ET2014 IPAブースプレゼンテーション

システム障害原因診断
～要求仕様検証マップ～

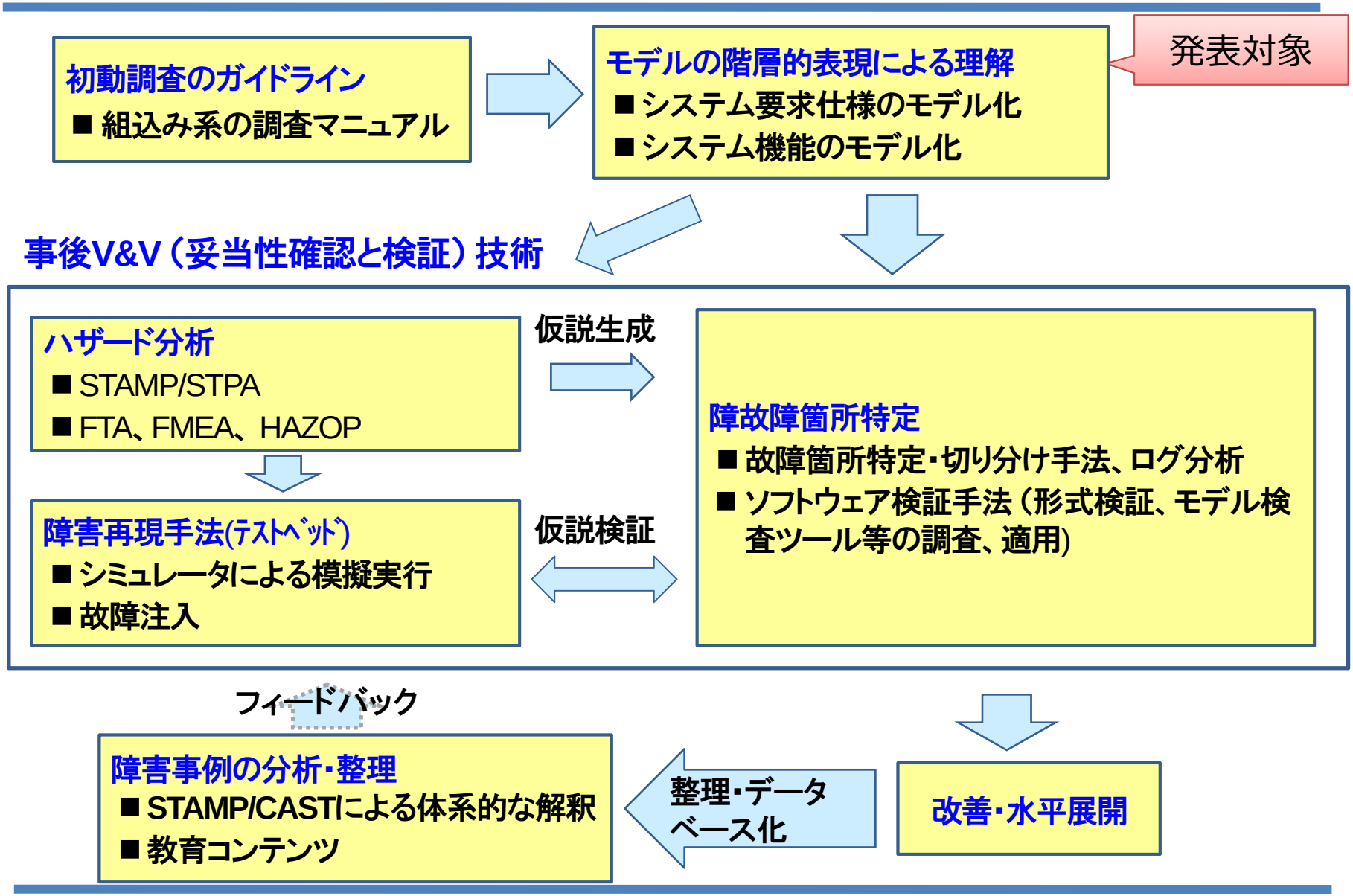
2014年11月

中村 洋
株式会社レンタコーチ
http://www.rentaco.jp/

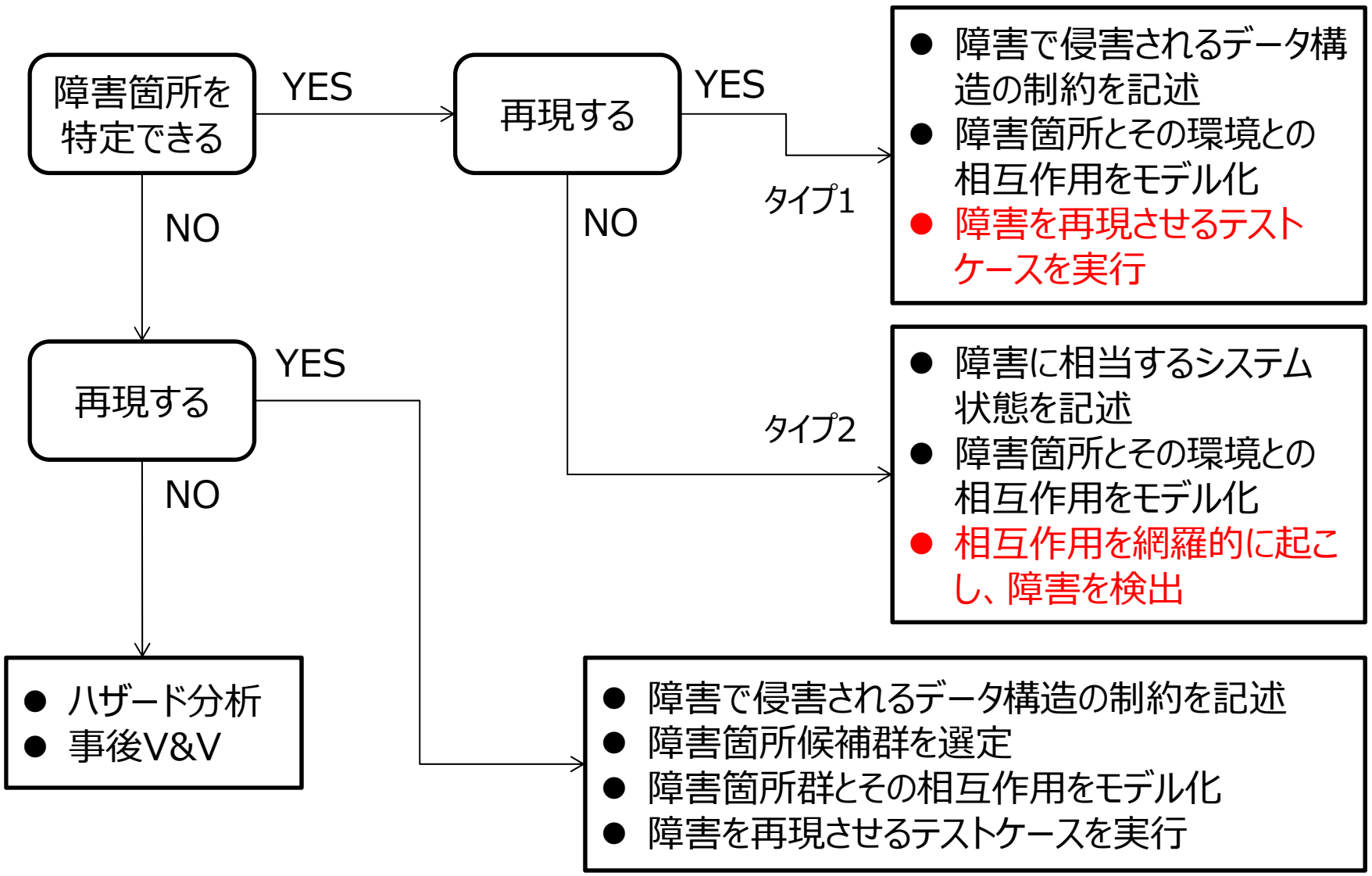
発表内容

- ◆事後V&Vのフレームワーク
- ◆モデルの階層的表現による理解
  - ■原因究明に使える診断手法の分類
  - ■要求仕様の検証に使える記述手法の特徴
- ◆例題
  - ■速度制限システム
- ◆まとめ

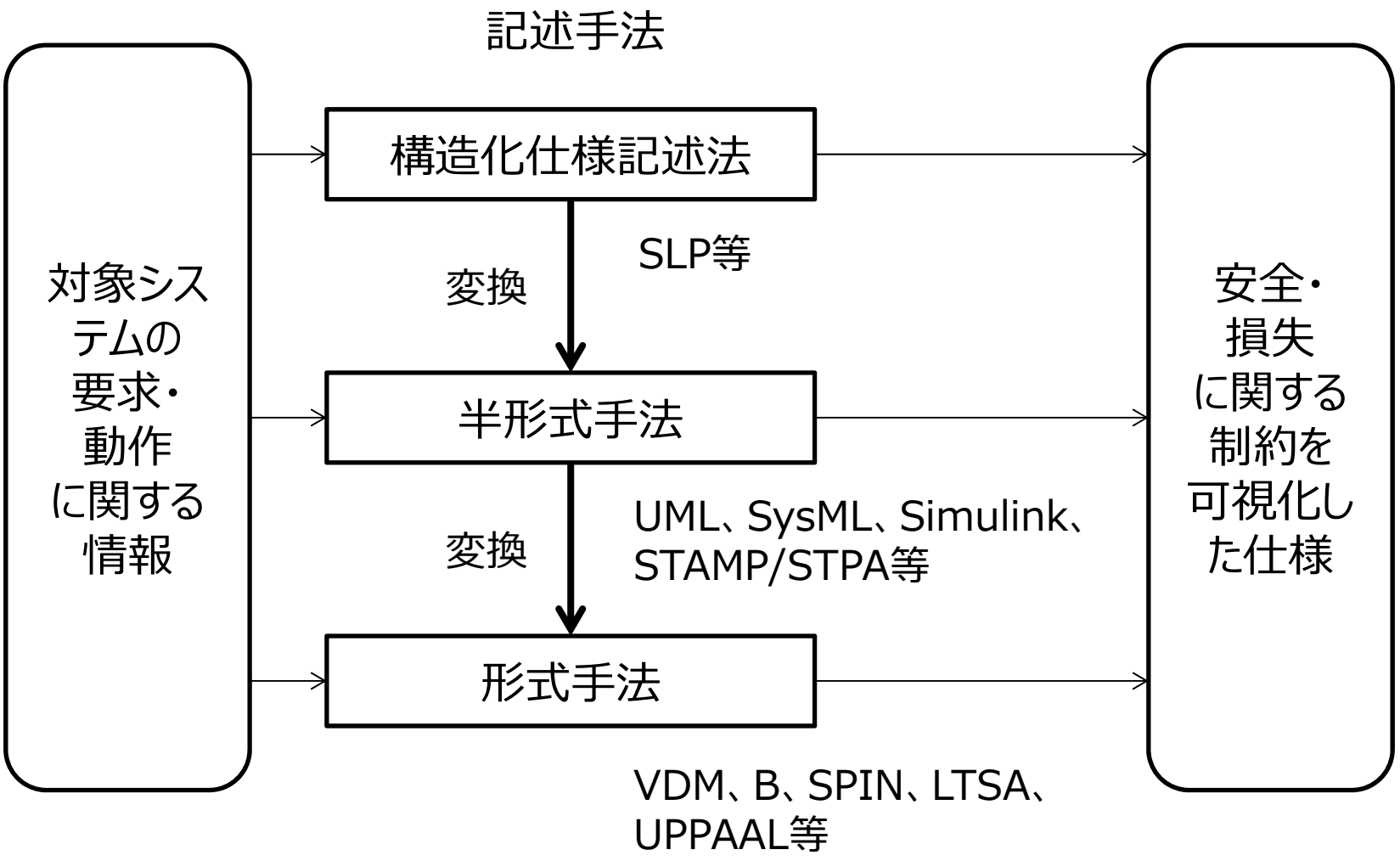
事後V&Vのフレームワーク



障害の発生形態に基づく診断手法



システム要求・動作の記述手法



注: STAMP/STPAはハザード要因に関する分析手法。SLPは要求記述言語。

記述手法の特徴

手法	ツール	要求・機能の記述	制約の記述と検証
モデル検査	SPIN	システムとその環境との相互作用を並行動作するプロセスとして記述。	制約を時相論理LTLで記述し、その反例をツールが見つける。
	LTSA	動作を有限状態プロセス(FSP)として記述。	時相論理、デッドロック等を用いて記述し、検出。
	UPPAAL	システムとその環境との相互作用を拡張時間オートマトンで記述。	制約を時相論理CTLで記述し、その反例をツールが見つける。
形式仕様記述法	VDM	機能をデータ構造とその操作とで記述する。	データ構造に関する不変条件を記述し、その侵害をテストによって検出。
	B	抽象機械の集まりとして記述。	不変条件として記述。
SysML	UML拡張版	構造と動作を図的に表現する。	構成要素の数値特性に関する制約条件を記述できる。
UML	多々	SysML同様。	OCLで記述できる。
MBD	Simulink		
STAMP/STPA	不明	構成要素とその相互作用を図的に表現。	ハザード要因に対する安全制約を識別し、記述する。

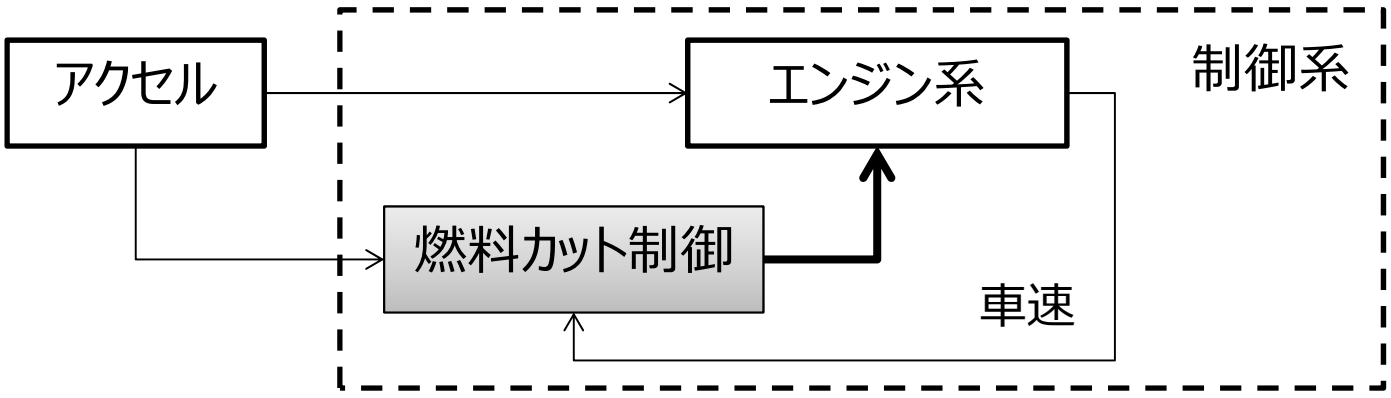
例題：速度制限システム

◆要求事項

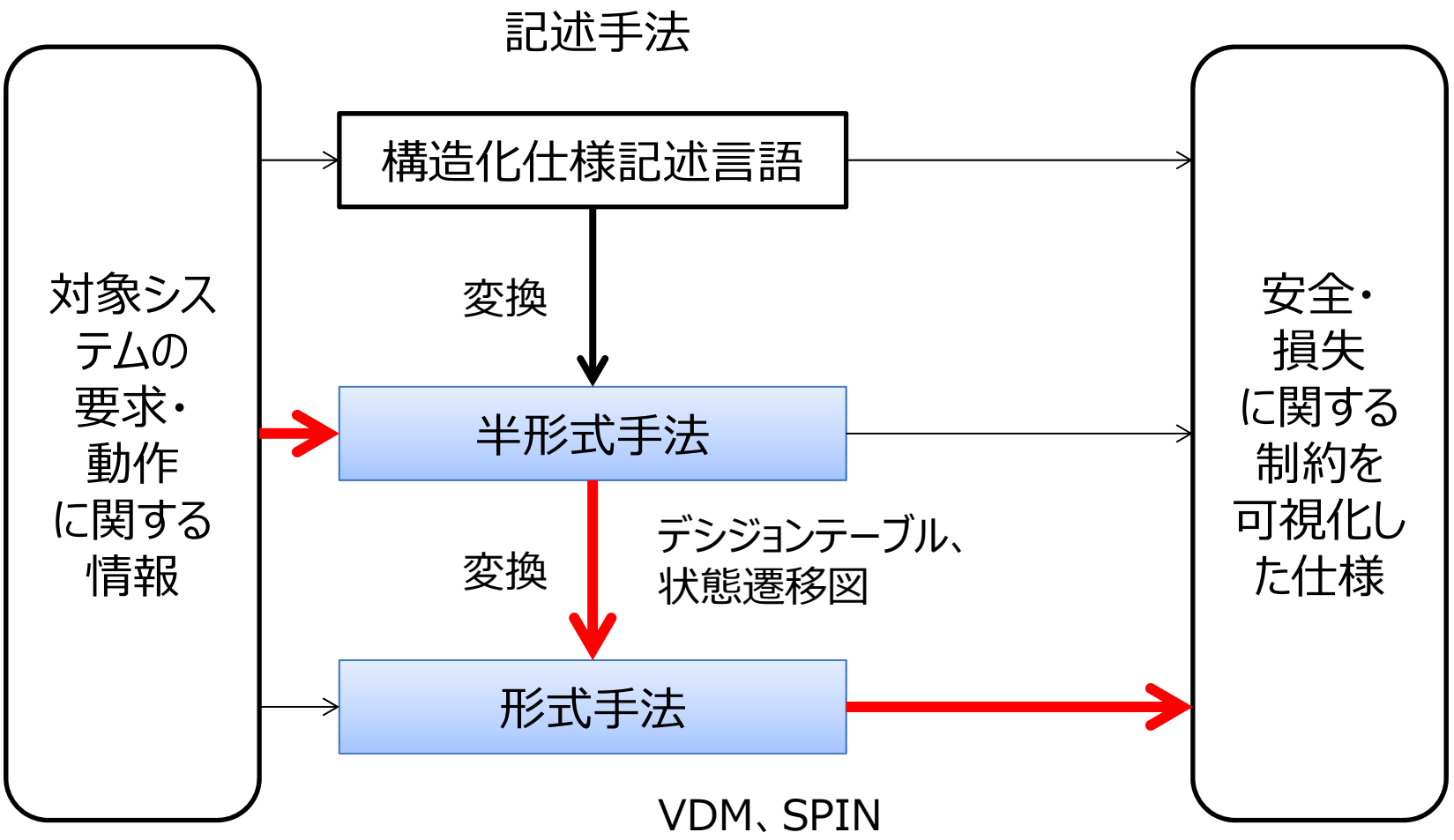
- ■ 車の速度を80km/H以下に制御する。
- ■ 速度制限は、燃料カットによって行う。

◆車速の動きに関して

- ■ アクセルが踏まれていると、車速は8km/Hずつ上昇する。
- ■ アクセルが離れていると、車速は4km/Hずつ減速する。
- ■ 燃料カットが働くと、車速は4km/Hずつ減速する。



使用する記述手法



まず、制御仕様をデシジョンテーブルで書いてみる

条件/動作		規則1	規則2	規則3	規則4	規則5	規則6
条件1	燃料カットがOFF	T	T	T	F	F	F
条件2	アクセルが踏まれている	T	T	F	T	T	F
条件3	(車速+8) > 制限速度	T	F	–	F	T	–
動作	何もしない		✓	✓		✓	
	燃料カットをONにする	✓					
	燃料カットをOFFにする				✓		✓

条件3:
次の状態で、車速が制限速度を超える

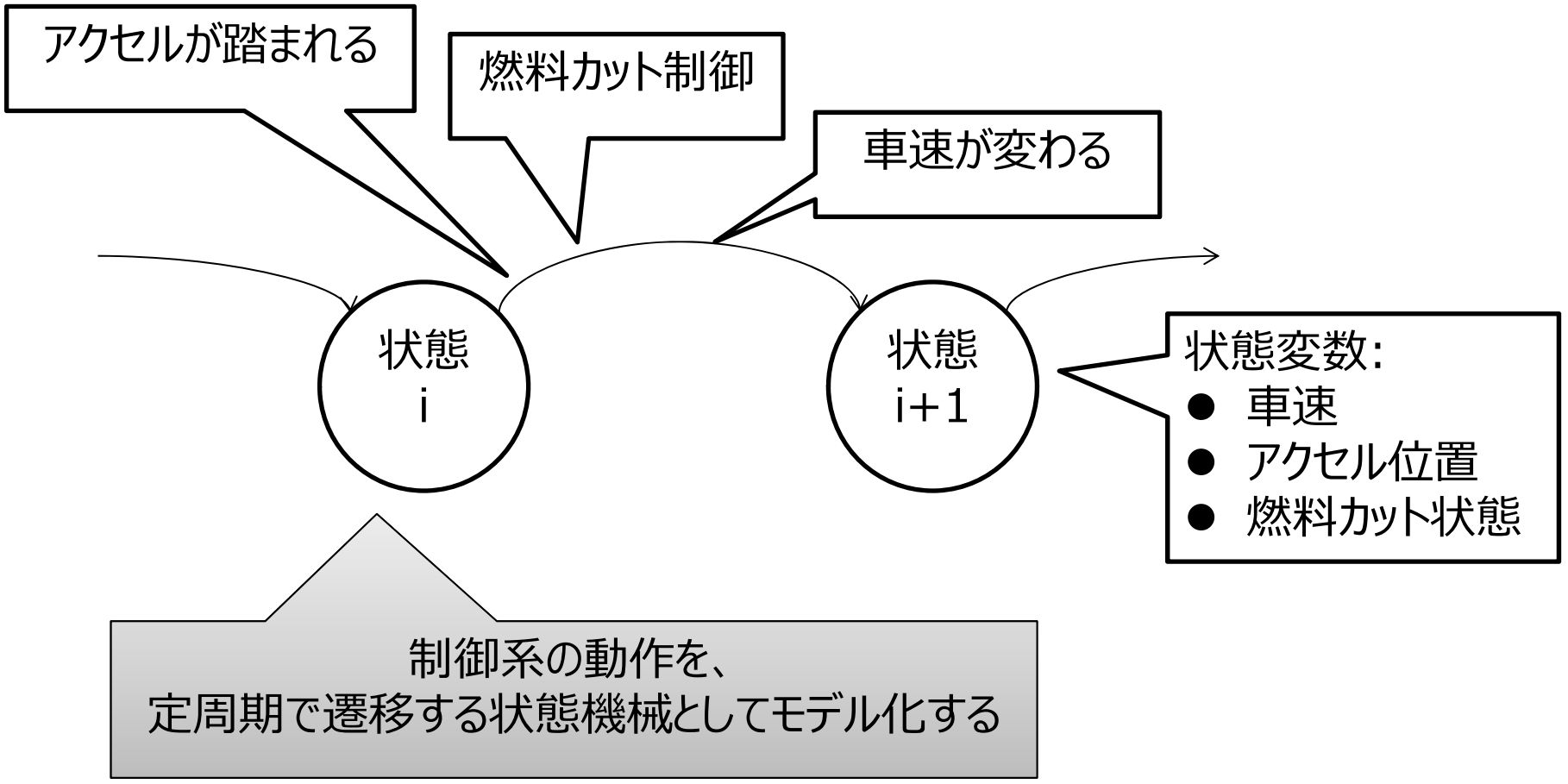
デシジョンテーブルをもとにVDM++で記述する

```
values
  制限速度 = 80 ; 加速幅 = 8 ; 減速幅 = 4 ;
instance variables
    車速 : nat := 0 ;
    アクセル : <on> | <off> ;
    燃料カット : <on> | <off> ;
inv 車速 <= 制限速度 ;
operations
燃料カット制御 : () ==> ()
燃料カット制御() == is not yet specified
  post
      (燃料カット~ = <off> and アクセル~ = <on> and
       車速~ + 加速幅 > 制限速度
       => 燃料カット = <on> )
      and
      ((燃料カット~ = <on> and アクセル~ = <on> and
        車速~ + 加速幅 <= 制限速度
        ) or
        (燃料カット~ = <on> and アクセル~ = <off>)
        => 燃料カット = <off> ) ;
```

データ構造に関する
不変条件として、
安全制約を記述

操作の
事後条件として
仕様を記述

制御仕様を検証するには



実行可能な仕様に変える

```
operations
燃料カット制御 : () ==> ()
燃料カット制御() ==
  ( if 燃料カット = <off> and
       アクセル = <on> and 車速 + 加速幅 > 制限速度 then
        ( 燃料カット := <on> ; return) ;
    if 燃料カット = <on> and
       ((アクセル = <on> and 車速 + 加速幅 <= 制限速度 ) or
        (アクセル = <off>) ) then
         ( 燃料カット := <off> ; return) ;
    return ) ;

車速変化 : () ==> ()
車速変化() ==
    if 燃料カット = <on> then 車速 := 車速 - 減速幅
    elseif アクセル = <on> then 車速 := 車速 + 加速幅
    else 車速 := 車速 - 減速幅 ;
```

アクセル動作をモデル化してテストする

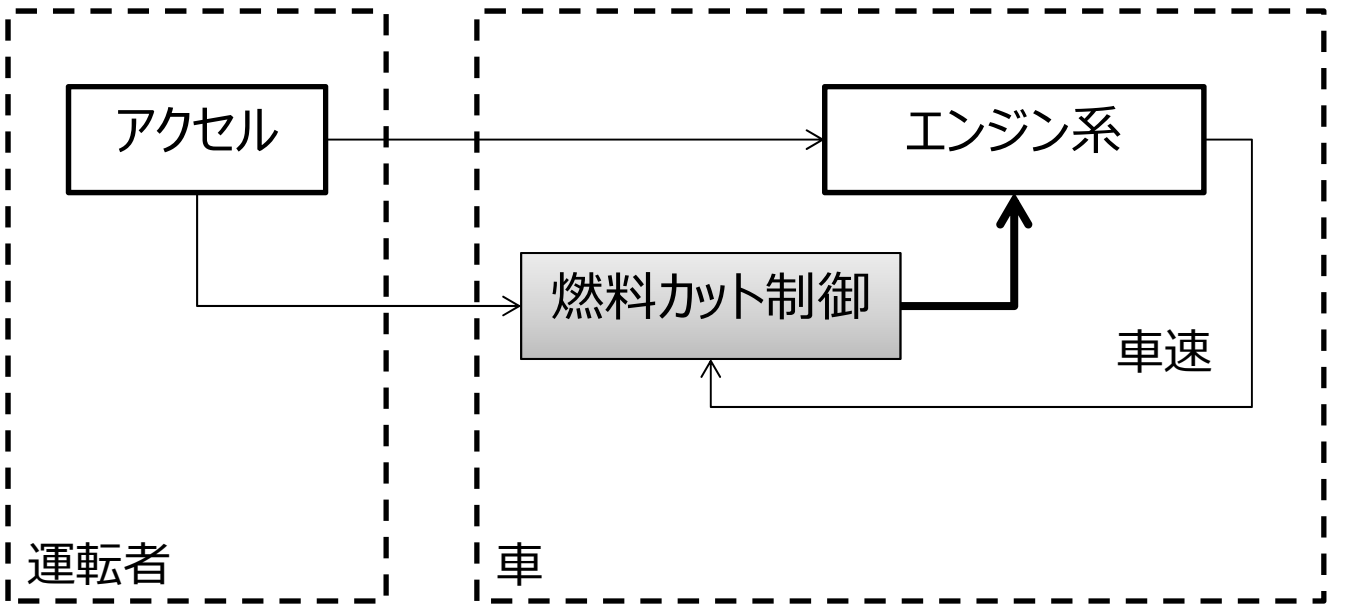
```
public test : seq of (<on> | <off>) ==> seq of nat
test(pattern) ==
  ( dcl ix : nat , ptn : seq of (<on> | <off>) , speeds : seq of nat ;
    if pattern = [] then ptn := [<on>,<on>,<on>,<off>]
    else ptn := pattern ;
    車速 := 0 ; アクセル := <off> ; 燃料カット := <off> ;
    ix := 0 ; speeds := [0] ;
    for i = 1 to 100 do
      ( アクセル := ptn(ix mod (len ptn) + 1) ; ix := ix + 1 ;
        燃料カット制御() ;
        車速変化() ;
        speeds := speeds ^ [車速] ) ;
    return speeds
  ) ;
```

入力:アクセル動作
出力:車速の変化

アクセル動作をモデル化

testの出力を見て、制限速度が守られていることを確認する

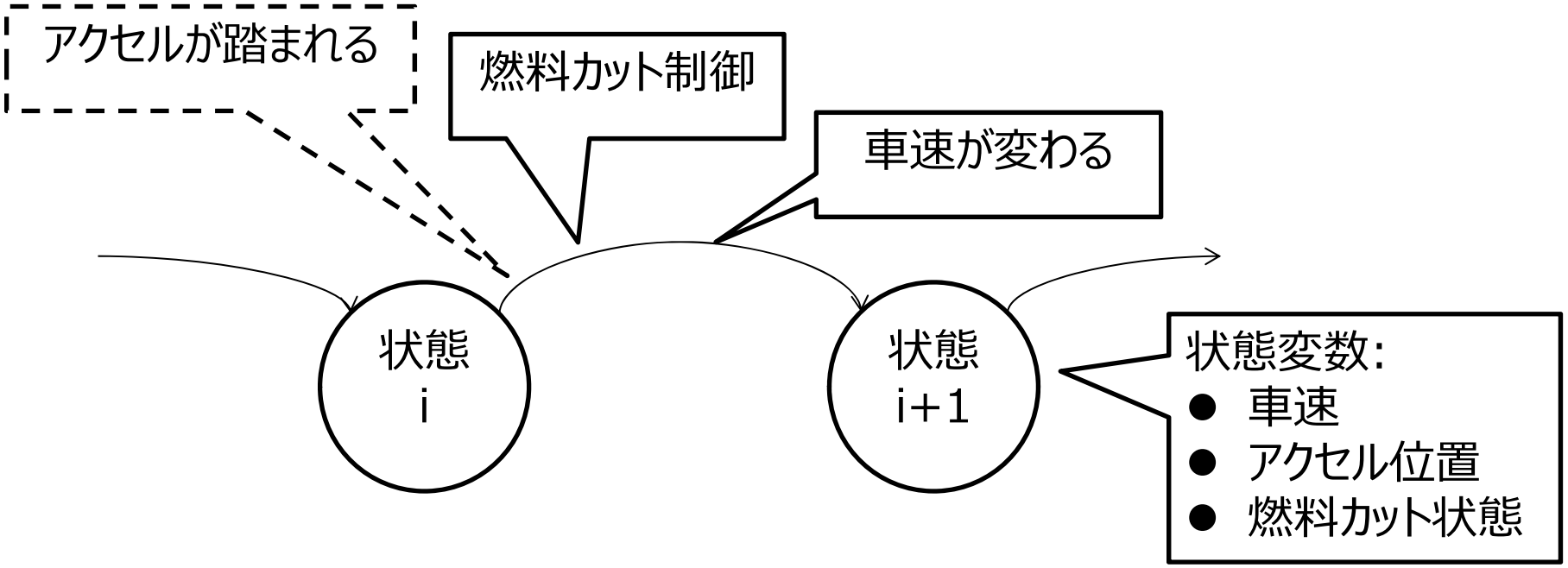
SPINで検証するには



運転者と車という2プロセスの相互作用としてモデル化する

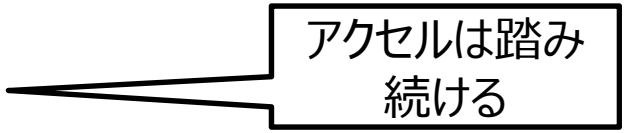
運転者はアクセルをずっと踏んでいると想定すれば、
1プロセスでモデル化できる

車の動作を状態遷移系でモデル化する



状態遷移系の動作をPromelaで記述する

```
mtype = {on,off} ;
mtype pedal, fcut ;
int speed ;
active proctype shasoku(){
speed = 0 ; fcut = off ; pedal = on ;
do
::true ->
  if
  ::fcut == off && pedal == on && speed + 8 > 80 -> fcut = on
  ::fcut == on && pedal == on && speed + 8 <= 80 -> fcut = off
  ::fcut == on && pedal == off -> fcut = off
  ::else ->skip
  fi ;
  if
  ::fcut == on -> speed = speed - 4
  ::fcut == off && pedal == on -> speed = speed + 8
  ::else -> speed = speed - 4
  fi
od
}
```



安全制約を時相論理LTLで記述する

◆安全制約

- 常に車速が制限速度を超えることはない、又は
- いつか車速が制限速度を超えることはない

◆LTLで記述すると

- []! (speed > 80) 、又は
- ！<> (speed > 80)

反例がでないことで、制限速度が守られていることを確認する

まとめ

- ◆障害原因の診断手法は、障害箇所の特定、現象の再現という発生形態の違いによって分類できる。
- ◆不十分な情報から、安全や損失に関する制約を可視化した仕様を作る必要があり、そのために半形式手法や形式手法等によるシステム要求や動作のモデル化が役に立つ。
- ◆SPINを使うと、対象システムを並行動作するプロセスとして記述し、制約を時相論理で記述し、その侵害を網羅的に検出できる。
- ◆VDMの場合には、制約をデータ構造に関する不変条件として記述し、その侵害をテストケースによって検出できる。

ご清聴ありがとうございました

IPAセミナーのご案内:

21日午後2:30−3:30
アネックスホール2階 F205にて

第7部
モデルベースアプローチに基づく障害原因診断手法