OSS 事前評価による開発リスク特定の取組み

岩崎 孝司　高山 修一　岩永 裕史
富士通九州ネットワークテクノロジーズ㈱
Iwasaki.takashi@jp.fujitsu.com
takayama_@jp.fujitsu.com
iwanaga_hiroshi@jp.fujitsu.com

鵜林 尚靖　亀井 靖高
九州大学
ubayashi@ait.kyushu-u.ac.jp
kamei@ait.kyushu-u.ac.jp

要旨

昨今のソフトウェア開発において、OSS を利用した開発が徐々に増えてきている。また、利用した OSS に起因する問題が開発後半で発生し、開発コスト増や納期遅延といった問題が発生することも珍しくない。OSS を利用する前に OSS の利用リスクを判断する事前評価の技術や開発プロセスは既存の研究でも実施されてきたが、開発現場で一般的に活用されるには至っていない。本研究では開発現場への導入を主眼に置き、開発者が OSS を利用する際に意識している品質要求から OSS の利用リスクを判断できる OSS 事前評価手法を考案した。本手法は、品質要求と OSS を事前に評価した事前評価指標を関連付け、開発者にリスク評価結果を示す事前評価レポートを作成する手法とした。考案した手法を、当社内で実際に開発を行ったプロジェクトに対して適用し、リスクの事前評価結果と実際の開発で発生した OSS 利用時の問題を比較した。結果、今回試行の対象としたプロジェクトで実際に発生した問題に対しては、本手法を用いてリスクを検知できる事を確認した。

1. 背景

(1) 企業での OSS 利用状況

近年のソフトウェア開発において、OSS(Open Source Software)を利用した開発が一般化しつつある。全国のソフトウェア関連企業の 50%を超える企業で OSS を利用した開発が行われている[1]。WEB系システム構築や企業向けの情報システムの様な一般的な情報システムの構築に加えて、通信ネットワークシステムやIoT（Internet of Things）システムの様な組込みシステムの開発においても、OSS を用いた開発が急増している。富士通九州ネットワークテクノロジーズ株式会社（以降、富士通 QNET）社内の場合でも年々OSS の利用率が上昇し、最近では約 50%のプロジェクトで OSS を利用した開発が行われている[11]。

OSS を利用したプロジェクトの問題として、利用した OSS の問題が開発工程後半で発覚し、納期遅延や開発コスト増大等のプロジェクト全体の問題になってしまう様な事態も発生しており、開発初期のリスク管理が重要視されている。

その様な OSS 利用時の問題に対応する技術として、開発開始前や開発工程前半で OSS 利用時のリスクを判断する OSS 事前評価手法[2-7]がある。OSS 事前評価手法は企業向け情報サービスやITソリューションを提供する企業では普及しつつあり、世の中では事前評価を含むコンサルタント業務をビジネスとして行う会社もある。一方、富士通 QNET を含む設計開発会社では事前評価手法は存在するものの十分に普及、運用されていない状況である。本研究では、OSS 事前評価指標を開発現場に浸透させるため、普及、運用を考慮した OSS 事前評価手法を考案した。普及、運用の課題については第二章で詳細を述べる.

(2) OSS 研究の状況

OSS を実際に利用する前に評価する事前評価技術の研究は2010年までに多く取り組まれ、その有効性が評価されてきた[2-7]。同時に事前評価を正しく実施するための開発プロセスの研究やツール開発も積極的に取り組みが行われた[8-10]。 現在ではソフトウェア工学の1つの独立した研究分野として「オープンソフトウェア工学」が提唱されソースコードの評価やプロセスに留まらず、OSS コミュニティの人的リソースの相互関係やビジネスモデルの研究が行われており、事前評価技術を含む、多面的な観点での研究となっている[13]。

(3) 論文の構成

以降、本論文の第二章では OSS の事前評価の課題を、開発現場での普及、運用の観点を含めて述べる。第三章では開発現場への普及を目的に、本手法を開発者が持つ品質要求を元に OSS 事前評価を行う手法とした事を述べる。第四章では、事前評価指標を品質要求に結び付けたやり方について、本手法での実現方法を述べる。第五章では本手法の全体像を実施手順、スコアリングルール、事前評価の結果作成されるレポートの例で紹介する。第六章では本手法を既存の開発プロジェクトを元に検証した結果、手法により、抽出したリスクがプロジェクトで発生した問題に対応しており、評価手法が有

効であったことを述べる。第七、八章にて、本研究のまとめと今後の課題を述べる。

2. OSS 事前評価手法の課題

OSS を利用する前に OSS の利用リスクを予測する OSS 事前評価手法は、OSS 利用の拡大とともに重要視されてきている。OSS 事前評価技術やプロセスの適用は、一部では普及しているが、組込み開発を中心として、設計開発を実施する会社では、必ずしも普及しておらず組織的な運用に至っていないのが現状である。

本研究で、現在の評価方法の問題を確認するため、OSS 事前評価を実施する既存の評価モデルである、QSOS[2]、OpenBRR[8]、RepOSS[4]、 OMM[10] の調査を実施した結果、「用語がおおまかで曖昧」,「スコアリングルールが曖昧な項目が半数」,「公開されたリポジトリがない」といった課題を認識した[11]。また、開発現場に事前評価手法が普及しない理由を以下の様に考察した。

・開発者は、提示された事前評価指標を1つ1つ解釈し、開発者自身が持つ品質要求と結び付けてリスク分析をしなければならない。そのため、リスク評価が開発者の能力に依存する。

・評価方法が曖昧で理解しづらい。例えば、既存研究では「ユーザビリティ」(使用性)という項目があるが、お客様の使い勝手を指すのか、OSS 自体の使い勝手を指すのか、一見して分かりづらい。

・事前評価指標が多種であり、全評価指標を調査し、検討するにはコストがかかり過ぎる。

・開発する製品の要求性能や要求品質により OSS への要求性能や要求品質が変わるため、事前評価が一意には評価できない

これらの課題から、OSS 事前評価手法開発の課題を以下とした。

(1) 開発現場が理解しやすい手法の確立

OSS 事前評価時の観点を開発者側から見た品質要求で定義して、開発者が理解しやすい評価手法を開発する事にした。例えば、「ユーザビリティ」は、開発者からみた OSS の使い勝手と、システムの利用者から見た使い勝手の品質要求に分けて定義し、開発者が理解しやすい評価手法とした。詳細は表 2 を参照、ユーザビリティの例は使用性に対応している。

また設計開発の場合は、開発者の利用する OSS に対する品質要求は開発するシステムや製品によって異なるため、品質要求のレベルも異なってくる。これらの異なる品質要求レベルに対応できる手法とする。

表1. 品質要求カテゴリと品質要求

カテゴリ	品質要求
機能適合性	利用するOSSの機能に関する品質要求
性能効率性	利用するOSSの性能に関する品質要求
使用性	利用するOSSのインストールやシステムへの組み込み等の利用容易性に関する品質要求
信頼性	利用するOSSの開発コミュニティやOSS自体の保守に関わる品質要求
セキュリティ	利用するOSSのセキュリティに関する品質要求
保守性	利用するOSSのプログラム構造や可読性に関する品質要求
移植性、互換性	利用するOSSを他システムや他OSで利用した場合を想定した移植性、互換性に関する品質要求

(2) 事前評価作業のコストの明確化と削減

OSS を利用する開発者は開発コスト削減や開発スピード向上を目的としており、事前評価作業や指標取得を実施する事に大きな負担を感じている。開発者の負担を軽減するためにも、事前評価作業におけるコストの明確化を図る共に、不要な作業を除去して、より小さなコストで評価を実施できる手法とする。

(3) 個人のスキルに依存しない事前評価作業の実現

従来研究の評価モデルでは、事前評価指標と OSS の品質要求を、開発者が自身の知識やスキルを元に対応させ評価する必要があった。本手法は開発者の知識やスキルに依存せず、一定の精度で利用する OSS のリスクを予測できる手法とする。

3. OSS 利用者の品質要求について

本手法を、開発者の品質要求から OSS の利用リスクを判断できる手法とするため、開発者の OSS 品質要求としてどの様な要求事項があるかを抽出した。

品質要求を抽出する際の観点としては、システム・ソフトウェア製品品質(JIS X 25010:2013)の品質要求のカテゴリを参考として、表 1 の7つの分類を採用し、OSS 利用の観点から各カテゴリに対応する品質要求を定義した。

具体的な品質要求の抽出に当たっては、研究チームの企業と大学側で、役割分担してまずは素案を作成した。企業側からは、OSS を利用した開発を行っている開発者(開発経験 10 年前後)複数名で実際の開発現場での品質要求をカテゴリ毎に抽出した。大学側は、先例の研究により明確になっている事前評価指標(表3参照)[11]から類推される品質要求を抽出して品質要求の素案とした。

次に、作成した品質要求の素案を元に当社の開発プロジェクトのメンバにヒアリングを実施した。ヒアリングの範囲としては、8プロジェクトの開発リーダ、開発担当の技術者を対象とした。ヒアリングにより追加された品質要求は 25 項目挙がったが、各カテゴリにより、項目数のばらつき

表2. カテゴリ毎の品質要求項目

カテゴリ	品質要求
機能適合性	OSS自身が実現したい機能を満たしている
	OSSのリリースが迅速で早期に機能を利用できる
	正常系が正しく動作しており，早期に機能を確認できる
性能	OSS自身が想定されるリソースで動作する
	OSSが指定された性能を満たす事が出来る
	性能測定のための環境作りが容易にできる
互換性・移植性	OSSが他のOSSやシステム内の機能と併存して利用可能
	複数種別のＯＳやハードに対応している。
	他の代替可能なOSSがある
使用性	OSS自体の操作性が利用者の要求を満たしている
	OSS自体にドキュメントや書籍が揃っていて，十分な情報が得られる
	OSSのインストールや組み込みが比較的容易にできる。
	OSSの利用例が多い
信頼性	サポートするコミュニティがしっかりしていて，保守されている
	OSSのバグ情報や対応状況が把握できる
	OSS自体の品質が良い
セキュリティ	脆弱性に関する報告が少ない
	信頼できるサイトで公開されている
保守性	OSSのプログラム構造が容易でホワイトボックス化しやすい
	ソースコード中のコメントが豊富で読解性が良い

があり、1 項目しか品質要求が挙がらなかったカテゴリもあれば、7 項目以上品質要求が挙がったカテゴリもあった。項目数を極力そろえるため、類似項目の統合や有識者による再検討を経て、各カテゴリの品質要求項目を 2～4 項目に見直し整理した。整理後の品質要求を表 2 に示す。

抽出した品質要求について当社において過去1年間で OSS を利用した開発リーダ（述べ 50 名）に対して、品質要求の重視度のアンケート調査を実施した。アンケートは、各品質要求に対しての重視度を、5:重視する 4:やや重視する 2:やや重視しない 1:重視しないの4段階で開発リーダに選択させた。

アンケート結果を図1に示す。アンケート時点の品質項目数は 23 件（後の整理統合により、最終的には表 2 に示す 19 件となった）であり、23 件中、ほとんどの項目は 3 点以上の評価結果となり、半分以上の項目は 4 点以上（重視する、やや重視する）という結果になった。

これらの結果により、定義した品質要求は、開発現場での品質要求を適切に捉えたものであると判断した。

一方、平均値として相対的に低い値となった品質要求

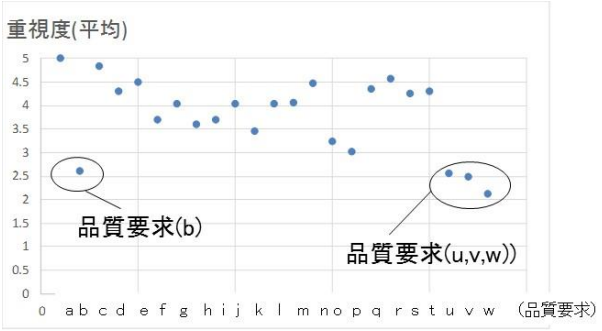

図1 品質要求のアンケート結果

が 4 項目存在した。それぞれ「OSS のリリースが迅速で早期に機能を利用できる」（品質要求 b）、「OSS のプログラム構造が容易でホワイトボックス化しやすい」（品質要求 u）、「ソースコード中のコメントが豊富で読解性が良い」（品質要求 v),「他の代替可能な OSS がある」(品質要求 w)であった。

品質要求 b については、既に OSS を開発母体に使用しており、早期の機能提供より OSS も安定を重要視するプロジェクトで低く評価された。当社の開発の多くはこの様な派生開発であることが平均値を低くした要因として挙げられる。ただし、先進分野(IoT 等）の新規開発においては、技術の進化速度が速く、リリースの迅速性を重視する開発も存在するため、品質要求として有益であると判断した。

品質要求 u,v については、OSS の利用方法によりプロジェクトの回答が分かれており、OSS をブラックボックスで利用するプロジェクトは重視度低く評価する傾向があり、改造して利用するプロジェクトは重視度を高く評価する傾向があった。OSS を改造して利用するプロジェクトが存在する事に加え、元々ブラックボックスで利用を計画していたプロジェクトでも結果的にコード解析や改造が必要になることを鑑み、品質要求としては有益であると判断した。

品質要求 w については、そもそも OSS の代替が念頭にないプロジェクトが多く存在した事が原因と考えられる。これは、派生開発の特徴に加え OSS の選択を顧客からの指定で決定されるプロジェクトが、品質要求の重視度を低く評価した事が原因であった。ただし、複数の OSS 候補に対して可能性を配慮することは、選択した OSS に大きなリスクが発生した場合には不可欠な品質要求となりえるため、開発者へ啓蒙の意味があると判断してそのまま採用した。

表3　RepOSS による事前評価指標の分類

OSS事前評価指標の分類
ライセンス情報
開発組織情報（コミュニティ、企業、主たる開発者、開発者の分布）
提供情報（メジャーリリース、メジャーリリース提供間隔）
コミュニケーション方法（オフィシャルサイト、ソースコードリポジトリ、バグトラッキングシステム、メーリングリスト）
仕様関係情報（標準準拠、ローカライゼーション）
構成（開発言語、プラグイン構造、GUIツール）
品質（バグ数、バグ修正率、平均バグ修正時間）
普及度（ダウンロード数、ユーザマニュアル、Webサイト数、書籍数、ユーザグループ、賞）
ポータビリティ（ディストリビュータパッケージ情報、インストーラ情報）
ビジネス（他ソフトでの利用、ベンダーによるサポートサービス）
事例

表3は日本OSS推進フォーラムプレスリリースを出展に作成　http://ossforum.jp/node/126

4. 品質要求項目と事前評価指標の対応

抽出した品質要求項目と、品質要求を評価可能な事前評価指標の対応付けを行った。対応付けの母集団として採用した事前評価指標は、過去の研究で提案された事前評価指標とソースコードの静的解析ツールで計測したソースコードメトリックス値とした。今回の研究ではコード解析ツールとし TechMatrix 社の Understand を採用した。

過去の研究で提案された事前評価指標は、日本OSS推進フォーラムが提供するオープンソースソフトウェア評価情報リポジトリ（RepOSS）の事前評価指標を採用した。これは他の研究組織の提唱する事前評価指標と比較し多種であり、系統立てて事前評価指標が定義されていたためであった。

加えて、Understand を用いて、オープンソースコードのソフトウェアメトリックスを計測した。Understand は、ソースコードを解析し、100 種類に及ぶソフトウェアメトリクスを計測可能である。ソフトウェアメトリックスには、複雑度を表すサイクロマチック複雑度や、結合性を表わす LCOM(結合性の欠如)、CBO（結合されクラスの数）、規模を表す LOC(コード行数)などが含まれる[11]。

これらの事前評価指標と抽出した品質要求の関連付けを実施した。表 4 に対応付けを行った品質評価指標の例と品質要求のカテゴリを示す。

表4. 品質要求カテゴリと採用した事前評価指標

品質特性	採用した事前評価指標（例）
機能適合性	書籍数・サイト数 マニュアル システムの開発事例 開発主体
性能（効率）性	動作環境:OS 動作環境：必要スペック 書籍数・サイト数
使用性	マニュアル パッケージインストーラ 書籍数・サイト数 他ソフトウェアへの適用
信頼性	開発主体 バグトラッキングシステムの有無 メーリングリストの流量 バグ数・フィックス率 Cyclomatic複雑度
セキュリティ	CVEEスコア 公開サイト
保守性	Cyclomatic複雑度 ネスト数 コード行に占めるコメント行の割合
互換性・移植性	ライセンス バージョンの互換性 マルチOS等への対応

特に、本研究の先行調査で、書籍数、Web サイト数、MaxNesting（ネスト数）、複雑度に関するソフトウェアメトリックス(5 種)の事前評価指標は OSS の品質と相関関係が深い事が確認されており[11]、該当する品質カテゴリの評価に結び付けることにした。

複雑度については、 5 種類の指標の相関関係がほぼ同一であったため、代表的な指標である AvgCyclomatic のみを採用した。

先行研究で研究された事前評価指標(RepOSS)とソフトウェアメトリックス(Understand)の事前評価指標を品質要求と対応付した結果、使用性の中にある書籍やドキュメントに関する品質要求や、信頼性にある OSS 自体のバグに関する情報やコミュニティの活動状況に保守性に関するリスク評価は、RepOSS の事前評価指標やメトリックス値と対応付ける事ができた。

しかし、品質要求によっては、品質要求を満たせる事前評価指標が存在しないものも発生した。性能や使用性のカテゴリに、そのままでは、事前評価指標が対応できない品質要求が存在した。

性能では、「OSS が指定された性能を満たす事ができる」や「性能測定のための環境作りが容易にできる」の品質要求がこれに相当し、使用性では「正常系が正しく動作しており，早期に機能を確認できる」が相当する。本来、対応する指標がないという事はリスクを判断する材料がなく、リスク有との判断をしてリスク管理の対象とするやり方が一般的である。今回の手法では極力、事前評価指標を品質要求に結び付ける事にした。例えば、「正常系が正しく動作しており，早期に機能を確認できる」について

は、多くのユーザに利用される OSS であれば、正常系動作が保証されている可能性は高いと考え、ダウンロード数や世の中の利用実績を関連指標として対応付けた。

しかしながら、ダウンロード数や世の中の利用実績が示す結果が良好であっても、品質要求が満たされるとは限らないため、結び付けた品質要求に対する事前評価指標の関連度を設定した。

品質要求に評価指標が直接関連する場合は関連度大(5点)とし、直接的に関連しないが、品質要求に何らかの関連があるものは、関連度を関連中(3 点)、関連小(1 点)として対応付けを実施した。

性能については、既存の事前評価指標には対応する評価指標は見つからなかった。例えば、他の導入事例等で性能に関する情報があっても、システムが動作する環境が違っていたり、性能評価のデータ自体が大変古いものであったりと有効に利用できないものが多かった。そこで、社内の利用実績やノウハウといった社内情報を評価指標に組み入れる事にした。これによって、実績の有無、件数に加えて、社内情報ならではの OSS を利用したシステムの特性や動作環境等の情報が把握でき、定性的に性能のリスクを検討できる可能性がある考えた。

最終的に採用した評価指標を表 5 に示す。以上の様に、当初 200 個強あった事前評価指標を今回抽出した品質要求に関連するもののみに絞り込んで、事前評価の仕組みに盛り込んだ。

5. OSS 事前評価手法について

本研究の OSS 事前評価手法は、開発者が利用する OSS への品質要件により変動する開発リスクを予め判断し、開発計画に反映するための手法である。従来手法と比べて、以下の特徴を持つものとした。

・開発者が OSS 利用時に意識している品質要求毎にリスク予測が可能となり、開発者の視点に近いリスク特定が可能となる。

・開発者が特別な知識やトレーニングなしに、一定のリスクを予測可能となる。

・開発者が複数の品質要求側面から網羅的に品質リスクを判断できる様になる。

表 5 採用した事前評価指標

事前評価指標	採用した評価指標
OSS評価指標 (REPOSS)	31種類 (詳細は表4参照)
ソフトウェアメトリックス	5種類 (Cyclomatic複雑度、コード行数、ネスト数 ファイル数、コメント行数、コメント行の割合)
社内利用実績	3種類 (社内の利用実績、トラブル報告数、ノウハウ数)

5.1. 手法の実施手順

OSS を利用して開発を行うプロジェクトでは、開発開始前に以下の手順で OSS の事前評価を行う。

(1) 品質要求アンケートの作成

OSS への品質要求の重視度を開発者の判断で記載する。重視度としては、重視する。やや重視する。やや重視しない。重視しない。の 4 段階の評価とした。

(2) 事前評価指標の算出

研究に先駆け当社で過去利用した OSS 約 120 種について、大学側で一般的な OSS 事前評価指標(RepOSS)とソフトウェアメトリックスについて、算出を行い、データベース化を実施した。

評価したい OSS の事前評価データがデータベース中にあれば、算出作業は不要であるが、データベース中になければ、新たに調査を実施して事前評価データを算出する。

(3) スコアリング

品質要求毎に予め事前評価指標を対応付けており品質要求毎のスコアが算出される。

(4) 事前評価レポートの作成

品質要求毎に品質要求の重視度と事前評価指標のスコアを開発者にレポート形式で通知する。開発者は品質要求の重視度に対して、事前評価のスコアが下回る場合、品質要求にリスク有と判断できる。

5.2. スコアリングルール

スコアリングを実施するに当たり、開発者が理解し易い様に、5 点満点で評価する様にした。

スコアは品質要求とそれらのカテゴリ毎に集計される。ここで品質要求 Qi(i=1～n)は、そのカテゴリ内で、図1に示したアンケート結果によって重みづけられる。重み付けの方法は、カテゴリに属する品質要求に対して、アンケートの結果から各品質要求の重みの合計が 100%となる様に、重み付けを行った。ただし重み付けについては厳密

には計算せずに、5%もしくは 10%単位で重み付けを実施した。

例えば、「機能適合性」の 3 つの品質要求の重視度の平均値は、「OSS 自身が実現したい機能を満たしている」は5、「OSS のリリースが迅速で早期に機能を利用できる」は 2.6、「正常系が正しく動作しており，早期に機能を確認できる」は 4.8 であった。これらの平均値からそれぞれの項目の重み(Bi)は、それぞれ 40%,20%.40%とした。

品質要求のカテゴリ毎の品質要求値(QV)は、品質要求の値(Qi)に重みを考慮し,以下の式で算出した。また、図2にスコアリングルールを示す。

$$QV= \sum_{i=1}^{n}(Qi * Bi)$$

品質要求に対するカテゴリ毎の事前評価指標のスコア(SV)も品質要求値のカテゴリ値(QV)と同様の重み(Bi)で算出した。品質要求毎の事前評価データのスコアを(Si)とするとき、品質要求カテゴリでの事前評価データのスコア(SV)は、以下の式で算出した。

$$SV= \sum_{i=1}^{n}(Si * Bi)$$

ここで各々の品質要求に対する事前評価データのスコア値(Si)は、複数の関連する事前評価指標より、算出される。品質要求に対応するそれぞれの事前評価指標に対して、品質要求に対する関連度(Ri)を決定した。関連度は、5 点満点として、関連大 5 点、中 3 点、小 1 点とした。それぞれの事前評価指標のスコア(Ei)は、5 点満点とし、指標毎にスコアの決定方法を定めた。例えば、書籍数や WEB サイト数の様に数値化可能なデータは、相対的に 5 段階評価でスコアを決定した。マニュアルの有無等の2極値であるものは、1 点、5 点とした。またデータ取得ができなかった場合は、スコアを 0 点とした。
事前評価指標の品質要求ごとのスコアは以下の式とした。また、図 3 にスコアリングルールを示す。

$$Si = \frac{\sum_{1=1}^{m}(Ei*Ri/5)}{(m-m_0)}$$

※m は品質要求に結びついた事前評価データの数、m_0 は Ei=0 データ取得不能で評価できなかった事前評価指標の数を示す。

5.3. 事前評価レポート

開発者から見た品質要求と事前評価指標によるスコアリング結果から、OSS の事前評価レポートを作成し、開発

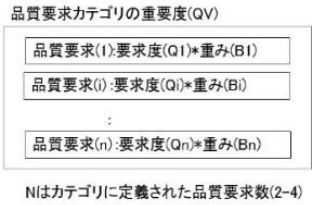

図 2 品質要求カテゴリのスコアリングルール

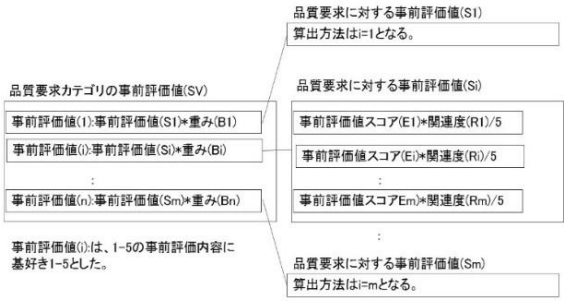

図 3 事前評価値のスコアリングルール

前にOSS利用時のリスクを特定できる様にした。事前評価レポートは、OSS の全体像を示す総括パートと個別の品質要求のスコアを示す品質要求評価パートの 2 つで構成される。

総括パート（図 4 参照）では、OSS 基本情報①として、OSS の公開ディレクトリやライセンス情報や機能概要が示される。総合判定チェック②として、各品質要求カテゴリのレーダチャートが記載される。レーダチャートでは、品質要求に対する期待度と事前評価指標によるスコアがカテゴリ毎に正規化され、レーダチャートの形状から、要求に対する期待度とスコアの関係が判断できる様になっている。ここで、事前評価指標のデータが一切取得できず、事前評価の評価値が決定できなかったカテゴリについては、スコア値を 0 としてレーダチャート上に表示した。レーダチャート上は要求を大きく下回る結果となり、リスク項目として表示される。

③に社内での OSS の利用状況、問題発生状況、社内ノウハウの一覧が表示される。

品質要求パート(表 6 参照)では、品質要求カテゴリ④毎に、品質要求⑤に対して、開発者が選択した重視度⑥と事前評価データによりスコアリングした事前評価スコア⑦（評点）が記載される。これらの品質要求毎のスコアを品質要求の重み付けを加味してカテゴリの重視度と価値を求め、（算出方法は 5.2 スコアリングルールを参照）、カテゴリ評価⑧とした。総合評価⑨は。カテゴリ評価の重視度と評価値を比較し、評価値が重視度以上であれば○:リスク低、評価値が重視度より低く差が 2 点以下であれば△:リスク中、2 点より大きければ、×;リスク大とした。

図 4. 事前評価レポートの例　総括パート

ここで△と×の閾値 2 点は 5 段階評価の中間点(1,3,5)の間隔から設定した。閾値の正当性については検討の余地があったが、後のトライアル結果を見て判断する事にした。総合評価が△、×の品要求項目については、評価結果に関する理由をコメント⑩として記載し、特に原因となった事前評価指標の内容を記載した

6 事前評価手法のトライアルと評価結果

研究した事前評価手法を用いて、事前評価手法の効果を評価した。評価対象として、2015 年度下期から 2016 年度上期に開発を完了したプロジェクト 12PRJ の内、任意の 4PRJ を対象とした。1 プロジェクトで複数の OSS を利用している場合があり、評価対象となった OSS は 6 種類であった。

トライアルは、開発メンバにより、品質要求の重視度を回答してもらい、トライアル実施チームにより、事前評価レポートを実施し、作成した事前評価レポートを元に実際に開発を実施した開発メンバとレビューを行い、事前評価レポートと開発実績を比較する事で実施した。

開発実績で問題が発生した項目は、事前評価時のリスクがどの様に検知されたかを確認するため、原因のレビューを実施した。また、カテゴリ毎に評価として、事前評価レポートを元に、カテゴリ毎にリスク予測と開発実績を比較した。開発実績は、品質要求のカテゴリ毎に〇：問題発生はなかった。△：問題までは至らないが、OSS導入に課題があった。×：OSS利用の問題が発生した。の三段階で評価した。

6.1 発生した OSS 利用時の問題とリスク検出の調査結果

今回のトライアルで評価を実施した OSS6 種について、延べ 5 件の OSS 利用時の問題が顕在化した。ここで挙げる問題については、OSS の機能的な障害(バグ)に加えて、OSS 利用時の問題を含んでいる。例えば、OSS の機能自体に問題はないが OSS の利用方法調査に想定外

表 6　事前評価レポート　品質要求パート

④カテゴリ	⑤品質要求	⑥重視度	⑦評価値	⑧カテゴリ評価 重視度	⑧カテゴリ評価 評価値	⑨総合評価	⑩コメント
使用性	OSS自体の操作性が利用者の要求を満す。	5	1.7	4.8	2.8	△	操作性とドキュメントの充実にリスクがある。 ・Webサイト数が少ないためOSS自体の情報が十分に得られない可能性がある。 ・社内ノウハウも未登録である
	ドキュメントや書籍が揃っている。	5	3.0				
	OSSの操作が容易にできる。	4.3	2.6				
	OSSの利用例が多い	5	4.0				
セキュリティ	脆弱性に関する報告小	2	3.0	2.0	3.0	〇	
保守性	プログラム構造がホワイトボックス化しやすい	5	2.1	5.0	2.0	×	ソースコードの構造、可読性ともリスクが高い。 ・複雑度の評価(スコア1) ・コメント率の評価(スコア2) ソースコードの解析のための工数を開発工程に盛り込む事を推奨する。
	ソースコードの読解性が良い	5	2.0				

表 7　発生した OSS 利用時の問題

発生OSS	問題内容	OSSの障害	利用時の問題
OSS1. OSS2	問題1：不用処理による性能問題	一部該当	該当する
OSS1 OSS2	問題2：情報不足による利用方法不明	該当しない	該当する
OSS3	問題3：ハードウェアの互換性不足	該当しない	該当する

の工数が発生した問題（問題 2）や初期調査が不完全であったため、開発後半でハードウェアの互換性が発生した問題（問題 3）がそれに当たる。これらの問題も何らかのリスク管理ができていれば、開発工程遅延や予定外の対応工数が不要であったと考え、問題として抽出した。

問題 1,2 は、問題の原因が OSS1,OSS2 のインターフェースや処理手順にあり、問題の解決には、OSS1,OSS2 双方の修正が必要となったため、両 OSS の問題とした。

問題 1 の内容は、OSS の処理に不要な処理が入っており、動作は正常に動作するが、システムの要求性能に至らかったという事象であった。また不要な処理を除去するために OSS の改造を実施したが、OSS の内容理解に多くの時間を費やすことになり、工程遅延を招いた。

問題 2 は、OSS の機能を利用する際、必要なパラメータ設定などのドキュメントやサンプルコードが不足しており、利用方法が不明であった。利用方法を理解するため、OSS 内部の解析に多くの時間を費やす事になった。

問題 3 は、OSS を利用した際、システムで利用するハードウェアに対応していないことが工程途中で判明し対応を実施しなければならなかった。この問題も急遽 OSS を解析して、独自の改造を実施した。

各問題ともに、直接の原因に対応する品質要求に加えて、「実現したい機能を実現できる。」とソースコードの解析で苦労をしたことから、ドキュメントやホワイトボックス化に対する品質要求が問題発生の品質要求として挙げられた。

表 8 発生問題と品質要求の対応

問題	OSS	カテゴリ	品質要求	要求スコア	評価スコア	事前評価
1	OSS1	機能適合性	実現したい機能を実現できる。	5	2.4	×
		性能	指定された性能を満たすことができる。	4	1.2	×
		保守性	構造が容易でホワイトボックス化しやすい。	5	2.1	×
2	OSS1	使用性	ドキュメントが整っていて情報が得られる	5	3.0	△
		保守性	構造が容易でホワイトボックス化しやすい。	5	2.1	×
1	OSS2	機能適合性	実現したい機能を実現できる。	4	1.6	×
		性能	指定された性能を満たすことができる。	4	0.6	×
		保守性	構造が容易でホワイトボックス化しやすい。	5	2.2	×
2	OSS2	使用性	ドキュメントが整っていて情報が得られる	5	2.5	×
		保守性	構造が容易でホワイトボックス化しやすい。	5	2.2	×
3	OSS3	機能適合性	実現したい機能を実現できる。	4.2	1.4	×
		互換性	複数種類のOSやハードに対応している	4	1.5	×
		使用性	ドキュメントが整っていて情報が得られる	5	3.2	△
		保守性	構造が容易でホワイトボックス化しやすい。	5	2.0	×

表 8 に問題発生に関連があった品質要求の一覧と事前評価の結果を示す。延べ 14 種の大半にあたる 12 種の品質評価指標でリスク大（×）の評価であった。

問題1の直接原因と対応する品質要求の評価詳細を表 9 に示す。性能については、社内での利用実績とノウハウで評価しており、利用実績無しの場合の評価は1、ありの場合は 3、類似システムでの利用有りの場合 5 の評価とした。ノウハウも同様に 1、3 として、ノウハウに性能の記述がある場合を 5 とした。これらのデータに品質指標との関連度を 5 点満点で設定し、5.2 項に示すスコアリングルールで計算した結果が関連度反映後のスコアとなる。いずれの OSS もノウハウ、実績もほぼないため、関連度反映後のスコアは 1.2、0.6 と大変低くなっており、要求の重視度 4 を大きく下回っている。これにより、リスク大（×）との評価結果となり、問題 1 の発生リスクを抽出できた。

問題 2 の直接原因と対応する品質要求の評価詳細を表 10 に示す。品質要求は、5 つの事前評価指標で評価される。WEB サイト数、書籍数、SLIDESHARE は、当社でデータを収集した全 OSS の事前評価指標から相対的に 1～5 点でスコアを算出している。マニュアルやユーザ会については有無で 1 点もしくは 5 点の評価としている。問題 1 の例と同様に関連度反映後のスコアを算出した結果は、それぞれ、3、2.5 であった。要求の重視度は 5 であり、OSS2 はリスク大、OSS1 はぎりぎりの点数リスク中と評価された。OSS1,2 共にほぼ問題 2 の発生リスクを抽出できた。

問題 3 の直接原因と対応する品質要求の評価詳細を表 11 に示す。互換性については、OSS のバージョン互換性と複数ハードウェアの対応で評価した。評価の詳細を表 11 に示す。複数バージョンの対応は、対応していたため 5 点となった。複数ハードウェア対応は、OSS3 は対応していないため、1 点となった。関連度は 3 であり、スコアは 1.5 点となった。結果、品質要求の重視度 4 に対して下回っており、問題 3 の発生リスクを抽出できた。更に、機能適合性や保守性についても、同様に事前評価結果を確認して、問題に対するリスクを抽出できたことを確認した。

表 9 性能の品質要求評価の詳細

事前評価指標	関連度	評価 (OSS1)	評価 (OSS2)
社内のノウハウの有無	3	1	1
社内での利用実績	3	3	1
スコア（関連度反映後）	—	1.2	0.6

表 10 使用性の品質評価の詳細

事前評価指標	関連度	評価 (OSS1)	評価 (OSS2)
WEBサイト数	5	1	1
書籍数	5	3	2
SLIDESHARE	5	–	2
マニュアルの有無	5	5	5
ユーザ会の有無	5	–	–
スコア（関連度反映後）	—	3	2.5

表 11 互換性の品質評価の詳細

事前評価指標	関連度	評価 (OSS3)
複数Version対応	3	5
複数ハードウェア対応	3	1
スコア（関連度反映後）	—	1.5

6.2 品質カテゴリ毎の評価

次に、作成した事前評価レポートを元にトライアルチームが開発チームに開発時の問題発生状況をヒアリングした。ヒアリング時には、開発チームの回答がリスク予測結果に左右されない様、カテゴリ毎のリスク予測結果は伏せて、カテゴリ毎の問題発生状況を、問題発生、多少問題あり（多少問題があったのもの開発に大きな支障はなかった）、問題なしの 3 択で回答して貰った。表 12 に事前評価レポートでのリスク判断件数と開発チームでのヒアリング結果を示す。

リスク予測と開発チームの評価を比較した件数の集計結果を以下に示す。

- リスク予測が実績評価と一致した件数　17 件
- リスク予測より実績評価と悪化した件数　2 件
- リスク予測より評価と好転した件数　22 件

(1) リスク予測が実績評価と一致したカテゴリの分析

リスク予測が評価と一致したカテゴリは、6.1 項で述べたリスク予測が問題として顕在化したカテゴリ 10 個に加え

表 12 品質カテゴリ毎のリスク予測と開発チームでの評価

事前評価レポートの評価結果		開発実績に伴う開発チームの評価	
事前評価	カテゴリ件数	実績	カテゴリ件数
リスク低	5	問題なし	5
		多少問題あり	0
		問題発生	0
リスク中	13	問題なし	9
		多少問題あり	2
		問題発生	2
リスク大	23	問題なし	11
		多少問題あり	2
		問題発生	10
評価不能	1	問題なし	1

表 13 リスク低で、問題が発生しなかったカテゴリ

OSS	カテゴリ	重視度	評価値
OSS1	セキュリティ	2.0	3.0
OSS3	セキュリティ	1.0	1.0
OSS4	保守性	1.0	2.4
OSS5	保守性	1.0	2.4
OSS6	保守性	1.0	2.5

て、リスク低で問題が発生しなかったカテゴリ 5 個とリスク中で多少問題があったカテゴリ 2 個であった。

リスク低で、問題が発生しなかったカテゴリの評価を表 13 に示す。いずれもカテゴリについての重視度が低く、開発者から見ても問題の顕在化の可能性が薄い品質要求であった。予測通り、問題は発生せず、リスク予測と結果が一致した。

リスク中で多少問題があった2カテゴリは、問題 2 に関連するカテゴリであった。

開発チームは、OSS1,2 の信頼性のカテゴリを多少問題ありと評価した。これは問題 2 の対応中にソースコードを解析し、ソースコードの構造が大変複雑であった事から開発上では問題ないものの、OSS の品質に開発チームが懸念を持った事から、多少問題ありの評価となった。信頼性に対する事前評価結果もリスク中との結果となっており、正しくリスク予測ができたと判断した

(2) 事前評価より結果が悪化したカテゴリの分析

事前評価より、結果が悪くなってしまった 2 項目については、リスクが高いにも関わらず、リスクに対する配慮が低下してしまう可能性があるため、重要な問題と認識した。その原因として、問題発生に対応した品質要求のリスク評価結果はリスク大であるものの、同一カテゴリの他の品質要求の評価結果がさほど悪くなかったため、カテゴリとしてのスコアが良くなってしまった事が挙げられる。

表 14 に該当カテゴリの 1 つである機能適合性カテゴリの事前評価詳細を示す。問題発生に関連する品質要求は「実現したい機能が実現できる。」であり、これはリスク大との判定結果であったが、残る 2 つカテゴリの内、1個がリスク低であり、カテゴリ全体としては、リスク中の判定となった。

表 14 機能適合性カテゴリの事前評価詳細

品質要求	要求の重み	重視度スコア	スコア	結果
実現したい機能を実現できる。	0.4	5	2.4	×
リリースが迅速である。	0.2	2	2.5	○
正常系が正しく動作する、	0.4	5	2.4	×
合計（カテゴリ評価）	1.0	4.4	2.4	△

表 15 OSS 毎のリスク予測＞実績のカテゴリ件数

OSS	問題あり	リスク予測＞実績	社内での利用実績
OSS1	有り	1	1
OSS2	有り	0	0
OSS3	有り	2	0
OSS4	無し	6	3
OSS5	無し	6	7
OSS6	無し	6	5

現在のカテゴリ評価は品質要求の重み付けでスコアを計算し判定しているが、これに加えて、カテゴリ内品質要求のリスク大の有無を考慮して、リスク大の項目がある場合、カテゴリ評価をリスク大にするといったカテゴリ評価方法の改善が必要と考えた。

(3)事前評価結果より結果が良好なカテゴリの分析

事前評価でリスク有と判断したが、問題なしとなったカテゴリが 22 個存在した。

リスク管理において顕在化しないリスクも当然あり得るため、事前評価結果より結果が好転することは止むを得ないが、不要なリスク項目が数多くある場合、開発者に余計な負担がかかってしまう。また多数のリスクがある場合、重点的に扱われるリスクを明確にするためにも、リスクの検出の精度を向上させる必要があると判断し、リスク予測より結果が向上した品質カテゴリの分析を行った。

分析の結果、大きく 2 つの原因があると分析した。

1 つ目は、OSS 評価方法の問題である。表 15 に、OSS 毎の問題発生の有無、リスク予測より実績が向上したカテゴリ数ならびに、事前評価レポートに記載された OSS の利用実績件数を示す。

OSS(OSS1～3)は、リスクが顕在化して、問題が発生した OSS であり、当然リスク予測の精度が上がっている。一方問題発生がなかった OSS は、ほとんどの品質カテゴリでリスク予測より、実績が向上しており、過剰にリスクを予測した結果になった。

OSS1～3 は、社内では新規導入（もしくは導入実績が少ない）が OSS4-6 は、社内の利用実績が比較的多く、複数のプロジェクトで使われており、従来の使い方であれば新たな問題が発生することは稀である。今回のトライアルでも OSS4-6 について問題発生はなく、予測したリスクも顕在化することはなかった。

利用実績が少なからず OSS リスク予測に有効な事は理解していて評価指標に盛り込んでいるが、利用実績がスコアに関与する影響は必ずしも高くない。利用実績が

スコアに与える割合を補正して、事前評価のスコアに与える影響を上げればリスク予測の精度向上に貢献できると分析した。

2 つ目は、品質要求に対する重視度の定義である。リスク予測より実績が向上した品質要求カテゴリの多くに重視度が想定よりも高いものが多く見受けられた。現手法では、重視度を開発プロジェクトの主観で決定している。

例えば、性能での「OSS が指定された性能を満たす事ができる」という設問は、文言の捉えようによっては、顧客から要求される性能とは無関係に、「重視する」と回答される可能性があり、今回のトライアルでもその様に回答したプロジェクトも見受けられた。品質要求の重視度についても、設問内容を含めて、改善の余地があると判断した。

7 まとめ

本研究では開発現場への普及や運用の容易性を目的に開発者の持つ品質要求を元に OSS 利用リスクを特定する手法を考案し、トライアルを実施した結果、以下の評価結果を得た。

・OSS 利用時発生した問題 5 件について、品質要求単位、品質カテゴリ単位の双方で、リスク高、中として検知できた。

・リスク予測と開発チームの評価実績が一致したカテゴリ数は 17 個であり、全カテゴリ数 42 個に対して約 40%の確度でリスク予測がトライアル結果と一致した。

以上により、本手法は予測精度向上の余地はあるが、リスク評価の手法として有用であると判断した。

8 今後の課題

(1) リスク予測精度向上の課題

リスク予測において、問題発生を検知できた反面、過剰にリスクを検知している可能性がある。品質要求の入力のやり方やスコアリング方法など、更なる改善の余地が残っている。本稿の執筆時、トライアルは継続中である。トライアル終了時に、データを再度集計、分析して、手法にフィードバックしリスク予測精度を向上させていきたい。

(2) 予測コストの明確化と更なる削減

本手法の実行において、大きく 2 つのコストが必要となる。1 つは、評価のための事前評価情報の収集コストであり、これは共同研究者である九州大学の研究にて、1OSS 当たり、30 分～2 時間とのベンチマーク結果が得られている[12]。取得手順マニュアルの整備や自動化を進めて、更なる運用コスト削減を進めていきたい、

2 つ目は、事前評価レポートの作成時間である。今回手法の検証作業も併せて、評価レポート作成を行ったため、最初のレポート作成には 2 週間、作業が慣れてきた後半でも 2-3 日を要している。正確な作成時間の計測を進めると共に、スコアリング等、一部ツール化、自動化を図り、コストの削減を実施していく。

謝辞

当社が過去に利用した OSS 約 170 種について、延べ 3 万種に及ぶ OSS の事前評価データを収集して頂いた九州大学）鵜林研究室の学生の皆様に感謝致します。

参考文献

[1] 独立行政法人情報処理推進機構：第3 回オープンソースソフトウェア活用ビジネス実態調査（2010）.

[2] Atos: Quali_cation and selection of open source software (QSOS), version 2.0. (2013).

[3] Duijnhouwer, F.-W. and Widdows, C.: Open Source Maturity Model, *Capgemini Expert Letter* (2003).

[4] Northeast Asia, O.: RepOSS: A Flexible OSS Assessment Repository (2012).

[5] Petrinja, E., Nambakam, R. and Sillitti, A.: Introducing the opensource maturity model, *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS'09. ICSE Workshop on*, IEEE, pp. 37{41 (2009).

[6] Taibi, D., Lavazza, L. and Morasca, S.: OpenBQR: a framework for the assessment of OSS, *IFIP International Conference on Open Source Systems*, Springer, pp. 173-186 (2007).

[7] Wasserman, T. and Das, A.: Using FLOSSmole Data in Determining Business Readiness Ratings, *WoPDaSD* (2007).

[8] Deprez, J.-C. and Alexandre, S.: Comparing assessment methodologies for free/open source software: OpenBRR and QSOS, *International Conference on Product Focused Software Process Improvement*, Springer, pp. 189-203 (2008).

[9] Petrinja, E., Sillitti, A. and Succi, G.: Comparing OpenBRR, QSOS, and OMM assessment models, *IFIP International Conference on Open Source Systems*, Springer, pp. 224{238 (2010).

[10] Petrinja, E. and Succi, G.: Assessing the open source development processes using OMM, *Advances in Software Engineering*, Vol. 2012, p. 8 (2012).

[11]情報処理学会：第190回ソフトウェア工学研究会製品開発におけるOSS導入のためのOSS事前評価に向けた初期調査　松本、山下、鵜林、亀井、深海、岩崎(2016)

[12]日本ソフトウェア科学会：第23会ソフトウェア工学の基礎ワークショップ(FOSE2016) 開発現場への導入を想定したOSS事前評価手法確立に向けた調査　松本、山下、鵜林、亀井、大浦、高山、岩崎(2016)

[13] コンピュータソフトウェア,Vol33.No1(2016) pp28-40,:Open Source Software Engineering 伊原　大平