imagio Neo C240/C320オブジェクト指向ソフトウェア開発

Obeject-oriented Software Development for Imagio Neo C230/C320

久我　雅人*　　米永　晃太郎*
Masato KUGA　　Kohtaroh YONENAGA

要　　旨

近年，ますます大規模複雑化する複写機などの組み込みソフトウェアにおいて，開発生産性の大幅な向上と品質の更なる安定の両立が必須課題となってきている．そのような中で，従来からのソースコード依存型開発から脱却し，より高いレベルでの再利用を志向した，オブジェクト指向開発手法とインクリメンタル開発プロセスを，エンジン制御ソフトウェア分野に適用した．

この開発手法を適用したカラーMFP：imagio Neo C230/320においては，要求仕様を分析する段階から全面的にUML記法に基づいて行い，プロッタエンジン及びスキャナエンジン制御ソフトのほぼ全体をオブジェクト指向手法及びインクリメンタル開発プロセスで行い，商品化することに成功した．あわせて，モデル・ベースによる高度な再利用を展開するための基礎固めができた．

ABSTRACT

In order to break away from the reuse process based on program source codes and to improve quality and productivity of software development, we adopt ″Object Oriented Design Method″ and ″Incremental Development Process″. With these method and process, we provided new Color MFP product “Imagio Neo C240/C320”. In this development process, we designed software using ″UML″ from understanding requirements phase to build almost whole engine control software through Object Oriented Technology, and laid the foundation for future model-based framework reuse.

* 画像システム事業本部　C&F第二事業部
C&F(Copier&Fax) Business Division 2, Imaging System Business Group

1. 目的

複写機やプリンタの世界では，デジタル化を契機として高機能化／ネットワーク化が進み，必然的にその制御ソフトウェアはますます大規模・複雑化してきている．そのいっぽうで商品開発サイクルはますます短くなる傾向にあり，ソフトウェアの品質確保がより重要な課題となりつつある．

開発サイクル短縮のため，ソフトウェア開発の生産性を上げるという観点からは，再利用を積極的に進めることが必須となるが，ソフトウェア再利用がもたらす副作用を考えると，事情はそう簡単ではない．

そこで本稿では，ソフトウェア品質を向上させつつ，効率の良い再利用をはかっていく手段として，オブジェクト指向開発手法とインクリメンタル開発管理プロセスを，ともに商品開発に適用した例を紹介する．

2. 従来手法の限界と新手法適用の背景

従来行われてきたソースコード流用型のソフトウェア再利用は，現流機に搭載されているソースコードを切り貼りすることによって進めるため，再利用が簡単に実現でき，生産性も上がる，という長所がある反面，切り貼りした箇所に数多く不具合が発生する，という大きな問題がある．

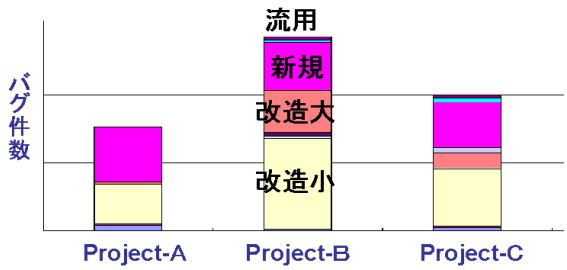

Fig.1　改造箇所とソフト障害の関係

Fig.1は，過去にリリースされたソフトウェアの障害の統計をとり，その障害が発生した箇所の設計内容／ソフトウェアのメトリクス値との相関でまとめたものの一例である．

Fig.1を見ればわかるように，直感的なイメージに反して，実は新規に作成した部分や大改造を施した部分ではなく，過去の資産の一部を改造した箇所に不具合が多く発生していることがわかる．つまり，生産性を上げるためにソフトウェアの再利用を繰り返すほど，むしろ品質は低下する傾向がある，ということである．

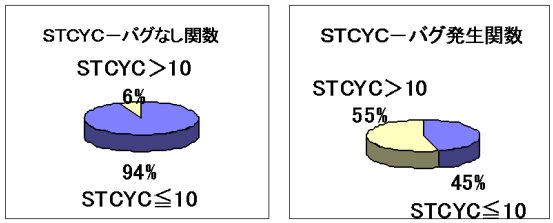

Fig.2　ソフトウェア複雑度とソフト障害の関係

また，Fig.2は「ソフトウェア複雑度」と障害発生の相関関係についての統計を示したものである．

Fig.2からは，ソフトウェアの複雑度が高い箇所ほど，障害が数多く発生する傾向が読み取れる．

すなわち，ソフトウェアの品質向上を達成し，効率的な再利用をはかっていくためには，

(1)　ソースコード・ベースの再利用から脱却し，再利用時の副作用（障害）を低減すること

(2)　ソフトウェア複雑度を低減したソフトウェアの構造とすること

(3)　それらを短納期で達成すること

という3つの問題を解決することが要点になる．

今回，これらの問題点を解決すべく，上流工程での品質作り込みが容易に実現でき，かつ，モデリングベースによる高度な再利用ができる，という特徴を持った「オブジェクト指向開発手法」の適用を試みた．さらに，これを短納期で実現するため，リスクを極小化しつつ容易に管理できる「インクリメンタル開発プロセス管理手法」と組み合わせて進めることにした．

3. 対象となったシステム

Imagio C240／C320は，ネットワークプリンタ/スキャナ／ファクス／コピー機能を備えたデジタルカラー複合機である．カラー機でありながらモノクロ機なみの省スペース設計や操作性を実現し，カラー原稿を直接電子メールで送信したりカラーの紙文書の電子化が手軽に行えたりネットワークカラープリンタ機能も有している．今回対象としたのは，このデジタルカラー複合機のプリント・エンジンおよびスキャナを制御する部分の制御ソフトウェアである．

4. インクリメンタル計画

インクリメンタル開発は，まず開発工程全体を，1～3ヶ月を目安とする期間（＝インクリメンタル）に区切り，あらかじめ各インクリメンタルでの開発目標を明確にしておく．そして，各インクリメンタルが終了したときに振り返りを行い，実績と計画の際を分析することにより，次回以降のイテレーション計画にフィードバックしつつ進める．短いサイクルを何度も回すやり方で，リスクを極小化し，管理しやすくできるところがポイントである．

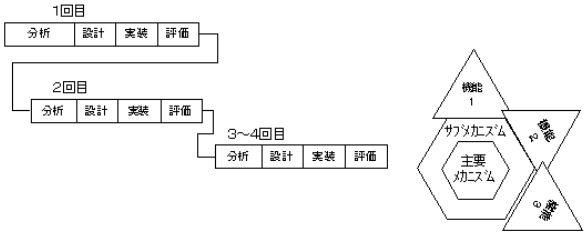

Fig.3　インクリメンタル開発

今回のエンジン制御ソフト開発では，つぎのような計画を組んで進めた．

●インクリメンタル0　～要求分析
　目標：対象領域の全体像を理解し，特定する
●インクリメンタル1　～準備
　目標：ワークフローを一通り経験する
　　：Rose2000やC++というツール／言語に慣れる
●インクリメンタル2　～骨格固め
　目標：主要ユースケースのシミュレーション完了
　　：モデルの骨格を固め，モデリング指針を決定する
　　：例外処理，異常処理も分析する
●インクリメンタル3　～骨格の肉付け
　目標：主要ユースケースの実機実装（動作確認）
　　：別の主要ユースケースのシミュレーション完了
　　：例外処理・異常処理の設計／シミュレーション完了
●インクリメンタル4　～主要機能の実装
　目標：主要ユースケースすべての実機実装
　　：例外処理，異常処理の実機実装
●インクリメンタル5　～全機能実装
　目標：全ユースケースの実機実装
　　：評価　→　障害対応
●インクリメンタル6　～仕様変更対応
　目標：評価　→　障害対応
　　：仕様変更に対する対応

これらのインクリメンタルのなかでも重要なのは，要求分析と骨格固めの期間である．

まず要求分析の基本はユースケース記述であるが，この時点で徹底的な抽象化を行うことが最大のカギとなる．ユースケースの記述に慣れないうちはアクティビティ図との併用で進めると，抽象化の視点を見つけやすい．

また，骨格固めで重要なのは，全体のアーキテクチャに対してインパクトの大きい機能を早期に認識し，骨組みの中にこれを解決する仕組みを盛り込んでおくことである．

組込みソフトの分野では，全体の90%を占めるといわれる，例外処理／異常処理がこれにあたるだろう．

今回は，この骨格固めの段階で，複写機制御ソフトウェアの最大の難所である「キャンセル処理」をどのように実現するかを検討し，その仕組みをモデルの骨格に盛り込んだ．

5. 開発チームの体制

開発チームの体制としては，スキャナ／プロッタ共に，下記の表1のように役割を明確に分担して進めるようにした．

Table 1　開発体制における役割

管理職／機種リーダー	マネージャー	プロジェクトのQCD管理 マスタープランの立案と運営
	テクニカル･リーダー	開発方針の決定 全体構想
横串機能	アナリスト	要件定義と要求分析 レビューアー
	モデリング･エキスパート	モデリング方針／設計指針の策定とモデルのチェック
	アーキテクト	システムレベルの設計 共通メカニズムの設計
機種ごとの開発者	デザイナー	要求仕様のモデル化 フレームワークモデルの開発
	プログラマー	開発言語にもとづくプログラミング
	テスター	ソフトウェアの評価 品質保証

ここでは，アナリスト及びアーキテクトに専任者をアサインし，いつでも必要に応じてレビューできる体制を敷くことが成功へのカギとなる．とくに，全てのサブシステム間のバランスや，モデルの均質化を行うアーキテクトの役割は重要である．

また，今回はとくに，対象領域＝ドメインの知識はないがモデ

リング・スキルの高いモデリング・エキスパートには頼らず，ドメインを良く知っているドメインエキスパートのモデリング・スキルを向上させながらモデリングを進めるようなやり方を採り，体制づくりを進めた．

6. 開発の進め方

体制以外に重要なのは，開発の進め方である．

とくに骨格固めのフェーズでは，毎日チーム全員が参加するウォークスルー形式で分析を進め，システムの骨格作りにあたって理解と意識を共有した．とくにこの期間では，だれか一人が分析を行い，その成果物を定期的に全員でレビューする，という従来のやり方では，効果が期待できないことを認識する必要がある．

また，開発を進めるにあたっては，従来手法同様，成果物に関する規約を決定し，それを徹底した．オブジェクト指向固有の成果物に関する規約としては，クラス図の記述ルール，クラス／属性／メソッド／ロール名などの命名規則，コード生成ルールなどを定めた．

このように規約を決めるほか，開発プロセスの定義を行い，各開発フェーズで作成すべき成果物を規定することも重要となる．管理すべき設計成果物としては，ユースケース・モデル（ユースケース図，ユースケース記述，コンテキスト・ダイヤグラムなど），概念モデル（概念図，用語辞書など），分析モデル（全体のクラス図と関係記述，オブジェクト記述と状態モデル，コラボレーション図とシーケンス図など．通常は，Rose2000による日本語表記モデルがこれにあたる），設計モデル（イベントのディスパッチメカニズムやインスタンス管理メカニズムも含めて，ソースコードが生成可能なモデル．通常は，Roseによるコード導出が可能な英文表記のモデルがこれにあたる），実装モデル（コンパイル可能なソースコード），およびPCシミュレーション・プログラムとその結果などがある．

Rose2000をはじめとする，オブジェクト指向開発をサポートするツールは，これらの成果物をひとつのファイルで一元的に管理でき，またできあがった設計成果物からソースコードを半自動的に生成できる．したがって，何か修正点があった場合でも，直接ソースコードで修正するのではなく，必要ならクラス図まで戻って修正を加えるやり方を進めることで，ソースコードともとの設計モデルはほぼ一致する．

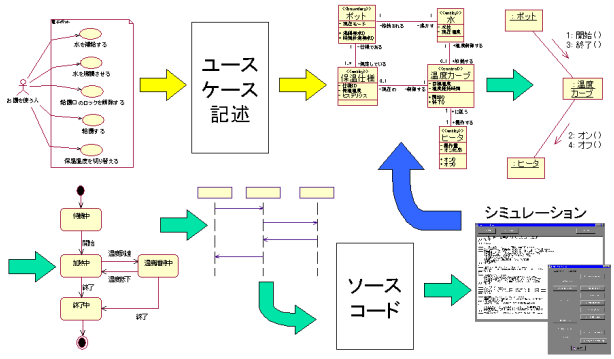

Fig.4　オブジェクト指向開発の成果物

そのため，設計レビューはモデル図をもとに行うことができ，ソースコードの詳細な内容を知らない他の担当者や管理者から見ても，密度の濃いレビューを行うことが可能になる．

オブジェクト指向開発手法によって，ソースコード・レベルの再利用ではなく，モデル・レベルの再利用＝上流での品質確保が容易に行えるようになる，といわれるゆえんである．

7. 作成したモデルの特徴

オブジェクト指向開発手法で構築されたモデルは実際にはどこが従来手法に比べて優れているのか，を見るため，今回オブジェクト指向開発手法をを導入したことで，もっともそのメリットが現れた，骨格部分のモデリングを例にとって説明する．

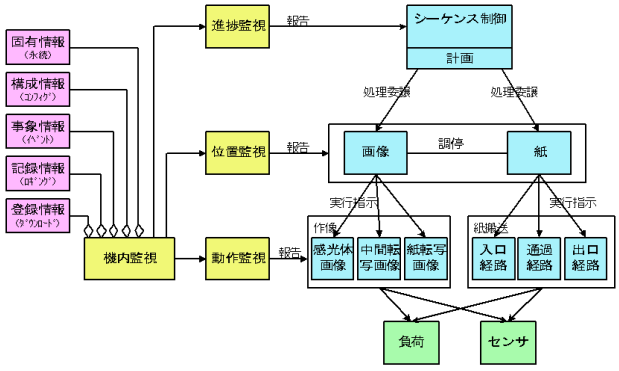

Fig.5　全体モデル図（プロッタ側）

今回のモデリングのポイントは，次の2点であった．

(1)　まずソフト障害が，ソフトウェア複雑度と深い相関があることに着目して，「複雑さ」を低減すること

(2)　そして，モデル・ベースでの再利用を進めるため，モデル・レベルで徹底して抽象化を行うこと

まず「複雑さを低減」するため，ここでは制御のレベルを階層化することにより，複雑さを分散させることにした.

すなわち，モデリングにあたっては，制御のレベルを階層化し，「計画（プラン）」層（投入された「指示」に基づき，作業のスケジューリングを行う），「調停（コントロール）」層（動作実体への実行指示と，異なる動作間のタイミング調整を行う）と，「動作（モーション）」層の3階層とすることに留意した.

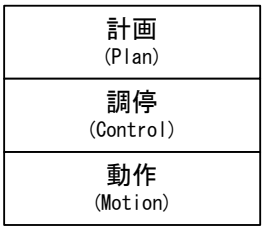

Fig.6　知的階層

また，「再利用性を高める」ため，まず従来どのような要求仕様の変更が多いかを振り返り，それにもっとも対応しやすい形でモデル化することを考えた．やはり従来最も多いタイプの仕様変更は，各負荷の制御タイミングの変更である．すなわち，制御タイミングの処理部分を局所化し，仕様変更に強いモデルとする必要がある.

そこで，前記の3階層を念頭に置いたうえで，制御タイミングの処理部分を局所化したモデリングを進めていった.

Fig.7に，今回作成した骨格モデルと従来モデルを比較する形で表現してみた.

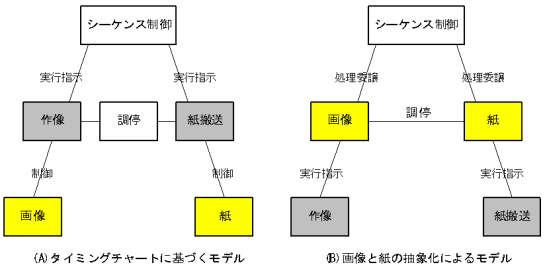

Fig.7　2つの抽象化視点

従来手法で構築されたモデルは，タイミングチャートの視点でモデル化するので，作像制御や紙搬送タイミングに対して，時系列に実行処理を指示することになり，Fig.7の左側（A）のような形になるのが一般的である.

すなわち，従来モデルでは，「シーケンス制御」で，「作像」と「紙搬送」へ順次実行処理を発行し，その2つのタイミングを調停するような存在も必要になるので，複雑な構成となる．また，このモデルでは，「画像」や「紙」は，実行処理の結果として動作するだけなので，モデルとしての意味（責務）は持ち合わせないものとなる．つまり，「画像」や「紙」は制御されるだけの存在となる.

従来モデルの問題点は，制御タイミングの処理部分がモデルの中心部分である「作像」と「紙搬送」部分に実装されていることである．すなわち，この部分が変動部となり，変更による修正範囲が広くなってしまうため，品質が劣化するだけでなく，異なる機種への展開が困難になる．つまり再利用性が低い.

これに対し，オブジェクト指向的アプローチで作成したモデル（B）は，制御対象としての「紙」と「画像」に着目してモデリングしたことを特徴とする.

このモデルでは，「シーケンス制御」が「画像」と「紙」に処理を委譲する構造となり，「画像」と「紙」は自分自身の状態を遷移させるために，作像や紙搬送へ実行指示を発行する．「作像」や「紙搬送」は，指示された動作を単純に実行するだけ，という役割分担となる．すなわち，（B）のモデルは，まずモデルの中心部分が抽象的な概念同士が連携して動作する，という汎用性の高い構造になっており，ちょっとしたタイミングの変更は，「作像」や「紙搬送」というモデルの下位層（知的情報を持ち合わせない）で対応できるため，局所的な修正で済む，という大きな特長がある．これにより，複数の機種にも適用可能なモデル化ができる．つまり再利用性が高い.

このように，オブジェクト指向的アプローチを採ることによって，従来のタイミングチャートの視点から，オブジェクト指向のメリットを最大限に活かした再利用性と品質の高いモデルを構築することができた.

8. 新手法の適用結果

オブジェクト指向開発手法にインクリメンタル開発プロセスを組み合わせた本取組みの成果はどうであったか，次に述べる．まず成果を客観的に評価するため，Q・C・D（品質／コスト／納期）それぞれの視点について，従来機種と数値で比較する.

(1)　Q：Qualityの比較結果：ソフトウェア複雑度

ソフトウェアの客観的な品質は向上したか？　という観点から

は，QAC解析結果で比較した．

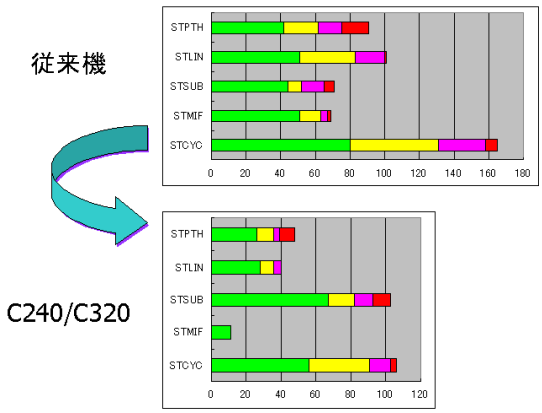

Fig.8　従来機種とのソフトウェア複雑度の比較

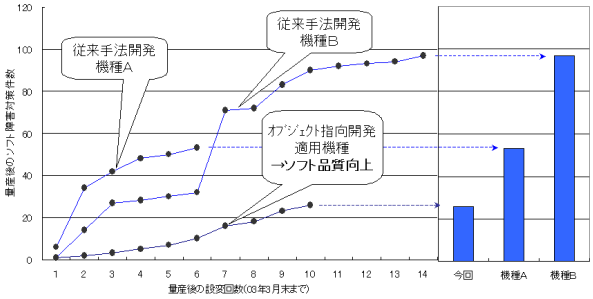

Fig.9　従来機種とのソフトウェア障害数の比較

従来に比べ複雑度がかなり低減されているのがわかる．実際，障害も15%程度減少した．とくに，量産後の品質は従来機に比べると格段に向上しており，新手法を適用した成果が良く現れている．

⑵　C：Costの比較結果：ROM/RAM容量

従来機種に比べて，コスト上昇となっていないか？　という観点からは，プログラムが格納されるROMの容量と，使用メモリ量としてRAM容量で比較した．

従来のカラープリンタ機種ソフトと比較すると，ROM容量は増加せず，実行ステップ数も，ほぼ同じであった．また，RAM容量はむしろ減少した．これはオブジェクト指向手法によるモデルでは，システムの処理を関数呼び出しの組合わせという形で実現できるため，システムを多くのタスク数に分割する必要がないためである．

タスク数に着目すると，従来機で50～70個必要だったものが，オブジェクト指向手法の適用機種は十数個ですむ．

⑶　Delivery比較結果：開発工数／期間

完成までに要した工数は低減できたか？　という観点からは，従来機種のソフトウェア開発工数と比較した．

オブジェクト指向手法を適用した1機種目の開発工数は従来比で1．5～3倍を要する，とされているが，結果はむしろ従来機と同等であった．工数ベースにすると10%減，開発期間では15%減を果たした．

やはり，骨格固めの時期で徹底的にウォークスルーを繰り返したことで，文字どおり上流での品質確保ができた結果である，と考える．

9．成果と反省点

エンジン制御ソフト（プロッタ＋スキャナ）のほぼ全体をオブジェクト指向開発手法により設計完了し，商品化できた．いきなり高いハードルを狙わず，小から大へ段階的適用をはかっていったやり方の成果が出たといえる．

諸般の事情によりテーマ開発中にはメンバーの入れ替わりも発生したが，従来手法では引継ぎ資料としてはソースコードのみという場合が多いが，今回はモデル図というプログラム構造が可視化された設計資料を元に引継ぎを行うことができ，より円滑に引継ぎを行うことができた．

今後の新規開発機種はすべてオブジェクト指向開発手法による，ということになるが，より上流へシフトし「つくらずにつくる」コンセプトを具現するため，C240／C320のモデルを洗練し，モデルベースのソフト再利用を加速していく方針である．