C Solution™向け開発方法論とプロジェクト管理

Application Development and Project Management Methodologies for *C Solution*™

山城　明宏　YAMASHIRO Akihiro
小尾　俊之　OBI Toshiyuki

アプリケーション開発とプロジェクト管理を支援する *C Solution*™向けの方法論(CSM/CSP)を開発した。この方法論は，*C Solution*™のアプリケーションフレームワークとソリューションプラットフォームを活用した繰返し／インクリメンタル型の開発プロセスをサポートする。S/V(Stable / Variant)分離，ラピッド開発，タイムボックスベースの管理技術がCSM/CSPの核となる。

We have developed a *C Solution*™ application development methodology (CSM) and a *C Solution*™ project management methodology (CSP). These methodologies facilitate the incremental and iterative development process, namely, rapid application development (RAD), by optimally utilizing the application framework and *C Solution*™ platform.

The core mechanism of CSM/CSP consists of three techniques: stable/variant separation, RAD, and time box-based management.

1 まえがき

情報システムは開発対象が大規模である。また，人間系を対象にした業務支援を機械化するので仕様を一様に決めがたい。さらに，開発されたシステムは寿命が長く，長期間にわたって継続的に保守が続く。

このような特徴をもつソフトウェアを開発するには，いくつかの問題を解決する必要がある。例えば，最初の特徴に対しては，開発技術の人依存と類似ソフトウェアの重複開発をいかに減らすかを考える必要がある。2番目の特徴に対しては，あいまいなユーザー要求をいかに把握し，開発途中のソフトウェアに要求を取り込むかを検討しなければならない。3番目の特徴に対しては，情報の一貫性維持や変更の局所化をいかに実現するかに取り組まなければならない。これらの問題を解決するために，現状では次のような方法が採用されている。

(1) データやトランザクションの正規化による冗長性の排除
(2) ユーザーインタビューやユーザー／メーカーの共同開発の実施
(3) 標準パッケージの導入とカスタマイズ／アプリケーションジェネレータの開発

ところが近年の情報システムは，プラットフォームがメインフレームからクライアント／サーバへ，さらにWebtopコンピューティングへ，というように情報システムのインフラは発展途上である。また，組織変更や合併吸収に伴う業務やその運用形態の見直し，さらに新技術の普及に触発された新たなビジネス分野の創出があり，情報システムが対象とする分野自体が広がり，流動的である。そのため，情報システムは絶え間のない改善と改変を余儀なくされている。

このような状況において，これからの情報システムの構築に対して従来のソフトウェア開発方法を採用するだけでは問題に対処しきることができない。前述の(1)への対応においては，正規化を行うこと自体に工数がかかる上に，前提にした画面や帳票のデータ項目あるいは業務自体が進歩するプラットフォームの活用に伴って変更される可能性がある。(2)に対してはメーカーが顧客にインタビューしたり，打ち合わせるだけでは，顧客の真意がメーカーには伝わらない可能性が高い。さらに(3)に対しては，標準パッケージを導入したり，アプリケーションジェネレータを活用すると，枠組みを越えたカスタマイズができなくなってしまう。

2 *C Solution*™の開発／管理方法論

このような情報システムの開発の問題点を考慮した上で，その解決の施策として，次のコンセプトを設定した。

①開発完了後の保守／拡張を見通しつつ(変更容易性)，②早期に顧客の機能要求／品質要求を確認し(ユーザー要求とシステムアーキテクチャの早期検証)，③開発サイクルをできるだけ加速する(開発期間の短縮およびプロセスと成果物の管理)というコンセプトである。このコンセプトを実現する上で次の三つの技術を開発方法論(CSM：*C Solution*™ application development Methodology)と管理方法論(CSP：*C Solution*™ Project management methodology)の核とした。

(1) S/V分離

(2) ラピッド開発(RAD)

(3) タイムボックスベースの管理技術

2.1 S/V分離

変更容易性は，保守性，拡張性，再構築，可搬性といった要素から構成される高品質ソフトウェア要件の一つである。すでに述べたように，寿命が長く，さらに近年はその応用分野と利用技術が広がりつつある情報システムの構築において，この変更容易性は非常に重要な品質要件といえる。変更容易性の観点から，対象の特徴に応じた開発を行なうために，アーキテクチャ決定をガイドする手段としてS/V分離の観点を取り入れた。

2.1.1 S/V分離の考えかた S/V分離とは，システム開発を進める上で，安定な部分(Stable)と変更が頻繁な部分(Variant)に分離する考えかたである。これはもともとハードウェア開発で培われた考えかたであり，変更部分を局所化することで開発や変更の容易性を向上できるため，ソフトウェア開発でも比較的古くから採用されている。

2.1.2 S/V分離のための従来／現在の技術 従来の情報システムの開発言語(COBOL)や実行環境(専用OS)では，安定部分と変更部分を切り分ける方法はサブルーチン化や共通パラメータのファイル化という方法に限定されていた。もし，気の効いたデータのパラメータ化や処理のテーブル化／パターン化によるS/V分離を実現しようとすると，アプリケーションを生成するジェネレータを，アプリケーションとは別に用意しなければならなかった。また，Warnier-Orr法やJackson法という古典的設計論はもともと個々のプログラム設計のための技術であり，システム全体の安定したアーキテクチャ標準の開発には不向きである。

これに対して，***C Solution***$_{TM}$の開発方法論は，Java(注1)，インターネット技術とともに，***C Solution***$_{TM}$が提供するアプリケーション フレームワークを利用する。これによって，継承や多様性，テンプレートを応用した“抽象化”と，インタフェースやパッケージ，名前空間による“カプセル化”を活用したS/V分離をプログラミング言語レベルで表現できるようになった。これらのメカニズムは，さらにシステムレベルのS/V分離にも応用可能である。

2.1.3 システムレベルのS/V分離の視点 システム全体の安定部分と変更部分を切り分けるための具体的な観点として，大きく三つの視点を設けた。

(1) アーキテクチャパターンによる方法 アーキテクチャパターンに関しては文献(1)に10種類のパターンの使い分けが述べられている。

(2) ファサードによる方法 複数サブシステムから構成されるサブシステム間にファサード層というインタフェースを設ける方式である。このインタフェースを通してサブシステム間の通信を行うことで，個々のサブシステムの変更に伴う他のサブシステムへの変更波及の局所化を図る。この構造は“アプリケーションファサード”として文献(2)に述べられているが，他にもさまざまな実装方法があり得る。

(3) レイヤの安定度に着目する方法 プレゼンテーション層，業務処理層，データベース層から構成される3層分散システムに対して，どの層がもっとも変更／拡張に対して安定しているかに着目する方法である。これらは過去の開発で得られた経験則である。このアプローチの概要を図1に示す。

(a) プレゼンテーション層に現れるユーザー対話のメニュー構造が他の層に比べて安定している場合，つまり個々のメニューとその構造が業務の構造を反映している場合は，MVC(Model View Controller)パターンやPAC(Presentation Abstraction Control)パターンの構造を利用すると変更が容易になる。

(b) 業務処理層が比較的安定している場合は，そこに現実世界(例えば業務を運用する組織)の構造をマッピングすることによってシステムの構造は変更に対して柔軟となる。

(c) データベース層が管理するデータ構造が比較的安定している場合，つまり管理される情報の種類とスキーマがもっとも重要で，アプリケーションの機能はそこに対する登録／検索／削除といった場合は，データオリエンテッドアプローチ(DOA)による正規化を進めることで，データだけでなく，このデータを基に構築するシステム全体が重複のない変更容易な構造となる。

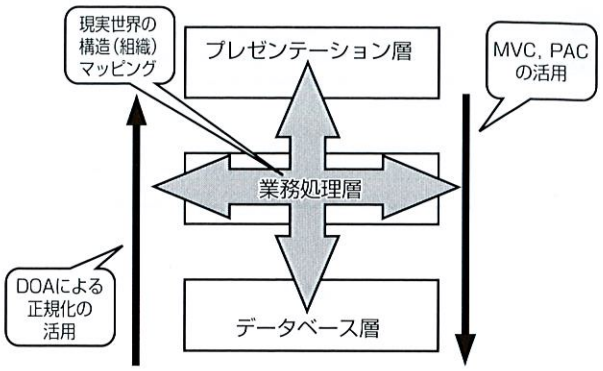

図1．レイヤの安定度に着目したシステムのS/V分離 3層分散システムの各レイヤの安定度に着目して，設計法を選択する。
Stable/variant separation of system from viewpoint of layers

2.2 ラピッド開発(RAD)

2.2.1 大規模システムの繰返し／インクリメンタル開発 人間系を対象にした業務支援を自動化する情報シス

(注1) Javaは，米国SunMicrosystems社の商標。

テムの構築には，ユーザー要求とシステムアーキテクチャの早期検証と見直しを行うことが重要である。ラピッド開発（RAD：Rapid Application Development）は，ソフトウェアの構築を，着手から出荷までの間進化的に設計，実装，検証を繰り返すインクリメンタル型の開発プロセスを支援する。この際，各担当者はそれぞれ，独立した機能の開発を請負い，その途中途中でシステム統合と試験／検証を繰り返す。これによって，システム全体ができあがる前に，システム全体の動作を確認することができる。

ラピッド開発は，ユーザー要件を満たす高品質なソフトウェアを開発する。実は，繰り返し型の開発プロセスが，分析から出荷までをシーケンシャルに行う古典的なウォータフォール型開発プロセスよりも優れていると論じられてからすでに10年以上過ぎている。しかし，大規模システム構築において，その成功例は少ない。理由は二つある。一つは共通部品化の観点なしに個々のメンバーが開発分担してそれぞれが並行して開発を進めると，開発する機能の中で重複する部分がどんどん増えてしまうからである。二つ目は，インクリメンタルにシステム統合を繰り返して，試験と検証を行うためには，完全に実装が完了していないうちにソフトウェアが実行可能でなければならないからである。

C Solution™は，アプリケーションフレームワーク（APF：関連論文p45参照）を，モジュールの共通化を図る観点として活用することで，前者の問題の解決を容易にしている。また，***C Solution***™プラットフォーム（SPF：関連論文p38参照）とスタブ生成機能を活用することで，後者の問題の解決を容易にしている。つまり，アプリケーションフレームワークと***C Solution***™プラットフォームの存在によって大規模システムの開発を対象にしたラピッド開発が初めて成り立つのである。***C Solution***™開発方法論は，これら二本柱の導入と活用を前提にした繰返し／インクリメンタル開発プロセスを定義している。

2.2.2 RADプロセスのドキュメンテーションと支援ツール　開発プロセスは複数のセグメントから構成される。個々のセグメント間をつなぐ中間成果物がドキュメントであり，また各セグメントでは複数のツール群が利用される。構築で開発者がインクリメンタル開発を進めていくにあたって，システム全体構造，つまりアーキテクチャのビジョンが共有されていることが重要である。そのために特に上流セグメントではドキュメンテーションを重視して，文書形式，フォームシート形式，およびダイアグラム形式を組み合わせて使うことにした。下流のセグメントではできるだけCASEツール（リバースツール）を活用して，ドキュメンテーションの簡易化を図っている。他方，開発ツールに関しては，上流のRational Rose(注2)，下流のNetscape Application Builder，また構成管理にはClearCase(注3)を中心にしたツールセットを連携活用するためのしくみを構築している。

2.3 タイムボックスベースの管理技術

タイムボックスベースの管理技術は，ラピッド開発を成功させるためのもう一つの重要な技術である。タイムボックスとは，固定期間内でのスケジュールと予算，資源を管理するプロジェクト管理のテクニックである。個々のタイムボックスは，ラピッド開発の繰返し開発の1サイクルに対して定義される。この開発方法に合せて，コンポーネントをベースとした見積り方法，タイムボックス内で完結させるための進捗（ちょく）管理方法，インクリメンタル開発に対応した構成管理方法などを，管理作業のガイドブック，ノウハウ集，ツールセットなどにまとめている。次項に管理技術に一つである見積り技法について概要を述べる。

2.3.1 見積り技法　見積りは，アプリケーションの実現手段に依存しないファンクションポイント（FP）法を用いて機能量を測定し，その機能量をベースとしてそれぞれの目的に沿った見積り値（開発工数，ソフトウェア価格など）の算出を行う。実施のタイミングは，引合い段階，要求抽出後，の2段階を中心に，その他，仕様変更があった時点，開発完了時点である。特に要求抽出後の2段階目では，先に行われたIT（Information Technology）コンサルテーション，および要求抽出でまとめられた業務機能単位ごとの成果物のうち表1に示す情報を利用し，より正確な機能量の測定を目ざしている。

これらの成果物は，図2に示すように業務機能で独立した単位ごとに作成される。この業務機能はそれぞれ複数のファンクション単位（IT機能）に分割される。このファンクションを機能量測定の単位とし，これを蓄積することによ

表1．業務機能を示すドキュメントの種類
Various documents describing application functions

ドキュメント名	分類	記述内容
業務機能要件	必須	ユーザーが要求する業務機能要件（What）を記述したドキュメント
業務機能階層図	必須	ユーザーが要求する業務機能の階層を示すドキュメント
業務処理フロー	必須	ユーザーの業務処理単位（ex.顧客新規登録）に入力データ，処理概要，出力データの関係を示したドキュメント
ER図	必須	論理データ間の関係を記述したドキュメント
ファイルマトリックス	OP	業務単位に処理と利用ファイルを表形式で示したドキュメント
画面一覧	必須	開発する画面の一覧を示すドキュメント
画面レイアウト図	必須	ユーザーインタフェースとなる画面のレイアウト図
画面遷移図	OP	画面の移り変わりの流れを示したドキュメント
帳票一覧	必須	作成される帳票の一覧を示すドキュメント
帳票レイアウト図	必須	帳票出力処理で作成される帳票のレイアウト図

ER：Entity-Relationship
OP：状況によって見積り時に考慮するドキュメント

(注2)，(注3) Rational Rose，ClearCaseは，米国Rational Software社の登録商標。

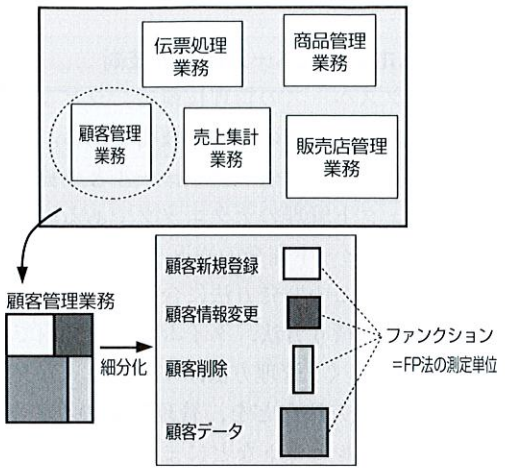

図2．業務機能単位の概念（流通分野の例）　業務機能単位は業務分野別にユーザー業務を意味的にまとめた単位であり，構成要素はFP法のファンクションである。
Concept of application unit

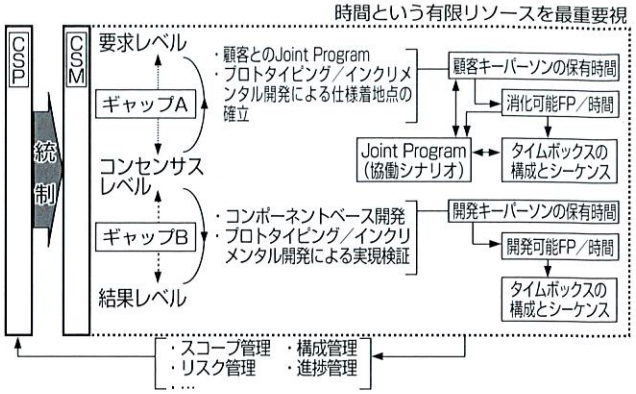

図3．システム開発における仕様内容のギャップ　CSM/CSPでは，システム開発に存在する仕様内容のギャップを埋めることが目的である。
Gaps between system specifications

り，測定作業の効率向上を果たす。

こうして出された機能量(FP)を利用して，開発工数や期間，ソフトウェア価格などの見積りを行う。なお，これについては，実績値の蓄積が必要不可欠であり，実績値の蓄積と合わせて，より適した見積式とする。

3 開発方法論(CSM)と管理方法論(CSP)

プロジェクト管理は，プロジェクトにおける品質，コスト，納期の最終目標を確保することが目的であり，そのため，異常や遅れを早期に発見し，それに対して適切でタイムリーな対応をとることが管理作業のベースとなる。また，プロジェクト管理は，一般に開発プロセスに大きく関係している。これは，***C Solution***TMを利用するプロジェクトにおいても変わることはない。したがって，CSPはCSMに沿って開発が進められることを，管理的な視点(品質，コスト，納期)から統制するものであると言える。

3.1 現状のシステム開発におけるギャップ

現在，システム開発のプロジェクトで起きているさまざまな問題の原因として，図3に示すようなギャップが存在することが挙げられる。つまり，ギャップAは，開発を開始する際の顧客側と開発側で合意した仕様内容(コンセンサスレベル)と，顧客が実際に考えていた仕様内容(要求レベル)との差であり，ギャップBは，このコンセンサスレベルと，実際に開発が終了し実現された内容(結果レベル)との差である。これらのギャップを埋めることがCSM，CSPに課せられた目的である。

3.2 CSMによる対応とCSPでの統制

これらのギャップを埋める手段として，CSMでは，プロトタイピングやインクリメンタル開発などを導入し対応を考えているのは既述のとおりである。また，特に重要なのが，顧客キーパーソン，開発キーパーソンの参画である。つまり，顧客側の意思決定を行えるキーパーソンの参画は必須(す)条件となる。しかし，キーパーソンは一般に，参画に費やせる時間の制約が厳しい。そこで，限られた時間を最重要視した開発の進めかたが必要となる。このようなニーズに対して，タイムボックス手法が有効である。

4 あとがき

C SolutionTM向け開発／管理方法論(CSM/CSP)について，S/V分離，ラピッド開発(RAD)，タイムボックス管理の三つの観点から述べた。この方法論は，***C Solution***TMが提供するアプリケーションフレームワークと***C Solution***TMプラットフォームの活用を前提として，大規模システムの開発をサポートする。

文　献

(1) F. Buschmann, et al. England, Pattern-Oriented Software Architecture, John Wiley & Sons, 1996, 457p.
(2) M. Fowler. Massachusetts, Analysis Patterns : Reusable Object Models, Addison Wesley, 1997, 357p.


山城　明宏　YAMASHIRO Akihiro
情報通信・制御システム事業本部 SI技術開発センター主務。オブジェクト指向技術推進と***C Solution***TM開発に従事。情報処理学会，IEEE, ACM会員。
System Integration Technology Center


小尾　俊之　OBI Toshiyuki
情報通信・制御システム事業本部 SI技術開発センター主務。プロジェクトマネージメント技術の開発・普及に従事。情報処理学会，PMI会員。
System Integration Technology Center