修士論文

# 防災 IT システムの耐災害化に関する研究

瀧島　和則

篠田　陽一

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和 4 年 3 月

Abstract

Natural disasters have become more frequent in recent years.In order to deal with disasters and reduce damage, disaster prevention systems using IT are in operation.The disaster prevention IT system is required to operate normally even in the event of a disaster.However, some parts of the system may be destroyed by a disaster.Therefore, we need to make it resilient to disasters.

As a method of improving disaster resistance, strengthening the components constituting the system is used.However, when the disaster prevention IT system does not operate normally, there is an event that it is not caused by the failure of the component.In other words, it is necessary to improve disaster resistance in consideration of the entire system. Based on the results, we propose a method to verify the phenomena that can occur in the system in the event of a disaster by simulation. This makes it possible to study disaster resistance when constructing a disaster prevention IT system.

The proposed method was implemented for the evacuation order system. As a result, an unsafe event was discovered. In addition, the reproduction was confirmed by simulation. Furthermore, it was shown that by making the simulator compatible with other systems, verification can be performed while interlocking with other systems that are operating.

# 目 次

# 図 目 次

# 表 目 次

# 第1章　はじめに

## 1.1　研究背景

近年、洪水や津波など自然災害は増加傾向にある。国土交通白書 [1] によれば、氾濫危険水位を超過した河川数は 2014 年から 2019 年にかけて約 5 倍に増加している。このように激甚化、頻発化する災害に対処し被害を減らすため、IT を利用した防災システム (以下、防災 IT システムとする) が運用されている。例として、気象庁の緊急地震速報や NTT ドコモのエリアメールによる避難指示システムなどがあげられる。

これらのシステムは災害時に正常に稼働することが求められるが、災害の被害によりシステムの構成要素が破壊されることがある。防災 IT システムを構成する要素としては、現実の物理量を測定する地震計や水位計といったセンサ、センサで取得したデータを保存するストレージ、データを基に解析を行う計算機、計算結果を人に伝達するデバイスやそれぞれを接続するネットワークなど、多くの要素が存在する。そのため、防災 IT システムはサイバーフィジカルシステム（CPS）の一種といえる。サイバーフィジカルシステムとは、現実世界においてデータを収集し、サイバー空間で分析を行い、結果を現実世界にフィードバックする仕組みである。

耐災害性の向上としてシステムを構成する各コンポーネントを冗長化するなど、構成要素をそれぞれ独立して強化する形で向上が図られてきた。例として、災害に強いネットワークに関する研究などが存在する [2]。また、防災システムは日常的に利用される SNS などのサービスを使って情報伝達を行っている場合もある。しかしながら、システムの一部分を高信頼にしても実際の被災時にはシステム全体として上手く動作する事が出来ず、避難指示が発令されなかった事象 [3] が発生している。これらの事例では、情報伝達ミスや伝達の遅れにより適切な避難指示を発令することが出来ていない。一方、市町村や消防に市民などからの電話が殺到していることからある程度の通信インフラについては確保されていたことがわか

る。電話の殺到により電話対応に追われ、避難対策に人手を割くことが出来なかった結果として重要情報の漏れや判断ミス、対策の不履行が生じていた。このことから、防災 IT システムにおいては運用に関わる人間など、システム全体を考慮して耐災害性を向上させる必要があるといえる。

## 1.2 研究の目的

防災 IT システムのように停止することが致命的損害となりうるミッションクリティカルなシステムにおいては、高い信頼性を保証するため安全性解析が行われている [4]。このようなシステムにおいてはシステムの構成要素それぞれの信頼性を向上させることはもちろん、それぞれを組み合わせてシステムを構成した場合にも高い信頼性を保証する必要がある。本研究では、防災 IT システムの特徴を考慮しながら、安全性解析を行い、シミュレーションで災害時にシステムに起こりうる現象の検証を行う手法を提案することで、防災 IT システムを構築する際の耐災害性の検討を可能とする。

## 1.3 論文の構成

本論文では、2 章にて本研究で対象とする防災 IT システムの分類と特徴について述べ、その耐災害性について課題を整理する。3 章では既存の安全性解析の手法について延べ、防災 IT システムへ適用する際に考慮する必要がある点を整理する。4 章では STAP を用いた防災 IT システムの安全性解析を行い、アクシデントにつながるハザードシナリオを識別する。ハザードシナリオを基にテストケースを作成しシミュレータで再現することでハザードの検証を行う手法について述べる。5 章では実施例で想定するシステムで提案手法を実施してその妥当性を評価する。最後に 6 章で本研究の成果と社会的な意義を述べる。

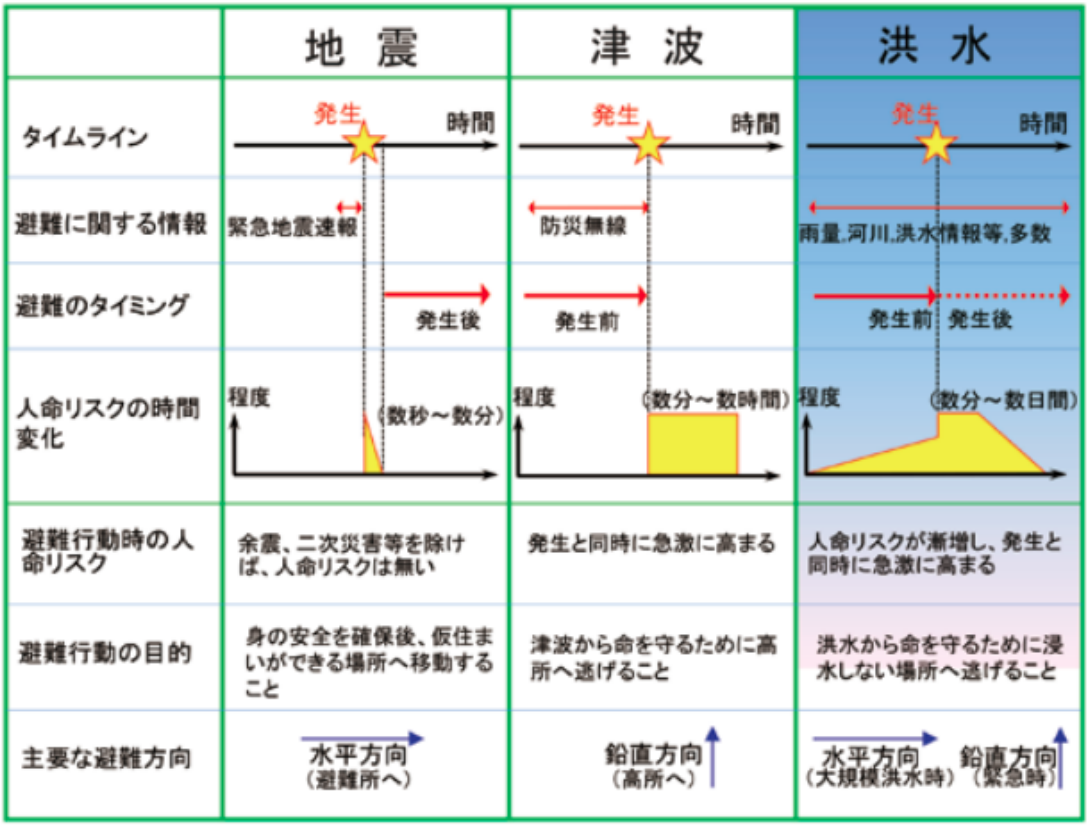
# 第2章　防災ITシステム

歴史を振り返るとITを用いない防災システムとしては、火の見やぐらなどがある。一方、現代社会においてはほぼすべての防災システムがITを利用している。本章では防災ITシステムの分類と、本研究で対象とする避難指示システムの特徴について述べる。

## 2.1　防災ITシステムの分類

防災ITシステムは様々な種類が存在し、対象とする災害に応じて使用される防災ITシステムは異なる。例として、地震では緊急地震速報、津波では津波警報システムなどがあげられる。災害の種類によって防災ITシステムに必要とされる機能は大きく異なるため、ここでは地震、津波、洪水避難の比較を行う。岡本 [5] による災害とその被害の分類表によると、地震では発生時が最も人命リスクが大きく、津波では発生時から数時間の間、人命リスクの高い状態が継続する。特徴的なのは洪水避難で、発生前から徐々に人命リスクが高くなり、発生後は徐々に人命リスクが低くなる。つまり、地震や津波では発生前に避難指示を発令すれば、発生後にその避難指示情報の更新が行われることは少ない。一方、洪水においては発生前から発生後まで連続的にリスクが変化することから、避難指示を発令すべきタイミングの検討が重要となる。
また、洪水には内水氾濫と外水氾濫が存在する。内水氾濫は市街地に排水能力を超える多量の雨が降り、排水が雨量に追い付かず発生する洪水である。これは、河川の増水によらない洪水であり河川周辺地域とは異なる場所でも発生する。外水氾濫は河川の水位が上昇し堤防を越え、破堤するなど堤防から水があふれることで発生する。これは、河川の増水に起因するものであり河川周辺地域で発生する。内水氾濫・外水氾濫どちらにおいても洪水の発生しやすい場所は存在するものの、実際に発生する場所を完全に予測することは困難であり、また降雨状況や河川水

位によって刻々と状況が変化する。したがって、洪水における避難指示発令システムは発生前から発生後まで継続して稼働する必要がある。

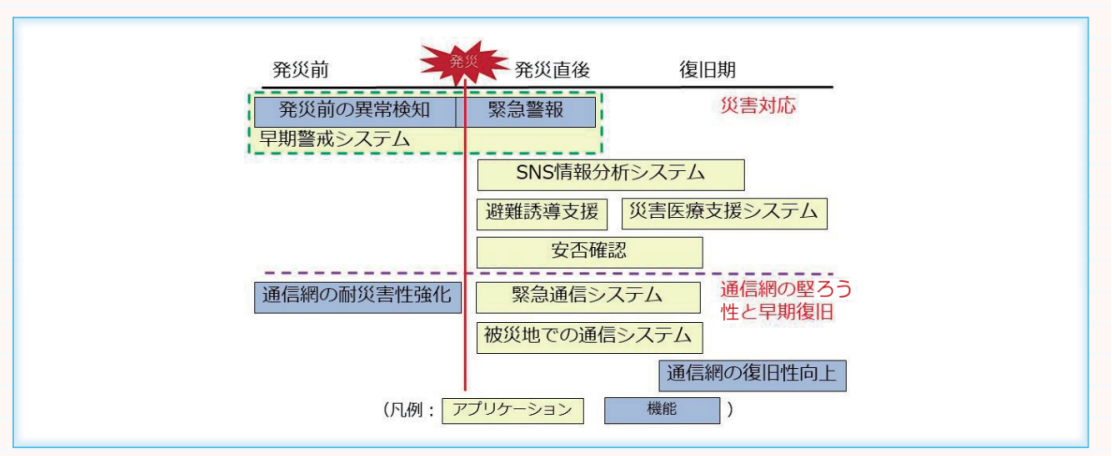



出典: 岡本 (2018)[5]

図 2.1: 地震、津波、洪水避難の比較

さらに、災害の発生前から発生後までを考えると、それぞれの場面で使用される防災 IT システムは異なる。ここでは災害の発生前、発生時、発生後という時間軸で防災 IT システムを分類する。長妻ら [6] は 2.2 に示す分類を行っている。災害の時間的経過により使用される防災 IT システムは変化する。発災前から発災前後では早期警戒システムが災害の被害を最小限に抑える役割を果たす。例として津波警報システムは津波が到達する前に予測し、津波警報を発令する。発災後は安否確認や避難誘導支援、災害医療支援システムなど被災者のリスクを低減するとともに被災状況を把握するためのシステムが動作することになる。近年は発災時に SNS などを通じた情報共有が活用されることも多く、発災後は通信網の迅速な復旧が求められる。災害のどの時点においても防災 IT システムは正常に動作することが求められるが、災害の発生時はシステム自体が被災し、正常に動作しなく

なる場合がある。また、津波や洪水では災害の進行によりシステムが災害発生中に故障する可能性がある。これは地震のように瞬間的に被害が発生する場合と比べ、システムの構成要素が故障する可能性のある時間が長くなるといえる。そのため、本研究では防災 IT システムのうち、災害発生中に動作する必要のある洪水での避難指示システムを対象とする。



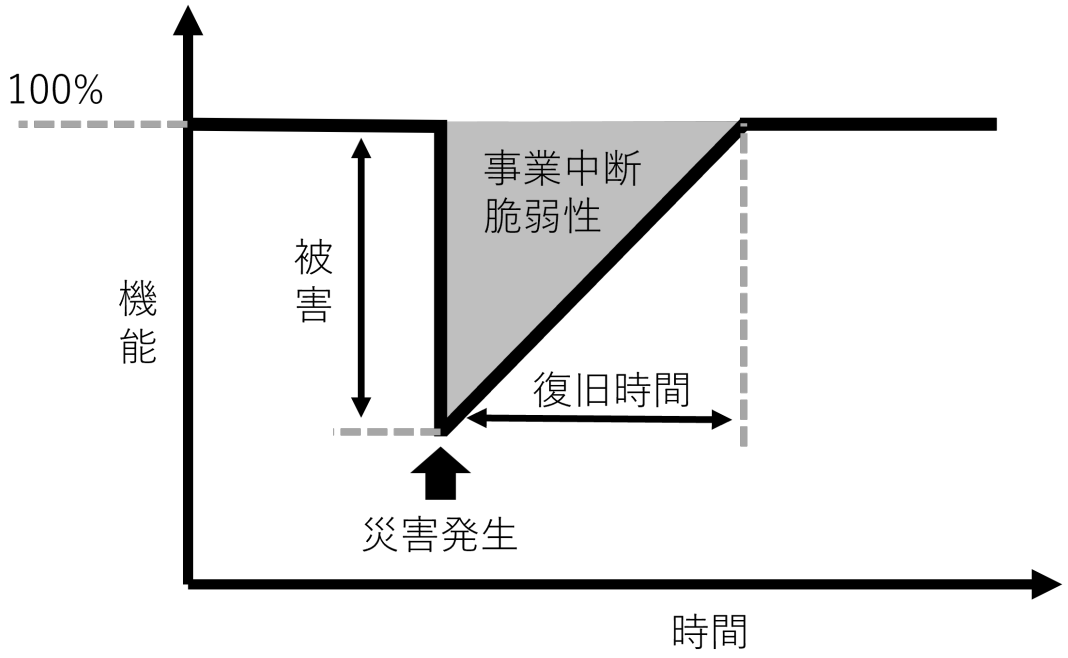

出典: 長妻 (2021)[6]

図 2.2: 災害フェーズと必要機能・アプリケーション

## 2.2 避難指示システムの特徴

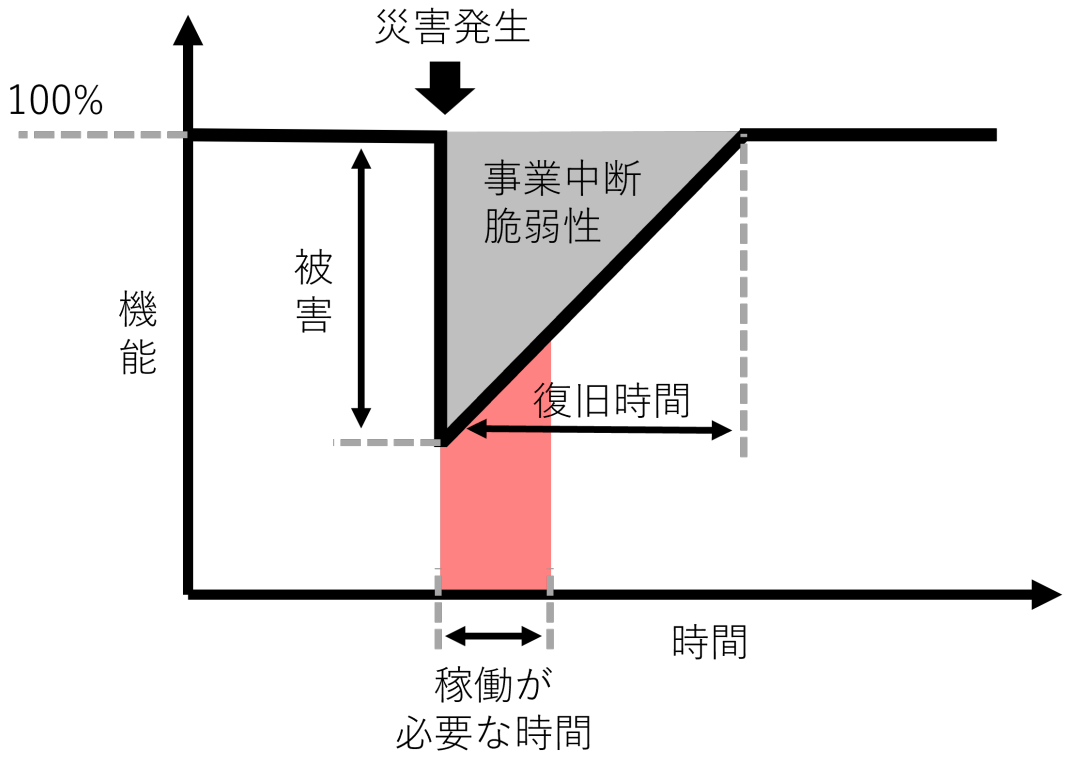
本研究で対象とする洪水の避難指示システムは、前述のとおり発災前から発災後まで正常に動作することが求められる。また、発災時に一度だけ避難指示を発令するだけではなく、洪水の発生中はいつでも避難指示を発令できる状態を維持することも求められる。また、災害において被害を軽減するだけではなく、被害の発生を前提としてそこからの立ち直る過程までを含めた総合的な視点が今後の防災・減災において必要とされている。この従来の予防力に加えて、災害を乗り越える力を加えたものは災害レジリエンスと呼ばれている。林 [7] によれば、災害レジリエンスのモデルは図 2.3 のように模式化される。ここでは発災前に個人であれ、組織であれ、社会から求められる機能を 100 %果たしていると仮定する。災害により被害が発生し、機能の一部あるいは全部が喪失する。そこから機能回復しようとするプロセスが開始される。これが災害を乗り越えるプロセスとされている。防災 IT システムにおいても、このレジリエントの考え方を適用することが

できる。災害の発生により防災 IT システムが被害を受けた場合、機能の一部あるいは全部が喪失する。通信経路が断絶した場合やシステムの重要なコンポーネントが被害を受けた場合には機能の全部が喪失すると考えられる。この場合において、洪水での避難指示システムでは、発災前から発災後まで動作することが求められるため、図における脆弱性 (事業中断) の面積を極力小さくすることが求められる。これは被害を受けないようにするために冗長化するなど、被害自体を小さくする方法や、迅速に復旧する方法、システムを縮退運転することで機能の低下を減らす方法などが考えられる。また、図 2.4 の赤線で示す範囲を避難指示システムが動作する必要のある時間とすると、この範囲における機能の面積を最大化することが求められる。この場合ではシステムを縮退運転することにより、面積を大きくすることが容易であると考えられる。したがって、洪水での避難指示システムでは、システムが必要とされる時間に可能な限り多くの機能を提供するため縮退運転が重要であると考えられる。




出典: 林 (2021)[7] を一部編集

図 2.3: レジリエンスのモデル




出典: 林 (2021)[7] を一部編集

図 2.4: 避難指示システムにおけるレジリエンス

# 第3章　システムを対象とした安全性解析技術

システムを対象とした安全性解析の手法はいくつか存在する。安全性解析手法はアクシデントモデルに基づき、そのモデルを実現するために手法化されている。主に設計時に使用される手法とそのアクシデントモデルについて説明し、避難指示システムに適用する場合について議論する。

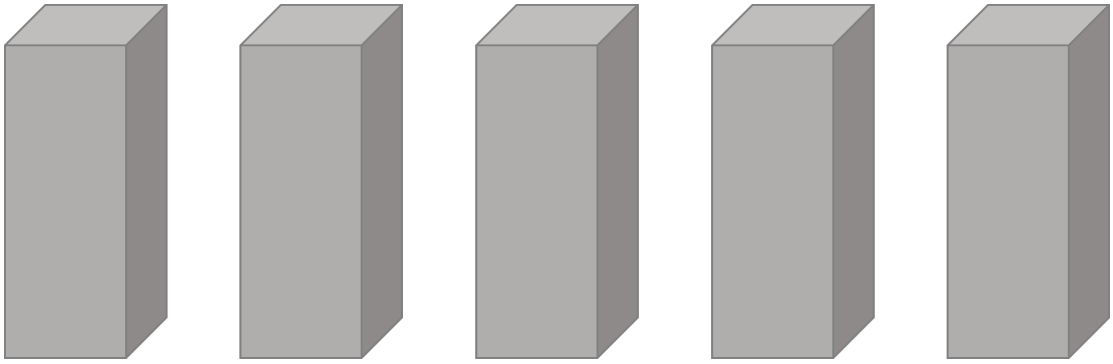
## 3.1　FEMA/FTA

あるアクシデントが発生したときには、システムを構成する構成要素のいずれかが故障するなど、要素のどれかに根本原因があると仮定する。Chain of Events モデルはこれを仮定したモデルであり、代表的なものとしてドミノモデルとスイスチーズモデルが存在する。FEMA(Failure Mode and Effect Analysis)[8] や FTA(Fault Tree Analysis)[9] はこのモデルに基づいた手法である。

### ドミノモデル

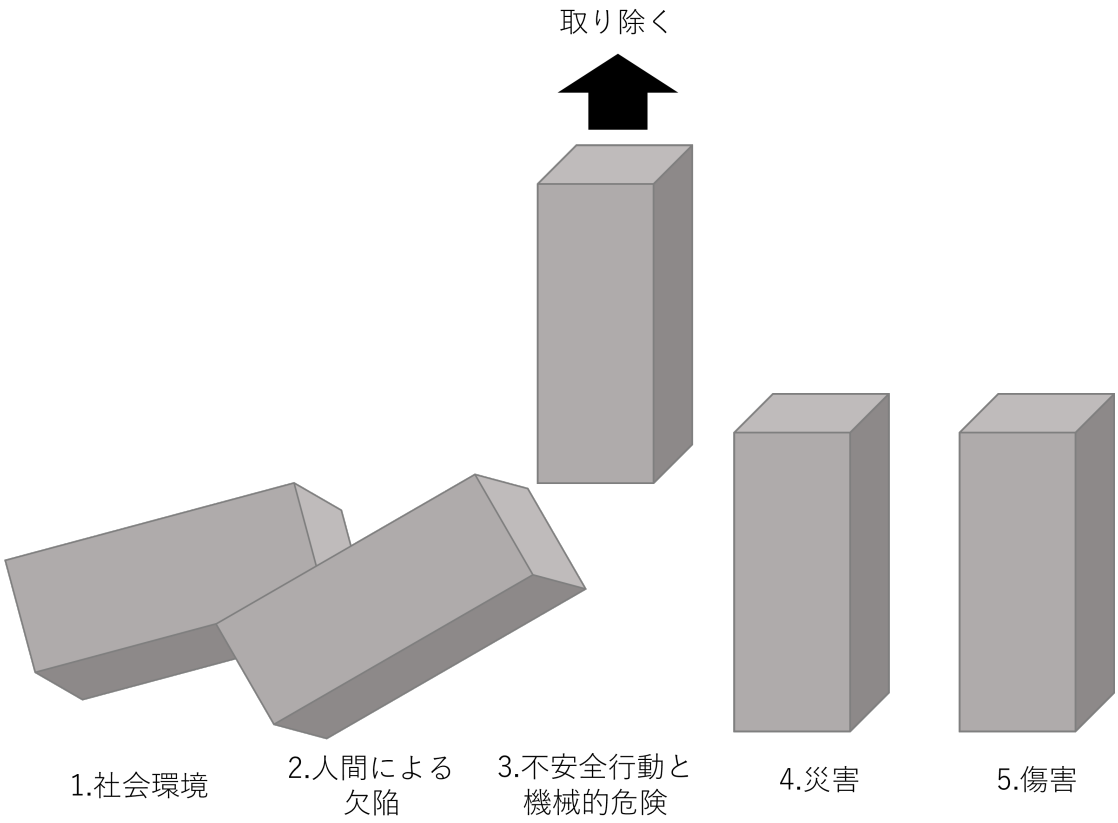
ドミノモデルは Heinrich[10] の提案したアクシデントモデルである。ドミノモデルは図 3.1 に示すように 1. 社会環境、2. 人間による欠陥 3. 不安全行動と機械的危険 4. 災害 5. 傷害の 5 つのドミノから構成され、事故や障害に至るプロセスを原因と結果の連鎖として説明したモデルである。このモデルでは１つのドミノが倒れることで次のドミノが連鎖的に倒れていき、最終的に災害や障害に行き着くことを系統的に表している。このモデルから連鎖するドミノのうち一つを取り除くことで連鎖的に倒れるのを止めることができるとしている (図 3.2)。ドミノの中でも不安全行為と機械的危険をその中枢であるとして、これを除去すべきとされて

いる。




出典: ハインリッヒ産業災害防止論を一部編集

図 3.1: 災害連鎖の五つの要因

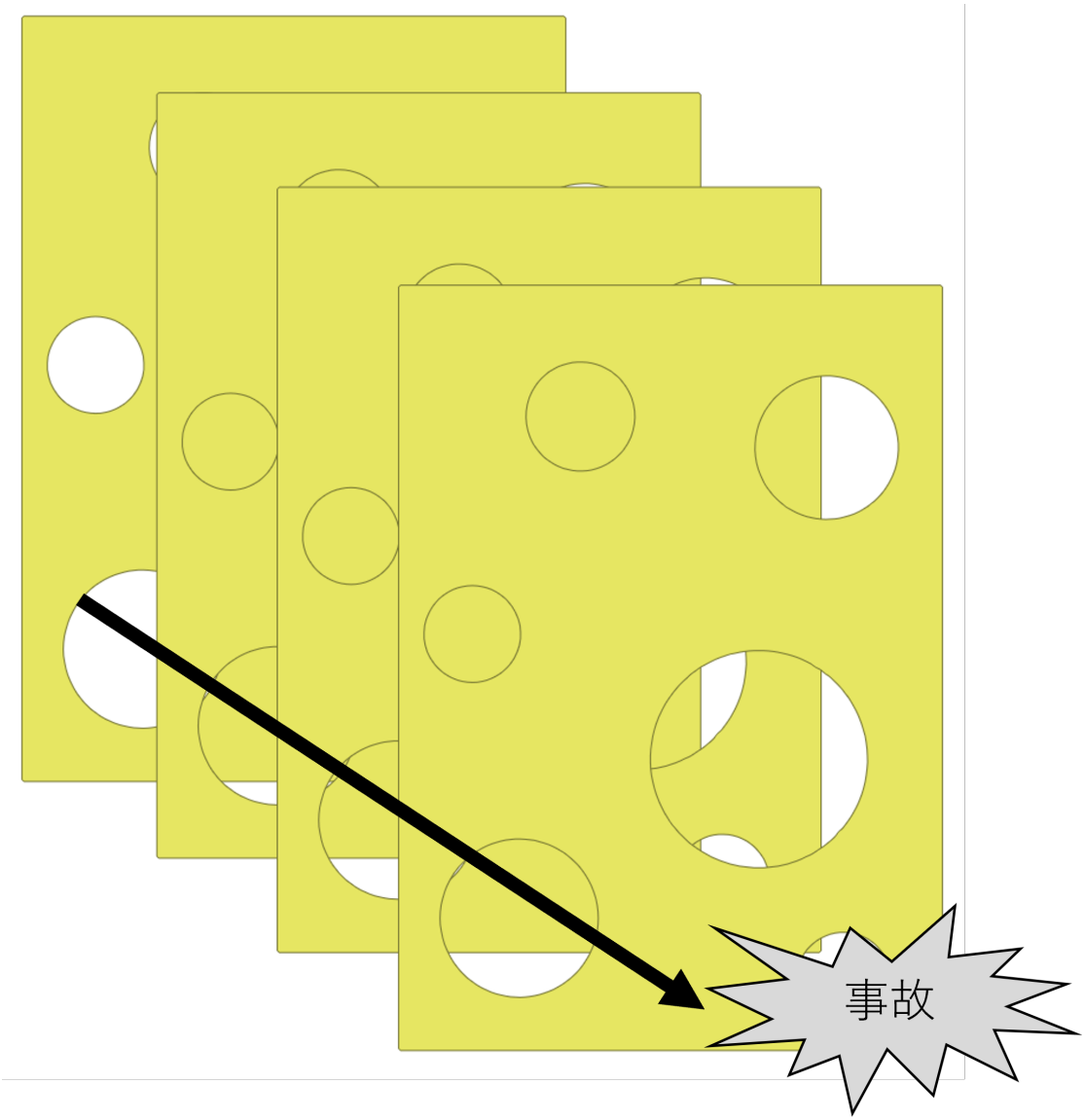



出典: ハインリッヒ産業災害防止論を一部編集

図 3.2: 災害の中心要素の除去による連鎖の無効化

## スイスチーズモデル

スイスチーズモデル [11] は James Reason が提唱したアクシデントモデルである。このモデルでは事故は各防護階層に生じた脆弱性が偶然重なりあうことで、システムが持つ潜在的な危険性が顕在化し発生することを示している。図 3.3 で示すようにスイスチーズモデルでは防護階層をスライスされた穴の開いたチーズ、穴を脆弱な部分としてチーズを何層にも重ねることでこのことを表している。穴の開き方が異なるチーズを重ねることでリスクがすべてのチーズを貫通する可能性を低くすることができ、事故へと繋がる可能性を下げることができる。この穴はヒューマンエラーや企業文化などの潜在的原因によって生じるとされており、人為的な行為や事情により大きさが変化したり移動したりする。




出典:Reason(2000)[11] を一部編集

図 3.3: スイスチーズモデル

## FEMA

FEMA は不完全な設計や欠陥を発見するために構成要素の故障モードを列挙し、故障発生時のミッションへの影響を解析するボトムアップ型の安全性解析手法である。FEMA は民間航空や自動車産業においても一般的に使用されている手法である。故障モードを列挙するため構成要素の受け持つ機能が定まっていることが精度の高い FEMA を行う前提となるが、設計の初期ではあまり細かい情報が得られ

ない場合がある。このことから設計の初期段階で機能 FEMA を実施し、詳細設計段階で詳細 FEMA を行う場合もある。FEMA における故障とは機能障害を指し、この障害を引き起こした要素の不具合が故障モードである。故障モードの分類は故障状態の形式（動かない、破損、摩耗、短絡など）で行う。FEMA では主に表 3.1 に示すようなワークシートを用いて解析結果を記述することが多い。

表 3.1: FEMA ワークシート

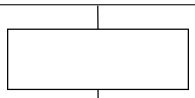
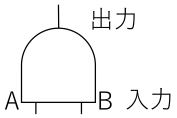
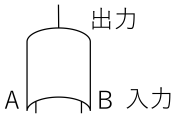
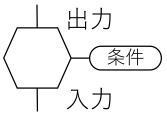
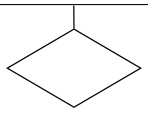
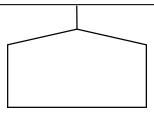
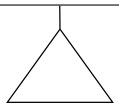
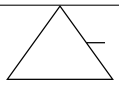
| 品目 (製品) | 機能 | 危険の程度 | 危険の原因 | 危険の影響 | 調べ方 | 防ぐ方法 | リスクレベル |
|---|---|---|---|---|---|---|---|
| 検討を行う製品、部品名 | 検討を行う製品、部品の機能を記入する | 検討を行う製品、部品の機能、性能が失われた場合を想定する | 検討を行う製品、部品の機能、性能が失われた場合を想定する | 危険が与える影響を記入する | 危険状態の検出方法として設計で対応可能な方法を記載する | 危険の発生を防止、抑制するために必要な設計、製造の対策を記入する | 危険が発生した場合の被害の度合いなどを記載する |

## FTA

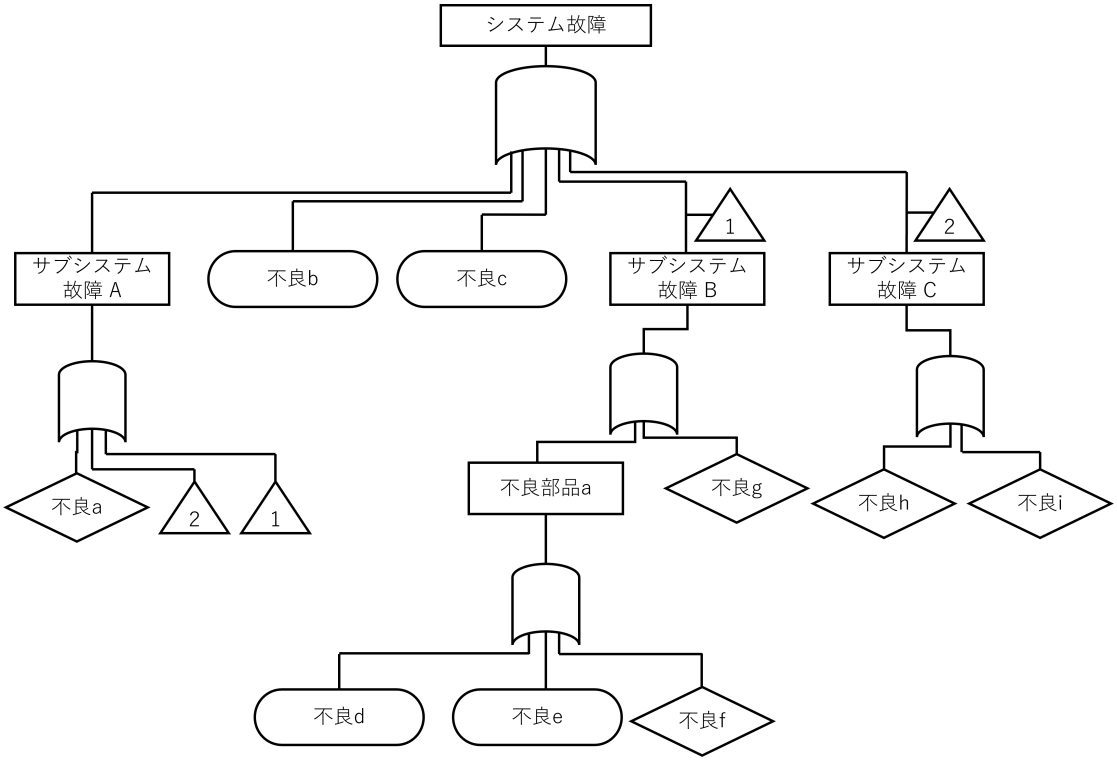
FTA は故障の木解析とも呼ばれ、FEMA がボトムアップ型の解析手法であるのに対し FTA は故障からその原因をたどるトップダウン型の解析手法である。JIS においては「下位アイテム又は外部事象，若しくはこれらの組合せのフォールトモードのいずれが，定められたフォールトモードを発生させ得るのかを決めるための，フォールトの木形式で表された解析」と定義されている。FTA では事故や損失をトップイベントとしてツリーのトップに配置する。このトップイベントからそれについて考えられる故障、事故に至った道筋を FT 図で表し分析していく。FTA の強みとして故障の組み合わせを考慮しながら解析を進めることができ、また解析結果は FT 図の木形式で表されるため結果に対する理解を促す効果を有し

ている。FT 図の作図では以下の表 3.2 に示す論理回路と似た記号を使用する。FT 図の例を図 3.4 に示す。

表 3.2: FT 図で使用する記号

| 名称 | 記号 | 意味 |
|---|---|---|
| 展開事象 |  | さらに展開されていく事象 |
| 基本事象 |  | これ以上は展開されない基本的事象 |
| AND ゲート (論理積) |  | 入力事象が共存して、はじめて出力事象が発生 (A∩B) |
| OR ゲート (論理和) |  | 入力事象のうち、少なくとも 1 つが存在するとき出力事象が発生 (A∪B) |
| 制止ゲート |  | 入力事象について、このゲートで示す条件があるときのみ出力事象が発生 |
| 非展開事象 |  | 情報不足などのために、これ以上天気出来ない事象 (基本事象と同様) |
| ハウス型 |  | 故障事象ではないが、通常発生すると思われる事象 |
| 移行記号 (IN) |  | FT 図の関連する他の部分からの移行を示す |
| 移行記号 (OUT) |  | FT 図の関連する他の部分への移行を示す |

出典: 山田 (2011)[12]




出典: 山田 (2011)[12]

図 3.4: FT 図の例

## 3.2 STAMP/STPA

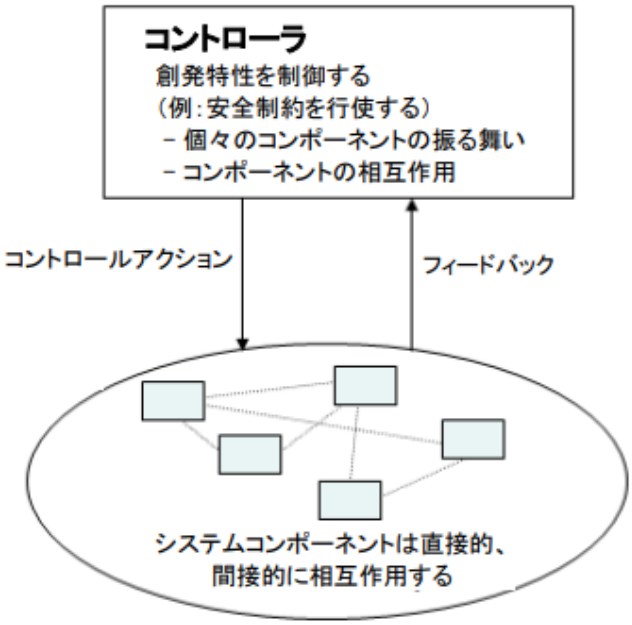
### STAMP

STAMP(Systems-Theoretic Accident Model and Processes)[13] モデルは従来の Chain of Events モデルに代わる新しいアクシデントモデルである。構成要素が少なく単純なシステムにおいては、Chain of Events モデルに基づきアクシデントが起きた時、根本原因はある構成要素であると分析することは容易である。しかしながら、構成要素の数が多く、要素間の相互作用が複雑なシステムでは根本原因となる構成要素を特定することが困難である。例として、次のような事象がある（STAMP HANDBOOK[14] より引用）

「ある海軍の航空機がミサイルをある地点から別の地点へ運んでいた。あるパイロットが、前の航空機を狙って（彼がそうやるように言われていたように）ダミー

ミサイルを発射して、計画されたテストを実行した。「賢い」ソフトウエアは 発射を指令されたミサイルが良い位置にない場合、代替のミサイルが発射されるように設計されていたことをどうやら誰も知らなかった。このケースでは、ダミーミサイルと照準の間にアンテナがあった。そして、ソフトウエアは代わりに異なる（良い）位置にある、実弾のミサイルを発射するよう判断した。ここでは航空機のどのコンポーネントが故障したのか？」

STAMP では故障や損失をシステムを構成する単一の要素の故障から起きるものとして扱うのではなく、複数の構成要素間における相互作用により発生するものとして扱う。そのため、引用で示したシナリオのように何かが故障したわけではないが、相互作用の誤りによって事故が発生した場合を扱うことができる。また、構成要素の振る舞いや相互作用による制御に着目して事故や損失を防ぐためのモデルであり、ハードウェアだけには限らず、ソフトウェア、組織、マネジメント、人による意思決定なども含めてシステム状態を扱うという特徴がある。STAMP においては、故障や損失はシステム構成する要素の故障によって起きるのではなく、システムの中で安全のための制御を行う要素と制御される要素における相互作用がうまく動かないことによって起きるとするアクシデントモデルである (図 3.5)。




出典:STAMP HANDBOOK[14]

図 3.5: 構成要素の相互作用の例

## STPA

STPA は STAMP モデルに基づく安全性解析手法である。個別の構成要素に着目するのではなく、構成要素間の制御に着目して解析を進めるという特徴がある。実際の手順はいくつかの Step に分かれている。以下に STPA の実施手順を示す。それぞれの手順の内容については章 4.2 で実際に解析を進めながら示す。

**Step 1** 損失とハザードの識別

**Step 2** コントロールアクションの作成

**Step 3** 非安全なコントロールアクションの抽出

**Step 4** 損失シナリオの識別

# 第4章　提案手法

本章では、防災 IT システムにおける安全性解析について説明し、安全性解析で判明した危険な状態を基にシミュレータでその状態を再現する手法について説明する。

## 4.1　提案手法の概要

本手法では、STPA を実施してその成果物である UCA や損失シナリオ、コントロールストラクチャを実行可能なシミュレーションシナリオとシミュレータによる実装に変換する。（図 4.1）また、シナリオに従ってシミュレーションを実行して分析を行う。本手法の特徴は、STPA の成果を利用しシミュレーションを行う点である。UCA が発生しなくなるようにシステム構成を変更して再度シミュレーションを行うことで、対策案の検討が可能である。このように対策とシミュレーションによる検証を繰り返しフィードバックすることでシステムを強くすることができる。

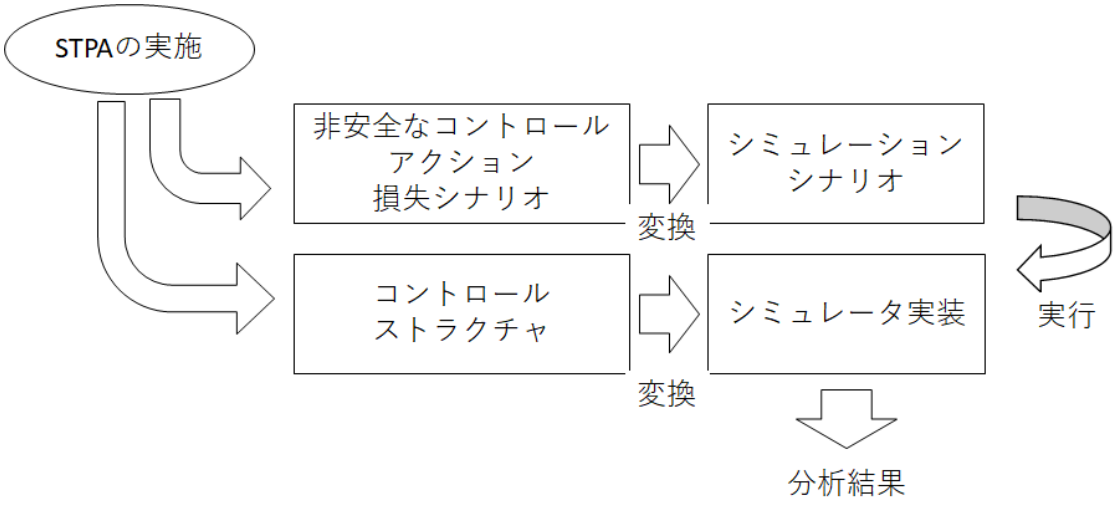



図 4.1: 本手法の流れ

## 4.2 STAMP/STPA の実施

STAMP/STPA はシステムだけではなく、それを運用する組織までを対象として安全性解析を行う場合もある [14]。本研究では安全性解析手法として STAMP を採用した。これは、避難指示システムには多くの構成要素があり、また構成要素間の相互作用が多く見られるからである。高梨ら [3] の調査によれば、平成 30 年 7 月豪雨の茨城県常総市において、防災行政無線を放送したところ、防災行政無線で放送した内容が聞き取りにくいための確認の電話などが殺到した。これにより市は電話対応に追われ災害対策本部機能不全の原因となった。この例が示すように、ある何かの構成要素の故障が発生したわけではないが、不適切な情報伝達が起きたことで適切に避難指示を発令することが出来なかった。

### Step1 損失とハザードの識別

STPA においては初めに解析対象とするシステムで受け入れられない損失を識別する。例としては人命の喪失やシステムの損傷、ミッションの喪失などがあげられる。本研究では、避難指示システムを対象とするため、損失を以下のように定義する。

**L1 人の負傷** 避難指示を受け取り、避難する住民などを人として、その人の負傷、喪失を受け入れられないものとする。

**L2 システムの喪失** システムの一部、もしくは全部が喪失することで、避難指示システムとして機能しなくなることを受け入れられないこととする。

つぎに、ハザードを定義する。ハザードとは、最悪ケースの環境条件で損失につながるようなシステムの状態または条件である。ハザードはシステムの状態や条件であり、システム外の環境の状態や条件を含まず、物理的なコンポーネントの詳細な状態を示さないようにする。これは、後の手順で目立たない原因を見落とさないようにするためである。また、ハザードは 1 つ以上の損失につながり、それぞれのハザードは結果として生じた損失と結びつく必要がある。さらに、ハザードを防ぐために満たす必要があるシステムの条件や状態を安全制約として、ハザードごとに定義する。本研究におけるハザードと安全制約の定義を表 4.1 に示す。

表 4.1: ハザードと安全制約

| ハザード | 安全制約 |
|---|---|
| H1 正しくない避難情報の提示 [L1,L2] | 被災の可能性があるときは正しい避難情報を提示しなければならない |
| H2 避難情報が提示されない [L1,L2] | 被災の可能性があるときは避難情報を提示し続けなければならない |
| H3 システムが過負荷を受ける [L3] | システムの応答時間は常に一定を下回らなければならない |

### Step2 コントロールストラクチャの作成

コントロールストラクチャとは、システムの安全制約の実現に関わるコンポーネント同士の制御とフィードバックの流れを説明するものである。コンポーネント同士の制御 (コントロールアクション) とフィードバックにより、コントロールループが形成される。例を図 4.2 に示す。

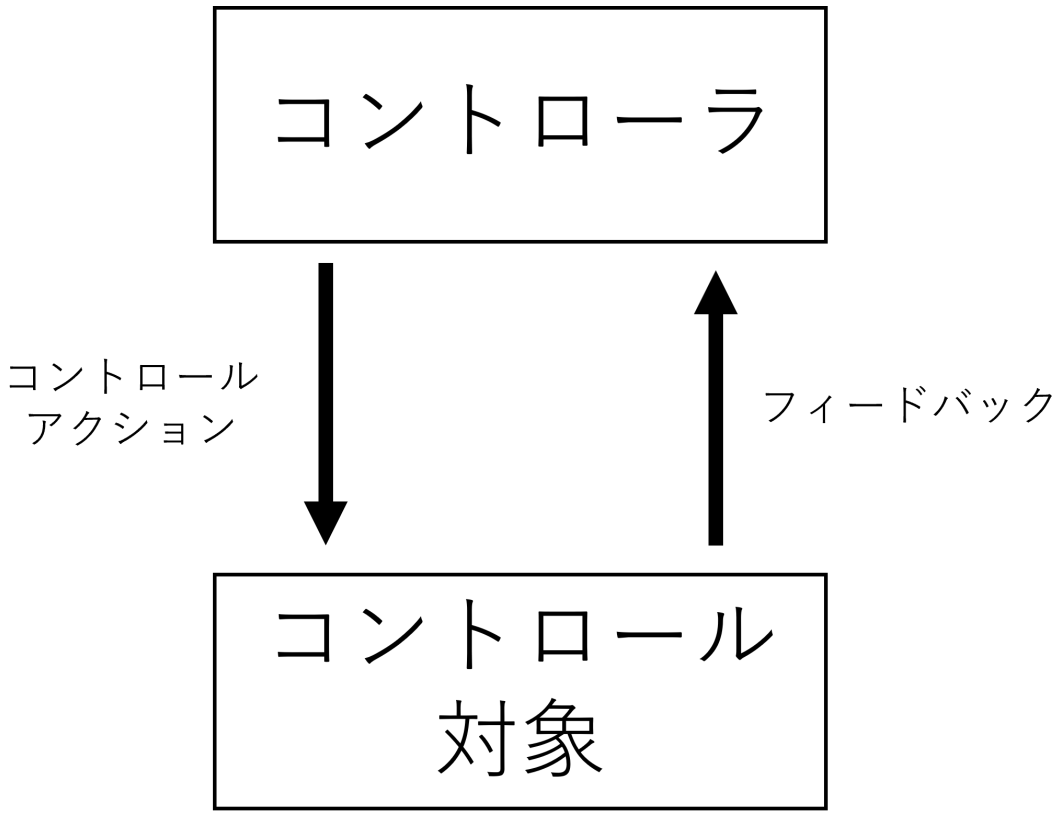
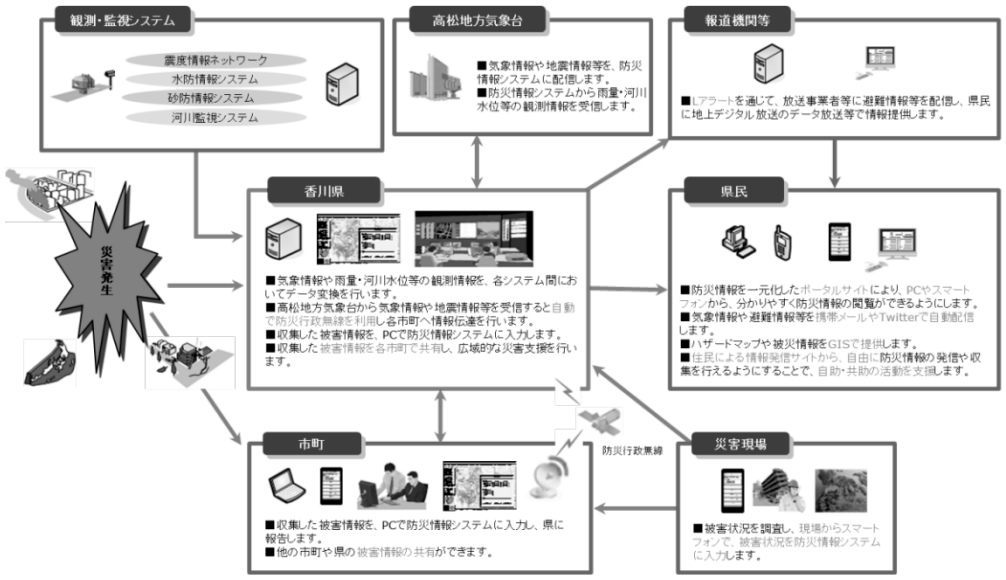



図 4.2: コントロールループを含むコントロールストラクチャ

コントロールストラクチャは対象とするシステムの構成を基に作成するため、ここでは避難指示システムの構成を考える。総務省情報流通行政局地域通信振興課の自治体における防災情報共有システムの導入に係る仕様書 [15] によると、香川県では県の防災情報や県内市町の災害情報等を迅速に収集・共有を行うとともに、住民へ L アラート、HP、携帯メール等のメディアを通じて情報提供するシステムが構築されている (図 4.3)。




出典: 総務省 [15]

図 4.3: 香川県防災情報システム

この構成では、観測監視システムや気象台から水位等の情報を香川県が受け取り、直接もしくは報道機関等を通じて県民へ防災情報を伝達する形となっている。本研究では、このような構成を想定し、コントロールストラクチャを作成した。具体的なコントロールストラクチャは想定する構成により異なるため 5 章で説明する。

### Step3 非安全なコントロールアクションの抽出

識別したコントロールアクションのうち、ハザードに繋がる（安全制約に違反する）非安全なコントロールアクション (UCA :Unsafe Control Action) を識別する。識別にはガイドワードである「与えられないとハザード」「与えられるとハザード」「早過ぎ、遅過ぎ、誤順序」「早過ぎる停止、長過ぎる適用」を活用した。ガイドワードは、解析に漏れがないように利用される言葉で、ガイドワードを解析対象で想定することで識別を行う。具体的な UCA は想定するコントロールストラクチャにより異なるため 5 章で示す。

### Step4 損失シナリオの識別

STPA HANDBOOK[14] によると、損失のシナリオは次の 2 つのタイプを考慮する必要がある。

(a) なぜ、非安全なコントロールアクションが起こるのか？

(b) なぜ、コントロールアクションは、不適切に実行される、または実行されず、ハザードに至るのか？

(a) では、

(1) 非安全なコントローラの動作

(2) 不適切なフィードバックと情報

などが原因として考えられる。
(b) では、

(1) コントロール経路を含むシナリオ

(2) コントロール対象のプロセスに関連するシナリオ

が考えられる。
この分類をもとに UCA ごとに損失シナリオの識別を行い、どのような場合、状態で UCA が発生するかを特定する。また、損失シナリオの作成には、はじめての STAMP/STPA[16] で紹介されている以下の 13 個のガイドワードを使用した。これらの step が完了すると、どのようなときに損失につながることがあるかという損失シナリオの出力が得られる。

1. コントロール入力や外部情報の誤りや喪失。
2. 不適切なコントロールアルゴリズム。（作成時の欠陥、プロセスの変更、誤った修正や適用）
3. 不整合、不完全、または不正確なプロセスモデル。不適切な操作。
4. コンポーネントの不具合。経年による変化。
5. 不適切なフィードバック、あるいはフィードバックの喪失。フィードバックの遅れ。

6. 不正確な情報の供給、または情報の欠如。測定の不正確性。フィードバックの遅れ。
7. 操作の遅れ。
8. 不適切または無効なコントロールアクション、コントロールアクションの喪失。
9. コントロールアクションの衝突。プロセス入力の喪失または誤り。
10. 未確認、または範囲外の障害。
11. システムにハザードを引き起こすプロセス出力。
12. アクチュエーターの動作が不十分
13. センサーの動作が不十分

## 4.3 シミュレータでの検証

安全性解析を活用するためにはSTPAで明らかにとなった損失シナリオをもとに対策を行う。しかしその対策が適当であるかどうかの検証は実際にシステムを稼働させる必要があると考えられる。システムの開発段階において稼働させることは難しいため、本研究では、シミュレータを用いて損失シナリオとその対策が適当であるかどうかの調査を行う。先行研究としてはSTPAの結果を基にSysMLでモデリングを行い、UPPAALでモデル検査を行うものがある[17]。この研究ではSTPAの結果の検証が可能となるが、防災ITシステムなどのシステム開発ではセンサなど現実にあるデバイスからデータを取得して動作の確認を行う必要がある。本研究では、現実のセンサや他のシステムとも連携ができるようにシミュレータの設計を行った。本研究では、Node-RED[18]をシミュレータ構築環境として選択した。Node-REDはフローベースのヴィジュアルプログラミング開発ツールであり、IoTの一部としてハードウェア、APIオンラインサービスを相互に接続するために開発された。Node-REDを選択した理由としては、

1. MQTTやREST APIによる他のサービスとの連携が容易。

2. STPA のコントロールストラクチャにおける構成要素を node, コントロールアクションとフィードバックを flow で表現することができ、STPA の結果からの変換が容易である。

の 2 点である。

## Node-RED でのシミュレータの実装

コントロールストラクチャにおける構成要素を node, コントロールアクションとフィードバックを flow で表現する。この場合、損失シナリオに基づいて node や flow の状態を変化させる必要がある。変化させる実装方法として以下の 2 つが考えられる。

(1) Node-RED でシナリオの読み込み、状態の変化を実装する。

(2) Node-RED とは別の環境でアプリケーションを作成してシナリオの読み込みを行い、そこから Node-RED の API を使用して node と flow の状態を変化させる。

(2) の方法はシナリオ読み込みが Node-RED に依存しないため柔軟な設定が可能となる。この方法で実装したところ、Node-RED の API を使用して node と flow の状態を変化させることが出来たが、Node-RED のバージョンアップにより node と flow を記述する json の形式が変化し動作しなくなることが判明した。json を書き換えることで動作するようになったものの、今後のアップデートにより破壊的変更が繰り替えされる可能性が考えられるため、本研究では、(1)Node-RED でシナリオの読み込み、状態の変化を実装する方針とした。シミュレータの構成図を図 4.4 に、実装したシミュレータを図 4.5 に、定義して使用するグローバルコンテキストのプロパティを表 4.2 に示す。

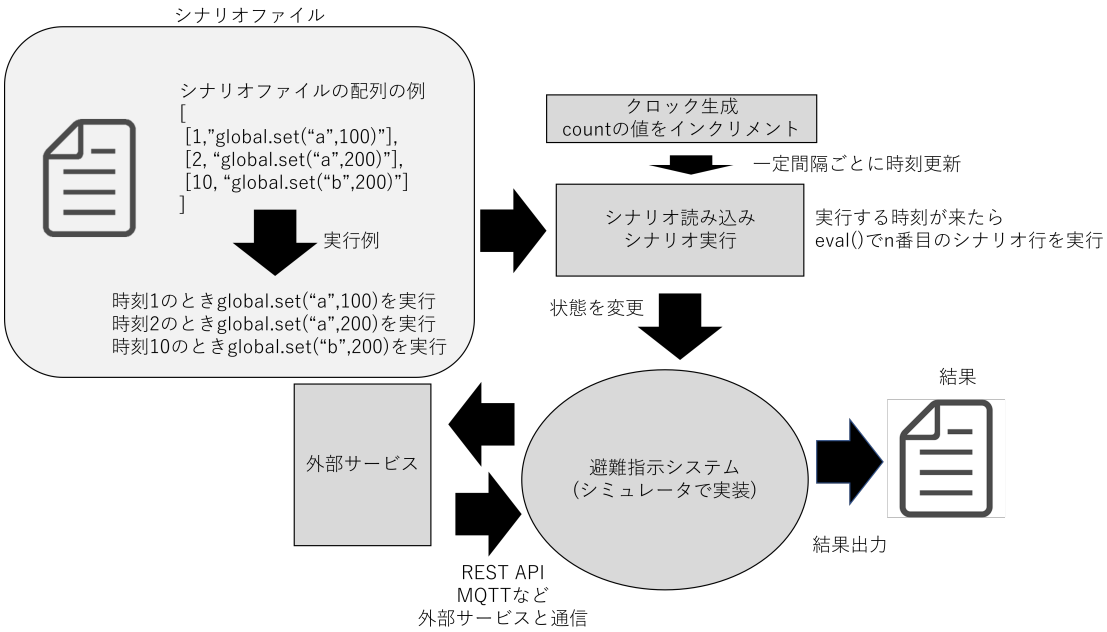



図 4.4: シミュレータの構成

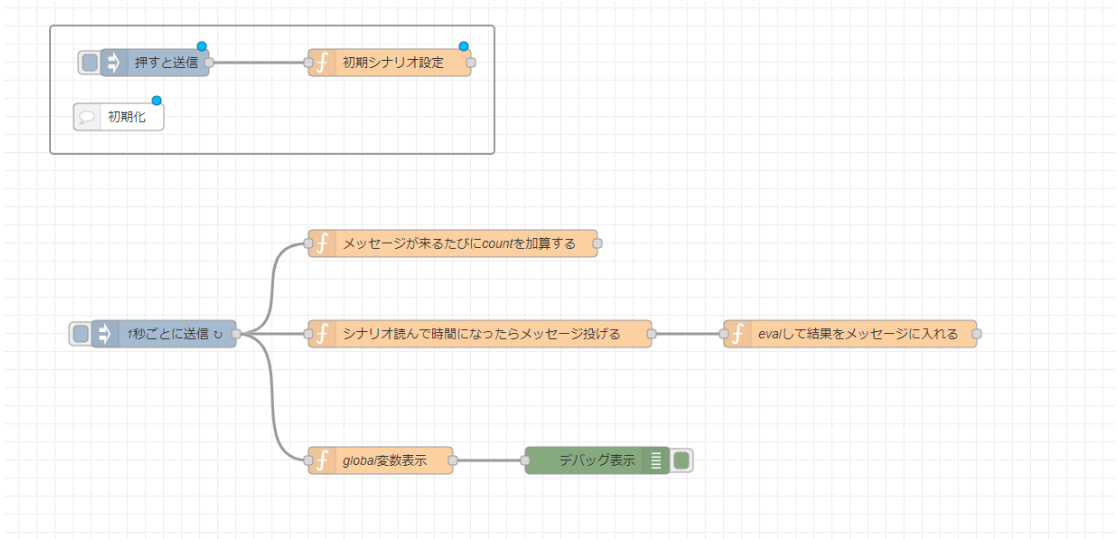



図 4.5: Node-RED によるシミュレータの実装

表 4.2: グローバルコンテキストの値

| プロパティ名 | 説明 |
|---|---|
| scenario | シミュレーションシナリオを json 形式で保持する |
| count | シミュレーション時間を保持する Int 型の値 |
| citizenEvacuated | 人が避難したかどうかを記録する Boolean 型の値 |

## シミュレータの実装

シミュレーション時間を定義し、一定間隔ごとに値をインクリメントすることでシミュレーションの進行を実装した。また、シナリオファイルは実行する時刻と実行するコードのペアの配列として定義し、シミュレーション時間がシナリオ配列の要素に達したとき、シナリオ中のコードを eval() 関数で実行する形とした。シナリオファイルの例を以下に示す。

[[10,”global.set(‘sensor1‘,200)”],[15,”global.set(‘sensor1‘,200)”]];

この例では、シミュレーション時間 10 のときに global.set(‘sensor1‘,200) を実行し、シミュレーション時間 15 のときに global.set(‘sensor1‘,200) を実行する。図 4.5 中の「１秒ごとに送信」ノードから１秒ごとに空の payload が送られ、「メッセージが来るたびに count を加算する」ノードでグローバルコンテキストの count をインクリメントする。「シナリオ読んで時間になったらメッセージを投げる」ノードでは、グローバルコンテキストの scenario 中にシミュレーション時間よりも大きいか同じ時刻があれば「eval して結果をメッセージに入れる」ノードに送り、シナリオ中のコードを実行する。これにより、シナリオファイルでグローバルコンテキストの値を変更し、対象とするシステムの状態を変更することができる。

## シナリオファイルの生成

シナリオファイルは STPA での損失シナリオをもとにして作成する。コントロールアクションがコントロール対象に届かないようにする場合は、届かないようにするかどうかを示すグローバルコンテキストの値を事前に定義し、コントローラとコントロール対象の flow の間に function ノードを追加して、グローバルコンテ

キストの値を使って flow を制御する。コントローラやコントロール対象の node 自体でグローバルコンテキストの値を読み取って flow を変化させる実装も可能である。また、変化させたいコントロールアクションの数だけグローバルコンテキストの値を定義することで、複数のコンポーネントの相互作用についても表現することができる。

### 4.3.1 STPA の結果からシミュレーションへの変換

本手法ではコントロールストラクチャを Node-RED のシミュレータ実装へ変換する。この変換では Node とコントロールストラクチャの構成要素、Flow とコントロールアクション、フィードバックが一対一に必ずしも対応するわけではない。なぜなら、コントロールストラクチャは実行可能なモデルではなく、抽象化されていたり、物理モデルと異なる場合があるからである。一方、シミュレータ実装はシナリオを実行するため、実行可能な形式である必要がある。また、コントロールストラクチャは STPA の分析で必要な情報のみ書かれているため、対象とするシステムの動作や構成を知ってる人間が補完しながら変換する必要がある。
さらに、UCA や損失シナリオからシミュレーションシナリオへの変換も、必要な情報を人間が補完しながら行う必要がある。シナリオは JavaScript の eval 関数で実行されるため、生成にはプログラミングの知識が必要であり、変換に要する時間はプログラミングの技能のレベルやシナリオの複雑さによって変化する。例として後述する単純な避難指示システムでの UCA13,16 を筆者が変換を行ったところおよそ 1 時間を要した。

### シミュレータの実行

実行は図 4.5 中の「１秒ごとに送信」ノードのボタンをクリックすることで開始する。実行結果として保存したい情報をグローバルコンテキストで定義し、「global 変数表示」ノードで表示することで結果の表示を行う。

# 第5章　実施例と評価

## 5.1　単純な避難指示システムの例

本節では避難指示システムとして単純な避難指示システムの例を示し、STPA を実施した結果を評価する。図 5.1 にコントロールストラクチャを示す。

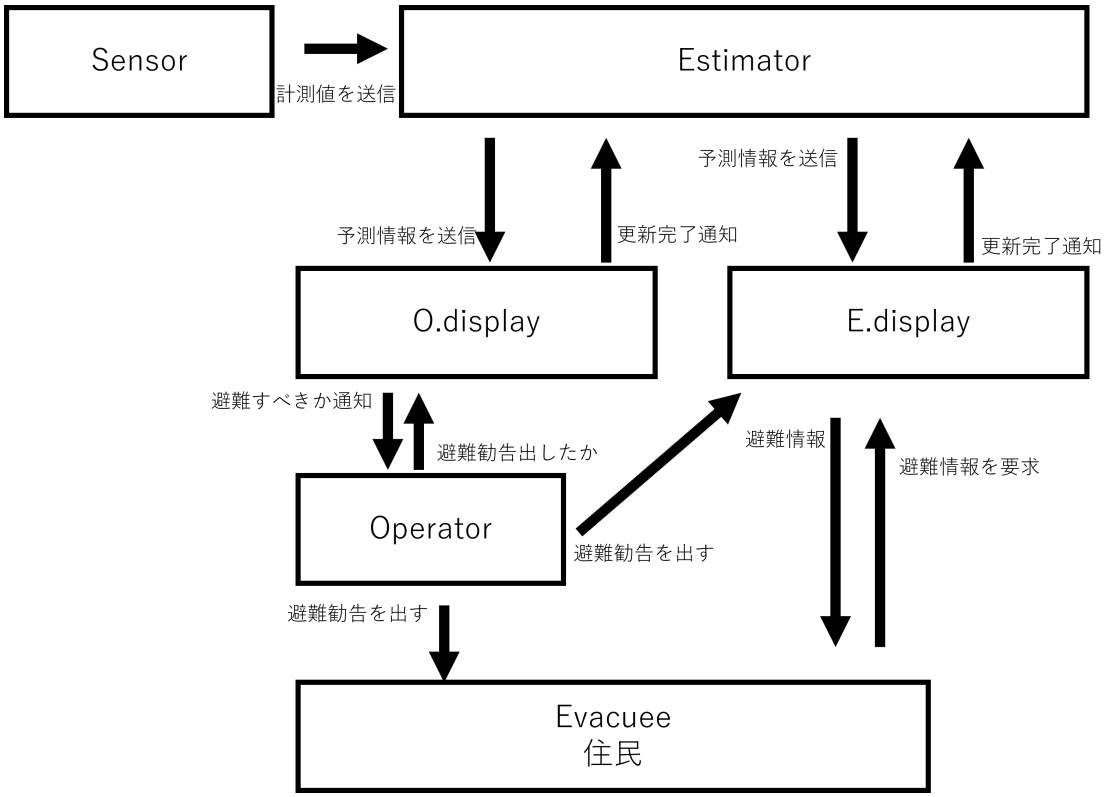



図 5.1: 単純な避難指示システムのコントロールストラクチャ

以下にコントロールストラクチャで登場する構成要素を説明する。

**Sensor**

水位計など、現実の物理量を測定するものであり、Estimator に計測値を送信する。

**Estimator**

計測値から気象状況の予測を行い、予測情報を O.display と O.display に送信して、情報が反映されたかどうかのフィードバックを受け取る。

**O.display**

気象の予測情報を Operator に提供し、避難指示を出すべきかどうかの通知を Operator に送る。Operator からは避難指示を出したかどうかのフィードバックを受け取る。

**E.display**

Evacuee から要求があった場合にフィードバックとして気象の予測情報と避難指示情報を提供する。Operator からの避難指示情報の更新があった場合は内部状態を更新する。

**Operator**

O.display から気象の予測情報を受け取り、避難指示を発令すべきかどうかの判断を行う。発令する場合は Evacuee と E.display に避難指示を送信する。

**Evacuee**

Operator から避難指示を受け取る。また、定期的に E.display に避難指示情報を要求し、フィードバックとして避難指示情報受け取る。

STPA での損失およびハザードは 4.2 章で示した表 4.1 として実施する。次に、識別したコントロールアクションからハザードにつながる UCA を識別した。表 5.1 に UCA 一覧を示す。

表 5.1: 単純な避難指示システムにおける UCA 一覧

| コントロールアクション | 与えられないとハザード | 与えられるとハザード | 早すぎ、遅すぎ、誤順序 | 早すぎる停止、長すぎる適用 |
|---|---|---|---|---|
| Operator → Evacuee | UCA-1: 住民が避難すべき状態のとき、避難指示を出さない。[H-2] | UCA-2: 住民が避難すべきでないとき、避難指示を出す。[H-1] UCA-3: Evacuee に送る避難指示が多すぎる。[H-3] | UCA-4: まだ避難指示を出すべきでないとき、避難指示を出す。避難指示を出すのが遅すぎる。 [H1,H2] | UCA-5: 避難指示を解除すべきなのにしない。避難指示を解除すべきでないのに解除する。[H1,H2] |
| Evacuee → E.display | UCA-6: Evacuee が避難すべき状態のとき、避難指示を出さない。[H2] | UCA-7: Evacuee が避難すべきでないとき、避難指示を出す。[H2] | UCA-8: まだ避難指示を出すべきでないとき、避難指示を出す。避難指示を出すのが遅すぎる。 [H1,H2] | UCA-9: 避難指示を解除すべきなのにしない。避難指示を解除すべきでないのに解除する。[H1,H2] |
| Operator → O.display | UCA-10: Evacuee が避難すべき状態のとき、判断に必要な情報が足りない。[H2] | UCA-11: 判断に必要な情報が多すぎて、Oparator が処理しきれない。 [H3] | UCA-12: O.display から得られる情報が古すぎる。[H2] | オペレータが対応する必要がないのに働く。 |
| Estimator → O.display | UCA-13: Evacuee が避難すべき状態の時、予測情報を送信しない。[H2] | UCA-14: Evacuee が避難すべきでないとき、不正な予測情報を送信する。[H1,H2] | UCA-15: 気象状況が変化しているときに、予測情報が古すぎる。[H1,H2] | |
| Estimator → O.display | UCA-16: Evacuee が避難すべき状態の時、予測情報を送信しない。[H2] | UCA-17: Evacuee が避難すべきでないとき、不正な予測情報を送信する。[H1,H2] | UCA-18: 気象状況が変化しているときに、予測情報が古すぎる。[H1,H2] | |

分析の最後に、UCA を各コントローラの動作上の制約 C に変換する。さらに、HCF（Hazard Causal Factor）を特定し、HS(Hazard Scenario) を識別する。制約とハザードシナリオを以下に示す。

**UCA1** C-1 Evacuee が避難すべき時は Evacuee に避難時指示を伝えなくてはならない。

**HS1-1**

コントロール入力の喪失。指示が通信路の故障などで消えてしまう。

**HS1-2**

Operator のミス (出し忘れ)。

**HS1-3**

O.display からの情報が不正確。

**UCA2** C-2 Evacuee が避難すべきでないときは避難指示を出してはいけない。

**HS2-1**

Operator のミス。誤って避難指示を出してしまう。

**HS2-2**

O.display からの情報が不正確。

**UCA3** C-3 避難指示を更新するときは一定時間の間隔を開けなくてはならない。

**HS3-1**

Operator のミス。誤って避難指示を出し過ぎてしまう。

**UCA4** C-4 避難指示は適切なタイミングで発令する必要がある。

**HS4-1**

Operator の動作の遅れ。

**HS4-2**

O.display からの情報が不正確。

**UCA5** C-5 避難指示は適切なタイミングで解除する必要がある。

**HS5-1**

Operator の動作の遅れ。

**HS5-2**

O.display からの情報が不正確。

**UCA6** C-6 Evacuee が避難すべき時は Evacuee に避難時指示を伝えなくてはならない。

**HS6-1**

コントロールの喪失　指示が通信路の故障などで消えてしまう。

**HS6-2**

E.display の情報が不正確。

**UCA7** C-7 Evacuee が避難すべきでないときは避難指示を出してはいけない。

**HS7-1**

Estimator からの情報が不正確。

**UCA8** C-8 避難指示は適切なタイミングで発令される必要がある。

**HS8-1**

Estimator からの情報が不正確。

**HS8-2**

E.display の動作の遅れ。

**UCA9** C-9 避難指示は適切なタイミングで解除される必要がある。

**HS9-1**

Estimator からの情報が不正確。

**HS9-2**

E.display の動作の遅れ。

**UCA10** C-10 O.display は判断に必要な情報を不足なく Operator に伝える必要がある。

**HS10-1**

Estimator からの情報が不正確。

**HS10-2**

O.display の動作の遅れ。

**UCA11** C-11 O.display は Operator が処理できない量の情報を提示してはいけない。

**HS11-1**

不適切なフィードバック (オペレータの判断に必要のない情報を含んでいる)。

**HS11-2**

不適切なコントロールアルゴリズム (要求に対して供給が多すぎる)。

**UCA12** C-12 O.display は最新情報を提供する必要がある。

**HS12-1**

Estimator からの情報が喪失。

**HS12-2**

O.display の動作の遅れ。

**UCA13** C-13 Estimator は被害が予測されるときに予測情報を送信しなければならない。

**HS13-1**

Estimator からの情報が喪失。

**HS13-2**

Estimator の動作の遅れ。

**UCA14** C-14 Estimator は常に正しい予測情報を送信しなければならない。

**HS14-1**

Estimator のアルゴリズムの欠陥。

**HS14-2**

Sensor の計測値の誤り。

**UCA15** C-15 Estimator は最新の予測情報を送信しなければならない。

**HS15-1**

Estimator の動作の遅れ。

**HS15-2**

Sensor からの入力の喪失。

**UCA16** C-16 Estimator は被害が予測されるときに予測情報を送信しなければならない。

**HS16-1**

Estimator からの情報が喪失。

**HS16-2**

Estimator の動作の遅れ。

**UCA17** C-17 Estimator は常に正しい予測情報を送信しなければならない。

**HS17-1**

Estimator のアルゴリズムの欠陥。

**HS17-2**

Sensor の計測値の誤り。

**UCA18** C-18 Estimator は最新の予測情報を送信しなければならない。

**HS18-1**

Estimator の動作の遅れ。

**HS18-2**

Sensor からの入力の喪失。

## STPA の評価

ここで UCA3 に着目する。これは、Operator から Evacuee に送られる避難指示が多すぎることで、どの避難指示が正しいのかの判断が出来なくなる状態であると考えられる。この事象は実際に発生している。避難情報などを携帯電話へ送信するエリアメールというサービスがある。2022 年 1 月 15 日トンガ沖噴火による津波予想において、エリアメールの送信回数が神奈川で 600 回を超えた [19]。これは 16 日深夜から明け方の間に 5〜10 分間隔でエリアメールが送信されおり、他の都道府県では多くても 40 件程度であった。本章で想定した単純な避難指示システムにより、実際のシステムで起きる問題の発見が可能である。

## シミュレーションの実施

本章で想定した単純な避難指示システムを、Node-RED 上で実装し、STPA で判明したシナリオを実行し、実際に損失が起きるのか調査を行う。Node-RED での単純な避難指示システムの実装を図 5.2 に示す。また、各部について以下で説明を行う。

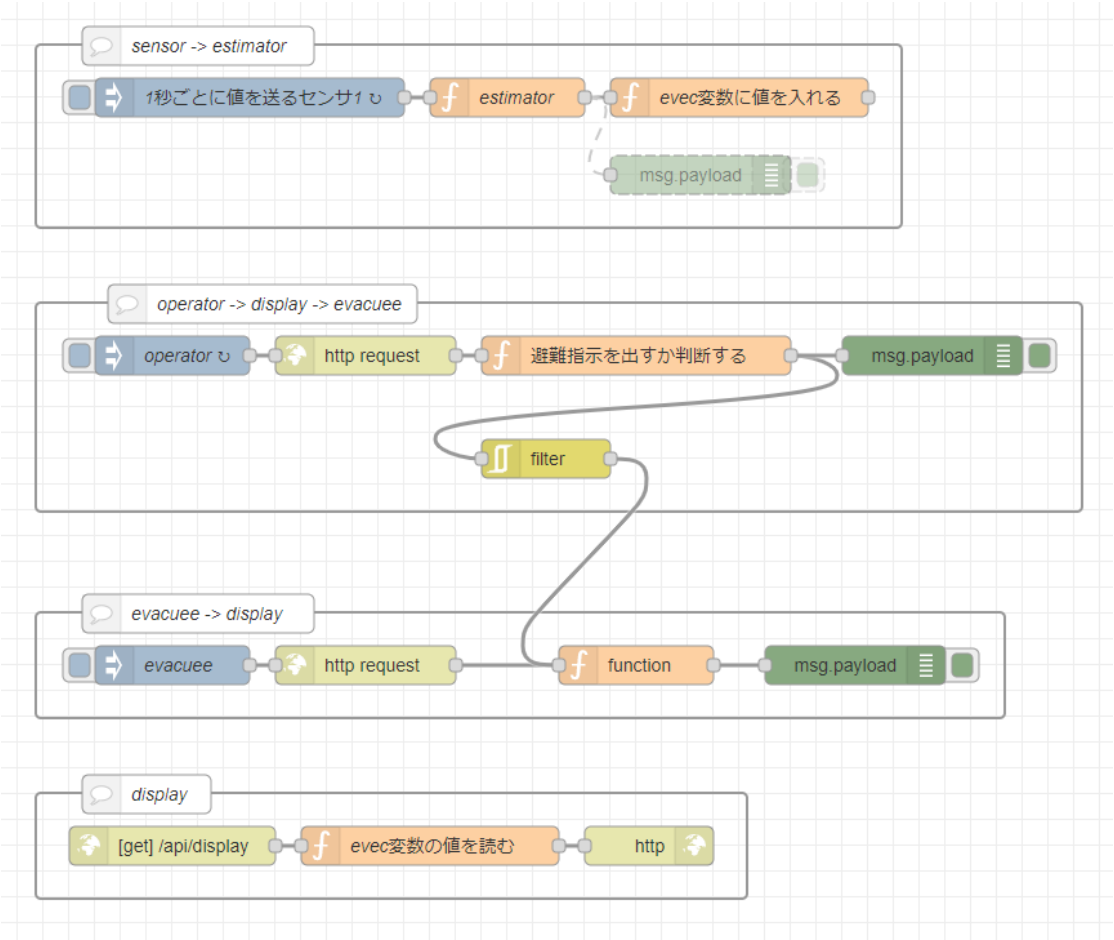



図 5.2: 単純な避難指示システムのシミュレータ実装

**Sensor → Estimator**

ここでは、センサが水位を計測しているとして、「1 秒ごとに値を送るセンサ」から水位として waterLevel を送信する。Estimater で閾値 (ここでは 100 とする。) を超えた場合はグローバルコンテキスト evac の値を true に、そうでない場合は false に変更する。これにより、Estimator の予測結果を保存する。

**Display**

簡単のため、コントロールストラクチャ上での O.display と E.dsisplay を統合し、Display とする。Web サーバとして機能し、アクセスがあった場合は evac の値を読み、避難すべきかどうかを真偽値として返す。

**Evacuee**

避難者をシミュレートする。Display にアクセスして避難すべきかどうかの判断を行い、避難した場合はグローバルコンテキスト citizenEvacuated の値を true にする。

**Operator → Display → Evacuee**

オペレーターをシミュレートする。Display にアクセスして避難指示を出すか判断し、出す場合は flow の payload を変化させ、Filter ノードを中継して Evacuee に伝える。

## シミュレーションシナリオと実行結果

### 1. 正常に動作する場合

シミュレーションのシナリオとして、センサの waterLevel を初期値 0 として、シミュレーション時刻 10 のときに 200 に変化するシナリオを実行する。これにより、シミュレーション時刻 11 のときに Evacuee が Display にアクセスして避難情報を受け取り、Operator が Evacuee に避難指示を出すため citizenEvacuated が true になり、住民が避難するという結果が得られる。この結果から問題なく避難が成功していることがわかる。

### 2. 正常に動作せず、損失につながる場合

Estimater の予測アルゴリズムが誤っている場合を想定して、waterLevel の閾値を本来 100 と設定すべきところを誤って 300 に変更したとする。このとき、1 と同様のセンサの waterLevel を初期値 0 として、シミュレーション時刻 10 のときに 200 に変化するシナリオを実行する。このシナリオは UCA-13 と UCA-16 が発生していると仮定したものである。結果として、シミュレーション時刻 11 のときにも避難指示が発令されず、citizenEvacuated が false となり、避難できないという結果が得られる。これは、STPA での「H2 避難情報が提示されない」というハザー

ドが起きているといえる。したがって、UCA が発生しているとするシナリオを実行した場合、ハザードが発生することが確認できる。

## 5.2 縮退運転がある例

避難指示システムでは、2.2 章で述べた通り、縮退運転が重要であるといえる。また、避難指示システムは災害が発生しそうな時に稼働することが求められるが、水害など、災害の発生確率が十分に低いときは一部機能を停止することでコストの削減が期待できる。本節では、単純な避難指示システム構成をもとにして縮退運転および機能停止を考慮したシステム構成を考える。このシステムにおいて、STPA を実施して UCA を識別し、単純な避難指示システムの場合と比較することで、機能が追加された場合にどのように解析結果が変化するかを明らかにする。
ここでは状態を管理する State controller という要素を追加し、縮退運転に必要となる状態の管理を行うモデルで検討を進める。他の方法としては、それぞれの構成要素が状態を持ち、自らの判断で縮退運転が必要かどうかの判断を行うモデルも存在するが、単純な避難指示システムとの比較を容易にするため前者のモデルを使用している。図 5.3 に縮退運転を想定したコントロールストラクチャを示す。また、追加したコントロールストラクチャの要素について以下で説明を行う。

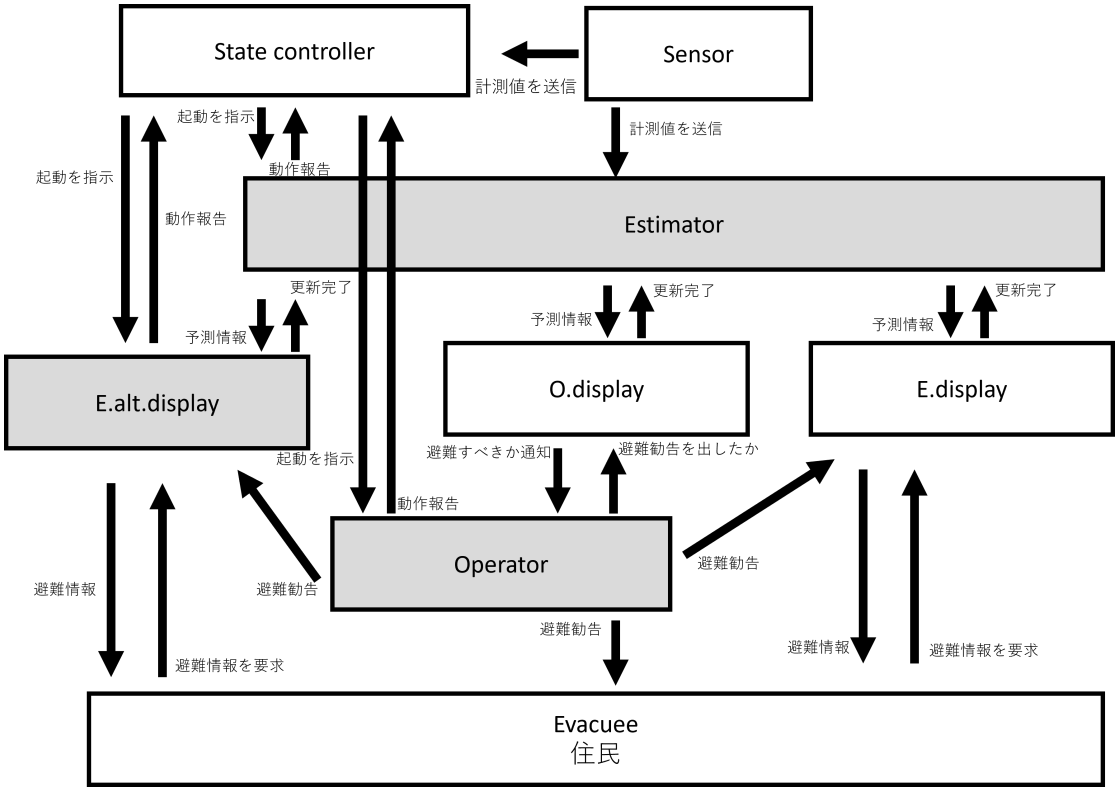



図 5.3: 縮退運転するシステムのコントロールストラクチャ

**State controller**

Estimator,Operator の起動と停止を管理を行う。Sensor から計測値を受け取り、避難指示システムを稼働させる必要があるときに Estimater と Operator の起動を行い、必要なくなった場合に停止を行う。また、E.display の動作が不十分なとき、E.alt.display を起動し、縮退運転を行う。

**E.alt.display**

E.display の動作が遅い、または動作しない場合に動作する。E.display の提供している情報のうち、重要な部分のみに限定して配信することで、避難に必要な最低限度の情報を Evacuee に提供する。

次に、識別したコントロールアクションからハザードにつながる UCA を識別した。単純な避難指示システムのコントロールアクションと同様のコントロールアクションでは、同様の結果が得られたため省略し、新たに識別された UCA を表 5.2 に示す。

表 5.2: 縮退運転がある例における UCA 一覧

| コントロールアクション | 与えられないとハザード | 与えられるとハザード | 早すぎ、遅すぎ、誤順序 | 早すぎる停止、長すぎる適用 |
|---|---|---|---|---|
| State controller → Estimator | UCA-1: 水位予測すべき時に、起動指示を出さない。[H2] | UCA-2: 水位予測すべき時に、終了指示を出す。[H2] | UCA-3: 水位予測すべき状態に変化したときに、起動指示を出すのが遅い。[H2] | UCA-4: 水位予測が必要ない状態になる前に、終了指示を出す。 [H2] |
| State controller → Operator | UCA-5: 避難指示が出そうなときに、起動指示を出さない。[H2] | UCA-6: 避難指示が出そうなときに、終了指示を出す。[H2] | UCA-7: 避難指示が出そうな状態に変化したときに、起動指示を出すのが遅い。 [H2] | UCA-8: 避難指示が必要ない状態になる前に、終了指示を出す。[H2] |
| State controller → E.alt.display | UCA-9: 縮退運転すべき時に、起動と切り替え指示を出さない。[H2] | UCA-10: 縮退運転が必要ないときに、起動と切り替え指示を出す。[H1] | UCA-11: 縮退運転が必要になる前に、起動と切り替え指示を出す。[H1] | UCA-12: 縮退運転が不要になったあとに、終了と切り替え指示を出すのが遅い。[H1] |
| Sensor → State controller | UCA-13: 水位予測が必要な時に計測値を送信しない。[H2] | UCA-14: 誤った計測値を送る。[H1,H2] | UCA-15: 計測値が古すぎる。[H1,H2] | UCA-16: 計測値の計測間隔が広すぎる、狭すぎる (時間分解能が不適切)。[H1,H2,H3] |

## 考察

単純な避難指示システムと比較すると、新たに追加されたコントロールアクションでのみUCAが追加で識別されている。このことから、避難指示システムにおいてSTPAで分析を行う場合には、システム構成を変化させた場合においても変更部分コントロールアクションを分析することで、網羅的に分析を行うことができると考えられる。また、新たに識別されたUCAではシステムの状態が変化したときに起きるコントロールアクションがある。シミュレータで実行する際はこの状態を保持する必要がある。STPAで状態を管理する方法として、先行研究としてSTPAとFSM(Finite State Machine)を組み合わせた手法が提案されている[20]。提案によれば、自動運転車の状態をFSMで表して、UCAと状態が遷移する条件を識別することで、安全でないコントロールアクションを明確にすることができる。したがって、状態が変化するようなシステムを扱う際はシミュレータでの実装としてFSMで状態を保持し管理することがシミュレーションシナリオの作成や実行で有用であると考えられる。さらに、網羅的にシミュレーションシナリオを作成することはコストが高い。そのため損失につながる重要なUCAを優先的に分析し、シミュレーションシナリオを作成する必要があると考えられる。UCAに優先順位をつける研究として、影響度によるハザードの点数評価を行い、UCAとハザードのトレーサビリティを利用してUCAに優先順位をつける手法[21]がある。この研究では、手法を放射線治療装置に適用することで安全性の高いシステムへの設計変更を可能としている。本研究においても、UCAに優先順位をつけてシミュレーションシナリオを作成することで、分析コストの削減が可能となると考えられる。

# 第6章　まとめ

本論文では、防災 IT システムの特徴の分析を行い、実際の防災 IT システムの構成を考慮しながら、STPA で安全性解析を行うことで、防災 IT システムの耐災害性について問題点が明らかになることを示した。また、実際の構成と想定した構成が異なる場合においても STPA による安全性解析の結果は一部有効であることが判明した。さらに、稼働しているシステムとも連携可能なシミュレーション環境を提案することで、STPA の解析結果の活用方法を示した。

## 6.1　本研究の成果

本研究により、防災 IT システムの耐災害性を向上させることが出来ると考えられる。また、提案するシミュレーション環境では条件を変えてシナリオ実行することで試行錯誤を繰り返すことが出来るため、システムの構成を変えて繰り返し実行することでより良いシステム構成を見つけるために役立つと考えられる。

## 6.2　社会的な意義

現代社会や都市は激甚化、頻発化する災害に対応するために、被害を前提としてそれを最小限にする能力であるレジリエンスを向上されることが求められている。これは、防災 IT システムにおいても同様であり、本研究では、防災 IT システムが縮退運転をすることによりレジリエンス能力が向上することを示した。レジリエンスは社会や組織においても使用される用語であり、本研究では、対象を防災 IT システムに絞って分析を行ったが、防災 IT システムを取り巻く環境も考慮しながら、レジリエンス能力を向上させることで、さらなる耐災害化が可能となると考えられる。具体的には、防災 IT システムの開発・運用を行う組織も STPA の解析対象とすることで、システムだけでなくシステムの開発・運用の安全性が確保される。この仕組みを社会に取り入れることで、耐災害性の向上が期待できる。

# 参考文献


[1] 国交省. 令和３年版国土交通白書

[2] 淡路 祥成, 廣田 悠介, 白岩 雅輝, 徐 蘇鋼, 久利 敏明, and 大和田 泰伯.NICT におけるネットワークの耐災害性強化の取り組み, 電気学会誌,2020 年 12 月号,p. 774-777

[3] 高梨 成子 and 坂本 朗一. 豪雨災害時の市町村災害対策本部の意思決定における情報処理の成功・失敗事例の分析及び対策に関する研究, 災害情報,2019 年 17 巻 2 号 p. 213-225

[4] 中島 尚紀, 竹澤 伸久, 坪根 直毅, 廣澤 祥泰, 中原 克彦 and 奥田 裕明,PSA を利用したオープンシステムの信頼性評価, 第 17 回秋季信頼性シンポジウム (2005)

[5] 岡本　俊彦, 洪水時の命を守る確実な避難の実現に向けて,JICE REPORT 第 32 号,pp.76-79(2018)

[6] 長妻 努, 今中 秀郎 and 井上 真杉.ICT を活用した災害対応と ICT の耐災害性に関する国際標準化動向, 電子情報通信学会　通信ソサイエティマガジン,15 巻 3 号 p. 228-235(2021)

[7] 林　春男, 災害レジリエンスと防災科学技術, 京都大学防災研究所年報第 59 号 A(2016)

[8] 日本規格協会, ディペンダビリティ　マネジメント―第４－３部：システム信頼性のための解析技法―故障モード・影響解析 (FEMA) の手順,JIS C 5750-4-3：2021 (2021)

[9] 日本規格協会, ディペンダビリティ　マネジメント―第４－４部：システム信頼性のための解析技法―故障の木解析 (FTA),JIS C 5750-4-4:2011 (2011)

[10] H.W. ハインリッヒ, ハインリッヒ産業災害防止論, 海文堂出版 (1982)

[11] Reason,J.,Human Error: Models and Management.British Medical Journal, Vol. 320, No. 7237,pp-768-770 (2000)

[12] 山田 茂, ソフトウェア信頼性の基礎, 共立出版 (2011)

[13] Leveson, Nancy,A New Accident Model for Engineering Safer Systems,Safety Science 42(2004) pp,237-270 (2004)

[14] Leveson, Nancy and John P.Thomas,STPA HANDBOOK 日本語版,MIT Partnership for Systems Approaches to Safety and Security (2018)

[15] 総務省情報流通行政局地域通信振興課, 自治体における防災情報共有システムの導入に係る仕様書,https://www.soumu.go.jp/menu_seisaku/ictseisaku/ictriyou/shiyousho.html,(参照 2022/01/28)

[16] 荒木 啓二郎, はじめての STAMP/STPA〜システム思考に基づく新しい安全性解析手法〜, 独立行政法人 情報処理推進機構（IPA),(2016)

[17] Fellipe Guilherme Rey de Souza,Combining STPA with SysML Modeling,2020 IEEE International Systems Conference(2020)

[18] Node-RED,https://nodered.org/,(参照 2022/01/28)

[19] 篠原修司,「うるさすぎて眠れない」神奈川県が緊急速報エリアメール夜中に十数回発信。通知オフ後の再有効化忘れずに,Yahoo ニュース,https://news.yahoo.co.jp/byline/shinoharashuji/20220116-00277593,(参 照 2022/01/28)

[20] Xingyu Xing,Tangrui Zhou,Junyi Chen,Lu Xiong, and Zhuoping Yu.A Hazard Analysis Approach based on STPA and Finite State Machine for Autonomous Vehicles,2021 IEEE Intelligent Vehicles Symposium (2021)

[21] 山口 晋一, システム理論に基づく安全解析手法（STAMP/STPA）の要件定義工程への適用評価, 慶應義塾大学大学院システムデザイン・マネジメント研究科システムデザイン・マネジメント専攻博士学位論文 (2019)

# 謝辞

研究を進めるにあたり、研究の進め方などご指導いただいた主指導教員の篠田陽一教授に感謝申し上げます。ゼミ発表などで意見をいただけたことで研究の進め方について多くの点を学ぶことができました。
丹康雄教授には、インターンシップ指導教員担当していただき、中間発表会などの場において研究に関して多くの助言をいただきました。感謝申し上げます。
ARIA の研究グループとして京都大学　廣井慧准教授には防災の専門家という立場から多くの意見を頂けただけではなく、研究自体のサポートも頂きました。ここに深く感謝申し上げます。
また、篠田研修了生の小比賀亮仁博士には論文の添削や研究方針など多くの相談に乗っていただきました。感謝申し上げます。
本研究室修了生の渡邊司揮氏、油布翔平氏、吉原昴司氏、知念研究室修了生の門脇真之佑氏、古寺雄馬氏には研究に関する様々な議論や研究生活を送る上での多大なご助力を頂きました。中でも修士 1 年時の輪講では多くのご指導をいただきました。深く感謝いたします。
本研究室の博士前期課程の馬越紘氏、本間可楠氏、梅内翼氏、岡田真一氏、片岡拓海氏とは同じ研究室の同期として研究内容に関する活発な討論など、研究を進めるため互いに切磋琢磨することが出来ました。深く感謝申し上げます。