AIエージェント導入による依存関係更新プロセスの変容
- OpenHands プロジェクトの事例分析

22T2005K 阿部航大

背景と目的

OSSライブラリは、脆弱性が増加傾向にあり、**依存関係の更新** は脆弱性の修正に不可欠である[1]

Dependabot などの自動化ツールは広く使われているが、定型的な更新に限定されるという限界がある

AI エージェントによる依存関係更新の事例は現れているが、AIエージェント導入が依存関係更新プロセスに与える影響は十分に分析されていない

OpenHands リポジトリを対象としたケーススタディとして、AIエージェントが
依存関係更新プロセスに与える影響を分析する

[1]Ponta, S.E., Plate, H. & Sabetta, A. Detection, assessment and mitigation of vulnerabilities in open source dependencies. Empir Software Eng 25, 3175–3215 (2020). https://doi.org/10.1007/s10664-020-09830-x

OpenHands 

LLMを用いてソフトウェア開発作業を**自律的に実行するAIエージェント**

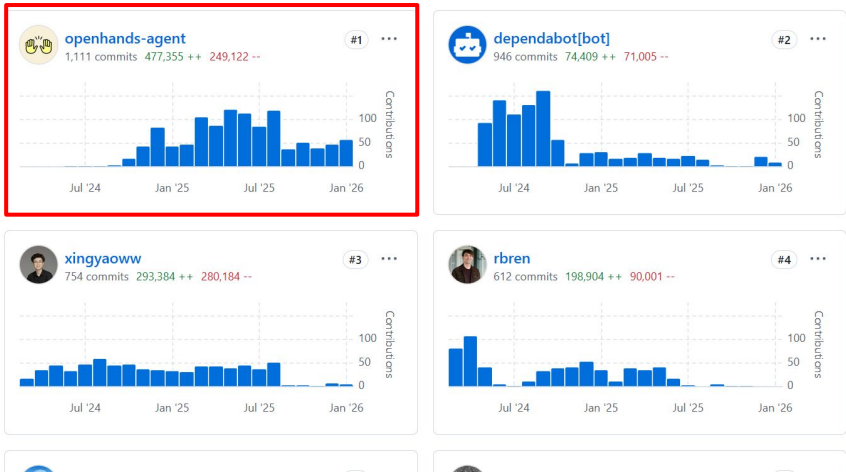
特徴

・コード生成だけでなく、**テスト実行・デバッグ・ git操作・PR作成**までを自動化可能

・GitHubと連携し、OSS開発フローに直接組み込める

OpenHandsは自らAIエージェントを開発しており、開発プロセス自体にAIエージェントを積極的に組み込んでいる最先端の事例

OpenHandsリポジトリで最もコミット数が多いのはopenhands-agentとなっている



Research Questions

RQ1：依存関係更新は時間とともにどのように変化してきているか？

- 更新主体ごとの依存関係更新数の推移
- 依存関係の鮮度（Libyear）の推移

RQ2：更新主体の違いは、依存関係更新の変化にどのような影響をもたらしているか？

- 更新主体ごとの鮮度（Libyear）への貢献度
- 更新主体ごとの更新タイプの違い

分析内容

対象リポジトリ: OpenHands

分析期間: 2024-03~2025-12

分析データ: poetry.lock、package-lock.jsonの全コミット履歴

GitHubのコミット履歴から以下の基準を用いて更新主体を3つのカテゴリに分類

- **OpenHands Agent（AIエージェント）**
  Author または Co-authored-by に"openhands agent" を含む場合
- **Dependabot**
  Authorが"dependabot"である場合
- **Human**
  上記いずれにも該当しない全ての更新

OpenHandsリポジトリでの依存関係管理

OpenHandsリポジトリでは、バックエンドはpoetry、フロントエンドはnpmという依存関係管理ツールが使用されている

ロックファイル：
実際にプロジェクトで使用されているバージョンが記録されている

フロントエンド	バックエンド
言語 ：TypeScript/JavaScript 設定ファイル： package.json ロックファイル： package-lock.json	言語： Python 設定ファイル： pyproject.toml ロックファイル： poetry.lock

Libyearの計算方法

Libyear：プロジェクトが利用している依存ライブラリのバージョンが、
最新版から合計で何年分遅れているか を数値化した指標[2]

計算方法

Libyear = Σ(最新バージョンのリリース日　- 使用バージョンのリリース日)

具体例：

- ライブラリA：2年前の版を使用 → 2.0 Libyear
- ライブラリB：0.5年前の版を使用 → 0.5 Libyear
- **合計：2.5 Libyear**

[2]J. Cox, E. Bouwers, M. van Eekelen and J. Visser, Measuring Dependency Freshness in Software Systems. In Proceedings of the 37th International Conference on Software Engineering (ICSE 2015), May 2015 https://ericbouwers.github.io/papers/icse15.pdf

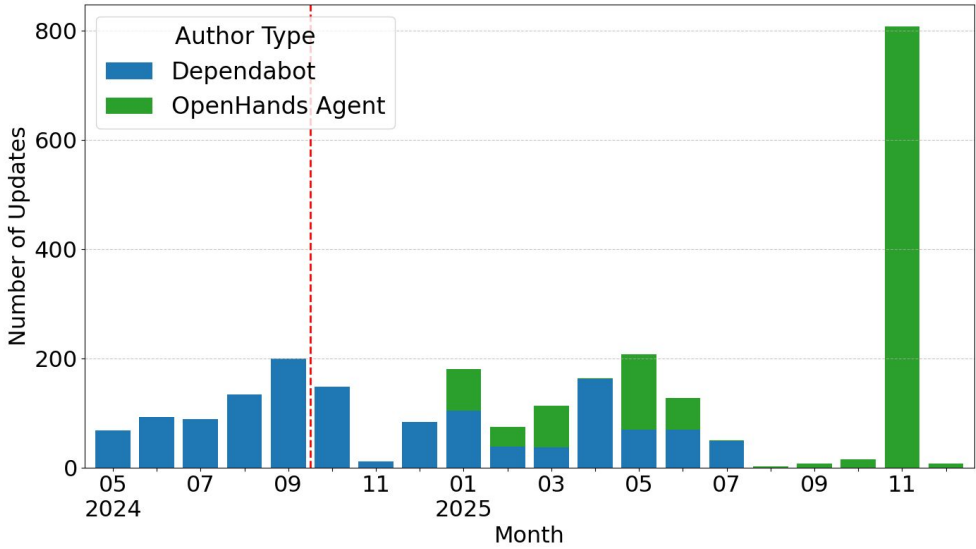
RQ1：バックエンド (poetry.lock) の依存関係更新数の推移

OpenHands Agent導入後（2024-10以降）の更新数

OpenHands Agent：1226回

Dependabot：775回

→OpenHands AgentはDependabotの約1.6倍の更新を行っている



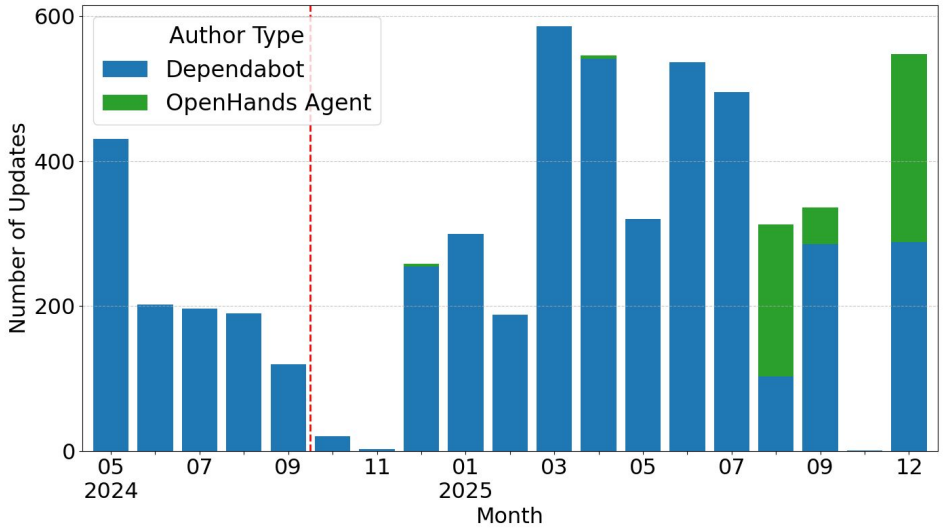
RQ1：フロントエンド (package-lock.json) の依存関係更新数の推移

OpenHands Agent導入後（2024-10以降）の更新数

OpenHands Agent：528回

Dependabot：3924回

→DependabotがOpenHands Agentの約7倍の更新を行っている



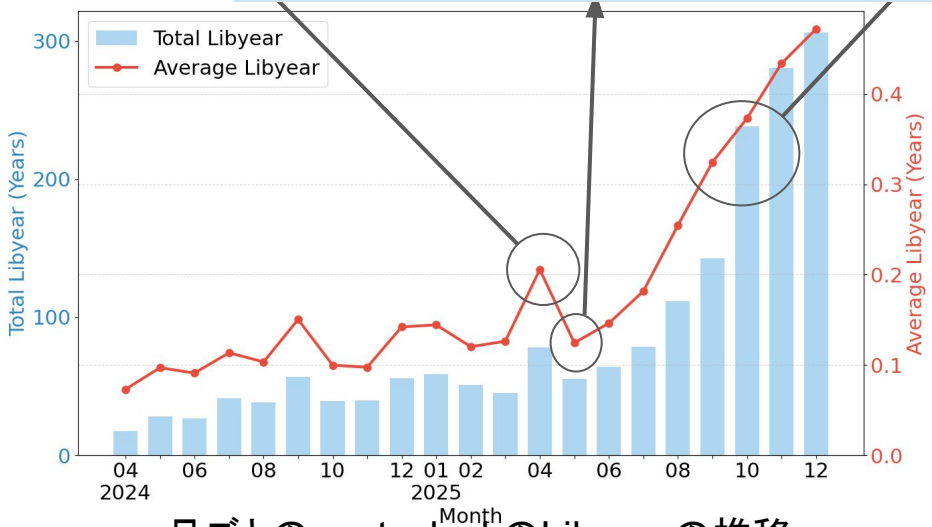
RQ1：依存関係の鮮度（Libyear）の推移

2025-04にLibyearの平均値が増加

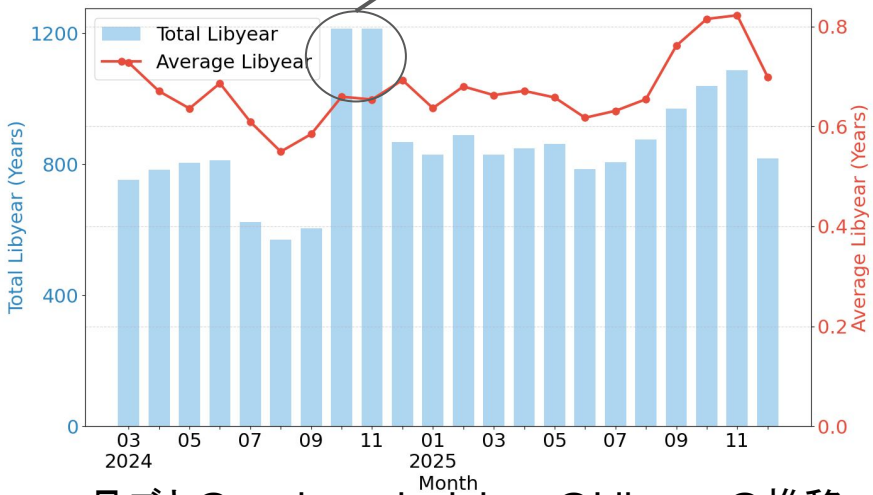
2025-10以降にLibyearが急増

2025-05にLibyearの平均値が減少

2024-10にLibyearが増加



月ごとのpoetry.lockのLibyearの推移



月ごとのpackage-lock.jsonのLibyearの推移

RQ1：依存関係の鮮度 (Libyear) の推移

バックエンド (poetry.lock) のLibyearの変化の要因

月	libyearの変化	主な原因
2025-04	増加(+33)	GPU対応の強化：nvidia-cuda-* 関連パッケージが大量導入されたが、導入時点で約1.8年の負債を抱えていた。また既存の rsa や h11 も新版リリースにより相対的に老朽化した。
2025-05	減少(-22.8)	一括アップデート: 前月に発生した遅れを解消するため、Nvidia関連、rsa, h11, 等を最新版へ更新し、負債を減らした。
2025-10	増加(+95.91)	パッケージ数の増加：201個 の新しいパッケージが追加された（439個→640個）。

RQ2：更新主体ごとの活動比較 (2024-10以降)

バックエンド (poetry.lock)

更新主体	更新回数(回)	Libyear削減量
OpenHands Agent	1025	82.37
Dependabot	381	35.45
Human	749	12.64

OpenHands AgentはDpendabotと比べ、2倍以上のLibyearを削減している

フロントエンド (package-lock.json)

更新主体	更新回数(回)	Libyear削減量
OpenHands Agent	789	107.03
Dependabot	2630	443.88
Human	1956	227.18

Dependabotが OpenHands Agentと比べ、4倍以上のLibyearを削減している

RQ2：更新タイプ (Major/Minor/Patch) の比率

バックエンド (poetry.lock)での更新タイプ別件数

更新主体	Major件数（比率）	Minor件数（比率）	Patch件数（比率）
OpenHands Agent	213 (21.0%)	442 (43.5%)	360 (35.5%)
Dependabot	19 (3.9%)	215 (44.2%)	252 (51.9%)
Human	199（26.1%）	352（46.2%）	211（27.7%）

Openhands AgentによるMajor更新の件数Dependabotと比べ多い

フロントエンド (package-lock.json)での更新タイプ別件数

更新主体	Major件数（比率）	Minor件数（比率）	Patch件数（比率）
OpenHands Agent	24 (3.1%)	241 (30.7%)	519 (66.2%)
Dependabot	128 (4.9%)	796 (30.3%)	1,705 (64.9%)
Human	76（4.0%）	677（36.0%）	1128（60.0%）

まとめ

- バックエンドではOpenHands Agentによる依存関係更新数がDependabotより多くなるという変化が起こっていたが、フロントエンドでは依然としてDependabotによる更新が大多数を占めていた

- OpenHands agentによる更新により、Libyearの削減が確認された

- フロントエンドにおいて、OpenHands agentのメジャーアップデート比率が21%に達し、Dependabotの約4%を大きく上回った
  →人間に近い高度な保守を行っていると考えられる