Oracle Applications 統合会計実装方法

Implementation Method for Oracle Applications' Financials Modules

稲 垣 理 佳

要 約 昨今，会計制度の変更をきっかけとして企業の経営管理者は迅速な財務報告とフレキシブルな管理会計による経営管理の実現を会計システムに求めるようになり，その実現手段としERPパッケージを選択する企業が増えてきた．

本稿ではOracle Applications統合会計モジュールの導入成功に直結する導入手法およびポイントについて，AIM導入工程のオペレーション分析フェーズにおけるConference Room Pilotの進め方を中心に紹介した．

Abstract Recently, with reformation of accounting principles as a trigger, management staffs of enterprises begin to require the realization of the business management using rapid financial reporting and flexible management accounting from the accounting system, and enterprises selecting ERP applications to realize their requirements have been on the increase.

This paper describes the installation method and the points to flow through to the success in installing Oracle Financials modules, focusing on mainly how to work Conference Room Pilot in the operation analysis phase of AIM installation.

1. はじめに

昨今，企業の多角化，国際化の急速な進展，国際証券市場への海外投資家の参入の増加など，日本企業を取り巻く環境の変化により連結財務諸表制度の変更をはじめとする会計制度の変更をきっかけとして，企業の経営管理者が会計システムに求める要件が変化してきた．その要件とは迅速な財務報告とフレキシブルな管理会計による経営管理の実現である．さらに，これらの要件を踏まえた会計システムを迅速に構築するための手段としてERPパッケージの存在が脚光を浴びてきた．一方，会計分野は生産管理などの他分野に比べ業種間，企業間での相違が比較的少ないという理由から，ERPパッケージ導入では最初に手がけられることの多い分野であるが，個々のERPパッケージの特性を活かした導入が重要である．

本稿ではOracle Applications統合会計モジュールのユーザへの導入実績により蓄積された事例を通じ，オペレーション分析フェーズで実施するConference Room Pilotを中心とした導入ポイントを紹介した．

2. オペレーション分析フェーズにおけるCRPのすすめ方

ERPパッケージ導入のメリットはOracle社などのERPベンダーから提供される「ベストプラクティス（最適モデル）」を短期間で低コストにてユーザに適用できることである．Oracle Applicationsにおいても標準導入工程「Oracle Applications Implementation Method（以下AIM）」が提供されており導入の支援となる様々なツール類を使用することが可能となっている．とはいうものの一歩間違えると導入が長期化

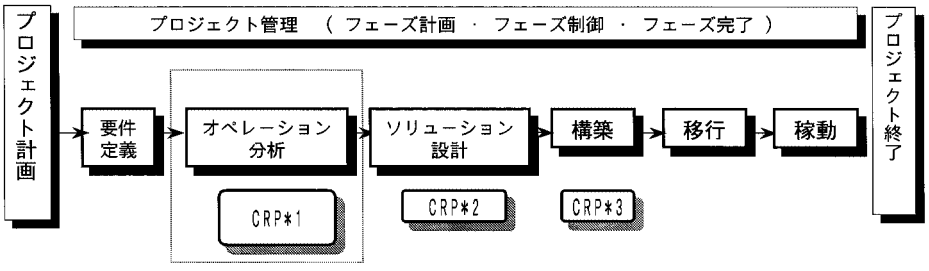
しコストも膨らみ，ユーザのめざす BPR（Business Process Re-engineering）にはとうてい結びつかないという結果になりかねない．本章では従来から行われている自社でのソフトウェア開発手法と ERP パッケージの導入手法の違い，AIM におけるオペレーション分析フェーズで行われる Conference Room Pilot（以下 CRP）について述べる．

2.1 ERP パッケージの導入手法

従来型のソフトウェア開発技法がウォーターフォール型のプロセスモデルであるのに対し，ERP パッケージの導入は初期段階ではスパイラル型のプロセスモデルを利用する．従来型のソフトウェア開発と異なる点は，従来型は工程が進むにつれて工数が増えていくのに対し，ERP パッケージの導入では全工程を通してほぼ一定となる．これは，ERP パッケージはあくまでも「パッケージ」であり本来はアド・オン（追加）開発なしで導入されるべきものだからである．むしろ ERP 導入では前半の工程での比重が大きいといえる．

2.2 CRP について

CRP とは，プロトタイピングとその評価を繰り返すという手法である．図 1 で示すように AIM のオペレーション分析（CRP 1），ソリューション設計（CRP 2），構築（CRP 3）において各フェーズの目的に合わせた CRP を実施するが，特にオペレーション分析フェーズにおいて行う CRP 1 は，ベストプラクティスを目指した新しいビジネスプロセスの設計と決定という意味で最も重要である．一般的に CRP とは CRP 1 を意味する（本稿では以降，CRP 1 を CRP と記述する）．CRP は，ユーザに合わせた形でパッケージをプログラミングすることなく，Oracle Applications の設定によりオン・マシーンにてビジネスプロセスの実現性の判断を繰り返し行うというところが特徴的である．



ＣＲＰ１：オペレーション分析フェーズで実施されるＣＲＰ。
オン・マシーンによりＦＩＴ&ＧＡＰを行う。

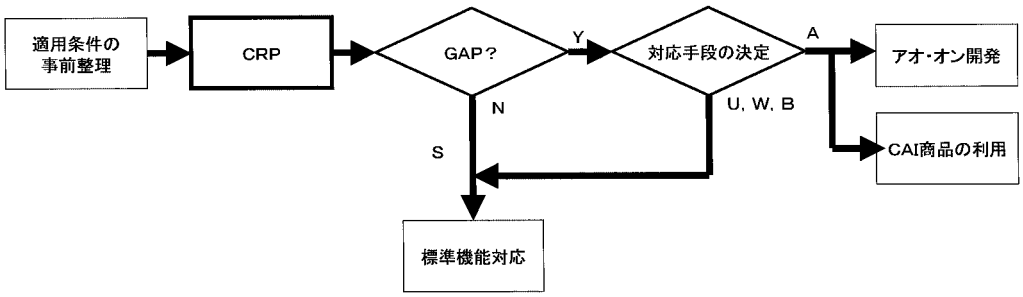
ＣＲＰ２：ソリューション設計フェーズで実施されるＣＲＰ。机上にてＦＩＴ&ＧＡＰを行う。
（論理設計レビュー時に実施）

ＣＲＰ３：構築フェーズの最終段階でユーザが行う最終検証のためのＣＲＰ。
（検収テスト確認時に実施）

図 1　AIM 工程における CRP 実施フェーズ

図 2 で示すのは，CRP を中心としたインプット成果物，アウトプット成果物との

関係である．CRP 実施前には Oracle　Applications 適用条件の事前整理（3 章参照）を行い，CRP にて定義結果の検証を行う．CRP の結果としてビジネスプロセスについて対応手段の決定を行う．



対応手段の種類	内　　容
S （ Standard ）	適合
U （ セットアップ ）	付加フレックス・フィールド、FSG、アラートで対応
W （ Work Around ：運用改善）	現場レベルの業務を変えてAPPSに合わせる。
A （ Add On ）	外付け、新規開発
B （ BPR ）	BPRで対応（プロセス・フローの再検討要）

図 2　CRP のインプット＆アウトプット

2.2.1　CRP 実施の重要性

ERP パッケージ導入のプロジェクトにとり，最も重要な作業はベストプラクティスを目指した新しいビジネスプロセスの設計と決定である．CRP では，ERP パッケージに合わせてユーザのビジネスプロセスの代替案を作成し，新しいビジネスプロセスの決定を行うことが要求される．CRP 実施の結果の如何により ERP パッケージへのアド・オン開発要件も大きく変動し，開発コストと期間を左右することになる．つまりパッケージでの代替案が見つからない場合はアド・オン開発要件が膨らみ，極端な場合はパッケージそのものが適用できずに従来の手作り開発という道を選択する結果にもなりかねない．

CRP 実施の結果，アド・オン開発と決定した部分は，ERP パッケージ・ベンダーの提供する開発ツール（Oracle　Applications では Developer 2000）を利用することによりパッケージと親和性のあるカスタムアプリケーションの開発を比較的柔軟に行えるが，本質的には手作り開発であることに変わりはなく，1）開発コストと本番運用後の保守コストがかかる，2）パッケージ・ベンダーによるバージョンアップの採用を困難にする，という点で ERP パッケージ活用によるメリットである 1）導入期間短縮，2）開発費用の低減，3）バージョン・アップによる IT 新技術の継続的な適用，等を失ってしまうことになる．

このような意味で CRP は ERP パッケージ導入プロジェクトの成功を左右する重要なプロセスという位置付けになる．

2.2.2　CRP 実施のメリット

先に述べたように，CRP ではプロトタイピングの手法により，ユーザがパッケー

ジ導入工程の早い時期に完成したシステムのイメージを確認でき，必要に応じてビジネスプロセスの代替案の調整をしながら導入を進めていくことができる．従来のソフトウェア開発ではシステムの完成イメージが段階的にしか確認できず，下流工程から上流工程への手戻りが発生し易い．CRP の実施は，上流工程でのコストはかかるが手戻りの発生が少ない，有効な手段と言える．

2.2.3 CRP の実施方法

CRP の実施方法は，簡単にいうと，パッケージをユーザのビジネスプロセス要件にあわせた形に設定し，実機で確認することである．

ここで CRP 実施の範囲となるユーザのビジネスプロセスは，AIM におけるオペレーション分析フェーズの前工程である要件定義フェーズで発生したギャップ（パッケージに合わない部分）の内，明らかにパッケージに機能がないものを除いた部分である．例えば「借入金システム」などという大きな単位で機能がない部分を除き，パッケージをうまく活用することにより代替案での実現が可能な部分を含む範囲（以下グレー・ギャップ）を対象とする．

グレー・ギャップに対する代替案の提示は，パッケージを利用できるかアド・オン開発が必要になるかを見極める重要なポイントである．CRP に参画するコンサルタントはグレー・ギャップに関する複数代替案を作成し CRP に臨むことが必要である．したがってコンサルタントにはパッケージに対する深い理解と柔軟な思考が要求される．

1） CRP 実施体制

CRP 実施時に限らず，ERP パッケージ導入のプロジェクト体制作りはユーザの BPR を成功させるための鍵を握る重要なポイントである．CRP 実施の体制は，ユーザの ERP パッケージ導入プロジェクト・メンバと我々パッケージ導入コンサルタントに加え，実業務に精通したエンド・ユーザから構成される．筆者の考える理想的な CRP 実施体制を図 3 に示す．

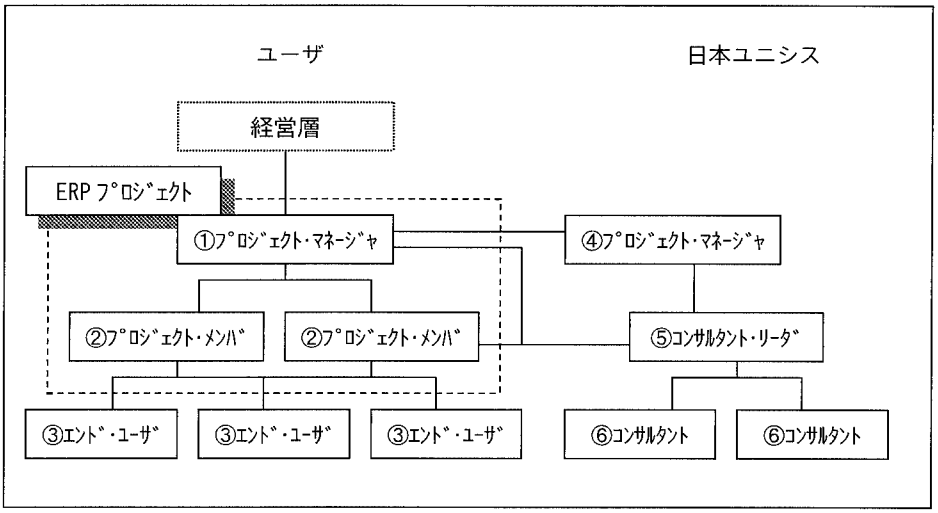


図 3 CRP 体制図

各メンバと役割は以下の通りである．

① ユーザ側プロジェクト・マネージャ

ERP パッケージ導入プロジェクトを統率して推進する役割を果たす．CRP 実施時においても，パッケージに対する設定要件，アド・オン要件等の承認権限を有する．一方，経営層に直結して密に連携する．また，ユニシス（以下，当社）側のプロジェクト・マネージャ，コンサルタント・リーダとも連携し，スムーズな CRP を実施できるようにする．

② プロジェクト・メンバ

ユーザのプロジェクト・マネージャをサポートする役割を果たす．一方，エンド・ユーザとの細かい調整も行う．また，当社側のコンサルタント・リーダとも連携しスムースな CRP を実施できるようにする．

③ エンド・ユーザ

業務に精通したエンド・ユーザであり，詳細なビジネスプロセスを説明する役割を果たす．

④ 当社側プロジェクト・マネージャ

当社側のプロジェクト全体を統括し，ユーザとの窓口となる役割を果たす．ユーザのプロジェクト・リーダと密に連携する．当社側ではコンサルタント・リーダとも連携しスムーズな CRP を実施できるよう努める．必要に応じて営業と連携し，最適なリソースの投入とユーザ支援を実現する．

⑤ コンサルタント・リーダ

ERP パッケージ導入コンサルタントチームのリーダ．実際に CRP を実施するコンサルタントの体制作り，技術支援等を行う．

⑥ コンサルタント

実際に CRP を実施するコンサルタント．

万全な態勢で CRP を実施するため，事前に体制を計画し各メンバ間の意識合わせを行うことが重要である．特に全体を調整し，決定権を持つユーザのプロジェクト・マネージャは必ず全ての CRP に参画することが必須である．

また，ユーザに体制の調整してもらうために当社側としても，プロジェクト・マネージャをはじめ関係各位に十分な理解を求める必要がある．

2） 効果的な CRP の進め方

オペレーション分析フェーズではスパイラル方式により CRP を数回にわたって実施する．図 4 は CRP の実施ステップを簡単に表現したものである．

CRP は通常 2〜3 回の実施となる．CRP を何回実施するかはプロジェクトの中で合意を得ていた方がよい．実施予定回数と 1 回での CRP 期間の決定をせずにいると，設定要件がなかなか決まらないまま時間とコストを費やし，次フェーズの作業スケジュールへのしわ寄せが発生する．また，各回の CRP にそれぞれの目的を持たせることも重要である．予め各回の目的とアウトプットを明確にすることにより次回の CRP，次フェーズへの移行がスムーズになる．

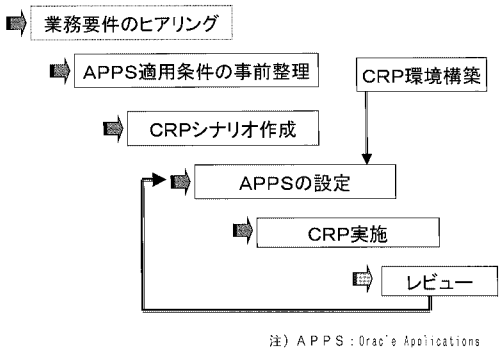


図 4　CRP 実施ステップ

3）　CRP のシナリオの作成

前工程の業務要件ヒアリングおよび Oracle　Applications 適用条件の事前整理のアウトプットを元に CRP のシナリオを作成する．シナリオはユーザのビジネスフロー（業務の流れ）に従い，各ビジネスプロセスを Oracle　Applications の機能にマッピングしたものである．シナリオをユーザに提出し承諾を得れば，それに従ったパッケージのセットアッププロセスに移行する．

4）　段階的 CRP の実施方法

ユーザからシナリオ承諾を得たら実機による CRP の実施フェーズとなる．ここでは図 4 に示す通り「Oracle　Applications の設定」→「CRP の実施」→「レビュー」を繰り返して行う．

筆者の個人的意見では，CRP は初回，中間，最終と 3 回にわたり行うことが望ましいと考える．3 回と想定した場合の目的を以下に示す．

第 1 回（初回）：ユーザのパッケージへの理解
第 2 回（中間）：イレギュラー処理も含むパッケージに対する設定要件の洗い出し
第 3 回（最終）：全ての設定要件の実機検証とユーザの合意のとりつけ

5）　テンプレートを利用した CRP の実施

CRP 実施において，FaSet シリーズ（FaSet Financial，FaSet Financial―EXT，FaSet　FA）をビジネス・ソリューション・モデルとして利用することも手法の一つである．FaSet シリーズは Oracle　Applications の短期導入モデルとして当社が開発したアプリケーションであり，一般的な会計キー・フレックス・フィールドのモデル（会計キー・フレックス・フィールドについては 4 章参照のこと），独自の入力画面と各種帳票が提供されている．4）で定義した第 1 回 CRP において，FaSet シリーズを Oracle　Applications 導入のテンプレートとして利用することにより，ユーザは導入イメージを実機にて確認することができる．大手企業は多種多様なビジネス要件があるため会計キー・フレックス・フィールドが一部固定されている FaSet シリーズをそのまま導入することは難しいが，ビジネス要件がマッチする企業であれば即時に導入が可能であり工程を短縮することができる．

3. Oracle Applications 適用条件の事前整理

Oracle　Applications 導入の際に決定すべきもののうち，各モジュール間で運用を統一すべきものがいくつか挙げられる．アプリケーション導入の際の Fit & Gap（ビジネスプロセスがパッケージに適合するか否かの判断）や CRP で討議すべき項目として忘れがちであり，方針を決定するまで予想外の時間を費やすケースがあるものに関し，カテゴリ別に討議していくべきである．あるユーザでは下記に挙げた番号管理に関して最終決定するまで半年以上の歳月を要した例もある．また，Oracle Applications は各モジュール単位で日付ひとつをとっても考え方が異なるという問題点もある．予め討議すべき内容の洗い出し時間を設ける必要がある．討議した結果は CRP のインプットとし，実機確認する必要がある．以下に討議すべき内容を挙げた．

3.1 番　号　管　理

一般的に，システムで管理する番号はマスタのコード番号とトランザクション・データの番号の 2 種類がある． Oracle Applications で管理するマスタのコード番号は，キー・フレックスのコードと各サブモジュールで提供されている仕入先マスタ等のコード番号の 2 種類がある．キー・フレックスのコードは基本的に手動による入力方式をとり，セグメント毎にユニークとなっている．仕入先マスタ等のコードに関しては自動発番/手動発番のパターンを選択できるものもある．

一方，トランザクション・データに関していえば Oracle　Applications で用意されている各種番号には下記のものがある．

① ユーザが入力するトランザクション・データの番号
② 文書番号
③ データの内部番号

Oracle　Applications で実現する会計システムは従来型の伝票会計システムではない．請求データ等のトランザクション・データは各サブ・モジュールで発生し，仕訳データが一般会計モジュールに転送されるという流れであり．「仕訳ありき」の伝票会計システムではない．そのため一般的な「伝票番号」という考えは持たない．

①の番号は一般会計モジュールの「仕訳名称」であり，買掛管理モジュールの「請求書名称」である．ただし自動発番の方式も採ることが可能である．

②は Oracle　Applications が提供する番号管理の考え方である．各種トランザクション・データ種別毎に発番される発番母体を持ち，トランザクション種別毎にユニークな番号が自動発番される仕組みである．また，当該番号は有効期間を設けることが可能であるため，会計年度等，一定期間単位でのシーケンスを保証する．

③はユーザの目に直接触れることはないが，システムで発番されるデータの内部番号であり正規化されたデータ・ベース表の内容を保証する番号である．

3.2 日　付　管　理

Oracle　Applications で管理されている日付には以下の種類がある．①，②はユーザが意識して入力する証憑の日付であり，③，④はユーザの目に直接触れることはないが Oracle Applications のデータ管理上重要な日付である．

① トランザクション・データの発生日付
② 仕訳データの計上日付

③　データ・レコードの作成日付

④　データ・レコードの更新日付

①，②は具体的には買掛管理モジュールの「請求書日付」と一般会計モジュールの「仕訳計上日」となる．請求書の日付と実際に仕訳を計上する日付という意味であるが，既存システムが伝票会計であった企業の場合，ニュアンスが異なる場合があるので十分に検討すべきである．また「仕訳計上日」とはいっても「実際に仕訳を一般会計に転送した日」はまた別の管理となっていることを知る必要がある．③，④に関してユーザの目に直接触れることはないと述べたが，処理日（システム日付）にあたる日付という意味で重要である．帳票をアド・オン開発する場合，その種類によっては処理日を元に出力したいというケースがありその場合に使用する日付である．文字通り③はデータ・レコードが作成された日付，④は更新日付であるが，④に関しては1）データの修正によってデータ・レコードが更新された場合，2）データ・レコードの状態変化（ex．未転記→転記）によって更新された場合，があるのでニーズによっては他の項目と合わせて参照するなどの考慮が必要である．

また，日付に関して以下に示すようなケースがあるので注意が必要である．各モジュール各機能の日付の扱いの違いを念頭に置いて方針を決定すべきである．

①　ある条件を元に計算により日付が自動的に算出される場合があり，上書き不可なものもある．

②　トランザクション・データがあるまとまった単位で管理され，日付もまとまった単位で特定の明細の日付になる場合がある．

3.3 マスタ管理

Oracle　Applications で提供されるマスタは大きく分けてキー・フレックスに組み入れられるマスタ類と各サブモジュールで用意されているマスタとの 2 種類に分けられる．マスタ管理について考慮すべき点を以下に述べる．

①　Oracle Applications のマスタに存在しない項目の管理

②　既存システムのマスタとの整合性の保持

③　既存システムと Oracle Applications とでマスタ構成が異なる場合の管理

④　テンポラリで発生する取引先の管理

⑤　導入時のマスタ一括投入

①は，Oracle　Applications で提供される各種マスタには固定入力項目以外にユーザが自由に入力できる付加フレックス・フィールド（4.1.2 項参照）が用意されており，それを用いることでほとんどの場合が対応可能である．

②が一番対応に悩むところであるが，1）Oracle　Applications 上で参照するマスタは Oracle Applications 内部で管理されていなければならない，2）トランザクション・データ投入前にマスタが存在しなければならない，3）マスタに直接更新ができない，という基本事項があるために，Oracle　Applications 側のマスタを正とし，更新情報を既存システムへ反映させる仕組み考えるということを基本方針とすべきである．

③の具体例は，Oracle　Applications では買掛管理モジュールで「仕入先」を管理し，売掛管理モジュールで「顧客」を管理している．企業により「取引先」として一元管理している場合も多く，Oracle　Applications 導入により 2 重入力という捕らえ

方をされてしまう場合がある．また，別管理であるがゆえに，仕入先でもあり，顧客でもある，取引先の債権債務の自動相殺機能はない．このようなケースの場合は「仕入先」と「顧客」の番号を統一し，運用により債権債務の相殺を行うことを提案する．

④は一見取引先といわれる，テンポラリ的な取引先の扱いである．パッケージによってはデータ投入の際に，その場で取引先の必要最低限な情報を登録できるものもあるが，Oracle　Applications の場合，②でも述べたが，必ず先にマスタを登録しておく必要がある．パッケージの考え方である旨を理解してもらい運用を検討する必要がある．

⑤に関しては，仕入先，銀行等 Oracle　Applications のほとんどのマスタは手動による入力となり，導入初期や期首の大量データ投入には苦しい部分である．最近ではキーボードやマウスの操作を覚えさせて登録しておき，実行により人の手を介さずにデータを入力できるようなツールも出回っており，これらツールの採用も検討材料にすべきである．

3.4　組織変更対応

Oracle　Applications に限らず，システムを構築する際に避けて通れないのが組織変更に関する問題である．組織変更に関するニーズには以下のものが挙げられる．

① 組織変更前/後のデータを参照したい．
　組織階層が変更になった場合は新旧階層で適切な参照をしたい．
② 組織変更時点で迅速なマスタ設定をしたい．
　組織変更は直前に発表になるケースがほとんどなので迅速に設定できること
③ 過渡期には変更前/後の組織ともデータ入力をしたい．
④ 組織コードが変わらずに名称のみが変更になった場合変更前データは変更前の名称で表示したい．
⑤ 人事組織マスタとの連動

①は会計キー・フレックス・フィールドの組織階層の考え方を用いて新旧組織を保持することが可能である．

②は 3.3 節のマスタ管理の項で述べたように，ツールを用いるなどの検討が必要となる．

③はフレックス・フィールド値（コード）に有効期間の設定が可能であるため，新旧コードが混在してもトランザクション・データ入力時に適切なコードを入力することが可能となっている．

④は同一コードで新旧名称を保持することが不可であるため，検討を要する事項である．

⑤に関しては，Oracle　Applications の統合会計モジュールから参照する人事モジュールのマスタ（従業員，アサインメント）があり，人事上の組織変更と会計上の組織変更のタイミングのずれが発生する場合の考慮が必要である．

3.5　セキュリティ管理

Oracle　Applications では様々な切り口でセキュリティ要件に対応できるようになっている．企業のセキュリティ要件をそれぞれの機能を用いて実現することが可能である．

① ユーザによるセキュリティ
② 職責によるセキュリティ
③ 会計キー・フレックス・フィールド・セグメント間のセキュリティ
（相互検証ルール）

①は Oracle Applications にサイン・オンするユーザ単位に使用する職責を分けることができるというものである．また，4 章で紹介するフォルダ機能もユーザ単位のセキュリティ管理が可能となっている．

②は職責別に使用できる処理の種類（メニュー，機能）を分けてセキュリティを管理するものである．また会計キー・フレックス・フィールドのアクセス権，会計帳簿のアクセス権もコントロールすることができる．また職責により，ある処理ボタンを非表示にして処理制限をかけることも可能である．

③は会計キー・フレックス・フィールド・セグメント間の検証ルールの設定により，例えば「勘定科目」と「部門」との検証を設定し，製造部門は製造関連の勘定科目のみ入力制限をかけるなどのセキュリティ要件が実現可能である．

3.6 データ移行

一般に新システムへの切替えの際には，データ移行は避けて通れない．本節ではデータ移行に関する留意点を挙げる．

① データ移行にかかわるスケジュールやコストの予定計上
② 新旧システムの管理体系マッピングの確定
③ 移行のタイミングとトランザクション・データの管理方針の決定

①はいうまでもなく移行は新システム切替えのため重要なタスクであり，期間と要員のスケジュールとコストを予定計上しておく必要がある．

②は Oracle Applications 導入にあたり管理体系をキー・フレックス・フィールドとして実現していくことになるが，既存システムにはない管理項目をどのように扱うかという問題である．1）代表コードに投入する，2）無意味コードを設け投入する，などのマッピングルールを予め確定しておく必要がある．

③は期首でシステムの切替えが発生する場合はまだしも，期中で切替えが発生する場合，債権/債務等のデータの扱いをどうするかという問題である．正確な移行を行わないと一般会計モジュールとサブモジュールとで 2 重計上などの不整合が発生する可能性があることを念頭に置くべきである．

4. オペレーション分析フェーズにおける CRP 実施上のポイント

Oracle Applications を導入するにあたり重要なポイントとなる項目がいくつか挙げられる．それらのポイントとは一つに Oracle Applications の優れた特長を活かし導入効果をもたらすもの，一つには Oracle Applications ではどうしてもカバーしきれない日本固有の商習慣に対応する方法である．この章では，CRP 実施にあたり，Oracle Applications を基盤とした企業に最大のメリットをもたらす導入が可能となるように，様々な事例の中から適切なソリューションを提供できる情報について記述する．

4.1 フレックス・フィールド

Oracle Applications の特長の一つにフレックス・フィールドの考え方がある．一

般的なパッケージはそのパッケージ内で管理するマスタの種類や桁数が固定的な場合がほとんどである．フレックス・フィールドはその常識を破った画期的な考え方である．つまりフレックス・フィールドはパッケージの制約にとらわれることなくユーザが自由に管理するマスタやその桁数を決定することができるものである．フレックス・フィールドには次に示す 2 種類がある．それぞれの特長について以下に述べる．

① キー・フレックス・フィールド
② 付加フレックス・フィールド

4.1.1 キー・フレックス・フィールド

キー・フレックス・フィールドを使用して柔軟にユーザに最適なビジネス・エンティティを提供することがあげられる．特に会計処理で使用する会計キー・フレックス・フィールドは企業としての会計データを参照/利用するための基本単位となる重要な役割を果たす．企業の ERP（Oracle　Applications）パッケージ導入が成功するか否かは，いかに有効な会計キー・フレックス・フィールド体系を築くかにかかってくるかということを理解して頂く必要がある．先に述べたように ERP パッケージを導入する際，会計分野から始めることが多い理由の一つとして，会計分野が企業管理体系の最終集約場所であることが考えられる．会計キー・フレックス・フィールド体系＝企業管理体系の最終集約場所の図式が成り立つのである（図 5）．

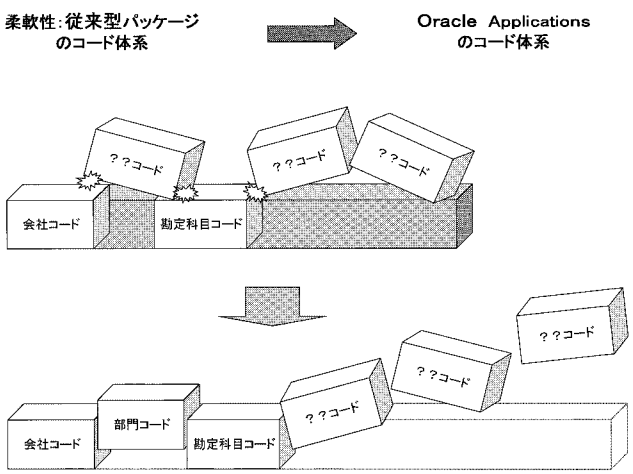


図 5　キー・フレックス・フィールド

表 1 はフレックス・フィールド事例である．上段に示すものが会計キー・フレックス・フィールドであり，下段が仕訳明細情報としての付加フレックス・フィールドである．貸借一致セグメントとは会計のバランスを合わせる単位であり，コストセンター・セグメントとはコスト負担部門を示している．会計キー・フレックス・フィールドは企業の業種やニーズに合わせて様々な体系を構成する．Oracle　Applications では，例えば「財務会計用」と「管理会計用」というように会計キー・フレックス・フィールド体系を分け，複数の会計帳簿を管理することも可能である（C 社の例）．

4.1.2 付加フレックス・フィールド

付加フレックス・フィールドはアプリケーションの画面に用意されている固定入力

表 1　フレックス・フィールド事例

会社	バージョン	業種	導入	Segment1	Segment2	Segment3	Segment4	Segment5	Segment6	Segment7	Segment8	Segment9	Segment10	Segment11〜
			モジュール	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	Attribute7	Attribute8	Attribute9	Attribute10	Attribute11〜
A 社	R10.6→R11	製造	GL	会社コード	*勘定科目*	予算費目	組織コード	機能コード	原価部門	工場コード	取引先	銀行	消費税区分	予備 1〜3
				相手科目	起票区分	本体勘定	起票者工番	起票者名称	仕訳起票日	相手先名称	伝票番号	自己科目		
B 社	R10.7	製造	GL	原価	*科目*	目的	プロジェクト	工程	製品群	事業	消費税	口座		
				短科目	短目的	短原価部門	組織部	アイテム	相手科目	計上日	得意先名称			
C 社	R10.7	サービス	GL,AP,AR	会社	*財務会計勘定科目*	課・センター	消費税	活動	ロケーション	エリア	グループ取引先会社	グループ取引先事業		
D 社	R10.7	商社	GL	会社コード	部所	*勘定科目*	補助科目	その他の科目	ダミー（連結区分）					
E 社	R10.7（NT）	システム開発	GL,AP,FA	*科目*	組織	商品	顧客	請求先	工番	内訳	会社	予備 1	予備 2	
F 社	R10.7	アパレル	GL	事業部門	計上部門	*勘定科目*	第二補助科目	消費税区分	プロジェクト	イベント	予備			
				依頼書番号	稟議番号	入力者 ID	仕分区分	相手先コード	相手先名称	取引番号	相手科目コード	行番号	店舗コード	
G 社	R11	電鉄	GL,AP,AR FaSeT FA	*勘定科目*	起票部署	予算部署	消費税区分	予算番号	金融機関	会社				
				取引先										
H 社	R11	建設	GL,AP,AR,PA PO,OE,INV	支店	*勘定科目*	消費税区分	事業部門	顧客区分	予備 2	予備 3				
				相手科目										
I 社	R11	流通	GL,AP,AR FaSet FA	会計単位	原価単位	*勘定科目*	科目区分	税区分	部門	口座	予備 3			
J 社	R11	公共	GL	勘定区分	*勘定科目*	消費税区分	契約担当契約職	仕訳担当出納職	仕訳発生部署					
				決算										

貸借一致（太字）	*科目（斜体）*	コストセンター（下線）

項目以外にユーザが自由に入力できるエリアのことである．パッケージの場合，どうしても入力項目が限られており，ユーザが付加情報として入力できるエリアが設けられていないことが多い．付加フレックス・フィールドはそのような制約に捕われずに管理したい項目を自由に設定することができ，項目に従った属性を入力できるエリアである．ただ，あくまでもユーザが任意に設定する項目であるため，導入にあたり以下のような制限事項について留意しなければならない．

① 標準画面の検索項目にはなっていない（一部例外がある）
② 標準帳票の出力項目にはなっていない
③ 個々の画面設定される項目であるためにモジュール間，処理間での連動性はない
（例えば，請求入力時に入力した付加情報は仕訳情報としては引継がない）

制限事項とはいうものの，次章に述べるオープン・インタフェースの存在や正規化されたデータ・ベース情報を最大限に利用した容易なアド・オン開発により Oracle Applications 導入のメリットを享受することも可能である．

4.2 複数会計単位

近年カンパニー制を導入する企業が増えている中で ERP パッケージも複数会計単位に対応できることが必須条件となっている．Oracle Applications には複数会計単位実現のためにいくつかの手段が提供されている．どの手段を用いて複数会計単位を実現していくか，それぞれの特徴を理解した上で決定する必要がある（図 6）．

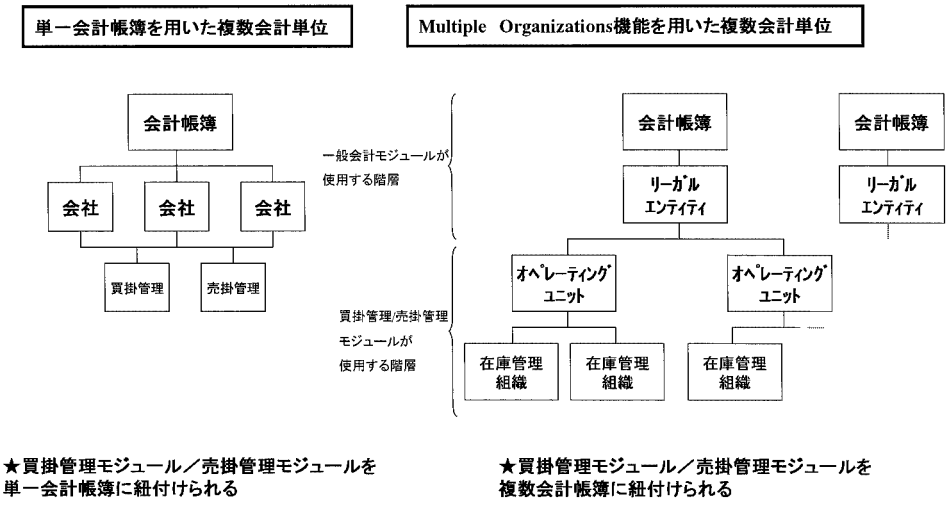


図 6　複数会計単位概念図

4.2.1 単一会計帳簿内での複数会計単位の実現

単一会計帳簿内で複数会計単位を実現するためには，会計キー・フレックス・セグメントの一つに会計単位となるセグメントを組み入れる．会計単位セグメントは「貸借一致セグメント」，すなわち残高をバランスさせる単位として設定する．これにより会計単位間の取引は会社間取引と認識され，仕訳転記時に各会社間でのバランスを保つために会社間取引仕訳が自動的に発生する．Oracle Applications 導入をきっか

けに部門別の採算制を実現するため，部門を貸借一致セグメントとした事例がある．ただしこの場合，部門間の取引毎に会社間取引仕訳が生成されるため，データが大量に発生することを念頭に置く必要がある．

単一会計帳簿の場合，買掛管理モジュールや売掛管理モジュール等のサブモジュールを導入する単位も単一である．一般会計だけで見ると複数の会計帳簿を導入することも可能であるが，サブモジュールと紐付けられるのは一会計帳簿のみという制限がある．勿論「貸借一致セグメント」の考えにより単一のサブモジュールにおいても会計キー・フレックス・セグメントという「仕訳」の目線での会計単位別の管理が可能であるが，本来会計単位別に管理すべき取引（例えば，請求書）は一括管理となってしまう．

また，会計単位でのセキュリティの設定は実質不可となる．自会計単位のみの入力/参照のセキュリティを設定することは会計単位間取引を入力できないことになるからである．尚，セキュリティに関しては 3 章を参照されたい．

4.2.2 複数組織（Multiple Organizations）機能を用いた複数会計単位の実現

Oracle Applications のリリース 10.7 より複数組織の機能が導入された．複数組織の機能の採用で，Oracle Applications の単一インストールにより，一般会計モジュール以外のサブモジュールを複数導入することが可能となった．この考えにより会計単位の会計帳簿をまったく別管理することが可能となった．勿論サブモジュールで管理する取引も会計単位別の管理となる．ただしこの場合は入金/支払業務も会計単位別となること，仕入先/顧客のマスタ管理も会計単位別になることに注意する必要がある．さらにサブモジュールで計上された会計単位間の取引も互いの会社間で別管理となることにも留意しなければならない．

4.2.3 会社間取引集中管理（Centra）

Oracle Applications のリリース 11 より一般会計モジュールに会社間取引集中管理機能が導入された．当該機能により会社間取引を発生させると相手の会計単位に自動的に仕訳を発生させることができる．4.2.1 項で述べた会社間取引と異なる点は異なる会計帳簿間での付替え仕訳を発生させることが可能である点である．これにより会社間での未達事項が削減でき，連結決算処理の期間短縮にも一役を担うこととなる．ただし当該機能は一般会計モジュールの機能であり，他のサブモジュールで発生した会社間取引には使用できないというデメリットもある．将来のバージョンにおいて，会社間取引が多く発生する買掛管理/売掛管理モジュールにも同様な機能が導入されることを望む．

4.3 オープン・インタフェース

Oracle Applications の最大の特長にオープン・インタフェースの存在がある．オープン・インタフェースとは Oracle Applications が提供するオープン・インタフェース・テーブルにデータを入力し，オープン・インタフェース取込プログラムにより，アプリケーション・テーブルにデータを投入するものであり，パッケージの基本的は枠組みを崩さずに外部システムとの連携が可能である．この項ではオープン・インタフェースの特長およびオープン・インタフェースを利用したアド・オン開発事例を紹介する．尚，以下オープン・インタフェース以外のテーブルを本稿では標準テーブル

と呼ぶことにする.

4.3.1　オープン・インタフェースの特長

オープン・インタフェースの存在はアプリケーション導入にあたり，以下のメリットを提供する.

① Oracle Applications 以外のシステムとの連動の容易性
② 多段階導入の容易性
③ 使用するモジュールの選択が可能
④ ユーザ固有の入力画面および追加プロセスをアド・オン開発可能

Oracle　Applications の各モジュールは一部例外を除いてオープン・インタフェースにより連動している（図 7 参照）. この特長により，例えば導入の第 1 フェーズは会計系モジュールを，第 2 フェーズは販売系モジュールを，というように無理なく導入計画を立てることが可能である. 第 1 フェーズでは既存システムを残し, オープン・インタフェースを通じデータを連動させることが可能となるからである. 導入時に使用するモジュールと要件を全て固めておく必要がある他の ERP パッケージとは思想が異なっている. また，企業固有の処理が多いフロントシステムは別に開発し，完成した部分から Oracle Applications に連動させることも可能である.

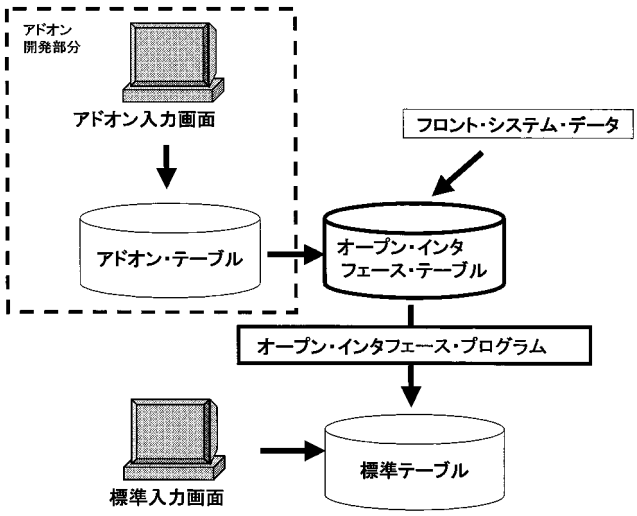


図 7　オープン・インタフェース概念図

図 8 は統合会計モジュールのオープン・インタフェースである. 一般会計モジュールの仕訳データのオープン・インタフェースをはじめとして各種インタフェースが用意されている.

4.3.2　オープン・インタフェースを利用したアド・オン開発

Oracle　Applications のオープン・インタフェース・テーブル以外の標準テーブルは直接更新することが許されていない. 標準テーブルは完全に正規化され複雑なリンク情報の元に成り立っており，アプリケーションが提供する画面やプログラムを通じて更新する以外に直接更新することはアプリケーションの動作を保証しないことに繋がる. また，バージョン・アップによりデータ・ベース構造やテーブル構造が変更に

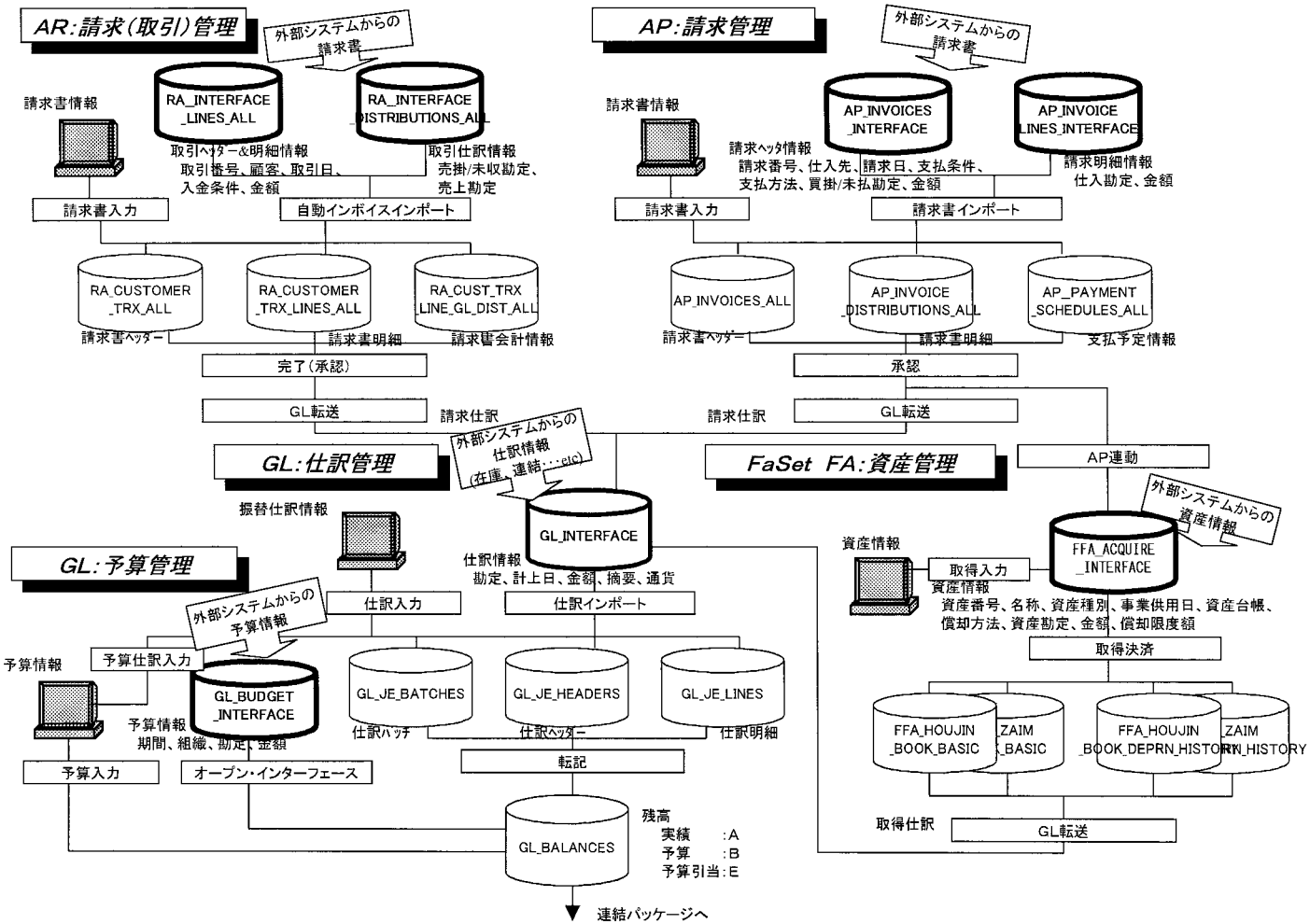


図 8　統合会計モジュールのオープン・インタフェース

なるケースがあり，直接更新するアド・オン開発をした場合，その都度開発した画面やプログラムを改修しなければならなくなる．一方オープン・インタフェース・テーブルは文字通り公開されているテーブルであり直接更新が許されている．

1)　オープン・インタフェースを利用した入力画面のアド・オン開発

アプリケーションで提供されている標準入力画面の使い勝手が悪い，アプリケーションで提供されていない複雑な承認ルートを使用したいなどのニーズがある場合，図7で示すように，入力画面それに付随する動作をアド・オン開発にて組み入れることができる．その場合，確定処理等何らかのトリガーにより，入力データをオープン・インタフェース・テーブル経由で標準テーブルに取込むのが一般的である．アド・オン開発のポイントとしてデータの整合性チェックをどの程度行うかという点がある．オープン・インタフェース・プログラムではデータ整合性チェックを行っており，不正なデータは取込みを拒否している．またエラーリストも出力する．アド・オン開発部分への同様な要件の組込プログラムの負荷がかかる上，オープン・インタフェース・プログラムとして提供されている部分を無駄にする．そこで以下のような開発方針を取る事例が多い．

①　キー・フレックス，その他マスタの整合性チェックはアド・オン部分に組込む

②　その他のチェックはオープン・インタフェース・プログラムにて吸収する

2)　オープン・インタフェースを利用した各種編集処理

先に述べたようにOracle　Applicationsのテーブルの内，ユーザに対し自由に更新が許されているのはオープン・インタフェース・テーブルだけである．そのような理由からオープン・インタフェース・テーブル上でデータを加工するアド・オン開発が可能となる．アド・オン事例として次のようなものが挙げられる．

①　摘要文言の編集

②　付加フレックス・フィールド編集

③　消費税内税分離処理

更新が可能とはいえ，編集の種類によっては標準テーブルの格納情報を得る必要があり難易度の高い場合もあるので注意が必要である．

4.4　日本企業へのOracle Applications導入の留意点

Oracle　Applicationsは日本で開発されたパッケージではないことから日本化対応が遅れている部分がある．新バージョンは国際標準であるベース機能と翻訳も含む各国固有の機能対応のパッチの組合わせによりリリースされる．また，各国固有の機能については組込まれない可能性もある．日本においても国際会計基準の導入が進められている一方，まだまだ日本固有の商習慣にこだわる部分が多いことは事実である．Oracle　Applicationsを導入するにあたり，日本の商習慣に対応しきれていない部分についてどのような対応方針をとるべきか考える必要がある．

4.4.1　各　種　帳　票

Oracle　Applicationsには各種標準帳票が用意されている．ところが英語を翻訳しているためのカラムのずれ，罫線がないなど美麗な帳票を要求する日本企業には受け入れられない部分がある．そのため標準帳票は使用せずアド・オンによる帳票の開発を望む企業が多い．結果としてアド・オン開発に予定外のコストがかかってしまうケ

ースがある．Oracle　Applications 導入目的に照らし，不要な帳票を削減するなどの措置を取ることが望まれる．帳票数の削減に関して以下に示すような方策がある．

① 分析/加工ツールによるデータの加工

② フォルダ機能の使用による複数帳票の統一

①に関しては先ごろ様々な分析/加工ツールが出回っている．一昔前ではエンド・ユーザのニーズを組み入れ情報部門が帳票を開発することが一般的であり，帳票の集計単位が異なるというだけで別プログラムとして開発をしていた．現在ではパーソナル・コンピュータとデータ・ウェアハウス・ツールの普及により，ニーズに合わせたデータ加工もエンド・ユーザ側で比較的簡単に行うことが可能になってきている．これらのツールの利用により帳票数の削減を図ることが望ましい．

②のフォルダ機能というのは Oracle　Applications の標準画面として組み入れられている機能のことで，マルチ・レコード形式でデータが表示されており以下の機能を持つ．1）データ項目の表示/非表示，2）データ項目の順序の入れ替え，3）検索条件の付加，4）データ項目プロンプトのユーザによる設定，5）データの Excel 等スプレッド・シートへのエクスポート，6）フォルダにセキュリティを付加することにより特定ユーザへのデータ公開範囲の制限，等である．これらの機能の組み合わせにより，画面による検索結果をそのままスプレッド・シートにエクスポートするなどエンド・ユーザのニーズに合わせたデータ抽出が可能となり，帳票削減に結びつく．

4.4.2 消費税対応

Oracle　Applications では表 2 で示すようにそれぞれのモジュールで消費税計算機能を持ち，消費税コードに登録された税率を元に計算を行うことができる．

表 2　統合会計モジュールの消費税機能

ﾓｼﾞｭｰﾙ	税金ｺｰﾄﾞ	ﾀｲﾌﾟ	率	内/外税区分	勘定科目	端数処理	計算ﾚﾍﾞﾙ
買掛管理	○	○	○	取引入力時に指定	税金ｺｰﾄﾞに指定	四捨五入	総額/明細
売掛管理	○	○	○	税金ｺｰﾄﾞに持つ	税金ｺｰﾄﾞに指定	四捨五入	総額/明細
一般会計	AP/ARの税金コードを使用			取引入力時に指定	税金ｺｰﾄﾞに指定	四捨五入/切上/切捨	総額/明細

買掛管理などのサブ・モジュールで入力された課税取引は，それぞれのモジュールにて本体金額と消費税金額とに分離してから一般会計モジュールへ仕訳として転送することが基本となる．内税で入力された取引も取引発生モジュールにおいて本体と消費税に分離しておかなければならない．サブ・モジュールからオープン・インタフェース経由で一般会計モジュールへ仕訳情報を転送する際に各モジュールで入力された税コードは引き継がれないため，一般会計側で分離することは実質不可能となっているからである．Oracle　Applications 以外の外部システムからオープン・インタフェース経由で仕訳を取り込む際も同様である．また，サブ・モジュールへオープン・インタフェース経由で課税取引を取り込む際も本体と消費税行を分離した形で取込みことを基本とする．

Oracle　Applications の消費税に関する基本的な考えは上記に述べたが，標準機能として不足している下記のような問題点がある．

① プロジェクト原価管理/請求管理モジュールは消費税計算機能が存在しない

② 買掛管理モジュールで相手先負担振込手数料の消費税計算が事実上不可能

これらの対応策として，Oracle　Applications の標準機能の範囲で消費税への振替取引を入力する考えもあるが，処理が非常に煩雑となってしまうため，アド・オン・プログラムの開発により税抜処理を行っている事例が多い．具体的には表 3 に示すように会計キー・フレックス・セグメントに消費税区分を設け，消費税区分のセグメント値を税率を含むコードとし，アド・オン・プログラムにより税抜き処理を行なう．処理タイミングと方式は，一般会計モジュール以外のサブ・モジュールまたは外部システムより仕訳データを転送する際,必ずオープン・インタフェースを経由するため,更新が許されているオープン・インタフェース上のデータを加工し図 9 で示すような税抜仕訳を生成する．

表 3　消費税区分設定例

ｾｸﾞﾒﾝﾄ値(ｺｰﾄﾞ)	摘要	税種別	課税/非課税/対象外	税込/税抜	率
11103	売上3%税込	売上	課税	税込	3%
11203	売上3%税抜	売上	課税	税抜	3%
11105	売上5%税込	売上	課税	税込	5%
11205	売上5%税抜	売上	課税	税抜	5%
12000	売上非課税	売上	非課税	－	－
19000	売上対象外	売上	対象外	－	－
21103	仕入3%税込	仕入	課税	税込	3%
21203	仕入3%税抜	仕入	課税	税抜	3%
21105	仕入5%税込	仕入	課税	税込	5%
21205	仕入5%税抜	仕入	課税	税抜	5%
22000	仕入非課税	仕入	非課税	－	－
29000	仕入対象外	仕入	対象外	－	－
99999	対象外	対象外	対象外	－	－

尚，図 9 は日次処理かつ明細単位の税抜処理の事例であるが，月次処理かつ総額での税抜処理であれば，アド・オンなしで税抜処理を行った事例もある．具体的には一般会計モジュールの機能として用意されている Financial　Statement　Generator（FSG）の機能を使用し税抜仕訳情報を作成した上で，オープン・インタフェース機能により一般会計に取り込むという事例である（図 10 参照）．

```
税込明細行　ｘｘ費（仕入 5%税込）105　/　買掛金　　105 ←元ﾃﾞｰﾀ
                        ↓
税込明細行　ｘｘ費（仕入 5%税込）105　/　買掛金　　105 ←元ﾃﾞｰﾀ
税抜仕訳　　仕払消費税 5%　　5　/　ｘｘ費（仕入 5%税込）　5 ←ｱﾄﾞ･ｵﾝ･ﾌﾟﾛｸﾞﾗﾑ
                                              により作成された行
```

図 9　アド・オン・プログラムによる税抜処理

★★　月次税情報　★★

名称	金額	計算式
01.70000.21105	-5	←01.70000.21105(ｘｘ費.(仕入5%税込)の当月発生金額)/105x5x-1
01.13500.99999	5	←01.70000.21105(ｘｘ費.(仕入5%税込)の当月発生金額)/105x5

↓

税抜仕訳　　仕払消費税 5%　　5　/　ｘｘ費（仕入 5%税込）　5

図 10　FSG 月次税情報

4.4.3 CAI商品

本節の冒頭で，Oracle Applicationsでは各国固有の機能については標準機能として組込まれない可能性もあることを述べた．そこで着目すべきはCAI商品の存在である．Oracle Applicationsを日本にて販売するには日本固有商習慣に対応できる機能が組込まれていることが必須条件である．ところがOracle Applicationsは全世界に販売展開されているものであるから日本固有の，さらに企業によっては使用しない場合もあるような商習慣までは対応しきれていない機能もある．CAI商品はそのようなOracle Applicationsを補完するツールである．CAI商品はOracle社が認定しているパートナーにより開発され，Oracle Applicationsの補完ツールとして認定を受けた商品である．個別のアド・オン開発に代えて，一般的に活用されている固定資産管理，連結決算処理，手形管理の代表的CAI商品を紹介する．

1） 固定資産管理

Oracle Applicationsの統合会計モジュールには固定資産管理を行うモジュールが存在する．ところが償却計算方法が定額法にしか対応されていないなど，日本の企業への導入を考えると，様々な問題を含んでいる．もちろん日本でも固定資産管理モジュールを導入している企業もある．ところが，例えば圧縮記帳等の多種の償却計算方法を採用している企業にとって，固定資産モジュールはパッケージとしてフィットせず，採用を見合さざるを得ない．極端な場合，Oracle Applications統合会計モジュールの導入を検討している企業で，固定資産モジュールの機能不足がクローズアップされた結果Oracle Applications自体の導入取消まで話が発展しそうになったケースもある．ERPパッケージという目線で捕らえた場合，固定資産モジュールの機能不足はパッケージ選定にまで影響を及ぼしてしまうのである．

このような背景のもと，当社では固定資産管理のアプリケーション「FaSet FA」を開発し，CAI商品として認定を受けている．開発には過去に固定資産管理のアプリケーションを開発に携わったスタッフとOracle Applications認定コンサルタントが加わり，優れたCAI商品として評価を受けている．

2） 連結決算処理

Oracle Applicationsの連結機能は一般会計モジュールの一機能としての位置付けとなっている．一般会計モジュールの連結機能は，1）オープン・インタフェース機能を使用しての外部からの連結データの取りこみ，2）会計キー・フレックス・セグメントのマッピング・ルールを使用しての組替処理，3）定型仕訳機能を使用しての連結消去仕訳の作成である．1）はOracle Applications以外のパッケージを使用している連結子会社からの連結データを容易に取り込むことを可能とし，2）は会計キー・フレックス・フィールドの柔軟性を活かし様々な管理体系を持つ連結子会社の連結データからの組替処理を実現するなどOracle Applicationsの優れた特長を連結機能として組み入れている．ところが連結開示項目に対応できていない等，連結処理そのものを売り物にしているパッケージに比べては見劣りがする．連結処理としては(株)ディーバ社が開発しCAI商品として認定を受けている「DIVA」の存在がある．「DIVA」は単独の連結処理パッ

ケージとしての実績もある CAI 商品である．

3)　手形管理

近年では手形を扱う企業が減ってきているが，まだまだ根強く残る日本固有の商習慣である．Oracle　Applications には手形管理のモジュールは存在せず，買掛管理，売掛管理モジュールにて支払方法，入金方法として扱われるレベルである．機能不足が固定資産管理ほど大きな影響を及ぼすわけではないが，手形を扱う企業に取っては手形管理の様々なニーズに対応できる機能を要求される．手形管理としては新日鉄情報通信システム(株)が開発し CAI 商品として認定を受けている「ENI　JAIS」の存在がある．手形を扱う企業では必ずといっていいほど挙げられる，印紙税節約のための手形分割等，細かなニーズに対応している商品である．

5.　アド・オン開発によるビジネス・プロセスの実現

アド・オン開発なしで Oracle　Applications を導入できればそれに越したことはない．2 章で述べたように CRP 等の手法を用いて極力アド・オン開発を減らすことが望ましい．とはいっても企業のニーズや Oracle　Applications で不足する機能に対応するためにはアド・オンが避けられないのが現状である．4 章でいくつか代表的なアド・オンの事例を述べたので，本章ではアド・オン開発にあたっての一般的な留意点を述べる．

5.1　アド・オン開発の見極め

2 章の図 2（CRP のインプット＆アウトプット）で示したように，CRP の実施によりユーザのビジネス・プロセスに対しる対応手段が決定される．ソリューションの種類を決定するにあたり，表 4 で示す指標を元に GAP の分類を行う．

表 4　GAP 評価基準

優先順位	判断基準
H	新業務プロセスにおいて必須。システムを稼働させる上で不可欠。
	外部提出の帳票。(法定帳票を含む)
	必須な個別システムへデータを渡す。(データ連携が必須)
	会社の制度上必要であり、別対応では著しく非効率または正確性の点で、管理上問題となる。
M	現業に著しい影響を与える。
	日常業務の遂行において、きわめて頻度が高く、手操作では非効率である。
L	当該機能の利用が特定業務(人)に偏っている。
	業務処理方法／手順を変えれば、対応可能。
	無くても対応可能であり、実現は容易である。
	使用頻度は低いが、管理上存在することが望ましい。
	使用頻度が高く、無い場合の業務負担の増加も予想されるが、当該機能の実現が、前提条件や他の重要機能の実現に、影響／障害を与える可能性がある。
	機能として使われていない、または、使用頻度が低い。
	現行システムに存在している。(必要性が不明確)
	あれば便利。
	方針として無くすことが可能。
	資料作成のためにのみ行っている。
	EUC (End User Computing：Excel等で対応) で、対応可能である。

アド・オン開発コストを削減するために，Oracle　Applications の標準機能の範囲で業務を行う方針とはなったが，機能不足を補うための各種チェックリストが必要となり，結果として開発コストが膨らんでしまったケースもある．またこのケースの場合，かえって業務が煩雑になり Oracle　Applications の導入効果が現れないことにもなりかねない．ユーザが業務に支障をきたさないための分岐点はどこであるのかを判断した上で，標準機能を用いるかアド・オンするかを見極める必要がある．

5.2　アド・オン開発のポイント

アド・オン開発にあたって，いくつかのポイントがあるが，重要なものとしては，1）異なる環境でも動作を保証する，2）バージョン・アップによる影響を受けない，3）標準モジュールと同様の動作を保証する，が挙げられる．そのためには下記の点を開発方針とするとよい．

① 名称，内部番号などのハード・コーディングは避ける．

② 表参照はビュー（仮想表）を用いて行う．

③ Oracle Applications の「コーディング・スタンダード」に従って開発を行う．

Oracle　Applications のマスタやトランザクション・データ項目は，特有の発番ルールに従った内部番号により管理されている．内部番号の発番は Oracle Applications の使用者が関知できない部分であり，例えば A マスタの 001 番の内部番号は B 環境によっては 1 番であり，C 環境では 2 番となる．すなわち本番環境と開発環境で同じマスタを投入しても同じ 001 番の内部番号は同じである保証がない．内部番号をハード・コーディングしてしまうと環境が変わることにより，全てを修正しなければならず，開発環境でのテスト結果を本番環境での動作保証ができないという事態が発生する．①で挙げたように名称や内部番号のハード・コーディングは避け，外部参照用の変数として定義する．

②は筆者が実際に体験したことに基づく提言である．Oracle　Applications の表項目はバージョン・アップにより変更になる可能性がある．実際にリリース 10.6 から 10.7 にバージョン・アップがあった際にキー・フレックス関連表の項目に変更があった．ある表にあった項目が別表に移ってしまい，関連するコーディングの手直しが発生した．当該項目を直接参照していたために発生した問題であり，表を直接参照するビュー（仮想表）を作成して，各プログラムはそのビューから項目を参照するようにさえしていれば 1 本のビュースクリプト（仮想表生成定義文）を改修することで済んだはずであった．

③で示すように Oracle　Applications の標準モジュールは「コーディング・スタンダード」の規約に基づいて開発されている．アド・オン開発も「コーディング・スタンダード」に基づき開発することを推奨しており，画面に関しては Developer 2000 Forms で作成された標準テンプレートが提供されている．この規約に従うことにより標準モジュールの画面とアド・オン開発した画面とで著しくユーザ・インタフェースが異なるという問題は発生しない．

5.3　アド・オン開発の留意点

ここでは筆者の実体験をもとにアド・オン開発の留意点を述べる．

5.3.1 正確な開発コスト算出の必要性

Oracle　Applications はデータ・ベースが公開されていることや親和性のある開発ツール（Developer 2000）が用意されているという理由から，開発が容易であるという錯覚に陥りやすい．従って開発工数を低めに見積もってしまいがちである．ところが，Oracle　Applications が完全に正規化されたリレーショナル・データ・ベース表の上に成り立っていることが反対にアド・オン開発の足枷になることが多い．というのは，表が正規化されているため，アド・オン帳票に表示したい項目を参照するにも表同士のリレーションを手繰っていかなければならいことが開発を難しくしているからである．このような現実を踏まえた開発スケジュールの作成と Oracle Applications の導入先企業への事前説明をした上でのコスト算出が必要である．

5.3.2 効率の考慮

アド・オン開発を行うにあたり，もっとも留意しなければならないことの一つとして効率の問題が上げられる．予め効率に関する指針を決定した上でプログラムを開発していくことが重要である．開発は完了したが，効率が悪く使用に耐えられないという結果になりかねない．リレーショナル・データ・ベースの特徴を正しく理解した上での開発方針の決定が必要である．

1)　ビュー利用の盲点

5.2 節で述べたようにバージョン・アップ時のリスク回避や開発コストの削減のためにビューを用いることは有用である．また，Oracle　Applications 上である程度必要な項目を網羅した標準ビューが提供されており，前項で述べた問題を回避するためも標準ビューはアド・オン開発に大変有用である．ところが一方，ビューに頼るがために陥りやすいのが効率の問題である．必要な情報を取り出すために何段階にもビューを参照することになる場合があり，結果として非常に効率が悪い処理になってしまうことがある．ビューが複雑になれば当然必要な情報を取り出すために時間がかかる．そこで，必要な情報を再編集したテーブルを新規で作成するというのも一つの手段である．

2)　リレーショナル・データ・ベースの特徴を理解した上での開発

リレーショナル・データ・ベースを用いて開発するには，その特徴を正しく理解しておく必要がある．開発プロジェクトのメンバが全員リレーショナル・データ・ベースの特徴を理解しているとは考えにくい．従って予め開発標準（方針）として規約を定義しておくべきである．ここでは，代表的な開発方針事例を挙げる．

①　表から複数の検索条件に従って特定のレコードを取り出すには，絞り込まれるレコードが多い条件から記述する．

②　マスタ項目を参照する場合，読み込む回数を減らすコーディングをする．

③　標準表を含め，必要に応じて索引を作成する．

①は考慮するしないによって著しく効率が異なることが一般的に知られている．

②③も同様に一般的にいえることだが，次に述べる Oracle　Applications の特徴に関連して重要なポイントである．

3） Oracle Applications の特徴を理解した上での開発

2）で述べたような一般的なリレーショナル・データベースの特徴に加え，Oracle Applications の特徴を正しく理解した上での開発が望まれる．2）の②に関連し，Oracle Applications のキー・フレックス・フィールドを代表的な例として，複数のマスタが一つの表に定義され特定のキーにより区別されているという構造になっている場合が多い．つまり一つのマスタから必要な情報を取り出そうとした場合に，非常に大量のレコードの中からの抽出になるからである．②のみならず①も考慮すべきであると考える．

4 章で述べたように，Oracle Applications の特長の一つに付加フレックス・フィールドが挙げられるが，付加フレックス・フィールドが何らかのキー項目として使用される場合がある．この際に，Oracle Applications 標準データ・ベース表の付加フレックス・フィールドの項目にあたる部分に索引を作成すると有効である．主キーに相当する部分はもちろん，表に外部キー等，予め索引が作成されている場合もあるがそれに新たな索引も加えることも効率改善の方策である．ただし，標準表に新たな索引を作成した場合は，バージョン・アップによって索引がない元の状態に戻ってしまうので，スクリプトの管理が必要である．

4） 開発ツールの選定

アド・オン開発をするにあたり予めツールの選定が必要になってくる．開発要件に従って適切なツールの選定をすることが望ましい．一例を挙げると，帳票作成ツールは選定したものによっては開発コストを大きく左右する恐れがあるので，事前に十分な評価をしておくべきである．また，入力支援ツール（画面のキー操作を予め登録しておいて実行させるタイプのもの）は通常の運用以外に移行時にも流用できる可能性があるため，プロジェクトにおける要件を踏まえて評価すべきである．

6. お わ り に

Oracle Applications 統合会計モジュールの導入において，特にオペレーション分析で実施する CRP が重要なプロセスであり，その実施ポイントについて述べてきた．本稿に記載した内容はこれまでの導入経験に基づくものであり，今後さらなる実績を積むことによりその結果が多岐に亘り利用性の高い部品となっていくことを目指したい．目標は，導入に携わる人の経験や実績に左右されることなく，誰がどのユーザに対しても適切なソリューションの提示ができることであり，結果として Oracle Applications の導入企業への効果をもたらすことである．さらに，これらの経験が FaSet シリーズに反映され，Oracle Applications の柔軟性を活かした商品として発展していくこに貢献したい．

参考文献 ［1］「ERP 入門」, 同期 ERP 研究所編, 工業調査会.
［2］「ERP 成功への方定式 Oracle Applications」, リックレレコム.
［3］「ERP 経営革命」, ERP 研究推進フォーラム, ダイアモンド社.
［4］「ERP 導入事例に学ぶ導入の進め方」, ERP 研究推進フォーラム, IPA/INES.

［5］「Applications Implementation Method（AIM）Advantage TM リリース 2.0」オラクル社

執筆者紹介　稲　垣　理　佳　（Rika Inagaki）

1986 年東京女子大学文理学部英米文学科卒業．同年日本ユニシス（株）入社．Mapper 会計アプリケーション UNIMACS Lite 開発，1997 年より Oracle Applications 統合会計モジュール適用に従事．Oracle Applications Certified Consultant/Gold．