自動販売機のテスト設計における目指したことと工夫点

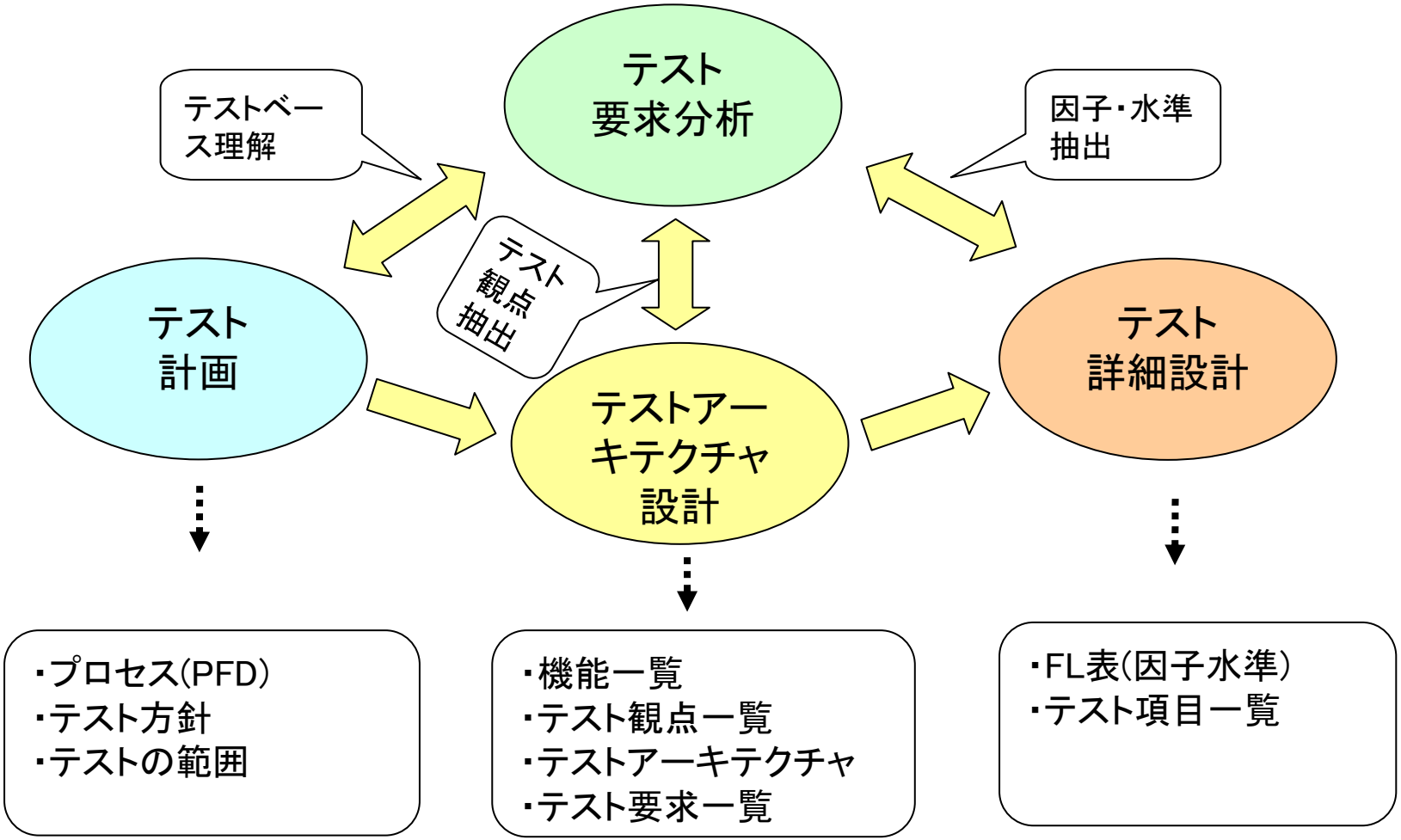
MKE98

AGENDA

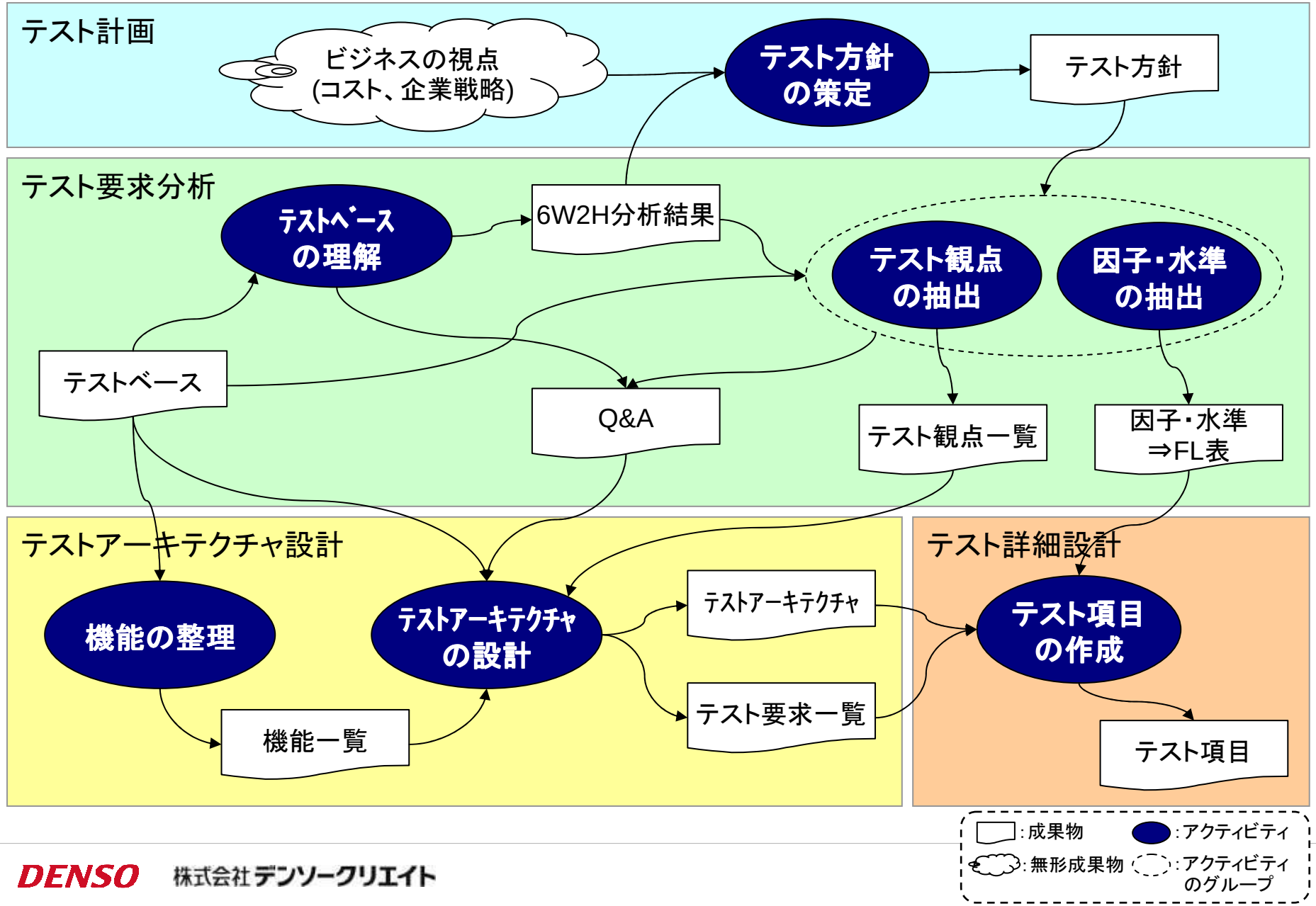
- テスト工程のプロセス
- 目指したこと
- テスト観点の漏れを防ぐための工夫(テスト要求分析)
- テストを効率的に進めるための工夫(テストアーキテクチャ設計)
- テスト詳細設計の工夫
- まとめ

テスト工程のプロセス(全体)

テスト計画、テストアーキテクチャ設計、テスト詳細設計の順でテストを進める。
各工程でテスト要求分析を実施する。(分析の粒度は少しずつ詳細になる。)



テストエ程のプロセス(詳細)



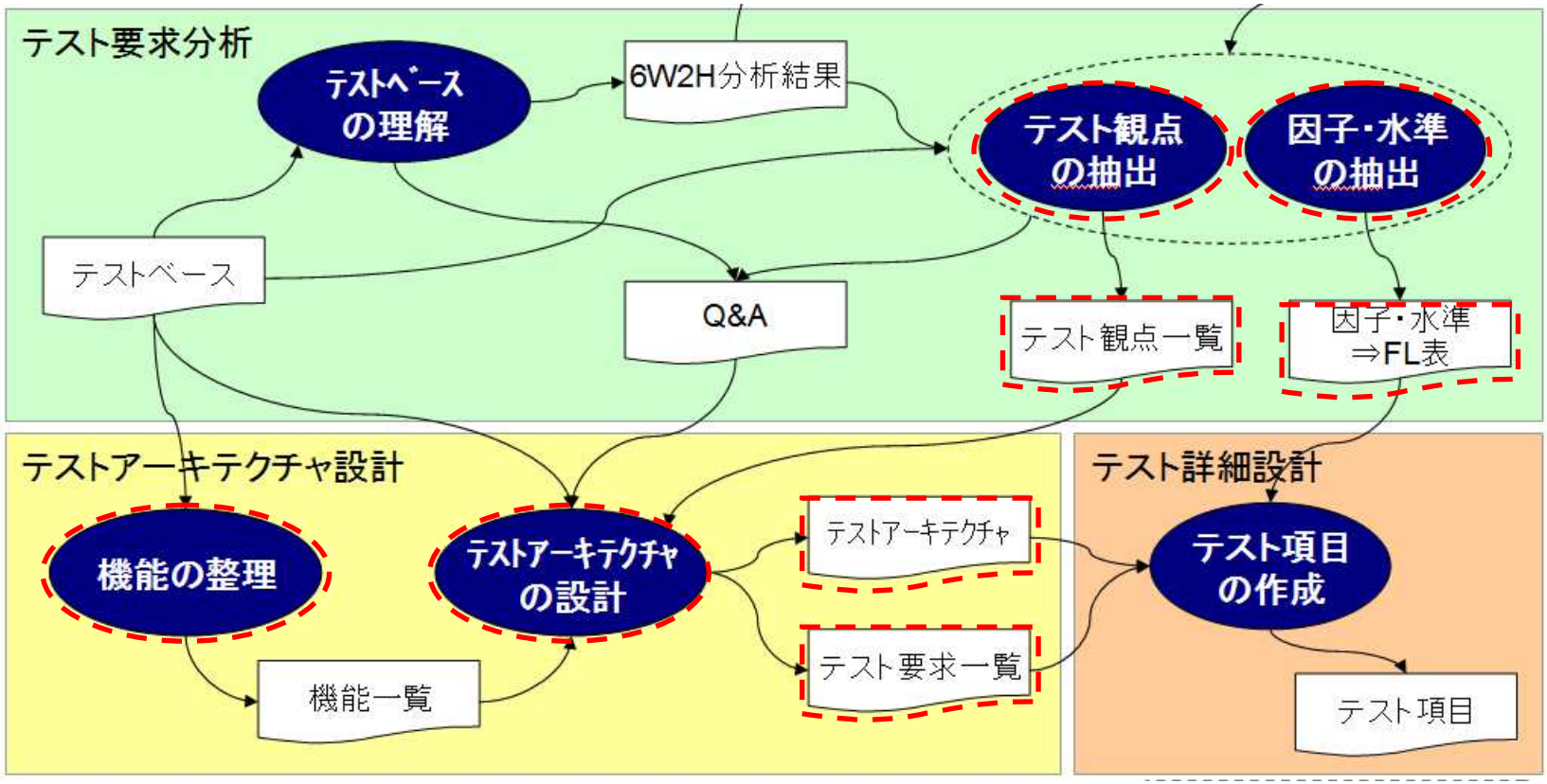
目指したこと

テストの実施漏れを
無くしたい

テスト要求分析の工程で、
できるだけ多くのテスト観点を抽出

テストを効率的に
進めたい

テストアーキテクチャ設計工程で、
テストを効率的に進めるためテストを設計



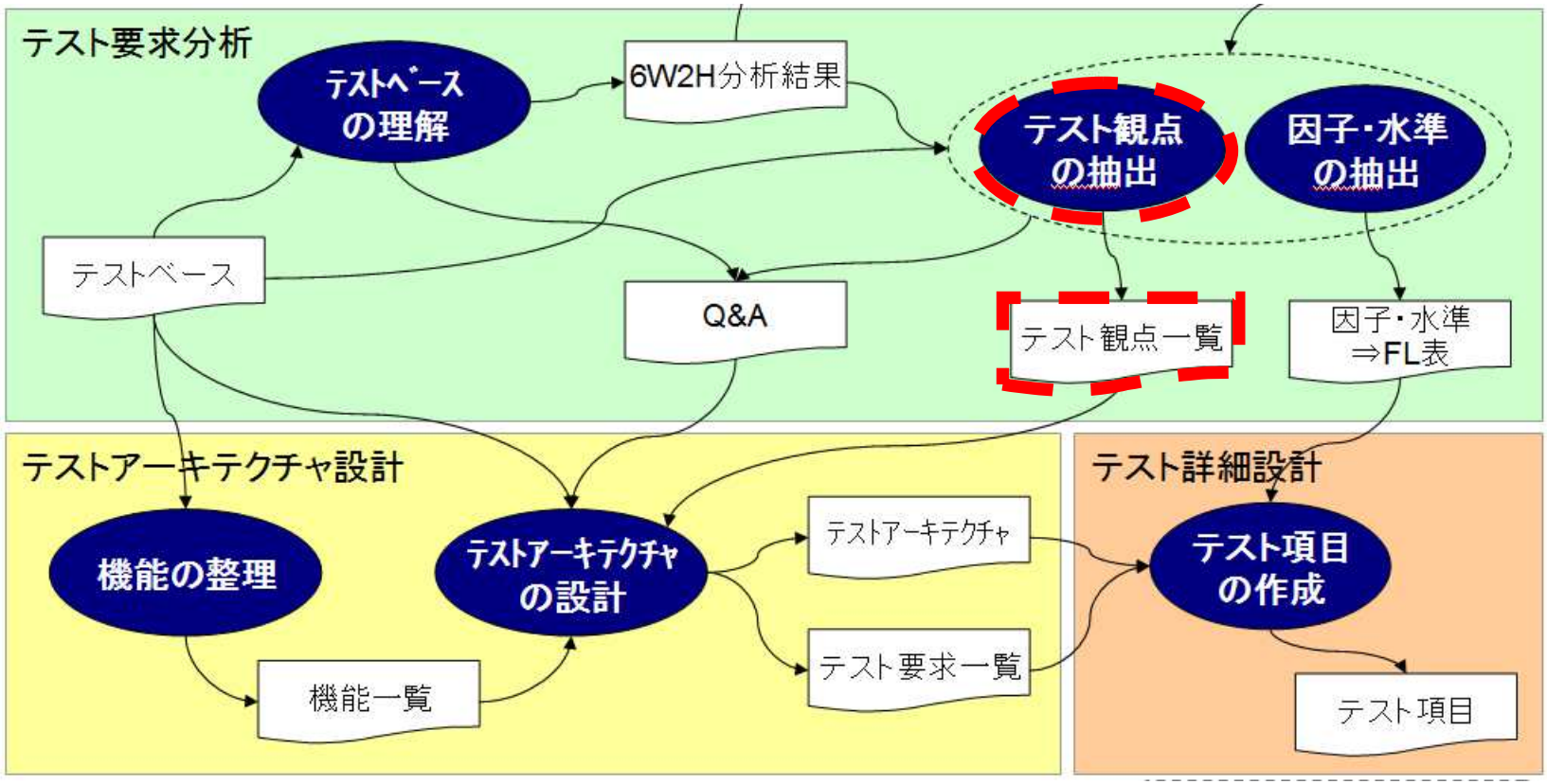
目指したこと

テストの実施漏れを
無くしたい

テスト要求分析の工程で、
できるだけ多くのテスト観点を抽出

テストを効率的に
進めたい

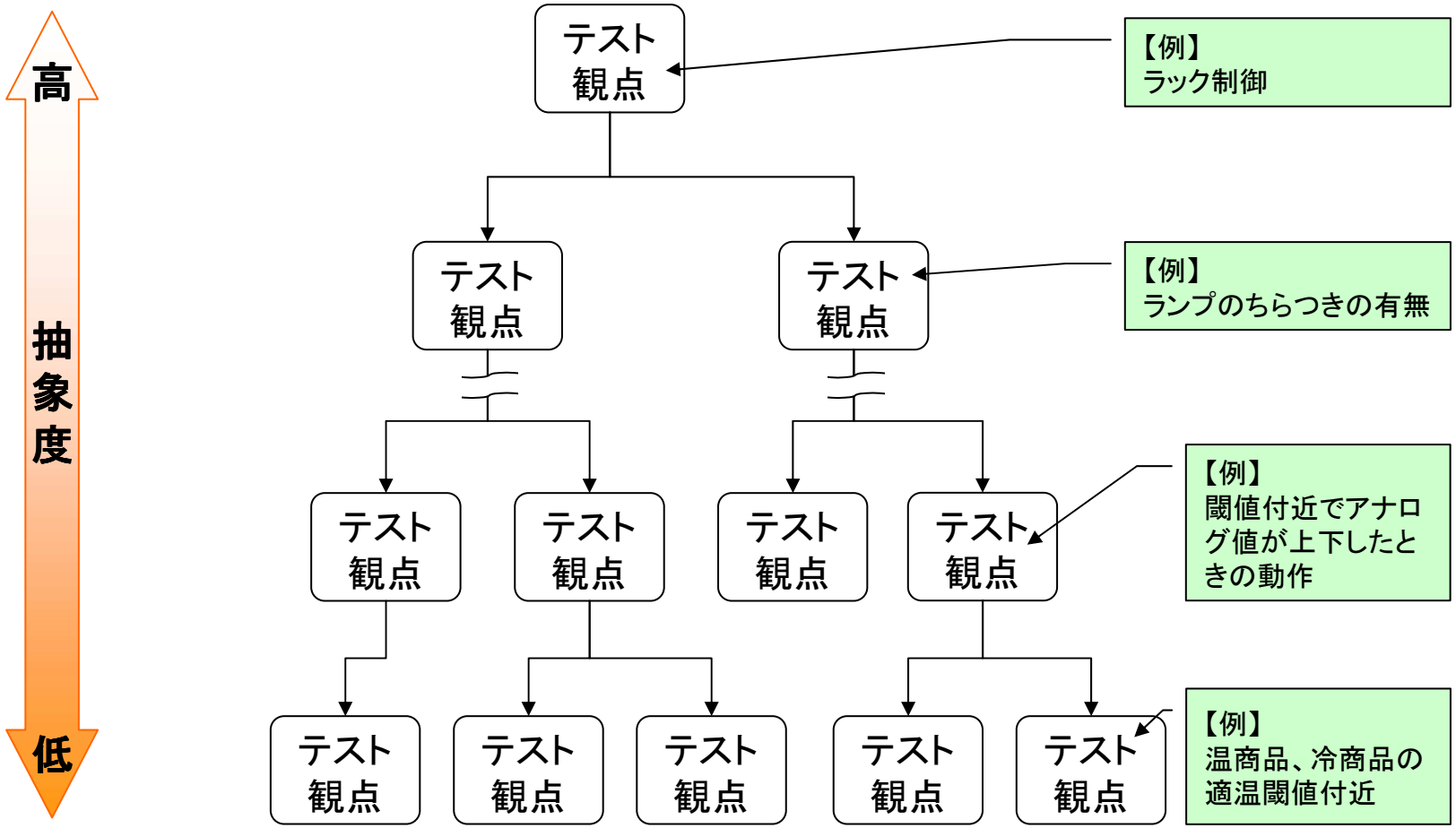
テストアーキテクチャ設計工程で、
テストを効率的に進めるためテストを設計



テスト観点の漏れを防ぐための工夫　(テスト観点の定義)

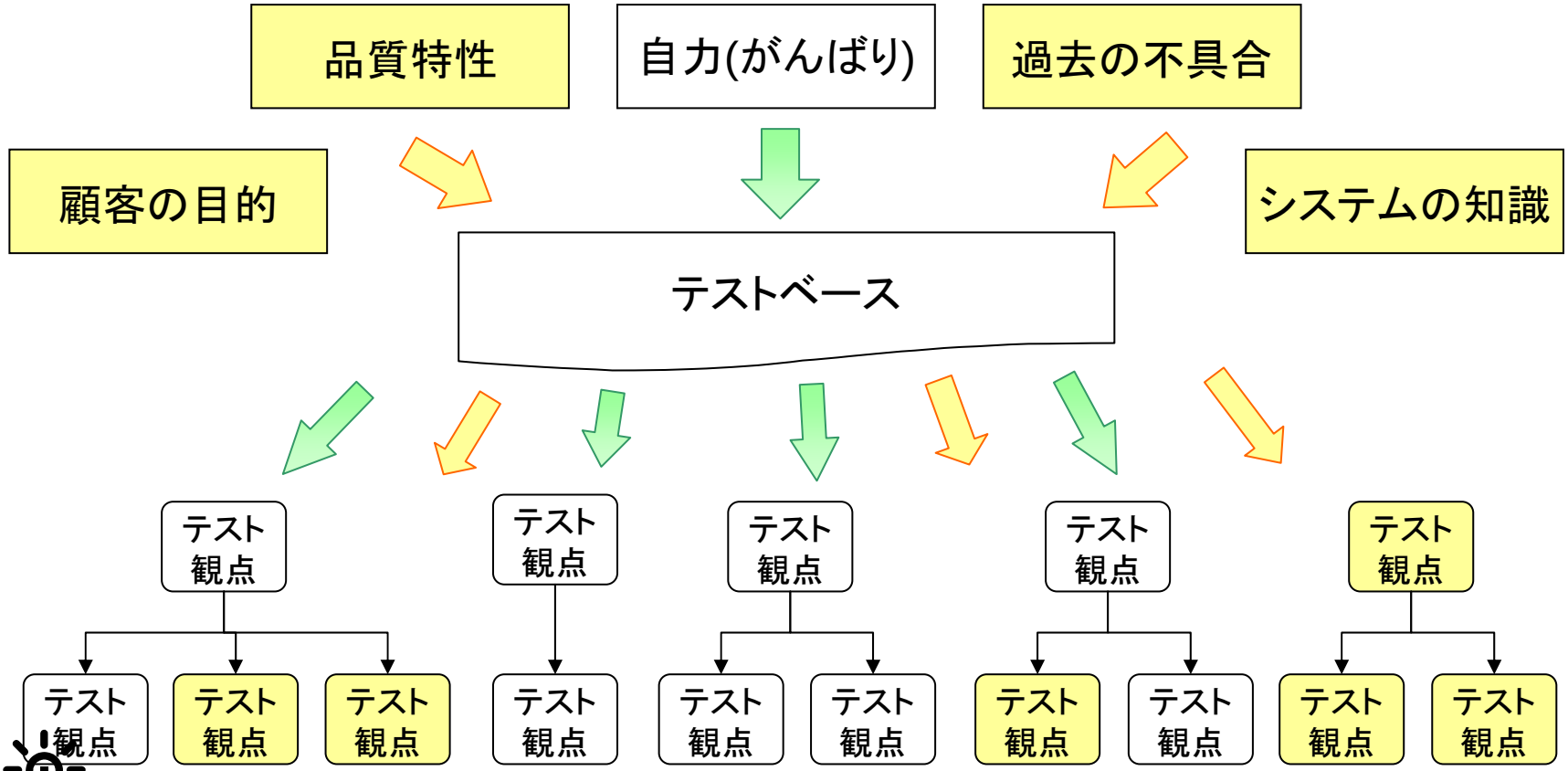
テスト観点

- テスト対象のテストすべき側面や、テスト対象が達成すべき性質。
- テスト観点は、以下のように階層構造を持つ。
- 基本的には、テスト観点を具体化していくとテストケースになる。



テスト観点の漏れを防ぐための工夫

自力(がんばり)では気づけるテスト観点に限界がある

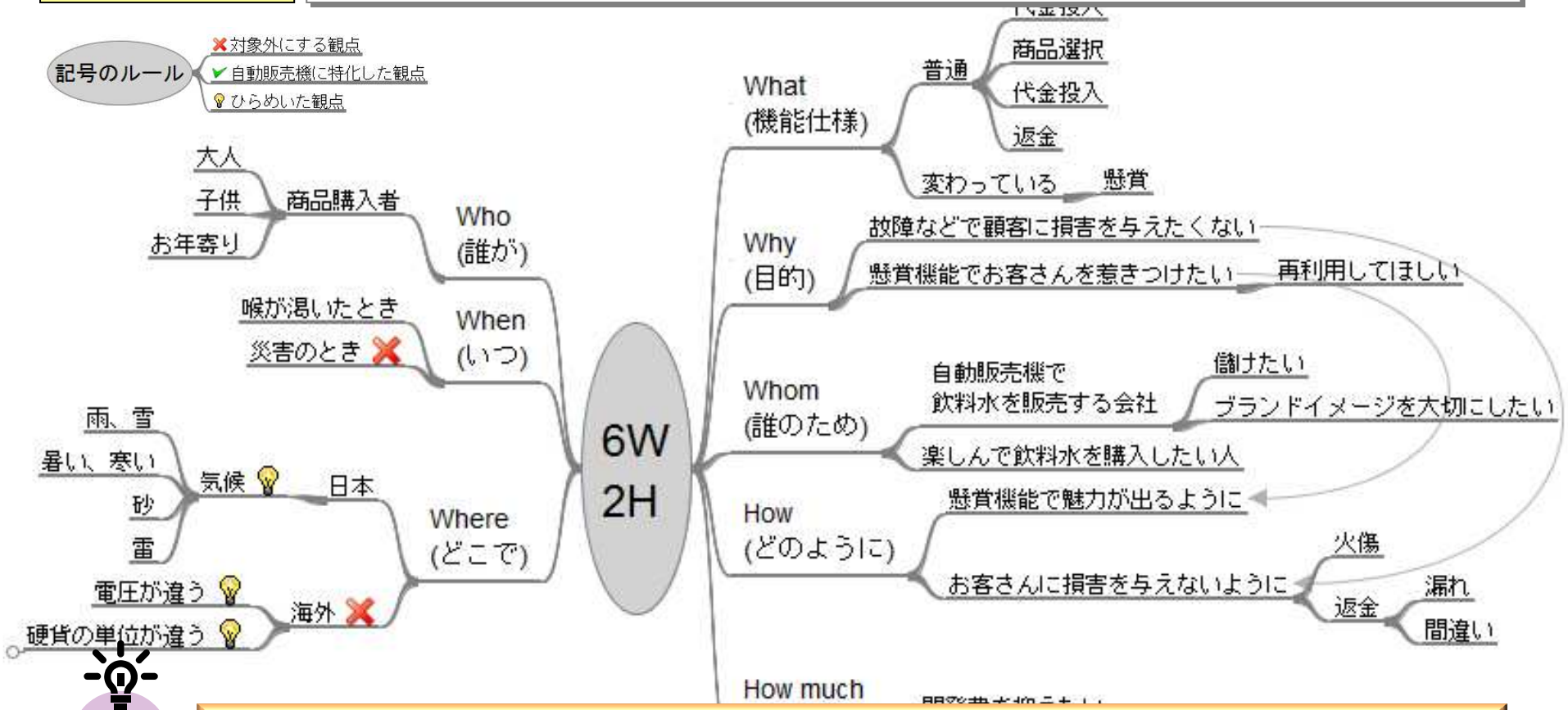


様々な拠り所を活用して、自力では気づかない視点を補足

「顧客の目的」を拠り所にテスト観点を抽出

顧客の目的

- 6W2H分析で顧客の目的を捉えて、それをテスト観点として抽出
- Why(目的)、How(どのように)を主にテスト観点として抽出

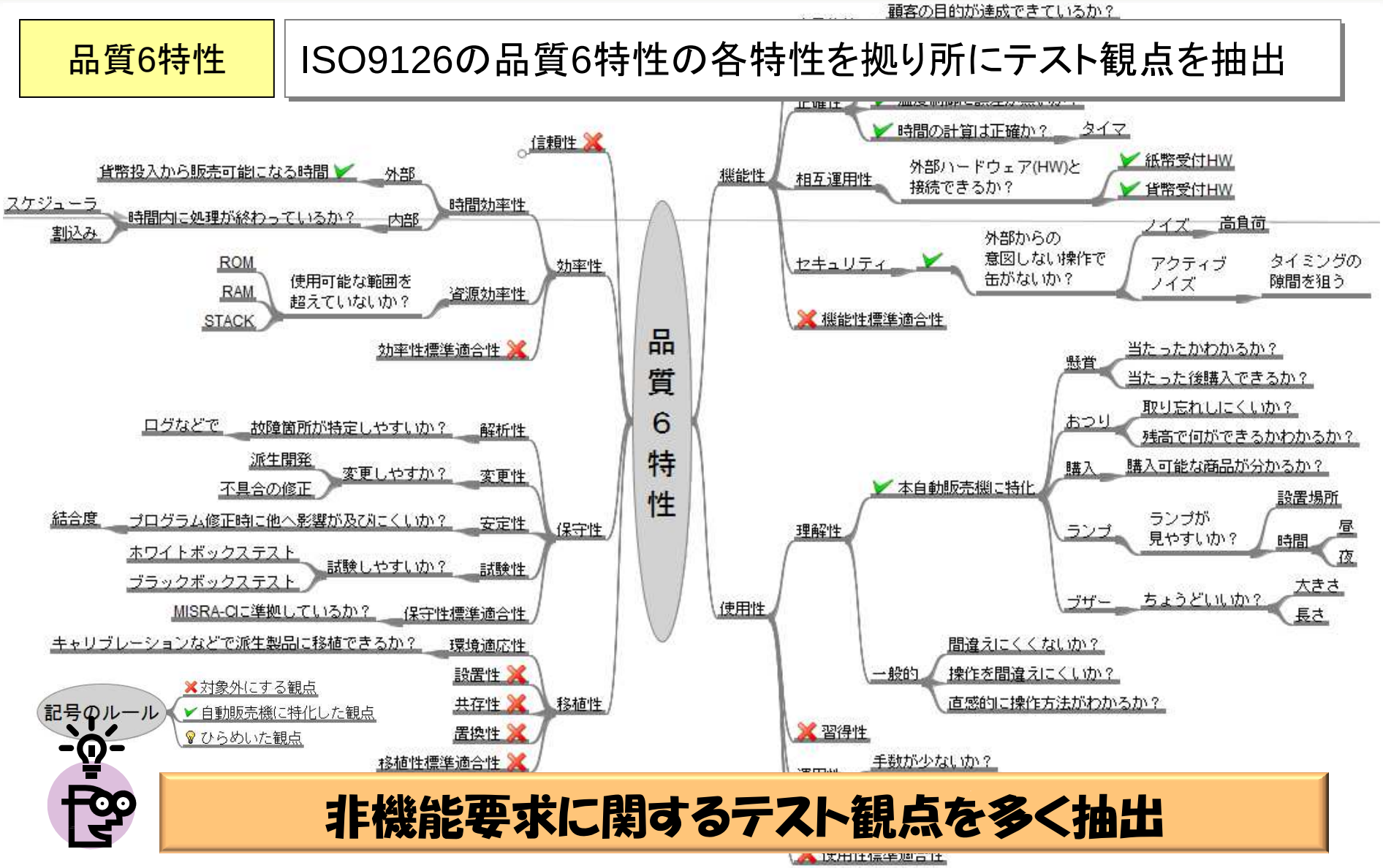


顧客の目的を満たすためのテスト観点を抽出

「品質特性」を拠り所にテスト観点を抽出

品質6特性

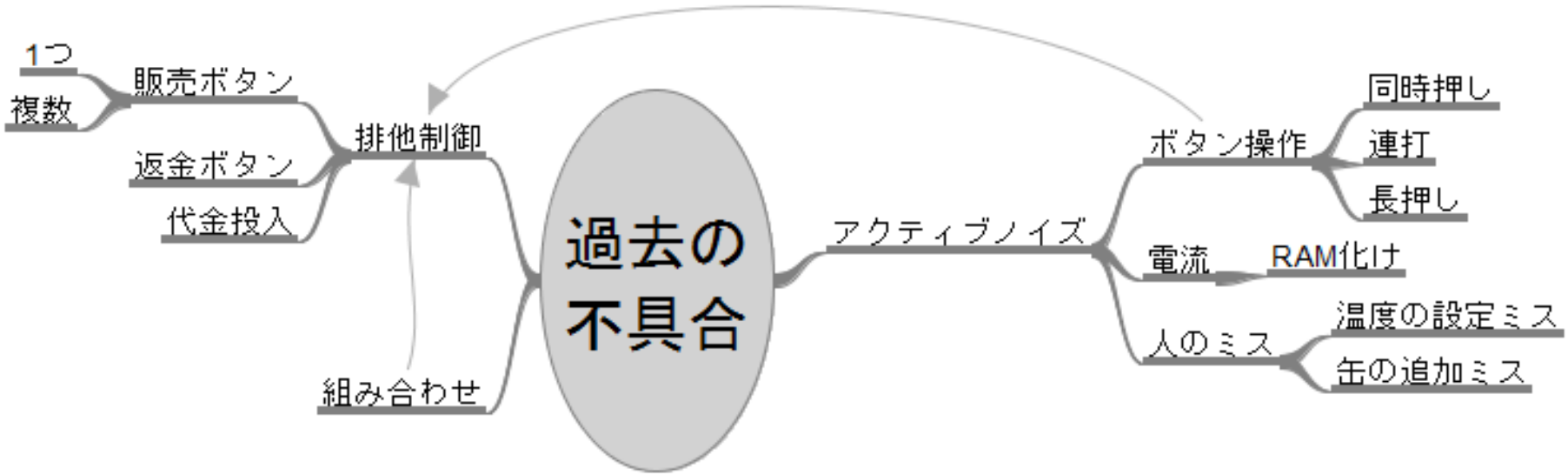
ISO9126の品質6特性の各特性を拠り所にテスト観点を抽出

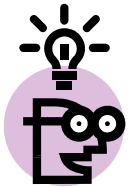


非機能要求に関するテスト観点を多く抽出

「過去の不具合」を拠り所にテスト観点を抽出

過去の不具合	自動販売機の過去の市場不具合を調査し、調査結果を拠り所としてテスト観点を抽出



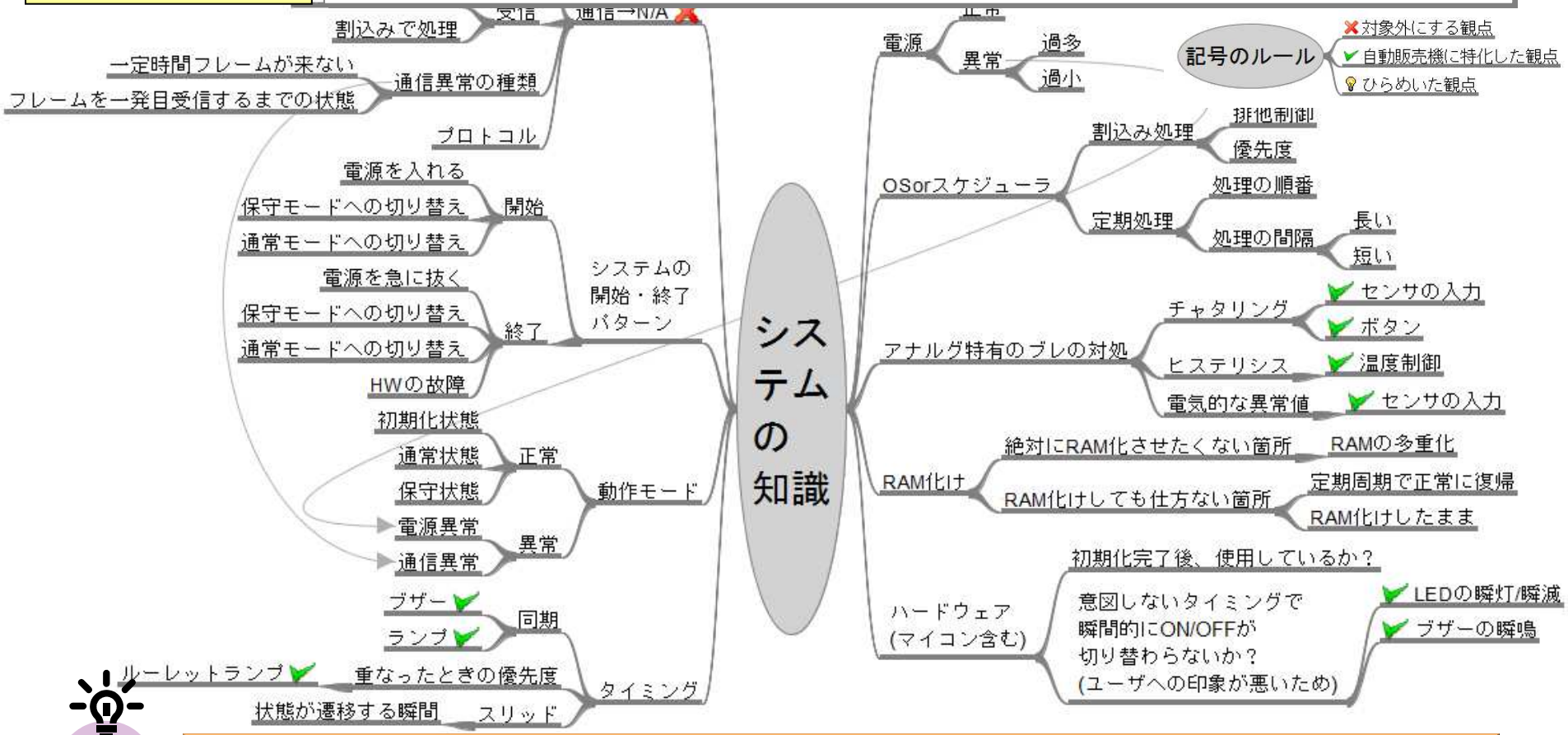


アクティブノイズに関するテスト観点を多く抽出

「システムの知識」を拠り所にテスト観点を抽出

システムの知識

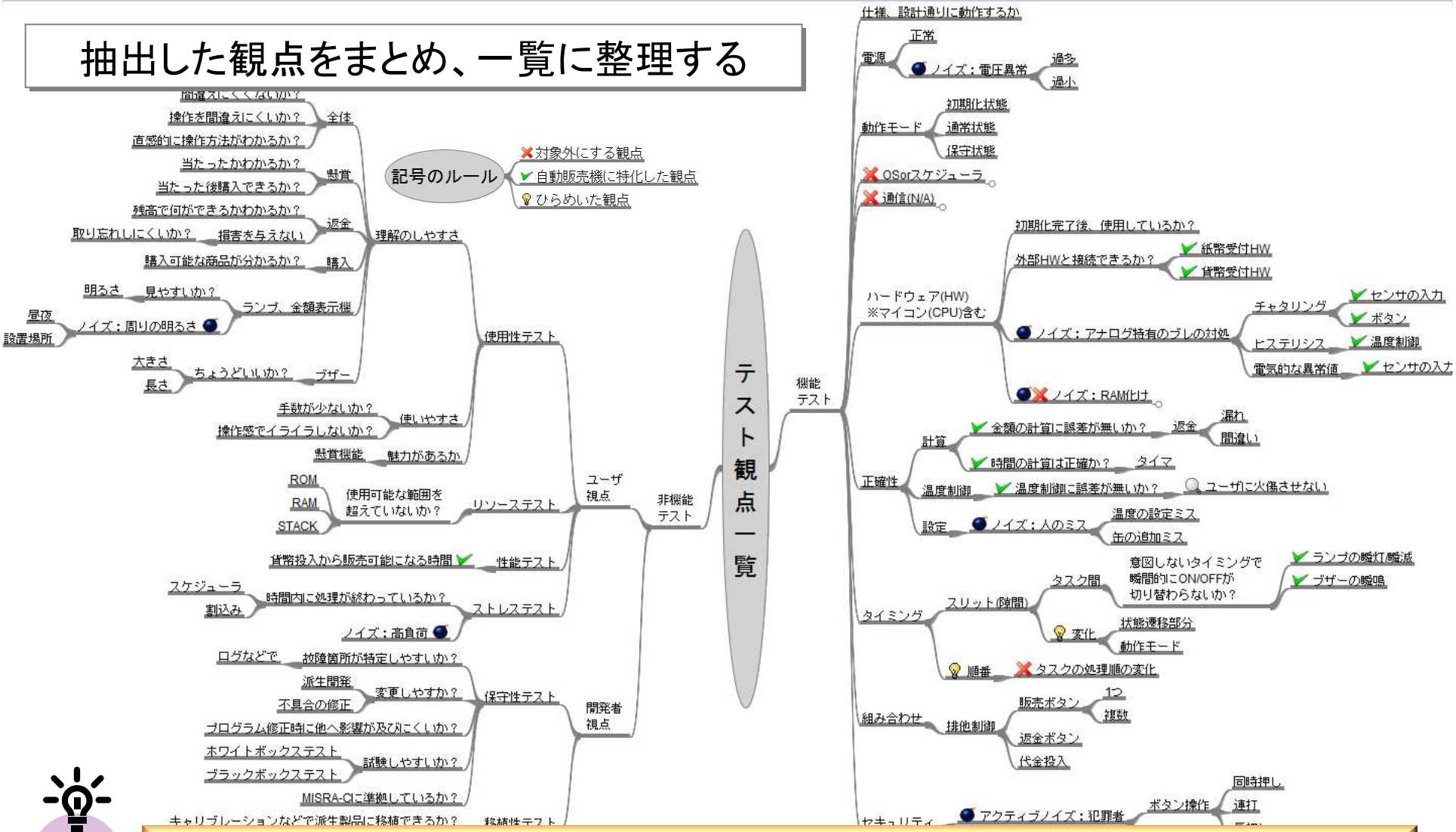
類似システムとして自動車ECUの知識を活用して、テスト観点を抽出

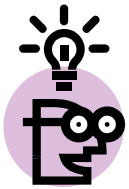


HW周り、ノイズ、タイミングなどのテスト観点を抽出

テスト観点一覧の作成

抽出した観点をまとめ、一覧に整理する





拠り所の活用で、自力では気づけないテスト観点に気づけた

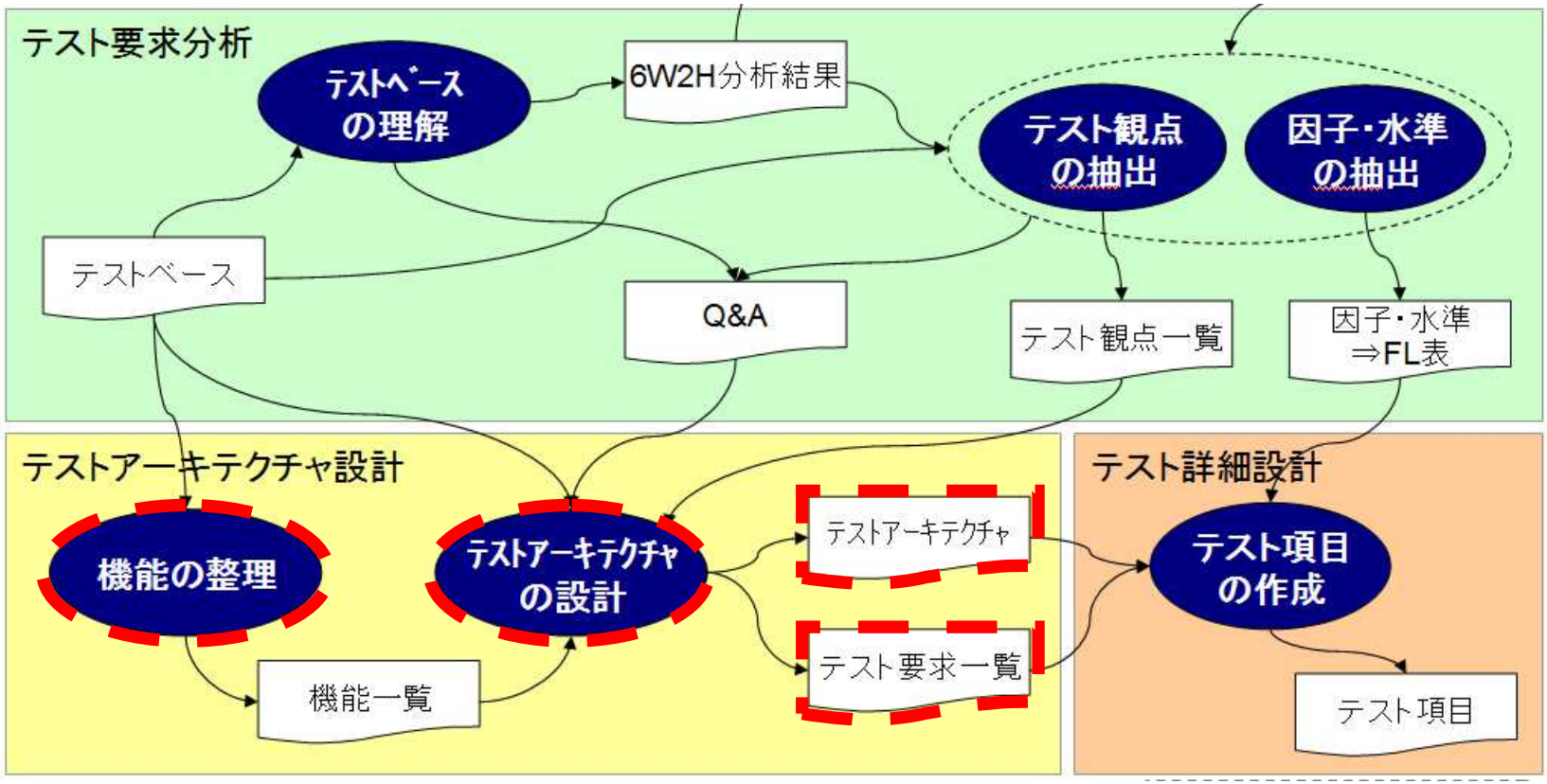
目指したこと

テストの実施漏れを
無くしたい

⇒ テスト要求分析の工程で、
できるだけ多くのテスト観点を抽出

テストを効率的に
進めたい

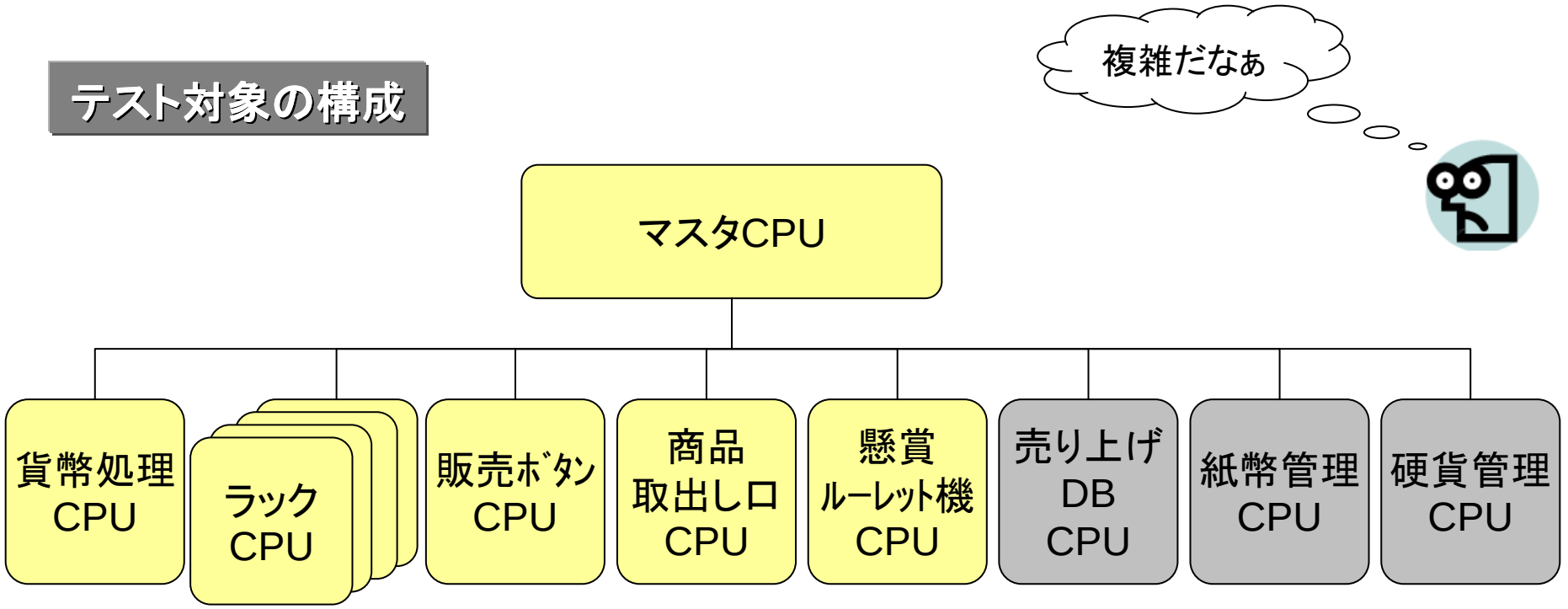
⇒ テストアーキテクチャ設計工程で、
テストを効率的に進めるためテストを設計



テストを効率的に進めるための工夫　(問題点①)

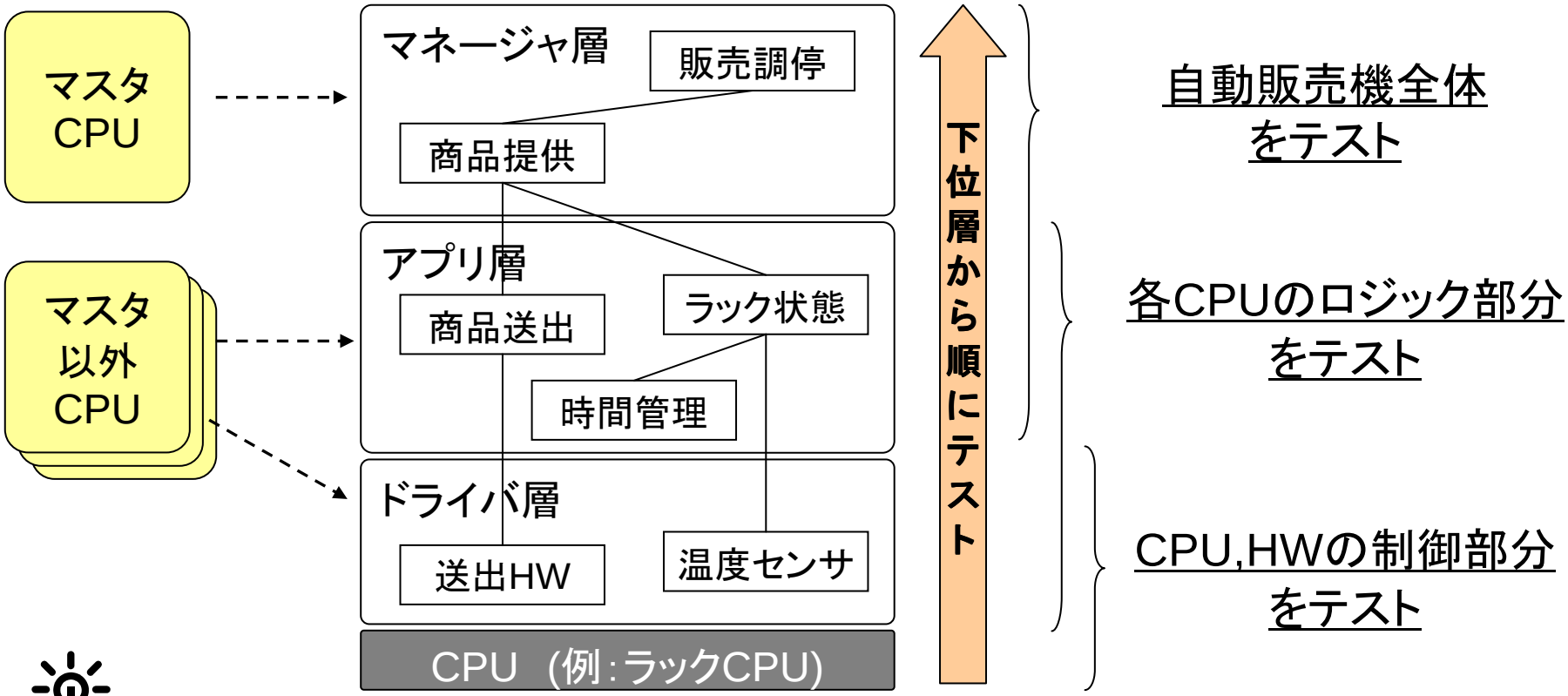
全CPUを統合した状態でいきなりテストすると、
不具合が発生したときに、不具合分析の範囲が広くて時間がかかる

テスト対象の構成



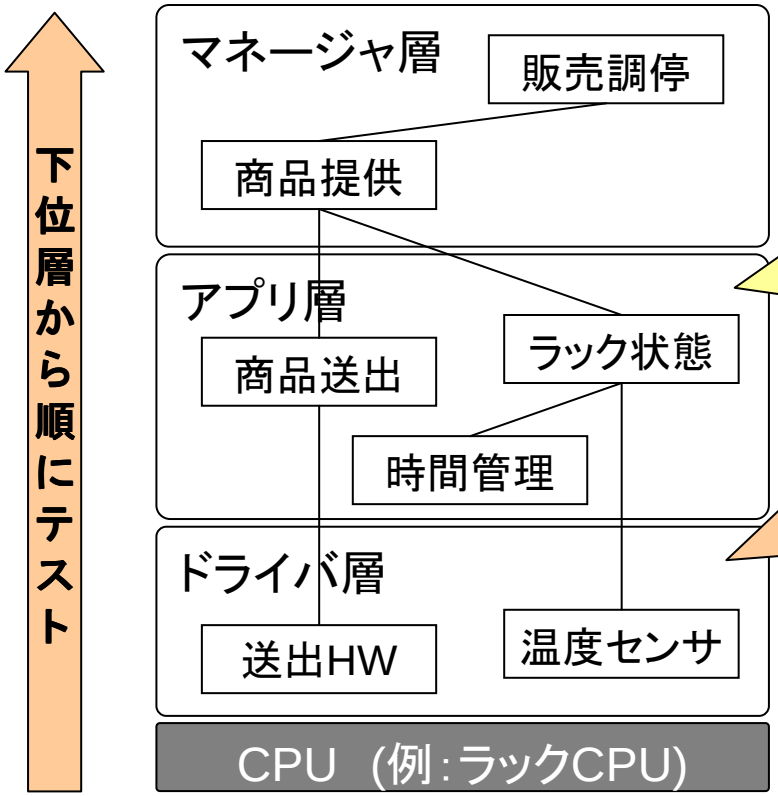
テストを効率的に進めるための工夫

- 各CPUが担う機能を「テストしやすい大きさ」「階層毎」に分割し、構造化
- ドライバ層、アプリ層、マネージャ層の順番でテストを実行



下位(ドライバ)層から順に保証範囲を少しずつ広げる

テストを効率的に進めるための工夫　(問題点②)



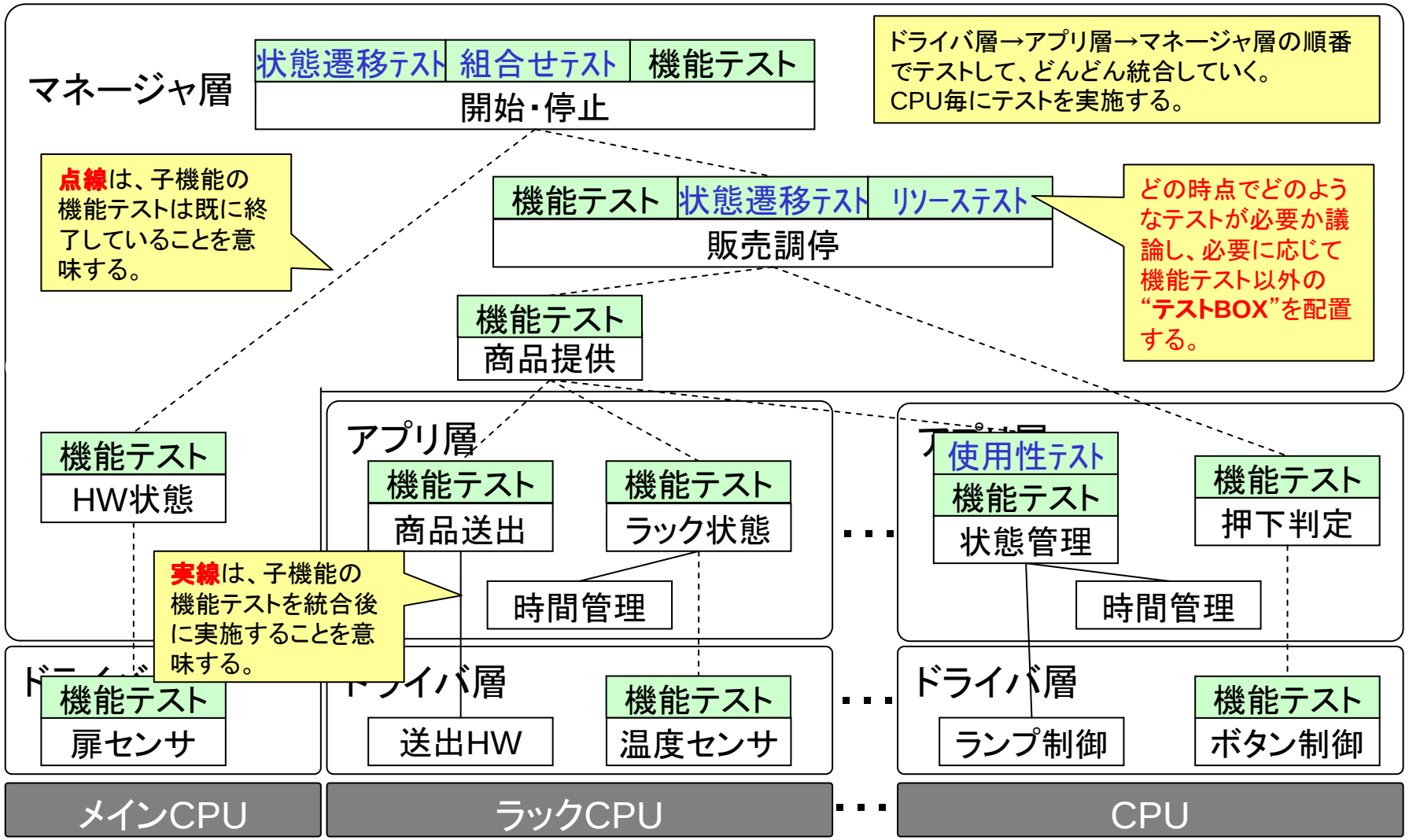
•出力する機能は特に難しい制御もしていないので、アプリ層の機能と統合してからテストしたい。
•結果的にドライバ層の機能確認が別のテストで確認できるから先に統合した方が効率的。

•何かを入力とする機能はチャタリング除去などテストを先にやっておきたいから単独で先にテストしたい。
•確認するパターンが複数ある場合は、単独で全パターンテストして、統合後のテストでは代表値だけ見たい。

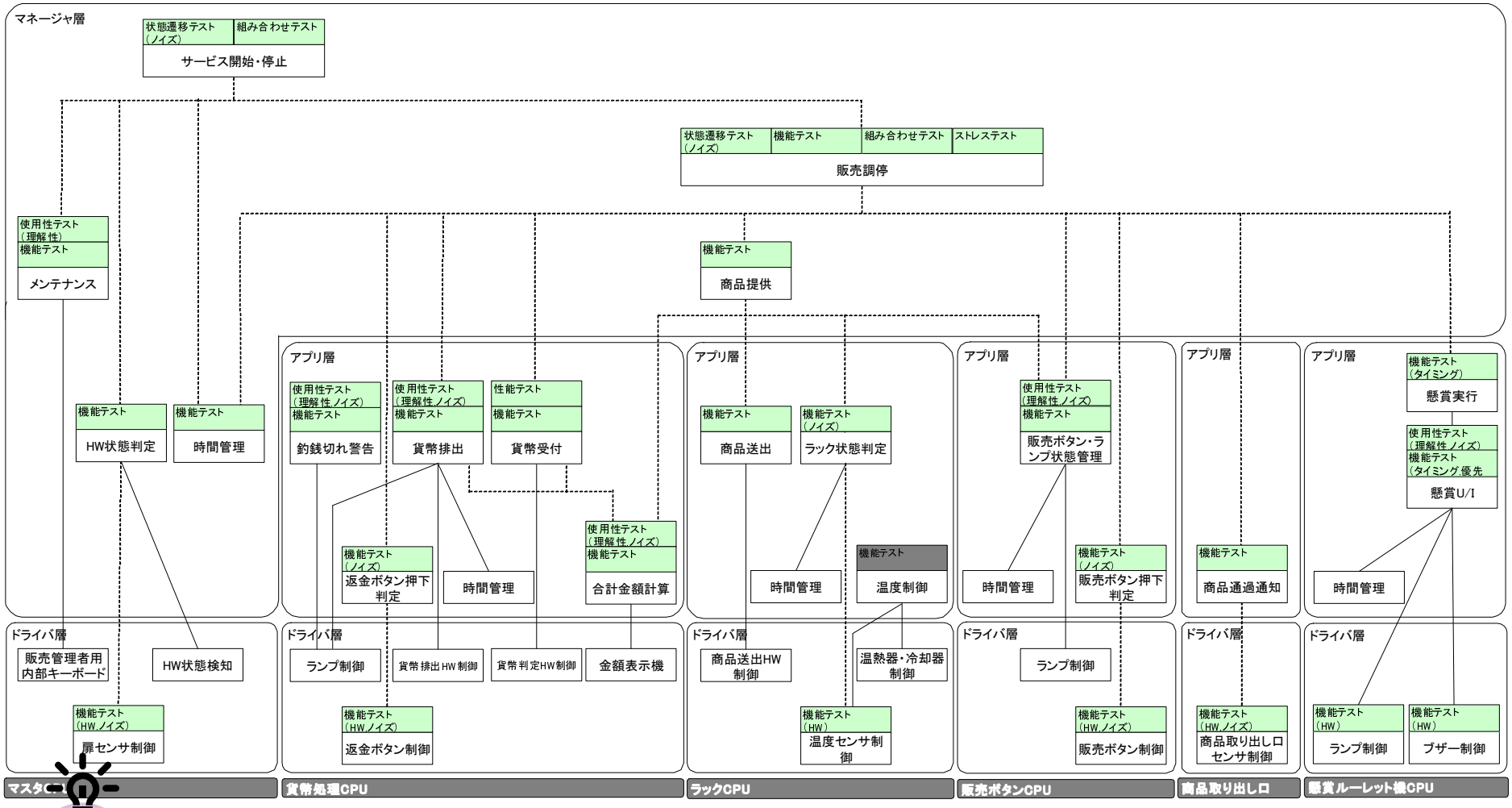
•状況により、単機能毎にテストしていくか、統合してからテストするか検討する必要がある。
•機能分割、階層化、構造化だけだと、統合していく過程でどのようなテストをするのかが見えにくい。

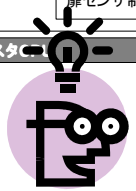
テストを効率的に進めるための工夫　(テストアーキテクチャー部抜粋)

テストタイプBOXと実線・点線で、統合の進め方とどんなテストをするか表現



テストを効率的に進めるための工夫 (テストアーキテクチャ)

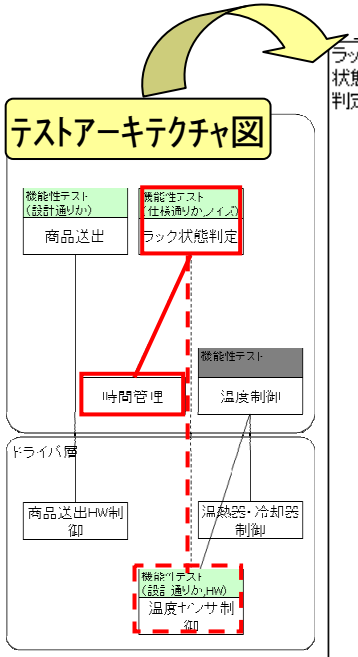




全体が見渡せるので、テストの検討が容易になった

テスト要求一覧の作成

- テストアーキテクチャ図をベースに、実際にテストすることを書き出し、(USDMに似せた)テスト要求一覧を作成する。
- 「テストベース(Q&A含む)、機能一覧、テスト観点一覧」を入力とする。



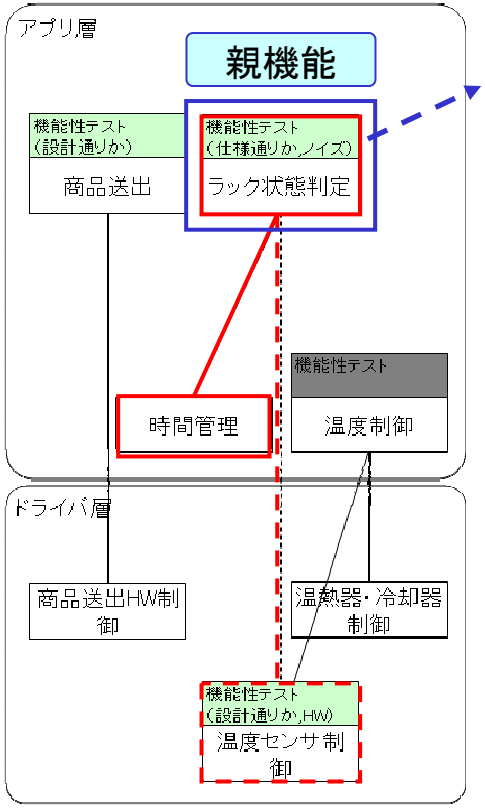
テストベース

機能一覧

テスト観点一覧

ラック状態判定	要求	ID	ラックの在庫数、温度制御が準備中か否か、温度制御の異常の判定を確認する	機能一覧ID	確認対象機能	トレーサビリティ(テストベース)	詳細	確認箇所	機能テスト
		理由	・ラックの在庫数、温度制御が準備中か否か、温度制御の異常の判定ができているか確認するため。 ・ノイズがこの判定にどのような影響を与えるか確認するため。	RA-2 RA-3	ラック状態判定－時間管理				
		説明	・外部からの確認が難しい場合は、RAMにて状態を確認すること。 ・「準備状態判定」機能ではデシジョンテーブルを作成し、以下の入力の組み合わせを確認する。 温度センサ検知温度、温度制御モード、適温範囲						
	在庫数管理								
	□□□	T_A_RA_B1_1	商品排出後、在庫数が1本減っていることを確認する。	RA-2-1	ラック状態判定	・ユースケース仕様書 2.5処理(メインフロー)	-	RAM	仕様通りか
	準備状態判定								
	□□□	T_A_RA_B1_2	「温度センサ検知温度」と「温度制御モード(冷却または加温)」、「適温範囲」に応じて、適温範囲内かが判定できているか確認する。 ※現状の仕様だと、適温範囲は以下のようになっている。 温商品：52℃以上 58℃以下 冷商品：1℃以上 6℃以下	RA-2-2	ラック状態判定	・ユースケース仕様書 1.5処理(メインフロー)	●	準備中ランプ	仕様通りか
	□□□	T_A_RA_B1_3	閾値付近の温度に設定した際の振る舞いを確認する。 ※適温範囲外になった際に点灯する「準備中ランプ」に着目し、ランプがチカチカ点灯/消灯していないか確認する。	RA-2-2	ラック状態判定	Q&A No52	-	準備中ランプ	ノイズ：アナログ特有のブレ
	温度制御異常判定								
	□□□	T_A_RA_B1_4	稼動後もしくは「温度センサ検知温度」が適温範囲から外れて18時間以上経過しても、適温範囲内にならない場合、異常と判断できていることを判定する。 ※「時間管理」の機能が18時間を計測できているかもあわせて確認する。 確認はストップウォッチですること。	RA-2-3 RA-3	ラック状態判定－時間管理	・ユースケース仕様書 5.5処理(メインフロー)	-	温度センサ RAM	仕様通りか
	□□□	T_A_RA_B1_5	適温範囲から外れて18時間以上経過する中で、一瞬センサから適温範囲内になってしまう温度を取得してしまったとき、どのような振る舞いをするか確認する。 ※センサ値を平均化せずに生値でそのまま使用していると、一瞬おかしなデータをセンサが拾ってしまっただけで18時間のタイマがリセットされてしまう。そのようなことが無いか確認する。	RA-2-3	ラック状態判定	Q&A No12,55	-	温度センサ RAM	ノイズ：電気的なごみ

テスト要求一覧の作成（各部位の説明1/2）



ラック状態判定	要求	ID	ラックの在庫数、温度制御が準備中か否か、温度制御の異常の判定を確認する
		理由	・ラックの在庫数、温度制御が準備中か否か、温度制御の異常の判定ができているか確認するため。 ・ノイズがこの判定にどのような影響を与えるか確認するため。
		説明	外部からの確認が難しい場合は、RAMにて状態を確認すること。
ラック状態判定			
□□□	T_RA_A_B1_1		商品排出後、在庫数が1本減っていることを確認する。
準備状態判定			
□□□	T_RA_A_B1_2		「温度センサ検知温度」と「温度制御モード(冷却または加温)」、「適温範囲」に応じて、適温範囲内かが判定できているか確認する。 ※現状の仕様だと、適温範囲は以下のようになっている。 温商品: 52℃以上 58℃以下 冷商品: 1℃以上 6℃以下
□□□	T_RA_A_B1_3		閾値付近の温度に設定した際の振る舞いを確認する。 ※適温範囲外になった際に点灯する「準備中ランプ」に着目し、ランプがチカチカ点灯/消灯していないか確認する。
温度制御異常判定			
□□□	T_RA_A_B1_4		稼動後もしくは「温度センサ検知温度」が適温範囲から外れて1時間以上経過しても、適温範囲内にならない場合、異常と判断できていることを判定する。 ※「時間管理」の機能が1時間を計測できているかもあわせて確認する。 確認はストップウォッチですること。
□□□	T_RA_A_B1_5		適温範囲から外れて1時間以上経過する中で、一瞬センサから適温範囲内になってしまう温度を取得してしまったとき、どのような振る舞いをするか確認する。 ※センサ値を平均化せずに生値でそのまま使用していると、一瞬おかしなデータをセンサが拾ってしまっただけで1時間のタイマがリセットされてしまう。そのようなことが無いか確認する。

テスト全体の内容
理由・説明

テストの内容

テストID

テスト要求一覧の作成（各部位の説明2/2)

■ラックCPU

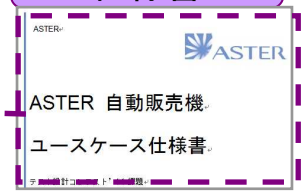
機能名	機能ID	機能
初期	RA-1	電源ON時に、ラックCPUを初期化する
	RA-1-1	電源ONのタイミングでCPU全体を初期化する
商品	RA-2	ラックから『商品』を送出する
	RA-2-1	ラックから『商品』を1つ送出する
ラック状態判定	RA-3	ラックの在庫数、準備状態、温度制御異常を判定する
在庫数管理	RA-3-1	「在庫数」を管理する
準備状態判定	RA-3-2	「温度センサ検知温度」と「温度制御モード」、「適温範囲の点灯/消灯」を判定する
温度制御異常判定	RA-3-3	『温熱器・冷却器』稼動後1時間以上経過しても、「温度を判定する
時間管理	RA-4	温度制御異常を判定するために、時間を計測する
時間計測	RA-4-1	温度制御タイムアウト時間(1時間)を計測する
温度制御	RA-5	適温範囲にするために温熱器・冷却器を制御する
温度制御	RA-5-1	「温度センサ検知温度」の変化に応じて、『温熱器・冷
商品送出HW制御	RA-6	商品送出HW(外部CPU)を操作する
商品送出HW制御	RA-6-1	商品送出HW(外部CPU)に商品の送出を依頼する
温度センサ制御	RA-7	温度センサ検知温度を通知する
温度センサ検知温度通知	RA-7-1	「温度センサ検知温度」を通知する
温熱器・冷却器制御	RA-8	温熱器と冷却器を操作する
温熱器制御	RA-8-1	温熱器を稼動/停止させる
冷却器制御	RA-8-2	冷却器を稼動/停止させる

機能一覧

機能一覧ID	確認対象機能	トレーサビリティ(テストベース)	詳細	確認箇所	機能テスト
RA-2 RA-3	ラック状態判定 －時間管理				
RA-2-1	ラック状態判定	ユースケース仕様書 2.5処理(メインフロー)	－	RAM	仕様通りか
RA-2-2	ラック状態判定	・ユースケース仕様書 1.5処理(メインフロー)	●	準備中ランプ	仕様通りか
RA-2-2	ラック状態判定	Q&A No52	－	準備中ランプ	ノイズ :アナログ特有のブレ
RA-2-3 RA-3	ラック状態判定 －時間管理	ユースケース仕様書 5.5処理(メインフロー)	－	温度センサ RAM	仕様通りか
RA-2-3	ラック状態判定	Q&A No12,55	－	温度センサ RAM	ノイズ :電気的なごみ

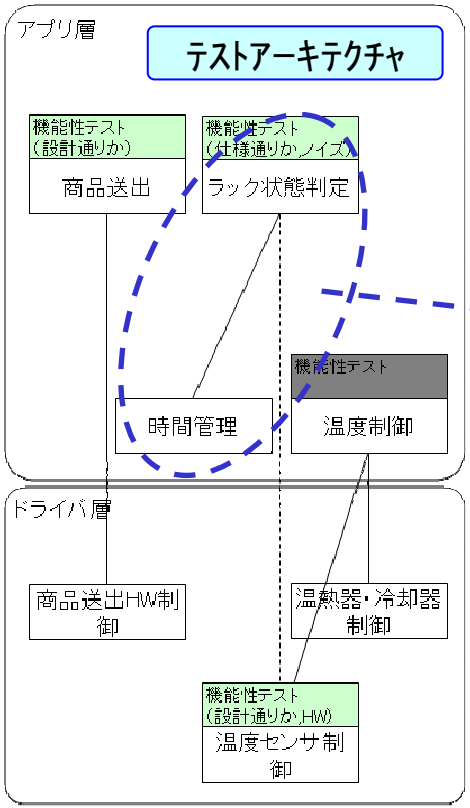
テストタイプ

テストベース仕様書

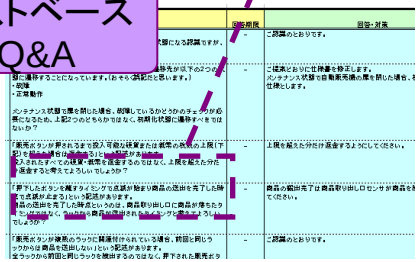


人が確認する物理的な点

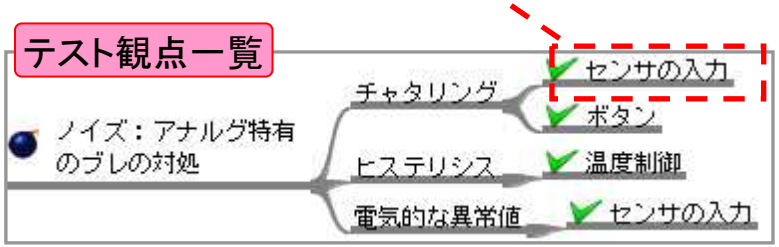
テストアーキテクチャ



テストベースQ&A

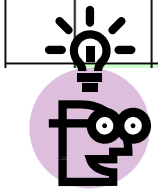


テスト観点一覧



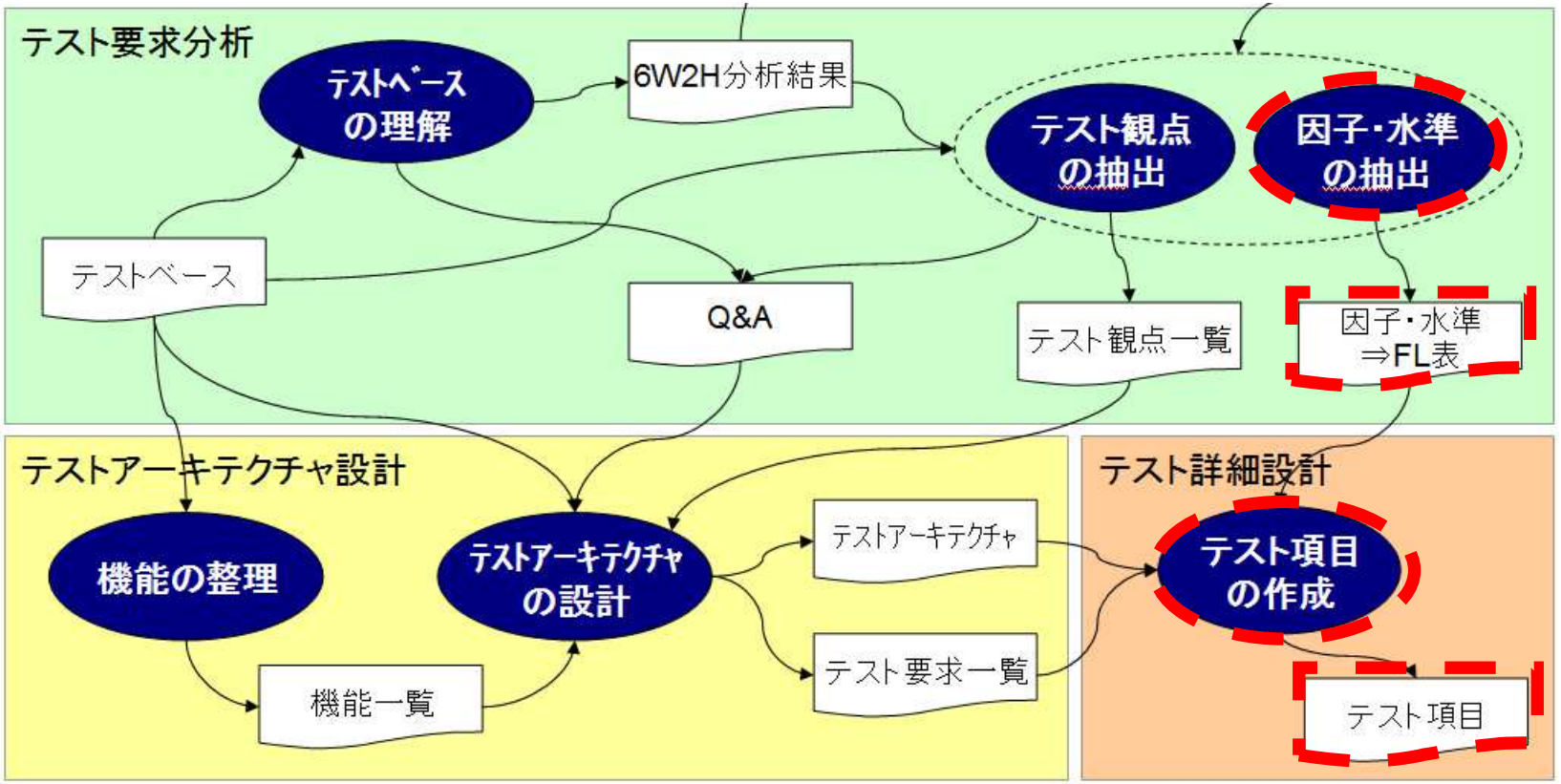
テスト要求一覧の作成

ラック状態判定	要求	ID	ラックの在庫数、温度制御が準備中か否か、温度制御の異常の判定を確認する	機能一覧ID	確認対象機能	トレーサビリティ(テストベース)	詳細	確認箇所	機能テスト
		理由	・ラックの在庫数、温度制御が準備中か否か、温度制御の異常の判定ができているか確認するため。 ・ノイズがこの判定にどのような影響を与えるか確認するため。	RA-2 RA-3	ラック状態判定 －時間管理				
		説明	・外部からの確認が難しい場合は、RAMにて状態を確認すること。 ・「準備状態判定」機能ではデシジョンテーブルを作成し、以下の入力の組み合わせを確認する。 温度センサ検知温度、温度制御モード、適温範囲						
	在庫数管理								
	□□□	T_A_RA_B1_1	商品排出後、在庫数が1本減っていることを確認する。	RA-2-1	ラック状態判定	・ユースケース仕様書 2.5処理(メインフロー)	-	RAM	仕様通りか
	準備状態判定								
	□□□	T_A_RA_B1_2	「温度センサ検知温度」と「温度制御モード(冷却または加温)」、「適温範囲」に応じて、適温範囲内かが判定できているか確認する。 ※現状の仕様だと、適温範囲は以下のようになっている。 温商品: 52℃以上 58℃以下 冷商品: 1℃以上 6℃以下	RA-2-2	ラック状態判定	・ユースケース仕様書 1.5処理(メインフロー)	●	準備中ランプ	仕様通りか
	□□□	T_A_RA_B1_3	閾値付近の温度に設定した際の振る舞いを確認する。 ※適温範囲外になった際に点灯する「準備中ランプ」に着目し、ランプがチカチカ点灯/消灯していないか確認する。	RA-2-2	ラック状態判定	Q&A No52	-	準備中ランプ	ノイズ :アナログ特有のブレ
	温度制御異常判定								
	□□□	T_A_RA_B1_4	稼動後もしくは「温度センサ検知温度」が適温範囲から外れて1時間以上経過しても、適温範囲内にならない場合、異常と判断できていることを判定する。 ※「時間管理」の機能が1時間を計測できているかもあわせて確認する。 確認はストップウォッチですること。	RA-2-3 RA-3	ラック状態判定 －時間管理	・ユースケース仕様書 5.5処理(メインフロー)	-	温度センサ RAM	仕様通りか
	□□□	T_A_RA_B1_5	適温範囲から外れて1時間以上経過する中で、一瞬センサから適温範囲内になってしまう温度を取得してしまったとき、どのような振る舞いをするか確認する。 ※センサ値を平均化せずに生値でそのまま使用していると、一瞬おかしなデータをセンサが拾ってしまっただけで1時間のタイマがリセットされてしまう。そのようなことが無いか確認する。	RA-2-3	ラック状態判定	Q&A No12,55	-	温度センサ RAM	ノイズ :電気的なごみ



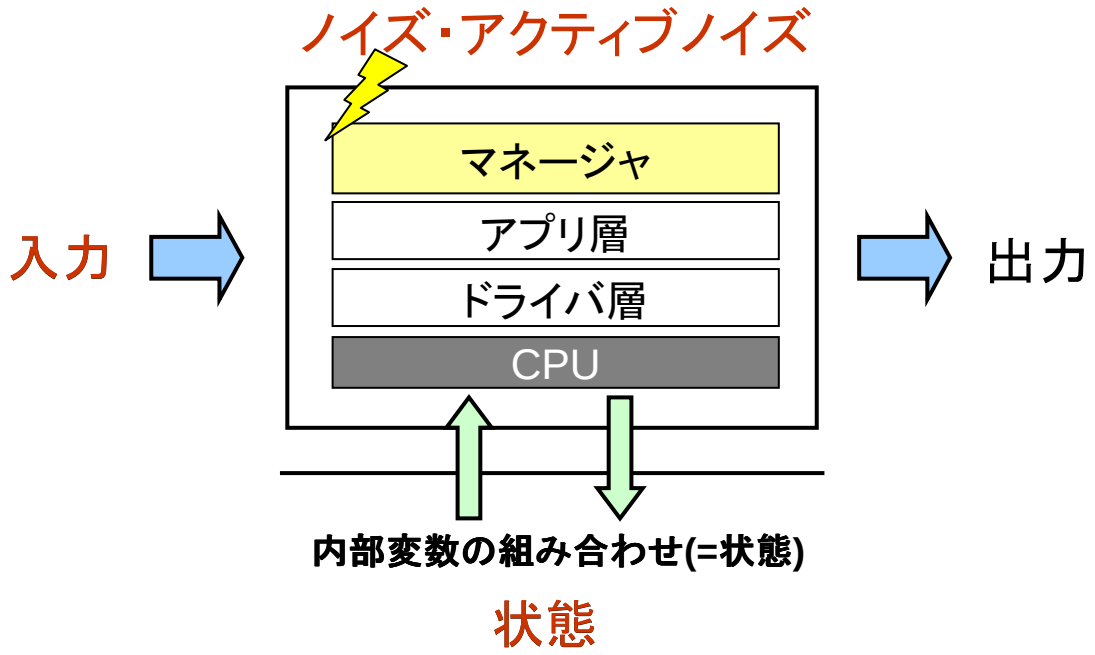
テストアーキテクチャの各テストで何をするか明確になった

テスト詳細設計の工夫点



因子・水準の抽出　(因子)

「入力、ノイズ・アクティブノイズ、状態」に着眼して因子を抽出

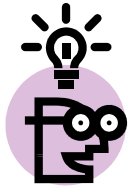


ノイズ	入力と出力の関係に悪影響を与える要因 例：電圧低下、高負荷、RAM化け、アナログ特有のブレ・・・
アクティブノイズ	人の動作によって発生するノイズ 例：不正アクセス、ボタンの連打・長押し
状態	状態を表す変数

因子・水準の抽出　(水準)

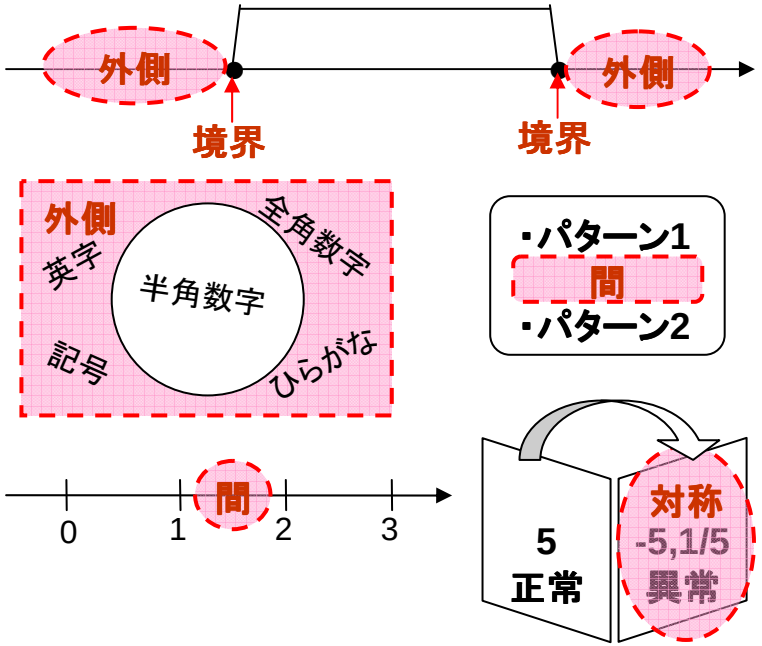
因子に対する水準を抽出する

分析の難しさ　網羅的に抽出することは難しい



境界・外側・間・対称の観点
を使用して分析する

	考える観点	具体例
境界	•データの境界	•「以下」or「未満」 •「0」or「1」始まり
外側	•補集合の外側	•1, 10 ⇒ 文字
間	•データの間 •条件の抜け	•1, 2 ⇒ 1.5 •パターン抜け
対称	•表に対する裏 •Aに対する非A	•5 ⇒ -5, 1/5 •正常 ⇒ 異常



因子・水準の抽出　(FL表)

FL表に因子・水準を整理

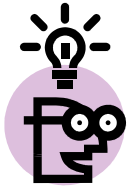
因子			水準				
Input	入力物	貨幣種類(紙幣)	1000円札	2000円札	5000円札	10000円札	その他
		貨幣種類(硬貨)	1円硬貨 (外側)	5円硬貨 (外側)	10円硬貨	50円硬貨	100円硬貨
			500円硬貨	その他 (外側)	-	-	-
	センサ	扉センサ	ON	OFF	-	-	-
		商品取り出し口センサ	ON	OFF	-	-	-
		温度センサ検知温度	X=[冷却適温範囲(最低温度)]-1 (外側)	X=[冷却適温範囲(最低温度)] (境界)	[冷却適温範囲(最低温度)]＜X＜[冷却適温範囲(最高温度)]	X=[冷却適温範囲(最高温度)] (境界)	X=[冷却適温範囲(最高温度)]+1 (外側)
			X=[加熱適温範囲(最低温度)]-1 (外側)	X=[加熱適温範囲(最低温度)] (境界)	[加熱適温範囲(最低温度)]＜X＜[加熱適温範囲(最高温度)]	X=[加熱適温範囲(最高温度)] (境界)	X=[加熱適温範囲(最高温度)]+1 (外側)
			X＜[冷却適温範囲(最低温度)]	[冷却適温範囲(最低温度)]≦X≦[冷却適温範囲(最高温度)]	[冷却適温範囲(最高温度)]＜X	X＜[加熱適温範囲(最高温度)]	[加熱適温範囲(最低温度)]≦X≦[加熱適温範囲(最高温度)]
			[加熱適温範囲(最高温度)]＜X	-1℃ (間)	0℃ (間)	100℃ (間)	-
		HW状態	正常	途絶 (対称)	故障 (対称)	-	-
	ボタン	販売ボタン	ON	OFF	-	-	-
		返金ボタン	ON	OFF	-	-	-
		内部キーボード	TBD(仕様明確化後に洗い出し)				
State	データ	合計金額	X=0 (境界)	0＜X	0＜X＜50	50≦X＜100	100≦X＜500
			500≦X＜1000	1000≦X＜9200	X=9200 (境界)	[商品価格]≦X	X＜[商品価格]
		投入貨幣枚数(10円玉)	X=0	1≦X＜[投入可能貨幣枚数(10	X＝[投入可能貨幣枚数(10		

キーワードを活用することで、多くの因子・水準に気づけた

因子・水準の抽出　(因子・機能一覧)

因子と機能の関係を因子・機能一覧で整理

因子			マスタOPU					貨幣処理OPU				ラック	懸賞
			サービス開始・停止	メンテナンス	販売調停	商品提供	HW状態判定	時間管理	貨幣受付	貨幣排出	釣銭切れ警告	合計金額計算	ラック状態判定	懸賞U/I
Input	入力物	貨幣種類(紙幣)			●				●			●		
		貨幣種類(硬貨)			●				●			●		
	センサ	扉センサ	●		●		●							
		商品取り出し口センサ			●	●		●						
		温度センサ検知温度	●		●	●							●	
		HW状態	●				●							
	ボタン	販売ボタン			●	●								
		返金ボタン			●									
		内部キーボード		●										
State	データ	合計金額			●	●				●		●		
		投入貨幣枚数(10円玉)							●	●				
		投入貨幣枚数(50円玉)							●	●				
		投入貨幣枚数(100円玉)							●	●				



機能毎に必要な因子がすばやく絞れ、確認できる

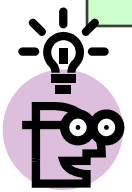
テスト項目一覧の作成(デシジョンテーブルの活用)

デシジョンテーブルを活用し、因子の組合せを表現する

テストID	T_M_MA_B10_1	ブロック名	マスタCPU(マネージャ層)																	
テスト要求	各販売ボタンに対応付いているラックの「在庫数」と「商品の温度」、「合計金額」に応じて、状態(売切れ、準備中、販売可能)が判定され、販売ボタンのランプが正しく点灯/消灯するかを確認する。																			
テスト目的	商品が選択可能な状態であるかを正しくユーザに通知し、ユーザが希望した商品を間違いなく提供し、不利益を与えないことを確認するため。																			
因子	在庫数、温度制御モード、温度センサ検知温度、合計金額																			
備考	下記の条件が成立するものとする。 ・冷却適温範囲(最低温度)≦冷却適温範囲(最高温度)≦加熱適温範囲(最低温度)≦加熱適温範囲(最高温度)										手法・技法・ツール					同値分割、境界値分析				
デシジョンテーブル				1	2	3	4	5	6	7	8	9	10	11	12	13				
条件	在庫数	X=0	Y																	
		0<X		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				
	温度制御モード	冷却	-	Y	Y	Y	Y	Y	Y											
		加熱	-								Y	Y	Y	Y	Y	Y				
	温度センサ検知温度	X<[冷却適温範囲(最低温度)]	-	Y							Y									
		[冷却適温範囲(最低温度)]≦X≦[冷却適温範囲	-		Y	Y						Y								
		[冷却適温範囲(最高温度)]<X<[加熱適温範囲	-				Y						Y							
		[加熱適温範囲(最低温度)]≦X≦[加熱適温範囲	-					Y						Y	Y					
		[加熱適温範囲(最高温度)]<X	-						Y							Y				
	合計金額	X<[商品価格]			Y															
		[商品価格]≦X	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y		Y					
アクション	販売ランプ	点灯				Y								Y						
		消灯	Y	Y	Y		Y	Y	Y	Y	Y	Y			Y	Y				
	準備中ランプ	点灯		Y			Y	Y	Y	Y	Y	Y				Y				
		消灯	Y		Y	Y							Y	Y						
	売切表示ランプ	点灯	Y																	
		消灯		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				

テスト要求一覧のテストIDと紐づく

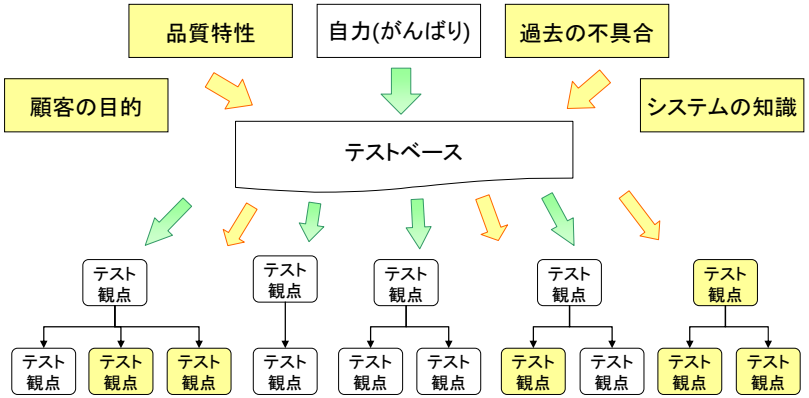
因子・水準はFL表から抜粋

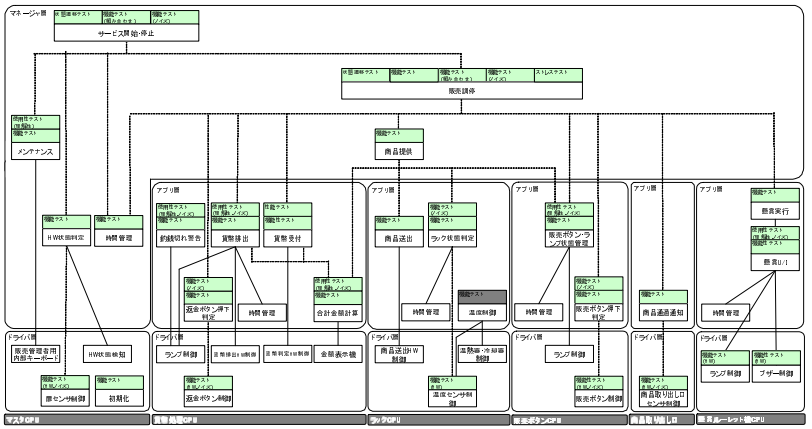


テスト項目を作成することでテストケースに落としやすくなった

まとめ

- テスト観点を抽出する際に必要な経験や知識を補うために、様々な拠り所を活用した。多くにテスト観点に気づけた。
- 機能をテストしやすい粒度に分割し、ドライバ層、アプリ層、マネージャ層に配置し構造化した。テストBOXの活用によって、どのような手順で機能を統合していくか、どの順番でどんなテストを実行するかが一覧視できるようになった。結果、どんなテストが必要なのか、これで十分かの議論がしやすくなった。





ご清聴ありがとうございました