ソフトウェア工学 103－2
（1995．3．9）

通信ボード組込みソフト開発プロセスの実際と障害の分析

端山 毅　　深海 悟

NTTデータ通信(株) 技術開発本部 情報科学研究所

概要

通信ボード用の組込みソフトウェア開発プロジェクトの調査に基づき、この分野で使用されている仕様の表現形式と基本的なソフトウェア・アーキテクチャについて報告する。また、このプロジェクトにおける再利用率と故障の発生件数を比較することにより、再利用の実態とその効果について考察した。この事例から、ソースコードの再利用は、単体試験工程までは障害の抑制に効果があるが、結合(統合)試験工程以降への効果は少ないことが判明した。

Analysis of Software Process and Fault Data in Software Development for Embedded Communication Program

Takeshi Hayama　　Satoru Fukami

Laboratory for Information Technology
Research and Development Headquarter
NTT Data Communications Systems Corp.

Abstract

A result of an investigation on an actual software development project is presented. The project developed an embedded software on a communication board for an application-specific terminal. We show the software architecture and the domain model which are assumed and used in the project. Some findings and effects on reuse are also discussed by analyzing the relation between reuse ratios and numbers of faults detected in the test phase. This case study shows that, by reusing source codes, the number of faults detected in the unit test phase is reduced, while the one detected in the integration test phase is not.

1 はじめに

通信ボード用の組込みソフトウェア開発プロジェクトを対象に、開発過程でどのような事態が発生し、それらがどのように関連しているか調べた。調査方法としては、設計文書や試験結果等の資料調査を中心に、合わせて担当者へのインタビューを行った。本報告では、この分野で使用されている仕様の表現形式と基本的なソフトウェア・アーキテクチャ[2, 3]について述べ、また、このプロジェクトにおける再利用率と故障の発生件数を照合することにより、再利用の実態とその効果について考察した。

2章では、調査対象プロジェクトの概要を紹介した。3章では、このプロジェクトの開発プロセスのあらましについて述べる。4章では、通信プロトコルの分野におけるドメインモデル[1]について述べる。5章では、ソフトウェアアーキテクチャについて述べる。6章では、再利用率と故障発見件数の関係から、再利用の実際について考察する。

2 調査対象プロジェクト

調査対象プロジェクトは、端末開発プロジェクトの一部を構成する通信ソフトウェア開発プロジェクトであり、当社開発担当者1人と協力会社要員数人で実施された。プログラム設計から結合試験まで(工程については3章参照)は協力会社が担当した。開発規模はC言語とアセンブラを合わせて数十KLOC程度である。

この通信ソフト開発プロジェクトは、親プロジェクトによって目的、仕様概要が決められている。基本的には既存製品との互換性を維持することが主眼であり、斬新な製品を開発するわけではない。

調査対象プロジェクトが開発するソフトウェアは、端末の通信オプションボード上で動作するもので、当社独自仕様の2種類のプロトコル(以下本稿ではAプロトコルとBプロトコルと呼ぶ)を実装している。両プロトコルとも不平衡なプロトコルであり、主局と従局の区別がある。

このボードは自社開発したもので、CPUを搭載したある程度汎用的なものである。また、専用OSが別途開発、搭載されており、端末本体とボード間の通信等共通的な機能を担当している。調査対象プロジェクトが開発した通信プロトコルソフトウェアは、プロトコルの違いと主局/従局の違いで合計4種類あり、起動時に端末本体からダウンロードされ、このボードおよび専用OSのもとで動作する。このボードと専用OSの組合せを用いて、この他にも複数のプロトコル用のソフトウェアが開発されている。

Aプロトコル、Bプロトコルとも、当社製品に古くから使用されているもので、このプロトコルの実装は同一組織内で何度も実施されている。今回は、新たなハード、OS等の環境に合わせて作り直すことになった他、細かな仕様が追加されている。また、開発担当者は、同種のボード用ソフトウェアについては十分な経験があるが、今回のプロトコルについては始めてであった。

なお、Bプロトコルの仕様書は厳密かつ詳細であるが、Aプロトコルの仕様書は不十分である。

3 開発プロセス

当社の社内標準では、

1. 基本検討(BI)
2. 基本設計(BD)
3. 詳細設計(DD)
4. プログラム設計(PD)
5. コーディング(C)
6. 単体試験(UT)
7. 結合試験(SI)
8. 総合試験(PT)
9. 総合運転試験(RT)

の順に実施されるウォータフォールモデルに基づく工程を定義している。各工程の終了時には、工程品質判定書が作成され、レビュー結果や試験結果の定量的データを沿えて、上級管理者に状況報告がなされる。

当プロジェクトでは、公式プロセスに則って基本検討書、基本設計書、詳細設計書、プログラ

ム設計書を作成しており、基本設計以降についてはそれぞれ工程品質判定書も作成されている。しかし、これらの生産物の内容を照らし合わせつつ読んだ結果、以下のことが判明した。

1. 基本検討書は再利用可能性検討書であった
   基本検討書は協力会社側で作成しており、過去のプログラムおよびドキュメントを当社側から提供し、どの部分が再利用できるか検討している。

2. 基本設計書と詳細設計書の内容はほんとんど同じであり、概要しか書かれていない。これらとは別にプロトコル仕様書がある。

3. プログラム設計書は、基本設計書や詳細設計書に比して量が多いが、実は概要部分についてしか書かれていない。

4. 基本設計工程と詳細設計工程で重点的に行われた作業は、既存プログラムの調査であった。

5. コーディング工程以降、総合運転試験工程に至るまで、繰り返しソースコードレビューが実施されている。

これらのことから、このプロジェクトは当社担当者がプログラムの細部まで理解し、細かく指示を出すことで成り立っており、ソースコードで管理されている。

4 通信プロトコルのドメインモデル

通信プロトコルの世界におけるモデルとしてはOSIの基本参照モデルが有名であるが、実用に供されているプロトコルは、必ずしもOSIのモデルに当てはまるわけではない。特に古くから使用されているプロトコルは、明確な階層化がなされているわけではない。

しかし、長年の経験に基づき、プロトコルの実現を表現する方法が醸成されてきている。その中心的役割を持つのが状態遷移表である。例えばJIS X5251「ローカルエリアネットワークの論理リンク制御」においては、表1のように整理している。また、JIS X5109「開放型システム間相互接続のコネクション型トランスポートプロトコル仕様」等では、表2の形式の状態遷移表を用いている。

現在の状態	要因	動作	次の状態

表 1: 状態遷移表形式1

事象 \ 状態	状態1	状態2	...
事象1	動作 遷移先		
事象2			
事象3			
⋮	⋮	⋮	

表 2: 状態遷移表形式2

今回調査した当社プロジェクトにおいては、以下のような状態遷移表と処理状態遷移表を使用している。表3の状態遷移表は、状態(1〜5)において入力事象(A〜M)が発生した場合に、どういう処理を実行して、どの状態に遷移するかが記入される。状態遷移表のすべての欄に処理が存在するわけではなく、各状態において無効であり無視される入力もある。表3の右下の記述は、5Mの処理を実行し、3の状態に遷移することを意味する。この5Mの処理が何であるかは、次の処理状態遷移表に記載される。

表4の処理状態遷移表の入力要因の欄には、状態遷移表の入力要因(A〜M)のうち有効な組合わせのみが記載される。索引番号は状態遷移表中の5M等の番号であり、この欄によって状態遷移表と処理状態遷移表が結びつけられている。処理状態遷移表が明確に決定されれば、ひとつひとつの処理を実行するプログラムを作成すればよい。

5 通信プロトコル実装における基本アーキテクチャ

上記の状態遷移表と処理状態遷移表は、基本的なソフトウェアアーキテクチャを想定してい

		上位モジュール				回線割込制御				タイマ制御			ビジー制御	
		A	B	C	D	E	F	G	H	I	J	K	L	M
状態	1													
	2													
	3													
	4													
	5													5M / 3

表 3: 状態遷移表の形式

状態	入力要因	処理	上位モジュールへの出力	回線制御モジュールへの出力	タイマ制御モジュールへの出力	索引番号
1						1A
⋮	⋮	⋮	⋮	⋮	⋮	⋮
5						5M

表 4: 処理状態遷移表の形式

る。すなわちプロトコルの処理を実行するソフトウェアにとっての入力事象および出力対象が網羅的に整理されている。

1. 上位モジュールからのコマンド
2. 回線制御部からの割込み
3. タイマからの割込み
4. 内部的に生じる事象 (ビジー)

の 4 種類の入力事象があり、それぞれに対応する出力がある。すなわち図 1の関係が想定されている。

この結果としてモジュール構成も概ね定まっている。(図 2) このうち、本体とのインタフェース部分は本体側のハードウェアや OS に大きく左右される。また回線制御部分は、通信ボードのハードウェアに依存している。共通処理部分はタイマ等ボード上のハードウェアに影響される部分もある。これに対して層管理部やマトリックス処理部はプロトコルの仕様によって決定されるものであり、ハードウェアや OS 等の変更から直接的影響を受け難い部分である。

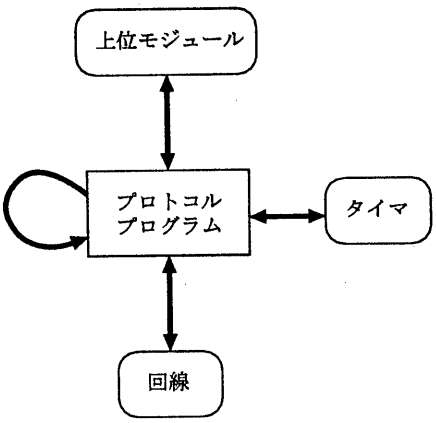

図 1: プロトコルプログラム周辺の実体

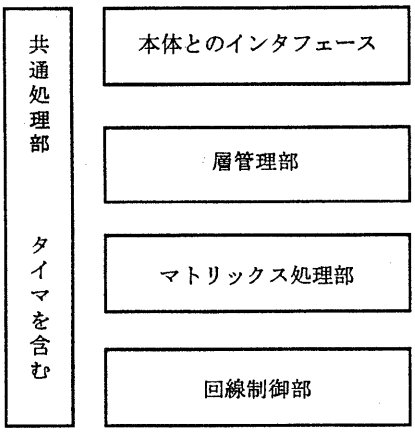

図 2: プロトコルプログラムの構成

今回の調査対象プロジェクトでは、本体とのインタフェース部分と共通処理部分をボード用の専用 OS としてまとめている。また B プロトコルのマトリックス処理部分は、既存ソフトウェアの再利用を図っている。

6 再利用

6.1 再利用率

マトリックス処理部と回線制御部について、A プロトコルと B プロトコルの再利用状況を表 5に示す。図 6は層管理を含めた再利用状況である。

プロトコル	完全再利用	改造再利用		新規作成
		再利用	改造	
A	0%	23.5%	35.4%	41.1%
B	1.7%	21.1%	62.0%	15.2%

表 5: マトリックス処理部と回線制御部の再利用率

完全再利用は、関数等まとまった単位を無修正で再利用した場合である。改造再利用は、変更を伴う再利用で、このうち変更しなかった部分を再利用、変更した部分を改造として分類している。表中の数字は行数を割合で示したものである。なお、行数で見た開発規模は A, B プロトコルで大差ない。ただし、A プロトコルの方がアセンブラを使用した部分が多い。

プロトコル	完全再利用	改造再利用		新規作成
		再利用	改造	
A	0%	16.4%	47.4%	36.2%
B	19.9%	34.5%	37.6%	8.0%

表 6: 全再利用率

マトリックス処理部と回線制御部について、A プロトコルの方は新規作成部分が多く、B プロトコルは大幅な変更を要したもののベースとなるプログラムがあったことを示している。

層管理については、A プロトコル用に作成したものを B プロトコルに流用しているので、全体で見ると B プロトコルの再利用率が大幅に高くなっている。

この再利用の違いが、試験工程における故障発見件数にどのように影響するか調査した。

6.2 故障発見数

当社の開発標準に基づけば、コーディング後、単体試験、結合試験、総合試験、総合運転試験が実施される。調査対象プロジェクトでは、ソースコードレビューで発見された故障も試験工程の障害報告書に記載している。

6.2.1 単体試験

	レビュー	マシン	合計
層管理	27.1	72.9	100
A	92.7	219.8	312.5
B	94.8 (66.7)	90.6 (46.9)	185.4 (113.6)

表 7: 単体試験の故障発見件数比率

表 7の数値は、層管理部分で発見された故障の合計を 100 としたときに、単体試験工程において各部で発見された故障件数の比率を表している。() 内の数値は、再利用したコードの不要な部分を削除する指摘の件数の比率であり、94.8 件

相当分の故障のうち、66.7件相当分がこのような削除の指摘であったことを表している。このような指摘が単体試験工程の障害報告書に多数記載されていた。

6.2.2 結合試験

	層管理	マトリックス処理部	回線制御	合計
A	18.9	8.1	73.0	100
B	32.4	21.6	56.8	110.8

表 8: 結合試験の障害発見件数比率

表8の数値はAプロトコルについて発見された故障数を100としたとき、結合試験工程において各部で発見された故障件数の比率を表している。

6.2.3 プロジェクトの特徴

調査対象プロジェクトでは、結合試験以降でも繰り返しソースコードレビューを実施している。これは、ドキュメントが十分整っていないことに起因し、ソースコードレベルで管理し、品質を維持しているものと考えられる。また、ソースコードレビューによって発見された故障に対しても、マシン試験によって発見された故障と同様に障害処理票を作成している。

6.3 故障発見数と再利用の関係

AプロトコルよりもBプロトコルの方が、故障が少なくなると予想される要因が存在する。

1. 層管理部分はAプロトコル用のものをBプロトコルにおいて流用している。
2. Bプロトコルのプロトコル仕様書が詳細かつ明確である。
3. Bプロトコルの方がアセンブラを使用した部分が少ない。
4. Aプロトコルを先に作成しており、要員の習熟が予想される。

このような要因があるにも関わらず、結合試験での故障発見数はむしろBプロトコルの方が多くなっている。

このことは、コードレベルの再利用の限界を示している。コードを再利用した場合、単体試験レベルでの故障発見数は減少するが、コード間の関係を試験する結合試験工程以降では、明確な効用は期待できない。

担当者のコメントによると、処理状態遷移表の各桝目に対応する処理プログラムを再利用する際に、大域変数の操作やバッファ等記憶領域の管理に対して、元のプログラムが様々な暗黙の制約を有しているため、複数のプログラム間の影響を考慮しつつ変更する必要があったとのことである。

新しいプログラムを開発するにあたり、旧来のものと異なる要求としては、

1. ハードウェア、OS等に依存する部分
2. プロトコルの細部に関わる部分
   (a) パケット長等、各種サイズの変更
   (b) エラー処理の変更

があり、図2の特定の部分に封じ込められるような変更ではなく、複数の部分に影響が及んでしまう。特にエラー処理については、エラー発生時にどのような反応を返すのか、あるいは返さないのかといった要求にしたがい、場合によっては、回線制御部からマトリックス処理部、層管理部を経て本体とのインタフェース部分にまで変更が及ぶ。すなわち、変更の影響を局所化できていない。

このような原因により結合試験における故障発見数は、Bプロトコルにおいても減少していない。

また、単体試験における「再利用したコードの不要な部分を削除する指摘」の多さも本プロジェクトの性格を示している。この指摘を故障と見なさなければ、単体試験におけるBプロトコルの故障発見数は、Aプロトコルのそれの1/4に減少したことになる。

これらの指摘事項はコーディングに入る前に、プログラム設計書に盛り込まれていることが、ソ

フトウェア工学の常道であろうが、現物を目の前に具体的に指示する方法が悪いのかどうか、一概に断定できない。

現状の処理状態遷移マトリックスを中心とした開発方法では、データよりも処理を重視した開発となっている。実装上は各種テーブルや作業領域など多くの大域的記憶領域が使用され、それらが様々なプログラムから操作されている。処理状態遷移マトリックス上のひとつひとつの処理がどのデータを操作するのか整理し、データを中心にソフトウェア・アーキテクチャを見直し、オブジェクト指向的なアプローチを採ることで、再利用性や保守性に改善が図れる可能性もある。

6.4 再利用と生産性

生産性にどのような指標を用いるかによって再利用の評価や普及度に影響が出る。例えば、作成行数を生産性の指標に利用している場合、冗長なコードを再利用したときに生産性が高くなったかのような評価を下してしまう恐れがある。

さらに、再利用はしたものの、変更部分が多い場合にはたして再利用のメリットがあるのかどうか、判断が難しい。単純に再利用率のみで管理すると、生産性や品質の向上に継らない再利用も推進してしまう。

したがって、再利用しようとする元のコードの質について、十分な評価、検討が行われる必要があり、組織だった検定・登録[4]の必要性が推測される。

6.5 上流工程の再利用

上流工程の設計等の再利用については、明確な尺度も定義されておらず、定量的に述べることはできない。しかし、当プロジェクトにおいても少なからず再利用が行われていることが見てとれる。

1. ボード用専用OS
   端末本体との通信やタイマ関連の処理等、共通機能を提供している。また、この専用OS自体として、一種のソフトウェア・アーキテクチャを想定しているが、再利用率の計算時には、算入していない。
2. ドメインモデル
   状態遷移表および処理状態遷移表によるプロトコルの記述方法および記述したものを利用している。
3. プロトコル仕様書から分からないエラー処理の細部等、実現に関わる詳細な仕様について、既存のプログラムを動作させてその振舞いを調べている。

これらの事項については測定されておらず、組織的な奨励が行われているわけではない。定量的に捉えるのが困難な側面であるが、品質や生産性に大きな影響を与えている事項であると予想されるので、これらの効果を可視化し推奨する必要がある。

7 まとめ

本報告では、通信プロトコルの実装時に経験的に形成され、使用されているドメインモデルやソフトウェア・アーキテクチャについて述べ、このような開発基盤のある程度整った状況下における再利用の効果について調査した。プログラムを再利用した場合、単体試験工程までの効果はあるが、結合試験工程以降での効果は観測できなかった。

このプロジェクトで暗に用いているドメインモデルおよびソフトウェア・アーキテクチャは処理に重点をおいたものであり、データについては明示的に管理していない。通信マトリックス上の各処理がどのデータを操作するのか明確にし、影響の伝搬を抑制することができれば、再利用による効果を拡大できる可能性がある。しかし、本報告が示す事例は、過去の経験の積み重ねによって形成されたドメインモデルとソフトウェア・アーキテクチャを前提として、過去に作成されたほぼ同じ機能のプログラムを再利用した場合の効果について、一定の限界を示していると考えられる。

参考文献

[1] ARANGO, G. and PRIETO-DÍAZ, R. Domain Analysis Concepts and Research Di-

rections, Domain Analysis and Software Systems Modeling, IEEE (1991), 9–26.

[2] PERRY, D. E. and WOLF, A. L. Foundations for the Study of Software Architecture, *ACM SIGSOFT*, 17, 5 (October 1992).

[3] SHAW, M. Larger Scale Systems Require Higher-Level Abstractions, 5th Int. Workshop on Soft. Specification and Design (ACM SIGSOFT, Vol.14, No.3) (May 1989).

[4] 磯田定宏 ソフトウェア再利用の管理的側面, 情報処理学会論文誌, 34, 5 (May 1993).