力で作成する。→L2 レベル

分析者(上級)・・・初期分析モデルを作成する。→L3 レベル
アーキテクト・・・アーキテクチャモデルを作成する。→L3 レベル
プロジェクトリーダー・・・モデル分析・設計方針のレビュー、決定を行う。→L3 レベル
プロジェクトマネジャー・・・プロジェクトの最高責任者→L1～L2 レベル

	L1-T1	L1-T2	L2	L3
実装者	○	○	-	-
分析者(一般)	○	○	○	-
分析者(上級)	○	○	○	○
設計者	○	○	△	-
アーキテクト	○	○	○	○
プロジェクトリーダー	○	○	○	○
プロジェクトマネジャー	○	○	△	-

○スキルアップ
UMLの知識(L1-T1)を得たり、モデリング初級(L1-T2)のスキルを習得することは、市販の書籍を1冊読むことで可能である。しかし、UMLのモデルを独力でつくるスキル(L2)については、書籍だけでそのスキルを習得することは困難である。まずは、トレーニングやメンタリングによりモデリングのコツを掴み、その後、たくさんのモデルを見て、理解し、描いてみる必要がある。
さらに、上級のモデリングスキルレベル(L3)では、実務レベルで経験を積む必要がある。

○参考情報
・UMTP 基準準拠
市販の書籍には、UMTP基準準拠表示されているものがある。
これは、モデリング用語集　UML 編　第 2 版に適合しているかなどを審査し、許可するものである。
UMTP 基準準拠の書籍には以下のものがある。
http://www.umtp-japan.org/modules/data4/index.php?id=7

・トレーニングテキスト認定
トレーニングテキストに関しては、モデリング用語集　UML 編　第 2 版に適合しているかに加えて、
モデリングに必要な知識をがトレーニングテキスト中で解説されているかを審査し、認定を行っている。
認定トレーニングコースは次のものがある。
http://www.umtp-japan.org/modules/data4/index.php?id=8

【関連情報】
なし

４． UML の適用範囲と開発のノウハウ(Hints & Tips)

2章のアンケート結果から、多数の人が、オフショア開発に UML を適用した時のメリットを認識していることが分かる。しかし UML をオフショア開発に適用すれば、すぐに効果がでるのではなく、開発のどの局面に、どのように適用するかが重要である。本章では、オフショア開発を成功させるには、各工程で、どのような作業をすればよいかのノウハウを説明する。ノウハウは、UML を中心に説明するが、UML に特化しないものも含んでいる。

図4．1に開発の流れ、表4．1に各工程の目的、作業内容、ワーカー、入力成果物、出力成果物を示す。図4．1は主要な成果物のみを記載しており、成果物の詳細は表4．1を参照すること。開発の各工程で注意すべき事項（ノウハウ）をポイント番号で図4．1に示す。各工程のポイント番号ごとに、ノウハウを以下の形式で説明する。

・ポイント番号とタイトル
・内容
・目的
・詳細・補足
・関連情報

なお、条件が整い実施できる場合には効果が高いが、オフショア開発の初期段階では無理して実施する必要のないノウハウについては、Option（状況に応じて選択可能）の扱いとしている。

図 4.1 開発の流れ

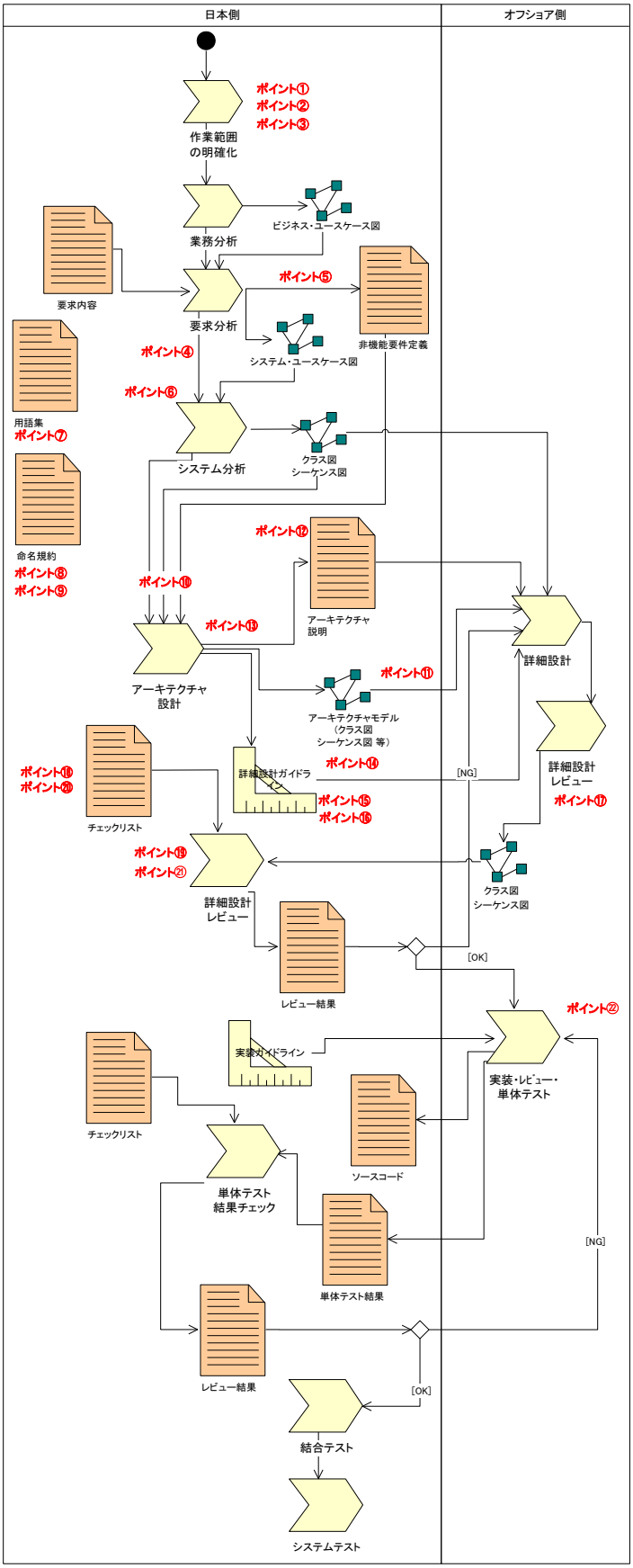

日本側
オフショア側
ポイント①
ポイント②
ポイント③
作業範囲の明確化
業務分析
ビジネス・ユースケース図
要求内容
要求分析
ポイント⑤
非機能要件定義
システム・ユースケース図
ポイント④
ポイント⑥
用語集
ポイント⑦
システム分析
クラス図
シーケンス図
命名規約
ポイント⑧
ポイント⑨
ポイント⑫
ポイント⑩
アーキテクチャ説明
ポイント⑬
詳細設計
アーキテクチャ設計
ポイント⑪
アーキテクチャモデル
（クラス図
シーケンス図 等）
詳細設計レビュー
ポイント⑰
ポイント⑭
[NG]
詳細設計ガイドライン
ポイント⑮
ポイント⑯
ポイント⑱
ポイント⑳
チェックリスト
ポイント⑲
ポイント㉑
詳細設計レビュー
クラス図
シーケンス図
[OK]
レビュー結果
ポイント㉒
実装ガイドライン
実装・レビュー・単体テスト
チェックリスト
ソースコード
単体テスト結果チェック
単体テスト結果
[NG]
レビュー結果
[OK]
結合テスト
システムテスト

表4．1各工程の説明

工程名：業務分析	
目的（日本側）： 今回のシステムの業務内容を理解し、明確にすることで、要求分析工程において、業務のどの部分をシステム化するかを明確にしやすくする。	目的（オフショア側）： 今回対象とするシステムの業務内容を理解することで、次工程以降の作業が円滑に進むように準備をする。
作業内容（日本側）： 業務内容を理解し、明確にする。必要に応じて業務の流れを整理する。	作業内容（オフショア側）： 今回対象とするシステムの業務内容を理解する。（上流に参加している場合）
ワーカー（日本側）： PM 分析者	ワーカー（オフショア側）： PM システム分析者（上流に参加している場合）
入力成果物： ・業務フロー	入力成果物： ・ビジネス・ユースケース図 ・業務フロー ・ビジネス・ユースケース記述 ・概念クラス図
出力成果物： ・ビジネス・ユースケース図 ・業務フロー ・ビジネス・ユースケース記述 ・概念クラス図	出力成果物：
参考情報： ・用語集 ・命名規約	参考情報 ・用語集 ・命名規約

工程名：要求分析	
目的（日本側）： システムの機能的要求および非機能的要求を明確にする。	目的（オフショア側）： 詳細設計以降を担当する場合は、直接的にこの工程の成果物を利用するわけではない。 この工程の成果物から、今回システム開発対象にしているシステムの機能を理解しておく。 システム機能を知っておくことで、なぜこのような設計をすべきかが理解できるようになり、詳細設計以降のコミュニケーションがスムースに行なわれる。
作業内容（日本側）： 顧客のシステムに対する要求を明確にする。	作業内容（オフショア側）： 出力成果物を参考にして、対象システムの機能を理解する。（上流に参加している場合）
ワーカー（日本側）： 要求分析者	ワーカー（オフショア側）： システム分析者（上流に参加している場合） 詳細設計者（上流に参加している場合）
入力成果物： ・ビジネス・ユースケース図 ・業務フロー ・ビジネス・ユースケース記述 ・要求内容	入力成果物： ・システム・ユースケース図 ・システムユースケース一覧 ・ビジネス・ルール定義 ・システム・シーケンス図 ・システム・ユースケース記述 ・画面遷移図 ・画面・帳票設計書 ・非機能要件定義
出力成果物： ・システム・ユースケース図 ・システムユースケース一覧 ・ビジネス・ルール定義 ・システム・シーケンス図 ・システム・ユースケース記述 ・画面遷移図 ・画面・帳票設計書 ・非機能要件定義	出力成果物：
参考情報： ・用語集 ・命名規約	参考情報 ・用語集 ・命名規約

工程名：システム分析	
目的（日本側）： システム開発対象の領域の知識を自然な形で分析する。 開発、運用環境に依存しないモデルを作成することで、再利用性、保守性を向上させる。	目的（オフショア側）： 詳細設計以降を担当する場合は、直接的にこの工程の成果物を利用するわけではない。 この工程の成果物から、今回システム開発対象にしている領域の知識の習得を目的する。 システム開発対象の知識を得ることで、なぜこのような設計をすべきかが理解できるようになり、詳細設計以降のコミュニケーションがスムースに行なわれる。
作業内容（日本側）： システムの概要レベルの構造・関連を示すシステムのコンテキストを定義する。その後、システムの地理的分散や運用の複雑さを理解するための概要を整理し、システムの振舞いを分析する。	作業内容（オフショア側）： 出力成果物を参考にして、システム開発領域を理解する。（上流に参加している場合）
ワーカー（日本側）： システム分析者	ワーカー（オフショア側）： システム分析者（上流に参加している場合） 詳細設計者（上流に参加している場合）
入力成果物 ・システム・ユースケース図 ・システムユースケース一覧 ・ビジネス・ルール定義 ・システム・シーケンス図 ・システム・ユースケース記述 ・画面遷移図 ・画面・帳票設計書	入力成果物 ・分析クラス図 ・分析シーケンス図 ・オブジェクト図 （・コミュニケーション図） （・ステートマシン図）
出力成果物 ・分析クラス図 ・分析シーケンス図 ・オブジェクト図 （・コミュニケーション図） （・ステートマシン図）	出力成果物
参考情報： ・用語集 ・命名規約	参考情報 ・用語集 ・命名規約

工程名：アーキテクチャ設計	
目的（日本側）： システムの基盤構造を明確にする。	目的（オフショア側）： 詳細設計のインプットとする。
作業内容（日本側）： 候補となるアーキテクチャを定義し、分析結果を元に、再利用可能なモデル要素を取り込みながら、定義されたアーキテクチャを洗練する。コンポーネント設計、データベース設計を行いながら、アーキテクチャの詳細検討、クラス/サブシステムの識別、サブシステム間インターフェースの識別を行っていく。	作業内容（オフショア側）： 出力成果物を参考にアーキテクチャを理解する。 （上流に参加している場合）
ワーカー（日本側）： アーキテクト	ワーカー（オフショア側）： アーキテクト（上流に参加している場合） 詳細設計者（上流に参加している場合）
入力成果物 ・分析クラス図 ・分析シーケンス図 ・オブジェクト図 （・コミュニケーション図） （・ステートマシン図） ・非機能要件定義	入力成果物 ・設計クラス図　・分析クラス図 ・設計シーケンス図　・分析シーケンス図 ・オブジェクト図 ・パッケージ図 ・コンポーネント図 ・配置図 ・アーキテクチャ説明 ・詳細設計ガイドライン ・詳細設計チェックリスト ・データ・モデル図（E-R図）
出力成果物 ・設計クラス図 ・設計シーケンス図 ・オブジェクト図 ・パッケージ図 ・コンポーネント図 ・配置図 ・アーキテクチャ説明 ・詳細設計ガイドライン ・詳細設計チェックリスト ・データ・モデル図(E-R図)	出力成果物
参考情報： ・用語集 ・命名規約	参考情報 ・用語集 ・命名規約

<table>
<tr><td colspan="2">工程名：詳細設計</td></tr>
<tr><td>目的（日本側）：

作業工数が多い部分について、中国側に作業を委託する。</td><td>目的（オフショア側）：

実装作業を行なえるほど、モデルを詳細化する。</td></tr>
<tr><td>作業内容（日本側）：

中国側の作業が、アーキテクチャから逸脱していないかをチェックする。</td><td>作業内容（オフショア側）：

アーキテクチャモデルやアーキテクチャ説明書を入力として、サブシステム間で調整を行いながら、詳細化して実装モデルを作成する。</td></tr>
<tr><td>ワーカー（日本側）：

レビューアー</td><td>ワーカー（オフショア側）：

アーキテクト（上流に参加している場合）
詳細設計者</td></tr>
<tr><td>入力成果物
・詳細設計クラス図
・詳細設計シーケンス図
（・オブジェクト図）
・詳細設計レビュー結果（オフショア側）</td><td>入力成果物
・設計クラス図　・分析クラス図
・設計シーケンス図　・分析シーケンス図
・オブジェクト図
・パッケージ図
・コンポーネント図
・配置図
・アーキテクチャ説明
・詳細設計ガイドライン
・詳細設計チェックリスト</td></tr>
<tr><td>出力成果物

・詳細設計レビュー結果（日本側）</td><td>出力成果物

・詳細設計クラス図
・詳細設計シーケンス図
（・オブジェクト図）
・詳細設計レビュー結果（オフショア側）</td></tr>
<tr><td>参考情報：
・用語集
・命名規約</td><td>参考情報
・用語集
・命名規約</td></tr>
</table>

工程名：実装・レビュー・単体テスト	
目的（日本側）： 引き続き、作業工数が多い部分について、中国側に作業を委託する。	目的（オフショア側）： システムを稼働させるソースコードを作成する。
作業内容（日本側）： 必要に応じてチェックを行なう。	作業内容（オフショア側）： 詳細設計の成果物をもとに、実装および単体テストを行なう。
ワーカー（日本側）： レビューアー	ワーカー（オフショア側）： 実装者
入力成果物 ・単体テスト結果	入力成果物 ・詳細設計クラス図 ・詳細設計シーケンス図 （・オブジェクト図）
出力成果物 ・チェック結果	出力成果物 ・ソースコード ・単体テスト結果
参考情報： ・用語集 ・命名規約 ・実装ガイドライン ・チェックリスト	参考情報 ・用語集 ・命名規約 ・実装ガイドライン ・チェックリスト

工程名：単体テスト結果チェック	
目的（日本側）： 品質に問題が無いかをチェックする。	目的（オフショア側）： 品質の最終確認をする。
作業内容（日本側）： 単体テストの結果のチェックを行う。	作業内容（オフショア側）： 必要に応じて、単体テスト結果を見直す。
ワーカー（日本側）： レビューアー	ワーカー（オフショア側）： 実装者
入力成果物 ・単体テスト結果	入力成果物 レビュー結果
出力成果物 レビュー結果	出力成果物 単体テスト結果
参考情報： ・用語集 ・命名規約 ・実装ガイドライン ・チェックリスト	参考情報 ・用語集 ・命名規約 ・実装ガイドライン ・チェックリスト

工程名：結合テスト	
目的（日本側）： 単体テストを行なったソースコードを、結合しテストを行なう。	目的（オフショア側）： ソースコードの品質をさらに高める。
作業内容（日本側）： 結合テストを行なう	作業内容（オフショア側）： 詳細設計、実装に関するバグがある場合は、修正する。
ワーカー（日本側）： テスター	ワーカー（オフショア側）： 詳細設計者 実装者
入力成果物 ・ソースコード ・単体テスト結果	入力成果物 ・詳細設計クラス図 ・詳細設計シーケンス図 ・ソースコード ・バグ票
出力成果物 ・結合テスト結果	出力成果物 ・詳細設計クラス図 ・詳細設計シーケンス図 ・ソースコード ・バグ票 ・チェックリスト
参考情報： ・用語集 ・命名規約	参考情報 ・用語集 ・命名規約

表 4.2 工程ごとの主要な成果物

工程	成果物セット	成果物	UML の図
業務分析	ビジネス・プロセス・モデル	ビジネス・ユースケース図	ユースケース図
		業務フロー	アクティビティ図
		ビジネス・ユースケース記述	
	ドメイン・モデル	概念クラス図	クラス図
			オブジェクト図
要求分析	ユース・ケース・モデル	システム・ユースケース図	ユースケース図
		システム・ユースケース一覧	
		ビジネス・ルール定義	
		システム・シーケンス図	シーケンス図
			相互作用概要図
		システム・ユースケース記述	アクティビティ図で作成する場合あり
	ユーザ・インターフェース・モデル	画面遷移図	ステートマシン図
		画面・帳票設計書	
	非機能要件	非機能要件定義	
	用語集	用語集	
システム分析	分析モデル	ロバストネス図	
		分析クラス図	クラス図
			オブジェクト図
		データ・モデル図(E-R 図)	
		相互作用図	シーケンス図
			コミュニケーション図
		状態図	ステートマシン図
アーキテクチャ設計/詳細設計	アーキテクチャ・モデル	アーキテクチャ説明	
		ソフトウェア構成	パッケージ図
			コンポーネント図
			合成構造図
		ハードウェア構成	配置図
		基本処理方式	シーケンス図
	設計モデル	設計クラス図/詳細設計クラス図	クラス図
			オブジェクト図
		データ・モデル図(E-R 図)	
		相互作用図	シーケンス図
			コミュニケーション図
		状態図	ステートマシン図
			タイミング図
		クラス仕様書	
実装	実装セット	実装コード	
単体テスト/結合テスト工程	テスト・セット成果物セット	テスト計画書	
		テスト仕様書	
		成果物	UML の図

表 4.3 開発のノウハウ（Hints & Tips） 一覧

工程	ﾎﾟｲﾝﾄ No	ノウハウ
作業範囲の明確化	01	作業範囲／作業分担の明確化 (*)
	02	利用する UML 図の確定
	03	必ずUMLである必要はない (*)
業務分析/要求分析	04	上流工程への参画 *（Option）*
	05	非機能要件定義
	06	分析モデルで業務を理解 (*)
	07	用語辞書を作成する
	08	命名規約を作成する
	09	モデルの作成規約を作成する
システム分析/アーキテクチャ設計	10	共通機能の明確化
	11	アーキテクチャ・モデル
	12	アーキテクチャ説明成果物の作成
	13	パターンの活用
	14	仕様書の記述レベル、書式の指定 (*)
	15	詳細設計ガイドライン作成
	16	仕様未決定部分は明確に
詳細設計/実装	17	オフショア側での成果物のレビュー実施
	18	日本側はチェックを繰り返し行なう
	19	Validation と Verification
	20	モデルで詳細設計のレビュー
	21	UML 図間の整合性 (*)
	22	実装はツールのコード生成機能を使用する *（Option）*
ー	補足	日本におけるシステム開発の特徴

(Option): 条件が整い実施できる場合には効果が高いが、オフショア開発の初期段階では無理して実施する必要のないノウハウ
(*): サンプルが記載されている。サンプルの説明は付録Aを参照。

【ポイント 01 作業範囲／作業分担の明確化】

各工程の作業分担を明確にする。各工程の成果物（UML 図、及び UML 図以外）を決定する。オフショアでの開発担当領域を明確にしてモデリング範囲／テスト範囲を決める。

【目的】

システム開発においては、複数の企業（国内、海外）にてプロジェクトを推進する。
企業により、各作業工程の作業内容、成果物が異なる為、プロジェクト開始時に作業工程別の作業内容を明確にして作業工程にずれが生じないようにする。
発注側／受託側での成果物を明確にして納品時のトラブルの発生を削減する。

【詳細・補足】

各工程でのオフショア担当作業対象を明確にし、作業効率を向上させる。

例えば、開発環境面から運用要件、障害対策要件、セキュリティ要件に関する内容はオフショアでは対応困難な面があり、予め日本側で担当する領域／オフショア側で担当する領域に分けておくと良い。

〈全般〉

- ・工程単位に使用する成果物を記述する。
- ・各工程のインプット、アウトプットを明確にする。
- ・成果物の、書式・内容の粒度のガイドラインを明記する。
- ・成果物間に関連性があるものは、成果物間の不整合を防止するため、確認方法も明記する。

〈アーキテクチャ設計、詳細設計 （モデリング）〉

- ・基盤・他開発部分の提供箇所の明確化

〈実装・単体テスト （製造、作成）〉

- ・開発環境（サーバ、ソフトウェア種類・バージョン etc）の状態
- ・提供部品スケジューリング （提供可能時期）
- ・提供部品（一覧、外部仕様）
- ・使用する開発ツール

〈テスト〉

- ・開発環境の状態の考慮
- ・テスト・データの提供有無 （サンプル・データを配布した方が理解度が高い）
  テスト・データを提供する場合は、できるだけ早めに提供すること。

特に、オフショアでの開発には、以下の点をプロジェクト開始当初から考慮しておく必要がある。

- ・オフショアに準備できる開発環境・テスト環境の制約

例: 日本語/中国語の文字の使用可能性、大型機の準備の可能性、ファイルの共有可能性。

・オフショアで設計・実装しにくい非機能要件部分に関する制約

例: 統合認証機能が必要になるテスト、準備可能なテスト機のパフォーマンスに依存するテスト、実環境でないことによる制約。

・オフショアで準備しにくいテスト・データの制約

例: セキュリティに基づく顧客データの提供。日本語/中国語の文字に依存するテスト・データ。

また、オフショア開発では、コミュニケーションが重要であり、コミュニケーションプランを明確にしておく必要がある。例えば、誰と誰が、どのタイミングで(例えば週次)、どのような方法で行うかを決めておく。

作業範囲/作業分担だけでなく、オフショア先での開発体制についての要件も明確にしておく必要がある。

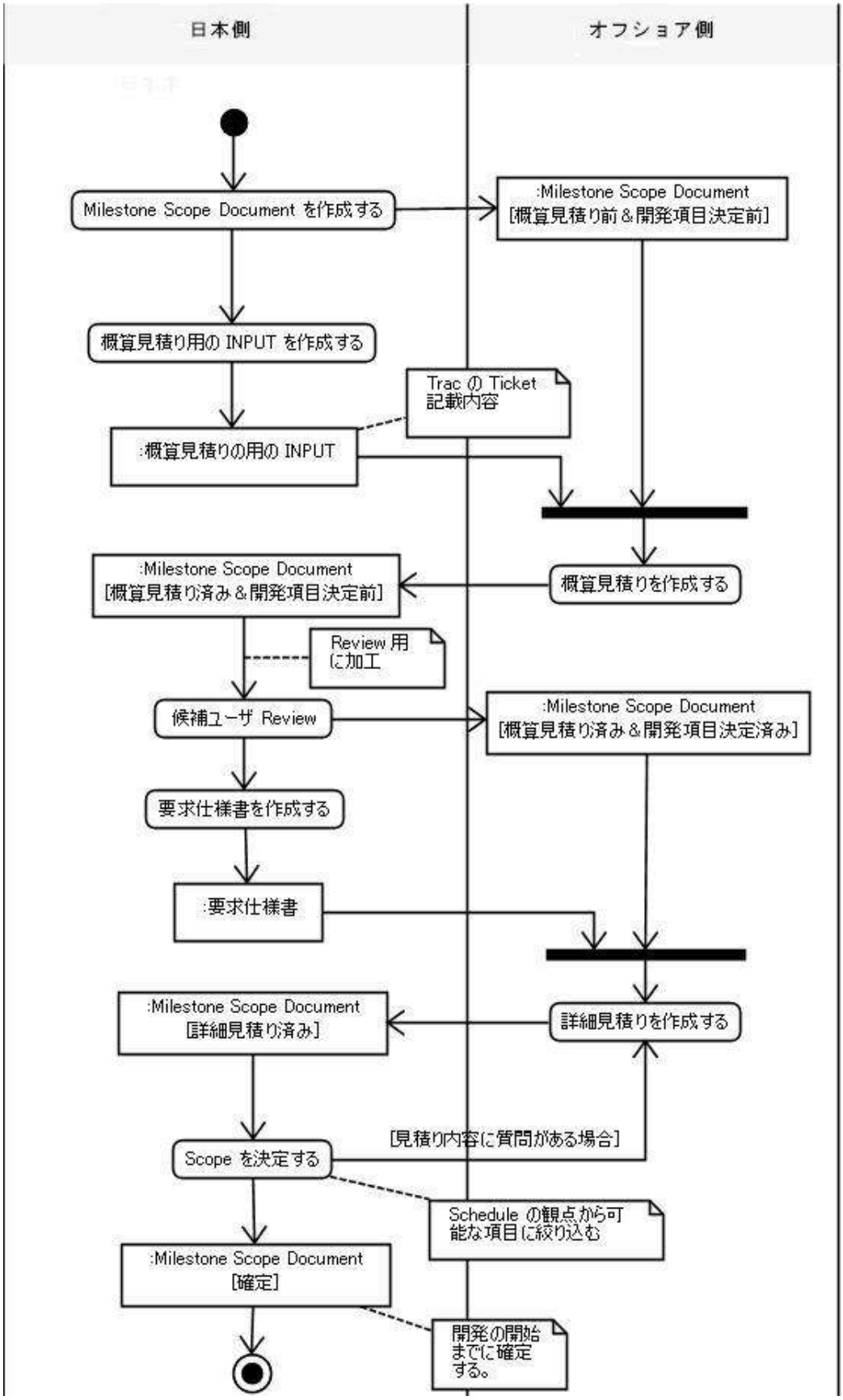
【関連情報】

図 4.1 開発の流れ

表 4.1 各工程の説明

【作業分担をアクティビティ図を使用して明確にした例】

日本側とオフショア側の作業をアクティビティ図のレーンを用いて区分けしている。



【ポイント 02 利用するUML図の確定】

各工程で使用するUML図、補助資料の種類を明確にする。

【目的】

開発の各工程においては、使用するUML図が予め決まっているわけではなく、内容に応じて取捨選択し利用する。プロジェクト開始時、工程ごとに、利用するUML図を決め、作業効率を向上させる。

【詳細・補足】

各工程で使用するUML図を明確にすることにより、各工程の作業開始時のロスを削減する。必須・任意に応じて作成（モデルの根拠の説明のため必要）などUML図のランク分けを行う。業務分析～詳細設計の各工程では、次に示すUML図(UML2．x)の利用を考えてみると良い。

〈業務分析〉

・ユースケース図	△（状況に応じて）
・アクティビティ図	○
・クラス図	△（状況に応じて）
・オブジェクト図	△（状況に応じて）

〈要求分析〉

・ユースケース図	○
・アクティビティ図	△（状況に応じて）
・シーケンス図	△（状況に応じて）
・相互作用概要図	△（状況に応じて）
・ステートマシン図	△（状況に応じて）

〈システム分析〉

・クラス図	○
・オブジェクト図	○
・シーケンス図	○
・コミュニケーション図	△（状況に応じて）
・ステートマシン図	△（状況に応じて）

〈アーキテクチャ設計、詳細設計〉
（上記の、前工程での作成図の見直しや詳細化に加

・コンポーネント図	△（状況に応じて）
・合成構造図	△（状況に応じて）
・クラス図	○
・コミュニケーション図	△（状況に応じて）
・ステートマシン図	△（状況に応じて）
・シーケンス図	○
・オブジェクト図	○
・パッケージ図	○
・配置図	△（状況に応じて）
・タイミング図	△（状況に応じて）

【関連情報】

表 4.2 工程ごとの主要な成果物

【ポイント 03 必ずUMLである必要はない】

UMLで、システム開発の全成果物を作成できない。UML として用意されている図以外も必要に応じて利用する。

【目的】

開発にあたって、UML として定義されている図だけを用いて設計書を記述しようと考える必要はない。開発対象の内容を、よりわかりやすく、より正しく伝えることが設計書の本来の目的であり、この点を見失わないことが目的である。

【詳細・補足】

UMLの図の特徴を捉えて、情報をまとめる上で最も適切な図を用いる。
UMLを適材適所で使用し、UMLで支援していな成果物は、UML以外の図を使用すべきである。たとえば、まとめる情報によっては、表形式にまとめたほうが、あきらかに分かりやすいものがある。これを、無理やりUMLのモデルで表現しても分かりにくいだけである。
システム開発のあらゆる面をUMLで作成することにこだわり、ドキュメント地獄に陥らないよう注意すべきである。

UML以外の図の利用には以下のようなものが考えられる。

- ・画面遷移図
- ・画面・帳票設計書
- ・データ・モデル図（E-R 図）
- ・データフロー図（DFD）
- ・ロバストネス図（UML の拡張）
- ・ネットワーク図
- ・一覧(表)-EXCEL 等

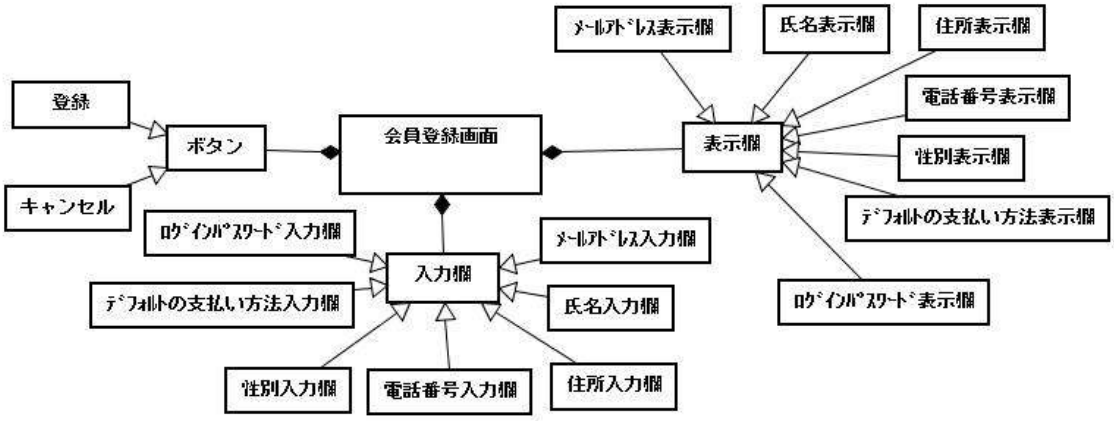
【関連情報】

表 4.2 工程ごとの主要な成果物

【UML 以外の図を利用した方がよい場合の例】

会員登録画面を設計する場合、UMLのクラス図を使用して作成した画面をサンプル1に示す。画面開発ツールやEXCEL等で作成した画面をサンプル2に示す。画面レイアウトの設計の確認やレビューを行う場合、サンプル1とサンプル2を比較すれば、明らかにサンプル2の方がわかりやすい。したがって、画面・帳票設計書には、サンプル2の図を載せ、UMLは使用しない方がよい。

＜サンプル1＞



＜サンプル2＞

会 員 登 録

メールアドレス（ログインID）	
氏名（漢字、ふりがな）	
住所	
電話番号	
性別	○ 男　　○ 女
デフォルトの支払い方法	--選択してください
ログインパスワード	

登録　　キャンセル

【ポイント 04 上流工程への参画】

Option

オフショア開発を行なう場合、日本語力が高く設計経験が豊富なSEがいれば、国内で行なう上流工程にもオフショア側のSEが参画することが望ましい。

【目的】

詳細設計以降の工程をオフショア開発する場合、詳細設計工程以降を効率的に推進する。

【詳細・補足】

上流工程をすべて日本企業で行い、詳細設計工程以降をオフショア開発とした場合、十分な成果物を作成したとしても、オフショア先で、完全にシステムを理解してもらうことは困難である。

したがって、上流工程にオフショア先のメンバが参画することにより、開発するシステムの目的、内容を理解でき、詳細設計工程以降の設計ミスを削減できる。また、システムの全体像を理解しているメンバがオフショア側にいることから、一部の結合テストについてもオフショア先へ委託できる可能性も出てくる。参加人数は、システムの規模にもよるが、数人程度でよい。いずれにしても、上流工程を任せられるようなSEがオフショア側にいるかどうかを見極めたうえで実施すべきで、徐々にオフショア先の担当領域を広げるのがよい。

【関連情報】

ポイント 01 作業範囲/作業分担の明確化
ポイント 06 分析モデルで業務を理解
ポイント11 アーキテクチャ・モデル

【ポイント 05 非機能要件定義】

非機能要件をまとめておく。 非機能要件とは、機能をどのように実現すべきか（システムの性質）を決める要件である。

【目的】

非機能要件は、アーキテクチャ設計時の重要な要件であり、アーキテクチャ設計前に明確にしておく必要がある。 非機能要件は、アーキテクチャ設計時の入力になる。

【詳細・補足】

システム要件には、機能要件と非機能要件がある。

ユースケース図で機能的な要求を明確にすると共に、非機能要件を定義する。 非機能要件の例を次に示す。

- ・開発／実行環境（ハード、ソフト）
- ・性能要件
- ・信頼性要件
- ・コスト（短期、長期）
- ・セキュリティ
- ・保守性要件
- ・運用要件

ここでは、オフショアの開発領域に限らず、システムに要求される非機能要件を指している。オフショア開発の場合には、“「常識」が常識でない”ために発生する誤解・問題が多いので、非機能要件も含めて成果目標を明確に定義しておくことは重要である。オフショア側も、非機能要件が不明確であると考える場合には、発注側に早目に確認することが肝要である。

【関連情報】

ポイント１１ アーキテクチャ・モデル

【ポイント 06 分析モデルで業務を理解】

分析モデルを作成することで、業務内容を視覚的に理解できるようにする。

【目的】

システム分析にモデルを使用することにより、業務内容の大枠を視覚的に理解できる。要求に精通している日本側で作成する。モデルの記述内容は日本語を使用するが、あまり長文を使用しないことが好ましい。モデルを使用することで、オフショア側プロジェクト・メンバの日本語のレベルがあまり高くない場合でも、情報の伝達が確実である。

【詳細・補足】

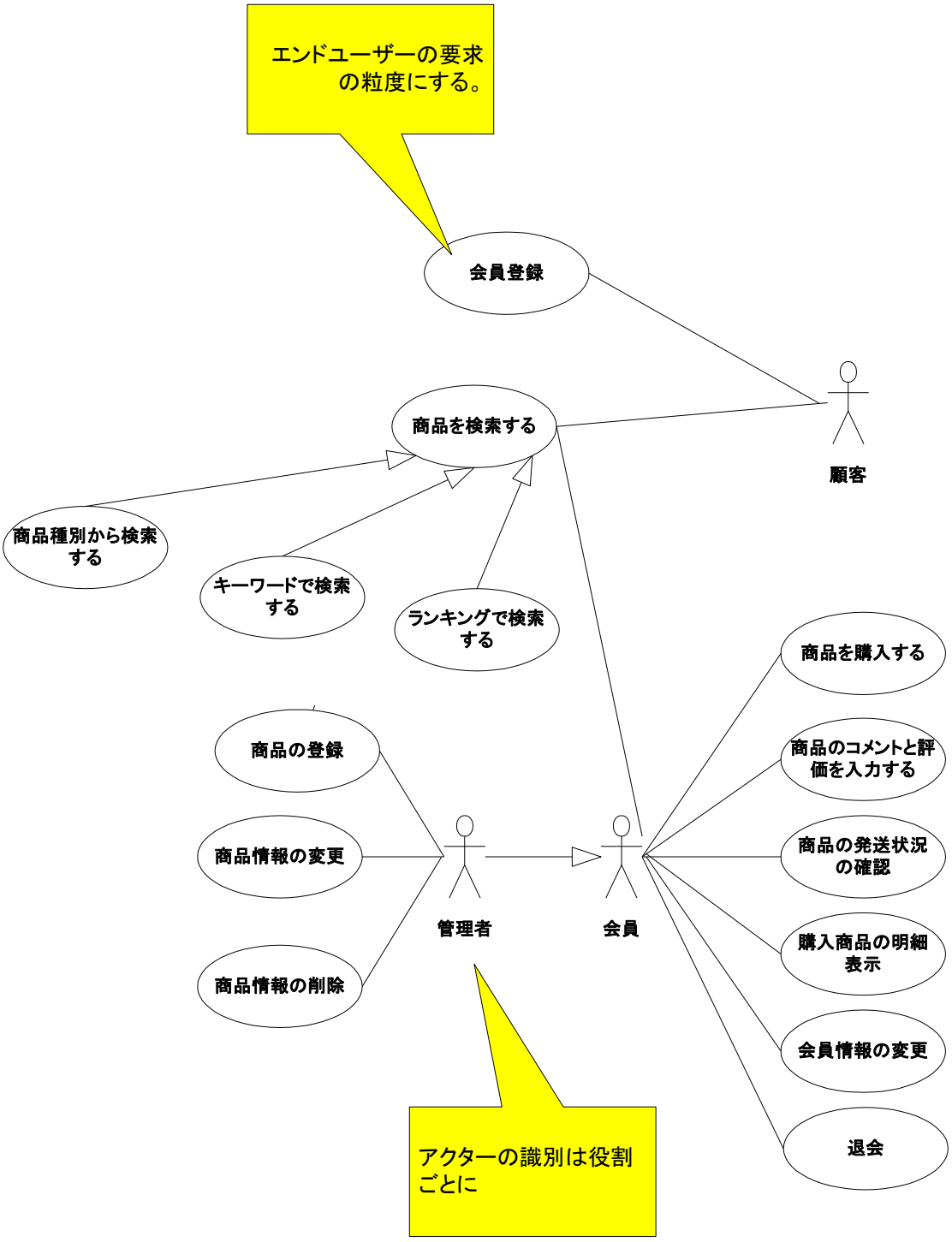
分析モデルはクラス図を中心に作成し、業務の内容を表現する。分析モデルは、実現方法を記述しない。矛盾があってはならないが、必ずしも詳細な内容を網羅する必要はない。共通理解ができるような情報をモデリングする。この時点で曖昧な点をなくすことは、それ以降の工程での手戻り削減につながる。逆にオフショア側で分析モデルの内容が理解できない場合、疑問がある場合には、早目に質問をすること。

【関連情報】

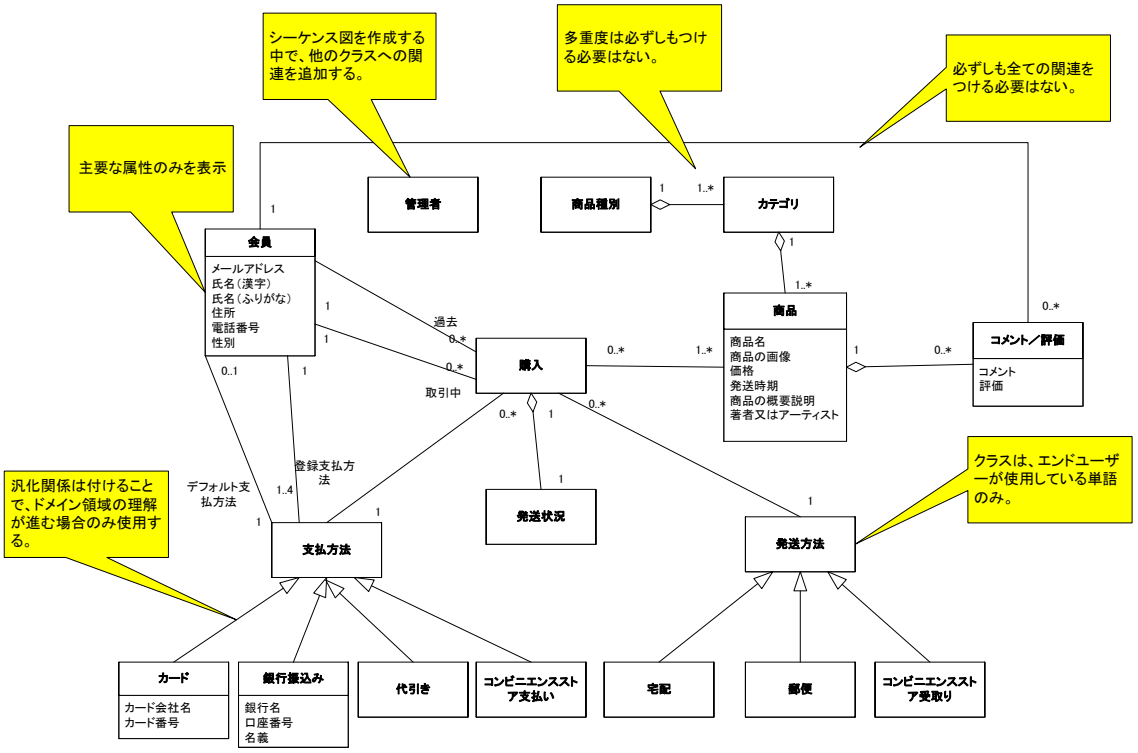
付録. 成果物サンプル （1）各工程の成果物 ③工程:システム分析

【業務を理解する分析モデルのサンプル】
ユースケース図や分析レベルのクラス図を使用することで、業務を理解する。

ユースケース図の例。



クラス図の例



＜注意事項＞

- エンドユーザー（業務）の視点で作成し、詳細になりすぎないようにする。
- ユースケース名は実際の業務名、クラス名、属性名は、実際に業務で使用している言葉にする。

【ポイント 07 用語辞書を作成する】

用語辞書を作成する。用語辞書は、業務用語と共通用語を分けた方が効果的である。
用語辞書は、日本語だけでなく、オフショア先の言語(中国の場合は、中国語又は英語)を入れた方がよい。
特に、業務用語は、オフショア先の言語を入れた方が、オフショア側プロジェクト・チームに正しく伝わる。
また、カタカナは使用せず、英語で記述するのがよい。

【目的】

業務内容の理解度向上と意識の統一を目的とする。
オフショア側プロジェクト・チームに対しては、ネーミング・ルールを明確にしておくことで、下記をねらう。

- ・日本側プロジェクト・チームから受け渡した設計書の理解度向上
- ・オフショア側プロジェクト・チームでの仕様書作成の向上
- ・受け入れ時のチェックに利用

【詳細・補足】

用語と内容説明を記述し、業務の理解度をあげるために使用する。
用語辞書のバージョン変更の連絡は明確にする。
なるべく日本側プロジェクト・チーム、オフショア側プロジェクト・チームでの辞書のバージョンをリアルタイムに近い形で統一すること。

【関連情報】

ポイント 08 命名規約を作成する
UMTP モデリング用語集
http://www.umtp-japan.org/modules/data4/index.php?id=2

【ポイント 08 命名規約を作成する】

命名規約(ネーミング・ルール)を作成する。少なくともシステム分析工程で、命名規約(ネーミング・ルール)の標準化を行う。

【目的】

開発で使用する各種名称の命名規則を標準化し、要求分析/システム分析工程の成果物の仕様理解と、以降の工程の成果物の効率的な作成が目的である。

これは、オフショア側プロジェクト・メンバにとって不慣れな言語で記述された設計書の読解と、以降の設計書の作成時に有用である。

さらに、オフショア側プロジェクト・メンバに対して、命名規約(ネーミング・ルール)を明確にしておくことで、下記をねらう。

・日本側プロジェクト・チームから受け渡した設計書の理解度向上

・オフショア側プロジェクト・チームでの仕様書作成の品質向上

・受け入れ時のチェックに利用

日本側プロジェクト・チームにとっても、日本語の誤用や説明不足がある設計書を理解し、レビューする上での助けになる。

【詳細・補足】

英語表記なのか日本語表記なのか、何桁なのか、各桁はどのように使い分けられるのかなど、事前に統一しておくとよい。また、英語と日本語の併記は効果的である。

命名規約(ネーミング・ルール)を決定しておく項目の候補としては

・サブシステム名/コンポーネント名、

・パッケージ名

・ユースケース名

・クラス名

・属性名/プロパティ名

・操作名/メソッド名

・イベント名

・メッセージ名

・状態名

・画面 ID、画面遷移 ID

など

【関連情報】

ポイント 07 用語辞書を作成する

【ポイント 09 モデルの作成規約を作成する】

ソースコードのコーディング規約などと同様に、モデル作成における規約（ルール）を作成する。

【目的】

モデル作成時、作成の規約（ルール）を作成しておくことで、プロジェクト・メンバの作成したモデルが同一基準で作成される。そのため保守性が向上する。

【詳細・補足】

UMLの要素（クラス、パッケージ、メッセージなど）を独自のルールを設けることにより、分類し、そのUML要素の意味をより明確にすることができる。

たとえばクラスを、BCE（Boundary、Control、Entity）に分けて、システム分析、アーキテクチャ設計、詳細設計を行なうことは、一般的に広く行われている。Boundary は画面や他システムを仲介するクラス、Control は流れを制御するクラス、Entity は現実にあるクラスを示している。

それ以外にも、「データの受け渡しに使用するクラス」、「データベースとのやり取りをするクラス」、「時間制約があるメッセージ」など、開発プロジェクトや企業単位の独自のルールを設けることで、プロジェクト・メンバ間の意思疎通を図りやすくする。

また、モデルに対する変更権限についても明確に定めておくと良い。例えば、オフショア側では private な操作は追加しても構わないが、public な操作を追加する時には日本側の承認が必要など。

必要に応じてこれらモデルの作成規約を設けるときに、UMLの拡張機能を利用することも考えられる。UMLでは、UMLを拡張する仕組みとしてUMLプロファイルを用意している。UMLプロファイルによるUMLの拡張は具体的には、ステレオタイプ、制約、メタ属性を用いる。

【関連情報】

ポイント 07 用語辞書を作成する
ポイント 08 命名規約を作成する

【ポイント 10 共通機能の明確化】

UML モデリング工程での共通機能化の検討を行う。
設計／実装時のオフショア側プロジェクト・チーム独自の共通化を事前に日本側プロジェクト・チームがアーキテクチャとして確立しておく。

【目的】

品質向上、開発効率の向上が目的である。
具体的には、オフショア側プロジェクト・メンバには、次の傾向があり、それを防止する必要がある。

- ・提供されたサンプルや最初に作成した開発物を繰り返しコピー＆ペーストし、 それを書き直して成果とする。
- ・コピー＆ペーストにより設計や実装がなされる結果、共通コンポーネント・共通機能・汎用モジュールとして考えられる類似の機能がいくつも作成されることになる。

【詳細・補足】

オフショア側プロジェクト・チームは内部での開発効率を向上させるために、サンプルを元に独自に共通化し、手順が確立後にコピーして作成していく場合が多い。
事前にアーキテクチャおよびガイドラインを日本側が提供し、日本側で早い段階で分析レビューを行う。
コンポーネントが適切に分割され、共通機能化が図られていることを確認するとよい。
ユーティリティと見なせる同じロジックが、複数のコンポーネントに散在することのないようにする。これには、

- ・作業工程の計画時に、共通機能化検討の作業項目を含めさせる
- ・レビュー内容に、共通機能化が行われているかのチェック項目を含めさせる

が考えられる。

【関連情報】

なし

【ポイント 11 アーキテクチャ・モデル】

「アーキテクチャ・モデル」とは、ハードウェアやソフトウェアの基本的な構造の設計のこと。アーキテクチャ・モデルをしっかりと作成し、内容を共有すること。

【目的】

ハードウェアとソフトウェア・アーキテクチャが顧客の業務的な要求を満足しているかどうか判断するためには、対象となる業務内容を十分理解しておく必要がある。このため、アーキテクチャ・モデルは、日本側で作成する。また、このアーキテクチャの変更は、システムに対して大きな設計変更・仕様変更を生じることになるため、長期的な視点に立ち、将来的なシステム拡張や変更を考慮してアーキテクチャを設計しておく必要がある。

アーキテクチャの設計をオフショア側に任せた場合には、将来のメンテナンスやシステム拡張を行う際に、発注側（日本側）のアーキテクチャへの理解が弱く、適切な変更・拡張の設計を行えないことにもつながる。（この点から、メンテナンスも含め、長期的にオフショアのある特定の企業に任せ続けるという選択肢は考えられる。この場合でも、内容について十分レビューし、日本側も把握しておく必要がある。）

【詳細・補足】

開発対象となるシステムのハードウェア、ソフトウェア・アーキテクチャは、要求に精通した日本側で作成すべきである。この時に考慮すべき事項としては、

・そのアーキテクチャは、機能面／非機能面で妥当であるか
・求められるパフォーマンスやキャパシティを、どう実現しようとしているか
・本番稼働環境／テスト実施環境からくる制約事項は、どう解決しようとしているか
・障害対策を含む運用上の要件を、どう実現しようとしているか
・セキュリティに関する要件を、どう実現しようとしているか
・開発に参加するメンバのスキルに、どう対応しているか

などが挙げられる。

そして、アーキテクチャ・モデルを日本側で検討、設計してから、オフショア側プロジェクト・メンバに成果としてそれを伝える。必要に応じて、「ハードウェア、ソフトウェア・アーキテクチャに従って開発を行う手順」などを記したガイドを作成するのもよい。

【関連情報】

ポイント 12 アーキテクチャ説明成果物の作成
付録. 成果物サンプル （1）各工程の成果物 ④工程：アーキテクチャ設計

【ポイント 12 アーキテクチャ説明成果物の作成】

システムのソフトウェア・アーキテクチャを説明する成果物を作成する。

【目的】

オフショア開発において、日本側での要求分析～アーキテクチャ設計からオフショア側での詳細設計へと開発の主体が移る際、開発対象システムのソフトウェア・アーキテクチャについて、日本とオフショア双方のプロジェクト・メンバが正しく同じように理解することが必要である。 開発対象システム全体がどのような考え方や構造に基づいて設計されているか、日本側プロジェクト・メンバからオフショア側プロジェクト・メンバへ効果的に伝える目的がある。

【詳細・補足】

プロセスへのインプットとして補足仕様書を用い、ソフトウェア・アーキテクチャを検討して、検討結果となるアーキテクチャを説明する成果物をアウトプットとして作成する。システム分析～アーキテクチャ設計レベルでのクラス図とシーケンス図、および、それらを補足して説明する資料群からなるのが、アーキテクチャ説明成果物である。

開発対象システムのためのハードウェア/ソフトウェア・アーキテクチャ、なぜそのように設計されたかの理由や判断ポイント、構造の狙いや目的を、主に機能要求面から各図に記述していく。 必要に応じて、コンポーネント図、パッケージ図、配置図も用いるとよい。 モデルを用いて図示することにより、日本語での説明記述であっても、より正確にオフショア側プロジェクト・メンバに伝えることができる。

すなわち、アーキテクチャ・モデル作成プロセスの関連成果物として、

・インプットとなる成果物: 非機能要件定義 （補足仕様書）

・アウトプットとなる成果物: アーキテクチャ説明

詳細設計ガイドライン

となる。

アーキテクチャモデル、アーキテクチャ説明成果物は、作るだけではダメで、使用しているかどうかの確認が必要である。

【関連情報】

ポイント 11 アーキテクチャ・モデル

付録. 成果物サンプル （2）アーキテクチャ説明書

【ポイント 13 パターンの活用】

アーキテクチャ設計において、あるいは、詳細設計において、アーキテクチャやデザイン上のよく知られた各種パターンを活用することが望ましい。

【目的】

アーキテクチャ設計におけるアーキテクチャの意図を伝える場合、あるいは、詳細設計におけるコンポーネントやクラスの責務や意図を伝える場合、プロジェクト・メンバの間によく知られるパターンを利用することにより、例え説明がなくても、その部分の目的・責務についての情報量が増えたことと同等の効果が得られ、双方の共通理解を促進する。

【詳細・補足】

ソフトウェア開発におけるデザインパターンは、UML との組み合わせで説明されることが多い。また、設計のノウハウがそれら各パターンに集約され、開発者間に浸透することで、パターンを再利用・適用することによる品質確保ばかりでなく、設計やレビュー時のコミュニケーションにおいても、パターンによる共通理解を得られる機会が少なくない。 アーキテクチャ設計におけるアーキテクチャレベルでのパターンとして、例えば MVC(Model, View, Control) などを適用することにより、詳細設計担当のプロジェクト・メンバにその意図を誤解なく伝えやすくなる。 一方、詳細設計におけるデザインレベルでのパターンであれば、例えば GoF（Gang of Four） デザインパターンなどを適用することにより、レビュー時にも設計各部の責務などが想像しやすく、チェックすべき箇所が絞られることで点検を要領よく進められることにつながる。 このようにプロジェクト・メンバ間の共通理解を促進するためのパターン利用は有効である。

オフショア開発にパターンの習得と利用は必須ではないが、以上の効果のため、開発対象やプロジェクト・メンバのスキルによっては、コミュニケーションをより円滑にできる手法のひとつとして、習得するとよい候補のひとつと考えられる。

【関連情報】

ポイント 06 分析モデルで業務を理解
ポイント 11 アーキテクチャ・モデル
ポイント 12 アーキテクチャ説明成果物の作成

【ポイント 14 仕様書の記述レベル、書式の指定】

オフショア側の成果物となる各仕様書について、UML の記述レベルや書式を指定する。

【目的】

UML に限らず、オフショア開発においては成果物サンプルを求められることが多い。オフショア側プロジェクト・メンバによる記述に関し、日本側からの指示やガイドの不足は、個々のオフショア側プロジェクト・メンバ間に記述詳細のばらつきや、書式のばらつきの問題を生じることになる。オフショア側では、プロジェクト・メンバ全員で記述レベルや書式を統一しようとする動きがされにくい背景もここにある。

オフショア側成果物をお客様に提出するのか、日本側プロジェクト・メンバが見るにとどまるのかは、このポイントを適用する上での考慮点になる。

【詳細・補足】

成果物に含まれる UML については、実際の書式に従って、日本側で求める記述詳細やモデル粒度で、オフショア側プロジェクト・メンバの参考にできるサンプルを作成し、ガイドの一部として提供する。 この時、その UML で何を表して欲しいのか、何が示されていればよいか、モデルの目的を指示できるのが望ましい。これにより、UML の詳細さ／正確さ／粒度が揃いやすくなる。

例外処理／エラー処理についても、記述して欲しいレベルで適切にサンプルに含めておく必要がある（当然追記されるであろうと期待し、サンプルに含めないでおくと、最終的な成果物にも含められてこない結果となりがちである）。

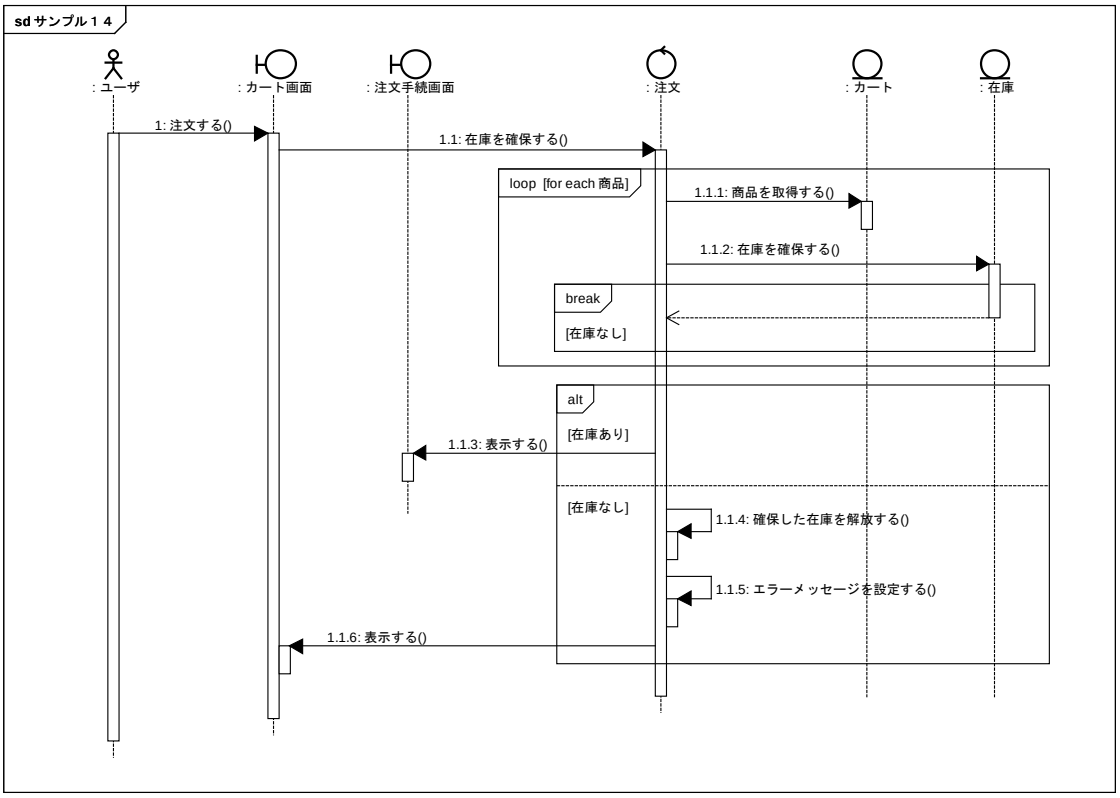
その他に、成果物の記述時の注意点として「あいまいな表現ではなく具体的な表現で記述すること」、「使用する用語について統一して使用すること」「カタカナの用語は英語で表記する」も合わせて指示・ガイドできるとよい。

【関連情報】

なし

【例外処理／エラー処理の例】

このシステムでは、商品をカートに入れたときに在庫の確保は行わず、「注文する」ボタンを押したときに在庫を確保する。カート中の商品全てに在庫があれば注文手続き画面へ進むが、1つでも在庫がなかった場合は、既に確保した在庫を開放した後、元の画面（カート画面）に戻りエラーメッセージを表示する。



・注意事項

① 特定の条件を満たす場合の処理（上記例では「在庫あり」）は記述するが、条件を満たさない場合の処理（上記例では「在庫なし」）は記述されないことがあるので、両者が必要であることをサンプルで明示しておく。

② エラー時の処理（上記例では「確保した在庫を解放する」）は、仕様書に記載がないと実装されないことがある。

③ エラー処理では何をするか具体的に記述する（上記例では、カート画面に戻る前にエラーメッセージを設定するなど）。

④ 実行環境や実装に依存した処理も必要に応じて追記する（エラーメッセージ ID を明記する、エラーメッセージの送信先オブジェクトの追加など）。

【ポイント 15 詳細設計ガイドライン作成】

アーキテクチャにそって、詳細設計を進めるガイドラインを作成する。

【目的】

アーキテクチャは、保守性、効率性、性能などを考慮して作成してある。
詳細設計では、そのアーキテクチャを踏襲することで、システム全体に対して、保守性、効率性、性能を維持することができる。
詳細設計の作業において、アーキテクチャから逸脱しないように、ガイドラインを作成することが重要である。

【詳細・補足】

アーキテクチャ説明においては、クラス図、シーケンス図などのモデルおよび、自然言語において、そのシステムのアーキテクチャの説明がなされている。
詳細設計では、そのアーキテクチャを、システム（モデル）の他の部分(機能)に適用する必要がある。

アーキテクチャ説明に記述されているのは、アーキテクチャの考え方だけである。
ガイドラインでは、どのようにモデリング・ツールを使用し、クラス図、シーケンス図をどこに作成し、クラス、オブジェクトをどこに配置していくか、などを具体的に説明する。
アーキテクチャ説明を全体の方針とすると、ガイドラインは具体的にブレイクダウンされたものになる。

【関連情報】

付録. 成果物サンプル (3)詳細設計ガイドライン

【ポイント 16 仕様未決定部分は明確に】

成果物における仕様未決部分は、オフショア側に明確に提示する。オフショア側も仕様が不明確な場合は、勝手に判断せず早めに確認をする。

【目的】

開発期間不足などの関係で、要求分析～システム分析～アーキテクチャ設計工程の、日本側作業での仕様未確定部分が多いまま、オフショア側に開発依頼をする場合も少なくない。その仕様未決部分が明確に示されていない成果物は、内容の整合性に欠け、オフショア側プロジェクト・メンバに多くの疑問・質問を引き起こすことになる。

【詳細・補足】

オフショア側プロジェクト・メンバに渡す成果物には、仕様未決（未決・仕様変更の可能性大）部分を明記して渡すことで、仕様吸収漏れを抑えることができる。

仕様未決（未決・仕様変更の可能性大）部分については、成果物 UML 内に、

・どの範囲が仕様未決であるのか、影響が及ぶのはどの範囲か

・現仕様にどのような変更が予想されるのか（今後さらに詳細化される方向性なのか、内容として異なる内容に置き換わる方向性なのか）

を、備考や注釈等として明記する。

日本側プロジェクト・メンバがこれら各項目を管理することで、仕様決定時・仕様変更発生時のオフショア側開発分への組み込み漏れ防止、仕様変更工数の低減を図ることができる。オフショア側との進捗状況確認会議においても、それら仕様未決事項の一覧を元に、反映状況更新の確認、仕様決定事項の確認ができるとよい。

【関連情報】

なし

【ポイント 17 オフショア側での成果物のレビュー実施】

オフショア側プロジェクト・メンバによるレビューを確実に実施する。

【目的】

成果物の内容レビューを日本側でも実施することを伝えると、オフショア側レビューが全く実施されない状態での成果物が届くことが起こり得る。「日本側でレビューするのだから、オフショアでもレビューを行うと二重レビューになってしまい、作業の無駄になる」との考え方がその背景にある。

【詳細・補足】

依頼する成果物に関し、オフショア側でレビューする対象の一覧、レビューでのチェック対象となるポイント、レビューでの合格基準について協議し、合意しておくとよい。 その際、オフショア側でのレビューの目的、日本側でのレビューの目的の違いも明確に理解してもらい、オフショア側でのレビューが確実に実施されるようにしておく。オフショア側は、必ずレビュー報告書を作成し、日本側は必要に応じてレビュー報告書をチェックする。

基本的なレビュー・ポイントとして;
・開発予定機能一覧にある機能がすべて網羅されているか
・前工程の設計内容に対応する仕様がすべて記述されているか
・指定の記述レベル、書式に従っているか
・用語は正しく統一された語が用いられているか
・開発予定機能の静的な構造が正しく伝わり、設計、記述されているか
・開発予定機能の動的な振舞い・挙動が正しく伝わり、設計、記述されているか
・開発予定機能の物理的な配置が正しく伝わり、設計、記述されているか
・クラスの責務に混乱はないか
・図間の整合性が確保されているか
・仕様変更が反映されているか
などが挙げられる。

【関連情報】

ポイント 18 日本側はチェックを繰り返し行なう

【ポイント 18 日本側はチェックを繰り返し行なう】

日本側では、開発中はオフショア作業開始直後からチェックを行う。

※なお、本ガイドラインでは成果物の厳密な確認・承認を行なう「レビュー」に対して、サンプリング的な確認や成果物だけでなく意識レベル・理解度も含めての確認という意味で「チェック」という用語を用いる。

【目的】

オフショア開発におけるオフショア側プロジェクト・メンバにとっては、やはり慣れない言語でのコミュニケーションとなることから、

・（日本側からは）オフショア側プロジェクト・メンバの理解の度合いがわかりにくい

・（オフショア側では）誤った理解で思い込んでしまう。関連する内容をお互いで確認することなく進めてしまい、誤解が解消されないことへの対処が目的である。

オフショア側プロジェクトに任せたままにしていると、成果物が出来ていなかったり、品質が著しく悪かったりすることがある。それらのチェックを確実に行なっておく必要がある。日本側で適切にチェックすることで、品質向上・開発工数低減につながる。日本側にチェックのための工数と人材を確保しておくことが必須。

【詳細・補足】

作業開始直後から、ウォークスルーなどにより日本側からもチェックを頻繁に実施し、

a）成果物の内容記述は適切か、

b）成果物の形式は標準に従っているか、

c）要件や設計内容の理解に誤解がないか

をチェックするとよい。 オフショア側でのレビュー内容を、日本側からも、途中段階で早期から繰り返し行うべきとも言える。日本側は、すべての成果物をチェックするのではなく、適切なタイミングで適切な成果物をチェックするようにする。

さらに、開発期間中を通して、お互いの理解や開発成果内容を、オフショア側プロジェクト・メンバ間で確かめるよう習慣づけられれば、さらに好ましい。

【関連情報】

ポイント 17 オフショア側での成果物のレビュー実施

【ポイント 19 Validation と Verification】

日本側の目的達成のための Validation（妥当性確認）と、オフショア側の実装内容確認のための Verification（検証）のふたつの観点からレビューを行うことが望ましい。

【目的】

オフショア開発においては明確に作業分担されることから、日本側では業務分析～システム分析の、主にお客様の要求実現に関心が集まり、オフショア側では詳細設計～単体テストの、主に日本側からの仕様実現に関心が集中する。 この視点の違いは、テスト範囲の漏れや要求機能の実現漏れを生みだす。日本側からの仕様に合わせた実装結果の検証だけでなく、実装がお客様の要求を満たしているかどうかの妥当性確認も行うことにより、これらの漏れを防止することにつながる。

【詳細・補足】

オフショア側は単体テストや結合テストにおいて、詳細設計あるいはアーキテクチャ設計とプログラムの実行結果とをつき合わせ、発注された仕様通りに正しく作られているかどうかの検証（Verification）に終始することが多い。 一方、日本側は、お客様のビジネス要求を満たす正しい実装が作られているかどうかも含めて確認（Validation）したいことが多い。

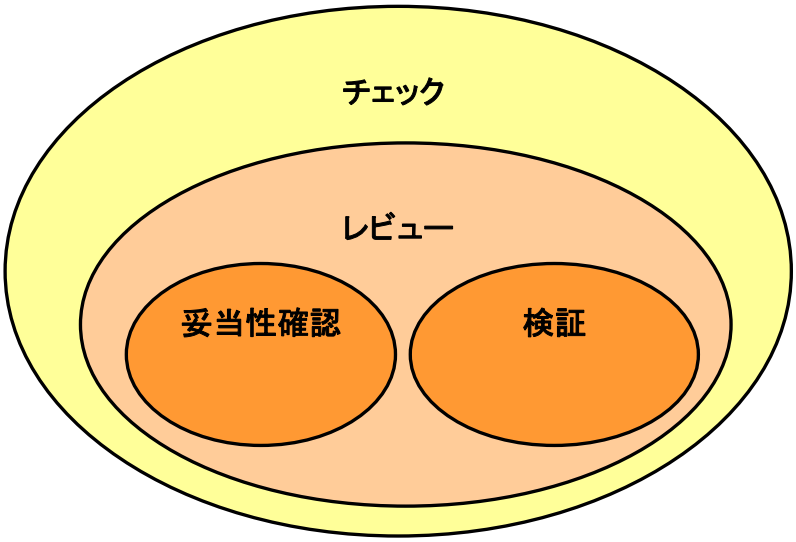
この視点の違いに注意し、オフショア側で行われがちな Verification 中心のレビューに加え、

- オフショア側にどのようなテストを実施して欲しいか、Validation 視点のテスト仕様として具体的に早期に明確に伝える
- オフショア側からの成果物受取り後、日本側で Validation 視点のレビューやテストを行えるように並行して準備する

という対応を行うことが望ましい。

例えばトレーサビリティ・マトリックスは、要求から実装までを結び付けることで、各要求はどの実装により満たされているかを確認することができる手法のひとつとして有効であるため、習得されているならば有効である。

なお、本ガイドラインにおいて、前出の「レビュー」「チェック」と本ポイントの「妥当性確認」「検証」とを、次の図のように位置づけて使用している。
成果物の承認を目的とするレビューは、妥当性確認と検証の2つの視点からレビューが望ましい、定義された成果物レビュー以外にも、離れた地点での開発において誤解なく円滑に進めるため各種の幅広いチェックが望ましい、と言い換えられる。



【関連情報】
ポイント 01　作業範囲／作業分担の明確化
ポイント 17　オフショア側での成果物のレビュー実施
ポイント 18　日本側はチェックを繰り返し行なう

【ポイント 20 モデルで詳細設計のレビュー】

詳細設計では、クラス図とシーケンス図、オブジェクト図、パッケージ図でレビューを行なう。

【目的】

実装を行なう前に、モデルによって方向性を確認する。ソースコードで、流れを追う必要はなく、モデル（シーケンス図など）で誰もが見やすい形でレビューを確実に行なうことが可能になる。
日本側でモデルにより詳細設計のレビューを行なうことで、実装の品質を向上させる。

【詳細・補足】

レビューは、通常の開発より、オフショア開発のほうがより確実に行なう必要がある。しかし、細かなチェック項目を作成しすぎて、日本側プロジェクト・メンバのレビュー工数が膨大になってしまいがちである。ここで、モデルの利用が有効である。

日本側プロジェクト・チームが作成した、アーキテクチャ説明、アーキテクチャのガイドラインに沿って、オフショア側プロジェクト・チームは、詳細設計を行い、成果物としてシーケンス図及びクラス図、オブジェクト図、パッケージ図を作成する。

日本側プロジェクト・チームは、オフショア側プロジェクト・チームが作成した、シーケンス図及びクラス図、オブジェクト図、パッケージ図のレビューを行なう。
レビューでは、アーキテクチャ説明を参照し、オフショア側プロジェクト・チームが作成したシーケンス図及びクラス図、オブジェクト図、パッケージ図がアーキテクチャから逸脱していないかのチェックを行なう。

【関連情報】

ポイント 09 モデルの作成規約の作成
ポイント 11 アーキテクチャ説明成果物の作成
ポイント 15 詳細設計ガイドラインの作成
ポイント 21 UML 図間の整合性

【ポイント 21 UML 図間の整合性】

オフショア側プロジェクト・メンバが設計した UML 図の間の整合性に注意する。

【目的】

UML に限らず、オフショア開発においては、成果物内容相互に整合性の欠落が見られることが多い。

日本側から成果物を渡す際の整合性の欠落は、オフショア側プロジェクト・メンバに多数の疑問・質問を引き起こす。日本側で成果物を受け取る際の整合性の欠落は、日本側プロジェクト・メンバが内容を理解するのに困難な状況となり、双方でレビューできない状況を引き起こす。読者に暗黙の了解を期待した成果物ではなく、必要な情報をすべて記載することも重要な点である。

【詳細・補足】

動的な振舞いの面で、

・クラス図/シーケンス図/コミュニケーション図の間

・シーケンス図/ステートマシン図/アクティビティ図の間

静的／物理的な構造の面で、

・クラス図/パッケージ図/オブジェクト図/コンポーネント図/配置図の間

など、関連する複数の図の間でお互いに対応する箇所があるはずの図の整合性は、日本側の詳細設計レビューにおいてチェックすべき対象となる。

また、それぞれの図において、同じものを表すのに異なる用語が使われていることがないかも、整合性の面からの点検すべき対象となる。

・あるシーケンスで動作すると記されたオブジェクトが、クラス図から読み取れない

・ある状態遷移を引き起こすトリガが、シーケンス図やアクティビティ図から読み取れない

・あるノードに配置されるはずのコンポーネントが、クラス図やコンポーネント図から読み取れない

といった整合性の欠落は、実装の誤りを生む。

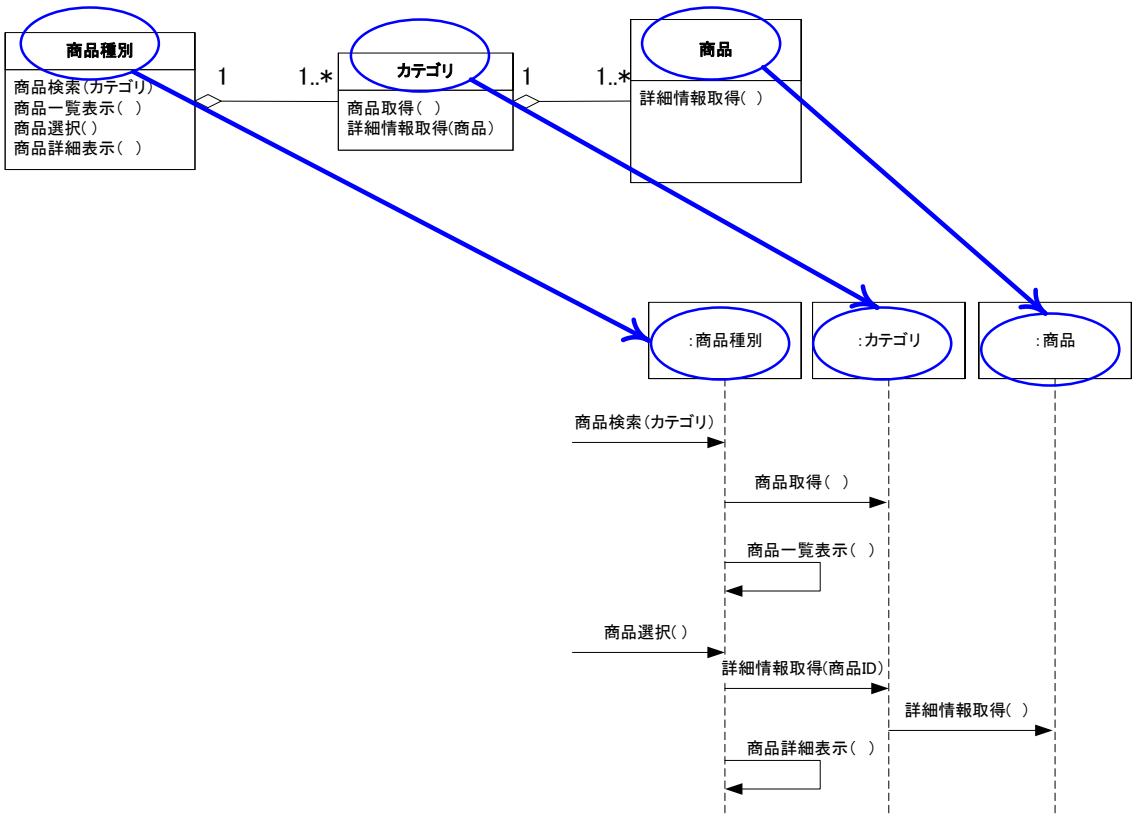
【関連情報】

なし

【UML 図間の整合性の例】

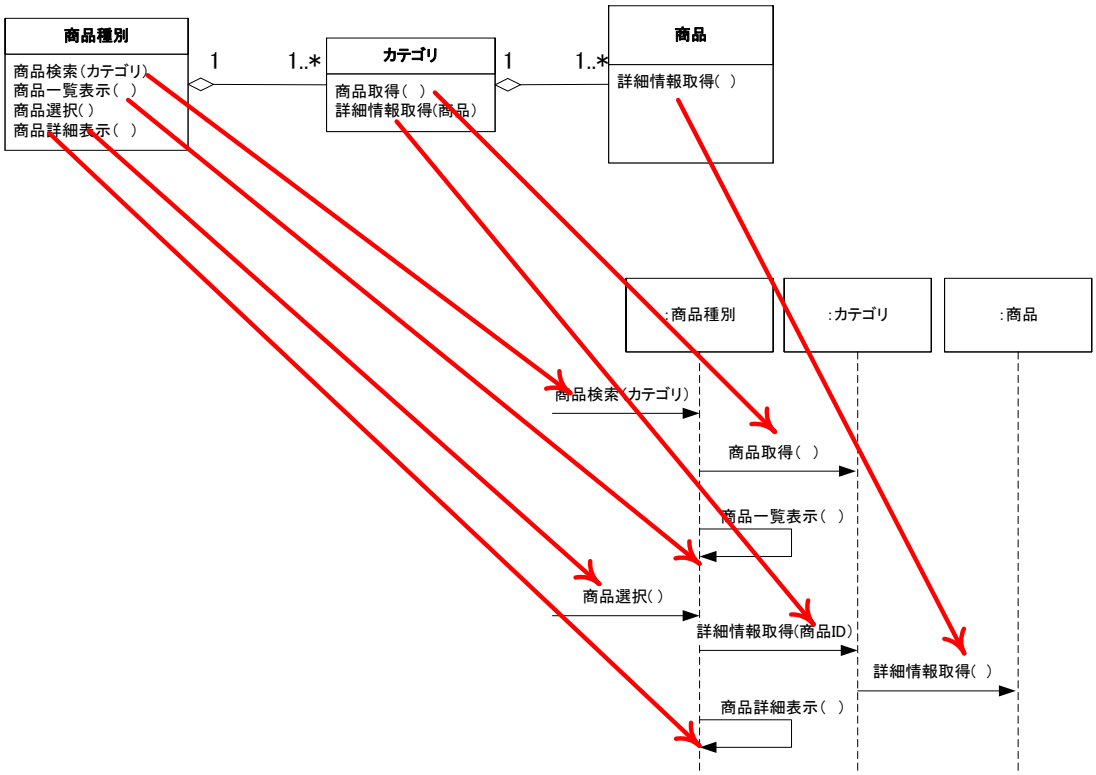
〇シーケンス図とクラス図の対応。
シーケンス図とクラス図においては、主に、次の3つの整合性に留意する必要がある。

例：商品種別で商品を検索する。
・ライフラインとクラス
シーケンス図のライフラインに対応するクラス図のクラスが定義されている必要がある。



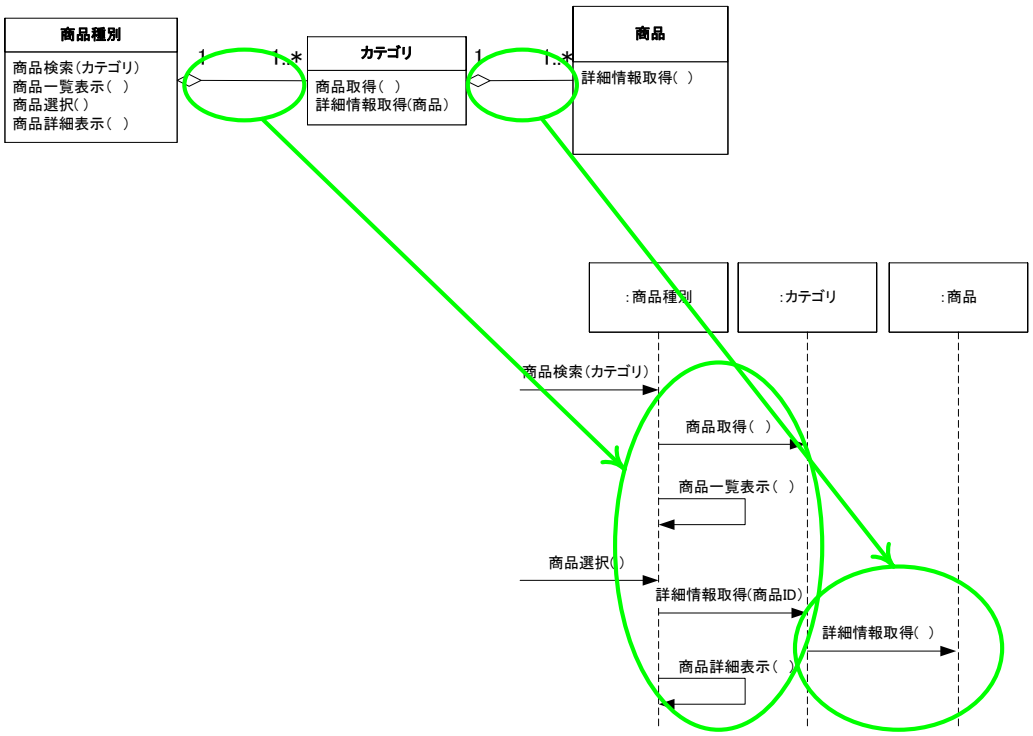
・メッセージと操作

シーケンス図においてメッセージが呼ばれている側のライフラインに対応するクラス図のクラスにその操作が定義されている必要がある。メッセージの呼び出しは操作の呼び出しにより実現する。

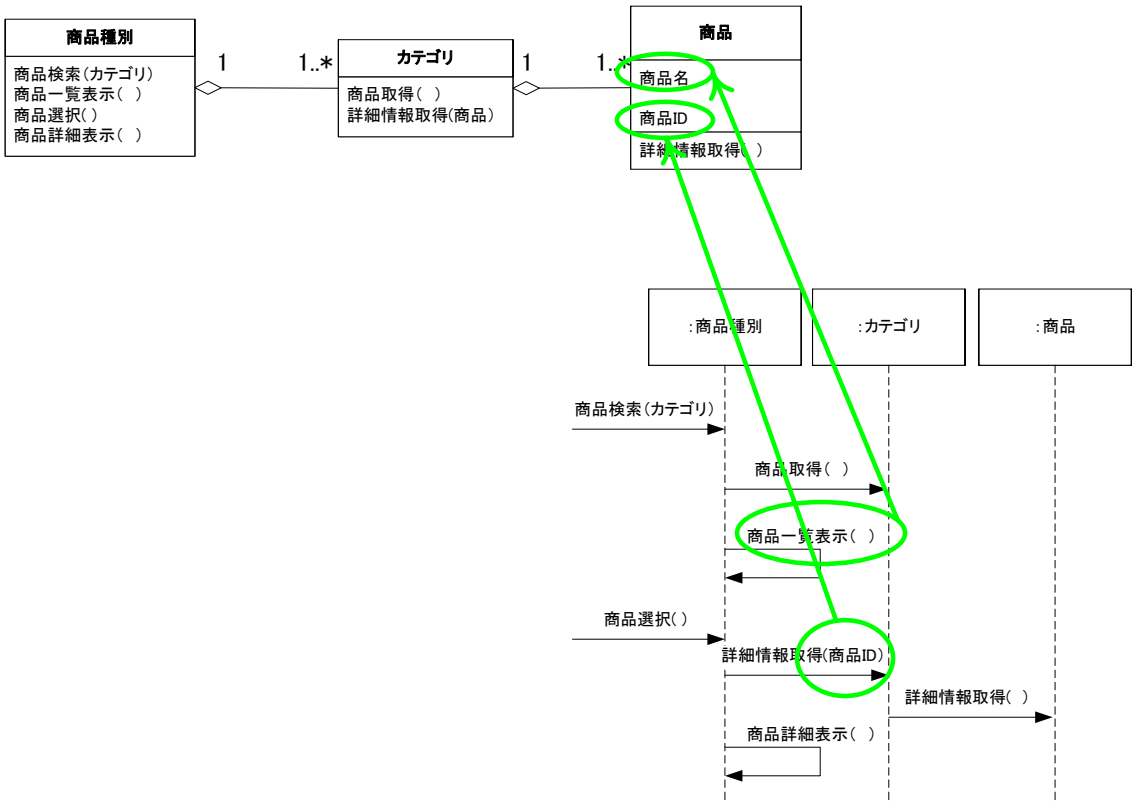


・メッセージと関係

シーケンス図において、メッセージの呼び出し関係があるライフラインに対応するクラス図のクラス間に、何らかの関係が定義されている必要がある。



※これ以外にも、シーケンス図におけるデータのやり取りから、クラス図の属性の必要性が発見される場合がある。



【ポイント 22 実装はツールのコード生成機能を使用する】

利用できるモデリング・ツールがあれば、実装時はツールのコード生成機能を使用して、ソースコードのテンプレートを生成し、それをもとにコーディングを行なう。必ずしもツールを使用する必要はないが、ツールを利用した方が効率的に信頼性の高いコードを記述できる。

【目的】

詳細設計において、オフショア側プロジェクト・チームが作成したクラス図及びシーケンス図などの成果物を日本側プロジェクト・チームがレビューする。
モデルにおいて問題がないことを確認されたレビュー後の(信頼性のある)モデルから、モデリング・ツールのソースコード自動生成機能を利用することで、ソースコードの信頼性も高める。

【詳細・補足】

クラスについては、モデリング・ツールのコード生成機能を利用することにより、ソースコードのスケルトンを生成することが可能になる。
これにより、クラス名、属性名、操作名などは、ソースコードに出力されるため、ソースコード内の内容を、モデル(クラス図)で確認することが可能になる。

処理の流れ(主にシーケンス図)については、一般的なモデリング・ツールではソースコードの生成を行なわないため、手作業でのチェック作業が必要になる(モデリング・ツールによっては、ソースコードの実行したものをリバースし、シーケンス図を作成できるものもある)。
モデリング・ツールのコード生成機能を利用することで、見やすくわかりやすいモデルのチェックにより、ソースコードの信頼性も高めることができる。

【関連情報】

なし

【補足　日本におけるシステム開発の特徴】

発注側（日本側）のシステム開発における特徴を明確にし、受注側（オフショア側）とのギャップを解消する。

【目的】

日本のシステム開発は、以下のような傾向がある。

・仕様変更を前提（当たり前）として、プロジェクトが進められる

・テストの網羅度や管理方法について厳密さを求められる

日本側のシステム開発の特徴を明確にすることにより、発注側／受注側双方の認識を合わせプロジェクトのトラブルの発生を軽減する。

【設計編】

仕様変更はプロジェクトの品質悪化に最も重大な要素であることを顧客、開発側（日本側とオフショア側）で共有することが重要である。

【詳細・補足】

仕様変更が発生する要因について以下に示す。

① 顧客の業界では一般的（当たり前）な決まりごと、慣習、特殊な条件における動作などについては記述されていないことがあり、テスト完了後の顧客確認時に発覚する。

② 日本の顧客は、操作方法や画面のレイアウトについて細部の要望を出す傾向がある。

③ 日本の顧客は、一度確定（合意）した仕様でも変更要望を出すことがある。

④ 仕様が確定していない段階でもオフショア側担当作業（詳細設計）の開始を依頼することがある。

【製造・テスト編】

【詳細・補足】

テストに関する要求事項について以下に示す。

① 全てのパスについてテストを要求される

条件のバリエーションや組み合わせ、全ての画面遷移、全てのコード（網羅率　100%）など

② エビデンス（テスト結果）について基本的には要求される

・処理パターンが明確になるようにエビデンスを要求される

・開発途中で品質に問題がある場合は、エビデンスを要求される

③ テスト項目の基準値が設定される

・ステップ数、画面項目数、DBテーブル数でテスト項目数が設定される

・正常、異常、境界の各ケースの比率が設定される

④ テスト中のバグ件数が設定される

・バグ件数が正常系、異常系それぞれの標準に満たない場合、再テストを要求される

・バグ件数が多い場合、見直しを要求される

⑤ テスト中の多くの管理資料を要求される

・バグ票、問題課題表、バグ発生曲線、テスト工程品質管理図など

付録 A. ポイントで使用のサンプルの事例説明

インターネットショッピング

A 社はインターネットでの商品販売システムを検討しています。

〇商品の検索方法
商品を検索するには、次の3つ方法があります。

①商品種別から検索する方法
②キーワードで検索する方法
③ランキング（売れている順）で検索する方法

①商品種別から選択する方法
商品には種別（本、雑誌、CD、DVD）があり、商品はそれぞれの種別ごとに、次のようなカテゴリ別で整理されています。

本のジャンル
・アニメ
・医学
・エンターテイメント
・音楽書
・科学技術
・学習参考書
・楽譜
・教育
・暮らし
・経済
・ゲーム
・工学
・語学
・児童書
・コンピュータ
・資格
・辞事典
・タレント写真集
・地理
・ビジネス
・歴史

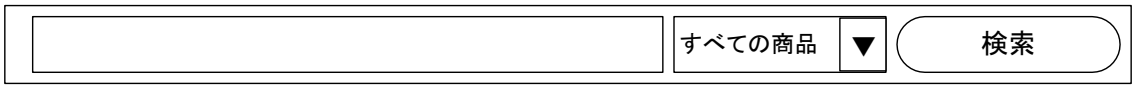
雑誌のジャンル
・アニメ
・音楽
・ゲーム
・結婚
・健康
・コミック
・コンピュータ
・情報誌
・クルマ
・バイク
・ビジネス
・文芸
・ライフスタイル
・料理
・旅行

CDのジャンル
・J-POP
・アニメ
・イージーリスニング
・演歌
・クラシック
・ジャズ
・ポップス
・ロック
・ゲーム音楽
・洋楽

DVDのジャンル
・外国映画
・日本映画
・テレビドラマ
・音楽
・アニメ
・お笑い
・バラエティ
・ホビー
・スポーツ
・アイドル
・アダルト

②キーワードで検索する方法

検索対象を選択し、キーワードを入力し、検索ボタンを押すと、該当する商品のリストが表示されます。



検索対象は次のものが選択できます。

・すべての商品
・本
・雑誌
・CD
・DVD

③ランキング（売れている順）で検索する方法

検索対象を選択して、“ランキング”ボタンを押すことで、売れている順に商品が表示されます。

検索対象は次のものが選択できます。

・すべての商品
・本
・雑誌
・CD
・DVD

〇商品の表示

商品を選択し、商品情報を表示すると、次のような情報が表示されます。

これは、全てのカテゴリに共通の情報です。カテゴリごとに、これ以外の情報が表示される場合があります。

・商品名
・商品の画像
・価格
・発送時期
・商品の概要説明
・著者又はアーティストの説明
・お客様のコメントと評価

〇会員登録

商品を購入したり、商品のコメントと評価を入力するには、会員登録をする必要があります。会員登録をするには、次の情報を会員登録画面に入力し、登録ボタンを押す必要があります。登録ボタンを押すと、仮登録の状態になります。システムは、入力したメールアドレスに確認のメールを送付します。メールにはURL が記述してあり、その URL の画面を表示し、確認ボタンを押すことで、正式登録がされます。

・メールアドレス（ログイン ID）
・氏名（漢字、ふりがな）

・住所
・電話番号
・性別
・デフォルトの支払い方法（カード、銀行振込、代引き、コンビニエンスストア支払い）
・ログインパスワード

正式登録されると、ログイン ID とパスワードでログインをすることができます。ログインすることで、次の機能を利用できます。

・商品購入
・商品のコメントと評価を入力
・商品の発送状況の確認
・購入商品の明細表示
・会員情報の変更
・退会

〇商品購入

商品購入をするには、事前にログインをして必要があります。
商品検索、表示し、“カートに入れる”ボタンを押します。ほしい商品を幾つでも、選択しカートに入れていきます。
“カートの中身を見る”ボタンを押すことで、カートに入れた商品の一覧を表示することができます。ここで、必要がない商品を削除したりすることができます。
“注文する”ボタンを押すことで、注文手続きに入ります。注文手続きでは、発送方法の選択（宅配、郵便、コンビニエンスストア受取り）し、支払い方法（カード、銀行振込、代引き、コンビニエンスストア支払い）を選択します。ここでは、会員登録時に入力してあったデフォルトの支払い方法が表示されます。異なる支払い方法を選択することもできますが、新たな支払方法に関する情報を入力する必要があります。入力した情報は、保存され次回に利用することができます。
最後に購入商品の明細、合計金額を確認し、“確認”ボタンを押すことで、購入が確定します。そのときに購入番号が表示され、またこの購入番号と購入商品の明細が記述されたメールが送付されます。
コンビニエンスストアで受取る場合は、この購入番号を伝えます。

〇商品のコメントと評価を入力

会員は、ログインすることで、商品に対するコメントと評価を入力することができます。この入力は、必ずしも自分が購入した商品でなくても入力することができます。
コメントは、３００字までで自由に入力をすることができます。
評価は１～５の五段階評価で、数字が大きいほど評価は高くなります。

〇商品の発送状況の確認

会員は、購入した商品の発送状況を確認することができます。
ログイン後、商品の発送状況の確認画面を表示し、確認したい商品の“購入番号”を選択することで、その商品の発送状況を確認できます。

準備中、受取店に発送中、受取店に到着、お渡し済み
準備中、発送中、お渡し済み

〇購入商品の明細表示
購入商品の明細を表示することができます。既にお渡し済みの過去の商品も含めて確認することができます。ログイン後、購入商品の明細表示の画面を表示し、明細を表示したい商品の購入番号を選択することで、その商品の明細を表示することができます。

〇会員情報の変更
既に登録していある会員情報を変更を変更することができます。ログイン後、会員情報の変更を選択し、会員情報変更画面を表示すると、既に登録してある会員情報が表示されます。必要に応じて、変更をし、“更新”ボタンを押すことで、会員情報が変更されます。

〇退会
ログイン後、退会を選択すると、確認画面が表示され“退会”を選択することで退会をすることができます。退会をすると会員情報は全て抹消され、ログインをすることが出来なくなり、会員としての全ての機能が利用不可能になります。

〇管理者の登録
非公開の画面で、管理者を登録することができます。管理者を登録するには、管理者 ID とパスワードを入力します。
管理者は会員と同等の機能が利用できるほか次の機能を利用することができます。

・商品の登録
・商品情報の変更
・商品情報の削除

〇商品の登録
管理者は、非公開画面で、ログインすることで、商品の登録をすることができます。
カテゴリを選択し、以下の情報を入力します。カテゴリによっては、以下の情報以外にも登録情報があるので、必要に応じて登録します。

・商品名
・商品の画像
・価格
・発送時期
・商品の概要説明
・著者又はアーティストの説明
・お客様のコメントと評価

〇商品情報の変更

管理者は、非公開画面で、ログインすることで、商品情報の変更をすることができます。

商品情報を変更するには、まず、商品選択をします。商品選択には次の方法があります。

- カテゴリを選択肢、そのカテゴリにある商品の一覧の中から目的の商品を選択する。
- 検索画面から商品を検索し、目的の商品を選択する。

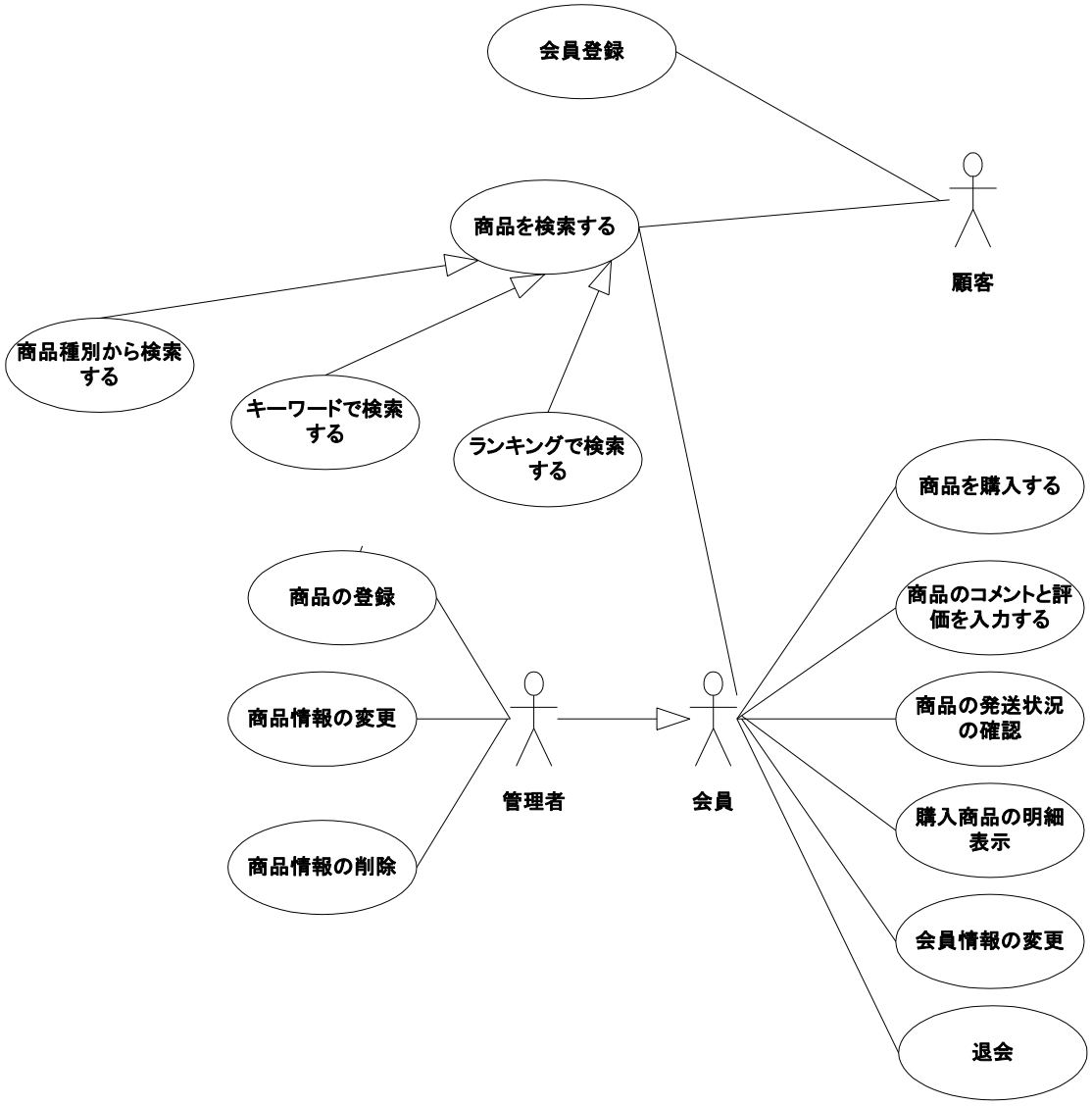
既に登録してある商品情報が表示されます。必要に応じて、変更をし、“更新”ボタンを押すことで、商品情報が変更されます。

〇商品情報の削除

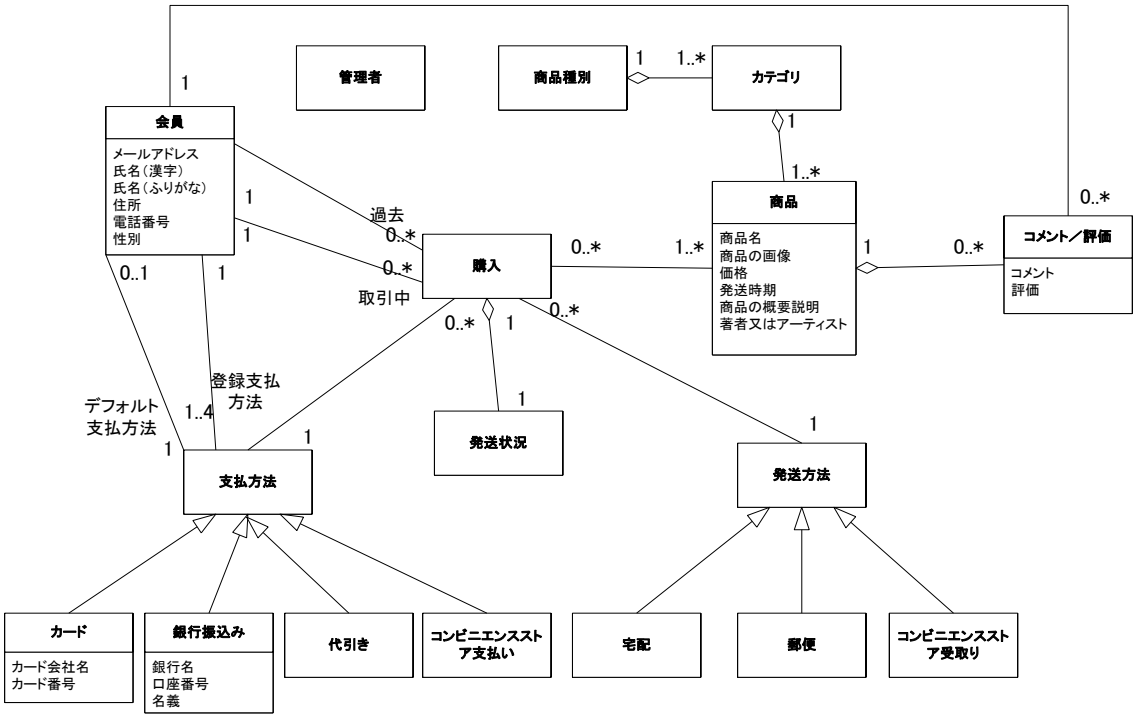
管理者は、非公開画面で、ログインすることで、商品情報の削除をすることができます。

商品情報の変更と同様に、商品を選択して、“削除”ボタンを押すと、確認画面を表示後商品情報が削除されます。

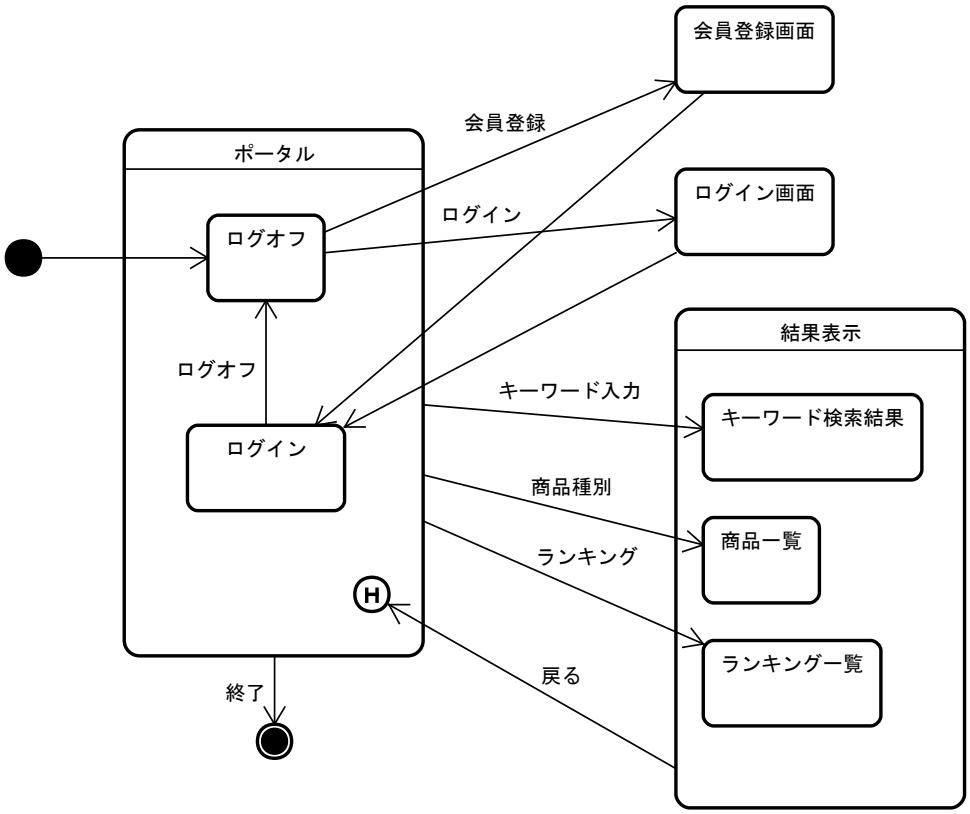
システム・ユースケース図



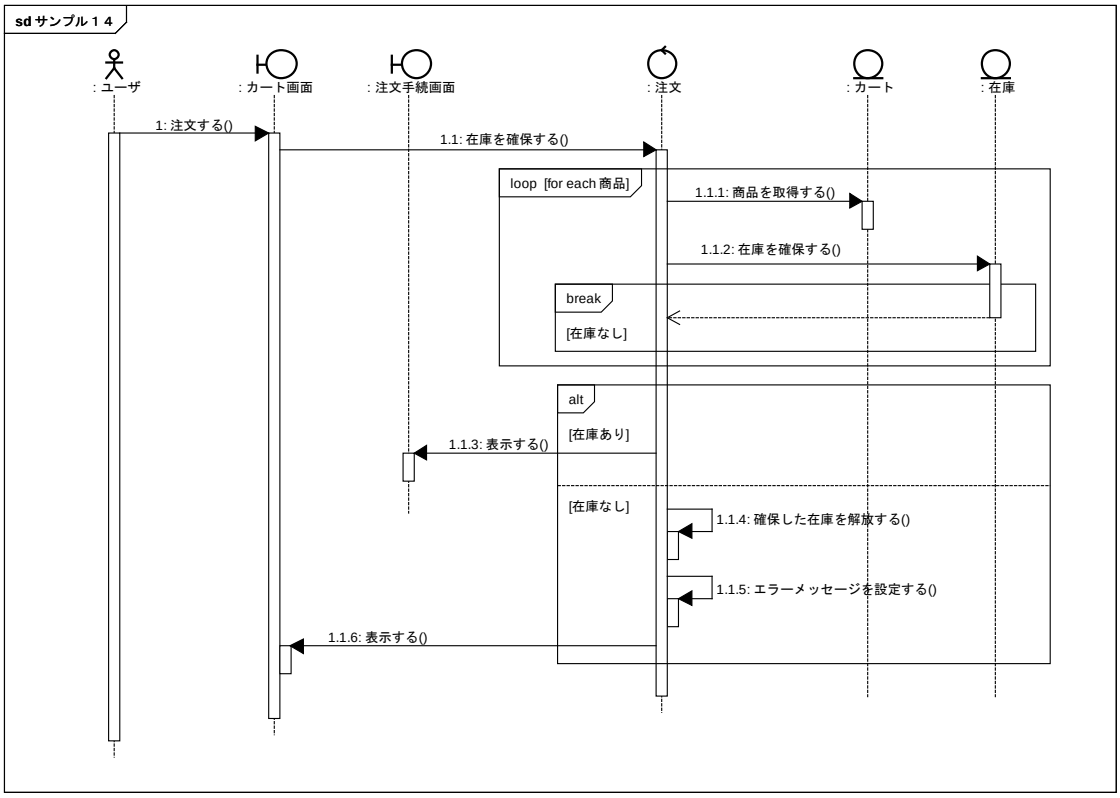
クラス図



ステートマシン図（画面遷移）　　※顧客、会員向け画面の一部



シーケンス図


sd サンプル１４
: ユーザ
: カート画面
: 注文手続画面
: 注文
: カート
: 在庫
1: 注文する()
1.1: 在庫を確保する()
loop [for each 商品]
1.1.1: 商品を取得する()
1.1.2: 在庫を確保する()
break
[在庫なし]
alt
[在庫あり]
1.1.3: 表示する()
[在庫なし]
1.1.4: 確保した在庫を解放する()
1.1.5: エラーメッセージを設定する()
1.1.6: 表示する()

付録 B. 成果物サンプル

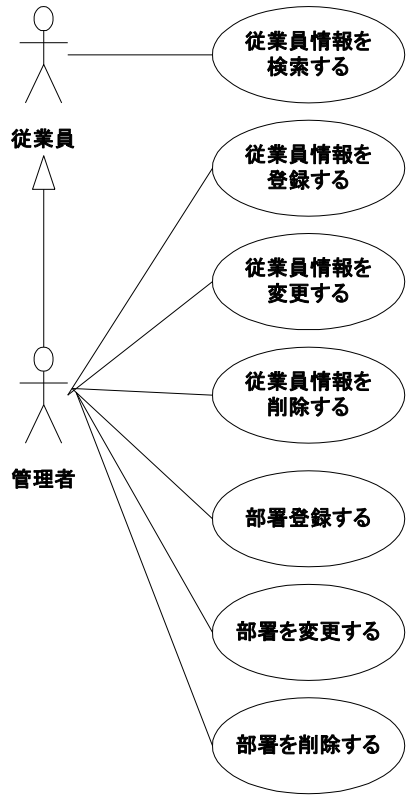
（1）各工程の成果物（一部分だけ）

①要求定義

A社では、従業員情報の検索システムの開発を検討しています。
従業員の部署、年齢、住所などのキーワードを入力して検索すると、それに適合する従業員情報を表示します。従業員ならば誰でもこの機能を使用することができます。
この会社の従業員は、技術職、営業職、事務職に分かれます。この会社には幾つかの部署があり、従業員はいずれかの部署に所属します。従業員の所属する部署は1つとは限らず、従業員は、複数の部署を兼務する場合もあります。
従業員情報は、あらかじめ管理者が登録しておきます。また、変更や削除をすることもできます。
従業員の所属する部署も管理者があらかじめ登録しておく必要があり、変更、削除ができます。

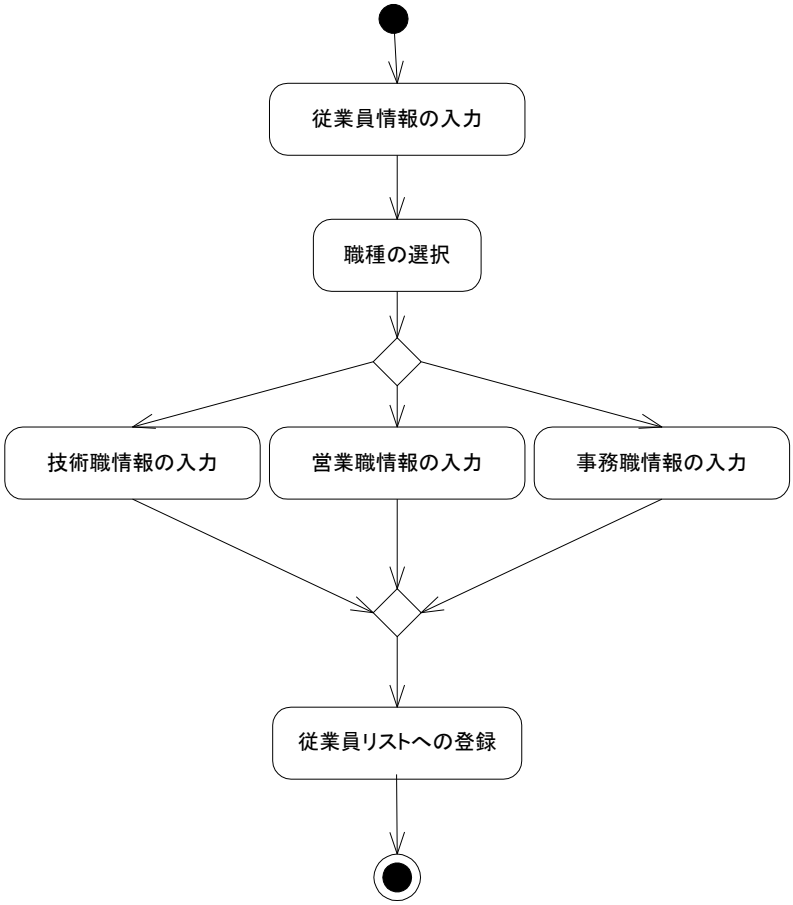
②工程：要求分析（日本側⇒オフショア側）

●ユースケース図

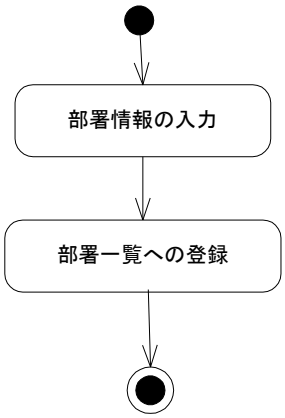


●ユースケースのアクテビティ図(イベントフロー)

ユースケース「従業員情報を登録する」のアクテビティ図(イベントフロー)



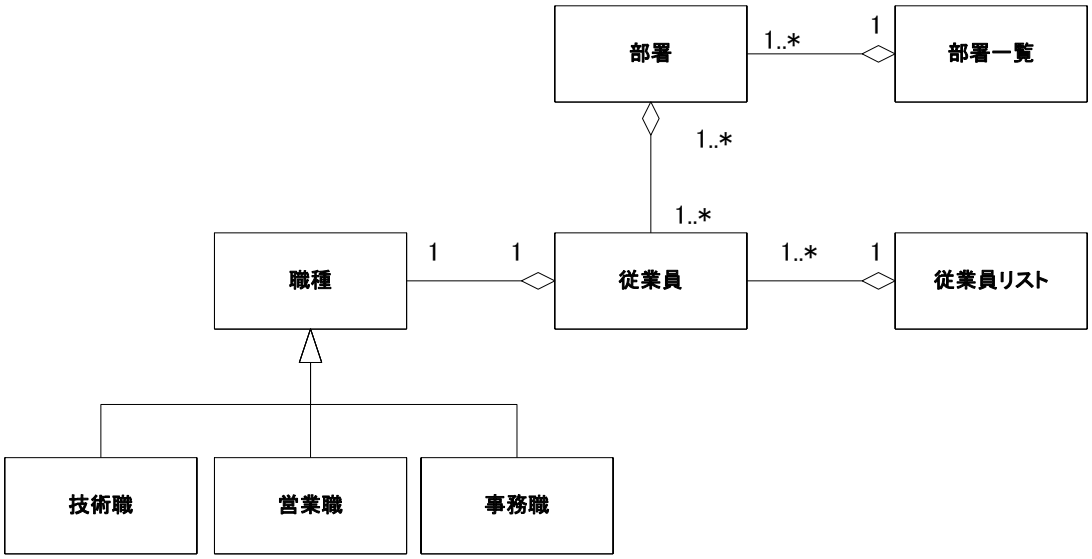
ユースケース「部署登録する」のアクテビティ図(イベントフロー)



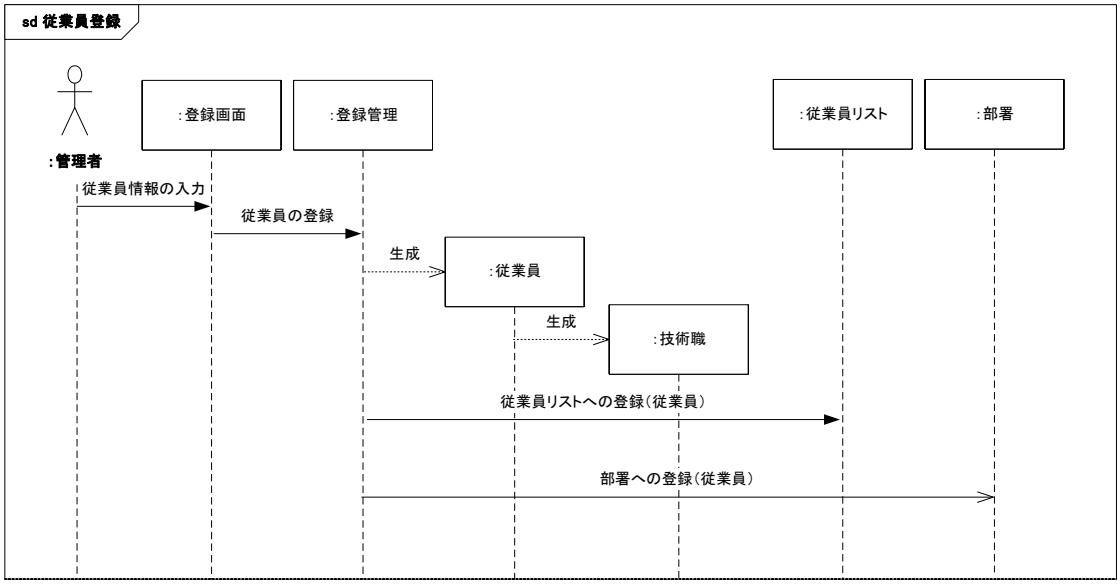
※その他、全てのユースケースに対して、このアクテビティ図(イベントフロー)を記述する。

③工程：システム分析（日本側⇒オフショア側）

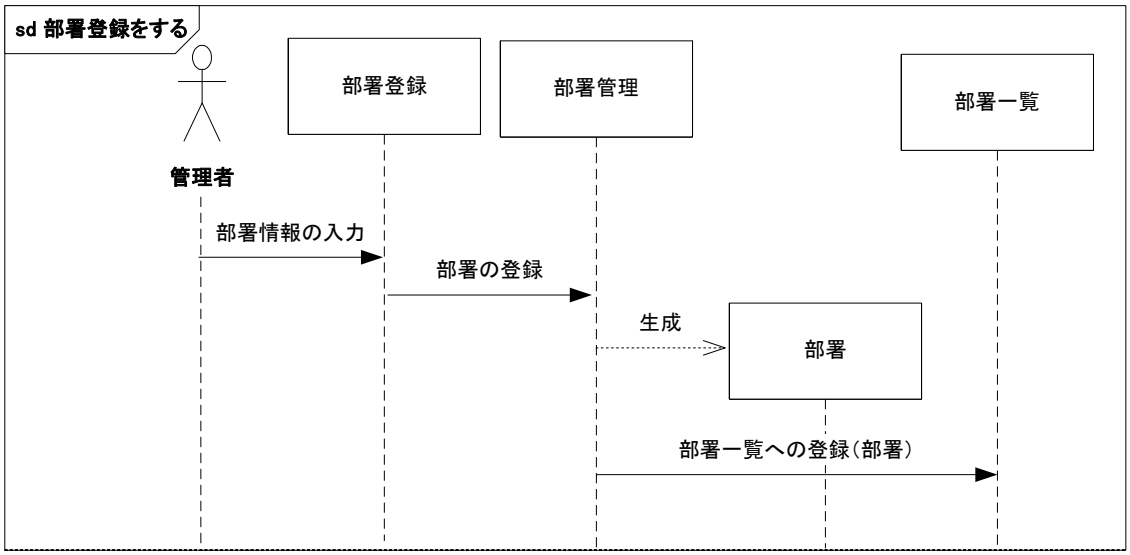
●クラス図



●「従業員情報を登録する」のシーケンス図

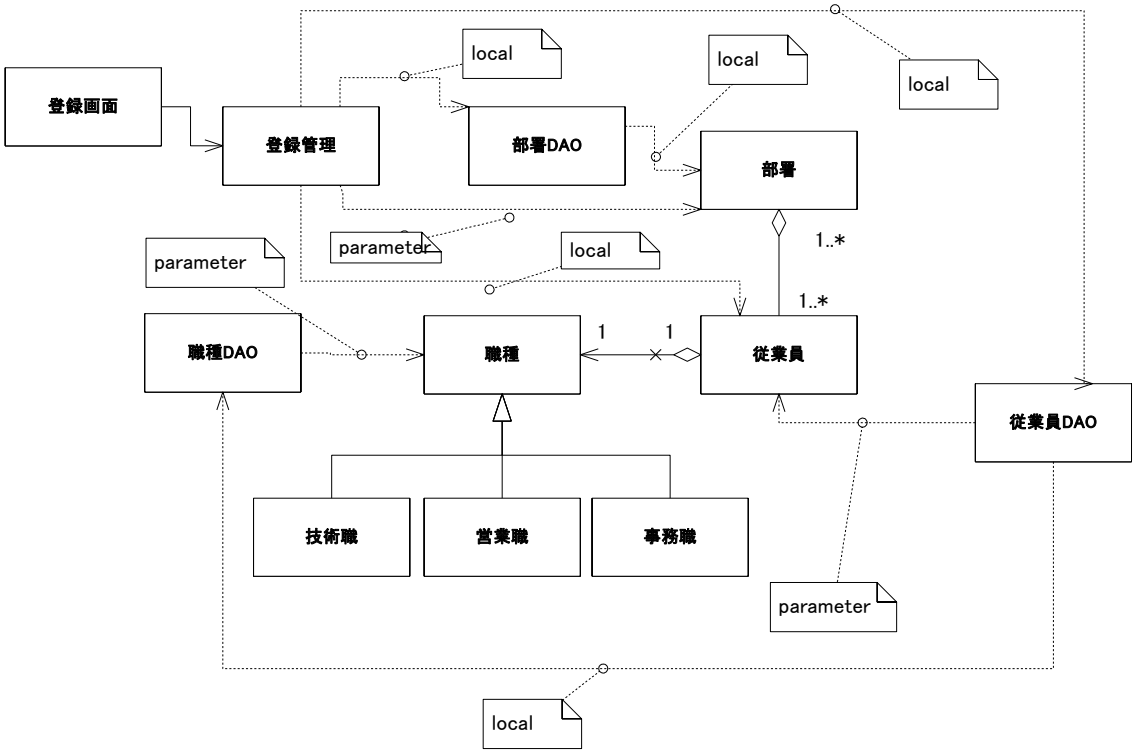


●「部署登録する」のシーケンス図

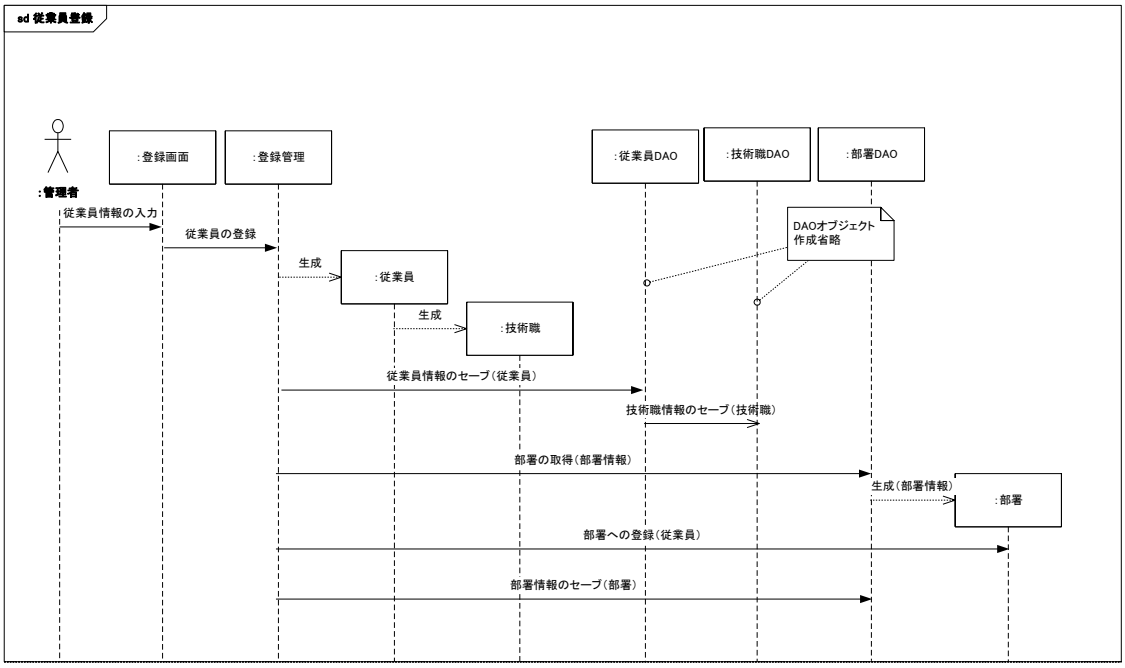


④工程：アーキテクチャ設計（日本側⇒オフショア側）

●クラス図

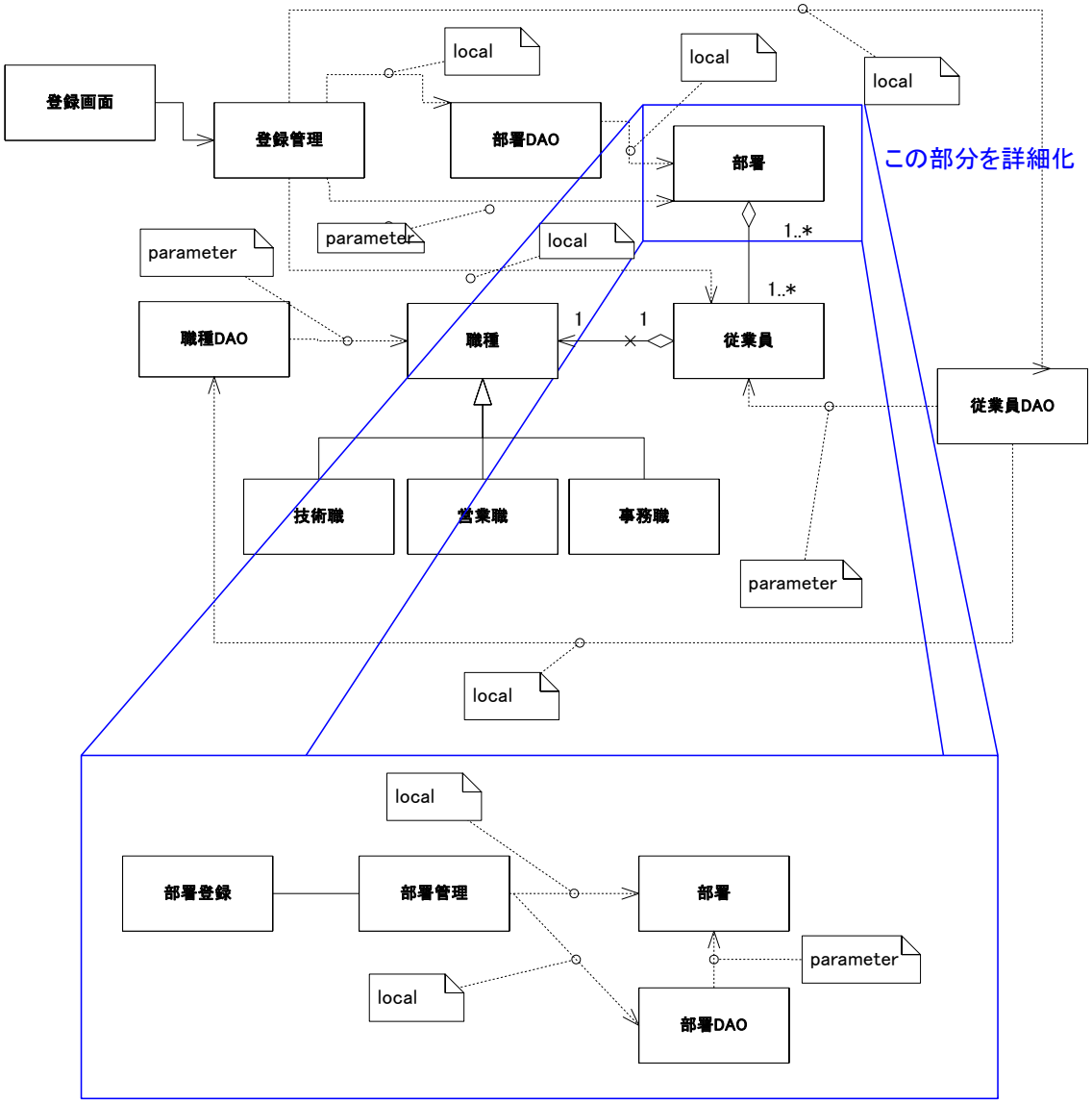


●「従業員情報登録」シーケンス図

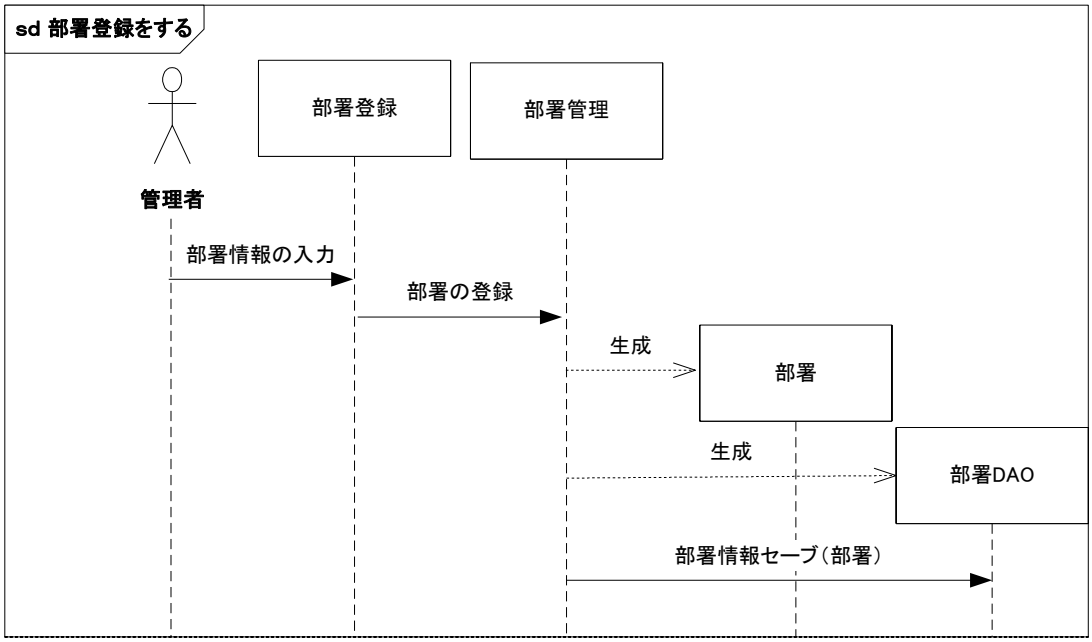


⑤工程：詳細設計（オフショア側⇒日本側）

●クラス図



●シーケンス図


sd 部署登録をする
部署登録
部署管理
管理者
部署情報の入力
部署の登録
生成
部署
生成
部署DAO
部署情報セーブ（部署）

（2）アーキテクチャ説明書

バージョン 0.1

〇永続化について（ユースケースビュー及び論理ビューの一部）

・Data Access Object (DAO)パターンの利用。

目的:

ビジネスロジックと永続化の実装を分離して、永続化の方法、データベースの違いを吸収する。

DAO の役割:

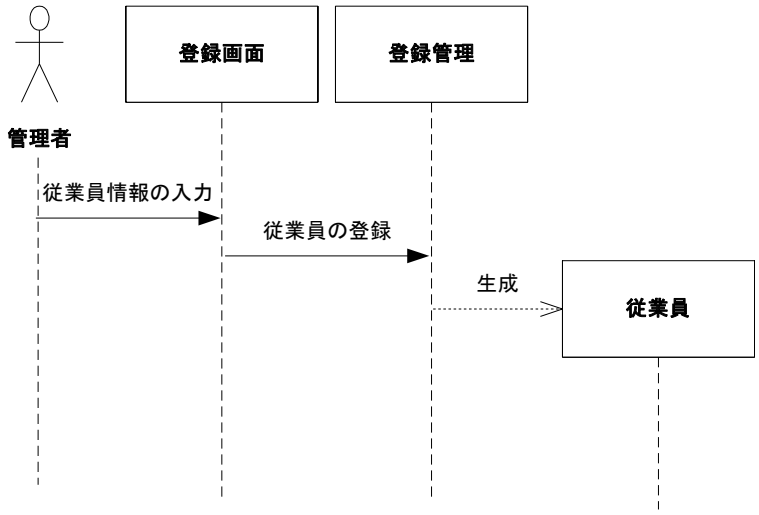
データの CRUD を DAO が受持つ

データベースのコネクションなどの処理を受持つ

・例　簡単な例（従業員登録）でシステム分析モデルに DAO パターンを導入する

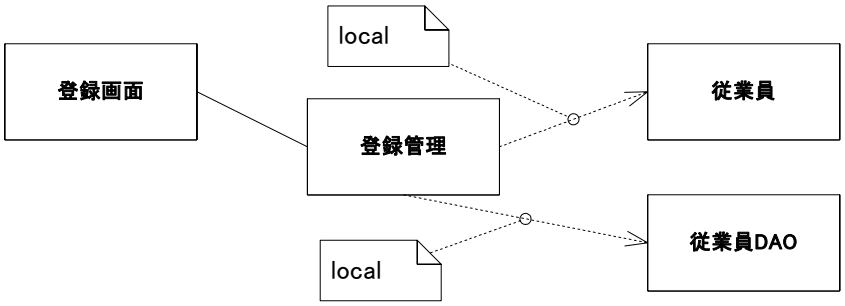
システム分析のクラス図

システム分析のシーケンス図



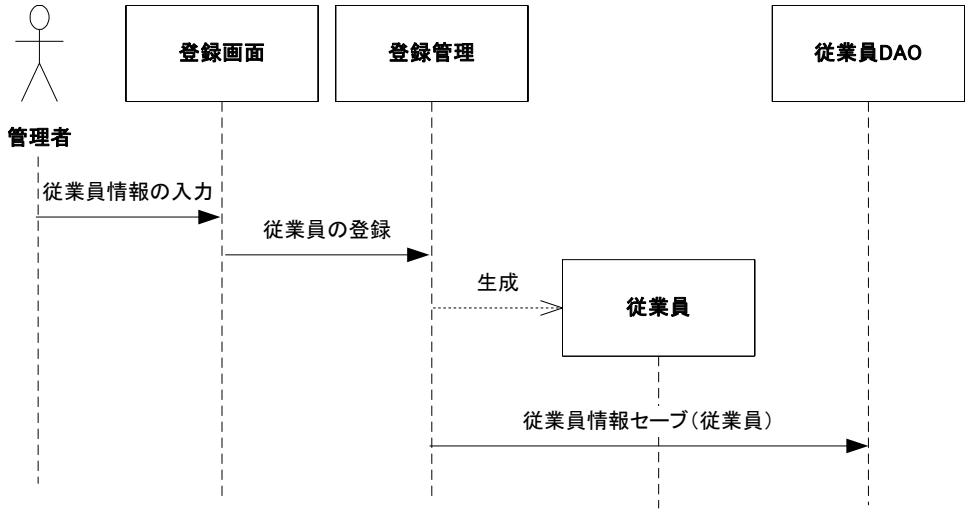
DAO パターンを導入した設計モデル

クラス図



※ local とは、依存関係元のクラスが持っており操作の中で、依存関係先のクラスのオブジェクトを生成することを示す。

シーケンス図



「従業員 DAO」クラスが、「従業員クラス」のデータベースへのデータのセーブ、ロードを受け持つ。

・
・
・

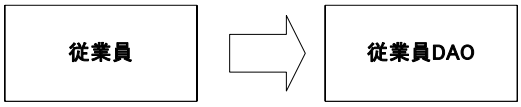
（3）詳細設計ガイドライン（一部）

設計クラスの追加

・永続化　Data Access Object (DAO)パターン

永続化を必要とする1つの entity クラスに対して、1つの DAO クラスを作成する。

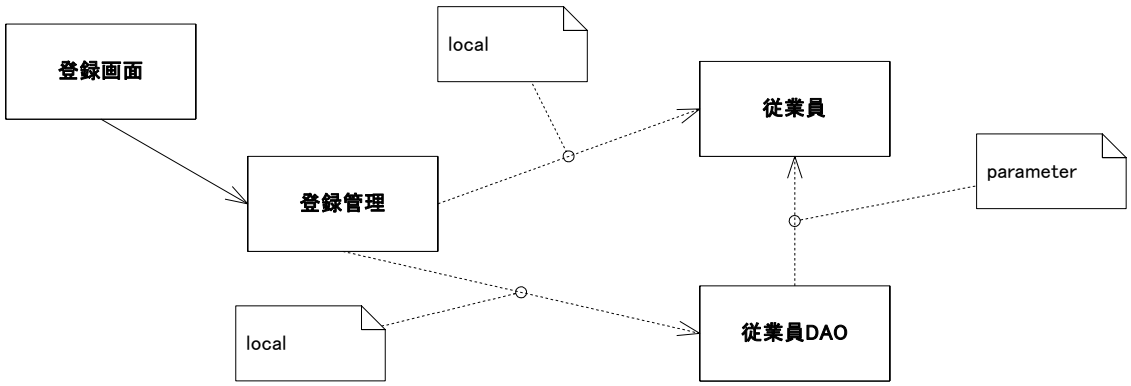
DAO クラス名は、entity クラス名 + ”DAO”　とする。



・クラス図

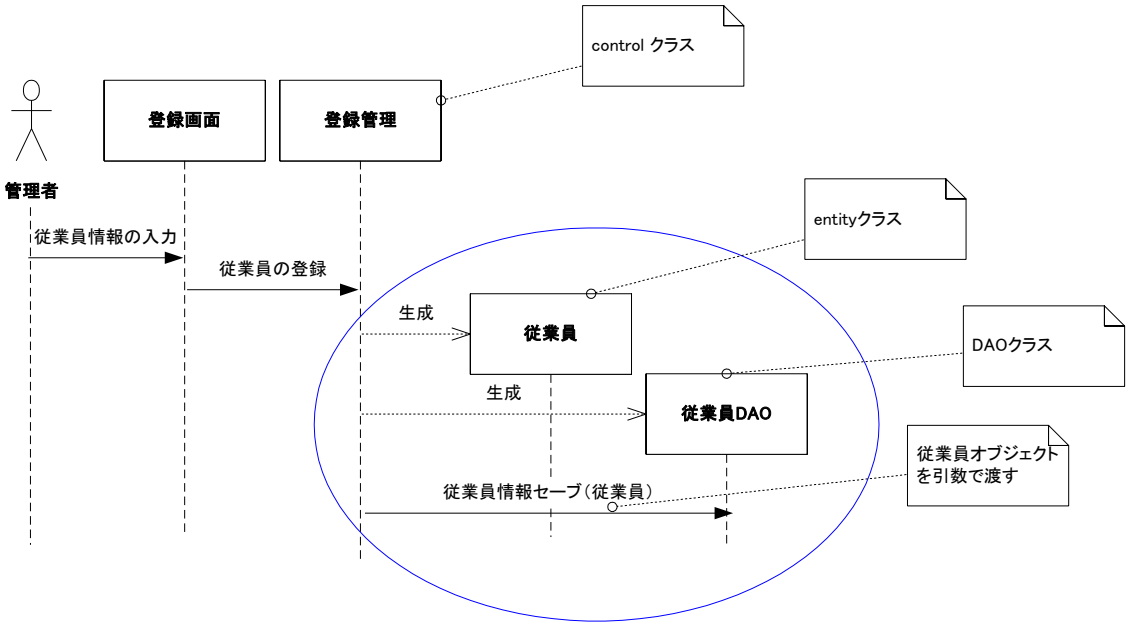
control クラスは、新規にローカルに、DAO クラスのオブジェクトを作成し、また entity クラスも新規にローカルに作成するので、依存関係を引き、「local」とコメントを入れる。

DAO　クラスは、entity　クラスを引数として渡されるので、依存関係を引いて、「parameter」とコメントを入れる。



シーケンス図

DAO クラスのオブジェクトは、control クラスがローカルに作成して、呼び出す。セーブする場合は、entity クラスを引数に渡す。



付録 C. モデリングツール Elapiz BE オフショア開発でのUML適用事例インタビュー報告

インタビューの内容

インタビュー日付:2009 年 6 月 23 日

インタビュアー:長田 真理亜（一部 中原俊政、竹政昭利、藤野博之）

インタビュイー:正田塁

開発の概要

まずプロジェクトの概要を教えてください。

対象は、Elapiz BE（イラピス ビジネスエディション）というツールです。

UMLとかビジネスモデリングをするためのツールですので、業務アプリケーションの開発とか組み込みの開発ではなくて、パッケージツールの開発が対象になります。

開発言語は C#で、2004 年の 8 月に開発をスタートし現在も継続中です。オフショア先は、中国上海です。特殊事情として、当社の関連会社を 2007 年 10 月に設立したため途中で委託先の会社が変わっています。ただ、主要メンバーは移籍してもらいました。

それでは、開発期間について教えてください。

開発バージョン	開発期間	体制（人数）		上海への委託範囲					
		日本	上海	外部設計	詳細設計	実装・単体テスト	結合テスト	システムテスト	ユーザマニュアル作成
v1.0	2004.12～2005.12	4	3→11	－	△	○	△	△	－
v1.1	2005.12～2006.8	6	11	△	○	○	○	△	－
v1.2	2006.8～2006.11	3	12	○	○	○	○	△	－
v1.3	2006.11～2007.2	2	14	○	○	○	○	○	△
v1.4	2007.2～2007.6	2	16	○	○	○	○	○	△
v1.5	2007.6～2007.8	2	16	○	○	○	○	○	○
v1.6	2007.8～2007.12	2	25	○	○	○	○	○	○
v1.7	2007.12～2008.2	2	25	○	○	○	○	○	○
v1.8	2008.2～2008.5	2	14	○	○	○	○	○	○
¦	¦								

バージョン 1.0 が 2004 年 8 月から 2005 年の年末ぐらいまでですね。8 月というのは日本での企画・計画の開始なのでオフショアは 12 月開始です。それから、翌年の夏 2006 年 8 月ぐらいまでがバージョン 1.1 です。その後はだいたい四半期に 1 回ぐらいのペースでマイナーバージョンアップしているという感じです。現在の最新バージョン 2.1 ですが、私が関わったのはバージョン 1.8 までです。

開発規模はどうでしたでしょうか。

当初は日本側と上海側が同じぐらいずつの 3～4 人の規模でスタートしています。開発が進むにつれ、上海側の人数が徐々に増えて 10 人余りに。バージョン 1.2 で日本側が急に減っているのは、私が上海に半年間常駐して上海側に技術移転を進めた時期だからです。機能を大幅に作りこんだバージョン 1.6 と 1.7 では、一時的に上海側の人数が増え 25 人体制となっています。

正田さんが上海に常駐をされている間、上海側の方々に上流工程も出来るように教えられたのでしょうか。

はい。常駐する前から上流を任せていくという方向性はありました。もちろん最初は、日本側でかなり多くの部分を担当し、実装とか単体テストとかだけを上海に任せていました。一部設計を任せるとか、テストの範囲も広げていくとか、徐々には常駐する前から準備は進めていました。それでも最初のバージョンは、それなりに動くものができるまでに 1 年ぐらいかかっています

最初の 1 年間の状況をもう少し教えて頂けないでしょうか。

機能的に最初のもの作りなので、土台となる部分を作らなくてはいけないという意味で開発ボリュームももちろん大きいのですが、UML を使用してオフショアに出すオーバーヘッドもあったと思います。試行錯誤の時間とか、教育コストも含まれて、1 年間という時間がかかってしまいました。モデルは読めるけど、十分には書けないという状態なので教育しながら進めていったという感じですね。

今は、上海側が UML を読めて書けるようになっているのでしょうか。

読めて書けるメンバーがいる、ということですね。全員ではないですけど。

最初は、ユースケース図とかなり詳細レベルの設計まで落としこんだクラス図と、そのクラス図を説明するための主要なシーケンス図も日本で書いていました。上海側には理解を確認する意味で、日本側のシーケンス図を参考にして上海側でもシーケンス図を作成してもらいました。なので、基本的には読めればいいというところからスタートしています。

その後、徐々に設計も上海側に任せましょうということで、設計レベルのクラス図やシーケンス図は上海側で作ってもらいました。最初の 1 年間はこれも完全に任せきれてはいなかったという状態ですね。

その後、画面や操作性などの外部設計やシステムテストも任せ、マニュアルとか日本語のヘルプも委託しています。日本側が最後まで握っているのは、要求定義です。つまり、どういう機能をどういう優先度で実現していくかは日本側で決定します。

現在は要求をするだけで、もの作りとテストを含めた残りすべてを上海側でできるようになっているということでしょうか。

はい、そうですね。自社のパッケージツールなので要求をコントロールできるという点と長期にわたって続けられるプロジェクトだったという点が、かなり広範囲にわたって上海に任せられた理由だと思います。お客様向けの

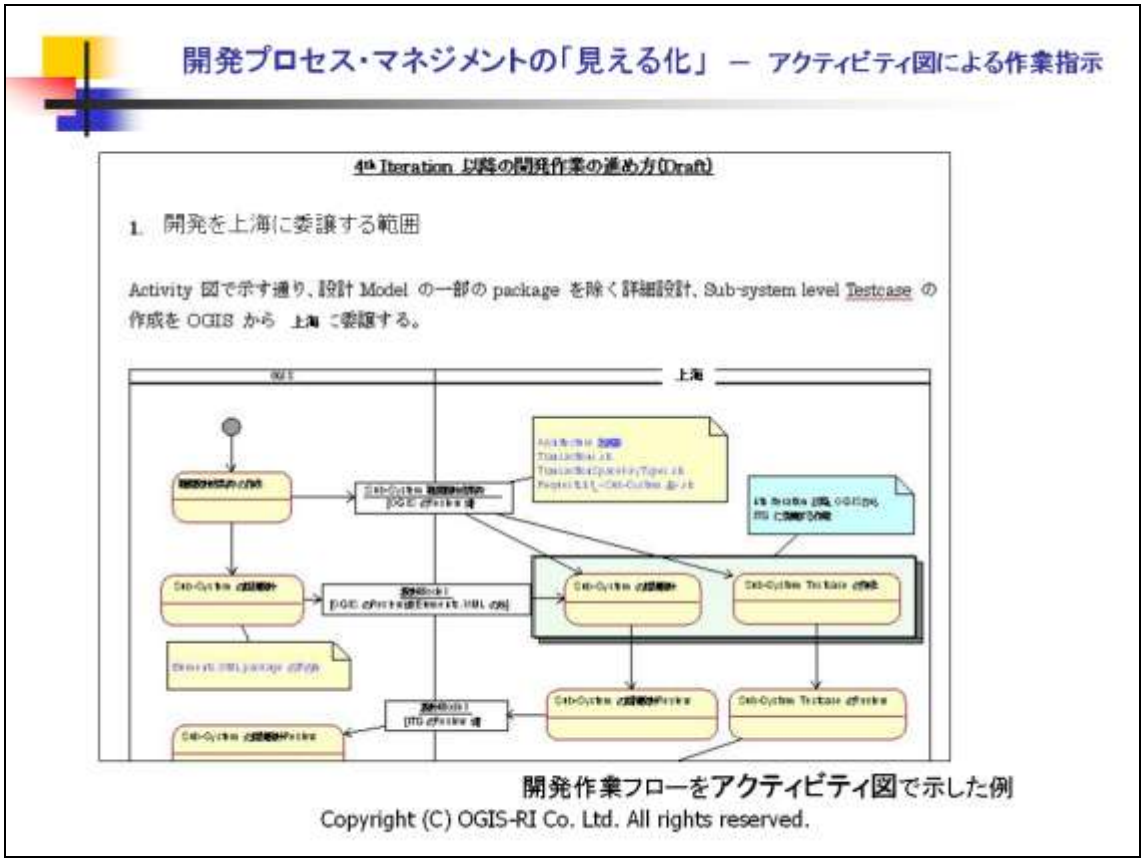
システムを3ヶ月間で仕上げて終わりといったプロジェクトでは、難しかったと思います。あとは、そうですね、会社は変わっていますけど、主要メンバーが変わっていないというのも大事な点かなぁ。

中国の社会全体で頻繁な転職が問題になっているようですが、主要なメンバーは変わらなかったのですね。

中国は離職率高いので、キーマンが一度にいなくなってしまうと、プロジェクトが長く続いていても、上流から任せるのは無理だったと思います。どちらが先か分からないですけど、他のオフショア企業に比べると上流の責任ある仕事を任せてもらえるというところが、モチベーション向上につながったのかなと思えるところもあります。

UML/ガイドラインの使用状況

UML 適用ガイドラインの使用状況について教えてください。



ガイドラインv2.0 の適用ポイントごとにいくつかトピックスをお話します。

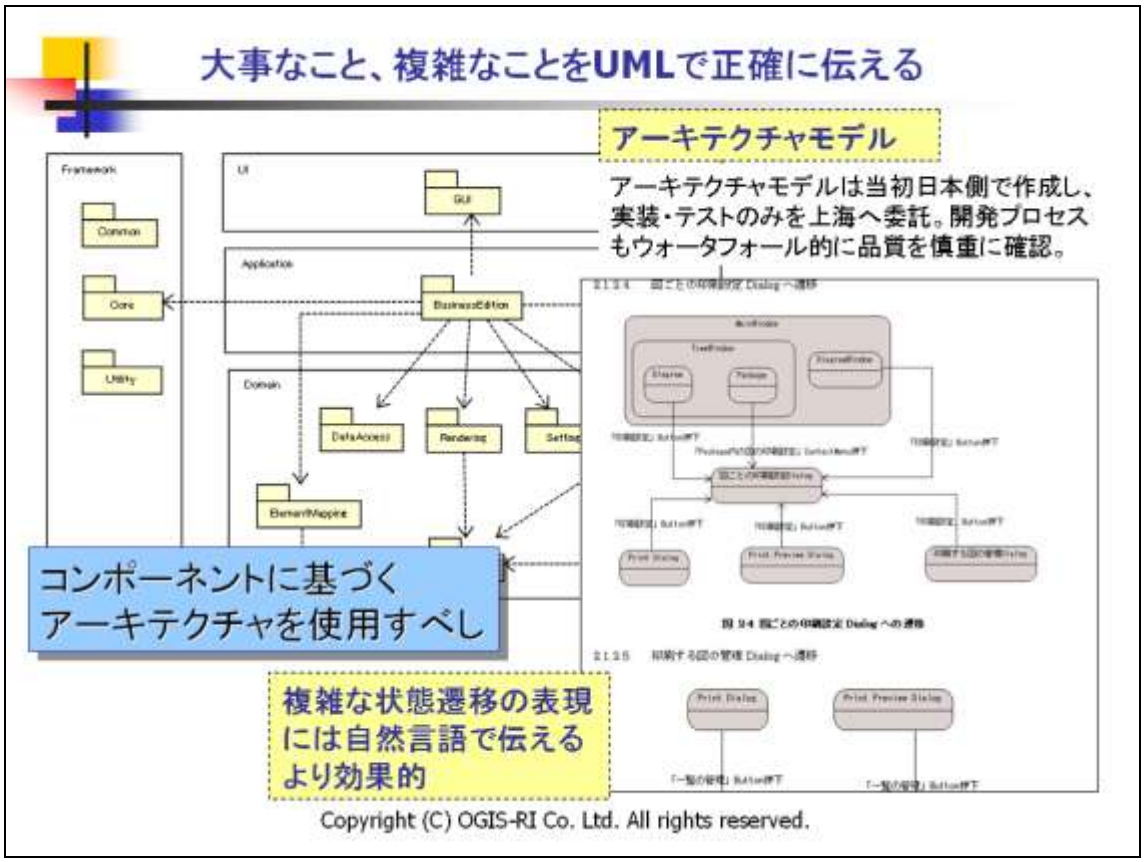
ポイント 1 番の「作業範囲/作業分担の明確化」ということでは、途中で 1 回混乱があって私が上海常駐したタイミングで、常駐しているがゆえに、日本側の立場なのか、上海側の立場なのか、お互いよく分からなくなって、責任分担があいまいになった時期がありました。ガイドラインを読み直したときに、「あれ、昔は明確化していたはずなのに、今はちょっとまずい状況になっているな」ということで、もう 1 回再確認したといった事象もありました。ちなみに役割分担は、アクティビティ図で定義していました。

ポイント 2 番の「利用する UML 図の確定」については、開発の概要でも説明しましたように、最初はクラス図・シーケンス図を日本側で書いて上海は読むだけ、徐々に上海が UML で書く範囲を広げていきました。その時々で日本側・上海側がどの図をどう使うかを明確にしていました。

ポイント 3 番「必ず UML である必要はない」に関しては、適材適所で UML を使用しています。要求定義以外は、すべて上海へ委託といいましたけれども、時々、リファレンス実装というか、実装のサンプルを日本側で作って上海側に渡すということもしています。リファレンス実装を渡すことによって、理解のギャップを埋めることができます。ドキュメントの中にはお絵描きツールで描いた画面も入っていますし、テストでは Excel の表を使っています。UML ではないですけど、「見える化」というか、視覚的に分かりやすいようなものをなるべくコミュニケーションツールとして使うように工夫しています。

ポイント 4 番「上流工程への参画」は、既にお話したように要求定義以外のかなりの部分を任せられており、本プロジェクトの大きな特長の1つです。

ポイント 11、12 番の「アーキテクチャ・モデル」「アーキテクチャ説明成果物の作成」にはエピソードがあります。開発スタート時に日本側で作成したアーキテクチャモデルの説明会も実施したので当初は上海側も認識していたはずなのです。それがいつの間にかお蔵入りになっていました。上海側のファイルサーバーの中には存在していたのですが、後からプロジェクトに加わったメンバーは存在すら知らない、ということが途中で発覚しました。反省点としては、最初に作っただけではダメで、それがきちんと使われていることを確認することが大事なのですね。



ポイント 16 番「仕様未決定部分は明確に」も失敗があって、日本と上海で同じ部分を作っていたことがあります。日本側としては全体像を理解してもらうつもりで日本側担当部分のクラス図も提供したのですが、意思疎通がうまくいっていなかったようです。その後は、UML 図上のパッケージの色分けで担当を区別するようにしました。

ポイント 17 番「オフショア側での成果物のレビュー実施」ですが、常駐して実際のレビュー状況を確認してみると、確かに形式的には時間も取って実施しているのですが、内容的にポイントをついてレビューをしていないということが分かりました。そこでレビュー用のチェックリストの作成などの改善をしてもらいました。

ポイント 20 番「モデルで詳細設計のレビュー」に関しては、モデルでレビューしているからこそ任せられる部分が増え、中身の実装の細かいところまで気にしなくてもよいというところがあります。レビューの観点の1つが、ポイント 21 番の「UML 図間の整合性」です。UML ツールを使っていればツールが整合性を保ってくれるので、UML の理解度が多少劣っていてもツールが補完してくれます。ツール利用のメリットですが、逆に上海側メンバーの実際の UML の理解度レベルに気付くのが遅れたというデメリットもありました。

［中原］ このプロジェクトが 2004 年からスタートして、ガイドラインが出来たのが 2007 年ですけが、ガイドラインを見て新たに追加したとか、追加してこういう効果があったとかはありますか。

正直目新しいことはないのですが、これだけきちんと網羅されていると、チェックリストとして有用だと思います。先ほど言ったように、スタート時点ではアーキテクチャモデルを作って説明したのにお蔵入りしているとか。作業範囲/作業分担の明確化も、忙しくなっていくとついつい忘れてしまって曖昧になっているとか。1 年に 1 回もしくは半年に 1 回ぐらいこのガイドラインでプロジェクトの状況をチェックすると、プロジェクトのその時々の弱点が浮き彫りになると思います。

［藤野］ 最初に 1 年ぐらいかけて教育されていますが、最初からラボ的なことまで意識して、育成計画を立てて教育されたのですか?

最初からそんな綿密な計画があったわけではないのですが、以前にインドのオフショアで痛い目にあった経験があり、ある程度上流まで任せていきたいという思いはありました。またツール開発ということで、相当コストを安く作らないと、ビジネス的に継続ができません。日本側で設計もやるし、作ったものの受け入れ試験もやるし、とやっていると、あまり日本側が楽じゃないというか、日本側がボトルネックになるわけです。受け入れ試験を綿密

にやっていると、次のバージョンのスペックが考えられないので、どうしても手抜きになっていきます。そうすると、中途半端なものが出来てきて、悪循環となってしまいます。中途半端に実装とテストだけ委託するというのは、日本側が苦しかったというのが正直な感想です。上海側にたくさん出さないと、オフショアを使っているメリットが感じられないという意識が現場にもありました。

[藤野] UML の教育にはどれぐらいの時間をかけたのですか。

UML の教育は、実は体系的にはほとんどやっていないです。たまたま UML ツールを作っているということで、副次的に理解している部分はあるかとは思うのですが、たとえば 1 ヶ月間教育するとかいうようなことは、当初はしていない。すればよかったかなとは、今では思うのですが。

[竹政] UML の図を段階的に理解してもらっているようですが、どこまで理解していれば任せられるとか目安がありますか。

実装、テストだけであれば読めればいいので、いわゆる L1 とかいうレベルで、逆に UML よりも実装能力の方が問われるのかなとは思います。しかし、一部と言えども設計を任せるあたりからは、1 人でもいいので、オフショア先にモデルをきちんと書ける人がいないと厳しいと思います。

[竹政] 体系的な教育というよりも実際にモデルを書いてもらい指摘しながら育成してきたということですね。

そうですね。でも、それは結構特殊事情だと思います。ある程度書ける人が最初からいて欲しいとは思います。英語とそれこそ UML でコミュニケーションし、試行錯誤で苦労しました。

[藤野] 最初は UML 適用ガイドラインもまだなかったわけですが、最初からガイドラインがあれば、立ち上がりは変わったでしょうか。

多少は、変わったかもしれないですね。

[中原] UML を使った効果が出だしたのはどの頃だったのですか。

実感として、効果が出始めたのは 1 年半とか 2 年ぐらいしてからですかね。もうちょっと上海側に委託する範囲を広げてもいいかなと思って実行すると、問題が起きて、また日本側に作業を戻したり、試行錯誤をしているので、1、2 年ぐらい経ってから回り始めたかな、という感じですかね。

[竹政] 設計から分析を任せるところのギャップというか、そこはどんな感じでしたか?

分析を任せる前に、外部仕様の一部を任せましたね。日本語ができるテスターのエンジニアに 3 ヶ月間東京に来てもらって、お客さん側の視点に立って外部仕様を一部作ってもらいました。そのテスターが上海に帰ってから分析を任せたので、ある程度上海側でどういうゴールになるかを理解したところで、分析をスタートしているのでスムーズにいきました。

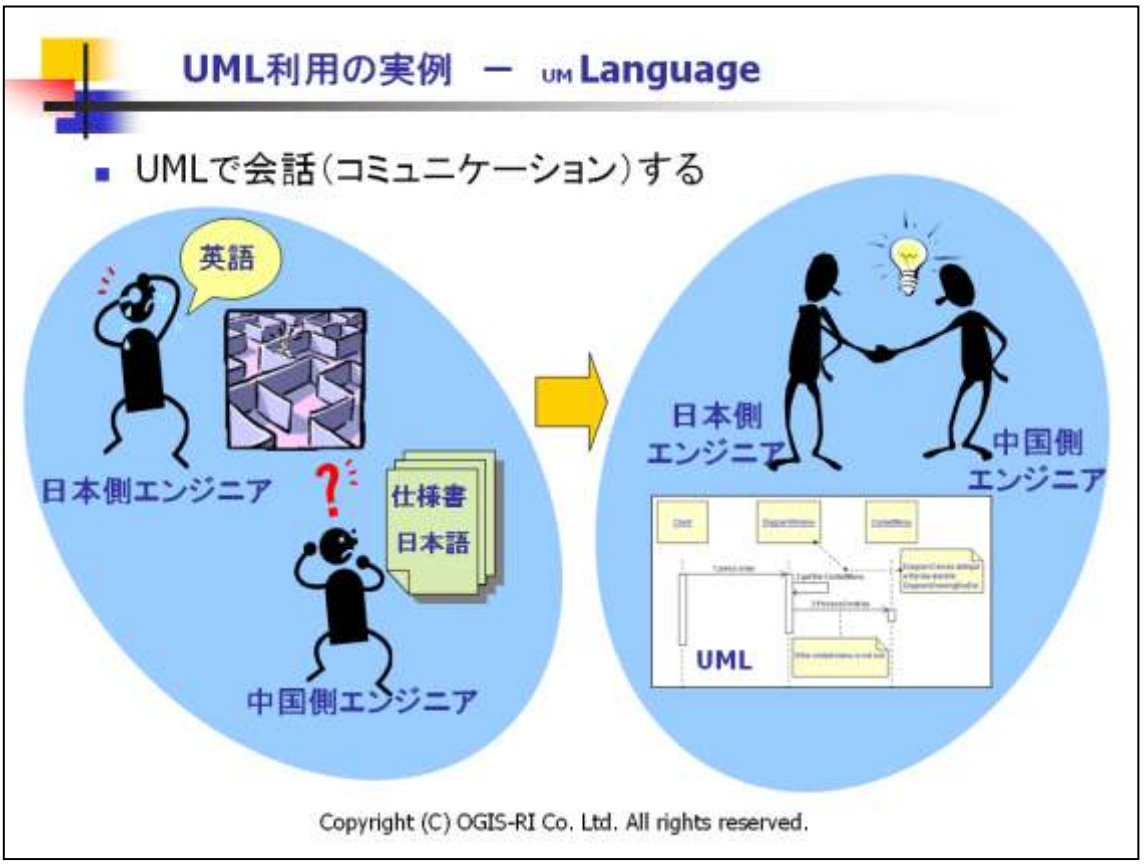
UML を使って良かった点・困った点

UML を使ってうまくいった点はどのような点だと感じていらっしゃるでしょうか。

逆に UML がないと開発はできなかったのかなぁと思います。日本語ができるエンジニアが増えてきたとはいえ、ドキュメントベースで長い日本語を書いてしまったりするとどうしても理解度は落ちるので、UML で仕様を書くということによって、何を作らなくてはいけないかというものに対する理解度は格段に高まっているという風に感じます。それを UML なしで、日本語にしても英語にしても中国語にしても、説明するのは非常にしんどいのかなという風には思います。

UML を使うことによって相手が理解しやすいようになったと感じていらっしゃるのでしょうか。

そうですね。後は、上海側がどれぐらい理解しているかというのを日本側が測る意味でも、上海側が書いた UML の図をレビューするのは意味があることだと思います。上海側のスキルもある程度みえてきます。一方通行ではなくて、両方向で役に立っているのかなぁという風に思います。たぶんそれは、相手が上海ではなくて日本人同士でも拠点が離れて開発していたとすると、やはり UML を使うメリットは出てくるのかなと思います。



物理的に距離がある場合、それは外国でなくても、UML を使う方がコミュニケーションを取りやすいと感じていらっしゃるのでしょうか。

そうですね。複雑なものを伝える場合とか、モデルでレビューすることによって品質を確保したい場合には効果が高いと思います。

モデルを使って品質を確保するというのはどういうことなのでしょうか。

モデルの段階でレビューするので、C#で書いたソースコードレベルの細かいところを見なくても、おおよそ間違ったものはできていないということが確認できます。逆にソースコードを全部見なくてはいけないとすると、ほとんど自分で作ったのと同じようにコストがかかってくるのですが、モデルをレビューすることによって、それほどコストをかけずに品質が高いものを作ることができるわけです。

一方で、問題点は何かありますでしょうか。

そうですね。悪いところは、モデルでレビューしてよしとしているので、実装レベルでのネーミングルールとかまでは、正直、目が行き届いていませんでした。本来であればそこもきちっとコントロールすることで、メンテナンスしやすいソースコードになります。それには、人間の目でレビューするだけじゃなくて、もうちょっと環境というかツールというか、そういった仕組みを作らないといけません。上流はモデル、下流はツールでチェックするという組み合わせが効果的だと思います。

他にも何かありますか。

さっき程も話しましたけど、UML を使うのであれば、最初にドカンと集合教育をすれば結果的には近道だったのではないかと思います。ある程度知っているメンバーもいたので OJT 的にやってきたのですけれど、時々「あれ、本当は分かっていなかったのかな」と思う問題に直面したのも事実です。今にして思えば最初に集合教育はしておけば、結果的には開発時間を短縮できたかもしれません。

[藤野] 読むほうは大丈夫でしょうけど、書かせるときに問題が発生するということですね。

そうですね。設計は比較的似たパターンを横展開していくだけなので大丈夫ですが、分析を任せる前に集合教育をしておけば良かったと思います。全員ではなく、キーマン 2、3 人の育成だけでも効果があったと思います。

[藤野] UML 利用の効果があったというのはどの点ですか。レビューの指摘件数が減ったとか、バグが減ったとか、工期が減ったというか、生産性が上がったとか。

一番顕著なのは日本側の要員を減らせているところです。そもそも最初のやり方では日本側だけが忙しく、日本側が頑張って上海側は定時で帰るという状況で、何のためにオフショアに出しているのかよく分からないような状況でした。UML を使って上流から委託することで、日本側の工数が減り、それでもユーザーに使ってもらえる品質のものが出来上がってきているというところです。

[竹政] インドの時と比べて中国ではどのへんがよかったとかはありますか。

インドの時はとりあえず作っちゃって、バグのもぐらたたきゲームに突入したのですけれど、中国委託時にはその反省も含めて、徐々に徐々に任せていったっていうこともあり、当然バグはありますけれどもコントロール不能な状態には陥っていません。

[藤野] よく言われるのは、ソフト開発に関してインドは優秀で中国はインド程ではない。逆に中国の場合は日本のやり方でやってくださいと指導までするけれども、インドの場合にはインド側のやり方についつい口出しできない、そういう観点はどうですか。

そういう観点もあると思いますね。ただ、インドにしても最初から日本側がコントロールしてなくてはいけなかったと思います。でも「我々は CMM レベル 5 持っています」とか言われると、「そうなの」っていうことで、事態を把握するまでに大分時間がかかったという反省があります。

[藤野] インドでも UML は使っていたのですか?

はい、使っています。

[藤野] では UML があるからといってトラブルが防げるわけではないということですか。

そうですね。インドから提出された UML の成果物をレビューしていました。確かに UML の知識は格段に上、というかアーキテクチャを任せられるぐらいの人がゴロゴロとは言わないけど確かにいます。そのエンジニアが真剣にやってくれたらすごいものができるだろうな、っていうのは中国ではないですね。ただ末端までいくと似たような状況なので、いくら優れた人がいても、訳が分からない人がたくさん入ってきてしまうと、出来上がったものは訳が分からなってしまいます。

[藤野] UML で記述された仕様を実装するときにバグが発生したということですか。

アーキテクチャ、分析の途中ぐらいまでは UML ベースで非常にうまく進んでいたと思います。そこから設計、実装と進むにつれ、UML の知識もない人間がどっと投入されて、混乱してしまいました。

[竹政] モデルとソースコードにギャップができたということですね。そこのコントロールがインド任せになってしまったので…。

分析ぐらいまでは素晴らしいのですが、設計あたりでは似たような機能を持ったクラスがそこら中に現れ始めるといった状況。逆に上流は素晴らしかったので、任せりゃ大丈夫だと(笑)。

[竹政] 中国の方は始めから怪しいなと思ってやっていたので、大きな問題になっていないということですね。

[藤野] 非常に貴重な話ですね(笑)。裏を返せば会社の規模が大きくても小さくてもちゃんと見るとこまで見なきゃいかんということですね。

そりゃそうです。

[竹政] モデルをうまくいったり来たりするような形で細かくチェックしていけば何とかなるけれども、逆に UML を使用していてもポンと任せたきりであまり見ていないとダメだよ、という話ですね。

UML を使っても、最初はやはり苦労しますね。相手のレベルが分からないから、どこまでチェックしたらいいかも分からない。立ち上がり時は、相当きちんと見なくてはいけないのかなとは思います。

今後もオフショア開発で UML を活用するか

「積極的に使用する」、「使用しない」、「条件により使用する」の中から今後の UML の活用について教えてください。

使用しないということはありえないです。個人的には積極的に使用したいですけれど、オフショアということであれば「条件により」ですね。先にも話したように、条件としては、たとえば長く続けられるプロジェクトであるとか、仕様が比較的コントロールしやすいとか。

オフショアで UML を使用する条件は、長く続けられるプロジェクトであるということですか。

パッケージ開発でなくても、システムの維持管理とかであれば、長く続けられますね。システムに対する理解が徐々に上がっていくことができます。結果的に一定期間後には、ある程度上流から任せることが可能になります。

[竹政] ユーザードメインの知識や UML 的な知識ということですね。

そういう意味ですね。

もう 1 つ、仕様をコントロールできるとはどういったことでしょうか。

ガイドラインの補足部分にも関係するのですが、日本におけるシステム開発の特徴として、基本的に仕様変更が前提となっています。エンドユーザーレベルで比較的短い間に仕様変更が入ってしまうと、それをオフショア先に伝えるというのは難しいわけです。だからある程度仕様の変化が少ないところを切り出して任せるとか、仕様変更が入ったとしても優先順位を調整できるようなことができないと、オフショアに出すオーバーヘッドを吸収できないですね。

[藤野] 長く続けられるプロジェクト、仕様をコントロールできるプロジェクトというのは、オフショアに出すときの条件であって、たまたま UML が使えればそれに越したことはないのだけれども、今の 2 つの条件というのは、UML を使う／使わないの判断にはならないのではないですか。

そうですね。加えて言うなら、お互いに UML を知っているという条件でしょうか。

[中原] 業務アプリの場合はどうすればいいのでしょう。

先ほども言いましたが、システムの維持管理とかであれば可能性があると思います。でも3ヶ月単位のバラバラのプロジェクトでは難しい。

[中原] そこなんですよね。開発の大半がそうなんですけど、安定したお客様の仕事ばかりでないので、そういうところでオフショアを使うにはどうすればよいか。

システムの維持管理でも、それなりの覚悟が必要ですね。

[竹政] 業務は、1ヶ月や2ヶ月では分かりません。中国から日本に来てもらって何年か開発してもらうか、あるいは日本側が中国に行って何年か一緒に開発するかといったことも必要なのではないですか。

資本まで入れてという話になれば出来ますが、全然違う会社だといつ仕事が切れるかもしれないので難しいですね。

[中原] 今後の課題ですね。

オフショアに UML を導入する際に必要なもの、ガイドラインについての意見・要望

オフショアに UML を導入する際に必要を感じた物があれば教えていただけますでしょうか。

集合教育をすればよかったという、さっきの話ですかね。これも話をしましたが、下流側のソースコードチェックツールもあれば良かったですね。

[中原] お薦めのツールなどはありますか?

コーディング規約のチェックツールなども大事なのかもしれないですけど、メトリクスツールとかでしょうか。モデルの段階ではいったん見ているのですが、実装に落ちたらグチャグチャになってしまったというところを救うために、いちいち全部読まなくてもある程度統計的にソースコードレベルの品質を表してくれるものとか。

[中原] クラスの凝集度とかですか。

それもあります。あるいは、1 クラスの中に大量のメソッドが詰まっているものを分解するとか。それはそれで使いこなす技術は必要かとは思いますが。上流でモデルレビューをしているのはいいのですが、ついつい下流がおろそかになってしまうので、テストツールも重要です。

[竹政] モデリングツールに求めたい機能は何かありますか。

赤ペンで添削するようなイメージで簡単にコメントが書けるといいですね。モデルの上に 1 枚透明なシートがかかっているようなイメージです。添削は透明なシート上に記載して、はがせばモデルには影響していないというのが便利です。

「オフショア開発向け UML 適用ガイドライン」についての意見やご要望事項があれば何でも。

初めて見る人にとってみれば、やはりサンプルとかが欲しいですかね。サンプル、テンプレートですね。用語辞書にしても、命名ルールにしても、サンプル、もしくはここらへんにそういう情報がありますよ、とか。

[中原] 初めて見る人は言葉では分かっても、じゃあどう使おうか?と思うので、あった方がいいですね。

[藤野] 読めるだけの活用レベルと、実際に書けるレベルまで持っていく活用レベルというのがあるのでは。いろんな活用のレベルがあるということと、それに向けてどんな教育をしていかなければいけないということが書いてあればよいのではないでしょうか。

そうですね。

[藤野] 今日の正田さんの話だと、設計まではある程度読んで理解できるだけでもよい。書いたものが良いものかどうかの判断することが難しいということですね。

そうですね。逆にそれは、オフショア側だけじゃなくて、日本側もそうでしょうね。いいか悪いかの判断ができないと。

[竹政] たぶん、読めれば比較はできるはずですね。こっちとこっちのどっちのモデルが優れているかといえば判断できるレベルはあるのだけれども、モデルを1つだけをみたときにどこが悪いっていうのが分からないと。だからそれを誘導してくれる誰かが「このへんはどう?」という話をすれば、たぶんそこを見直すことはできる。

そこまでいかないと、UML を使用する効果は薄いですね。でもそれはオフショアだからという話ではなくて、なかなか日本で UML が普及していかない根本的な問題がそこにあるような気はします。

以上です。ありがとうございました。

成功へのポイント

●定期的(1 年もしくは半年ごと)にガイドラインの各ポイントの適用状況をチェックすることで、プロジェクトのその時々の弱点が浮き彫りになり改善に生かすことができる。

●UMLをオフショア開発で利用するのであれば、委託範囲は実装・単体テストに限定せず上流(分析・設計フェーズ)にも参画してもらう方が品質・費用の両面から効果が高い。

●オフショア委託先に上流から参画してもらう場合、分析フェーズを任せる前にキーとなるエンジニアにUMLの集合教育を行なった方がよい。

付録 D. 財務会計システム FAST オフショア開発でのUML適用事例インタビュー報告

インタビューの内容

インタビュー日付：2009 年 8 月 25 日
インタビュアー：長田 真理亜（一部 中原俊政、正田塁）
インタビュイー：鴨下 明

開発の概要

今日は、オフショア開発でUMLを使用されたということで、お話を伺いたいと思います。まず、プロジェクト名を教えてください。
「地方自治体向け財務会計システムの改修と機能追加」です。

その中にはいくつかのプロジェクトが含まれているそうですが、オフショアに出した先の企業は変わってきているのでしょうか。
今回、複数プロジェクトを発注したオフショア先は、中華人民共和国の 1 社です。

仮に A 電子さんとします。これが中国の発注先ですね。
はい、そうです。

所在地はどこですか。
遼寧省の大連です。

どれくらいのお付き合いですか。
今年で 3 年目になります。

では、最初から 3 年間ずっと、地方自治体向けの改修をされているのですか。
はい。私の部門としては、この製品だけです。相手先も 1 社だけですし、私の部門としてオフショアに出している他の発注事案はありません。

その 1 製品について、A 電子さんにこの 1～19 の案件を発注されているということですね。

No.	案件名	作業工程	作業量		当社計画工数（人月）	BP実績工数（人月）	差異	作業期間	備　考
1	○○区予算執行	製造・単体試験	伝票修正	25 本	1.5	1.4	0.1	2008/6/16 ～ 2008/7/25	
2	〃　２次開発		帳票新規	5 本	0.8	1.5	-0.7	2008/7/22 ～ 2008/8/22	
3	〃　３次開発		画面新規	1 本	0.5	0.5	0.0	2008/8/18 ～ 2008/9/12	
4	○○区郵送料金		帳票新規	7 本	1.5	1.7	-0.2	2008/7/7 ～ 2008/8/22	
			伝票新規	1 本					
			画面新規	1 本					
			画面修正	2 本					
5	○○区契約管理		伝票修正	30 本	1.5	2.5	-1.0	2008/7/8 ～ 2008/8/14	
6	〃　２次開発		画面新規	3 本	2.5	2.4	0.2	2008/7/18 ～ 2008/8/28	
7	〃　３次開発		画面新規	1 本	2.1	2.3	-0.1	2008/8/1 ～ 2008/9/10	
			画面修正	2 本					
			帳票新規	2 本					
8	〃　４次開発		伝票修正	20 本	1.0	1.1	-0.1	2008/8/8 ～ 2008/8/29	
9	○○区備品管理		伝票修正	7 本	0.4	0.6	-0.2	2008/7/14 ～ 2008/8/15	
10	〃　２次開発		帳票修正	11 本	0.9	0.9	0.1	2008/7/22 ～ 2008/8/29	
11	○○区財産管理		帳票新規	8 本	1.6	2.0	-0.4	2008/8/25 ～ 2008/9/26	
12	〃　２次開発		帳票新規	3 本	1.2	1.2	0.0	2008/9/1 ～ 2008/9/30	
13	〃　３次開発		帳票新規	6 本	1.2	1.3	-0.1	2008/10/10 ～ 2008/10/31	
14	○○区用品管理		帳票修正	9 本	0.8	0.6	0.1	2008/7/7 ～ 2008/8/15	
15	○○市予算執行	設計・製造・結合試験（帳票設計除く）	伝票修正	12 本	2.1	2.2	-0.2	2008/11/25 ～ 2008/12/19	
			帳票新規	3 本					
16	○○市契約管理	製造・単体試験	伝票修正	18 本	0.7	0.7	-0.0	2008/11/25 ～ 2008/12/19	
17	○○市予算執行	設計・製造・結合試験（帳票設計除く）	伝票修正	11 本	1.8	2.0	-0.2	2008/12/1 ～ 2009/1/9	
			帳票新規	3 本					
18	○○市予算編成	製造・単体試験（ﾃｽﾄﾊﾟﾀｰﾝ作成）	帳票修正	4 本	1.3	1.3	-0.0	2008/12/1 ～ 2009/1/9	
			画面修正	5 本					
19	○○市行政評価	製造・単体試験（ﾃｽﾄﾊﾟﾀｰﾝ作成）	帳票新規	2 本	1.0	1.0	-0.0	2008/12/26 ～ 2009/1/23	
			一括修正	1 本					
合　計					24.2	27.0	-2.7		

はい、そうです。

そうすると、最初のもので 2008 年 6 月から、至近のものは 2009 年 1 月 23 日ですね。

はい、その通りです。

対象製品の名前を教えてください。

「地方自治体向け財務会計システム」です。製品名は「FAST」で、「ファスト」と発音します。

開発工程について教えてください。

現状私の部署では、個々のお客様にパッケージ製品を適用するという形で各プロジェクトが走ります。つまり、もともとのパッケージ製品があり、お客様に実際にそのパッケージをお使いいただくに当たって、細かい個々の機能がお客様のお仕事に合っているかを確認するということが要件定義の工程になります。通常のプロジェクトであれば、色々な資料類やユースケース図等を使って「このようなお仕事の手順で間違いないですか?」という確認をお客様とすり合わせするかと思いますが、パッケージ製品の顧客適用の場合は、実際の画面の動きを見て、システム導入後の仕事進め方を想像していただきながら、打ち合わせを進めるという形になります。要件定義については、現在のところは国内で弊社プロパーの SE が担当します。

パッケージ製品として各業務処理は基本仕様ベースで完成していますので、本来の意味での基本設計の工程は通常は実施しません。要求定義からいきなり詳細設計に入ります。続く詳細設計の進め方のスタンスとして 2 種類に大別できます。パッケージで持っている既存の機能を部分的に改修するというケースと、機能的に不足していて、お客様がご予算のご用意が可能な場合に、その機能を新規に追加するというケースです。この 2 種類のケースに対する詳細設計については、現状ではほとんどが、まだ国内で弊社のプロパーSE が行っています。

当部門の場合、これ以降の工程からがオフショアの対象になるのですが、今のところ、オフショア対象プロジェクトの内の 8 割くらいは製造と単体テストの工程を中国の A 電子さんに発注し、結合テストから後は弊社で引き受けるという形になっています。残りの 2 割については、詳細設計を含めて A 電子さんに依頼しています。現在は発注対象の工程を広げ始めているところです。詳細設計を発注すると、通常は当然に結合テストも受注先が行う形になります。総合テストと現地の運用テストなど、テストの後半部分は、お客様先で実施しなければいけない工程となりますので、以降は弊社のプロパーSE が担当する形になります。

役割分担についてもう一度確認させてください。要件定義はジャパンシステム様で実施されるのですね。
　はい。要件定義については全て弊社で担当しています。

パッケージ製品のエンハンスは基本設計に影響しないため、基本設計の工程は実施しないのですね。詳細設計の役割はどうですか。
　現在はほぼ弊社の担当です。課題として、詳細設計もオフショア先に出したいと思っています。

将来的には詳細設計からＡ電子さんにお願いしたいということですね。
　はい。現状は 2 割くらいしかできていません。

製造・テストは全面的にＡ電子さんに依頼されているのですね。
　はい。発注対象サブシステムごとに原則製造とテストは A 電子に発注しています。

結合テストはどうですか。
　結合テストは詳細設計とリンクしており、詳細設計を担当したところが結合テストをするというのが通常は効率的だと考えているので、同じような割合になります。

ということは、ジャパンシステム様でほぼ結合テストを行っていて、将来的にはＡ電子さんに移行したいということですね。
　はい、そう考えています。双方の体制構築も含めて急ぎ対応すべき課題となっています。

総合テストについて最後に触れていただきましたが、こちらはジャパンシステム様が現地に赴いて実施されているのですね。
　はい。お客様から稼動判断を頂かなければいけないので、どうしてもプロパーが担当する必要があると考えています。

では次に、オフショアを実施した規模について教えてください。基本設計は省いて、要件定義、詳細設計、製造・テスト、結合テスト、総合テストについて教えていただければと思います。
　中国側企業の方についてですが、発注全体で 27.0 人月です。
　各プロジェクトごとの全体工数と中国側の工程別工数内訳は表をご参照ください。

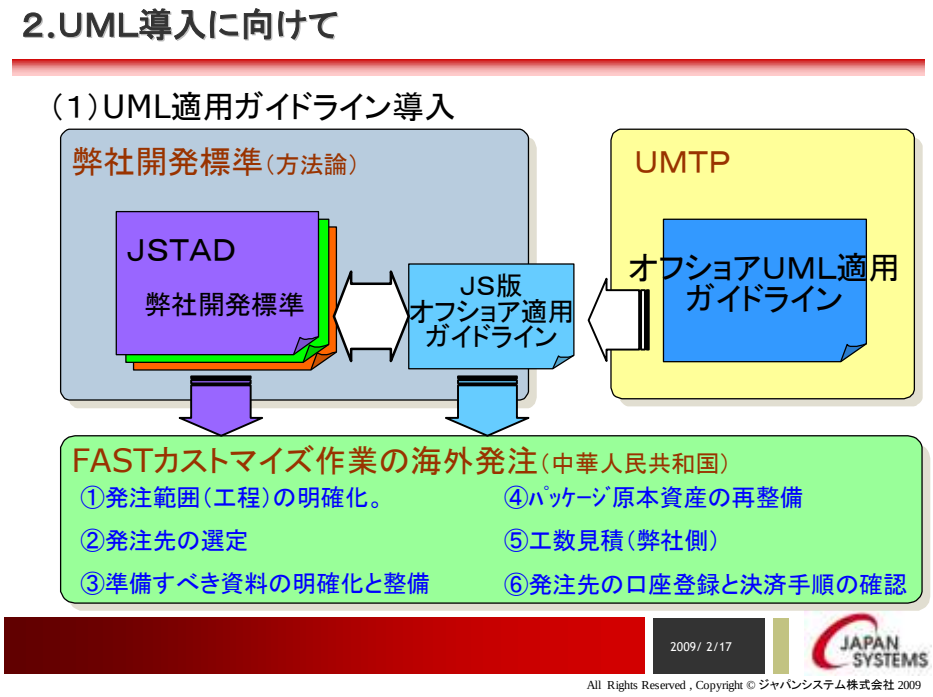
No.	案件名	作業工程	発注作業量		BP実績工数（人月）	実績工数　工程別内訳			ﾌﾟﾛｼﾞｪｸﾄ全体規模（人月）	備　考
						詳細設計	製造	結合試験		
1	○○区予算執行	製造・単体試験	伝票修正	25 本	1.4		1.4			
2	〃　２次開発		帳票新規	5 本	1.5		1.5			
3	〃　３次開発		画面新規	1 本	0.5		0.5			
4	○○区郵送料金		帳票新規	7 本	1.7		1.7		79.3	
			伝票新規	1 本						
			画面新規	1 本						
			画面修正	2 本						
5	○○区契約管理		伝票修正	30 本	2.5		2.5			
6	〃　２次開発		画面新規	3 本	2.4		2.4			
7	〃　３次開発		画面新規	1 本	2.3		2.3		48.5	
			画面修正	2 本						
			帳票新規	2 本						
8	〃　４次開発		伝票修正	20 本	1.1		1.1			
9	○○区備品管理		伝票修正	7 本	0.6		0.6		15.9	
10	〃　２次開発		帳票修正	11 本	0.9		0.9			
11	○○区財産管理		帳票新規	8 本	2.0		2.0		15.1	
12	〃　２次開発		帳票新規	3 本	1.2		1.2			
13	〃　３次開発		帳票新規	6 本	1.3		1.3			
14	○○区用品管理		帳票修正	9 本	0.6		0.6		6.0	
15	○○市予算執行	設計・製造・結合試験（帳票設計除く）	伝票修正	12 本	2.2	0.6	1.1	0.5	7.1	
			帳票新規	3 本						
16	○○市契約管理	製造・単体試験	伝票修正	18 本	0.7		0.7		8.4	
17	○○市予算執行	設計・製造・結合試験（帳票設計除く）	伝票修正	11 本	2.0	0.5	1.0	0.5	11.5	
			帳票新規	3 本						
18	○○市予算編成	製造・単体試験（ﾃｽﾄﾊﾟﾀｰﾝ作成）	帳票修正	4 本	1.3		1.3		6.6	
			画面修正	5 本						
19	○○市行政評価	製造・単体試験（ﾃｽﾄﾊﾟﾀｰﾝ作成）	帳票新規	2 本	1.0		1.0		11.8	
			一括修正	1 本						
合　計					27.0	1.1	24.8	1.0	210.2	

UML/ガイドラインを使用したきっかけ

では、ここからインタビューの中枢部分である UML の使用、その方法、成果、ポイントなどについて伺います。UML を使用したきっかけを教えていただけますか。

UML 導入の契機としては、弊社の上層部から UML を使って生産性を上げるよう指示があり、UML 関係の資料やノウハウをチームで収集、研究して、導入したことがきっかけです。

ガイドラインについては、先ほどの説明で、ジャパンシステム様が使っている JSTAD の開発標準と、オフショア UML 適用ガイドラインがうまく符合するということで、そのすり合わせをして実際の適用に臨んだとおっしゃっていましたが、ガイドラインを使用したきっかけとしてはそれでいいでしょうか。



ガイドラインを使用したきっかけも、上層部から UMTP を紹介され、ガイドラインのドラフト版の提供を受けたことを契機としています。そのときは Ver1.0 だったのですが、実際に弊社で持っている開発標準と、オフショアに向けてのチェックポイントを取りまとめていた WBS とをすり合わせして、有効なところを使わせていただこうという方針で UML 導入に着手しました。

当初弊社で考えていた WBS に漏れている事項があれば補足するという視点もあり、その情報源として使用しました。

上層部のメンバーが UMTP のオフショアの分科会に参画させていただいており、比較的早いタイミングでガイドラインを入手できましたので、その情報を個別のプロジェクトにも配布して、活用させていただきました。

[中原] 御社開発標準と比べて、モデリングに関してガイドラインの情報量はいかがでしたか。

表現や分類の違いはありましたが、8 割程度は弊社の必要事項を満たしていたと思います。

[中原] 2 割程度を追加したということですね。

そうです。「必ず UML である必要はない」というところは斬新でした。UML バージョン 1.1 の頃に別の大手ベンダーさんのお仕事で UML を使ったことがありましたが、総合的に導入すべきかと考えていました。その頃は UML をよく分かっていませんでした。このときはスクラッチの開発だったのでユースケース図から始めたのですが、大枠の図を描いて、その下に説明文と称してダラダラと従来方式のように仕様条件等を書くなど、UML 本来の要点すら分からずにやっていました。それに対して、UML の利点を活かす形で導入するという意味では、ガイドラインの各項目に記述されている あたりまえのことも邪魔だとは感じませんでした。 私のように導入当初

UML のスキルが低かった者から見ると、あたりまえのことも完結に記述していただいた方が、理解が正しくなります。 事例とまではいかなくても多少実務ベースに近い形の説明がガイドラインに記載されていることで、テキスト類（入門書や参考書）から理解した「UML はこうあるべき」の実践指針が間違っていないのだと再確認でき、安心できます。

ガイドラインに指針や、UML 導入の方向性に関して 22 の項目別に示されているという点からも、要点の抽出がしやすい構成となっているので一読で概要もつかみやすく参考になりました。

[中原] ガイドラインに不適切なところはありませんでしたか。

特には思い当たりません。

今のお話について少し確認したいのですが、当初ガイドラインをお使いの際、ご自身の UML のレベルはどのくらいでしたか。

初心者でした。

UML 初心者の鴨下さんにとって、JSTAD とガイドラインをすり合わせる際に、ガイドラインが実務ベースに近かったので読みやすかった、それまで手探りでやっていたけれどガイドラインの中に指針や方向性を見出せたということでしょうか。

はい。以前に別の大手ベンダーのお仕事で UML を使ったときは、参考にすべきものが入門書、参考書とリファレンスしかありませんでした。ですから、これら参考書類に出ているサンプルくらいしか事例のダイアグラムが手元にありませんでした。

今回は、実務に近いサンプルが、資格取得用の問題集などに豊富に入っていることを知っていたので、そこから図の書き方などのスキルを身につけた記憶があります。個別のダイアグラムをどう書くかとか、文法がどうとか、こういうクラス図はいい/悪いということとは別次元で、プロジェクトを推進する上でこういう点に気をつけて導入すべき、といった通常のテキスト類（入門書や参考書）には比較的記載の少ない留意事項や指針の理由等がガイドラインの各項目に記述されていました。

上流のプロジェクトマネージャとして有効性等の判断しなければいけない項目に関わるヒントとなる内容が記載されていて参考になりました。

上流工程に関するところが役立ったということですか。

実際のプロジェクトで UML を使って、たとえばどの図を選んで先方に提供するか、先方にどの図をどのように書かせる、といったことを決定し先方と調整する際に、弊社としてどう指針を設定するか検討する際の参考になりました。

UML をどのようにプロジェクトに導入していくかという部分ですね。

はい。UML のダイアグラムを描くスキルというより、UML をプロジェクトとして採用するにあたって留意なければいけない点が指針として示されているので、プロジェクトマネージャの立場として、比較的安心してオフショアプロジェクト進める事ができたと思っています。

UML/ガイドラインの使用方法 ― 使用した図とタイミング/効果

ここまで、UML とガイドラインを使用したきっかけ、ガイドラインのよかった点などを伺いました。次に、A 電子さんとのやり取りで実際に使用したクラス図とそのタイミングを教えてください。

使ったのはクラス図、シーケンス図、ステートマシン図の 3 つです。詳細設計の内容を先方に伝えるために導入しました。

既存のパッケージ製品の設計を先方に知ってもらう必要があったので、そのときに 3 つの図を使ったということですか。

そうです。パッケージ詳細設計の基幹的な部分と改修対象の改修前の仕様がどうなっているかを知ってもらうのに必要な部分について 3 種類のダイアグラムを作って、それを先方に元資料として渡しました。

作成者は鴨下さんですか。
私ではなく、私の配下のメンバーです。

どのレベルのメンバーですか。
中堅メンバーで、サブリーダークラスです。

サブリーダークラスの方がクラス図、シーケンス図、ステートマシン図を詳細設計の時に作成して、中国側に渡したということですね。そのときのそれぞれの図の目的を教えていただけますか。
クラス図は、個々の部品プログラムがどういう構造で作られているかを理解してもらうために使いました。

それによってどのような効果がありましたか。
クラス図の導入効果は、各プログラムからそれぞれどんなサブプログラムが呼ばれているか、逆に、末端のプログラムがどの上位のプログラムから関連付けられて呼ばれているか、が明確に理解できることです。

シーケンス図を詳細設計で使ったときの目的を教えてください。
シーケンス図にはスコープがいろいろとあると思いますが、まずスコープを除外してお答えします。ある 1 つのプログラムの中で各サブプログラムを呼び出すときに、プログラム間通信が行われます。上位のプログラムが下位のプログラムに対して「××をしなさい」という命令を投げ、下のプログラムは「命令を受けた結果」を上に向かって返す、という動きです。その命令が何をトリガーに発行されるか、命令を受けて処理した結果をどのように返すか、というやり取りの関係を明記します。それが 1:1 から 1:N になった場合や、階層が 2 階層から 3 階層になった場合にも、それぞれ同様の関係があります。これらそれぞれの場合に応じて、呼ぶ/呼ばれる、どういう答えを返す、といった処理の順番を意識した時系列的なタイミングを明確に記載できるのがシーケンス図の導入効果です。

[中原] シーケンス図を書くと作業量が大変なことになると思いますが、複雑なクラス間のシーケンス図を書くときに何か工夫をされましたか。
多少工夫しました。いったんは、あるブロックについて全部書くように指示をしたのですが、期限を経過しても仕上がってこないので状況を確認すると、ミクロフォーカス的に延々と作業をしていたのでブレーキをかけました。
以降はスコープ条件を設定させて、一定階層以降をブラックボックス化することとしました。先ず 3 階層までに限定して概要レベルの図を書かせました。ところが、3 階層に限定すると、シーケンス図として役に立たないものになりました。細分化されすぎていてドキュメントの沢山のページを渡っていかないと全体を理解できないので、シーケンス図の意味を持たないものとなってしまいました。
そこで、必要となるクラスグループのブロックをクラス図の中で経験的に切り出して、それに関していったんシーケンス図をざっくりと書いて、深くなる部分はブラックボックスにして止めるようにしました。

おおまかなシーケンス図を書いて、細かく枝分かれしていくところは抜き出して別にシーケンス図を書いたということですか。
はい。私たちは当初作図の経験が少なかったために失敗をしたのですが、いったんはざっくりとした記載で止めて、レビューをしながら最適な粒度を割り出していく方法でしかまとめられませんでした。品質に関わる重要な部分については、スコープを変えて切り出しながら書いていくのが現実的には良いと思います。
これは既にあるシステムについて図を作成する場合です。スクラッチ開発のケースで UML を導入する場合は、基本的な部分や処理の錯綜が予想できる箇所について優先的にシーケンス図を作成して、極力整合性を明確にしておくことは設計品質の確保につながると思います。
また、当初想定外の成果として、今回は1件だけでしたがシーケンス図を書くことによって障害を発見することができました。

[中原] まずは概略を作って、あとはレビューしながら必要なところを書いていくということですか。
はい。それが良いと思っています。

では次に、ステートマシン図の目的を教えてください。
ステートマシン図は、あるプログラムが何をきっかけにどういう順序でどんな処理を実行するかを、比較的フリーな形で各処理単位にボックスを設けて、その関連を記述していく図です。ある機能が、何をトリガーとして、どの

条件の時に、どんな処理を実行するのかを、時系列に関係なく記述します。（開始、終了、と条件別の処理内容を記述します）

それもサブリーダーの方が書かれたわけですね。同じ方ですか。

着手当時に L2 の資格を持っていた人が 2 名いたので、そのメンバーを専任にして書かせました。

クラス図、シーケンス図、ステートマシン図それぞれを別の方が作られたわけですか。

担当リーダ皆が 3 種類の図を書きました。

ステートマシン図を実際に使って、どのような効果がありましたか。

それぞれの部品のプログラムが何をトリガーにどういう動きをするかを図で表しているので、処理の全体像が一目で理解しやすいという点です。

工夫された点は何かありますか。

ステートマシン図は従来のフローチャート図に近いので、従来型の仕事をしていたメンバーも、ステートマシン図の文法を覚えるだけでかなり深く（的確に）理解、作図することが可能となりますので、比較的導入は容易でした。もちろん、ステートマシン図も全部を書こうとすると多くの労力がかかりますので、どの部分をステートマシン図に落とすかの選択に注意が必要でした。

先ずは設計全体の中で出現頻度の多い典型的な処理について一連の処理セットの開始から終了までを作図対象として選定しました。

次に作図対象として、過去に新任メンバーがよく誤解して障害等を起こしがちな部分をピックアップして対象箇所として選定しました。このケースは作図対象を徐々に増やす事ができますが、ある程度ソースレベルまで読み込んだあとに補足的にダイアグラムを見るようにしないとダイアグラム自体を理解しにくいかと思います。

とはいえ、ステートマシン図については、フローチャートに近いダイアグラムなので、間違いなく発注先に伝えなければならないのはこの部分だという絞り込みに、担当 SE とレビューアーでズレがほとんど無く、すんなり導入できました。 特殊な工夫をしたという記憶がありません。

実際の効果としては、フローチャートよりも表現できる範囲が少し広いので、フローチャートを描くくらいならステートマシン図を描いた方が、同じ労力でも正確に仕様が相手方に伝わると思います。

[正田] この 3 つの図はすべてジャパンシステム様で書かれていて、中国側はそれを読んで理解するだけということですか。

そうです。ただ、昨年の後半の何本かのプロジェクトでは、新規で作成する機能について何枚かステートマシン図を作ってもらいました。成果物を UML で書いてもらうことにはまだ本格的に着手していないので、今年も含めてその形を増やしていきたいと思っています。

[正田] その場合には中国側で書かれた図を日本側でレビューするということですか。

はい。少しづつですが中国側にも設計工程を発注していますので、その際に担当リーダが必要と判断した場合はステートマシン図やシーケンス図を作成させます。中国側で作成した図は当方のレビュー承認後に次工程へ進むよう指示する手順となります。

今のところは、ジャパンシステム様でクラス図を作成して、A 電子さんがそれを読んで理解すると言う状況ですね。

はい、大半はそうです。確実に実績があるのはジャパンシステムで作図して先方に渡すというところにとどまっています。

それは詳細設計の部分ですね。実際に A 電子様がこの図を使うことで効果があると感じたことはありますか。

初回にクラス図もシーケンス図もステートマシン図も部分的に作成したのを用意して先方に説明した際は、実際にプログラミングを担当する方が 7、8 名の出席でした。彼らは、ドキュメントに日本語を書くことはできるけれども、会話はできないというレベルの日本語力であったのですが、仕様説明の冒頭から、「なぜこういうクラス構成になっているか」というような、クラス図の意味合い自体は正しく理解したうえでの、質問をしてきました。

パッケージ製品といっても、お客様から改修の要望がよく出てきがちな部分があります。スクラッチでの構築の場合なら動作の効率や処理の汎化などを考えるかと思いますが、パッケージの場合は、「この部分の仕様はお客様によってかなり要件が違う」という箇所があり、一見非効率でも、あえてその部分を別のクラスに引き継がせて

後で入れ替えるような作りをする部分があります。そのため、出来上がっているプログラムだけを見ると、多少奇異な感じがする箇所があります。そういった部分について、何故そうなっているのかという質問を受けましたので、「この人はダイアグラム中の仕様を正しく解っているな」という印象を受けました。従来型の設計資料を渡した場合には、おそらく今回と同じ様な短い時間でここまでの理解はされなかっただろうと思いました。

従来型の設計資料よりも、UML を使ったクラス図などの方が、日本語が不得意な中国側の SE にとっても理解しやすいということですね。

母国語が違うということが障壁になりにくいのだと思います。

その他に成果はありましたか。

ステートマシン図には処理に関する動詞や名詞が記述されるのですが、各 PG にダイアグラムを渡すにあたってその翻訳が必要です。この際、図の中に表記されているものが基本的に単語/熟語のレベルなので、資料を中国語に翻訳する作業は無償で先方がやってくれました。初回訪中時はステートマシン図が 25～26 枚くらいだったのですが、先方の日本語ができるメンバーがおそらく 1 日で翻訳できるだろうということで、特に翻訳についての経費はかかりませんでした。従来型のドキュメントなら、翻訳に経費がかかる上、ニュアンスは伝わらず問題が起きていたと思います。そもそもニュアンスが入っていること自体が設計書として問題があると思います。比較はできていませんが、UML の導入によって曖昧さ排除の効果があったのだと思っています。

[中原] 従来のドキュメントとは具体的に何を指していますか。

機能概要書などです。

[中原] たとえば DFD などだと変わらないようにも思うのですが。

DFD なら同じかもしれません。私の知っている範囲では、中国語は英語の語順に近いですね。そのためか比較的白黒はっきりしているので、日本人のように仕様書の行間を読んでくれなどということは通用しません。中国だからというより、日本以外、グローバル標準的にそうだと思います。設計書に書いてあることはやるが、書いてないことはやらないという姿勢で仕事を進めます。そういう意味でいうと、文章表現の設計書はトラブルの元だと思います。ですから、ビジュアルドキュメントである UML をどんどん導入すべきだ　　と思っています。

[中原] 成果として質問の質がよかったという話がありましたが、後工程での品質やバグ密度などについて、感覚的にでも効果はありましたか。

UML を最初から使ったので、使ったときと使っていないときとの比較ができていません。

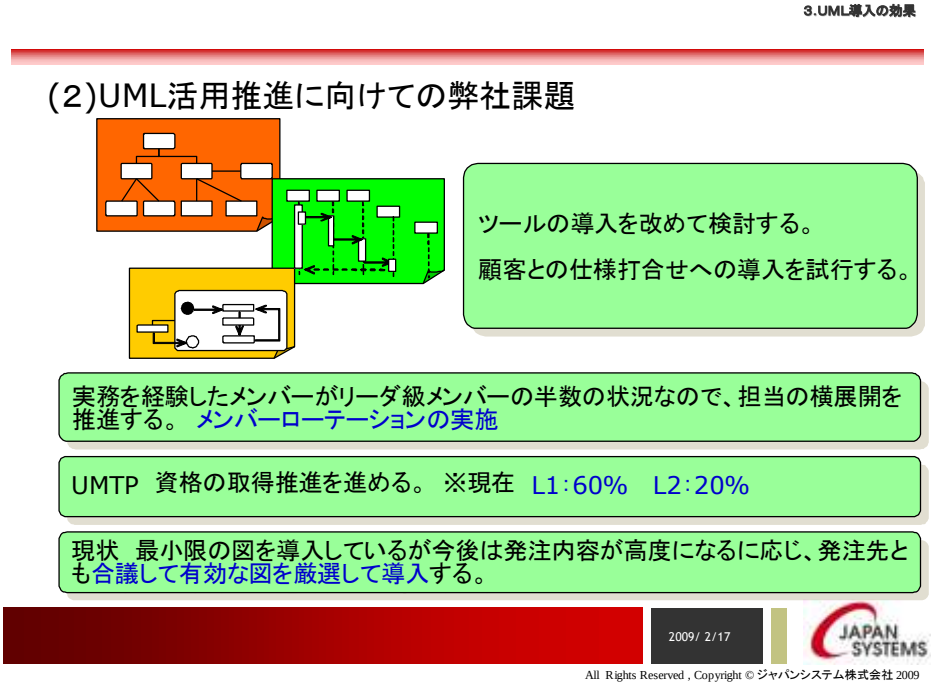
[中原] 逆に、UML を使わないで、質問のやり取りがひどく大変だったとかトラブルが生じたというプロジェクトがよくあるのですが、そういうトラブルはなかったのですか。

今の所は発注範囲が限定的であることもあり、トラブルと言うほどの問題は発生していません。

先ほどの説明には含めませんでしたが、このシステムは VB で作っているため、発注先にプログラムスキルが付いてきた後は、ドキュメントを省略して、「このサンプルでよろしいでしょうか」というように VB のプログラムで返してくることが増えてきました。VisualStudio の開発環境はとても充実していますので、システムの大筋が分かってしまえば、直接プログラムのソースのサンプルをやり取りすることは仕様確認の範囲であれば有効な方法です。当初はステートマシン図を介して要件の確認等をしていましたが、あまり形式だけに拘ってしまうと効率を下げるケースがありますので留意が必要でしょう。発注内容が新規か改修か、また規模や段階（仕様伝達段階か、実装方法の質疑応答段階か）によってはサンプルソースを使うことも、ケースや状況によっては正解となりえるものと認識しています。

UML/ガイドライン活用のポイント

UML 活用推進に向けてのジャパンシステム様の課題として、先ほどの発表の中に 3 点挙げられていましたが、それを含めて、今後の活用に向けて、ポイントを教えてください。



弊社の課題をふまえての今後のポイントについては、先ほどの正田さんからの質問と重なりますが、まだまだ一方通行で、弊社のパッケージシステムの改修前の仕様を説明するためにしか UML を使っていません。設計の工程も含めてオフショア先に依頼できるようにしないと、オフショア対象の業務量の上限が低くなってしまうので、発注工程を広げなければならないという課題があります。この課題をクリアする過程として設計段階が終わった中間の成果物を UML で納品してもらい、レビューを実施する必要が出てくると認識しています。

また、UML に関して言うと、今導入しているのはベーシックな 3 種類の図だけで、たとえば配置図なども場合によっては必要になると想定していますので、対象のダイアグラムの種類を増やしていきたいと思っています。目的はあくまでもオフショア対象の工程範囲を広げるためです。

UML とガイドラインを使用したことを概観して、今回のやり取りの中でのポイントとして PR するものはありますか。

UML の採用を検討しているプロジェクトでも、使える経費の範囲内でダイアグラムを選んで導入できるか否かがポイントだと思います。まずはパイロット的にでも UML を使ってみて、効果を体感しながら導入してもらいたい。部分的導入も可能ということで、それが可能である点も良いところだと思います。

[中原] ガイドラインに対して、こういった内容があればいいなどの要望はありますか。

私の場合は導入当初はサンプルの情報が一番欲しかったものでした。

ガイドラインの中に入れるのは構成上難しいですし、サンプルのダイアグラムに馴染まない項目もたくさんあると思いますが、テクニカルノートのような別冊のサンプル集があると導入検討しやすいかと思います。上流工程のメンバーには今のガイドラインの内容で十分役に立つと思いますが、具体的にどのダイアグラムをいつまでに作成するのかを検討するサブリーダーレベルの方にはサンプル集があると判断がしやすいと思います。UML の経験が少ないメンバーでも方針判断ができるように、事務系、制御系、学術系など、主だった分野のダイアグラムが別冊のサンプル集としてあると良いと思います。ISO のテクニカルノートのようなイメージです。想定した業務サンプルとそれに対応するクラス図や、処理想定とそれに対応するステートマシン図のように、キーワードで分野が選べるようになっていると良いと思います。というのも、私の場合 L2 資格試験向けの問題集サンプルが当初非常に参考になったからです。具体的な例が含まれていて、設問を要件、回答例がサンプルダイアグラムとみなした場合、要件と成果物の両方が揃っているので理解を深めるのに役にたちました。

当然ですが、リファレンスは UML の何たるかは理解している人が文法の確認をするために使う形が本来的なものだと思いますので導入当初の段階では参考にはなりにくいと思います。

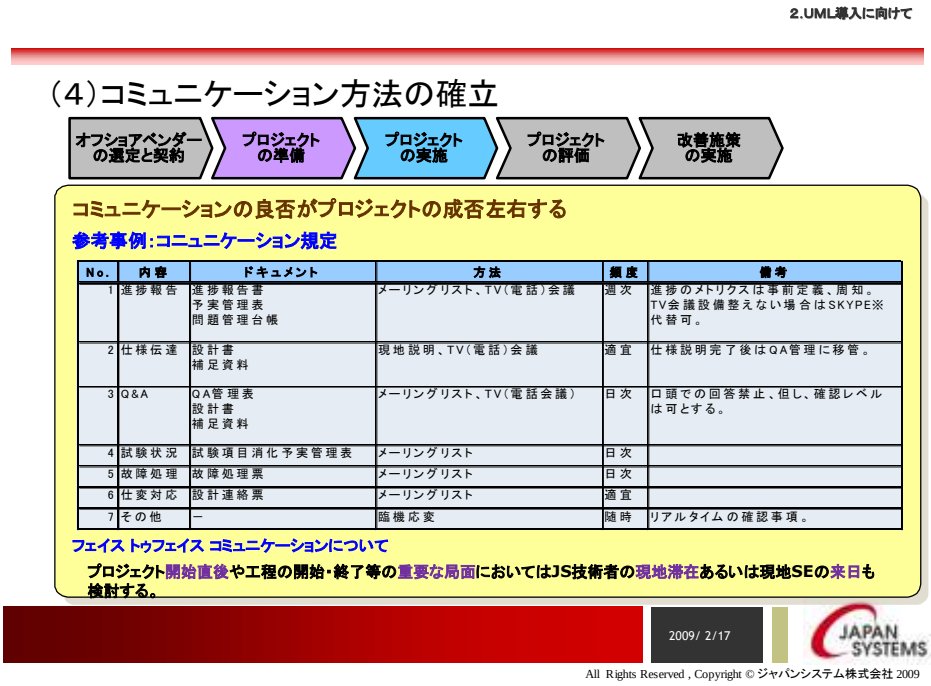
質疑応答

質問者：中原俊政、正田塁

回答者：鴨下　明

[中原] 工程会議などで進捗をフォローした頻度はどのくらいですか。

製造工程の進捗確認は、結構粗くて、週次で済んでいました。スライドで少し触れていますが、進捗報告のところです。



[中原] オフショア側では日次で、国内との会議を週次で行っていたということですね。それで特に問題はありませんでしたか。

はい。オフショア先が優秀だったことが大きいですが、最初の 2 週間くらいは月水金で状況確認を実施していましたが、「順調」という報告と、質問内容もしっかりしていて作業精度も高いと評価できたため、週次にサイクルを変更しました。最初は質問が非常に多かったので、逆に安心した記憶があります。サブリーダーは最初の 2 週間、毎日 20～30 件もの質問にメールで返答していて、苦労していました。

[中原] 質問の内容にもレベルがあると思うんですが、よく分かっている質問が来たということですか。

そうです。8 割くらいは内容をよく分かっている質問でした。2 割は的外れな質問もありましたが、人に問題があったようで、その人を外してもらった後は非常に順調でした。ただ、初回はエース級のメンバーを充ててくれているはずですので、2 年目、3 年目と逆に留意する必要があると考えています。

[中原] 次に、仕様変更についてお伺いします。通常、業務アプリケーションですと、仕様変更で結構トラブルがあるのですが、パッケージのカスタマイズの場合は、そういうトラブルはあまりないのですか。

仕様認識の齟齬による大きな問題は今のところ発生していません。まだ内容浅いというか、比較的誤解のしようがない発注内容のためだとご理解ください。

[中原] 仕様が膨らんでくると、見積もりなおしをするようなことはありませんでしたか。

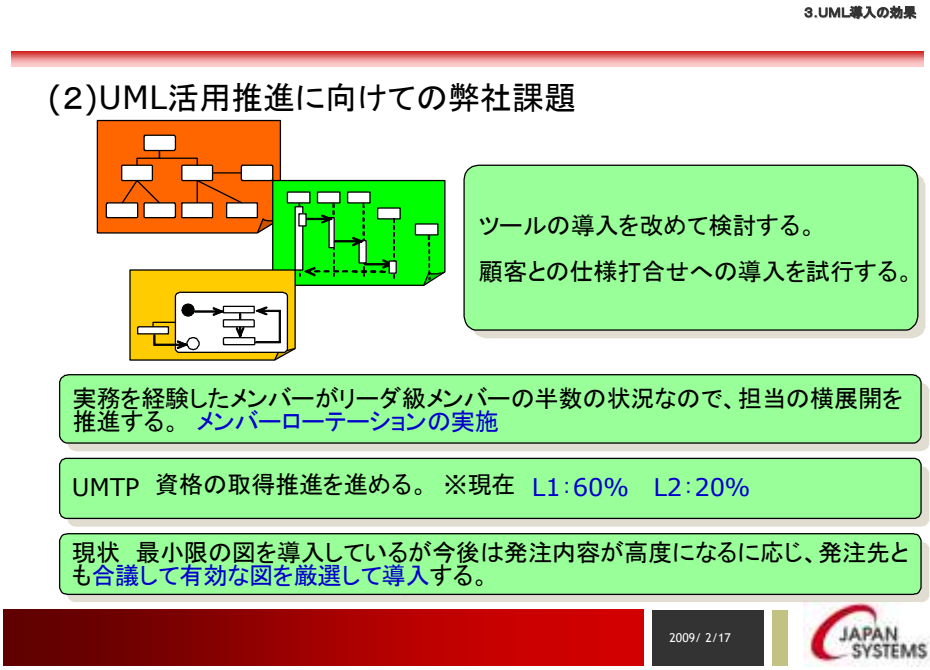
見積りなおしは 2 回くらいありました。

[中原] それで納期が遅れるなどのトラブルはありませんでしたか。

今回は少なかったです。通常は見積もりなおしをする時点では納期も危なくなります。今回は、もともと安全を見たスケジュールで発注していたので、たまたま収まったともいえます。

中国も今年は仕事の絶対量が減っているようで、投げやりな成果物にはなったりはしていません。オリンピックイヤーと同じような好景気が続くのであれば注意が必要かもしれません。プロジェクト管理者は中国に限らず常に疑ってかかる必要があるかと思います。

[中原] UML 導入の課題のスライドですが、資格取得の推進について、現在は L1 が 60%、L2 が 20%と結構高いのですが、目標はどのくらいに置いていらっしゃいますか。



弊社経営層からは基本的に、L1 は SE 全員が取るようにと指示されています。ですが、それはなかなか難しいので、当面、私の部門の今期の人事考課上の目標は L1 が 80%として、対象者を指名して必ず取得するように指導しています。L2 は 30%を目標にしています。L3 は、5%を目標としていますが、まだ 1 人しかいません。

[中原] 全員に取らせようとすると、かなりの投資が必要ですね。

そうですね。予算を取って、バウチャーを先着 50 人に割引で分けるようなことを実施しています。他にも資格手当てとして一時金を出しています。それよりも、私が L2 を取ったのが効いているかもしれません。「上司に先に取得されてしまったので仕方がない…」といったプレッシャーは有効なようです。

[中原] 資格を取らせても、そういう仕事がなかなかないと、取らせるほうの立場がないですね。

それはあります。事実「取れ取れと言った割にはそんなに仕事がないじゃないか」とは言われています。ここは問題です。オフショアのレベルを上げて、プロパーの SE は上流工程だけにシフトしようという目標設定は良いのですが、実務配置を十分に推進できていない点は課題です。

個別のプロジェクトで見ると、まだまだオフショアを怖がっていて、自分の手元でやったほうが安心感があるようです。常識的な納期で受注できればいいのですが、最近はお客様も業者選定が慎重であったりするため業者決定が 1 ヶ月、2 ヶ月、遅れるような事例が多くあって、工期が短くなっています。そんな関係もあって、注文が取れたのは良いが、改修の期間が 3 ヶ月しかないというような仕事が多く、オフショア実施の準備が間に合わないので二の足を踏んでしまうことも多くあります。プロジェクトが細かくなりがちなのが私の部署の今年の傾向です。

[中原] 全体のうちで UML を適用する業務の割合はどのくらいですか。半分以上あれば資格も取らせやすいですが。

1割くらいです。私が在籍しているパッケージ製品を扱っている部隊よりも、中央の公官庁や銀行などのお客様の方が概してロットが大きいので、そこで UML を導入できると良いのですが、規模の大きいものはなかなかフロントで受注できず、他社の二次請けで入ることが多くなっているという事情があります。この場合、開発手法を弊社側で選択することができません。

このためプロジェクト推進の基本指針を自由に決められるということで、どうしても自社パッケージの部隊に指示が下る形になってきます。これらが当社の課題となっています。

[正田] オフショア開発評価シートの UML の導入評価という項目がありますが、これはもう少し詳細化されていますか。

現状はしていません。

[正田] その評価ポイントがあれば、それが次の人のためのいいサンプルになるかもしれませんね。

今回のインタビューは以上とさせていただきます。
皆さん　長時間にわたりご苦労様でした。

成功へのポイント

●UML を適用する場合、プロジェクトのメンバーのスキル、経費等を考慮し、その範囲内で　導入可能なダイアグラムを選択するのがよい。

●いきなり本番適用ではなく、パイロット的に UML 使って効果を体感しながら導入するのがよい。

特定非営利活動法人 UML モデリング推進協議会
オフショアソフトウェア開発部会

ガイドライン作成メンバー

・足立　勝　株式会社ニコンシステム
・王　春生　Oriental Standard Human Resources Co., Ltd. Beijing
・大下　美穂　バブ日立ソフト株式会社
・岡　純一　日本インフォメーション株式会社
・小倉　博　オープンワークス株式会社
・神岡　太郎　一橋大学商学研究科
・鴨下　明　ジャパンシステム株式会社
・許　炎　通華科技(大連)有限公司
・今野　隆一　ジャパンシステム株式会社
・齋藤　肇　日本海隆株式会社
・サザーランド　真理亜　横浜国立大学
・正田　塁　株式会社オージス総研
・高橋　和司　アドソル日進株式会社
・竹政　昭利　株式会社オージス総研
・田中　繁　オープンワークス株式会社
・中島　拓　ITエンジニアリング株式会社
・中原　俊政　バブ日立ソフト株式会社
・幅　洋平　バブ日立ソフト株式会社
・日高　綱弘　日本インフォメーション株式会社
・広井　政彦　株式会社ニコンシステム
・藤野　博之　NEC ネクサソリューションズ株式会社
・保坂　成利　千代田化工建設株式会社
・牧野　宏　オープンリソース株式会社
・森　稔貴　日本インフォメーション株式会社
・山口　昭二　さくら情報システム株式会社
・吉田　亮　日本アイ・ビー・エム株式会社
・若林　良二　NEC ネクサソリューションズ株式会社