要求定義工程での品質保証とシステムテストのテストケース自動生成新技法

テクマトリックス（株）

TechMatrix

Agenda

- 動機
- FOTとは
- 検証結果

動機

動機(上流工程)

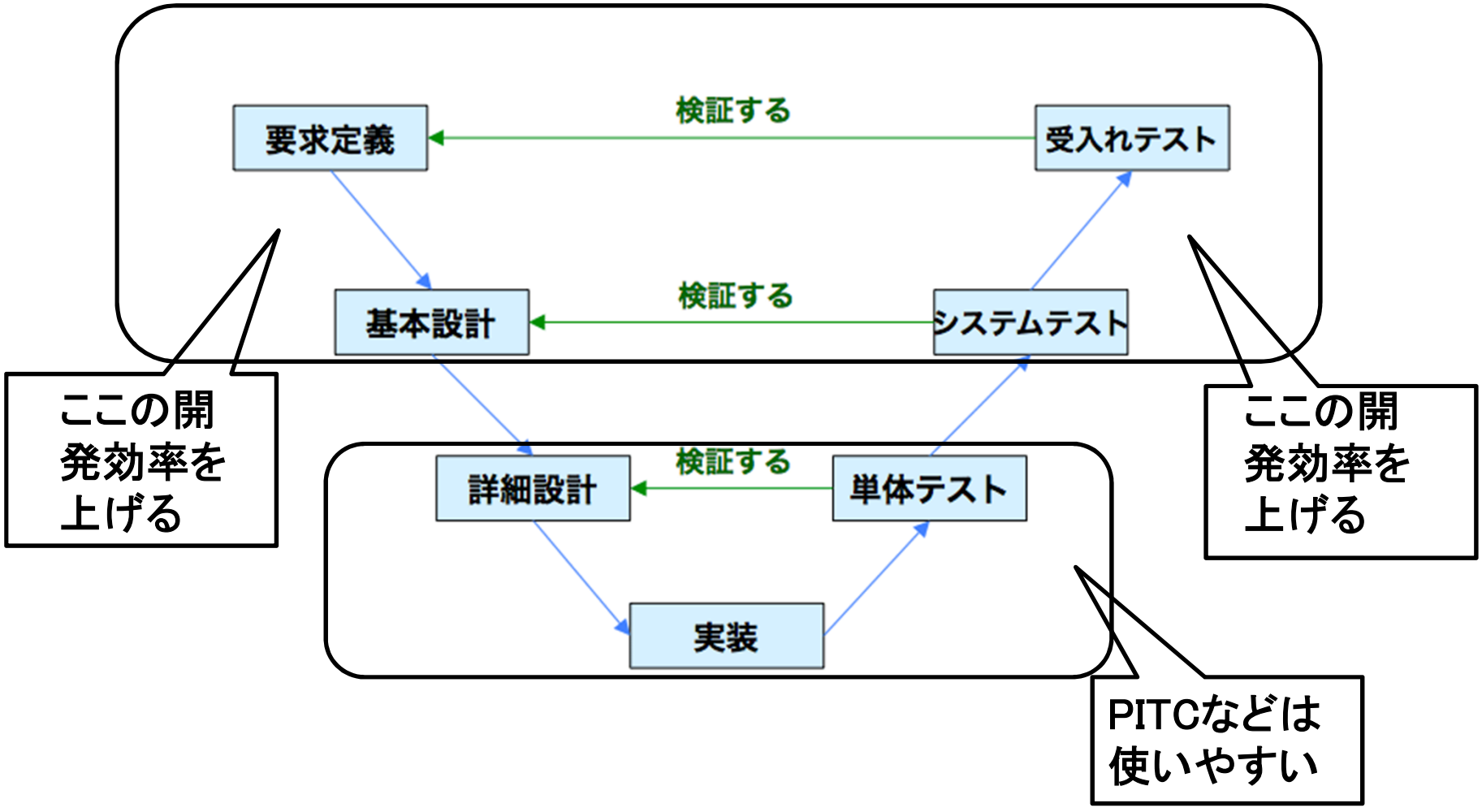
- システムが肥大化、高機能化による、テスト工数の増大。
- 増大の原因はテスト時に考慮するべきテストパラメータ数の増大による、組合せテストの負担増。
- 組合せテストのツールは存在するが(PICT、ACTS)、使い方が煩雑で、今一つ活用されていない。ほとんどが、マニュアルでの手計算による組合せテスト(⇒抜け、漏れ、**重複が頻発**)。**特に、上流でのテストケース作成は、複雑な制約条件を考慮する必要がある。**
- FOTは、上記の課題を解決するための方法論で、組合せテストを効率良く実施するための方法。

PICTの記述例（上流工程）

```
# 因子とその水準の列挙
サイズ: 大, 小
色: 赤, 青, 緑
形: 丸, 三角形, 四角形
三角形: 正三角形, 二等辺三角形, 不等辺三角形, dummy

# [要求８] サイズ小のブロックは緑色のみである。
IF([サイズ] = "小") THEN [色] = "緑";

# [要求７] 赤色で三角形のブロックはない。
NOT ([形] = "三角形" AND [色] = "赤");

# 階層構造を表現するための制約式
IF([三角形] = "正三角形") THEN [形] = "三角形";
IF([三角形] = "二等辺三角形") THEN [形] = "三角形";
IF([三角形] = "不等辺三角形") THEN [形] = "三角形";
NOT([形] = "三角形" AND [三角形] = "dummy");
```

後出の、コンピュータビジョンの記述例

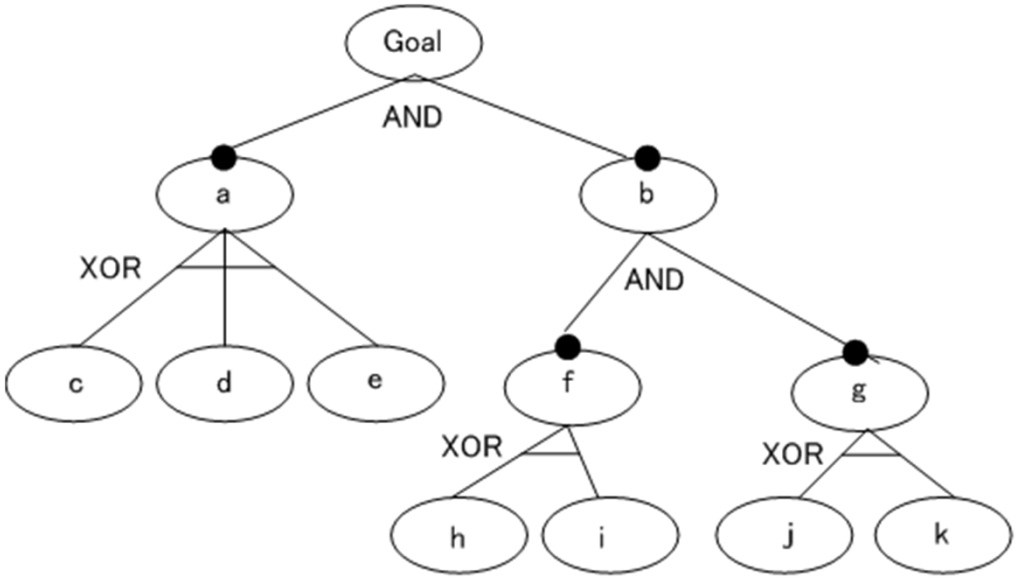
動機(上流工程)



Feature Oriented Testing(FOT) とは

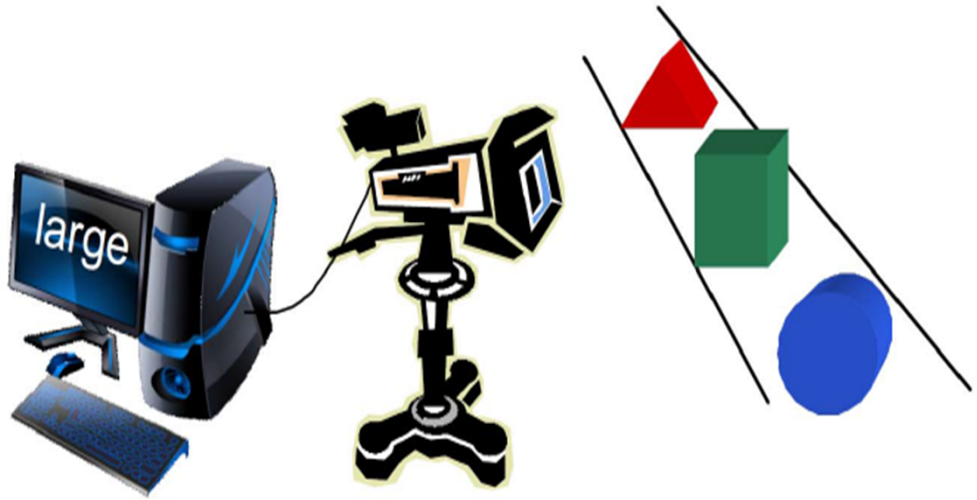
FOTとは

- FOT は、拡張ロジックツリーを用いた系統的なテストモデルの作成（テスト設計）の支援、及び、そうしたテストモデルからPair-wise 網羅基準を満たすテストケースの機械（自動）生成の機能を持つ。



例題をもとに拡張ロジックツリーを説明

例:様々なブロックのサイズを認識するコンピュータビジョンシステムのテストケース設計



【要件1】 ブロックは1個ずつ流れる

【要件2】 ブロックは、サイズ、色、形の属性を持つ

【要件3】 サイズは、大、小

【要件4】 色は赤、緑、青

【要件5】 形は、四角形、丸、三角形がある

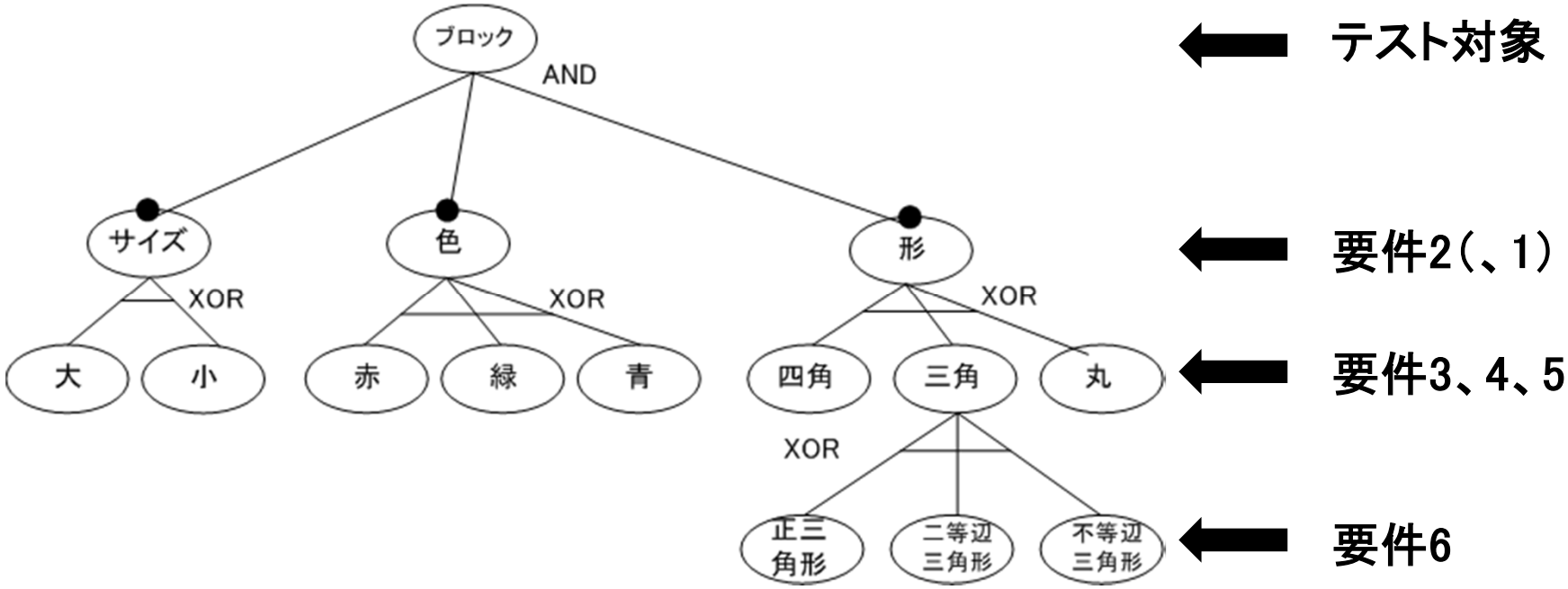
【要件6】 三角形には、正三角形、二等辺三角形、不等辺三角形がある。

【要件7】 赤色で三角形のブロックはない。

【要件8】 小さいサイズのブロックは緑だけ。

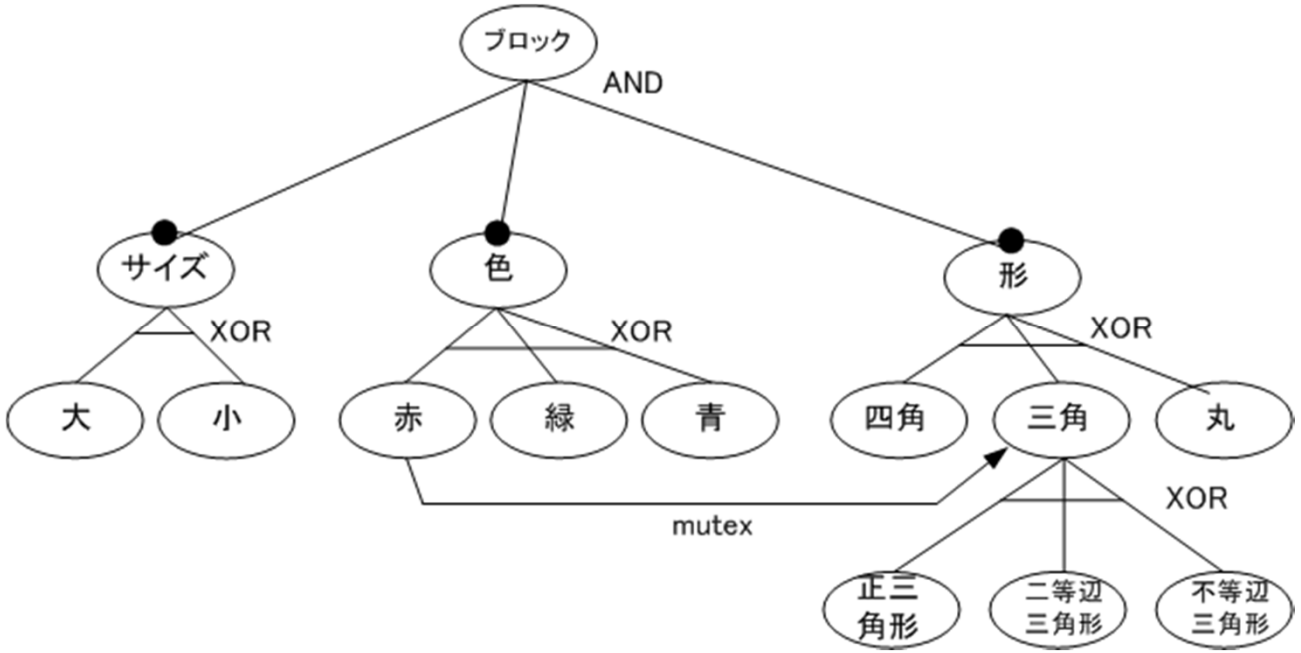
例題をロジックツリーで表現

- FOT の基本的な分析は、「根ノードを頂点とした分解（division）の繰り返しによる分析」



拡張ロジックツリー　要件7への対応

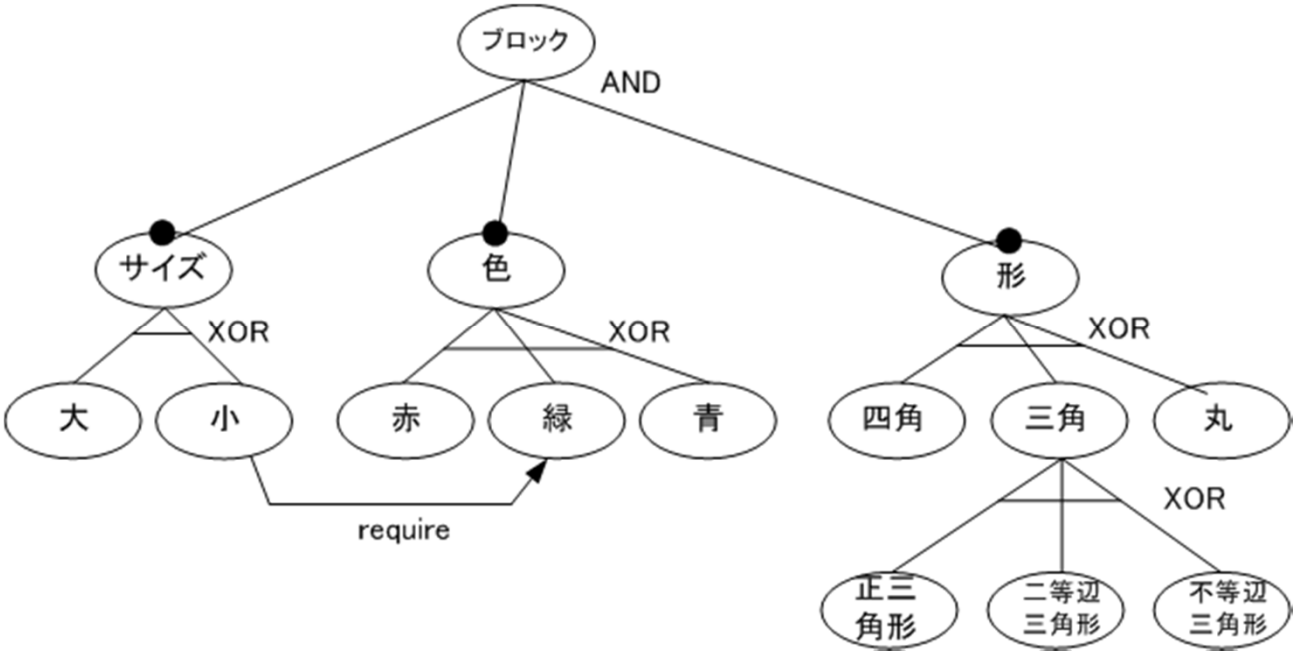
- 「～はない」を記述可能にするために、mutexを準備。
- a mutex b　は「a と b は両立しない」 意味する。



「赤」 mutex 「三角」 : 「赤」と「三角」は両立しない

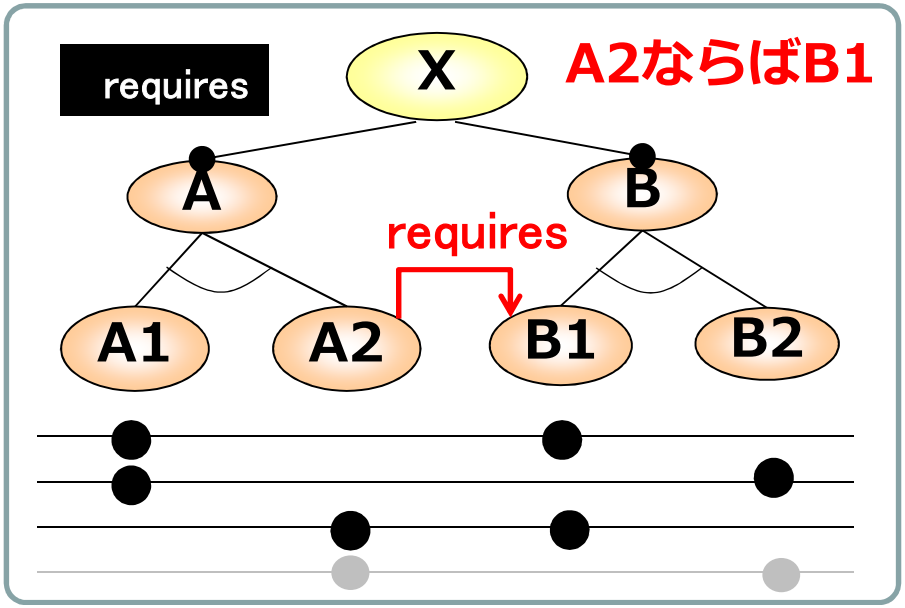
拡張ロジックツリー　要件8への対応

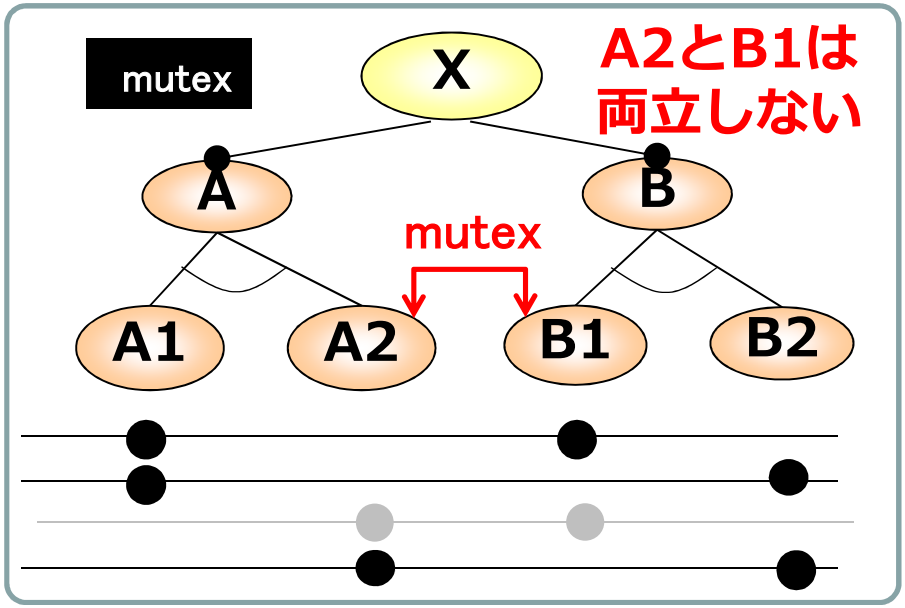
- 「～時は必ず～」を記述可能にするために、requireを準備。
- a require b　は「a の時は必ず b である」 を意味する。

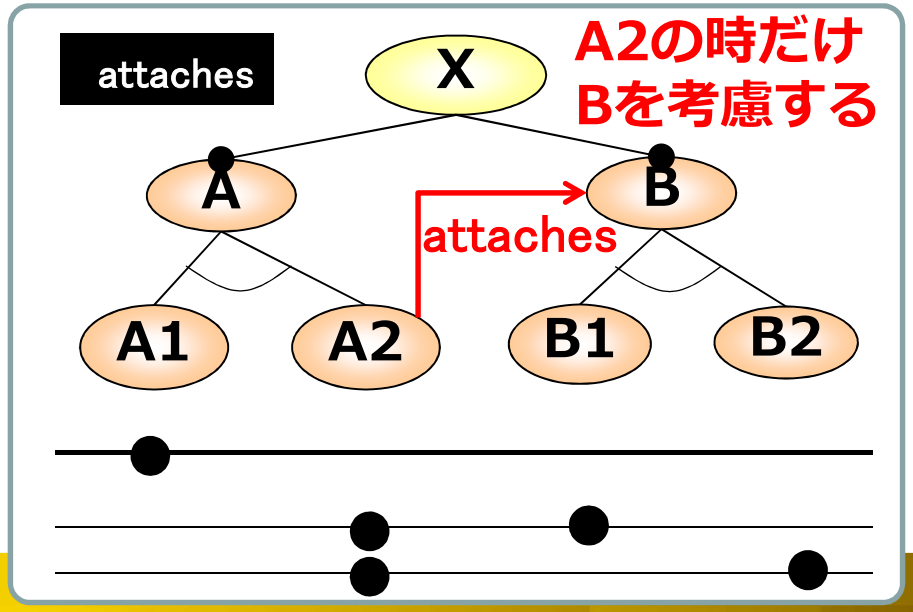


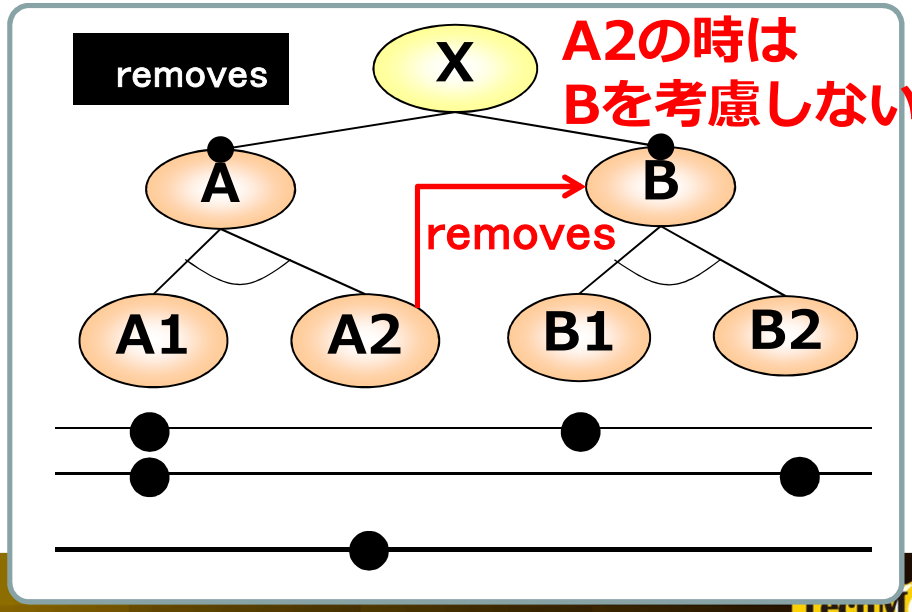
「小」 require 「緑」 ： 「小」の時は必ず「緑」

その他の拡張制約、及び、再整理





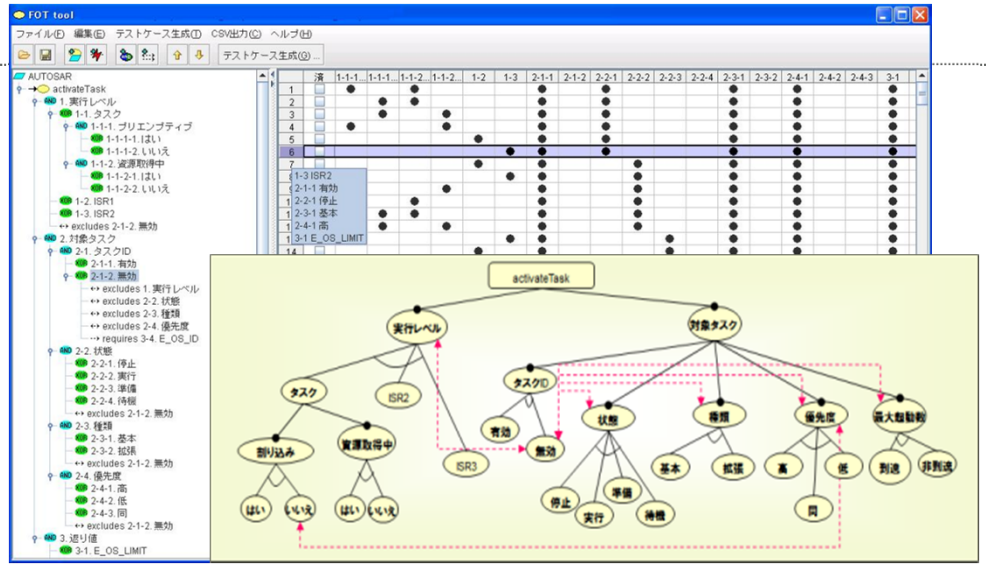




FOTの運用方法

開発者・顧客

入力



出力

変換

変換アルゴリズム

activateTask

テストケースNo.	Execution_level			Task					日付
	Normal								
	Preemptive	No_Constraints		TaskID	status	type	priority	Max_activation	
1	–	–	–	Invalid	–	–	–	–	
2	–	–	ISR3	Valid	suspend	extended	high	reached	2012/10/1
3	–	–	ISR3	Valid	ready	extended	high	reached	2012/10/1
4	–	–	ISR3	Valid	run	extended	high	reached	2012/10/1
5	–	–	ISR3	Valid	run	extended	high	not_reached	2012/10/1
6	–	–	ISR3	Valid	run	extended	equiv	not_reached	
7	–	–	ISR3	Valid	run	basic	equiv	not_reached	
8	–	–	ISR3	Valid	run	basic	low	not_reached	
9	–	–	ISR3	Valid	wait	basic	low	not_reached	
10	Yes	No	–	Valid	wait	basic	low	not_reached	
11	No	No	–	Valid	wait	basic	–	not_reached	
12	No	No	–	Valid	suspend	basic	–	not_reached	
13	No	No	–	Valid	ready	basic	–	not_reached	
14	No	No	–	Valid	ready	basic	–	reached	
15	No	No	–	Valid	ready	extended	–	reached	
16	No	No	–	Valid	run	extended	–	reached	
17	No	Yes	–	Valid	run	extended	–	reached	
18	Yes	Yes	–	Valid	run	extended	equiv	reached	
19	Yes	Yes	–	Valid	wait	extended	equiv	reached	
20	Yes	Yes	–	Valid	suspend	extended	equiv	reached	
21	Yes	Yes	–	Valid	suspend	extended	equiv	not_reached	

出力

```
実行レベル: タスク, ISR2, ISR3↓
割り込み: はい, いいえ, ～↓
資源取得中: はい, いいえ, ～↓
タスクID: 有効, 無効↓
状態: 停止, 実行, 準備, 待機↓
種類 : 基本, 拡張↓
優先度 : 高, 同, 低↓
最大起動数 : 到達, 非到達↓
↓
# CONSTRAINTS↓
IF([割り込み] = "はい") THEN [実行レベル] = "タスク";↓
IF([割り込み] = "いいえ") THEN [実行レベル] = "タスク";↓
IF([資源取得中] = "はい") THEN [実行レベル] = "タスク";↓
IF([資源取得中] = "いいえ") THEN [実行レベル] = "タスク";↓
NOT([タスクID] = "無効" AND [状態] in {"停止", "実行", "準備", "待機"});↓
NOT([タスクID] = "無効" AND [種類] in {"基本", "拡張"});↓
NOT([タスクID] = "無効" AND [優先度] in {"高", "同", "低"});↓
NOT([タスクID] = "無効" AND [最大起動数] in {"到達", "非到達"});↓
NOT([タスクID] = "無効" AND [最大起動数] in {"タスク", "ISR2", "ISR3"});↓
NOT([割り込み] = "～" AND [実行レベル] = "タスク");↓
NOT([資源取得中] = "～" AND [実行レベル] = "タスク");↓
```

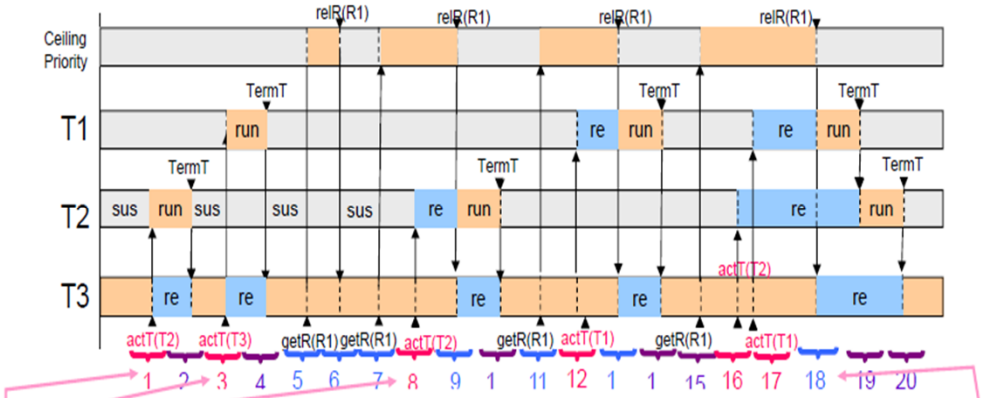
テスト設計書Excel出力

PICT・ACTS形式

有効性の検証

FOT有効性の検証 【その1】 OSEK Kernel

実際のテストケースの観点



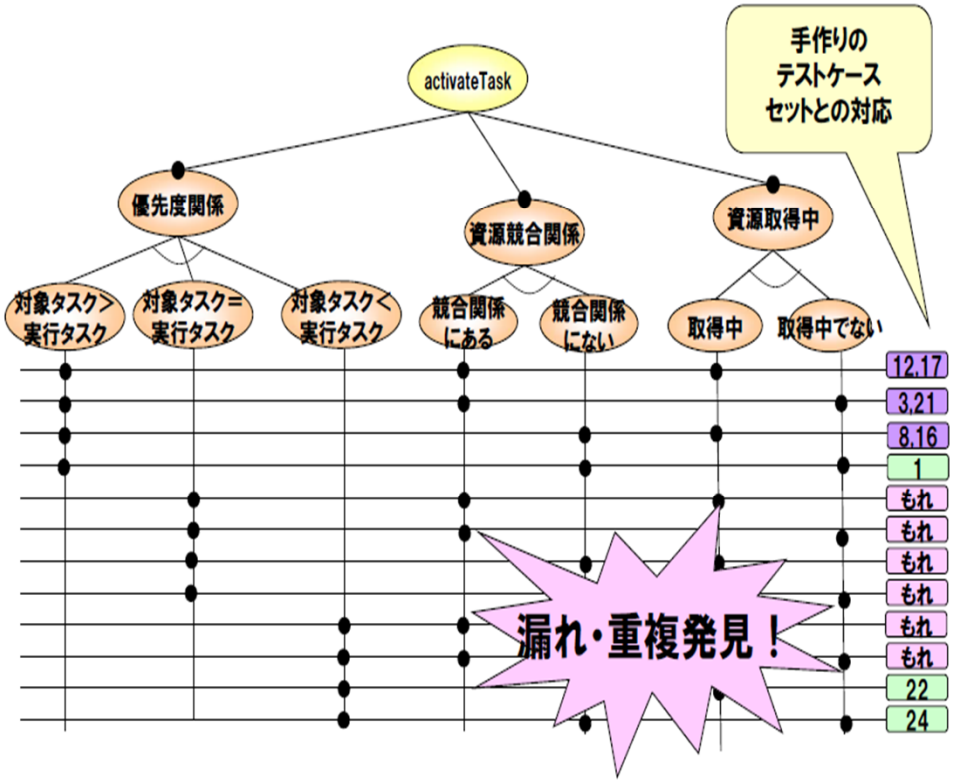
17. ある資源を取得中に，実行タスクと資源競合関係にあり，かつ，実行タスクより優先度の低いタスク（T1）をActivateする場合

8. ある資源を取得中に，実行タスクと資源競合関係になく，かつ実行タスクより優先度の高いタスク（T2）をActivateする場合

3. 実行タスクと資源競合関係にあり，かつそのタスクより優先度の高いタスク（T2）をActivateする場合

1. 実行タスクと資源競合関係になく，かつそのタスクより優先度の高いタスク（T2）をActivateする場合

このテストケース設計のテストケースセット



FOT有効性の検証 【その2】

狙い

- 実装されている制約で、テストケースの記述は「十分に」可能か。
- 人手（KKDです）で作成されたテストケースは最低でもきちんとカバーされているか。

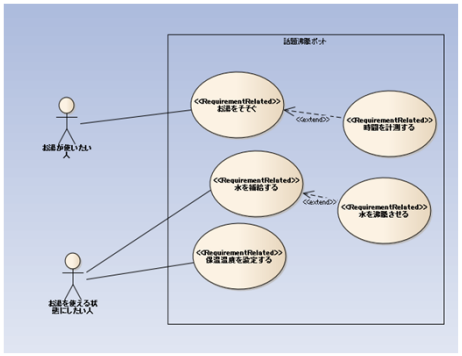
方法、対象

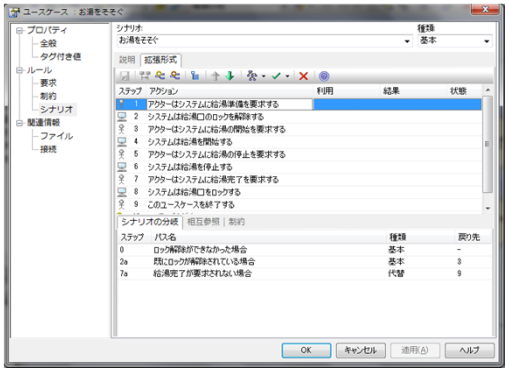
- ドメイン分析から導出されたUMLのユースケースを使用する。特に、状態遷移からのテストケース生成が不可（FOTの制限）であるため、ユースケースシナリオ（アクティビティ図）を重視。
- FOTで生成されたテストケースを使用して、ユースケースシナリオを「動かす」。即ち、「モデル」のテストを行い、要求分析の間違いを早期に発見。合わせて、テストケースの妥当性も確認する。
- 既開発済みの（エンタープライズ系）プロジェクトに適用。

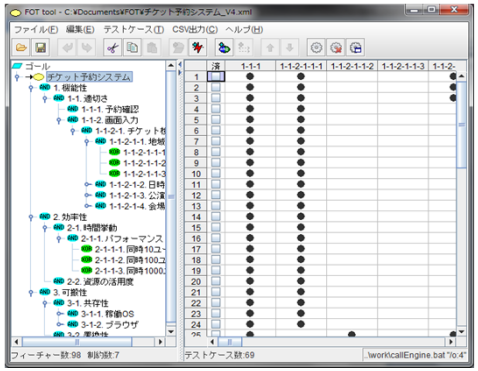
FOT有効性の検証　【その2】

結果

- 制約条件に関しては、十分使用可能であることが確認できた。即ち、FOTで既実装済みの制約で十分表現可能。
- テストケース作成までの工数も、約1/3に短縮。特に、レビュー工数の短縮が大きい。
- アクティビティ図で、14STEP、画面数5のサブシステムで、手動による（既存のテストケース）では20ケース、FOTによるテストケース生成では23ケース。3ケースの漏れを発見。
- 要求仕様から直接因子、水準を決定することは非常に難しい、また、因子間の制約（禁則）は要求仕様に書かれていない場合も多い　⇒　普遍的な困難。



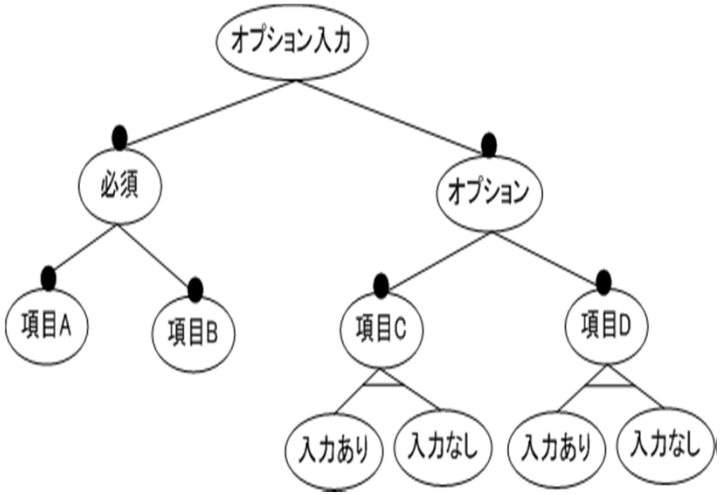


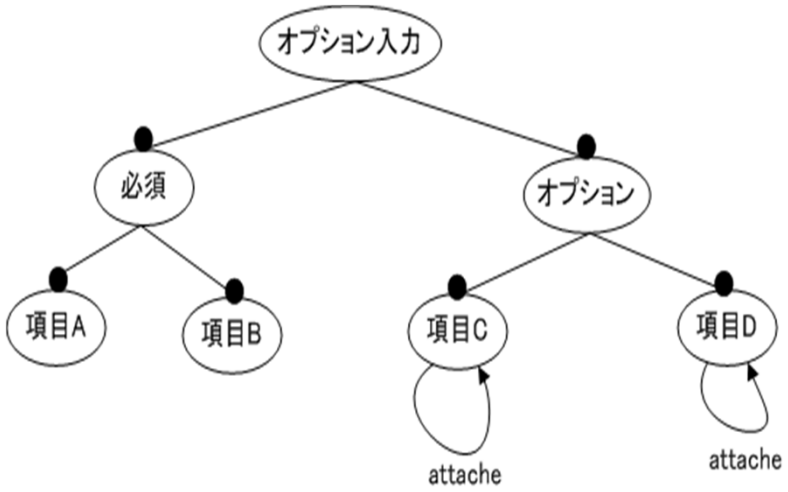


今後の課題

FOT有効性の検証の結果

- 拡張ロジックツリーの質の問題。現状、拡張ロジックツリーは作成者に依存する部分も多い。 例えば、AとBは必須だが、CとDはオプションの場合。下記は同じことを表現しているがどちらが良い？





- 仕様書からの自動生成など。