FRAM（機能共鳴分析手法）による成功学に基づく安全工学


野本 秀樹※1


道浦 康貴※1


石濱 直樹※2


片平 真史※2

レジリエンス・エンジニアリングのモデリング手法である，FRAM（Functional Resonance Analysis Method：機能共鳴分析手法）を使用し，実製品開発で安全分析を行うための方法及び手順，とくにモデリング方法と評価方法を構築した．FRAM を使った安全分析では，従来のリスクや失敗原因を追及するだけではなく，反対に成功要因を分析する．成功学に基づく安全分析で具体的に着目すべき観点をまとめた．

New Safety Engineering using FRAM (Functional Resonance Analysis Method)

Hideki Nomoto※1, Yasutaka Michiura※1, Naoki Ishihama※2, Masafumi Katahira※2

This paper describes modeling and analysis methodology and procedure using FRAM (Functional Resonance Analysis Method). FRAM is the methodology to realize safety analysis based on resilience engineering. The analysis will be performed based not on failure but on success. Required perspectives to conduct success-based safety analysis will be shown.

1　はじめに

FRAM(Functional Resonance Analysis Method: 機能共鳴分析）は，レジリエンス・エンジニアリングにおける安全分析のための手法である [Hollnagel 2013]．機能共鳴とは，複数の機能が相互にインタラクションした結果外乱に柔軟に対応する一方で，逆にエスカレーションなどを起こし，安全を脅かすことを指す．つまり，FRAM における安全分析とは，複数の機能が互いにどのようにインタラクションするのかを明らかにし，その関係性の中に安全にかかわるシステムの長所や短所を見出すことであると言うことができる．本稿では，FRAM の分析の考え方を述べた後に，具体的な分析手法を記述する．

2　従来の安全工学との違い

2.1 FRAM の特徴

従来の安全解析は，ハザードの発生など，システムが失敗する事象を定義し，その原因を分析していくものである [FAA]．これに対して，FRAM はシステムの失敗事象を何ら定義せずに分析を行うことに大きな特徴がある．これは，安全に対する思想の相違から生まれる特徴である．FRAM

※1　有人宇宙システム株式会社　Japan Manned Space Systems Corporation
※2　研究開発法人宇宙航空研究開発機構　Japan Aerospace Exploration Agency

を生んだ安全工学をレジリエンス・エンジニアリングと呼ぶ．レジリエンス・エンジニアリングは，システムの安全は環境変動や意図しない入力に対する柔軟さによって達成されると考える [Hollnagel 2012][Holling 1973]．一方，その柔軟さが逆に意図しないシステム挙動を生む可能性もある．硬質のボール同士の衝突によるその後の挙動は予測可能であるが，柔軟なアメーバ同士の衝突に続いて次に何か起こるのかは予測が難しい．

つまり，入力の変動に対して柔軟に変動できる能力の高さが安全性の高さ（成功要因）であると同時に，変動することそのものは安全のリスク（失敗要因）でもあると言うことができる．これを，レジリエンス・エンジニアリングでは，「失敗と成功の同義性」と呼んでいる [Hollnagel 2016]．すなわち，安全を脅かす要因は，「失敗」や「故障」だけではなく，「成功」も安全を脅かす原因となる．

2.2 従来の安全解析手法との比較

従来の安全解析手法では，FTA(Fault Tree Analysis) のように，ハザードの原因を，明示的に「故障」に求める [FAA]．また，STAMP/STPA(System Theoretic Accident Model and Process) のように，コンポーネント間のインタラクションが「遅れる」「間違う」「途中で止まる」「提供されない」など，故障以外の要因を求める技術もある [Leveson 2012]．いずれにしても，これらの技術は，物事の「失敗」に着目するものである．

従来の安全工学とのもう一つの大きな違いは，分析がボトムアップに行われる性質にある．FTA はハザードをトップ事象として，その原因を詳細に分解していく．STAMP も同様にハザードをトップ事象とし，ハザード制御がどのように破たんし得るのかをトップダウンに詳細化していく．一方 FRAM は，まず個々の機能の詳細な定義から始め，分析の結果として全体ネットワークがモデル化され，システムの成功要因が導出される．従来の手法が演繹法であり，「失敗に基づく分解」を行っているのに対して，FRAM は帰納法であり，「成功に基づく統合」を指向している．

FRAM では，上述したように，最初のステップで個々の機能の詳細な定義を行う．機能の変動につながる表 1 に示す 6 つの要素を分析する．

2.3 制御システムの分析例

ここで，人工衛星の姿勢制御に関する分析を考える．従来の安全解析では，例えば，姿勢レートセンサと姿勢制御装置とのインタラクションを分析する際，姿勢レートセンサから出力される姿勢レート値が届かない場合，姿勢レートが間違っていた場合などのケース分けをして影響評価を行っていく．その多くは，異常処置の内容を精査することにつながる．

表 1　機能の 6 要素

I: 入力	機能の動作トリガー
P: 前提	機能が動作開始するための事前条件
C: 制御	機能の挙動方法を操作する事後条件
R: 資源	機能の動作に必要な資源（事後条件）
T: 時間	機能実行可能時間（事後条件）
O: 出力	機能の出力

一方，姿勢制御機能と姿勢レート測定機能がインタラクションするという関係性を FRAM により分析する場合，やり取りされるデータの変化よりも，機能そのものの変化に着目する．すなわち，姿勢制御機能がその機能を変えるのは何によってであるのかを分析する．姿勢制御機能がその制御方法を変更するのは，入力されるセンサデータの有効フラグの値が変わったとき（前提条件），周期処理スロット内にデータが入力されたか否か（時間），小さなデッドバンド幅で制御を行う精姿勢制御モードと大きなデッドバンド幅で制御を行う粗姿勢制御モードが入れ替わったとき（制御）など，とくに入力の異常に限定せず（と言うよりも，むしろ大部分が正常ケース）識別していく必要がある．

このような，機能間インタラクションのモデリングを行う場合は，通常のインターフェース仕様に定義される「データフォーマット」など，シンタクスを規定するのではなく，「前提条件」「制御」「資源」といったセマンティクスを規定することになる．それにより，単純にデータが来ない・間違う・遅れるなどの異常系の評価を指向せず，正常・異常を全く分け隔てすることなく，機能間の関係を定義する．

現代的システムでは，ダイナミックに機能同士が通信し，互いに制御し合う分散的な関係を持つアーキテクチャが主流になりつつある [Ding 2007][Aoki 2014]．このようなカップリングを多用するシステムにおいては，ある機能とある機能が極めてまれな条件の組み合わせのときのみ，危険な相互作用をするというような，検出しにくい問題の方が重大な問題を引き起こしやすい傾向を持つと考えられる．なぜなら，まれにしか発生しない問題は，そのような問題が存在すること自体が認識されていないことが多いからである．

発見しがたいカップリングを発見するためには，カップリングの様子を詳細に可視化する手法が必要となる．しか

も，そのモデルには，依存関係，タイミング，制御・被制御関係，制約・被制約関係など，様々な関係を一挙に俯瞰できることが求められる．なぜなら，単純な制御・被制御の関係だけから問題が発生することはまれだが，例えば「制御する側が制御される側よりも遅いタイミングで起動される」というような複数の側面の特殊なカップリング関係こそが発見しがたいまれなカップリングになり得るからである．そのようなカップリングを見出すためには，タイミング・モデルと制御・被制御モデルを別々に作る手法よりも，一つのモデルにそれらをすべて取り込んだモデルが効果的となるであろう．なぜなら，モデルを使った分析とは一種のひらめきを喚起するための道具の使用法であるが，ひらめきとは，複数の認知リソースの同時活性化によって生まれるとされているからである [Suzuki 2009].

上述したように，モデルを見てカップリング挙動をイメージするためには，機能間のつながり方の情報量の豊かさが重要となる．そのためには，機能と機能との結び付き方は，単純な矢印 1 本で描画されるのではなく，結合の意味（セマンティクス）を伴うモデルの作成が必要となる．FRAM が機能間の接続子として，「入力」「前提条件」「制御」「資源」「時間」「出力」の 6 つの意味的接続子を使っているのはそのためである．図 1 には，全く同一の結び付きを，STAMP の Control Structure Diagram (CSD) 図と，FRAM モデル図とで表した．FRAM のモデルが STAMP のモデルに比べて情報量が多いことを示すため，それぞれの機能やコンポーネントの意味は伏せたままで，それぞれがそのままでどこまで評価可能なのかを以下に示す．

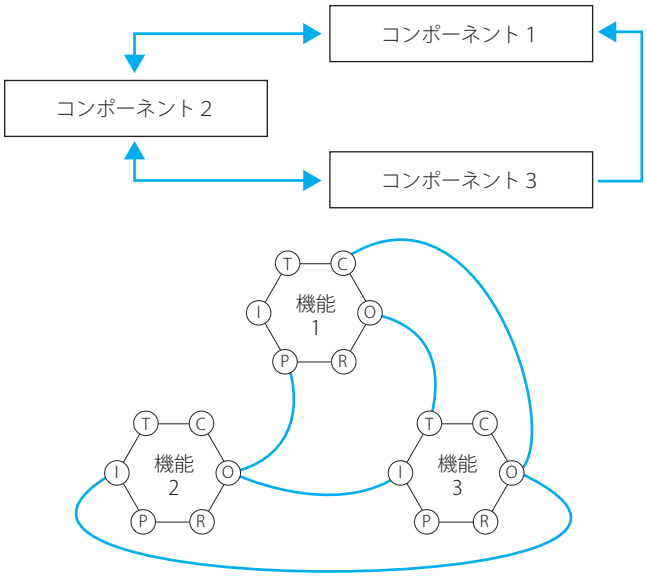

図 1　STAMP モデルと FRAM モデル

FRAM モデルを見ると，機能 2 と機能 3 との間にフィードバックループの関係があり，そこに機能 1 が割り込むような形で時間制約を与えている．その時間制約が行われるか否かは，機能 2 の出力が前提条件となって決定される．かつ，その時間制約は機能 3 の出力によって内容が変動するため，不意なタイミングで機能 3 が処理時間超過を宣告されるようなリスクがあることを読み取ることができる．

機能 2 と機能 3 との間に定常的に存在しているフィードバックループは無限ループの形を取っている（入力と出力が双方に接続）ため，基本的には安定したループであるが，機能 1 の存在によって，その安定した関係に変化が生じる．そして，その変化を生み出しているのは，機能 2 及び機能 3 自身の出力である．つまり，機能 1 と機能 2，そして機能 3 の間には，非常にダイナミックな共鳴関係が築かれているということになる．

上述したような評価を可能としているのは，FRAM モデル特有の 6 つの要素という情報量の多さである．図としてのシンプルさとは裏腹に，極めて多くを語るモデル化手法と言うことができる．一方，図 1 の STAMP モデルからは，モデルが持っている情報量が FRAM モデルと比較して少ないため，上記のような分析を導くことは難しい．STAMP モデルに描かれているインタラクションはやり取りされるデータにラベルが付けられていなければ理解することはできない．そして，インタラクションのタイミングに関しては，たとえラベルを付けられていても，明らかにすることができない．タイミングに関する情報量はほぼゼロとなっている．

相互影響のありようをモデル化した後，その相互影響のありようにどのような変動要素があるか否かを分析し．次に，そのような変動要素は，本質的に危険な変動なのか，それとも安全な変動なのかを最後に評価する．この一連の手順は自明なものではない．本研究では，この評価手順を 3 節に示すように構築した．

3　FRAM分析手順

提案する FRAM による分析手順は以下の 2 つステップから成る．

(1)　モデリング手順

(2)　評価手順

3.1　FRAM モデリング手順

FRAM モデリング手順は，以下の順序で実施する：

(1)　質問による機能の把握

(2)　各機能の定義

(3) 可視化

(4) 可視化したモデルを使った分析

FRAM のモデリング手順は，分析対象システムの設計者，あるいは，当事者などに対するインタビューを通じて，システムの中の重要「機能」の特徴を明らかにするところから始める．その手順を 3.1.1 項に示す．

次に，質問の答えをもとに，各機能の入出力を定義する．これがモデル化作業である．その手順を 3.1.2 項に示す．

各機能の入出力データ名が明確になれば，それをツールに入力することによって可視化が完成する．その手順を 3.1.3 項に示す．

最後に，可視化されたツールを俯瞰しつつ評価を行う．評価の詳細は，3.2 項「モデルの評価」に示すが，その全体像を 3.1.4 項に示す．

3.1.1 質問による機能の把握

解析対象を俯瞰して，最も重要な機能というものを把握する．FRAM は機能間の関係に着目するので，最も重要な機能を決めたら，そこからつながりのある機能を数珠つながりにモデリングしていく．その際，重要な機能とほかの機能との結び付きをいきなりモデリングしようとしてしまうとうまくいかない．なぜなら，ある機能がある機能に対して「制御」を提供しているのか，「前提条件」を提供しているのか，などという問題は，単純ではないからである．そのような微妙な作業を実施するためには，様々な関連情報を揃えておく必要がある．そこで，まずは機能の概要を把握するために以下の質問を行う．

(1) その機能の目的は何か？

(2) 機能はどのような処理を行っているか？

(3) 機能にはどのような入出力が存在するか？

続いて，FRAM のインタビュー技法 [Hollnagel 2013] に基づき，以下の質問を行い機能の詳細を把握する．

(1) その機能の開始トリガー（入力）は何か？（I に関する質問）

(2) 条件が変わった場合，どのように適応するか？

(3) オフノミナル条件にどう反応するか？

(4) リソースは安定的に供給されるか？不安定要因は？（R に関する質問）

(5) 外部環境はどのくらい安定？不安定要因は？

(6) オフノミナル条件はたびたび発生？

(7) 「当然」と思われている前提条件はあるか？（P に関する質問）

(8) 時間制約によるプレッシャーはどこにかかるか？（T に関する質問）

(9) 特別なスキル，特別な高機能，特別な高信頼性を必要とする個所は？

(10) 最適な実行方法というものが存在しているか？（C に関する質問）

FRAM においては，ハザード解析のように，リスクパターンを網羅的に分析するのではなく，機能の 6 要素を網羅的に分析する．この網羅性によって，機能間のインタラクションの見落としがないことが保証され，システムの特徴を漏れなくモデル化することができる．システムの特徴とは，そのシステムが成功できる理由そのものである．

3.1.2 各機能の定義

FRAM の 6 つの機能要素は，表 1 に示したように，1 つの出力と 5 つの入力で構成されている．これらの要素は，3.1.1 項の質問を参考にして，重要な機能から順に定義していく．

定義に必要なのものは，各要素の名前（やり取りされる情報・データ・もの・締め切り時間）と相手の機能の名前である．

3.1.3 モデルの可視化

モデルの可視化は FRAM Visualizer[FMV] を使って行う．ツールには，3.1.2 の機能の定義を入力すれば，図 1 のような機能カップリングの図が描画される．

機能を示す 6 角形がグレーになっているものは，バックグラウンド機能と呼ばれる変動せずに安定的にデータを定期的に出力する，あるいは，出力データそのものに大きな変動が存在しないため，後続の機能に対する変動要因とならず，後続機能のモデル化の必要がないと判断されたような機能を指す．FRAM の解析は，機能の変動に着目するため，これらのバックグラウンド機能は，FRAM モデルの「外縁」に相当する．注目する機能からバックグラウンド機能までが FRAM 分析の対象となる．言い換えると，FRAM においては，分析の結果，分析対象が後付けで決まる．これにより，分析対象から外れる「想定外のインタラクション」を最小化できる．分析範囲を前もって想定しないため，原理的に想定外というものが存在し得ないからである．

3.1.4 モデルを使った分析方法

3.1.1 項に示したインタビュー技法を使い，機能の特徴を十分に引き出すことができれば，次は，システムの成功要因とリスク要因を分析する．

分析は必ず以下の順序で行う：

(1) このシステム（モデル化した範囲全体）の成功要因は何か？

(2) このシステムのリスク要因は何か？

モデル全体を俯瞰して分析を行う場合，必ずシステムの成功要因を分析するところから始める．レジリエンス・エンジニアリングは，リスクの排除によるシステム構築を目指すのではなく，成功要因を識別し，それを育てると共に，成功要因の実現を阻むリスクを抽出することを目指す．

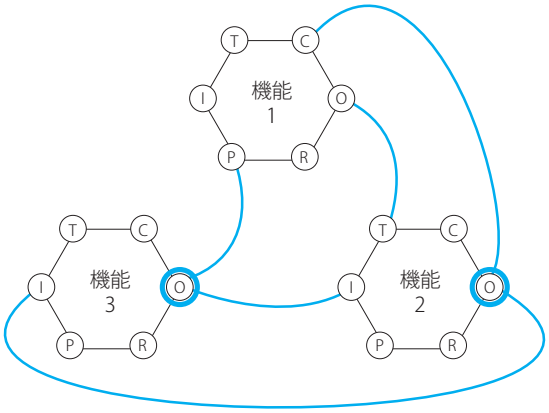

図 3.1.4-1　簡単な FRAM モデル

システムの成功要因は，幾つかの特徴的な点に現れる．例えば，上記の例では，機能 2 や機能 3 からの放射線状の出力や，機能 3 と機能 2 の間のループ構造，機能 1 と機能 2 の間のループ構造，そして，機能 3 から機能 2・機能 1 を経由して機能 3 に帰る大ループ構造など，特徴的な点が幾つかある．成功要因分析では，まず，これらの特徴的な個所を使い，システムの成功要因を明文化する作業を行う．一つの成功要因として，機能 3 と機能 2 のループに着目すると，この 2 つの機能は，通常はシーケンシャルに処理を行っている．したがって，機能 1 からの時間制約が入っても，それによって機能 3 と機能 2 の処理順序が逆転するような並列処理のリスクはなく，常に順序は守られるという特徴を持っている．これが成功要因の一つである．

次に，リスクの分析作業を行う．例えば，上述した成功要因（必ず順序が守られる）が得られると，順序が守られるため発生する問題点が見えてくる．システムは厳密な時間制約によって動いている．その制約を満たせない場合は，機能 2 は処理をストップする．機能 2 が処理をストップすると，機能 3 は開始トリガー（入力）を失うため，システム全体がストップすることになる．通常は，そのように耐性のないシステムは作られないため，機能 3 はタイムアウトを検知して次の周期の処理を実行するであろう．しかし，ここでは機能 2 からの開始トリガー（入力）がないため，通常の処理とは異なる出力が行われる可能性がある．もし機能 3 の出力が異常なジャンプをした場合，それを入力する機能 2 に何が起こるのかは，重要なリスク要因となる．

以上のように，成功要因の分析からリスク要因の分析につなげることにより，レジリエンス・エンジニアリングの思想に準拠した活動が可能となる．3.2 項では，この分析を効果的に進めるための，モデルのパターンごとの具体的な評価手順を示す．

3.2 モデルの評価

本項では，FRAM モデルの評価方法について，その手順を示す．なお，この方法は，ウッズらによる複雑なマルチスレッドシステムの典型的なパターン分析 [Woods 2006] を参考に構築した．

モデルの評価として，機能と機能の結び付き方のパターン（ネットワークトポロジー）に着目した評価方法を以下に示す．ウッズらは，複雑なシステムが示すパターンとして，Tempo（速度変動），Escalation（相互増強），Coupling（相互依存），Reframing（構造変動），Dilemmas（相互干渉）などを挙げ，システムの持つダイナミクスが当初の形態から次第に変化していく典型的な形態を，以下のように示した．

- Tempo はインタラクションの速度の変化
- Escalation はインタラクションの強度の増大
- Dilemmas はインタラクションの強度の減少
- Coupling は，複数のインタラクション間の依存
- Reframing はインタラクション構造の再構築

こうしたインタラクションの変化や複数のインタラクション間の依存性のような，複雑な問題を分析するために，FRAM のモデルをネットワークトポロジー的な特徴で分類し，カテゴリごとに，評価の観点を考察した．

3.2.1 やや複雑な機能共鳴の評価

非常に頻繁に見られ，かつ，複雑なネットワークの中でも比較的単純なパターンとして，下図 3.2-1-1 のような，同一種類の複数入力がある．ネットワークトポロジー的には，ツリー型に属する．

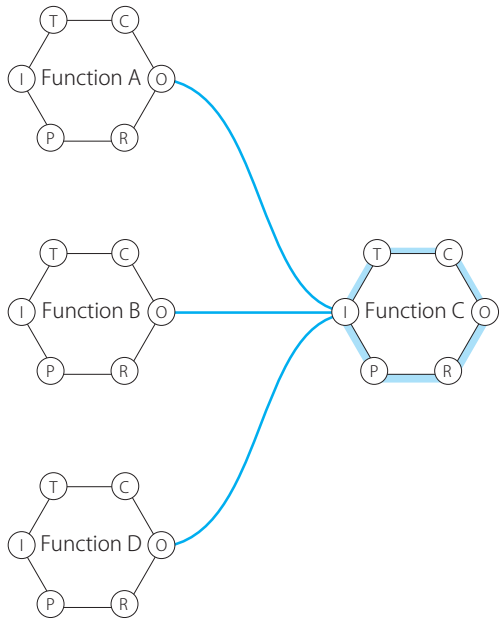

図 3.2.1-1　ツリー型

(1)　入力

- 複数の入力がある場合，どちらかに優先度はあるか？
- 複数の入力がある場合，想定される入力順序はあるか？
- 優先順位，入力順序が狂った場合，何が起こるか？

(2)　前提条件

- すべての前提条件が常に完璧に揃うことが想定されているか？
- すべての前提条件が揃わなかったら何が起こるか？
- すべての情報が揃わず，有効期限切れのデータを使ってしまうと何が起こるか？

(3)　制御

- 複数の制御が入力される場合，互いにエスカレートし合うか？
- 複数の制御が入力される場合，互いに相殺し合うか？

(4)　リソース

- リソース供給が不安定時に，出力はどう変わるか？

(5)　時間

- 時間が不足すると出力はどう変わるか？

(6)　出力

- 出力先から何かを入力している場合，エスカレートし合うか？
- 出力先から何かを入力している場合，相殺し合うか？

3.2.2　複雑な機能共鳴の評価

前項よりも，複雑な機能結合には，幾つかのパターンが存在する．以下，接続パターンごとに評価方法を示す．

(1) 放射線状の出力（出力が複数の相手先にある：スター型）

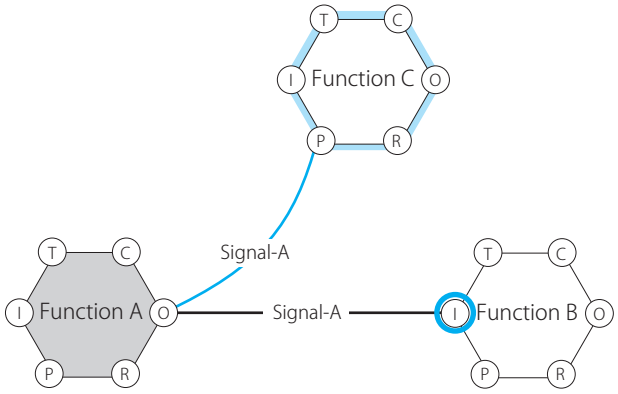

図 3.2.2-1　スター型

この場合，同じ Signal-A であっても，利用されるタイミングが異なる．Function B は，Signal-A 入力と同時に機能を実行し，Function-C は Signal-A が入力されても即座に実行せず，機能開始トリガーの入力待ちを続ける．つまり，Signal-A の内容は，Function-C が使用するときには古い情報になる可能性がある．また，入力タイミングのズレがあるということは，定常状態とデータの不安定な状態とで，システム挙動が変わることを意味する．データの不安定時には，タイミングのズレが Signal-A の利用される時間的ズレを増幅する．

(2) ループ構造（出力先からのフィードバックが返ってくる：リング型）

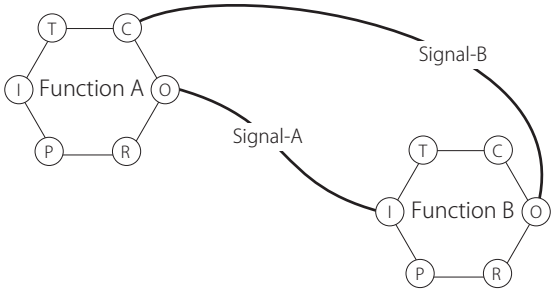

図 3.2.2-2　リング型

この場合，Function A と Function B は，互いに共鳴により強め合う，または弱め合う，もしくは，その複合という関係になることがある．

上図のようにフィードバックの帰り先が「制御」である場合は，いわゆるフィードバック制御を行うので，上記で言う複合関係となる．

下図のように，制御と制御が結合する場合，制御同士の競合，またはエスカレーションを防止するために，中間バッ

ファ的な機能（下図 Function-C）を作る必要があることがある．

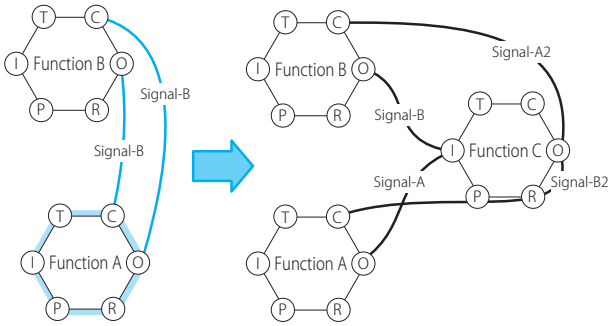

図 3.2.2-3　制御同士の競合を防止するための中間バッファの追加

(3) 入り組んだ親子関係（親が孫の子供になるような関係：メッシュ型／フルコネクト型）

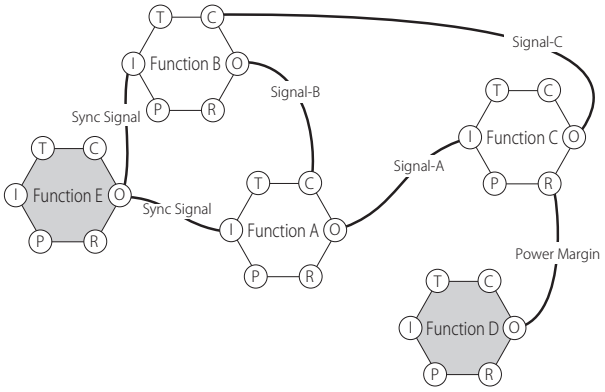

図 3.2.2-4　メッシュ型／フルコネクト型

上図では，Function-B が Function-A の親になっているが，Function-A の子供である Function-C は Function-B の親になっている．

上図の例だと，Function-A と Function-B は Function-E からの同期信号で起動しているので，実行開始は正確に同時であるが，Function-A が処理完了前に Signal-B が Function-A に届くか否かによって，Signal-A の内容は大きく変わる．Function-A と Function-B の実行時間が可変ならば，このシステムはタイミング上の変動要因を持っていることになる．そのとき，Function-C からフィードバックされてくる Signal-C は，Function-B にとっては前周期の自分の出力が反映された結果なのか，それとも前々回の出力が反映された結果なのかが認識できないため，ここを暗黙的に「前回値の反映結果」として使うと共鳴挙動が意図しないものになる可能性がある．

(4) 入り組んだ親子関係（ダイナミックに変わる関係）

あるシーンでは親，あるシーンでは子供というように，シーンによって親子関係が変わるパターンがある．下図の例だと，Power margin が不足したとき，Function-C は強権を発動して Function-B を機能制限するような制御を行うことがある．このシーンにおいては Function-C は Function-B の親である．しかし，通常時においては，Function-C は Function-B の孫に過ぎない．

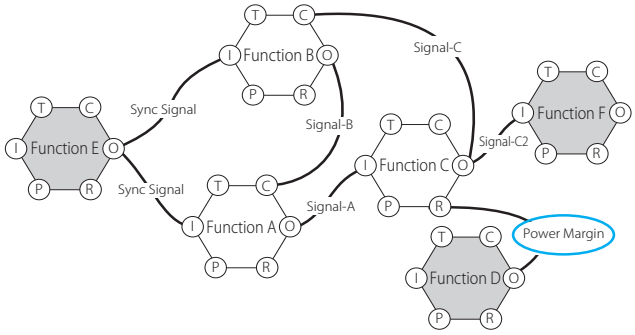

図 3.2.2-5　ダイナミックに変わる入り組んだ親子関係

上記のような例も人間組織や分散システムでよく見られるパターンである．手術室の医療チームの例として，普段は淡々と助手としてサポートを行っている看護師が，手術後の残留物確認の際に，全数チェックの結果 NG であれば縫合延期のトリガーを与えるような機能を持つ例がある．この場合，看護師の提供する機能が安全上のキーポイントになっており，これが同時にリスク源ともなる．かけ持ちで複数の手術が平行しているときなど，この全数チェック機能が最適なタイミングで行えずに医療事故に至った例がある．

また，子機能が親機能の動作を制限するような関係は，構造化プログラミングのアーキテクチャでは原理上発生することはまれだが，分散システムでは十分あり得る．外乱に対して多数のエンティティが並列的・分散的に反応することのできるシステムは柔軟性が高く，自然界の生物のようにロバストなものに進化することが可能であるが，挙動の変動が大きく，変動の条件の組み合わせが無数に存在するため，人工的なシステムとして構築することには困難が伴う．

(5) 三体問題

３つ以上のループが組み合わされることにより，これまで述べてきたような２つのループが接続されているような関係が更に複雑化する要因となる．例えば，下図のような構成は，青のループと灰色のループに，点線のループが絡むことによって，共鳴的な影響が増幅される．

青のループと灰色のループは，途中 Func2 を接点にして枝分かれするが，最終的には，同じ場所に返る．つまり，

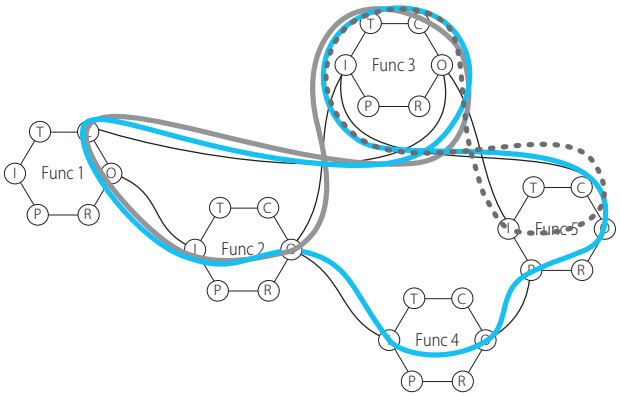

図 3.2.2-6　三体問題

途中青のループが回り道をすることになる．回り道をしている間は，別途時間がかかるが，Func5 への入力が Func5 の前提条件になっているため，青のループが灰色のループと同じタイミングで実行されるか否かは，ひとえに，点線ループが灰色ループと同じタイミングで周期運動するか否かにかかってくる．

つまり，灰色と青のループが同期を取って動作できるかどうかは，点線の実行タイミングによって決定されてしまう．点線の実行周波数が灰色の 1/10 であれば，自動的に青の実行周波数も 1/10 となるという強力な依存関係が存在している．つまり，灰色と青との関係は，灰色と青とのみからでは評価できず，むしろ第三者である点線がどのように関係するかによって決定されるという仕組みになっていることが分析される．

4　課題

FRAM を使った安全性評価が真価を発揮するのは，人間組織や人工知能など，機能が流動的に変化し，環境変動に対して柔軟に対応するタイプのシステムである．これらのシステムは，高い適応能力（つまり柔軟な機能変動）により成功を実現するシステムの典型例である．しかし，そうした高い柔軟性を持つシステムにおいては，成功と失敗の同義性 [Hollnagel 2016] も高く，システムの動作を保証することが困難である．

将来セーフティ・クリティカル・システムに人工知能を搭載することも一般的になる日が来ないとは限らない．そのとき，我々はそうしたシステムの安全保証をせねばならないという課題を突き付けられることになる．自然知能や人工知能のように，その柔軟さ故に，決定論的には動作せず，100% の安全を立証することができないシステムをどのように安全保証するのかが，今後の課題となる．

5　まとめ

FRAM を使ったモデリング方法，評価方法，実例，及び今後の課題について記述した．FRAM の安全評価手法はレジリエンス・エンジニアリングを実現するためのものであり，以下の特徴を有している：

(1)　システムの失敗要因ではなく，成功要因に着目する

(2)　個別のコンポーネントやデータではなく，統合的な視点でネットワークトポロジーに着目する

つまり，「成功学に基づく統合」という特徴が FRAM の特徴である．従来の安全解析手法が「失敗学に基づく分解」であったことの完全に逆の考え方となっている．従来の失敗学に，FRAM の成功学を組み合わせることにより，大規模・複雑な現代システムの安全がより強固に実現可能となる．失敗学，成功学共にどちらが欠けてもそれは不可能であろう．

【参考文献】

[Hollnagel 2013] エリック・ホルナゲル , 社会技術システムの安全分析―FRAM ガイドブック , (2013), p.25-38, 海文堂出版

[FAA] FAA System Safety Handbook (2000), Chapter 9: Analysis Techniques

[Hollnagel 2012] Hollnagel, E. (2012) “A Tale of Two Safeties”

[Hollnagel 2016] Hollnagel, E. (2016) “Introduction to FRAM - The Four Underlying Principles”

[Holling1973] Holling, C. S. (1973) “Resilience and stability of ecological systems”, Annual Review of Ecology and Systematics

[Leveson2012] Leveson, N. (2012) “STPA Primer”

[Wang2011] Hou, C., Wang, Q. (2011), "Software Interface Failure Modes and Effect Analysis Based on UML", DOI: 10.1109/ICM.2011.375

[Ding2007] W.Ding, Integration of MEMS INS with GPS Carrier Phase Derived Velocity: A New Approach," Proc. of 20th International Technical Meeting of The Institute of Navigation, pp.2085-2093, 2007.

[Aoki 2014] 須田 義大 , 青木 啓二 (2014) 自動運転技術の開発動向と技術課題 57 巻 (2014) 11 号 p. 809-817

[Suzuki2009] Suzuki. H. (2009). “Dynamics of insight problem-solving: Its generative, redundant, and interactive Nature” In S. Watanabe, A.P. Blaisdell, L. Huber, & A. Young (Eds), Rational animals, irrational humans. Tokyo: Keio University Press.

[Woods2006] Woods. D. & Hollnagel. E (2006) “Joint cognitive systems: Patterns in Cognitive Systems Engineering”, CRC Press ISBN 0-8493-3933-2, Chapter 9

[FMV] The FUNCTIONAL RESONANCE ANALYSIS METHOD, FRAM Model Visualizer (FMV), http://functionalresonance.com/FMV/index.html