半導体製造装置の FA 間通信ソフトウェア開発
～オブジェクト指向リファクタリングのプロセス改善～

M2007MM002 安藤貴広
指導教員：野呂昌満

1　序論

我々は OJL として，半導体製造装置の FA 間通信ソフトウェア開発を行った．FA 間通信ソフトウェアは，手続き指向言語で作成されており，作成され始めてから 10 年以上経過している．その間，場当たり的な機能追加や変更の繰り返しによって記述が散在し，開発者以外の者がそのコードを理解し，修正することは容易ではなくなっている．原因は，設計思想がソフトウェア全体に行き渡っておらず，ソフトウェアの保守性が低下していることが挙げられる．

FA 間通信ソフトウェアは，複数の作業をいくつかの装置上で処理する時，それらの最適な処理順序を求める問題であるスケジューリング問題を解決するソフトウェアである．プロセスを管理する構造を持つ既知のソフトウェアとして，オペレーティングシステム (以下，OS) のプロセススケジューラ [4] がある．FA 間通信ソフトウェアは，OS のプロセススケジューラのソフトウェアアーキテクチャに基づいて開発されるべきである．

ソフトウェアの保守性に対する有効な手段の一つとして，データ抽象化と多相性の概念 [6] を背景としてソフトウェアの振舞いを変更せずに構造を変更する [7] 技術がある．本稿では，これをオブジェクト指向リファクタリングと呼ぶ．一般的に，リファクタリングを行うにあたり，系統的な開発プロセスが決定されていない．また，ソフトウェア開発においてソフトウェアアーキテクチャが重要である．これらのことから，ソフトウェアアーキテクチャ [5] に基づいたオブジェクト指向リファクタリングを行うことで，設計思想を統一し，系統的な開発を行うことができると考えた．

本 OJL の目的は，オブジェクト指向リファクタリングを行う際の開発プロセスを提案し，FA 間通信ソフトウェアを再開発することである．

提案する開発プロセスでは，問題を解決する既知のソフトウェアアーキテクチャを基に対象ソフトウェアのソフトウェアアーキテクチャを構築する．

構築したソフトウェアアーキテクチャを運用し，開発プロセスを繰り返し行うことで，対象ソフトウェア全てをリファクタリングすることが可能である．新たな仕様を復元することと，他のドメインへの適用を行うことで提案する開発プロセスの有効性を考察する．

また，提案する開発プロセスは，OJL の期間毎に定義する必要があった．このことから，対象ソフトウェアの特性を加味した PDCA[3] サイクルに基づいた開発プロセスの改善プロセスを定義し，その妥当性を検証する．

本 OJL において習得する問題解決の基礎となるベース技術は，ソフトウェア開発プロセスとして，オブジェクト指向リファクタリング，ソフトウェアレビュー，ソフトウェアテストがある．また，ソフトウェアアーキテクチャとしてオブジェクト指向アーキテクチャスタイル，デザインパターン等がある．

ベース技術を選択，組合わせて適用することで問題を解決する技術であるメタ技術として，ソフトウェア開発プロセスの改善プロセスを習得する．

2　背景技術

2.1　オブジェクト指向リファクタリング

オブジェクト指向技術の背景知識となる本質的な概念は，データ抽象化と多相性である．また，リファクタリングとは，外部から見たときの振舞いを保ちつつ，理解や修正が簡単になるように，ソフトウェアの内部構造を変化させる技術である．

本稿では，これらの技術を合わせたものをオブジェクト指向リファクタリングと呼ぶ．オブジェクト指向リファクタリングは，データ抽象化と多相性の概念を背景としてソフトウェアの振舞いを変更せずに構造を変更し，オブジェクト指向コードに既存コードをマッピングする技術である．

本 OJL では，オブジェクト指向リファクタリングを行うことで，今後の機能追加等の変更に対して柔軟な構造に変更することができる．

2.2　反復型ソフトウェア開発プロセス

ソフトウェア開発プロセスは，ソフトウェア製品の開発工程を体系化し，生産性と品質を向上する．

反復型ソフトウェア開発では，短いサイクルでソフトウェアを作成することができ，開発期間中の要求の変更に対応することができる．

本 OJL では，オブジェクト指向リファクタリング時に反復型開発プロセスの利点を利用した．具体的には，リファクタリングする機能毎に開発プロセスを繰り返し，最終的に対象ソフトウェア全てをリファクタリングすることである．初めにリファクタリングする箇所は，仕様が明確に決定しているものから順に行う．

2.3　ソフトウェアレビュー手法

レビューには，管理目的のレビュー，改善目的のレビューがある．レビュー対象がどういう特徴があり，どの観点からレビューを行うかによって実施するレビュー手法を選択することで効率の良い開発を行う手助けとなる．

本 OJL において，仕様書，設計書，アーキテクチャ構築指針書に対してプロジェクト内で設計思想を統一するためにウォークスルーレビューを実施した．テスト仕様書に対しては，欠陥を見つけるためにインスペクションレ

ビューを実施する．

2.4 PDCA サイクル

PDCA サイクルとは，Walter A.Shewhart，W.Edwards Deming らによって提唱された生産管理や品質管理等の管理業務を計画通りに進めるための TQM(Total Quality Management) の一つの要素である．
PDCA サイクルは Plan(計画)，Do(実行)，Check(評価)，Act(改善) から成り，これら 4 段階を繰り返し行うことで，管理業務を改善していく．
OJL の実施期間毎に開発プロセスを定義する際に，以前に行ってきた開発プロセスの問題点を把握し，その問題を解決するために，開発プロセスをどのように改善すれば良いかを導く一連の改善プロセスとして用いる．

3 提案する開発プロセス

3.1 再開発とリファクタリング

我々は，半導体製造装置の FA 間通信ソフトウェアをオブジェクト指向リファクタリングにより再開発した．
一般的に再開発とは，リエンジニアリング [1] 等のソフトウェアを再利用するための技術や，リファクタリングといったソフトウェアの構造を整理する技術等を用いて，既存システムと同等，あるいはそれ以上の機能を持つ新たなシステムを作成することを指す．一般的なリエンジニアリングのモデルを図 1 に示す．

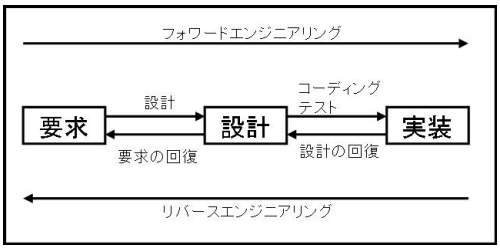

図 1 リエンジニアリングモデル

リエンジニアリングとは，リバースエンジニアリングとフォワードエンジニアリングを用いて，既存システムから新たなシステムをつくり出す技術のことを言う．リバースエンジニアリングとは，既存のソフトウェアのプログラムを分析し，既存の仕様を明確にする技術である．フォワードエンジニアリングとは，リバースエンジニアリングにより導き出した仕様に対して新たな要求を加えて，新たなソフトウェアを作成する技術である．
また，一般的なリファクタリングプロセスを図 2 に示す．ソフトウェアの振舞いを変更することなく，構造を変更する技術としてリファクタリングがある．リファクタリングプロセスは，既存のソースコードレビューを行い，どの箇所にリファクタリングを適用するか，どのリファクタリング技術を適用するかを決定し，その効果を測定した後，プログラムを修正する [2]．また，既存の仕様書や設計書等の成果物がある際には，それらもレビューし，リファクタリングを行う．
プログラムの修正後，テストを行い，テストをパスしたら設計書を作成・修正する．これらの一連のプロセスを繰り返すことで，対象ソフトウェア全てをリファクタリングすることが可能である．

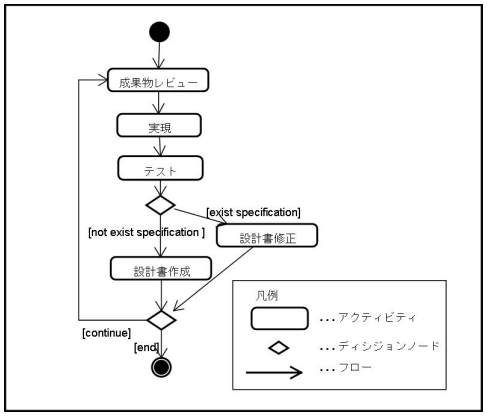

図 2 一般的なリファクタリングプロセス

一般的なリファクタリングの特徴として，ソースコードの一部分を機能を変えないで書き換えたり，リファクタリング途中で作業を打ち切ってもプログラムは正常に動作することが挙げられる．以上のことから，リファクタリングは，リエンジニアリングと同様，再開発のための一つの技術であると言える．しかし，リファクタリング自体が場当たり的なものになってしまい，保守性の低いソースコードが生成される可能性も存在する．
そこで，本 OJL では，ソフトウェアアーキテクチャに基づくオブジェクト指向設計によるモジュール化，リファクタリングを用いて再開発を行った．

3.2 提案する開発プロセス概要

提案する開発プロセスの概要を図 3 に示す．

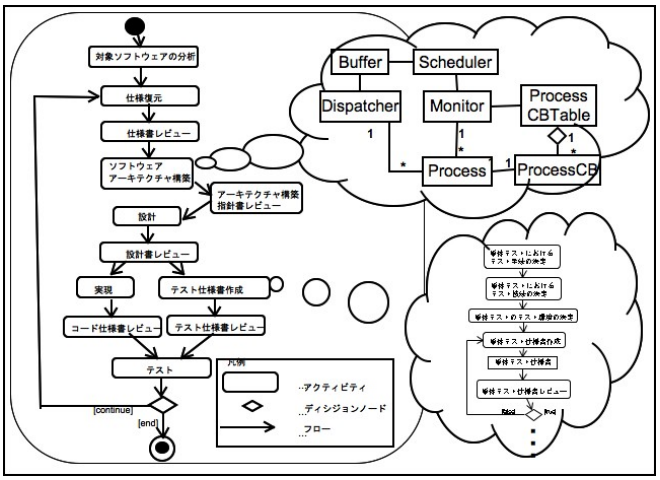

図 3 開発プロセスの概要

提案する開発プロセスは，対象ソフトウェアの分析，仕様復元，ソフトウェアアーキテクチャ構築，ソフトウェア設計，実現，テストの順でオブジェクト指向リファクタリングを進める．つまり，リエンジニアリングのプロセスに沿う．理由は，ソフトウェアアーキテクチャ中心に開発することで設計思想を統一した体系的なリファクタリングを行うことができるからである．これは，開発規模が大きいということや，開発期間，開発人数等の制約も関連し，序々にリファクタリング範囲を拡大していくことができるという意味を持つ．

提案する開発プロセスの各プロセスにおいて作成される成果物は，仕様書，設計書，アーキテクチャ構築指針書，オブジェクトコード，テスト仕様書である．これらの成果物に対するレビューを実施し，プロジェクト内での設計思想を統一し，誤りの発見を行う．
対象ソフトウェアの分析を行い，対象ソフトウェアの特性を解決するソフトウエアのソフトウエアアーキテクチャを構築する．構築したソフトウェアアーキテクチャに基づいてリファクタリングを進めることで，保守性の高い構造を保つことができる．
提案する開発プロセスの利点は，既知のソフトウェアアーキテクチャを基に開発を進めることで，保守性の高い構造のまま，対象ソフトウェア全てをリファクタリング可能ということや，体系的にリファクタリング可能なこと等が挙げられる．

3.3 開発プロセス改善プロセス

OJL の各期間毎に渡って様々な問題が生じてしまったことから，開発プロセスを改善してきたが，これは PDCA サイクルを用いた改善プロセスと同等のものであると言える．この場合，PDCA サイクルの Plan は，開発プロセスの定義，Do は開発，Check は開発プロセスにおける問題点の把握、Act は問題点の改善，開発の終了もしくは，継続に相当する．
一般的に，ソフトウェア開発プロセス改善の指標として SW-CMM[8] がある．しかし，SW-CMM は，いくつものプロセスエリアを尊守しなければならず，手順書や文書化に関する整備不良等のプラクティスの省略が多く，実際に組織へ適用する上では困難な場面も存在する．
このことから，PDCA サイクルを用いることで，成熟度モデルを用いることなく開発プロセスを継続的に最適化することができると考えた．実際の SW-CMM では成熟度レベルを越えて次のレベルへ移ることはできないが，本 OJL で用いたソフトウェアアーキテクチャに基づく PDCA サイクルは，SW-CMM における成熟度レベル 5 である「最適化しているプロセス」引く，レベル 4 である「定量的に管理されたプロセス」と同等な意味を持つものであると考える．

4 考察

4.1 提案する開発プロセスの有効性

提案する開発プロセスに基づいて「プロセス間の同期」を復元した例を説明する．
プロセス間の同期を復元する際に，OS のプロセススケジューラのソフトウェアアーキテクチャに基づいて，装置制御プロセスを管理するシステムのソフトウェアアーキテクチャの変更箇所を明確にした．一般的な OS におけるプロセス間の同期の実現は，ProcessCBTable にプロセス識別子を持たせる．よって，装置制御プロセスを管理するシステムのソフトウェアアーキテクチャの ProcessCBTable にプロセス識別子を持たせることで解決できる．
ソフトウェアアーキテクチャ構築の際に，アーキテクチャ構築指針書を作成し，レビューすることで，プロジェクト内で設計思想を統一することができた．アーキテクチャ構築指針書 (抜粋) を図 4 に示す.

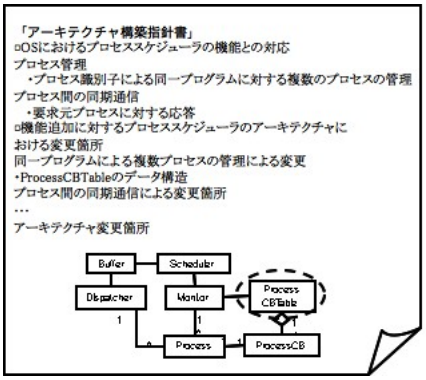

図 4 アーキテクチャ構築指針書 (抜粋)

アーキテクチャ構築指針書レビュー，ソフトウェアアーキテクチャ構築後，ソフトウェア設計，実現，テスト全てにおいて，それまでの OJL で作成した成果物を利用することができる．これは，リファクタリング箇所を明確にすることで，ソフトウェアアーキテクチャの変更箇所が明確になり，それに対する各開発プロセスが決定されると言い替えることができる．初期の OJL における開発プロセスと提案する開発プロセスの比較を図 5 に示す.

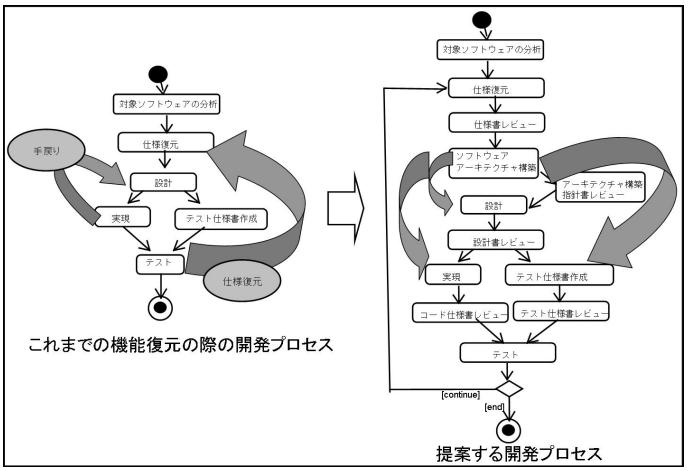

図 5 これまでの機能復元時の開発プロセスと提案する開発プロセスの比較

これまでの機能復元時の開発プロセスは，設計思想を統一させる指針もなく，レビューも行っていないので手戻りや無駄な作業が生じてしまった．
提案する開発プロセスに基づいて開発を行った結果，構築したソフトウェアアーキテクチャに対応した仕様復元，ソフトウェア設計，テストを行うことができ，手戻り作業や無駄な労力を軽減できた．
提案する開発プロセスに基づいて開発を行うことで，開発効率が向上し，系統的なリファクタリングを行うことができる．以上のことから，提案する開発プロセスは，オブジェクト指向リファクタリングを行う際に有効だと言える．

4.2 他のドメインへの適用による開発プロセスの妥当性の検証

本 OJL における対象ソフトウェアと同様の特性を持つものとして SOA(Service Oriented Architecture)[9] のミド

ルウェアであるサービスブローカが挙げられる．サービスブローカには，ケジューリング，ルーティング，トランザクション管理等の機能がある．サービスリクエスタと，サービスプロバイダ間のサービスを実現するサービスブローカのソフトウェアアーキテクチャを構築することで提案する開発プロセスに基づいて再開発できると考える．提案する開発プロセスに基づいてサービスブローカの再開発を行った．開発プロセスの工程であるアーキテクチャ構築の際に，機能に対するソフトウェアアーキテクチャの変更箇所を明確にする．この開発プロセスに基づいて開発を行うことで，特にスケジューリングやルーティングに関する箇所，例えば，サービスプロバイダからサービスブローカへサービスのインタフェースを登録する等の機能に対するテストを徹底的に行うことがわかる．
また，復元する機能が決定されると，SOA における UDDI(Universal Description,Discovey,and Integration) は，プロセススケジューラにおける ProcessCBTable クラスに対応する等がわかり，機能を復元するためのソフトウェアアーキテクチャ構築の指針が決定される．
以上のことから，OS のプロセススケジューラの特徴を持つ他のソフトウェアに対しても提案する開発プロセスは有効である．提案する開発プロセスはソフトウェアの再開発時の開発プロセスとして妥当だと考える．

4.3 開発プロセス改善プロセスの妥当性の検証

提案する開発プロセスと SW-CMM を比較し，PDCA サイクルを用いた開発プロセス改善プロセスと SW-CMM を比較することで，PDCA サイクルを用いた提案する開発プロセス改善プロセスが妥当であることを検証する．
提案する開発プロセスは，SW-CMM の成熟度レベル 3 の「定義された段階」にあたると考える．理由は，開発プロセスとして定義し，それに基づいて再開発を行っているからである．
PDCA サイクルを用いた提案する開発プロセスの改善プロセスは，成熟度レベル 5 の 3 個のプロセス領域を達成する仕組みを備えている．しかし，SW-CMM では，実際に SW-CMM の成熟度レベルを飛び越して次のレベルへ移ることはいけない．何故ならば，各々のレベルでは，次のレベルを達成するのに必要な基盤を形成しているからである．しかしながら，提案する開発プロセス改善プロセスは，継続的に改善され，常に最適化されていることは事実である．
また，PDCA サイクルは，一般に問題点の把握である Check が困難であると言われている．その原因として，問題点を問題として捉える基準がないことが挙げられる．問題点を判断する基準がないことから，改善である Act もより困難とされる．ソフトウェアアーキテクチャに基づく PDCA サイクルの場合は，ソフトウェアアーキテクチャを Check の基準にすることができる．つまり，ソフトウェアアーキテクチャの妥当性の検証から問題点の把握を始めることができる．ソフトウェアアーキテクチャの妥当性の検証後，ソフトウェアアーキテクチャに基づくソフトウェア設計手法，ソフトウェアレビュー手法，ソフトウェアテスト手法を検証することで開発プロセスにおける問題点を把握する．言い替えると，ソフトウェアアーキテクチャに基づいた PDCA サイクルにより，Check の手順を明確にすることができたと言える．
このことから，ソフトウェアアーキテクチャに基づいた PDCA によって Check した結果，Act の手順も決定されることがわかった．Act において決定された開発プロセスは，次開発における Plan の要因となり得る．
以上のことから，本稿では，提案する開発プロセスの改善プロセスは，改善プロセスとして妥当であると考える．

5 結論

本 OJL の成果は，リファクタリングをソフトウェアアーキテクチャに向かって進める開発プロセスの提案と，開発プロセス自身の最適化である．
一つ目は，プロジェクトの成功に必要な本質的なこととして，ソフトウェアアーキテクチャに基づいたオブジェクト指向リファクタリングプロセスを定義することができたことである．二つ目は，開発プロセス自身を最適化するために，PDCA サイクルを利用し，継続的なプロセスの改善を行うメタ技術を習得することができたことである．
今後の課題として，メトリクス等の定量評価手法を用いた改善度合の調査，ソフトウェアアーキテクチャの変更により，開発プロセスが明確に決定される箇所の特定が挙げられる．

参考文献

[1] Rovert S. Arnold, “Software Reengineering,” *IEEE Computer Society Press*, 1993.

[2] T. Mens and T. Tourwe, “A survey of software refactoring,” *IEEE Transaction on Software Engineering*, Vol30, p.126-139, 2004.

[3] W. E. Deming, *Quality, Productivity, and Competitive Position*, MIT, Center for Advanced Engineering Study, Cambridge, Mass., 1982.

[4] Andrew S. Tanenbaum, *Modern Operating System*, Prentice-Hall, 2001.

[5] L. Bass,P. Clements and R. Lazman, *Software Architecture in Practice 2nd ed*, Addison-Wesley, 2003.

[6] G.Booch, “Object-oriented development,” *Proc.IEEE Transaction on Software Engineering*, Vol12, p. 211-221, 1986.

[7] Martin Fowler,*Refactoring Improving the Design of Existing Code*, Addison-Wesley, p. 53-399, 1999.

[8] M. C. Paulk, B.Curtis, M. B. Chrissis and C. V. Weber, SEA-SPIN/CMM 研究会訳, ソフトウェア成熟度モデル 1.1 版, Software Engineering Institute, Carnegie Mellon University, 1993.

[9] Dirk K. Banke,and D. Slama 著, 山下真澄監訳, SOA 大全, 日系 BP 社, 2006.