人工知能学会
第13回知識流通ネットワーク研究会
SIG-KSN-013-02

オープンソースソフトウェア開発コミュニティの成熟度モデルとソフトウェア品質に関する考察

○桑田喜隆(*1)　三浦 広志(*1)
(*1) NTTデータ

A Study on a Maturity Model for Open Source Development Community and the Estimation of the Quality of Software

Yoshitaka Kuwata(*1) and Hiroshi Miura(*1)
(*1) NTT DATA Corporation, Japan

概要

オープンソースソフトウエア（0SS）はコミュニティによりソフトウェアを開発する手法である．ソフトウェア開発および維持管理手法として注目されている．ソースコードを公開することにより，多くの開発者の目に触れる機会があり，不具合なども発見されやすいという特徴がある．このため，企業内で開発されたソフトウェアなどに比べ，ソフトウェア品質が高くなる可能性がある．しかし，0SSに関する評価方法や具体的な評価項目に関して確立した方法がないのが現状である．
筆者らは，0SS 開発コミュニティの成熟度モデルに注目しモデルに基づく評価方法について提案する．

Abstract

Open Source Software (OSS) is widely noticed as a unique way to develop and maintain software by community. Because the development process, source code, and related documents are open, it is expected that many developers and testers examine their software and thus the quality of the software can be higher than those developed by companies, which is called proprietary software. However, there are no established methods for the evaluation of OSS, neither actual terms of the evaluation of OSS.
We propose an evaluation method which is based on the maturity model of OSS development community.

1. はじめに

オープンソースソフトウエア（0SS）は開発コミュニティでオープンに開発を進める，新しいタイプのソフトウェアである．成果物であるソフトウェアのソースコードやドキュメントは公開される．

0SS の定義については，参考資料 1)にあるが，0SS の現状に沿った特徴を以下に述べる．

(1) 開発プロセスや体制が公開されており，仕様の策定や開発，試験に参加することが可能である．
(2) ソースコードやドキュメントなどの，プロジェクト成果物がインターネット上で公開されている．
(3) オペレーティング・システムからライブラリ，アプリケーションまで，非常に多くの分野に渡ってプロジェクトがある．また作成されているソフトウェア規模も異なる．

[1] Yoshitaka Kuwata
NTT データ　基盤システム事業本部
東京都江東区豊洲 3-3-9 豊洲センタービルアネックス
kuwatay@nttdata.co.jp

(4) 一人から数千人規模のプロジェクトまで開発コミュニティの規模が異なる. またプロジェクトの体制が異なる.
(5) プロジェクトの状態が異なる. 構想段階のものから, 既に商用で利用されているもの, 陳腐化して利用されなくなってしまったものまである.
(6) 利用ライセンス条件が多様である. 著作権を放棄した Public Domain から二次利用に関して強く制約を課す GPL[2]まで幅が広い. また, OSS 同士でも相容れないライセンスも存在する.
(7) ソフトウェアの品質が異なる. 商用製品と同等の品質を持つものから, コンパイルさえ通らないものもある. いわば, 玉石混交である.

OSS は公開されたプロセスのもとに開発が進められ, ソースコードも公開されていることから, 多くの人によって改良される機会があり, 結果としてソフトウェアとしての品質が向上することが期待される. ソフトウェア自体が無償で利用可能であり, 商用ソフトウェアのようなライセンス費用がかからないため, コスト的なメリットも大きい.

他方で, 成果物や試験に関する考え方がプロジェクトによって異なっているため, 一律にその成果物であるソフトウェアを評価することが難しい. また, 仮に品質の高いソフトウェアであることがわかっている場合でも, 開発の継続性や第三者権利の侵害などの法的リスクも課題となる.

筆者らは OSS を使って実際的なシステム構築することを前提に, ソフトウェアを評価したいと考えている. 特にソフトウェアの品質や開発の継続性に関しては非常に重要な項目である. そこで, 本稿ではコミュニティの成熟度に注目し, OSS を評価する方法に関して考察した.

2. 本研究の目的

OSS コミュニティによって開発されたソフトウェアを利用することを想定し, 採否を決定する. まず, OSS の評価のために必要なメトリックス定義する. また, メトリクスを基にしてソフトウェアの品質の推定をするためのフレームワークを提供することを目的とした.

ソフトウェアの機能的な要件に関しては, 目的によって合致するかどうかが決まるため, 一概に評価が出来ない. ここでは非機能要件に注目し, 評価することを目的とする. 具体的な項目としては, 以下の要件を評価することを最終的な目的とする.

- ソフトウェア品質
- 開発および維持管理の継続性
- 拡張性

3. これまでの取り組み

開発工程や中間成果物が公開されており, 定量的に分析しやすいことから, ソフトウェアサイエンス分野で OSS を題材にした研究は多い.

例えば, 参考文献 3)は開発工程に注目し OSS 開発で使われているソースコードマネジメントシステムのログから, OSS のリリース時期などの状態を推定する方法を提案している. 参考文献 4)は OSS コミュニティのメーリングリストのログからプロジェクトのアクティビティを推定する方法の提案である. 参考文献 5)ではニューラルネットワークを利用してソフトウェアコンポーネントの重要度を学習し, バグトラッキングシステムの故障データから推定される重要度との比較を行っている.

これらの先行研究では OSS プロジェクト成果物をソフトウェアの事例として扱っているが, OSS 固有の開発プロセスに注目して分析を行っているわけではない. 本取り組みでは, ソフトウェアそのものを対象として扱う代わりに, OSS コミュニティの構造に注目し, 評価メトリクスを抽出することで評価を行うアプローチを取っている点が異なる.

4. 分析のアプローチ

まず OSS コミュニティの構造の分析を実施した. 図 1 に本稿で提案する OSS の関係

モデルを示す.

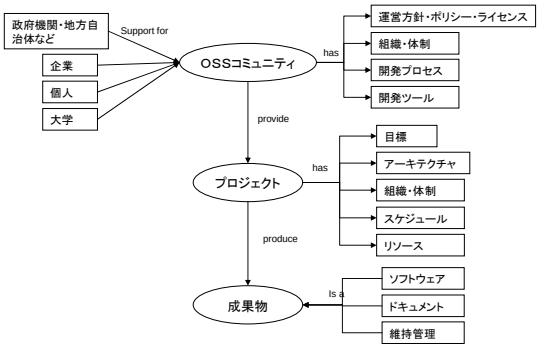

図１　OSS の関係モデル

この関係モデルでは，OSS コミュニティがプロジェクトを提供するという親子関係にある．プロジェクトにはそれぞれ目標やスケジュールなどがある．ソフトウェアやドキュメントなどの成果物はプロジェクトによって作成される.

図２に OSS モデルの派生モデルを示す.

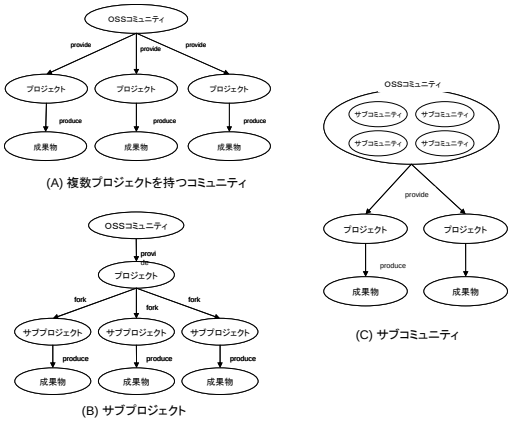

図２　OSS コミュニティの派生モデル

(A) ひとつのコミュニティが複数のプロジェクトを提供することがある．(B)規模の大きなプロジェクトにおいて機能的にプロジェクトを分割したサブプロジェクトを持つ場合がある．(C) OSS コミュニティの構造も単一ではなく，地区や分野などによるサブコミュニティが形成されることが考えられる.

5. OSS 開発コミュニティの成熟度モデル

OSS 開発コミュニティを「組織」として捉えた場合，CMMI の定義する組織の能力成熟度モデルを参考にして，状態を定義することが可能である.

以下に，OSS 開発コミュニティとプロジェクトの状態に基づき定義した，OSS 開発コミュニティの成熟度モデルを提案する.

表1　OSS 開発コミュニティの成熟度モデル

レベル	状態	定義
1	初期状態	プロジェクトが開始された状態．まだ明確なコミュニティは形成されていない.
2	管理された状態	コミュニティが形成されて，プロジェクトが継続的に行われるようになった状態．多くのプロジェクトでは，この段階で成果物のリリースが開始される. この段階では，プロジェクトは，暗黙の合意に基づき運用され，明示的に運用ルールが決められているわけではない.
3	定義された状態	コミュニティによってプロジェクトの運用ルールやソフトウェアの開発プロセスが明示された状態. 実際にルールやプロセスに従ってプロジェクトが実行され，ソフトウェアがリリースされる.
4	定量的に管理された状態	コミュニティによってプロジェクトの状態が定量的に測定されている状態. 例えば，組織的に品質管理を実施してリリースの判断をおこなう.
5	最適化している状態	定量的なデータに基づき，開発プロセスやコミュニティの運営などが常に見直されながら運用されている状態. 例えば，品質を向上するためバグの収束状況を把握し，検証プロセスを見直しながら改善する.

6. 成熟度モデルの検証

成熟度モデルの検証のために，実際の OSS コミュニティがどのような状態にあり，どのような管理を行っているかを調査した．以下に調査結果を述べる.

6.1 Linux Kernel Project

OSS のオペレーティング・システムである Linux は The Linux Foundation によって開発されている．参考文献 8)およびホームページをもとに調査した Linux Kernel Project の概要を以下にまとめる．

- 組織
  - The Linux Foundation
- プロジェクト開始
  - 1991年
- 開発者数 5000人以上
  - リリースに関わったエンジニアの延べ人数
  - 少数精鋭：上位30人の開発者が25%の変更を実施している。
- ソースコード規模
  - 11.5M行
  - レポジトリ管理システムgitで管理
  - ライセンス GPL2
- リリース管理方式
  - 100程度のサブシステムによる分散管理
  - サブシステムのメインテナーによる承認
  - ステージングリリース

特徴としては，Linux は非常に大きなソフトウェアであり，関連する開発者数も 5000 人以上と多いことが上げられる．プロジェクトは 100 程度のサブプロジェクトに分割され，それぞれにメインテナーと呼ばれる責任者が割り当てられている．プロジェクトの形態 B に相当する．メインテナーを中心にプロジェクトが進められる管理方式を取っている．リリース判断もメインテナーが行う．

テストスーツなどによって数値的な情報把握も実施されている．（レベル４に相当）

加えて，リリース方法などの改善も行っているので，コミュニティの成熟度としてレベル５に相当すると考えられる．

6.2 GNU Project

Free Software Foundation (FSF)では，エディタやコンパイラといったソフトウェアツールを開発している．ホームページの情報をもとに調査した GNU Project の概要を以下にまとめる．

- 組織
  - Free Software Foundation
- プロジェクト開始
  - 1985年
- サブプロジェクト数（GNUパッケージ数）
  - 364
- 開発者数
- ソースコード規模
- リリース管理方式
  - パッケージのメインテナーがソースコードの管理を実施する
    - Information for Maintainers of GNU Softwareに手順が定義
  - Git、Savannahによるソースコード管理
    - ソースコードの変更履歴の参照が可能
  - Hydraによる移植性のチェック
  - ボランティアによるテスト
    - リリースにあたっての品質に関する記述はなし

特徴としては，プロジェクトは 364 個の GNU パッケージと呼ぶサブプロジェクト群から構成されている．図２の分類では，プロジェクトの形態 A に相当する．

各パッケージのプロジェクト管理はメインテナーに任されているが，手順などはドキュメント化 [9]されているため，レベル３に相当すると考えられる．しかじ，手順どおりに実施されているかどうかはプロジェクトおよびメインテナーに依存する．また，ソースコードの管理はシステム化されているため，数値データの把握が可能であるが，プロジェクトによってリリース管理などに利用されているかどうかは明らかではない．このため，コミュニティの成熟度はレベル３ないしはレベル４であると判断される．ただし，プロジェクトによってはレベル３の要件を満たさない可能性がある．

6.3 Apache Software Foundation

Apache Web サーバの開発で有名な Apache Software Foundation は，他に多くのプロジェクト群サポートしており，コミュニティを形成している．ホームページの情報をもとに調査した Apache Project の概要を以下にまとめる．

- 組織
  - The Apache Software Foundation
- プロジェクト開始
  - 1985年
- サブプロジェクト数
  - 199
- 開発者数
- ソースコード規模
- リリース管理方式
  - Project management Committee (PMC) による管理（把握）
  - 個別のプロジェクトに管理やリリースサイクルは任されている
    - Apache Antプロジェクトの例
    - http://ant.apache.org/bylaws.html
  - SVNによるソースコード管理, Issue Tracking

Apache Software Foundation は 199 の

サブプロジェクト群[10]をサポートしている．複数のプロジェクトをサポートしている点はFSFと同様である．プロジェクトの承認などはProject Management Committee (PMC)によって全体で管理されているが，個別プロジェクトの管理やリリースはプロジェクトに任されている．加えて，管理ポリシーもプロジェクトごとに決定することが出来る．プロジェクトによる裁量範囲が高い反面，管理ポリシーを明示していないプロジェクトもある．

以上の調査結果から，コミュニティの成熟度はレベル２ないしはレベル３に相当すると考えられる．

6.4 OpenStack Project

OSSのクラウド基盤を開発するOpenStack Projectは2009年から開始された比較的若いプロジェクト[11]である．ホームページおよび技術情報を管理するためのlaunchpad[12]の情報をもとに調査したOpenStack Projectの概要を以下に示す．

- 組織
  - Open Stack Foundation
- プロジェクト開始
  - 2009年
- サブプロジェクト数
  - 主要 5プロジェクト
- 開発者数（登録者）
  - 5600人
- ソースコード規模
- リリース管理方式
  - ボードメンバー（有償企業会員メンバー）による意思決定
  - テクニカルコミッティーによる管理
  - 定期的なサイクルリリース（6ヶ月）
  - プロジェクト毎の分割管理
  - ステージングリリース(ベータ、RC,Final)
  - githubによるソースコード管理, launchpad によるIssue Tracking

プロジェクトの特徴として，OSSでありながら商用利用を強く意識したプロジェクト運営がなされている点である．企業からの寄付をベースにプロジェクトが進められており，主に有償会員メンバによる意思決定が行われる．またソフトウェアのリリース間隔も６ヶ月と決められており，リリースのためのマイルストン管理をしっかり実施されている．

プロジェクトの管理方法やテストなどの方法が明示されており，レベル３の要件を満たす．また，定量的に状態も把握されているため，コミュニティの成熟度はレベル４であると考えられる．

7. コミュニティの成熟度とソフトウェア品質に関する考察

本章ではコミュニティの成熟度分析を通じて得られた知見および考察を述べる．

(1) 成熟度と開始時期

前章で述べたプロジェクトはレベル２からレベル５に相当するが，必ずしも開始が早いプロジェクトのレベルが高いわけではないことがわかる．コミュニティの運営方針や管理ポリシーによって，そのレベルが変わってくると考えられる．

(2) プロジェクトの継続性

レベル４，５のコミュニティは，すでに確立されたコミュニティであり，プロジェクトの高い継続性が期待される．

他方，企業のサポートしているプロジェクトでは，トップや管理体制が変わることにより管理ポリシーなどの重要な項目が変化するリスクが存在する．

(3) 成熟度とソフトウェア品質

一般に成熟度が高いほどソフトウェアの品質が高いことが期待される．

レベル４以上の成熟度のコミュニティでは，Issue Tracking Systemの情報分析が可能である．リリース履歴を調べて，その後のバグの修正状況の確認が可能であるため，直接ソフトウェアの品質評価を実施すれば良い．

図３にApache Aries ProjectのIssue Tracking Systemの例を示す．

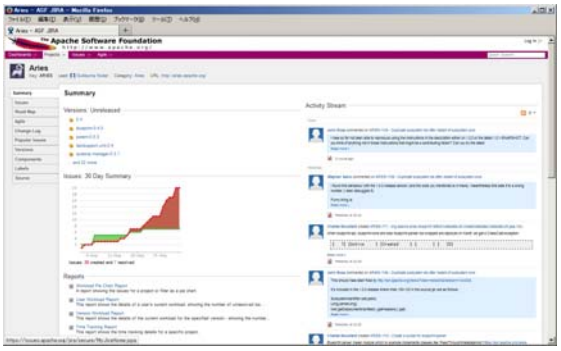

図３Issue Tracking Systemの例
(Apache Aries Project[13])

他方，開発中でリリースのたびに多くの新機能を取り込んでいるような段階のプロジェクトでは，単純に Issue Tracking System の状況から把握することが難しい．プロジェクトとしてしっかり管理できる仕組みを持っているかどうかが重要となると考える．

8. 課題と今後の取り組み

本稿では，OSS コミュニティの成熟度に基づき成果物の評価を実施する方法を提案した．いくつかの典型的なコミュニティを取り上げて，成熟度の分析を実施した．

特に成熟度 4 以上のコミュニティはソフトウェアの状態を定量的に把握する手段を保持している場合が多く，成果物の品質に関する評価は容易であると考えられる．他方，レベル 3 以下のコミュニティに関しては, メーリングリストや Wiki などのコミュニケーションのログを参照することで，その活性度が推定可能である．

今後の課題としては，本研究の目的である成果物の評価するために，OSS コミュニティの成熟度別に評価の方法を検討することが必要であると考える．

A. 参考文献

1) オープンソースの定義（日本語訳），Open Source Group Japan, http://www.opensource.jp/osd/osd-japanese.html, 2004 年 2 月
2) GNU GENERAL PUBLIC LICENSE, Free Software Foundation, Inc, http://www.gnu.org/licenses/gpl.html
3) オープンソース開発における SCM の自動分類に基づく evolution の傾向分析と品質評価, 載 家豪, 海谷 治彦, 海尻 賢二, 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス 110(458), 79-84, 2011-02-28
4) メールスレッドのクラスター分析による OSS プロジェクトのアクティビティ予測手法, 大蔵 君治, 大西 洋司, 川口 真司, 大平 雅雄, 飯田 元, 松本 健一, 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス 107(275), 41-46, 2007-10-15
5) オープンソースソフトウェアに対する遺伝的アルゴリズムに基づく信頼性評価法, 田村 慶信, 山田 茂, 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス 106(120), 31-36, 2006-06-15
6) A social networking approach to F/OSS quality assessment, Tawileh Anas, Rana Omer, and McIntosh Steve, Proceedings of the First international conference on Computer-Mediated Social Networking, ICCMSN'08, pp157-170, 2009
7) Open source software maturity model based on linear regression and bayesian analysis, Zhang Dongmin, Texas A & M University, 2007
8) 誰が Linux を開発しているか（第 2 版）, Greg Kroah-Hartman, Jonathan Corbet, and Amanda McPherson, the Linux Foundation
9) Information for Maintainers of GNU Software, Free Software Foundation http://www.gnu.org/prep/maintain/maintain.html
10) Apache Software Foundation Index: Project Listing, http://projects.apache.org/indexes/quick.html
11) The OpenStack Foundation, http://www.openstack.org/
12) OpenStack Compute (Nova) in launchpad, https://launchpad.net/nova
13) Apache Aries Project, http://aries.apache.org/