2022 年 12 月 19 日
第 2 版

ソフトウェアテスト技法ドリル【第 2 版】

演習問題解答例

第 1 章

1.1

まずは、重要な機能と、入力について明らかにします。

重要な機能は、「体温を計測し結果を表示する」、「体温が 38℃以上の場合には発熱警報を鳴らす」でしょう。入力は、基本的に体温となりますが、「外気温」、「設置後の時間（精度に関係するはず）」、「電池の残量」といった環境やシステムの条件に対するテストがどれだけなされているかを確認する必要があります。ソフトウェアテストはハードウェアの評価後に実施されることが多いので、ハードウェアの評価結果を入手し、重複しそうなテストは最小限にします。HW+SW のシステムとしての評価もありますので、ハードウェアの評価がなされているからといって、ソフトウェアテストでは一切そこは実施しないというのは得策ではありません。多少の重複を許容しながらテストすることが大切です。

また、気になる点としては、「測定結果の表示が消えるまでの時間」があります。測定した体温の表示時間が短ければ、読み損ねが発生するでしょうし、長すぎれば、次の人を待たせることになるからです。また、38℃以上のときに鳴る発熱警報音については、警報音を手動で止めるのか、タイムアウトで止めるのかなどの仕様の確認が必要です。

さらに、低体温側について、何℃（例えば 35℃）以上に対応しているのかも気になります。冬の外気では手や額の温度が低いケースも想定されるからです。また、電池の減り具合もテストで確認したいところです。細かなことをいえば「1 秒以内に測定し」とありますので、測定に要する時間が 1 秒以内であって、その後に表示されるまでタイムラグがあるかもしれません。タイムラグの要因としては、データの保持が考えられます。昨今では近くにあるサーバーに Bluetooth で接続して測定データを記録する処理が走っているかもしれません。

上記仕様には書かれていませんが、ユーザーとのインターフェース以外の仕様は別の仕様書に書かれることも多いので注意が必要です。

次に、入力については、「間」、「対称」、「類推」、「外側」を考えます。

① 間

測定距離について、5～10cm とあります。体温計から 5cm までの間に手をかざしたときにはエラーになるのでしょうか。確認が必要です。

② 対称

外気温が体温を超えたケースでどうなるかの確認が必要です。（日本でも 2018 年と 2020 年に 41.1℃の記録があります。建物の入り口に設置されることを考慮しますとテストで結果を確認する必要があるでしょう。）

③ 類推

人間以外（例えば、ペットの犬）を測定したらどうなるでしょうか。体温計がペットのお腹と、人の手の違いを識別するとは思えません。犬の平熱は、38～39℃とのことですので、発熱警報が出るだけだと思いますが、例えば、動物病院の入り口に設置されたらこのようなユースケースがそこそこの頻度で発生するかもしれません。

④ 外側

音量設定は 5 までとあるため、6 の設定ができないことを確認します。また、操作する人に入力が上限値を超えていることが伝わるかどうかも評価したいところです。（例えば、一瞬点滅することで入力が受け付けられる一方で、処理としては受け付けないことが伝わるようにする、など）です。

これらについて、すべてテストするわけではありません。しかし、テスト設計を行うためには、まずはさまざまな視点からテスト対象物を眺めてみる必要があります。

電気通信大学の西康晴先生は、NGT/VSTeP という方法でテスト観点を整理しながらテストをつくる方法を提案しています。とても有効な方法ですので興味のある方は調べてみてください。

1.2

「間違え」というと、開発者に叱られてしまうかもしれませんが、「仕様に明記されていない要求の実装漏れ」は発生しやすいものです。この設問でいえば、ボタンを押した後に、入力カーソルがどこに行くかが書かれていません。

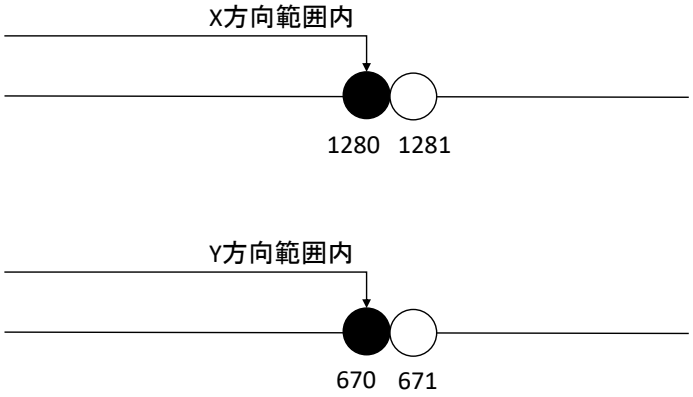
町名まで入力するのだから町名の後にカーソルが移動し、ユーザーが「1 丁目 2－3」というように、町名に続いて残りの番地を入力しやすくするべきです。このように明示されていない仕様のことを「暗黙の要件」といいます。また、郵便番号の桁数やハイフン「-」の有無、ハイフンの位置について、どこまで受け入れ、どこからはエラーにすべきでしょうか？　例えば、ハイフンが 2 つあった場合、筆者はエラーとしたほうが良いと思いますが、いかがでしょうか。

また、暗黙の要件のほかにも、開発者は自分の環境でのみ動作確認をすることが普通です。したがって、例えばこれが、ウェブアプリであればブラウザによる差の確認が必要です。実際に特定のブラウザのときに住所の自動入力で文字化けをするケースがありました。

第 2 章

2.1

まず、線と●○を描きましょう。



X（横方向）について 1280 と 1281、Y（縦方向）について 670 と 671 とそれぞれ 2 つずつ境界値が見つかりました。したがって、1280×670、1280×671、1281×670、1281×671 の 4 つの画像サイズが見つかりました。でもちょっと待ってください。「有効な値は複数同時に確認できるが無効な値については 1 回に一つずつしか確認できない」のでした。今回の仕様を「トリミングしたときに、メッセージを出力する」と解釈した場合、「1281×671」のテストをしても、1281 という値によってメッセージが出たのか、それとも 671 という値によってメッセージが出たのか区別できません。

よって、答えは、「1281×671」を除く「1280×670、1280×671、1281×670」の 3 つとなります。

※ 最小サイズの「0×1、1×0、1×1」（0×0 はどちらも無効な値なので除いている）が気になった方がいらっしゃるかもしれません。余裕があれば、「1×1」のテストを実行しても良いかもしれません。なお、（0×1 と 1×0 の画像は、そもそも、作成できないため優先度は低くなります。）

ただし、「1×1」のテストは、ユーザー視点の優先度が低くなります。

2.2

風量を記憶していた整数型の変数の値が▽ボタンを押すごとに小さくなり、0 になったときに、風が止まった。さらに▽ボタンを押したところ、本来は 0 のままであるべきところを、0 から 1 を引いてしまい、結果として、－1 という値になったが、－1 を正の整数として取り扱ってしまった（通常は、その変数型の最大の整数値になる）ことにより、強い風が吹いてきたと考えられます。

第 3 章

3.1

デシジョンテーブルを作成する前に、入り組んだ用語を図 A.1 のように、木構造で整理します。文章から図にすることで地域の定義という込み入った仕様が明確になります。

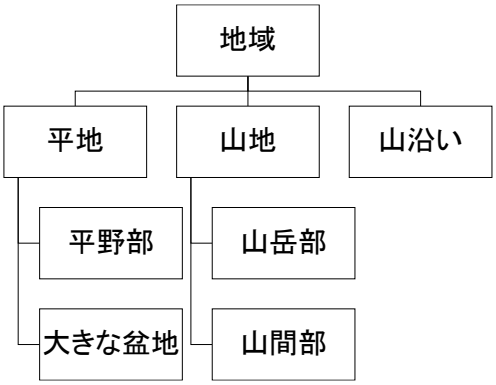


図 A.1　気象庁における地域の定義の木構造

次に、図 A.1 を表に書き直すことで、図 A.2 のような判定結果だけ埋まったデシジョンテーブル表をつくります。

平地	平野部	
	大きな盆地	
山地	山岳部	
	山間部	
山沿い		

図 A.2　地域の定義のデシジョンテーブル（判定結果のみ）

ここまでできれば、あとは判定結果をもたらす条件を記入し、デシジョンテーブルを完成するだけです。完成したデシジョンテーブルは図 A.3 のとおりです。

条件と判定結果		ルール1	ルール2	ルール3	ルール4	ルール5
起伏が極めて少ない		Y	Y	N	N	-
山に囲まれている		N	Y	-	-	-
山と山の間		-	-	N	Y	-
山に沿った地域		-	-	-	-	Y
平地	平野部	X				
	大きな盆地		X			
山地	山岳部			X		
	山間部				X	
山沿い						X

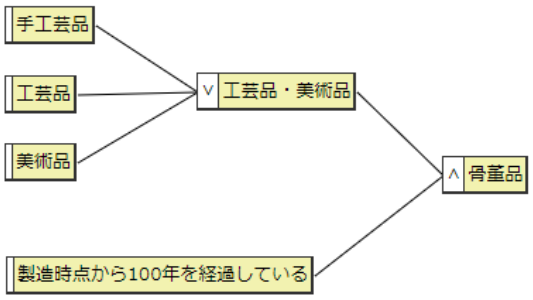
図 A.3　地域の定義のデシジョンテーブル（完成版）

こうして、デシジョンテーブルを作成すると、ここに現れてこない「海上」、「海岸」、「沿岸」、「沿岸の海域」、「沖」といった、陸に相対する海の用語が気になるかもしれません。仕様を整理すると、「仕様に記載されていない事項に気がつく」という効用もあります。

3.2

まず、骨董品の定義に関係する原因を抽出します。すると、「製造時点から 100 年を経過」、「手工芸品」、「工芸品」、「美術品」の 4 つが見つかります。ここで、絵画・掛け軸」「焼き物・陶磁器」は、骨董品の定義ではなく、骨董品の種類を例示したものであることに注意してください。

原因となるノードが 4 つありますからすべての組合せは $2^4=16$ となりますが、原因結果グラフを描くことで 16 のうちの 4 ケースが抽出されました（図 A.4)。この 4 ケースは各条件（4 つの原因ノード）の True/False を確認する網羅性（MC/DC カバレッジ）をもっています。



デシジョンテーブル

ノード名			#1	#2	#3	#4
原因:		手工芸品	T	F	F	F
		工芸品	F	T	F	F
		美術品	F	F	F	T
		製造時点から100年を経過している	T	F	T	T
中間:	{obs}	工芸品・美術品	T	T	F	T
結果:	{obs}	骨董品	T	F	F	T

図 A.4　骨董品の原因結果グラフとデシジョンテーブル

第 4 章

4.1

まず、テレビの目的機能を考えます。テレビはさまざまな目的で使われますから、目的機能も抽象的な記述になります。「何かの情報を得るために、テレビ放送を視聴する」です。

次に目的機能に対する入力を考えます。今回は仕様に 5 つしかありませんのでそれをすべて入力においています。なお、組合せテスト時は「電源」は入力とはせずに「電源が入っていること」前提条件の一つとします。

出力はテレビなので、「音」と「映像」です。状態変数として「音量値」と「チャンネル番号」としたのは、テレビは電源を入れたときに前回の音量とチャンネルを表示するからです。

ノイズは、入出力を妨げる要因ですから、外から聞こえる生活音や部屋の照明（蛍光灯と白熱灯でも見え方が異なるかもしれません）で、アクティブノイズは、人が行うノイズですが、あまり思いつきません。ここでは、「真っ暗に」としました。真っ暗なときに手探りでリモコンを操作できる機能性があるかどうかをテストしたいと考えたからです。

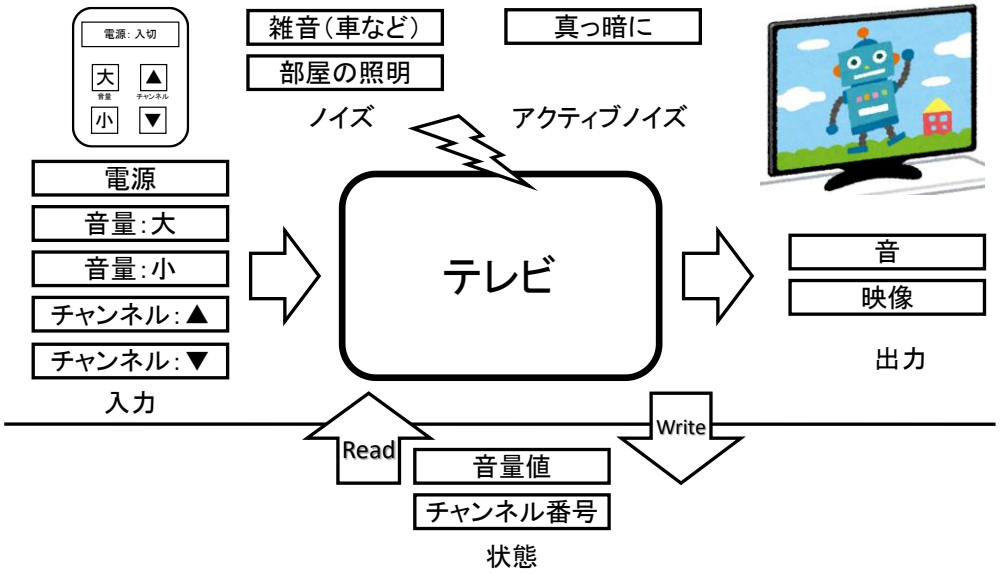


図 A.5　テレビのラルフチャート

4.2

ここでは、表 A.1 のとおり「画像の種類」と「カメラの種類」を追加しました。これら 2 つの因子は、開発側では制御できない要因です。したがって、テストの際にはシステムに対して厳しい水準をテストすべきです。厳しいとは、例えば、大きなファイルであったり、色再現が難しい微妙な緑色だったりします。

また、実際の組合せテストのときには、用紙サイズに「フォト用紙」を選択するとカラーモードの「白黒」と「印刷時に指定」は選択できないなどの禁則を回避する必要があります。

表 A.1　プリンターの FL 表

因子名	水準 1	水準 2	水準 3	水準 4
用紙サイズ	フォト用紙	普通紙 A4	普通紙 B4	はがき
カラーモード	カラー	白黒	印刷時に指定	
暗証番号	指定しない	指定する		
画像の種類	写真	イラスト	文章	
カメラの種類	スマホ	デジカメ	スキャン	

第 5 章

5.1

図 A.6 はカーオーディオの状態遷移図です。[＜][＞] ボタンは、[曲送り] というイベントにまとめています。この図を状態遷移表にすると、表 A.2 のとおりになります。

表 A.2 をよく見ると、リピート再生状態で曲送りボタンを押したときに N/A すなわち遷移しないことになっています。たしかに問題文には書いていないのですが、通常のカーオーディオでは入力した結果をできるだけ反映するような動作をしますので、リピート再生状態で［＞］を押したら次の曲に移り状態は通常再生状態に遷移する可能性があります。

仕様書またはテストで要確認事項です。

関係行列と、1 スイッチカバレッジ表は状態遷移表から表 A.3、表 A.4 のように機械的に変換できます。

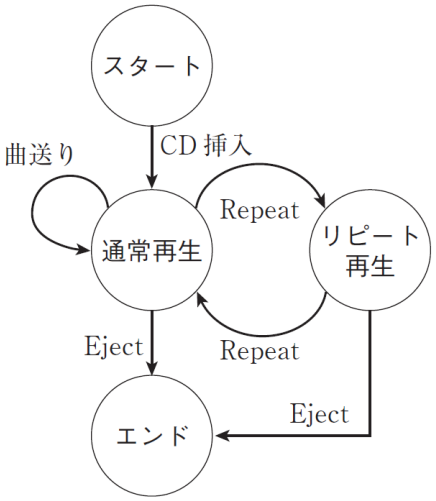


図 A.6　カーオーディオの状態遷移図

表 A.2　カーオーディオの状態遷移表

	スタート	通常再生	リピート再生	エンド
CD 挿入	通常再生	N/A	N/A	N/A
曲送り	N/A	通常再生	N/A	N/A
Repeat	N/A	リピート再生	通常再生	N/A
Eject	N/A	エンド	エンド	N/A

表 A.3　カーオーディオの関係行列

遷移先の状態 / 遷移元の状態	スタート	通常再生	リピート再生	エンド
スタート		CD 挿入		
通常再生		曲送り	Repeat	Eject
リピート再生		Repeat		Eject
エンド				

表 A.4　カーオーディオの 1 スイッチカバレッジ表

遷移先の状態 / 遷移元の状態	スタート	通常再生	リピート再生	エンド
スタート		CD 挿入×曲送り	CD 挿入 ×Repeat	CD 挿入×Eject
通常再生		曲送り×曲送り +Repeat×Repeat	曲送り ×Repeat	曲送り ×Eject+Repeat× Eject
リピート再生		Repeat×曲送り	Repeat× Repeat	Repeat×Eject
エンド				

5.2

図 A.7 の状態遷移図を見て違和感をもたれる方も多いのではないかと思います。ソフトウェアの開発を行うための仕様書にはこのような状態遷移図ではなく「小人チケットが購入できる状態」、「大人チケットが購入できる状態」といった抽象的な状態となっている場合のほうが多いと思います。

プログラムを書く場合には、そのような状態遷移図のほうが良いのですが、そこから状態遷移表を作成しても今度はテストデータを考える必要があります。

ソフトウェアテスト用の状態遷移図は一見、泥臭い図になりますが、あり得る状態（内部変数の組合せ）を並べ、そこにイベントを書き込むようにしてください。そうするとテストする範囲が明確になります。

なお、この方法だと図 A.7 のとおり有限の状態しか表せませんので 250 円や 350 円、400 円などがテストされません。状態遷移としてどこまでの状態をテストするかを考えてそこまでの図を描けばよいのです。今回であれば大人 2 人と子供 2 人の家族が動物園に来園することを考えると、500 円程度までは状態遷移図を描いたほうがよいでしょう。

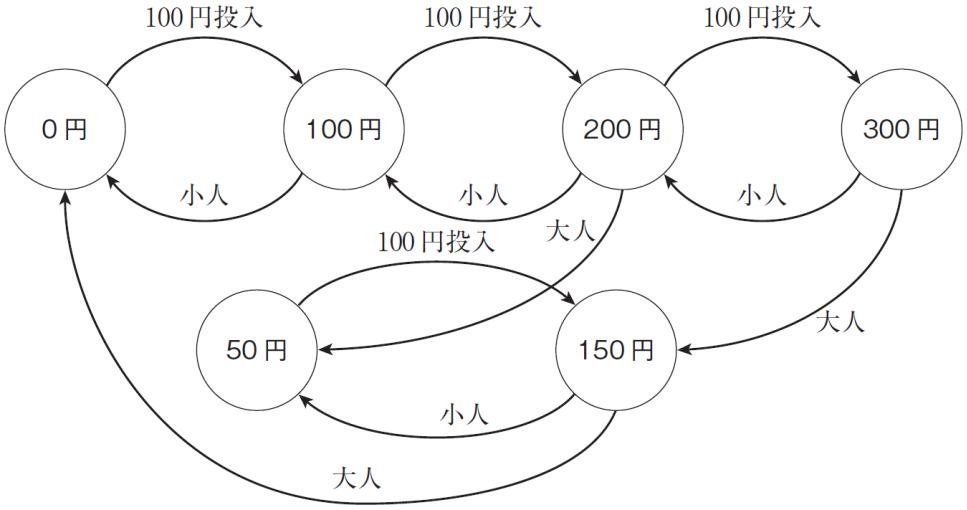


図 A.7　動物園の入園料の状態遷移図

第 6 章

6.1

入出国手続の目的は、人が異なる国を行き来するときに問題が発生しないようにすることです。問題とは、主に防犯・防疫・経済保護の 3 点です。

ソフトウェアテストでいえば、防犯はセキュリティテスト、防疫は構成テストもしくは共存性テスト、経済保護（滞在期間の確認や外国人登録）は導入準備テストのようなものと考えられるでしょう。

いずれにしても、開発側からお客様へ商品をお渡しするときに顧客先で問題が発生しないことを顧客側(相手国側の利益)から見て考える必要があります。

6.2

シナリオテストをつくるときに大切なことは次の 3 点でした。

① ユーザーシーン（When、Where、Who）を明確にする。

② 固有名詞や定数値を使う。

③ 変わった行動や失敗を犯すシナリオをつくり、そこから回復させながらユーザーのしたいことを継続させる。

それでは、上記に気をつけながら、さっそく書いてみましょう。

水元さんの家は、夫婦（剛司、優理）と、小学 1 年生になる女の子（華音）の 3 人家族です。朝 8 時に夫の剛司さんと子供が出勤や通学に行った後、妻の優理さんは洗濯をしようと思いつきました。

まず、お風呂場の脱衣場に置いてある洗濯機に洗い物を入れて、電源［入］ボタンを押しました。

「そういえば昨日、華音は自転車で転んで泥んこになって帰ってきたなぁ」と、服が泥で汚れていることを思い出し［ゴシゴシ］コースを選び［スタート／一時停止］ボタンを押しました。

洗濯機が洗い物の重さから、水量 57 リットルと洗濯洗剤カップ 2／3 を要求してきたので、ふたを開き洗剤ポケットに洗剤を入れました。少し洗剤を多く入れてしまったので、水量を 60 リットルに変更するとともに近くにあったタオルを追加投入。「まっ、これでトントンだわ」とつぶやき内ぶたを閉めると運転スタートです。

洗濯が終わるまで、テレビでも見るかと思ってリビングに戻ったところ、夫の剛司さんが脱ぎ散らかした靴下を発見！　「しょうがないなぁ」と思いながらそれを手に取り洗濯機に戻り［スタート／一時停止］ボタンを押して、ストップし、靴下を追加し再始動しました。