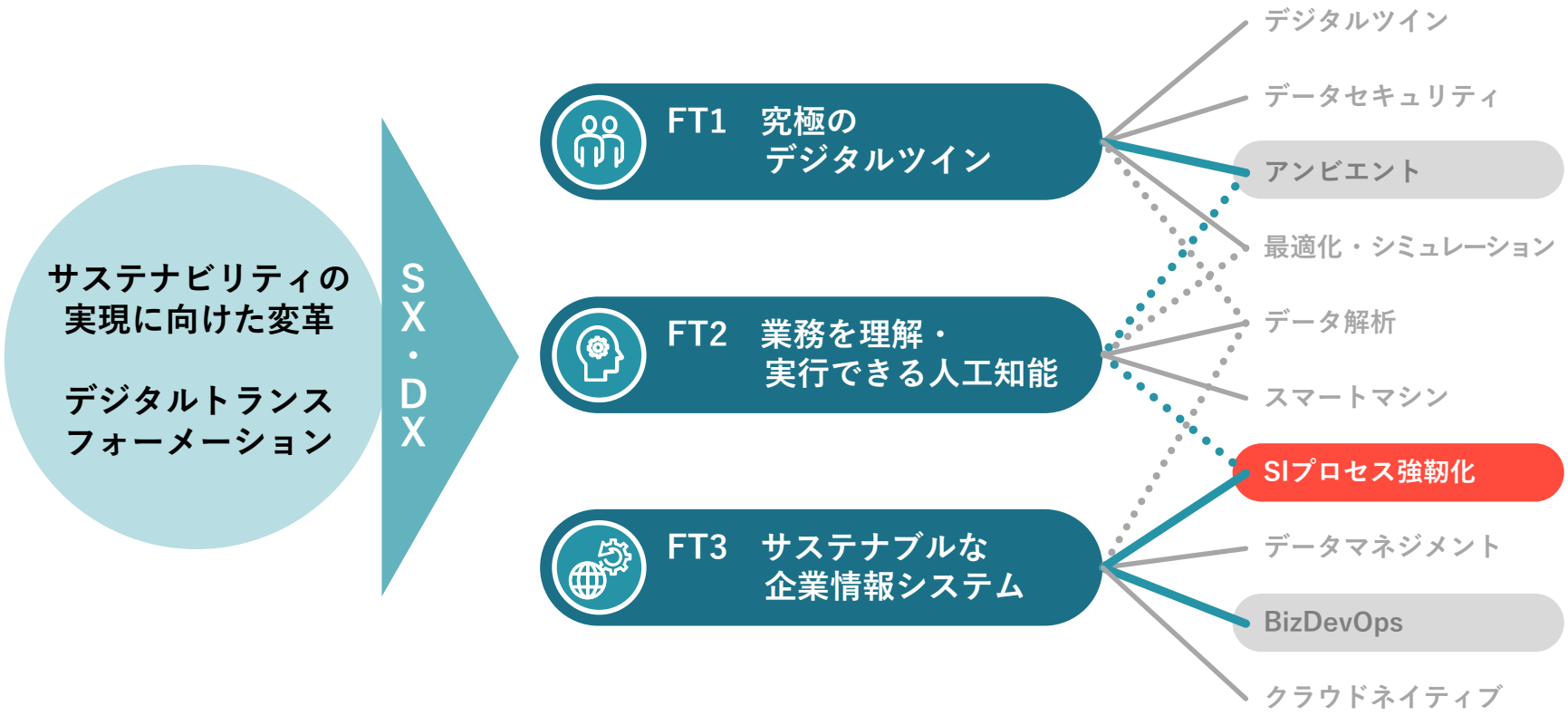
Web画面テスト自動化AI Curatis（キュラティス）

技術本部　システム研究開発センター　研究員
米倉　千晶

技術本部　システム研究開発センター　研究員
坂田　雄亮

未来目標における本セッションの位置付け



アジェンダ

1 Curatisが解決を目指す課題

2 Curatisの紹介

3 今後の展望

1 Curatisが解決を目指す課題

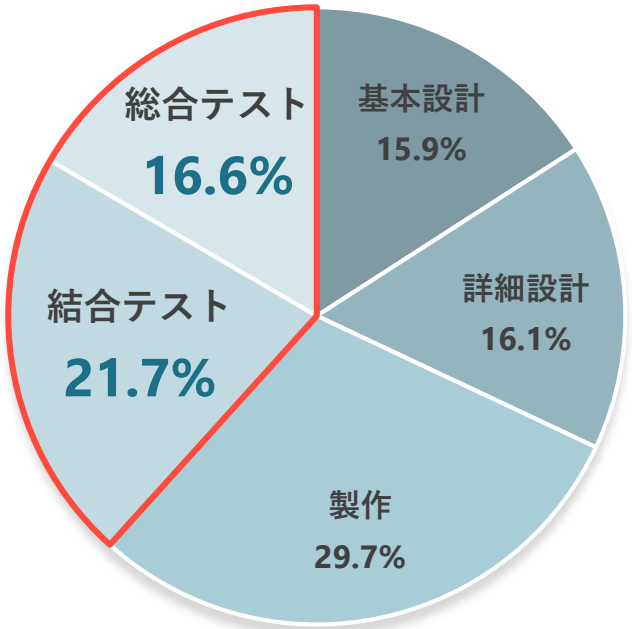
未来目標との関連「業務を理解・実行できる人工知能」

システム開発業務を対象とした取り組み

- システム開発の現場では、まだ手作業部分が多い
- システム開発の手作業部分をAIにより自動化することで、
  - 開発プロセスをより高速化できる
  - 開発リソースをより高付加価値な業務へ集中させることができる
- システム開発のテストフェーズに着目し、AIによる自動化にチャレンジ

テストを自動化できるとコスト・アジリティに効果大

- テストの工数はシステム開発工数の
  1/3〜1/4 を占めるボリュームゾーン
- Web画面テストはテスト工程中の半分ほど



工数別の実績工数の比率の基本統計量（改良開発）平均値,
情報処理推進機構社会基盤センター「ソフトウェア開発データ白書2018-2019」

Web画面テストの手作業の実行はQCDに深刻な影響

- Web画面テストとは
  - Webアプリケーション開発の最終段階で、システムが期待通りに動作するか画面を実際に操作して確認する工程

テスト仕様書作成

Web画面テスト実施

Web画面テストを手作業で地道にやると大変

- テスト仕様書の指示に従いながら、画面を操作し、結果を確認し、画面をスクショし、結果を記録し…
  （例）5000件のテストケースの場合、画面操作数は約35000回
    - 1人でやると、1日に900クリック×40日ほどかかる
- しかもこれをシステム改修のたびに実施する

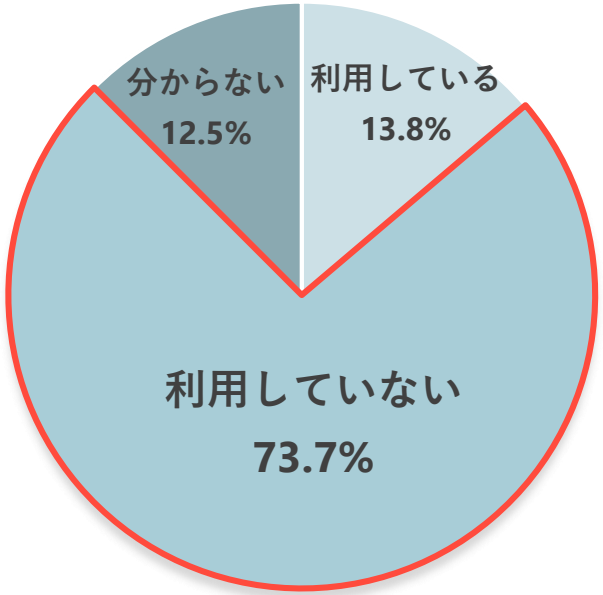
現時点の主なWeb画面テスト自動化の方法

- テストプログラムによる自動化
  - 画面操作を行うテストプログラムをコーディング

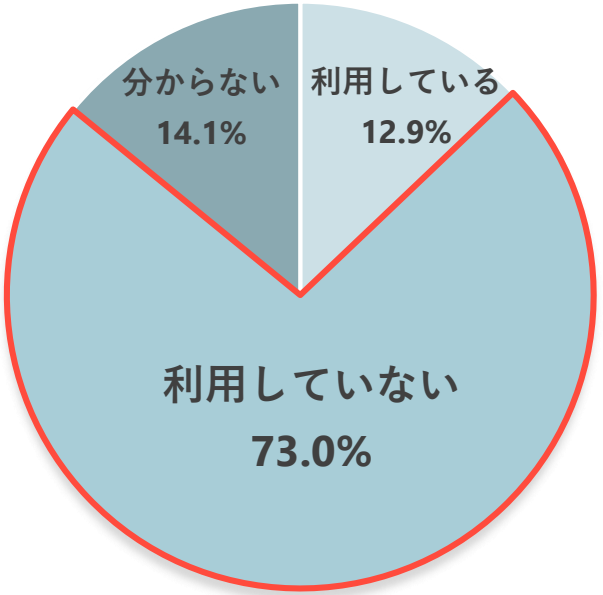
テストプログラム開発のコストが大きい

- **テスト仕様書をもとに、画面操作数分にあたるプログラムをコーディング**
  （例）5000件のテストケースの場合、画面操作数は約35000回
  ⇒テストプログラムは数万行

Web画面テスト自動化の実施率の低さ



結合テストのツール利用率



システムテストのツール利用率

- 日経クロステック「開発支援ツール徹底調査2011 テスト編」より。
- JUAS「ソフトウェアメトリックス調査2022 システム開発・保守調査」でも同様の結果が得られている。

Web画面テストプログラム開発を妨げる3つの壁

テストプログラムによる自動化は、壁の克服が難しく二の足を踏む

1 スキルの壁

- テストプログラムの開発に特別なスキルが必要で人材調達に課題あり。
- レビューも容易ではない。

2 開発コストの壁

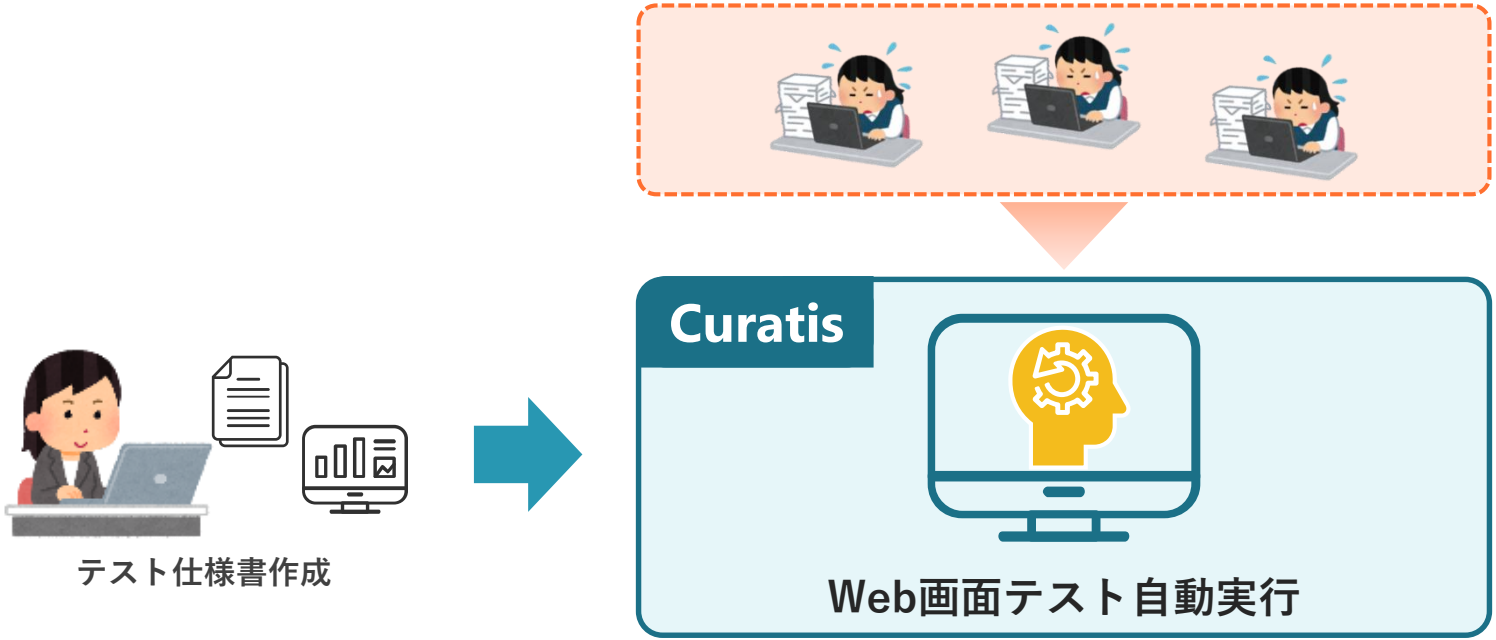
- テストプログラムの開発コストは、8回ほどテストが実行されないと元がとれないほど高い。

3 保守の壁

- 画面の仕様変更があればテストプログラムも改修しなければならない。
- 場合によっては画面改修そのものよりテストプログラムの改修の方が難しい(本末転倒)。

Web画面テスト自動化の壁を打破するAIを開発

- 人間が行っていたWeb画面テスト実行部分をAIに置き換える
- テストプログラムコーディングなしで自動実行



2 Curatisの紹介

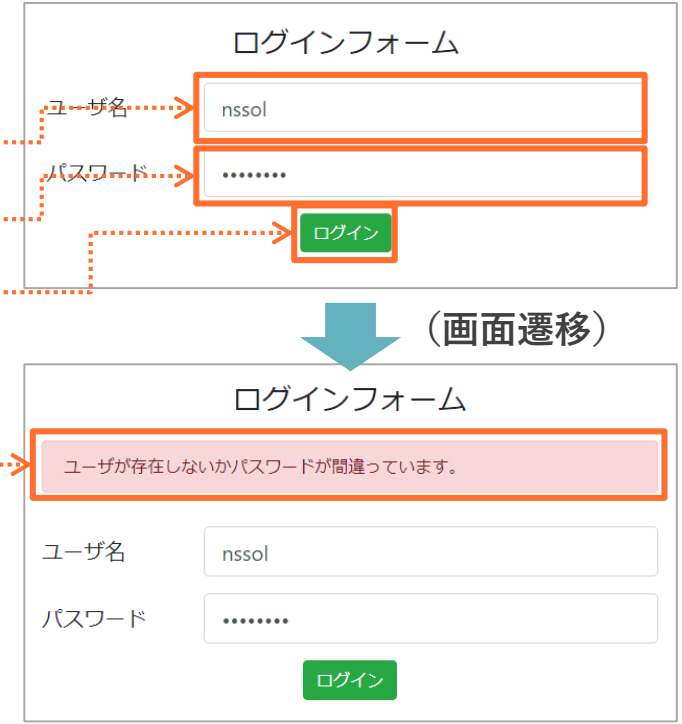
Web画面テスト自動化AI「Curatis」概要

日本語で書かれたテスト仕様書の文章を理解し、
その通り画面操作をしてテストを自動実行し、スクリーンショットの取得等をしてくれるAI

テスト仕様書

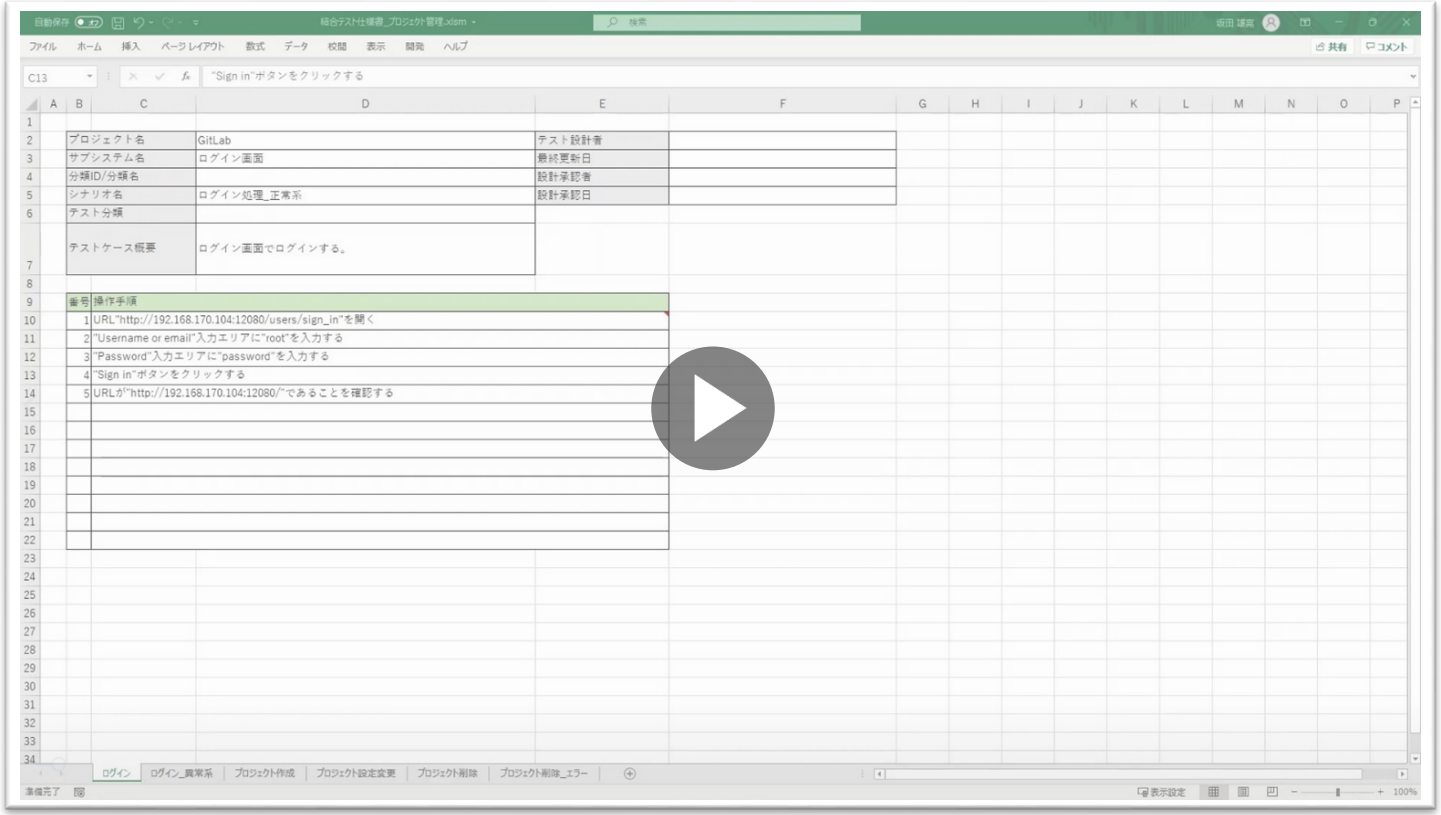
番号	操作内容
1	“ユーザ名”入力エリアに“nssol"を入力する
2	“パスワード”入力エリアに”password”を入力する
3	“ログイン”ボタンをクリックする
4	テキスト”ユーザが存在しないかパスワードが間違っています。”が存在することを確認する
	⋮

テスト画面



デモムービー

実際のWebアプリケーションのテスト自動実行をデモ



Curatisの効果

スキルの壁

ノーコード

- 非プログラマでも作成可能。
- レビューも容易。

開発コストの壁

生産性6倍※

- 記述量削減。
- テスト仕様書を書くだけで英語モードのシステムも日本語モードのシステムもテスト可能。

※p.16に示すケースの場合

保守の壁

改修コスト減

- テスト仕様書変更が不要なレベルなら改修不要。
- AIは画面レイアウトの軽微な変更にも強い。

システムの品質を高められる
安心して迅速な機能の追加改修が行える

Web画面テストが自動化できる。
素早く何度でもテストができる。

ノーコード・生産性6倍

```python
import time

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select, WebDriverWait
from webdriver_manager.chrome import ChromeDriverManager


class TestPlanisphere:
    def test_reserve_otoku(self):
        driver = webdriver.Chrome(ChromeDriverManager().install())
        wait = WebDriverWait(driver, 10)

        driver.get("https://hotel.testplanisphere.dev/ja/reserve.html?plan-id=0")
        # NOTE: wait.until(EC.presence_of_all_elements_located) <- 上手く動作せず
        time.sleep(1)  # HACK
        driver.save_screenshot("selenium_1.png")  # スクリーンショット撮影

        date = driver.find_element_by_id("date")  # 宿泊日
        date.clear()
        date.send_keys("2020/12/24")
        date.send_keys(Keys.TAB)
        driver.save_screenshot("selenium_2.png")  # スクリーンショット撮影

        driver.find_element_by_id("username").send_keys("ベリ坊")  # 名前
        driver.save_screenshot("selenium_6.png")  # スクリーンショット撮影

        Select(driver.find_element_by_id("contact")).select_by_value("tel")  # 「電話連絡を希望」
        driver.save_screenshot("selenium_7.png")  # スクリーンショット撮影

        driver.find_element_by_id("tel").send_keys("80120828828")  # 電話番号
        driver.save_screenshot("selenium_8.png")  # スクリーンショット撮影

        driver.find_element_by_id("submit-button").click()  # 予約確認ページへ遷移
        wait.until(EC.presence_of_all_elements_located)
        driver.save_screenshot("selenium_9.png")  # スクリーンショット撮影

        assert "15,000" in driver.find_element_by_id("total-bill").text

        driver.quit()
```

従来のテストプログラム

記述量
約1/6に削減

番号	操作内容
1	URL”https://hotel.testplanisphere.dev/ja/reserve.html?plan-id=0”を開く
2	“宿泊日”入力エリアに”2022/07/24”を入力する
3	“氏名”入力エリアに”taro”を入力する
4	“選択してください”をクリックする
5	“電話でのご連絡”をクリックする
6	“電話番号”入力エリアに”01234567890”を入力する
7	“予約内容を確認する”をクリックする
8	テキスト”合計 9,750円（税込み）”が存在することを確認する

Curatis

Curatisの効果

スキルの壁

ノーコード

- 非プログラマでも作成可能。
- レビューも容易。

開発コストの壁

生産性6倍※

- 記述量削減。
- テスト仕様書を書くだけで英語モードのシステムも日本語モードのシステムもテスト可能。

※p.16に示すケースの場合

保守の壁

改修コスト減

- テスト仕様書変更が不要なレベルなら改修不要。
- AIは画面レイアウトの軽微な変更にも強い。

システムの品質を高められる
安心して迅速な機能の追加改修が行える

Web画面テストが自動化できる。
素早く何度でもテストができる。

Curatisを実現する先端AI技術

Curatis

文章の意味理解

日本語と英語をどちらでも理解、
軽微な表記揺れの吸収

×

画面レイアウトの理解

マルチモーダル

複数メディア（文章データと視覚データ）を
ミックスしたより人間らしい意味理解

AI技術を用いた研究開発実績	長年のシステム開発高度化の研究開発実績
Cogmino, Lumisis, etc.	Pitalium, Lacat, etc.

3 今後の展望

今後の展望

今後のシステム開発では、Curatisがメンバーとして参画する時代がきます！

- NSSOLのシステムインテグレーション力の強化、QCDの抜本的改善
  - 実証実験を重ね、ソリューションとして作りあげ、機能を洗練させていく

マルチモーダル深層学習技術の研究開発を進め、
人間のような知覚や読解能力を備えたAIを生み出すことで、FT2の実現を目指す

商標について

NS Solutions、 NSSOL、NS(ロゴ)、Curatis、 Cogmino、Lumisis、Pitalium、Lacatは日鉄ソリューションズ株式会社の商標または登録商標です。

その他本文記載の会社名及び製品名は、それぞれ各社の商標又は登録商標です。