ゴール指向要求分析法を用いたソフトウェアテストの品質向上

Improve Quality of Software Test using Goal-oriented Requirements Analysis

沖 汐 大 志

要　約　ソフトウェアの品質を確保するためには，要求を正しく理解して設計や実装に反映するとともに，適切なテストを行う必要がある．しかし，開発の工程が進む途中で要求が誤解されたり，設計の考慮漏れが発生したりすることがあり，それがテストの漏れや誤りによって見逃されることがある．本稿では，KAOS法の四つのモデルにテストモデルを追加することによって，要求獲得時にテストで確認するべきことを明らかにし，テストモデルとゴールモデルを使用してテストの品質を向上させることを提案する．

Abstract　It is essential for the quality of software to test the program appropriately. But the errors are often caused by the misunderstanding of requirements and the lack of careful design considerations. And faults caused by those errors are often missed in the test. It is possible to prevent these problems by the testing based on a correct understanding of the software requirements. In this paper, we propose a method to improve the test quality by adding a new model to KAOS method viz. test model. Finally, we present examples to improve the quality of software test by applying the test model and goal model.

1. はじめに

顧客に高品質なソフトウェアを提供するためには，要求仕様書や設計書が正しく記述されている必要がある．プロジェクトの失敗要因の40%は要求定義工程にあるといわれているが，まず要求を正しく理解し，正確な要求仕様書を作成することによって，要求定義以降の工程に反映させていくことが必要である[1][2]．そして，こうした正しい要求や設計を基に，ソフトウェアが正確に作りこまれていることを，適切なテストによって確認しなければならない．

テスト設計では，はじめに要求仕様書や設計書を基にテストシナリオが作られ，そのテストシナリオを基にテストケースが作成される．すなわち，テストの品質に大きな影響を与えるテストケースの品質は，テストシナリオの基になる要求仕様書や設計書が正しく記述されていることによって確保できる．しかし，開発工程が進む中で，要求の認識誤りや設計時の考慮漏れなどによって，要求が要求仕様書や設計書に正しく反映されていないことがある．そしてそれがテストケース漏れの発生や誤りの原因になることがある．

このようなことを防ぐためには，要求の基になった上位ニーズや意図を理解する必要がある．要求の上位ニーズや意図を明確に理解するには，ゴール指向要求分析法[3]によって作成されたゴール木を参照することが有効である．ゴール木の上位ゴールから，テストの目標，すなわち，テストゴールを設定することが可能である．一方，有効なテストケースを作成するには，要求の意図や背景を基にテストのための具体的な条件を定義しておく必要がある．これをテストプロパティと呼ぶことにする．テストゴールやテストプロパティは，要求獲得フェーズで作成され，ゴール木の中に組み込まれることによって，テストケース作成時にテスト設計者が参

照できるようになり，有効なテストケースの作成に寄与することができる．

本稿では，要求獲得手法の一つである KAOS 法[4]の手順の中に，要求を達成した場合に満たすべき状態と，その状態を確認するために必要なテストゴールやテストプロパティをテストモデルとして取り入れることを提案する．KAOS 法にテストモデルを取り入れ，2. 1 節で説明するゴールモデルと合わせてテストケースのチェックに使用することにより，テスト設計時に発生するテストケースの漏れや，不必要なテストケースの作成，勘違いによる誤ったテストケースの作成などを回避できる可能性が高くなる．

例えば，ユーザから「1 秒当り 100 件程度の利用」という要求が提示されたとする．ユーザの意図が「1 秒間に 100 件の取引成立」であるのに，要求分析者が「1 秒間に 100 件の Web アクセス」という解釈をし，それを基に設計や実装が行われてしまうと，必要な性能要求を満たせない最終製品が作成されてしまうことになる．このような勘違いによる解釈の違いは，最終的にはテストによってチェックできなければならないが，要求の定義にテストという視点を持ち込むことによってこうした勘違いを防止できる可能性が高くなる．また，テストゴールやテストプロパティを作成しておき，これらを使用してテストケースをチェックすることによって，テストケースに勘違いを反映してしまった場合でも，これを検出することが可能となる．

2. KAOS 法とテストモデル

2. 1 KAOS 法のモデル

本来の KAOS 法には，次の四つのモデルが提案されている[4][5]．

1） ゴールモデル

ユーザのニーズや意図から個別の要求までを，目標とする状態によって系統的に表現したモデルである．要求は，ゴール木の最終末端ゴールとして表現される．

2） 責務モデル

どのエージェントが，どのゴールの達成に責務を持っているのかを表現したモデルである．KAOS 法では，対象エージェントが特定できた時点でゴール分解を停止することになっている．

3） 操作モデル

ゴール達成のための操作と，物理的なオブジェクトに対する入出力を表現したモデルである．KAOS 法では，要求（末端ゴール）の形式的な記述から，操作を導出する．これを Operationalization という[5]．

4） オブジェクトモデル

ゴールを実現するための操作で使用するオブジェクトを，オブジェクト指向ではなく，オブジェクトベース[6]によって表現したモデルである．

四つのモデルとその関係を図 1 に示す[7][8]．ゴールモデルと責務モデルの代表的な構成要素の定義と図式は，表 1 の通りである．

KAOS 法のゴールモデルでは，ゴール分解を通して作成されたゴール木を基に，要求同士の衝突や，要求の実現を阻害する障害を洗い出し，代替案や要求の優先順位付けを行いながらゴールを洗練し，責務モデルを使って要求を導出していく．そこで得られた要求は，Operationalization によって，具体的な操作に変換される．

本稿では，これら四つのモデルに新たな「テストモデル」を追加して，導出された要求が設計工程やテストに正しく反映されていることをチェックする方法を提案する．

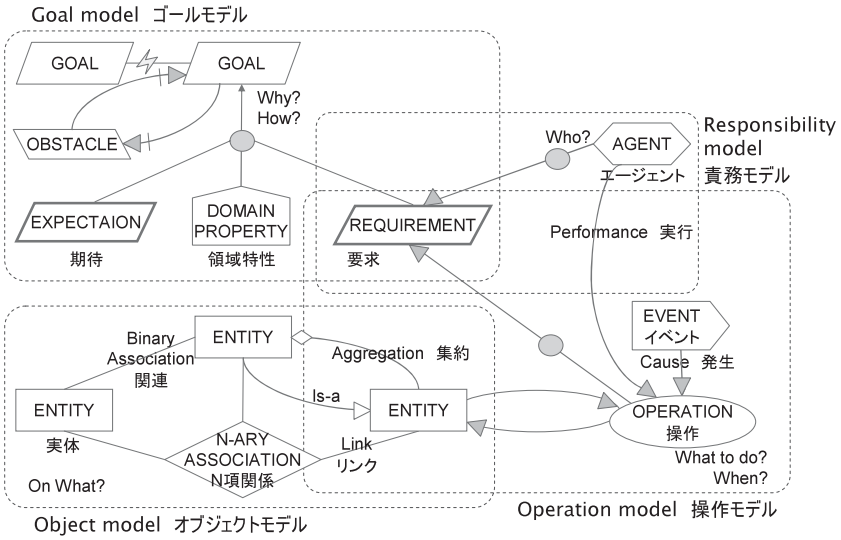

図 1　KAOS 法の四つのモデル

表 1　代表的な構成要素の定義と図式

モデル	構成要素	定義	図式
ゴールモデル	ゴール	達成すべき状態	
	ソフトゴール	達成が望ましいゴール	
	要求	一つのソフトウェアエージェントに割り当て可能な程度に十分小さく分割されたゴール	
	領域特性	システムに依存せず，問題領域で考慮すべき性質．仮説，不変性質を扱う．	
	洗練化	AND-OR分割によるゴールの洗練化	G → G
責務モデル	エージェント	個々のゴール達成のために，操作を実行する能力を持つ実体	
	責任	要求または期待の達成に関して，エージェントが責任を持つ．要求または期待に対して責任を持つことができるエージェントは一つのみ．	Ag → Re

2.2　テストモデルと構成要素

KAOS 法のゴール木には，ヒューマンエージェントによって実現される要求と，システムエージェントによって実現される要求が混在している．テストモデルは，このうち，システムエージェントに関係する要求やソフトゴールについて，どのような状態を確認できれば要求を実現できるか，そのためにはどのような事前条件や事後条件を設定するべきかを記述し，適切なテストケースの作成を支援するためのモデルである．図 2 に，KAOS 法の四つのモデルとテストモデルの関係を示す．テストモデルは以下に挙げる三つの要素で構成され，要求を介して他の四つのモデルと関連を持つことになる．

・要求仕様（Requirement Specification）
・テストゴール（Test Goal）
・テストプロパティ（Test Property）

ヒューマンエージェントに関わる要求はシステム化されないので，テストゴールの適用対象外である．

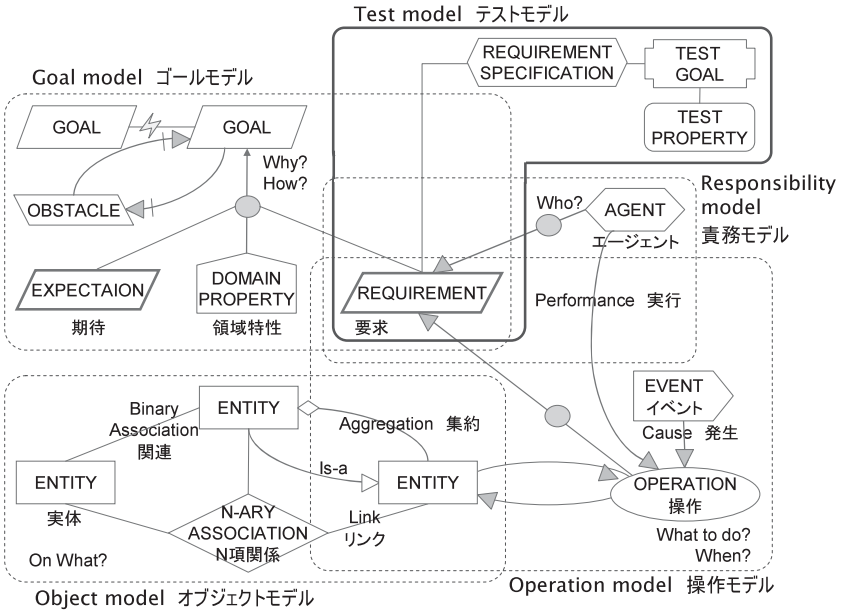

図2　KAOS法の四つのモデルとテストモデル

2.2.1　要求仕様とテストゴール

KAOS法では，目標が達成された状態をゴールとして定義する[4][5]．それゆえ，テストモデルにおいても，テストゴールには，システムが要求を実現できていることをテストによって確認するための「目標状態」を記述する．

要求は現実世界の中に存在している．しかしながら，テストは作成されたプログラムに対し実施されるものであり，要求そのものをテストすることはできない．テストゴールを設定するには，現実世界の要求をコンピュータ世界に写像しなければならない．Jackson，M.らは，共有現象という概念を使って要求を要求仕様に写像すべきとしている[9]．共有現象とは，現実世界とシステムの双方が観察することのできる現象のことである．この定義に従えば，テストできるのは要求ではなく，要求仕様である．テストモデルの中で，要求は要求仕様に変換され，その要求仕様毎にテストゴールが設定される．

Jacksonらは，文献[9]で，動物園の入園管理を例に，要求と要求仕様の違いを説明している．すなわち，現実世界の入園者をシステムが直接カウントすることはできないが，動物園の入口にある回転ドアの回転数によってこれを知ることが可能である．そして，このドアの回転数は，仮想世界に構築された虚構ではなく，現実世界でも観察可能な共有現象である．こうして，動物園への入園者数という要求は，ドアの回転数という仕様に変換されなければならない．この例を使って，意図，要求，要求仕様，テストゴールの関係を説明すると，

意図：売上金額が正しいことを確認する．

要求：売上金額と入園者数が一致していることを確認する．

要求仕様：売上金額とドアの回転数が一致していることを確認する．

テストゴール：売上金額から，ドアの回転数に入園料を乗じた値を引いた値が，常にゼロである．

となる．図 3 は，上述の要求仕様とテストゴールを図式化した例である．

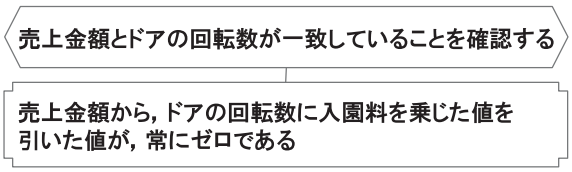

図 3　要求仕様とテストゴールの例

2. 2. 2　テストプロパティ

個々のテストゴールごとに，そのテストに関する条件をテストプロパティとして定義する．すなわち，テストプロパティとは，テストにおいて確認する必要がある基本条件の集合である．典型的なテストプロパティは，以下のとおりである．

・事前条件：要求仕様が実現されるためのシステムの状態
・事後条件：要求仕様が実現した後のシステムの状態
　－定常事後条件：定常状態で要求仕様が実現した時のシステムの状態
　－特殊事後条件：要求仕様実現中に非定常状態が発生した時のシステムの状態
・期待値：要求仕様が実現した後のシステムの処理
　－定常期待値：定常状態で要求仕様が実現した時のシステムの処理
　－特殊期待値：要求仕様実現中に非定常状態が発生した時のシステムの処理
・使用データ：テストで使用されるデータ
　－定常使用データ：定常状態のテストで使用される入出力データ
　－特殊使用データ：特殊条件をテストするための入出力データ
・不変定数：要求仕様実行中に変更されてはならない定数
・制約事項：処理モードやアーキテクチャなど，テスト時の制約事項

具体的なテストプロパティの項目は，要求やシステムの特性によって異なると考えられる．プロジェクトごとに適切なプロパティ項目を追加する必要がある．ここに挙げた例は，最低限の項目である．

図 4 は，2. 2. 1 項の例に対応するテストプロパティの例である．

2. 2. 3　テストゴールの記法

KAOS 法に追加されるテストモデルの中で，要求仕様，テストゴールとテストプロパティは，図 5 のような図式で表現するものとする．

◆事前条件

・ドアの回転数（入園者数）がカウントされている

・入園料が徴収されている

◆事後条件

▲定常事後条件

・ドアの回転数に入園料を乗じた値が売上金額に一致する

▲特殊事後条件

・ドアの回転数に入園料を乗じた値が売上金額に一致しない

◆期待値

▲定常期待値

・ドアの回転数に入園料を乗じた値を印刷する

▲特殊期待値

・ドアの回転数に入園料を乗じた値と売上金額の差額
　およびメッセージを印刷する

◆使用データ

▲定常使用データ

・ドアの回転数（入園者数）

・売上金額

▲特殊使用データ

・入園料を支払い済みで，ドアの1回転中に共連れで入園した人数

・入園料を払い戻した人数

・入園者を伴わず，ドアだけが回転したときの回転数

◆不変定数

・入園料

◆制約事項

・閉園後にバッチ処理で計算する

図4　テストプロパティの例

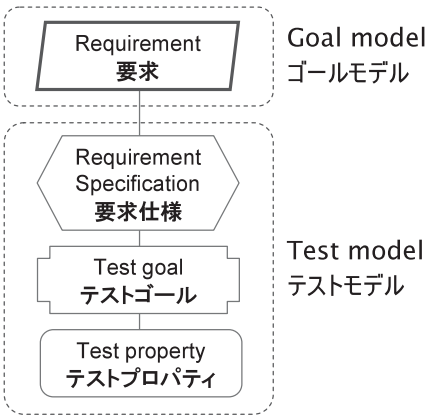

図5　テストゴールとテストプロパティの図式

3. テストモデルの設計

3.1 要求仕様の定義

KAOS 法では，導出された要求の中で，システムエージェントに関係した要求をソフトウェア要求としている．テストでは，こうして導出されたそれぞれの要求ごとに，それらがプログラム上で達成されていることを確認しなければならない．しかし，2.2.1 項で述べたように，要求ではなく要求仕様に基づいて作成されたプログラムをテストするには，テストゴールも要求仕様に基づいていなければならない．仕様化するために選択された現象によってテストゴールも異なってくる．2.2 節の例で言えば，入園者数の確認のために，ドアの回転数ではなく，入園券の半券を使用するとすれば，テストゴールだけでなくテストプロパティも異なってくる．したがって，テストモデルでは，要求とテストゴールの間に要求仕様の定義が必要となる．ここで採用された要求仕様は，当然，要求仕様書に反映されることになる．

3.2 テストゴールの設定

2.2.3 項で述べたように，テストゴールは，KAOS 法のゴールの表現方法に従い，状態表現で記述する．システムの目的が不具合の解消であれば，テストゴールはその不具合を解消した状態をプログラムとして定義したものとなるし，目的が業務効率の改善であれば，テストゴールは効率改善の目標値を実現した状態の定義となる．

テストゴールは，機能要求については，達成したかどうかをプログラム上で具体的に確認できるよう形式的あるいは定量的に記述するのが望ましい．非機能要求についても，できるだけ定量的にゴールの達成を判断できるように記述する．FURPS モデルの品質属性ごとにテストゴールを記述した例を表 2 に示す[10]．定量化できない場合は，その達成を確認するための方法を付記しておく必要がある．

表 2　品質属性とテストゴールの記述例

品質属性	テストゴールの記述例
Functionality	xx機能について，指定したデータを入力したときの出力が期待値〇〇となること
Usability	エンドユーザが，90%の処理を，マニュアルを見なくても操作できること
Reliability	xx障害発生時に，システム担当者が復旧をhh時間以内に完了できること
Performance	xx機能の応答時間がss秒以内であること
Supportability	システム担当者が，システムの導入をhh時間以内に完了できること

3.3 テストプロパティの作成

テストゴールによって示された「システムが要求仕様を満足した状態」を確認するために必要となる事前条件，事後条件，使用データ，制約事項などをテストプロパティとして記述する．テストプロパティは，KAOS モデルが完成している場合には，操作モデルとオブジェクトモデルを参考にすることができる．

事前条件や事後条件は操作モデルを作成する際に使用した事前条件と事後条件を参照して作成する．事前条件は，要求を実現する処理が実行されるための状態を記述する．それは，とりもなおさず，その要求に関する仕様の境界を決めることを意味している．事後条件は，要求仕様に記述された処理が実行された後の状態を記述する．

使用データは，操作モデルの入出力データを参照して作成する．操作モデルやオブジェクトモデルが作成されていない場合は，操作シナリオなどを作成することによって，テストで使用する入出力データを特定する．入出力データから，定常条件が成り立つ場合に使用するデータを定常使用データとして選択し，特殊事後条件が成立する場合に使用するデータを特殊使用データとして選択する．

不変定数は，操作モデルやシナリオの中で使用するパラメータや変数をリストアップし，その中から変更してはならないものを選択する．2.2.2 項の図 4 の例では，この処理の中で入園料が変更されてはならないことを表している．

処理モードやアーキテクチャによってテスト環境が異なり，そのことによってテストプロパティが影響を受ける場合がある．テストモデルの前提となっている条件は制約事項として記述しておく．図 4 の例では，この処理が動物園開園中にリアルタイムに行われる場合と，閉園後にバッチ処理で行われる場合で，テストの初期状態（既売上金額の値など）が異なる場合があるので，処理モードを指定してある．

これらの情報を利用することにより，システムが要求の目標値や基準を満たしていることを明確に確認できるテストケースの作成が可能になる．

4. ケーススタディ

本章では，ケーススタディとしてテストモデルを使用してテストケースの漏れや曖昧性を検出する例を示す．

4.1 異常系のテストケース漏れの検出例

我々は通常，次のような手順でブラックボックスのテストケースを作成している[11]．

① 設計書の処理仕様を基に，デシジョンテーブルなどの技法を使ってテストシナリオを作成する
② 設計書の入力仕様を基に，同値分割や境界値分析などの技法を使ってテストデータを作成する
③ テストシナリオとテストデータを組み合わせて，テストケースを作成する

この手順の中でブラックボックスのテストケースに漏れが発生する原因としては，次のようなことが考えられる．

① 設計書の考慮漏れや記述漏れに気づかずにテスト設計を進めてしまい，間違ったテストケースを作成したり，テストケースに漏れが発生したりする
② 設計書に問題はないが，テストシナリオ作成時に不備があり，テストシナリオに漏れが発生することによってテストケースにも漏れが生じる
③ 設計書とテストシナリオには問題はないが，テストデータの選択に漏れがあり，テストケースに漏れが生じる

これら三つの原因のうち，①についてはテストモデルによる検証が有効である．設計書の考

慮漏れや記述漏れにはさまざまな要因が考えられるが，正常系の処理よりも異常系の処理の場合に多くみられる[11]．この理由の一つは，設計者が，正常系の処理があれば最低限の機能を満たしていると安易に考えてしまうことにある．また，異常系の処理はどれだけ挙げれば十分といえるか判断し難く，稀に発生するような異常系を見逃してしまう場合もある．そして，異常系の処理は漏れていても気づきにくい．このような，設計段階での異常系の処理の漏れに伴うテストケースの漏れについては，要求獲得段階で作成したテストモデルを利用することによって，その有無を確認できる．これについては，本節の後半で例を説明する．

②のテストシナリオが漏れるケースについては，テスト設計に失敗しないようにテスト技法の習熟を図ることや，テストシナリオのレビューのためにデシジョンテーブルなどを成果物として残すことなどが有効である．例えば，デシジョンテーブルによって作成したシナリオから必須のもの以外を省略しようとして，必須のシナリオまで省略してしまうこともある．このような失敗は，レビュアがデシジョンテーブルなどのテスト設計の成果物を見て設計過程を確認しながらテストケースをレビューすることによって，検出の可能性を高めることができる．

③のテストデータの漏れについては，同値分割や境界値分析によるテストデータの選択はできていても，Special Value や Error Based と呼ばれる技法によってテストデータを選択する際に漏れが発生することがある．この理由の一つとして，同値分割や境界値分析のように機械的に選択するのではなく，Special Value や Error Based の場合は，過去の経験やシステムの背景などの知識が必要となることが挙げられる．こうしたケースはゴール指向要求分析法のゴールモデルによって補い，漏れを防ぐことができる．また，テストモデルの特殊事後条件や特殊使用データによって，どのようなテストデータを追加するべきかのヒントを得ることができる．これが，テストモデルを KAOS 法のゴールモデルに連動させた理由である．

ここでは，①の設計の考慮漏れや記述漏れが原因となる異常系のテストケース漏れについて，テストモデルを利用して検出する例を説明する．

2 章で取り上げた動物園の例では，異常系として，ドアの回転数に入園料を乗じた値と売上金額が一致しない場合の処理がある．この不一致は，通常は一人ずつ回転ドアに入るところを，稀に複数人が同時に入ってしまったり，入園料は払ったが実際には入園しなかったりということが原因で生じる．このような発生頻度の低い特殊な処理系は設計段階で見落としやすくなるが，この見落としを原因とするテストケースの漏れは，テストモデルの特殊事後条件に対応するテストケースが存在するかどうか確かめることで検出することができる．この場合の特殊事後条件と特殊期待値は，2.2.2 項の図 4 に示したように次の通りである．

▲特殊事後条件

・ドアの回転数に入園料を乗じた値が売上金額に一致しない

▲特殊期待値

・ドアの回転数に入園料を乗じた値と売上金額の差額，およびメッセージを印刷する

この特殊事後条件とテストケースを見比べ，対応するテストケースが漏れていることが分かった場合は，表 3 の 2 行目および 3 行目のようなテストケースを追加する．そして，テストケース漏れの原因となった設計書も修正する．

表 3　テストケースの例

条件	期待値
売上金額から，ドアの回転数に入園料を乗じた値を引いた値がゼロのとき	ドアの回転数に入園料を乗じた値を印刷する
売上金額から，ドアの回転数に入園料を乗じた値を引いた値がゼロより大きいとき	ドアの回転数に入園料を乗じた値と，売上金額の差額およびコメント「入園料金過剰」を印刷する
売上金額から，ドアの回転数に入園料を乗じた値を引いた値がゼロより小さいとき	ドアの回転数に入園料を乗じた値と，売上金額の差額およびコメント「入園料金不足」を印刷する

4. 2　テストケースの曖昧性の検出例

次に，ゴールモデルによりテストケースの曖昧性を検出する例を示す．2 章で取り上げた動物園の例で，テストモデルの基になるゴールモデルは，図 6 の通りとする．

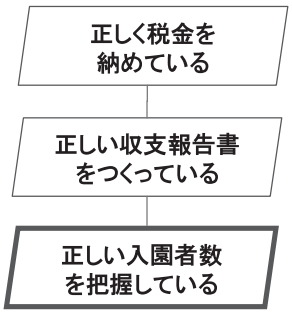

図 6　ゴールモデルの例

このゴールモデルでは，正しい収支報告書を作成するためのソフトウェア要求は「正しい入園者数を把握している」である．一方，テストゴールは「売上金額から，ドアの回転数に入園料を乗じた値を引いた値が，常にゼロである」である．これらの二つの情報から，ソフトウェア要求が示す「入園者」とは，来園した人数を指していることがわかる．

つまり，この動物園が，一度入園料を支払えば終日入園と退園が自由である場合，テストデータとして使用するドアの回転数は，再入園者を除外した数でなければならないことがわかる．表 3 のように，テストケースがこのことを十分に表現しきれていない場合は，説明を追加して区別できるように修正する．

5. お　わ　り　に

本稿では，ゴール指向要求分析法の一つである KAOS 法にテストモデルを追加し，テストモデルとゴールモデルを利用してテストケースの漏れや誤りを発見する方法を提案した．

テストケースの漏れや誤りには，設計工程での考慮漏れや認識違いに起因するものがある．このようなテストケースの問題を検出するには，要求分析時にテストに必要な要件を明らかにしておき，テストケースのチェックに利用することが方法の一つである．テストモデルは，そうしたテストケースのチェックに有効である．また，テストの品質を向上させるには，テスト技術の適用に加え，要求を正しく理解して，その実現を適切に確認できるようにすることが必

要である．ゴールモデルは要求の背景を理解するのに有効なモデルであり，本稿では，ゴールモデルとテストモデルの融合がテストの品質向上に役立つことを示した．今後は，テストモデルを実務に適用することによりブラッシュアップしていきたい．

最後に，本稿の執筆あたりご指導いただいた関係者の皆様に厚くお礼申し上げます．

参考文献
［1］ Standish Group Chaos. Standish Group Inc, 1994.
［2］ IEEE, Recommended Practice for Software Requirements Specifications. Std 830-1998, 1998.
［3］ Anton A. I. Goal-based requirements analysis. Proc. of 2nd International Conference on Requirements Engineering, 1996, pp. 136-144.
［4］ A. van Lamsweerde, “Goal-Oriented Requirements Engineering: A Guided Tour”, Symposium on Requirements Engineering, 2001, pp. 249-263.
［5］ A. van Lamsweerde, Requirements Engineering, Wiley, 2009.
［6］ CIW ファンデーション公式テキスト，CIW，2008.
［7］ ゴール指向要求分析，トップエスイープロジェクト，国立情報学研究所，2008 年 9 月
［8］ 妻木俊彦，白銀純子，要求工学概論，近代科学社，2009
［9］ Jackson, M. and P. Zave, Deriving specifications from requirements: An example. Proc of 17th International Conference on Software Engineering, 1996, pp. 15-24.
［10］ Robert Grady, Practical Software Metrics for Project Management and Process Improvement, Englewood Cliffs, NJ: Prectice-Hall, 1992
［11］ 大塚俊章，荻野富二夫，ソフトウェアテスト技術，ユニシス技報，日本ユニシス，Vol.27 No.2，通巻 93 号，2007 年 8 月

執筆者紹介　沖　汐　大　志（Motoji Okishio）
1990 年日本ユニシス（株）入社．CAD/CAM システムの開発・適用支援に携わった後，2007 年より品質保証に関する作業に従事し，現在は品質保証部品質管理室に所属．