# 確率的モデル検査器を用いたFRAM モデル理解の支援

青木 善貴[1,a)]　小形 真平[2,b)]

**概要**：FRAM (Functional Resonance Analysis Method:機能共鳴分析手法) は複雑なシステムの機能の構成と機能間の相互作用をモデル化して分析する手法である．FRAM では，事故は機能間のパフォーマンスの予期せぬ変動の組み合わせ（機能共鳴）から生じ，このパフォーマンスの変動を監視し減衰させることによって防止できるとしている．しかしながら，FRAM モデルの分析は，分析者のドメイン知識に頼った定性的評価でなされ，定量的評価を行う方法は未確立である．そこで本研究では，分析者が FRAM モデルの振る舞いを理解することを支援する目的で，確率を扱えるモデル検査ツール PRISM によりモデルの振る舞いを数値化して表す手法を提案する．本稿では，提案手法を FRAM 分析の事例に適用して，数値化により振る舞いを理解しやすくできること示した．また，変動を想定したモデル修正による振る舞いの数値の変化を追うことにより，端的にパフォーマンスの変動を捉えることができた．

# Support of FRAM model understanding using a probabilistic model checker

***Abstract:*** FRAM is a method to model and analyze the configuration of functions and the interaction between functions of complex systems. In FRAM, an accident results from a combination of unexpected fluctuations in performance between functions. It can be prevented by monitoring and attenuating this change in performance. However, the analysis of FRAM model is a qualitative evaluation that relies on the analyst's domain knowledge, and there is no established method for quantitative evaluation. So, in this research, in order to support the understanding of the behavior of FRAM model, we propose a method to represent the behavior of the model with the model inspection tool PRISM that can handle the probability. In this paper, we apply the proposed method to the case study of FRAM analysis and show that it is easy to understand the behavior by quantification. By comparing the numerical values of the verification results obtained by changing the model, it was possible to catch the fluctuation of the performance.

## 1. はじめに

IoT や CPS は，サイバー空間とフィジカル空間をまたがり，かつ構成要素が多様で相互作用が複雑である．このような相互作用が複雑なシステムの信頼性・安全性を検証する手法については，まだ確立されているとはいえない．

相互作用が複雑なシステムの安全性解析の手法のひとつとして， FRAM(Functional Resonance Analysis Method: 機能共鳴分析手法)[1] がある．FRAM は，システムを構成する機能間の相互作用における関係を六つの側面でモデル化し，想定外の事象の要因を分析できる手法である．FRAM については，2 章で詳述する．

FRAM は，機能を積み上げていくボトムアップによりモデル化を行うため，対象システムの規模に応じてモデルが複雑化しやすい．また FRAM は，定性的評価のための四つのステップ [1] が定義されているが，定量的評価については考慮されていない．定性的評価では，想定外の事象の具体的な発生要因が発見できる，一方，定量的評価では端的な良し悪しの客観的な判断指標が提示できる．したがって，複雑なシステムの安全性向上のためには，双方の評価方法の両立が重要である．分析者が FRAM モデルを理解しやすいようにモデルの可読性をあげる支援を考慮する必要がある．また，定量的評価は分析者間での FRAM モデルの評価を共有する場合にも有効であると考える．

FRAM モデルは，機能内部は状態遷移としてモデル化でき，機能間の相互作用によって状態遷移に関するイベン

[1] 日本ユニシス株式会社
東京都 江東区豊洲 1-1-1
[2] 信州大学
長野県長野市若里 4-17-1
[a)] Yoshitaka.Aoki@unisys.co.jp
[b)] ogata@cs.shinshu-u.ac.jp

トが生じると考えれば，全状態を漏れなく検証できる形式手法の一つであるモデル検査の利用が有効といえる．

本稿では，モデル検査を用いた FRAM モデルの定量的評価に向けて，まずは機能間における相互作用の振る舞いの理解を支援するために，モデルの振る舞いを数値化する手法を提案する．さらに，提案手法を参考文献にある FRAM 分析の事例に適用して提案手法の有効性の確認を行う．本研究は，FRAM モデルの振る舞いの確率的モデル検査器による数値化とモデル生成ツールによる PRISM モデルの自動生成を実現した．

## 2. FRAM について

FRAM は Hollnagel [1] により提唱された想定外の事象や事故が発生したとしても，その影響を最小にすることを目的とするレジリエンスエンジニアリングの考え方に基づく分析手法である．FRAM は，機能に基づいてシステムをモデル化し，パフォーマンスの変動の原因を特定しそれを管理，抑制，または監視するための手段を決定するために使用される四つのステップで構成されていいる．

ステップ 1 では，必要な全てのシステム機能を定義する．すべての機能は，機能を表すモデル，六角形のセル（**図 1**）とその 6 つの側面の形（**表 1**）で説明される．

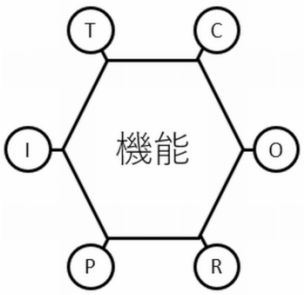


**図 1** FRAM セル
**Fig. 1** FRAM cell

**表 1** FRAM の六つの側面
**Table 1** Six Aspects of FRAM

| 側面 | 役割 |
|---|---|
| I:入力 | 機能が処理するもの，動作のトリガー |
| P:前提条件 | 機能を実行する前に満たすべき条件 |
| C:制御 | 機能がどのようにモニター，制御されるか |
| R:資源 | 機能を実行するのに必要なもの |
| T:時間 | 機能の実行に影響を与える時間制約 |
| O:出力 | 機能の処理結果 |

ステップ 2 は，FRAM モデル内の機能の変動性を識別する．人間を含む機能のパフォーマンスは大きく変動する傾向があるが，技術的機能は通常，時間の経過とともに安定する傾向がある．ステップ 3 では，パフォーマンスの変動がシステム全体にどのように広がる可能性があるかを確認する．FRAM モデルは制御の流れを表すものではなく，意味論に基づく機能間の関係を記述している．そのため分析者はシステムの振る舞いについて意味論に基づく解釈が可能となり，変動を読み取ることができる．

最後のステップであるステップ 4 は，パフォーマンスの変動を監視，管理，または排除できる方法を提案する．ステップ 2 と 3 の結果に基づいて，変動性を最適に監視および管理する方法を分析する．

FRAM では，想定される作業 WAI(work as imagined) と実際の作業 WAD(work as done) をモデル化し，それぞれの分析結果から成功要因を導き出す方法がよく用いられる．

また FRAM においてパフォーマンスの変動を評価する指標として以下の五つがある [2], [7].

(1) バッファ容量
　バッファ容量は，パフォーマンスが根本的に低下することなくシステムが吸収または適応できる度合い
(2) 柔軟性
　柔軟性は，外部の変化や圧力に応じてシステムが再構築される度合い
(3) マージン
　特定の性能境界に対して，システムがどの程度厳密に動作しているか，どれだけ不安定であるかの度合い
(4) 耐久力
　圧力が適応能力を超えた時に，システムが境界付近で適切な劣化または急速な崩壊の線に沿ってどのように動作するかの度合い
(5) クロススケールインタラクション
　あるスケールで定義されたシステムが他のスケールで定義されたシステムからの影響に依存するかの度合い

### 2.1 FRAM のモデルの評価の難しさ

FRAM は六つの側面を用いてモデルを作成するため，振る舞いが捉えにくくなる一面もある．**図 2** はやや複雑な FRAM モデルの例である．

四つの機能があり，機能 1，機能 2 の出力が機能 4 の入力へ入り，機能 3 の出力が機能 4 の前提条件に入っている．基本的には入力が機能実行のトリガーとなるが，複数の入力がある機能 3 の合流部分 (図 2 機能 3 の I:入力部分) については，どのような入力条件で機能 4 が実行の状態になるかの考慮が必要である．その他にも前提条件の有無による振る舞いの違いなどの考慮も必要である．

こうしたモデルの解釈は，分析者のドメイン知識やスキルへの依存度が高いため，他の分析者に正確に伝えることは難しい．また，FRAM はシステムを構成する機能を積み上げていくボトムアップの手法であるため，機能を積み

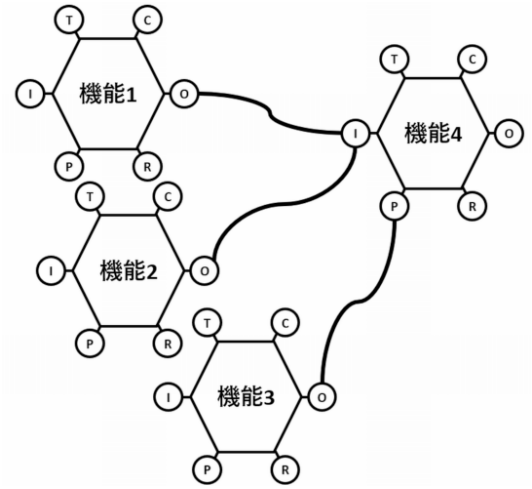


図 2　FRAM サンプルモデル
**Fig. 2**　FRAM Sample Model

上げた後に，複雑な振る舞いから問題となるポイントを指摘することも難しい．

## 3. 本研究の目的と課題

### 3.1 研究目的

本研究の目的は，モデル検査を用いた FRAM モデル理解の支援である．FRAM モデルには機能共鳴のによる変動の理解が難しいという問題がある．変動を理解するには，ある時点 p からある時点 q までの間で発生しうる振る舞い（以降，過渡的な振る舞い）を漏れなく検証することが重要であるが，分析者が漏れなく行うことは難しく，これができれば評価の精度向上が望める．また，振る舞いの数値化により客観的な評価のための指標を提供し，システムの振る舞いの評価および評価の共有の容易化を図る．ただし，これを実現するためには解決すべき以下の課題がある．

### 3.2 検証に関する課題

モデル検査とは，形式手法の一つであり，システムの仕様の正当性を数学的に検証するものである [3]．システムの振る舞いを数学的モデルで定義し，満たすべき性質を表わす時相論理式（検査式）でその充足関係を検証する．もし性質が満たされない場合はその満たされない状態に至る状態遷移（反例）が出力される．満たすべき性質は時相論理式で記述される．

検証できる性質は，「到達可能性」，「安全性」，「公平性」，「活性」に限られる [4]．「到達可能性」の意味は「いつか p が成り立つ」であり，「安全性」の意味は「p が成り立つことは決してない」であり，「公平性」の意味は「p が無限回成り立つ」であり．「活性」の意味は「p が成り立てばいつか q が成り立つ」である．

上記の性質を用いた検証は，時間軸を無限にとり，最終的に収束する状態を検証するものであり，反例はその最終的な状態に収束できないことがある場合に，その過程を示しているにすぎない．最終的な状態の検証も重要ではあるが，FRAM モデルで検証したい特性は，機能共鳴による過渡的な振る舞いであり，上記特性だけで明らかにすることは難しい．

そのため本研究では，確率モデル検査器である PRISM[5] を用いる．PRISM は，ランダムまたは確率的な振る舞いを示すシステムの形式的モデリングおよび分析のためのツールである．PRISM は上記の時相論理をサポートすると同時に，モデルの定量的特性の自動分析もサポートする．PRISM を用いれば，FRAM モデルの振る舞いを任意の検査式で定量的に検証できると考える．

### 3.3 検査モデル生成に関する課題

提案手法を実現するためには，六つの側面による複雑な関連をもつ FRAM モデルを解釈してモデル検査ツール PRISM のモデルを生成する必要がある．FRAM モデルに対応する PRISM モデルを手作業で作成することは可能であるが，FRAM モデルの複雑さやモデルに何度も修正が入ることを考えると手作業による作成は現実的とはいえない．

また FRAM モデルの PRISM モデルへの置き換えは，FRAM の六角形セル間の関連を PRISM の状態遷移のコマンドへの変換となるで，定型的な処理の繰り返しとなる．そのため本研究では，FRAM モデルから PRISM のモデルを生成するツールを用意することとした．

## 4. モデル検査ツール PRISM について

PRISM[5] は確率的モデル検査器であり，確率的な挙動を示すシステムのモデリングと分析のツールである．検査対象のシステムは，時相論理に基づいた PRISM の特性仕様言語で記述され，検査特性は時相論理式で記述し，PRISM により自動的に検証される．PRISM はいくつかのタイプの確率モデルをサポートしている．以下の特徴がある．

- 仕様として書かれた状態へ到達できる確率が得られる
- 状態及び遷移に任意の reward（報酬）を定義し，検証結果としてその累積値が得られる
- 時間を扱えるため，モデルの過渡的な状態の検証結果が得られる
- 確率 (rate) で遷移の発生頻度を調整できる

次に PRISM モデルの例を示す．**図 3** は，0⇒ 1⇒ 0 もしくは 0⇒ 2⇒ 0 で循環する状態遷移のモデルである．

PRISM モデルは複数のモジュールで構成され，ガードと一つまたは複数の更新処理のコマンドで構成される．この状態遷移モデルを PRISM で記述すると**図 4** になる．先頭の dtmc（Discrete-time Markov chains : 離散時間マルコフ連鎖）は確率モデルのタイプである．

変数 pi は確率を表す，0.5 を設定する．変数 state の値は 0,1,2 のいずれであり初期値は 0 である．一つ目のコマ

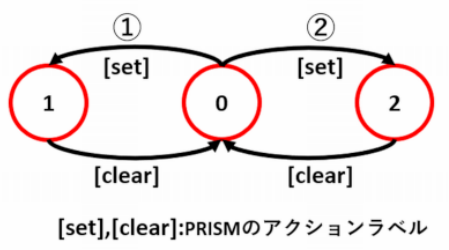


図 3　状態遷移モデル
**Fig. 3**　State Transition Model

```
dtmc
const double pi;
module StateTran
  state:[0..2] init 0;
  [set]state=0 -> pi:(state'=1) + 1-pi:(state'=2);
  [clear]state!=0 -> (state'=0);
endmodule
rewards "State"
  [clear] state=1 :1;
endrewards
```

図 4　PRISM コード
**Fig. 4**　PRISM Code

ンドの意味は，「state が 0 ならば，確率 pi で state は 1 になるか，確率 1-pi で 2 になる」である．二つ目のコマンドの意味は，「state が 0 でないならば sate は 0 になる」である．これらのコマンドの先頭の [set]，[clear] はアクションラベルであり，異なるモジュール間で同名のアクションラベルのコマンドは同期をとって実行される．

以下で，上記で記述した PRISM の特徴について，検証例を示す．検査式の記述にある「P =?」は確率を求め，「R=?」は reward（報酬） の累積を求めるものである．「C<=t」は，t 時間単位が経過するまでに累積された reward（報酬）に対応する．「F<=t」は，t 時間単位を上限とした確率に対応する．

(1) ある状態に到達できる確率の検証
`P =? [F state= 2]`
state がいつか 2 になる確率を求めている．いつか必ずいつか実現するため実行結果は 1.0 となる．

(2) ある時点に至るまでの過渡的な状態の検証（確率）
`P =? [F <= 2 state= 2]`
state が 2 単位時間内で 2 になる確率を求めている．確率モデルタイプ dtmc の場合，1 単位時間は 1 遷移にあたる．state が 2 単位時間内に 2 になるのは，図 3 の①へ進まず②を進んだ場合だけなので，pi=0.5 なので実行結果は 0.5(50%) となる．

(3) ある時点に至るまでの過渡的な状態の検証 （`reward`）
`R =? [C <= 100]`
100 時間単位以内の reward（報酬）の累積値を求めており，実行結果は 50.0 となる．遷移は [set] と [clear] しかなく，これらは均等に発生するため，100 時間以内の [set] の遷移は 50 回となる．

(4) 状態遷移の発生頻度の調整
pi を 0.5 から 0.75 にすると (3) の結果は，50.0 から 75.0 になり，状態遷移の発生頻度が変化する．

## 5. 提案手法

### 5.1 検査モデル生成の基本的な考え方

検査モデルは，機能ごとに PRISM のモジュールを定義する．機能の目的は何らかの状態変化をもたらすことであるので出力 (O) は機能を実行した結果の状態であり，出力の接続先機能が実行前の状態である．実行前の状態は，機能が発火できるかを表せればよいので，二つの状態（待機中，実行中）で表せる（**表 2**）．また各モジュールの機能自体の発火状態も，変数として検査モデルにもたせる．

表 2　六つの側面の状態
**Table 2**　State of Six Side

| 状態名 | 状態値 |
|---|---|
| 機能自身 | 待機中，実行中 |
| I:入力 | 待機中，実行中 |
| P:前提条件 | 待機中，実行中 |
| C:制御 | 待機中，実行中 |
| R:資源 | 待機中，実行中 |
| T:時間 | 待機中，実行中 |
| O:出力 | 待機中，実行中 |

出力 (O)-入力 (I) 間の状態遷移の考え方は，「ある機能の出力 (O) が待機中から実行中になり（送信），その接続先の機能の入力 (I) が待機中から実行中になり（受信），出力 (O) は待機中に戻る」となる．他の側面も同様である．

**図 5** に FRAM モデルの例を示す．機能 1 と機能 2 の二つの機能があり，機能 1 の出力が機能 2 の入力に入り，機能 2 が実行中になる例である．状態遷移の過程は次の通りである．一つの機能は基本的に「受信」「発火」「実行」「送信」の順番で動作する．

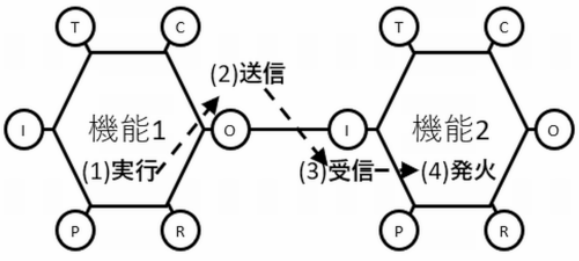


図 5　PRISM 変換時の基本的な遷移の考え方
**Fig. 5**　Basic Transition Concept in PRISM Transformation

(1) 実行：機能 1 自体が待機中から実行中になる
(2) 送信：機能 1 の出力 (O) が実行中になる
(3) 受信：機能 2 の入力 (I) が実行中になる

機能 1 の出力 (O) が待機中になる
機能 1 自身が実行中から待機中になる
(4) 発火：機能 2 自身が待機中から実行中になる
機能 2 の入力 (I) が待機中になる

### 5.2 モデル生成ツールの基本的な考え方

我々は IoT・CPS のアーキテクチャをモデリングする手法として TORTE[6] を提案している．TORTE が提供するエディタでは，六つの階層（要求/監視/制御/データ転送/エネルギー供給/使用）に分けてモデルを記述する．このような方法はモデルの見た目上の複雑さを軽減できる．そのため，本研究では FRAM の側面に対応させて記述した TORTE のモデルから PRISM のモデルを生成することとした．

TORTE では，UML クラス図記法を応用してアーキテクチャエンティティをクラスで表し，エンティティ間の関係を関連で表す．これらを階層ごとに図を分けて描く上では，登場するエンティティは図間で整合させ，多量なエンティティ間を関連線で結ぶ必要があり，手間がかかる．そこで TORTE では，**図 6** に示すように，常に最新のエンティティ集合を表示し，階層ごとの関係を容易に与えられるよう表形式のエディット支援ツールを提供しており，分析者が本質的な分析作業に集中しやすい環境を提供している．なお，モデル全体を俯瞰できるように，全ての階層の関連を一つの図に表示する機能も提供する．

FRAM では，関連は必ず“出力”が起点となるため，TORTE では終点となりうる五つの階層（入力，前提条件，制御，時間，資源）に分けて関連をモデル化する．

これらを踏まえ，**表 3** に TORTE 要素と PRISM 要素の対応関係を示す．この関係に基づき，自動変換では，TORTE による FRAM モデルを入力とし，検査式を除く PRISM モデルを出力とする．表 2 に基づき，各状態変数がとりうる状態数は 2 状態（待機中，実行中）であり，自動生成直後の初期値は一律に待機中を表す値となる．

**表 3** TORTE 要素と PRISM 要素
**Table 3** TORTE and PRISM elements

| TORTE | PRISM |
|---|---|
| クラス | モジュール，ならびに機能自身の状態を表す状態変数 |
| 関連<br>（有向） | 送受信の状態を表す変数．<br>変数は方向性を区別して定義する．たとえば，<br>機能 A 出力 (O)→ 機能 B 入力 (I) の関連があった場合，<br>機能 A は入力 (I) への出力 (O) の送信状態の<br>変数（A_f_input_active_B）を持ち，<br>機能 B は入力 (I) における受信状態の<br>変数 (B_f_input_passive_A) を持つ． |

モジュールの動作を表すコマンドは，5.1 節の「受信」「発火」「実行」「送信」のロジック定義に基づき生成する．

対応関係にある送受信に対し，状態遷移を同期させるためにアクションラベル を用いるが，つぎの命名規則により実現する．なお，命名規則は状態遷移の位置づけを理解容易にするものでもある．

(1) 「受信」アクションラベル：Sync+送信側モジュール名+関係+受信側モジュール名・・・（辞書順にあるだけ繰り返す．）
(2) 「発火」アクションラベル： Fire+自モジュール名
(3) 「実行」アクションラベル： Exec+自モジュール名
(4) 「送信」アクションラベル：Sync+送信側モジュール名+関係+受信側モジュール名・・・（辞書順にあるだけ繰り返す．）

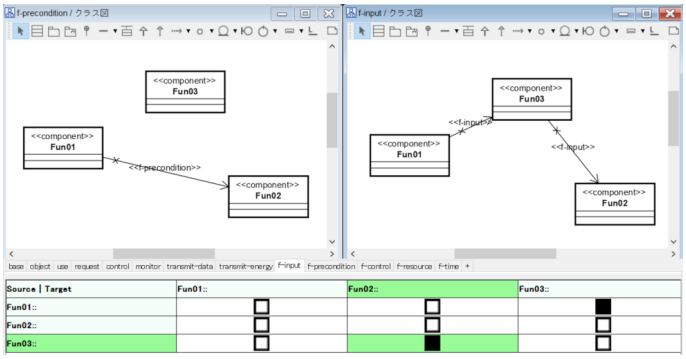


**図 6** 拡張 TORTE ツールのビュー
**Fig. 6** Extended TORTE tool views

提案手法では，全ての機能が画一的な順序で処理が実行されることを想定している．これは分析者に依らない系統的な解析を軽量に実現する一手段となる．自動生成直後の確率は全て 1 と設定し，reward（報酬）は設定していない．

検査式や確率，reward（報酬）は，分析者が分析目的に応じて細かく手動で調整する余地があり，合理的な一つの設定値を究めることは重要でないと考えており，自動生成の対象とはしていない．

### 5.3 検証方法の基本的な考え方

FRAM モデルの分析の目的は，ある時間内に取りうる過渡的なシステムの振る舞いを変動として分析することである．モデル検査で扱える時相論理だけでは，システムの過渡的な振る舞いは検証できない．

PRISM は確率と reward（報酬）という定量的な特性が扱え，それらの検証に時間特性も付加できるため，これを利用すれば過渡的な振る舞いも検証できると考える．

また，FRAM はその名が示すとおり機能の共鳴を扱うため，使用する確率モデルタイプは，dtmc ではなく ctmc（ContinuousTime Markov chains：連続時間マルコフ連鎖）とする．ctmc は dtmc が確率的に遷移を選択するのと異なり，rate のラベル付けによる競合により遷移を選択するため，FRAM モデルの分析により適していると考える．

### 5.3.1 reward を用いる検証

遷移の reward を時間経過で累積した値により，指定した時間内の FRAM モデルの振る舞いを確認できる．FRAM は六つの側面を用いてモデルを作成するため，複雑な関係性も記述できる一方振る舞いが捉えにくくなる一面もある．

**図 7** は三つのループが重なる FRAM モデルの例である [8]．五つの機能に対して三つのループが組み合わさっている．Func2 でいったん分かれた後，Func3 で合流して元の Func1 に戻る．合流部分 (図 7 の Func3 の I:入力部分) の同期については特に考慮が必要である．野本ら [8] は，ループの共鳴的な影響を解釈してループ 1 とループ 2 が同期を取って動作できるかどうかは，ループ 3 の実行タイミングによって決定されると分析している．

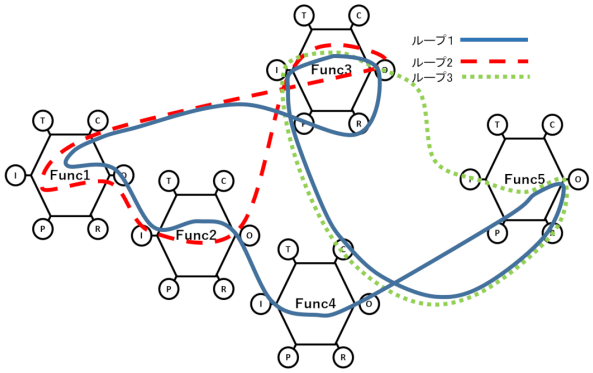


**図 7** 三つのループをもつ FRAM モデル
**Fig. 7** FRAM model with three loops

この FRAM モデルに対して，提案手法を適用する．各機能の振る舞いは以下のように定義した．

- Func1 は，制御 (C) を受信すると発火する．
- Func2 は，Func3 と Func4 へ個別に出力 (O) を行うが，必ず両方向に送信を行う．
- Func3 は，Func2 からの入力 (I) への受信と Func5 からの入力 (I) への受信が揃った時に発火する．機能 1 と機能 5 へ出力 (O) を送信する．二つの出力 (O) は個別に行われるが，必ず両方向に送信を行う．
- Func5 は，入力 (O) と事前条件 (P) が揃った時に発火する．

上記の定義を満たす PRISM モデルをモデル作成ツールにより生成した．**図 8** は，各機能を置き換えたモジュール内のアクションラベルの同期の関係を表した図である．アクションラベルは 5.2 の (1) から (4) の定義に基づいて作成されている．

同名のアクションラベルは条件が揃った時に同期して実行される．図 8 の①の同期を表すアクションラベルのコードを**図 9** に示す．アクションラベル [SyncFun03InpFun01] は両方の条件が揃った時点で同時に実行される．Func3 側の出力 (O) は待機中になり，Func1 側の入力 (I) は実行中

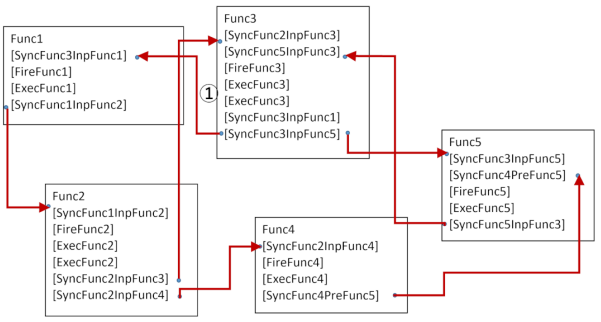


**図 8** モジュール内のアクションラベルの同期の関係
**Fig. 8** Synchronization relationship of action labels in module

```
module Func1
[SyncFunc3InpFunc1]
func1_f_input_passive_func3=1 & func3_f_input_active_func1=0
  -> (func1_f_input_passive_func3'=0); //待機中に戻る
```

```
module Func3
[SyncFunc3InpFunc1]
func3_f_input_active_func1=0
  ->(func3_f_input_active_func1'=1) ;//実行中になる
```

**図 9** PRISM コード
**Fig. 9** PRISM Code

になり，遷移が行われたことになる．

各ループの 100 単位時間内の reward を求める．Loop2 の reward の定義は**図 10** になる．各遷移が起きれば無条件に reward に 1 が加算される．Loop2 の場合一周すると reward は 3 となる．reward は，ループ 1 が 31.95（6.39 周），ループ 2 が 19.37（6.46 周），ループ 3 が 12.18（6.09 周）となった．

```
rewards "Loop2"
   [SyncFun03InpFun01]true:1;
   [SyncFun01InpFun02]true:1;
   [SyncFun02InpFun03]true:1;
endrewards
```

**図 10** Loop2 の reward 定義
**Fig. 10** Loop2 reward definition

Func3 の入力 (I) への受信が揃った時点で Func3 は発火する．Func2 出力 (O)⇒Func3 入力 (I) の遷移の reward は 6.59 であり，Func5 出力 (O)⇒Func3 入力 (I) の遷移の reward は 6.19 である．従って Func3 の発火は，reward6.19 の遷移により決定される．

この遷移は Func5 の出力 (O) であり，Func5 の発火は Func4 出力 (O)⇒Func5 事前条件 (P) と Func3 出力 (O)⇒Func5 入力 (I) が揃ったとき発生する．Func4 出力 (O)⇒Func5 事前条件 (P) の遷移の reward は 6.39 であり，Func3 出力 (O)⇒Func5 入力 (I) の遷移の reward は 5.99 である．Func5 の発火は rewrd5.99 の遷移により決定される．この遷移はループ 3 に属している．従ってループ 1 とループ 2 の関係は，ループ 3 により決定されるといえる．

本提案手法による FRAM モデルの定量的評価から，野本ら [8] と同じ分析結果を導くことができた．

#### 5.3.2 確率を用いる検証

特定の時間内にある状態に到達できるかの確率を求めることによりモデルの振る舞いが分析できる．FRAM モデルに修正を加えた場合に修正の効果を確認できる．

先行研究 [9] は，鉄道沿線で発生した火災に対する電車側対応について WAI と WAD（**図 11**）の FRAM モデルを作成し，本提案手法を適用した．この事故では，緊急停止ボタンで電車は停止したが，運転手は停車位置が火災現場から近く危険であることに気づかず，第三者からの指示で移動を行っている．WAD では第三者からの指示で電車の走行開始が可能なように変更した．50 単位時間以内に，緊急停止ボタンが押されてから，再度電車が走行できる確率を求めた．再発進できる確率は，WAI が 53%から WAD が 99%となり，少なくとも変更前後の期待される効果（特定時間内に再発進できる可能性）の差異が確認できた．

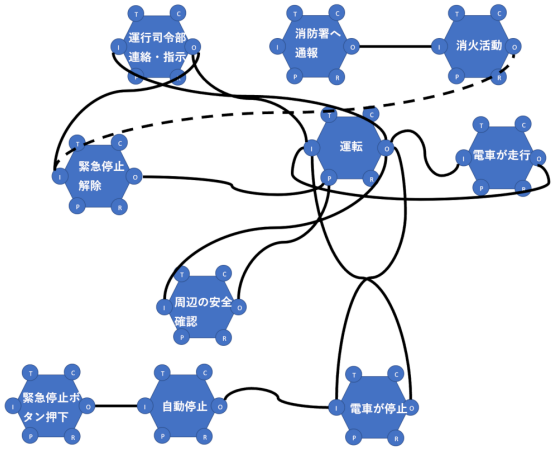


**図 11** 鉄道沿線火災 FRAM モデル　WAD
**Fig. 11** Railroad Fire FRAM Model WAD

## 6. 適用事例

適用事例は航空交通管理システム（Air Traffic Management System 以降 ATM) の下で，RVSM (Reduced vertica separation minimum：航空機間の垂直方向の距離が 1000 フィートの意味）の条件で飛行した航空機が，実際に起こした衝突事故を FRAM で分析した事例に対して，提案手法を用いて追試する [7].

### 6.1 事故の概要

航空機 GLO1907 便（ボーイング 737）と航空機 N600XL（エグゼクティブジェット）が，2009 年 9 月 29 日にアマゾンの上空で衝突事故を起こした．事故はブラジル航空交通管制部（Area Control Centers 以降 ACC）とアマゾン ACC の監視が交差する空域で発生した．衝突の結果，GLO1907 は墜落し，N600XL はダメージを受け着陸した．

N600XL は，サン・ジョゼ・ドス・カンポス空港を離陸し，マナウスのエドワルド・ゴメス国際空港へ向かった．N600XL のフライトプランでは三つの異なる巡航高度があり，フライトレベル 370（37,000 フィート）⇒ フライトレベル 360⇒ フライトレベル 380 の順で指示されていた．GLO1907 便は，ブラジリア国際空港へ向けてマナウスのエドワルド・ゴメス国際空港を離陸した．フライトレベルは 370 で，衝突までのこの高度が維持された．

本来ならば 1000 フィートの高度差ですれ違えるはずだったが，N600XL のトランスポンダが停止しており，パイロットがそれを認識できなかったことや，ACC も N600XL の高度がフライトプランと違っていることを認識できなかったこと等が重なり，衝突事故が起きた．

### 6.2 ATM のシステム構成

ATM の主なシステム構成は以下のとおりである．

- ATC（Air Traffic Control）
  航空管制のこと．
- パイロット
- 管制官
- 管制官とパイロットの間の通信機器
- レーダシステム
  主監視レーダ（PSR）と二次監視レーダ（SSR）がある．SSR は航空機のトランスポンダの動作が前提．
- TCAS（Traffic alert and collision avoidance systems）
  他の航空機のトランスポンダ信号に基づく衝突回避システムである．パイロットに回避指示を行う．

### 6.3 事故の原因

事故が起きた主な原因は以下のとおりである．

- パイロットは，フライトレベル 370 でマナウスに向かったので，変更指示がない限り高度を維持するべきであると認識しており，その後通信の不調により高度変更の指示が受けられなかった
- N600XL のトランスポンダの停止により，TCAS が動作しなかった，またパイロットがそれを認識できなかった（システムの警告なし）
- N600XL のトランスポンダの停止により，二次監視レーダの高度が不正確だったため管制官が危険を認識できず，警告を出せなかった（システムの警告なし）

上記の原因により，パイロットも管制官も，トランスポンダ信号なしの飛行を認識できず衝突事故が発生した．

### 6.4 FRAM 分析

分析は，「ATM の基本機能」，「離陸機能」，「飛行制御機能」に対して行われた．FRAM の評価特性ごとに，以下の分析結果が記述されているが，これらの結果を FRAM モ

デルからどのように読み取ったかの理解は難しい．

#### 6.4.1 バッファ容量

パイロットは飛行開始以来，制御ループからはずれており，管制官からの指示を待っていた．管制官は利用可能な情報からは衝突状況を知覚することができなかった．

#### 6.4.2 マージン

この空中衝突では，通例の通信などルールや手順に記載されているパフォーマンスとは異なる予想外のパフォーマンス変動によってマージンが時間とともに変化している．マージンは減少する傾向を示した．

#### 6.4.3 柔軟性

ＡＴＭシステムは非常に小さな柔軟性を示した．RVSM 空域で誤った航空路で，どの ACC からも認識されず，制御されずに航空機が飛行できた．

#### 6.4.4 耐久力

この事故は調整の背後にある理論的根拠を理解し，パフォーマンスの望ましくない変動を減衰させるためにシステム機能の継続的なパフォーマンス監視の必要がある．

#### 6.4.5 クロススケールインタラクション

パイロットがフライトプランや地域の航空チャートを知らないと，彼らは制御ループから外れ，システムはクロススケールのインタラクションの機会を逃す．N600XL のセクタを変更時に，制御ループを閉じるクロスチェックの機会が失われた．

### 6.5 提案手法の適用対象

「飛行制御機能」の FRAM モデル（**図 12**）に提案手法を適用する．RVSM 下を TCAS とレーダシステムによる航空管制により，安全に飛行するための機能の連携を記述したモデルである．点線の部分はパイロットと管制官が行う通信の制御ループである．

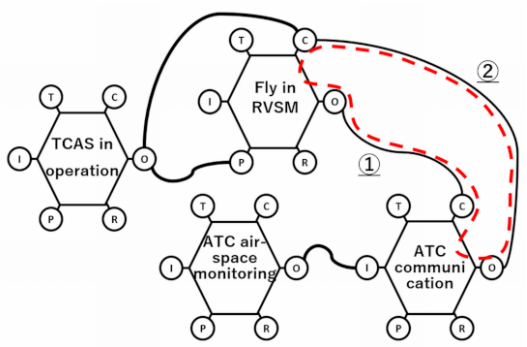


**図 12** 飛行制御機能　FRAM モデル
**Fig. 12** Flight control function FRAM Model

図 12 に記述されている機能は以下のとおりである．

- TCAS in operation
  他の航空機トランスポンダ信号の自動検出に基づく衝突回避システムである．
- Fly in RVSM
  RVSM 下で航空機を飛行させる．
- ATC air-space monitoring
  飛行計画とレーダシステムに基づいて空域監視する．高度を正確に計測する二次監視レーダはトランスポンダに依存する．
- ATC comunication
  管制官 - パイロット間の人間中心のシステム，パイロットへの命令送信により，RVSM を飛行する航空機を制御する．適切な空間監視に依存する．

また FRAM モデル図には，各機能の側面に対して注釈が記述されていた．以下に主なものを**表 4** に記す．規則，手順，技量などは変動に影響を及ぼす可能性が低いと判断して今回は省略する．

**表 4** 飛行制御機能　FRAM 側面の注釈
**Table 4** AFlight control function FRAM side annotation

| 機能 | 側面 | 内容 |
|---|---|---|
| TCAS in operation | 入力 | トランスポンダ信号の受信 |
| | 事前条件 | トランスポンダが動作中 |
| Fly in RVSM | 制御 | パイロットへ管制官から通信 |
| | 事前条件 | トランスポンダが動作中 |
| ATC air-space monitoring | 入力 | 主監視レーダ信号の受信 |
| | 入力 | 二次監視レーダ信号の受信 |
| ATC communication | 制御 | 管制官へパイロットから通信 |

FRAM モデルにおける機能連携および機能側面への注釈により，各機能のについて以下の条件があることが言える．

- 機能「TCAS in operation」には，トランスポンダの信号の受信および動作中であることが必要である．
- 機能「ATC air-space monitoring」には，主および二次監視レーダの信号が必要である．
- 機能「ATC comunication」には，パイロットのからの通信もしくは，機能「ATC air-space monitoring」からの出力が必要である．
- 機能「Fly in RVSM」には，管制官からの通信，トランスポンダが動作中であること，機能「TCAS in operation」および機能「ATC comunication」からの出力が必要である．

上記の内容を考慮して，PRISM モデルの生成を行う．

### 6.6 PRISM モデルの生成

図 12 の FRAM モデルを基に，5.2 で説明したモデル生成ツールを用いて作成した．図 12 の FRAM モデルは，一本の入力 (I) の遷移，三本の制御 (C) の遷移，一本の前提条件 (P) の遷移で構成されている．モデル生成ツール上でこれらの遷移を定義する．制御 (C) の定義が**図 13** となり，入力 (I) の定義が**図 14** となり，前提条件 (P) の定義

が図 **15** となる．図 13 から図 15 までを統合して階層構造に置き換えたイメージは，図 **16** となる．各側面内の関係が明確になり，他の側面にどのように影響するのかが分かる．これらの定義を基に PRISM モデルを自動生成した．

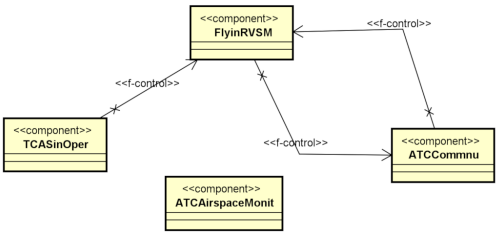


図 **13**　エディタ上の定義 制御 (C)
**Fig. 13**　Editor definition control

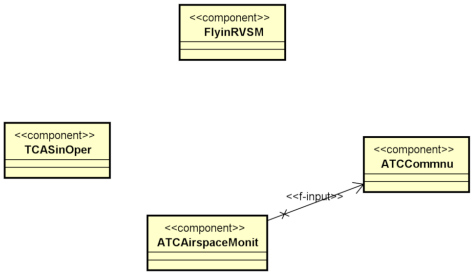


図 **14**　エディタ上の定義 入力 (I)
**Fig. 14**　Editor definition input

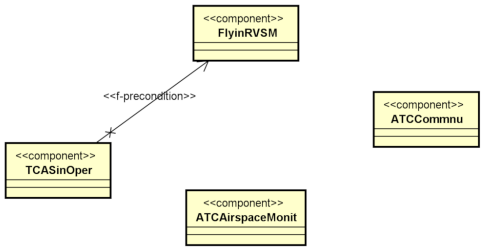


図 **15**　エディタ上の定義 前提条件 (P)
**Fig. 15**　Editor definition precondition

自動生成直後のモデルでは，機能に複数の受信がある場合は，全ての受信が揃った時点で機能が発火するように定義されている．そのため，6.5 節で示した各機能の実行制限については，手動で PRISM モデルに修正を入れ調整を行った．また，表 4 にある注釈条件が満たされる確率が筆者が想定した値になるように，各機能ごとに ctmc による rate を考慮して設定した．各機能は各々の注釈条件が満たされないと実行されない．各モジュール内のアクションラベルの同期の関係は，図 **17** になる．関連付けされている同名のアクションラベルは，条件が揃った時同時に実行され，接続先のモジュールへ状態を渡す．図 17 の①②の同

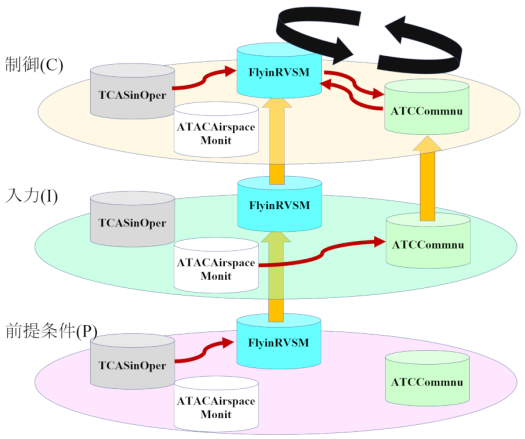


図 **16**　TORTE 階層イメージ
**Fig. 16**　TORTE hierarchical image

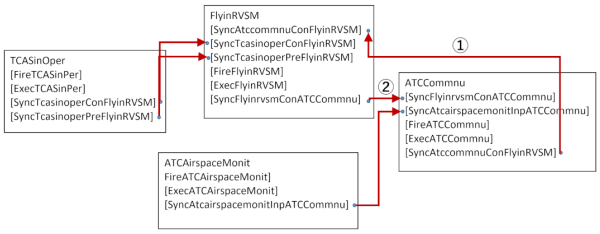


図 **17**　モジュール内のアクションラベルの同期の関係
**Fig. 17**　Synchronization relationship of action labels

期を表すアクションラベルのコードを図 **18** に示す．対応する受信処理，送信処理は同じアクションラベルが付けられており，条件が揃った時点で同時に更新が行われる．

```
module FlyinRVSM
[SyncAtccommnuConFlyinRVSM]
  flyinrvsm_f_control_passive_atccommnu=1
   & atccommnu_f_control_active_flyinrvsm=0
   -> (flyinrvsm_f_control_passive_atccommnu'=0);//①受信
[SyncFlyinrvsmConATCCommnu]
  flyinrvsm_f_control_active_atccommnu=0
   -> (flyinrvsm_f_control_active_atccommnu'=1);//②送信
```

```
module ATCCommnu
[SyncFlyinrvsmConATCCommnu]
  atccommnu_f_control_passive_flyinrvsm=1
   & flyinrvsm_f_control_active_atccommnu=0
   -> (atccommnu_f_control_passive_flyinrvsm'=0);//②受信
[SyncAtccommnuConFlyinRVSM]
  atccommnu_f_control_active_flyinrvsm=0
   -> (atccommnu_f_control_active_flyinrvsm'=1);//①送信
```

図 **18**　PRISM コード
**Fig. 18**　PRISM Code

### 6.7　PRISM モデルによる検証

6.4 節の分析で指摘された課題で，「飛行制御機能」に関連するのは以下のものである．

- パイロットは制御ループから外れて指示待ち，管制官

も危険を認識できず指示を出せない
（通信の制御ループが回らない）

- 継続的なパフォーマンス監視が必要である

上記の課題について，FRAM モデルの理解を支援する目的で PRSIM による振る舞いの数値化を行う．

#### 6.7.1 制御ループの振る舞いの数値化

図 12 の制御ループに reward（報酬）を設定する (**図 19**)．

```
rewards "State"
  [SyncFlyinrvsmConATCCommnu]true:1;
  [SyncAtccommnuConFlyinRV]true:1;
endrewards
```

**図 19** reward コード
**Fig. 19** reward Code

SyncFlyinrvsmConATCCommnu（図 12 の①）と SyncAtccommnuConFlyinRV（図 12 の②）はモデル生成ツールが自動作成したアクションラベルである．ループが一周すると reward（報酬）は 2 累積される．以下の 1000 単位時間までに累積される reward を求める検査式を実行する．

$$R\{\text{"}State\text{"}\} =? \quad [C <= 1000] \qquad (1)$$

全ての注釈条件の rate を「成り立つ」を 1，「成り立たない」を 0.01 とした（成否の比率 100:1）結果は，reward（報酬）は 490.9（約 245 周）であった．次にトランスポンダの信号の不調による変動を検証するため，機能「TCAS in operation」の注釈条件が「成り立たない」の rate を徐々に 30 まで上げると reward（報酬）は 77.5（約 38 周）まで低下した（**図 20** の実線）．トランスポンダの信号の頻度が，管制官とパイロットの間の制御ループの回転数の上下を決定していることが定量的に確認できた．

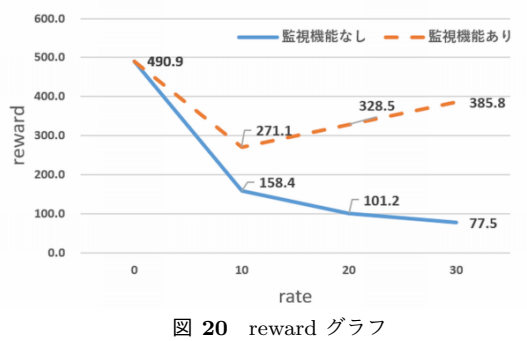


**図 20** reward グラフ
**Fig. 20** reward Graph

#### 6.7.2 パフォーマンス監視機能追加の効果の数値化

改修案として，継続的なパフォーマンス監視が必要との指摘がある．定期的に「Fly in RVSM」の入力 (I) へループを起こす出力 (O) を送る監視機能を追加した．監視機能実行の rate をトランスポンダの不調が生ずる rate の 1/100 に仮想して reward を 6.7.1 と同様に検証したところ，制御ループの回転数の低下を防ぐ効果を定量的に確認できた（図 20 の点線）．

### 6.8 考察

6.7.1 では，rate のラベル付けによる競合と時間的な制限を用いることにより，管制官とパイロットの間にある制御ループの過渡的な特性を数値化できた．また，6.7.2 では，パフォーマンスの変動を管理する仕組みを追加した場合の効果を確認できた．どちらも定量的な評価により振る舞いを理解して，パフォーマンスの変動が捉えられた．本提案手法は FRAM の分析のステップ 3，4 で有効と思われる．

## 7. まとめ

本稿では，モデル検査ツール PRISM を用いて，FRAM モデルの振る舞いを数値的に表す手法を提案した．適用事例の FRAM モデルは，シンプルなため結果自体は予想できるものであったが，この手法は修正後の効果の予想が難しい，もっと複雑なモデルでも同様に適用することができる．今回，モデル作成ツールを用意したことにより，PRISM モデル作成の手間が軽減されより効率的に検証が進められるようになった．今後の課題として，入出力を複数持つ振る舞いを表す FRAM モデルの定義方法の簡易化がある．また，状況に応じて手動で修正を行う必要がある．検証を積み重ねて，必要なパターンを選別してツールに実装していきたい．そのためにも今後，実践的な課題に取り組んでいく予定である．

### 参考文献

[1] エリック・ホルナゲル，社会技術システム－ FRAM ガイドブック，pp.25-38， pp.45-47， 海文堂出版，2013.
[2] Erik Hollnagel，David D.Woods，レジリエンスエンジニアイング　概念と指針，p.23，日科技連，2012.
[3] 吉岡信和，SPIN による設計モデル検証，pp.11-14，近代科学社，2009.
[4] 産業技術総合研究システム検証研究センター，モデル検査上級編 － 実践のための三つの技法－，pp.4-5，近代科学社，2010.
[5] M. Kwiatkowska, G. Norman, and D. Parker, PRISM 4.0: Verification of Probabilistic Real-time Systems, In Proc. 23rd International Conference on Computer Aided Verification (CAV'11), pp.585-591,2011.
[6] S. Ogata, et al., A Tool to Edit and Verify IoT System Architecture Model, MODELS 2017, pp.571-575, 2017.
[7] de Carvalho PVR, The use of Functional Resonance Analysis Method (FRAM) in a mid-air collision to understand some characteristics of the air traffic management system resilience, Reliab Eng Syst Saf 96(11),pp.1482-1498,2011.
[8] 野本秀樹, 道浦康貴, 石濱直樹, 片平真史, FRAM（機能共鳴分析手法）による成功学に基づく安全工学，SEC journal , Vol.14, No.1, pp.42-49, 2018.
[9] 青木善貴，井上祐司，モデル検査を用いた FRAM モデルの機能共鳴の解析，第 16 回ディペンダブルシステムワークショップ，2018.