STEM™ 2.0
――クリーンなソフトウェアを生み出す科学的で規律のある方法

STEM™ 2.0
―― A Scientific and Disciplined Way to Produce Clean Software

T. Ashok
大川　鉄太郎，小玉　哲博　訳

要　約　本稿はクリーンなソフトウェアを生成するテストに対するエンジニアリングアプローチを概説する．STEM バージョン 2.0 は，STAG 社のテストエンジニアリング方法論であり，三つのフェーズにわたる八つのディシプリンを活用して実践される 32 個の科学的コンセプトから構成される．STEM は開発の最終ステージでの製品評価，初期ステージでのサブシステム/コンポーネント評価，要件の妥当性確認に適用される．

Abstract　This article outlines a engineering approach to testing to produce clean software. STEM Version 2.0 is STAG's test engineering method that consists of thirty two scientific concepts implemented using eight disciplines over three phases. STEM has applied in late stage product evaluation, early stage subsystem/component evaluation and has been applied in valiudating requirements.

1. は　じ　め　に

クリーンなソフトウェアをリリースすることは，挑戦的な課題である．クリーンなソフトウェアをリリースするために様々なアプローチがある．規律のあるプロセスに依存するもの，ツール/自動化に依存するもの，個人のドメイン知識と創造性に依存するもの，繰返しの評価に基づくプロセスモデルに依存するものなどがある．

本稿ではクリーンソフトウェアを生み出す科学的アプローチを概説する．STAG 社では，専門的なソフトウェア技術者がソフトウェア品質を保証できるようにする「テストの科学とエンジニアリング」が存在すると考えている．STEM[*1]（STAG Test Engineering Method）は，中核となる科学的コンセプトとこの考え方を実践する一連の個人的ステップに依存する方法中心のアプローチである．STAG 社は欠陥を検出しクリーンなソフトウェアを生み出すための科学的で規律のある方法に焦点を当てている．

IV Square 社と STAG 社は 2006 年 3 月に日本ユニシスと技術協定を締結し，同社品質保証チームと密接に協力して，STEM を適用することで同社における品質エンジニアリングを改善してきた．

現在，IV Square 社と STAG 社は，開発の上流工程からクリーンなソフトウェアを開発する手法である CSD（Clean Software Development）の開発・推進を同社にて試行し始めた．

2. 優れたシステムの特性

我々の目的は，欠陥を検出しクリーンなソフトウェアを生み出すための優れたシステムを構

築することである．優れたシステム（クリーンなソフトウェアを生み出すシステム）は，効果的で，一貫性があり，効率がよく，拡張性があり，可視性が高く，機敏性が高くなければならない（表 1）．

表 1　優れたシステムの特性

有効性	システムはビジネスに大きく影響する欠陥を発見する．
一貫性	システムは個人に完全には依存しない．
効率性	システムは費用効率が高く実用的である．
拡張性	システムは大規模チームにより大規模ソフトウェアに展開できる．
可視性	システムは作業状況を明確に見ることができるので，効果的なマネジメント意思決定/行動が可能になる．
機敏性	システムはソフトウェアの変化するニーズに迅速に対応できる．

3.　クリーンなソフトウェアへのアプローチ

クリーンなソフトウェアを生み出すために採用されている種々のアプローチの概略を表 2 に示す．濃い丸はシステム属性を十分に満たしていることを表し，薄い丸は十分には満たしていないことを表している．

表 2　クリーンなソフトウェアを提供するためのアプローチ

アプローチ	キー特性	有効性	一貫性	効率性	拡張性	可視性	機敏性
プロセス駆動型アプローチ	テストプロセスがキーである 独立したテストチーム	○	●	○	●	○	○
ドメイン中心プロセス	ドメインスペシャリストが欠陥を取り除く	●	●	○	○	○	●
自動化駆動型アプローチ	ツール/自動化がキーである	○	●	●	○	○	○
アドホック/創造的/探査的アプローチ	欠陥を発見するために個人の能力に焦点をおく	○	○	●	○	○	○
頻発かつ連続的な評価アプローチ	欠陥を発見するためにサイクル数に焦点をおく	○	●	●	○	○	●
エンジニアリング手法に基づくアプローチ	欠陥検知/予防のための科学的なアプローチ	●	●	●	●	●	●

優れたテストプロセスに依存する「プロセス駆動型アプローチ」は一貫性があり拡張性があるが，このアプローチの有効性は個人の能力に依存する．欠陥を検出するために個人のドメイン知識に依存する「ドメイン中心型アプローチ」は有効性があるが，個人の知識に依存するため拡張性はない．欠陥を発見するためにツール/自動化に依存する「自動化駆動型アプローチ」は効率的で一貫性があるが，有効性は戦略とテストケースに大きく依存する．「アドホック/創造的」アプローチは確かに効率的であるが，個人のスキルに大きく依存しているので拡張性はない．「頻繁で連続的な」テストに依存するアプローチはツールの活用により機敏性が高く効率的で一貫性があるが，有効性はテストケースの品質に依存する．このように各アプローチは優れたシステムの幾つかの属性を満たしている．欠陥検出のために科学的視点を持つ「エンジニアリング方法論中心型」アプローチはすべての属性を満たすより高い能力を持つ．

STEM 2.0 は方法論中心型アプローチであり，設計，戦略立案，自動化，測定，管理の各テ

ストライフサイクルにおける異なる活動を実施する科学的コンセプトから構成されている．これらが首尾一貫して適用されることを保証するために，STEM 2.0 は科学的コンセプトに加えて，同じように実践するための一連の規律あるステップを備えている．

4. ソフトウェアテスティング―フィッシングアナロジー

湖で漁をする漁師が収入を増やしたいと望んでいる状況を考えてみる（図 1）．どうしたら彼は望みを達成できるだろうか？　1）より多くの魚を捕る，2）より高い値段で売れる魚を捕る．

これを実現するための科学的アプローチは次のようになる．

1. どのような種類の魚を捕りたいのかを明確にする．
2. 頻繁に集まる場所や住処を理解する．
3. 捕まえる魚の大きさに基づき魚網のサイズや網の穴の大きさを明確にする．
4. 魚を捕まえるのに役立つ技術を理解する．
5. 湖をどこまでカバーする必要があるかを決める．
6. 必要となる漁船の種類を決める．

図 1　欠陥発見のためのフィッシングアナロジー

ここで重要なことは捕捉したいもののゴールを設定することであり，捕まえるための手段を決めることである．ここでは，魚をソフトウェアに存在しうる欠陥になぞらえている．このような欠陥を科学的に検出するためには，欠陥のタイプの仮説を立て，実施すべきテストのタイプを考え，検出するためのより早期のステージを特定し，これを実行するために必要な技法/ツールを明確にする必要がある．

5. 効果的なテストへの主要な課題

ソフトウェア開発ライフサイクル（SDLC：Software Development Life Cycle）のどのステージにおけるテストでも，目的は欠陥を検出することである．テストに加えて，レビューやインスペクションも欠陥検出の技法であり，これらの技法は初期ステージの成果物である要件定義書，仕様書，設計書，およびコーディングの初期段階での「スモールコード」の欠陥を検出するのに適している．

有効なテスト（レビュー/インスペクションを含む）に対する主要な課題は次の通りである．

1. 捕まえるべき欠陥のタイプは明確になっているか？　ゴールに強く焦点を合わせているか？　即ち，捕まえるべき魚が分かっているか？
2. テストシナリオ/テストケースあるいはチェックリストのような初期のステージのテスト手順書は完全であるか？　これらが欠陥を検出することに対してどの程度の信頼性があるか？　即ち，魚を捕まえるのに魚網が十分な大きさがあり，十分に網目が細かいか？

上記の二つの課題に答えることにより，必要な品質レベルを明確にし，各レベルで検出すべき欠陥を特定し，その結果としてテストケースを明確に組み立てることができる．テストケースを作成し，様々な基本機能/モジュールにおける欠陥を検出するのに最も適した技法を科学的に特定できる．最後に，ツールニーズを科学的に明確にし，設計したテストケースを実行するために適切なツール/自動化に投資することができる．

6. STEM ―クリーンなソフトウェアへのエンジニアリング方法論中心のアプローチ

STEM すなわち STAG Test Engineering Method は，クリーンなソフトウェアを生み出すための科学的で規律のある方法論である．STEM はゴールに焦点を合わせた方法であり，ゴールを達成するためにフェーズとディシプリンにより構成された明確なステップから成り立つ．中核となる科学的コンセプトは，STEM のそれぞれのディシプリンにおける各ステップを強化し，各ステップが科学的な方法で実施されることを保証する（図 2）．

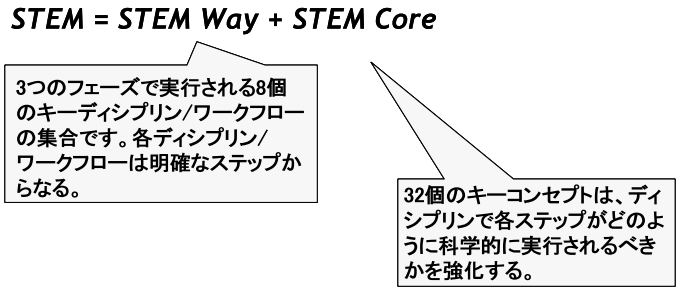

図 2　STAG テストエンジニアリング方法論（STEM）の概要

STEM 2.0 はクリーンなソフトウェアを生み出す活動を 3 フェーズに分けている．3 フェーズとは，1）何をすべきかを明確に理解しているか？　2）どのように実施すべきかを知っているか？　3）どのようにうまく実施されたか？

STEM 2.0 は様々なフェーズに適用されるディシプリンによってグループ化されたステップから構成される．この三つのフェーズとそれぞれのディシプリンを STEM Way と呼び，これは実行のためのプロセスモデルである．STEM Way は三つのフェーズと八つのディシプリンの組み合わせである（図 3）．これらのステップは科学的なコンセプトにより強化される．科学的なコンセプトを STEM Core と呼び，八つのディシプリンに適用する 32 個のコアコンセプトがある．

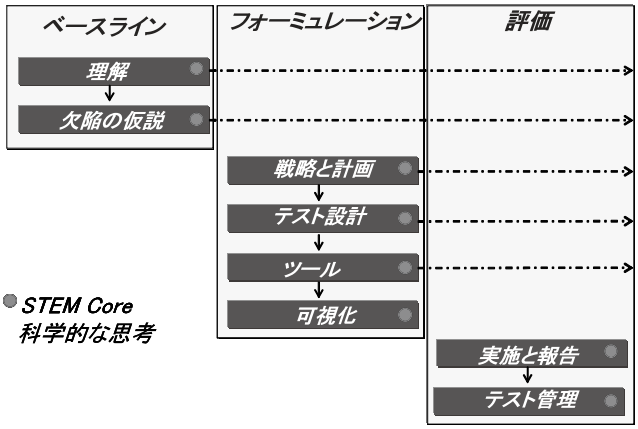

図3　STEM ―フェーズ，ディシプリン，Core

7. STEM Way

STEM Way は STEM 2.0 の活動を実践する規律のある方法である．これらの活動（ステップ）は三つのフェーズで使用されるディシプリンでグループ化される．

最初のフェーズであるベースラインでは，主として，基本機能，属性，技術，「汚れ」を引き起こす欠陥のタイプの観点で製品を理解する．ベースラインのフェーズでこれを達成するための二つの重要なディシプリンが，ビジネス価値の理解と欠陥の仮説である．

第 2 のフェーズであるフォーミュレーションでは，活動計画を明確にするために，戦略，計画，テストケース，テストスクリプト，および測定を立案する．このフェーズは，戦略と計画，テスト設計，テストツール，可視化の四つのディシプリンから構成される．

第 3 のフェーズである評価では，最終的にリリースするソフトウェアが十分にクリーンであり，リリースと展開に耐えるものであることを保証するためにソフトウェアを評価する．このフェーズは，実施と報告，テスト管理の二つのディシプリンから構成される．

STEM Way の詳細と各ディシプリンのゴールを図 4 に示す．

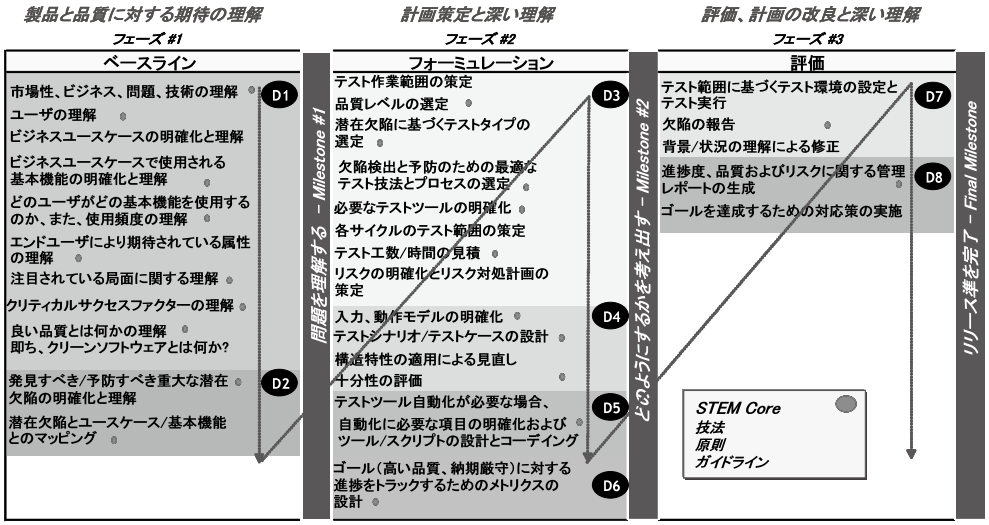

図4　STEM Way の詳細

STEM 2.0 の各フェーズで各ディシプリンがどれだけ適用されるかを図 5 に示す．最初のディシプリンである D1（ビジネス価値の理解）は，第 1 フェーズにより多く適用されるが，後のフェーズでも適用される．各ディシプリンの様々なステップはひとつのフェーズに限定されるのではなく，三つのフェーズに渡って連続的に適用されることを理解することが重要である．

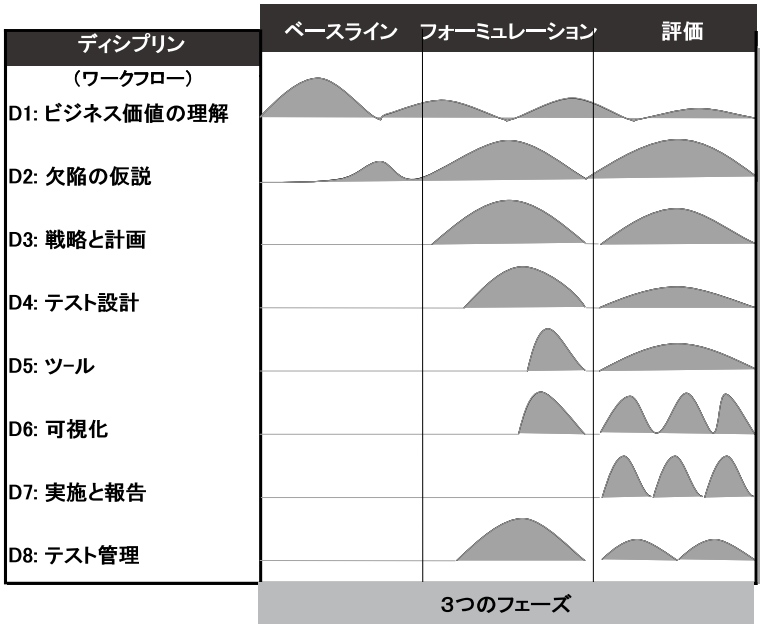

図 5　フェーズにおけるディシプリンの適用

STEM においてディシプリンは図 6 に示すような明確な構造を持つ．ディシプリンはゴール駆動型であり，ゴールを達成するための活動の集まりである．各活動が効果的に実施され，個人に依存しないことを保証するために STEM コアコンセプトによって強化される．

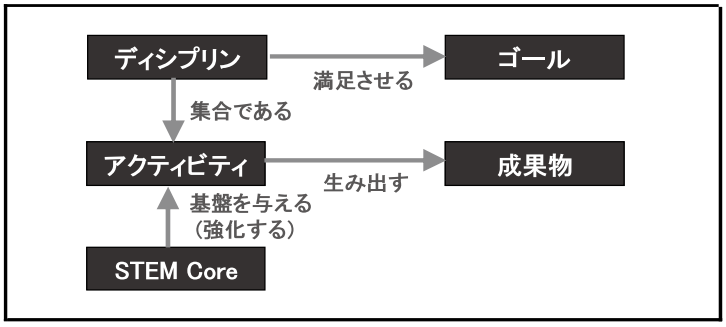

図 6　STEM におけるディシプリンのアーキテクチャ

テストエンジニアリング方法論である STEM 2.0 のゴールは，クリーンなソフトウェアを生み出すことである．このゴールはフェーズ毎のゴールに分割され，さらにディシプリン毎のゴールに分割される．各ディシプリンのゴールを表 3 に示す．

表 3　ディシプリンのゴール

ディシプリン	ゴール
D1: ビジネス価値の理解	エンドユーザの期待およびビジネス価値が明確に理解すること
D2: 欠陥の仮説	潜在欠陥を仮設することによる欠陥中心主義の考えとその結果としてゴール指向の考えを徹底すること
D3: 戦略と計画	効果的でかつコスト効率の高い妥当性確認のアプローチを考え付くこと
D4: テスト設計	ソフトウェア全体をカバーする最小でかつ有効なテストケースを考え出すこと
D5: ツール	ツールの必要性を明確にし、コスト効果の高いテスト自動化を構築すること
D6: 可視化	明確に洞察し、客観性がある決定を速やかに実施できるようにすること
D7: 実施と報告	詳細なアセスメントを実施し、状況を継続的に把握し、欠陥を修正するために十分な情報を提供すること
D8: テスト管理	エンドユーザの期待およびビジネス価値に合っていることを家訓すること

8.　STEM Core

STEM Core は各ディシプリンでのどのように活動すべきかを支援する科学的コンセプトの集まりである．活動成果の有効性が個人の知的能力に依存するリスクを低減するために，ディシプリンの各活動を科学的方法により強化している．

STEM Core は 32 個のコンセプトから構成される．中核となる科学的コンセプトは次の 3 レベルから成る．1）技法―問題を解決するための明確な方法，2）原則―問題を解決するための基礎，3）ガイドライン―問題を解決するための指針，である．

STEM Core により活動が科学的な方法で実施でき，STEM Way により活動が規律のある方法で実施できるようになる．したがって，STEM Core は各活動が効果的に実施されることを保証することに焦点を合わせており，STEM Way は効率性，一貫性，拡張性に焦点を合わせている．

STEM Core の 32 個の科学的コンセプトの概要を表 4 に示す．

表 4　STEM Core の概要とディシプリンでの適用

	コアコンセプト	概要
1	Value prioritization （価値の優先順位付け）	システムに実装されるフィーチャ（機能）のビジネス価値および優先度を理解する手法
2	Attribute analysis （属性分析）	システムに実装されるべき属性（非機能性）を理解する手法
3	Landscapes （ランドスケープ）	マインドマップ手法を適用し、テスト対象の製品をより深く理解する技法
4	Viewpoint （ビューポイント）	エンドユーザの視点でシステムを見ることにより、各ユーザの実際の使用方法を理解する手法
5	Reduction principle （低減主義の原則）	システムを段階的に詳細化することにより、テストすべき機能および属性を明確化するための基本的な考え方
6	Reduction principle （低減主義の原則）	システムを段階的に詳細化することにより、テストすべき機能および属性を明確化するための基本的な考え方
7	EFF Model （EFFモデル）	エラー、フォールト、ファイリャーの視点でシステムをモデルし、発見すべき問題を理解する手法
8	Defect centricity principle （欠陥中心の原則）	システムに潜在する可能性がある欠陥のタイプ（パターン）を明確化するための基本的な考え方
9	Negative thinking （ネガティブシンキング）	システムに潜在する問題点を種々の視点で見つけ出そうとする思考方法
10	Orthogonality principle （直交性の原則）	発見すべき欠陥、テストタイプ、テストレベル、テスト技法の関係を理解するための基本的な考え方

表 4　STEM Core の概要とディシプリンでの適用（続き）

コアコンセプト		概要
11	Approximation principle （近似の原則）	科学的な近似を実施するための基本的な考え方
12	Tooling needs assessment （ツールニーズ アセスメント）	テストツールの必要性を評価する技法
13	Defect centered activity breakdown （デフェクト指向のテスト作業細分化）	デフェクトを発見するためのテスト手順に基づき、テスト工数等を見積るため技法
14	Quality growth principle （品質成長の原則）	ソフトウェアの品質レベルを特定するための基本的な考え方
15	Techniques landscape （テスト技法ランドスケープ）	適用すべきテスト技法を特定するためのガイドライン
16	Process landscape （プロセスランドスケープ）	適用すべきテストプロセスモデルを特定するためのガイドライン
17	Behavior stimuli approach （BESTアプローチ）	機能性テスト設計を実施するために適用する技法
18	Operational profiling （操作プロファイリング）	実環境でのエンドユーザベースでの使用パターン（操作プロファイル）をモデル化する技法
19	Box model [y=f(x)] （ボックスモデル）	入力と出力の関係からシステムの機能（動作）を理解する技法
20	Input granularity principle （入力粒度の原則）	テストレベルに対応した入力の粒度（細かさ）を理解するための基本的な考え方
21	Test coverage evaluation （テストカバレージ評価）	テストケースの十分性/カバレッジを評価する技法
22	Complexity assessment （複雑度評価）	機能/要求の複雑度を評価する技法
23	GQM (Goal Question Metric)*2 （ゴール指向評価指標）	品質目標に基づきソフトウェア品質の評価指標を設定する技法
24	Quality quantification model （品質定量化モデル）	ソフトウェア品質を定量的評価することにより、品質に対する客観的評価を与える技法
25	Automation complexity （自動化複雑度評価）	フィーチャ(機能)/要求に対するテスト自動化の複雑度を評価する技法
26	Minimal babysitting principle （ミニマル・ベビーシッティングの原則）	テストの自動化において、無人でテスト実行を可能にし、テストの実行効率の最大化を実現するテストスクリプトを設計するための基本的な考え方
27	Separation of concerns principle （問題分離の原則）	高いメンテナス性を保証するテストスクリプトを設計するための基本的な考え方
28	Tooling need analysis （ツールニーズ分析）	テストツールに対するROIを最適化するために、テストツールの必要性を分析する技法
29	Contextual awareness （状況の認識）	テスト状況から得られた情報に基づき、より効果的なテストを実施するための基本的な考え方
30	Defect rating principle （欠陥レーティングの原則）	重大度および優先度に基づき発見された欠陥をレーティングすることにより、欠陥を分類するための基本的な考え方
31	Gating principle （ゲートの原則）	各テスト工程に効果的な終了基準（ゲート）を設定し、テストを最適化するための基本的な考え方
32	Cycle scoping （サイクル・スコーピング）	複数のテスト・サイクルを実施する場合に、各テスト・サイクルでのテストスコープを策定するための技法

9.　STEM Way と組織プロセスへの関係

STEM Way は STEM Core に基づく科学的思考技法に支えられたテストのパーソナルプロセスである．STEM Way はソフトウェアテストに従事する個人（開発者/テスト担当者）に優れた思考プロセスを提供することで，組織プロセスモデルを補完する（図 7）．

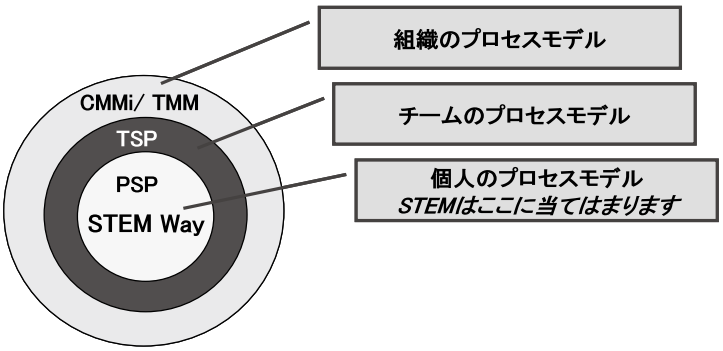

図 7　プロセスリング：STEM Way とプロセス階層構造における位置付け

10.　STEM Way と STEM Core の図解

ディシプリン D4，テスト設計の詳細を図 8 に示す．このディシプリンは七つの主要なステップから構成される．このディシプリンは八つの STEM Core コンセプトを使用する．C1 ～ C8 は STEM Core コンセプトで，各ステップにおける適用方法を表す．

このディシプリンは問題を機能要件および非機能要件に分解し，動作モデルを作成し，テストシナリオとテストケースを設計するステップである．その後，構造特性を使用し，テストケースを改善する．最終的に，テストケースは要件にトレースされるだけでなく，潜在欠陥にもトレースされる．テストケースの潜在欠陥へのトレースをフォルトトレーサビリティと呼ぶ．

ディシプリン			D4: テスト設計
目的			すべてのソフトウェアをカバーする最小限で効果的なテストケースを作成する。
アクティビティ			
	1		分割 – 機能性および非機能性の要件を明確にする。
C1		1.1	各要件に対する非機能性を明確にする。
	2		動作モデリング – 略式の表記法を使用し機能性の動作をモデル化する。
		2.1	意図された動作を理解するために仕様書を読んだり、ソフトウェアを探査する。
C2		2.2	要件に関係する様々な入力を特定する。 入力はユーザインタフェースあるいはシステムからの入力になることに注意する。
C3		2.3	入力の仕様を理解する。「入力粒度の原則」を適用することが重要になる。
C4 C5 C7		2.4	「ボックスモデル」を適用することにより、機能性仕様に対する動作モデルを作成する。
		2.5	意図されたビジネスロジックを記述する。
		2.6	「技術ランドスケープ」で概説された技法を使用し、ボックスモデルで記述されたものを適切な表記法に変換する。
C5	3		作成 – テストシナリオを設計し、対応するテストケースを作成する。
		3.1	表記法で記載された動作に対する様々なテストシナリオを作成する。
		3.2	与えられた仕様からシナリオが明確にならない場合には、明確にし、作業を継続するか、設計上の欠陥として記録し、解決する。
		3.3	各シナリオに対する入力を明確にする。
C6	4		コード構造を考慮した改善 – コード構造/設計法を考慮し、TS/TCを追加・削除する。
		4.1	コード複雑度を使用し改善する。
		4.2	リソース使用パターンを使用し改善する。
		4.3	ランタイム環境に対するコード依存性を使用し改善する。
		4.4	潜在的な例外処理/エラー処理を使用し改善する。
C8	5		要件トレース – テストシナリオ/テストケースを要件/機能/コードにトレースする。
C8	6		フォールトレース –テストシナリオ/テストケースを潜在欠陥にトレースする。
	7		テストシナリオ/テストケースのカバレージを評価する。
		7.1	モデル化された動作が考慮されたかを確認する。
		7.2	正常系テストケースと異常系テストケースの比率が適切であるかを確認する。
		7.3	すべての潜在欠陥がカバーされているかを確認する。

STEM Core
1. 低減主義の原則
2. 入力粒度の原則
3. ボックスモデル
4. BESTアプローチ
5. 技術ランドスケープ
6. 複雑度評価
7. 操作プロファイリング
8. テストカバレージ評価

図 8　テスト設計ディシプリン詳細

11. 成果

STEM は顧客に対して大きな成果を提供した．主要な成果の例を図 9 に示す．開発の初期のステージに適用した場合，開発からリリースまで欠陥見逃しが 30 ～ 50%減少した．システムテストに適用した場合は，リリース後に報告される欠陥が 1/3 ～ 1/10 に低減した．STEM に基づくコンピテンシーモデルを適用することにより，優れたテスト成功要因が個人のスキル/ドメイン知識に依存しなくなり，テストチームを迅速に立ち上げ，拡大することができた．STEM を適用することにより，テストケースが大幅に増加し（2 ～ 5 倍），その結果として欠陥の捕捉率が改善された．また，ツール活用と自動化に対する科学的アプローチは，自動化投資効率を大きく改善した（最大 5 倍）．

顧客との機密保持契約の関係により，顧客名およびプロジェクトの詳細は公開できないが，STEM を適用した事例を次に解説する．

速い段階での検出	30 – 50% 改善
より高い品質のソフトウェア	300 – 1000% 改善
スキル依存性の低減とテストチームの拡張	100 – 200% 改善
欠陥を捕える良いネット	200 – 500% 改善
ツール活用の高いROI	100% まで

図 9　STEM 適用の結果

11. 1　事例 1

.NET 技術を採用した製品を開発する企業の事例である．この企業は品質管理プロセスを持っていたが，結果は満足できるレベルではなかった．主要な問題のひとつは，品質の問題により開発が遅れることであった．品質チームの指摘による問題に対する訂正作業および製品リリース日のようなマイルストーンの変更がたびたび発生した．主要なビジネスゴールは次の通りであった．1）欠陥見逃しを低減すること（欠陥見逃しは 60%であった）．2）テストサイクルタイム予測を改善すること．3）アドホックテストより計画性のあるテストの信頼性を向上させること（欠陥の 80%はアドホックテストにより発見されていた）．

詳細な解析の結果，明確になった問題は次の通りであった．1）極端に厳しいスケジュールのため，効果的な単体テストが実施されていない．2）コードの多くは受け継がれており，実施されたテストケースはほとんど正常系である．3）問題の検出が遅いため，QA 完成日が遅れる．4）設計されたテストケースは有効ではない．即ち，アドホックテストの方が有効である．

これらの問題に対して STEM を適用することにより，次のような結果が得られた．

- 単体テストケースが 70%改善された．
- 単体テストでの欠陥見逃しが 24%から 3%に低減された．
- リリース後の欠陥見逃しが 60%から 5%に低減された．
- アドホックテストにより発見される欠陥数は大幅に減少した．欠陥の 90%は設計されたテストケースで発見された．
- ブランチカバレージが 65%から 85%に改善された．

11. 2　事例 2

デジタルビデオの分野で組み込み製品を開発するハイテク企業の事例である．リリース後に多くのサポート費用がかかることが問題であった．明確にされた問題は次の通りであった．1）テストケースが十分ではない．2）正常系テストケースに対する異常系テストケースの割合が低い．3）欠陥見逃しが高い．4）初期に発見されるべき欠陥が最終ステージで検出される．

テストチームがSTEMを採用し，テストケースを再設計し，テストケース数を5倍に増加させた．テストケースの欠陥指向性をより明確にするテストタイプにより，テストケースをグループ化した．この結果，欠陥見逃し率は70%から10%に低下した．10%の欠陥見逃しはエンドユーザの使用環境に依存するものおよび装置がないためテストできなかったものであった．STEMを適用した自動化によりテスト時間が1/2に削減された．

11. 3　事例 3

テレコムアプリケーション製品を開発する企業の事例である．製品は市場投入されて数年経つが，新バージョンのリリースが計画され，STAG社はリリース検証テストを依頼された．STEMの適用により約860件のテストシナリオが特定され，結果として，186件の欠陥を発見し，内39件が重大欠陥であった．このテストは5週間で完了し，リリース後の製品安定化の問題を解決した．

11. 4　事例 4

銀行向け製品を開発する企業の例である．J2EE技術を採用した製品で，市場では6年以上の実績があった．この製品は顧客の要求に応じてカスタマイズを実施しており，顧客対応のために多くのコードベースが開発された．これらの異なるコードベースをマージするための新しいアーキテクチャの開発に着手した．また，テストケースも顧客要求に応じて修正されてきた．問題はテストケースの十分性と効果性に疑念があることであった．

STEMを適用し，テスト十分性を保証するためのテストケースの再設計を実施した．テストシナリオとテストケース作成にSTEMアプローチを適用し，結果として，テストケース数が80%増加し，欠陥タイプによるテストケースの明確な分類によりテストケースの信頼性を向上させることができた．

12.　SDLC における STEM のアプリケーション

クリーンなソフトウェアを生み出すエンジニアリング方法論としてのSTEMは，ソフト開発ライフサイクルのテストフェーズだけに限定されるものではない．STEMはシステムのレベルと統合レベルでの製品のテストに適用される．さらに，STEMは開発の初期段階においてクリーンな開発プラクティスを実践するためにも適用される．コードの妥当性確認に加えて，STEMは「要件テスト」として要件定義書のクリーン度を検証するためにも適用される（図10）．

STEMはSDLCの各フェーズにおいてクリーンであることを保証するためのエンジニアリング方法論である．初期の要件定義フェーズ，設計フェーズ，開発フェーズ，統合およびシステムテストフェーズに適用できる．リリース後はソフトウェアの保守フェーズにも適用できる．

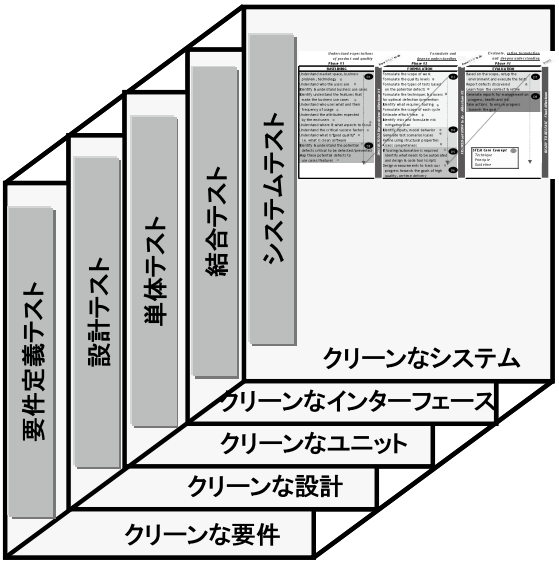

図 10　STEM と SDLC の異なるステージへの適用

13.　クリーンなソフトウェアを生み出すためのゴール指向型アプローチ

STEM において重要なことは，クリーンなソフトウェアというゴールに対する強い目的意識である．STEM の主要なポイントは，欠陥を仮説し，効果的でかつ効率的な方法で，いつ，どこで，どのように欠陥を検出するかを科学的に明確にすることである．STEM がどのようにこの目的意識を実現するかを図 11 に示す．

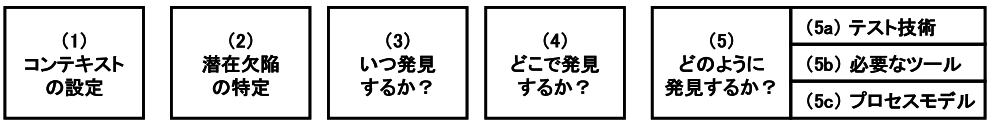

図 11　クリーン度に対するゴール指向型アプローチ

⑴　コンテキスト設定―最初にコンテキストの設定に焦点を合わる．要件/仕様を明確に一覧化し，ビジネス価値を理解し，ビジネスリスクをランク付けし，成功要因を明確にし，そして，クリーンであることの基準を特定する．

⑵　潜在欠陥の特定―技術，展開環境，使用方法，実装，使用ドメインに基づいてソフトウェアにある潜在欠陥を仮説する．この作業の結果は，潜在欠陥タイプを特定するために集められた潜在欠陥の一覧表である．これによりソフトウェアを汚染する原因を明確に理解できる．

⑶　品質レベルとステージング―潜在欠陥タイプを明確にし，ソフトウェア開発ライフサイクルにおける欠陥検出の最も早いポイントを特定する．このための古典的方法はテストレベルを三つに分けることである．即ち，開発環境での単体テスト，統合テスト，システムテスト，および顧客環境における受入テストとアルファ/ベータテストである．STEM では古典的なレベルに限定せずに，複数の品質レベルに分割する．その結果として，複数のステージからなる品質成長システムを構築する．あるタイプの欠陥は検出

されるか，あるいは，存在しないことが証明されるため，ソフトウェア品質は向上するとの考えに基づいている．従って，様々なタイプの欠陥を分離し，ランク付けできれば，複数の品質レベルを設定することができ，開発中のソフトウェアの品質成長モデルを構築することができる．

⑷ トレーサビリティ―うまく定義された品質成長モデルを構築するために，欠陥タイプとランク付けを確定し，潜在欠陥タイプが存在する要件/基本機能/サブシステムを特定する．これにより潜在欠陥を何処で探せばよいか明確に理解できるようになる．このプロセスにおいて，潜在欠陥と要件/仕様/サブシステムの間のトレーサビリティを作成する．

⑸ 技法，ツール，プロセス―コンテキストと範囲，どの潜在欠陥をいつどこで探せばよいかを明確に理解できると，次に潜在欠陥を発見するのに最も効果的なテスト技法，評価に最も適したプロセスモデル，プロセスステップを実行するためのツール要件等を決める必要がある．

14. お わ り に

STEM は 2000 年に考案され，過去 8 年以上にわたり着実に進化してきた．これまで，STEM は様々な技術を採用した大規模なビジネスアプリケーション，組み込みソフトウェア，通信ソフトウェアのような様々なドメインにおいて，多くのプロジェクトで採用され成功を収めてきた．STEM はドメインと技術に依存しない方法論であり，特定の自動化ツールの使用時においてのみ技術に依存する．

STEM は効果的なテストのための強力な基盤を提供することに加えて，STAG 社ではドメインにおける個人のスキル/経験に依存しない優れたテストチームを作り上げることに成功してきた．

STEM の適用によりクリーンなソフトウェアを生み出すことに加えて，STEM に基づいたテストチームの能力アセスメントサービスを提供している．現在，STAG 社では STEM の実装/展開を支援するツールスイートである STEM ツールキットを開発している．

＊ 1　STEM™ は STAG Software Private Limited. の商標である．
＊ 2　Victor R. Basili が提唱する技法をテストエンジニアリング分野に適用したものである．

参考文献　［1］ Scott W. Ambler, “A Manager’s Introduction to The Rational Unified Process (RUP)”, Dec 4, 2005 (Paper on Web)
［2］ “The Science and Engineering of Testing Course Material”, STAG Software Private Limited.
［3］ “Rational Unified Process- Best Practices for Software Development Teams”, Rational Software White Paper, TP026B, Rev 11/01

執筆者紹介　T. Ashok

クリーンなソフトウェアの開発方法論に特化したSTAG Software Private Limited.（www.stagsoftware.com）の創立者であり，最高経営責任者（CEO）である．熱烈な向上心を持ち，クリーンなソフトウェアを作り出す方法論の探求を使命としている．STAG社のテストエンジニアリング方法論であるSTEM^{TM}のチーフアーキテクトを務める．知識を共有することに情熱を持ち，ワークショップを開催し，いくつかの重要なフォーラムで講演している．イリノイ工科大学（シカゴ）とアンナ大学（インド）を卒業し，業務経験は23年を超えている．

STAG社は，東京に本社を持ちテストエンジニアリングに特化した日本の企業であるIV Square社（www.ivsquare.co.jp）と独占的業務提携を結んでいる．AshokはIV Square社の役員も務めている．

訳者紹介　大川　鉄太郎（Tetsutaro Okawa）

1974年日本ユニシス(株)入社．汎用機オペレーティングシステムと通信ソフトウェアの保守・サポートに従事．1996年より現在まで品質管理，プロセス改善に従事．

共著に『ソフトウェア品質知識体系ガイド― SQuBOK Guide』（オーム社2007年刊，2008年度日経品質管理文献賞），共訳書にリック・スターム他著『標準サービスレベルマネジメント』（オーム社2003年刊），論文にCONQUEST2004，3WCSQ（2005），CONQUEST2007，4WCSQ（2008）他がある．

小　玉　哲　博（Tetsuhiro Kodama）

1976年(株)IHI入社，航空機エンジンの自動計測システム開発に従事．1980年日本ヒューレット・パッカード(株)入社，製品開発，システム構築，ITコンサルティング等のプロジェクト業務に従事．2007年にSTAG社Ashok社長の提唱するテスト方法論STEMに共感し，(株)アイ・ブイ・スクエア入社，テスト技術教育，テスト技術コンサルティング等を担当．同社IV&V事業部長．大阪大学大学院修士課程修了．