情報システム大賞 日経コンピュータ NIKKEI COMPUTER

清水建設株式会社様 工事建物データベースの構築 ～生涯顧客サービスへの取り組み～

Development of Construction Building Database for Shimizu Corporation - Approach toward Lifetime Client Service -

堀内　健司
Kenji Horiuchi

概要

長引く建設不況により新規ビルの構築が伸び悩む中、清水建設株式会社（以下、清水建設）様では、顧客に対して的確なアフターサービスを提供する「ビルライフケア事業」に力を入れている。その基幹ツールとして、「建物」や「工事」の情報を統合管理する「工事建物データベースシステム」を稼動させた。各種情報は着工から施工中、竣工後の建物のライフサイクルの中で、ワークフローを通じて一貫して蓄積し、活用される。このシステムは、アフターサービスを推進して顧客満足度を向上させ営業拡大を図る為の「建物版CRM」と捉える事ができる。

システム化の計画から稼働までは10カ月という短期間であったが、本格的な「オブジェクト指向」の採用や開発プロセス・プロジェクト体制の工夫等の取り組みにより、スケジュール通りに稼動を開始した。

尚、このシステムは、経営戦略と融合している点が高く評価され、日経コンピュータ誌「第7回情報システム大賞」でグランプリを受賞した。

1. はじめに

清水建設様は、「顧客第一」という経営理念に基づき、生涯に渡っての「ライフサイクルパートナー」として顧客をサポートしている。これを「生涯顧客サービス」と位置づけ、経営戦略と融合させた取り組みを行っている。その一環として、建設業の商品・サービスそのものである「工事」や「建物」の情報を一元化し活用してゆく為の「工事建物データベースシステム」を2002年6月に本格稼動させた。インテック（以下、当社）は、「ITパートナー」としてシステムの分析・設計・開発を行った。清水建設様は、このシステムを基幹ツールとして活用し、顧客満足度の向上と営業拡大に取り組んでいる。

一般的に、顧客満足度の向上と営業拡大を図る為のソリューションとして、CRMが挙げられる。当社でもCRMを基幹ソリューションとして展開しており、以下の様に定義している。

【インテックが考えるCRMソリューションの定義】

（1）企業が、顧客との距離・時間を短縮する目的で、顧客情報を保有し、社内共有化を図り、分析・対応することにより業績向上に結びつける動きをサポートする「システム全体」をいう。

（2）企業が顧客に対応する業務部分をカバーするシステムである。

(3) システムを構成するコンポーネントは状況により変化する。

建設業の商品である「建物」のライフサイクルは非常に長い為、「建物」を「顧客」と同一視する事ができる。この為、当論文においては清水建設様の今回の取り組みを「広義のCRM」と捉え、構築事例として紹介する。

尚、「工事建物データベースシステム」は、経営戦略と融合させたシステムである点が高く評価され、日経コンピュータ誌「第7回情報システム大賞」でグランプリを受賞した。

2. システム化の背景と目的

長引く建設不況により新規ビルの構築が伸び悩む中、清水建設様では、建物を維持する為の診断やメンテナンス、資産価値を高める為の改修・リニューアルを中心としたアフターサービスを提供する「ビルライフケア事業」に力を入れている。「建物」のライフサイクルは長く、数十年に及ぶ事が多い。新築工事が終了しても、そこから顧客との新たな長い付き合いが始まるのである。その中で顧客からの問合せに迅速に対応し、定期保全等の適切なアフターサービスを行う事で顧客との信頼関係を築く為には、「建物」や「工事」等の情報を高い精度で統合管理し、関係者が容易に検索可能なツールが必要とされた。この様な背景の下、「工事建物データベースシステム」が計画された。「工事建物データベースシステム」により適切なアフターサービスを推進する中で、新規受注等の新たな営業展開が期待できる他、各種文書や竣工報告書作成での入力業務が省力化され、全国の作業所での業務効率化が期待できる。「工事建物データベースシステム」は、いわば、顧客満足度を向上させ営業拡大を図る為の「建物版CRM」ともいうべきシステムなのである。

3. システムの概要

このシステムの最大の特徴は、従来、社内各部門に散在していた各種情報と、各事業所や倉庫等に分散して管理されていた設計図や竣工写真、工事の際の様々な帳票や記録などをデータ

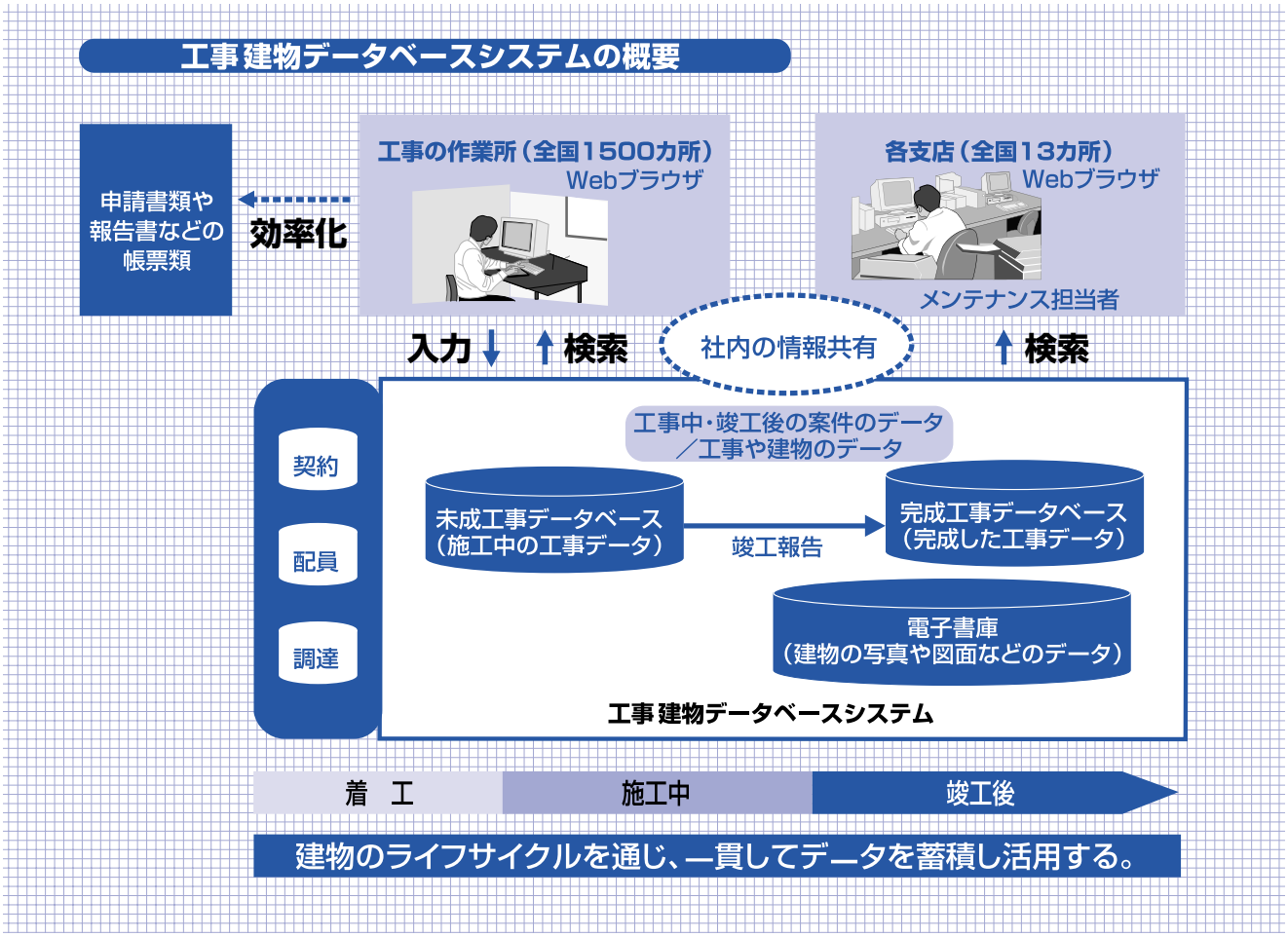


図1 工事建物データベースシステムの概要

ベース化し、一元的な管理を可能にした事である。

「工事建物データベースシステム」では、「建物」や「工事」に関わる様々な情報を施工中の段階から登録を開始する。竣工時には、入力された情報から電子的に竣工報告書を作成する。竣工報告はワークフロー機能によって関係各部門の承認・確認を得る。竣工後には点検・診断・修理等、建物のライフサイクルの各タイミングで発生した履歴情報を随時追加してゆく。つまり、「工事建物データベースシステム」はひとつのワークフローシステムであり、建物のライフサイクルを通じて、一貫してデータを蓄積・活用する為の仕組みである、と言える。この、「建物のライフサイクルを通じた一貫性」という点が非常に重要なポイントである。例えば、数十年あるいは100年を超える建物のライフサイクルの中では、担当者が変わることもあるし、所有者が変わる事さえある。又、保全工事が繰り返され、時には大規模なリニューアル工事が行われるかもしれない。さらに、一つの建物を新築し保全してゆく為には社内の様々な部門が関わる為、情報が分散してしまう可能性がある。この様な状況で、顧客・清水建設様双方にとって有益なアフターサービスを推進する為には、建物の諸元や工事情報の履歴、設計図面等を「ライフサイクルを通じて一貫して蓄積する事」が必須なのである。システムの概念を図1に示す。

「工事建物データベースシステム」は全体的な概念であり、実際のデータベースは大きく以下の3つに分類される。

① 未成工事データベース

施工中の「建物」・「工事」・「工事担当者」・「発注者」・「協力会社」・「技術」等に関する文字（あるいは数値）情報。

② 完成工事データベース

竣工後の「建物」・「工事」・「工事担当者」・「発注者」・「協力会社」・「技術」・「保全担当者」・「点検」・「診断」・「改修」等に関する文字（あるいは数値）情報。

③ 電子書庫

設計図面、建物診断書、長期保全計画書、竣工写真等の文書や提出物。

これらのデータベースには、現在、全国約44,000棟の建物と約77,000件の工事情報が蓄積されている。未成工事や完成工事の登録・閲覧は、全国1,500カ所の作業所と13の支店からWebブラウザを通じて行われる。建物の増改築や修理をする場合には、過去に行った工事の設計図面や竣工写真、各種文書を閲覧する必要があり、かつてはこれらの情報はマイクロフィルムや紙で大量に保管されていた為、簡単に閲覧する事ができなかった。「工事建物データベースシステム」は、これらの設計図面や文書もWebブラウザを通じて閲覧する機能を持っている。又、この他に、エンドユーザに対して集計ツールを提供し、データの抽出・加工を可能にしている。これらの様々な機能を使用して、工事担当者・技術スタッフ・営業マン等が建物の図面や点検記録・診断記録・改修の資料等を共有し、顧客からの問い合わせ対応や、保全作業を始めとするアフターサービスを行ってゆくのである。

次にシステムの構成について述べる。システム構成は図2の通りである。

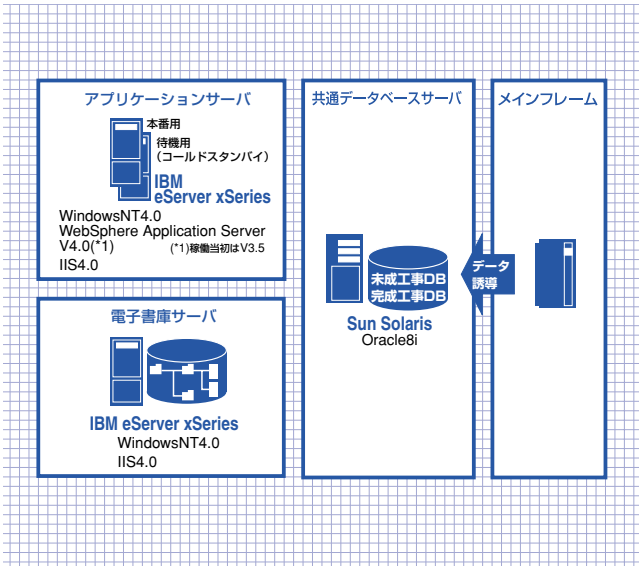


図2　システム構成図

アプリケーションサーバにおいては、未成工事データベースや完成工事データベースを登録・閲覧するアプリケーションが稼動している。電子書庫サーバは、図面や文書等のデータを保管するサーバである。共通データベースサーバには、未成工事や完成工事の各種データや管理情報を持つ。この外に集計ツールによるデータ活用や連携ツールによる他システムとの連携が行われている。この様に、文字情報と図面・文書を統合管理し、様々な角度から活用が可能になっている。

4.　短納期・低コストへの挑戦

4.1　オブジェクト指向への取り組み

清水建設様では開発期間の短縮とコスト削減を目的として、オブジェクト指向を積極的に採用している。「工事建物データベースシステム」の開発でもオブジェクト指向とJ2EE（Java）が採用された。

今回の開発でのオブジェクト指向の適否について次に述べる。オブジェクト指向とはそもそも「世の中に実在するオブジェクト（物や概念）をモデリングしてそのまま実装する」という考え方である。建設業のビジネス領域には様々な「物」や「概念」が存在する。「物」の代表例が「建物」であり、「概念」の代表例は「工事」である。その他、「工事担当者」・「発注者」・「協力会社」「技術」「保全」等、様々な概念が存在する。これらの情報は、既にデータベースとして存在していた。しかし、長期的に統合管理して様々な切り口から活用する為には、今一度見直して整理分類してゆく必要があった。この様に、ビジネス領域に存在するオブジェクトを洗い出してモデル化する事がオブジェクト指向のアプローチそのものである。正しくモデル化して実装に繋げる事が、より実世界に近いシステムを作り上げる条件である。その意味でオブジェクト指向によるアプローチは最適であったと言える。但し、正しいモデリングを行う為には、オブジェクト指向による高い分析能力と業務への深い理解が必須である。例えば、「工事情報」ひとつを取り上げる場合でも、部門によっては「建物」の切り口で捉える事もあるだろうし、「協力会社」の切り口で捉える事もあるだろう。これは、オブジェクトに対する「観点（ViewPoint）」の違いにより発生する現象であり、モデリングの初期段階で「構築対象システムにとっての観点（ViewPoint）」を明らかにし、共通認識する事が必要である。ユーザ企業である清水建設様が自らオブジェクト指向に対するノウハウを持っているからこそ、この様なアプローチが有効に機能したと言えるであろう。オブジェクト指向のノウハウと業務知識のどちらが欠けていても、概念モデリングの過程で混乱に陥ってしまったと思われる。

「観点（ViewPoint）」を確立するという点で、清水建設様は独自の工夫を行っている。通常はUML(Unified Modeling Language)のユースケース図を利用することで「観点（ViewPoint）」を確立してゆくが、清水建設様ではLFD(Lane Flow Diagram)と呼ばれるビジネスフローダイヤグラムの一種と画面モックアップ（HTMLによるプロトタイプ）を使用して検討を進める。まず、LFDと画面モックアップを用いて実際にシステムを利用するユーザ部門と共にビジネスフローの検討を行う。要求定義段階での成果物は、原則としてLFD・画面モックアップのみであり、エンドユーザ用・開発用と二重のコストをかける事が無い。要求定義の次は、「工事」や「建物」などのオブジェクトを洗い出し、そのオブジェクト間の関連を明らかにして業務全体を表す概念的なモデルを作成する。この段階ではUMLのクラス図を使用する。そしてさらに概念クラス図に対して詳細な検討を加えるのである。

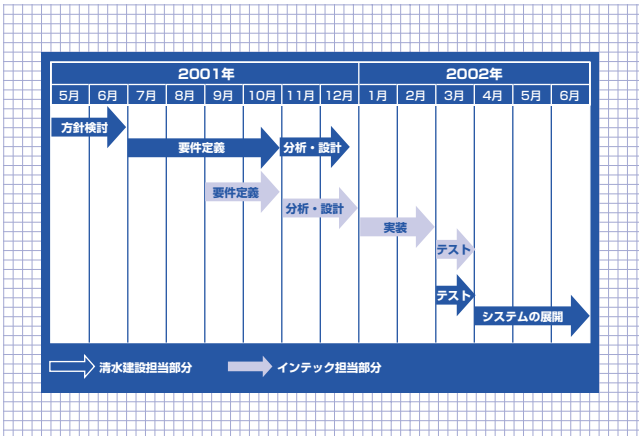


図3　開発スケジュール

ここで開発スケジュールについて触れておく事にする。図3は今回のプロジェクト開発スケジュールである。

今回の開発では、システム化の計画から稼働までが10カ月という短期間で実現されている。オブジェクト指向のメリットとして、「複雑な物をモデリングにより細分化し、単純にする」という点が挙げられる。最終的にはオブジェクトの単位はプログラムの単位となってくる為、オブジェクトの分割が適切に行われていれば、プログラムが単純化され実装工程はプログラマを大量投入する事で工期が短縮できる。今回の開発では設計から結合テスト迄が5ヶ月、そのうち実装（プログラム開発）を2ヶ月という短期間で実施しており、その効果を実感できた。但し、短期間のプログラマ大量投入に対して、プロジェクト体制を考慮する必要がある。この点については後に述べる。

ここまで述べてきた様に、ユーザ企業である清水建設様自ら開発プロセスや手法について定義を行う事でエンドユーザと情報システム部門・SIerの距離を近づけ、開発期間の短縮とコスト削減を狙っている。

4.2　プロジェクト体制の工夫

清水建設様内のプロジェクト体制にも工夫がされている。全体指揮とユーザ部門や経営層との折衝を行う「プロジェクト・リーダー」、設計や技術面の指揮、プロジェクト・リーダーの補佐を行う「オペレーション・ディレクター」、システムの設計・開発とベンダー、SIerとの調整を行う「オフィサー」という3層に分け、役割分担を明確にした。当社はSIerとして各オフィサーと協議をしながら設計から開発までの全ての工程を担当した。このような役割分担の明確化が、効率的なプロジェ

クト推進を可能にした。

一方、SIerである当社内部のプロジェクト体制であるが、「より高度な分業化」を目指している。Javaを使用したオブジェクト指向開発では技術要素が多岐に渡る上に短納期であることが多い。又、システム要件が充分に固まらずに納期だけが決まっている場合もある。さらにオブジェクト指向に熟練した技術者の不足はIT業界共通の課題である。この様な状況下では、より専門化・分業化されたプロジェクト体制を取り、専門の技術者を様々なプロジェクトでシェアしながら、さらに新たな技術者を育成してゆく必要がある。具体的には、標準的なプロジェクト体制や開発プロセス、人材育成のスキームを作成し、明示する事が重要である。図4に、標準的な開発プロジェクト体制の例を示す。

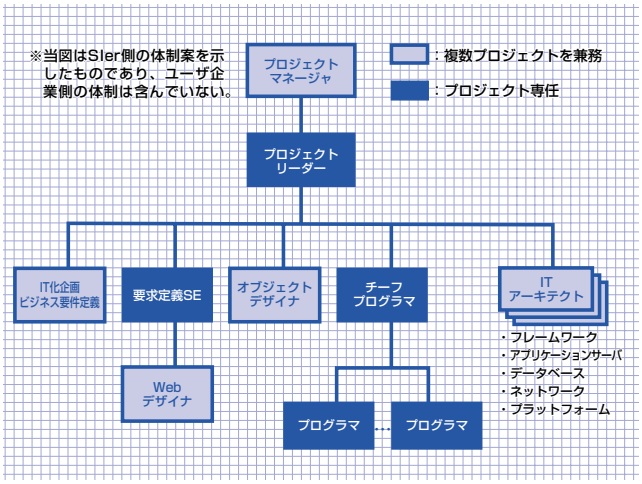


図4　オブジェクト指向開発での標準的なプロジェクト体制（例）

開発に必要なスキルは様々である。システム化企画、要求定義、モデリング、アプリケーションフレームワークの適用技術、アプリケーションサーバの導入チューニング等多岐に渡る。これらを同時に満たす技術者を育成する事は困難を極める為、専門化・分業化が必要になってくる。又、先に述べた様に、短期間のプログラマ大量投入に対する考慮も必要である。プログラマの作業割当を確実にコントロールしつつ、コードインスペクション等により設計に対して正しい実装がされているかをチェックしてゆく為に、チーフプログラマの役割が必須となってくる。特にオブジェクト指向とJavaによる開発では技術面及び短納期によるリスクが高く、実装に特化したリーダーシップが重要なのである。

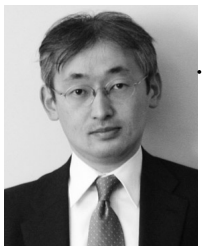
4.3　実装を進める上での考慮点

実装を進める上でも専門化・分業化が有効であり、MVCモデル[*1]の様な階層モデルにおいて特に効果が出る。例えば、画面制御部分とビジネスロジック部分で各々担当を分ける事で開発効率が向上する。又、Struts[*2]等のフレームワークを利用する事で、プログラマがアプリケーション固有の部分に注力する事が可能である。フレームワークの導入は前述した体制図に示した「ITアーキテクト」が行い、一般のプログラマが深く理解する必要は無い。

モデリングツールの利用も効果的である。現在、当社にモデリングツールを導入し、分析・設計から実装までを最短で結ぶ為のインフラストラクチャとして利用している。モデリングでの効率UPももちろんであるが、実装に入った段階でスケルトン（雛形）を作成する事が可能であり、プログラマの生産性が上がる。又、スケルトンの利用はプログラマへの属人性を下げる事になり、ひいては品質向上に繋がる。

5.　おわりに

清水建設様では「生涯顧客サービス」を加速させる為に、「工事建物データベースシステム」の拡充を進めている。当社としても開発手法の研究・人材の育成を進め、清水建設様のビジネス要件にリアルタイムに対応できる体制を作っていきたいと考えている。又、技術の横展開を図り、当社の他プロジェクトとの相乗効果を生んでゆきたい。



堀内　健司
Kenji Horiuchi

・アウトソーシング事業本部
　システム・ソリューション部
・オブジェクト指向を始めとする開発プロジェクトのプロジェクトマネジメントに従事

*1　MVCモデル…アプリケーションプログラムを、Model（ビジネスロジック）・View（表示・入出力）・Controller（ViewとModelの制御）の3層に分けた設計モデルの事
*2　Struts…オープンソースのWebアプリケーションフレームワーク。ApacheやTomcatと同様、「Jakarta Project」で開発・提供されている