大規模・複雑化した組込みシステムのための障害診断手法

モデルベースアプローチによる事後 V&V の提案

独立行政法人情報処理推進機構
技術本部 ソフトウェア高信頼化センター
ソフトウェア高信頼化推進委員会
障害原因診断 WG

（Ver. 2.0）

2016 年 3 月

独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan

本書は 2014 年度に公開した「大規模・複雑化した組込みシステムのための障害診断手法 〜モデルベースアプローチによる事後 V&V の提案〜 Ver. 1.0」の改訂版である。

今年度、当 WG では主に、要求仕様のモデル化による理解と障害原因の診断手法の検討や STAMP/STPA 手法の適用、Simulink を用いた新たなサンプルシステムの開発について取り組み、それらの結果を Ver. 2.0 として本書に採り込んだ。

Ver. 2.0 で大きく追加・変更した記述部分は以下となる。

- 4.2 節：システム記述言語 SysML によるシステム要求記述とハザード分析
- 4.3 節：STAMP 構成要素であるコントロールストラクチャーなどの SysML による記述
- 5.3 節：STAMP/STPA による解析事例と従来法である FTA などとの比較
- 6.3 節：障害原因仮説の妥当性の推定しと絞り込み
- 付録：要素技術検証のサンプルシステムとなる倒立二輪車の制御システムの開発

はじめに

製品・制御システム（いわゆる組込みシステム）の大規模化・複雑化、インターネットとの統合などに伴って、その障害が、人間へ危害や大規模な経済的ロスなどの社会への大きな影響を与える可能性が広がっている。これを未然に防ぐために、多くの設計技法や国際規格が提案されているが、それでも障害が起こっているのが現実である。旧来の安全解析技法では、コンピューターによる複雑な制御機能を持つ大規模システムの安全設計に対応できなくなってきてもいる。また、これらの障害の原因に関して、社会への強い説明責任が求められ、その説明如何で、製品の死命をも決せられかねない時代でもある。

このような現実を踏まえると、起こった障害を、製造者の責任として追及するだけではなく、産業界の共通課題として、第三者の立場として、障害の原因を迅速に追究し、再発防止のための提言を行う体制が必要とされる。

本報告書は、特に、製品・制御システムのソフトウェアならびに付随するネットワークコミュケーションシステムの障害に注目し、それの原因を追究し、解決への提言を行う方法論として、「事後 Verification & Validation（V&V)」という考え方を提案している。この V&V という考え方は、本来、製品開発段階で用いられるものであるが、設計時点での想定不足を見つけるという視点で考えると、障害を起こしたシステムの原因診断にも使える。また、障害と原因の関係を体系的に整理することで、社会への説明責任を果たすことにもつながるだけでなく、将来の改善設計に役立つ知見が得られ、新たな設計ツールの開発・提供も期待できる。

本提案では、整合性をもった事後 V&V の方法論を確立するために、近年着目されているモデルベースアプローチを中核の考え方として利用する。「モデル」という形でシステムの機能を抽象化・階層化して理解し、そこに起こりうる障害をシステマティックに把握するという考え方である。大規模な製品・制御システムの障害では、実装設計のミスだけでなく、要求段階の仕様ミス、市場での運用ミス（ヒューマンエラーなど)、長期運用での環境変化への対応ミス、セキュリティアタックによる悪意のある侵害など、多様な原因が考えられるため、このような体系的な考え方が必要とされる。さらには、製造者以外の利害関係者（ステークホルダー）の間の共通理解を得るための方法としても期待できる。

本書は、製品・制御システムに関わる開発者、事業者、管理監督者などを読者として想定している。事後 V&V の体系を説明するとともに、そこで用いられる要素技術を具体的なケーススタディをとおして説明することで、障害診断に関わるステークホルダーの理解を得ることを目指すものである。

目次

図表目次

1. 障害原因診断の問題意識

製品・制御システム（いわゆる組込みシステム）の大規模・複雑・複合化が急速に進んでいる。組込みシステムのソフトウェアは、その規模（ソースコード行数）が 2000 年から 2010 年の 10 年間で 3〜5 倍程度に増大したという調査結果があり、特に自動車向けソフトウェアでは 5〜10 倍の規模増大が見られる[刀川 2011]。

ソフトウェア規模の増大には複数の要因が考えられるが、組込みハードウェアの性能向上が主要なものの一つであろう。近年の組込み機器向け MPU（Micro-Processing Unit）は、省電力化のためピーク性能は PC 向けのものに劣るが、そのアーキテクチャーや機能は遜色がない。MPU が多量のメモリーを管理できるようになり、メモリーデバイスの集積技術や実装技術も向上したことから、1GB 以上のメモリーを搭載する組込み機器も市販されている。さらに、今日では多くの組込み機器がネットワークに接続されるようになっている。

ハードウェアの性能向上とネットワーク化によって、組込みシステムは従来の単一装置による単独システムから複数の機器やソフトウェアが協調する複合システムになっている。複合システムでは必然的にシステム間インターフェース（I/F）が必要となり、システム間 I/F の増加が今日の組込みシステムをより複雑なものにしている。

システムとソフトウェアが大規模・複雑・複合化したことによって新たな課題や問題が生じている。例えば、マーズ・クライメイト・オービターの事例[1] は、設計・開発時のコミュニケーション不足が原因であるが、サブシステム間での I/F 齟齬の典型的なものであると言える。また、複数のサブシステムが協調するような大規模システムでは、システム全体の動作を把握できる人を置くことが難しいという問題としても捉えられるだろう。日本の組込みソフトウェア開発体制は米国や欧州と比較して外部委託率が高いと言われている。近年の調査では外部委託率は低くなっているものの、こういったこともシステム全体の俯瞰を難しくする一因となろう。

さらに、自動車のブレーキ制御やステアリング制御など（X-by-wire）に見られるように、従来は電気的、機械的に直接的な手段で制御されていたものが組込みソフトウェアによって間接的に制御される例が増えている。組込みシステムをこのような場所で使用するにはソフトウェアに高度な信頼性が求められる。ソフトウェアを製作する者、それを運用する者に求められる社会的責任（事故発生時の説明責任など）も増大していると言え、製造者の社会的責任は 2009〜2011 年のトヨタ車リコール問題で大きな話題となった。

また、ネットワーク化された組込み機器では、プライバシーやセキュリティ、風評問題など人間社会との関与が生じることも特徴である。ネットワーク家電がプライバシーに影響を与えること（2015 年のサムスン製 TV の事例[2]）や迷惑メール送信の踏み台にされるなど、セキュリティリスクとなり得る事例が生じている。

今日では製品・制御システムに事故が生じた場合、その原因調査は容易ではない。組込み系開発企業では、エンタプライズ系ソフトウェア開発企業（ベンダー）に比較して、ソフトウェア不具

---

[1] NASA が火星の気象観測のために打ち上げた探査機が予定と異なる軌道で火星に接近して消失した事故（1998 年 12 月発生）。探査機の一部のソフトウェアではメートル法、別の部分ではヤード・ポンド法を用いていたことが原因。

[2] サムスンのスマート TV（音声認識機能付きテレビ）のプライバシーポリシーで「テレビの前で行なわれた会話が音声認識機能を通じて第三者に送信される」とされたことがプライバシー問題として報道されたもの。サムスンは誤解があったとしてプライバシーポリシーを改訂した。

合が生じた際の障害情報の共有、原因調査結果の公表・情報公開や独立した組織による原因調査といった他の組織を巻き込んだ再発防止策が進んでいないとの調査結果もある[IPA2013]。

事故は複数の要因が絡み合って生じたものと考えられ、人命に関わるソフトウェアの製造者責任、立法の問題、社会に対する説明責任、また風評被害や訴訟費用の負担増大の恐れなど様々な論点から評することができる。加えて、ソフトウェアの再利用や過去に使用実績のあるソフトウェアに対する過度な信頼など、ソフトウェア開発の問題だけでなく開発者と運用者との意思疎通の問題、さらには政府・監督省庁に関係した問題が含まれることもある。

障害原因診断においては原因の究明を目指すだけでなく社会的な責任の遂行のために、根拠に基づき広く社会が合意、納得できるような説明のできる調査・分析が求められていることが分かる。重要な制御ロジックとしてソフトウェアが含まれる複雑な組込みシステムでは、製造者や監督省庁だけによる原因調査では不十分であるという点を本書では問題意識としている。

このような問題意識に基づいて、次章以降で製品・制御システムの障害原因診断手法の調査と研究を行う。特に「事後 V&V」と呼ぶ新たな原因調査手法を提案することにより、社会が十分に納得しうる根拠をもった障害原因診断の提言を目指す。

2. 事後 V&V の全体像

2.1. 事後 V&V の提案

大規模・複雑化した製品・制御システムに発生する障害の原因を体系的に究明するには、設計段階で用いられる検証と妥当性確認（V&V：Verification and Validation）で用いられる方法論と同じ考え方を用いることが大事である。V&V は、設計段階での考え落としや実装ミスを防ぐ方法であるが、障害発生時の原因究明は、まさに、これと同じことを行う必要があるためである。さらには、抽出した原因仮説により発生した障害から観測される諸事象をすべて再現できるという証明まで必要とされるため、設計段階での V&V よりも具体的できめ細かい方法論を確立しておく必要がある。

本書では、これを「事後 V&V」と呼び、そこで使う方法論を体系化し、具体的な応用に必要なツール群とその使い方までを整理して提供することを目指している。多くのステークホルダーが協働して問題解決に当たるための共通のプラットフォームが、モデルベースアプローチによる事後 V&V という方法論であるといえる。図 2.1-1 は、事後 V&V の体系をまとめたものである。各要素技術の概要を以下に示す。

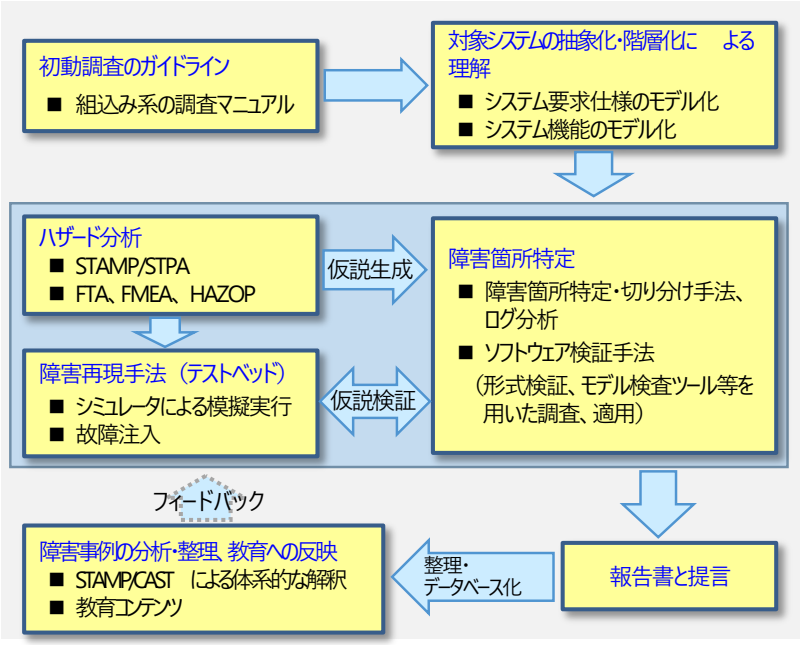

図 2.1-1　事後 V&V の全体概要

1. 初動調査のガイドライン

   障害原因分析するには、まず初動調査として分析に必要な情報の収集を行う。ソフトウェアをその核とする製品・制御システムの障害が発生した際の対応では、他分野におけるものとはかなり異なるアプローチが必要となる。製品・制御システムにおいて障害分析に必要な情報を収集するためのガイドラインを作成する。

2. 対象システムの抽象化・階層化による理解

   製品・制御システムの障害原因究明において重要なものとして、第三者による対象システムの理解がある。ここには、対象システムの要求仕様、要求を実現するためのシステム機能、機

能を実現するための構造が含まれる。要求仕様障害発生に関連する部分の第三者による理解のために、抽象化及び階層化を行い、モデルベースアプローチで用いられている方法論とツールを利用し、事例としてシステム記述言語のひとつである SysML での記述とハザード分析を試みた。

3. ハザード分析と障害原因仮説の生成

システムの要求仕様や機能がモデル化されると、そのシステムが起こしうる障害（ハザード）が予測可能になる。このハザードの要因（原因）を体系的に分析する方法として、FTA（Fault Tree Analisys）や FMEA（Failure Mode and Effect Analysis）、HAZOP（HAZard and OPerability study）などの方法論、手法がある。これらの障害解析法を状況に応じて使い分け、ハザードとその要因を障害原因仮説としてリストアップすることができる。

大規模・複雑化したシステムのハザード分析法として、近年、MIT の Nancy G. Leveson 教授が提唱するシステム理論に基づく事故モデル STAMP（Systems-Theoretic Accident Model and Processes）という考え方とそれに基づく手法 STPA(System-Theoretic Process Analysis)と CAST（Causal Analysis based on STAMP）[Leveson2012]が注目されている。これは、障害の原因をコンポーネント間やコンポーネントと人間の間のインターフェースのミスなどの幅広い視野で分析するというシステムズエンジニアリングアプローチといえる。

4. 仮説検証の要素技術

障害を引き起こすサブシステムが絞り込まれ、その障害原因仮説が抽出された場合、その仮説を検証することが必要になる。伝統的な静的解析技術に加えて、モデル検査などの最新の形式手法に関する研究成果を取り入れることで、ソフトウェアの設計に組み込まれたフォールトを顕在化させることが可能になる。

障害再現シミュレーションにより予測される様々の観測事象が現実の観測結果と一致すれば、そこでの原因仮説が実証されたことになる。この障害再現シミュレーションを体系的に行うモデルベースアプローチも仮説検証の要素技術として大事になる。

障害原因仮説が絞り込まれたあとの、もうひとつ大事な視点が、障害を引き起こした背景要因や真因の考察である。

5. 報告書と提言

1〜4 で調査した内容は、報告書としてまとめ、今後の障害を防ぐことに役立てていかなければならない。システムズアプローチといった体系的な原因究明は、将来の本質的な開発プロセスの改善に結びつける提言に役立てることができると考えられる。

6. 教育への反映

上記 1〜5 で調査・報告した内容は具体的な障害事例に基づき最新のモデルベースアプローチによって得られた結論である。そこには新しいエンジニアの教育に有用な結果が多数含まれていると考えられ、使用したツールは具体的な適用例をもった演習教材としても利用できる可能性がある。これらを教育コンテンツとしてまとめ、教育や訓練に利用することが重要である。

また、障害の組織としての再発防止や改善、教育を考えていく上では組織のプロセスを評価しておくことも重要となる。

2.2. システム障害発生のパターン

2.2.1. 障害発生の全体像

障害の発生には多くの要因が係っている。システムの構成要素に何らかの故障が発生し、ある日突然システム障害（例えばシステムダウン）に至っても、システム障害に対するバックアップ機能や応急処置、復旧施策等の備えができていれば、障害時間や被害の波及を最小限に抑えることができる。また、障害を引き起こす因子としては、例えば操作ミスや保守保全の不徹底、情報セキュリティの脆弱性、製作過程で内在させてしまったプログラムバグなど複数のものがあっても、そのどれかを未然に手当てしておけば大事には至らなかったであろう事例は多々ある。

障害の発生に直面した時にいったいどこから手を付けて原因究明したらよいのか、発生してから慌てることのないようにするためには、障害がどのような構図で起きうるものか予め全体像を掴んでおくことが重要である。

1950 年代に全米安全評議会（NSC）が開発した家庭内事故についてのモデル（[レブソン 2009]で紹介）を参考に、製品・制御システムの障害発生の全体像を作成した（図 2.2-1）。

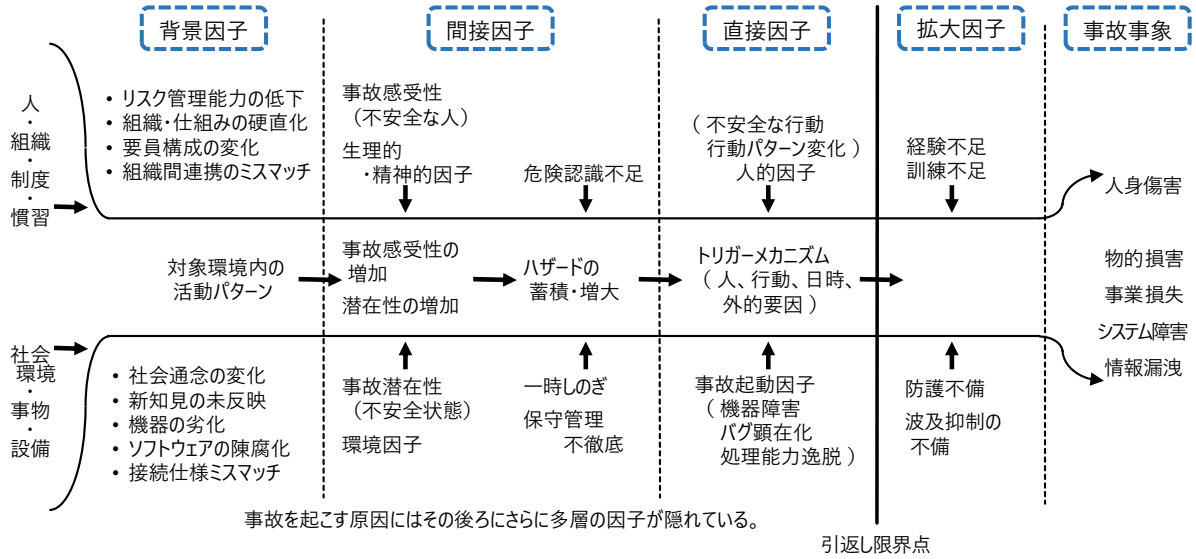

図 2.2-1　製品・制御システムの障害発生の全体像

観測される事故事象は、図 2.2-1 の右側に示された人身傷害や物損、システム障害などである。多くのシステムには利用者、運用者、保守員がいて、メーカーが控えている。これら関係者の観測情報や事実・事象を集め、障害現象の直接原因の推定から始めて時間経過を遡りつつ事実関係を整理し、FTA やなぜなぜ分析を行うことで事故に至った要因を洗い出すことが可能になる。このような複雑で大規模なシステムの障害で注意が必要な点は、障害を引き起こす要因が一つではなく、しかも、顕在化していない状態にあることである。しかも、その要因は、ソフトウェアのバグといった単純なものだけでなく、要求仕様の欠陥、使用される社会環境の変化に伴う欠陥の顕在化、システムを取り扱うステークホルダーの変化など、複雑な相互作用により引き起こされる。このような複雑な障害では、従来型の故障モデルである故障要因とそれが引き起こす事象の連鎖（FTA など）だけで説明ができないことも想定され、前記の STAMP のような考え方に基づく障害原因究明も必要になってくる。

調査の結果判明した直接因子や拡大因子は、再発防止のためにも速やかに対策を打つ必要がある。

間接因子はその組織体の風土や経営理念に起因していたり、あるいはシステムの基本設計思想に内在していたりということがある。それらには直ぐには対策できないものもあり、うやむやになりがちであるが、将来への課題として次のシステム開発に反映するような努力が必要である。

背景因子は、そのシステムや組織体を越えて社会全体の課題であり特定の技術分野における課題でもある。よりオープンな議論を重ね広く共有するべきである。

2.2.2. ソフトウェア障害原因の特性

システム障害原因の調査においては、まず、システムがどのようなサブシステムやパーツから構成されているかをしっかり把握して調査に臨むべきである。原因特定だけでなく再発防止や障害拡大防止のためにもシステムの全体構成を俯瞰しておく必要がある。

今日のシステムでは、System of systems とも呼ばれるように、サブシステムや構成機器が自律的に動作し、全体として連携・協調するようになって、複雑な機能の実現にソフトウェアの役割が飛躍的に増大している。このようなシステムでは、単一の設計者やレビューアがその挙動の全てを把握することが難しく、一貫した設計思想や安全管理思想でシステムを作ることが難しくなる。さらには、ソフトウェアという柔軟で巨大な機能を簡単に実現できるツールの存在が、一貫した設計思想でシステムを作ることをさらに難しくしている。特に、ソフトウェアに起因する障害では、各要素（サブシステム）に関わる担当設計者が、全体システムの目的や設計思想を理解せずに一部の要求仕様に基づいて設計せざるを得ないという実態が大きな背景要因となっている。

社会の変化が早くなっている状況下で、多くのシステム開発では、その前提条件や要求仕様を完全に定めて設計することは困難であり、特にユーザーの操作がかかわる場合、開発時には予想し得なかった状況に陥ることもある。動作環境が変化することによって要求仕様と合わなくなり、その対応がソフトウェアに皺寄せされることで重大障害に結びつく不具合が発生する。

このようなソフトウェアに特有の障害の原因を究明するには、単に、要求仕様通りに作られていないソフトウェアのバグを見つけるという視点だけではなく、要求仕様がどのようなユーザーニーズから出され、そこに欠陥がなかったかというような視点や、ユーザーニーズが出された際の社会環境とシステムが提供される時点での社会環境に差異がないかといった視点が必要とされる。前記の STAMP は、このような広い視点からの障害の解釈が必要であることを主張しており、ソフトウェアという新しい道具を手にした時代における、製品・制御システムの安全も含めた品質管理に関わる大事な概念を示唆してくれている。

3. 初動調査のガイドライン

本章では、重要インフラを担う製品・制御システムに障害が発生し、通報が入ってからの初動対応について、必要性、一般的な対応パターン、調査の指針と注意点の順に述べる。過去の事故調査事例の詳細は、前年度報告書[IPA2015]を参照されたい。

3.1. 製品・制御システムの障害調査の必要性

障害が発生した場合、発見者の通報が、すべての事実を客観的に捉えていると考えるのは危険である。本人が気づいていない潜在的な事実や要因は常にあるからだ。思い込みや誤解もある。

また、システムは生き物とも言われるように、時間と共に状況は変転していく。障害を取り除こうとしたり、影響を最小限にしようとしたりする諸々の努力が、隠れた原因の証拠を消失させたり、覆い隠してしまうこともありうる。

このような状態で正確な調査を行うためには、周到な準備と細心の注意、そしてスキルが必要になる。障害復旧後に根源的な原因究明や的確な再発防止策を講じるためには、客観的で分析可能な調査記録を作成しておくことはきわめて重要である。

このことは、後述の航空機や鉄道事故のような過酷な事故調査ではしばしば指摘されており、失敗学会等でも調査報告の重要性が訴えられている。

ソフトウェアに起因する障害に特有の難しさもある。物理的な事物は、あとで手を加えても何らかの痕跡が残るものである。しかし、ソフトウェアの実行履歴は、跡形もなく消えてしまうことがある。さらにソフトウェアは、処理量も膨大であり、しかも設計思想の全く異なる複数のプログラムが、お互いに協調することもなく、同じプラットフォーム上に共存している。このような実行環境下でトラブルが起きると開発者本人でさえ、原因究明に手を焼くというのがソフトウェアの世界である。

このような悩ましい課題を抱えながらも、ソフトウェアがエンジン制御やブレーキ制御、あるいは鉄道網の管理や航空機の運航など、重要インフラに深く組み込まれてきた現在、障害原因究明の社会的要求は強くなる一方である。それ故、重要インフラに組込まれたソフトウェアの調査はできるだけ体系化され、仮に基本原因が不明でも、障害発生に至る幾つかの要因をできるだけあぶり出し、後日の再調査による解明が可能なように客観的なものにする必要がある。

3.2. 障害発生時の一般的な対応

3.2.1. 障害発生時の一般的な対応パターン

障害発生時の一般的な対応モデル例を時間的な推移で示す（図 3.2-1 参照)。

まず、障害発生が認知され通知される（第１ステップ)。次に障害原因の一次判定（第２ステップの１)、障害の除去または隔離と復旧（第２ステップの２)、障害の記録と報告（第５ステップ）となる。この場合は、システム運用者や待機中の保守員の対応で済む。

しかし、障害の発生が収束せずに被害拡大した場合（第３ステップの１)、復旧までに時間がかかり、被害抑制策（第３ステップの２）を講じるなど、応援要請・体制強化が必要になり対策本部の設置も必要になる（第４ステップ）だろう。対策本部では、障害事象の特定、被害拡大阻止、報告・広報などの一連の作業が同時進行的に進められる。規模の大きなシステムでは、対策に従事する人員は、数十人から千人に迫ることもありうる。

システム規模が大きければ、防災対策と同様、平時から緊急時の連絡体制、対応マニュアル、訓練

が必須になる。

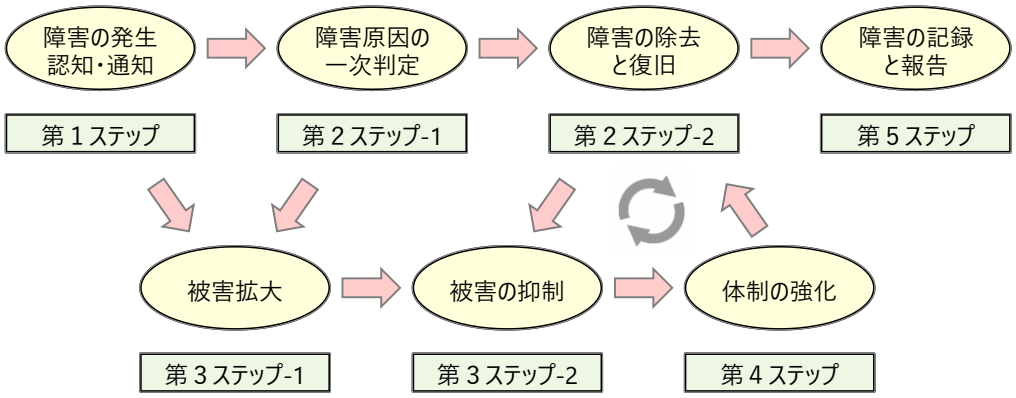

図 3.2-1　障害発生時の一般的な対応パターン

3.2.2. 障害調査の事例

社会的に影響の大きい事故については、以前から、関係機関毎に調査手順を定めて対応してきている。例えば、以下のような事例がある。詳しくは前年度の報告書に記載している。

ア）運輸安全委員会の例
イ）火災事故の報告書例
ウ）昇降機の事故報告書例
エ）医療事故調査の例
オ）金融システムの報告書例
カ）情報セキュリティ障害報告例
キ）IPA のソフトウェア障害調査報告例

3.3. 製品・制御システムの障害調査指針

3.3.1. 障害発生時の初動調査の指針と注意点

障害発生の通報を受けて調査を開始する場合、個人情報や組織機密に接する可能性があり、また、刑事事件や民事事件に至る可能性があることを認識しておく必要がある。

調査者は、調査の目的と自分の権限範囲を知っておくべきである。初動調査の目的は、一般的には①障害の直接原因と損害を明らかにすること、 ②障害の再発防止策、拡大防止策に必要な基礎事実、基礎資料を得ることである。

従って、原因究明が犯人探しになってはならないし、根拠なく間接要因や背景要因を断言するようなことがあってはならない。客観的な事実確認に徹し、科学的かつ合理的な観点で事実関係を立証していくべきである。

調査者の心得として以下を提示する。

1）システムの調査に必要な知識やシステム運用に係る関係法令や規則を習得しておくこと。専門が多岐にわたる場合は、専門家の招聘、調査委託も考えておくこと。
2）関係者の民事的紛争に関与しないようにすること。個人の権利侵害や調査過程で知りえた秘密を他に漏らさないようにすること。
3）調査場所、調査設備に立ち入る場合は、原則、関係者の立ち合いを得ること。

4）関係者と随時密接に連絡を取り相互協力して調査すること。
5）調査の結果は随時公表し、調査終了後は調査経過も公表するよう努める。

3.3.2. 初動調査のステップと真因究明

通報を受けてからのアクションは、概ね次のようなステップになる。

1）通報受理
- 通報受理時刻、通報者氏名、通報手段、障害発見時刻、場所、システム名などの基本情報の収集。
- システム基礎情報（事前管理資料）との照合。関係先への連絡。広報担当や警察機関の活動に協力する。

2）影響拡大阻止への協力
- 通報者からの応援依頼などに対処する。

3）前調査
- システム基礎資料から障害発生個所、影響の進展範囲、などの推定。
- 現場調査に必要な資料や調査優先事項の整理。
- 現場調査者を決める（むやみに現場へ入らせない)。

4）現場調査
- 第一通報者の確認、状況ヒアリング。
- 顧客の被害や社外への影響把握、関係者の安全の確認。
- エラーログなど後日調査に必要な記録の保管、保存。
- 機器の状態確認、障害応急対応の作業内容の確認。

5）関係資料入手、または閲覧
- 設計資料、実装ソフトウェアのバージョン、更新履歴、システム運用履歴、システム保守履歴、過去のトラブル記録等の確認、収集。
- ソフトウェアの実行環境の確認。ハードウェア保守記録の収集。

6）関係者ヒアリング
- システム運用者、保守員、改造・改良担当者、システム利用者（外部ユーザー）から個別、独立にヒアリング。
- ヒアリング内容の整合性確認。

7）原因調査、判定
- 複数人で客観的に事実関係に基づき原因を絞り込む。

8）報告
- 再発防止策、障害拡大抑制策に利用できるよう、正確にわかり易く記載。
- 事実内容と推定内容は明確に分ける。
- 後日の再調査や統計分析に利用できるよう分類可能な書式とする。かつ、秘密情報が保護されるよう区分しておく。
- 広報担当、顧客などの被害者に対して説明可能なわかり易い文章に努める。

3.3.3. 障害発生に備える

障害はある日突然発生する。発生してから必要な基礎資料を集めるような泥縄では初動調査の重要

なタイミングで作業遅滞を招いてしまう。そのため、3.3 節で示した構成要素に係る基礎データは予めファイルしておくか、すぐにアクセスできるようにしておくことが望ましい。また、前節に示した調査ステップがスムーズに進むように必要な記録帳票の様式を事前に用意しておくべきである。

航空機にはフライトレコーダーやボイスレコーダーがあり、事故調査の体制も法的裏付けがある。消防も消火器や避難経路など様々な予防・被害抑制設備を義務付けている。

重要インフラへの利用拡大が進んでいる製品・制御システムも機種に依存されない標準的な運用・通信記録や障害抑制機能、保護機能の義務付けや業界指針が必要と思われる。

4. 対象システムの抽象化・階層化による理解

製品・制御システムの障害原因究明は、開発者でもなく利用者でもない第三者によって実施される。したがって、第三者による対象システムの理解が、特に重要になる。対象システムの提供時の要求（要求仕様）と、これを実現するためのシステムの機能と構造が、理解の対象となる。ここには、システムを開発又は利用した企業のノウハウが多く含まれるが、障害発生に関連する部分の第三者による理解のために、抽象化及び階層化を行い、モデルベースアプローチで用いられている方法論とツールを利用する。

4.1 節では、システム障害を診断するための手法と、対象システムの要求や機能を記述するために使える手法を説明しているが、この部分は 2014 年度の報告書と同じであり、要約だけを述べるにとどめる。

本年度の活動を反映して全面的に書き直したのは、4.2 節と 4.3 節である。4.2 節では、システム要求仕様のモデル化の事例として、2014 年度に開発した化学プラントシミュレーターを題材として、システム記述言語のひとつである SysML でシステム要求を記述し、それを参照して、潜在的な障害原因を識別するためにハザード分析を試みた。

4.3 節では、システム開発とハザード分析の統合を図ることを目的に、STAMP 構成要素であるコントロールストラクチャーなどを SysML で記述することを試みた。

4.1. システム理解のためのモデル化

事後 V&V は、システム障害が起きてから、ソフトウェアに原因があれば、その原因を究明するために適用される手法のフレームワークである。この原因究明は、一般的には、障害箇所を特定できているか、障害現象が再現できるか、この 2 点に依存するであろう。したがって、原因究明に用いる診断手法は、障害の発生形態に適するものを選ばなければならない。

障害の発生形態に基づいて診断手法を分類すると、図 4.1-1 のように表現することができる。たとえば、障害箇所を特定でき、かつ、障害現象が再現する場合には、次の手順による診断が有望である:

1）障害発生によって侵害されるデータ構造に関する制約を記述する。
2）障害箇所とその環境との相互作用をモデルの階層表現によって記述する。
3）障害を再現させるテストプログラムを作成し、実行する。
4）障害が再現したときに、データ構造の制約がどのように侵害されたかを分析する。

障害箇所を特定できず、さらに、障害が再現しないとなると、原因究明は困難を極め、事後 V&V フレームワークにおけるハザード分析などが、特に必要となる。

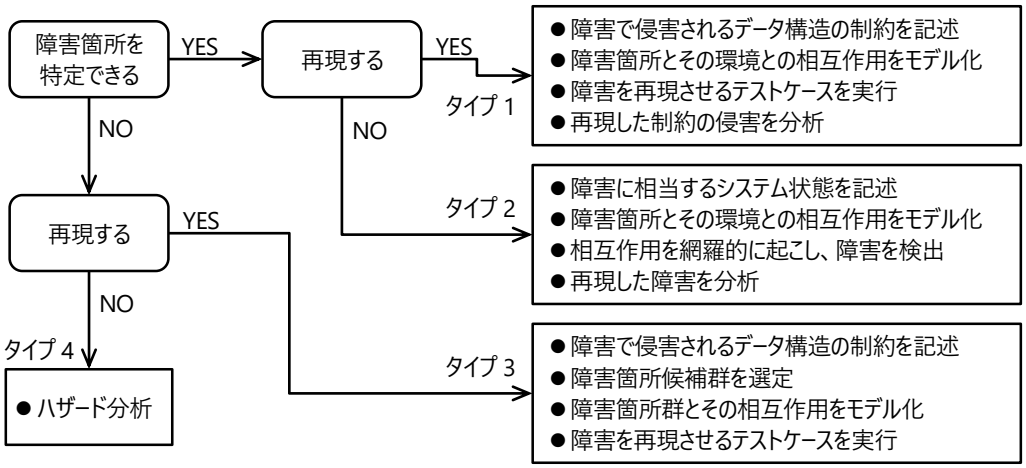

図 4.1-1　障害の発生形態に基づく診断手法の分類

システム障害を究明するための診断手法は、障害の発生形態に依存するだけではなく、対象となるシステムの要求及び機能の理解を必要とする。システム要求には機能要求だけではなく、非機能要求もあり、システム機能としてはシステムの動作及び構造が関わる。しかし、実際のシステム障害に際しては、これらに関する情報が、十分に文書化されていることは少ないのが現状である。したがって、障害発生時に存在する、対象システムに関する要求又は機能に関する情報をもとにして、安全や損失などに関する制約を可視化した仕様を記述することが課題となる。

そのための記述手法は、構文化手法[3]と、半形式手法、形式手法の 3 種類に分けて考えることができ、代表的な手法を表 4.1-1 に示す。それぞれの記述手法には特徴があり、これらを適切に選択してモデル化、仕様化することが求められる。たとえば、UML[4]のような半形式手法だけを使って、不完全な情報から安全制約を可視化した仕様を記述するだけで十分なケースもある。あるいは、仕様の検証が必要であれば、その記述結果をもとに形式手法で書き直さなければならないケースもある。

重要な点は、状況や必要などに応じて、これらの手法を組み合わせ、相互に変換し、安全や損失などに関する制約をモデル化し、仕様化することである。

[3] 定められた構文規則に従い、自然言語を用いて仕様を記述する手法を意味する。

[4] Unified Modeling Language の略で、ソフトウェア構造と動作の記述に用いられる。

表 4.1-1 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。	制約を時相論理で記述し、その反例をツールが見つける。
	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。	制約を実時間時相論理で記述し、その反例をツールが見つける。
形式仕様記述	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。	不変条件を記述し、その妥当性を演繹的に証明する。
	Event-B	イベントや制約条件を記述し、段階的に詳細化してゆく。	不変条件を記述し、その妥当性を演繹的に証明する。
構文化手法	SLP	要求を主語又は目的語と述語からなる文と、その構文規則で記述する。	述語論理を用いて、論理整合性と用語の統一を図る。
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。	オブジェクトに関する制約を記述できる。
MBD	Simulink	構造と動作をブロック線図や状態遷移図で表現。	制約を状態遷移や時相論理で記述し、その侵害を検出。
STAMP/STPA	研究レベル	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。

4.2. システム要求仕様のモデル化

対象システムに関する要求（要求仕様）を理解しようとすると、それを適切に記述する方法論又はツールが必要となる。今年度の活動として、化学プラントシミュレーターに関する要求を SysML で記述し、その記述内容を参照して、潜在的な障害原因を洗い出すためにハザード分析を行った。システム記述のための方法論として、SysML を表記法として利用するひとつの分析設計手法を、ハザード分析としては、最近、注目を集めている STAMP/STPA 手法をそれぞれ使用した。その試行の概要を述べる。

4.2.1. システム要求の検証

システムを記述するために用いられる SysML は、システムの要求と、機能、構造を表現することはできるが、単なる表記法のひとつであり、システム要求をモデリングするための手法を提供しない。そのため、システム要求の全体像を記述するには、何らかの方法論が必要となる。今回の試行では、システム開発プロセスにおいて、システム構成要素の開発に先立って、与えられた要求をもとにして、構成要素の要求仕様を作成する活動を支援するための分析設計技法[5]を使用した。その作業の流れ及び内容の概要は、表 4.2-1 に示す。この作業手順に沿って、システムの全体像を SysML を用いて記述した。

[5] システム要求設計技法と呼び、表記法として SysML を使用する。詳細は次を参照：
http://homepage2.nifty.com/rent-a-coach/library.html

表 4.2-1　システム記述に用いた作業の流れと内容

手順	作業項目	使用する SysML 図
要求分析	要求を獲得する	要求図
	システムとその境界を決める	ブロック定義図
	システムの使われ方（機能）を定める	ユースケース図
	ユースケースの動作を表現する	シーケンス図 アクティビティ図 状態機械図
アーキテクチャー設計	システムを構成要素に分解する	ブロック定義図
	部品の相互作用を定義する	シーケンス図 アクティビティ図
	部品の相互接続を定義する	内部ブロック図
制約評価	システムの安全制約を獲得する	構造に関する図 動作に関する図
	ハザード分析し、設計を修正する（繰返す）	
要求割当て	構成要素の要求仕様を定める	ブロック定義図
	要求の追跡性を確立する	要求図

(1) 要求を獲得する

題材とする化学プラントシミュレーターに関しては、その概要を説明する要求文書[6]が存在する。この種の文書は、一般的には分析に適するようには書かれていないので、分析を行いやすいように、まず、その記載内容をもとにして要求の整理を行った。経験に従えば、書かれている要求事項を構成と、制御、操作、安全制約に分類するのが妥当であろう。たとえば、制御要求と安全制約を取り上げると、次のように記述した：

制御要求 1：タンク 1 の水位を目標値に制御する

制御要求 2：タンク 2 は自動給水する

制御要求 3：タンク 1 の水位がアラートレベルを超えた場合、緩和排水する

制御要求 4：タンク 1 の水位がアラームレベルを超えた場合、緊急排水する

安全制約 1：タンク 1 の溢水は防止しなければならない

分類した後で、個々の要求を必要に応じて原子的要求に分解し、結果は省略するが、要求間の関係を要求図を用いて明確に記述した。

構成に関しては、要求文書に詳細な説明があり、その一部を図 4.2-1 に示す。

[6] 昨年度の報告書[IPA2015]の付録を参照。

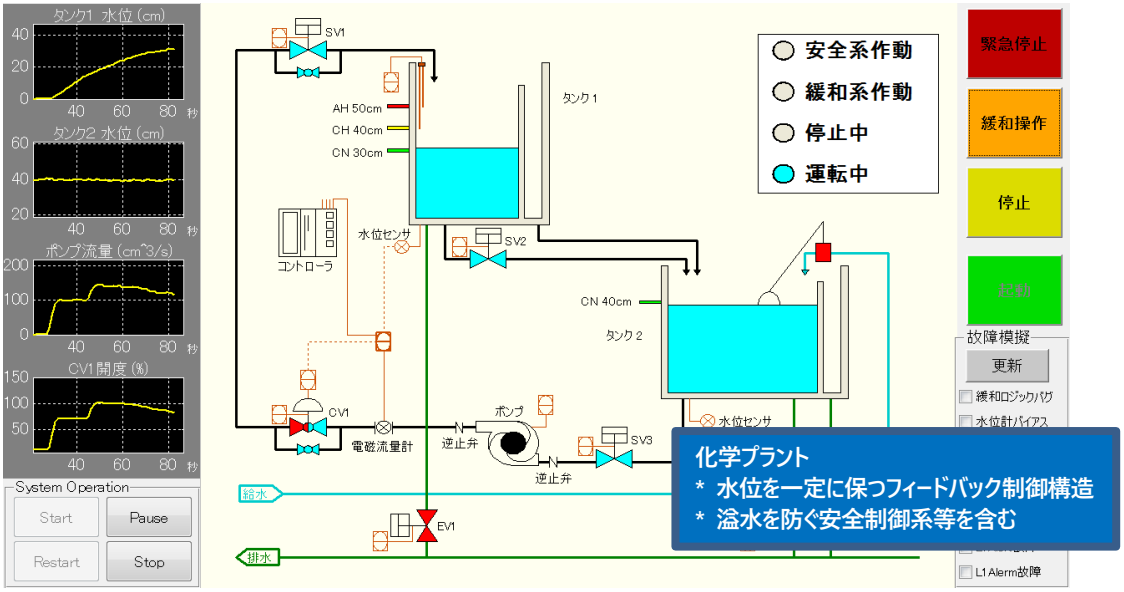

図 4.2-1　化学プラントシミレーションの系統図

(2) システムとその境界を決める

システムを開発するためには、そのシステムの内側と外側を明確に決める必要がある。通常、システムに含まれない周辺の設備や環境などをシステムコンテキスト図に表現することが多い。SysML を使うと、このシステムコンテキスト図をブロック定義図で表現することができる。

障害診断においてもシステムの境界は重要なので、システムコンテキスト図を作成し、この図で、対象システムを「化学プラント」と命名し、電源や給排水などのための設備を対象システム外とし、システムの利用者として運転員を明記した。

(3) システムの使われ方を定める

次の作業は、システムが提供する機能を決めることである。通常、システムの利用者から見た機能を洗い出し、ユースケース図を作成することが多い。システムコンテキスト図で運転員を利用者としたから、化学プラントが運転員に提供する機能を調べ、次のユースケースを定義した：

- 化学プラントを起動する
- 緊急排水する
- 緩和排水する
- 化学プラントを停止する

(4) ユースケースの動作を表現する

次に、各ユースケースの動作を記述する。運転員からの指令などの事象に反応するシステムでは、状態機械図（状態遷移図）を用いることが一般的である。状態機械図を使うと、ユースケースに対応する動作だけでなく、運転員からの指令によらない自動制御の動作も合わせて記述できる。その記述結果を図 4.2-2 に示す。

この図において、太線の流れは、「化学プラントを起動する」というユースケースに対応し、事象の中で、青字表記は自動制御に対応する。各状態においては、その状態で実行する機能を対応づけた。特に、緊急排水制御状態へは運転状態のどのサブ状態からでも、水位の変化又は運転員からの指令という事象で遷移することが、赤字表記で明記されている。

各状態で実行される水位制御機能などの処理概要は、与えられた要求文書から読取って、結果は省略するが、アクティビティ図を用いてそれぞれ記述した。

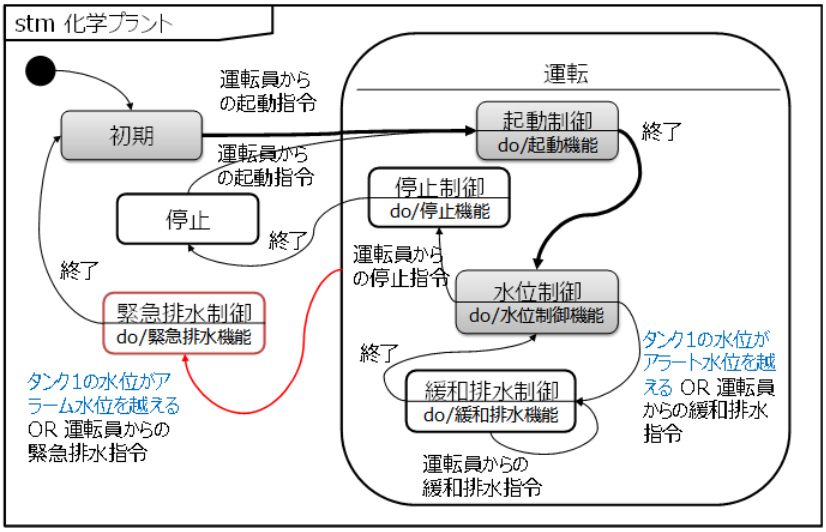

図 4.2-2　全体の動きを表す状態機械図

(5) システムを構成要素に分解する

ここまでが表 4.2-1 における要求分析であり、次の段階のアーキテクチャー設計として、まず、システムの構成を分析した。与えられた要求文書は、構成を詳細に述べているが、アーキテクチャーレベルの視点から主要な構成要素を洗い出し、ここでは、次のように定義にした：

- コントローラー
- 液体循環装置
- 操作卓

ここで、制御されるプラント側を液体循環装置としてまとめた。これは、タンクや、ポンプ、弁、配管、水位センサーなどを含むが、その詳細は省略する。

(6) 部品の相互作用・接続を定義する

主要な構成要素の内部構造には触れずに、それらの相互作用・接続を分析するのが次の作業になる。その結果の一部を図 4.2-3 に示す。

この図において、コントローラーから液体循環装置へは、弁制御指令とポンプ制御指令という 2 種類のインターフェースがあること、逆方向へは、測定水位というインターフェースがあることが示されている。操作卓とコントローラーの間には、運転指令と運転状況という 2 つのインターフェースが存在している。

それぞれのインターフェースの中味は、ブロック定義図を用いて明確にすることができる。たとえば、図は省略するが、弁制御指令は、排水弁制御と、止め弁制御、給水弁制御、制御弁制御から構成され、各弁制御は、「開く」か「閉める」かの値をとることを定義した。運転指令については、これは運転制御で構成され、運転制御は「起動指令」、「停止指令」、「緊急排水指令」、か「緩和排水指令」の値をとることを定義した。

最後に、状態機械図に示された、各状態で実行する機能が、コントローラーによってすべて実行されることを、ブロック定義図を用いて明確にすれば、与えられた要求文書をもとに表 4.2-1 の手順で

分析したシステム記述は完成する。

このように、システム開発で用いられる分析設計手法を活用すれば、障害原因診断のときにも、対象システムの要求仕様をモデル化して記述することができる。次節では、このようにして作成されたシステム記述を参照して行われるハザード分析について述べる。

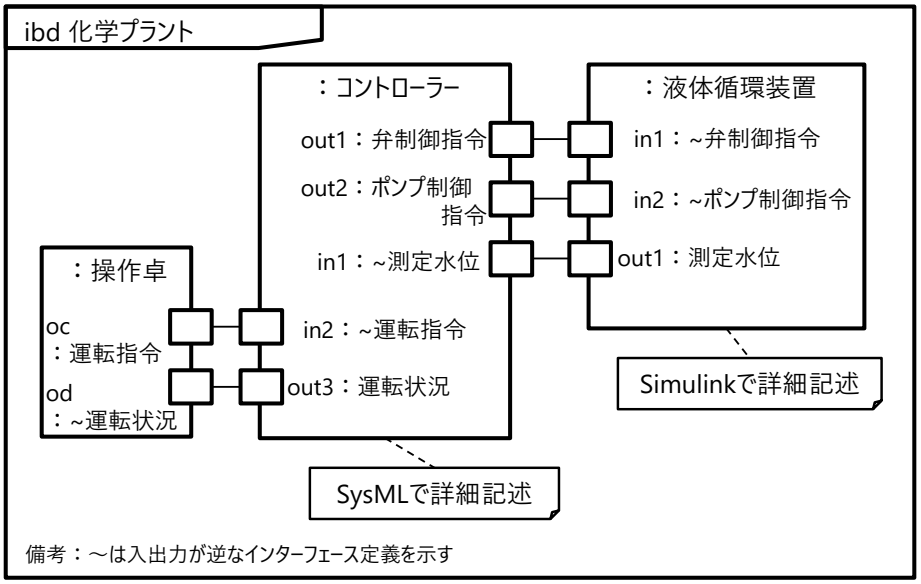

図 4.2-3　主要な構成要素間の相互接続

4.2.2. STAMP/STPA によるハザード分析の試行

事後 V&V フレームワークにおいては、障害原因に対する仮説生成を行う手法のひとつに、STAMP/STPA を位置づけている。この手法は、ハザード分析手法のひとつであり、基本的には、システム開発の初期の段階、まだ設計が進んでいない状態で、ハザードを引き起こす要因を識別することを可能にする。この手法が、既に開発を終えて、稼働しているシステムに潜在する障害原因を識別することができるかを考察するために、化学プラントシミュレーターに適用してみた。その際に、4.2.1 節で作成した SysML で書かれたシステム記述を参照して STPA 分析を進めた。その概要を手順に沿って説明する。

(1) コントロールストラクチャー図を作成する

一般的な STPA 分析の手順に従うと[7]、最初に実施することは、アクシデントと、ハザード、安全制約の識別である。これらは、与えられた要求文書を解釈して、次のように識別した：

アクシデント：化学汚染

ハザード：タンク 1 の溢水

安全制約：タンク 1 の溢水を防止する

次に、安全制約を守らせるためのコントロールストラクチャーを分析しなければならない。このときに、図 4.2-3 の内部ブロック図やインターフェース定義などを参照した。しかし、最初の試みでは、運転員とコントローラーが与えるコントロールアクションの詳細度が揃わないことに気づいた。そのため、ユースケース図と状態機械図を参照して、両者のコントロールアクションを機能レベルに揃え

[7] 一般的な STPA 手順は、IPA/SEC 発行の「STAMP 手法に関する調査報告書」(2015 年 5 月)を参照。

ることを行った。その結果が、図 4.2-4 である。この図に示すように、コントローラーが与えるコントロールアクションは、最初は、弁及びポンプの制御指令としていたが、運転員からの指令に対応する機能に変更した。

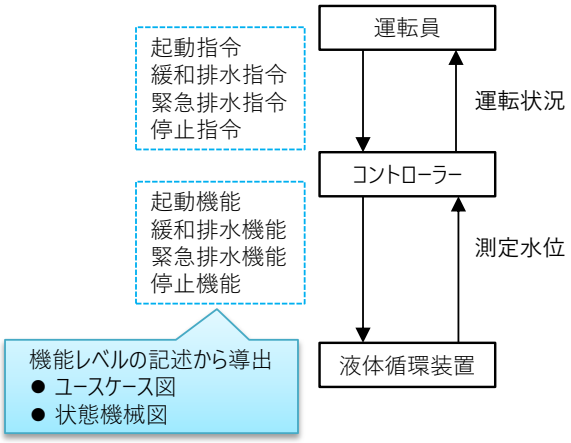

図 4.2-4　化学プラントシミレーターのコントロールストラクチャー図

(2) 非安全なコントロールアクションを識別する

ここまでが準備段階で、次は、ハザードを引き起こすかもしれないコントロールアクションを識別する段階である。このために、一般的には四つのガイドワードが使われる。運転員とコントローラーのコントロールアクションを縦軸に、ガイドワードを横軸に置く表が使われ、すべてのマス目で、ハザードが引き起こされる条件や状態などを考える。

その結果の一部を表 4.2-2 に示す。この表において、赤枠表記の部分に注目すると、コントローラーが緊急排水機能を途中で止める要因が、運転員が与える起動指令、停止指令、又は緩和排水指令に関係していることがわかる。

さらに分析の詳細度を高めるために、図 4.2-2 の状態機械図と各種のアクティビティ図を参照して、コントローラーの状態ごとに検討を続けた。その結果の一部を表 4.2-2 に示す。この表では、縦軸にはコントローラーの状態を置き、横軸には運転員のコントロールアクションを置き、コントロールアクションが与えられないとき、又は与えられるときに、ハザードを引き起こすかどうかが検討されている。たとえば、コントローラーが緊急排水制御状態にあるときに、運転員が停止指令を与えると、コントローラーは緊急排水機能を途中で止め、停止機能を実行してしまい、ハザードに至る可能性を識別できる。もうひとつは、コントローラーが運転状態にあるときに、水位がアラームレベルを超えているのに、運転員が緊急排水指令を与えないと、ハザードに至るケースである。

表 4.2-2　非安全な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、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
コントローラー： 緊急排水機能	アラーム水位で緊急排水機能を行なわないとハザードになる	不必要に排水されるだけ	開始が遅れるとハザードになる	途中で止めるとハザードになる
運転員： 起動指令	N/A	緊急排水機能の途中に起動機能を行うと、ハザードになる	N/A	N/A
停止指令	N/A	緊急排水機能の途中に停止機能を行うと、ハザードになる	N/A	N/A
緩和排水指令	N/A	緊急排水機能の途中に緩和排水機能を行うと、ハザードになる	N/A	N/A
緊急排水指令	アラーム水位で緊急排水機能が行なわれていないときに指令しないとハザードになる	N/A	同様に指令が遅れるとハザードになる	N/A

表 4.2-3　運転員に関する非安全なコントロールアクション

停止指令を例にして

CA 状態	停止指令 与えられないとハザード	緊急排水指令 与えられるとハザード
停止	N/A	N/A
運転	停止機能が正常に行なわれる	アラーム水位を越えているのに、指令しないとハザードになる(UCA2)
緊急排水制御	緊急排水機能の途中に停止機能を行なうと、ハザードになる(UCA1)	N/A

動作に関する記述を参考にする
- 状態機械図
- アクティビティ図

(3) ハザード誘発要因を識別する

次の段階では、識別できた非安全な制御行動に対して、それを引き起こす要因を分析する。このためには、制御行動ごとにフィードバックを含む制御ループを描き、対象分野に適したガイドワードなどを使って、ハザード誘発要因を識別することが行われる。今回の試行では、ヒューマンエラーに関する分類を参考にして、ガイドワードを次のように設定した：

① 正しくない指令を与える（コミッションエラー）
② 正しい指令を与えない（オミッションエラー）
③ 正しい指令を間違えて実行する（スリップ）

④ 組織的な圧力を受ける
⑤ 表示を見落とす
⑥ 表示を間違えて解釈する
⑦ コントローラーが正しくない表示を行う

このガイドワードにしたがって誘発要因を識別した結果を、表 4.2-4 に示す。この表では、非安全なコントロールアクションごとに、ハザードを引き起こすシナリオを洗い出している。たとえば、シナリオ 1-1 によれば、次のような進行でハザードを引き起こす：

1. コントローラーは緊急排水制御状態にいて、緊急排水機能を実行し、その処理の一環として排水弁を開いて、タンク 1 からは緊急排水を始める。
2. 運転員は、タンク 1 の水位がアラームレベルを超えていることに驚き、間違えて停止指令を与える（ガイドワード①)。
3. コントローラーはその停止指令を受けて、緊急排水機能を中断し、停止制御状態へ遷移し、停止機能を実行する。
4. コントローラーは停止機能の一環として排水弁を閉じる。
5. 排水弁が閉じられた結果、タンク 1 の溢水が起きる。

その他のシナリオも同様であり、表中の丸数字は上述のガイドワードに対応している。

表 4.2-4　ハザード誘発要因の識別

UCA	シナリオ	
UCA1： 緊急排水機能の途中に停止機能を行うと、ハザードになる	シナリオ 1-1	1. コントローラーは緊急排水機能を開始し、排水弁を開く。 2. 運転員はアラーム水位に驚き、間違って停止指令する（①）。 3. それを受けて、コントローラーは緊急排水機能を中断して、停止機能を実行する。 4. コントローラーは停止機能の一環として、排水弁を閉じる。
	シナリオ 1-2	1. コントローラーは緊急排水機能を開始し、排水弁を開く。 2. 運転員は緊急排水が終了したと勘違いし（⑥）、早く復旧したいという組織の要求に対応して（④）、停止指令する。 3. コントローラーは停止機能の一環として、排水弁を閉じる。
UCA2： アラーム水位を越えているのに、指令しないとハザードになる	シナリオ 2-1	1. アラーム水位を越えたのに、コントローラーが緊急排水機能を行わない。 2. コントローラーはアラーム水位を越えていることを表示しない（⑦）。 3. 運転員はアラーム水位を越えていることを認識しない。
	シナリオ 2-2	1. アラーム水位を越えたのに、コントローラーが緊急排水機能を行わない。 2. コントローラーはアラーム水位を越えていることを表示。 3. 運転員はアラーム水位を越えていることを見落として（⑤）、又は無視して（②）、何も実施しない。
	シナリオ 2-3	1. アラーム水位を越えたのに、コントローラーが緊急排水機能を行わない。 2. コントローラーはアラーム水位を越えていることを表示。 3. 運転員は緊急排水指令しようとして、間違って停止指令する（③）。

(4) 安全制約を追加する

STPA 分析の最後の段階は、識別できたハザードシナリオを防止するために安全制約を検討することである。これには特にガイドワードなどは用意されていないが、図 4.2-2 で記述されたコントローラーの動作に限定すれば、技術的な知見から安全制約を検討することは容易であろう。たとえば、次の 2 件を追加することができる：

- 緊急排水機能を実行中には、運転員指令を受け付けない。
- 運転状態では緊急排水指令を即時受け付け、遅滞なく実施する。

このようにして導き出された安全制約は、システム開発中に STPA 分析を実施するときには、設計に反映することになる。障害原因診断に適用する場合は、潜在的な障害原因として活用できそうである。つまり、診断対象とするシステムはこの安全制約を守っているか、守っていないとすれば、その侵害によって障害現象が発生するか、こういう仮説生成に活用できると考える。

4.3. SysML と STAMP によるシステム統合モデル化

STAMP に基づく STPA 等の分析手法は、従来の分析手法では分析が困難であった複雑に相互作用するシステムの安全分析に有効な手法であると期待されている。STAMP に基づく分析を実施するためには、準備として STAMP の構成要素である安全制約、コントロールストラクチャー、プロセスモデルを用意する必要がある。これらの構成要素は、要求仕様書等の様々な資料やステークホルダーへのインタビューから作成されるが、通常の開発工程の成果物と連携させることは有用であろう。実際、4.2 節「システム要求仕様のモデル化」では、システムズエンジニアリング用モデリング言語 SysML の図を参照して STAMP/STPA を実施している。

本節では、通常の開発工程で構築されるモデルと STAMP に基づく分析工程で利用されるモデルを相互に活用することで通常の開発工程と STAMP に基づく分析工程の統合を図ることを目的とし、SysML の要求図を用いた安全制約の記述事例、及び SysML のブロック定義図と内部ブロック図を用いたコントロールストラクチャーの記述過程と記述事例を紹介する（図 4.3-1）。なお、プロセスモデル記述に SysML を用いていないため、プロセスモデルの記述は割愛する。他方、工程統合の副産物として高機能な SysML 記述ツールが STAMP の構成要素の記述に利用可能となるため、人手による作業と比較して作業効率が向上した。

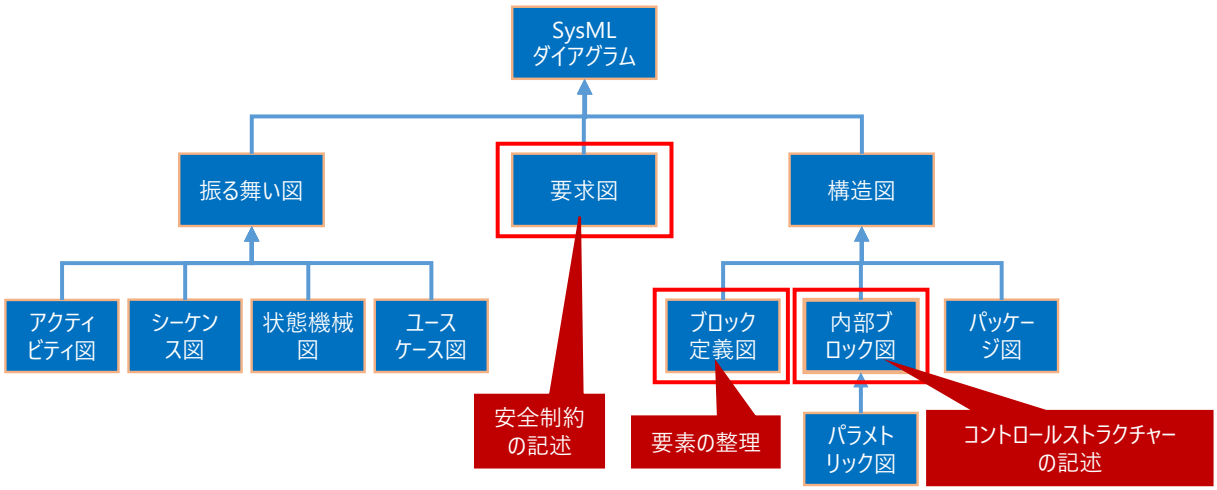

図 4.3-1 SysML ダイアグラムの分類と本事例での利用方法

4.3.1. 準備 1：要求図を用いた安全制約の識別

準備 1 では、アクシデント、ハザード、安全制約の識別が行われる。今回の事例では、SysML の要求図を用いて、これらの識別を実施した。

要求図は、「要求のトレーサビリティをサポートするために、テキストベースの要求と、他の要求、設計要素、テストケースとの関係を表す」図と定義されている[フリーデンタール 2012]。要求図では、ある要求 R1 から別の要求 R2 が導出されるという関係を、R1 から R2 への導出関係として定義できる。本事例では、アクシデント、ハザード、安全制約を要求として記述し、あるアクシデント A1（ハザード H2）からあるハザード H1（安全制約 SC1）が識別されたという関係を H1（SC2）から A1（H2）への導出関係として記述する。さらに、安全制約を表す要求とシステムに対応するブロック間を充足関係で結ぶことで、システムとシステムが満たすべき安全制約間の関係も記述できる。また、SysML のモデル要素「根拠」を用いることで、その導出の理由を「根拠」として付記できる（図 4.3-2）。

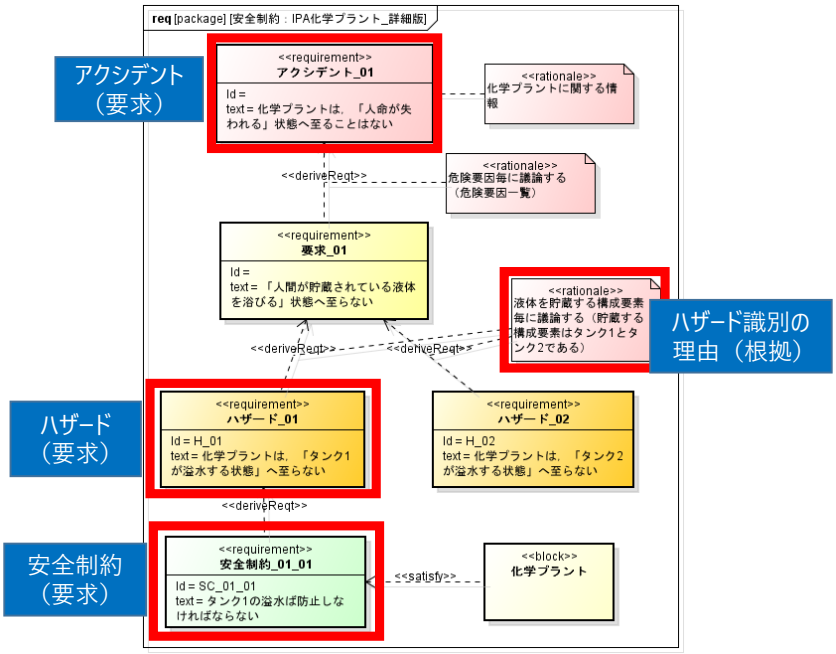

図 4.3-2　要求図を用いたアクシデント、ハザード、安全制約の識別

4.3.2. 準備 2：内部ブロック図を用いたコントロールストラクチャーの記述

準備 2 では、コントロールストラクチャーの構築が実施される。内部ブロック図は「ブロックのパート間の相互接続とインターフェースを表す」図と定義されている[フリーデンタール 2012]。他方、コントロールストラクチャーはコンポーネント間の機能動作を示したシステムの設計図であり、コンポーネント間でやり取りされる制御の指示やフィードバック等を矢印で結んで表す。そこで今回の事例では、SysML の内部ブロック図を用いてコントロールストラクチャーを記述した。

今回の事例での記述手順の概略（図 4.3-3）

1. システム構成要素の階層的整理：ブロック定義図 BDD を記述する。
2. 抽象 CS の構築：BDD 内のコントロールストラクチャー記述対象ブロックの直下ブロックのみで内部ブロック図を作成し、それを抽象コントロールストラクチャー（抽象 CS）とする。
3. 詳細 CS の構築：抽象コントロールストラクチャー内のブロックに対しそれぞれ内部ブロック図を記述し、それらを抽象コントロールストラクチャーに埋め込んで得られた内部ブロック図を詳細コントロールストラクチャー（詳細 CS）とする。
4. 最終 CS の構築：分析の観点を設定し、その観点に基づき詳細 CS のモデル要素を整理する。また、必要に応じてブロック定義図や各内部ブロック図の修正を繰り返す。

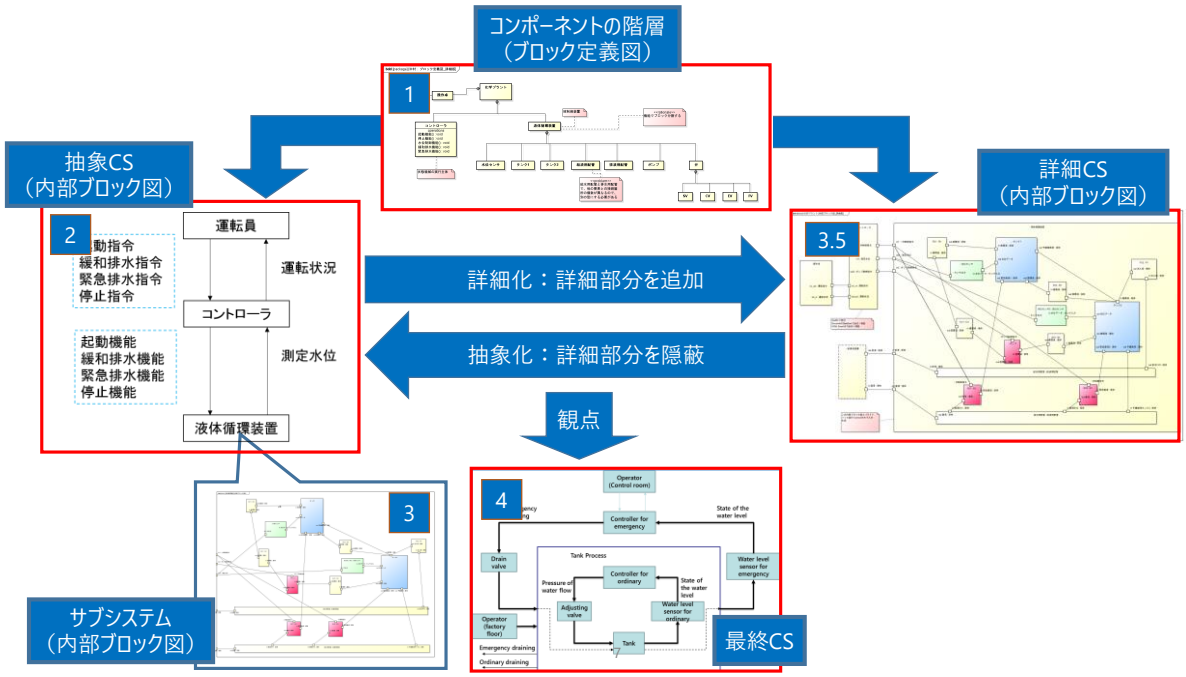

図 4.3-3　本事例でのコントロールストラクチャー構築手順

コントロールストラクチャーを記述する前に、コントロールストラクチャーに含まれる要素を階層的に整理することは、この後のコントロールストラクチャー記述の見通しをよくする。本事例では SysML のブロック定義図を用いて STAMP/STPA の分析対象である化学プラントの構成要素を整理した。（図 4.3-4）

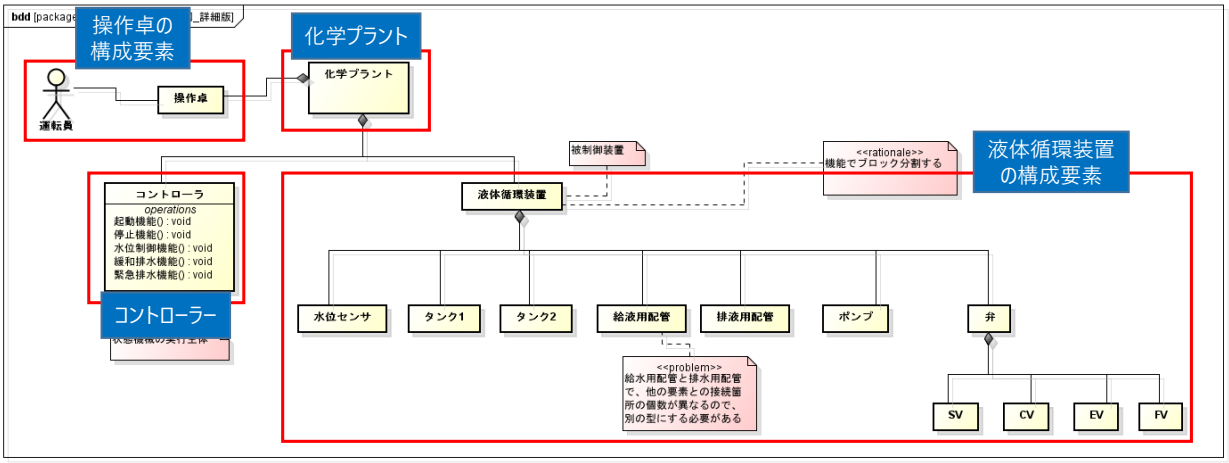

図 4.3-4　ブロック定義図による構成要素の整理

記述したブロック定義図の中で、今回の分析対象である化学プラントの直下の要素（操作卓、コントローラー、液体循環装置）と関係する要素（給排水設備）から化学プラントの抽象コントロールストラクチャーを構築した。すなわち、これらの要素を配置して、要素間のコントロールアクションを追記した。（図 4.3-5）

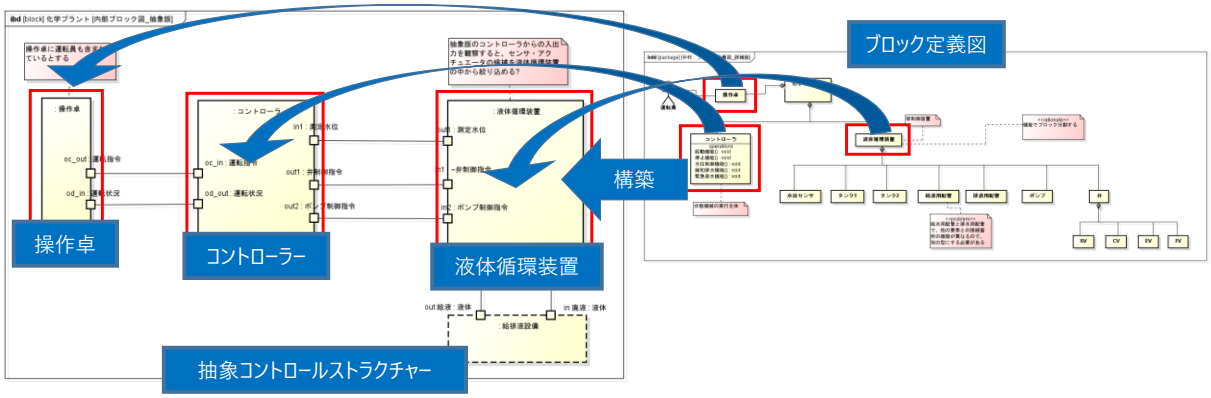

図 4.3-5　抽象コントロールストラクチャーの構築

抽象コントロールストラクチャーはシステム全体を概観するのには適しているが、分析で有用な情報を得るには、より詳細なコントロールストラクチャーが必要になることもある。本事例の抽象コントロールストラクチャーも抽象度が高すぎたため、抽象コントロールストラクチャー内のブロックに対しそれぞれ内部ブロック図を記述し（図 4.3-6）、それらを抽象コントロールストラクチャーに埋め込んで得られる内部ブロック図を詳細コントロールストラクチャー（詳細 CS）とした。（図 4.3-7）本事例では、サブシステムの内部ブロック図として、化学プラント内の体循環装置の内部ブロック図を記述した。（図 4.3-5 左側）この内部ブロック図の作成では、階層的な要素整理で使用したブロック定義図（図 4.3-4）及び化学プラントに関する文書類を参考資料とし、含まれるコンポーネントを識別し、それらコンポーネント間のコントロールアクションを追記した。

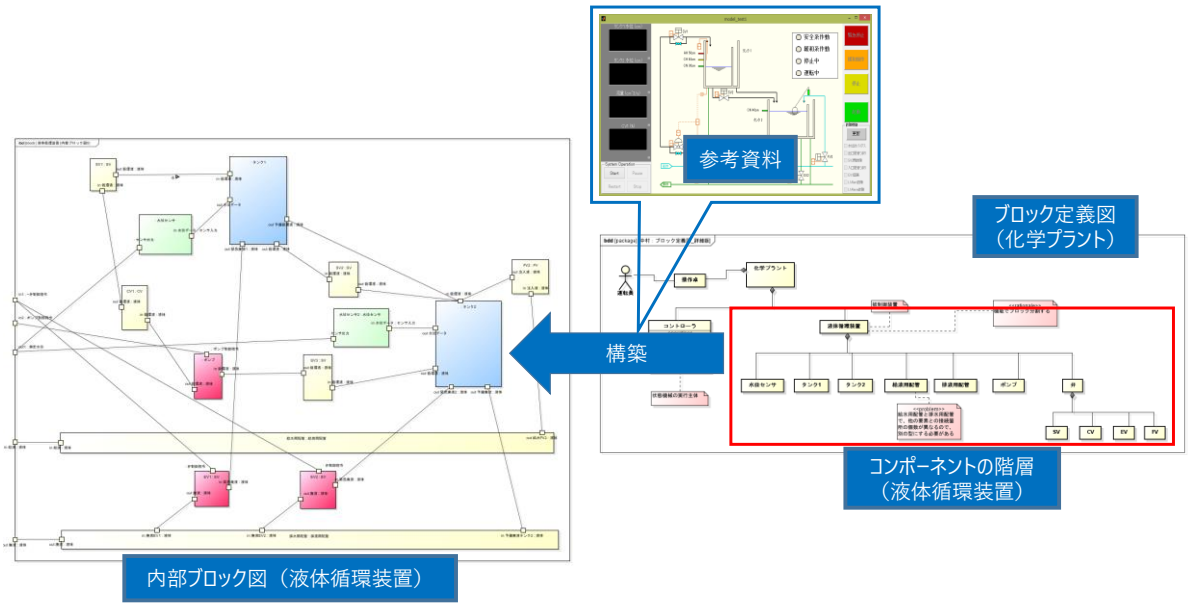

図 4.3-6　内部ブロック図（サブシステム）の構築

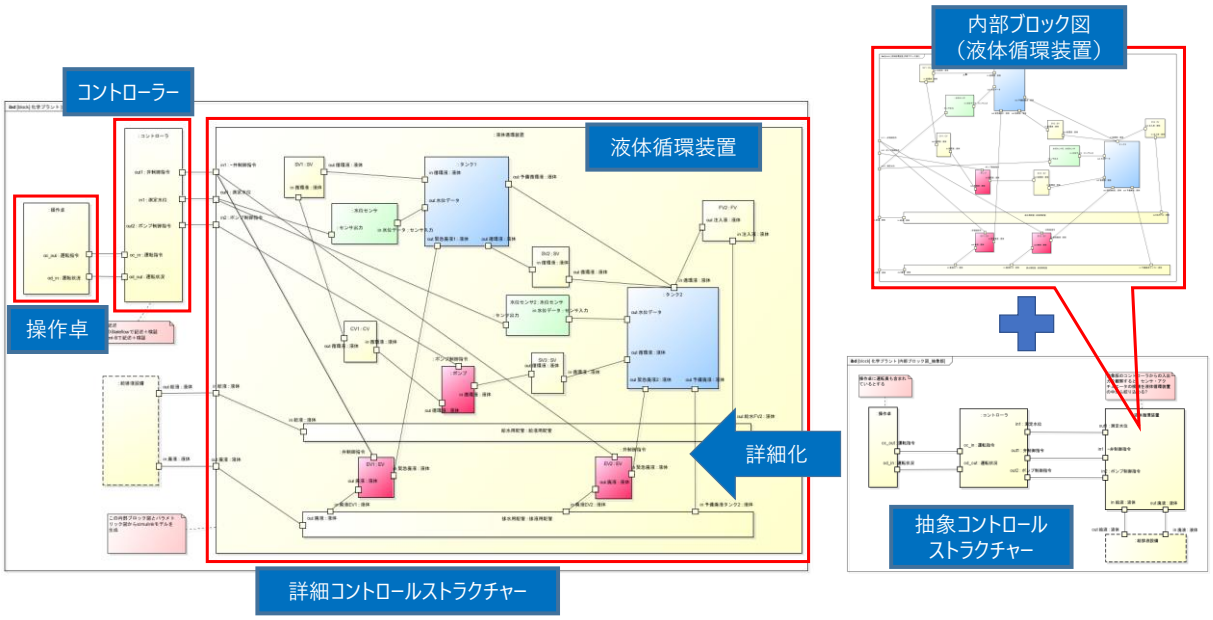

図 4.3-7　詳細コントロールストラクチャーの構築

詳細コントロールストラクチャーは分析のための十分な情報を含んでいるが、一般に複雑になるために、優先的に分析すべきコントロールループを見つけにくい。そこで、分析の観点を定め、その観点に基づき詳細コントロールストラクチャーを整理することで、優先的に分析すべきコントロールループが分かりやすいコントロールストラクチャー（最終コントロールストラクチャー）を構築した。なお、本事例の最終コントロールストラクチャーは、これまでの手順を一度実施しただけで得られたものではなく、ブロック定義図や各内部ブロック図の修正を繰り返す必要があった。

4.3.3.　まとめ

本節では、通常の開発工程で構築されるモデルと STAMP に基づく分析工程で利用されるモデルを相互に活用することで、通常の開発工程と STAMP に基づく分析工程の統合を図ることを目的とし、SysML の要求図を用いた安全制約の記述事例、及び SysML のブロック定義図と内部ブロック図を用いたコントロールストラクチャーの記述過程と記述事例を紹介した。

工程統合の副産物として高機能な SysML 記述ツールを STAMP の構成要素の記述に利用できたことは有用であった。しかし、使用したツールは SysML を記述するためのツールであり、STAMP の構成要素記述や STPA 支援を主たる目的に作られてはいない。実際、今回記述したよりも抽象度の高いコントロールストラクチャーの記述に適した記述法や、記述した安全制約やコントロールストラクチャーに基づく STPA 支援の方法はさらに検討が必要である。他方、研究レベルでは STMP に基づく STPA/CAST プラットフォームとして XSTAMPP[XSTAMPP]等が開発されており、これらのツール自身の改善及び通常の開発工程で利用されるツールとの統合もさらなる検討が必要であろう。

5. ハザード分析と障害原因仮説生成

5.1. 障害原因仮説の生成

システム障害に関する諸情報が整理され、システムの要求仕様や機能がモデル化されると、そのシステムが起こした障害（ハザード）の要因（原因）分析が可能になる。システムの故障原因を体系的に分析する方法は、大規模システムの設計には欠かせないものであり、多くの手法が提案されている[JASA2010]。故障の原因をトップダウンで分析する FTA や、ボトムアップで故障原因がシステム全体に及ぼす影響を解析する FMEA が代表的なものである。また、化学プラントのようなサブシステム・コンポーネント間の複雑な干渉があるシステムのハザード分析法としては、HAZOP が広く使われている。

一方で、複雑・大規模化する製品・制御システムでは、このような旧来のハザード分析法に限界があることも指摘されている。これらのシステムは、人間や社会と関わりがより深くなっており、複雑なサービスを提供する制御ロジック、多数のサブシステム間のコミュニケーションによるサービスの最適化ロジックなどが組み込まれている。その障害は、人間への身体的な危害を及ぼすだけでなく、大規模なサービス停止やプライバシー情報の漏洩のような社会活動への影響まで起こしうる。このような大規模・複雑化したシステムのハザード分析法として、近年、STAMP/STPA という考え方が提唱され注目されている[Leveson2012]。これは、障害の原因を、コンポーネントの故障や人間のエラーに押し付けるだけでなく、コンポーネント間やコンポーネントと人間の間のインターフェースのミスなどの幅広い視野で分析するというシステムズエンジニアリングアプローチといえる。

これらの新旧様々の障害解析法を、状況に応じて使い分け、ハザードとその要因を障害原因仮説としてリストアップすることができる。ここで大事な点は、これらのハザード分析手法を適切に使いこなすには、その前提として、適切なシステムのモデル化が必要であるということである。多くの場合、システムの設計に関わったエンジニアの暗黙の知識が必要とされるが、それでも、システムの振る舞いを「モデル」という形で明示化することで、ハザードに関わる要因を、複数のステークホルダー間で共有化し、その可能性を客観的に議論することが可能になる。このモデル化技法としては、プラント設計分野での配管計装線図（P&ID）、電気制御展開接続図（ECWD）、ソフトウェア設計分野での UML、SysML のようなものが一般的である。これらのモデルからハザード分析を行うには、各要素の振る舞いの解釈に専門家の暗黙の知識が必要であり、そのままでは実行できないモデルともいえる。また、ソフトウェア設計でしばしば用いられる状態遷移図や、サブシステム間の入出力関係を定性的ないし定量的に記載したブロック図、MATLAB/Simulink のような数値シミュレーションまで可能なモデリングツールなどは、要素の入出力間、要素間の関係に曖昧性が少なく、そこからのハザード分析がより少ない専門家の関与で可能になる。これらは実行可能なモデルといえるが、一方で、モデル表記法としての制約も多く、設計の意図がわかりにくくなるという問題も抱えている。

本章では、これらのハザード分析法の概要を、その前提となるモデル化手法と絡めて説明する。

5.2. 従来型のハザード分析による障害原因仮説の生成

前章で説明している化学プラントシミュレーターを例に、従来型のハザード分析で用いられている FTA と ETA（Event Tree Analysis）を説明する。FTA と ETA は、演繹的及び帰納的手法として大規模プラントのリスクアセスメントに欠かせないものである[Wang2013]。図 4.1-1 に示した系統図で、このシミュレーターにおけるハザードは、タンク 1 からの溢水と仮定しており、タンク 1 の水位を一

定に保つ制御弁（CV1）とコントローラー、溢水を防ぐための水位センサー、アラーム判定のコントローラー、緊急排水のための排水弁（EV1）から成っている。FTA では、この望ましくない事象（溢水）をトップ事象として、その原因となる事象を順次展開していく。この図では、何らかの故障で水位増加になった状況から、安全系の動作が失敗して溢水に至る故障原因を列挙している。シミュレーターでは、水位のアラーム状態を検出して自動排水するロジックの他に運転員による手動での緊急排水機能も設けており、これが最後の砦になっていることが FTA からもわかる。大規模プラントの設計では ETA も場合に応じて使い分けている。今回の例では、図 5.2-2 に示すように、水位増加という起因事象に続いて安全系が順次働いてゆき、その失敗により最終的に溢水につながるシナリオをリストアップでき、最終的な砦が運転員の操作であることも示されている。両図から分かるように、今回対象とした簡単な事例では、FTA も ETA も等価になるが、ある事象がシステム全体に与える影響を評価する場合は ETA が考えやすい。一方で、その事象の原因を考察する場合は FTA の方が考えやすい。障害原因究明では、その目的に応じて両者を組み合わせて使えば良い。

この例で分かるように、FTA も ETA も、システム全体でのコンポーネントの振る舞いや繋がりを知った上で作成する必要がある。今回は、図 4.1-1 に示した系統図を元に、エンジニアの常識的知識を加味して作成したが、このような方法で、実際に起こったシステムの障害の原因仮説をリストアップすることができ、原因究明に資することができる。モデル化にあたっては、ここで述べた系統図の他、現場でよく使われる P&ID や SysML のようなシステム機能の表現法が最も実用的であるが、エンジニアの知識を含めて個々のコンポーネントの振る舞いを把握する必要がある。一方、各コンポーネントの入出力関係が定量的に定義でき、直接システムの振る舞いを予測できるモデルの場合、FTA の自動化が可能になることもあり、いくつかの研究事例[Gofuku2008]はあるが、まだ、実用段階には至っていないと見るべきであろう。

FTA と双対の FMEA は、ボトムアップの故障分析法で、対象とするコンポーネントの故障モードをリストアップし、その故障原因、システムへの影響評価、さらには、影響の深刻度を表にまとめる方法である。安全設計だけでなく、障害の原因究明にも良く用いられる方法であるが、機器の物理構造を熟知したエンジニアと、システム全体への影響を評価できるエンジニアの協力で行う必要がある。特に、故障モードのリストアップには、機器の設計に関わった人間の知識が必要であり、多くのノウハウも含まれるので、今回のような第三者検証では使いにくい方法かもしれない。

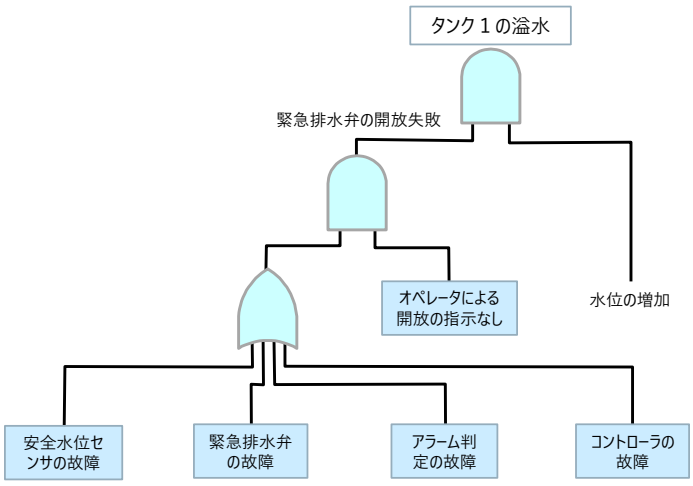

図 5.2-1　化学プラントシミュレーターでの FTA の例

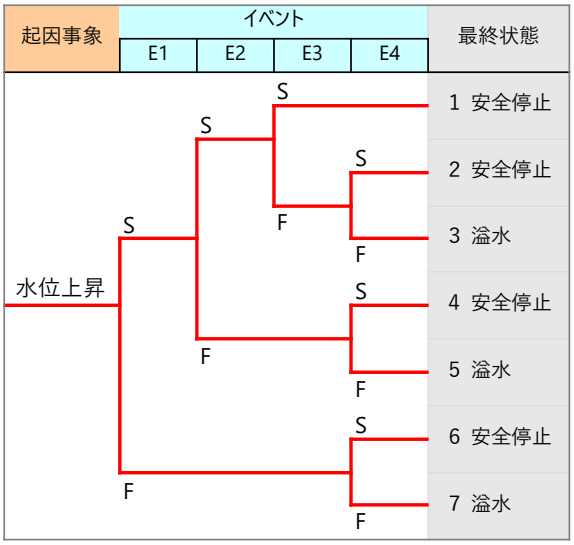

図 5.2-2　化学プラントシミュレーターでの ETA の例

5.3.　新しいハザード分析・STAMP による障害原因仮説の生成

現在の複雑な人工物システムでは、システムを構成するサブシステムやコンポーネントに不具合がなくとも、サブシステムやコンポーネントの相互作用によってアクシデントが発生することがあり、従来型のリスク分析手法では限界がある。そのため、前章でも述べたように STAMP/STPA という手法が注目されており、IPA/SEC の WG でも、その有効性の検討や普及活動を行っている[IPA2016]。ここでは、図 4.2-1 に示した化学プラントシミュレーターを例にとって、STAMP/STPA によりどんな解析が可能かを述べ、さらに、従来法である FTA などとの比較も行っていく。

図 5.3-1 に、対象とした化学プラントのアクシデント、ハザード、安全制約、機能要求を、図 5.3-2 にコントロールストラクチャーを示す。本節での分析では、アクシデントをタンク 1 からの溢水と仮定し、その前段階であるハザードは水位がアラームレベルを超えた状態としている。また、図に示したように、水位を一定に保つフィードバック制御系と、溢水を防ぐ安全制御系を階層的に配置した。本節での目的は、この外側の安全制御系に含まれるハザード誘発要因を分析することである。このコントロールストラクチャーにおいて、緊急時のコントロールアクションは、緊急排水弁の開指示である。これに STPA で与えられる四つの非安全コントロールアクション(UCA)を適用すると、図 5.3-3 UCA 表によるハザードシナリオのように、三つの具体的な UCA（A, B, C）が抽出される。図 5.3-4 は、この三つの UCA の原因となるハザード誘発要因（HCF）を、STPA により与えられるガイドワードに沿って抽出し、それをコントロールストラクチャーの図内にまとめたものである。なお、ここでは下記の 6 通りの HCF ガイドワードを想定して分析を行った。図内で示した記号で、例えば A-2 は、抽出した UCA の種類と下記の HCF のガイドワード番号を対応させている。

HCF1：入力が危険である
HCF2：制御アルゴリズムが危険である
HCF3：プロセスモデルが矛盾している、不安定、不正確である
HCF4：不十分な計測に基づく制御

HCF5：駆動装置や制御対象に欠陥がある

HCF6：その他、外部に影響される、複数のコントローラー同士の調整ミスや状況、環境の違いが原因となることがある

このようにして抽出した HCF を、図 5.2-1 に示した FTA による結果と比べたものが、図 5.3-5 である。両者を比べると、STPA では、ハザード誘発をより具体的に記述している他、通信系に関わる故障要因まで抽出されていることがわかる。

ACCIDENT：タンク 1 からの溢水

HAZARD：水位がアラームレベル以上に上昇

SYSTEM SAFETY CONSTRAINT：

タンク内の水が危険水位（アラーム）を超えてはいけない

FUNCTIONAL REQUIREMENTS：

- 制御器は、水位が高くなったら加減弁で水流を弱め、低くなったら水流を強める（一定レベル制御）
- 制御器は、ポンプ起動、各種弁の開閉操作を行ってタンクに水を一定レベルまで給水できる（起動制御）
- 制御器は、危険水位を確認したら排水弁を開く（緊急排水）
- 運転員は、計算機を介して制御器の操作に介入できる
- 運転員は、現場で直接に各種弁、ポンプを操作できる

図 5.3-1　化学プラントシミュレーターのアクシデント、ハザード、安全制約

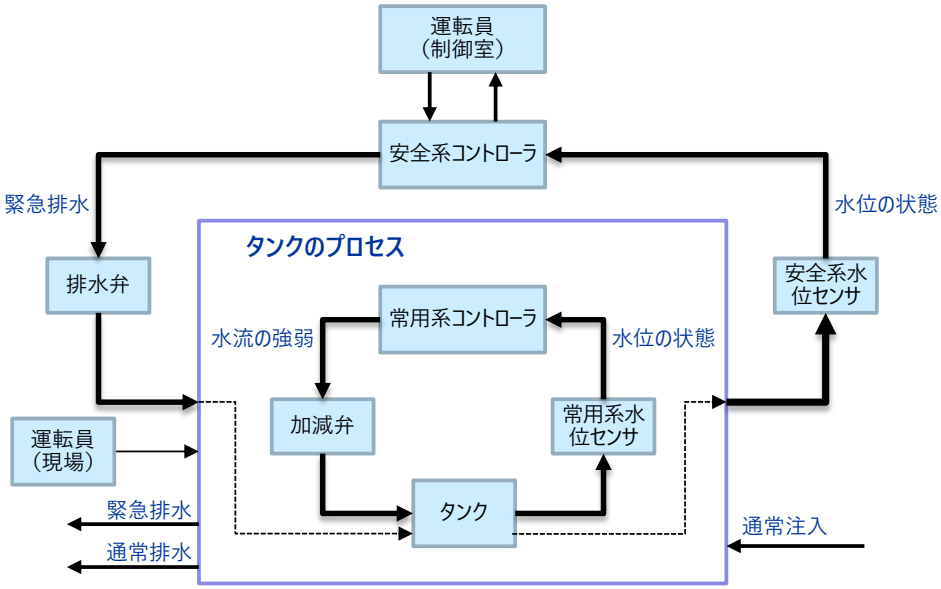

図 5.3-2　化学プラントシミュレーターのコントロールストラクチャー

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、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
緊急排水をする	UCA-A 緊急時に排水が行われない	ハザードなし （水位が下がる）	UCA-B 水位が高くなってから排水が始められるまでの時間が長い	UCA-C 水位が十分に下がり切っていないのに排水を中止する

図 5.3-3　UCA 表によるハザードシナリオ

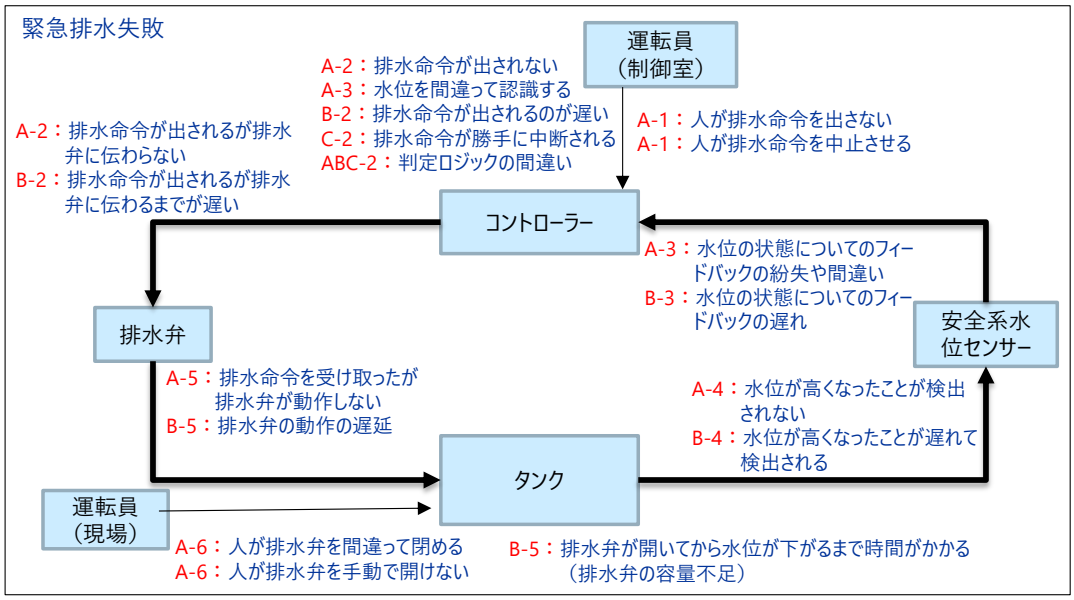

図 5.3-4　ハザード誘発要因（HCF）の抽出結果

FTA の要因	STPA の要因
安全水位センサーの故障	A-4：水位が高くなったことが検出されない B-4：水位が高くなったことが遅れて検出される
アラーム判定の故障	ABC-2：判定ロジックの間違い
コントローラーの故障	A-2：排水命令が出されない A-3：水位を間違って認識する B-2：排水命令が出されるのが遅い C-2：排水命令が勝手に中断される（制御ロジックエラー）
人間による開の指示なし	A-1：人が排水命令を出さない A-1：人が排水命令を中止させる A-6：人が排水弁を間違って閉める A-6：人が排水弁を手動で開けない
緊急排水弁の故障	A-5：排水命令を受け取ったが排水弁が動作しない B-5：排水弁の動作の遅延 B-5：排水弁が開いてから水位が下がるまで時間がかかる（排水弁の容量不足）
その他（通信系）	A-2：排水命令が出されるが排水弁に伝わらない B-2：排水命令が出されるが排水弁に伝わるまでが遅い A-3：水位の状態についてのフィードバックの紛失や間違い B-3：水位の状態についてのフィードバックの遅れ

図 5.3-5　FTA と STAMP/STPA による故障原因抽出結果の比較

今回の分析対象とした化学プラントシミュレーターの事例では、STAMP/STPA を用いた場合、四つの非安全コントロールアクションという分類と HCF ガイドワードを用いることで、FTA よりも容易にハザードを引き起こす故障要因をリストアップできている。今回の適用事例では、プラントの挙動に関する専門的知識が少ないエンジニア（実際には学生）でも STPA の利用が可能であるというメリットも明らかになった。

一方、この結果を Leveson 教授にレビュー頂いたところ、本質的には、FTA 解析を行っているに過ぎない、というコメントを頂き、代わりに、図 5.3-6 に示すようなコントロールストラクチャーと

それに基づく分析結果を教えて頂いた［Leveson2015］。以下にその概要を示し、STPA の本来の長所を説明して行く。まず、図 5.3-6 の制御構造図では、制御対象を緊急排水弁（EV1）としており、先に示した図 5.3-2 がタンク水位を制御対象としている点が異なる。また、運転員のコントロールアクションとプラント側からのフィードバックを明示している点も異なる。制御対象（排水弁と水位）の違いは本質的ではないが、運転員の挙動を明示化する点は重要である。このコントロールストラクチャーに基づいて作成した UCA の分析結果が図 5.3-7 である。ここで、前記の図 5.3-3 との違いは、コントロールアクションとして、緊急排水以外に、起動時などの給水操作と運転員による干渉操作が定義されている点である。このため、図にあるように、五つの UCA が抽出されている。先に述べた図 5.3-3 での 3 つの UCA-A, B, C は、ここでは、UCA1 に含まれている。これらの各 UCA について、非安全な行動を引き起こすハザード誘発シナリオをまとめたものが下記である。

[UCA1]

- 水位アラームレベル状態で、コンピューターが給水弁開とドレン弁閉指示を出す
- 水位アラームレベル以下になる前に、コンピューターが給水弁開とドレン弁閉指示を出す
- 水位アラームレベルに達した際、コンピューターが給水弁を閉しない、または、ドレン弁を開しない
- 水位アラームレベルに達した後 X 秒以内に、コンピューターが給水弁を閉しない、または、ドレン弁を開しない

  [Scenario 1-1]

  - コンピューターは、水位がアラームレベルに達したことに気付かなかった、または、気づくのが間に合わなかった
  - 水位情報がコンピューターに伝わらない、または、間違った情報が伝わった

  [Scenario 1-2]

  - コンピューターが、バルブ操作の指示を適切にできない
  - バルブ操作に関するソフトウェア設計エラー
  - バルブ開閉の指示は出たが、バルブがうまく動かなかった

[UCA2]

- コンピューターが給水弁が完全に閉まる前に指示をやめる、または、ドレン弁が完全に開く前に指示をやめる

  [Scenario 2-1]

  - コンピューターはバルブ開閉信号を出したので、バルブは指示どおりの状態にあると、コンピューターは思っている
  - バルブの位置情報のフィードバックがないか、間違っていたため、開き終わったと勘違いして、開閉指示をやめる

[UCA3]

- コンピューター操作が溢水を引き起こしそうな時に、手動操作で介入しない
- コンピューター操作が溢水を引き起こしそうな時に、手動操作の介入が X 秒以上遅れる

  [Scenario 3-1]

  - 運転員が、コンピューターが適切に動作していないことに気づかない、または、気づくのが遅れる
  - 運転員への情報の提供がないかまたは不適切（プラント状態が適切に得られない）
  - 他のタスクに気を取られて気づかない
  - コンピューターが正しく動作しているとの思い込み

[Scenario 3-2]

- 運転員がバルブ操作を適切にできない
- バルブ操作方法を知らない、または、コンピューターへの介入方法を知らない
- 設計ミスでオペレーター指示を受け付けない（運転員からの矛盾する介入操作など）

[UCA4]

- コンピューターが意図通りに動いている時に、運転員が介入して溢水を引き起こす

[Scenario 4-1]

- コンピューターが適切に制御しているのに、運転員がこれを不十分と誤解する
- 運転員への、水位に関する間違った情報提示、または、不適切な操作ガイドの提示
- コンピューターの動作に関する設計を理解していない

[Scenario 4-2]

- 不注意によるコンピューターへの間違った操作介入
- 操作介入機能の設計ミス（間違えやすい設計）

[Scenario 4-3]

- 運転員の意図的な操作介入（安全マージンを犠牲にした生産効率の向上など）
- 過負荷やノルマなどのプレッシャー
- 安全文化の不備

[UCA5]

- 誤解により運転員が緊急排水を中断する、または、給水を継続する

[Scenario 5-1]

- 運転員が適切な操作をおこなったのに、それを認識せずに排水を中断したり給水を継続したりする
- 運転員が適切な操作を行った後に、それが終わったと思って操作を中断する

これらのシナリオを少し一般化して、運転員とコンピューターの間、ならびに、運転員とプラントの間の HCF としてまとめたものが、図 5.3-8 である。Leveson 教授の教科書[Leveson2012]で示されている HCF ガイドワードに比べて、人と機械（コンピューターとプラント）の間の HCF が詳しく記載されている。これらの誘発要因は応用領域によって異なるとはいえ、過去の事故原因を考えてみると、一般的に成り立ちうる要因でもある。近年の組込みシステムでは、コンピューターを介した制御が一般的になっているが、このような制御では、状態表示画面がコンピューターの不具合でフリーズした際に、それに気づかないこともありうる。また、すべての制御がコンピューターを介して行われる設計と、人間がプラントにコンピューターを介さずに直接アクセスして制御することもできるような異なる設計の考え方もある。安全が最重要視されるシステムでは、コンピューターがダウンした際の対処方法も設計に組み入れておくべきであるが、ここで示したようなハザード誘発要因をまとめておくことで、設計の際の気づきとして用いることもできる。

本節での分析結果を、図 5.2-2 に示した従来の分析結果を比べると、図 5.3-9 に示すような FTA の限界が見えてくる。FTA では、定格運転を想定して、そこで発生した故障への対応を考えていたが、起動時の複雑な制御操作時のミスや、緊急対応時の機械と運転員のミスマッチによる不具合、特に、機械の制御操作に運転員が介入することで却って事故を引き起こしてしまう可能性などのハザード要因の分析で限界があることが分かる。一方で、STAMP/STPA では、四つの UCA というトップダウンの分析で、前記の複雑な状況下でのハザード分析を効果的にできているように見える。

STAMP/STPA は新しい技術であり、研究レベルでは、多くの複雑な安全制御への応用が検討されている[Thomas2012]。今後も、どのような可能性があるかを具体例を通して見極めてゆくことが大事である。

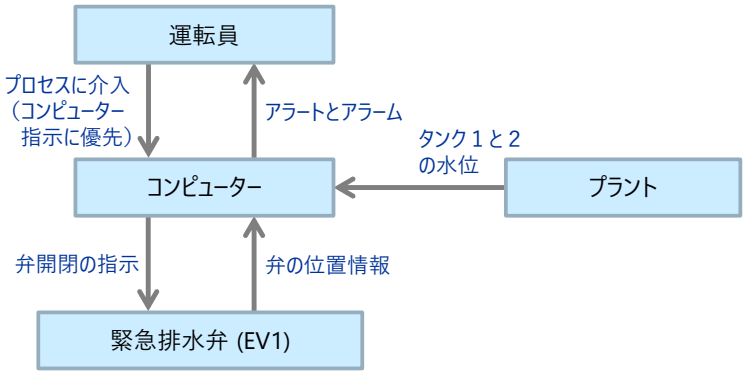

図 5.3-6　化学プラントのコントロールストラクチャー（MIT 分析）

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、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
Computer Action 給水弁(SV1)開とドレン弁(EV1)閉		水位アラームレベル状態で、コンピューターが給水弁開とドレン弁閉指示を出す（UCA1）	水位アラームレベル以下になる前に、コンピューターが給水弁開とドレン弁閉指示を出す（UCA1）	
Computer Action 給水弁(SV1)閉とドレン弁(EV1)開	水位アラームレベルに達した際、コンピューターが給水弁を閉しない、または、ドレン弁を開しない（UCA1）		水位アラームレベルに達した後 X 秒以内に、コンピューターが給水弁を閉しない、または、ドレン弁を開しない（UCA1）	コンピューターが給水弁が完全に閉まる前に指示をやめる、または、ドレン弁が完全に開く前に指示をやめる（UCA2）
Human Action 運転員の手動操作の介入（コンピューター指示に優先）	コンピューター操作が溢水を引き起こしそうな時に、手動操作で介入しない（UCA3）	コンピューターが意図通りに動いている時に、運転員が介入して溢水を引き起こす（UCA4）	コンピューター操作が溢水を引き起こしそうな時に、手動操作の介入が X 秒以上遅れる（UCA3）	誤解により運転員が緊急排水を中断する、または、給水を継続する（UCA5）

図 5.3-7　UCA 表によるハザードシナリオ

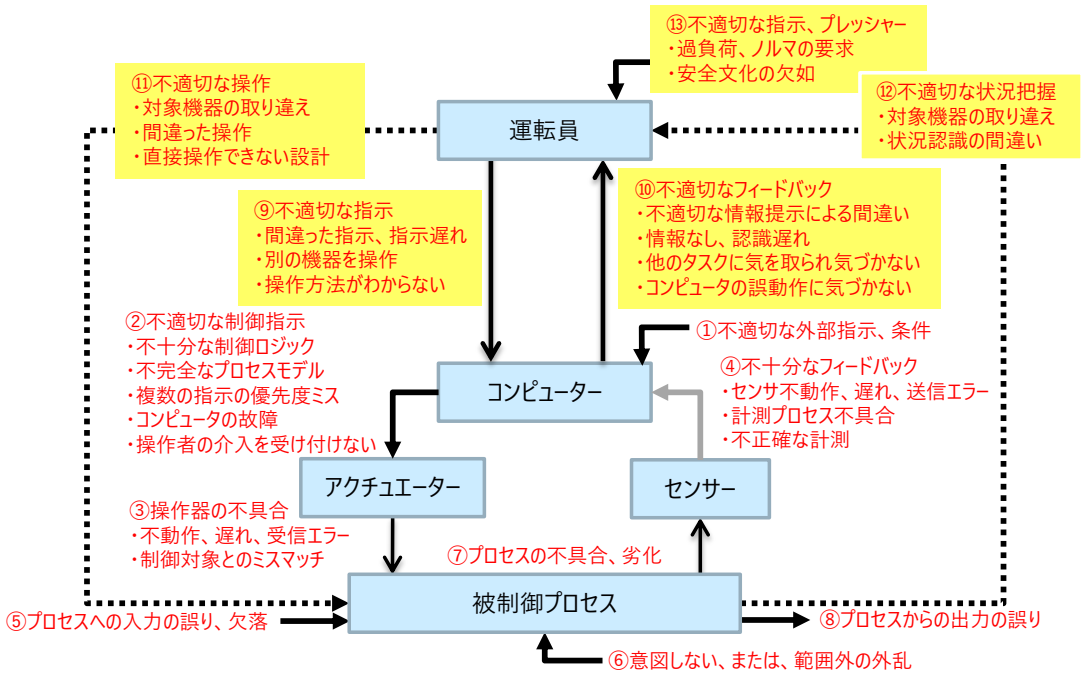

図 5.3-8　人間系も含めたハザード誘発要因のまとめ

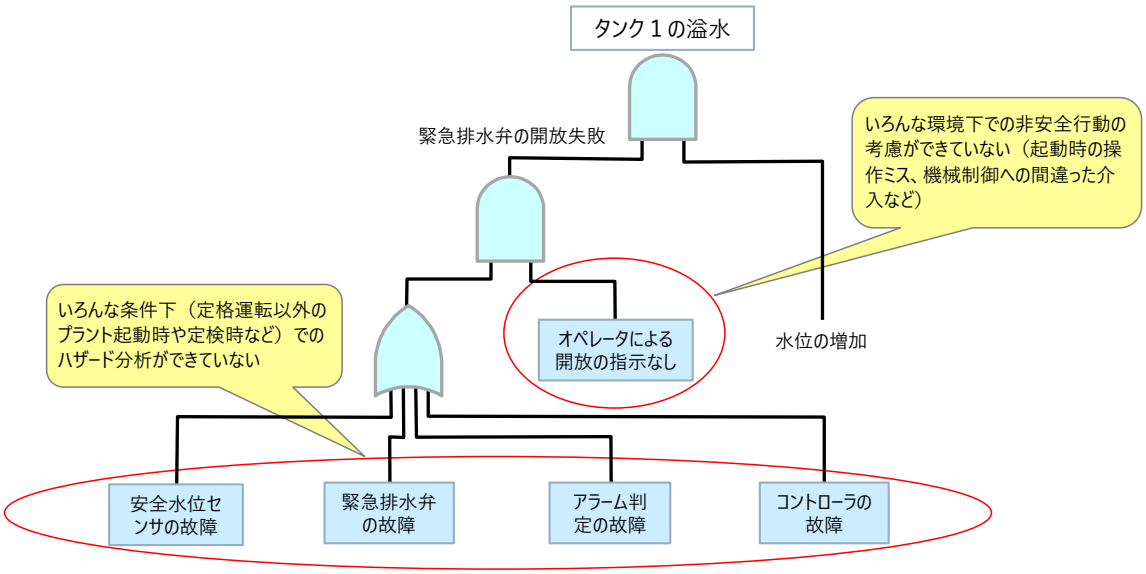

図 5.3-9　FTA の限界

6. 仮説検証の要素技術

6.1. ソフトウェア形式検証

6.1.1. 形式手法の概要

一般にある手法が形式的であるとは次の要件を満たす必要がある。

1. 記述手法が厳密に定義されていること。厳密に定義されているためには構文が形式文法で定義されていることは当然ながら、構文の意味が数学モデルなどで精緻に定義されている必要もある。これにより、記述には曖昧さがなく、計算機上で処理可能となる。
2. その記述上で有用な性質を調べるための数理的理論、手続き、アルゴリズムなどが提案されていること。有用な性質とは例えば、記述の整合性、安全性、進行性など情報システムに共通して現れる望まれる性質、記述間のある種の関係式による順序付け（例えば、詳細化、抽象化、模倣関係など）があげられる。

多くの形式手法は 2. に基づいて計算機処理可能なツールが提案されている。またこれらのツールでは多くの場面でスケーラビリティの効果が有用性の基準として議論になっている。というのは、多くの検証手続きあるいはアルゴリズムが対象とするシステムのサイズに比例して大きな計算量を要することがあり、少しでも大きなシステムになると今日の計算機でも手に負えない（膨大なメモリー量を必要とする、計算が常識的な時間で終了しない）という状態爆発の問題を抱えているからである。この状態爆発に対するさまざまな対策が理論面、アルゴリズム面、ツール運用面のさまざまな視点から精力的に研究されている[Clarke2001]。

システムの記述は上記のように構文およびその意味とともに数学的に厳密に定義されている必要がある[中島 2012]。

多くの情報システムはオートマトンモデル（状態遷移モデル）との親和性が高い。従って対象とする情報システムを適切に抽象化し、オートマトンモデルで表現することができれば、そのオートマトン上で多くの形式手法の恩恵を受けることができる。

オートマトンでシステムをモデル化した場合、以下のような性質を調べることが可能である。別途与えたレファレンスモデルに対する等価性、言語包含関係、特定の状態への到達可能性、デッドロック判定、別途与えた論理式に対する充足可能性などである。とりわけオートマトンを意味モデルとみなし、別途与えた時相論理式の充足可能性判定を行う方法はモデル検査[Clarke1999]と呼ばれ、近年急速に研究が進み、実用的ツールも多く公開されている。

ユーザーからの視点に立てば、書かれた記述に対して、さらにユーザーが高度な論理判断を行いながら、望ましい性質の証明（保証）を半自動で作っていくもの（証明支援システム）と計算機がすべての論理的可能性を自動かつ、もれなく調べていくもの（モデル検査的アプローチ）の 2 つに大別することができる。

前者のアプローチには多くの証明支援システムが含まれる。後者についても多くの使用モデルのバリエーションに基づいた分類が可能である。

- SPIN [SPIN], (Nu)SMV [NuSMV]
  - モデル　： Kripke 構造
  - 検証性質 ： LTL / CTL
- UPPAAL [UPPAAL]

- モデル　：時間オートマトン
- 検証性質：拡張 CTL

• PRISM [PRISM]
- モデル　：Markov 鎖
- 検証性質：PCTL / CSL

その他（分類からはずれる新規方法論）

• Alloy Analizer [Jackson2011]
• EventB [Abrial2010]

情報システムのモデリング（形式的な記述方法での記述）はシステムの設計段階で行うことが多いが、この報告書のテーマであるように事故事後の原因究明にも役立つ。システムで発生した障害は、システムにとって望ましくない性質と捉えることができ、モデル検査により、そのモデル上でその障害が発生するかどうかをもれなく調べていくことが可能である。

ここでは種々のモデリングのうち、モデル検査を前提としたモデリングについて UPPAAL で用いられている拡張時間オートマトン(以下 TA と呼ぶ)による制御システムのモデリングを例に説明する。

TA は、ロケーションという遷移状態とトランジションと呼ばれる遷移からなる。グローバル状態変数として有限個のクロック変数を持つ。UPPAAL の拡張ではさらに有限の値を持つ有限個の変数（以下離散変数と呼ぶ）をグローバル状態変数として持つことができる。トランジションではクロック変数と整数定数の比較などをガード（遷移するための制約）として課すことができる。またロケーションではクロック変数と整数定数の比較式を不変条件（モデルにおいて成立すべき式）として記述できる。クロック変数は通常の時計と同様に時間経過にともない一様にそれぞれの値を増加させる。

トランジションにおいては、一部のクロック変数の値を 0 に設定（リセット）することができる。また UPPAAL の拡張では離散変数についてそれらの値を制限のある数式で更新することができる。

TA の形式的定義と意味モデル。簡単な記述例などは[IPA2015]を参照のこと。

[IPA2015]のライトの例題の動作概要を以下で説明する。システムは、ライトの利用者とライトの２つのサブシステムからなる。ライトの利用者は勉強中かリラックス中かの２つの状態を持ち、ライトのボタンを押して、各状態にあった光の強さを調整する。ライトはスイッチを１つ持ち、ライト消灯中にスイッチが１回押されれば暗く(dim)光り、その状態で 10 単位時間経過後に再びボタンが押されれば消灯(off)になり、10 単位時間経過前にボタンが押されれば明るく(bright)光る(図 6.1-1)。

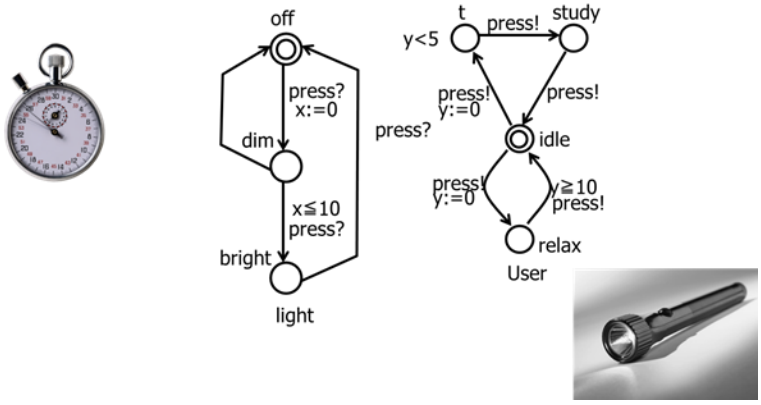

図 6.1-1　ライトの動作モデル

このようなモデルに対して例えば以下のような性質を検証する。

A□(User.relax -> light.dim)

この式はユーザ(User)がリラックスしている(relax)ときにはライト(light)は常に暗く光っている(dim)という性質を意味している。この式がモデルにおいて成り立つか否かをモデル検査で検証することができる。利用者がボタンを十分早い時間で操作し(5 単位時間以内)勉強モードになるようにして、かつ 5 単位以上 10 単位時間未満の間はボタンの操作をしない場合は上記の性質は成り立つが、これ以外の操作系列がある場合は、上記の性質が成り立たない状況が起こりえる。その場合モデル検査器は成り立たないという判断とともに、成り立たない状況に陥る状況列を示す。

このようなことはシステムにおいて起こってほしくない障害が再現できるかということを調べることにもつかえる。

6.1.2. 形式検証法を用いたシステム障害診断

システムにおいて発生した障害に対してその原因を診断する方法の 1 つとして形式的検証方法を利用することを考える。システムを形式的にモデル化し（モデルを形式的に記述し)、システムにおいて成立してほしい性質として発生した障害が起きないこととする (性質を定義する)。そのモデルと性質に対してモデル検査器を用いることができれば、モデル検査器はその性質がモデルにおいて成り立つか成り立たないかを判断し、成り立たない状況があれば、判例を示してくれる。その判例を用いることにより障害の特定及び障害の除去に役立てられる可能性がある。

形式手法を用いた例として化学プラントを取り上げる。化学プラント水槽制御プログラムの仕様を以下に示す。このプログラムは水槽の溢水防止のための緩和系ロジックである。

タンク 1 について

- 運転員は緊急停止と緩和操作が手動でできる
- 緩和操作では 5 秒間排出弁を開き、同時に 15 秒間（含む 5 秒間）は新たな開指示を受け付けない
- 運転員の指示は常に優先する（15 秒間の禁止区間でも受け付ける。自分自身の過去の指示に対しても優先する）
- 水位がアラートレベル（40cm)を越えると上記と同様の緩和操作を行う

この仕様に対して、5 秒と 15 秒のワンショットタイマー（ディレイタイマー）を用いた実装に対して問題を調べてみる。

この水槽に対して以下のトラブルが発生したと仮定する。

水槽溢れに対して手動とセンサー検出の両方を起因とした緩和操作が起こりうる。想定される正常シナリオでは、センサー、手動ともに排水すべき状況で排水できる。しかしながら実際に発生しえる異常シナリオでは以下の状況が起こりえる。

- センサーがオンで、緩和期間をすぎているのに排水できない
- さらに運転員の手動の操作も受け付けなくなる

故障診断のためのモデリングを以下のように進める。

原因究明の立場からは網羅的に動作を調べる必要がある。この観点からは Simulink を用いたシミュレーションでは網羅的な探索は現実的ではない。網羅性に着目しモデル検査可能性の観点からの妥当性を考慮すると制御系にモデルを絞って、時間オートマトン、通常のオートマトンを用いて形式手法を適用するのは妥当なモデル化と考えることができる。

水槽システムの制御ロジックはディレイタイマーを用いて比較的容易に設計できる。ディレイタイマー自体もその動作仕様は設計者によって曖昧に定義されていることがある。

システムのモデル化においては、この部分の定義を精緻化（形式化）することからはじめる必要がある。[Björkman2009]で使われているディレイタイマーの UPPAAL の時間オートマトンなどを参考に、通常のディレイタイマーとリセットを持つディレイタイマーをまずはモデル化した。このような 2 つのディレイタイマーを組み合わせて全体の制御部を時間オートマトンでモデル化できる（図 6.1-2 参照)。

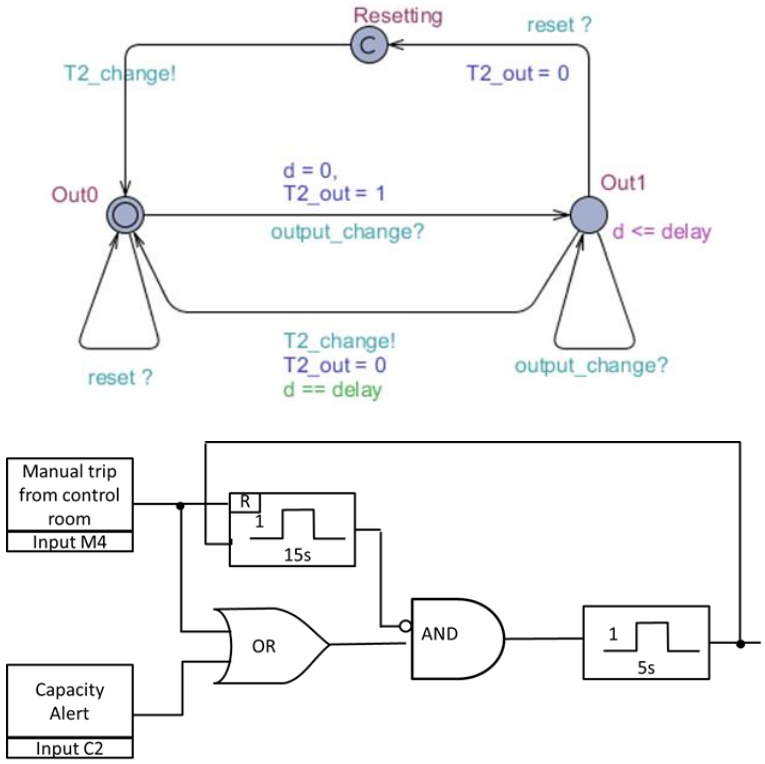

図 6.1-2　リセット付きタイマーの時間オートマトンモデルとシステム全体のブロックダイヤグラム

詳細な記述例などは[IPA2015]を参照のこと。

この水槽制御モデルに対して検証したい検証性質 Pは次の意味であり、それを以下の時相論理式で表す。

“排水が抑制されていなく、かつ、水位レベルセンサーが警告レベルの状態であるときに、いつか「排水モード」になるべきである。”

(In.C2 and T2_out == 0) --> T1.Out1

この検証性質に対して実際にモデル検査を行うと、P は成立しないことが確認できる。

このような結果のときは、モデル検査結果に対し反例を解析することにより故障原因を解析できる

ことがある。

反例トレースの一例の概要は以下のようになる。

- 最初の 5 秒の緩和操作の間に、再度、運転員からの緩和指示が出た場合、15 秒の操作禁止はリセットされるが、そのとき水位がアラート越えの状態に留まっていた場合、AND 出力は 0→1 に立ち上がるが、5 秒タイマーは動作中で、これを受け付けない。5 秒経過後も AND 出力が 1 で継続（水位アラート状態）の場合、AND 出力は 1 のままなので、5 秒タイマーは駆動されない
- この状態で、運転員が緩和指示を出しても受け付けなくなる（運転員優先の逸脱）

このようにモデル上に存在する障害原因が診断された。

化学プラントの例で考えると、水槽の制御を表現するモデルに対して「排水が抑制されていなく、かつ、水位レベルセンサーが警告レベルの状態であるときに、いつかは排水モードになるか。」を調べ、もし成り立たなければ、反例すなわち障害を起こす実行系列を得ることができる。

モデル検査のアプローチをまとめる。

利点：網羅性からくる検査結果の妥当性
欠点：適切なモデル化が必要。モデル化できていないこと（例えば抽象化しすぎて、障害が発生する状況が発生しえないモデルを作ってしまった場合）は調べられない。
限界：そのクラスで表現できないなら、より広い能力をもつ手法を選ばないといけない。

形式手法を用いるにあたっては、システムに対する適したモデル化、妥当な検査式の作成、高速なモデル検査ツール環境の整備、反例からのシステム障害の推測などの課題がある。これらにおいては、従来からの多くの形式手法を用いた報告から障害原因診断に活用できる内容もあるし、今後障害原因診断を形式手法の一つの分野として発展できる可能性もあると考えられる。

6.2. 障害再現法

障害原因診断をモデルベースで実施するためには、様々な障害を再現できるモデルの構築が肝要となる。実際、システムを構成するソフトウェアやハードウェアの誤りを模擬できるモデルを構築し、そのモデルを用いてシステム障害を再現できれば、実際に障害を再現させることが極めて困難なシステムの障害原因診断に有用であり、一般的なシステムに対しても診断コストの低減を期待できる。

本節では、モデルベースの障害再現法に関し、障害を再現できるモデルの構築方針について解説し、Simulink による障害を再現可能なモデルの構築事例を紹介する。さらに構築したモデルによる障害模擬シナリオを紹介する。

6.2.1. 障害を再現できるモデルの構築方針

システム障害を再現できるモデルは、通常時の動作を模擬でき、かつシステムを構成するソフトウェアやハードウェアの誤り時の動作を模擬できる必要がある。誤り時の動作を表すモデル（誤りモデル）は通常動作を表すモデル（通常モデル）に依存して構築される。

誤りモデルとして、システムのコンポーネントの誤り時動作を模擬するモデルと、各コンポーネント間の情報供給の誤りを模擬する動作モデルの二種類のモデルを構築する [Delange2014, レブソン 2009, Leveson2012]。システムコンポーネントの誤りモデルは、対象とするコンポーネントとそのコンポーネントに対する既存の類型化された誤りを基に構築する。また、各コンポーネント間の情報供給の誤りを模擬する動作モデル（誤りモデル）の構築には、STAMP/STPA の以下の 4 つのガイドワ

ードが利用できる[レブソン 2009, Leveson2012, IPA2014]。ここでは、ガイドワードに基づくこれら 4 種類の誤りモデルを一つのコンポーネントとしてまとめ、そのコンポーネントを故障模擬コンポーネントと呼ぶことにする。

通常モデルに加え、これらの誤りモデルを利用することで、例えば排水バルブが閉故障しかつ安全・緩和装置から緊急排水バルブへの緊急排水命令が供給されないときに、タンクが溢れるといった障害を再現できる。もしタンクが溢れるといった障害が実際に観測され、「排水バルブの閉故障」と「安全・緩和装置から緊急排水バルブへの緊急排水命令が供給されない」という二つの誤りモデルからこの障害が再現できるのであれば、これらの誤りを引き起こす不良が障害原因として推定できる。さらに、各コンポーネント間の情報供給の誤りを模擬する動作モデルを構築することで、各コンポーネント内に誤りがなくても、複数のコンポーネントを組み合わせることで、システム全体として障害が発生するという事象も再現できる。

6.2.2. 事例：化学プラントシミュレーターの誤りモデル

昨年度報告書[IPA2015]の「付録 A 化学プラントシミュレーター」で解説される Simulink を使った化学プラントシミュレーター（以下単にシミュレーターと呼ぶ）を基に、6.2.1 項の方針に沿って誤りモデルを構築する。本節では各コンポーネント間の情報供給の誤りを模擬する動作モデルを STAMP のガイドワードを基に構築する。例えば、シミュレーター内のコンポーネント間の情報供給として、監視装置から安全・緩和装置へのコントロールアクション：緊急停止命令にガイドワード：Not Provided を適用する場合、その誤りモデルは「監視装置からの緊急停止命令を安全・緩和装置へ供給しない」動作を模擬する。これらの誤りモデルをまとめたものとして不良挿入（Fault Injection）を実施するためのコンポーネント（故障模擬コンポーネント）を構築し、シミュレーターの監視装置コンポーネントから安全・緩和装置コンポーネントへの情報供給経路の途中に、その故障模擬コンポーネントを挿入することで、これらの誤りを模擬する（図 6.2-1）。

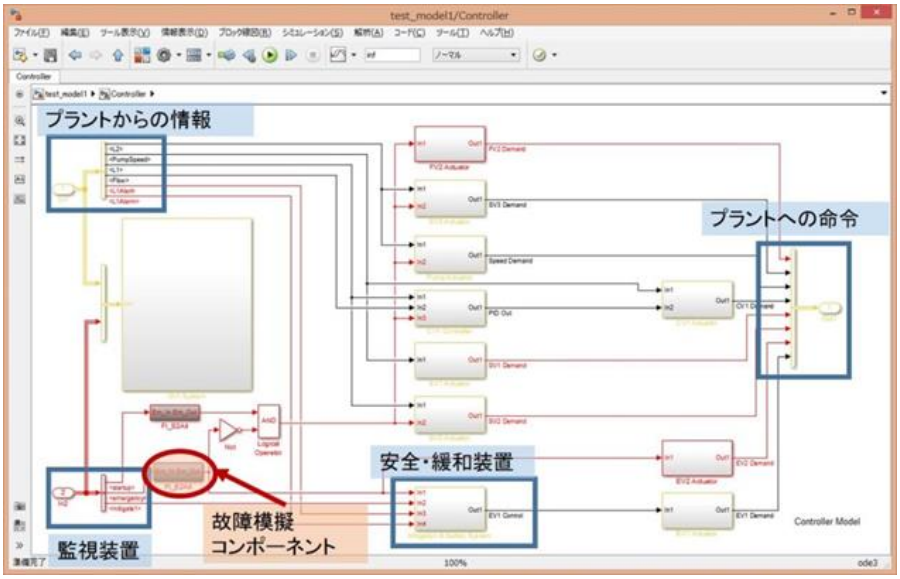

図 6.2-1　故障模擬コンポーネントを含む化学プラントシミュレーター

6.2.3. 障害模擬シナリオ

シミュレーターと前述の誤りモデルによる三つの障害模擬シナリオを紹介する。本項では、障害と

してタンク 1 の溢水を扱うことにする。(原典では三つの障害模擬シナリオを紹介している）各コンポーネント間の情報供給の誤り動作モデルは 6.2.2 項で構築したものを用いる。

障害模擬シナリオ：「SV2 閉故障」＋「EV1 固着」＋「緊急停止命令 NOT Provided」に加え、「監視装置から安全・緩和装置への起動命令が Incorrectly Provided（一旦起動命令を供給した後は、起動命令が供給され続ける)」であると仮定する。(注意：最後の仮定を再現するには、監視装置と安全・緩和装置間の起動命令の供給経路に対して故障模擬コンポーネントを別途挿入する必要がある）すると、タンク 1 が満水になった後も注水用の各バルブが閉じず、タンク 1 は溢水に至ることがシミュレーション結果から分かる。この結果から、このシナリオの仮定が障害原因として推定される。

6.3. ハザード要因の切り分け

5 章で述べた方法で障害原因の仮説が生成された場合、この仮説の妥当性を観測データから推定し、少数のサブシステムの原因仮説として絞り込む必要がある。これが、ハザード要因の切り分けに相当し、古くから、電力・化学プラントなどで、システム異常診断技術として研究が進んできた分野であり、フォールト検知と識別（Fault Detection & Isolation, FDI)というように呼ばれている。この呼び方から分かるように、フォールト（異常）検知技術と、フォールト識別（障害原因箇所の切り分け）技術という二つの技術からなっている。この中の異常検知技術は、近年大きく進展している機械学習理論を用いて一般化され、機械やプラントだけでなく、下記のように、多様な情報分野での応用が進んでいる。[山西 2009, 井手 2015a, 井手 2015b]これらに共通に含まれている考え方は、正常時（通常の状態）の観測信号の挙動を、何らかの統計モデルで表現しておき、異常時の観測信号の挙動を検定のような統計的手法で検出するものである。

- 機械系：故障検知
- 製造系：不具合発生検知
- 不審者の検知
- コンプライアンス：情報漏洩検知、不正検出
- マーケティング：流行のブレイク検出
- セキュリティ：攻撃・侵入検出。不審者の検知
- Web：トピック潮流の変化検出
- ネットワーク運用系：トラフィック異常検出
- システム運用系：異常検出

一方、FDI のもう一つの要素技術である識別に関しては、複数のサブシステムのどこが故障しているかという視点以外に、どんな故障モードか（突発的な外乱なのか、漸近的なドリフトなのかなど)、観測信号は正しく計測されているか、などのシステム的な切り分けがまず必要になる。その上で、機械系の故障では、サブシステム内の要素の劣化や、観測センサーの故障モード、さらには、劣化などを加速した環境要因などの物理的要因を推測することになる。対照的に、情報システムの障害場所の切り分けでは、サブシステムそのものの故障はあるが、同時に、サブシステム間の通信タイミングのズレなどの論理的要因の推測が必要である。

診断対象に応じて有効な診断アルゴリズムは異なるため、魔法の杖のような汎用的な方法はないが、プラント系の異常診断で確立されたいくつかの手法をここでは紹介する。まったく同じような異常診断の考え方が、情報システムの性能監視にも使われるようになっていることもあり、代表的な手法の

紹介は有意義であろう。

異常診断システムの一般的な構成を図 6.3-1 に示す。システムから得られる観測データ（時間履歴データ）は、後述する参照モデルを通して予測される値との残差系列に変換される。対象システムが正常であれば、この残差系列は、平均値ゼロの周りにばらついたヒストグラムになる。一方、異常が発生すると残差系列は平均値ゼロからずれるため、より明確な形で異常発生を検出することができる。この中で研究されてきた診断法は、モデルに基づく方法と、観測時系列データの統計分析に基づく方法に大別される。このうち、モデルに基づく方法を図 6.3-2 で説明する。これらのモデルは、まず、物理モデルや設計モデルなどの第一原理に基づく方法と、観測データからモデルを作成する経験則による方法に分けられ、さらに、後者は、パラメトリック法とノンパラメトリック法に分けられる。個々のモデルの説明は省略するので、文献[Hines2008, Jiang2007, Wegerich2004, 渡邉 2014, NEC2015, 井手 2015a, 井手 2015b]等を参照されたい。ここでは、典型的な監視法を、前記の化学プラントシミュレーターを例にして説明する。

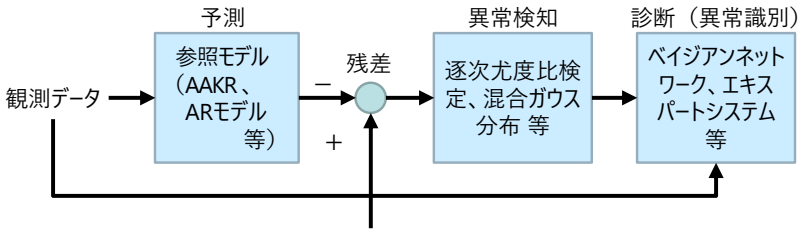

図 6.3-1　異常診断システムの一般的構成

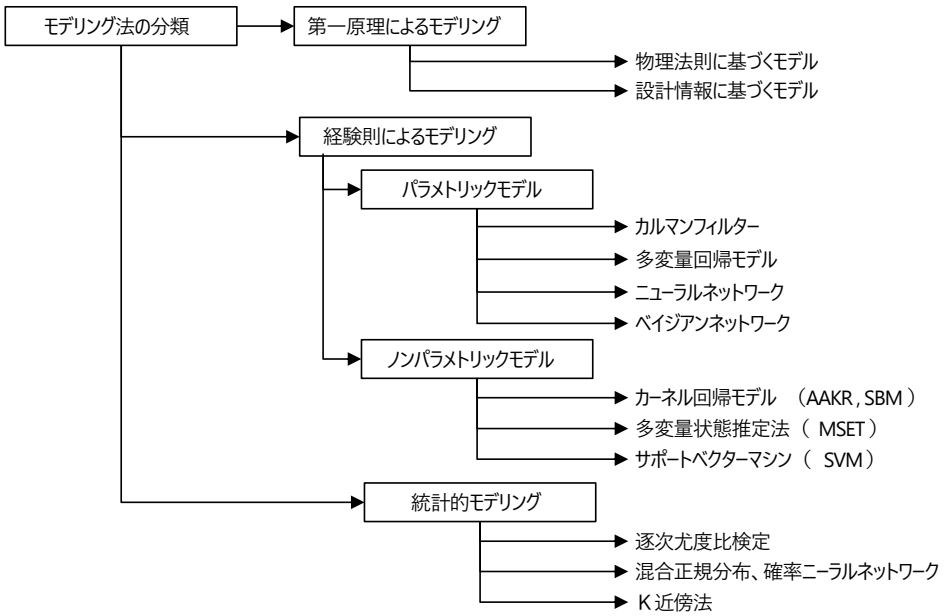

図 6.3-2　異常診断で用いられるモデル化手法の分類

以下で説明するのは、図 6.3-1 で紹介した異常診断システムにおいて、参照モデルとして自己参照型カーネル回帰モデル（Auto-Associative Kernel Regression, AAKR）を異常検知手法として、逐次尤

度比検定（Sequential Probability Ratio Test, SPRT）を用いて、化学プラントシミュレーターの監視を行った例である。AAKR モデル[Hines2008]は下記式で表現される簡単なモデルである。

$X_i(i=1,N)$ は、m-次元、N 点からなる参照データ（訓練データ）であり、プラントの正常運転時の観測データから構成される。x は、異常の有無を判定すべき特定の時点の m-次元の観測データである。また、$\hat{x}$ は、その予測値であり、ε が残差になる。カーネル関数としてはガウス関数を用いており、h はパターン類似度を定義するパラメータである。簡単なモデルであるが、これは m 次元空間において、過去の類似パターンを調べ、その類似度に応じて重みを求め、さらに、その重み付き平均で予測値を求めるということを意味する。このノンパラメトリックモデルは、予測の際に線形回帰モデルやニューラルネットワークのように間にパラメトリックモデルを介さない分だけロバストになる。一方で、過去に経験した m 次元空間の領域内でしか予測できないことに注意が必要である。

$$w_i(x) = K(x, X_i) = \frac{1}{\sqrt{2\pi h}} \exp\left(-\frac{|x - X_i|^2}{h}\right) \ (i = 1, N)$$

$$\hat{x} = \frac{\sum w_i X_i}{\sum w_i}$$

$$\varepsilon = x - \hat{x}$$

このようにして残差の時間履歴が求まると、次に、SPRT を用いて異常判定を行う。SPRT は、正常状態の仮説 H_0（例えば、平均a_0，分散 σ^2 の正規分布に従う）と、異常状態の仮説 H_1（例えば、平均a_1，分散σ^2 の正規分布に従う）の確からしさの比を用いて異常判定を行う。この場合、確からしさの比（尤度比）の対数をとったもの $\lambda(t)$は、下記のように、時刻 t での残差 $\varepsilon(t)$を得るごとに、逐次的に更新できる。

$$\lambda(t) = \log\frac{P(\varepsilon(t)|H_1)}{P(\varepsilon(t)|H_0)} = \lambda(t-1) + \frac{(a_1 - a_0)}{\sigma^2}\left\{\varepsilon(t) - \frac{1}{2}(a_1 + a_0)\right\}$$

異常判定は、誤検出率(False alarm rate，α)と非検出率(Miss alarm rate，β)を用いて、下記のように行う。

$\lambda(t) < B$　　：　H_0を採択（正常）

$\lambda(t) > A$　　：　H_1を採択（異常）

$B < \lambda(t) < A$ ：　監視継続（判定保留）

ただし、A=log((1-β)/α)、B=log(β/(1-α))　である。

このようなアルゴリズムを用いて、前記の化学プラントシミュレーターの定格・定常運転時の挙動を監視した事例を以下に示す。図 6.3-3 は、t=300 秒でランプ状に 60 秒かけて水位センサーが 5%マイナス側にドリフトした事例のトレンド波形である。水位低下を補償するため、コントローラー指示(PIDout)が増加し、流量も増えている。60 秒過ぎると見かけ上の水位が元の定格値に戻り、流量も元に戻るという変化である。しかし、実質水位は 5%増加した状態になっている。水位計測値が制御系に組み込まれていることと、非定位系（水位の真値はどのレベルでも安定した定常状態になる）であるために、複雑な挙動となっている。この 5 次元の観測データの 0-150 秒の区間を参照データ（訓練データ）とし、全 600 秒の観測データに AAKR を適用し、その残差を表示したのが、図 6.3-4 である。ドリフトの始まった 300 秒付近から水位データも含め、全部のデータにはっきりとした変化が観測さ

れている。水位信号に関しては、観測信号そのものよりも、残差の方がはっきりと変化をとらえていることが分かる。本来、水位信号だけに異常が見られるはずであるが、前述のように、観測値が制御系を通して他の信号に影響するため、全信号に異常兆候が見られる。この異常兆候の水位信号を例にして、SPRT で異常判定した結果が、図 6.3-5 である。ここでは、正常仮説は、0-150 秒のデータから求めた平均と分散による正規分布とし、異常仮説は、平均値 1%増加(H_1)、平均値 1%減少(H_2)、分散 2 倍増加(H_3)、分散 0.5 倍減少(H_4)という 4 通りを用いて監視した。判定閾値は、誤識別率 α =0.001、非識別率 β =0.01 としている。赤丸が異常判定したことを示しているが、t=350 秒付近から、平均値増加仮説と分散増加仮説の尤度比が閾値を越えて異常を示していることが分かる。

このように、AAKR と SPRT を組み合わせた手法は、化学プラントの異常監視に有効に働いていることが分かる。この手法は、米国の原子力発電所での監視に用いられている方法でもあり、今回の化学プラントのような対象には適した方法であるが、IT システムのような別の対象の異常監視には、図 6.3-2 に示したいくつかのアルゴリズムから適切なものを選ぶ必要がある。

もう一つの問題は、水位計のバイアスにもかかわらず、制御系のフィードバックを通してシステム全体に見かけ上の異常状態が波及しており、障害個所の切り分けが困難になっている点である。この点については、下記でもう少し説明をする。

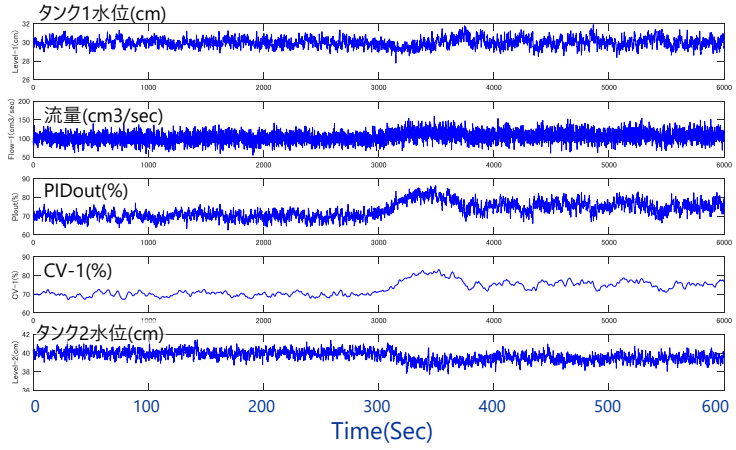

図 6.3-3　化学プラントシミュレーターによる水位計ドリフト模擬データ

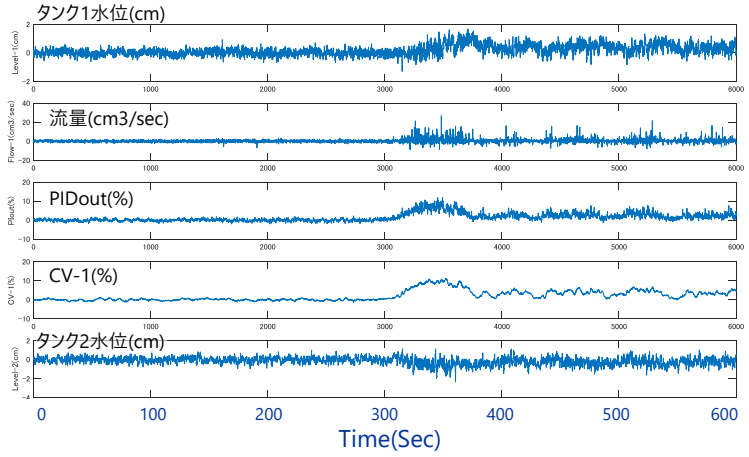

図 6.3-4　AAKR 法による残差の監視結果

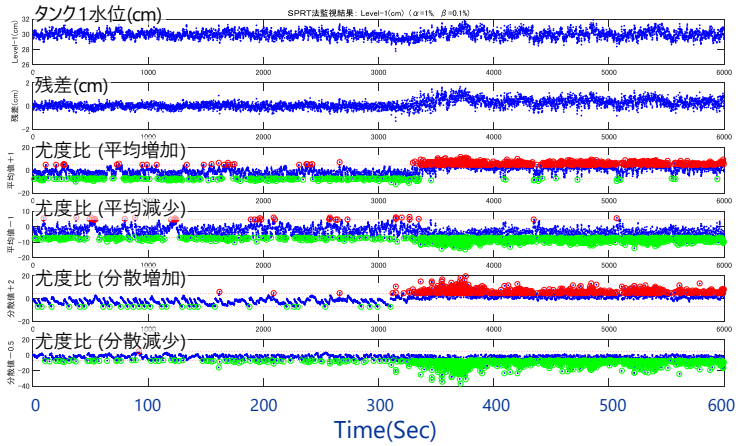

図 6.3-5　SPRT による水位信号残差の検定結果

多くのコンポーネントと観測信号からなるシステムに発生した障害の切り分けは、簡単ではなく、従来より多くの研究がなされている。その一つは、古くから人工知能の分野で研究されてきたモデルベース診断法である[Kutsuna2012]。図 6.3-6 に示すように、システム内の各コンポーネント（C1, C2, C3）の入出力関係が定義され、さらに、信号 a～f の正常・異常が論理的に識別できるものとしている。信号 d のように、非観測量で正常か異常かが不明の場合も許容される。このシステムでは、コンポーネントが正常かつ入力が正常ならば出力が正常という仮定をしており、これを System Description（SD）という形の論理式で表現している。これに観測情報（Observations, OBS）の論理式を併せて、各コンポーネントの正常・異常を推論する。図に示したように、C1～C3 の各コンポーネントが異常という仮説を検証すると、C1 異常の場合のみ真となり、他は棄却される。これが、モデルベース診断の原理であり、形式手法と同じく、仮定した世界の範囲では論理的に閉じた完全な診断法（証明法）である。

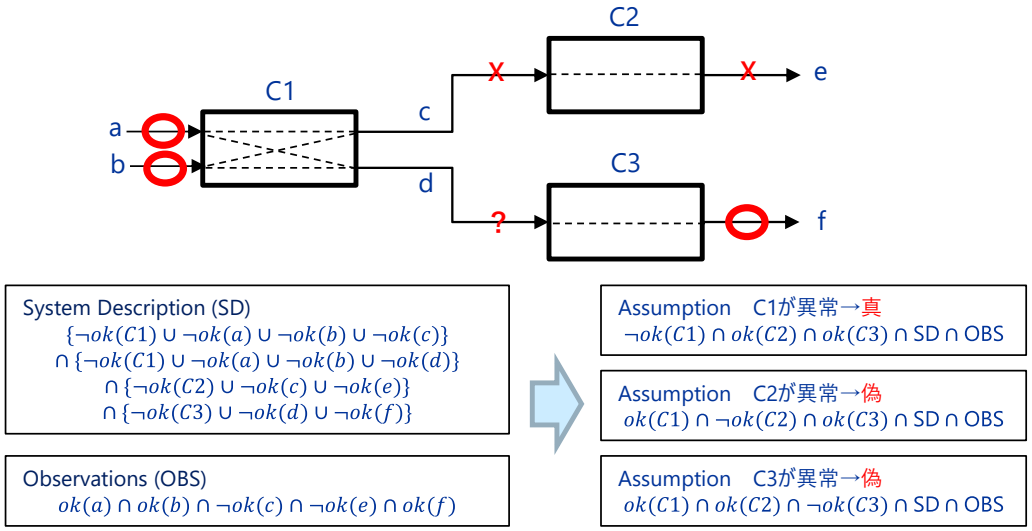

図 6.3-6　論理回路のモデルベース診断

一方で、多くの実際のシステムでは、コンポーネントの入出力関係が正確には分からなかったり、観測データの正常・異常の判断にも曖昧さが残ったりすることが多い。このような場合に、コンポーネントの入出力関係を条件付き確率（遷移確率）で定義し、観測量の正常と異常（真偽）も確率分布

で定義して、ベイジアンネットワークを適用することで、各コンポーネントの正常・異常の確率を評価できる。図 6.3-7 に、化学プラントシミュレーターを想定したベイジアンネットワークの例を示す。

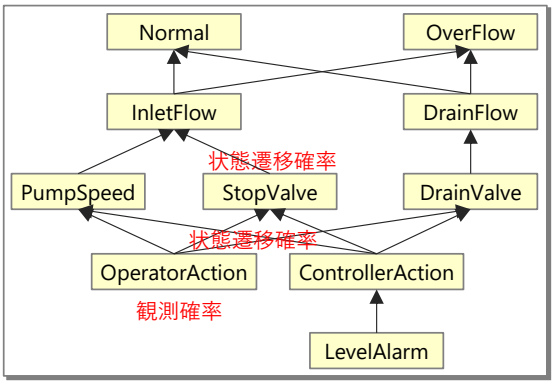

図 6.3-7　ベイジアンネットワーク

溢水が発生した場合を想定すると P(OverFlow)=1 となるが、その他の観測情報を入力することで、残りの変数の状態確率を評価することができる[本村 2002, HUGIN2015]。ベイジアンネットワークの特徴は、順方向の確率計算だけでなく、逆方向の確率も併せて評価でき、観測情報をすべて加味して残りの非観測量の状態が推定できるので、障害の原因として可能性の大きい機器を絞り込むことができる。

図 6.3-1 に示した一般的な異常診断システムでは、最後の段階で上記のモデルベース診断法、ベイジアンネットワークなどの方法や、場合によっては、過去の故障事例を参照して推論するエキスパートシステムなど適切な方法を集めて、真の原因を突き止め再発防止を図る必要がある。

6.4. 最近の形式手法技術による障害原因究明

モデル検査による方法ではモデルで表されたシステムの振る舞いが性質を満たすか否かの情報が得られる。満たされないときの反例は、なぜ満たされないかの情報を含み、原因究明の大きな情報源になると考えられる。ただし反例を原因究明に用いるときには注意点がある。

反例は性質が成り立たない動作列を与えるが、必ずしも人間にとって有用な情報を直接与えるとは限らない。化学プラントの例では不具合を引き起こす具体的な動作系列例を挙げているが、この系列は UPPAAL のモデル検査の反例としてはツールから得ることはできない。ツールで得られるのはこの系列の初期の短い系列だけである。この制約は、UPPAAL の採用している論理が状態に基づく CTL であることや、この例題で用いた検査式が特殊な形（ p → q すなわち A□(p imply A◇ q）という時相オペレーターがネストした特殊な形）の論理式であることなどから来ている。

ユーザーが一般的に故障原因究明に用いたい情報の一つとしては、モデル（すなわちシステムに）における誤り箇所の候補の特定を挙げることができる。

このような障害箇所候補の同定について、有界モデル検査技術を応用してスマートタブレットの電力異常消費問題の究明に適用した中島の研究結果[Nakajima2015]について簡単に紹介する。

中條らによるモデルベース診断法に基づくコンポーネント単位での故障診断法（[IPA2015]6.3 節）ではコンポーネントの入力・出力の関係を「コンポーネントが正常かつ入力が正常ならば出力は正常という関係」に抽象化し、これらの接続ネットワーク情報とともに観測されるデータの異常・正常のデータから故障を起こしているであろうコンポーネントを論理的に導出する。この手法と中島の方法はともに論理制約から故障箇所を同定するという点では共通しているが、同定する粒度は大きく異なるともいえる。

一般にシステムの振る舞いは状態遷移系としてモデル化できることはすでに述べたとおりである。このような状態遷移系はさらにその遷移関数などの形で形式化することができ、最終的には制約式の積からなるブール式として表現することが可能である。さらに有界モデル検査技術では初期状態からの最大遷移上限値 n を用いて、初期状態から n ステップまでで到達できる状態やそのための条件をブール式の多大な制約式の積として表現する。

このように状態遷移系 T に対し最大遷移上限値 n で到達可能な状態をブール式で表現した式を ϕ_T^nで表すことにする。さらに検査したい性質も同様にブール式で表現することができる。これを ϕ_P で表すことにする。

有界モデル検査では $\phi_T^n \rightarrow \phi_P$ という式を SAT ソルバーといわれる、ブール式の真偽判定ツールに与えることにより、ϕ_T^n の元で Φ_P が成り立つか（すなわち ϕ_T^n というモデル下での ϕ_P のモデル検査問題）という問題を解く。

今、ここで、$\phi_T^n \rightarrow \phi_P$ が偽であるということが得られたとする。

すなわち、状態遷移系 T では n ステップ以内の任意の系列で ϕ_P が成り立つとはいえない、ということが得られる。ここで、この反例を機械的にもとめ、その論理表現 ϕ_{CE}（反例の系列の論理表現）が得られたとする。

当然、この状況下では、$\phi_{CE} \wedge \phi_T \wedge \phi_P$ は偽であるが（ここで ϕ_T は ϕ_T^n から n ステップ制限をなくした論理式)、ここで ϕ_{CE} や ϕ_P を真とする真偽割り当ての中で、ϕ_T の中のどの複数の節が原因で式全体を偽としているのかということを問う問題を考えることができる。

一方 SAT ソルバー[SAT2009]については近年多くの研究がなされ、相当に大きなブール式を実用

上問題のない程度に高速に解くツールが複数実用化されている。

SAT ソルバーの変種として MSS（Maximal Satisfiable Subset）や MUS（Minimal Unsatisfiable Subset）を解くアルゴリズムやソルバーも近年研究されている。MSS では与えられた SAT 式が複数の節 Ci の積である（∧ Ci）と仮定し、与えられた式が論理的に満たされない場合にできるだけ多くの節 Ci を真にするブール変数割り当てを解く。

通常 MSS では与えられた節の積と、その各節を H（Hard 制約）、S（Soft 制約）の二つの集合に分類した入力に対して、H に属する節はすべて真にし、S に属する節をできるだけ多く真にする割り当ての組み合わせをすべて具体的に与える。例えば $\phi_C = a \wedge \neg a \wedge (\neg a \vee b) \wedge \neg b$ という論理式は偽である。この式を 4 つの節 $C_1 = \{a\},\ C_2 = \{\neg a\}, C_3 = \{\neg a, b\},\ C_4 = \{\neg b\}$ の積とする。このときすべての節を S に属するとして MSS を求めると $\{\{C_2, C_3, C_4\}, \{C_1, C_4\}, \{C_1, C_3\}\}$ が得られる。直感的には例えば元の式から節 $\{C_2, C_3, C_4\}$ だけを取り出せれば真にできるということを意味している。

MSS の各要素（である部分集合）の補集合を要素とする集合は MCS（Minimal Correction Subset）と呼ばれる。これは直感的には元の式を偽にする根本的な原因となる節の集合を意味する。先ほどの例では MCS は、$\{\{C_1\}, \{C_2, C_3\}, \{C_2, C_4\}\}$ となる。

元の問題で、$\phi_{CE} \wedge \phi_P$ を原則 H に属するとし、ϕ_T を S に属するとして MSS を解く。さらに MCS を得る。ここから何が本質的に悪さをしているか（すなわちどこがエラー箇所か）を見つけることができる。

中島の研究ではスマートタブレットの電源管理の状態遷移系を時間と累積消費電力量が計算できるオートマトンモデルに一旦モデル化した後、この有界区間の有界モデル検査の方法にのっとり、MSS 問題に帰着し、消費電力量の問題箇所を特定する道筋を示している。具体的には累積消費電力量が計算できるオートマトンモデルとして時間オートマトンを拡張し、時間経過とともに、状態（ロケーション）に割り当てられた重み（消費電力）で増加する電力量という変数を追加したハイブリッドオートマトンを考案し、それでスマートタブレットの電源管理機構をモデル化している。

このモデルではロケーションの滞在時間が長ければ長いほど電力量の値が（その重み付けである電力値に応じて）増加するようになっている。

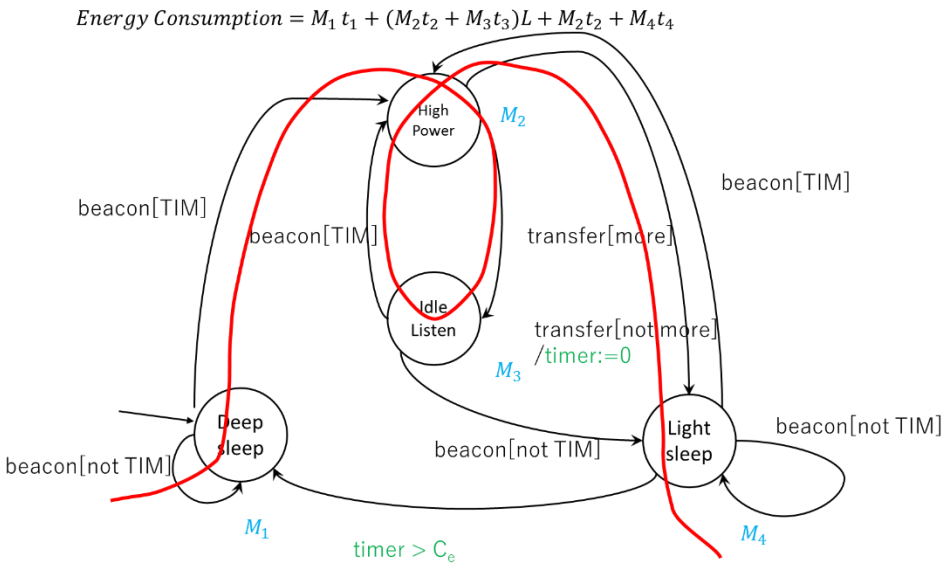

図 6.4-1　電力消費オートマトンの例（Wi-Fi Station のモデル化）

例えば図 6.4-1 の例では通常の時間オートマトンと同様にクロック変数 $timer$ についてリセット（ $timer \coloneqq 0$ ）や遷移のガード指定（ $timer > C_e$ ）などができるほか、各ロケーションに重み係数 $M_1, M_2, .., M_4$ が与えられており、例えばロケーション Deep sleep にコントロールが滞在している間経過時間が t_1であれば $M_1 t_1$の値だけ電力量変数の値が増加しているとみなす。

また、例えば赤の経路で実行され、かつ High Power と Idle Listen のロケーションを L 回繰り返した場合の総消費電力量（Energy Consumption）は各ロケーションの滞在時間 t_1, t_2, t_3, t_4を用いて図の上式で与えられる。

このモデルは Wi-Fi Station の動作をモデル化しており、消費電力量削減のための退避状態として Deep sleep と Light sleep が使い分けられ、beacon 信号をあわすアクションと TIM（Traffic Indication Message）の条件の組み合わせによりこれらの状態間の遷移がモデル化されている。

[Sorea2002]では時間オートマトンの各遷移動作を SAT にエンコードする方法が具体的に与えられており、この方法を上記の電力消費オートマトンに拡張することにより中島は SAT への具体的なエンコード方法を与えている。

ここで時間オートマトンにおけるガード式やリセット、不変条件などプログラムのよくある誤り箇所に相当する部分をエンコードした式を S に属するとし、標準的な構造のエンコード式は H に属するとして MSS に問題を帰着している。これにより、消費電力量について問題のある制御の条件などの同定を行っている。

中島の研究の例題では、たとえば電力量消費を抑える待機状態に遷移する遷移のガード式の時間閾値を少し小さくする（すなわち早く待機状態に遷移する）と異常電力量消費のバグが解決できることを示唆している。

先に述べた中條の研究のように、コンポーネントの外部インターフェースの情報のみを用いてシステムをモデル化できる場合は、システムのスケーラビリティの点では実用上で有利である。ただし、時間情報や例外などを扱うと外部インターフェースの異常・正常だけでのモデル化は表現上の限界もある。一方、中島のアプローチは時間を扱うモデルや技法を複数駆使しながら（適切な段階で適切なモデルとアプローチをとりながら）問題解決を図っている点で興味深い。実用上の課題としては大規模システムへのスケーラビリティへの対応が考えられる。

複雑なネットワークやマルチタスクシステム間の非同期の通信があるシステムの診断には、モデルベース診断法に基づくコンポーネント単位での故障診断法でコンポーネント単位の同定は有効であると思われる。その後コンポーネント内部での粒度の細かな故障診断には中島の方法は有効と思われる。

7. 報告書と提言

本章では、報告書作成にあたっての諸注意について述べる。内容は前年度から大きな違いはないが、本年度の議論を通して、若干の語句訂正と追記を行った。

7.1. 報告書の作成

障害原因を調査、検証した内容を報告書としてまとめておくことは、以後の障害原因調査の参考となるだけでなく、障害の未然防止にも有効である。そのためには、報告書を客観的で比較評価可能な形にまとめておく必要がある。

報告書をどのようにまとめるか、誰が書くのか、公表範囲をどうするのかなど、事態発生後に方針が定まらず混乱することのない様、予め手順書や様式などを定めておくことが望ましい。

報告書はいろいろな立場の人が読むことになるが、誰宛の報告書なのかは、重要な要素になる。原則的には、調査組織を設置した責任者宛である。

この責任者宛の報告書を基本報告書とするなら、それを元に、顧客や被害者向け、あるいはマスコミや一般市民向けなどに要約版や専門用語、業界用語を排したわかり易い報告書を派生的に作成する必要もある。また、他の開発チームが事実に基づく再発防止策を検討できるよう、より技術的で具体的な報告書も必要になるだろう。

7.2. 報告書の構成

調査組織設置責任者向けの調査報告書は、一般的に以下の項目が必要となる。

1) 調査組織構成員の氏名と連絡先、専門範囲
2) 調査の経過、各種情報収集の日時と場所、情報提供者とそのミッション
3) 障害の概要、詳細経緯（障害発生直前の状況、障害の推移、対処経緯）
4) 被害状況、顧客や社会への影響、報道内容、システム復旧状況
5) 障害原因の分析、検証の前提と検証方法、検証プロセス、検証結果
6) 障害に対する処置内容、当該システムの再発防止策と水平展開の範囲
7) 組織としての再発防止策、類発防止策の提案

なお、必要に応じて以下の調査記録を詳細資料または添付資料としてまとめておく。

1) 障害発生時の通知、覚知記録
2) システム基礎情報
   - システムの開発と運用開始時期、設置場所、システムの用途
   - システムの構成と設計図書、構成機器の仕様諸元、電源仕様
   - ソフトウェアの構成・コード等の設計資料、購入、調達の内容
   - ネット接続仕様、セキュリティ対策
   - システムに関係する組織体制、利用者の概要
   - システムの試験や移行・試運用に関する記録
3) 現場調査記録（エラーログ、システム状態記録など）
4) 関係者ヒアリング記録
5) 第 4 章〜第 6 章にある解析、検証の結果
6) 保守、改造・更新、トラブルのこれまでの履歴

7） 障害予防活動、防災活動の記録
8） 過去の類似障害事例
9） 関連法規、関連規格
10）関連の可能性を否定できない障害発生前のイベント、自然現象、社会現象

7.3. 報告書記載時の注意

記載は、事実に基づいて、時系列に正確に客観性を保って進めなければならない。

調査においては、すべての事実を収集できるわけではないので、事実と推定は用語もしっかり使い分けることが望ましい。

国土交通省の昇降機事故調査報告書では、次のような用語の使い分けを行っている。

① 断定できる場合　：　「・・・認められる」
② 断定できないが、ほぼ間違いない場合：　「・・・推定される」
③ 可能性が高い場合　：　「・・・考えられる」
④ 可能性がある場合　：　「・・・可能性が考えられる」
　　「・・・可能性があると考えられる」

また、火災原因調査の例では、調査の結果を、「断定」、「判定」、「推定」、「不明」　に区分して報告するよう定めている。

① 断定： 信頼性の多い各資料を総合することにより全く疑う余地なく、その原因が具体的かつ科学的に確定され、何らの推理を必要としない場合
② 判定： 信頼性の多い各資料を総合することにより、具体的かつ科学的にその原因を断定するまでには至らないが、多少の推理を加えることにより疑う余地を残さない場合
③ 推定： 信頼性のある資料によっては直接判定できないが、推理すれば合理的にその原因が推測できる場合
④ 不明： 原因決定の基礎となる資料がなく、または若干の資料はあっても信頼性が少なく、かつ多少の推理を加えても合理的にその原因を推定できない場合

製品・制御システムに関する障害原因分析の報告書においても、上記のような用語の使い分けを行っていくことが望ましく、報告書の冒頭で定義しておくとよい。

その他の注意点として以下があげられる。

1） 特殊用語、専門用語、略語は必ず補足説明をつける。同じ用語でも業界や技術分野により意味が異なる場合があるので注意が必要である
2） 用語、文体について、統一、一貫性を保つ
3） 意見が分かれたり、複数見解が発生したりという場合、すべて併記する
4） 事実関係の整理ができていない状態で、偏った思い込みに走らないよう、あるいは不用意に過失を批判したりすることのないように心がける
5） 個人情報、営業秘密、システムのノウハウ等が漏洩することないよう配慮する
6） 当該システムの「処置（対策）」と「再発防止策」を明確に分ける
7） 特に処置は、緊急対策、仮処置、本処置など、場合により使い分ける
8） 障害を発生したシステム以外で同様の問題がないか、類発防止策も提言する

9） 関与組織全体の再発防止策も意見として示す

10）顧客、被害者の心証を傷つけることのないよう、またマスコミに誤解を与えることのないよう文章表現については複数人でチェックする

7.4. 報告書の利用と提言について

報告書はこれまで何度か触れてきたように事実に基づき客観的に原因に迫るべきである。背景要因まで充分な分析に至らない状態のまま、性急に再発防止策や教訓を得ようとすると事実誤認や客観性を失う恐れがあり注意が必要である。

障害から教訓を引き出そうとする場合は、調査報告書を査読した者に委ねることが望ましい。習得する教訓は、その人やチーム毎に、経験と組織内の役割分担により、深さ的にも方向的にも大きな違いが出てくる。従って調査報告書の作成者は改善のための意見を提示しても直接的に教訓を提示することのないように気をつけたい。

一つの事例から得られる教訓は一面的なものになりやすい。また、一人が得る教訓は、しばしばその個人の経験や知識によりバイアスがかかりやすい。従って、共有されるべき教訓というものは、複数の障害報告や成功事例を丹念に分析して総合的に得られる抽象化された知識体系となるべきである。このような観点からも可能な限り報告書は、複数の人が世代を超えて相互に比較検証を実施することができて、分析的なものを目指したい。

原因調査の過程でコンプライアンスに触れる問題も見つかる可能性があるが、コンプライアンス問題等、組織と人の責任のあり方の問題については、システム障害原因診断における技術的な原因分析とは切り離して対処することが望ましい。

また、外部発表にあたっては、メディア・バイアスの存在にも注意を向ける必要がある。説明の仕方や語句の使い方が、内部の者や専門家には、妥当な表現であっても、一般には否定的に見られることがある。さらにメディア自体が、背景知識や報道時間の制限から事実を歪めた報道をしてしまうこともある。広報には細心の注意と事前の準備も必要になる。

8. 教育への反映

事後 V&V の体系に沿って調査した障害事例の状況や原因究明などの結果には、エンジニアのための教育コンテンツとして有用な事項が多く含まれていると考えられる。また、その結果を導き出すために用いたツールは、具体的な適用例をもつ演習教材としても利用できる可能性がある。これらを教育コンテンツとしてまとめ、教育や訓練に利用することが大事である。また、障害原因究明の分析結果はデータベースとして蓄積・整理することで、教育コンテンツとともに広く関連エンジニアへの普及に役立てることができる。

ここで、教育コンテンツを利用する対象者は二つに分けられる。一つは事後 V&V 活動に携わる第三者組織のエンジニアであり、もう一つは障害を起こしたシステムの開発組織の管理者やエンジニアである。それぞれの立場に応じて教育コンテンツは異なると考えるべきである。

第三者組織に関わるエンジニアは、まさに事後 V&V の方法論として、初動調査からシステムの要求仕様や機能のモデリングに関する方法論や原因診断に関わるいろいろな形式・半形式手法を学ぶ必要がある。同時に、対象とするシステムのドメイン知識もある程度の深さで学ばねばならない。起こった障害に応じて問題解決に必要な知識は異なるので、これらのすべてを学ぶことは困難であるが、例えば、この知識を分類し、その分野ごとにスキルの習熟度をスキル標準のレベル（[IPA2012]）によって定義することで、事後 V&V に携わる分析者の能力を客観化できる。これにより障害原因診断の際に必要な人材をバランスよく集めることができる。

必要な知識をどのような分野として整理していくかは今後の課題であるが、事後 V&V の要素技術だけでなくシステムズエンジニアリングや障害診断にかかわる一般的知識として、例えば、プロジェクト管理の方法論や組織としての技術標準化､安全文化の醸成なども必要になってくると考えられる。

一方、障害を起こしたシステムの開発組織の管理者やエンジニアに対する教育は少し異なったものになる。障害を再び起こさないための改善などの提言は、事後 V&V に含まれるが開発に関わった組織そのものの診断も重要である。

一般に、安全に重大な影響を持ちうるシステムに開発には組織の安全文化が大事であるが、これは放っておくと劣化する。特に、障害発生後の再発防止策として、組織内の一部の有識者や外部のコンサルタント、または特定の規格の認証機関の意見のみを鵜呑みにして組織標準プロセスを構築または改善をすると、現場においては、定義されたプロセスの意味を考えなくなり、やがて不遵守が多くなるという安全文化の劣化を招くことが多い。

プロセスに従う現場の管理者やエンジニア一人一人が、プロセスの意味やその効果、さらには不遵守になったときのリスクを考え、プロセスのさらなる改善と個人のスキルアップを目指すことが重要である。

また、障害を起こすシステムの開発では、エンジニア個人、プロジェクト、組織のどこかの段階で何らかの欠陥があることが多いということも現実である。障害を発生させないシステムを開発するためには､先に述べた安全文化を含めた組織としての PDCA だけではなく､開発を行う個々のプロジェクトにおける PDCA、そして管理者やエンジニア個人での PDCA が必要であり、いずれかが欠けても障害の再発防止は保証できないと言えよう。

日本の文化として、個人レベルでの工夫や改善としての PDCA は比較的高い水準にあると言われることもある。しかしながら、個人でモノづくりのすべてを担う職人的な開発ではなくある程度の開発規模になると、プロジェクトとしてのマネジメント、カイゼンが必要となる。ここで、片仮名でカ

イゼンと書いたがプロジェクトレベルにおける PDCA は、日本の改善の文化が輸出され、体系立ったフレームワークとして逆輸入されたカイゼンに近い考え方とも言えよう。さらに、組織としての PDCA という観点では、まだまだ仕組みが確立されていない組織も多く、改善が外国語的な KAIZEN として捉えられてしまっていることもある。

このような、組織としての再発防止や改善、教育を考えていく上で、組織のプロセスを評価しておくことも重要となる。製品事故やシステム障害に至る要因は、特定の技術的要素の誤りがきっかけとなることが多いが、その誤りを発生させるプロセス的な要因を取り除かない限り製品事故やシステム障害の再発を回避することができない。そのため、障害原因分析を実施し改善を行ったプロジェクト及び組織に対しては、その経過を評価し、再発防止の状況を確認する必要がある。

ここでは、開発プロジェクト及び組織における障害発生要因の診断手法として FMEA を用いて分析を行う方法を提案する（詳細は[IPA2015]参照）。設計またはその結果に対して適用される設計 FMEA（DFMEA）と開発プロセスや生産工程に対して適用されるプロセス FMEA（PFMEA）に大別される。

DFMEA は、システムを構成するコンポーネントやアイテムに発生し得る故障状態（故障モード）を列挙し、その影響と原因を分析して対策を検討する分析手法である。一方、PFMEA は従来、生産工程におけるミスや不遵守の影響と原因を分析して対策を検討する手法として適用されてきたが、ここでは特にソフトウェアの開発プロセスに適用し、プロセスが適切に実施されなかったり不遵守となるケースを列挙し、その影響と想定される原因を分析することで対策を検討する。DFMEA や PFMEA を用いることで、プロセスの欠陥や改善点を網羅的に抽出することが可能となる。

障害を発生させた組織に限らず障害の再発防止や未然防止を進められる組織においては、このような手法を用いてプロセスに潜む潜在的な障害発生のリスクを評価し、そのリスクを回避・抑制するための組織、プロジェクト、個人の PDCA が望まれる。

おわりに

事後 V&V の全体像と、その中で用いられる要素技術を紹介した。コンピューターを組み込んだ高度な制御、人間との協調制御に加えて、インターネットを介した制御などますます大規模・複雑化する製品・制御システムが増えてゆく中で、その安全設計やトラブル対応には、既存の技術で対応しきれなくなってきており、システム理論に基づく事故モデル STAMP のような考え方が提唱されている。このようなシステムにおいて障害原因究明が必要とされた場合には、その障害の状況に応じて、既存の技術だけでなく STAMP のような最新の技術を、迅速かつ適切に組み合わせて解決していくことが必要になる。

本稿では、化学プラントシミュレーターを仮想の対象として、様々の障害原因究明にかかわる要素技術の適用方法を検討した。要求仕様や機能仕様の SysML を用いた記述法、SysML と STAMP を組み合わせた複雑システムのハザード分析、機械学習・人工知能技術を用いた障害診断法、形式手法（モデル検査）を用いた人間・機械の協調制御アルゴリズムの検証、などである。このように具体例でその使い方をショーケースのように可視化しておくことは、いざというときのための準備として必要不可欠なことであろう。トヨタプリウスの電子制御トラブルでの NASA のアプローチのように、実装コードの解析・ロジックの解析・モデルベースのテスト・実時間性の解析など、あらゆる既存のツールを活用して解析された例もある。結果的に、ソフトウェアには問題が見つからなかったことが示されたが、このような対応能力を国内で持っておくことは極めて大切である。

本年度の活動では、要素技術の検証のためのサンプルシステムとして、化学プラントシミュレーターに加えて、倒立二輪車の自立制御と人間との協調制御システムを作成した。現実の世界の障害を直接扱うことは必ずしも容易ではないことから、今後も、これらのサンプルシステムを用いて、障害原因究明のためのツールの準備とその利用方法の蓄積を行ってゆく予定である。これは、また、障害診断に関わるエンジニアの育成にも大きく寄与できると考えられる。さらには、この中で、既存の枠を越えた新しい技術やツールが生まれれば幸いである。

謝辞

本ワーキンググループにてご講演を賜り、活動推進にご支援、ご指導をいただいた下記のみなさまに心より感謝の意を表します。

2015 年度ご講演

・株式会社日立製作所 研究開発グループ 主任研究員 松原 正裕 氏
「自動車制御ソフトウェアの形式検証技術」
・株式会社ニルソフトウェア 代表取締役 CEO 伊藤 昌夫 氏
「CARDION：概念段階におけるハザード・脅威の抽出手法」
・株式会社日立製作所 研究開発グループ 主任研究員 小川 秀人 氏
「モデル検査を用いたフォールトアナリシス手法の提案」
・国立情報学研究所 教授 中島 震 氏
「モデル検査法の適用経験」
・東北大学 教授 / 技術研究組合 制御システムセキュリティセンター（CSSC）多賀城本部長
高橋 信 氏
「サイバーセキュリティとヒューマンファクタ、レジリエンスエンジニアリングについて」

（講演順）

2014 年度ご講演

・有人宇宙システム株式会社（JAMMS） 主幹技師 星野 伸行 氏
「安全手法 STPA 及び不具合分析手法 CAST の概要と事例紹介」
・明治大学 名誉教授 向殿 政男 氏
「事故調査委員会の活動内容とその調査事例について」
・九州大学 准教授 日下部 茂氏
「ハザード分析手法 STAMP/STPA を用いた障害原因分析の体系化（事後 V&V）に向けて」
・株式会社フォーマルテック 代表取締役 早水 公二 氏
「モデル検査の事例・動向に関する解説」
・北陸先端科学技術大学院大学 准教授 青木 利晃 氏
「形式手法の産業応用」
・愛知工業大学 教授 中條 直也 氏
「車載電子システムのログデータ収集と故障診断」

（講演順）

2016 年 3 月
独立行政法人情報処理推進機構
技術本部ソフトウェア高信頼化センター
ソフトウェア高信頼化推進委員会
障害原因診断 WG

用語説明

本書をとおして使用されている重要な用語について、JIS や IEEE などの標準規格や業界標準に沿って説明します。但し、これらの用語に「正解」はなく、「本書ではこういう意味で使用している」という約束事を示すものです。

【製品・制御システム】

- 様々な分野の製品に組み込まれて動作するマイクロプロセッサーを利用したソフトウェアを含む電気・電子制御製品、システム
- それらの製品や機器を物理的、論理的につなげることにより社会生活を支える基盤となるシステム

【故障（failure)】

- 要求された機能を遂行する製品や機能単位の能力が尽きる状態、無くなること
- システムまたはシステムのコンポーネントが指定された制限内で要求された機能を果たさない事象

【フォールト（fault)】

- 要求された機能を機能単位が遂行できなくなる偶発的条件。ソフトウェアに対しては、計算機プログラム内の不正確なステップ、プロセスまたはデータの定義

「障害」という用語は JIS において「fault」に対応する訳語とされているが、「fault」が原因で引き起こされた「故障（failure)」の結果、系がサービスを提供できない現象を「障害」として指すことも多い。本書では、概ね後者の意味で「システム障害」などとして「障害」を用い、「fault」に対しては「フォールト」を用いている。

【ハザード】

- 潜在危険。潜在的に危険の原因となりうるもの。システムが誤動作などをして利用者や関係する社会などに危害や損害をもたらす場合に、それらの誤動作を引き起こす要因となるもの

【リスク】

- ハザードによって引き起こされる、ある（有害な）事象生起の確からしさと、それによる負の結果の組合せ

【原因・要因】

- ある事象を引き起こした事柄としてほぼ同じ意味で用いられることも多いが、「原因」は問題や現象を引き起こしたものとして特定される事柄。「要因」は事柄やその原因に影響を及ぼす（可能性のある）事柄

【妥当性確認（validation)】

- 定められた用途に対する特有の要求事項が満たされていることを、客観的証拠の審査及び提出によって確認すること

【検証（verification)】

- 規定要求事項が満たされていることを、客観的証拠の審査及び提出によって確認すること

参考文献

[Abrial2010] J.-R. Abrial, “Modeling in Event-B: System and Software Engineering”, Cambridge University Press 2010.

[Björkman2009] K. Björkman et al., “Verification of Safety Logic Designs by Model Checking”, 6th American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT), 2009.

[Clarke1999] E. M. Clarke et al., “Model Checking”, MIT Press, 1999.

[Clarke2001] E. Clarke et al., “Progress on the State Explosion Problem in Model Checking”, LNCS Vol. 2000, pp. 176-194, 2001.

[Delange2014] J. Delange and P. Feiler, “Architecture Fault Modeling with the AADL Error-Model Annex”, 40th EUROMICRO Conference on Software Engineering and Advanced Applications, pp.361-368, 2014.

[フリーデンタール2012] サンフォード・フリーデンタールほか著, 西村秀和監訳: “システムズモデリング言語 SysML”, 東京電機大学出版局, 2012.

[Gofuku2008] A. Gofuku and A. Ohara, “Fault tree analysis of chemical plants based on multi-level flow modeling”, Toward Innovative Nuclear safety and Simulation Technology (ISSNP 2008), vol.2, pp.11-17, 2008.

[Hines2008] J. W. Hines et al., “Technical Review of On-Line Monitoring Techniques for Performance Assessment”, NUREG/CR-6895, vol.2, 2008.

[HUGIN2015] http://www.hugin.com/productsservices/demo/hugin-lite

[井手 2015a] 井手剛, 杉山将, “異常検知と変化検知”, 講談社, 2015.

[井手 2015b] 井手剛, “入門機械学習による異常検知”, コロナ社, 2015.

[IPA2010] 独立行政法人情報処理推進機構 ソフトウエア・エンジニアリング・センター, “形式手法適用調査報告書”, http://www.ipa.go.jp/sec/softwareengineering/reports/20100729.html, 2010.

[IPA2012] 独立行政法人情報処理推進機構 IT スキル標準センター, “IT スキル標準はやわかり”, http://www.ipa.go.jp/files/000025745.pdf, 2012.

[IPA2013] 独立行政法人情報処理推進機構 ソフトウエア・エンジニアリング・センター, “2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書”, http://www.ipa.go.jp/files/000026799.pdf, 2013.

[IPA2014] 独立行政法人情報処理推進機構 ソフトウェア高信頼化センター, “情報処理システム高信頼化教訓集（IT サービス編）障害対策手法・事例集、障害分析手法・事例集 2013 年度版”, 2014.

[IPA2015] 独立行政法人情報処理推進機構 ソフトウェア高信頼化センター ソフトウェア高信頼化推進委員会 障害原因診断 WG, “大規模 複雑化した組込みシステムのための障害診断手法 ~モデルベースアプローチによる事後 V&V の提案～”, https://www.ipa.go.jp/sec/reports/20150331_2.html, 2015.

[IPA2016] 独立行政法人情報処理推進機構 ソフトウェア高信頼化センターソフトウェア 高信頼化推進委員会 システム安全性解析手法 WG, “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～”, 2016.

[Jackson2011] D. Jackson, “Software Abstractions, revised edition Logic, Language, and Analysis”, MIT Press, 2012.

[JASA2010] 組込みシステム技術協会, “組込み系技術者のための安全設計入門”, 電波新聞社, 2010.

[Jiang2007] G. Jiang, et al., “Efficient and Scalable Algorithms for Inferring Likely Invariants in Distributed Systems”, IEEE Transactions on Knowledge and Data Engineering, 19(11), pp.1508-1523, 2007.

[Kutsuna2012] T. Kutsuna, et al., “Efficient Root Cause Detection in Complex Embedded Systems with Abstract Model-based Diagnosis”, Journal of Information Processing, 20(3), pp.767-773, 2012.

[レブソン2009] N. G. レブソン著, 松原友夫ほか訳, “セーフウェア ～安全・安心なシステムとソフトウェアを目指して”, 翔泳社, 2009. （原著：N. G. Leveson, “Safeware: System Safety and Computers”, Addison-Wesley, 1995）

[Leveson2012] N. G. Leveson, “Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems)”, The MIT Press, 2012.

[Leveson2015] N. G. Leveson, Private communication, August, 2015.

[本村 2002] 本村陽一, “ベイジアンネットワークソフトウェア”, 人工知能学会誌, 17(5), pp.559–565, 2002.

[中島 2012] 中島 震, “形式手法入門 ロジックによるソフトウェア設計”, オーム社, 2012.

[Nakajima2015] S. Nakajima and S.-M. Lamraoui: “Fault Localization of Energy Consumption Behavior using Maximum Satisfiability”, Proc. 5th International Workshop on Design, Modeling and Evaluation of Cyber Physical Systems, pp.99-115, 2015.

[NEC2015] http://jpn.nec.com/websam/invariantanalyzer/

[NuSMV] A. Cimatti, et al., “NuSMV 2: An openSource tool for symbolic model checking”, In Proceeding of International Conference on Computer-Aided Verification (CAV 2002), 2002.

[PRISM] M. Kwiatkowska, et al., “PRISM: Probabilistic Symbolic Model Checker, “Computer Performance Evaluation: Modelling Techniques and Tools Volume 2324 of the series Lecture Notes in Computer Science pp 200-204 2002.

[SAT2009] A. Biere et al. eds., “Handbook of Satisfiability”, IOS Press, 2009.

[Sorea2002] M. Sorea, “Bounded Model Checking for Timed Automata”, 3rd Workshop on Models for Time-Critical Systems (MTCS 2002), ENTCS, 68(5), pp.116–134, 2002.

[SPIN] G. J. Holzmann, “The SPIN Model Checker: Primer and Reference Manual”, Addison-Wesley, 2004.

[刀川 2011] 刀川眞, “我が国の社会的特性に着目した組込みシステム開発の方向性 ―エレクトロニクス化された耐久消費財におけるソフトウェア開発の強化策―”, 科学技術動向 2011 年 9・10 月号, pp.12-22, 科学技術政策研究所 科学技術動向研究センター, 2011.

[Thomas2012] J. Thomas et al., “Evaluating the Safety of Digital Instrumentation and Control Systems in Nuclear Power Plants”, MIT Research Report: NRC-HQ-11-6-04-0060, 2002.

[UPPAAL] G. Behrmann, et al., “A Tutorial on Uppaal”, LNCS Vol. 3185, pp 200-236, 2004.

[Wang2013] J. X. Wang and M. L. Roush（日本技術士会訳）, “リスク分析工学”, 丸善（株）, 2003.

[渡邉 2014] 渡邉将也 他, “最新の機械学習法を用いた回転機の異常監視手法の提案”, 日本保全学会第 11 回学術講演会, pp.453-457, 2014.

[Wegerich2004] S. W. Wegerich, “Similarity Based Modeling of Time Synchronous Averaged Vibration Signals for Machinery Health Monitoring”, IEEE Aerospace Conference Proceedings, 2004.

[XSTAMPP] XSTAMPP (An eXtensible STAMP Platform), http://www.iste.uni-stuttgart.de/se/forschung/werkzeuge/xstampp.html

[山西 2009] 山西健司, “データマイニングによる異常検知”, 共立出版, 2009.

編著者（敬称略）

障害原因診断 WG（主査、副主査以下 50 音順）

主査	兼本　茂	公立大学法人会津大学
副主査	金田 光範	地方独立行政法人東京都立産業技術研究センター
	青木 利晃	国立大学法人北陸先端科学技術大学院大学
	漆原 憲博	株式会社ジェーエフピー
	大原　衛	地方独立行政法人東京都立産業技術研究センター
	岡野 浩三	国立大学法人信州大学
	岡本 圭史	仙台高等専門学校
	北道 淳司	公立大学法人会津大学
	髙村 博紀	株式会社アトリエ
	田渕 一成	ビジネスキューブ・アンド・パートナーズ株式会社
	中村　洋	株式会社レンタコーチ

IPA/SEC（50 音順）

石井 正悟
石田　茂
佐々木 千春
玉置 哲男
十山 圭介
松田 充弘
三原 幸博