

Sim4stamp

STAMP/STPAシミュレータの紹介

— The simulation tool for STAMP/STPA —

積田 恵一

Agenda

1. STAMP/STPA
2. STAMP/STPAを実施する上での課題
3. シミュレーションを用いた思考
4. Sim4stamp
5. モデリングから言えること
6. まとめ
7. 今後の課題

STAMP/STPA

STAMPとは.....

- 事故はイベントの連鎖よりも複雑な動的プロセスを含むものとして捉える。
 - 事故を(要素または部品)故障の問題ではなく、**コントロールの問題として捉える。**
 - コンポーネントの振る舞いと相互作用の制約を課すことにより、事故を防ぐ。
- ➔ 事故のより深い原因を捉えることが可能になる。

STAMP : System-Theoretic Accident Model and Processes

STPA : System Theoretic Process Analysis

STAMP/STPAを実施する上での課題

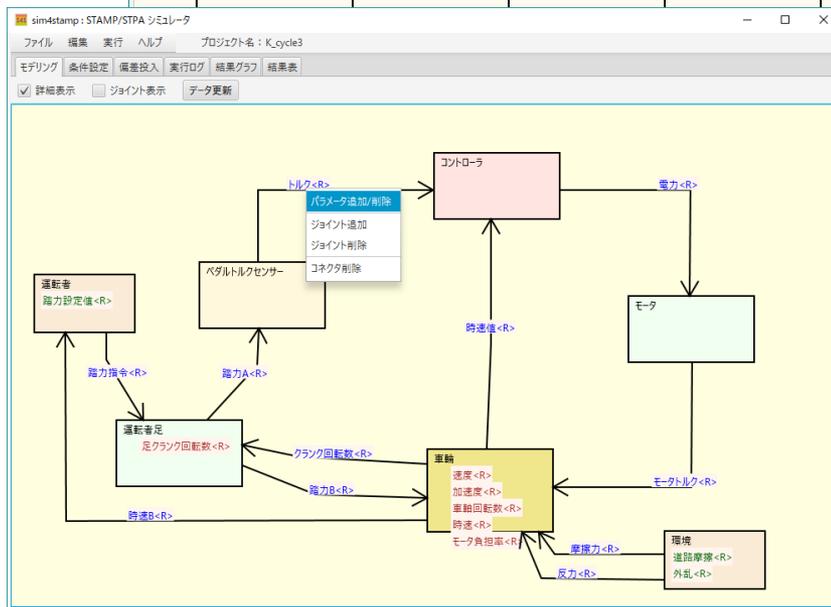
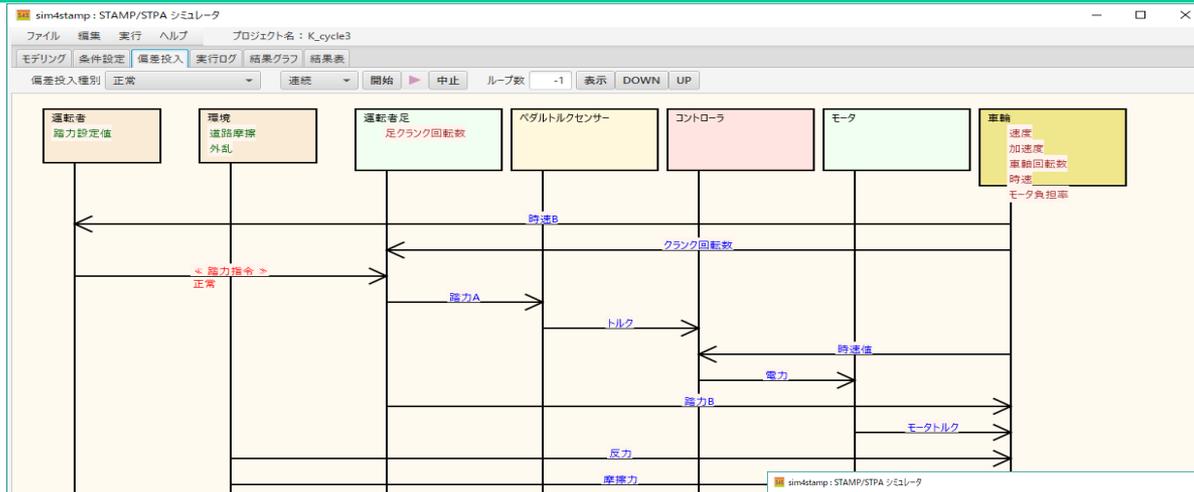
- 人は具体的なもので考える方が考え易い。
--- 抽象的なものだけでは思考が深まらない。
- 「具体論 ⇒ 抽象論」は扱い易いが、反対は困難。
--- 「後知恵」は容易、未知のもの想像は困難。

何か、考えるための「手がかり」が欲しい！

シミュレーションを用いた思考

- ある程度具体化したモデルで、現象を具体的に考える。
--- 考え易い。イメージが湧く。
- ロジックの動きを目に見えるようにしてくれる。
--- ロジックを動かすのに頭を使わなくて済む。
- 素早く、試行錯誤が可能になる。
- 直観をすぐに試せる。
- 変更のコストは最小。

Sim4stamp



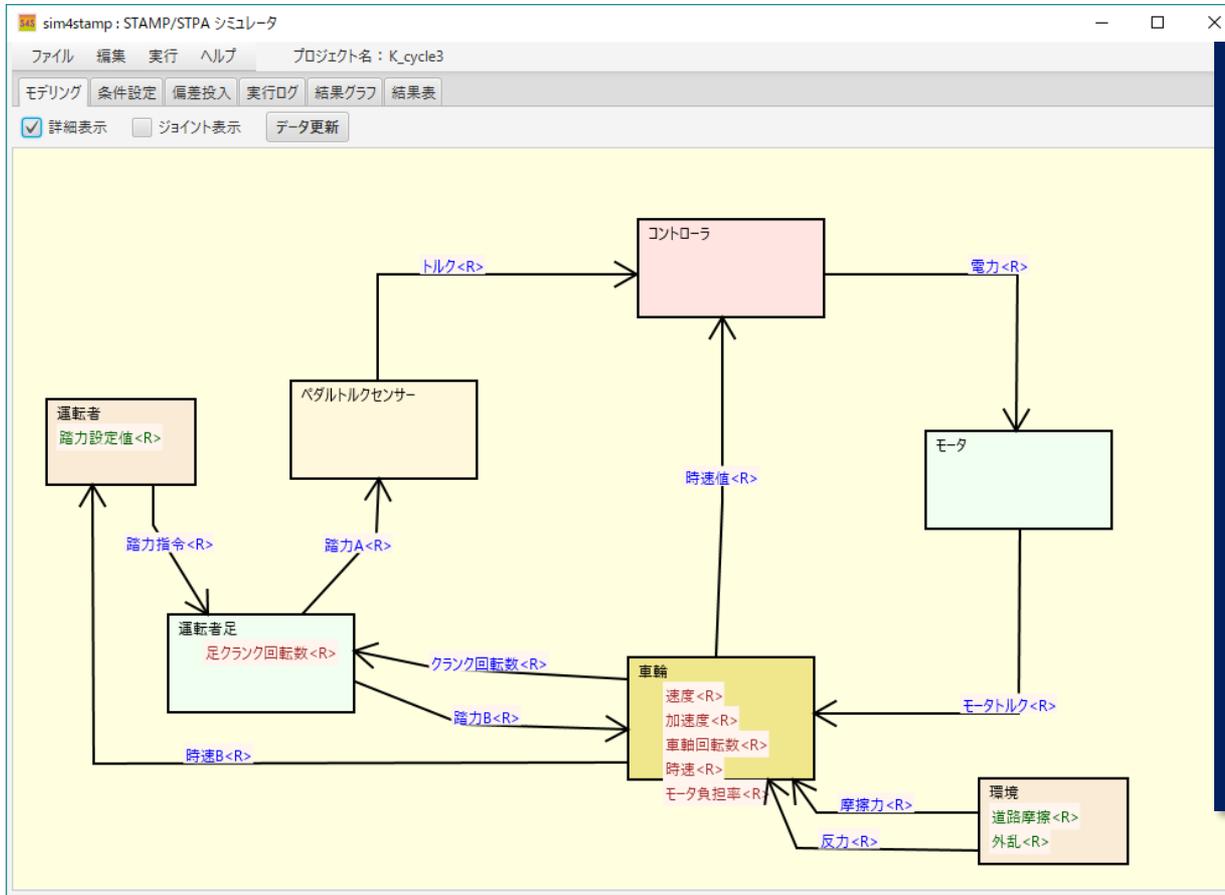
Sim4stamp

- Guiベースのモデリング。
- ロジック実行は「Overture」を使用（VDM++使用）。
--- Guiモデリングから、VDM++ 半自動生成。
- データ入力サポート。
- STAMP/STPA 偏差設定機能。
- シミュレーション結果は表、グラフで表現（制約値付き）、偏差設定なし、ありと比較表示。

注) Overture: Overture Tool, Formal Modelling in VDM
EclipseベースのVDM用のIDE(統合開発環境)
<http://overturetool.org/>

Sim4stamp

Guiベースのモデリング



モデル要素は

- 被コントロールプロセス

- センサ

- コントローラ

- アクチュエータ

- インジェクタ

の5種類

Sim4stamp

VDM++生成後、OvertureによるVDM++ロジック追加作成



Overture IDE

```
1 class 『コントローラ』 is subclass of 『作用素』
2 types
3
4 values
5   アシスト係数 : real = 0.25;
6   最大アシスト速度 : real = 24.0;
7   通常アシスト速度 : real = 10.0;
8   ds : real = 最大アシスト速度 - 通常アシスト速度;
9 instance variables
10
11 operations
12
13 public 『コントローラ』 : () ==> 『コントローラ』
14 『コントローラ』() ==
15 (
16   init("コントローラ", <コントローラ>);
17 );
18
19 public func : () ==> ()
20 func() ==
21 let
22   トルク : real = getData("トルク"),
23   時速 : real = getData("時速値")
24 in
25 (dcl 注入電力 : real;
26  if 時速 < 通常アシスト速度 then
27  (
28    注入電力 := トルク * アシスト係数;
29  )else if 時速 < 最大アシスト速度 then
30  (
31    注入電力 := トルク * アシスト係数 * (1.0 - (時速 - 通常アシスト速度)/ds);
32  )else(
33    注入電力 := 0.0;
34  );
35  setData("電力", 注入電力);
36 );
37
```

モデルからVDM++を
自動生成する

Sim4stamp

VDM++生成後、OvertureによるVDM++ロジック追加作成



```
public 『コントローラ』 :() ==> 『コントローラ』
『コントローラ』() ==
(
  init("コントローラ", <コントローラ>);
);

public func : () ==> ()
func() ==
  let
    トルク : real = getData("トルク"),
    時速 : real = getData("時速値")
  in
    (dcl 注入電力 : real;
     if 時速 < 通常アシスト速度 then
     (
       注入電力 := トルク * アシスト係数;
     )else if 時速 < 最大アシスト速度 then
     (
       注入電力 := トルク * アシスト係数 * (1.0 - (時速 - 通常アシスト速度)/ds);
     )else(
       注入電力 := 0.0;
     );
     setData("電力", 注入電力);
    );
);
```

VDM++スケルトン
自動生成

ロジックを追加して
完成させる

Sim4stamp

データ入力

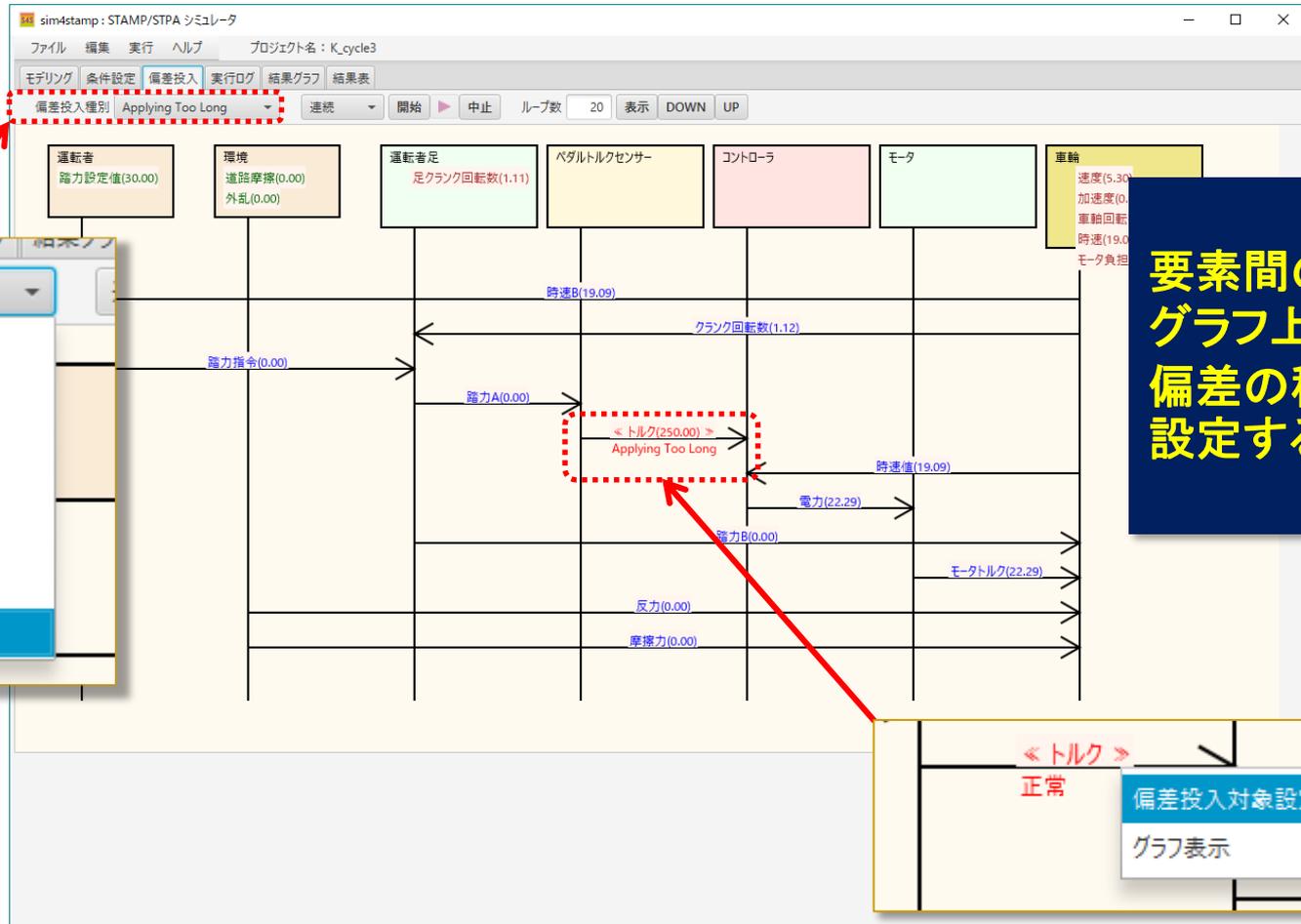
The screenshot shows the 'sim4stamp : STAMP/STPA シミュレータ' window. The '条件設定' (Condition Setting) tab is active. On the left, there are input fields for 'タイトル' (Title: XXXX), 'データ数' (Data Count: 20), and '投入開始位置' (Input Start Position: 0). A '設定' (Settings) button is highlighted with a red dashed box. Below these are '構成要素パラメータ' (Component Parameters) with a table where the '初期設定' (Initial Setting) column has checkboxes for '運転者_踏力設定値', '環境_道路摩擦', and '環境_外乱', all of which are checked. A red arrow points from this table to the '設定' button, which in turn points to a large table on the right. This table is also enclosed in a red dashed box and contains 20 rows of simulation data.

No.	運転者_踏力設定値	環境_道路摩擦	環境_外乱
# 1	0.00	0.00	0.00
2	250.00	0.00	0.00
3	250.00	0.00	0.00
4	250.00	0.00	0.00
5	250.00	0.00	0.00
6	250.00	0.00	0.00
7	250.00	0.00	0.00
8	250.00	0.00	0.00
9	30.00	0.00	0.00
10	30.00	0.00	0.00
11	30.00	0.00	0.00
12	30.00	0.00	0.00
13	30.00	0.00	0.00
14	30.00	0.00	0.00
15	30.00	0.00	0.00
16	30.00	0.00	0.00
17	30.00	0.00	0.00
18	30.00	0.00	0.00
19	30.00	0.00	0.00
20	30.00	0.00	0.00

初期値設定を
マークすると初
期値を設定する
表中に初期設
定欄が出現す
る。

Sim4stamp

シーケンス図から偏差設定



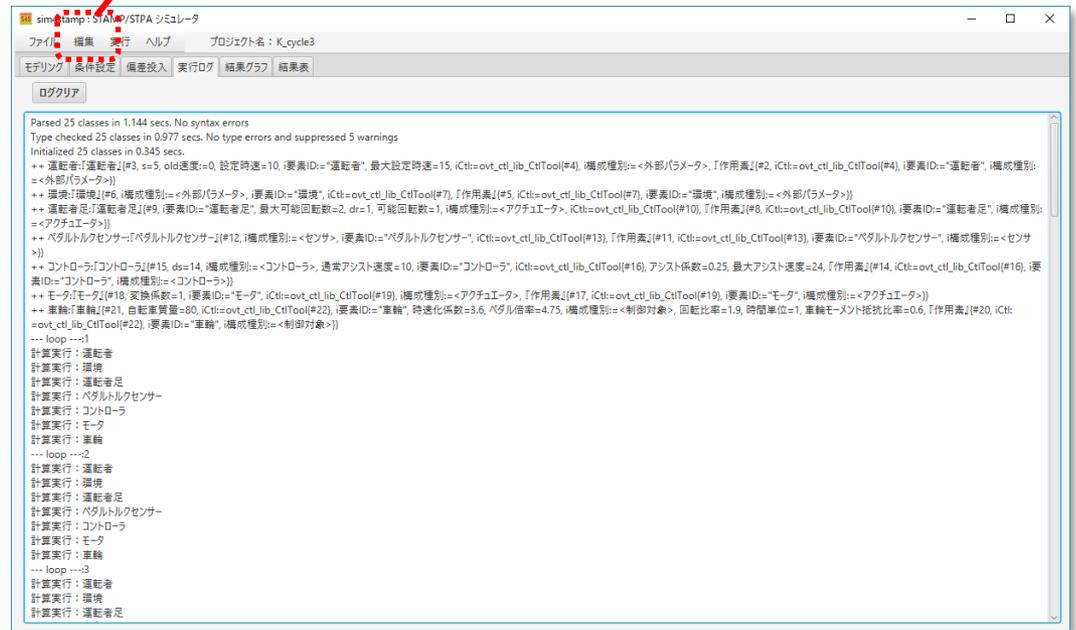
要素間の有向
グラフ上で
偏差の種類を
設定する！

Sim4stamp

シミュレーション実行



Sim4stamp
アダプタ Lib



Sim4stamp

Sim4stamp

結果表示(構成要素データ値グラフ)



正常ケースと偏差投入ケースの比較を行う。

Sim4stamp

結果表示(表)

sim4stamp : STAMP/STPA シミュレータ

ファイル 編集 実行 ヘルプ プロジェクト名: K_cycle3

モデリング 条件設定 偏差投入 実行ログ 結果グラフ 結果表

7 : Applying Too Long

No.	運転者			環境				運転者足				ペダルトルクセンサー		コントローラ		モータ		速度		
	踏力設定値	踏力指令	時速B	道路摩擦	外乱	反力	摩擦力	足クランク回転数	踏力B	踏力A	クランク回転数	踏力指令	踏力A	トルク	トルク	電力	時速値		電力	モータトルク
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	250.00	250.00	3.11	0.00	0.00	0.00	0.00	0.00	250.00	250.00	0.18	250.00	250.00	250.00	250.00	62.50	3.11	62.50	62.50	0.86
3	250.00	250.00	6.13	0.00	0.00	0.00	0.00	0.18	250.00	250.00	0.36	250.00	250.00	250.00	250.00	62.50	6.13	62.50	62.50	1.70
4	250.00	250.00	9.08	0.00	0.00	0.00	0.00	0.36	250.00	250.00	0.53	250.00	250.00	250.00	250.00	62.50	9.08	62.50	62.50	2.52
5	250.00	250.00	11.94	0.00	0.00	0.00	0.00	0.53	250.00	250.00	0.70	250.00	250.00	250.00	250.00	62.50	11.94	62.50	62.50	3.32
6	250.00	250.00	14.49	0.00	0.00	0.00	0.00	0.70	250.00	250.00	0.85	250.00	250.00	250.00	250.00	53.84	14.49	62.50	62.50	4.12
7	250.00	25.40	16.67	0.00	0.00	0.00	0.00	0.85	250.00	250.00	0.97	250.00	250.00	250.00	250.00	42.45	16.67	62.50	62.50	4.91
8	250.00	-83.39	18.52	0.00	0.00	0.00	0.00	0.97	250.00	250.00	1.08	250.00	250.00	250.00	250.00	32.73	18.52	62.50	62.50	5.70
9	30.00	0.00	19.93	0.00	0.00	0.00	0.00	1.08	229.20	229.20	1.17	250.00	229.20	229.20	229.20	22.42	19.93	62.50	62.50	6.49
10	30.00	0.00	20.99	0.00	0.00	0.00	0.00	1.17	208.62	208.62	1.23	250.00	208.62	208.62	208.62	15.16	20.99	62.50	62.50	7.28
11	30.00	0.00	21.80	0.00	0.00	0.00	0.00	1.23	193.16	193.16	1.27	250.00	193.16	193.16	193.16	10.39	21.80	62.50	62.50	8.07
12	30.00	0.00	22.43	0.00	0.00	0.00	0.00	1.27	181.29	181.29	1.31	250.00	181.29	181.29	181.29	7.12	22.43	62.50	62.50	8.86
13	30.00	0.00	22.94	0.00	0.00	0.00	0.00	1.31	172.02	172.02	1.34	250.00	172.02	172.02	172.02	4.81	22.94	62.50	62.50	9.65
14	30.00	0.00	23.34	0.00	0.00	0.00	0.00	1.34	164.68	164.68	1.36	250.00	164.68	164.68	164.68	3.13	23.34	3.13	3.13	6.48
15	30.00	0.00	23.66	0.00	0.00	0.00	0.00	1.36	158.81	158.81	1.38	250.00	158.81	158.81	158.81	1.88	23.66	1.88	1.88	6.57
16	30.00	0.00	23.92	0.00	0.00	0.00	0.00	1.38	154.09	154.09	1.40	250.00	154.09	154.09	154.09	0.93	23.92	0.93	0.93	6.65
17	30.00	0.00	24.14	0.00	0.00	0.00	0.00	1.40	150.25	150.25	1.41	250.00	150.25	150.25	150.25	0.21	24.14	0.21	0.21	6.70
18	30.00	0.00	24.32	0.00	0.00	0.00	0.00	1.41	147.13	147.13	1.42	250.00	147.13	147.13	147.13	0.00	24.32	0.00	0.00	6.76
19	30.00	0.00	24.49	0.00	0.00	0.00	0.00	1.42	144.43	144.43	1.43	250.00	144.43	144.43	144.43	0.00	24.49	0.00	0.00	6.80
20	30.00	0.00	24.63	0.00	0.00	0.00	0.00	1.43	142.03	142.03	1.44	250.00	142.03	142.03	142.03	0.00	24.63	0.00	0.00	6.84

詳細値を表で
表示する。

モデリングから言えること

- 試行錯誤することにより、対象の理解が深まる。
- シミュレーションしてみて、初めて気が付くシナリオがある。
- 既存の思考に引きずられ見逃すことが少なくなる。

まとめ

- 条件を変更し、シミュレーションを繰り返すことにより、いろいろなシナリオを確認できる。
- STAMP/STPA専用シミュレータなので、STAMP/STPA分析のモデリングでは使い易い。
- モデル図がそのまま実行されるので、絵を書けば、接続関係が自動的に生成される。モデル図を修正すると評価順やパラメータが自動的に変更される。
- ロジックはVDM++で書き、Overture IDEを用いるので、インクリメンタルな文法チェックが行われ、デバッグや動作確認がやり易い。

今後の課題

- シミュレーション条件と結果のプリント用へExcel等への保存。

最新のSim4stamp入手先



GitHubリポジトリ :

<https://github.com/KeiTsumuta/Sim4stamp>