

Beyond pip install: Evaluating LLM agents for the automated installation of Python projects

MILLIKEN Louis¹, KANG Sungmin², YOO Shin¹

¹*Korea Advanced Institute of Science and Technology,*

²*National University of Singapore*

The 32th IEEE International Conference on Software Analysis, Evolution and Reengineering
(SANER2025)

目次

研究背景

目的・アプローチ

実験設定

結果

まとめ

目次

研究背景

目的・アプローチ

実験設定

結果

まとめ

研究背景

- | LLMのコーパスにソースコードが含まれている場合、そのLLMはソフトウェア関連のタスクを実行可能
- | 従来は関数の合成、単体テストの生成といった**小規模**なタスクを実行させることが多かった
- | しかし、Chain-of-thoughtやReActといった高度なプロンプト技術や、LLMベースのアーキテクチャ等により**リポジトリレベル**のタスクが実行できるようになってきている。

研究問題：LLMエージェントと リポジトリレベルのタスク

- | LLMエージェントの実行能力が研究されている
リポジトリレベルタスク
- リポジトリレベルコーディングタスク
- リポジトリレベルコード生成タスク
- | 今まで研究されてきたものはコード管理タスクに
寄っている.
- 環境管理タスクの実行能力に関してはあまり研究され
ていない

研究背景：インストールタスク

- | 本論文では環境管理タスクの1つとしてインストールタスクを挙げている.
- Dagenaisらはソフトウェアプロジェクトに新しく参加する人にとって、不十分なdeveloperドキュメントが障害になることを発見した.
- Aghajaniらの調査では、開発者の63%が、不適切なインストール手順が共通の問題であると主張している.
- 自動インストールを試みる使用できるツールは、開発者のフラストレーションを軽減し、生産性を向上させるために必要

目次

研究背景

目的・アプローチ

実験設定

結果

まとめ

研究目的とアプローチ

- | **研究目的**：リポジトリレベルのインストールタスクの実現可能性の調査
- | **アプローチ**：環境管理タスクの1つである「コードリポジトリのインストール」をLLMベースのエージェントであるINSTALLAMATICに実行させ、それを用いて評価を行う
- 評価には、github上で公開されている40のPythonプロジェクトを元に、本研究で作成されたデータセット(後述)を使用

リポジトリレベルのインストールタスクの実現可能性を調査

本論文におけるインストールタスクの流れ

① インストールに関連するドキュメントの収集

- README.mdやINSTALL.mdのようなインストールに関する説明があるドキュメントを収集

② リポジトリをインストールおよび、インストールが正しく行われたかテストするためのDockerfileを構築する.

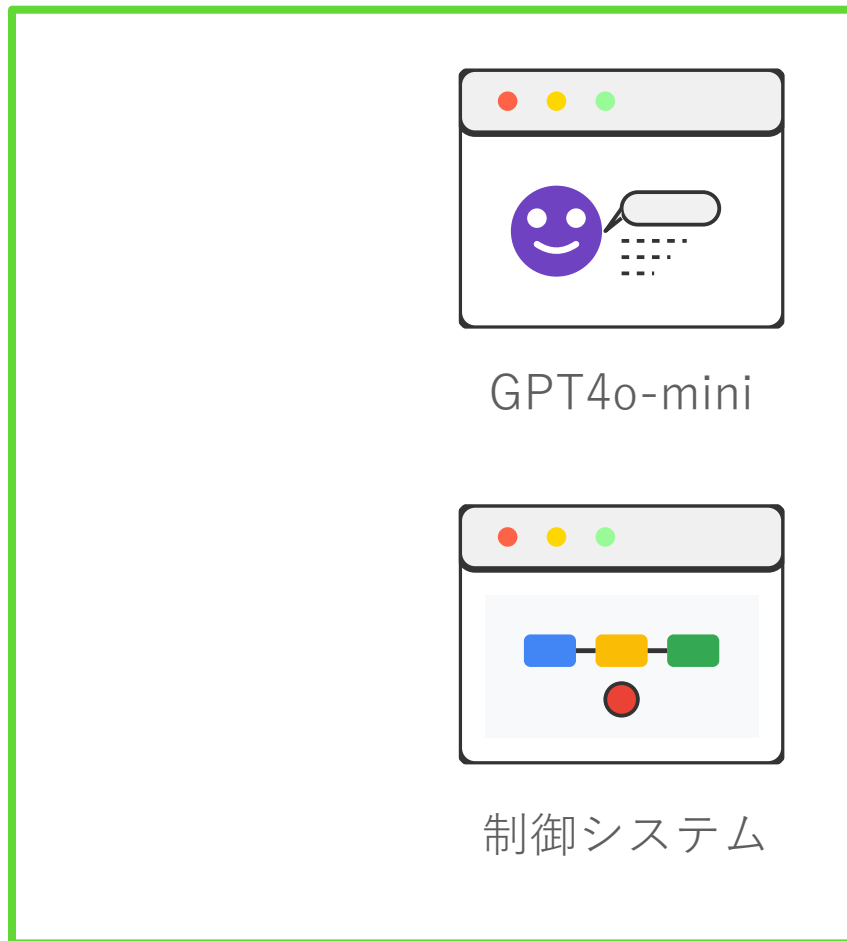
- Dockerfileは、仮想OSを選択した上で、そのOS上で使えるコマンドを実行できるファイル
- テストスイートはリポジトリに備え付けられているものを流用

具体例

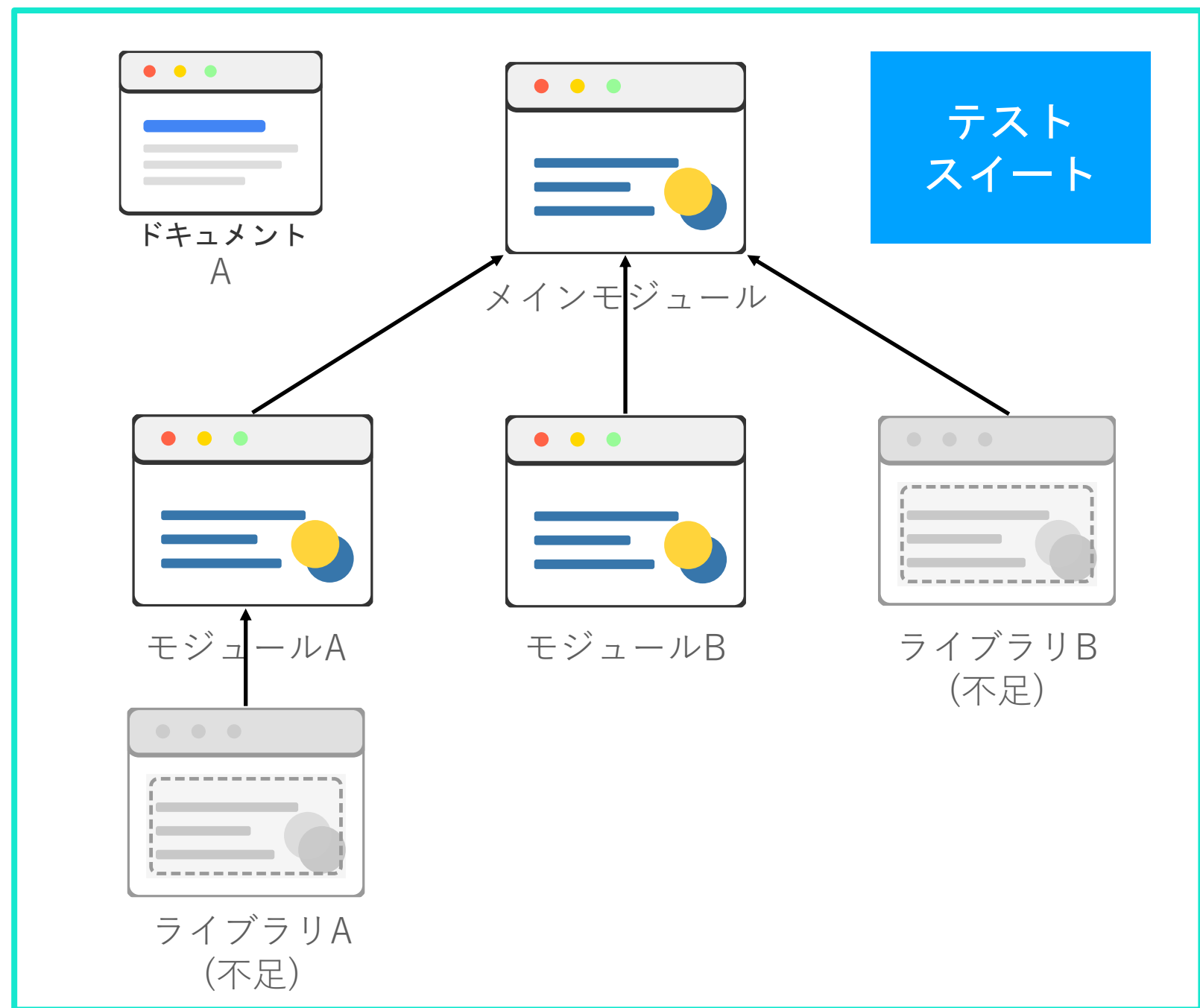
| Pythonプロジェクトを中心としたコードリポジトリを
想定

具体例

INSTALLAMATIC



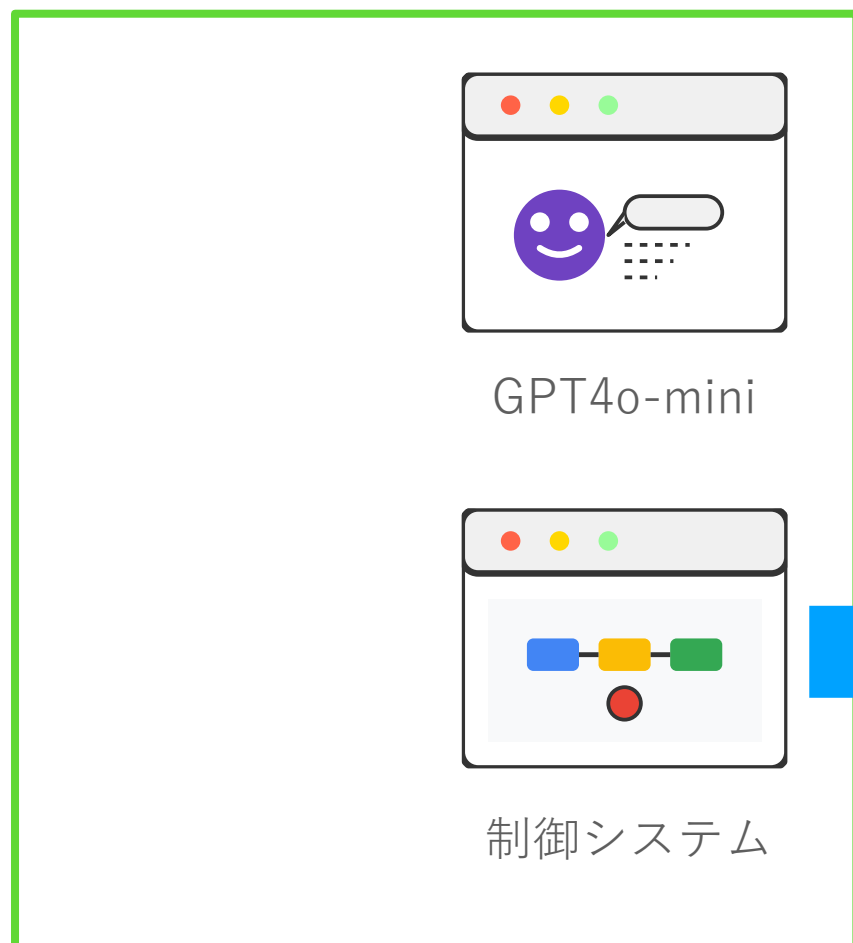
リポジトリ



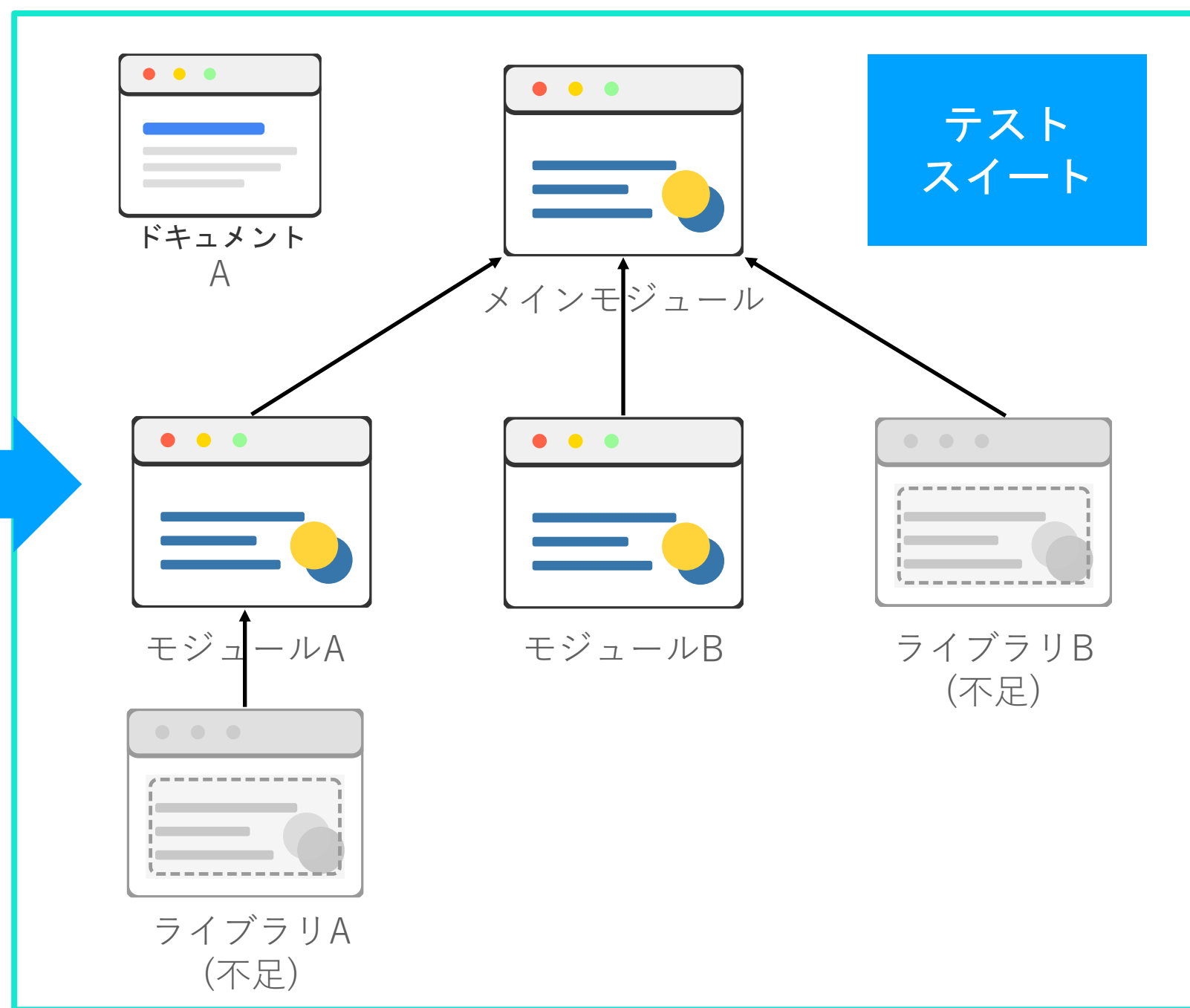
具体例

INSTALLAMATIC

リポジトリ

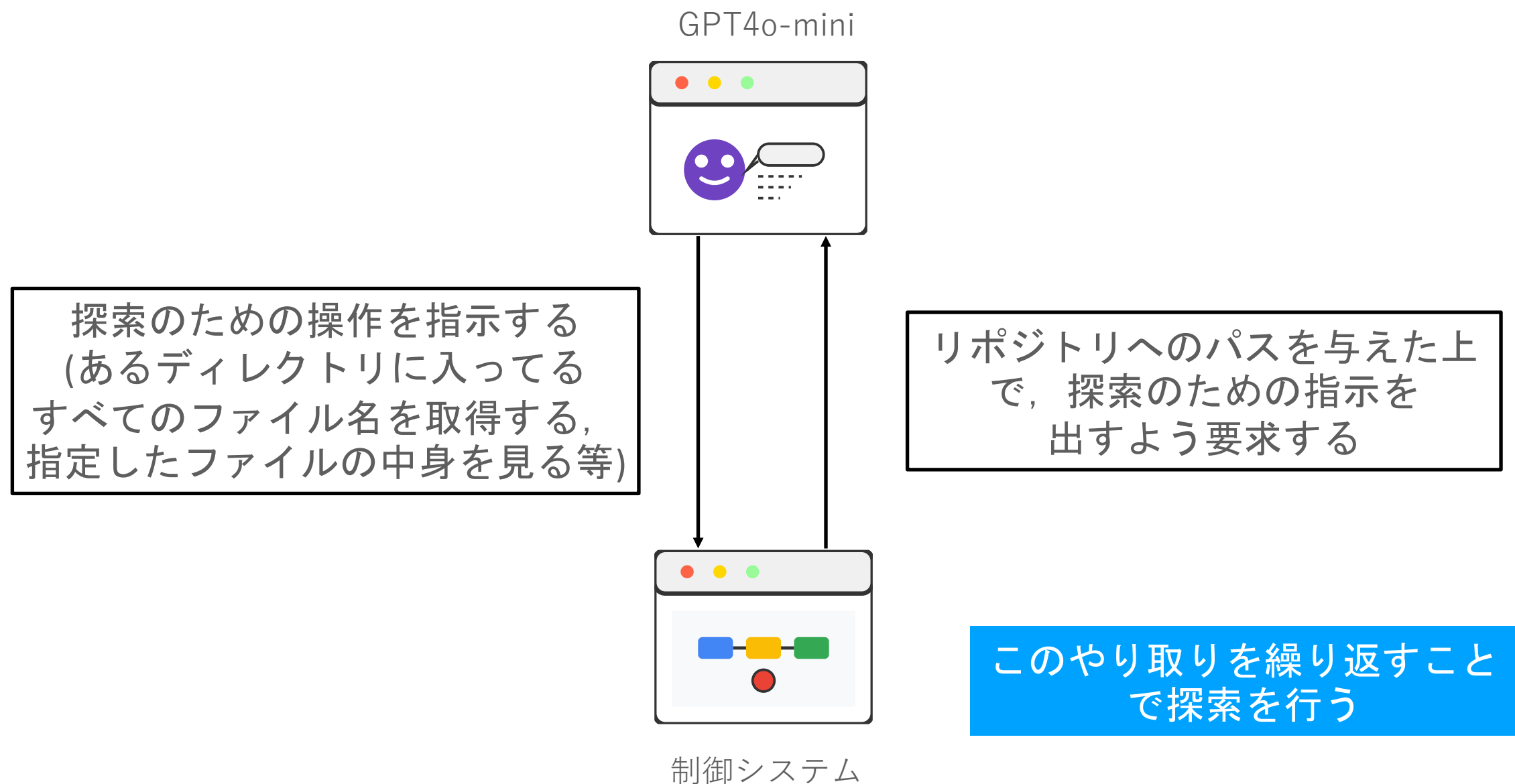


ダウンロードに関する記述
のあるドキュメントを
リポジトリから探す



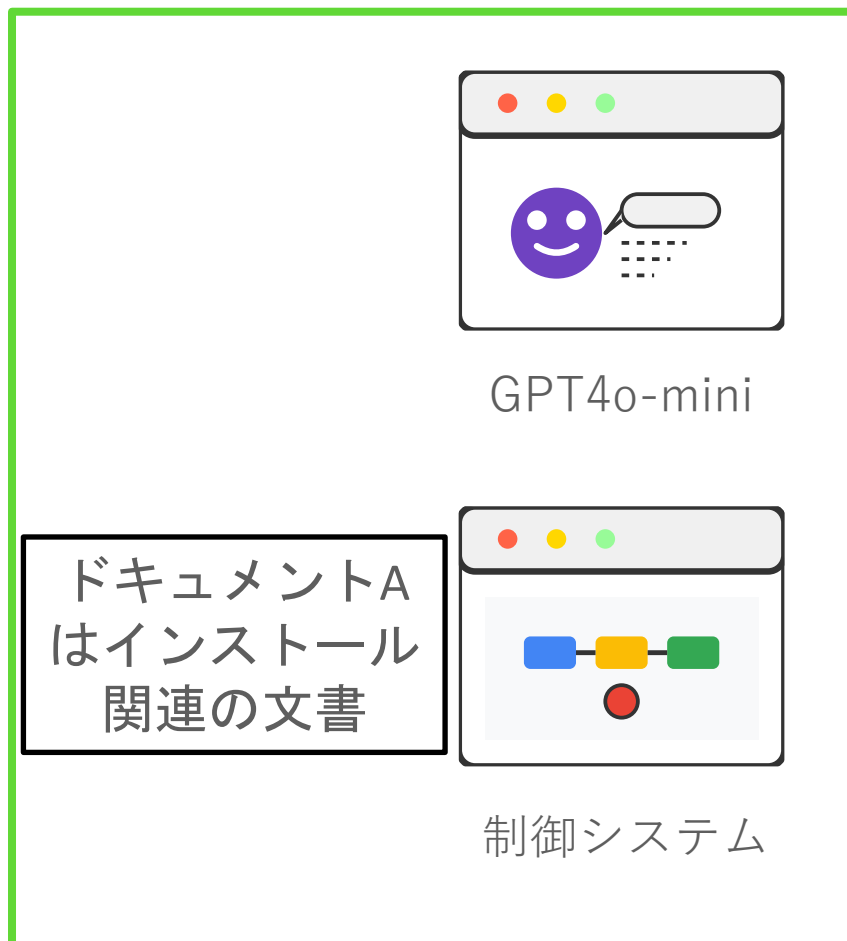
探索法について

- ある特定のアルゴリズムに基づく探索ではない
- LLMの指示に基づき，探索を行う



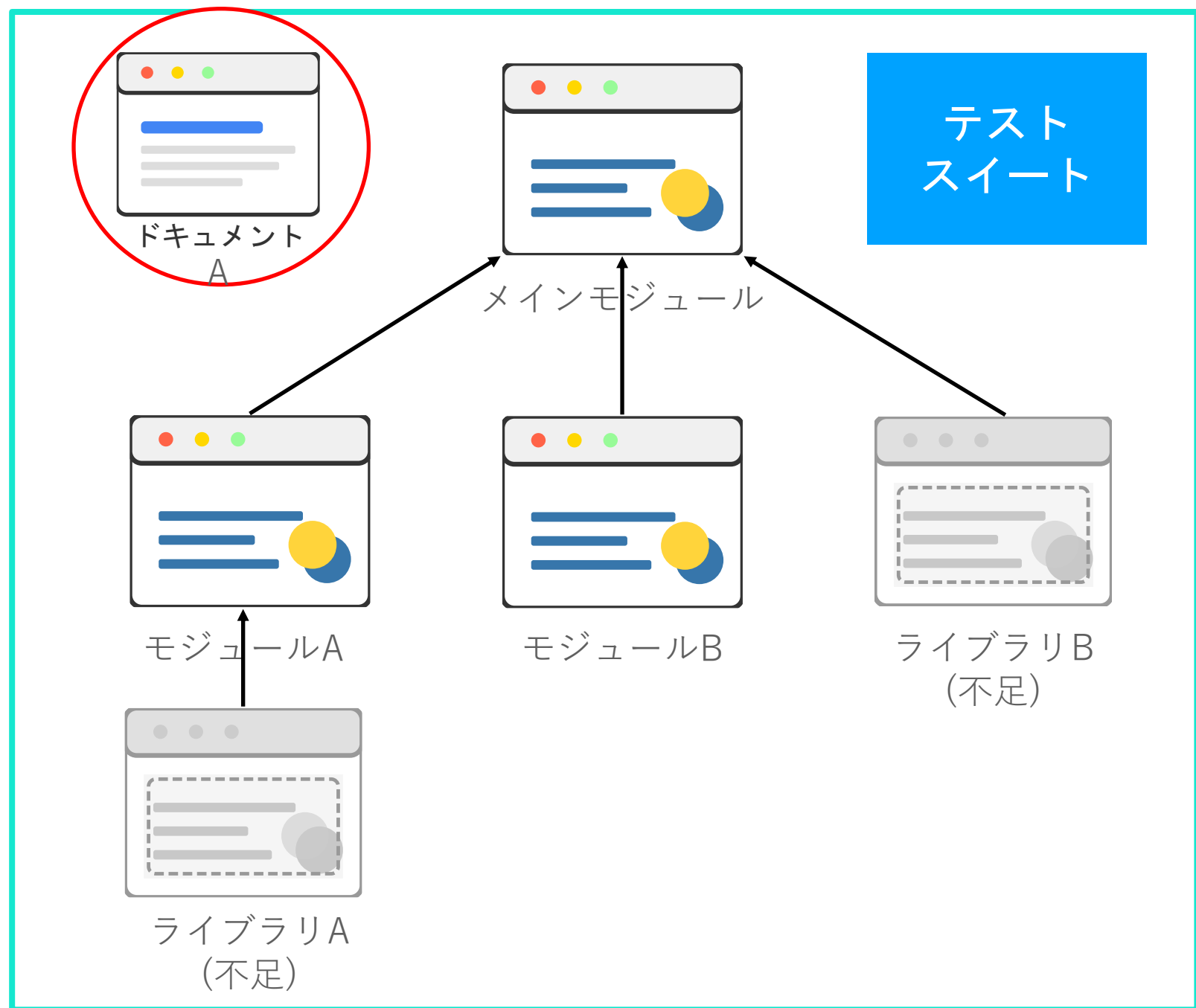
具体例

INSTALLAMATIC



発見したドキュメントを
インストール関連の文書
として記録

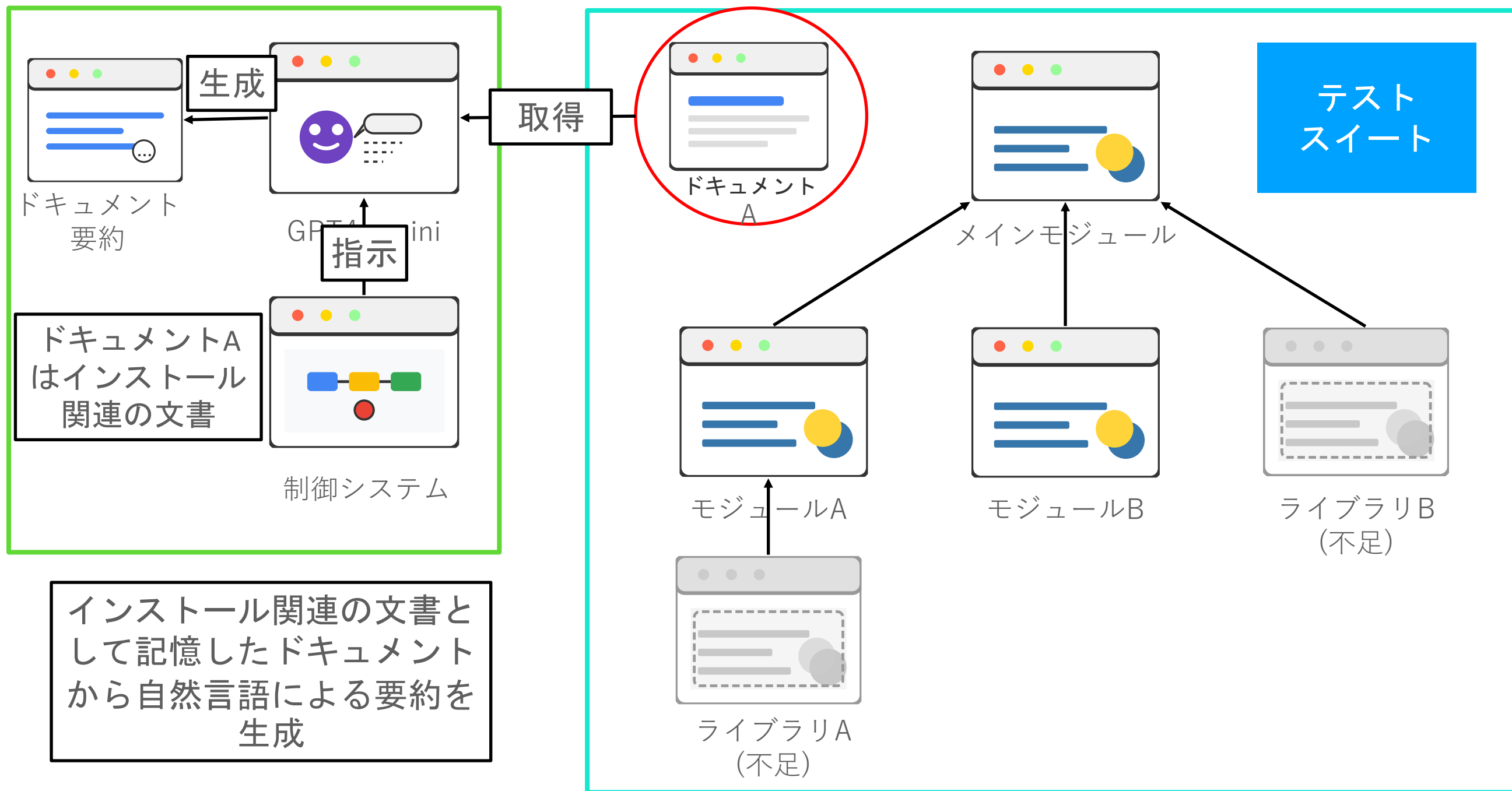
リポジトリ



具体例

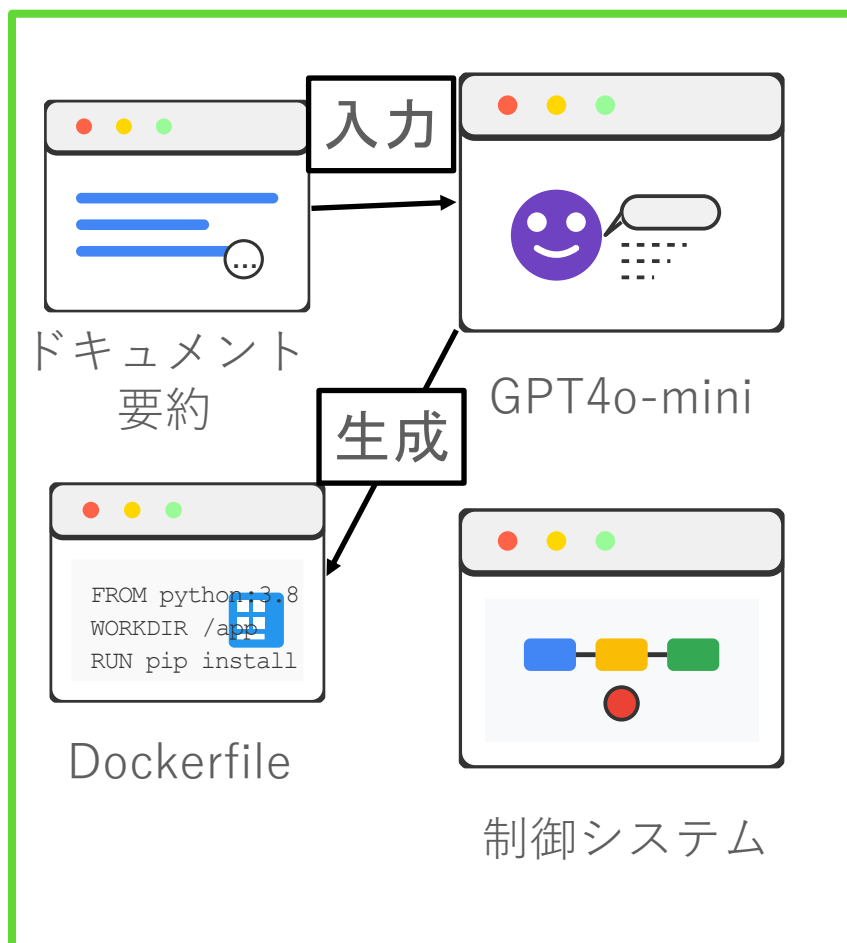
INSTALLAMATIC

リポジトリ



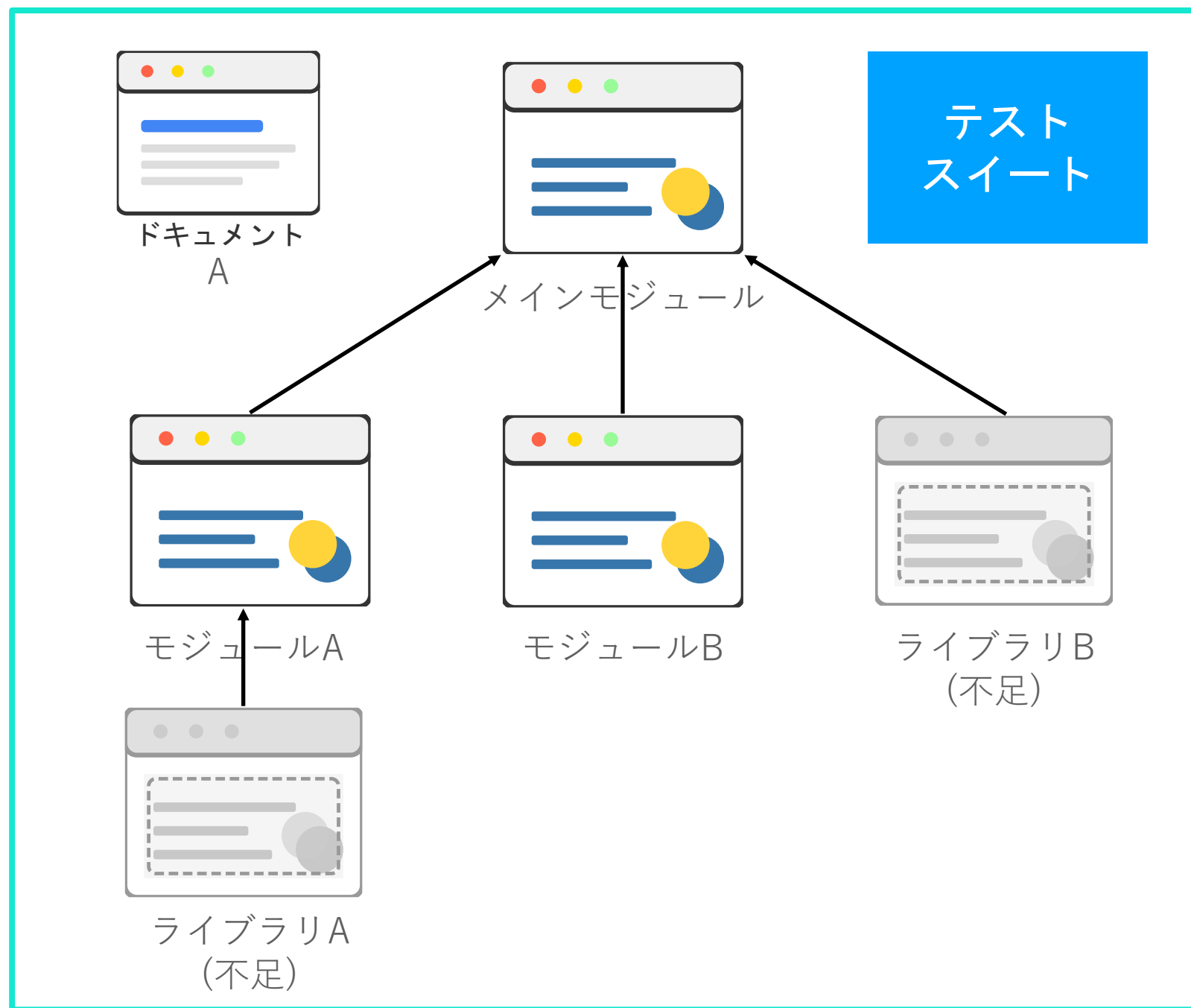
具体例

INSTALLAMATIC



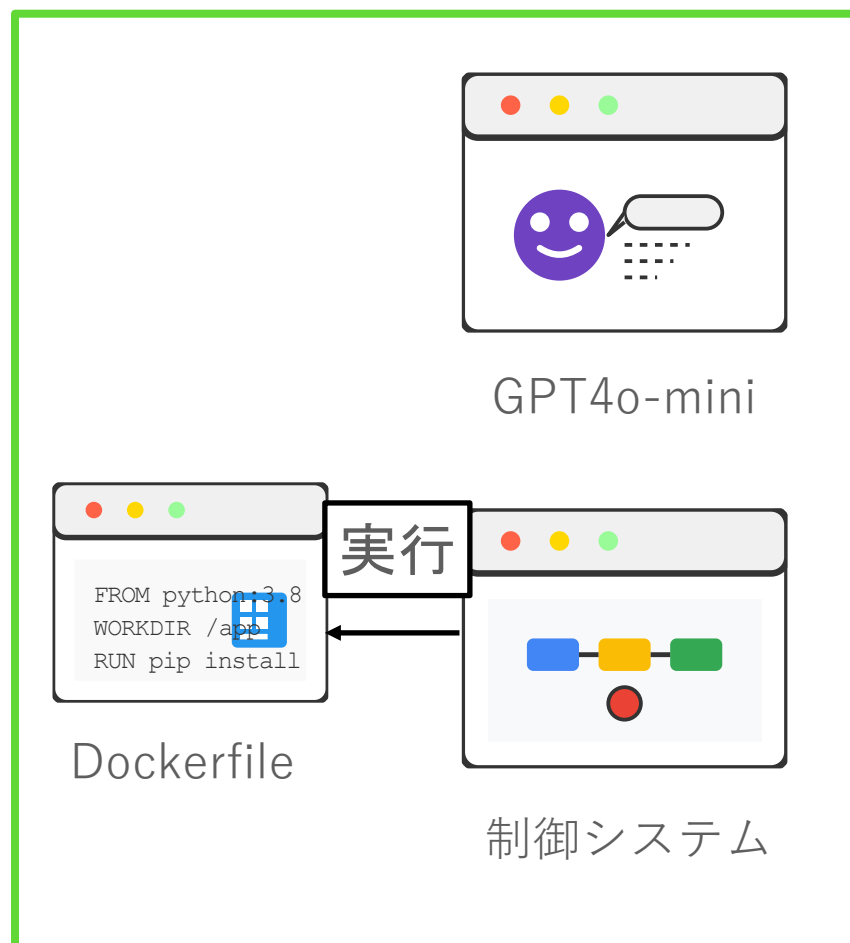
ドキュメントの要約をLLMに入力し、Dockerfileを生成する

リポジトリ



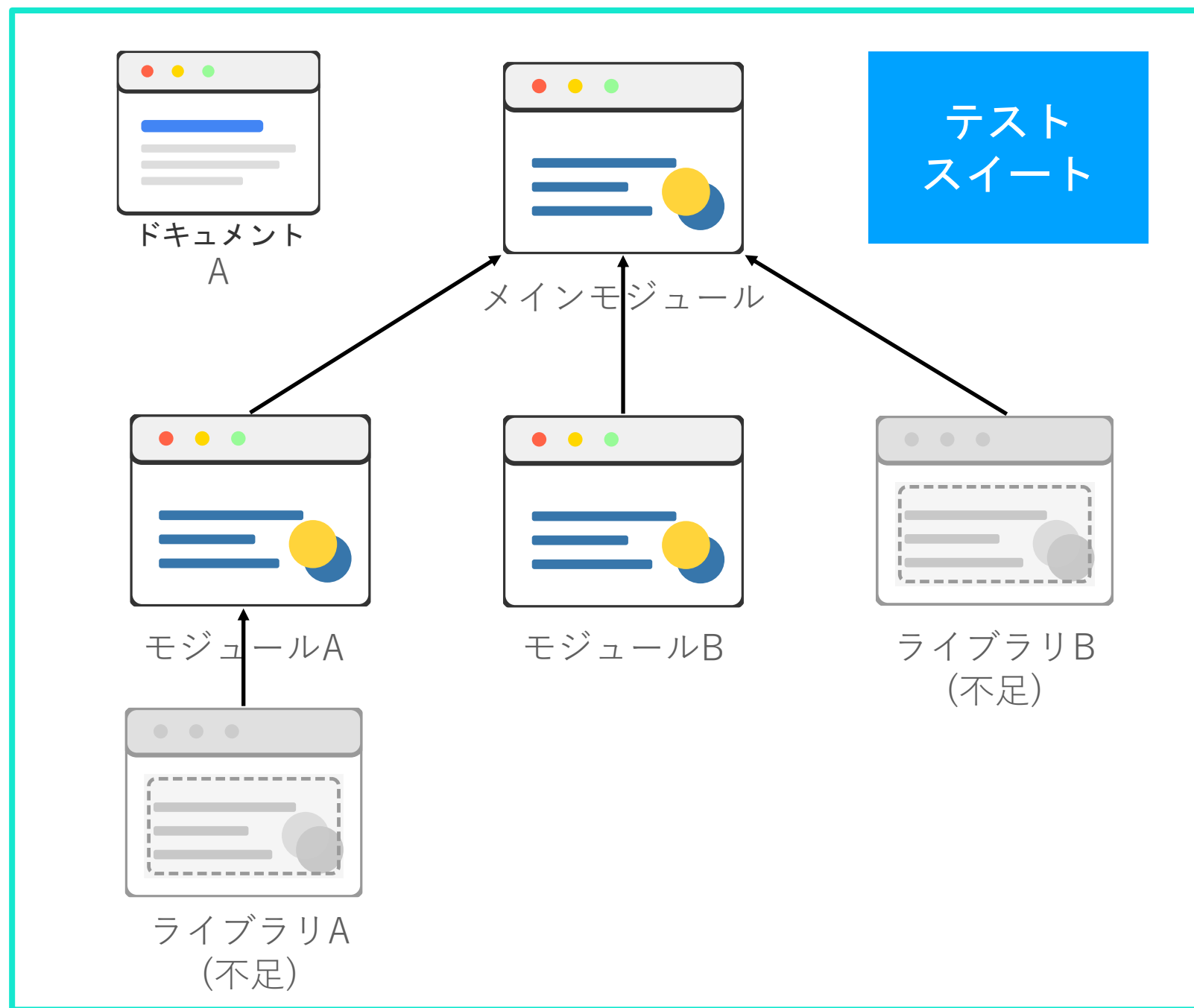
具体例

INSTALLAMATIC



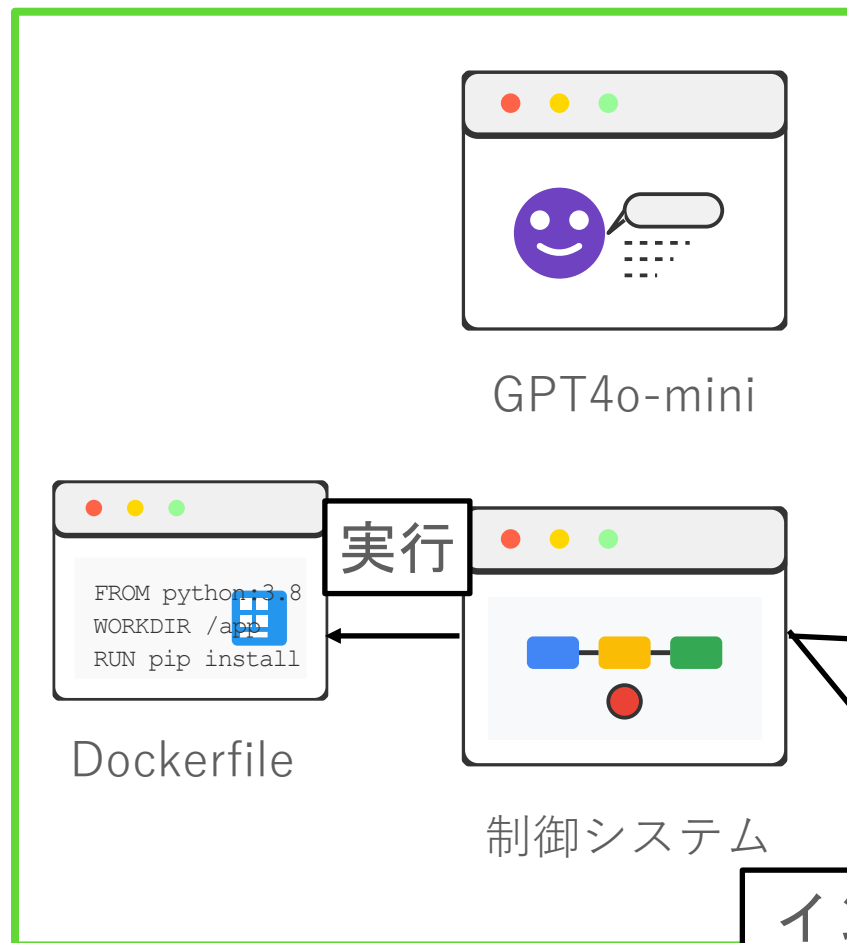
生成されたDockerfileを
実行する

リポジトリ



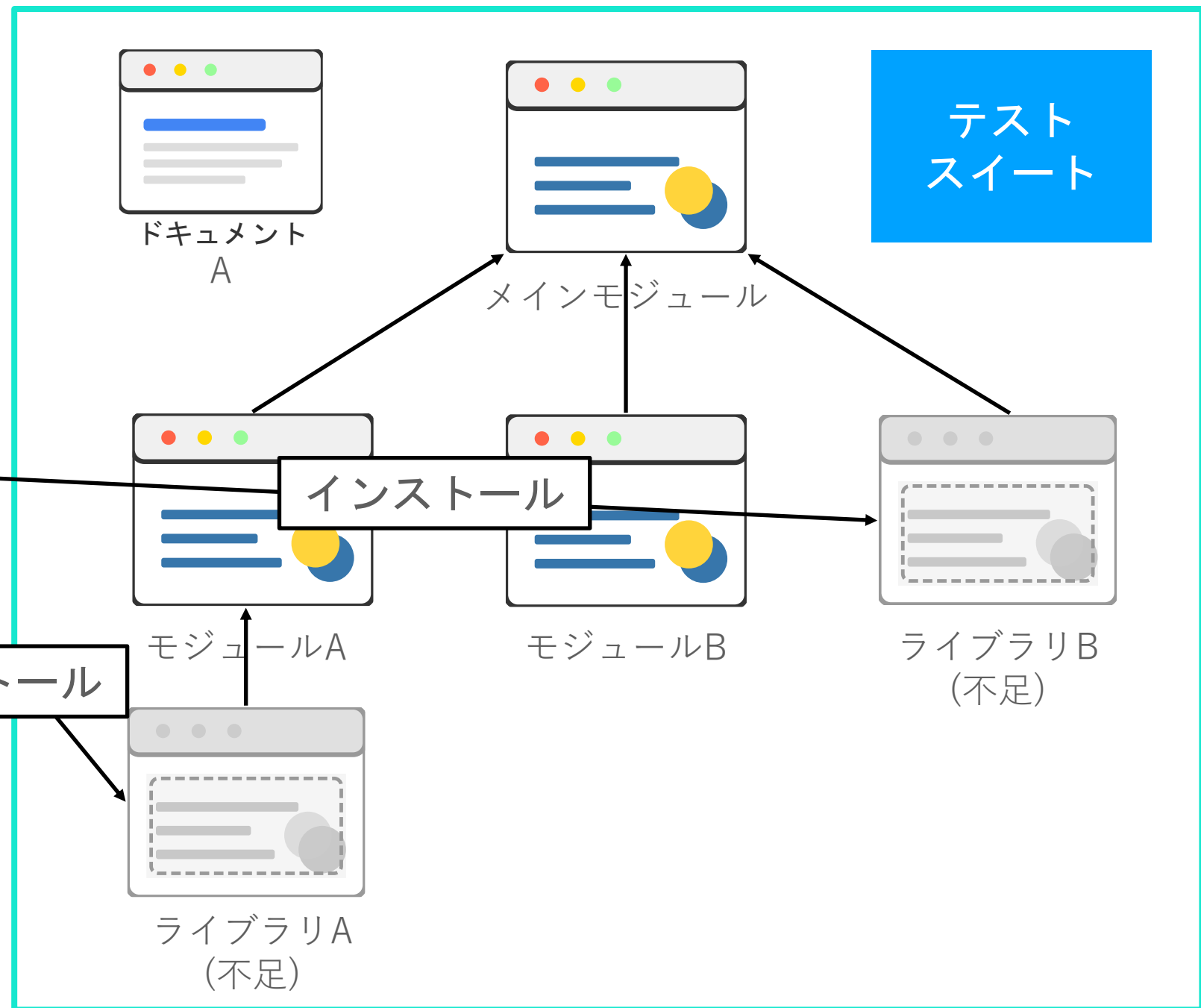
具体例

INSTALLAMATIC



(正しいDockerfileなら)
依存関係のインストールを
行う

リポジトリ



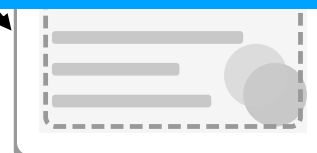
具体例

INSTALLAMATIC

リポジトリ

ここでインストールが成功するか失敗するかで
処理が分岐する

(正しいDockerfileなら)
依存関係のインストールを
行う

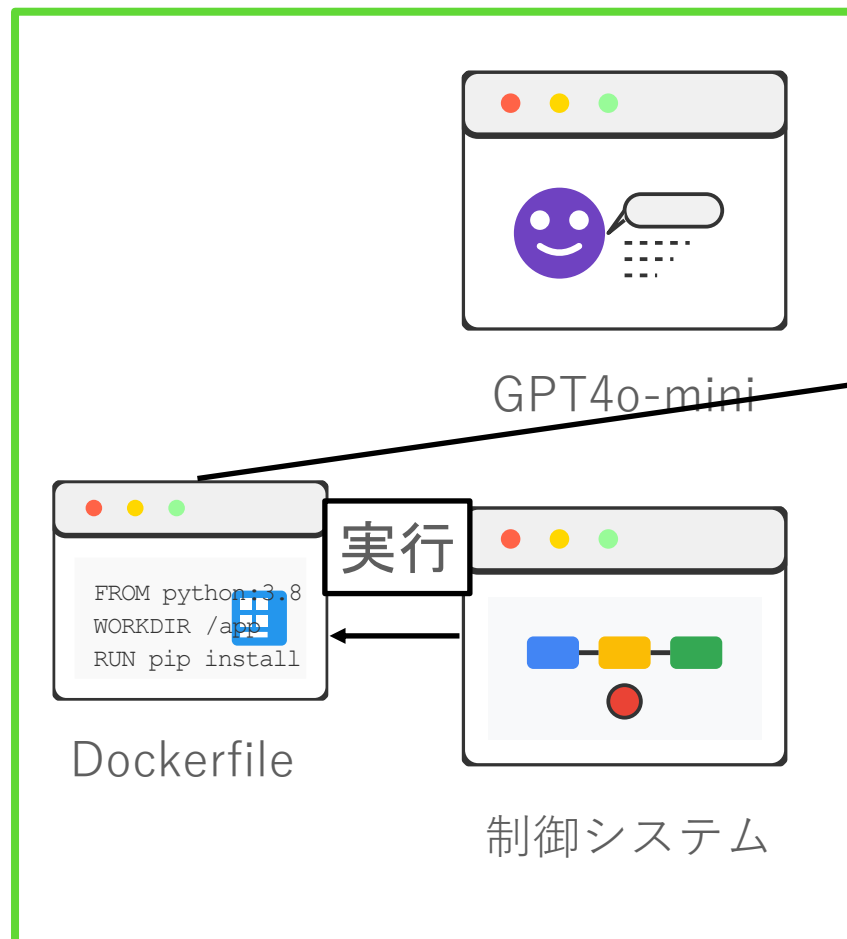


ライブラリA
(不足)

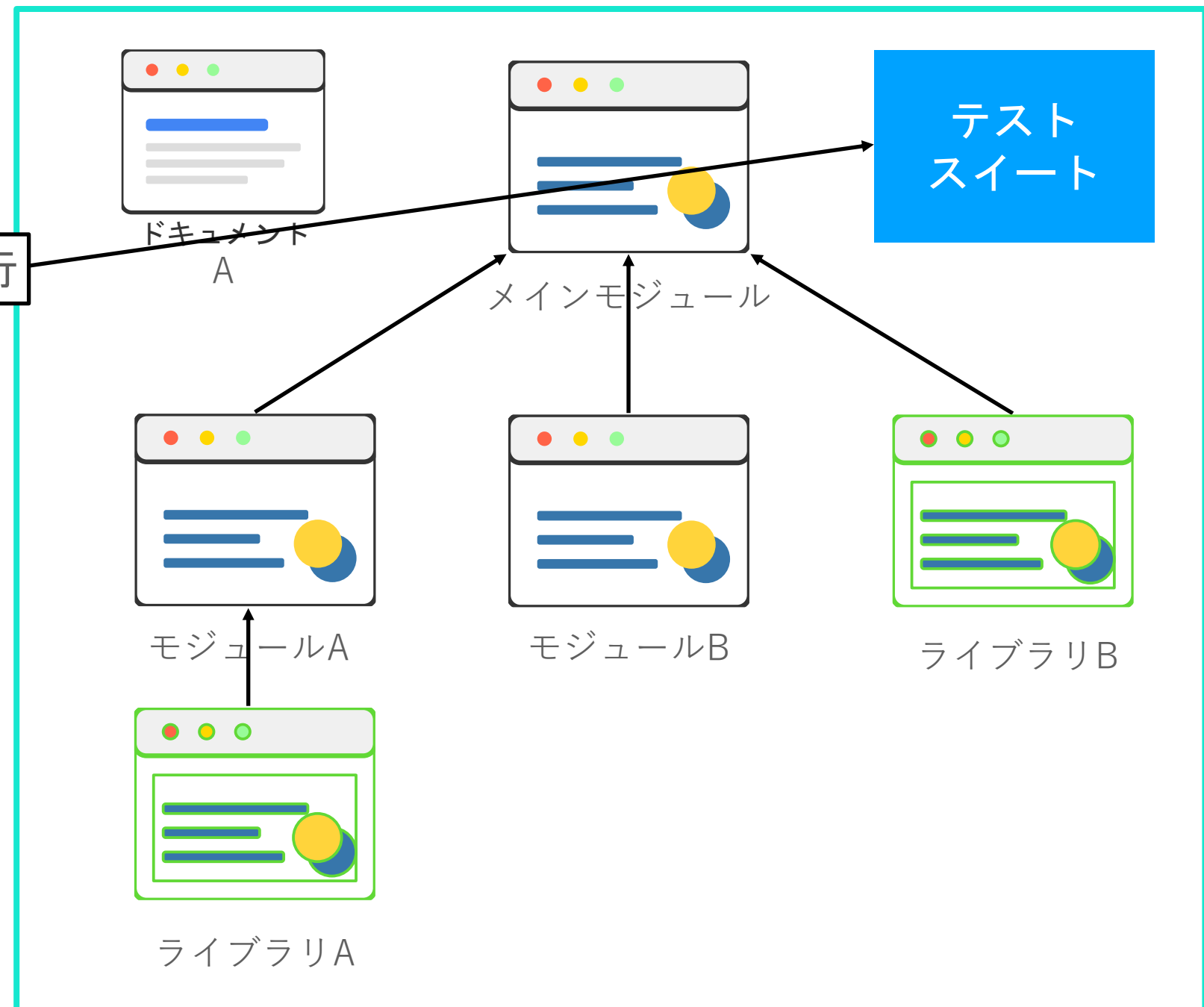
具体例(インストール成功)

INSTALLAMATIC

リポジトリ

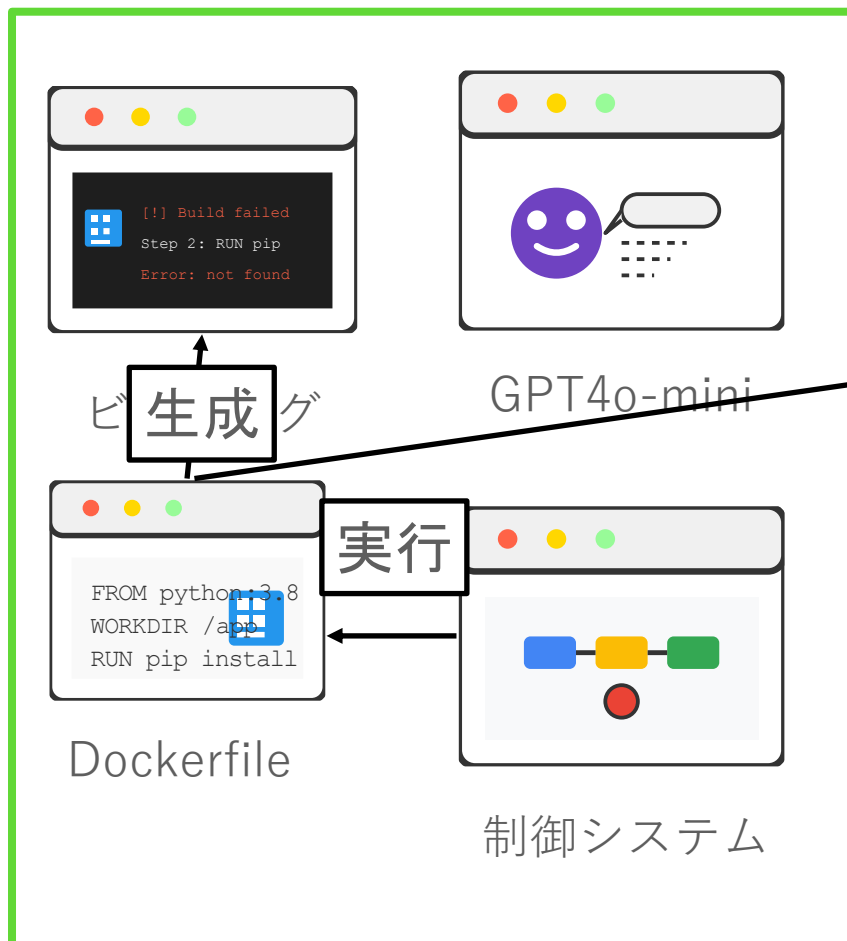


Dockerfileに記述されている
複数のテストを実行し、
そのうち1つでも通れば
インストール成功とみなし
処理を終了

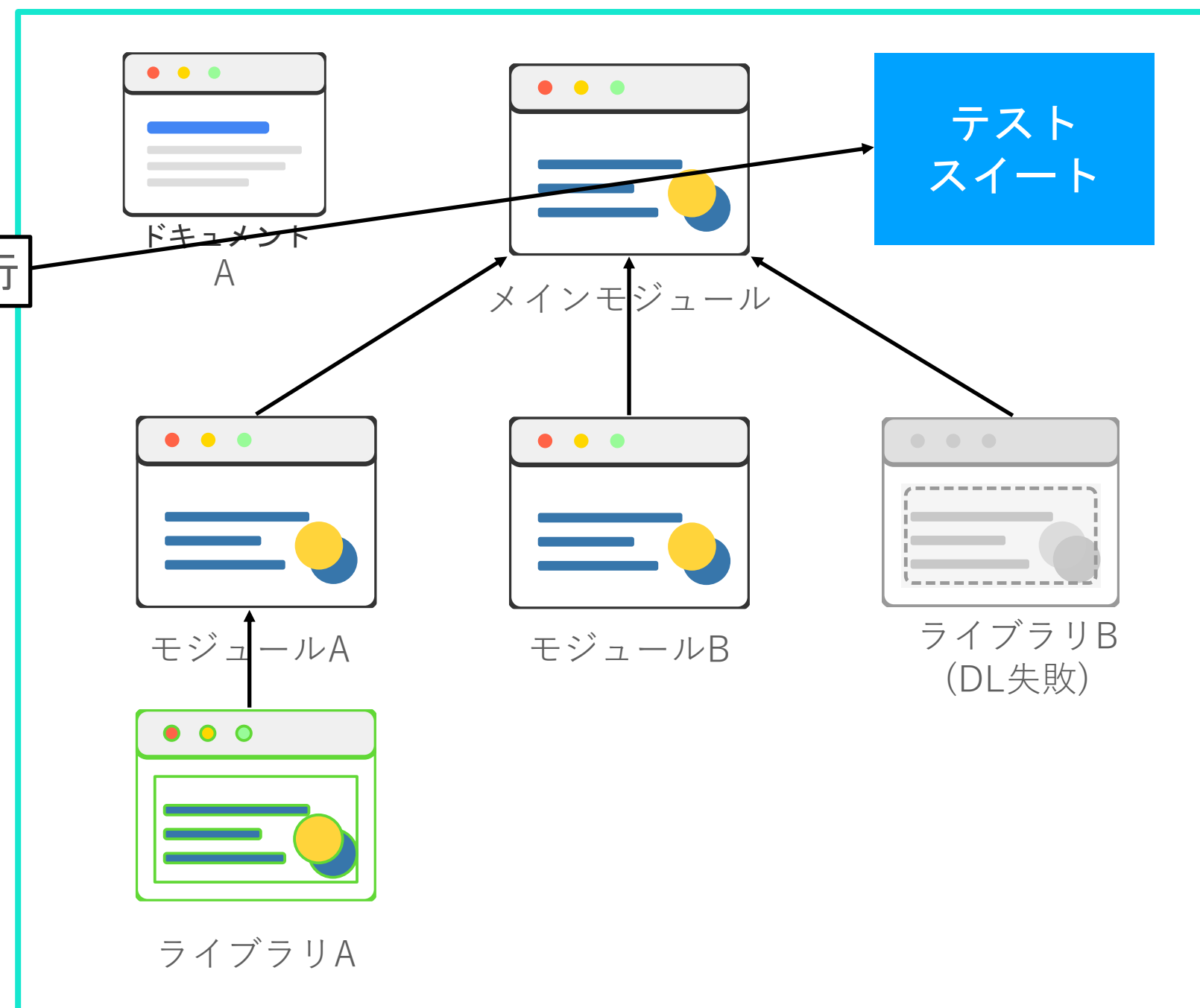


具体例(インストール失敗)

INSTALLAMATIC



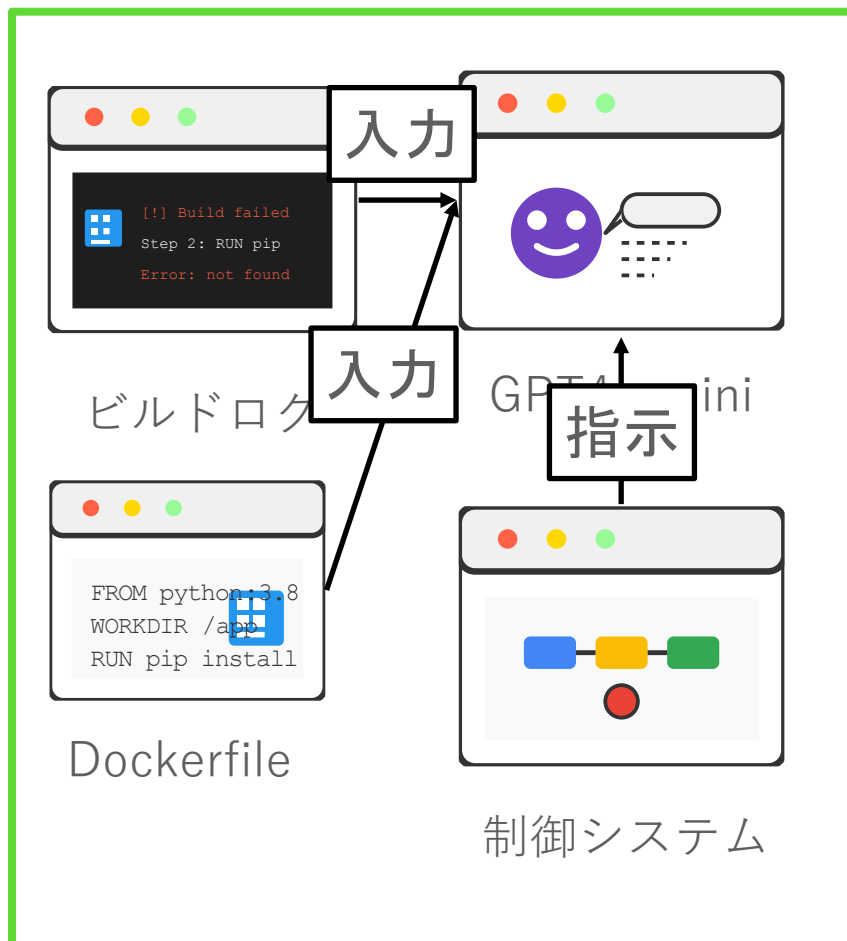
リポジトリ



もしテストが1つも通らない,
もしくはDockerfileの実行が途中
で終了した場合, インストール
失敗とみなす

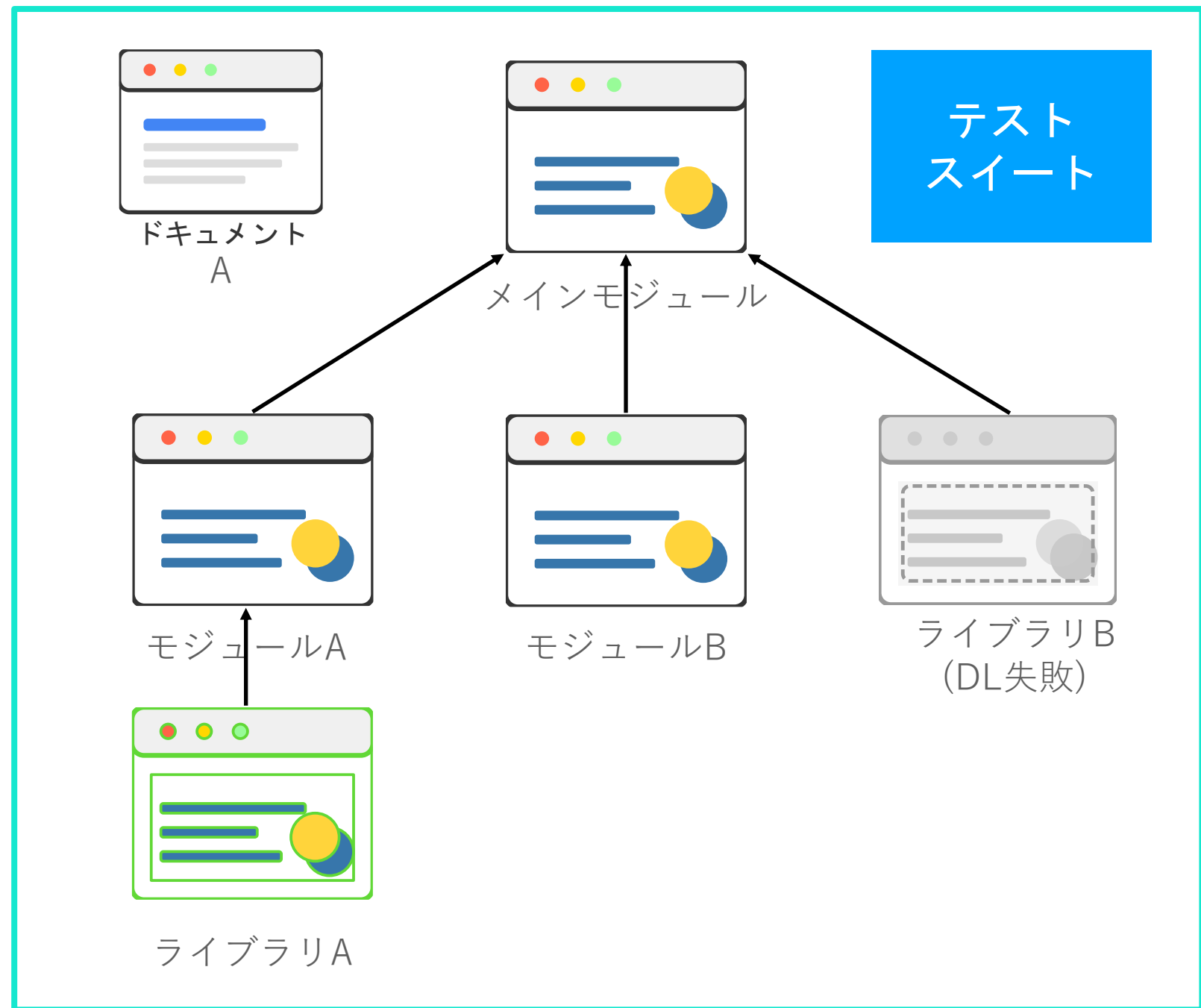
具体例(インストール失敗)

INSTALLAMATIC



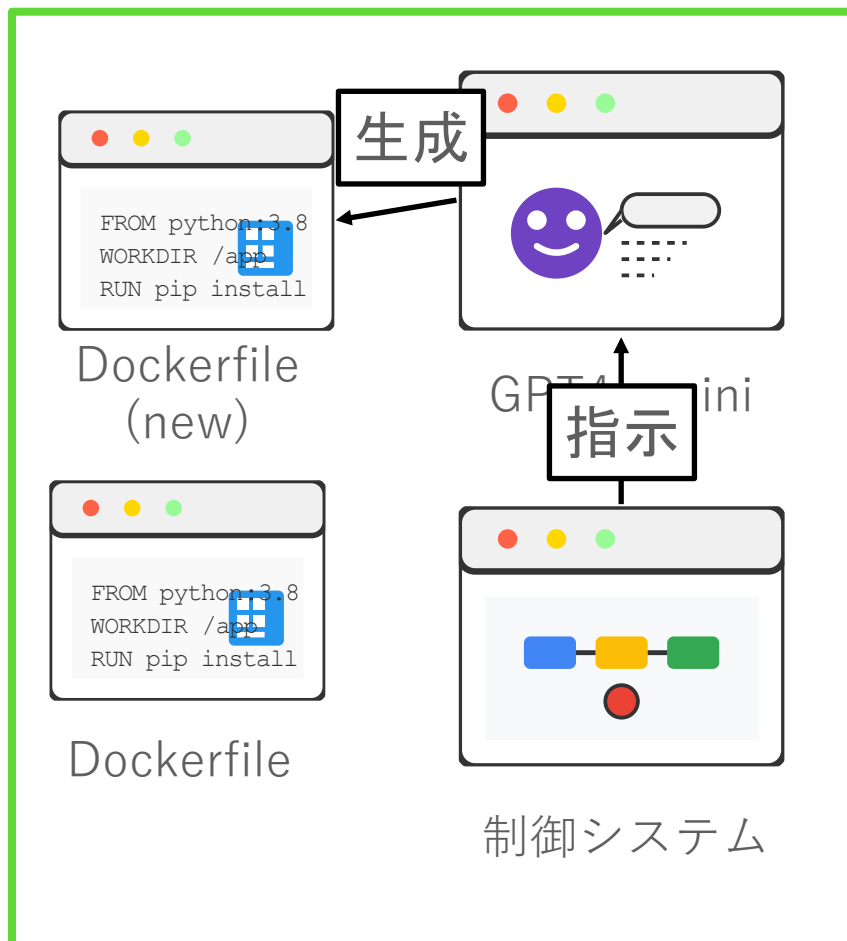
ビルドログとDockerfileを
LLMに与えた上で、LLMに
エラーの原因特定
を指示する

リポジトリ



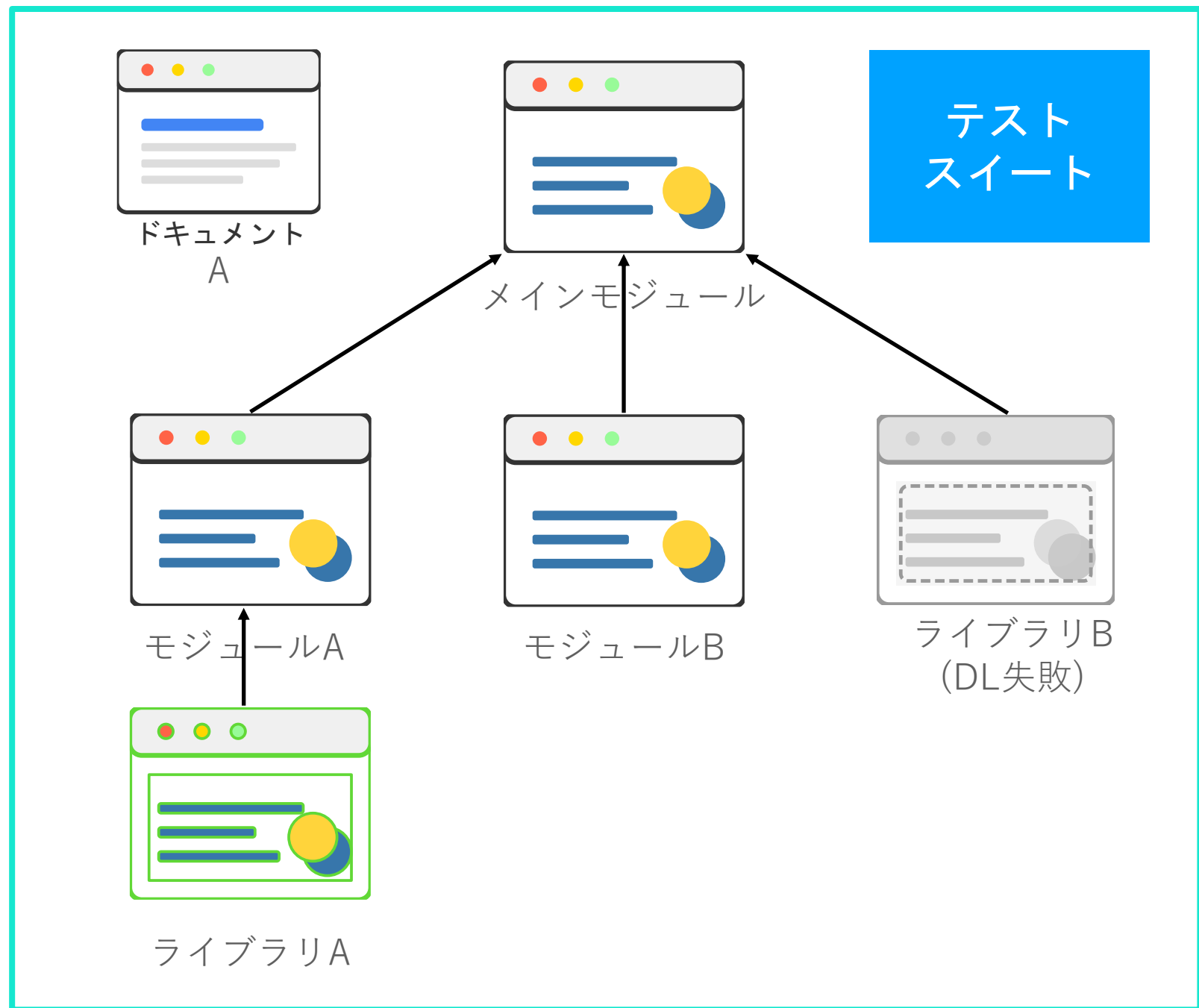
具体例(インストール失敗)

INSTALLAMATIC



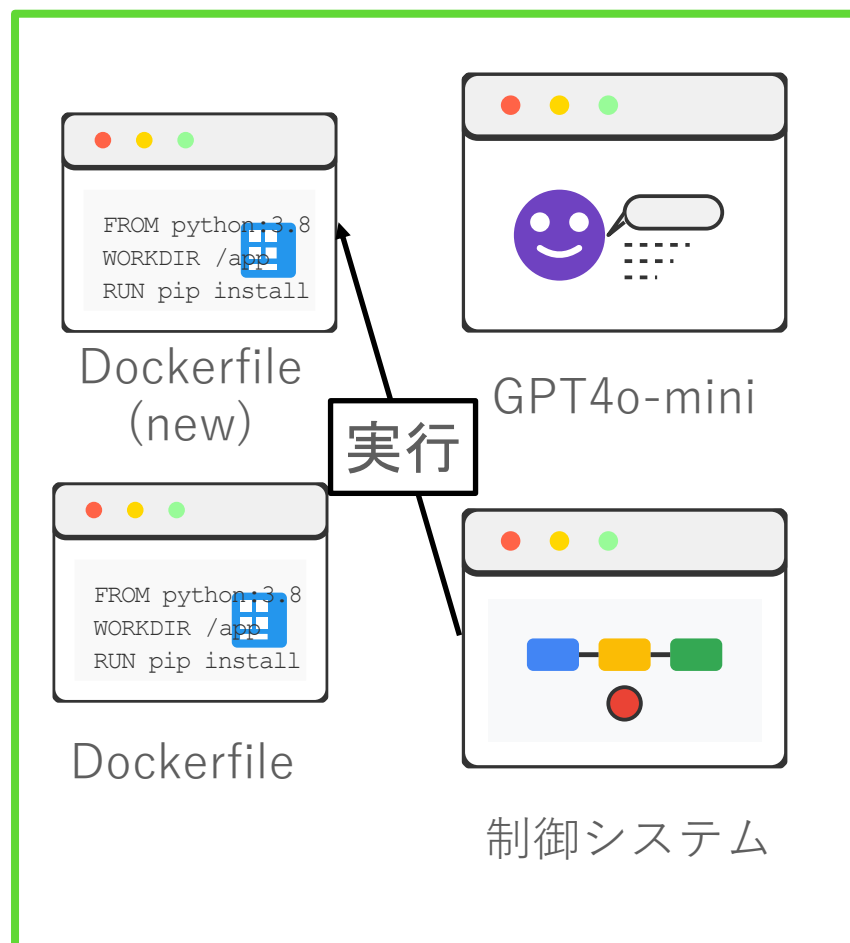
LLMがエラーの原因を特定した後、エラーの修正方法及び、その方法で修正されたDockerfileの生成を指示する

リポジトリ



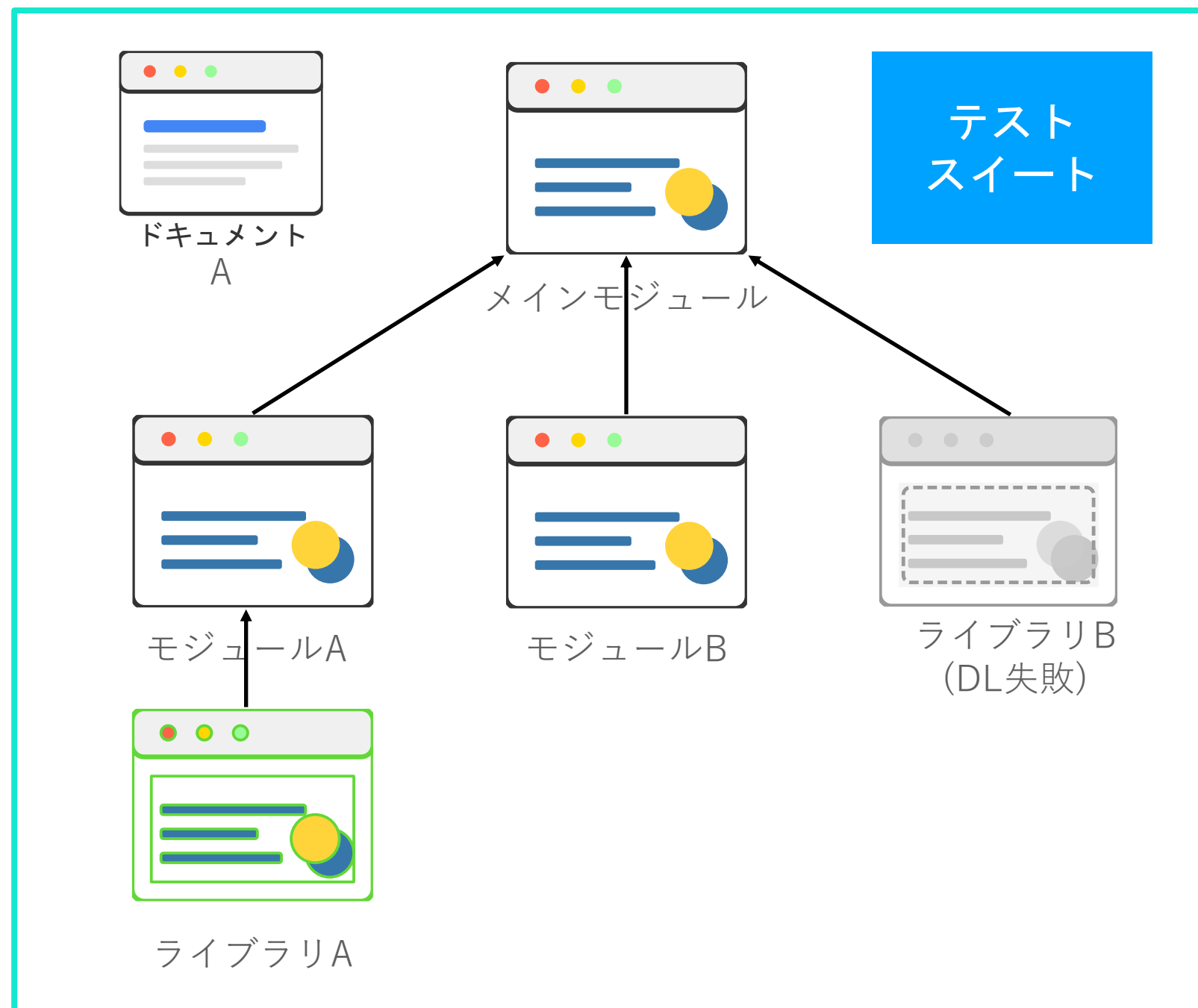
具体例(インストール失敗)

INSTALLAMATIC



新たに生成されたDockerfileを
実行する

リポジトリ



具体例(インストール失敗)

INSTALLAMATIC

リポジトリ

インストール成功→処理を終了
インストール失敗→再びインストール失敗時の
処理を行う
修復試行の時間コストの関係で、修復試行は最大で2回
までしか行われない。

新たに生成されたDockerfileを
実行する



ライブラリA

目次

研究背景

目的・アプローチ

実験設定

結果

まとめ

実験設定

- | 通常通りLLMがドキュメント収集を行う**標準構成**と、INSTALLAMATICのドキュメント収集を人手で行った**パーフェクトリコール構成**でそれぞれ実験を行う
- | 各リポジトリに対してINSTALLAMATICを10回実行し、その結果を集計

実験設定：データセット

- | 使用するデータセットはGithubから収集した40のリポジトリから構築
- 2024年8月以降に更新されているリポジトリのみ使用
- 4つの異なるスター(人気度)の範囲でリポジトリを10ずつ選択
- 収集したリポジトリは必ずインストールテスト用のテスト・スイートを持つ
- 各リポジトリは必ずしもインストールに関するドキュメントを持たない

実験設定：データセット

TABLE I: List of repositories in the dataset

★	Name	URL	★	Name	URL
1k-5K	icloud-drive-docker	https://github.com/mandarons/icloud-drive-docker	10k-20K	yfinance	https://github.com/ranaroussi/yfinance
	django-stubs	https://github.com/typeddjango/django-stubs		beets	https://github.com/beetbox/beets
	pennylane	https://github.com/PennyLaneAI/pennylane		starlette	https://github.com/encode/starlette
	X-AnyLabeling	https://github.com/CVHub520/X-AnyLabeling		datasets	https://github.com/huggingface/datasets
	opencompass	https://github.com/open-compass/opencompass		mypy	https://github.com/python/mypy
	R2R	https://github.com/SciPhi-AI/R2R		sympy	https://github.com/sympy/sympy
	Torch-Pruning	https://github.com/VainF/Torch-Pruning		ydata-profiling	https://github.com/ydataai/ydata-profiling
	scvi-tools	https://github.com/scverse/scvi-tools		spotify-downloader	https://github.com/spotDL/spotify-downloader
	sabnzbd	https://github.com/sabnzbd/sabnzbd		qlib	https://github.com/microsoft/qlib
	dlt	https://github.com/dlt-hub/dlt		scapy	https://github.com/secdev/scapy
5k-10K	camel	https://github.com/camel-ai/camel	20K+	fastapi	https://github.com/tiangolo/fastapi
	boto3	https://github.com/boto/boto3		black	https://github.com/psf/black
	cloud-custodian	https://github.com/cloud-custodian/cloud-custodian		tqdm	https://github.com/tqdm/tqdm
	aim	https://github.com/aimhubio/aim		rich	https://github.com/l textualize/rich
	speechbrain	https://github.com/speechbrain/speechbrain		open-interpreter	https://github.com/OpenInterpreter/open-interpreter
	nonebot2	https://github.com/nonebot/nonebot2		core	https://github.com/home-assistant/core
	moto	https://github.com/getmoto/moto		sherlock	https://github.com/sherlock-project/sherlock
	instructor	https://github.com/jxnl/instructor		spaCy	https://github.com/explosion/spaCy
	numba	https://github.com/numba/numba		you-get	https://github.com/soimort/you-get
	pymc	https://github.com/pymc-devs/pymc		textual	https://github.com/l textualize/textual

実験設定：メトリクス

- | インストールのパフォーマンスを測るために以下の2つの指標を使用
- **ドキュメント収集ステップのリコール**
 - $$\frac{\text{実際に回収されたインストール関連ドキュメントの数}}{\text{リポジトリ全体のインストール関連ドキュメントの数}}$$
- **Dockerfile生成ステップのインストール成功率**
 - 各リポジトリにINSTALLAMATICを10回適用した内、何回インストールに成功したかを示す割合

実験設定：メトリクス

インストールの成功要因を分析するために、以下の2つのメトリクスを計測

➤ (リポジトリの)可視性

- インストールに関連するドキュメントを見つけるのがどれだけ簡単かを示す指標

- $$(\text{可視性}) = \frac{1}{(\text{ディレクトリ数}) + (\text{ファイル数})}$$

➤ 情報性

- ドキュメントが実際のインストールプロセスに関してどれだけ包括的であるかを示す指標

- $$(\text{情報性}) = \frac{(\text{正解のDockerfileとドキュメントに共通して現れる行の数})}{(\text{正解のDockerfileの行数})}$$

実験設定：インストールタグ

インストール手法による傾向を調べるため、インストール手法(コマンド)を17に分類した後、それぞれのリポジトリに対応したインストール手法のタグを手動で付与している

TABLE II: List of Installation Tags

Tag	Description	Tag	Description
requirements	Installation of dependencies using <code>pip install -r requirements.txt</code> .	install-other	Installation of dependencies through means not listed above, such as a custom script contained in the repository
requirements-extra	Installation of dependencies from additional requirements files, such as <code>requirements-test.txt</code> .	pytest	Tests are run using <code>pytest</code> .
pip-extra	Requiring the installation of something other than Poetry or the contents of a requirements file.	pytest-extra	Additional arguments need to be provided to <code>pytest</code> , such as specifying the location of the tests or additional flags.
Poetry	Installation of dependencies using the Poetry dependency manger ¹	tox	Tests are run using <code>Tox</code> .
Poetry-extra	Installation of dependencies using Poetry, with additional arguments, e.g. <code>poetry install --no-interaction --with sentry-sdk</code> .	unittest	Tests are run using Python's built in <code>unittest</code> command.
make-install	Installation of dependencies using a makefile, typically commands such as <code>make install</code> or <code>make init</code> .	make-test	Tests are run using a makefile with a command such as <code>make test</code> .
install-self	The project itself needs to be installed in the working environment, e.g. by running <code>pip install -e .</code>	test-other	Tests are run some other way, such as a <code>test.py</code> file.
install-pytest	The <code>Pytest</code> ² library needs to be installed manually.	bash-extra	Requiring additional bash commands to set up the repository, such as creating new directories or granting permissions to certain files.
install-tox	The <code>Tox</code> ³ library needs to be installed manually.		

実験設定：実行環境

LLMのモデル：gpt-4o-mini-2024-07-18

Dockerfileにおける仮想OS:Ubuntu 22.04.4LTS

Dockerのバージョン：27.1.2

目次

研究背景

目的・アプローチ

実験設定

結果

まとめ

結果

| **研究目的**：リポジトリレベルのインストールタスクの実現可能性の調査

| 評価の指針として3つのリサーチクエスチョンを設定

1. エージェントはどこまで任意のPythonリポジトリにおいてインストールタスクを実行できるか？
2. エージェントがインストールタスクを成功するにはどのような要因があるか？
3. 自動インストールや同様のリポジトリレベルのタスクを行うエージェントの設計には、どのような課題が残っているか？

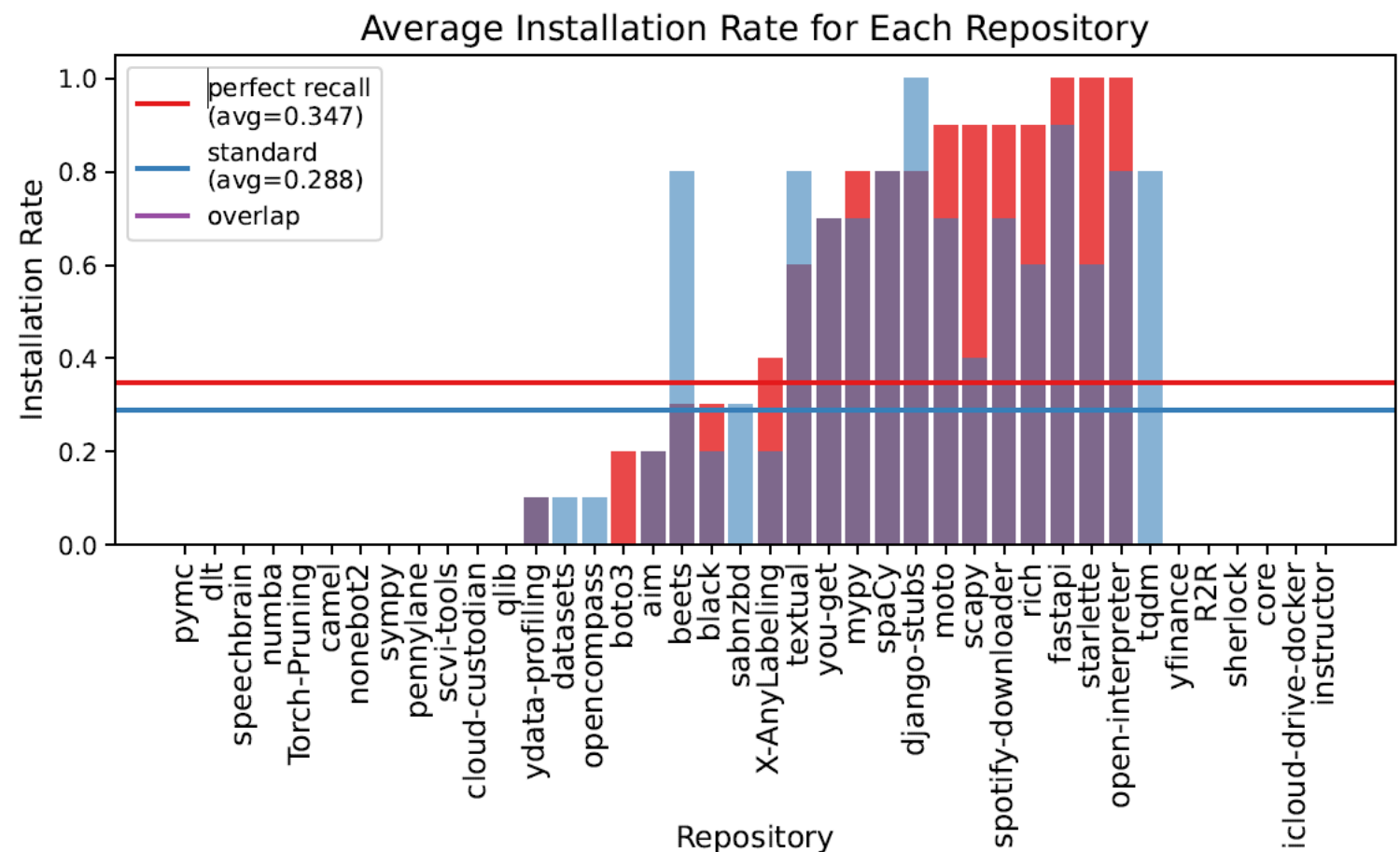
結果：RQ1について

RQ1: エージェントはどこまで任意のPythonリポジトリにおいてインストールタスクを実行できるか？

結果(各リポジトリの平均インストール率の図)

➤ 赤 → パーフェクトリコール構成

➤ 青 → 標準構成



結果：RQ1について

結果

- **標準構成**では、40リポジトリのうち、21リポジトリにおいて少なくとも一回インストールに成功した。
 - 全リポジトリの平均インストール成功率は28.8%
- **パーフェクトリコール構成**ではINSTALLAMATICはテストした34のリポジトリのうち18をインストールできた。
 - 平均インストール率は34.7%
- **パーフェクトリコール構成**より、**標準構成**の方がインストール率が勝っているリポジトリがいくつかある
 - これはLLMの出力の不安定性によるものと思われる

結果：RQ1について

- | インストール手法に関して、poetry-extra, pytest-extra が付与されていたリポジトリは1回もインストールに成功していない
- いずれもインストールに伴い、必ず引数を与えなければならない手法
- **必要な引数の一部がドキュメントに記載されていない**ことが、インストール失敗の原因と考えられる

結果：RQ1について

結論：

- INSTALLAMATICは、調査した40リポジトリのうち21リポジトリのインストールに成功
- 平均インストール成功率は28.8%
- インストールに失敗する最も一般的な理由は、インストールに成功するために必要なコマンドや引数が、関連ドキュメントに記載されていないことである。

結果：RQ2について

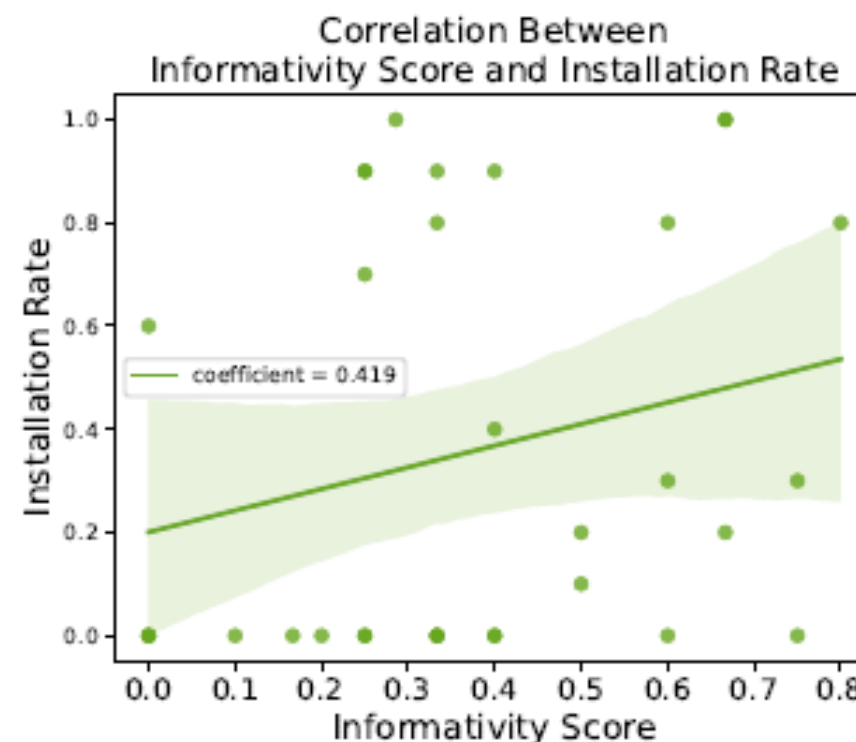
RQ2: エージェントがインストールタスクを成功するにはどのような要因があるか？

結論

- インストール修復ステップがよい影響を及ぼした
- 修復ステップを1回はさむだけで30%ほどインストール成功率が上がった.
- インストール関連文書を回収させることがよい影響をおよぼした
- よりよい構造のドキュメントがよりインストールを成功に導いた

結果：RQ2について

- これはあくまで「ドキュメントがより正確であるほどインストール成功率が上がる」というだけの話であり、ドキュメントの詳細さや、ドキュメントの表現方法による影響はわからない



(b) Correlation between the informativity score and the installation success rate

結果：RQ2について

- | インストールタグの中でもpip-extraのようなextraのつくタグを持つリポジトリと、そうでないリポジトリとではビルド率に明確な差が出た
- Extraのタグがつくインストール手法は複雑であるため、この結果から**インストールプロセスが複雑であるほどビルド成功率が下がる**と推測される。
- このようなリポジトリもまた、ドキュメントが正確に記されて入ればビルド率は上がる。

結果：RQ2について

結論：

- リポジトリのインストールプロセスの複雑さがインストールの可否に最も大きな影響を与える
- この影響は、**明確な文書構成**と、**インストール手順にコード例を使用**することで軽減できる

結果：RQ3について

RQ3：自動インストールや同様のリポジトリレベルのタスクを行うエージェントの設計には、どのような課題が残っているか？

結果：RQ3について

結論：

- ドキュメントが無秩序に配置されているリポジトリへの対処が難しい
- LLMがタイプミス等の低レベルのミスを犯すが故に、適切なDockerfileが出力されない
- 探索プロセスにLLMを使用した故に、LLMの使用コストが関連研究より高くなっている
- LLMは任意のインストール手法に対するテスト手法を生成することができない

結果：RQ3について

結論：Pythonリポジトリを自動的にインストールするためにLLMベースのエージェントを使用する場合、いくつかの制限があります。

目次

研究背景

目的・アプローチ

実験設定

結果

まとめ

まとめ

- | 40のPythonリポジトリからなるデータセットを作成した
- | リポジトリレベルのエージェントがドキュメントを理解し、リポジトリを正しくインストールする能力の有効性を評価した
- | リポジトリレベルのインストールタスクを実行するエージェントであるINSTALLAMATICの作成を行った
- | ドキュメント用のリポジトリレベルのエージェントを開発する際に直面した課題と、将来の開発者のためにこれらの課題を克服するための提案を行った

所感

| 研究の軸がイマイチよくわからなかった

| ドキュメントなしでインストールできるわけではない
あたり、まだまだ研究の余地がありそう

| 研究やその評価には相当な工夫が施されていると思った。

補足



研究背景 | 「リポジトリ」レベルのタスク

- | この論文内には「リポジトリレベルのタスク」に関する明確な定義は存在していない
- ネットで調べてもそれらしきものは出てこなかった
- | 本研究においてリポジトリレベルのタスクとされているもの
- リポジトリレベルコーディングタスク
- リポジトリレベルコード生成タスク
- コードリポジトリのインストール

研究背景 | 「リポジトリ」レベルのタスク

- | この論文内には「リポジトリレベルのタスク」に関する明確な定義は存在していない
- ネットで調べてもそれらしきものは出てこなかった

| 本研究においてリポジトリレベルのタスクとされているもの

- リポジトリレベルコーディングタスク
- リポジトリレベルコード生成タスク
- コードリポジトリのインストール

リポジトリというスコープで、
何かソフトウェア関連のタスクを行うこと
→ リポジトリレベルのタスク。