ウォーターフォールモデルに基づく情報システム開発における成果物の品質管理手法について

宇田川 佳久[†1]

今日の高度情報化社会では，情報システムの障害による業務・サービスの停止や機能低下は，社会活動に深刻な影響を与えている．その一方で，情報システムの高機能化・大規模化・短納期化など，品質確保を難しくする要因が増大しており，体系的な手法に基づく品質管理の必要性が高まっている．本文は，ウォーターフォールモデルに基づくシステム開発における設計およびテスト工程での成果物の品質管理手法について論じている．この手法の特徴は，工程をまたがる成果物の網羅性を確認する手段を提供していることである．上流工程での成果物が下流工程に引き継がれていることを確認するために“トレース関連”という概念を導入する．要件定義および設計工程での成果物は，木構造で表現できる．そのため，“設計工程のトレース関連”を上流工程の成果物の木構造と下流工程の木構造との写像として定義する．一方，テストは，設計工程での成果物とテストケースの組合せで実施される．この組合せを，設計工程の成果物の木構造とテストケースの集合との写像として定義し，これを“テスト工程のトレース関連”と呼ぶ．トレース関連をグラフ理論に基づいて定義し，各工程での品質管理基準について論じる．また，本文で定義したトレース関連をシステム開発における事例に適用し，成果物の網羅性を確認する手段としての有効性を検証している．

A Work Product Quality Management Technique in Information System Development Based on the Waterfall Model

Yoshihisa Udagawa[†1]

In today's high information-oriented society, a trouble of an information system causes stop of business resulting in a serious influence on the social activity. On the other hand, system developers often face adverse factors that make difficult to keep the quality of information systems, such as short-term development and high complexity in a system architecture, etc. This paper discusses a quality management technique for a set of work results in system development based on a waterfall model. The technique features to offer means to identify exhaustiveness of work products among processes in a system development. This paper introduces a "trace-relation" to make sure that items in work products are succeeded by the subsequent process. Items of a work product in requirement/design processes construct a tree structure. Thus, "trace-relation in design" be defined by a mapping between two tree structures of a subsequent requirement/design processes. In the same manner, "trace-relation in test" be defined by a mapping between a tree structure of items in design and a set of test-cases. This paper gives a strict definition of "trace-relation" based on graph theory and discusses management criterion. Effectiveness of the management technique is verified by means of an example in a system development field.

1. はじめに

コンピュータの処理性能の飛躍的な向上やインターネットをはじめとする通信技術の普及により，情報システムは現在の高度情報化社会を支える基盤として社会活動に広く適用されている．同時に，情報システムの障害による業務・サービスの停止や機能低下は，社会活動に深刻な影響を与えており，システムの品質向上は，情報システム構築ベンダの社会的な責任をともなう課題となっている[3),4)]．

システムトラブルの要因は，システムの外的要因と内的要因に分類できる．外的要因とは，システムを取り巻く環境が，想定していた環境を逸脱したときに発生するもので，想定以上の処理負荷によるシステムトラブルが典型的な例である．内的要因とは，システムに在留しているバグを指す．十分な開発期間とリソースがあればバグは排除できるが，競争激化による短期

†1 三菱電機インフォメーションシステムズ株式会社
Mitsubishi Electric Information Systems Corporation

開発や新技術の導入などにより，開発プロセス管理や実装技術が充足されず，バグを内在したシステムが稼動しているのが現実である．

本文は，情報システム開発における設計およびテスト工程での成果物の管理手法について論じている．本手法は，システム開発で広く用いられているウォーターフォールモデル[1]に基づいており，システム開発における要件定義項目と設計項目の網羅性，設計項目とテストの網羅性を管理することを特徴としている．情報システム開発の成果物を体系的に管理するためにはツールによるトレーサビリティの管理が必須であるが，既存のツールでは，トレーサビリティを“項目間のリンク”として扱っている[2),8)]．一方，大規模システム開発における成果物は，成果物の構成を決める計画フェーズと内容を詳細に定める実施フェーズで別々の担当者によって作成することが一般的であるが，従来の“項目間のリンク”としてトレーサビリティ管理機能では計画と実施フェーズでのトレーサビリティ区別が難しかった．本文では，計画フェーズで定義するトレーサビリティ（以降，包括トレース関連と呼ぶ）と実施フェーズで定義するトレーサビリティ（以降，個別トレース関連と呼ぶ）の概念を導入する．包括トレース関連と個別トレース関連が定める関連性を分析することにより，成果物の項目間の整合を効率的かつ正確に管理できるのが特徴である．

2章では，ウォーターフォールモデルの各工程における成果物について概論する．要件定義からプログラム作成までの設計工程におけるトレーサビリティと，単体テストからシステムテストまでのテスト工程におけるトレーサビリティの特徴を述べるとともに，トレーサビリティ管理の対象を明確にする．

既存の手法では，トレーサビリティをツールの実装機能としての“項目間のリンク”で扱っていた．3章では，トレーサビリティを成果物の概念に基づいて定義する．これにより，トレーサビリティを成果物に付随する普遍的な属性として扱うことができることを論証している．また，トレーサビリティを使った成果物の品質管理のポイントについて論じる．

4章ではログイン処理を事例とし，3章で述べた管理手法の有効性を検証する．5章では，3章の定義に基づきトレース関連に関する主要な概念の実装方針を示す．

2. 情報システム開発のプロセスと成果物

2.1 開発プロセスモデル

システム開発プロセスは，ウォーターフォールモデル，プロトタイプモデル，スパイラルモデルに大別できる[1)]．ウォーターフォールモデルは，開発工程を要件定義，外部設計，内部設計，プログラム作成，結合テスト，システムテストなどに分け，各々の工程を完結した状態で次の工程に進む．ウォーターフォールという名前が示すとおり，上流工程から下流工程へと後戻りせずに開発を進めていくという特徴がある．このモデルは多くの情報システム開発に適用されている．

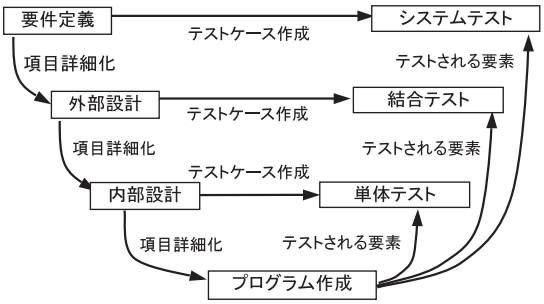

図 1 ウォーターフォールモデルの各工程と成果物
Fig. 1 Stages of waterfall model and documents.

プロトタイプモデルは，要件定義を作成する段階で，重要な仕様を確認するための試作ソフトウェア（プロトタイプ）を作り，利用者に試用・評価してもらい，修正を繰り返しながら詳細な仕様を確定するという技法である．外部設計以降は基本的にはウォーターフォールモデルと同じ手順で開発を進めるモデルである．

スパイラルモデルは，大規模なシステムを独立した部分単位に分割し，部分ごとに設計/開発/テストの工程を反復しながら，完成度を高めていく技法であり，ウォーターフォールモデルとプロトタイプモデルの両方の技法を合わせたモデルである．

本文では，情報システムの開発で広く採用されていること，また，プロトタイプモデル，スパイラルモデルにも共通する技法を提供していることから，ウォーターフォールモデルに基づくシステム開発を前提とする．

2.2 ウォーターフォールモデルと成果物

図 1 は，ウォーターフォールのV字モデルと各工程の関連について示したものである[1)]．要件定義からプログラム作成までの各工程では，仕様を大項目から詳細な項目までブレークダウンし，木構造によって階層的に管理している．一方，品質管理の観点では，要件定義から始まる仕様が後続の工程に網羅的に引き継がれていることを保証する必要がある．ISO9000[6)]やCMMI[7)]では仕様が工程間で引き継がれることをトレーサビリティと呼んでおり，品質管理の重要指標として規定しているが，具体的な管理方法については言及していないため，プロジェクトごとに固有の管理を

実施しているのが現状である．

要件定義からプログラム作成までの各工程では，仕様を木構造によって管理していることから，トレーサビリティを木構造から木構造への写像として定義することができる．

工程をまたがる仕様の引き継ぎは，下記のケースがあり，本文では，これらを総称して**設計工程のトレース関連**と呼ぶ．

- 要件定義項目から外部設計項目への関連
- 外部設計項目から内部設計項目への関連
- 内部設計項目からプログラム要素への関連

一方，テストは，設計項目に関連付けられたテスト項目に基づいて実施される．テストケースは，テストを完遂するために必要な環境（条件）を備えており，個別に扱うことができる．すなわち，テストは，テスト対象とするプログラムとテストケースの集合から構成されるものと定義することができる．設計項目とテストケースの関連は，下記があり，本文では，これらを**テスト工程のトレース関連**と呼ぶ．

- 内部設計項目から単体テストケースへの関連
- 外部設計項目から結合テストケースへの関連
- 要件定義項目からシステムテストケースへの関連

設計工程のトレース関連と**テスト工程のトレース関連**とをあわせて**トレース関連**と呼ぶ．

3. グラフ理論に基づくトレース関連の定義

この章では，トレース関連に関する概念をグラフ理論[5]に基づいて定義する．

3.1 設計工程のトレース関連の定義

Ts でトレース関連のソース側（上流工程）の設計項目の木構造を表すものとする．すなわち，Ts を構成する頂点集合を Vs，辺集合を Es，写像 $\mu_s: Es \to Vs \times Vs$ とするとき，$Ts = (Vs, Es, \mu_s)$ と定義する．ただし，任意の点 $x \in Vs$，$y \in Vs$ を結ぶ辺の集合はただ1つであり，かつ Ts はループを含まないものとする．

同様に，Tt でトレース関連のターゲット側（下流工程）の設計項目の木構造を表すものとする．Tt を構成する頂点集合を Vt，辺集合を Et，写像 $\mu_t: Et \to Vt \times Vt$ とするとき，$Tt = (Vt, Et, \mu_t)$ と定義する．ただし，任意の点 $x \in Vt$，$y \in Vt$ を結ぶ辺の集合はただ1つであり，かつ Tt はループを含まないものとする．

写像 $\tau: E \to Vs \times Vt$ とするとき，グラフ $\rho(Vs, Vt, E, \tau)$ を Ts から Tt への**設計工程のトレース関連**と定義する．グラフ ρ は連結グラフである必要はない．ただし，写像 τ の集合が空（ϕ）である場合は上流と下流工程に関連性がないことを意味するため，

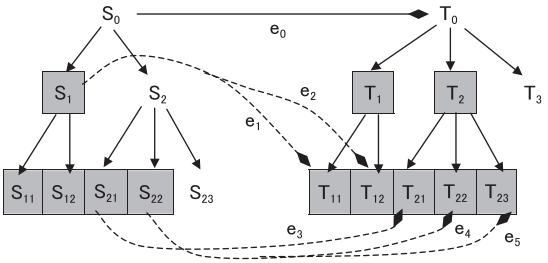

図 2 設計工程のトレース関連の例
Fig. 2 An example of trace-relation in design.

写像 τ の集合は空ではないものとする．以降，Ts をトレース関連のソース木，Tt をターゲット木と呼ぶ．

Tt の終端ノード（葉）の集合を $L(Tt)$ で表す．任意の Vs の点から Vt の終端ノードへの写像 $\tau': E' \to Vs \times L(Tt)$ とするとき，グラフ $\rho' = (Vs, Vt, E', \tau')$ を Vs から Vt への**個別トレース関連**と定義する．個別トレース関連ではないトレース関連を**包括トレース関連**と定義する．すなわち，写像 $\tau'': E'' \to Vs \times \{Vt - L(Tt)\}$ とするとき，グラフ $\rho'' = (Vs, Vt, E'', \tau'')$ と表す．次の節で述べるが，個別トレース関連と包括トレース関連により，ソース木およびターゲット木で対応関係のないノードを検出することができ，これを設計工程における品質管理の一基準とする．

図 2 は設計工程のトレース関連の例である．図 2 の左側の木構造がソース木 Ts で，Vs は $\{S_0, S_1, S_2, S_{11}, S_{12}, S_{21}, S_{22}, S_{23}\}$ である．同様に，図 2 の左側の木構造がターゲット木 Tt で，Vt は $\{T_0, T_1, T_2, T_3, T_{11}, T_{12}, T_{21}, T_{22}, T_{23}\}$ である．写像 τ は $\tau(e_0) = (S_0, T_0)$，$\tau(e_1) = (S_1, T_{11})$，$\tau(e_2) = (S_1, T_{12})$，$\tau(e_3) = (S_{21}, T_{21})$，$\tau(e_4) = (S_{22}, T_{22})$，$\tau(e_5) = (S_{22}, T_{23})$ であるから，E は $\{e_0, e_1, e_2, e_3, e_4, e_5\}$ であり，トレース関連はグラフ $\rho(Vs, Vt, E, \tau)$ である．

一方，Vt の終端ノードへの写像 τ' は $\tau'(e_1) = (S_1, T_{11})$，$\tau'(e_2) = (S_1, T_{12})$，$\tau'(e_3) = (S_{21}, T_{21})$，$\tau'(e_4) = (S_{22}, T_{22})$，$\tau'(e_5) = (S_{22}, T_{23})$ であるから，E' は $\{e_1, e_2, e_3, e_4, e_5\}$ であり，個別トレース関連はグラフ $\rho' = (Vs, Vt, E', \tau')$ である．

Vt の非終端ノードへの写像は $\tau''(e_0) = (S_0, T_0)$ であるから，E'' は $\{e_0\}$ であり，包括トレース関連はグラフ $\rho'' = (Vs, Vt, E'', \tau'')$ である．

3.2 ソース木，ターゲット木の刈込み

個別トレース関連と包括トレース関連を使って，ソース木およびターゲット木の刈込みを定義する．刈込みされたソース木 Ts とは，Ts に対し下記の操作によって得られる木であり $[Ts]$ と表記する．

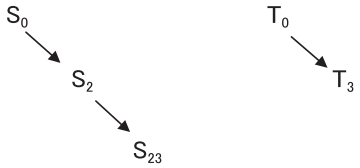

図 3 トレース関連の刈込みの例
Fig. 3 An example of truncated tree.

(1) トレース関連の定義に $x \in Vs$ が含まれるとき，Ts からノード x を削除する（x が非終端ノードであるとき x を頂点とする Ts の部分木を削除する）．

(2) x の上位ノードを $P(x)$ とする．(1) の結果，$P(x)$ に下位のノードがなくなったとき，$P(x)$ を Ts から削除する．

同様に，刈込みされたターゲット木 Tt とは，Tt に対し下記の操作によって得られる木であり $[Tt]$ と表記する．

(1) トレース関連の定義に $y \in Vt$ が含まれるとき，Tt からノード y を削除する（y が非終端ノードであるとき y を頂点とする Tt の部分木を削除する）．

(2) y の上位ノードを $P(y)$ とする．(1) の結果，$P(y)$ に下位のノードがなくなったとき，$P(y)$ を Tt から削除する．

図 3 は図 2 のトレース関連の刈込み結果である．$[Ts]$ は，ターゲット木と対応がないソース木であり，下流工程との対応がない要素から構成されている．同様に，$[Tt]$ は，ソース木との対応がないターゲット木の要素であり，上流工程との対応がない要素，または，下流工程で新たに出現した要素を示す．$[Ts]$ と $[Tt]$ のシステム開発における意味については，次章で述べる．

3.3 テスト工程におけるトレース関連の定義

ウォーターフォールモデルにおけるテストはテストケースに基づいて実施される．テストケースとは，入力データ，実施する操作，期待する結果をセットにしたものであり，より多くバグが見つかるようにテスト対象を漏れなく網羅することが肝要である[1),6),7)]．

図 1 のウォーターフォール V 字モデルで示したように，テストケースは要件定義または外部・内部設計と関連付けて作成される．テストケースでは，テスト対象を特定するための入力データおよび操作を指定するため，テストケースとテスト対象の関連は存在する．一方，入力データおよび操作は，テストケースごとに設定されるため，テストケース間での関連は，一般的には存在しない．そのため，テストケースを単なる集合として扱う．

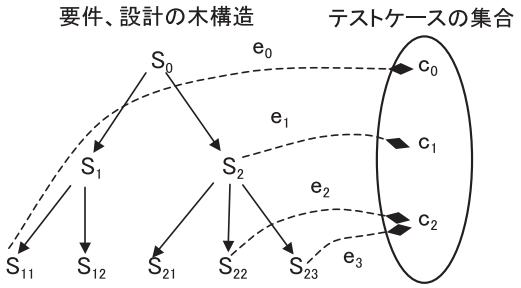

図 4 テスト工程のトレース関連
Fig. 4 An example of trace-relation in test.

テスト対象＼テストケース	c_0	c_1	c_2	テスト済
S_0	○			○
S_1				
S_{11}				
S_{12}				
S_2		○		○
S_{21}		○	○	○
S_{22}		○	○	○
S_{23}		○	○	○

図 5 トレース関連のマトリックス表現
Fig. 5 A matrix representation of trace-relation in test.

要件定義または外部・内部設計とテストケースとのトレース関連は，要件定義または外部・内部設計を表すソース木 Ts と，テストケースの集合 C との関連として定義する．形式的には，写像 $\psi: E \to Vs \times C$ とするとき，グラフ $\sigma(Vs, C, E, \psi)$ を Ts からテストケース C への**テスト工程のトレース関連**と定義する．

テストケースのトレース関連の例を **図 4** に示す．Vs は $\{S_0, S_1, S_2, S_{11}, S_{12}, S_{21}, S_{22}, S_{23}\}$，$C$ は $\{c_0, c_1, c_2\}$，E は $\{e_0, e_1, e_2, e_3\}$ である．写像 ψ は，$\psi(e_0) = (S_{11}, c_0)$，$\psi(e_1) = (S_2, c_1)$，$\psi(e_2) = (S_{22}, c_2)$，$\psi(e_3) = (S_{23}, c_2)$ である．写像 ψ が意味するのは，テストケースとテスト対象との関連である．たとえば，テストケース c_2 は，$\psi(e_2)$ と $\psi(e_3)$ により，S_{22} と S_{23} をテストすることを示している．なお，$x \in Vs$ が非終端ノードであるとき，x より下位のノード集合を示すものとする．

図 5 は，テストケースのトレース関連をマトリックスとして表現したものである．各テストケース (c_0, c_1, c_2) に対応する列は，テストケースとテスト対象の関連を表現している．テストケースは集合なので，縦の列の順番を入れ替えても同じマトリックスである．マトリックスに記入された ○ 印は，テスト対象がテストされたことを示している．図 5 の “テスト済み” の列は，テスト対象の行ごとに ○ 印をまとめたもので，テストケースの集合 C によってテストされる対象をリス

トしており，テストの網羅性を確認することができる．

4. 適 用 事 例

4.1 要件定義と外部設計

本章では，情報システムへのログイン処理機能を事例として取り上げ，この事例に対しトレース関連を適用し，設計項目およびテストの網羅性管理の有効性について検証する．

ログイン処理は，許可されたユーザだけが情報システムを使えるようにするための機能である．**図 6** は，ユーザ ID とパスワードによる本人認証ができること，不正に使用されないこと，ユーザ ID とパスワードを管理（登録，変更）できるという要件を示している．

図 7 は外部設計の例である．外部設計ではシステムの外側から見える機能について定めている．たとえば，パスワードの有効期限切れが近づいたとき警告を出す（ED-1-3），パスワードは 8〜12 文字以内とする（ED-2-2）ことなどが記述されている．

図 8 は要件定義と外部設計のトレース関連を示している．個別トレース関連を菱形矢印と破線で，包括トレース関連を菱形矢印と実線で示している．要件定義（ソース木）で塗り潰された項目（ノード）は，個別トレース関連のソースに対応する要素であり，白抜きされた部分が刈込みされたソース木を示す．刈込みされたソース木のノードは，下記の場合に発生する．

(1) 要件定義の要素に対応するトレース関連が定義されていない．
(2) 要件定義の要素に対応する外部設計が実施されていない．

(1) はトレース関連の定義漏れであり，該当するトレース関連を定義することにより，ソース木の対応するノードを削除することができる．(2) は外部設計の漏れに該当する．現実の情報システム開発では，要件定義のすべての項目が外部設計に展開されなければならず，刈込みされたソース木が空であることが，外部設計完了の必要条件となる．図 8 では，要件 “R-2-1” に対応する外部設計が行われていないことを示している．

一方，外部設計（ターゲット木）で塗り潰された項目（ノード）は，個別トレース関連のターゲットに対応する要素である．白抜きされた部分が刈込みされたターゲット木を示す．刈込みされたターゲット木は，下記の場合に空ではない．

(1) 外部設計の要素に対応するトレース関連が定義されていない．
(2) 外部設計の要素に対応する要件定義がない．

(1) は，トレース関連の定義漏れであり，トレース関連を定義することにより，ターゲット木の対応するノードを削除することができる．(2) は外部設計で新たに追加された要素を示している．上流工程よりも下流工程の方が詳細化されており，新たな要素が付け加わる．したがって，一般に刈込みされたターゲット木は空にはならない．(2) のケースは，該当する外部設計の内容を精査し，妥当性を判断する必要があることを示している．

たとえば，図 8 の “ED-2-1” の場合，パスワード変更に際し「パスワード変更画面より，新パスワードを取得する」仕様であるが，これは図 6 の要件定義には明記されていない．本件への対応は下記が考えられるが，どちらを選択するかは，プロジェクトの方針に依存する．

(1) 外部設計で新たに作り出された仕様であり，要

ID	項目	
R-1	本人認証	
R-1-1		ユーザ ID，パスワードを不正に使用されないこと．
R-1-2		ユーザ ID，パスワードを他人に知られないこと．
R-2	ユーザ管理	
R-2-1		ユーザ ID パスワードの管理ができること．

図 6 要件定義の例
Fig. 6 An example of requirement definition.

ID	項目	
ED-1	ログイン機能	
ED-1-1		ログイン画面より，ユーザ ID とパスワードを取得する．
ED-1-2		パスワードが正しければ，ログインする．
ED-1-3		パスワードの有効期限が期限切れ 10 日前から期限までの場合，警告を発生し，パスワード変更に誘導する．
ED-1-4		パスワード入力を連続して 3 回ミスするとユーザ ID をロック．
ED-2	パスワード変更	
ED-2-1		パスワード変更画面より，新パスワードを取得する．
ED-2-2		パスワードは，8〜12 文字以内とする．
ED-2-3		英字と数字が混在すること．
ED-2-4		パスワード履歴管理（過去 5 回のパスワードと違うこと）
ED-2-5		ED-2-2，ED-2-3，ED-2-4 を満たすとき，正しいパスワードと判定する．

図 7 外部設計の例
Fig. 7 An example of external design.

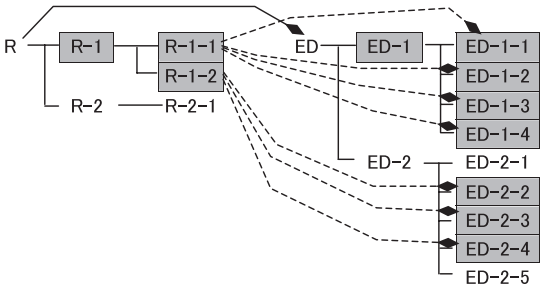

図 8 要件定義と外部設計のトレース関連と刈込み
Fig. 8 An example of trace-relation and truncated tree.

ID	項目
ID-1	ログイン機能
ID-1-1	ログイン画面を表示し、ログイン ID とパスワードを取得する.
ID-1-2	パスワードチェック機能
ID-1-2-1	if (パスワードが正しい)
ID-1-2-2	{パスワード入力ミス回数を0に設定する;
ID-1-2-3	if(パスワード期限切れの 10 日前から期限以内)
ID-1-2-4	{パスワード期限切れ警告画面を表示する;
ID-2	パスワード変更に誘導する};
ID-1-2-5	ログインを実行する}
ID-1-2-6	else {パスワード不正メッセージを表示する;
ID-1-2-7	パスワード入力ミス回数を+1 する;
ID-1-2-8	if(パスワード入力ミス回数 ＞ 3)
ID-1-2-9	{ユーザ ID をロックする} }
ID-2	パスワード変更機能
ID-2-1	パスワード変更画面を表示し、新パスワードを取得する.
ID-2-2	パスワードの正当性をチェックする.
ID-2-2-1	if (パスワードが8文字未満または 13 文字以上)
ID-3	{ パスワード不適正処理};
ID-2-2-2	if (パスワードが英字だけまたは数字だけ)
ID-3	{ パスワード不適正処理};
ID-2-2-3	if (過去5回分のパスワードに一致するものがある)
ID-3	{ パスワード不適正処理};
ID-2-2-4	正しいパスワードと判定する
ID-3	パスワード不適正処理機能
ID-3-1	パスワードが不適正であるメッセージを表示する.
ID-3-2	ユーザに処理継続を確認する.
ID-3-2-1	if(パスワード変更を継続する)
ID-3-2-2	{ パスワード変更を実行する}
ID-3-2-3	else {パスワード変更を中断する}

図 9 内部設計の例
Fig. 9 An example of internal design.

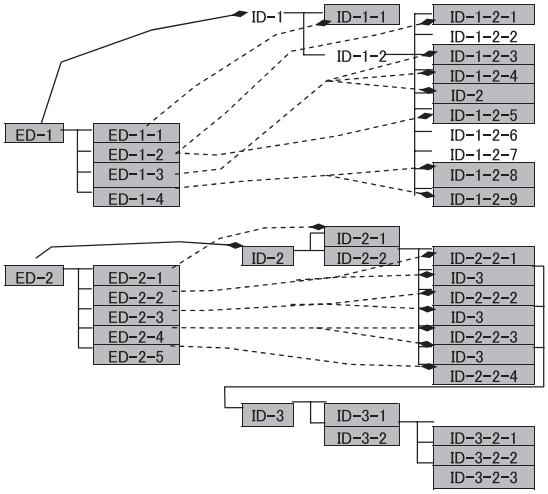

図 10 外部設計と内部設計のトレース関連と刈込み
Fig. 10 An example of trace-relation and truncated tree.

件定義で明記する必要はない.

(2) 外部設計で発覚した要件定義漏れであり，対応する要件を要件定義に追加する.

4.2 外部設計と内部設計

図 7 の外部設計に対応した内部設計を**図 9** に示す. 一般に，内部設計は，情報システムの実装方法についての仕様を記述している．ソフトウェアに関しては，システムの状態を表現するためのデータ構造，および処理順序などの仕様を定める．図 9 では，説明の都合で擬似コーディングによって内部仕様を表現している．外部設計の “ED-1” は，内部設計の “ID-1” に対応しているが，外部設計では宣言的な仕様表現になっているのに対し，内部設計では，処理順序や状態を実装するための方法が定められている.

テストケース	内 容
1	ログイン画面で入力したユーザ ID とパスワードがサーバ側で正しく取得できる.
2	期限切れ 11 日前の正しいパスワードを入力する.
3	①期限切れ 10 日前から期限以内の正しいパスワードを入力する. ②パスワード変更画面で数字8桁の新パスワードを入力する. ③パスワードが不適正と表示されるので、パスワード変更を中断する.
4	3 回連続して正しくないパスワードを入力する.

図 11 テストケース
Fig. 11 Testcases.

No	内部設計項目	テスト 1	テスト 2	テスト 3	テスト 4	済み
1	ID-1 ─ ID-1-1	○			○	○
2	ID-1-2 ─ ID-1-2-1		○	○	○	○
3	ID-1-2-2		○	○	○	○
4	ID-1-2-3		○	○		○
5	ID-1-2-4			○		○
6	ID-2			○		○
7	ID-1-2-5		○			○
8	ID-1-2-6				○	○
9	ID-1-2-7				○	○
10	ID-1-2-8				○	○
11	ID-1-2-9				○	○
12	ID-2 ─ ID-2-1			○		○
13	ID-2-2 ─ ID-2-2-1			○		○
14	ID-3					
15	ID-2-2-2			○		○
16	ID-3			○		○
17	ID-2-2-3					
18	ID-3					
19	ID-2-2-4					
20	ID-3 ─ ID-3-1			○		○
21	ID-3-2 ─ ID-3-2-1			○		○
22	ID-3-2-2					
23	ID-3-2-3			○		○

図 12 内部設計からテストケースへのトレース関連
Fig. 12 Trace-relations from internal design to testcases.

図 10 は，外部設計と内部設計のトレース関連と刈込みを示している．ソース木の刈込みは空であるので，外部設計の項目はすべて内部設計に引き継がれている．一方，刈り込まれたターゲット木は空ではない．この場合，図 8 と同様の，プロジェクトの方針に依存した対応を行う．図 10 に関しては，刈り込まれたターゲット木は ID-1-2，ID-1-2-2，ID-1-2-6，ID-1-2-7 から構成されているが，いずれのノードもシステムの実装面から内部設計で新たに作り出されたものであるので，外部設計への反映は不要と判断するのが妥当である.

4.3 内部設計とテスト

ウォーターフォールモデルでは，内部設計の設計項目からテストケースを作成し，単体テストを実施する.

図 11 はテストケースの一覧である．テストケース 1 は，内部設計項目の ID-1-1 をテストする．同様に，テストケース 2 は，内部設計項目の ID-1-2-1，ID-1-2-2，ID-1-2-3，ID-1-2-5 をテストする．**図 12** は，内部設計項目とテストケースとのトレース関連を図示している．図 12 の “テスト済み” と表示した列は，テストケースによってテストされた内部設計項目を示している．テストケース 1〜4 を実施することにより 23 項目中 18 項目のテストが実施され，設計項目の件数ベースで約 78%のテストが完了することを示している．

5. トレース関連の実装方針

本章では，トレース関連に関する主要概念の実装方針を述べる．なお，データ構造の実装はリレーショナルデータベースを前提とする．

(1) ソース木ならびにターゲット木

木は 1 個の親ノードに対し，0 個以上，複数個の子ノードが対応する．これを実装するには，**図 13** に示した子ノードをキーとするリレーション（R_TREE）によって実装する．

(2) トレース関連

トレース関連は，ソース木からターゲット木への関連，または，ソース木からテストケース集合への関連である．トレース関連は，多対多の関連があり，**図 14** に示したように，“ソース” と “ターゲット” の組合せに対しキーを定義したリレーション（R_TRACE）によって実装する．個別トレース関連と包括トレース関連の区別は，下記の処理によって実装する．

- R_TRACE テーブルの “ターゲット” に記載されている識別子が R_TREE テーブルの親ノードとして存在するとき包括トレース関連と判断する．これ以外の場合は，個別トレースと判断する．

(3) 刈込み処理

ソース木の刈込みは下記の処理で実装する．

① 項目木テーブル（R_TREE）から刈込み対象とするソース木を抽出する．たとえば，外部設計の場合，R_TREE テーブルの子ノード（ID）が “ED” で始まるものを抽出する．

② ① のソース木から，トレーステーブル（R_TRACE）のソースに登録されているノードを削除する．

③ ② のソース木で，下位のノードがない親ノードを削除する．この処理が停止するまで繰り返して得られた結果が刈り込まれたソース木である．なお，ターゲット木の刈込みは，“ソース” を “ターゲット” に置き換えることで実装する．

ID	P_ID	Terms
R-1	R	本人認証
R-2	R	ユーザ管理
ED-1	ED	ログイン機能
ED-1-2	ED-1	パスワードが正しければ、ログインする.
・・・		・・・

図 13 木構造を示すテーブル（R_TREE）
Fig. 13 Table for a tree structure (R_TREE).

Trace_ID	Source	Target
TR 1	R-1-1	ED-1-1
TR_2	R-1-1	ED-1-2
TR_21	ID-1-1	ﾃｽﾄ1
TR_22	ID-1-1	ﾃｽﾄ4
・・・	・・・	・・・

図 14 トレース関連テーブル（R_TRACE）
Fig. 14 Table for a trace-relation (R_TRACE).

表 1 トレース関連テーブルのレコード数
Table 1 Number of records for R_TRACE table.

成果物	項目数	個別トレース	包括トレース
要件定義	500	—	—
外部設計	5000	5000	500
内部設計	8000	8000	800
単体テスト	16000	16000	1600
結合テスト	10000	10000	1000
システムテスト	1000	1000	100
合計	40500	40000	4000

(4) 規模と性能の見通し

本文で想定している大規模システムでのトレーサビリティの管理項目数を**表 1** に示す．テスト項目数は設計項目数の 2 倍とする．また，包括トレース関連数は個別トレース関連数の 10 分の 1 とする．この場合，リレーション（R_TRACE）は，44,000 個のレコードを含む．このリレーションから特定の Trace_ID 44 個を検索するのに 21 ミリ秒を要した．Source および Target をキーにした検索も，同様に 21 ミリ秒であった．性能測定は，データベース ACCESS 2003，検索プログラム Java 1.5.0 で，ハードウェアは，ノート PC（Core™ Duo，メモリ 512 MB）で実施した．この結果から，トレーサビリティの管理に関する性能上の問題はない．

6. おわりに

ウォーターフォールモデルに基づくシステム開発では，各工程が完了する際に，綿密なチェックを行い，工程内の品質を確保する．工程間の成果物の整合性管理については，ISO9000[6] や CMMI[7] でトレーサビリティと呼ばれ，品質管理の重要指標としているものの，開発現場での具体的な品質管理方法については言

及していない．本文はトレーサビリティの実装方法の一手法について論じた．

現実のシステム開発では，設計以降の工程で作成される文書が数千ページになることも珍しくない．成果物の整合性管理が，開発プロセスの品質確保のポイントになるが，文書の分量に圧倒され，各工程間の完了チェックとしての整合性管理が十分に行われていない場合もある．

本文が対象としたのは，システム開発の一断面における成果物管理である．ほとんどのシステム開発では，要件および設計変更が発生し，各工程間の整合性管理をより複雑なものにする．要件および設計変更への対応や，大量の文書からの管理項目の抽出方法などについては，稿を改めて論じる予定である．

参 考 文 献

1) シャリ・ローレンス・プリーガー（著），堀内泰輔（訳）：ソフトウェア工学—理論と実装，ピアソン・エデュケーション (2001).
2) レフィングウェル，ウィドリング（著），石塚，荒川（監訳）：ソフトウェア要求管理，ピアソン・エデュケーション (2002).
3) 山田敬嗣（編集）：シンビオティック・システムの実現に向けて，情報処理，Vol.47，No.8，pp.809–866 (2006).
4) 東証の挑戦，日経コンピュータ，2007年1月8日，pp.34–48 (2007).
5) West，D.B.: *Introduction to Graph Theory*, *2nd ed.*, Englewood Cliffs, NJ: Prentice-Hall (2000).
6) ISO 9001-2000, ISO (International Organization for Standardization) ISO/TC 176 (2000).
7) Understanding and Leveraging a Supplier's CMMI Efforts — A Guidebook for Acquirers CMMI Guidebook for Acquirers Team, Technical Report CMU/SEI-2007-TR-004, Carnegie Mellon University (2007).
8) 高野　敦：成果物・文書管理にすぐ使えるオープンソース (2005). http://www.atmarkit.co.jp/farc/rensai/open02/open02b.html

(平成19年 5 月16日受付)
(平成19年11月 6 日採録)

宇田川佳久（正会員）

1954年生．1982年東京大学大学院博士課程修了．工学博士．同年三菱電機（株）入社．2006年より三菱電機インフォメーションシステムズ（株）に勤務．データベースの研究を経て，直近の約10年間は，製造業および金融業向けWebシステムの設計，開発，プロジェクト管理に従事．情報システムの品質向上，開発の効率化に興味がある．著書に，『オブジェクト指向データベース入門』（(株）ソフト・リサーチ・センター，1992年）等．元学会誌編集委員，論文誌査読委員．