2009－SE－163 (27)
2009／3／19

ソフトウェアプロジェクト診断のためのチェックリスト導出

出張 純也，水野 修，菊野 亨
大阪大学 大学院情報科学研究科 情報システム工学専攻
E-mail: {j-debari, o-mizuno, kikuno}@ist.osaka-u.ac.jp
菊地 奈穂美, 平山 雅之
独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター
E-mail: {n-kiku,m-hiraya}@ipa.go.jp

概要

ソフトウェア開発現場ではプロジェクト管理の重要性が益々高まってきている．本研究では，ソフトウェアプロジェクトの混乱を未然に予測して対策をとるための「ソフトウェアプロジェクト診断」を提案する．この診断のためには，ソフトウェア開発の問題点を指摘するチェックリストを作成する必要がある．従来，こうしたチェックリストは経験や知識をベースに作成されることが多かった．本研究ではIPA/SECによって収集された開発データに対して相関ルールマイニングとクラスタ分析を適用することによってチェックリストの生成を試みた．そして，生成されたチェックリストの妥当性を評価した．

On Deriving Checklists for a Software Project Checkup

Junya Debari, Osamu Mizuno, and Tohru Kikuno
Graduate School of Information Science and Technology, Osaka University
E-mail: {j-debari, o-mizuno, kikuno}@ist.osaka-u.ac.jp
Nahomi Kikuchi, Masayuki Hirayama
Software Engineering Center, Information-technology Promotion Agency
E-mail: {n-kiku,m-hiraya}@ipa.go.jp

Abstract

In software development project, necessity of project management has been increasing in recent years. It is thus required to discover runaway projects at an early stage. This paper proposes a project management framework "Software Project Checkup." This framework requires checklists indicating problems in a development project. Such checklists were usually based on the experience of project managers. In this study, we try to make such checklists automatically by applying association rule mining and clustering analysis to the project data collected by IPA/SEC. Then we evaluate the validity of the checklists.

1 はじめに

ソフトウェアプロジェクトの開発現場では，プロジェクトの混乱状態からの回避が急務となっている．しかし，開発プロジェクトの成功・失敗の要因は多岐にわたるため，その特定や特定した後の回避方法の検討は経験をベースに行われることが多く，しかも回避方法選択の判断は定性的なことが多い．一方で，現場で観察収集される様々なデータを有効に活用してプロジェクトを制御する手法の確立が求められている．

ソフトウェア開発プロジェクトにおけるリスク分類はBoehmらによって古くから行われてきており[2]，リスクを分析してプロジェクトの最終状態に関する予測を行う研究もある[3,6]．

我々の研究グループでは，ソフトウェア開発プロジェクトの早期に実施する問題分析アンケートによって，そのプロジェクトが最終的に混乱状態に陥るかどうかを判定する手法を提案してきた[1,4,5]．しかし，これらの研究に用いられてきたアンケートの項目は，開発者の経験をベースに作られたものであり，定量的なデータ分析によって得られたものではなかった．

本研究では，実際のプロジェクトのデータが多数存在する場合に，定量的なデータ分析によって有効なチェックリストを導出するための方法を議論し，実際に導出を行った．その結果，経験ベースのリストアップでは漏れてしまうようなチェック項目を導出できることが確認できた．

2 研究の目的

2.1 プロジェクト診断

本研究の目的は，ソフトウェアプロジェクト管理への新たな取り組みとしてソフトウェアプロジェクト診断を提案することである．提案する診断では，あるプロジェクトが健全であるかどうかを簡便な方法で定期的に診断する．なお，健全でないと診断されたプロジェクトについては改めて詳細な分析を行い，適切な改善を行う．図1は，提案するソフトウェアプロジェクト診断の概要を示している．

プロジェクトの健全度チェックとは，プロジェクトの初期に分かる情報や計画内容を調査・分析して，そのままプロジェクトがある程度進んだ時に品質・コスト・工期に関するトラブルが起きるかどうかを初期に予測するという事である．本研究では，プロジェクトの最終成果物の不具合数によってトラブル発生の可能性を判断する．

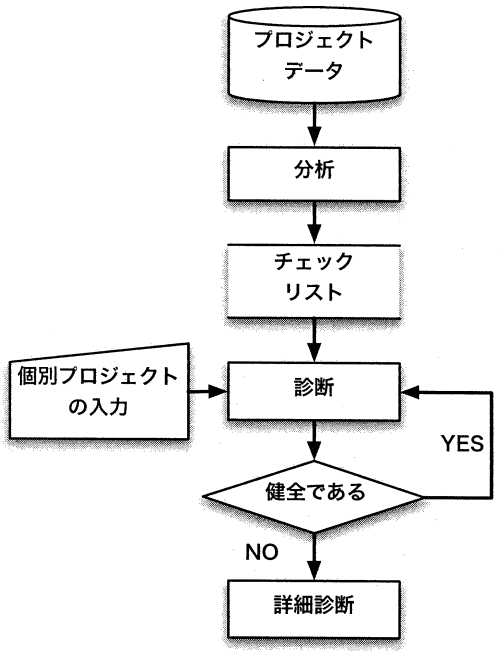

図1: プロジェクト診断の概要

2.2 チェックリスト導出

前節で示したソフトウェアプロジェクト診断において，定期的に行う簡便な方法での診断として，チェックリストを用意しその内容に応じて診断を行う方法を対象とする．

図2はチェックリスト導出手法の概要である．まず，過去に収集されたプロジェクトのデータから，失敗に終わっているプロジェクトの特徴を相関ルールマイニングを用いて抽出する．次に，抽出された相関ルール群に対してクラスタ分析を適用し，得られる樹形図を基に抽出された相関ルール群の特徴を解析する．得られた相関ルール群の特徴からチェックリストを考案する．

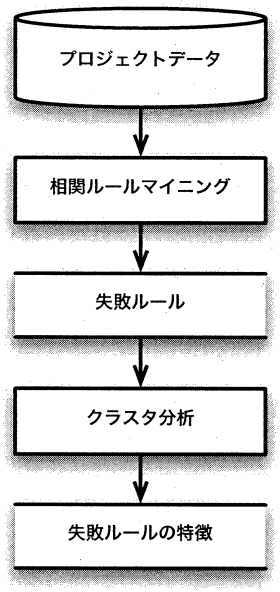

図 2: 導出手法の手順

3 準備

3.1 相関ルールマイニング

相関ルールマイニングは，事象間の強い関係を相関ルールの形で発見する，データマイニング手法の一つである．A ⇒ B という形で表される相関ルールは，A という事象が発生した場合に B という事象が発生することを意味しており，A を前提，B を結論と呼ぶ．

この相関ルールを評価するパラメータとして，支持度 (support) と信頼度 (confidence) がある．支持度は相関ルールの出現頻度を表しており，データ集合全体の中で A と B が同時に発生する確率 (p(A∩B)) である．信頼度はルールの前提と結論の結びつきの強さを表しており，事象 A が発生しているという条件下で事象 B が発生する確率 (p(B|A)) である．相関ルールを抽出する際には，これらのパラメータに最低値を設定し，その条件を満たすルールのみを抽出する．

相関ルールマイニングを用いることで，単一の要因が結果に与える影響だけでなく，要因間の関係も考慮に入れた「ルール」の抽出が可能となる．

本研究では，相関ルールの前提，結論などの事象を全てメトリクス m_i と，そのメトリクスの観測値 v_j の組み合わせ $\{m_i = v_j\}$ によって表現する．この組み合わせをトランザクションと呼ぶ．前提は 1 個以上のトランザクションによって構成され，結論は 1 個のトランザクションによって構成される．

本研究では，プロジェクトの最終成果物の不具合数によってトラブルの発生の判断を行うため，相関ルールの結論は “不具合数 = 多い” 及び “不具合数 = 少ない” の 2 種類に限定する．なお，最終成果物の不具合数は中央値で分割して多いと少ないの 2 値にしている．より具体的には，不具合数が 0 または 1 ならば「少ない」，2 以上であれば「多い」となる．

3.2 クラスタ分析

クラスタ分析とは，個体間の距離を基に類似度の高い個体同士を集めてクラスタを作り，対象である個体を分類するデータ分析手法である [7,8]．個体間の距離としてはユークリッド距離や Jaccard 係数などが利用される．

クラスタ分析には階層的手法と非階層的手法が存在するが，本研究では階層的手法を用いている．階層的クラスタ分析は，n 個の対象からなるデータが与えられた時に，初期状態として 1 個の対象だ

けを含む n 個のクラスタを作成する．この状態から，クラスタ C_1 と C_2 の距離 $D(C_1, C_2)$ を計算し，最も距離の近い二つのクラスタを逐次的に併合する．この併合を，全ての対象が一つのクラスタになるまで繰り返す．この結果得られる階層構造が樹形図の形で得られる．

クラスタ間の距離の求め方には，最小距離法や最大距離法などがあるが，本研究では最も分類感度が高い方法の一つであるウォード法を用いている [8]．ウォード法は $D(C_1, C_2)$ を次のように設定する．

$$D(C_1, C_2) = E(C_1 \cup C_2) - E(C_1) - E(C_2)$$

ただし，$E(C)$ はクラスタ C の全ての要素とクラスタ C のセントロイドの距離の二乗の総和である．

本研究では，クラスタ分析を行う対象が相関ルールであるため，相関ルール r_1 と r_2 の距離 $d(r_1, r_2)$ を次のように定義する．

$$d(r_1, r_2) = \frac{|T(r_1) \cap T(r_2)|}{|T(r_1) \cup T(r_2)|}$$

ただし，$T(r_1)$ は相関ルール r_1 に含まれる全てのトランザクションの集合である．

3.3 利用データ

本研究で利用したデータは，IPA/SEC が収集したものである [9]．このデータは国内の企業 19 社から収集されたもので，汎用計算機上で動作するアプリケーションソフトウェアを開発するプロジェクトのものである．収集されたプロジェクトの形態は 9 割以上が受託開発である．それぞれのプロジェクトのデータには，規模・工数・工期・不具合数などの量的なプロジェクトデータ項目に加えて，プロジェクト特性を示す質的データが含まれている．

収集されたメトリクスの詳細については文献 [9] に詳しいため，ここでは省略する．プロジェクト数は 1,419 件であり，2005 年 6 月頃までに終了したものとなっている．

表 1 は分析に利用したメトリクスの一覧である．月当たりの工数，月当たりの FP 値，工数当たりの FP 値，工数当たりの SLOC 値は，文献 [9] に記載されているデータを加工して設定したものである．工数は実績工数 (開発 5 工程)，工期は実績月数_プロジェクト全体，SLOC 値は実効 SLOC 実績値，FP 値は FP 実績値 (調整前) を利用した．本研究では，簡便にソフトウェアプロジェクトの状態を診断する手法の開発を目的としているため，利用する全てのメトリクスは協議の上で 2 値に分割している[1]．

4 チェックリスト導出

4.1 手法適用の結果

まず，ソフトウェアプロジェクトデータに対して相関ルールマイニングを行った．その結果，不具合が多くなる状況を示す相関ルール 2,199 件を得た．この相関ルール群に対してクラスタ分析を行い，図 3 に示す概形の樹形図を得た．

各クラスタには様々なルールが含まれるが，クラスタ内のルールに 90%以上の頻度で出現しているトランザクションをクラスタの支配的要因と見なした，そして各クラスタに支配的要因が見つかる高さで樹形図を切断した．この結果，相関ルールは C_1〜C_{10} の 10 のクラスタに分類された．表 2 は各クラスタの支配的要因である．

4.2 支配的要因の評価

まず，抽出された各クラスタの支配的要因が妥当であるか否かについての議論を行った．その結果，クラスタ C_9 の支配的要因の一つである”開発の種別 = アプリケーションソフト”に関しては，分析対象としたデータのほぼ全てがアプリケーションソ

[1] 多くのメトリクスでは中央値を利用して 2 値に分割している．A,B,C,D のような順序データは A と B,C,D のように分割したものが多い．

表 1: 本研究で利用したメトリクス一覧

開発プロジェクトの種別	母体システムの安定度
開発プロジェクトの形態	受託開発の場合の作業場所
新規の顧客か否か	新規の業種・業務か否か
新規協力会社か否か	新技術を利用する開発か否か
開発プロジェクトチーム内での役割分担・責任所在の明確さ	達成目標と優先度の明確さ
作業スペース	プロジェクト環境 (騒音)
計画の評価 (コスト)	計画の評価 (品質)
計画の評価 (工期)	業種
利用形態	利用者数
利用拠点数	同時最大利用ユーザ数
システムの種別	業務パッケージ利用の有無
処理形態	アーキテクチャ
Web 技術の利用	オンライントランザクション処理
DBMS の利用	運用ツールの利用
類似プロジェクトの参照の有無	プロジェクト管理ツールの利用
厚生管理ツールの利用	設計支援ツールの利用
ドキュメント作成ツールの利用	デバッグ・テストツールの利用
CASE ツールの利用	コードジェネレータの利用
開発方法論の利用	開発フレームワークの利用
要求仕様の明確さ	ユーザ担当者の要求仕様関与
ユーザ担当者のシステム経験	ユーザ担当者の業務経験
ユーザとの役割分担・責任所在の明確さ	要求仕様に対するユーザ承認の有無
ユーザ担当者の受け入れ試験関与	要件決定者の人数
要求レベル (信頼性)	要求レベル (使用性)
要求レベル (性能・効率性)	要求レベル (保守性)
要求レベル (移植性)	要求レベル (ランニングコスト要求)
要求レベル (セキュリティ)	法的規制の有無
PM スキル	開発要員スキル_業務分野の経験
開発要員スキル_分析・設計経験	開発要員スキル_言語・ツール利用経験
開発要員スキル_開発プラットフォームの使用経験	平均要員数プロジェクト全体
ピーク要因数プロジェクト全体	月あたりの工数
月あたりの FP 値	工数あたりの FP 値
工数あたりの SLOC 値	不具合数

フトの開発であったことから，チェックリストとしては意味が無いという結論に至った．

次に，クラスタ C_1〜C_8，C_{10} の支配的要因に関して，これらの要因がソフトウェアの不具合数に影響を与えているかどうかについて，統計的検定を用いて確認した．表 3 は C_1 の支配的要因である”母体システムの安定度”と，不具合数についてのクロス表である．この表をもとに母体システムの安定度と不具合数の関連についてカイ二乗検定を用いて検定した結果，有意差が認められた ($p < 0.05$)．同様に，C_2〜C_8，C_{10} の支配的要因と不具合数に関連があるかどうかをカイ二乗検定で検定した結果が表 4 である．

その結果，クラスタ C_1，C_3，C_5，C_7 の支配的要因はプロジェクトの最終成果物の不具合数と 5%有意で関連があることがわかった．また，クラスタ C_4 と C_8 の支配的要因に関しては，20%の有意水準であっても強い関連があるとは言えないことがわかった．

4.3 得られた要因の解釈

分析の結果得られた各クラスタの支配的要因からチェックリストを得るために，強い関連が見られなかった C_4 と C_8，そもそも意味がないと判断された C_9 以外の要因について解釈を行った．

C_1: 母体システムの安定度 = 安定

母体システムの安定度は，既存のシステムを拡張する場合に収集されるメトリクスであり，拡張を行う対象のシステムの安定度を表している．

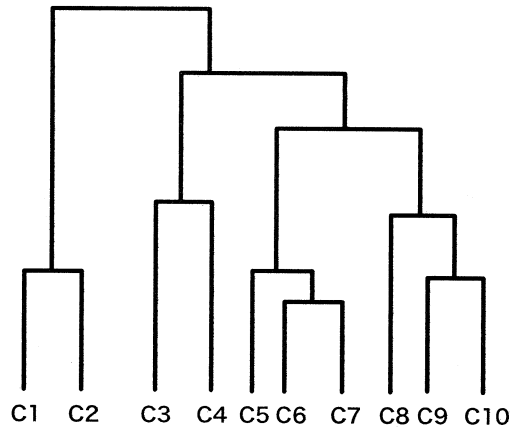

図 3: 樹形図の概形

一般的には母体となるシステムが安定していれば拡張したシステムも安定すると考えられている．この結果からは，既存ソフト資産をベースにした開発では，もとの構造によっては不具合が入り込む可能性が高くなると考えることができる．

C_2: 構成管理ツールの利用 = 無し

構成管理ツールは，システムの開発から運用終了までを通しての，ハードウェア，ソフトウェア，文書化，テストなどによる変更を考慮した機能の一貫性を保証するためのツールであり，バージョン管理ツールなども含まれている．構成管理ツールを利用しい場合は，開発が混乱して不具合が入り込む可能性が高くなると予想される．

C_3: ドキュメント作成ツールの利用 = あり，デバッグ・テストツールの利用= 無し

ドキュメント作成ツールを利用しなければならないプロジェクトは開発規模が大きいことが予想される．開発規模が大きいプロジェクトにおいて，デバッグ・テストツールを利用していない場合は，不具合が入り込む可能性が高くなる．

表 2: 各クラスタの支配的要因

クラスタ	支配的要因
C_1	母体システムの安定度 = 安定
C_2	構成管理ツールの利用 = 無し
C_3	ドキュメント作成ツールの利用 = あり デバッグ・テストツールの利用 = 無し
C_4	要求仕様の明確さ = 曖昧
C_5	ドキュメント作成ツールの利用 = あり ユーザ担当者の要求仕様関与度合 = 高い
C_6	工数あたりの FP = 低い 開発プロジェクトチーム内での 役割分担・責任所在の明確さ = 不明確
C_7	工数あたりの FP = 低い プロジェクト管理ツールの利用 = 無し
C_8	要求レベル (性能・効率性) = 低い
C_9	月あたりの FP = 高い 開発の種別 = アプリケーションソフト
C_{10}	月あたりの FP = 高い 母体システムの安定度 = 安定

表 3: C_1 の支配的要因に関する分析

		母体システムの安定度		
		不安定	安定	合計
不具合数	少ない	18	13	31
	多い	17	36	53
	合計	35	49	84

C_5: ドキュメント作成ツールの利用 = あり，ユーザ担当者の要求仕様関与度合 = 高い

一般的に，要求仕様を作成する際にユーザ担当者と密に連絡を取り，厳密な要求仕様を作成する事は品質向上に繋がると考えられている．しかし，この結果は要求仕様を記述する際にユーザの関与が高すぎる場合は不具合数が増加する可能性が高いことを示唆している．その原因としては，要求仕様の作成に当たりユーザが過度な要求をした結果，要求仕様の中に不正確であったり非機能的な要求事項が含まれたりすることが考えられる．

C_6: 工数あたりの FP 値 = 低い，開発プロジェクトチーム内での役割分担・責任所在 = 不明確

工数あたりの FP 値が低いということは，1 つの機能を実装するのに必要であった工数が高いとい

表 4: 各クラスタの支配的要因の p 値

クラスタ	p 値
C_1	0.023
C_2	0.107
C_3	0.018
C_4	0.221
C_5	0.026
C_6	0.091
C_7	0.022
C_8	0.504
C_{10}	0.155

うことであり，開発の難易度が高い事を示している．こうした状況で設計，実装，テストなどの役割分担や責任の所在が不明確であれば，チーム間の情報交換ミスなどが発生し，不具合を作り込む可能性が上がると考えられる．

C_7: 工数あたりの FP 値 = 低い，プロジェクト管理ツールの利用 = 無し

プロジェクト管理ツールは，開発の進捗状況の把握などによってプロジェクトの管理をサポートするためのツールである．開発の難易度が高いプロジェクトにおいて，プロジェクト管理ツールを利用するなどの適切なプロジェクト管理が実施されていない場合，プロジェクトの混乱が発生して不具合を見逃す可能性が高くなると考えられる．

C_{10}: 月あたりの FP 値 = 高い，母体システムの安定度 = 安定

月あたりの FP 値が高いということは，そのプロジェクトの開発の速度が速いということを示している．拡張対象であるシステムを十分に吟味せずに機能追加などを急ぐと不具合が入り込む可能性が高くなると考えられる．

4.4 チェックリスト

4 節で得られた要因とその解釈から，ソフトウェアプロジェクトの健全度を診断するためのチェックリストを作成する．チェックリストの作成は，プロジェクトの早期の段階にチェックが行われることを想定して行った．

抽出されたクラスタの中で，例えば C_1 は経験ベースのリストアップでは漏れてしまう可能性を含んでいる．母体システムが安定していれば不具合は少なくなると考えがちであり，そうした常識を否定する要因が得られたことは，本手法の狙いが達成されていることを示す．

表 5 は得られたチェックリストの一覧である．なお，前述のクラスタ C_1 はチェックリストでは「1. 既存ソフト資産の利用を前提としている」として項目化されている．

5 まとめ

本研究では，ソフトウェアプロジェクト管理の新しい方法としてソフトウェアプロジェクト診断を提案した．さらに，ソフトウェア開発プロジェクトデータから，あるプロジェクトが健全であるか否かを判断するためのチェックリストを導出する方法について検討を行い，プロジェクトの最終成果物の不具合数を不健全さの指標としてチェックリストの導出を行った．導出されたチェック項目には，直感と反する内容の項目も含まれていた．また，チェックリストの妥当性を評価するために，統計的検定を行った．

今後の課題としては，コストや工期の面からのチェックリスト導出や，チェックリストによって健全か否かを判断するための基準の考察などが挙げられる．

表 5: 分析の結果得られたチェックリスト

No.	チェックする内容
1	既存ソフト資産の利用を前提としている
2	構成管理ツールを利用していない
3	プロジェクト規模が大きい開発においてデバッグ・テストツールを利用していない
4	プロジェクト規模が大きい開発において，ユーザが過度の要求仕様を求めている
5	開発の難易度が高いにもかかわらず，役割分担が不明確である
6	開発の難易度が高いにもかかわらず．プロジェクト管理ツールを利用していない
7	既存ソフト資産を再利用し，開発ピッチを上げようとしている

参考文献

[1] Amasaki, S., Hamano, Y., Mizuno, O. and Kikuno, T.: Characterization of Runaway Software Projects Using Association Rule Mining, *Proc. of 7th International Conference on Product Focused Software Process Improvement (PROFES2006)*, pp. 402–407 (2006).

[2] Boehm, B. W.: Industrial software metrics top 10 list, *IEEE Software*, Vol. 4, No. 5, pp. 84–85 (1987).

[3] Jiang, J. and Klein, G.: Software development risks to project effectiveness, *Journal of Systems and Software*, Vol. 52, pp. 3–10 (2000).

[4] Mizuno, O., Kikuno, T., Takagi, Y. and Sakamoto, K.: Characterization of risky projects based on project managers' evaluation, *Proc. of 22nd International Conference on Software Engineering*, pp. 387–395 (2000).

[5] Takagi, Y., Mizuno, O. and Kikuno, T.: An Empirical Approach to Characterizing Risky Software Projects Based on Logistic Regression Analysis, *Empirical Software Engineering*, Vol. 10, No. 4, pp. 495–515 (2005).

[6] Wohlin, C. and Andrews, A. A.: Prioritizing and Assessing Software Project Success Factors and Project Characteristics using Subjective Data, *Empirical Software Engineering*, Vol. 8, pp. 285–303 (2003).

[7] 奥野忠一, 芳賀敏郎, 矢島敬二, 奥野千恵子, 橋本茂司, 古河陽子, 続多変量解析, 日科技連出版社 (1976).

[8] 古谷野亘, 多変量解析ガイド, 川島書店 (1988).

[9] (独) 情報処理推進機構 ソフトウェア・エンジニアリング・センター：ソフトウェア開発データ白書 2006, 日経 BP 社 (2006).