

新開発二輪車用部品コスト管理システムの紹介

Newly Developed Cost Management System for Motorcycles

土 屋 晴 嗣*

Seiji TSUCHIYA

渡 辺 雅 子*

Masako WATANABE

山 本 勝 暢*

Katsunobu YAMAMOTO

荒 牧 崇*

Takashi ARAMAKI

要 旨

企画から試作、量産へと機種開発が進む中で二輪車一台分の部品コストを効率よく積み上げるシステムを開発した。これは生産部門で管理している生産部門部品表と、開発部門で管理している開発部門部品表を一体化させ、量産部品や開発部門の新機種部品にコストおよびサプライヤ情報を連携することを特徴とする。また、本システムはWeb要素技術であるJava言語を採用して開発した。さらに海外での活用を考慮し、保守およびネットワークや端末への負荷を抑える手法を取り入れて開発した。これらの結果、新機種開発のコスト変動をとらえることが容易になり、日本だけでなく海外の事業所においても機種コスト算出作業の効率が向上した。

ABSTRACT

The system, which efficiently adds up parts cost of one vehicle of motorcycle, amid the progress of model development from concept stage to mass production through prototype stage, has been developed. This is characterized by the capability of linking cost and suppliers' information of mass-produced parts, and parts of new model of Development division, between Manufacturing BOM (Bill of Material) managed by Manufacturing division and Development BOM managed by Development division. This system has been developed with Java, one of the fundamental technologies of webification. Moreover, such methods has been introduced as lowering the maintenance cost and the load on network, PC clients, to consider the overseas use. Consequently these made it easier to see the cost changes of a new model being developed, and helped to improve operation efficiency in calculating model cost in overseas offices as well as offices in Japan.

1. まえがき

二輪車開発のコスト管理は購買部門で一部ホストコンピュータを用いて、主に表計算ソフトで管理していた。しかし、近年、開発機種数が増加傾向にあり、海外生産も拡大し、このような従来のコスト管理手法では情報の収集に多大な時間がかかり、開発期間の短縮に寄与することが難しくなった。また、海外生産されている機種一台分のコスト管理を行う上では現地調達部品のコストやサプライヤ情報も、日本から供給される部品のコストと併せて管理する必要がある。そのため開発部門部品表⁽¹⁾と生産部門部品表を連動させたコスト管理システムが必要になってきた。今回、開発部門・購買部門・生産部門で効率よく活用できるコスト管理システムを開発した。

2. 開発コンセプト

2.1. 機種コスト管理における課題

従来、開発部門部品表は機種の開発に関する技術情報を、生産部門部品表は生産にまつわる部品コストやサプライヤを管理することを目的とし、それぞれの部門で独自に管理されている。そのため新機種のコストを管理するためには二つの部品表を開発部門および生産部門から取り寄せ、多くの工数を費やし手作業で一つにまとめる必要があった。また、海外の生産部門で管理している現地調達部品のコストを取り寄せるためにも多くの時間を要している。これらのことから機種コスト算出を行なうためには、多くの時間と工数が必要とされていた。さらに、これは日本だけでなく海外事業所でも同じ課題を抱えていた。課題を以下にまとめた。

*朝霞研究所

- ・開発部門部品表と生産部門部品表が一体で管理されていない
 - ・新機種コストを効率よく算出する道具がない
 - ・海外事業所の現地調達部品の情報取得に要する時間が大きい
- 上記の課題を解決するため、本システムでは、二輪車の部品におけるコスト管理システムとして共通治具化を目指し、日本および海外事業所において機種開発に必要な情報を双方向で効率よく活用できる情報共有ツールを提供することを目標に開発を行った。

2.2. システム要件

課題から求められるシステム要件を以下に記す。

- (1) 開発部門部品表と生産部門部品表の一元化
- (2) 機種一台分のコスト算出機能提供
- (3) 海外事業所における現地調達部品情報の共有
- (4) グローバル活用の実現(高レスポンス, 24時間稼働, 現地語対応)

本システムは海外利用ニーズが非常に多いため、基本アーキテクチャを Web 形式としてシステム開発を行った。

3. 開発手法

3.1. システム開発プロセス

3.1.1. 開発スタイルの変革

近年、めまぐるしい時代の変化に伴い多様化しているユーザーニーズに即応できるシステム開発が必要になってきた。また、分散、並行、レプリケーション、セキュリティ、負荷バランス、フォールトトレランスといった繰り返し現れるアーキテクチャ上の問題は、より複雑になってきている。システムの構築と保守には、基盤になるシステムモデル(プログラムとデータベース構造)の設計が重要な要素になっている。複雑なシステムをモデル化する理由は、システム全体を完全に把握することが困難になってきたからである。コンピュータがメインフレーム中心からパソコン・インターネット中心にシフトしたようにシステム開発手法の考え方も変化してきている。従来のシステム開発の手法では表現できなかったような設計図を部品化設計(Object Oriented Approach: 以下, OOA という)やデータ流設計(Data Oriented Approach: 以下, DOA という)といった相反する開発方法論を融合してシステム開発を行う必要性が出てきた。

3.1.2. 開発プロセスと目標

本システムでは OOA, および DOA が持っている長所を融合し、柔軟性、再利用性のあるシステムを目指した。開発プロセスの各フェーズごとの具体的目標は以下のとおりである。

(1) ユーザ要件定義

- ・ユーザ要件書(ユースケースシナリオ)
- ・業務フロー(Business Flow Diagram)
- ・プロトタイプ画面

(2) 分析・設計

- ・ユースケースを元にしたクラス図(プログラム構造)
- ・データベースモデル
- ・スケルトンソースコード(プログラムひな形)
- ・テスト仕様書
- ・運用手順書

(3) プログラム開発・テスト

- ・プログラムソースコード
- ・テストレポート

以上、各フェーズごとの成果物としてまとめを行った。

3.2. システム設計手法

3.2.1. ユーザ要件定義

ユーザ要件定義として以下の目的を具体的に表現するために本システムでは開発の初期段階に業務フローを作成した(Fig. 1)。業務フローを元にプロトタイプ画面を作成し、使用者と最終的なシステムイメージを相互で確認した。開発の初期段階でこの作業を行うことによってシステム開発の方向性にずれが発生しないよう考慮した。

(1) 組織構成やその流動性を理解する

(2) 効率よい運用体制を目指して、使用者、開発者が共通の理解を持つ

(3) システム要件を引き出す

(4) システムに必要とされる要件に関して使用者や関係者と同意を得る

(5) プログラム開発者にシステム要件をしっかりと理解してもらう

(6) システム化の範囲を明確にする

3.2.2. DOA によるデータベース設計

現場で使われている帳票や表計算データそのものに着目し、業務フローより時系列にとらえ論理的・分析・モデル化を図ったデータモデリング^{(2)~(4)}を行った。本システムでは先々のビジネスモデルの変更に對して、データベース構造の論理性を保持したシステム変更が容易に行えるように市販のツールを採用した。

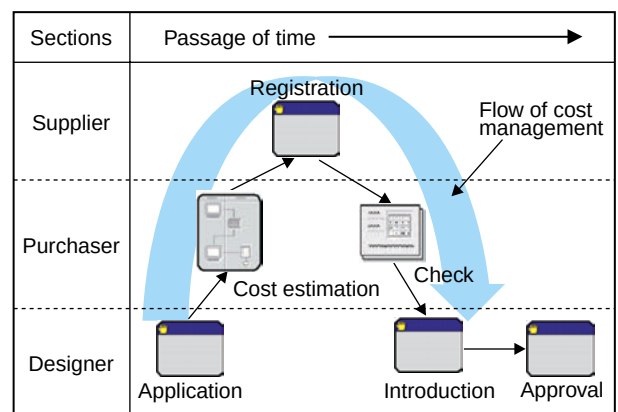


Fig. 1 Business flow diagram

3.2.3. OOA によるプログラム設計

本システムは設計段階でプログラム数量がおおよそ 700 本と試算されていた。そのためプログラム全体の関連性や再利用性、品質といった面を考慮しオブジェクト指向言語を採用しプログラムの開発を行った。これは直面している課題をすべて一から開発するのではなく、過去に用いた課題に対するプログラムをシステム全体で再利用し積み上げていくといった、いわば積木を重ねるようにプログラムを連結していく手法である。設計段階では、繰り返しこのプログラムの再利用を試み、そのたびに積木の修正を繰り返すよう設計が完成される。また、3.2.2. 項で述べたようにデータをモデル化していく工程と同様にプログラムをモデル化する必要がある。そこでプログラム自体の構造もビジネスロジックレイヤ(Model)、画面表示レイヤ(View)、制御レイヤ(Control)とに役割を明確に分け、共通に使われるプログラム群を抜き出した上で積木を部品化⁽⁵⁾し再利用を図った。これにより先々のシステム変更に対応できるよう設計した。

4. システム化

4.1. システム構成

本システムは Fig. 2 のようなシステム構成を採用した。クライアントから要求される処理はすべてアプリケーションサーバで制御され実行する。また、定期的に要求される処理はジョブコントローラにより管理され、定められた時間によりアプリケーションサーバで実行される。アプリケーションサーバに処理を集中させることで、クライアントに対する負荷の軽減を行い、システムのメンテナンス性を向上させた。アプリケーションサーバで実行された処理はデータベース管理サーバ上のデータベースにアクセスし参

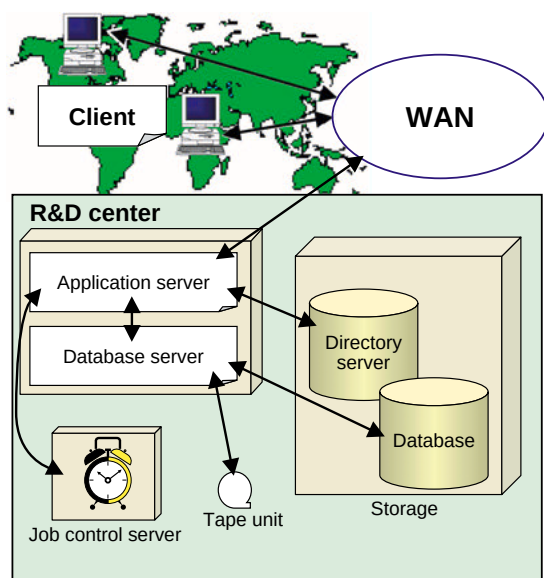


Fig. 2 System architecture

照および更新を行う。サーバをアプリケーションサーバとデータベース管理サーバとに分け、サーバに対するシステムの負荷を分散させた。

4.2. システム化技術

4.2.1. 技術部品表と生産部品表の一元管理の実現

本システムでは Fig. 3 のように管理形態の違う開発部門部品表(Engineering Bill Of Material : Engineering BOM)と生産部門部品表(Factory Bill Of Material : Factory BOM)をデータベース上で定義し相互の関連づけを行い、あたかも一つの部品表であるように表現させた。

新規図面が出図され部品番号が発行されると技術部品表システムから本システムに出図情報が転送される。また、生産部品表システムからコスト情報を取り込む。さらに技術部品表の情報を基本とし生産部品表のコストを掛け合わせることで一つのコスト部品表(Cost Bill Of Material : Cost BOM)を生成する仕組みを本システムで開発した。

これにより一元管理された技術部品表と生産部品表を同時に把握することが可能となった。使用者が従来のように二つの部品表を読み取り、表計算ソフトへ再入力することなく本システム上でコスト算出が自動的に行えるため、作業効率が大幅に向上した。また、技術部品表情報に影響を与えることなく生産部品のコストに関する登録や改廃が行えるため、開発部門の技術仕様の変化に追従したコスト変動管理が可能となった。

4.2.2. セキュリティ管理の実現

本システムではセキュリティに関するロジックを Fig. 4 のように設定することで、使用者に対してのセキュリティが機種単位に制限されるよう設計した。

予め使用者管理サーバ(以下、ディレクトリサーバという)へは使用者情報を、またデータベースには機種権限情報を設定している。

使用者が本システムへログインする際に取得したログイン ID とパスワードをディレクトリサーバにて認証を行い、

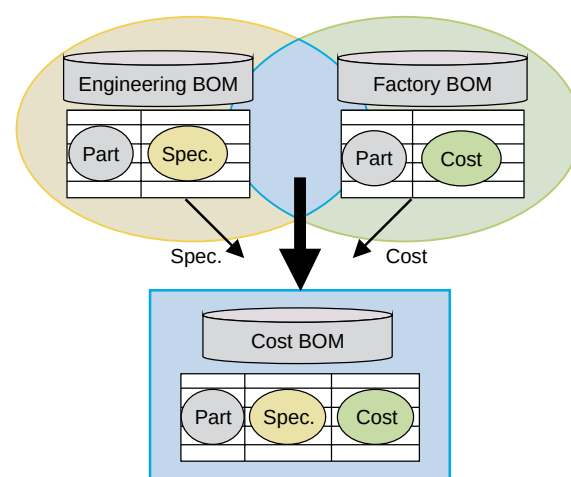


Fig. 3 Structure of cost BOM

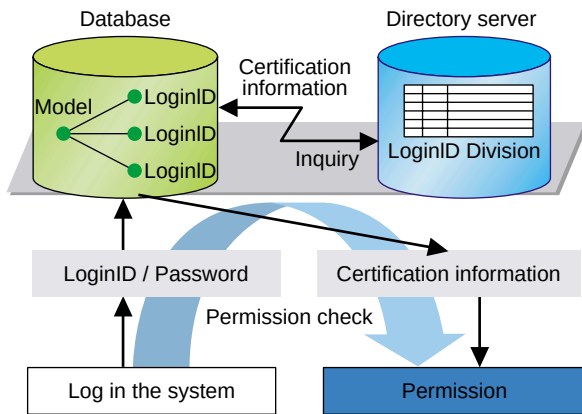


Fig. 4 Setting of authorization information

次に機種認証プログラムにてログインした使用者の機種に対する権限をデータベースに問い合わせ最終的に判断する。これによって使用者ごとに異なる権限コントロールが可能となった。この権限設定では、機種のデータ改廃に対する制限をかけるだけでなく、参照に対しても権限によって制限がかかるように設計した。結果として本システムでは事業所および他機種のコスト情報漏えいに対する高い安全性を実現した。

4.2.3. 機種一台分のコスト集計

本システムのコスト部品表は部品ごとに変化点経歴管理を行っているため、使用者が登録したコストに関する情報をどの時点においても機種一台分として集計でき、かつ、その変化点経歴時点のコストも簡単に参照できるようにした。

コスト算出方法は、部品単価に依存度(取引先別に発注する割合)、勘合率(部品の使用度合)、適用個数を掛け合わせたものに全部品の依存度、勘合率、適用個数を掛け合わせて算出する。この部品ごとのコストを集計することによって機種一台分のコストが得られる。

4.2.4. グローバル化への対応

本システムでは海外事業所への展開を視野に入れてシステム設計を行った。海外事業所は二輪事業の特色として小規模な事業所が多いため、ネットワークが弱い環境にある事業所に対して最低限のネットワークレスポンスを保証する必要がある。また、現地調達部品情報の共有化を行うにあたり、全事業所の使用者が簡単に使用できるユーザインタフェースを提供する必要がある。これらを実現するために本システムでは直感的に操作方法が分かり、どの事業所の使用者でも使用経験があるWebブラウザにて動作するシステム設計を行った(Fig. 5)。海外事業所へシステム展開するときの課題として以下のことがあげられる。

(1) 運用保守

小規模な事業所では、システムのトラブル時やバージョンアップ時にシステムをサポートする保守要員がない場合があり、トラブル対応やプログラムの維持や改廃に時差が生じる。

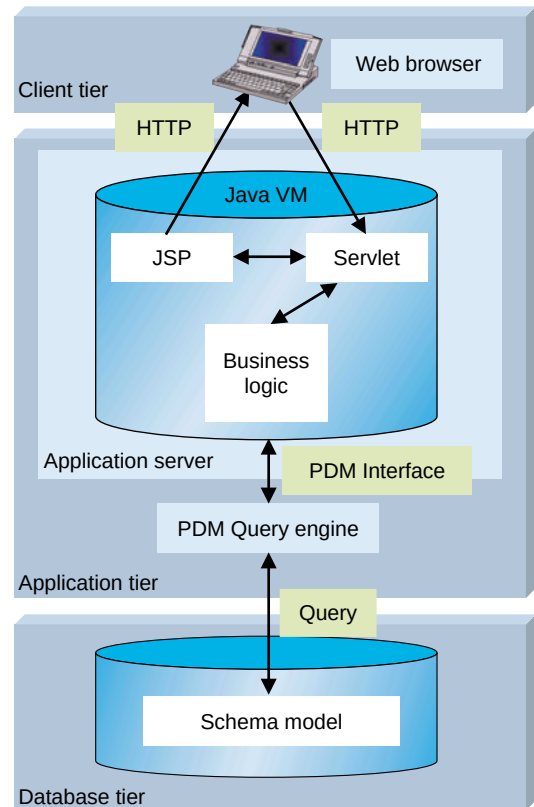


Fig. 5 Logical architecture

(2) ネットワーク

海外事業所といった遠隔地は日本とのネットワーク上に多くの通信制御機器(以下、ルータという)が介在する。これらは通信を行う上で必ず必要であるが、ルータの処理能力およびルータ数に比例して通信データがルータで滞るルータ遅延を発生しがちである。

以上の課題を解決するため、本システムではサーバサイドで処理を行う形式(JSP/Servlet形式)^{(6), (7)}を採用し、クライアントで処理する形式(Applet形式)と比べ、クライアントの負荷、ネットワークの負荷とも消費を抑え、良好なレスポンス確保を実現した。

本システムではネットワークの負荷の要素として伝送量(トラフィック量)だけでなく伝送回数(パケット数)を考慮したシステム開発を行った。Fig. 6は同一処理におけるクライアントで処理する形式とサーバサイドで処理する形式での秒間当たりの伝送回数(パケット数)を表したものである。

4.2.5. プログラム構造

プログラムは、ビジネスロジックレイヤ(Model)・画面表示レイヤ(View)・制御レイヤ(Controller)に分けたMVCモデルを採用した。Modelはシステムの中でデータベースアクセス、データ加工などのビジネスロジックを記述し、システムの本体部分にあたる。Viewは画面表示、データ入出力を担当する。ControllerはViewとModelを制御する。Controller自身は画面表示・ロジックの実行は行わず、入力に応じて必要なロジックの実行をModelに指示し、その結

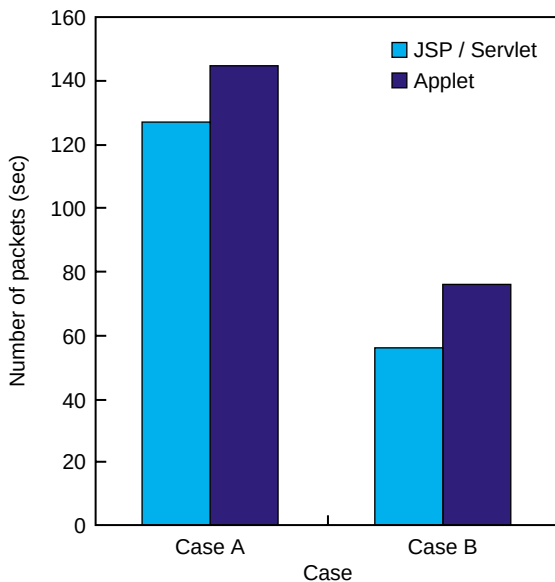


Fig. 6 Difference of the volume of packet transferred in retrieving BOM

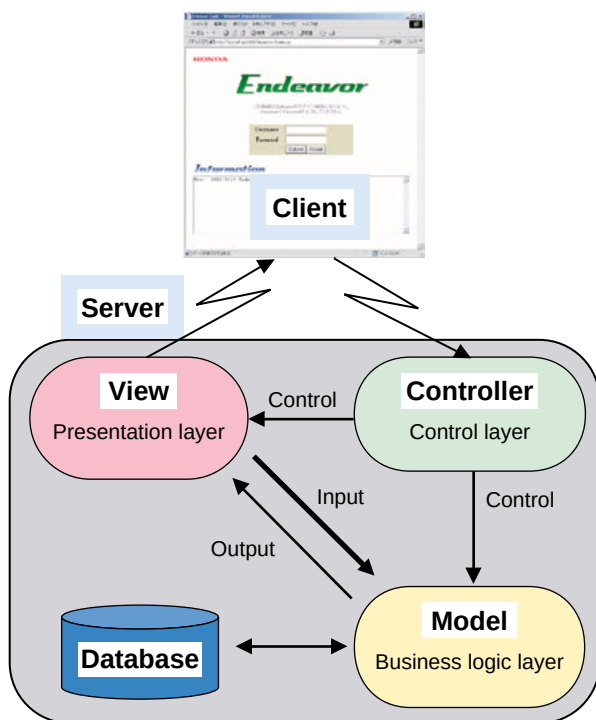


Fig. 7 MVC architecture

果を View に渡して画面表示をさせる (Fig. 7)。このモデルに従った設計を行うことで機能ごとの分離が明確になり、それぞれの独立性が確保される。開発においては、各機能が明確に分離しているため、機能ごとに分担した開発が可能となる。同時に、レイヤ間の依存が最小限に抑えられるため、ほかの部分の変更による影響を受けにくいプログラム構造にすることができる。そのため、データベースの設計を変更する場合、ビジネスロジックの変更のみで対応でき、

データベースの設計を繰り返しながらプログラムを作成していくといった柔軟な開発を行うことが可能となる。

4.2.6. 多言語対応の実現

多言語対応を行うために従来の手法ではそれぞれの言語の画面を作りシステムを提供してきた。しかし、システムの修正が発生するごとに各画面をメンテナンスするため、システム維持の効率が悪かった。

本システムではこのような問題をさけるため多言語対応可能なプログラム部品群を導入した。このプログラム部品群を組み合わせて実装することで、開発者のレベルに依存することなく統一された品質のプログラミングが可能となる。いくつかのプログラム部品群を策定した中で、MVCモデルと一体になって表示コントロールできる Struts (標準化組織である Jakarta Project によって開発された Web アプリケーション開発のためのプログラム部品群)^{(8), (9)} がシステム維持に優位であると判断した。この Struts は表示メッセージを言語ごとに管理し、必要に応じて言語を切り替える方法を提供している。Struts を利用することで、以下の4つの利点を得ることができた。

- ・メッセージは定義ファイルに一元管理されているため、各画面で文体を統一できる。表示メッセージを変更する場合もプログラムの変更は必要ない。
- ・新たに対応言語を増やす場合には、その言語に翻訳したメッセージ定義ファイルを追加するだけで実現可能となった。
- ・従来の JSP では HTML ファイル上に Java 言語を書き込むため複雑になりがちであったが、Struts が提供するタグ (画面プログラムの部品化) を使うことで整理され、開発工数を少なくできた。
- ・画面遷移情報が XML ファイルで一元管理されるため、ビジネスロジックレイヤに画面遷移の制御処理を作成する必要がなくなり、プログラムの生産性や品質が向上した。

4.2.7. 負荷分散と耐障害性

本システムではアプリケーションサーバを使用して、負荷分散 (ロードバランス) と耐障害性 (クラスタ構成) に対応したシステムを実現した (Fig. 8)。複数クライアントからの処理要求に対して負荷状況を認識し、負荷の少ないアプリケーションサーバに対して処理要求を分散する。負荷の異なる二つの処理プログラムをテストケースとして負荷分散の有効性を検証した。Fig. 9 にその検証結果を示す。この負荷分散機能により処理要求が集中している場合でもクライアントに対して均一のレスポンスを実現した。耐障害性では、複数のアプリケーションサーバを接続し、一つのアプリケーションサーバに障害が発生した場合に接続された別のアプリケーションサーバに切り替わるようにした。そのため、アプリケーションサーバに障害が発生した場合でも利用者へのシステムサービスを停止することの無いシステム運用が実現できた。

6. 今後の展開

今後、技術仕様やコスト情報の変化点がより効率よく使用者にとらえられるよう、各部門ごとに必要な情報を整理した使用者専用画面(ポータル画面)を提供していく。さらに海外事業所への展開を行っていく中で海外の要件を取込み、システムの熟成を計る。

参考文献

- (1) 山本雅貴ほか：技術部品表データマネジメントシステムの開発, Honda R&D Technical Review, Vol.14 No.1, p.77-84 (2002)
- (2) James Martin：インフォメーション・エンジニアリングⅠ～Ⅲ, トッパン(1991-1994)
- (3) 佐藤正美：クライアント/サーバデータベース設計テクニック, 株式会社ソフト・リサーチ・センター(1993)
- (4) 佐藤正美：T 字形 ER データベース設計技法, 株式会社ソフト・リサーチ・センター(1998)
- (5) Wolfgang Pree, 訳 佐藤啓太, 金澤典子：デザインパターンプログラミング, トッパン, p.75-78 (1996)
- (6) Inderjeet Singhほか：Java 2 Platform Enterprise Edition アプリケーション設計ガイド第2版, サンマイクロシステムズ株式会社, p.79-180 (2002)
- (7) 原田洋子：Java サーバーサイドプログラミング, 技術評論社, p.164-269 (2000)
- (8) James Goodwill, トップスタジオ:マスタリングJakarta Struts, 翔泳社, p.87-156 (2003)
- (9) Chuck Cavaness：Programming Jakarta Struts, O'Reilly, p.104-193 (2002)

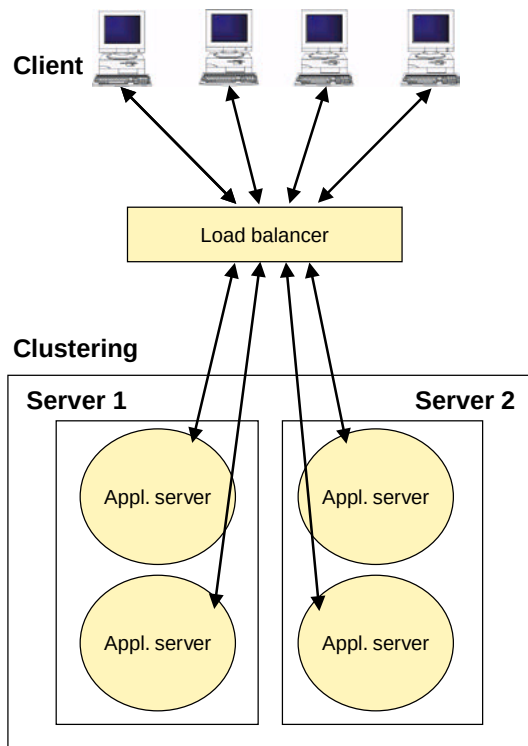


Fig. 8 Load balancing and clustering

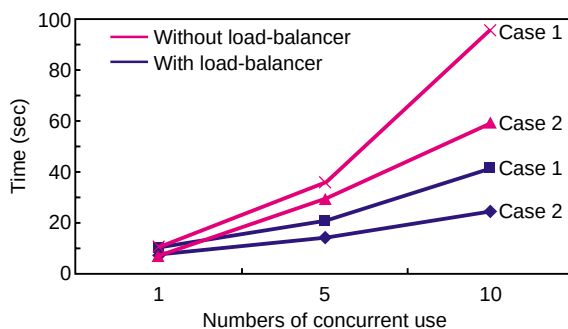


Fig. 9 Load balanced performance

5. まとめ

開発したシステムは、以下のような特徴や効果を持ち、新機種のコスト管理に有効なシステムとして構築できた。

(1)特徴

- ・将来の管理項目追加, 増加するデータに対応できる柔軟性.
- ・多言語環境およびネットワーク負荷削減の対応を行い, 海外への展開も容易にできる拡張性.

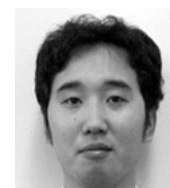
(2)効果

- ・海外生産機種も含めた開発イベントごとの機種一台分のコスト管理, およびサプライヤ情報の共有ができるシステムとした.
- ・海外生産機種の現地調達部品個々のコストや納入形態を一目瞭然で参照できるシステムとした.
- ・新機種一台分のコスト算出に要する時間を従来の 2/3 に短縮した.

■著者■



土屋 晴嗣



山本 勝暢



渡辺 雅子



荒牧 崇