4 UML2.0 Testing Profile/ Eclipse Test & Performance Tools Platformの解説

小野塚荘一
日本IBM

モデル駆動型開発の現状

ソフトウェア開発モデル化について，情報システムの各開発フェーズのうち，要求分析・設計フェーズ用の開発手法・ツールは研究・開発が進んでおり実用化されてきている．たとえば，要求仕様のモデルがUML（Unified Modeling Language）で表記され，モデルからコードを生成する手法・ツール等がある．

モデルを使うことにより“複雑さ”や“詳細さ”を隠し，理解が容易になる．すなわちソフトウェアの分析設計において，維持管理の容易性，再利用性，コミュニケーションの向上（さまざまな利害関係者が業務を理解するために必要な概念と，情報システムとして実現するためのプログラミング言語との間にある意味体系のギャップを減少させる）などのメリットを享受できるようになる．

テストのモデル化の現状はどうであろうか．前述のモデル駆動型開発のテスト版にあたるモデル駆動型テストは，モデル駆動型開発の分析・設計フェーズにおいて作成したモデル検証の色彩が強い．テストケースなどの試験フェーズでの成果物モデル化への取り組みついては，近年研究開発が進みつつある．実用段階に達したテストのモデル化の例として，以下の2つがよく知られている．

［UML2 Testing Profile (U2TP)］

UMLを使ってテストのモデルを定義したもので，Object Management Group（OMG）によって開発された．このモデルはオープンソースのEclipse Test and Performance Tools Platform（TPTP）のテストツールとして実現されている．

U2TPの仕様のサイト
http://www.omg.org/technology/documents/formal/test_profile.htm

［Testing and Test Control Notation (TTCN-3)］

当初，電気通信とデータ通信機器のテスト目的で，ETSI（欧州電気通信標準協会）と通信方式の国際標準化作業を手がけるITU（国際電気通信連合）により開発された．

TTCN-3のサイト
http://www.ttcn-3.org/

本稿では上記のうち，今後幅広い分野で適用が期待されているU2TPを中心にテストツールの基盤としてのEclipseとの連携も含めて解説する．

次章では，ソフトウェア開発の分析・設計フェーズでのモデル化を前提に，試験フェーズにおけるテストのU2TPによるモデル化について解説する．

U2TPを利用したテストのモデル化

U2TPは，テストシステムの設計，視覚化，仕様化，分析，構築，文書化を容易にするために定められた表記法である．使用される分野は問わない．

U2TPでテストシステムをモデル化することにより試験フェーズのさまざまな成果物の関連の明確化が可能になり，再利用率が向上するなどで，テストの生産性向上が期待できる．筆者が考える具体的な適用方法，メリットについて以下に述べる．

［テスト概念・用語のモデル定義による明確化］

一般的なテストの準備作業では，用語集や作業手順書を準備し，プロジェクト作業の標準化を図るケースが多い．用語集は外部の標準化団体の用語集を参考にしている場合もある．たとえば，Japan Software Testing Qualifications Board（JSTQB）が提供している，ソフトウェア・テスト標準用語集（日本語版）などが利用される．

用語集の問題は，用語の関連についての情報が希薄なことで，たとえば，テストケースとテストデータやテスト・ログがどのように関係しているかを明確にするためには，作業の手順書など別の成果物を参照しなければならない．

概念・用語をモデル化することで，複雑さや詳細を隠蔽し，それぞれの関連について分かりやすく表現するこ

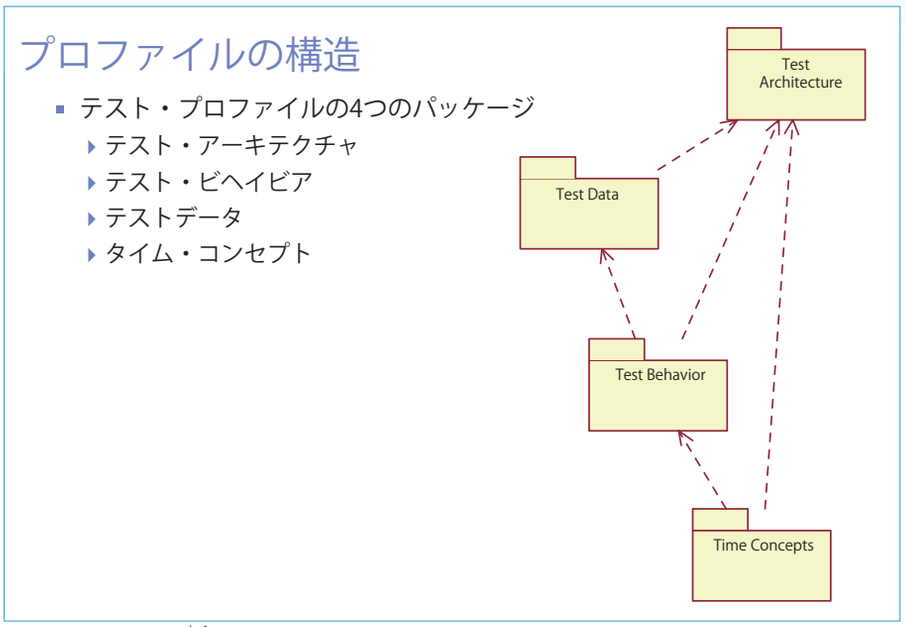

図 -1　U2TP の構造☆1

とが可能になる．また，プロジェクト固有の状況を反映させて作成したテスト・モデルは手動や自動を問わず，テスト成果物の関連を示す重要なドキュメントとなる．

［テスト・プロセスの記述力向上］

U2TP では，テストをシステムとしてモデル化しているが，モデルの各要素について作成／実施の開始条件，作成者（役割），作業フローなどのプロセスの記述は定義されていない．そのため，U2TP で定義したテスト・モデルを基にテスト作業を進めるためには，プロセスの表記法と連携が必要になる．モデル駆動型開発でも作業のプロセスの表記法については特に定義がなく，プロセスの表記法と併せて使われる．

ソフトウェア開発プロセスの記述用に OMG によって整備された Software Process Engineering Metamodel（SPEM）がある．これにより，テスト成果物の作成について，役割，作業，ワークフロー等のプロセスを明確に記述できる．テストの成果物と作業フロー等のプロセスの両方が明確になることで，開発・テスト作業の隘路を工数や障害の数などから特定し，改善を実施することが可能になる．

［自動テストツールの設計仕様の明確化］

Eclipse Test & Performance Tools Platform（TPTP）は，テストツールの概念モデルから実際に動作するテストツールの分析・設計・開発まで一貫して行われた例としてよく知られている．

TPTP のプロジェクトでは，コンピュータに依存しないテストツールの概念モデルである U2TP を使用し，コンピュータでの動作を前提としたテストツールのモデルが開発された．このように，U2TP はこれから新規に自動テストツールを開発する際の標準的な仕様の記述法としても利用可能である．

テストの自動化を推進している他の例として，IBM Haifa Research Lab（イスラエル）で研究・開発が進められている GOTCHA（Generation of Test Cases for Hardware Architectures）などがある [1]．

［アスペクト指向技術を応用したテストへの応用］

アスペクト指向の基本的な考え方は，「横断的関心事」と呼ばれる，オブジェクトに散在する処理を単一の「アスペクト」として本体のロジックから分離することで，ソフトウェアの開発効率や保守性の向上を目指すものである．

たとえば，ログの出力処理が「横断的関心事」の代表例である．このログを確認するテスト・プログラム，テストケースを U2TP で記述することによりテスト・プログラム・インタフェースの定義が容易になると考えられる [2]．

本章で紹介したテストのモデル化のメリットを実現するために，テストの概念を定義したものが，U2TP である．次章では U2TP の内容について解説する．

U2TPのパッケージの詳細

U2TP では，テスト･コンポーネント，テストケース，テスト目的などのテスト固有の概念について定義している．U2TP では 26 の概念について定義しており，これらの概念をテスト・アーキテクチャ，テスト・ビヘイビア，テストデータ，タイムの 4 グループにパッケージとして分類している（**図 -1**）．

それぞれ，テスト・アーキテクチャはテスト成果物の静的な構造・構成関連，テスト・ビヘイビアは動的側面のテスト実行関連，テストデータはテスト実行で使われるテストデータ関連，タイムはテスト実行で定義される時間関連について記述されている．

各パッケージについては，OMG から UML 2.0

☆1 図 -1 は U2TP のパッケージ間の関係を UML パッケージ図で表記したもので，破線と矢印はパッケージ間の依存関係を示す．

Testing Profile Specification として公開されている．モデルのダイアグラムも各パッケージについているためぜひ参照いただきたい．ここでは各パッケージについての要点とその中で定義されている代表的な概念について解説する．

［テストアーキテクチャ・パッケージの特徴］

テストアーキテクチャ・パッケージは，テストシステムの静的構造を定義したものである．これはテスト・コンポーネント間およびテスト対象システム（System Under Test：SUT）とのインタフェースとして記述される☆2．

このパッケージの特徴について解説する．「テスト・コンフィグレーション」は，テスト・コンポーネントとSUT の関連から構成される．複数のテストケースをテスト・コンフィグレーションごとにグループ化することで，「テスト・スイート」が構成される．

「テスト･スイート」から得られるテスト結果は，「アービター」（Arbiter；調停者）が取りまとめ，最終的なテスト結果を算出する．

実際のテスト作業で，テストケース数やテスト結果が膨大になり，その結果，管理や取りまとめのための作業工数が膨大になる場合がある．そこであらかじめ管理や取りまとめのための成果物を用意しておき，管理向けに処理された情報を使うことで取りまとめの工数の増大を防ぐことが可能になる．

野球の例で考えると，アウト・セーフの判定と試合の勝ち負けの判定のロジックは異なる．アービターは試合の勝ち負け判断の概念である．アウト・セーフをひとつひとつ確認する概念をバーディクト（Verdict；判定）としている．

［テスト・ビヘイビア・パッケージの特徴］

テスト・ビヘイビア・パッケージは，テストがどのようにふるまうか(動的側面)について定義したものである．

このパッケージの特徴について解説する．「テストケース」はテスト・ビヘイビア・パッケージの中心となる概念で，次のように定義されている．

- 「テスト目的」に対して，テスト・コンポーネントが「SUT」をどのように検証するか，そのふるまいを定義する．
- テスト結果として「バーディクト」を返す（先の野球の例では，アウト・セーフの判定にあたる）．
- 「評価アクション」の結果が「アービター」に渡され，結果の総合的な評価を行う．

「テストケース」でどのように（How）テストを実行するか，「テスト目的」で何を（What）テストするかについての記述をする．「テストケース」からは「評価アクション」，「ログアクション」などが関連付けられて，さらにテスト実行プログラムとの関連付けを行う．

このパッケージでは，テスト目的→テストケース→テスト実行→テスト結果→総合判定と一連のテストの成果物の流れについてふるまいを定義する．

［テストデータ・パッケージの特徴］

テストデータ･パッケージは，SUT へ送信されるデータ，SUT から得られる観測データなどのデータ型や値を定義するために用いられるパッケージである．

このパッケージの特徴について解説する．

「データプール」はテストデータの表を定義している．「データプール」は「テスト･コンポーネント」によって「テストケース」実行時の変数の値として使われる．

［タイム・パッケージの特徴］

タイム・パッケージは，実行タイミングおよび実行継続時間を定義したものである．

次に，このパッケージの特徴について解説する．

「タイムゾーン」で「テスト･コンポーネント」のグループ化を定義する．各「テスト・コンポーネント」は「タイムゾーン」に属する．

「タイマー」は一定時間経過後のタイムアウトを生成する．

U2TP で定義された概念をベースに開発されたテストツールが TPTP のテストツールである．U2TP と TPTP 連携の解説の前に，次章では，Eclipse プロジェクトにおける TPTP の位置づけについて解説する．

Eclipseプロジェクトにおける TPTPの位置づけ

Eclipse TPTP は，一般にオープンソースのテストツールとして知られ，TPTP プロジェクトでは，単体テスト，パフォーマンステスト，マニュアルテストなどのテストツールを提供している．本稿では，個別のツールの説明は避け，U2TP を応用した TPTP のテストツールの共通基盤としての価値について解説する．

Eclipse のプロジェクト・アーキテクチャを**図 -2** に示す．Eclipse はオープンソースの Java の統合開発環境やテストツールとして，広く知られている．これらのツール群に共通の機能を提供する基盤として，Frameworks, Tools Platform, Rich client Platform

☆2 インタフェースはクラスが外部に公開する操作であり，UML で記述される．

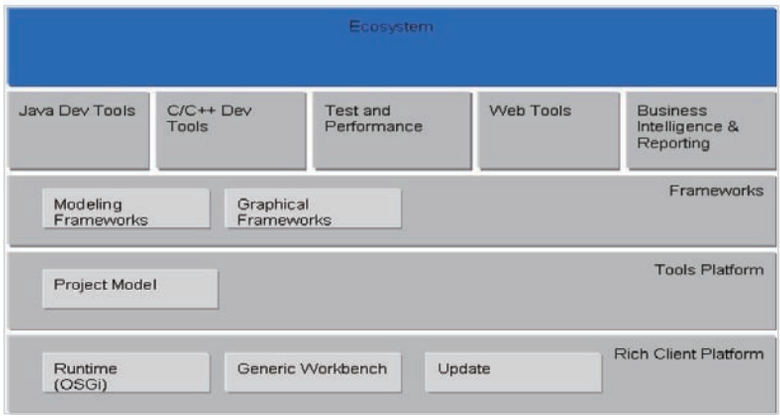

図 -2　Eclipse プロジェクト・アーキテクチャ ― Eclipse の代表的なプロジェクト（出典：http://www.eclipse.org/org/councils/roadmap_v2_0/AC_v2_0/index.php）

のプロジェクトがある．

TPTP は Test and Performance プロジェクトのことであるが，前述の基盤となるプロジェクトで開発された機能を使用している．

次に TPTP プロジェクトのサブプロジェクトを**図 -3** に示す．TPTP の中心となるサブプロジェクトは Workbench の中にある，プラットフォーム，テストツール，トレースとプロファイリング（Java メソッドの実行状況），モニタリング（メモリ，CPU などの使用状況），レポーティングの 5 つのサブプロジェクトで構成される．

TPTP プロジェクトの目的は，テスト実行／性能監視ツールのための共通基盤の提供であり，自動テストツールの開発・提供が目的ではない．

TPTP のサブプロジェクトで特に重要なプラットフォーム・プロジェクトのミッションは，ユーザ・インタフェース，データモデル，データ収集，通信コントロール，リモート実行環境，拡張機能に関連する共通のインフラストラクチャの提供である．**図 -4** には，分散環境でのテスト実行および実行結果のデータ収集のアーキテクチャを示す．左側が TPTP のツール本体でこちらからテスト実行，結果の表示を行う．右側がエージェントで分散環境でのテスト実行を可能にしている．

このアーキテクチャで，多くのテストツールに共通のテスト実行，結果表示，統計処理などの機能を提供しているため，新規にテストツールを開発するにあたって，TPTP ですでに実現されている機能を活用できる．

TPTP とテスト・モデル連携では EMF（Eclipse Modeling Framework）が使用される．EMF は一般に Eclipse の UML モデリングおよび UML モデルから Java コードの生成などの機能が知られている．

TPTP のテスト・モデルでは EMF データモデルが使われている．TPTP には次の 4 つの EMF データモデルが含まれている．これらは，①トレース，②ログ，③統計情報，④テスト実行で，この中で特にテストツールのテスト実行のモデルである④テスト実行が重要で，TPTP の中心となるモデルである．

次章では，この 4 つのデータモデルのうち④テスト実行のモデルが U2TP とどのように関連しているかについて解説する．

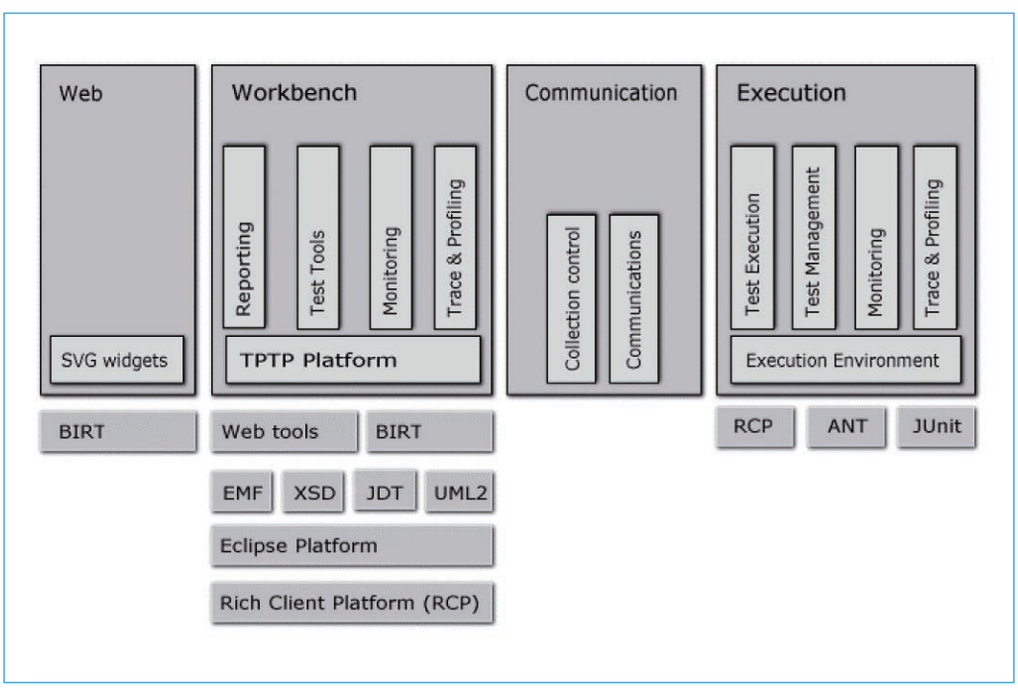

図 -3　Eclipse プロジェクトの中の TPTP プロジェクトの位置づけ ― Workbench が TPTP プロジェクト．5 つのサブプロジェクトがある．（出典：http://www.eclipse.org/org/councils/roadmap_v2_0/AC_v2_0/index.php）

4 UML2.0 Testing Profile/Eclipse Test & Performance Tools Platformの解説

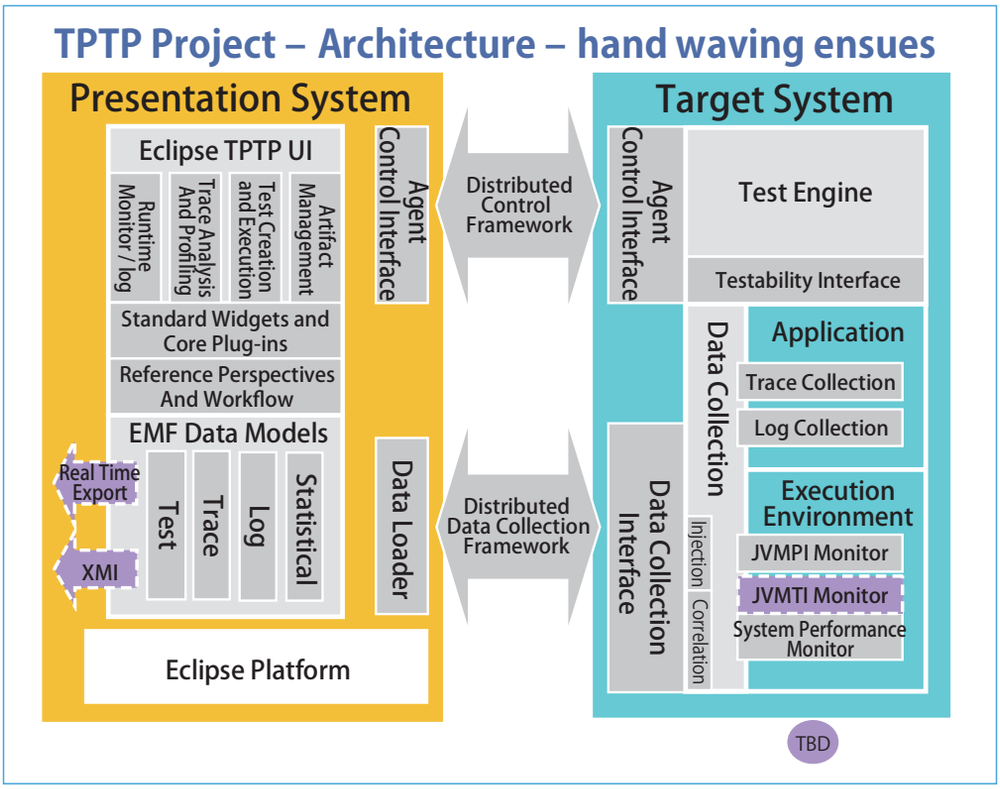

図 -4　Eclipse TPTP のアーキテクチャ

U2TP と TPTP

TPTP の概念モデルは「テスティング・プロファイル」,「ビヘイビア・モデル」,「実行ヒストリー・モデル」の 3 つのモデルにまとめられ，テスト・モデルまたはデータモデルと呼ばれている．**図 -5** は，TPTP の設計のベースになっている「テスティング・プロファイル」の UML 図であり，U2TP をベースに作成されている．図 -5 からは詳細な情報が読み取れないが，このダイアグラムは URL[☆3] で公開されているので，ご参照いただきたい．URL[☆3] からテスティング・プロファイルのダイアグラムを詳細に見ると，TestObjective, TestSuite, Behavior, Arbiter, TestCase, SUT, TestComponent など U2TP で定義された用語が TPTP のモデルとして使われていることが分かる．このように，U2TP のモデルを TPTP の「テスティング・プロファイル」として実現することで，概念モデルとしての U2TP と実体としての TPTP のテストツールが関連付けられる．

TPTP のテスト実行部分は「ビヘイビア・モデル」として記述される．テストの実行は，テスト･スイート（実行可能なテスト･プログラム）として定義され，ループ，同期などによって，ふるまいが定義される．「ビヘイビア・モデル」にはインタフェースが提供されている．インタフェースを使用するメリットは以下のとおりである．

- プログラミング意味論的に分かりやすくなる
- 異なるツールベンダ製品によるテストの相互運用性を推進する
- モデル化されていない他の TPTP ベースの製品を使用することを可能にする（両方のツールのインタフェース実装を経由する）．

テスト実行の結果については，「実行ヒストリー」のテスト・モデルで定義される．テスト実行中のログの定義，生成，管理，永続化についてのモデルである．

このように，TPTP では U2TP によって規定されたテストツールの概念に基づいてテストツールの開発が可能になっている．次章では，テストツールを便利に使うための TPTP を用いたテスト開発環境について解説する．

TPTPを用いたテスト環境開発のメリット

共通フレームワークとしての TPTP プラットフォームを利用して，テストツールおよびテスト環境を開発することで，Eclipse の標準機能を活用することが可能になる．このメリットを次に挙げる．

［操作性の向上］　複数の TPTP をベースに開発されたツールが共通の操作性になる．Eclipse を終了することなく，Eclipse 標準の機能（パースペクティブ）でツー

☆3 Rose※で作成された TPTP モデルの URL
http://www.eclipse.org/tptp/platform/documents/resources/models/index.htm
※ Rose は，IBM Rational の UML モデリングツールである．

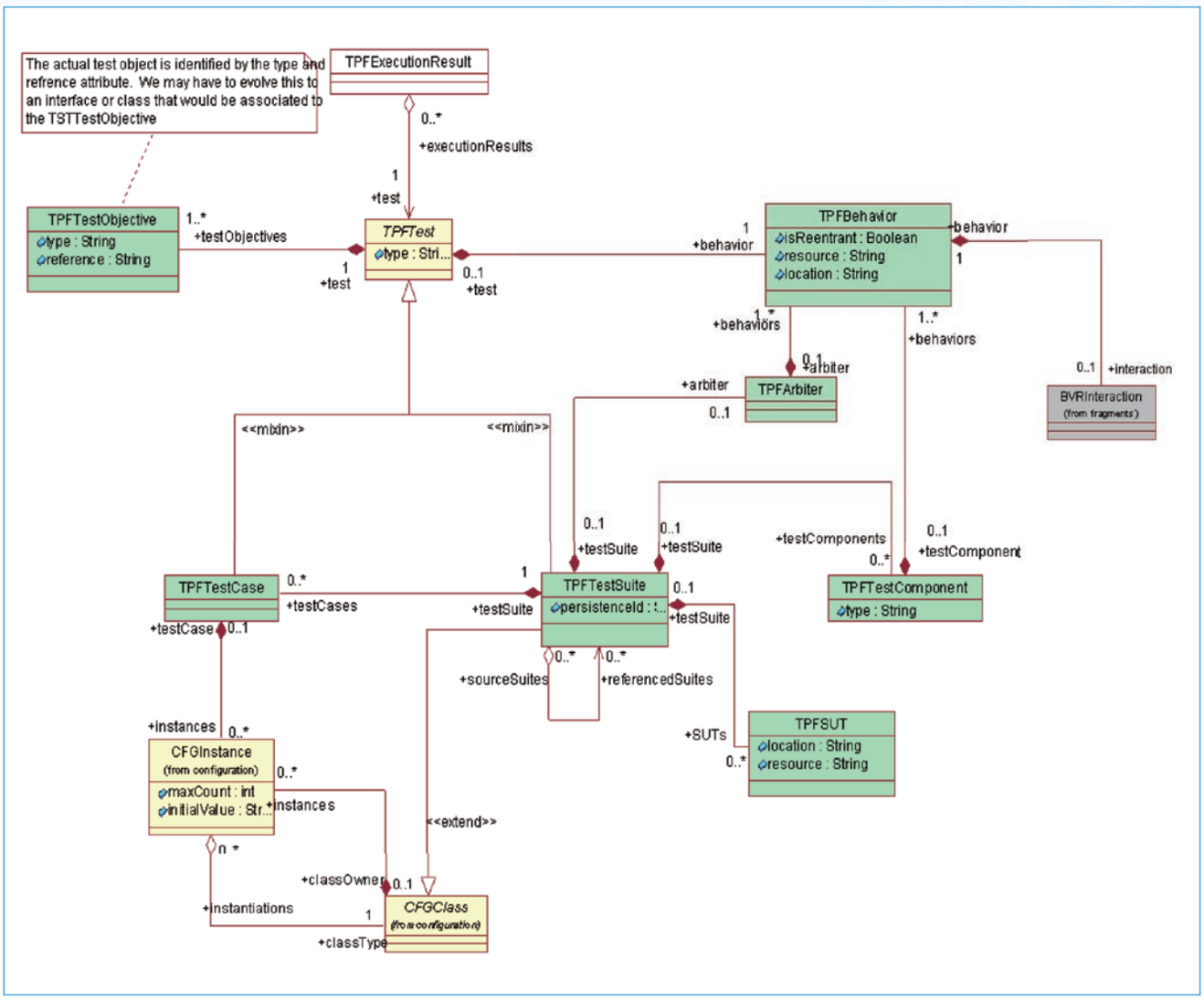

図-5　テスティング・プロファイルの UML 図（部分）
（出典：URL☆3 のリンクを参照．このリンクから，Logical View のツリーを展開し，common → testprofile → Main をクリック）

ルを切り替えることによって，他のツールを同じ操作性で使用することができる．

［分散テスト環境の実現］　TPTP では，複数台の構成による分散テスト実行やリモート・マシンからの監視等のデータ収集ができる．

［テストデータ・セットの変数化］　データプールが活用できる．データプールはテストの入力データと期待される出力データのセットを表で準備する．データの編集は手動入力または，事前にテストデータが作成された CSV ファイルのインポートを行うことで容易に実現できる．

［テスト・ログのフィルタリング］　TPTP のテストツールの実行結果はテスト･ログ･ビューワに表示され，フィルタ機能によりサマリーや詳細について抽出することができる．このビューから障害データベースへ障害情報の登録を行うことができる．

［テスト・レポート生成］　テスト結果の取りまとめは，Business Intelligence and Reporting Tools（BIRT：Eclipse のレポート機能）によってレポートを作成することができる．

TPTP は Eclipse のツールのプラグインで，集中，オープンソース，柔軟，拡張可能などを特徴としたテストのフレームワークである．この特徴を活かすことで，テストの作成，管理，実装，データプール，ログ，レポートとテストに関連する一連の機能を Eclipse を通して一貫した操作感によって使用することができる．

また，すべての API がドキュメント化されており公開されている☆4．

次章では，まとめとして，U2TP および TPTP を拡張した実用方法の現状と将来の展望について筆者の考えを述べる．

☆4 TPTP4.1.0 の API の URL
http://archive.eclipse.org/tptp/4.1.0/javadoc/Platform/public/index.html

4 UML2.0 Testing Profile/Eclipse Test & Performance Tools Platformの解説

U2TP, TPTP の実用化の現状と将来への展望

コンピュータに依存しない概念のモデルが U2TP で，コンピュータでの動作を目的とし，テストツールの共通基盤が TPTP であることはご理解いただけたと思う．今後の方向性として，U2TP については，概念の拡張と表記可能な範囲の拡大を，TPTP については，共通基盤として，さらなるツールとの連携が期待される．先に示した U2TP の適用方法について，それぞれ現状とその将来について述べる．なお情報は原稿執筆時のものに基づいており，すでに実現済みの項目もあるかもしれない．また，筆者の個人的な希望も含まれているので，実現の可能性については保証できないが，ご容赦いただければ幸いである．

［テスト概念・用語のモデル定義による明確化］

［現状］プロセスの用語については，Rational Unified Process（RUP），Basic Unified Process（BUP）を参照，テストの成果物については，U2TP などを参照し，ツールは Eclipse Process Framework（EPF）などを使用することで，プロセスと成果物を独自に作成する．

［将来］組込み／ Web などの技術，金融業／製造業／流通業など業種，地理的分散テストなど形態ごとにテストのプロセスと成果物の体系が標準化され，選択することで利用可能になる．

［テスト・プロセスの記述力向上］

［現状］Eclipse でテスト管理ツールとテストツールの連携ができる．これにより，手動および自動テストを一元的に管理し，テスト作業のフローの標準化が可能である．テスト管理ツールから障害の状況やテストの進捗状況を確認することできる．

［将来］テストのスケジュール，人材のスキル，テスト環境，予算などリソース管理と連携が可能になる．テストの進捗やアプリケーションの品質の状況やリスクを考慮したテスト・プロセスが記述できるようになることで，迅速かつ適切な人材，機材，スケジュールの調整が可能になる．

［自動テストツールの設計仕様］

［現状］単体テスト，GUI 機能試験，負荷試験などのテストツールを利用して，自動テストを作り込む必要がある．また，テストツール単体での使用が多い．

［将来］ユースケースなどの上位のドキュメントから制約や仕様などを与えることで，自動テスト・プログラムを生成．性能やセキュリティ検査などの非機能テストも実施可能になる．

［アスペクト指向プログラミング］

［現状］AspectJ などのプログラミング言語によって，メソッドの開始／終了時ロギングなど基本的なトレース情報のコントロールが可能である．

［将来］(1) JUnit[☆5] がアスペクト指向に拡張．(2) Eclipse Process Framework（EPF）などのプロセス記述ツールがアスペクト指向アプローチ（AOA）に対応．作業開始，終了時のレポートで共通の項目を作業横断的に記述することが可能になる．

常にテストの手法，技術，ツール，方法論について新しい情報が発信されている今日，テストシステム全体のモデル化によって，テストを概観するためにも，U2TP は有効である．

参考文献

1）Model Driven Engineering Technology：http://www.haifa.ibm.com/dept/services/mdet.html

2）日本ソフトウェア科学会第 23 回大会（2006 年度）論文集 1，アスペクト指向プログラミングにおけるテストに基づいたポイントカットの提案．http://www.ipl.t.u-tokyo.ac.jp/jssst2006/papers/Sakurai.pdf

（平成 20 年 1 月 15 日受付）

☆5 JUnit とは Java で開発されたプログラムの単体テストを自動的に行うためのフレームワークである．

小野塚荘一

SONOZUKA@jp.ibm.com

日本 IBM ラショナル・テクニカル・セール & サービス シニア IT スペシャリスト．テスト・プロセス，テスト管理，自動テストツールのコンサルティングおよびツールのサービスを中心に活動．