

# 金融向けソリューションへのSDAS活用

## Adaptation of SDAS to Financial Business Solutions

あらまし

金融業のお客様からは、システムの新規構築や再構築において従来どおりの信頼性・品質を確保し、さらなる短納期が求められている。とくに近年の流動化するビジネス環境やIT技術の絶え間ない変化に追従すべく、最新の技術や開発手法を取り入れることも必要な時代となっている。この実現のためには従来の開発手法では限界があり、SDASを活用した金融向けソリューションの確立と展開を図る必要がある。

本稿では、まず金融業界におけるリース業向けソリューションの体系に必要な要件とリースシステム構築ソリューションの設計思想（業務サイドの設計思想、システムサイドの設計思想）を説明する。つぎに、システムサイドの設計思想を実現させている取組みを紹介し、確立したソリューションを繰り返し展開することで高品質と短期開発を実現している効果について述べる。

### Abstract

Fujitsu's customers in the financial industry have restructured and expanded their systems and now need not only the reliability and quality they enjoyed the past but also require them much more quickly. Especially these days, it is necessary to use the latest technology and development methods to keep up with the constant changes in the business environment and IT techniques. To achieve this, the previously used development methods are becoming inadequate, and it is now necessary to establish and develop financial solutions that adopt SDAS. In this paper, we first explain the requirements of the solution architecture for the leasing industry and the design concept of a solution for lease system construction that covers both the business side and system side. Next, we introduce an approach that covers the design concepts on the system side. By repeatedly using the solution that adopt SDAS, we have achieved high-quality, short-term system development.



小島 浩(おじま ひろし)  
金融ソリューションBG APソリューション部 所属  
現在、金融BGで共通技術支援に従事。



黒田正剛(くろだ せいご)  
金融ソリューションBG APソリューション部 所属  
現在、金融BGで共通技術支援に従事。

## ま え が き

最近、金融業のお客様において新規構築システムのみならず現行の大規模な基幹系メインフレームシステムをオープン環境で再構築したいという案件が多くなってきている。しかも、従来どおりの信頼性・品質を確保した上で、さらに短期間での開発が求められている。これらの要望に応えるために金融業向けソリューションでは、SDASで提供している開発技術に業務知識とプロジェクトマネジメントを包括した体系を備えている。

金融業におけるリース業向けソリューションでは、1999年にSDAS（AA/BRMODELLING<sup>(注1)</sup>ベース）を活用したリースシステム構築ソリューションを確立し、これをベースに再利用型の短期開発プロセスで多数の大手リース会社の基幹システムを再構築してきた。リースシステム構築ソリューションは、大きく業務サイドの設計思想とシステムサイドの設計思想がある。

本稿では、まず業種向けソリューションの体系に必要な要件と、上記の二つの設計思想について説明する。つぎに、システムサイドの設計思想であるフレームワークを実現させている「アプリケーション構造の体系化/標準化」「プラットフォームの隠ぺい」「性能向上への取り組み」「再利用によるWebシステム化への取り組み」「業種向けソリューションの再利用による効果」について述べる。

## 業種向けソリューションの要件とリースシステム構築ソリューションの設計思想

本章では、業種向けソリューションの体系に必要なとされる要件と、リースシステム構築ソリューションにおける二つの設計思想について述べる。

業種向けソリューションの体系に必要なとされる要件

業種向けソリューションの体系には、以下の要件を備えることが必要であると考えている。

## (1) 業種別の知識に基づく業務モデルの確立

システム化した業務の流れの可視化、およびビジ

ネスデータの基本データ構造を明確化する。

## (2) SDAS標準フレームワークに、業種特性を付加したフレームワークの確立

開発環境の枠決めを行い、その中で動作する機能を実装したソフトウェア部品（通信制御・画面制御・業務仕様など）によりコンポーネント化を図る。

## (3) 開発の手法・環境・ツールの統一

開発者による「要求定義」から「テスト」までの開発手法を統一することで、開発者ごとに存在する技術的スキルの差の改善を図る。

## (4) 開発標準、プロジェクトマネジメント基準・ツールの確立

体系化/標準化されたドキュメントを使用し、実践的なドキュメントにより記述内容の均一化を図る。また、作成されたドキュメント管理やレビュー管理などもツールにより進捗の可視化を図る。

なお、業種向けソリューションの展開では、上記で確立したソリューションに最新の技術や開発手法を取り入れながらプロジェクトで繰り返し適用し、フィードバックを行い、さらに適用ノウハウ（フレームワークの知識を前提としたコンポーネントの作成・組合せ）を有する開発体制を強化し、開発生産性と品質を高めることが重要である。

リースシステム構築ソリューションの設計思想

リースシステム構築ソリューションは、前述のように業務サイドの設計思想とシステムサイドの設計思想がある。

## (1) 業務サイドの設計思想「リース業務のモデル化」

リース業務は商品種類が多様で、リース期間中に契約内容の変更や解約など取引の状態が複雑に推移する。また、さらにそれに伴って、物件・固定資産の支払内容やリース料の回収条件も変化するため、各種機能が強固に連携したシステムを構成する必要がある。このような複雑な要件に対処するため、リース業務の基本的な商品の取引から事務処理までを一貫してモデル化（業務モデル、データモデル・論理モデル）することで、お客様が必要としている要件を整理し、追加・変更ポイントを明確にすることができる。

## (2) システムサイドの設計思想「業務仕様と制御の分離」

業務仕様とコンピュータ制御を分離するために、

(注1) Application Architecture/Business Rules MODELLINGの略。データ中心のモデリング技法を中核にした開発方法論に基づいて、分析や設計ドキュメント（モデル）の表記と意味を規定した技法。ここで規定した書き方をソフトウェアで支援するのが、SDAS統合CASEツールのAA/BRMODELLER。

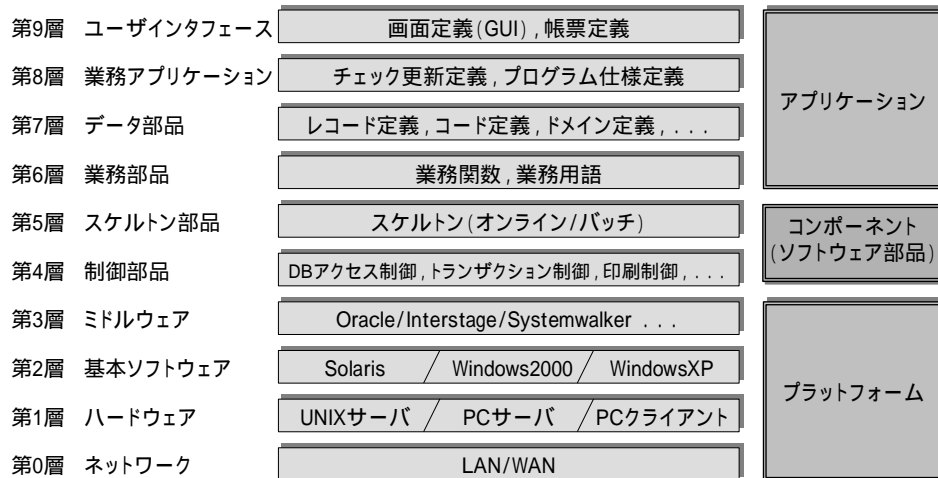


図-1 アプリケーションからプラットフォームまでの分割  
Fig.1-Devision from application to platform.

アプリケーションからプラットフォームまでを10の階層に分割している。これらは大きなブロックであるアプリケーション、プラットフォームとその二つの役割を吸収するコンポーネント（ソフトウェア部品）によって分類される（図-1）。

アプリケーション開発時において、リース業務のモデルをベースにすることで業務部品・データ部品の再利用率の向上が期待でき、既存の部品がない場合でも体系化/標準化されたインタフェースによって業務部品やデータ部品への追加・変更を容易にしている。また、アプリケーションとプラットフォーム（ハードウェア・ミドルウェア）とのインタフェースをコンポーネントで隠ぺいし分離することで、業務部品・データ部品についても再利用率が向上する。

## アプリケーション構造の体系化/標準化

体系化/標準化されたインタフェースによるアプリケーション構造を繰り返し使用することにより、設計・構築の期間を短縮し、制御ロジックでの品質低下を抑制することができる。また、その構造に沿って設計ドキュメントの体系化/標準化を図ることで、設計ドキュメントの記述レベルの向上も図ることができる。

アプリケーションはオンラインプログラムとバッチプログラム、およびサブプログラムに大別することができる。これらの内、オンラインとバッチについてはプログラム構造を処理形態によって体系化し、

22パターン（単純編集・振分け・集計編集・マッチング・テーブル更新・帳票編集など）の標準化したスケルトン<sup>(注2)</sup>を準備している。

これらのスケルトンにはトランザクション制御、DBアクセス制御、エラー処理、ログ処理、メッセージ処理、リカバリ処理などの部品が組み込まれている。

とくにオンラインスケルトンにはこれらに加え、画面制御、イベント制御、項目制御、電文制御などの部品が組み込まれている。また、サブプログラムでは業務部品・データ部品・共通部品などを標準的なインタフェースで作成するためのスケルトンとして準備している。

オンライン/バッチ/サブプログラムのスケルトンを効率的に使用するため、プログラム分割基準と各スケルトン仕様をSS（システム構造設計）工程開始から明確にし、設計書に使用するスケルトンを記入している。プログラマは指定されたスケルトンを使用することで、プログラム作成の期間短縮、制御処理の品質の維持が可能になる。

このアプリケーション構造を実現するため、SDAS統合CASEツールであるAA/BRMODELLERを使用し、スケルトンや制御部品の関数化で業務仕様とコンピュータ制御の分離・隠ぺいを実現している。

（注2）各業務仕様に合わせたひな型であり、業務フローを参考に業務パターンを洗い出す。

プラットフォームの隠ぺい

近年の開発環境は、ハードウェア・ソフトウェアの世代交代が早いいため、アプリケーションを多世代にわたり継続して使い続けるには、アプリケーションからプラットフォームとのインタフェースを隠ぺいし、変更に対応できることが重要である。

アプリケーション構造の体系化/標準化によりスケルトン、制御部品でプラットフォームは隠ぺいされているが、バッチプログラムをテストする場合において、Windowsはバッチファイル、Solarisはシェルスクリプトといった具合にプラットフォームごとに相違があるため、開発時にはプラットフォームを意識せざるを得ない。

その対策として、メインフレーム時代のJCL(ジョブ制御言語)のように、ファイルの割当てとプログラムの実行を記述し、ソートユーティリティやファイルの削除などは「ソート」「削除」と記述することで対応した。OSに依存する分岐などの記述は自動展開することによってプラットフォームの隠ぺいを実現している(図-2)。

シェルスクリプト/バッチファイル(以下、ジョブ)の自動展開機能としてプラットフォームの隠ぺい以外に考慮しておくべき以下の事項についても対応した。

- (1) ジョブの再実行が考慮されていること
- (2) 各コマンド(プログラム、ユーティリティなど)の復帰コードを自動判定すること

- (3) ジョブ終了時に不要ファイルを自動削除できること

- (4) 単体テストで使用したものが、そのまま運用ジョブとして使用できること

つぎに完成したジョブを順序よく起動するためのスケジューラへの登録が必要である。しかし、運用を考慮するとライブラリ管理や変更管理といった手順が必要で、各開発者がスケジューラをダイレクトに操作するわけにはいかない。したがって、スケジューラ情報の入力に使い勝手の良いツールが必要となる。

使い慣れたExcelを使用して、スケジューラのジョブネット情報を入力する方法は、セル(先行ジョブ名)とセル(後続ジョブ名)を線やコネクタで結ぶことでジョブネット情報を抽出し、テキストとして生成する方式とした。

スケジューラへの登録機能として以下の事項に対応した。

- (1) ジョブ間の関連情報がライブラリ管理できること
- (2) サーバ(スケジューラ)を使用しないで入力できること
- (3) ジョブの関連が視覚的に記入できること
- (4) スケジューラへ登録する前にチェック(ジョブ間のループなど)できること

現在、このツールとスケジューラであるSystemwalker OperationMGRと組み合わせて使っている。

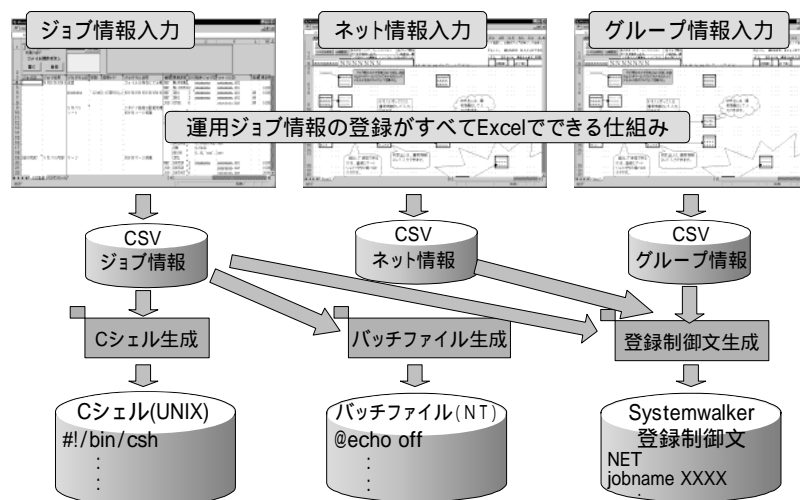


図-2 プラットフォームを意識させない運用ジョブ登録  
Fig.2-Registration of operation job not to recognize platform.

## 性能向上への取組み

品質悪化の要素として、性能問題が挙げられる。性能問題は、大量データを使用したシステムテスト工程で発覚することが多く、アプリケーションの見直しは難しい時期である。したがって、性能問題は早い段階に把握することが重要となる。

### (1) すべてのSQL内容について評価

性能問題が発生する最大の要因は、SQL命令で発生する。中でも一番の原因はデータへの実アクセス（物理ページをアクセス）である。「性能が悪く、数十分かかっていた処理がチューニングしたら数秒で終わった」などは、物理的なアクセスがチューニングによりなくなったためである。SQL文を起因とした性能問題を事前回避するために、業務アプリケーションでのSQLの直接呼出しは全面禁止し、テーブルアクセスは提供されたアクセス部品を使用する。アクセス部品の作成も一元管理し、スキルある管理者がアプリケーション開発者から申請されたSQL文をすべて評価した。

### (2) レスポンススループット情報の自動取得

性能情報を自動的に取得する仕組みをスケルトンに組み込み、結合テスト段階から性能情報を自動的に取得・累積し、テストサイクルごとに業務開発者へフィードバックを繰り返し、性能改善を支援した。累積したプログラムごとの性能情報を使用して、最終のシステムテスト/運用テストで性能的に改善されていることを確認した。

### (3) 特定トランザクションのOracleトレース情報取得

サーバのオンラインアプリケーションは、一つのプロセスが複数クライアントと会話するため、プロセスを終了させない限り特定のトランザクションのみのトレース情報を取得できない。また、取得しても複数クライアントのトランザクションが混在しているため目的のトレース情報の解析は困難である。通常であれば誰も使用していないタイミングで取得するか、専用の環境を構築することによって取得することになる。

対応策として、トランザクション制御部品に特定情報（担当のIDやプログラムのIDなど）を識別し該当のトランザクションの場合に、トレース取得開始と終了命令を発行することで、トランザクション

ごとのOracleトレースの情報を取得することを可能とした。その結果、問題SQL文の特定が早まり、調査時間の短縮が図れた。

## 再利用によるWebシステム化への取組み

リースシステム構築ソリューションは、クライアントサーバシステム方式のソリューションであったが、近年ではインターネットの普及に伴いWebシステム化した構築が一般的になっている。本章では、クライアントサーバ方式で確立した「リースシステム構築ソリューション」の再利用を考慮したWebシステム化に向けての取組みポイントを紹介する。

一つ目のポイントは、リースシステム構築ソリューションのAPサーバ側は極力再利用し、Webサーバ側のみフレームワークを構築したことである。二つ目のポイントは、業務開発要員に二つ以上の言語を使用させないことである。

Webシステムの方式には、Servlet方式と操作性を重視したApplet方式があるが、開発期間と業務要件からServlet方式を採用し、Servlet版リースフレームワークを構築した（図-3）。

このフレームワークではWebサーバ関連で業務開発者が作成するものは、画面設計として作成するHTMLと画面に対する入出力レコード定義の属性情報だけである。これらをツールに入力すると、画面ごとにJSP（JavaServer Pages）が自動生成され、属性の基本チェック・入出力時の項目編集・桁数チェックおよび共通コードによるプルダウンなどは新しく開発したServlet版フレームワークが制御するため、業務ロジックを記述する必要がない。

また、動的な情報の入出力は、アプリケーションよりServlet版フレームワークに必要な情報を指定することで実現した。

したがって、画面の制御はすべてServlet版リースフレームワークで実装しており、業務開発者がJavaなどで開発する必要がない（図-4）。

現在、Applet版リースフレームワークを構築しているが、基本的なコンセプトは同じである。

## 業種向けソリューションの再利用による効果

本章では、リースプロジェクトの確立した業種向けソリューションの利用による効果について紹介する。

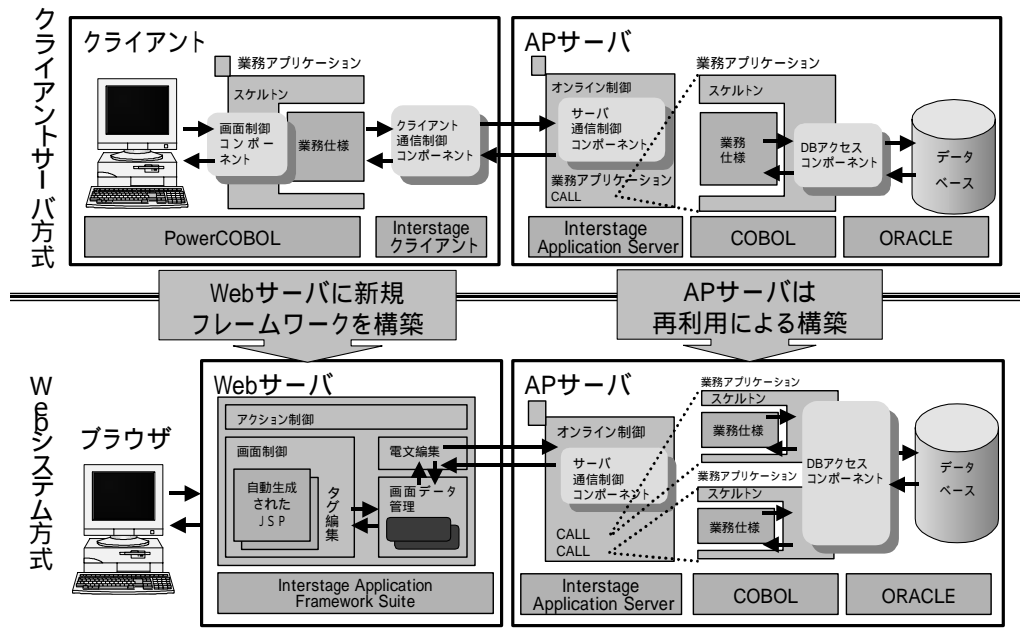


図-3 Servlet版リースフレームワーク  
Fig.3-Lease framework of Servlet version.

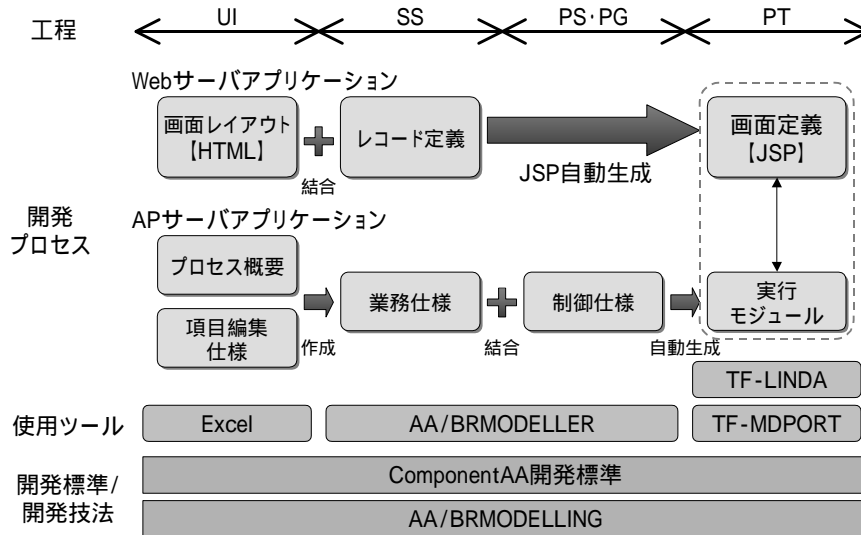


図-4 Servlet版JSP自動生成  
Fig.4-JSP automatic generation of Servlet version.

業務面では、リースシステム構築ソリューションを再利用し、「リース業務のモデル」による「Fit/Gap分析」(注3)を行うことによって、要件の抽出を実施し、顧客ニーズを詳細に仕様化することがミス・漏れの減少につながる。

システム面では、アプリケーション構造が体系化・標準化され、プラットフォームにより隠ぺいさ

れていることで、再利用を繰り返し行っても業務部品やデータ部品が再利用可能となっている。ただし、再利用する上でもう一つ重要なことは、リースシステム構築ソリューションの再利用と併せてノウハウを持った要員を継続して要員体制に入れることである。部品の利用方法やノウハウを持った要員が繰り返し携わることで、期間短縮・品質向上にとって非常に効果が大きいと考える。

本稿ではシステムサイドのフレームワークに絞っ

(注3) ERPパッケージが保持している機能とビジネスプロセスを成立させるために必要となる機能を比較する作業。

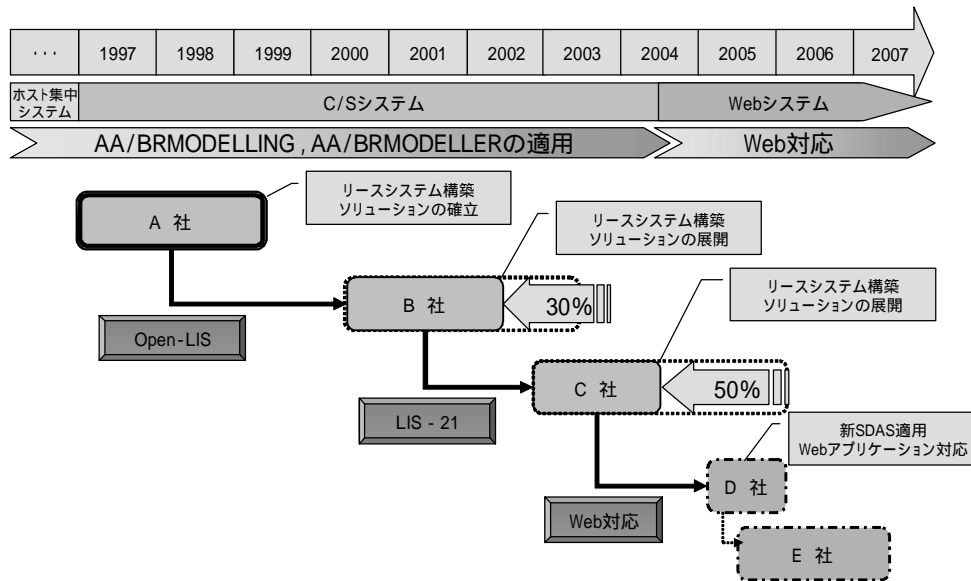


図-5 リースシステム構築ソリューションの展開  
Fig.5-Expansion of lease system construction solution.

て紹介したが、開発期間の短縮・品質の向上を実現するためには進捗管理，品質管理，障害管理，仕様変更管理，問題点課題管理，構成管理なども含めたプロジェクトの推進体制が必要不可欠である。

これまでのリースプロジェクトで「リースシステム構築ソリューション」を繰り返し再利用することによって、開発規模に対する開発期間の比率は30%～50%の削減効果があった（図-5）。

## む す び

本稿では、金融業におけるリース業向けソリューションについて、開発技術を中心に繰り返し再利用することで開発期間の短縮、品質の向上の効果を紹介した。しかし、その効果を発揮するためには開発技術だけではプロジェクトは成功しない。プロジェクトにはルールが必要で、全体を効率良く統率する

ためのツールによって可視化し、問題点を早期発見し対策を講じることが期間短縮・品質向上の大きな成功要因である。

今後、金融業界では業態を越えた商品・ビジネスの多様化により総合金融サービスへの対応が必要となる。このため提供チャネルの拡大，異業種との提携などオープンな連携による金融ビジネスモデルの変更に適応する業種向けソリューションが求められている。

富士通では、「サービスのコンポーネント化」，「SOA（Service Oriented Architecture）基盤を中心としたチャネル/サービスの連携」，「インタフェースの標準化」を特長とし，変化する金融ビジネス環境に適応する次世代金融ソリューションの拡充を進めている。