

# LatteArt : テスト活動を収集・分析・ 可視化するNTT発のOSS

日本電信電話株式会社  
ソフトウェアイノベーションセンタ  
但馬将貴・切貫弘之・丹野治門

# はじめに

- 皆さん、ソフトウェアのテストってどうやっていますか？
- 事前にドキュメント化する「記述式テスト」ですか？
  - テスト対象の仕様に従って、事前にテスト設計する
  - 何をテストして何をテストしないかが明確のため、説明性が高い
  - テスト設計から外れるテストを行いにくいため、テスト中に気づいた違和感や無駄をテストに反映しにくい

No	項目名	試験種別	試験手順	確認内容				
					実施対象	予定日	試験者	開始日
1	指定OS上でGUI部が起動すること。(Windows7 64bit)	正常系	1. 「9ペア試験準備手順」を実施する。 2. エンジンを起動しGUIを出力する。(Windows7 64bit) 3. GUIを起動する。(Windows7 64bit)	1. GUI部が起動すること。 2. エラーが発生しないこと	○	10/20(金)	xx	10/13(
2	指定OS上でGUI部が起動すること。(Windows10 64bit)	正常系	1. 「9ペア試験準備手順」を実施する。 2. エンジンを起動しGUIを出力する。(Windows10 64bit) 3. GUIを起動する。(Windows10 64bit)	1. GUI部が起動すること。 2. エラーが発生しないこと	○	10/20(金)	xx	10/18(
3	指定OS上でGUI部が起動すること。(RHEL出力結果+Win7)	正常系	1. 「9ペア試験準備手順」を実施する。 2. エンジンを起動しGUIを出力する。(RHEL) 3. GUIを起動する。(Windows7 64bit)	1. GUI部が起動すること。 2. エラーが発生しないこと	○	10/20(金)	xx	11/2(フ
4	指定OS上でGUI部が起動すること。	正常系	1. 「9ペア試験準備手順」を実施する。 2. エンジンを起動しGUIを出力する。(RHEL) 3. GUIを起動する。(Windows10 64bit)	1. GUI部が起動すること。 2. エラーが発生しないこと	○	10/20(金)	xx	11/6(フ
5	指定OS上で起動したGUI部が終了できること。、	正常系	1. 「9ペア試験準備手順」を実施する。 2. エンジンを起動しGUIを出力する。 3. GUIを起動する。(Windows7 64bit) 4. ブラウザの閉じるボタンを押下する。	1. GUI部が終了すること。 2. エラーが発生しないこと 3. 終了確認ダイアログが表示されること	○	10/20(金)	xx	10/13(

## 事前設計されたテスト設計書の例

- 最近注目<sup>(\*1)</sup>の「探索的テスト」ご存じですか？
  - 今まで見つかったバグや違和感，ドメイン知識に基づいてテスト実施し，その振る舞いを見極めてテスト中に動的にテスト設計する
  - アジャイル開発と相性が良く，バグ発見効率も良い
  - 手順書が無いため，どのようなテストを行ったか，記録を残すのが難しい
  - テストの振り返りやバグ再現が難しく，レポーティングも難しい
  - テストの質がテスターのスキルに依存し，属人性が高い
  - テストを管理するのが難しい

(\*1)テスト手法の調査によると，記述式テストは約60%，探索的テストは約80%。

PractiTest: "STATE OF TESTING REPORT 2021",

<https://www.practitest.com/assets/pdf/state-of-testing-report-2021.pdf>

- 最近注目<sup>(\*1)</sup>の「探索的テスト」ご存じですか？
  - 今まで見つかったバグや違和感，ドメイン知識に基づいてテスト実施し，その振る舞いを見極めてテスト中に動的にテスト設計する
  - アジャイル開発と相性が良く，バグ発見効率も良い
  - 手順書が無いため，どのようなテストを行ったか，記録を残すのが難しい
  - テストの振り返りやバグ再現が難しく，レポーティングも難しい
  - テストの質がテスターのスキルに依存し，属人性が高い
  - テストを管理するのが難しい

## 課題

(\*1)テスト手法の調査によると，記述式テストは約60%，探索的テストは約80%。

PractiTest: "STATE OF TESTING REPORT 2021",

<https://www.practitest.com/assets/pdf/state-of-testing-report-2021.pdf>

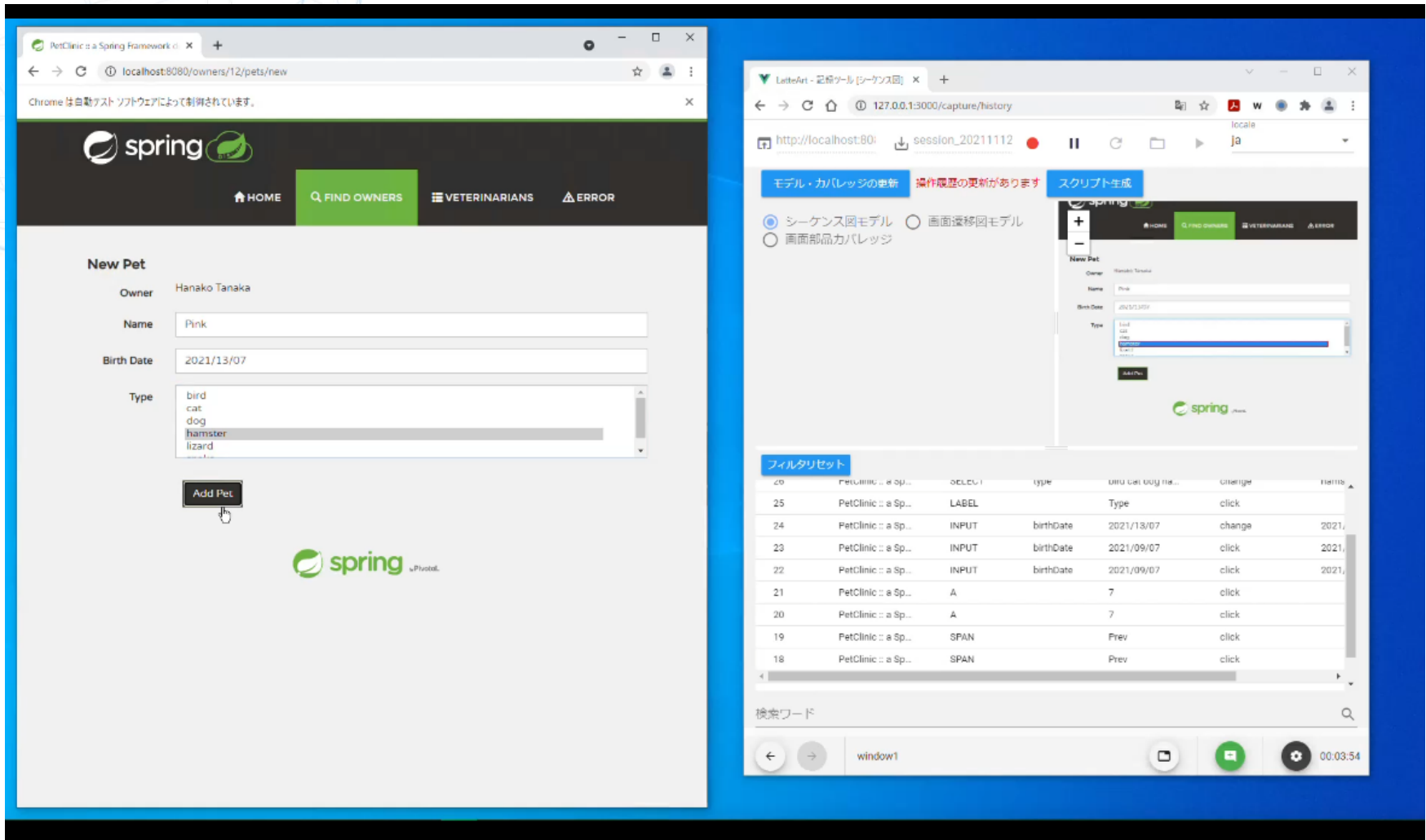
- 「LatteArt」は探索的テストの課題を解決します
  - テスト中にテストの目的や気づきをテスト実施履歴に記録でき、どのような観点でテストを行ったか、自身や他人が後から分かる
  - 画面遷移図や画面要素カバレッジなどの可視化により、テスト実施状況がより俯瞰的に分かる
  - 実施履歴や入力値を自動記録して可視化するので、テスト後の振り返りやバグ再現、レポーティングも容易
- 効果（探索的テストがどうよくなるのか）
  - テスターが振り返りを行えることで、より効率良く探索的テストを実施できる
    - テスターが振り返りをするときに他者からレビューをすることができる
  - バグ再現やレポーティングの稼働が減る



- テスト活動の収集
  - テスト実施履歴の自動記録
    - クリック・入力操作, 入力値, 画面遷移, スクリーンショットの記録
  - テスト目的と気づきの手動記録
    - テスト目的: テストの意図や知見を記録
    - 気づき: 怪しい挙動やバグに気づいた際に記録
- テスト活動の分析・可視化, 活用
  - シーケンス図, 画面遷移図, 画面部品カバレッジによる可視化
  - テストスクリプトの自動生成による回帰テスト自動化支援
- テストの管理
  - テスト計画のスケジューリング
  - テスト計画の進捗可視化
  - テスト結果のレビュー

- テスト活動の収集
  - テスト実施履歴の自動記録
    - クリック・入力操作, 入力値, 画面遷移, スクリーンショットの記録
  - テスト目的と気づきの手動記録
    - テスト目的: テストの意図や知見を記録
    - 気づき: 怪しい挙動やバグに気づいた際に記録
- テスト活動の分析・可視化, 活用
  - シーケンス図, 画面遷移図, 画面部品カバレッジによる可視化
  - テストスクリプトの自動生成による回帰テスト自動化支援
- テストの管理
  - テスト計画のスケジューリング
  - テスト計画の進捗可視化
  - テスト結果のレビュー

本日のデモ





The screenshot shows a web browser window with the URL `http://localhost:8080/owners/12`. The page displays a sequence diagram on the left and a pet registration form on the right. The sequence diagram is titled "[ (1)ペットを新規登録する ]" and shows interactions between a window and a pet registration form. The steps are:

- (1)window1
- (2)click: Add New Pet
- (11)click: Add Pet
- (13)click: Add New Pet
- (28)click: Add Pet
- (36)click: Add Pet
- (38-0)誕生日に未来の日付を入れて登録できてしまう

The pet registration form on the right has the following fields:

- Name:
- Birth Date:
- Type:

At the bottom of the browser window, there is a table with the following data:

Filter Reset	Step	Page	Element	Action	Time
	29	FetClinic :: a Spring Framework d...	Edit Pet	click	13:57:01
	38	FetClinic :: a Spring Framework d...	dog	click	13:56:37
	37	FetClinic :: a Spring Framework d...		screen_transition	13:56:32
	16	FetClinic :: a Spring Framework d...	ADD PET	click	13:56:31

The screenshot shows a web browser window displaying a pet management application. The URL is `http://localhost:8080/owners/12`. The application has a navigation bar with "HOME", "NEW OWNERS", "PET OWNERS", and "ERROR". The main content area shows a "New Pet" form with fields for "Name", "Age (days)", and "Type", and an "Add" button. The test runner overlay on the left shows a flowchart with nodes: "ペット一覧画面" (top), "ペット新規登録画面" (bottom left, highlighted in pink), and "ペット編集画面" (bottom right). Transitions are labeled with "click: Add New Pet", "click: Add Pet", and "click: Edit Pet". Below the flowchart, there are checkboxes for "明示的に入力された値以外のセルをクリアアウトする" and "hidden要素は表示しない". At the bottom, a table lists test cases:

要素ID	要素名	type	1回目	2回目	3回目
			[ペット新規登録画面]	[ペット新規登録画面]	[ペット新規登録画面]
			click: Add Pet	click: Add Pet	click: Add Pet

```
File Edit Selection View Go Run Terminal Help • test.spec.js - Visual Studio Code
pec.js > describe('generated test cases') callback > it('ペット一覧画面 -> ペット新規登録画面 -> ペット一覧画面') callback
4 * @namespace generated test cases
5 * @mermaid
6 * graph TD;
7 *   ペット一覧画面 --> ペット新規登録画面;
8 *   ペット新規登録画面 --> ペット一覧画面;
9 *   ペット一覧画面 --> ペット編集画面;
10 *   ペット編集画面 --> ペット一覧画面;
11 */
12 describe('generated test cases', () => {
13   beforeEach('open top page', () => {
14     browser.url('http://localhost:8080/owners/11');
15   });
16
17   /**
18    * @function ペット一覧画面 -> ペット新規登録画面 -> ペット一覧画面
19    * @memberof generated test cases
20    * @mermaid
21    * graph TD;
22    *   ペット一覧画面 --> |go| ペット新規登録画面 | ペット新規登録画面;
23    *   ペット新規登録画面 --> |doAddPet| ペット一覧画面;
24    *   ペット一覧画面 --> ペット編集画面;
25    *   ペット編集画面 --> ペット一覧画面;
26    */
27   it('ペット一覧画面 -> ペット新規登録画面 -> ペット一覧画面', () => {
28     new ペット一覧画面()
29     // ペット一覧画面.page.js
30     .go ペット新規登録画面()
31     // ペット新規登録画面.page.js
32     .doAddPet({
33       name: 'Black',
34       birthdate: '2010-11-10',
35       type: 'cat'
36     });
37   });
38
39   /**
40    * @function ペット一覧画面 -> ペット編集画面 -> ペット一覧画面
41    * @memberof generated test cases
42    * @mermaid
43    * graph TD;
44    *   ペット一覧画面 --> ペット編集画面;
45    *   ペット編集画面 --> ペット一覧画面;
46    */
47 });
```

Class: ペット新規登録画面

ペット新規登録画面()

new ペット新規登録画面()

```
classDiagram
    class ペット一覧画面
    class ペット新規登録画面
    class ペット編集画面
    ペット新規登録画面 --> ペット一覧画面 : doAddPet
    ペット編集画面 --> ペット一覧画面
```

Source: [page\\_objects/ペット新規登録画面.page.js, line 11](#)

### Methods

doAddPet()

1. Change [ name ]
2. Change [ birthdate ]
3. Change [ type ]
4. Click [ addPet ]
5. Move to [ ペット一覧画面 ]

Source: [page\\_objects/ペット新規登録画面.page.js, line 27](#)

Documentation generated by JSDoc 3.6.7 on Wed Nov 10 2021 11:03:45 GMT+0900 (GMT+09:00)

# 想定するユースケース

- 探索的テストをよくする
  - 今回ご紹介した実施例
- 記述式テストをよくする
  - テスト証跡を取りたいときに活用できる
  - バグ報告に使える
- 記述式テストを探索的テストに置き換える（野心的！？）
  - 品質説明ができれば置き換え可能？

- GitHub 公開リポジトリURL（2021/11/25より公開）
  - <https://github.com/latteart-org/latteart>
- 利用方法
  - [リリース](#)より最新のlatteart-〇〇〇.zipをダウンロード後、[README\\_ja.md](#)に従い事前設定後すぐに利用可能
- 利用にあたり
  - 本技術は評価版です。サポートやバグ修正対応はベストエフォートとなります。
- 皆さまへ
  - バグや機能改善要望がありましたら、ぜひ[イシュー](#)を起票ください。
  - コントリビュータ大募集！一緒にNTT発のOSS:LatteArtを育てませんか？

## ハードウェア要件

#	項目	内容
1	端末	PC
2	CPU	Intel Core i5 3.20Ghz以上
3	メモリ	8GB以上

## ソフトウェア要件

#	項目	内容
1	OS	Windows 10 (64bit) MacOS (実験的)
2	ブラウザ	- Google Chrome - USB接続したAndroid/iOS上のGoogle Chrome (実験的)
3	その他	Chrome Driver ※Google Chromeに対応したバージョン

## テスト対象の要件

#	項目	内容
1	実装形態	HTMLで構成されるWebアプリケーション
2	記録可能な操作	クリック, 文字入力のみ 特殊な操作 (ダブルクリック, ドラッグ, ホバー等) を記録することはできません。

# 将来の展望

- NTTグループ内外に広く普及展開
- テスト活動データの活用
  - テストスクリプトの自動生成
  - テスト推薦
  - テスト教育