IPA

# 2025 年度 未踏 IT 人材発掘・育成事業 採択案件評価書

## 1. 担当 PM

竹迫 良範（神山まるごと高専 デザイン・エンジニアリング学科 教授）

## 2. クリエータ氏名

髙橋 侑大（東洋大学 情報連携学部 情報連携学科）
浅野 凌輔（東洋大学 情報連携学部 情報連携学科）
鮎澤 颯人（電気通信大学 情報理工学域 III類）

## 3. 委託金支払額

2,880,000 円

## 4. テーマ名

OS と Web ブラウザを統合したローカル LLM 支援型オペレーティングプラットフォームの開発

## 5. 関連 Web サイト

- Floorp OS Web サイト: https://ja.floorp.app/floorp-os
- Floorp Web ブラウザ GitHub リポジトリ: https://github.com/Floorp-Projects/Floorp
- Floorp OS Core GitHub リポジトリ: https://github.com/Floorp-Projects/Floorp-OS-Automator-Backend
- Sapphillon GitHub リポジトリ: https://github.com/Sapphillon/Sapphillon

## 6. テーマ概要

本プロジェクトでは、Firefox ベースの Web ブラウザである Floorp（以降、Floorp Web ブラウザ）を中核として、ローカル OS 環境と Web ブラウザ環境の分断を解消し、自然言語による指示から Web 操作とローカル操作を一連の処理として実行できる LLM プラットフォーム「Floorp OS」を開発した。

## 7. 採択理由

本提案は、高レイヤのウェブブラウザと低レイヤのローカル OS を統合し、さらに先進的な生成 AI のローカル LLM を活用してユーザの作業自動化を実現

するという、極めて野心的な取り組みである。従来分断されていたウェブアプリケーションとローカル環境をシームレスに連携させるという発想は思いつくものの技術的には非常にチャレンジングであり、実際にアウトカムを生み出すことができたときの社会的インパクトが高い点を評価した。

提案者らは既にウェブブラウザ「Floorp」の開発実績があり、5 万人以上のアクティブユーザを抱えるなど、実装力と影響力の裏付けがある。また、ローカル LLM を導入するという選択は、外国製クラウド依存に伴うセキュリティ課題や経済安全保障に対する解決策となっており、日本企業や政府機関など幅広いニーズに応えうる設計思想となっている。

本プロジェクトの成果は、単なる個人向けの便利なブラウザという枠を超え、次世代のオープンな AI ネイティブ OS の実装として、AI の民主化やデジタル主権の確立に資する可能性を持つ。これは、GAFAM を中心としたプラットフォーム支配とは一線を画し、国産・オープンソースとして展開されることに大きな意義がある。

本提案はローカル LLM の導入・管理・実行という技術課題に加え、セキュリティ・サンドボックス内でのプラグインの動作、さらには Deno ベースのカスタムランタイムの実装など、挑戦的かつ多層的な技術課題に挑戦している。

提案代表者の浅野氏はすでに独自のブラウザ開発を一から立ち上げ、継続的に成長させている実績があり、髙橋氏・鮎澤氏もバックエンドや生成 AI 領域での開発経験を持つ。開発スケジュール、役割分担、使用言語・ツール類も具体的かつ妥当であり、限られたリソース下でもプロジェクトを着実に前進させる能力があると判断した。

世の中の生成 AI のエコシステムの発展は非常に速いスピードで進行しており、採択時点では MCP サーバとの連携による自然言語によるソフトウェア制御も急速に実現が進んでいる。プロジェクト採択後も幅広く時流を捉え、当初の技術選択以外の技術に対しても客観的に動向を把握・評価し、LLM と統合した OS・ウェブブラウザのエコシステムをいち早く実現することを期待している。

## 8. 開発目標

本プロジェクトの開発目標は、ブラウザを単なる閲覧ソフトではなく、OS 全体を制御するハブ（Operating Platform）へと拡張し、Web とローカル OS の境界をなくす LLM 活用基盤を構築することである。具体的には、自然言語による指示を元に、ローカルのファイル操作と Web サイト操作を横断して自動実行できる仕組みを実現し、分断されていた両者をシームレスに繋ぐユーザ体験を提供することを目指した。さらに、ユーザの PC 上で動作するローカル LLM を活用することで、機密性の高いローカルファイルや業務データを外部サーバのクラウドに送信することなく処理できる環境の実現を目指した。また、LLM が

実行する自動化ワークフローについても、権限を細かく管理できる実行環境の上で動作させることで、セキュリティとプライバシーに配慮した安全な LLM 実行基盤の構築を目標とした。

## 9. 進捗概要

本プロジェクトでは、Web とローカル OS に分断されていた処理を自然言語による指示から一連の操作として実行できるようにするため、ユーザの指示に対して実行時に逐次判断を重ねるのではなく、まず実行手順を明示的なワークフローとして生成し、その内容を事前に定めたうえで静的に解釈しながら必要なプラグインを順次呼び出して処理を実行するように開発した（図 1）。

(1) LLM が生成したワークフローを実行する独自のワークフローランタイムである Sapphillon（サフィヨン）を開発した。これにより、Web ページの操作、ローカルファイルの処理、外部機能の呼び出しなどを、自然言語の指示から一連の処理として実行できるようにした。
(2) ワークフローから利用できる機能を拡張可能にするため、プラグインランタイムを実装するとともに、プラグインシステムを設計した。ブラウザ制御、ファイルシステム、検索、光学的文字認識（OCR）、ローカル LLM 連携などの機能をプラグインとして提供し、用途に応じて機能を追加できる構成とした。
(3) ワークフローとプラグインはそれぞれ独立したサンドボックス環境で、システムと隔離して実行する設計とした。さらに、権限に基づいてアクセス可能な機能やリソースを細かく制御することで、柔軟性を保ちながら安全に動作する実行基盤を構築した。
(4) Floorp Web ブラウザ側にも改修を行い、生成したワークフローからブラウザを直接操作できる仕組みを実装した。あわせて、自然言語入力からワークフローの生成・実行・結果確認までを扱うユーザ向け UI を整備し、Web とローカル OS を横断する処理を分かりやすく利用できるようにした。

以上の機能を Floorp Web ブラウザに統合することで、ブラウザ操作とローカル OS 操作を一つの流れとして扱える構成を実現した。開発した Floorp OS の技術スタックを図 2 に示す。

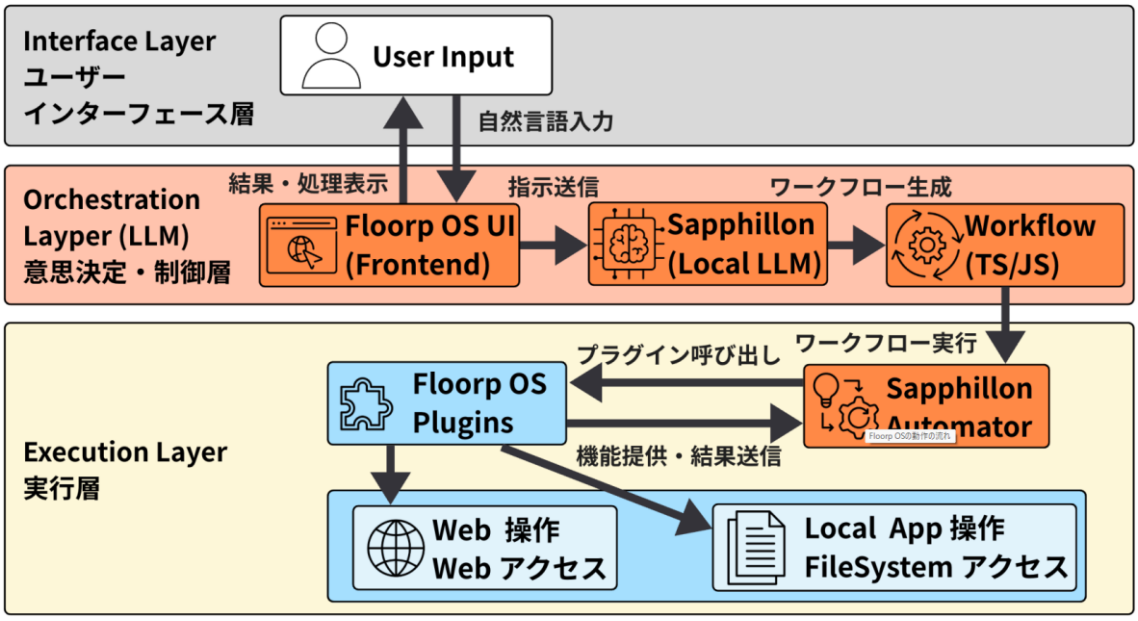


図 1：Floorp OS の操作の流れ

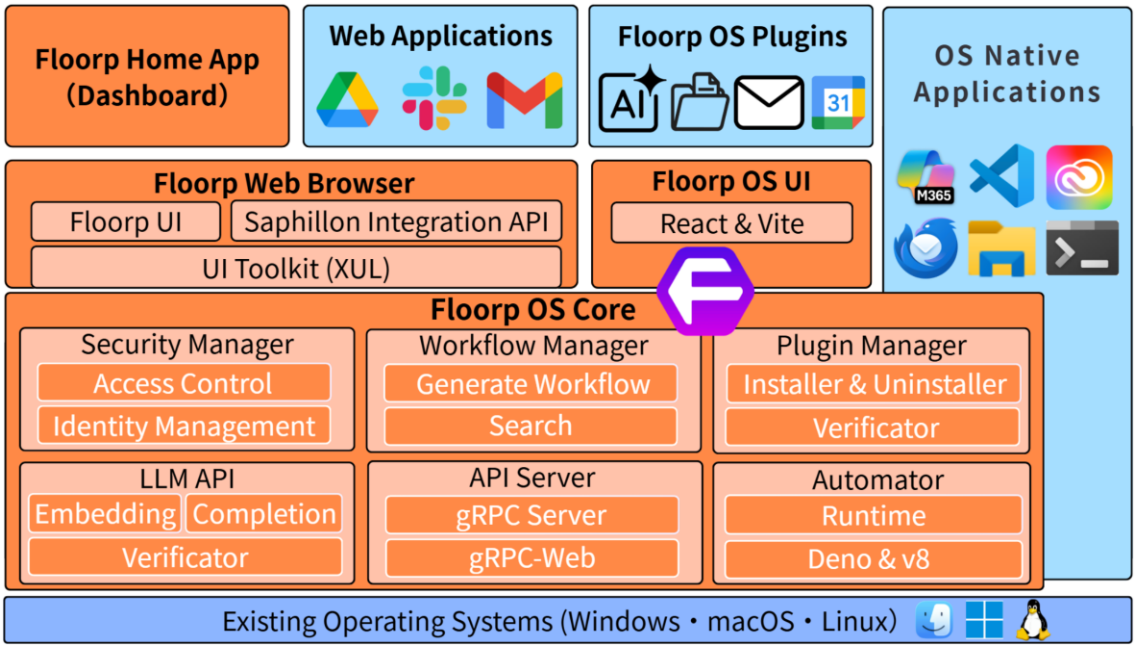


図 2：開発した Floorp OS の技術スタック

## 10. プロジェクト評価

一連のシナリオが一気通貫して動くまでに時間がかかったが、Web ブラウザとローカル OS で分断されていた実行環境を、LLM を中核とした統合プラットフォームとして再構成するという当初の目標は達成できた。自然言語の指示を直接 LLM が実行するのではなく、一度 JavaScript のワークフローへ変換し、それを安全に実行するための基盤として Sapphillon を設計し実装し切った点は、大きな技術的成果であると考える。プロジェクト採択後に Atlas（OpenAI）や

Comet（Perplexity）、Dia（The Browser Company）など、各社 AI ブラウザが市場に展開されていったが、ローカルファイルと Web を統合してワークフローを安全に実行できるという点で Sapphillon の設計の優位点が目立った。プロジェクト終盤では OpenClaw が登場し、大きな人気を得たが、セキュリティ上の理由からその導入には普段使いの PC とは異なる Mac mini を専用マシンとして使わないといけないなど、この問題に対してきちんとアプローチできているものは少ない。例えば、Claude Code で業務を進める際、実行中に AI から都度確認を求められたら承認する、もしくは却下して改善案を伝えるなどの関わり方が必要となっており、AI 実行の安全性を高めると人間のストレスが増えるようになっている。その点、Floorp OS はワークフロー実行直前に一括して権限リストとリスクを表示し、一回の承認で済むように UI 上の工夫もされている。また、LLM が生成したコードを安全に実行するために、Deno および V8 isolate を用いたサンドボックス実行環境を構築し、権限制御を組み込んでおり、きちんとリスクを踏まえた設計がされていると言える。LLM を実運用環境に組み込む上で本質的な課題に正面から取り組んだ成果である。

## 11. 今後の課題

一通り動く仕組みは出来上がっているため、今後は、プラグインストアの公開や開発者向け SDK の整備が必要である。今回開発したプラグインシステムは Controller・Core・Plugin Manager・Workflow Manager といった責務分離が明確な構成で、拡張性と再利用性を意識したアーキテクチャ設計ができているため、オープンソースの開発コミュニティとして新しくプラグインの開発が進むようになると、開発、事務、調査、研究支援など幅広い領域への応用が期待できる。

LLM の進化のスピードは速く、プロジェクト終了後に Gemma4 E2B や 1-bit Bonsai 8B などエッジ向けのローカル LLM も小さいサイズで性能が良いものが出ているため、LLM によるワークフロー生成の精度や安全性を定量的に評価する仕組みが不可欠である。LLM が生成するワークフローの品質は一定ではないため、ワークフロー実行における耐障害性やエラーハンドリングなど異常系を含めた実行管理の強化が必要と考える。

また、マルチモーダル LLM も実用的なレベルで登場してきているため、表示画面の内容や画面操作を認識するなど新しい体験の拡張も今後の発展課題として考えられる。