TERASOLUNA

ZIPC Testerで加速する NTTデータのソフトウェア生産技術革新

2014年10月10日 ZIPCユーザーズカンファレンス

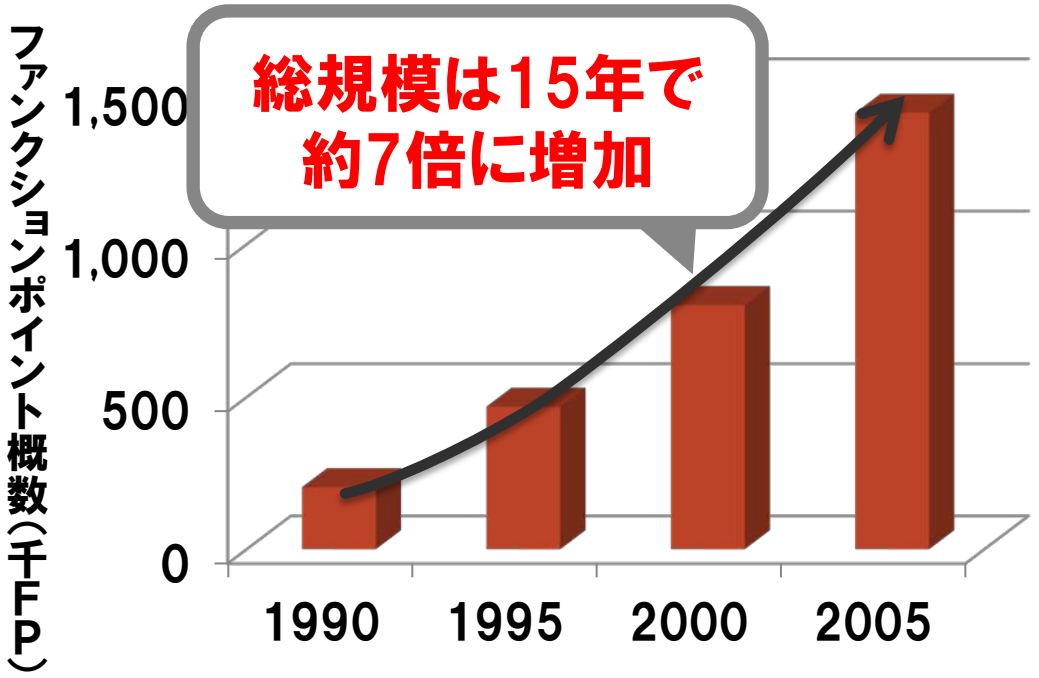
技術開発本部 ソフトウェア工学推進センタ
課長 金子 武彦

NTT DATA

はじめに

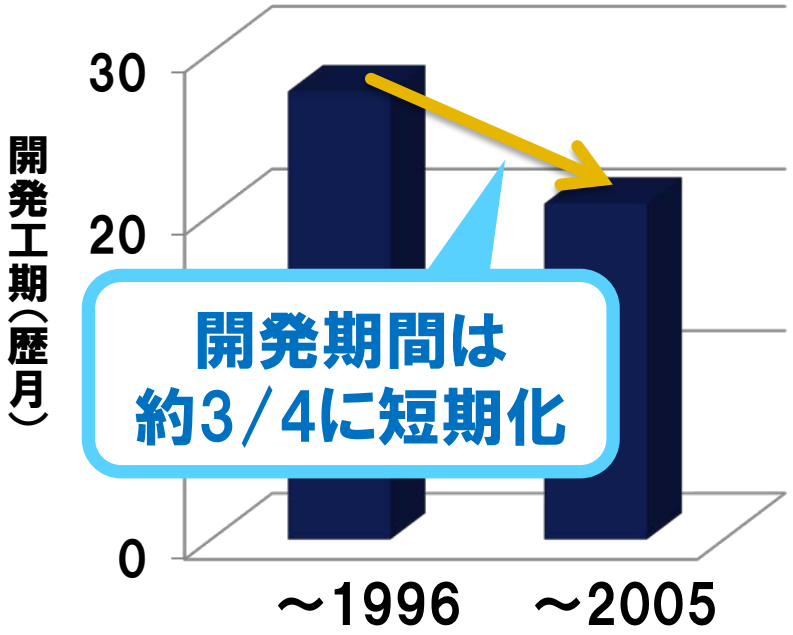
企業が保有するソフトウェアは激増、開発期間は短期化している

企業が保有するソフトウェア総規模の推移（米国）



※全分野ソフトウェアの平均FP概数（国防総省を除く）

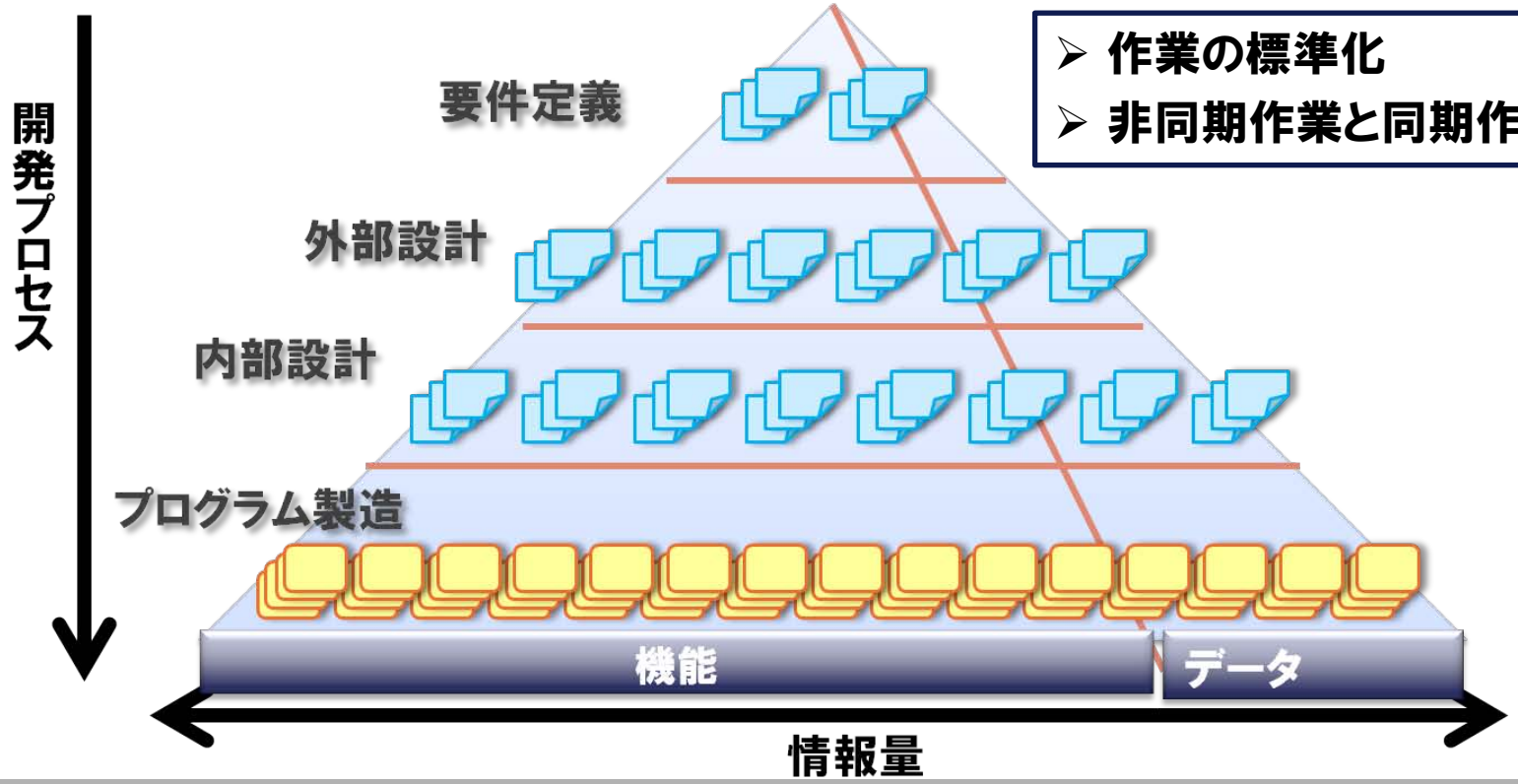
ソフトウェア開発工期の推移（米国）



※10000FP規模プロジェクトの平均工期

Capers Jones著「ソフトウェア開発の定量化手法第2版／第3版（共立出版）」掲載データよりグラフ作成

大規模かつ高品質なシステム開発を行うための「工程分担・水平分散モデル」



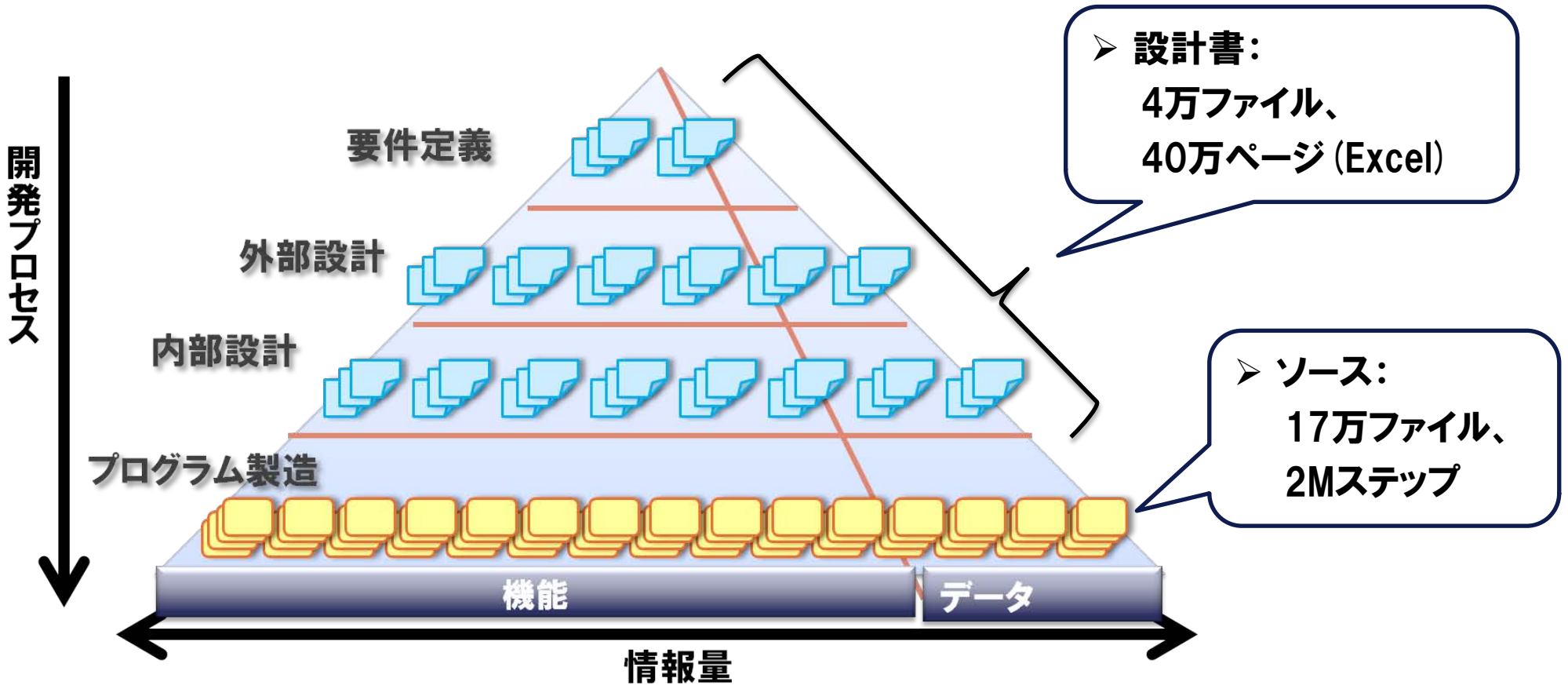
- 作業の標準化
- 非同期作業と同期作業の高レベルの管理

開発が進むにつれて増加する大量のドキュメント・コードを、一か所に集めて品質を管理

現在の開発モデルの限界　ー某プロジェクト事例

NTT DATA

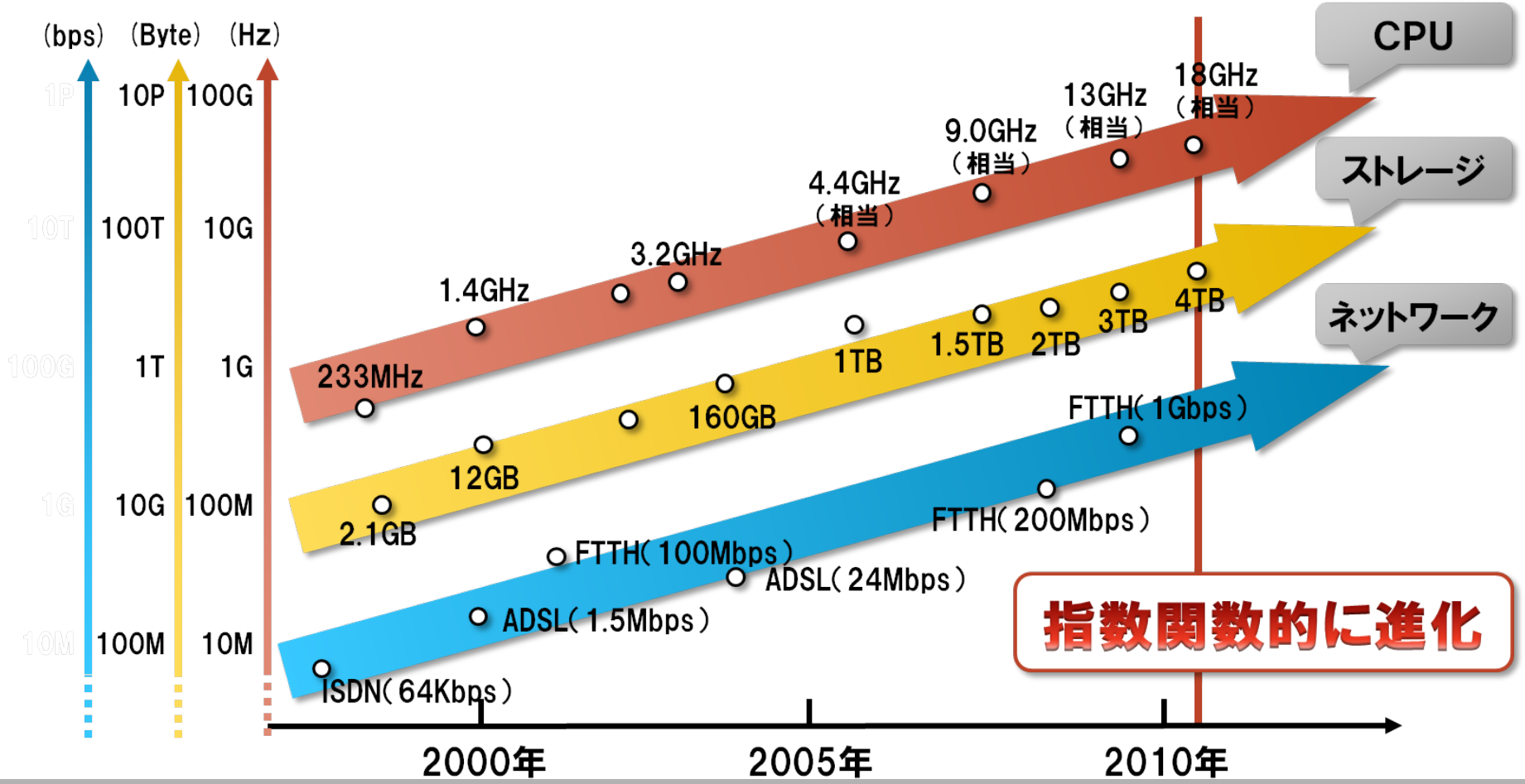
工数　　:2年間　20,000人月



人手による管理の限界

NTT DATA

三大要素技術の爆発的な進歩はつづく



余剰リソースを用いた高速化

【注釈】(相当)とはマルチコアプロセッサをシングルコア換算をしたもので、マルチコアプロセッサについて、2コア、4コア、8コア、10コアの性能を、それぞれ通常のシングルコアプロセッサ処理能力の1.5倍、3倍、6倍、7.5倍と評価。
2006年から順に、2コア2.93GHzの1.5倍で4.4GHz、4コア3GHzの3倍で9GHz、8コア2.26GHzの6倍で13GHz、10コア2.4GHzの7.5倍で18GHzとした。

ソフトウェア開発自動化

NTT DATA

従来、開発者が人手で行っていた開発作業の一部を
コンピュータで正確・迅速に実行させる

生産性の向上

- 人に依存しない開発
- 少ないリソースで大量生産

＋

品質の確保

- 人的ミスの排除
- スキルによらない品質の画一化

ソフトウェア開発自動化

生産技術の革新によって
労働集約型から知識集約型の開発スタイルへ

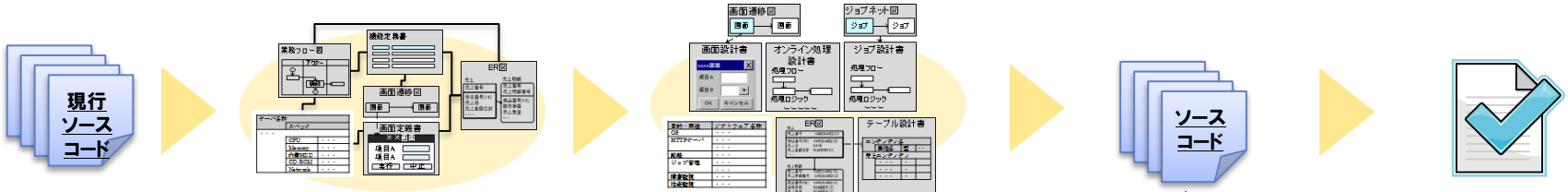
TERASOLUNA Suite

TERASOLUNA Suite の全体像

一貫した開発自動化により「高品質」を保った「高速開発」を実現

従来開発

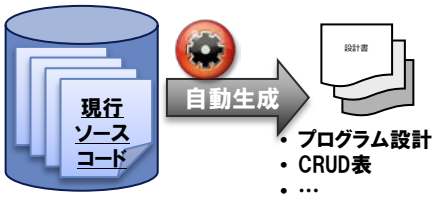
要件定義 | 設計 | 製造 | テスト



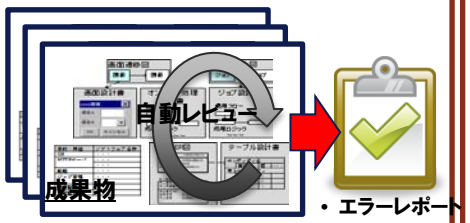
開発自動化

キャッツと協業

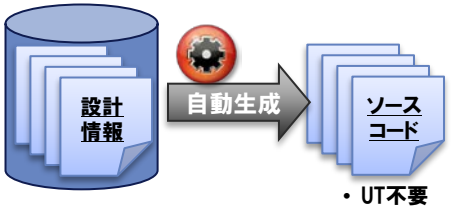
現行コード自動解析



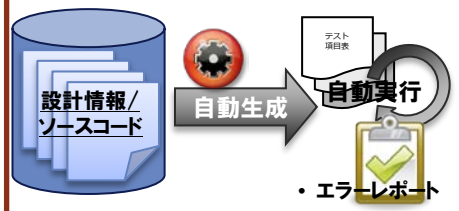
開発成果物自動レビュー



ソースコード自動生成



テストの自動実行



TERASOLUNA Reengineering	TERASOLUNA DS	TERASOLUNA IDE 開発機能 TERASOLUNA ViSC	TERASOLUNA IDE 試験機能

現行システムのプログラムや運用ログから、処理や機能、仕様を 正確にスピーディに低コストで解析するソリューション

現行システム

COBOL、JCL、
Java、PL/I　対応

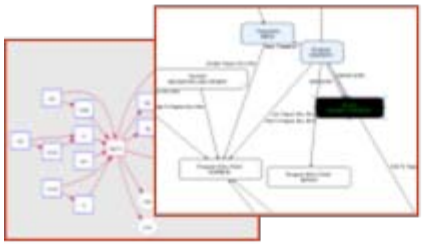
ソースコード

プログラム解析

資産整理

現行分析の範囲、コスト、リスク等の把握のための分析資料作成

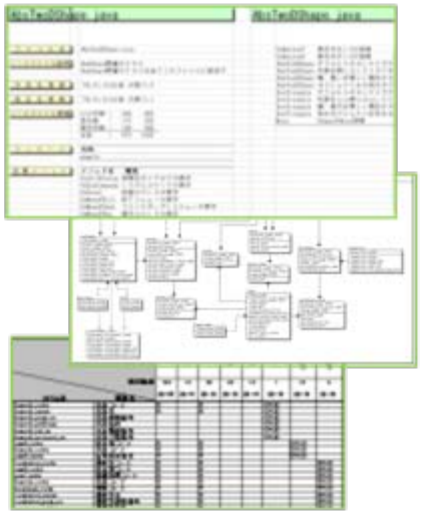
不要・不稼動資産分析



資産構造分析

「処理」の理解

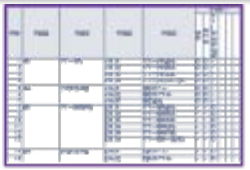
プログラム設計書・内部設計書レベルの成果物生成



業務処理ロジック抽出

「機能、仕様」の理解

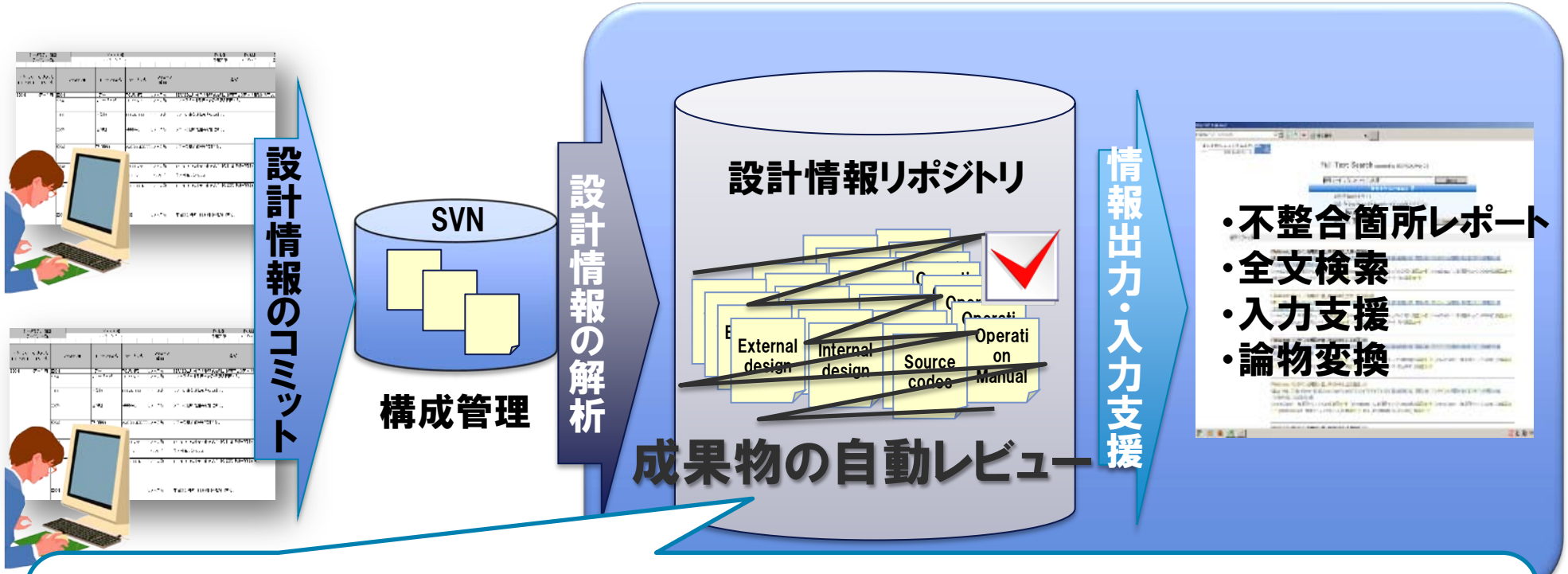
外部設計書・要件定義書レベルの成果物生成（研究中）



データ編集仕様抽出

個別業務ルール抽出

設計情報間の整合性確保を自動化し、レビュー時の形式チェック作業削減等、設計作業を大幅に効率化



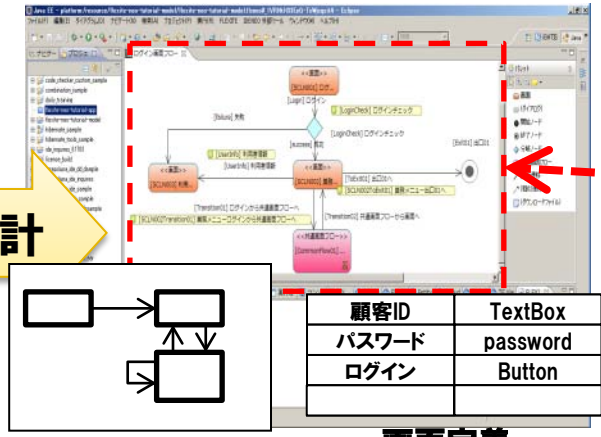
特長：**様々なフォーマットの設計書**に対応できます。
：プロジェクト特性にあわせた**柔軟なチェック**ができます。
：工程を跨る設計書でも**横断的に整合性チェック**ができます。

画面コードや設定ファイルを、設計モデルから自動生成し
設計情報と製造成果物の一貫制維持を実現

GUIベースの専用エディタで
モデル指向設計を行うことで効率化と
品質向上を実現

設計者

設計



顧客ID	TextBox
パスワード	password
ログイン	Button

画面遷移図

画面定義

モデルの双方向の対応
関係をルールで規定

設計モデル

反映ルール

コードモデル

テキストベースの
コードをモデル化

自動生成

設計情報をモデルで一元管理し、モデル
からTERASOLUNAに準拠したソースコード
や設定ファイル、設計書を自動生成

多様なアプリケーションでも業務ロジックのソースコードを完全自動生成することで、製造・単体試験を削減

Microsoft® Excel®で入力した情報から、「設計書」と「実行可能なプログラム」を完全自動生成

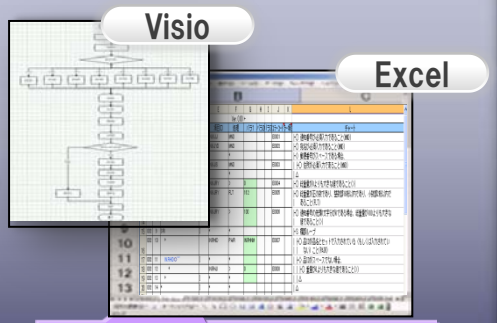
ツールのカスタマイズ

プロジェクトの固有性に最適化

プロジェクト個別にカスタマイズ可能

適用

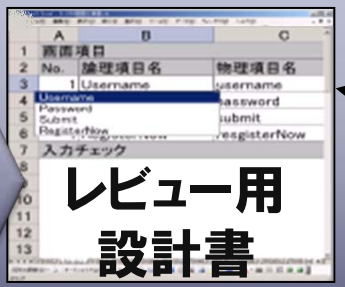
設計情報入力



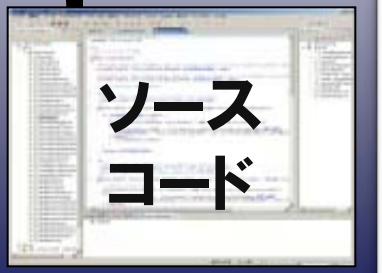
Microsoft Visio®
Microsoft® Excel®
フォーマットで作成

自動生成

ソースコードを100%自動生成



完全同期



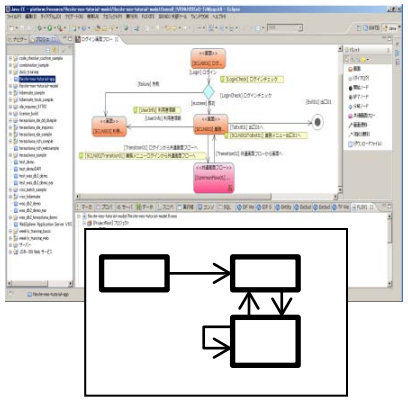
導入準備 → 設計 → 製造・単体試験 → 試験

プロジェクトの固有性を組み込んだカスタマイズ
（場合によってはゼロカスタマイズで適用可能）

100%自動生成
製造・単体試験が削減できる！UTバグ削減

設計モデルの画面遷移情報からテストシナリオを自動生成
シナリオからテストコード、テストケース表を自動生成

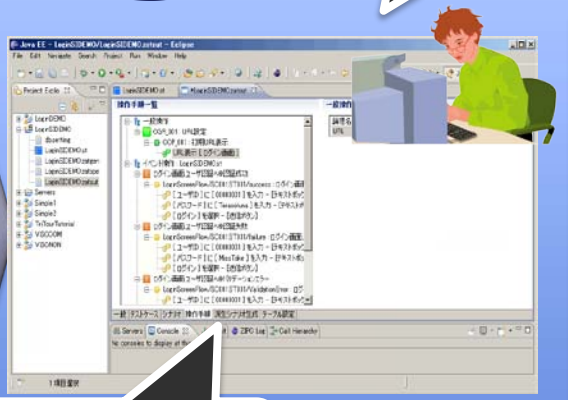
設計モデル



入力

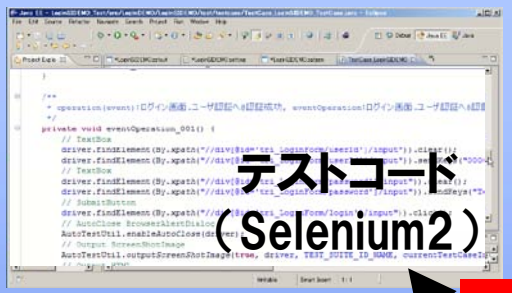
TERASOLUNA IDE

テストデータは開発者が入力



テストシナリオを抽出

自動生成



一致

キャッツと協業し、機能結合テストの実行を自動化

TERASOLUNA IDE ver.3

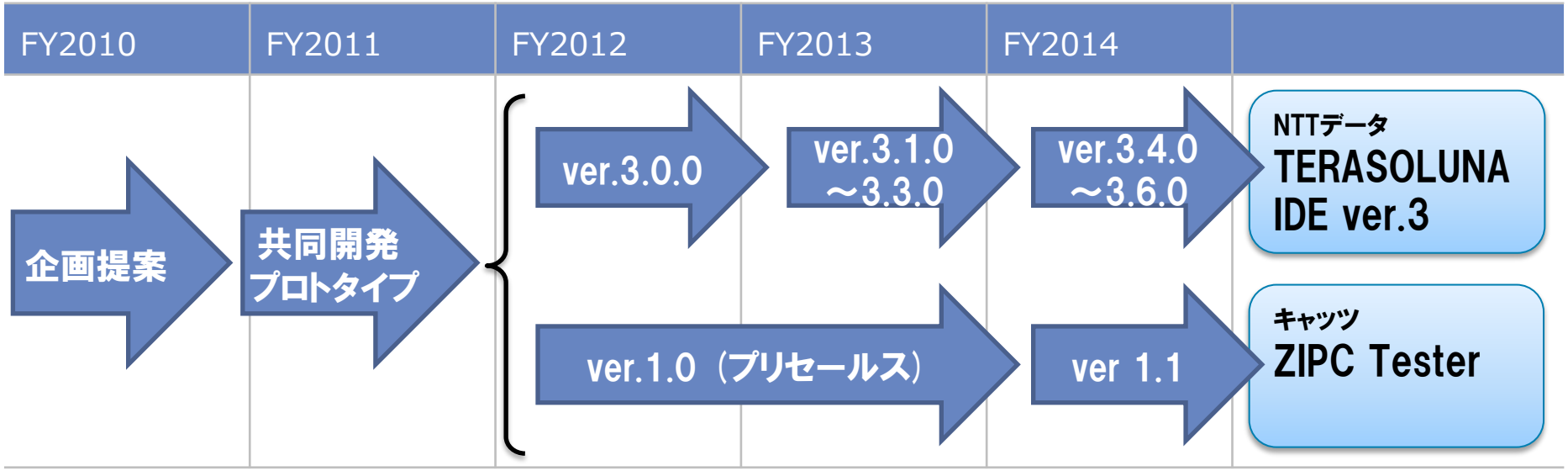
協業によりテストケースの設計ツールを共同開発

TERASOLUNA IDE ver.3

- NTTデータの社内ツール
- 情報システム分野向け
- テストケースを設計し、テストの実行を自動化するソースコードを生成

ZIPC Tester

- キャッツの商用ツール
- 組み込み分野のテストに向け
- テストケースの設計（基本機能）を製品として販売し、カスタマイズを受注



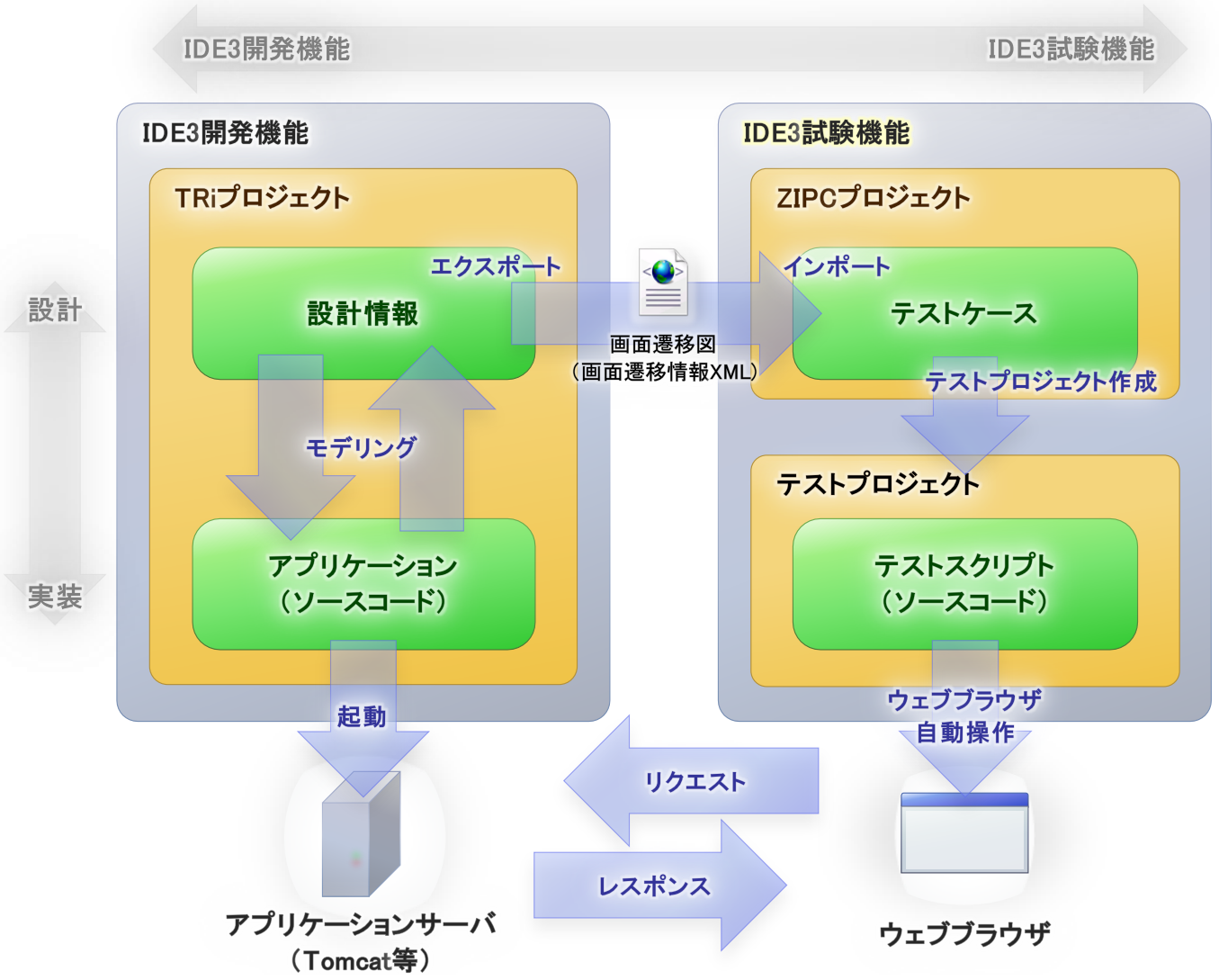
試験機能が解決する課題

NTT DATA

人手でテストケースの設計や実行を行う場合に起こりやすい、以下の課題を解決

課題	解決法
設計情報の解釈の誤りにより、テストケースに誤り（試験バグ）を埋め込む	設計情報を持ったツール上でテストケースを編集することによって、設計情報とテストケースの不整合を防ぐ
正常系のシナリオは作りやすいが、異常系のシナリオは見落としやすく抜け漏れが発生しやすい	基本的な画面遷移のシナリオをベースとして、遷移を追加した派生シナリオを自動生成する
開発者によってテストケースの解釈や実施方法にブレがあって、バグの発見や再現が難しい	テストケースを自動的に実行するためのプログラム（テストコード）を自動生成する

設計情報をテストケースの編集と自動実行に活用



設計情報から画面遷移図をエクスポートし、ZIPCの状態遷移モデルとしてインポート

テストケース表だけでなく、テスト自動化フレームワーク（Selenium）を用いたテストコードを自動生成

テストコードがWebブラウザを操作してテストケースを自動実行し、証跡を記録

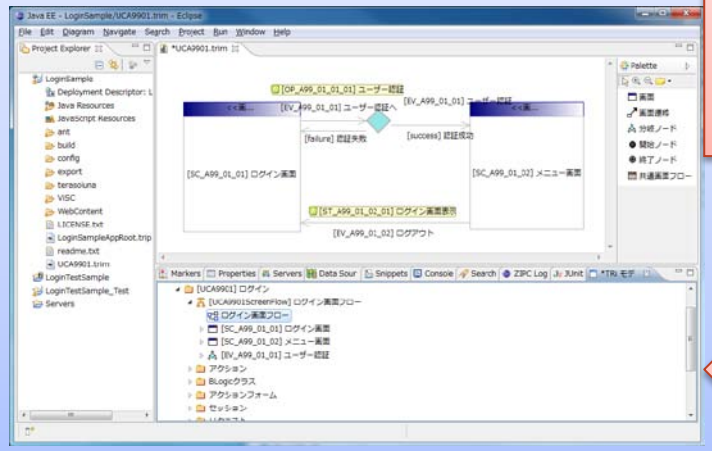
Java Webアプリケーションの設計を支援し、製造を自動化

(TERASOLUNA Server Framework for Java を利用)

① **画面項目や画面遷移などの設計情報を入力**

③ **ソースコードのひな形を編集し，画面デザイン，業務ロジックを製造**

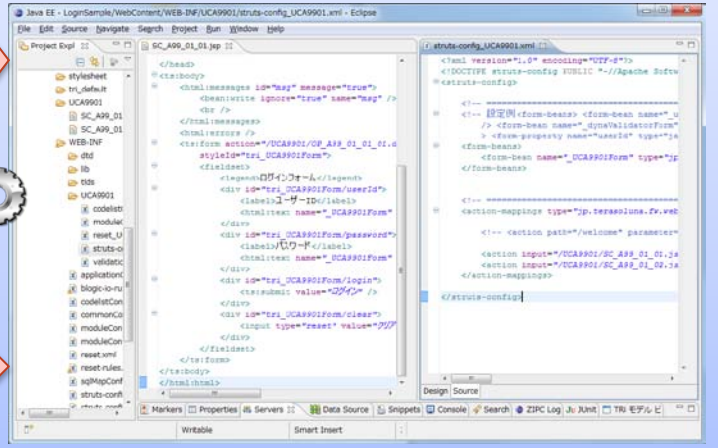
設計



②ソースコードのひな形・設定ファイルを自動生成

④変更内容を同期

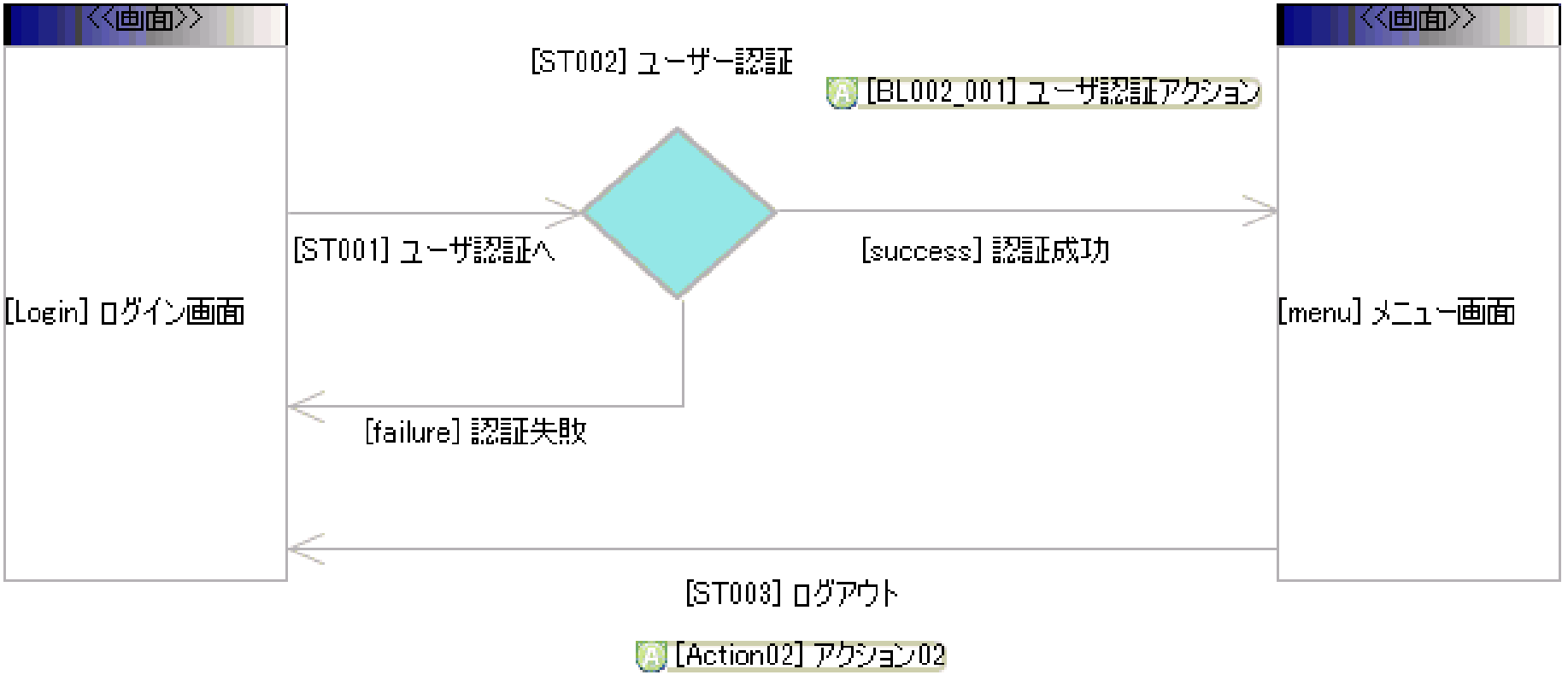
製造（コーディング）



簡単な画面遷移図の例

NTT DATA

Eclipse ベースの GUI により画面遷移図や画面の表示内容を編集することにより、ソースコードを自動生成

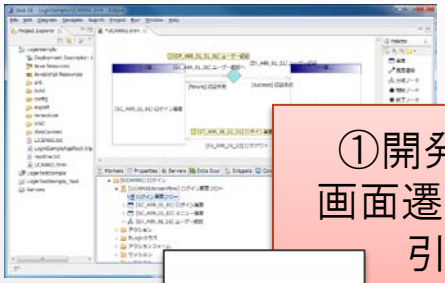


設計情報を活用してテストケースを編集し、テストを自動実行するためのスクリプトを出力

② 代表的なテストケースを入力し，バリエーションを生成させる

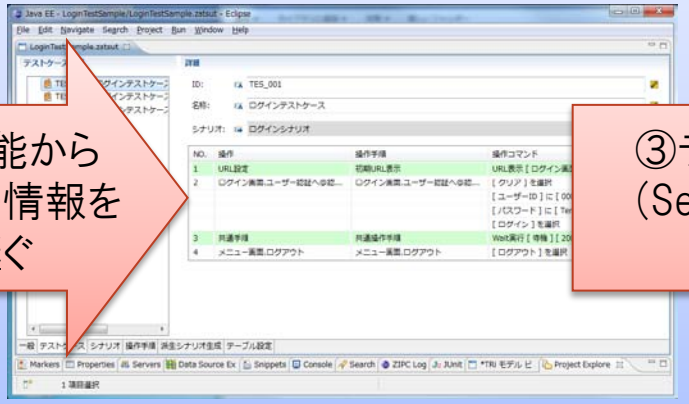
④ テストスクリプトを実行し，証跡を確認する

設計（入力済）



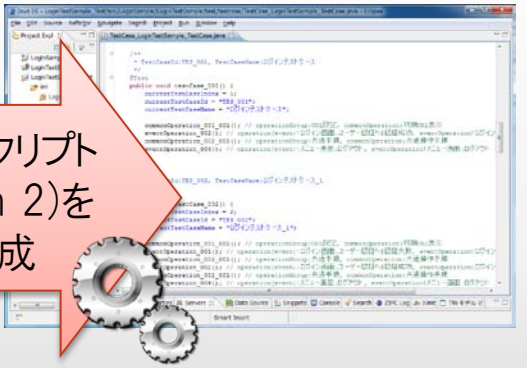
設計情報（XML）

テストの準備



①開発機能から画面遷移の情報を引き継ぐ

テストの実施



③テストスクリプト（Selenimum 2)を自動生成

試験機能を用いたテストの手順

NTT DATA

1. 基本シナリオの設計

- 開発機能から画面遷移図、画面設計の情報を引き継ぐ
- 機能を実行するための基本的な画面遷移の系列（パス）を指定

2. 派生シナリオの生成

- ZIPC Testerと共通の経路抽出エンジンを採用
- 基本シナリオをベースとして，寄り道などの条件に応じたシナリオを自動生成
- 生成されたシナリオを「派生シナリオ」と呼ぶ

3. テストデータの割り当て

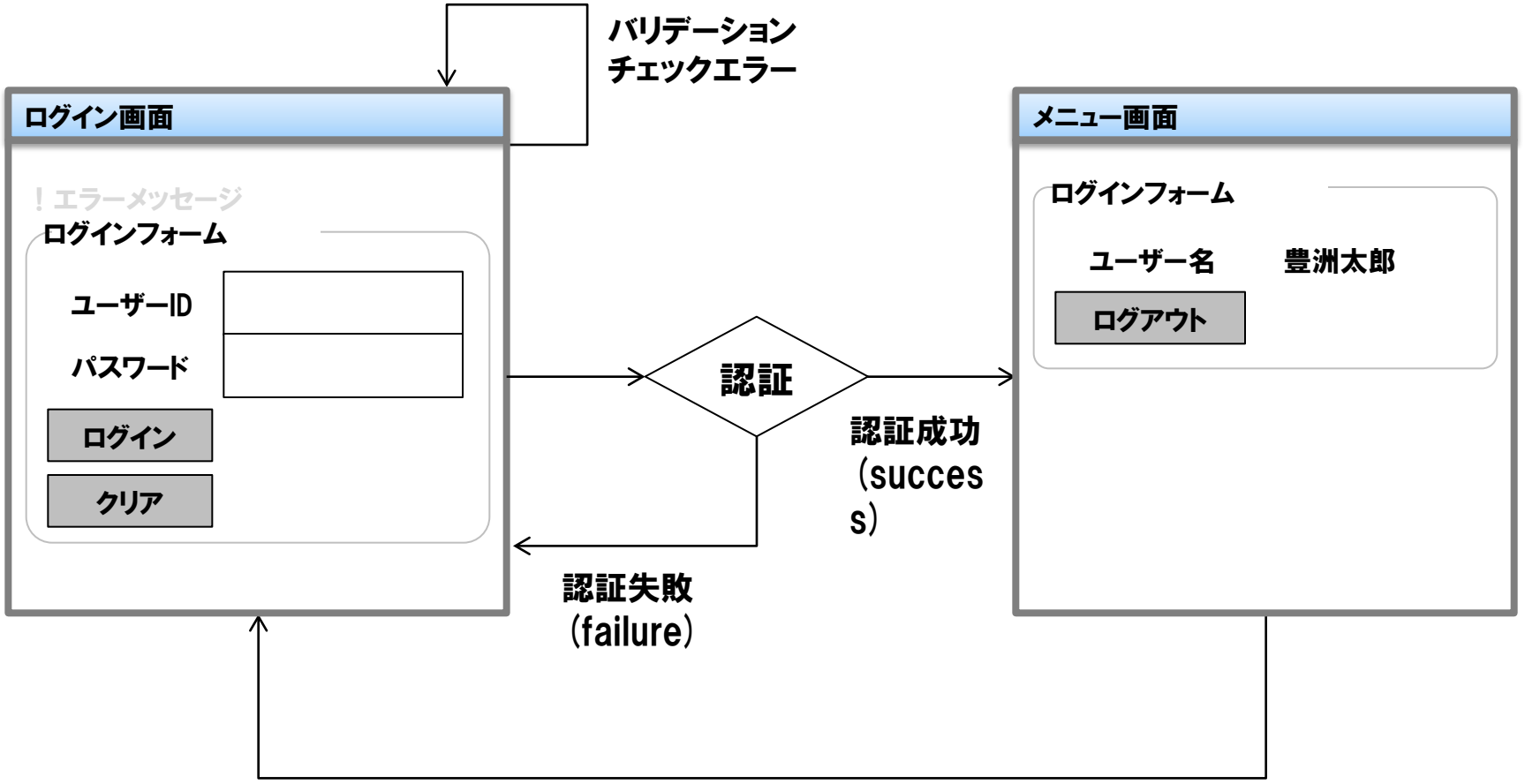
- 基本または派生シナリオに対して，シナリオ内の各画面での入力値を編集
- 画面上に表示されるボタンやリンクは画面遷移から自動的に決定される

4. テストコードの生成と実行

- Webブラウザやデータベースの種類を指定する
- テストケースを自動的に実行するためのテストコードを生成する

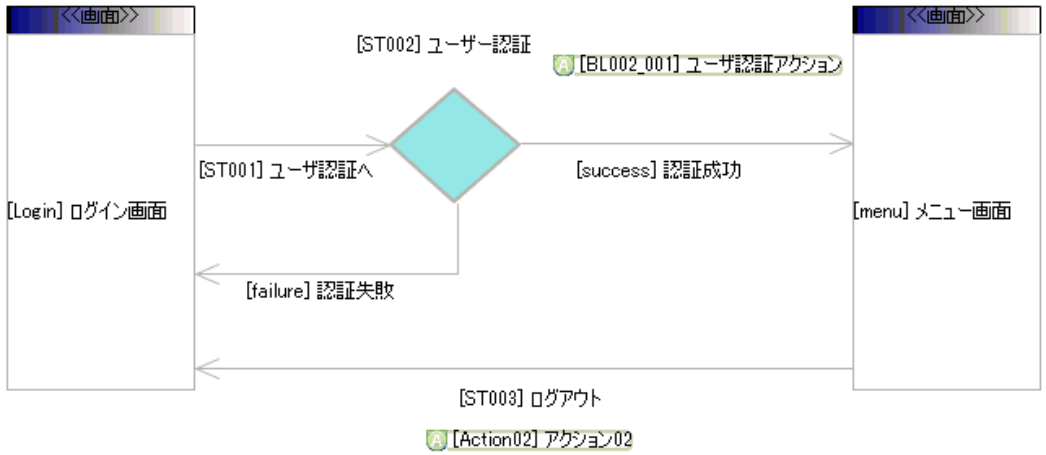
例題

ログイン画面とメニュー画面だけの簡単な画面遷移を例として、試験機能の利用イメージをご説明します。



1. 基本シナリオの設計

開発機能から画面遷移の設計情報をインポートし、テスト実行の基本となるシナリオを編集する



画面遷移図

インポート

jointNameSI	S	入力画面	加算結果画面	減算結果画面	入力画面	出力画面
E		0	1	2	3	4
入力画面.加算	0	加算結果画面	×	×	×	×
入力画面.加算@バリデーションエラー	1	入力画面	×	×	×	×
入力画面.減算	2	減算結果画面	×	×	×	×
入力画面.減算@バリデーションエラー	3	入力画面	×	×	×	×
入力画面.名前連結	4	×	×	×	出力画面	×
入力画面.計算機能へ	5	×	×	×	入力画面	×

状態遷移モデル（試験機能）

設計情報の解釈を誤って無効なテストを作るかも!

画面遷移の系列（パス）を順番に指定するだけ

基本シナリオ

シナリオ一覧

- SNR_001：減算入力チェックシナリオ
  - SNR_001_001：減算入力チェックシナリオ_1
- SNR_002：加算入力チェックシナリオ
  - SNR_002_001：加算入力チェックシナリオ_1

詳細

ID: SNR_001_001

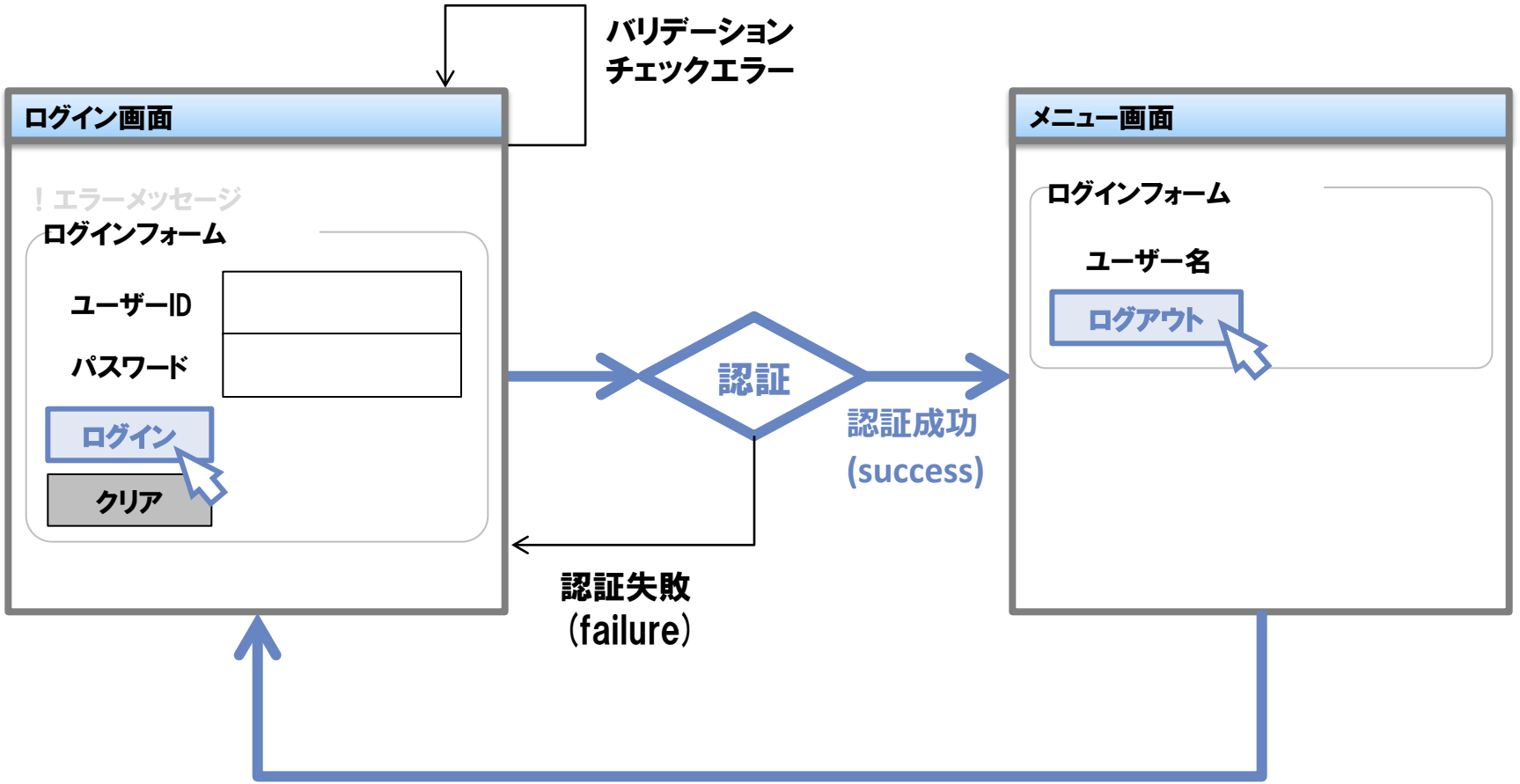
名称: 減算入力チェックシナリオ_1

NO.	遷移元	イベント
1	入力画面	入力画面.計算機能へ
2	入力画面	入力画面.減算@バリデーションエラー
3	入力画面	入力画面.減算
4	減算結果画面	

一般 | テストケース | シナリオ | 操作手順 | 派生シナリオ生成 | テーブル設定

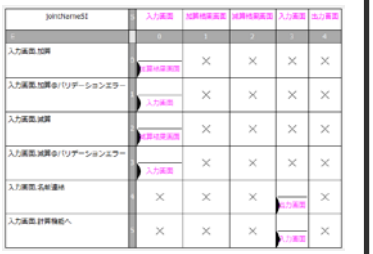
例題: 基本シナリオの設計

ログイン画面から認証処理に成功してメニュー画面に遷移し、元のログイン画面に戻る

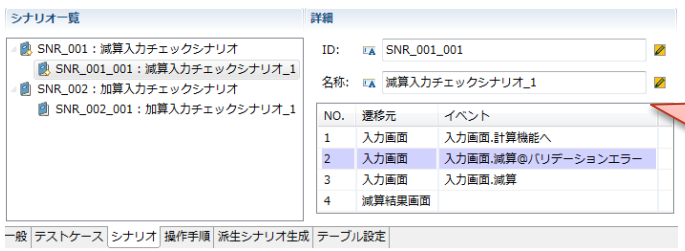


2. 派生シナリオの生成

画面遷移を網羅するため、基本シナリオに含まれていない遷移（寄り道）を追加して新たなシナリオを生成する



状態遷移モデル

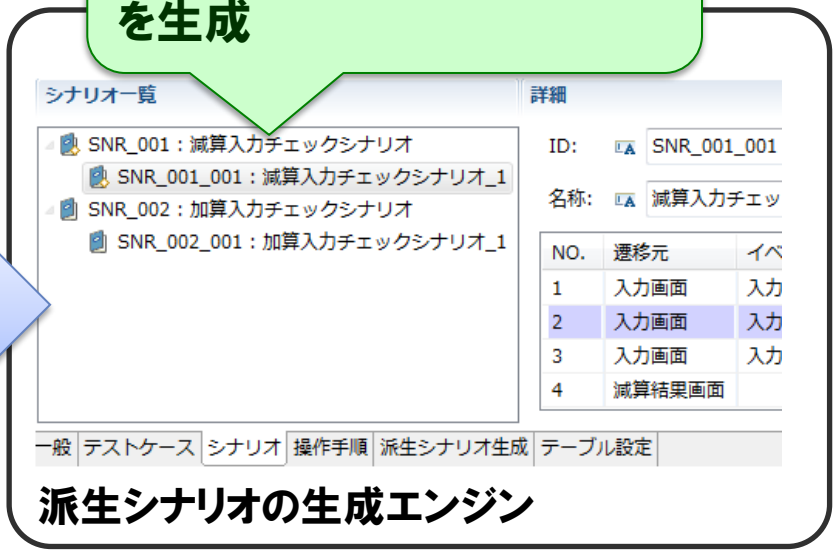


基本シナリオ

画面遷移を網羅するためのテストシナリオの作成が大変

基本シナリオをベースに遷移を追加し、派生的なシナリオを生成

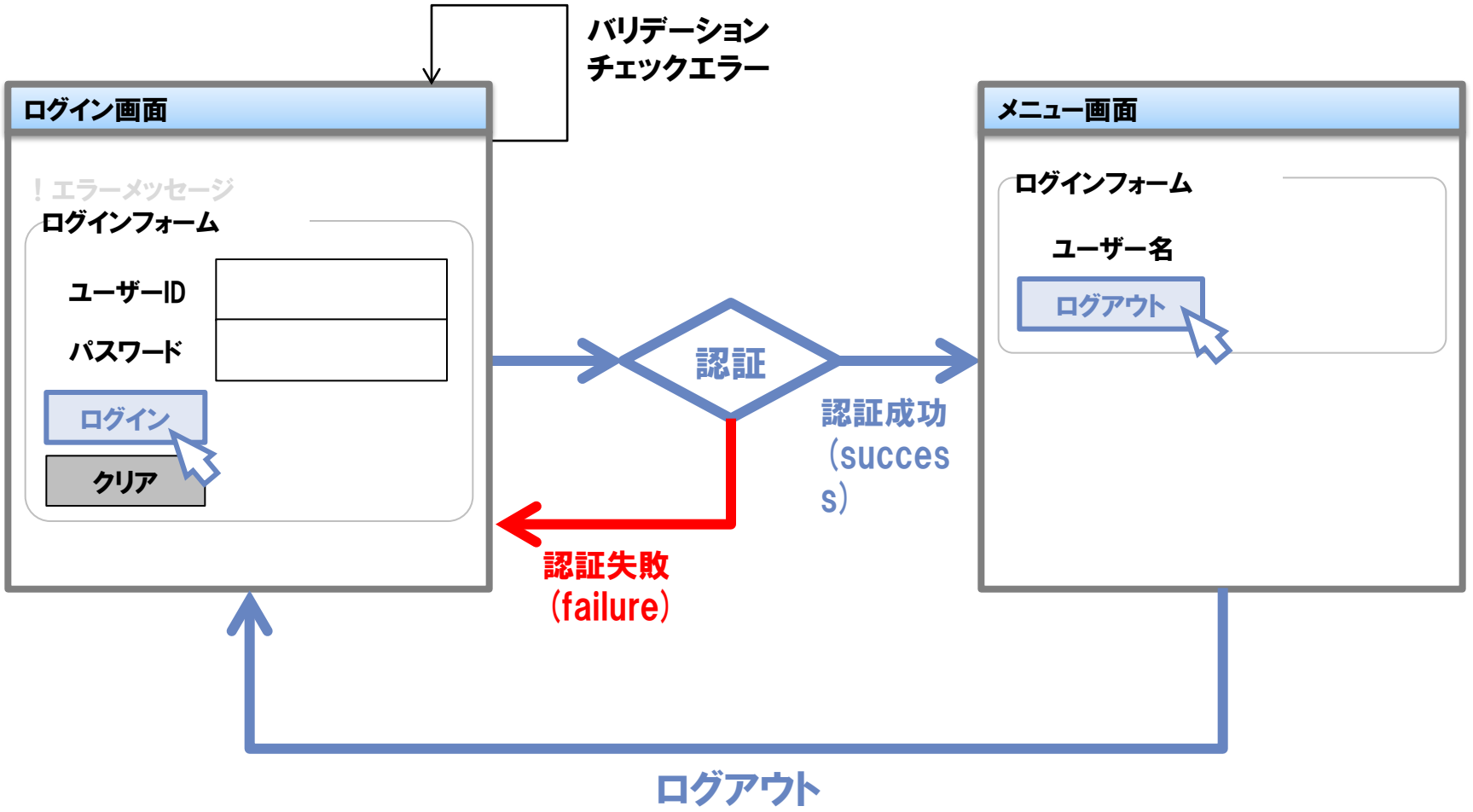
派生シナリオの生成エンジン



派生シナリオの生成エンジン

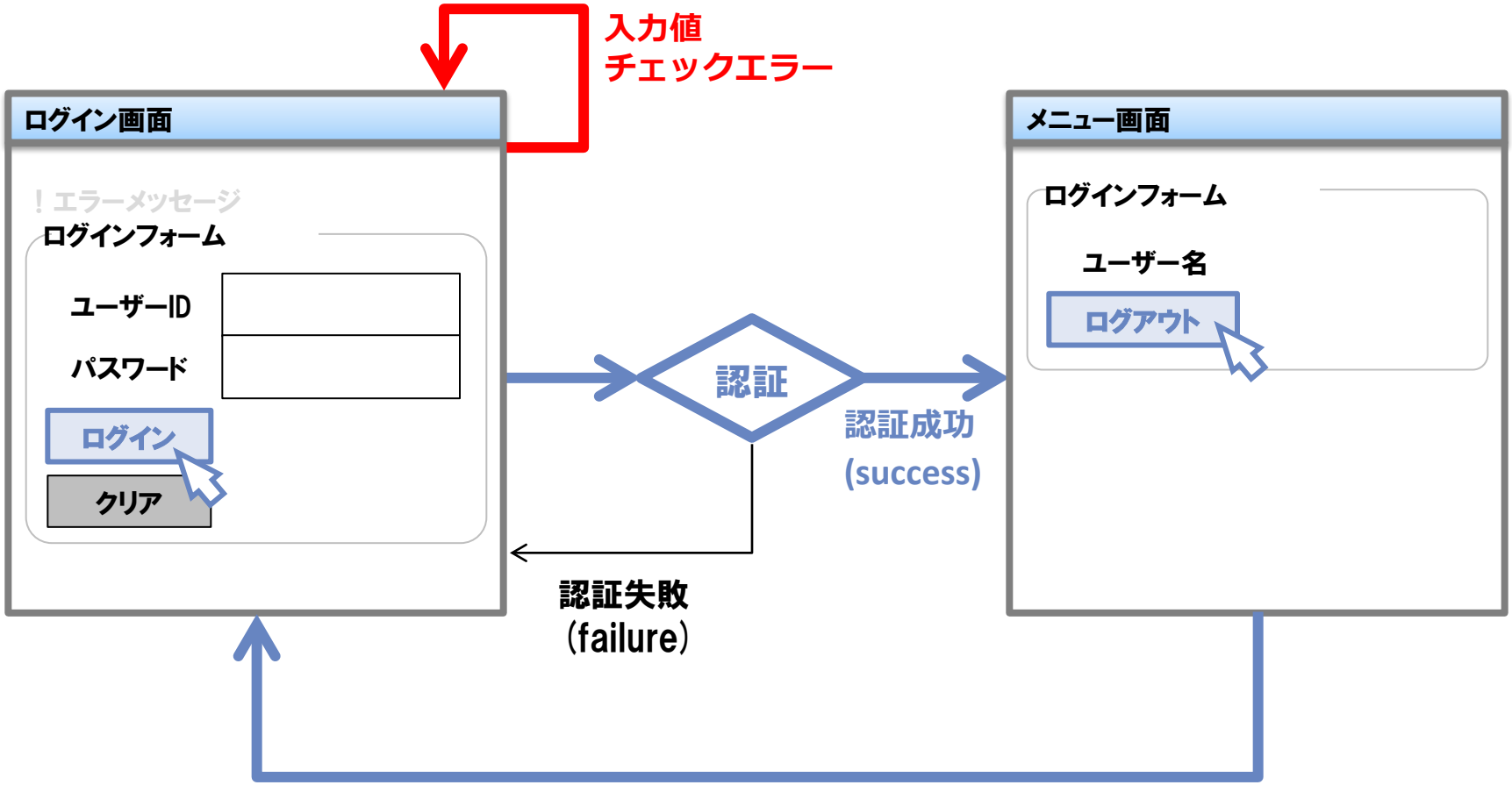
例題: 生成される派生シナリオ①

ログイン画面で認証処理に失敗し、二回目で成功して
メニュー画面に遷移し、元のログイン画面に戻る



例題: 生成される派生シナリオ②

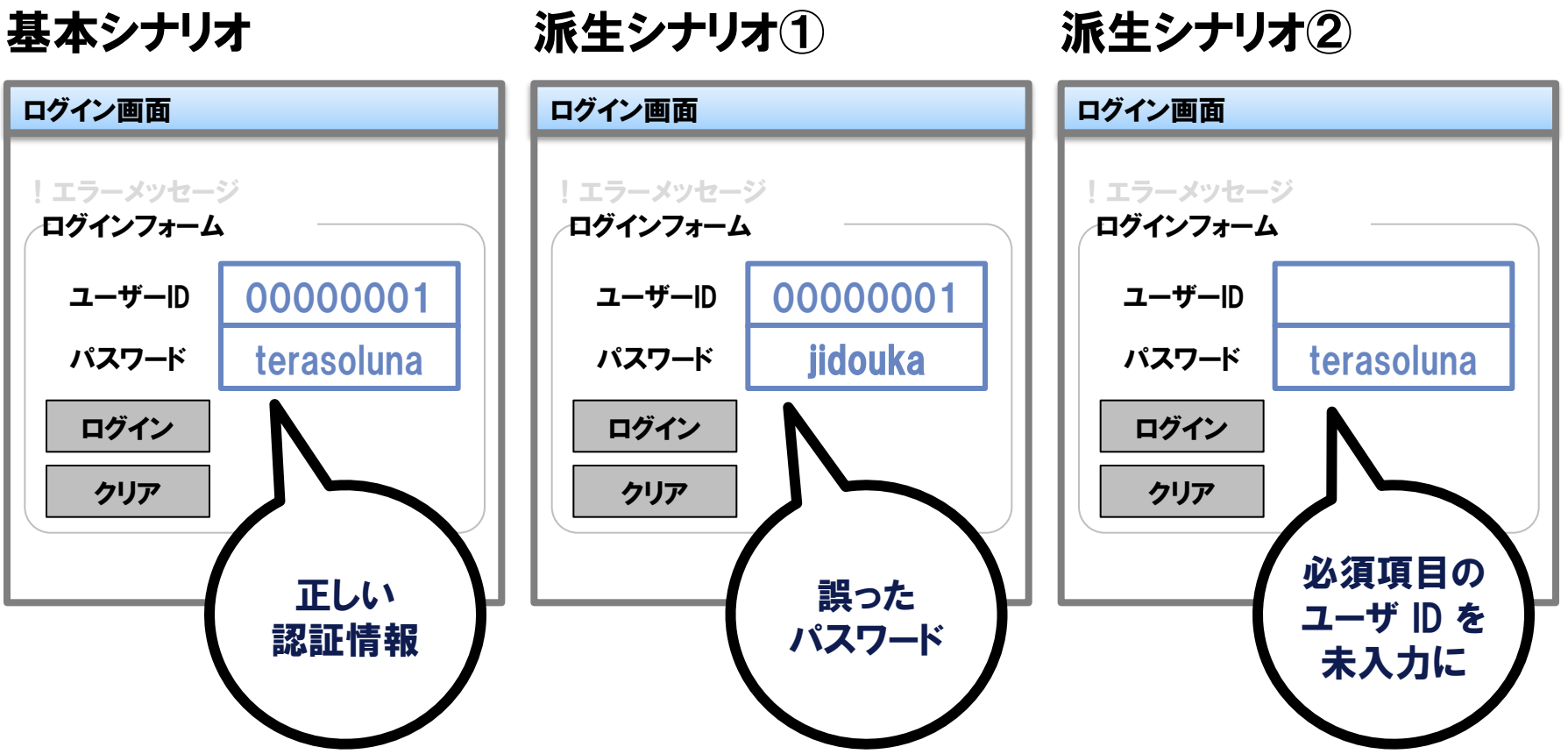
**ログイン画面で入力値チェックに失敗し、二回目で成功して
メニュー画面に遷移し、元のログイン画面に戻る**



例題: テストデータの割り当て

NTT DATA

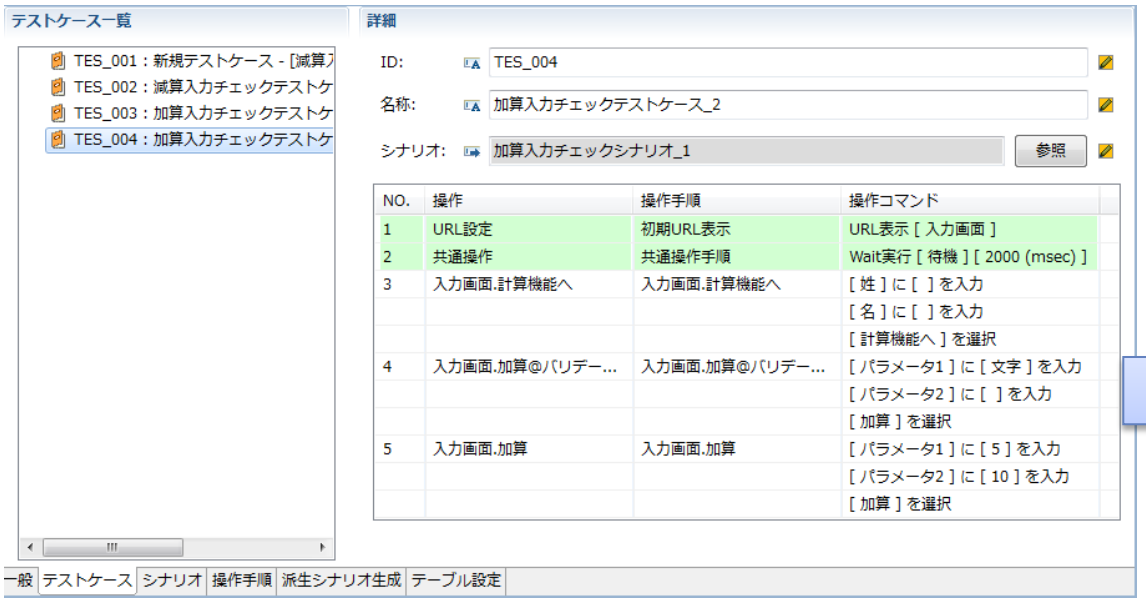
基本シナリオ・派生シナリオの各画面で入力する テストデータを設計する



テスト実行自動化フレームワーク（Selenium）を用いた テストコードを自動生成し、テストケースを自動的に実行

- テストケースとテストコードの整合性を担保
- テストケース表（Excel 形式）も出力可能
- 画面証跡を自動取得

テストケースの実施を自動化し、バグの再現や修正後の再試験を容易化



テストケース

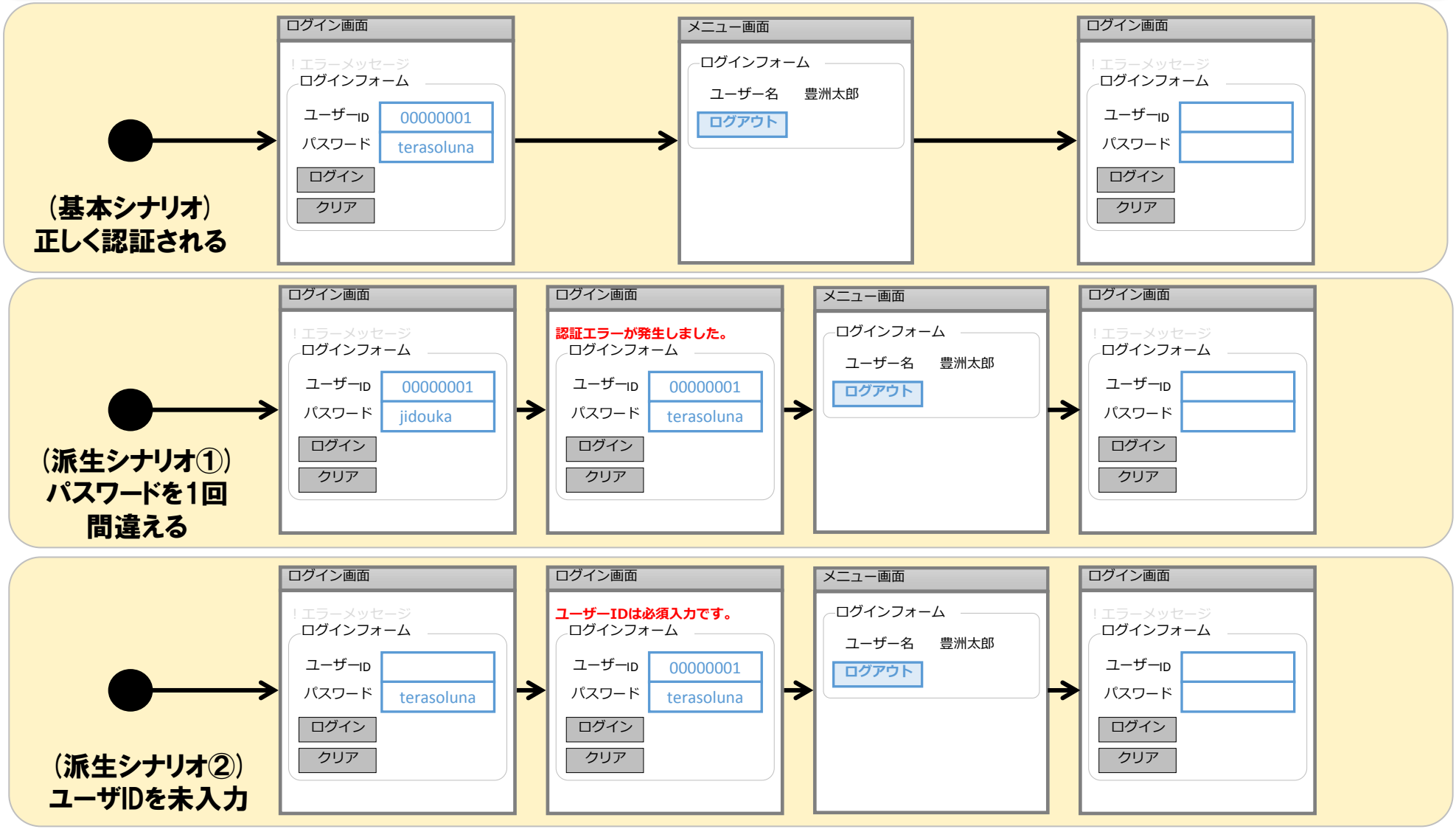
```
/**↓
 * TestCaseId:TES_004, TestCaseName:加算入力チェッ
 */↓
@Test↓
public void testCase_004() {↓
    currentTestCaseIndex = 4;↓
    currentTestCaseId = "TES_004";↓
    currentTestCaseName = "加算入力チェックテスト
    ↓
    commonOperation_001_001(); // operationGroup:
    commonOperation_002_001(); // operationGroup:
    eventOperation_006(); // operation(event) : 入
    eventOperation_002(); // operation(event) : 入
    eventOperation_001(); // operation(event) : 入


/**↓
 * operation(event) : 入力画面.加算, eventOperation
 */↓
private void eventOperation_001() {↓
    // TextBox↓
    driver.findElement(By.xpath("//div[@id='tri_Fo
```

テストコード（Selenium）

例題: テストコード自動実行のイメージ

Webブラウザを自動操作してテストを実行し、証跡を記録



おわりに

NTT DATA

本講演ではNTTデータグループの生産技術革新の中核となる「TERASOLUNA Suite」をご紹介しました。

NTTデータ

- 大規模な情報システム開発のノウハウ蓄積
- 生産技術革新のための R&D

TERASOLUNA®

×

キャッツ

- 組み込み分野を中心としたツール開発の豊富な実績
- 状態遷移ベースの自動化技術における強み

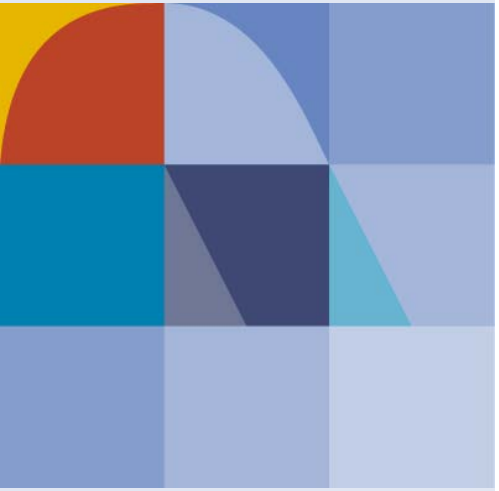
ZIPC® Tester

ソフトウェア開発自動化

生産技術の革新によって
労働集約型から知識集約型の開発スタイルへ

株式会社NTTデータ
技術開発本部
TERASOLUNA窓口

Phone: 050-5546-2482
E-mail: terasoluna@kits.nttdata.co.jp
URL: http://www.terasoluna.jp



NTT DATA

Global IT Innovator