DS-671.1 デジタル社会推進実践ガイドブック

ユーザビリティ導入ガイドブック

Introduction to Usability

2026 年（令和 8 年）6 月 1 日
技術検討会議サービスデザインタスクフォース

Informative

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

1

概要

「DS-671.1 ユーザビリティ導入ガイドブック」（デジタル庁）は、デジタル社会推進標準ガイドラインのサービスデザイン関連文書です。利用者中心の視点に立ち、誤操作を起こしにくく安全で、利用ニーズに合致した使いやすい情報システム及びユーザーインタフェース（User Interface: UI）を提供するための基本的な原則や手法について紹介しています。

行政官だけでなく、デザイナー、エンジニア等、異なるドメイン知識を持つ方々に利用いただけるように、あえて標準ガイドラインの用語を用いずに平易な表現を用いる、文末表現を常体ではなく敬体で構成する、重要な概念にはイラストをつける等、理解しやすいよう工夫しました。

デジタル庁の業務を通じて得られた知見やノウハウをもとに、行政機関の方にとってサポートとなる内容を多く記載しています。様々なプロジェクトで活用いただけることを願っています。

1.1 標準ガイドラインにおける本文書の位置づけ

本ガイドブックは、各府省が情報システムを介した行政情報及び手続等の機能を、利用者にとってより使いやすく安全に、活用しやすい形で提供できるよう見直す際に、「DS-670.1 ユーザビリティガイドライン」（デジタル庁）とあわせて参考にすることができます。

1.2 背景と課題

情報システムに関連したインシデントの多くに、人の操作ミスや、理解・操作が困難な難解なインタフェース及びコンテンツ等のユーザビリティに関する考慮不足が関わっています。人に起因した事故をなるべく引き起こさない・事故の影響を低減させるユーザビリティに対する取り組みは、機器やサービスの安全な利用に欠かせない活動です。ユーザビリティ及びアクセシビリティの確保は、以下のような側面でメリットがあります。

- 必要な情報の適切な伝達や理解
- 申請・手続の利用の促進
- 機会損失、経済的損失や環境資源損失の回避
- 心身・生命等の安全の保障
- 障害者、高齢者等の様々な特性のある利用者への利用機会の保証

なお昨今、政策の検討・推進が、単独の組織が提供する独立した情報システムによって完遂できることは少なくなってきているため、どのような機能をどのようなレベルで、誰に向けてどのような形（範囲）で実現するべきか、見通しにくくなっています。そこで本ガイドブックでは、ユーザビリティを高めるために導入しておくと良い事項や基本的な取り組み方を紹介しています。

1.3 対象読者

本ガイドブックは、以下の読者を対象にしています。

- 使いやすい情報システムを実現したい行政の担当者
- ユーザビリティの評価を行いたい方
- 操作ミス等を予防し、被害を軽減するための手法を知りたい方
- ユーザビリティに関連するリスクアセスメントの手法や考え方を知りたい方
- システム思考に基づいた安全設計の手法や考え方を知りたい方

1.4 関連文書

各府省が参照すべきユーザビリティのガイドラインを探している方は、「DS-670.1 ユーザビリティガイドライン」（デジタル庁）を参照してください。

1.4.1 導入のための参考になる文書

ユーザビリティやアクセシビリティを確保した情報システムの導入を目指す方向けには、以下のガイドブックも用意されています。

- 「DS-671.2 ウェブアクセシビリティ導入ガイドブック」（デジタル庁）

なお、ウェブアクセシビリティについては、行政機関は「みんなの公共サイト運用ガイドライン（2024年版）」（総務省）を参照する必要があり、標準ガイドライン群におけるウェブアクセシビリティに関する資料も、当該ガイドラインの内容に即した記述になっています。

1.4.2 利用の手引き

DS-600番台のサービスデザイン関連文書は、目的に応じて体系化し、負担が少ない形で利用できるように整理されています。以下の手引きを参考にしてください。

- 「サービスデザイン関連ガイドラインの読み進め方・資料の探し方」（デジタル庁）

1.5 記述範囲と構成

本ガイドブックは、ユーザビリティの概念、ユーザビリティに配慮した設計、ユーザビリティの評価を実施すること、及びリスクアセスメントに関するガイドブックとして以下の内容で構成されています。

- 用語や概念の紹介（第2章）
- 使いやすい情報システムを設計するために実施すべき活動（第3章・第4章）
- 使いやすさの評価（第5章）
- 使用エラー対策や分析の実施、安全設計の考え方（第6章～第8章）
- システム思考に基づく安全設計の考え方（第9章）
- リスク情報の活用と安全文化の醸成（第10章）

1.6 利用と配布

1.7 改訂履歴

2026年6月1日　誤字脱字・誤記等の修正及び参考文献の記載
2025年9月30日　初版発行・デジタル社会推進標準ガイドライン編入

2

基本的な概念を理解する

2.1 ユーザビリティとはなにか

ユーザビリティは直訳すると「使える状態」という意味ですが、専門的には「特定のユーザーが特定の目的を達成するために、システムや製品をどれだけ効果的、効率的に、そして満足して使用できるか」（注記）を指します。

産業革命以降、機械を人間が扱えるようにするために様々な取り組みが行われてきました。機械がシンプルだった頃は、トレーニングやマニュアルを用意することで、十分に対応できました。しかし、機械が飛行機のように複雑になってくると、人を訓練するだけでは対応が難しくなってきました。機械が扱いやすくないと、人の対応能力を超えてしまうためです。

図2.1 シンプルな機械から複雑な機械まで。

原子力発電所のシステムでユーザビリティが考慮されていなかったために、重大事故につながった事例もあります。こうした「ヒューマンファクター（人的要因）」に対処するため、ユーザビリティの概念や評価・改善手法が開発されてきました。

ユーザビリティが確保されている＝使いやすいことは、単に使いやすいだけではなく、サービスの導入率向上や、サポートコストの低減、サービス提供者への信頼感の醸成等、様々なメリットを提供してくれます。このことに感覚的に異論がある人はまずいないでしょう。

誰だって、難解で分かりにくく、非効率的で、すぐによく分からないエラーを通知してくるシステムを操作したくはありません。ECサイトの問い合わせ電話番号を探したいだけなのに、チャットボットとQ&Aの間をたらい回しにされて1日を費やしたくないのです。

注記）JIS Z 8521:2020 人間工学―人とシステムとのインタラクション―ユーザビリティの定義及び概念（ISO 9241-11:2018の一部変更規格）

2.1.1 ユーザビリティの観点

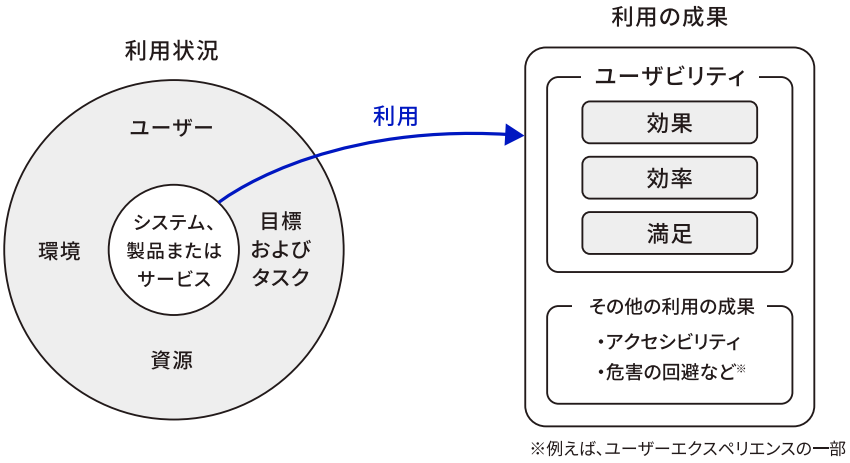

図2.2 ユーザビリティの概念。JIS Z 8521:2020 図1.2を元に作図。

ユーザビリティは、以下のような観点から評価することができます（いずれも「利用者にとってどうなのか」という観点なので、注意が必要です）。

- 有効性
- 効率性
- 満足度
  - 実用度
  - 信用度
  - 快感度
  - 快適度
- 学習のしやすさ
- 記憶のしやすさ
- エラー等による危害の回避及び低減
- 利用状況の網羅性

本ガイドブックでは現場で判断しやすくするため、以下のいくつかのJISに分散している観点を集約して記載しています。ソフトウェアの品質標準を定めているISO/IEC 25000 SQuaREシリーズでは、ユーザビリティは、利用文脈での評価観点である利用時の品質モデルの特性のひとつである便益性の副特性として記載されています。

- 「JIS Z 8521:2020（ISO 9241-11:2018のMOD規格）：人間工学―人とシステムとのインタラクション―ユーザビリティの定義及び概念」におけるユーザビリティの定義
- 「JIS X 25019:2025（ISO/IEC 25019:2023）：システム及びソフトウェア技術－システム及びソフトウェア製品の品質要求及び評価（SQuaRE）－利用時品質モデル」における利用時品質の定義

2.1.2 利用者

図2.3 利用者の分類。

利用者には、直接情報システムを操作する直接利用者と、情報システムを介して間接的に影響を受ける間接利用者がいます。また、サービスの提供や管理を行う責任組織、広くサービス利用に関わってくる公共・社会への影響も考える必要があります。

直接利用者には、以下のような人たちがいます。

- 申請・手続等の申請者
- ウェブサイトの訪問者
- 行政機関のシステム運用担当者・政策立案担当者
- 申請・手続等の審査事務局の担当者
- システムの保守運用オペレーター
- コールセンターのオペレーター

直接操作を行わないが、そのシステムの運用によって、個人や特定のグループ、組織等で影響が及ぶ人を間接利用者といいます。間接利用者に以下のような影響が及ばないか、注意深く調査する必要があります。

1. 健康・生活等への影響
2. 経済的価値への影響
3. 環境への影響
4. アクセスできる情報への影響
5. 特定のコミュニティへの影響
6. 人権の尊重への影響
7. 法律・規制・規範への影響

例えば、子育て支援センターの業務システムの設計は、間接的にこどもや保護者に影響を与えます。特定の通報の登録漏れやアラートを見逃すと、支援が必要な状況を見逃してしまうかもしれません。このような場合、実際に間接利用者にどのような影響があるのかを特定し、そのような状況に至るシナリオについて十分な対応策を検討しなければなりません。

2.1.3 使用エラー（ユースエラー）

利用者に起因する誤操作や間違いは「ヒューマンエラー」と呼ばれてきました。しかし、こうしたエラーは複合的な要因から操作ミスや認識ミスが引き起こされるのであり、人に帰責すべき問題だと強調されすぎると、根本的な課題解決からは遠のいてしまいます。インシデントの直接的な引き金を引いているのは人の操作ミスかもしれないが、根本的な原因はシステムの考慮不足にあるものが見過ごされ、二次被害を引き起こすことがあるのです。

そこで近年では「ヒューマンエラー」とは呼ばずに、「使用エラー」又は「ユースエラー」と用語が改められています（注記）。使用エラーを「引き起こさせない」そして「起きたとしても被害を軽減する」ため、システム側で十分な考慮を行っておくのはとても大切です。詳しくは6章で紹介します。

注記）この概念について、JIS Z 8521:2020 では「ユースエラー」、JIS T 62366-1:2022 では「使用エラー」と表記されています。JIS T 62366-1:2022 は「医療機器」、JIS Z 8521:2020 は「インタラクティブシステム」を扱っていますので、情報システムを取り扱う上では厳密には「ユースエラー」表記の方が適切ですが、本ガイドブックの執筆時点ではユーザビリティエンジニアリングを主題として取り扱っている観点から「使用エラー」表記を採用しています。

2.2 ユーザビリティは使っている時の文脈や環境に影響を受ける

紙に書かれた文字は、明るい場所でなければ読むことができません。普段は書きやすいお気に入りのボールペンも、山道で揺れる車の中では使いにくいでしょう。利用時の環境や文脈を理解しておくことは、ユーザビリティを確保する上で重要な要素です。

文脈の重要性を理解してもらうために、もう少し事例をもとに考えてみましょう。例えば車の運転において、ハザードランプは以下のような様々な意味を持っています。

- 路肩に停車している（停車する）とき：後続車に注意を促す
- 高速道路で渋滞に追いついたとき：後続車に危険を知らせる

ここまでであれば、ハザードランプの「文脈」を利用者（この例では後続車のドライバー）が取り違えても、あまり致命的な齟齬は起きそうにありません。

しかし「ありがとう」の意味でハザードランプを点滅させる「サンキューハザード」は少し厄介です。停車することを知らせるハザードが、サンキューハザードと勘違いされてしまうと、後続車は減速せず事故につながりかねません。このように、環境や文脈によって違う意味が成立してしまうことでアクシデントが起きる危険性は、日常の様々な場所に潜んでいます。

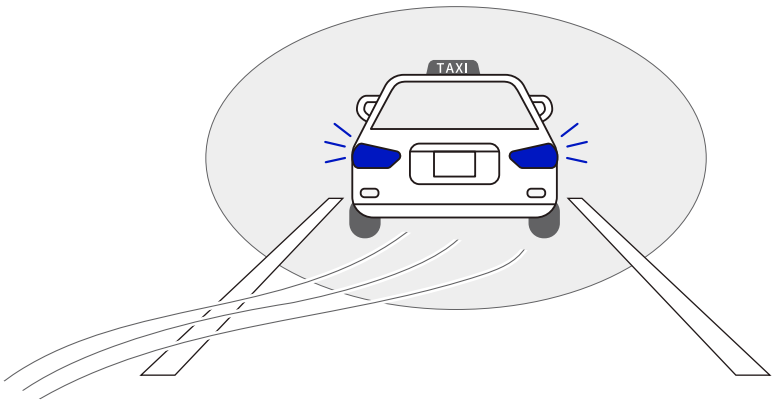

図2.4 停車していたタクシーが車道に出てきてハザードを点滅させている。

2.3 慣れているものが使いやすい

車のハザードランプは、危険予測が上手な人は回避できるかもしれませんが、運転に慣れていない人は見誤ってしまうかもしれません。「ユーザビリティ」とは「どれだけ利用できるか」という意味でもあり、利用者の熟達の度合いや慣れに大きく左右されます。

飛行機のコックピットには無数のスイッチが並んでいますが、パイロットは使いこなしています。このように、専門家が用いる複雑なシステムは、一見しただけでは何をどうしたらいいのか分野外の人にはよく分からないことでしょう。これらは使いやすさが熟達や知識に影響を受ける例です。

2.3.1 慣れ（熟達）が間違いを誘発する

熟達が逆に間違いを誘発するケースもあります。これも事例で考えてみましょう。車の運転中、カーナビゲーションが「ななめ左方向です」と通知してきたとします。前提知識なしに情報を受け取ったときは、疑いなく左側に進むかもしれません。

しかし、付近の地理に詳しく、カーナビゲーションを経験上あまり信用しておらず、現在位置から「目的地は右側にあるはず」と知っていたとしたら、判断に迷ったり、道を間違えてしまうかもしれません。

実際には道が大きくカーブしていて、正しい方向に進めることが明示されていれば、熟達者も判断に迷いません。情報を提示したりフィードバックを返す際には、初心者だけではなく熟達者がどのように判断をするかを考慮にいれ、曖昧さを排除する必要があります。

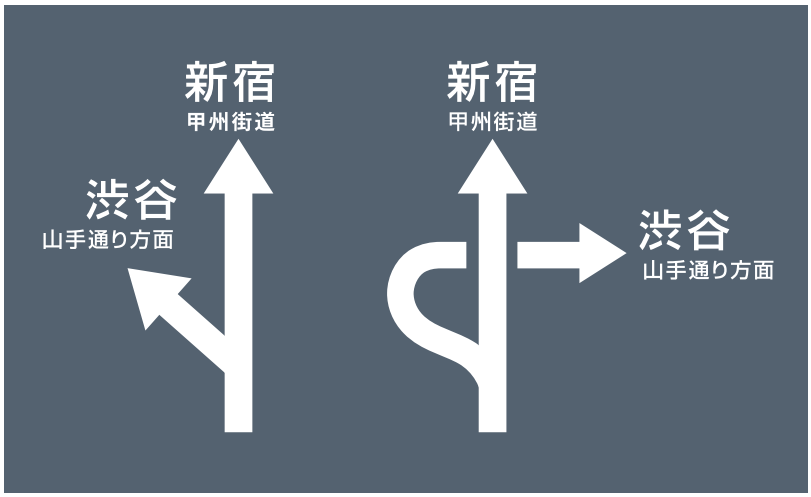

図2.5 示す位置に応じて使い分けるべき道路標識（道なりの分岐を示す道路標識と、地理上の位置関係を示す道路標識）。

2.4 ユーザーエクスペリエンス（User Experience: UX）

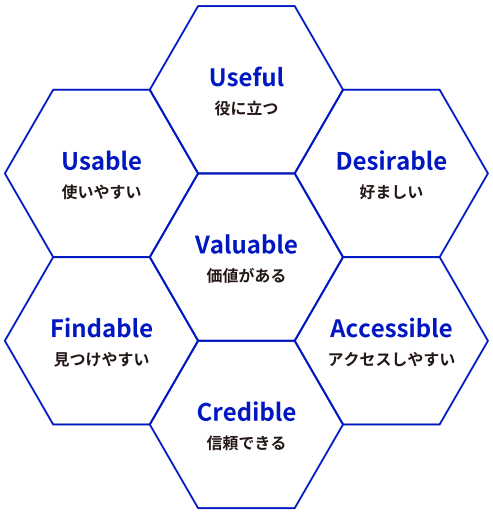

図2.6 User Experience Honeycomb（UXハニカム：ユーザーエクスペリエンスがユーザビリティを超えた概念であることを示す概念図）。
https://semanticstudios.com/user_experience_design/

これまでみてきたような情報システムの「使いやすさ」は、個別の機能の適切性のように要素分解可能なものだけでは決まりません。以下のような全体論的観点（Holistic Approaches）で捉えられるべきものです。

- システムを利用している環境や経時的な過程において、利用者がどの程度機能的かつ効率的に目的を達成できるか（合理性の観点）
- システムの利用が関係する活動全体に対する経験

いかに効率的なシステムであっても、利用者が怯えながら操作していては、使いやすいとは言えません。こうした各観点からの総合的な利用者の評価のことをユーザーエクスペリエンス（UX）（注記）といいます。ユーザーエクスペリエンスは以下の観点で評価できます。

- システムの利用を通じ、どのような活動が可能になるのか予測・イメージできること
- 利用者がイメージ通りに操作できること、また、操作結果がイメージと異なる場合には、その違いが適切にフィードバックされ、安全に試行錯誤できること
- 活動を通じて、利用者がやりたかったことが実現できたと（利用者自身が）評価できること
- 個人が尊重され、心地よく活動を実施できたと（利用者自身が）評価できること

ユーザーエクスペリエンスについて議論する場合、人の生理的な機能・認知特性によって、責任分界点を合理的に切り分けできないことを常に意識し、人間工学の観点を適切に導入する必要があります。利用者の活動のうち、システムが責任を負える範囲は常に限定的だからです。目覚まし時計のアラームが適切に作動したとしても、実際に起床できるとは限りません。このように、ユーザーエクスペリエンスは、それ自体を直接デザインすることはできません。

注記）日本でよく流布している解説で「User eXperience」とXを強調する表記をすることがありますが、本来は最初の音節「Ex」からきているため、本ガイドブックでは「User Experience」と記載しています。例えば「XML」は「Extensible Markup Language」の略称ですが「eXtensible Markup Language」とは書かないのと同様です。

2.5 認知とバイアス

人は、常に合理的に判断して行動しているわけではありません。複雑な環境下や時間的余裕のない状況では、過去の経験や目に見える情報をもとに「とっさの判断」や「思い込み」によって行動してしまうことがあります。こうした人間の思考の特性は、ユーザビリティや安全性の評価において、単純な操作手順の整合性だけでは説明しきれない「使用エラー」や「リスク」の温床となることがあります。

例えば、複数のボタンが並んだ画面で、意図せずキャンセルボタンを押してしまうケース。あるいは、確認画面を見落として手続きを完了したと誤認するケース。これらは単なる不注意ではなく、「どのように知覚し、どう判断し、どんなバイアスが働いたのか」を理解することで、設計上の改善余地が明らかになります。

本節では、人の認知過程の特徴と、判断や行動に影響を与える代表的な認知バイアスについて概観します。システム設計においてどのようにバイアスの影響を受けない、あるいはバイアスを逆手に取った支援設計が可能かを考える視点としてご活用ください。

2.5.1 文脈効果

人が与えられた情報を理解しようとするときには、2種類の情報処理プロセスが働きます。

- 周囲の状況やあらかじめ知っている概念の中で理解をしようとするトップダウン処理（概念駆動型処理）
- 与えられた情報を最初の手がかりにして高次の情報処理へと進んでいくボトムアップ処理（データ駆動型処理）

特にトップダウン処理をしているときには、全く同じ情報を受け取っていたとしても、人の置かれた状況や文脈によって全く異なる認知をしてしまうことがあります。これを文脈効果といいます。

図2.7 トップダウン処理(概念駆動型処理)の例。ABC…と続けて読むと一番右端は英字のIにみえるが、987…と読んでいくと数字の1にみえる。

システムの設計者は、そのシステムの最も優れた熟達者であり、システムの文脈を熟知しています。[2.3.1 慣れ（熟達）が間違いを誘発する]で示した車のナビゲーションの事例を思い出してください。設計者は完璧な地図が頭に入っているので「ななめ左」という情報表示に違和感を持ちません。

2.5.2 デザインモデルとメンタルモデル

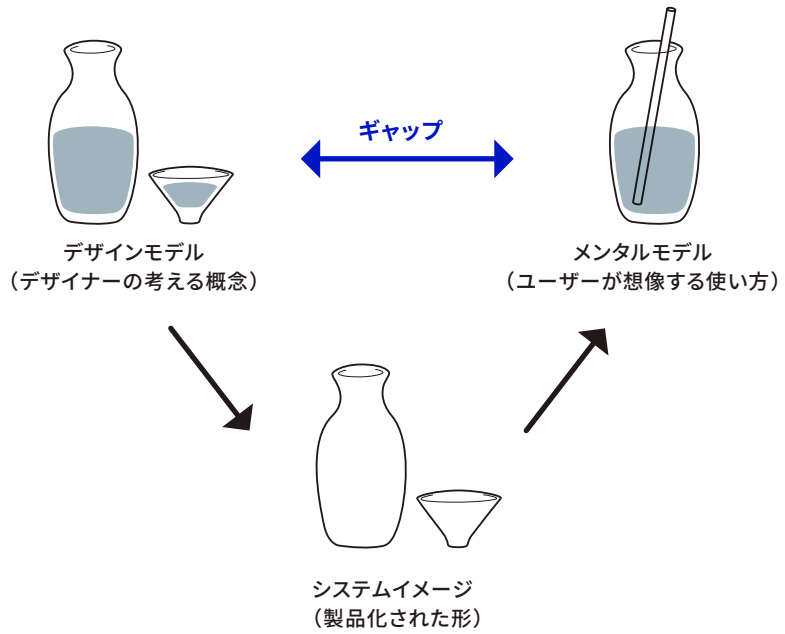

図2.8 システムイメージからユーザーが認識するメンタルモデルは、システムイメージの元となったデザインモデルとは全く異なる場合がある。

利用者が、操作しようとしている情報システムの機能や構造、データの処理方法に対する認識（メンタルモデル）と実際（システムイメージ）の違いが、ユーザビリティの低下を引き起こします。利用者の持つメンタルモデルは、マニュアルや実際にシステムを操作してみての経験から獲得されるため、設計者が認識しているもの（設計段階の想定にあたるデザインモデルや、実際にできあがったシステムイメージ）とは全く異なります。

このあとの使用エラーのセクションでは、利用者の「操作ミス」について触れますが、意図的な失敗（試行錯誤）なしに、利用者がメンタルモデルを獲得することはできません。失敗を許容しないのではなく、安全に失敗できることが大切です。

このため、飛行機の操縦のようにミスが致命的な結果を導くシステムでは、シミュレーターのように利用者が安全に失敗できる状況を作り、メンタルモデル構築を支援する必要があります。

2.5.3 バイアス

ヒトの生理学的な認知機能の働きにより、意図しない（事実とは異なる）認知や判断をしてしまうことをバイアスといいます。人は多くの情報を瞬時に処理するために、様々なバイアスに頼っています（肉食動物に捕食されてしまう危険性と常に隣り合わせだった私たちの祖先は、瞬時に判断ができる必要があったのです）。

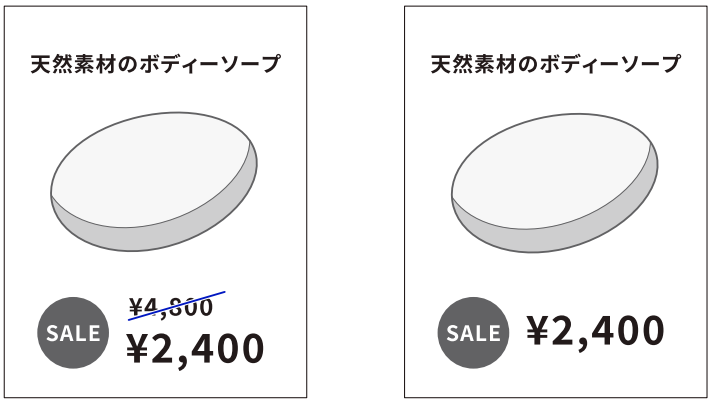

図2.9 アンカリング効果の例。左側の値札のほうが、元値4,800円が2,400円に下がったように見え、お得に感じる。

以下に示すような、様々なバイアスがヒトには常に働いています。たとえバイアスについて熟知していたとしても、バイアスに打ち勝ったり回避することは（意識とは無関係に働くため）できません。

- 正常性バイアス
- 確証バイアス
- 後知恵バイアス
- サンクコストバイアス
- 帰属の誤り
- アンカリング

バイアスはシステムの企画・設計者にも等しく働きます。システムの開発者と同じ理解や振る舞いを、利用者もしてくれるものだと無意識に期待する確証バイアスや、過去の投資や意思決定に対するサンクコストバイアスが働いてしまうことがあります。こうした「思い込み」と実際がずれてしまうと、利用者目線では使いにくいサービスが企画されてしまうことになります。

開発者の期待	利用者の印象
スマートフォンを用いた認証サービスは便利だ	スマートフォンを落とすと怖いので使いたくない

こうしたバイアスや思い込みの影響を排除するため、企画・運用しているサービスの利用状況や、利用者目線での課題を把握していくことが大切です。

3

デザイン原則を理解する

3.1 デザインと倫理

情報技術やデザインは、人間の社会生活に大きな影響を与えます。例えば、プライバシーの尊重や著作権の考え方は、インターネットの普及によって大きく変化してきました。システムの道具的な価値や機能の有効性・効率性・安全性のみをデザイン活動の規範としていると、本来社会にいる多様な人々のニーズや、アーキテクチャが潜在的に持ちうる不公平さに対してのアプローチが欠如してしまうことがあります。

特殊詐欺やフェイクニュースは、ヒトの認知機能を悪用しています。UXデザインの領域に限ってみても「解約の導線を意図的にわかりにくくする」というように、デザインの機能を悪用する「ディセプティブパターン（ダークパターン）」が問題になっています。

このため技術やデザインが与える影響を意識し、その道具的価値・機能的価値だけではなく多様な価値に焦点を当て、人の様々な権利や尊厳を尊重できるように、サービスデザインに関わる人が守るべき倫理原則を考えていくことは非常に大切です。ここではいくつかの概念（注記）を紹介します。

- 利用者の人権を尊重すること（Human Rights）
- 利用者の利益を優先すること（Beneficence）
- すべてのステークホルダーの助けになること（Beneficence for all）
- 公平かつ公正であること（Justice）
- ウェルビーイングを追求すること（Well-being）
- 環境・文化を含めた持続可能性を追求すること（Sustainability）

注記）これらの概念は、ベルモントレポート、アマルティア・センやマーサ・ヌスバウムらによるケーパビリティアプローチ（Capability Approach）等を参照しています。

3.1.1 ディセプティブパターン

「解約手続がなかなか見つけられない」というように「運営者側がやってほしくない操作」から利用者を遠ざけたり、あるいは逆に近づけたりするために、認知の仕組みや心理を悪用する手法を「ディセプティブパターン（欺瞞的なデザインパターン）」といいます（かつては「ダークパターン」と呼ばれていましたが、国際的に訴訟も増え、我が国でも社会問題となっていることから、現在ではより厳密な表現に言い換えられています）。ディセプティブパターンには、例えば以下のようなものがあります。

- 誤認させる（例：定期購読なのに買い切りの契約のように見せかける）
- 気づかない（例：定期購読になっていることに気づかない）
- サービスにとって都合の良い選択肢へ誘導する（例：有料プランと無料プランがあるときに、有料プランだけが視認しやすい）
- 特定の操作を困難にする（例：退会するまでにいくつものステップが必要である）
- 焦らせる（例：「この割引はあと5分で終了します」と時間を区切る）
- 社会的信用を強調する（例：「今日この部屋は8人がチェックしています」と人気を示す）

図3.1 文章中で例示したディセプティブパターンの手法をすべて盛り込んだ例。「無料でお試し」を申し込むと、2,400円のボディーソープの定期購入を契約してしまうことになる。在庫は実際にはたくさんあり、閲覧者数はランダムな数が表示されている（本図は、実際にあったパターンを組み合わせている）。

行政サービスにおいては「個人情報の利用に同意する」ボタンを強調して「同意しない」ボタンを目立たなくしてしまい、行政側に都合の良い意思決定（個人情報利用への同意）に誘導してしまうケースがあります。この表現はディセプティブパターンであり、またGDPRや個人情報保護法にも抵触してしまう可能性があります（注記）。

注記）イギリスのICO（Information Commissioner's Office）とCMA（Competition and Markets Authority）は、2023年8月にディセプティブパターンに関する共同声明を発表し、こうした実装はGDPRの公平性及び透明性の原則を侵害する可能性が高いと指摘しています。
https://www.drcf.org.uk/publications/papers/ico-cma-joint-paper-on-harmful-design-in-digital-markets/

ディセプティブパターンは、局所的にみれば「加入率の上昇」「解約率の低下」のように、見かけ上KPIを改善させるように見えますが、実際にはKPIを歪め、中長期的には利用者の評価を低下させてしまうだけではなく、「契約行為について適切な説明ができておらず、契約相手が同意していない」ことにもつながります。ディセプティブパターンに陥るのを避けるためには、プロダクトの評価時に、以下のような対策をしておく必要があります。

- あくまで利用者中心の視点で意思決定が適切に行われるように設計・評価を行う
- プライバシーやセキュリティ、関連法の動向を常に把握する
- プロダクトの体験が評価される定性的な指標を導入する
- KPIを歪める施策になっていないか、別の指標でも確認する

3.2 ユーザビリティデザインの原則

本ガイドブックでは、デザインの実践にあたり以下の4原則を満たすことを推奨しています。

- アクセシビリティファースト
- 対話の原則
- 情報提示の原則
- 人間中心の原則

各原則の関係性を図に示します。

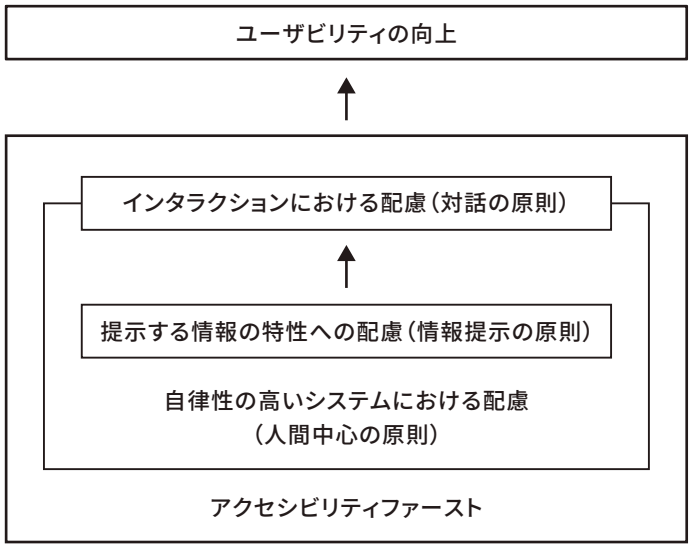

図3.2 本ガイドブックにおいて参照するデザイン原則の関係。アクセシビリティファーストを基盤として、対話の原則、情報提示の原則、人間中心の原則を満たすことが、ユーザビリティの向上に寄与する。

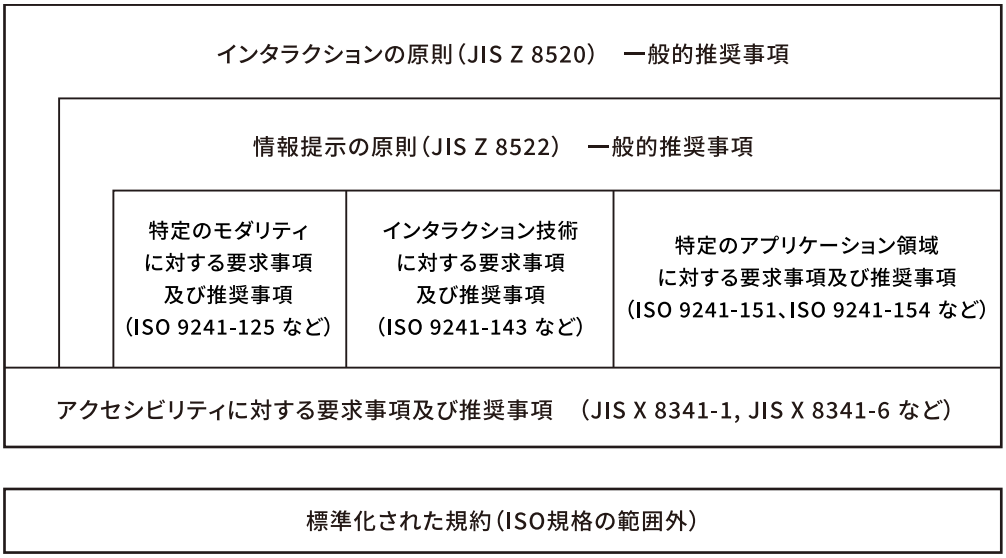

図3.3 情報提示に関する規格群の関係（JIS Z 8522:2022）

注記　JIS Z 8520では、インタラクションの原則及び情報提示の原則が並列に示されている

3.3 アクセシビリティファースト

万人にとって使いやすいものを作るのは困難です。例えば、大人向けのスツールは子どもには大きすぎますし、右利きの人向けにデザインされたハサミは左利きの人には使いにくいものです。つまり「使いやすさ」は「特定の誰か」にとって、という前提があって成り立ちます。

しかしながら「特定のユーザーに向けて」というと、多くの行政職員は違和感を覚えるのではないでしょうか。行政機関が提供するサービスや情報は、できるだけ多くの人にとって、使いやすかったり、分かりやすいものでなければなりません。従って、ユーザビリティを向上させる時に無数の選択肢からひとつの選択肢を選ぶ時には「できるだけ多くの人や、利用状況、利用文脈において使いやすい」ものを選ぶこと、多様な使い方を保証するアクセシビリティの確保とセットで取り組み、ユーザビリティとアクセシビリティの間で齟齬が生じる場合は、アクセシビリティを優先して取り組んでいく必要があります。

3.3.1 アクセシビリティとは

情報システムにアクセスできる能力、つまり（製品やサービスを）利用できること、またはその到達度を指します。ユーザビリティが「特定のユーザーにとって」の品質を示す概念なのに対して、アクセシビリティは「最もターゲットを広げた（多種多様な）利用者にとっての使いやすさ」を示す概念です。行政サービスは代替手段が存在せず、「使いやすい」という実感を持って多種多様な特性のある利用者に受容される必要があります。デジタル庁ではアクセシビリティの確保を最優先事項としています。

アクセシビリティの対象を国民向けの情報システム、あるいは、各府省が設置・公開するウェブサイトに掲載するコンテンツのみに限定して捉えるのは適切ではありません。「みんなの公共サイト運用ガイドライン（2024年版）」（総務省）では、公式ホームページだけでなく、関連サイト、ウェブシステム、さらには職員向けのウェブコンテンツも含めたすべてのウェブコンテンツがアクセシビリティ対応の対象であることが明記されています。たとえば、団体内の職員向けに運用されるイントラネットのほか、文書管理・財務会計・住民情報管理等の業務アプリケーションも含まれます。これは、行政サービスの公平性だけでなく、障害のある職員を含めたすべての職員が適切に業務を遂行できるようにするための環境の整備にもつながります。

詳しくは「みんなの公共サイト運用ガイドライン（2024年版）」（総務省）「DS-671.2 ウェブアクセシビリティ導入ガイドブック」（デジタル庁・2025改訂）等を参考にしてください。関連するISO及び日本産業規格、国際的に参照されている標準には、JIS X 8341シリーズがあります。

3.3.2 JIS X 8341シリーズ

JIS X 8341シリーズは、高齢者、障害者、不慣れな方を含めた幅広い利用者を対象に、FAX等の情報通信機器やソフトウェア及びサービスの情報アクセシビリティについて、開発者及び経営者（企画者）が配慮すべき要件をまとめたJISのシリーズ群です。

- JIS X 8341-1 第1部：共通指針
- JIS X 8341-2 第2部：パーソナルコンピュータ（ISO/IEC 29136）
- JIS X 8341-3 第3部：ウェブコンテンツ（ISO/IEC 40500）
- JIS X 8341-4 第4部：電気通信機器（ITU-T F.790を参考にしている）
- JIS X 8341-5 第5部：事務機器（ISO/IEC 10779）
- JIS X 8341-6 第6部：対話ソフトウェア（ISO 9241-171）
- JIS X 8341-7 第7部：アクセシビリティ設定（ISO/IEC 24786）
- JIS Z 8529:2024（ISO 9241-20:2021）

なお、元々枝番1（JIS X 8341-1）と対応関係にあったISO 9241-20は2021版の更新で大幅な改訂が行われたことから、枝番1を更新するのではなく、JIS Z 8529:2024として新設されました。このため、JIS X 8341-1は対応国際規格無しのJISに改訂されています。

JIS X 8341シリーズの中で、ウェブコンテンツ（ウェブサイト）のアクセシビリティについて定めているのは枝番3（JIS X 8341-3）です。最終改訂が2016年に行われていますが、2016年版は参照元であるW3Cの「Web Content Accessibility Guidelines（WCAG）」2.0と互換性があります。WCAG 2.0は2008年12月にW3Cが勧告したもので、当時はまだスマートフォンの普及が始まる前でした。2024年現在、多くの利用者はスマートフォンを使っています。このため、JIS X 8341-3:2016に基づいた試験のみでは十分に対応できません。前掲の総務省・デジタル庁の各ガイドライン・ガイドブックでは、WCAG 2.1や2.2を併せて参照するように求めています。

WCAGはW3Cが勧告している、ウェブコンテンツ（ウェブサイト）のアクセシビリティについて定めた国際標準で、最新版はWCAG 2.2です。「デジタル庁デザインシステム」（デジタル庁）では、WCAG 2.2に対応できるよう整備をしています。

また、WCAG の達成基準の多くはウェブサイトのアクセシビリティを対象としており、iOSやAndroid等のネイティブアプリのアクセシビリティを考慮するときには、WCAGでは対応できないところがでてきます。各OSのプラットフォームが提供しているインタフェースガイドライン等の技術仕様を確認しながら開発・試験手法を定める必要があります。なお、現在議論が進んでいるWCAG 3.0では、ネイティブアプリも対象とすることが検討されています。

3.3.3 環境整備・合理的配慮としての代替手段の提供

アクセシビリティを確保したくても、数万点に及ぶスキャンされた画像があったり、様々な事業者からPDFが提供されるようなシステムでは、アクセシビリティを確保するのは困難です。

このような場合は、代替手段を予め企画し準備していく環境整備や、障害者から要望があったときには合理的配慮を実施できる体制を整えておかなければなりません。

また、アクセシブルではないコンテンツを扱う場合には、ウェブアクセシビリティ試験の例外扱いにするのではなく、対応が難しいという事実や今後対応する計画についてを試験結果に記載するほうが好ましい対応であるといえます。

3.4 対話の原則

利用者は情報システムと対話する（情報を取得したり操作をする）とき、

- システムの利用を通じ、どのような活動が可能になるのかを予測・イメージする
- イメージ通りに操作ができるか、試行錯誤を繰り返す

という過程を通じ、そのシステムを使って「何ができるのか」だけではなく、その「限界はどこか」「どのタイミングでその機能を使えばいいのか」「リスクはなにか」等を学習していきます。

例えば車のブレーキの操作を通じて、利用者は以下のようなことを学んでいきます。

- ブレーキを踏むと車を減速させることができる。
- 空走距離と制動距離の制約がある（車は急には止まれない）。
- 路面状況によって適切なタイミングでブレーキを踏み始めなければならない。
- 急ブレーキを踏むと後続車の追突を招きかねず危険である。
- 先行車が急ブレーキを踏んだときに追突しない程度の車間距離をあける必要がある。

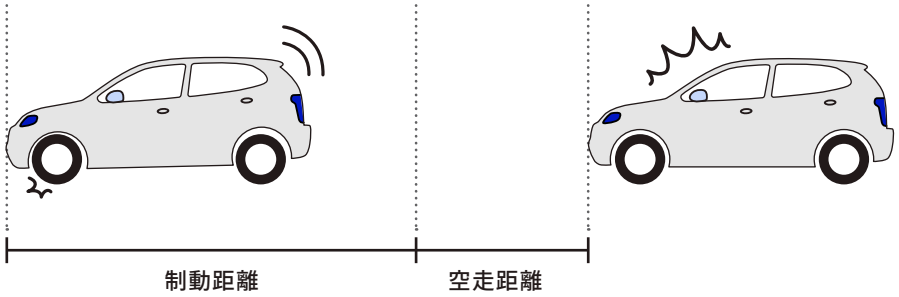

図3.4 ブレーキを踏んでから車が止まるまで。

ブレーキひとつを取ってみても、このように複雑な情報を汲み取ってタスクが実行されています。こうした学習の過程をなるべく安全かつ円滑に実現するために守るべき原則のひとつが、対話の原則です。関連する規格として「JIS Z 8520 人間工学―人とシステムとのインタラクション―インタラクションの原則」があります。

3.4.1 ユーザーが行うタスクへの適合性

タスクへの適合性とは、利用者にとってそれが合目的であるかどうかを指しています。

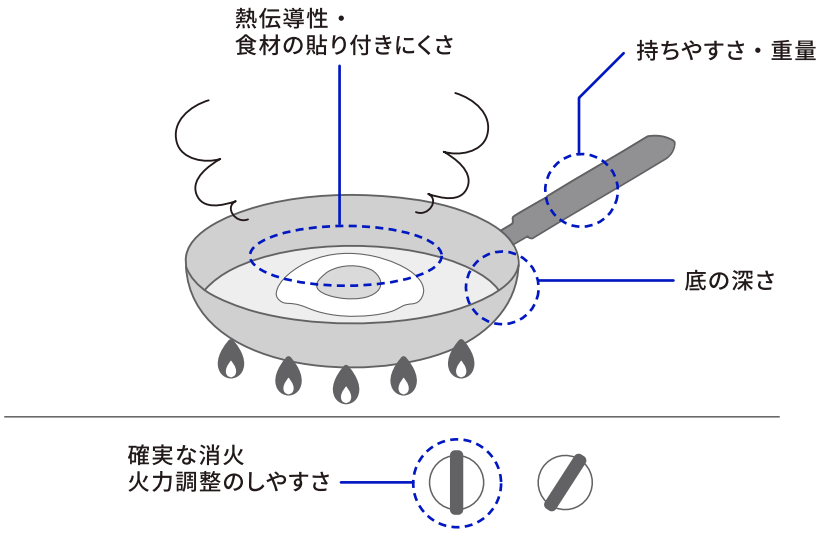

図3.5「目玉焼きを焼くこと」への適合性の例。

利用者には「目玉焼きを焼く」といった具体的な活動目標があり、単に「火力が調整できる」ことではなく「目玉焼きを焼くのに適切な火加減が設定できる」ことを期待しています。システムが提供している機能やインタフェースは、利用者のタスクにうまく合致していなければなりません。

情報システムでは、以下のような工夫が必要になります。

ポイント

- タスクの完了に影響がある情報を提示する
- 必要のない情報を表示しない
- 適切な形式で表示する

3.4.2 インタラクティブシステムの自己記述性の確保

利用者に対して、システムの状況を適切な内容・タイミングでフィードバックしているかを指しています。

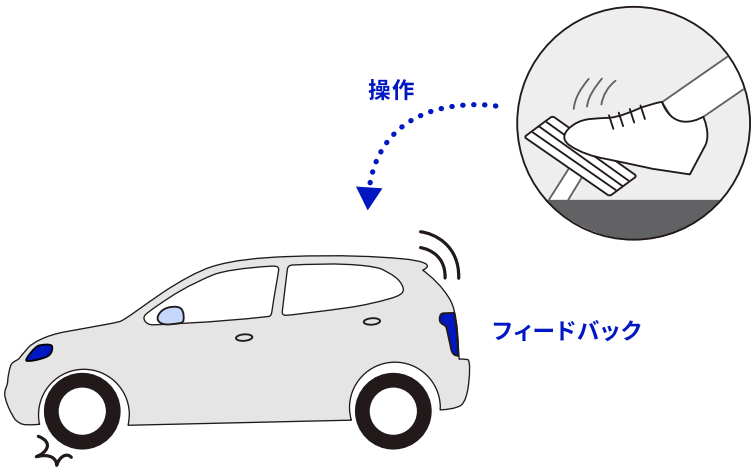

図3.6 スピードメーターの表示や減速によって働く加重から、ブレーキ操作が行われたことが分かる。

車を運転していると、自動車の状態はすぐに把握することができます。道を逸れていれば逆の方向にハンドルを切りますし、スピードが出ていればブレーキをかけることができます。目玉焼きを焼くフライパンは、加熱しすぎていれば油の弾ける音や煙で、すぐに状態が把握できます。

こうした物理的な道具と違って、情報システムは「そのように設計しなければ」、システムの状態を伝えることができません。そのシステムが「どのようなデータや現実の事物に対して、どのような操作が実施でき、結果として何が起きるのか」「今どの機能を操作していて、どの過程にあるのか」を利用者に伝える仕組みを人工的にデザインする必要があります。これを「自己記述性」といいます。自己記述性を確保するための仕組みには、以下のようなものがあります。

自己記述性を確保するための仕組み

- ナビゲーション
- 状態表示
- 通知

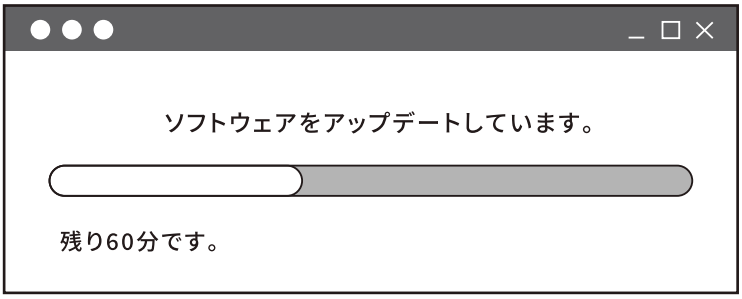

図3.7 進捗状況が分かるプログレスダイアログの表示。

3.4.3 ユーザーが抱く期待への一致

利用者が期待・想定したとおりに情報が表示され、機能が動作するかを指しています。

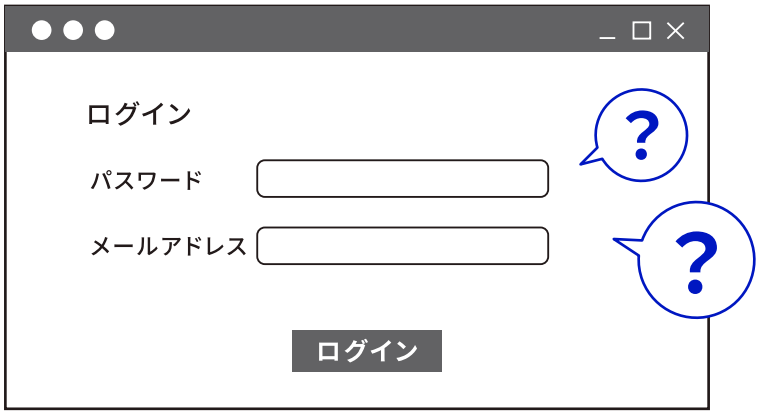

図3.8 一般的なログインフォームとは異なる順序で要素が提示されたことで、利用者が戸惑ってしまう。

利用者の期待する動作や、利用者が学習している既存の概念とは違う振る舞いをシステムがすると、利用者は戸惑ってしまうことになります。[2.5.2 デザインモデルとメンタルモデル] も参考にしてください。以下にいくつか事例を示します。

利用者の期待とシステムの振る舞いが一致しない例

- 「伝票」といって経理担当者が想像するのは「入金や出金等の取引内容や金額等を記載した書類」です。経理担当者向けのシステムで「見積を決裁者に伝えるための帳票」のことを「伝票」と呼んでしまうと混乱が起きてしまいます
- ログイン画面では通常、「メールアドレス」を入力したあとに「パスワード」を入力します。これを逆にすると利用者は混乱します
- 姓名を分割して入力することに慣れている利用者は、姓名をつなげて入力することに不安になります

何が「利用者にとっての常識なのか」は「テストしてみるまで顕在化しない」ことが多いため、インタフェースの設計途中で繰り返し利用者を被験者とした操作試験（ユーザビリティテスト）を行うことが重要になってきます。

3.4.4 ユーザーによる学習性の確保

簡単にいうと、安全かつ容易に試行錯誤ができるかを指しています。

図3.9 二度失敗した目玉焼きだが、三度目に成功した。

「学習性が確保されている」状態は「安全に失敗することができる」ことと同義です。 図3.10に示した目玉焼きのように、現実の世界では自然と失敗することができます。利用者は試行錯誤を通じて、どこまでなら安全なのか、何をしたら失敗するのか、様々なパラメーター（フライパンの熱伝導率やテフロン加工が、目玉焼きの剥がしやすさにどう影響するのか等）を学びます。

こうした学びは柔軟性につながります。つまり、「兄は半熟のサニーサイドアップが好きだが、妹は堅焼きのターンオーバーが好き」というような様々な状況に対応できるようになっていきます。

情報システムでは、利用者が「そのシステムの使い方を理解でき」「様々な状況で使える」ように、うまく学習の機会に配慮された設計にする必要があります。ゲームのチュートリアルでは安全に失敗ができるよう設計されています。自動車教習所ではシミュレーターを使った危険予知訓練や、安全な場所での急ブレーキ操作等を通じて「車の挙動」の限界を学んでいきます。

情報システムにおいても原則は同じです。例えば、普段使っているのとは別のメールソフトを初めて使うときは、BCCとCCを間違えないように気をつけたり、見逃してはいけないメールを放置しないよう工夫するため、いろいろと機能を確認するのではないでしょうか。

学習性確保のポイント

- 試行錯誤をしても安全な領域を特定し、確実にフィードバックすることで、突発的に「ギリギリのラインでの操作」をするときに、どこまでが安全なのかを試せる
- 本当に間違えてはいけないものは、システム側で（可能な限り）責任を担う
- 初心者が必要なフィードバックと、熟練者が必要なフィードバックを分ける

明示的なシミュレーション環境以外にも、学習性を担保するための工夫を設けることができます。例えば以下のような工夫が考えられます。 普段用いる機能を活かして、学習性を担保することができるのです。

- 公開先を自分だけに設定して、原稿を公開することができる
- 下書きを保存することができる
- データを複製しておける
- 操作のUnDo機能がある（やり直すことができる）

3.4.5 ユーザーによる制御可能性の確保

「ユーザーの実際の手順や、時間の使い方の邪魔をしない」というのがこの原則です。システムの設計者は、利用者の入力操作を「理想的で直線的な流れ」で考え「中断なしに終わらせようと」してしまいがちですが、そのとおりに入力できることはほぼありません。

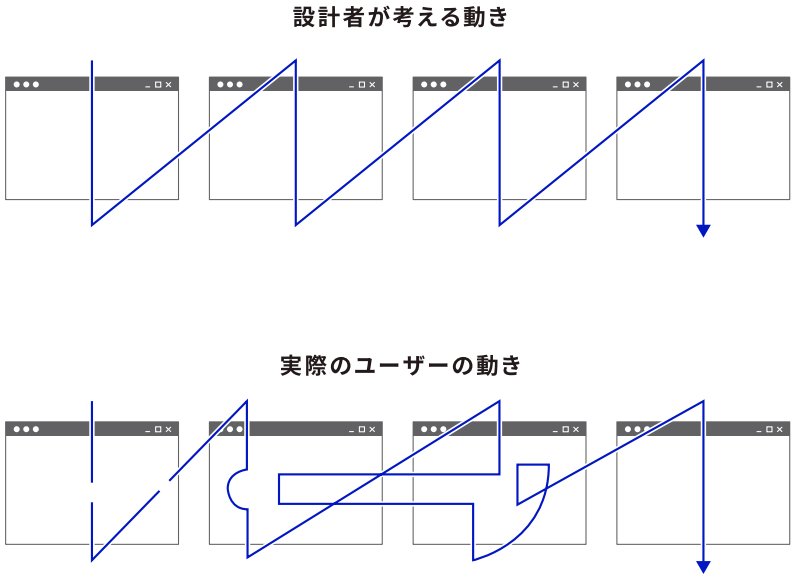

図3.10 開発時には上段のようにシームレスな操作を前提にしてしまうが、実際には下段に示すように、操作の中断や、入力しやすい項目からの操作、画面の行き来があり、より複雑である

車の運転のように一般的に操作はマルチタスクであり、操作をしている最中に利用者に影響を与えている要素は多数あります。料理をするときに「レシピ通りの手順で」実施できることは稀です。大抵は複数の料理を同時に作っていますし、飼い猫が足に飛びついてきたり、宅配便の配達業者が呼び鈴を鳴らすかもしれません。

制御可能性確保のポイント

- ユーザーが手順をコントロールできる
  - 操作を中断し、状態を保存することができる
  - 時間を制限しない
  - 手順を制限しない（先に入力できる項目から入力・保存しておける）
  - 操作を（必要に応じて）キャンセルできる
- システムの都合はシステム側で吸収する

また、操作や情報の入力を行うタイミングや順序は、なるべく利用者が決められるようにして、中断できるようにしておきましょう。利用者は、システムが期待する順序では情報が入力できないかもしれないからです。「パスポートの入力が必要だけれど手もとにはないので、先に入力できるものだけ入力しておこう」といった具合です。

> セキュリティ上、時間制限を用いる必要がある場合は、時間を利用者が明示的に延長できる等、代替手段を用意する必要があります。拡大鏡等の機能を使い、画面を拡大して読んでいたり、入力に時間が必要な操作端末を使っていたりする利用者がいるためです。

3.4.6 使用エラーへの耐性の確保

人はなんらかの意思決定や行動を行ったときに、認知・心理・行動特性等の要因で、本来意図した結果に至れないことがあります。分かりやすいのはバイアスです。情報システムを利用しているときに、正しく使おうとしているにも関わらず、予期したものと異なる結果を引き起こしてしまったり、操作を省略してしまうことで起きるエラーのことを使用エラーといいます。

図3.11 エラーが発生していることは分かるが、どのようなエラーであるか、またどのように対処すべきかが分からない。

使用エラーへの耐性の確保とは、損害が発生しないように工夫したり、損害を軽減するようなしくみや制約を設けておくことを指します。より具体的にいうと、なんらかのエラーが起きた時には、利用者が単独で状態を回復したり、障害を回避できるようになっている必要があります。

使用エラーへの耐性確保のポイント

- エラーが起きても、エラーの内容が自分で判断できる（復帰できる）
- エラーの内容を、サポートや管理者に伝えやすい
- どこにエラーが起きているのか把握しやすい
- 回復・回避のための機能が用意されている

詳しくは6章以降で紹介します。

確保ができていない例

- 「NC201 Error」：エラー番号だけが表示されても、利用者にはその場で何もできない
- 「管理者に相談してください」：利用者は管理者にコンタクトを取らなければ問題を解決できません（利用者も困るでしょうが、同じ問い合わせを何回も受ける管理者も大変です）

3.4.7 ユーザーのエンゲージメントへの配慮

第一印象が「使いにくいな」とか「わかりにくいな」と感じたシステムは、なかなかその後も使う気にならないものです。また、適切なゴールや操作の筋道が提示されていないと、操作のモチベーションはどんどん低下していってしまいます（ECサイトで、電話サポートを受けるための窓口が、なかなか見つからないときのことを思い出してみてください）。

図3.12 適切なゴールにいつまでも到達できない例。Q&Aを開くとチャットボットで問い合わせるよう指示され、チャットボットで問い合わせるとQ&Aに誘導される。

利用者のモチベーションが適切な状態に維持されるよう、以下のような状態を避けたり、モチベーションを調整するための工夫をする必要があります。

- ニーズとの不一致
- 利用者を尊重していないと感じられる表現や機能
- 効率が悪い機能やシステムのパフォーマンス
- 不合理にみえる要求

自分が損をしてしまうようなことでも、モチベーションを維持できない・興味が湧かないものに人はリソースを割きません。毎日使用するシステムに対しては学習のモチベーションを維持できますが、年1回しか使わないようなシステムに対して学習意欲はほとんど維持できません。結果として起きるのは、こんな操作です。

- 最初の見出ししか見ないで読み飛ばす
- 内容を確認せずにチェックボックスをオンにする

ウェブサービスの運営者はしばしば「誤ってアカウントを削除してしまった」というサポートリクエストに直面します。何重にわたって「本当に消していいのか」と確認していても、アカウントを消してしまうことがあるのです。

- 自分に関係する情報への手がかりが十分に用意されていると感じる
- 現状支払えるリソースに対して、適切な情報の粒度・量である

といった条件を満たせれば、利用者は（使えるリソースに応じて）じっくりと情報を読もうとします。きちんと情報を届けるには、利用者の利用状況とモチベーションを十分に把握しておく必要があるのです。

3.5 情報提示の原則

ヒトはすべての情報をそのままの形で受け取れるわけではありません。バイアスが働いたり、動いているものに気を取られたりします（こうした特性を上手に利用しているのがマジックです）。ヒトの生物としての認知特性にあわせて、情報システムが情報を提示できるようにするために遵守すべき原則が、情報提示の原則です。

- 気づきやすくする
- 注意を逸らさないようにする
- 区別しやすくする
- 解釈しやすくする
- 簡潔にする
- 内部一貫性及び外部一貫性を保つ

関連する規格として JIS Z 8522:2022 人間工学―人とシステムとのインタラクション―情報提示の原則（ISO 9241.112）があります。

3.5.1 気づきやすくする

重要な情報や操作のための要素を、すぐに気づいてもらえる形式で提示する必要があります。
よくある問題の例

- 重要な通知が他の情報に埋もれて目立たない
- 重要なボタンやリンクが画面の下部に配置され、スクロールしないと見えない
- 画面の隅に表示された通知に気づかず、情報を見過ごす

改善例

- 重要な通知は色やサイズ等の差別化を行う
- 重要な情報を、再度リマインドする

3.5.2 注意を逸らさないようにする

画面要素のアニメーションや通知が、利用者の集中を妨げないようにする必要があります。スライドショーのように動きのある要素は、必ず利用者が停止・制御できるようにする必要があります。

よくある問題の例

- 過剰に動くアニメーションや動画が目立つ
- 多くの通知が届き、集中力が削がれる

改善例

- 重要な通知を優先順位ごとに表示する
- ポップアップや通知を必要最低限にする

3.5.3 区別しやすくする

異なる情報や機能が視覚、操作、構造の面で明確に区別される必要があります。色・アイコン・テキストの併用のように、複数の手段で違いを示せるようにすると、より使いやすくなります。

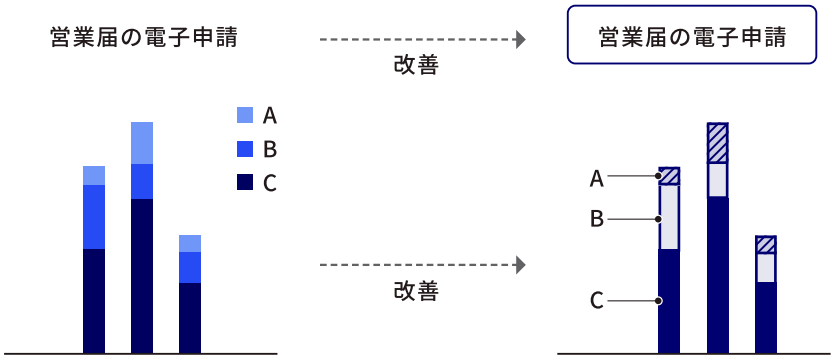

図3.13 隣接するデータ要素が似た色で区別しにくいグラフを、区別しやすい表現に改善した。

よくある問題の例

- ボタンとテキストが同じ色と形で区別しづらい
- グラフや図で色のみを使ってデータを区別している
- 似たような名称のラベルが並んでいる

改善例

- UI要素のコントラスト比を3:1以上に設定する
- 色に加えてラベルやパターンを使って情報を伝える
- スクリーンリーダー等の支援技術でも判別・判読できるように、HTMLマークアップを適切に実施する

3.5.4 解釈しやすくする

情報や機能が直感的に理解できるように、シンプルで明確な言葉や構造を使用します。

よくある問題の例

- フォーム入力にラベルがなく、目的が分からない
- エラーメッセージが曖昧で解決方法が示されていない

改善例

- 各入力項目に明確なラベルを付ける
- エラーメッセージには具体的な解決策を含める（例:「パスワードは8文字以上必要です」）
- HTMLマークアップを適切に実施し、論理的な順序で情報を提示する

3.5.5 簡潔にする

必要な情報のみを簡潔に提示し、利用者が混乱しないようにします。

よくある問題の例

- 長文が多く、要点が分かりにくい
- リンクのテキストが「こちら」等曖昧で目的が不明瞭

改善例

- 文章を箇条書きや短文でまとめる
- リンクテキストを具体的にし、目的が明確になるようにする（例:「詳細な仕様はこちら」→「製品仕様」）

3.5.6 内部一貫性及び外部一貫性を保つ

システム全体で一貫した設計を保ち、一般的な慣習や操作ルールに合わせることで、利用者が迷わずに使えるようにします。この原則に適合するため、デジタル庁では「デジタル庁デザインシステム」（デジタル庁）を整備しています。

関連するWCAG達成基準

- 3.2.3: 一貫したナビゲーション
- 3.2.4: 一貫した識別（同じ要素は同じデザインで提供）

よくある問題の例

- ページごとに異なるデザインやナビゲーションが使われている
- 同じ機能でもラベルやアイコンが統一されていない

改善例

- システム全体で統一されたナビゲーションとUIを提供する
- 同じ操作要素には一貫したラベルやアイコンを使用する

3.6 人間中心の原則

生成AI等の知能・自律型のロボットやシステム等のいわゆるRIAS（Robotics, Intelligent and Autonomous System）を通じ、自律性の高いシステムによるインタラクション（利用者との対話や相互作用）を提供する場合、利用する技術の特性及び用途に照らして、人間の判断を介在させる必要があります。利用者中心の観点からは、以下を満たす必要があります。

- 公正であること（透明性及び説明責任）
- 利用者とシステムとで、適正な役割分担が行われること
- 利用者の制御可能性を確保すること
- 利用者にとって適正なデータを用いること
- 利用者にとって適切なデータ処理が行われること
- 利用者の必要に応じて、相互接続性と相互運用性が確保されていること

RIASの応答を見て、利用者は情報システムの振る舞いに対する自身の理解が正しいかどうかを判断しようとします。

複雑なシステムの一部としてRIASが振る舞う場合、この応答だけでは、どこから取得されたデータを用いているのか、どんなシステムに影響されてその応答をしているのか、利用者には全体像が理解できず、以下のような問題が起きてきます。

- メンタルモデルと実際のシステムイメージの乖離が激しい
- RIASの責任範囲がどこまでなのか理解できない
- 自身のインプットがどこまで伝達できたのか判断できない

本原則の参考になる関連文書は、以下の文書です。

- ISO/TR 9241-810:2020 Ergonomics of human-system interaction
- 「人間中心のAI社会原則」（平成31年3月29日、統合イノベーション戦略推進会議決定）
- 「AI事業者ガイドライン」（2024年4月19日、総務省・経済産業省）
- 「広島AIプロセス」（2023年12月1日、G7合意）
- 「セキュアAIシステム開発ガイド ライン」英国国家サイバーセキュリティセンター（NCSC）が米国サイバーセキュリティ・インフラストラクチャー安全保障庁（CISA）等とともに作成し、内閣府科学技術・イノベーション推進事務局及び内閣サイバーセキュリティセンターが推奨しているガイドライン
- 「DS-920 行政の進化と革新のための生成AIの調達・利活用に係るガイドライン」（2025年5月27日、デジタル社会推進会議幹事会決定）

また、以下のような文書やツールも参考になります。

- 「データ倫理ガイドブック」（「イノベーションを支えるデータ倫理規範の形成」プロジェクト）
- 「The Digital Ethics Compass」（Danish Design Center）

4

ユーザビリティを確保するデザインプロセス

4.1 人間中心設計 (Human Centered Design: HCD)

情報システムを利用する利用者にフォーカスをあてて、利用者の置かれた環境や、利用者の目標、タスクを明確化し、それらに適合したデザインを行うプロセスが人間中心設計 (HCD) です。

以下のようなプロセスに大別できます。

- 利用者の利用状況、利用環境の特定
- 利用者の課題及びニーズの特定 (利用者要求事項)
- 仕様の策定
- 品質の測定及び評価
  - 利用時品質の評価
  - 安全目標に対するリスク評価

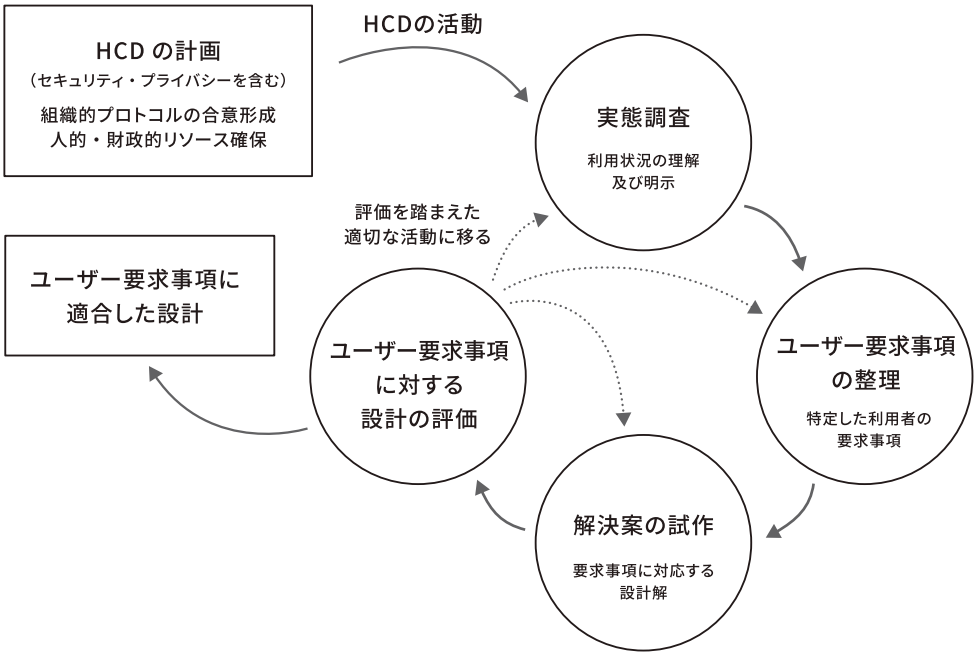

図4.1 HCD 活動のプロセス図。JIS Z 8530:2021 図1 人間中心設計の活動の相互関連性 を元に作図。

HCDのプロセスは、端的にいうと利用者（ユーザー）視点での課題仮説検証、発想支援、そして合意形成のアプローチを組み合わせたものです。プロジェクト内で反復的に実施され、課題解決にあたるチームの発想を発散させたり収束させたりしながら、機能する計画を立て、関係者間で合意を形成していくプロセスです。

注記）HCDの日本語表記は人間中心デザインと呼ばれることもありますが、同じものを指しています。類型化にあたっては、ISO/TR 9241-810:2020を参照しています。

4.1.1 実装前に、利用者による試験を繰り返し実施する「ダブル」V字モデル

システム開発でよく参照されるV字モデルでは、要求定義・設計の各段階と試験を紐づけて考えます。要件定義を満たしているか受入テストで確認する、といった具合です。しかしここで問題になるのは、受入テストで重大な問題や設計ミスが発覚した場合、後戻りできないということです。

要件定義や基本設計の段階でユーザー要求を満たしているか確認できれば、設計を見直すことは十分可能です。そこで、実際の実装の代わりとなるプロトタイプを作成して実際に利用者の評価を受けてから次のステップへ進んでいくのが人間中心設計の基本的な考え方です。

V字モデルで説明すると、「実装後にチェックする」V字の中に、さらに小さな「デザインの段階でチェックする」V字があるのです。とはいえ、この段階ではまだ実装されて動くシステムは存在しないので、紙芝居のようなモックアップを簡易に制作して評価を行います（一般的にプロトタイピングといったときは、このプロセスを指しています）。

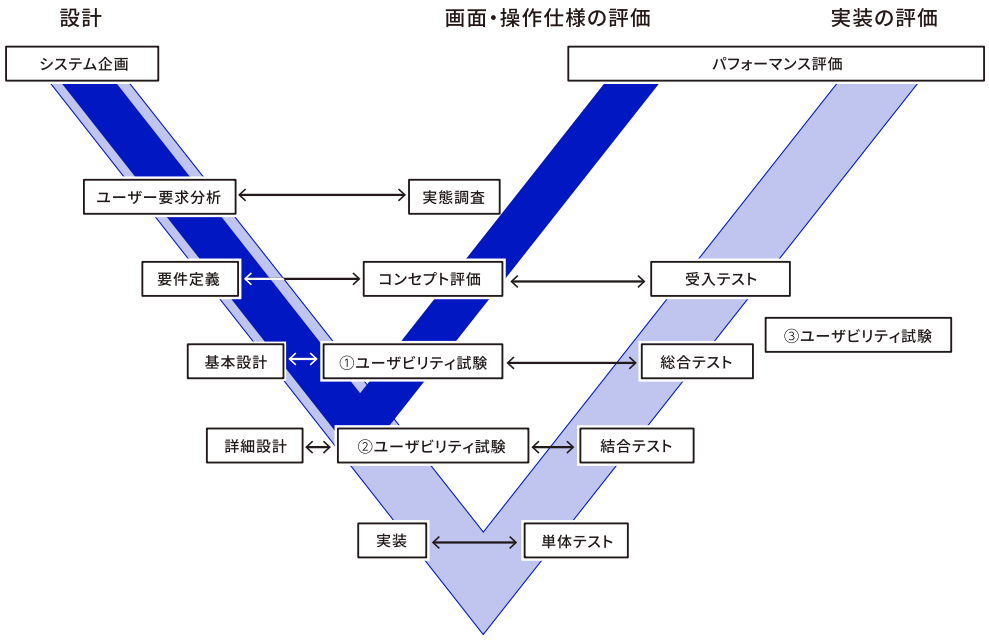

図4.2 デジタル・ガバメント推進標準ガイドライン実践ガイドブックの解釈を元に、ダブルV字モデルを定義。

4.2 ユーザビリティ設計活動で解決すべき課題

では実際に実装に移る前までに、どんなことを確認しなければいけないでしょうか。人間中心設計は様々な課題解決に適用されていますが、情報システムを対象とした場合、解決すべき課題は以下のように類型化することができます。

- 個人の活動や営みにおける課題に適合しているかどうか [4.2.1]
- 情報システムの操作や指示がきちんと行えるか [4.2.2]
- その情報システムを取扱う組織の特性に左右される課題に適合しているかどうか [4.2.3]
- 法律・ガイドライン・社会規範に適合しているかどうか [4.2.4]
- セキュリティの確保、プライバシーの保護、使用エラーの防止等のクリティカリティへの対処ができるか [6]

4.2.1 個人の活動や営みにおける課題への適合

端的にいえば「実用に耐えるのかどうか」です。オンライン手続であれば手続が完了できること、業務システムであれば業務が円滑に実施できること等、情報システムの利用者や二次利用者（間接的に影響を受ける人）の課題解決や目的達成に寄与するのかどうかを指します。システムがいかに使いやすくとも、前後のタスクや状況とうまく接合されていなければ台無しだ、ということでもあります。

以下のような状況を考えてみると分かりやすいでしょう。

- 真っ暗な場所では紙の指示は読めません。
- 手が塞がった状況で指紋認証は使えません。
- 通勤途中の電車の中では、予約確認の電話はできません。
- 表計算ソフトで入力したデータを取り込めない統計システムは役に立ちません。

4.2.2 情報システムの操作や指示がきちんと行える状態

システムの操作を通じて、以下の状態が確保されていることを指します。

- 利用者が適切に情報を取得・理解できること
- 利用者が適切な指示をシステムに与えられること

一般的に「使いやすさ」といったとき、多くの人が思い浮かべるのはこの「操作のしやすさ」のことでしょう。言い換えると「思ったとおりにシステムが動く」ことです。第3章で紹介してきた各原則を満たせるようにする必要があります。

- 利用者が適切かつ効率的にシステムの利用に対するメンタルモデルを構築できる [2.5.2]
- 適切なメンタルモデル構築のために、安全な試行錯誤の機会が与えられる [3.4.4]
- 利用者の指示とシステムのプロセスの対象・範囲・挙動が一致している [3.4.1]
- 適切なフィードバックが与えられる [3.4.2]

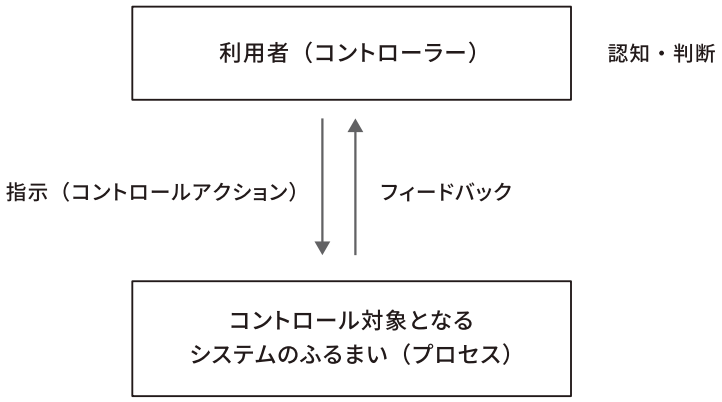

図4.3 指示とフィードバックでつながる利用者とシステムのふるまいの関係

UIコンポーネントレベル、タスクシナリオベースで各原則を満たし、更に以下のような場合にハザード（利用者等のステークホルダーにとって意図しない結果）を起こさない仕組みになるよう設計活動を行います。詳しくは6章以降で再度紹介します。

- 間違った指示が行われたとき
- 必要な指示が行われないとき
- 必要な指示が適切なタイミングで実施されないとき
- 手順が省略されたとき
- 手順が想定した順序から逸脱したとき

ここでは機能の信頼性が高くても、システムの安全性が高いとは限らない点に注目しましょう。ブレーキの性能がいくら高くても、適切なタイミングで踏まれなければ、安全は保たれません。

4.2.3 組織的課題への適合

情報システム同士が連携したり、それぞれのシステムを操作する利用者（多くの場合オペレーター）がいる場合に、適切な情報伝達が行われないと、各システムのユーザビリティが低下してしまうことがあります。

- 社会や多様なステークホルダーに伝達すべきシステムの役割や振る舞いが明確である
- 社会や多様なステークホルダーに対する組織の責任分界点が明確である
- 各組織の個別判断が、他のシステムに影響を与えるとき、どのような影響が及ぶか明示されている（オペレーターに理解されている）

4.2.4 法律・ガイドライン・規範への適合

法律・ガイドラインの遵守という観点と、情報システムを導入することで社会や規制に与える変化への対応について検討します。

- 遵守すべき法律、ガイドライン、規範等を満たしているかを確認します
- 法律・ガイドラインと情報システムの相互関係において、それぞれが安全を担保するための制約として機能するかを確認し、機能しない箇所があれば是正する必要があります

4.3 アクセシビリティ設計活動と評価

本活動のプロセスについては「DS-671.2 ウェブアクセシビリティ導入ガイドブック」（デジタル庁）の第4章を参考にしてください。

4.4 ユーザビリティのための産業共通様式（CIF）

ユーザビリティを確保するための調査・設計プロセスにおいて作成される成果物に含まれるべき観点や要素は、SQuaREシリーズのうち「ユーザビリティのための産業共通様式」（Common Industry Format for usability: CIF）シリーズ（ISO/IEC 2506x）として整理されています。CIFを用いることで、デザインプロセスの品質を一定に保つことが期待できます。

5

デザイン評価の実施

デザインが有効に機能しているかを評価するには、以下2軸での確認が必要になります。

- 解決すべき課題の設定が適切なのか（課題仮説の評価）
- 課題を解決するための機能や仕様案が適切なのか（解決案のパフォーマンス評価）

本ガイドブックでは主に後者の「できあがった画面（解決案のパフォーマンス評価手法）」を解説します。具体的には、インタビューのように個別の現象を掘り下げていく質的調査と、統計調査のように一定の母集団を対象として行う量的調査があります：

質的調査

- ユーザビリティテスト [5.1.1]
- ヒューリスティック評価 [5.1.2]
- エキスパートレビュー [5.1.3]

量的調査

- アンケート
- アクセス解析
- A/Bテスト

それぞれ、得意とする点と不得意とする点があり、開発中の評価で主に使うのは質的調査です。

	質的調査	量的調査
被験者数	利用者数が限定的でも行える	ある程度の利用者数（母数）がある場合に行う
特徴	特定の利用者群が行動や選択を行った背景や理由、複雑な要因が絡んだ課題を掘り下げて特定できる	特定の利用者集団（クラスター）の行動や選択を特定できる
注意点	得られたインサイトが、どの利用者群を対象にしたものかは量的調査での検証が必要	特定の利用者集団が、なぜそのような行動や選択をするのか掘り下げるには質的調査が必要

それぞれの評価手法で「分かること・分かりにくいこと」があり、何を明らかにしたいのかによって適切な手法を選択し、評価を行っていきます。できるだけ複数の調査を組み合わせるようにすると、調査の妥当性を別視点から検証することもできます。

5.1 代表的なユーザビリティの評価手法

ユーザビリティの質的評価手法には、実際の利用者を招いて行うユーザビリティテストと、専門家による評価（インスペクション評価）があります。操作課題等を出して実際に達成できるか等を評価するユーザビリティテストは「特定の手続が完了できるか」といった特定のユースケースの評価に向いており、システム全体を評価したい場合には、専門家による網羅的な評価のほうが向いています。

5.1.1 ユーザビリティテスト

実際にシステムの利用者に画面案を操作してもらう評価手法です。利用者が初めて機能に触れる場合は、以下の「認知・判断・行動」のステップを適切に達成でき、学習性が確保できているか、あわせて効率性や機能性等についても評価を行っていきます。

- 認知：操作に必要な機能を利用者が認知できるか
- 判断：正しい理解や判断を利用者が行えるか
- 行動：正しいアクションを利用者が行えるか
- フィードバック：情報システムからのフィードバック（アクションの結果起きたこと）を利用者が正しく認知・理解できるか

主要な課題を発見するには（システムの規模や成熟度にもよりますが）、ユーザビリティテストを少なくとも5人以上の利用者に実施することを目標にすると良いでしょう（コラム参照）。ユーザビリティテストを実施するには通常以下のステップで、1ヶ月〜2ヶ月程度の期間が必要です。

- アンケート等による被験者のスクリーニング（適切な対象者の選定）の設計と実施
- スクリーニング後の被験者のリクルーティングとスケジュール調整
- 会場設営と試験実施（オンラインで実施する場合もある）
- データ分析とレポート作成

ユーザビリティテストに使える時間は、被験者1人あたり1時間〜1時間半程度なので、網羅的にすべてのシナリオをテストすることはできません。テストできる操作課題は2個〜3個くらいだと考えておきましょう。システム全体を網羅的にチェックしたい場合は、次項で紹介するヒューリスティック評価やエキスパートレビューと組み合わせることで、より効率的・効果的な分析ができます。

ユーザビリティテストの実施人数

ユーザビリティテストで発見できる課題の数や重要性は、アプリケーションの規模や開発のフェーズ、テストの品質や被験者の性格等にも左右されますが、n人の被験者によって発見できる課題の数Nは以下の数理モデルを用いて表現できることが分かっています（つまりソフトウェアの信頼性評価と同様に、ユーザビリティテストで発見できる課題の数もポアソン分布（ある事象が決まった時間内に発生する回数の分布）を示すことが分かっています）。

$$Found(n)=N(1-(1-\lambda)n)$$

λは1人の被験者がテストをして発見できる課題の割合。先行研究では様々なシステムをテストした結果のλの平均値は31%とされており、nが5のときに約85%の課題が発見できることになります。こうした数理モデ

ルを応用することで、個別の情報システムにおける課題の総数や、追加すべき被験者数をある程度判断することも可能になります。とはいえこれは、実際に発見できる課題の深刻度等は考慮されていませんし、被験者によっても発見できる課題にはばらつきが生じます。

複数のクラスターを考慮すべきであればそれぞれテストする、単純に被験者数を増やす等、プロジェクトの状況によって対象人数を検討するのが適切です。

5.1.2 ヒューリスティック評価

ヒューリスティック評価は、人間工学の原則に基づいた専門家による評価を行う評価手法です。原則はいくつか考案されていますが、ユーザビリティ研究の第一人者であるヤコブ・ニールセンの提唱によるものが有名です。複数の専門家が同じ対象の評価を行い、あとで集約して課題の優先度を判定していくことで信頼性を高めることができます。

5.1.3 エキスパートレビュー

（そのデザインのプロジェクトに関っていない）デザインの専門家（評価対象のプロダクトの専門知識や経験のある人を選ぶことも推奨されます）が評価を行うもので、ヒューリスティック評価をより簡易に実施し、その専門家の知見や、過去のユーザビリティテストの結果等を踏まえたレビューを行う評価手法です。デザインの世界における「セカンドオピニオン」だと考えると分かりやすいかもしれません。複数のデザイナーが働いている組織では、簡易評価として日常的に実施されています。

5.1.4 認知的ウォークスルー

初めてシステムを利用する初心者の視点に立って、必要な操作方法を直感的に理解でき、次のステップに迷わず進めるか、システムの「学習のしやすさ」をシミュレートする評価手法です。そのため、操作課題は必ず「受付を完了できるか」といったタスクベースで行われることになります。操作のステップごとに、大きなゴール（目標）に向けて利用者が適切に判断・行動ができるかどうかを検証していきます。

- 認知：操作に必要な機能を利用者が認知できるか
- 判断：正しい理解や判断を利用者が行えるか
- 行動：正しいアクションを利用者が行えるか
- フィードバック：情報システムからのフィードバック（アクションの結果起きたこと）を正しく認知・理解できるか

認知的ウォークスルーは、一般的に慣れ親しんでいるとは言えない機能をテストするときに有効です。ウェブサイトのように、他に類似しているUIに利用者が慣れていると考えられる場合は、ヒューリスティック評価 [5.1.2] をするほうが効果的なことがあります。

5.2 リプレゼンテーションの確保

図5.1 性別、年齢、障害の有無等が異なるさまざまな人たち。

行政サービスは他のサービスによって代替不可能である場合が多く、多くの人に影響があります。このため、社会や対象地域を構成するさまざまなグループ（性別、年齢、文化的背景、障害の有無、社会的立場等）の視点を、適切に反映していること（リプレゼンテーション）は、行政サービスの前提として特に重要です。

本ガイドブックでは詳しく取り上げませんが、行政サービスのみで改善できる社会課題はごくわずかです。どのようなシステムがあると、多種多様なステークホルダーの協働のネットワークがうまく機能するかを確かめながらプロジェクトを実施する必要があります。

ユーザビリティの向上という本ガイドブックの趣旨に限定してみたときには、リプレゼンテーションの確保は以下の観点から特に重要です。

- 利用者グループによる特定の機能や情報アクセシビリティに対する要求の差異の把握。
- 個別最適化・全体最適化の判断を行い、より多くの利用者の利便性や効率性の向上に寄与する機能・非機能要件の工夫。

事例で考えてみましょう。中小企業向けの補助金申請システムで考えてみます。利用者を「事務局担当者」と「補助金申請者」で2グループに分けてみました。

	利用者要求	ポイント
事務局担当者	補助金交付の妥当性を判断するための情報を「網羅的かつ正確に」入力してほしい。また、毎日何百もの申請を処理するので、効率性を重視してほしい	・公正な審査の実施 ・効率的な審査の実施
補助金申請者	なるべく負担のない形で、補助金の交付申請ができるようにしてほしい。いつどのような補助金が予定されているのか、予めわかるようにしてほしい	・申請コストの見積 ・資金繰りの判断

一見すると妥当なグルーピングや利用者要求の記述ができているように見えますが、残念ながら十分ではありません。例えば、補助金の申請を初めて行う企業と、普段から補助金を活用している企業では、課題意識やニーズはまったく異なるものになるからです。企業の規模によっても、制約は変わるかもしれません。数名規模の企業では補助金専任の担当者を置くことはできませんが、大企業では専任チームを作ることができます。

このように、利用者のグルーピングにあたっては、利用者の役割だけではなく、熟達度や利用頻度、利用の文脈、関わる人の数等に配慮することによって情報システムが利用しやすくなったり、課題が解決したり、生産性が向上するポイントを丁寧に把握する必要があります。

- 利用者の役割や関わり方
- 利用者の熟達度
- 利用頻度
- 地域や利用される場所
- 割り当てられるコスト（時間・経済・人的資源等）
- ニーズや課題

5.2.1 話を聞きやすい利用者グループのニーズを優先していないか

情報システムの担当者からすると、直接やり取りを行うことも多い事務局の担当者の話のほうを聞きやすく、補助金の申請を行う中小企業の担当者に話を聞くのはやや困難です。補助金申請システムを使う前に申請を諦めてしまった企業に対するアプローチは更に困難なものになるはずです。こうした利用者へのコンタクトのしやすさの違いがバイアスとして働いてしまうと、意図せず排除されてしまう利用者が出てきます。

- 世代別や業界別でも、異なるニーズがあるかもしれません
- 都市部であればすぐに用意できる書類も、地域によっては準備が困難かもしれません
- 障害のある経営者や高齢の経営者、あるいは学生起業家が申請をするかもしれません

多様なステークホルダーが自らの要望や課題を適切なタイミングで言語化できるように、プロジェクトに安全に参加できる機会を設計することで、バイアスを低減できます。多様なステークホルダーによってプロジェクトの妥当性がチェックされることには、社会的信頼が醸成されるようにプロセスの質を高める効果も期待できます。可能な限り、プロジェクト初期から多様な利用者の視点を導入するようにすること、現場に出向いてできる限り多くの利用者に接することを心がけましょう。

5.3 機能案・画面案の評価を実施するタイミングと手法の選定

利用者のニーズを網羅的に把握したり、ユーザビリティの評価を行うべき時期と手法は、プロダクトの成熟度や評価対象によって決まる側面もあります。例えば、A/Bテストは以下のようなケースではうまく評価ができません。

- 改善点を大幅に反映したナビゲーション全体の試験
  いろいろな利用シナリオがありうるので、テストする箇所が多すぎます
- 年間100件くらいしか申請がない
  アクセス数が極端に少ない画面では、有意差が生じるのに数年かかってしまいます

ここでは、「古いシステムを総入れ替え（リプレース）する場合」と「新しいサービスを企画する場合」で、設計した画面や機能の有用性を、どのように評価すればいいのか紹介します。

5.3.1 古いシステムをリプレースする場合の評価

古くなった、多くの課題がある情報システムをリプレースする場合には、既存のシステムに対してのアセスメント（評価）をまずは行う必要があります。どのような手法を用いるべきかは、以下のような要素を加味して決定していきます。

- 利用者目線での課題をどの程度把握できているか
- 新しい技術で作り直したときの変更の大きさ
- 業務等の利用者の活動への対応状況

現状把握をする際に、既に大きな課題が見えているのであれば、ユーザビリティテストを実施するのは有効ではないことがあります。開発チームにとって既知の課題に被験者がつまずいてしまい、未知の課題を抽出するのが難しい傾向にあるからです。そのようなときは、専門家によるエキスパートレビューを実施したほうが費用対効果に優れています。

しかし、これまで利用者の声を聞けていなかったり、アクセス解析等を十分に導入できていない場合は、利用者が何を求めているのか課題の抽出を兼ねて、実際にどのように画面や機能を使っているか見せてもらい、利用文脈（コンテクスト）を理解するための現地調査やデプスインタビュー（1人の対象者の行動や判断、価値観についえ掘り下げて話を聞く定性調査の手法）を実施するのが出発点としては好ましいでしょう。UIの専門家等によるレビューをあわせて実施しながら現行システムの課題整理を進めていき、そのあとで、次項 [5.3.2 新機能・新サービスを開発するときの評価] のステップに進むと効果的です。

5.3.2 新機能・新サービスを開発するときの評価

[5.1 代表的なユーザビリティの評価手法] で解説したリプレゼンテーションが十分に確保され、多様なニーズを抽出・整理した上で、プロトタイプ（解決案）を作成していくことを前提に解説します。

ユーザビリティが適切に確保できているかどうかは、プロジェクトの各段階で反復的にユーザビリティテスト等の評価を行い、検証していく必要があります。同時並行で「後戻りが難しい」開発サイドの検討が進んでいるので、例えば結合テストを実施する最終段階では、発覚した問題のほとんどが解決できないからです。

現代のモダンなフロントエンドの設計では、情報システムのUIを「画像」「リストアイテム」のように頻出要素ごとに抽象化した上で、それらの要素を組み合わせて画面を作っています。フロントエンドの開発では、単に画面のデザインを再現するだけではなく、以下に示すように多角的な検討を行っています。

- APIから、どのようなデータを、どのタイミングで取得・更新できるようにするか
- 通信が中断した場合やデータがない場合どうするか
- 権限ごとに適切な情報を表示できるか
- 適切な時間でデータを取得・表示できるか（キャッシュするかどうか）
- 不適切な入力が行われた場合、どのように対処するか（和暦の入力に西暦で入力された場合等）

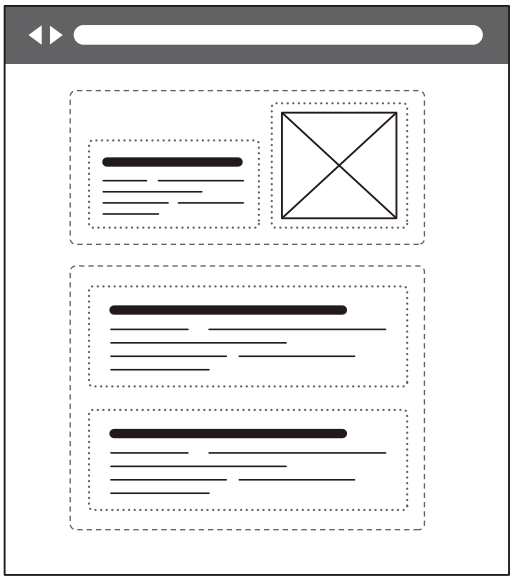

図5.2 UIコンポーネントの分割イメージ。多くの画面を同じUIで表現することで、開発管理とUIの一貫性確保を同時に満たすことができる。

これらの検討・仕様決定・実装の区切りごとに「後戻りできないこと」が増えていきます。デザイン検討においては少なくとも、以下の3段階でデザインの評価を実施するのが理想的です。

- ワイヤーフレームやペーパープロトタイプを作成したタイミング
- HTMLを仮に組み上げた静的プロトタイプを作成したタイミング
- データ連携が行われた動的プロトタイプを作成したタイミング

これらのテストは、アクセシビリティテストと同時期に実施するとより効果的です。

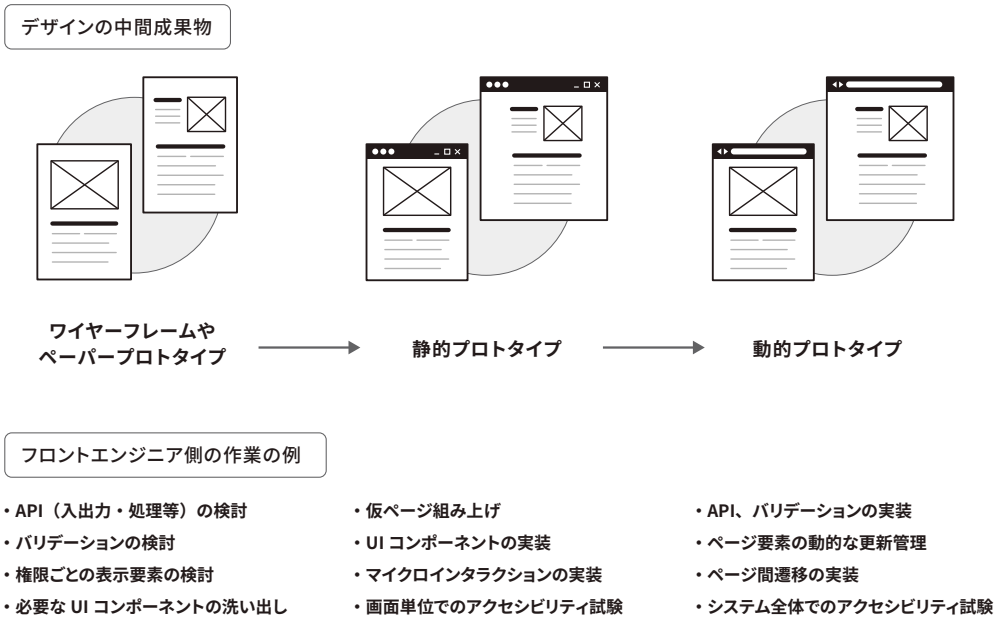

図5.3 デザイナーとフロントエンジニアの作業の対応関係。デザインの中間成果物を起点に様々な技術的検討や仕様策定、実装が行われる。ここでのバリデーションは「正しいデータが入力されることを保障する仕組み」のこと。

ワイヤーフレームで検証しておくべきこと

まず初めのステップ、ワイヤーフレームやペーパープロトタイプ（その名の通り実際に紙で作っても構いませんが、ここではUIデザインツールで作成された紙芝居レベルのプロトタイプのこと）では、

- システムが取り扱う情報、オブジェクトの関係を理解できるか
- 直感的なナビゲーションが提供できているか
- 画面の切り分け方は適切か
- 画面内の要素や機能に過不足はないか
- 画面をみて、利用者がどのように認知するか（認知負荷が低減できているか）

等を検証することができます。一般的に、「大きな画面の区分け方やナビゲーション」は、このあとの工程では（追加コストをかけないと）変更できなくなります。

静的プロトタイプで検証しておくべきこと

静的プロトタイプとは、HTMLレベルまで組み上げたプロトタイプです。ブラウザ画面を用いて、限定的な操作（スクロールを試したり、リンクを押したりする）を試すことができます。静的プロトタイプは、利用者が普段用いている端末やブラウザで再生しやすいため、ワイヤーフレームやペーパープロトタイプよりも現実に則したテストが実施しやすくなります。

- 画面内でのマイクロインタラクション
- 画面内で重要な情報やボタン等の要素を見失わないか
- データ処理の流れを利用者が理解できるか
- UIコンポーネント設計は妥当か

一般的に、このあとの工程では「UIコンポーネントをまたぐ情報の増減」や「APIレベルでの変更」は（追加コストをかけないと）実施が難しくなります。

動的プロトタイプで検証すべきこと

動的プロトタイプとは、実際にデータをつなぎこんだプロトタイプ（モックアップ）です。現実のアプリケーションのようにデータの呼び出しや処理、保存を行うことができ、現実の環境に限りなく近い状況でのテストをすることができます。

- 効率的に操作が可能か
- 十分な性能・速度を発揮しているか

一般的には、この段階で調整できるのは細かいマイクロインタラクション（ボタン等のUIを押したときに発動するアニメーション）程度です。対応できなかった課題は、改修や次期開発で対応していくことになります。

5.3.3 運用を始めたサービスの改善を行う場合

量的調査を実施し、解決策の効果を実際に利用者の反応を通して検証することができるのが、運用フェーズのメリットです。以下のような手法が一般的です。

- アクセス解析
- ヒートマップ
- A/Bテスト
- 利用者向けアンケート・インタビュー

実際に利用している利用者の利用動向を、アクセス解析等の量的調査を中心にして分析し、課題がないかを探っていくことができます。ただ、アクセス解析のような量的調査だけでは「なぜ利用者がボタンを押さないのか」といった質的な情報を抽出できないことに注意してください。

運用段階でデザインの改善案をテストする代表的な手法のひとつがA/Bテストです。複数の異なる解決案を実際に提供し、効果が高かったものを採用していく手法です。A/Bテストは、ボタンの配置や色といった、細かい改善をテストし、PDCAサイクルを回していくのに向いています。逆に、画面全体や手続一式を対象にA/Bテストを実施しても、変数が多すぎてどの施策が適切だったかを評価するのは困難です。大きな課題がある程度解決した段階で、A/Bテストを取り入れていくのが良いでしょう。

利用者の声は、わざわざアンケートの実施等を調達しなくてもできるように、フィードバックフォーム等を設置できるようにしておき、手続の完了直後等に感想を聞けるようにしておくと効果的です。「喉元過ぎれば熱さを忘れる」のことわざ通り、利用が完了してしばらくすると、何に困っていたかやどんな感想を抱いたかは忘れてしまうからです。

5.3.4 情報システムを使っていない非利用者等の話を聞く

情報システムの開発者やデザイナーは情報システムを機能させることに注力しがちですが、他にも考慮するべき要素はたくさんあります。

- マニュアル
- 初心者向けチュートリアル
- 周囲で先に使っている人の評価（レピュテーション）
- 情報技術への不信感や信頼感
- 広報

等、様々な要素によって「情報システムを使うか使わないか」は左右されてきます。また、利用者が皆、熱狂的に新機能や更新を追いかけているわけではありません。情報システムがコモディティ化すると、以下のようなケースが出てきます。

- より便利な新機能があるが、ついつい古い機能を使い続けている
- あるサービスのヘビーユーザーなのに、別の（一般的にはより有名な）サービスの存在を知らなかった

こうした人たちの評価を定期的に集めておくことは、情報システムをエコシステムとして捉え改善していく上で非常に重要です。

6

使用エラーのメカニズム

原則でも紹介したように、システムを正しく使おうとしているにも関わらず、予期したのと異なる結果を引き起こしてしまうこと、操作を省略してしまうことで起きるエラーのことを使用エラーといいます。

使用エラーが発生したときには、利用者が復帰できるよう支援をすることが重要です。[3.4.4 ユーザーによる学習性の確保] で例示した「UnDo機能」は意図しない入力をしてしまったときに、利用者が復帰できる手段の例です。

なお、こうしたエラーは、システムのメッセージを正しく理解できる利用者ばかりが引き起こすとは限りません。知識がない人が正しくない使い方をしてしまうことや、「猫がキーボードの上で寝ていて、延々とEnterキーが押されていたとき」や「赤ちゃんが親のスマートフォンを握りしめているとき」等のイレギュラーなケースが発生することがあります。こうした通常の「使用」を超えた使われ方を「異常使用」といいます。情報システムを開発する際は、使用エラーと異常使用、両方の対策が必要です。

- 使用エラー
- 異常使用の中で予め対策を取れるもの（合理的に予見可能な誤使用）

機器やサービスの設計者が想定している利用方法で使用したときに安全であることはもちろんですが、異常使用があったとしてもハザードが起きないように、システム全体で安全をコントロールしていく必要があります。また、リリース後には実際にどのように使われているのかを確認し、「想定していなかったものの、正しい使い方」等の実情に沿った対策を実施していくことも大切です。

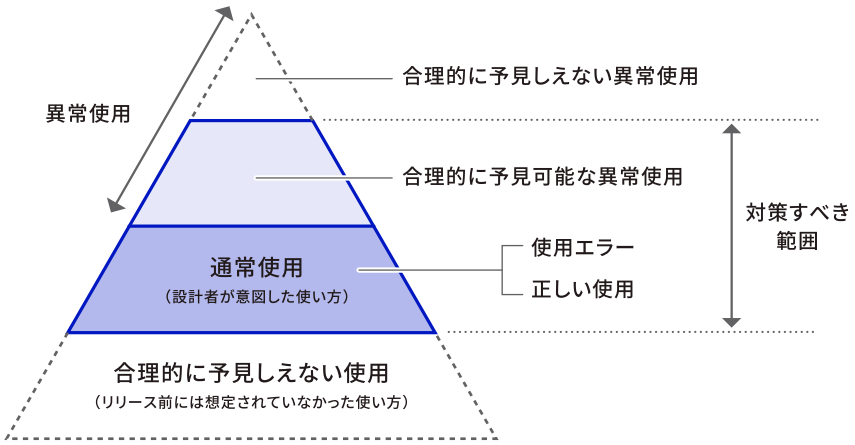

図6.1 「使用エラー」の位置づけ。詳しくは JIS T 62366-1:2022 図 A.4を参照すること。

[3.4.5 ユーザーによる制御可能性の確保] でも見てきたように、一般的に操作はマルチタスクであり、操作をしている最中に利用者に影響を与えている要素は多数あります。シンプルで線形的な操作手順が、そのまま実行されることはほとんどありません。また、以下のような要素にも利用者は大きな影響を受けます。

- 利用のコンテクストや環境
- 作業上の制約やプレッシャー
- 体調
- メンタルモデルと実際のシステムイメージとの不一致
- 組織文化

使用エラーの多くは、実際には「機械側の原因」や「組織的・環境的要因」によって引き起こされます。使用エラーへの対処策を練るときに重要なのは、人をコントロールしようとする（例えば手順やマニュアルで縛る）のではなく、利用環境を含めたシステム全体に対する安全をコントロールしようとすることです。使用エラーを安易に個人の「判断の問題」として結論づけると、思わぬ二次被害を招きかねません。

なお、「どれくらいのリスクが許容されうるか」は社会情勢の変化や、製品の普及の度合い等によって変化します。ポケットベルや固定電話の使い方を若年層の人たちは知らないかもしれませんし、それまでは許容されていたことが、重大事故の発生等によって許容されなくなることもあります。

使用エラーに関連する規格、参照しておくことを推奨する主な規格は以下の通りです。

- ISO/IEC Guide 51:2014 Safety aspects — Guidelines for their inclusion in standards　JIS Z 8051:2015
- JIS T 14971:2020 医療機器―リスクマネジメントの医療機器への適用
- JIS T 62366-1:2022 医療機器―第１部：ユーザビリティエンジニアリングの医療機器への適用
- JIS T 60601-1-6:2023 医用電気機器-第 1-6 部:基礎安全及び 基本性能に関する一般要求事項-副通則:ユーザビリティ

6.1 代表的な使用エラー

使用エラーは、「どんなことを間違えたのか」という結果（外見＝行動主義心理学的分類）や、その原因となっている心の働き（認知心理学的分類）で分類してみると分かりやすくなります。まずはスウェインとグートマンの整理（Swain & Guttmann, 1983）を始めとした、「どんなことを間違えたのか」を軸とした使用エラーの類型を見てみます（後述するリーズンは「なぜ起きたか」という使用エラーの認知的メカニズムに焦点を当てます）。

種類	内容
オミッションエラー （Errors of Omission）	するべきことをしなかったために、起きるエラー
コミッションエラー （Errors of Commission）	想定と異なることをした・行動を間違えて発生したエラー
不要な行為 （Extraneous act）	本来の手順にないことをしてしまったために発生したエラー
手順エラー （Sequence Errors）	本来の手順どおりの順序で作業ができなかったために発生したエラー
タイミングエラー （Timing Errors）	タイミングが早すぎる・あるいは遅すぎるために発生したエラー

こうしたエラーには、慣れていないときや疲労、ルーチン等で「つい・うっかり」引き起こしてしまう場合と、熟練者が「あえて」やってしまう違反行為とがあります。

6.2 意図しない使用エラーと、意図的な使用エラー

人間は、外部環境からの刺激を認知し、それに基づいて判断し、具体的な行動をとるという一連のプロセスに従って日常的に作業を行っています。この「認知→判断→行動」の過程を構造的に理解することで、使用エラーが、どこで・どのように発生するのかを明確に可視化することが可能になります。

使用エラーが「認知・判断・行動」のどの段階で発生したのかによって、端的にいえば「うっかり」やってしまったのか「あえて」やってしまった違反行為なのかで、対策は大きく異なってきます。

- 過失に基づくもの（うっかり）
  - 記憶の錯誤
  - 認知ミス
  - 判断ミス
  - 行動ミス
- 知識や技量の不足
- 意図的・故意的なもの（あえて）
  - 近道行動
  - 省略行動
  - 違反・不遵守

6.3 「うっかり」や「あえて」はどのようなときに発生するのか

人が「認知し、判断をし、行動する」プロセスをモデル化することで、「うっかり」や「あえて」がどこでどのように起きるのかも視覚化しやすくなります。こうした行動形成プロセスのモデルで有名なのはラスムッセンのSRK（Skill-Rule-Knowledge）モデル（Rasmussen, 1983）です。J・リーズンは、SRKモデルやスキーマ理論（Norman & Bobrow, 1976; Rumelhart & Ortony, 1977）、ノーマンのATS（Activation-Trigger-Schema System）（Norman, 1981）などをベースとして、GEMS（Generic Error-Modelling System）（Reason, 1987）を開発し、以下のような「なぜ起きたか」に焦点を当てたエラー分類を提案しました。

行為レベル	種類	例
Skill Base	スリップ （行動段階）	意図せず発生。エレベーターに駆け込んできた人がいるときに、慌てて「閉じる」ボタンを押してしまう。
	ラプス （記憶の錯誤や失念）	意図せず発生。薬の飲み忘れ等。
Rule Base	ルールベースのミステイク （判断が適切に行われない）	意図して発生。与えられた状況に対して(正しいと思って)選択した行動が間違えている。
Knowledge Base	知識ベースのミステイク （認知処理が適切に行われない）	与えられた状況に対する認知が不完全（正しく状況を理解できていない）で間違った行動を取ってしまう。

スリップやラプスは本人が意図せず無自覚に発生してしまうエラー（うっかり）です。これに対して「ミステイク」は「自覚的」に（あえて）発生するエラーです。なお、エラーではないものの対策が必要なものに「違反」があります。正しい手順を知っているが、自覚的に逸脱する行為です。

人の認知行動が熟練してくると、目標を達成するための判断・行動の一連の流れが自動化され、あたかもプログラムであるかのように意識しなくてもできるようになってきます。これをスキーマと言います（ぼーっと考え事をしていても家に帰ってこられるのはスキーマが構築できているおかげです）。ATSやGEMSでは、スキーマを起動させる刺激と働きに注目します。スキーマは単独で機能するわけではなく、他のスキーマを刺激したり、複数のスキーマが自動的に活性化したりします。スリップは、適切なスキーマが存在しない（知識不足）か、誤った刺激が波及してしまった場合に発生すると考えます。次節からは、典型的な使用エラーを見てみましょう。

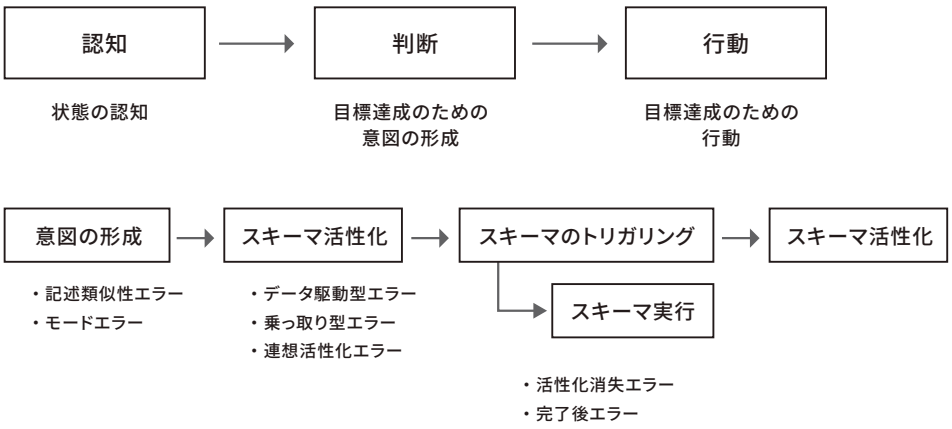

図6.2 認知から行動までのどの段階で、どんなエラーが発生しやすいか。

6.3.1 モードエラー（状況を正しく認識できない）

判断に必要な状況や、状況の変化を適切に認識できなかったことによって、いわゆる「モード選択を間違えたとき」に起きるエラーです。

- スライドショーモードで編集しようとしてしまう
- サイドブレーキがかかっているのにアクセルを踏んでしまう

6.3.2 記述類似性エラー（ゴールがあいまいで、似たものがあるとき）

似たものがあったときに、正しい行動を間違った対象に対して行ってしまうことです。

- ミニトマトのヘタを取り分けているときに、ヘタをいれる袋にミニトマトを入れてしまった

6.3.3 データ駆動型エラー（目の前のことに引っ張られる）

いつもやっている行動（熟練行動）が、与えられた刺激に反応してすぐ実行されてしまうことで発生するエラーです。

- 電卓アプリで、電話番号を押してしまった

6.3.4 乗っ取り型エラー（いつもの行動に引っ張られる）

直近で実施した（あるいは習熟している）共通した要素を持つ現象や概念（スキーマ）にとらわれてしまうエラーです。

- 郵便局に寄ってから帰ろうと思っていたのに、ぼーっと歩いていたらそのまま帰ってきてしまった
- 顔を洗おうと水をだして、つい石鹸を手にとってしまった

6.3.5 連想活性化エラー（頭で考えたことに引っ張られる）

連想しやすい概念が、たまたま頭の中で呼び出されてしまったために引き起こされるエラーです。

- いうべきではないと思ったことの方を口に出してしまった

6.3.6 活性化消失エラー（物忘れ）

行為の元になった意図が消失してしまい、なぜその行動をしようとしていたのか分からなくなるエラーです。

- 辞書を開いた途端、何を調べようとしていたのか忘れた

6.3.7 完了後エラー（やり残しエラー）

メインタスクの完了後に実施すべきタスクを忘れてしまうエラーです。

- コピー機に原稿を置き忘れた
- ATM にキャッシュカードを忘れた
- オンライン会議を始めるのに気を取られて録画を忘れた

例えば、コンビニの証明書交付機能付きマルチコピー機では、マイナンバーカードを取り出さないと次のステップに進めないようになっています。これは完了後エラーを防止するための仕組みです。

6.4 エラーチェーン（事象連鎖）

重大な事故や障害は、単一の原因によって発生するのではなく、複数の小さなミスや不具合が連鎖的に積み重なった結果として生じる――このような考え方に基づく概念モデルを「エラーチェーン（事象連鎖）」と呼びます。この考え方を視覚的に示した代表的なモデルとしては、ハインリッヒ（Heinrich, 1931）のドミノモデルやリーズンのスイスチーズモデル（Reason, 1990）が広く知られています。

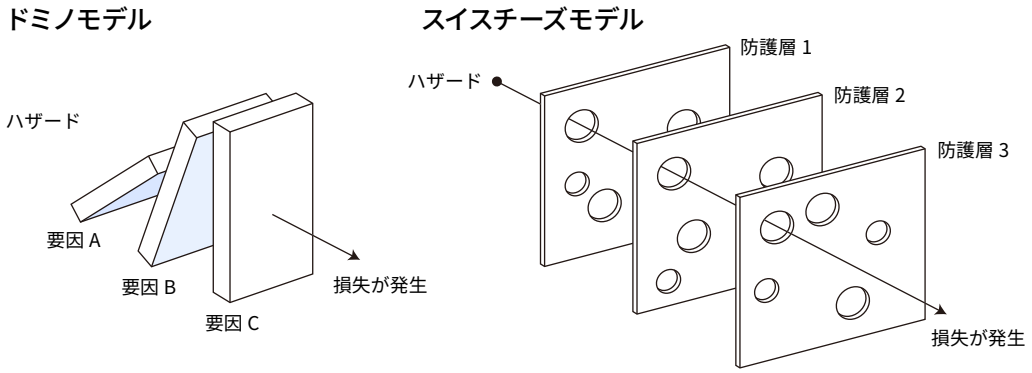

図6.3 チェーンモデルの基本的な考え方を視覚化したドミノモデルと、スイスチーズモデル。

- ドミノモデル
  作業者の不注意や教育不足といった初期要因が、環境的・組織的要因を介して事故へと至る一連の因果関係を、倒れるドミノの列になぞらえて説明します。

- スイスチーズモデル
  スイスチーズモデルは多層的な防御システムを構成している各々の防御システムに（万が一）“穴”（＝防御の弱点）が空いていると仮定しても、それらの穴が同時に重ならなければ、事故は発生しないという構造を示しています。これは、組織的・技術的・人的な多重の防御層とエラーの関係を強調するものです。

重大事故の発生を防ぐためには、各段階で生じるエラーの連鎖を早期に断ち切ることが効果的だ、というのが基本的な考え方です。例えば、フェイルセーフ [8.3.2] やフールプルーフ [8.3.3] といった安全設計の考え方、あるいはリスクアセスメントの基本的な構造は、このエラーチェーンモデルの前提に立っています。特にユーザビリティの観点からは、システムや UI 設計の中でユーザーのエラーが他の要因と連鎖しないようにする、または早期に検出・是正できる仕組みを組み込むことが求められます。

6.4.1 エラーチェーンモデルの限界と展開

ご注意いただきたいのは、エラーチェーンモデルは「多層防護」の重要性を説くのではなく、むしろ「多層防護」の限界や難しさを示すために用いられている点です。多層防護が行われているような巨大で複雑なシステムでは、エラーの発生が検知されにくく影響範囲も分かりにくいため、組織文化レベルで内在されている過ち（例えば管理者レベルでの不備）等が見過ごされる傾向にあります。

特に、以下のようなケースにおいては、「多層防護」モデルに基づくリスク分析や対策が十分に機能しない場合があります（ペローによる整理がよく知られています）。

- 原因と結果の関係が直線的ではなく、因果関係の特定がうまくいかないとき
- 分析や判断が主観的になりやすく、客観的な根拠に乏しいとき
- 事象間の因果関係を（特に専門家が）複数説明することができるために、恣意性や専門家自身のバイアスを完全に排除できないとき
- 扱うシステムが多層的・動的・非線形であり、非常に複雑なとき
- ステークホルダー構造が複雑で、全体像の把握が困難なとき
- 組織課題や伝統のような非技術的な側面が重視されてしまうとき

こうした状況では、エラーや事故の発生を「単一の線形因果関係の結果」として捉えるのではなく、システム全体の構造、相互作用、組織的文脈等を含めて捉える視点が求められます。そのため、近年ではよりシステミックなアプローチに基づく概念モデルやリスクアセスメント手法の開発が進められています。例えば、STAMP（Systems-Theoretic Accident Model and Processes）[9] や FRAM（Functional Resonance Analysis Method）等の手法は、複雑な社会技術システムを対象とした新しい安全評価モデルとして注目されています。

とくに原子力、航空、医療等の複雑系を扱う分野では、エラーチェーンモデルのみで安全性を語ることは少なくなっており、多層的な要因を同時に扱える枠組みの必要性が高まっています。

7

使用エラーの可視化・分析

ここでは使用エラー等の要因の可視化・分析の基本的な手法をいくつか紹介します。近年の複雑なシステムでは、これだけで対応するのは困難になってきていますが、基本的な分析と対策を行っていくこと自体は欠かせない取組です。本章で紹介する手法は人間中心設計でよく用いられる手法とも親和性が高く、デザイナーが取り組みやすいアプローチといえるでしょう。

7.1 使用エラーの要因分析と対策

7.1.1 基本的な調査・分析手法を知る

本節で紹介する手法は人間中心設計のプロセスでも頻繁に利用されています。どの手法にも長所・短所があり、現実にはデザインのプロセスにおいて最適なものを選択・組み合わせていく必要があります。本節では基礎的な分析ツールを2つ紹介します。

- 5 Whys分析（過程関連性分析）
- SHEL分析モデルとSHELL分析モデル（要因分類型分析）

7.1.2 5 Whys 分析

根本原因を探るために「なぜ起きたのか」を逆順に「5回」（を目安として）分析していく方法です。必ず5回やらなければならないわけではなく「真因と思われる原因まで突き止める」のが目標です。ポイントは「単に5回なぜを繰り返す」のではなく「掘り下げられるかどうか」です。掘り下げた結果が「真因」かどうかは、以下のような基準で判断します。

- その原因に対する対策が繰り返されうるものの場合、それは真因ではない（例：「うっかり忘れ」の原因を人に求めると「指差し確認をする」のが対策になりますが、再発を防げません。この場合真因を「うっかり忘れに対応できなかったバリデーションの不備」に置くと対策では再発を防げます）
- 他責にせず、対策が立てられうる真因を探す（例えば「雷が落ちたから」というのは雷自体を防ぐことができないという意味において、他責的思考です）
- 現実に対応できるか十分に確かめる

7.1.3 SHEL分析モデルとSHELL分析モデル

エルウィン・エドワーズによって提唱されたSHEL分析モデル（Edwards, 1972）を、フランク・H・ホーキンスが拡張したのがSHELL分析モデル（Hawkins, 1975）です。人（利用者）とシステム（人の周囲の要素）の相互作用を分析するためのモデルです。使用エラーが引き起こされたときに、その当事者を中心に置き、周囲の環境や人物、物事等との関係性を可視化し、分析を円滑化する手法です。

要素	概要
S（Software）	手順、マニュアル、チェックリスト等の非物理的なシステムの構成要素
H（Hardware）	機器や装置等の物理的なシステムの構成要素
E（Environment）	作業環境や気象条件、組織文化等の環境要因
L（Liveware）	人的要因。特に能力や機能的限界等。当事者と当事者以外の要因がある

SHEL分析では、これらの要素の相互作用の中での不整合や問題の所在を視覚化するツールです。例えば以下のような組み合わせです。

- 不適切なマニュアル（Software）と人間（Liveware）との相互作用がエラーの原因となる場合
- 操作環境（Environment）が原因で、機器（Hardware）が正しく利用されない場合

SHELL分析モデルでは、SHEL分析モデルをさらに視覚的に拡張し、Lを当事者本人とそれ以外の関与者に分けて考えます。

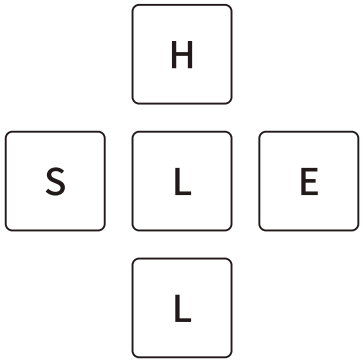

図7.1 SHELLモデルの関係図。真ん中に配置されたLは利用者本人（当事者）、その周囲に配置されたLは他の関係者を指す

SHELL分析モデルでは、要素間の相互作用が視覚的に強調されています。

1. L-H (Liveware-Hardware)
   人間と機械の相互作用。
   例：操作しやすいインタフェースや、誤操作を防ぐ設計。

2. L-S (Liveware-Software)
   人間とソフトウェア（手順やマニュアル）の相互作用。
   例：理解しやすいマニュアルの必要性、システムエラーの予防。

3. L-E (Liveware-Environment)
   人間と環境の相互作用。
   例：適切な照明、騒音やストレスの低減。

4. L-L (Liveware-Liveware)
   人間同士の相互作用。
   例：チームワーク、コミュニケーション、役割分担。

それぞれの状態や関係は刻々と変化していきます（例えば、人間は疲弊します）。ある相互作用のファクターが変化したときには、他の関係にも影響がある（他の影響も考慮しなければならない）と考えるのです。

7.2 規格に基づくアセスメントの枠組み

国際標準には、世界中の専門家の良質な知見が集められています。規格に基づき、アセスメントの基本的な枠組みや手法を網羅的に確認したい場合は、JIS Q 31010:2022、特に表A.3や、附属書A及びB等を参照してください。このうち、HAZOPについては [8.7] で紹介しています。

- ブレーンストーミング
- デルファイ法
- ノミナルグループ技法
- 構造化インタビュー、半構造化インタビュー
- 質問紙調査
- チェックリスト
- HAZOP（Hazard and Operability）スタディーズ
- シナリオ分析
- ハザード分析及び必須管理点（Hazard analysis and critical control points: HACCP）
- 構造化“what-if”技法（Structured “What-IF” Technique: SWIFT）
- 事業影響度分析（BIA）

リスクアセスメントに関連して、参考にしておくと良い規格は次の通りです。

- JIS Q 31000:2019 リスクマネジメント－指針
- JIS Q 31010:2022（IEC 31010:2019）リスクマネジメント－リスクアセスメント技法
- JIS Q 0073:2010 リスクマネジメント－用語

7.3 インシデント発生時の原因調査のポイント

サービスや業務の運用中に発生したインシデントに、使用エラーが関与している可能性がある場合には、再発防止と設計改善を目的として、構造的・認知的・組織的な観点から原因分析を行う必要があります。
このとき重要となるのは、インシデントに関与した個人のミスを単純化して捉えることを避けることです。ここで「自分が悪かった」「なぜ〇〇できなかったのか」といった自責的・後知恵的な評価は、原因の本質を見誤るおそれがあります。

後述する CAST（Causal Analysis based on STAMP）[9.3] は、分散的な意思決定が行われる複雑なシステムで、このような後知恵バイアスを回避し、インシデントの因果構造を体系的に明らかにする手法として有効です。CAST では「誰が悪かったか」を問うのではなく、「なぜその時にそうするのが合理的だったのか」を明らかにすることで、予防可能な構造上の課題を抽出するアプローチです。あわせて参考にしてください。

簡易の原因調査を行う場合であっても、以下のようなポイントに注意しましょう。

なるべく多くの情報を収集する

インシデントに関わった関与者について、それぞれの置かれていた状況、認知・判断・行動の経緯、情報システムの振る舞いを時系列順で列挙して事実関係を整理し、視覚化します。

「誰が悪かったか」ではなく「なぜそうするのが合理的だったのか」にフォーカスする

それぞれの関与者にとって、どうしてその判断・行動が最も合理的だったのかを解析します。言い換えると、どのような情報を受け取ったために（あるいは受け取れなかったために）その判断が合理的にみえたのかを分析し、対策につなげます。

どうすれば合理的な選択や、仕組みによる防止ができるか検討する

分析を踏まえて、どのようにすれば、本来するべき判断が最も合理的に映るか検討する。情報システムや、組織体制側でどのような仕組みが導入されていれば、そのような合理的判断を促進したり、本来好ましい状態を確保できたりしたのかも検討します。

局所的に合理的でも、全体として合理的ではないケースを特定する

個々の関与者による分散的な意思決定が行われざるを得ないシステムや、複数のシステムが連携して機能提供が行われている場合は、個々の関与者にとって合理的にみえても、全体として脅威のリスクを高める箇所を特定して対策を行う必要があります。

8

安全設計とリスクマネジメント

ユーザビリティデザインに取り組むべき最大の理由のひとつが、労働災害や個人情報漏えい等のインシデントの防止です。過去、多くの情報システムや工場、産業機械や航空機等の事故が、オペレーターの使用エラーを最終的なトリガーとして引き起こされてきました。令和5年の労働災害発生状況調査では、機械等による「はさまれ・巻き込まれ」の死傷者数は13,928人（死者108人）と報告されています。こうした事故を防いだり被害を軽減したりする上でも、ユーザビリティの確保は非常に重要です。

8.1 リスクマネジメントの主要概念を理解する

8.1.1 脅威（ハザード）とリスクとはなにか

ユーザビリティ上のリスクとは、発生すると、情報システムの提供者、利用者、その他のステークホルダーにとって意図しない結果、なんらかのアクシデントによって脅威（ハザード）がもたらされることを指します。

脅威は必ずもたらされるわけではなく、何らかの事象の結果として引き起こされる不確定性を持っています。そこで近代以降の科学は、脅威がもたらされる可能性をリスクとして捉え、コントロールしようとしてきました。リスクには様々な考え方や捉え方がありますが、本ガイドブックでは、脅威によって発生すると予測される危害や損害、不快感の大きさが、日常生活や社会生活等の活動の中で通常遭遇するものよりも大きくなることを指しています。

8.1.2 リスクアセスメントとマネジメント

本ガイドブックにおいては、ヒューマンファクター（人的要因）に関するリスクアセスメント及びマネジメントについて記述していますが、リスクアセスメント自体はより広範囲の領域を対象として行われるべきです。

操作ミスや認知・認識の誤り、定められた手順の省略等の誤使用によって使用エラーが発生し、重大事故につながることがあります。使用エラーによって引き起こされる結果の中には、プライバシーやセキュリティ上のインシデントを引き起こすものがあります。セキュリティ・バイ・デザインやアクセシビリティ・バイ・デザイン等の関連した活動と共に、ユーザビリティを向上させリスクを明確化する人間中心設計のプロセスを確実に遂行しながら、リスクマネジメントを実施していく必要があります。

8.2 安全設計とリスクマネジメントのプロセス

安全性の評価や対策の基本的な考え方には、決定論的評価（DSA）と確率論的評価（PRA）があります。やり方が決まっていれば同じ結果にたどり着くのが決定論、ランダムに事象が発生し同じ結果にたどり着くとは限らないのが確率論です。 落雷による停電やシステムの故障等の「予想外の出来事」「たまたま起きること」が脅威をもたらすことがあり、それらは確率論的です。

安全評価や対策を行う場合は、決定論的評価（DSA）と確率論的評価（PRA）を、検討対象課題に応じ組み合わせて、リスクを定量的に評価しながら対策を考えていくのが基本的なアプローチです（ただし、情報システムの安全評価・設計においてはこのアプローチだけでは限界があることを9章以降で見ていきます）。

決定論的評価	確率論的評価
確定的事象・脅威が対象	偶発的事象・脅威が対象
特定の条件下でのシステムの振る舞いのプロセスを評価し対策を行う	システムやプロセスが不確実性の影響を受ける場合に、その結果の発生確率を評価し対策を行う
確実性の高い結果が発生するのが前提。特定の操作・入力に対して結果が一意に決まる	リスクや失敗の可能性を数値的に表現する。不確実性が前提

以下に、具体的な流れを参考として示します。

初期分析

- インシデントに関連した事象を抽出する [7.3]
- 使用エラーを分類する [6.1] 〜 [6.3]
- SHELL分析等を用いて、要素間の関係を特定する [7.1.3]
- 問題点を仮説として整理し、リスクの優先順位付けを行う

詳細分析（決定論的評価や確率論的評価）の実施

- システムの特定の条件（シナリオ）や偶発的事象発生等の課題発生条件を詳細化する [8.4] や [8.5]
- 課題発生条件下におけるシステムの限界やパフォーマンスを定量化する
- リスク低減のための設計に向けた要件を明確化する

改善

- インタフェースを改善し、インターロック等の防護策を導入する [8.3]
- 手順を簡略化したり、明確化する
- 教育訓練を実施する
- 環境条件や組織文化の改善を行う

8.3 基本的な安全対策

使用エラーに対処するために、様々な手法や考え方が提案されています。これから紹介する考え方を考慮した設計にすることが非常に大切です（複雑なシステムでは、これらの対策だけでは十分とはいえません）。

8.3.1 エラーレジスタンスとエラートレランス

使用エラーをゼロにすることはできませんが、訓練や意識付けによってエラーの発生を起きにくくすることはできます。これを「エラーレジスタンス」といいます。また、仕組みでの工夫によって、使用エラーが発生しても事故に結びつかないようにすることを「エラートレランス」といいます。

8.3.2 フェイルセーフ

エラートレランスを実現する代表的な設計手法として、フェイルセーフとフールプルーフがあります。エラーが発生しても安全側に収束するように工夫することをフェイルセーフといいます。フェイルセーフには以下のような例があります。

- 倒れると停止するストーブ
- ドアを開けると止まる電子レンジ
- 停電すると自動的に下がる踏切の遮断機

8.3.3 フールプルーフ

フールプルーフとは、全く使い方を知らない人が異常使用をしても問題を引き起こす機能を働かせない仕組みのことです。

- ブレーキを踏んでいないとドライブモードに入らない車のギア
- 両手で操作しないと使えない裁断機
- 異なる端子は接続できない形状になっている

防ぐべきハザードがシンプルで発生条件が特定できていれば、誤使用によりエラーが誘発されそうになっても、問題を防いだり被害を低減する仕組みを作ることができます。

フェイルセーフやフールプルーフを実現するこうした仕組みは「インターロック」とも呼ばれます。

8.3.4 タンパープルーフ

専用工具を使わないと開封できない機械の止めネジのように、適切な権限や対処能力のある人だけが操作できるようにすることを、タンパープルーフといいます。

8.4 イベントツリー分析（ETA）を用いた評価

イベントツリー分析（Event Tree Analysis: ETA）は、使用エラーの初期事象発生から、より重大な事象に至るまでのイベントの連鎖や分岐が発生するシナリオを順方向に検討し、モデル化して分析を行う評価手法です。二分樹で業務等の手順を表現していくことで、エラーが生じたときにどのような事態に至るのかを視覚化することができます。

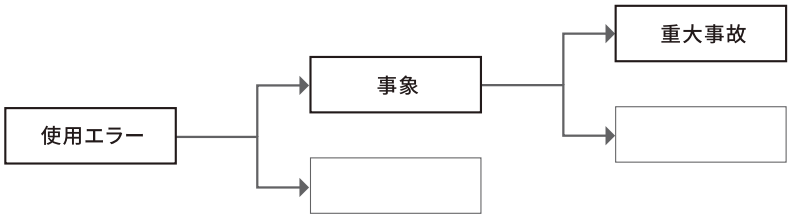

図8.1 イベントツリー図の例。使用エラーの発生を起点に重大事故につながる事象の分岐や連鎖を順方向に検討する。

8.5 フォールトツリー分析（FTA）を用いた評価

フォールトツリー分析（Fault Tree Analysis: FTA）は、重大なインシデントや障害の原因を体系的に分析する手法です。特定の望ましくない事象（トップイベント：Top Event, TE）をツリーの頂点に設定し、それがどのような要因によって引き起こされる可能性があるかを論理的な因果構造として視覚的に表現します。

例えば、「個人情報の漏洩」というトップイベントに対して、「外部からの不正アクセス（事象A）」または「内部者による誤操作（事象B）」という原因が考えられる場合、それぞれをORゲートで接続します。さらに、「誤操作（事象B）」が「研修未実施（事象C）」と「確認手順の欠落（事象D）」の両方によって引き起こされるときは、ANDゲートで接続されます。

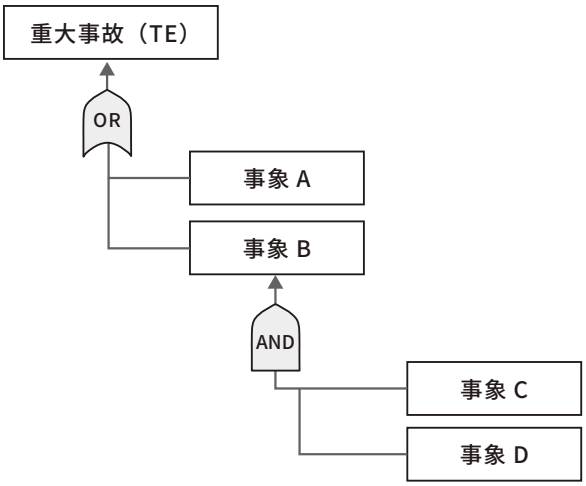

図8.2 フォールトツリー図の例。重大事故(TE)は事象Ａ又は事象Ｂが発生したとき(OR条件)発生し、事象Ｂは事象ＣとＤの両方が発生したとき(AND条件)に発生する。

このようにFTAでは、トップイベントに至るまでの原因構造をツリー状に描き、構造上の弱点を可視化することが可能です。各事象に対して発生確率を設定することで、トップイベントの発生確率を算出することもできます。不確実性を伴う事象については、ベイズ統計（ベイズ推定）を用いて、知識や条件の変化を反映し、動的に確率を更新する場合もあります。FTAは特に次のような目的で活用されます。

- システムや運用フローにおける脆弱性の構造的把握
- 重大事故の発生要因の組み合わせと優先度の特定
- 安全設計におけるリスク低減策の優先順位付け

ただし、FTAは基本的に静的なツリー構造を前提としており、プロセスの動的な変化や認知的な要因の反映には限界があります。そのため、使用エラーや人的要因に焦点を当てる場合には、STAMP/STPA等のより現代的な手法を用いた分析を行っていくこともできます。

フォールトツリー分析と、[6.3] で紹介した使用エラーのメカニズム（スリップ、ラプス、ミステイク）等の観点を組み合わせることで、限定的ではありますが、実践的なリスク評価と対策立案が可能になります。以下に、実際の業務システムにおける分析手順の一例を示します。

8.5.1 FTAを用いた分析例

トップイベント（TE）は重要業務プロセスの中断（データ損失を伴う）とします。この場合、考えられうるエラーには次のような事象があります。

- ユーザーが間違って別のボタンを押す（スリップ）。
- ユーザーが操作手順を省略する（違反）。
- マニュアルが不明瞭で操作手順を誤解する（ミステイク）。

これらの事象を因果構造としてツリー図で表現することで、それぞれのエラーがトップイベントにどうつながるかを可視化でき、リスクシナリオの評価ができるようになります。ただし、現代の情報システムにおけるFTA適用では、以下の制約があることを認識しておく必要があります。

定量的評価の限界

情報システムでは、ハードウェアのような故障率データが存在しないため、確率計算は参考値に留まります。

動的な性質

利用者の学習やシステムアップデートにより、エラー率は時間とともに変化します。

文脈依存性

同じ操作でも、状況や利用者の振る舞いにより異なるエラーが発生します。

ツリー図の作成例

各事象を中間事象とした、AND／ORゲートによって構造化されたFTAを作成します。

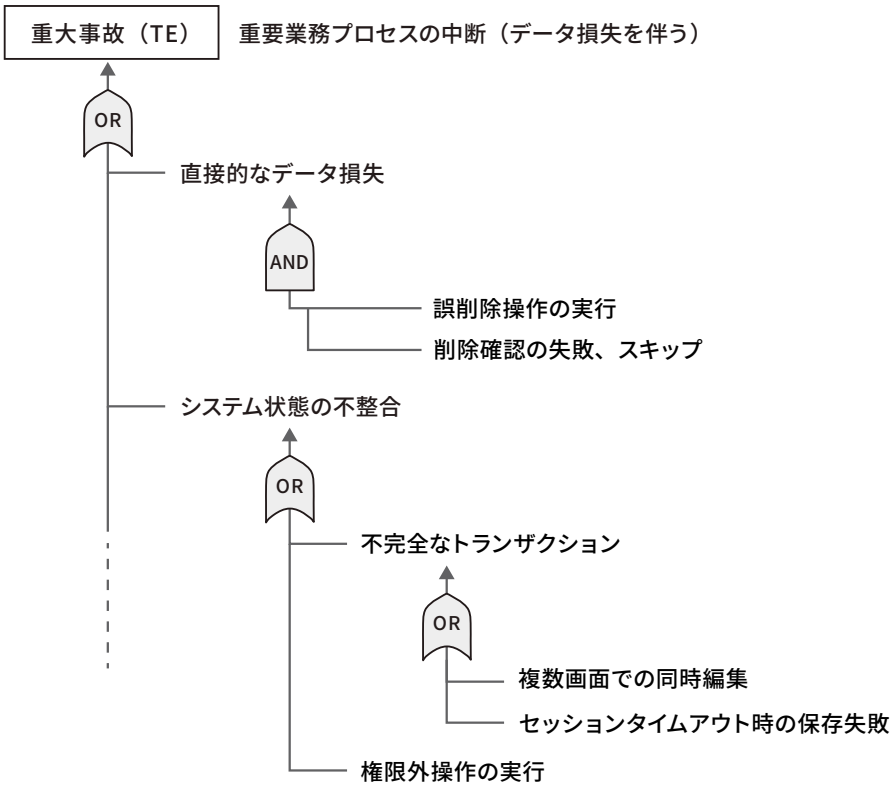

図8.3 フォールトツリー図の具体例。データ損失を伴う重要業務プロセスの中断が発生しうる要因を書き出し、さらにそのような事象が発生する要因をツリーとして書き出していく。

基本事象（使用エラー）の分類と分析

各事象の発生要因やシナリオ、低減策を検討します。表は直接的なデータ損失に関連する分析例です。

エラー	要因例	発生シナリオ	低減策の例
スリップ	削除ボタンの誤押下	隣接ボタンとの混同	ボタン間隔の確保、物理的分離
ラプス	削除確認ダイアログの無意識スキップ	「OK」の反射的クリック	自動バックアップ、重要度別の確認手法の差別化
ミステイク	選択対象の誤認	表示モードの誤解	プレビュー機能
違反	バックアップ手順の意図的な省略	時間圧力、熟達による利便性の追求	自動バックアップ

このように、発生しやすい使用エラーの特定と、その影響の構造的理解、再発防止策の導出が一体的に行えます。FTAは単なる技術的リスク評価にとどまらず、利用者の行動と設計の接点を分析するツールとして活用できます。頻度や影響度等を加味した分析を行うと優先順位の検討に活かすこともできます（次節のFMEAの分析例で、優先順位付けの事例を紹介しています）。

8.6 FMEA (Failure Mode and Effects Analysis)

故障モード影響解析 (FMEA) は、想定されうる故障やエラー等の一覧を作成し、それぞれの影響度や発生確率に基づいて優先的に対策すべき項目を特定する手法です。使用エラーが引き起こすトラブルを事前に抽出し、リスクを定量的に把握する際に有効です。

8.6.1 FMEAの作業手順

作業プロセスの明確化

システムや業務の一連の手順 (分析の対象を細かく区切る) で具体的な操作や手順 (作業プロセス) を洗い出す。

想定されうるエラーの列挙

作業プロセスごとにエラーが起きうるもの (故障モード) を列挙する。

影響評価

エラーによって引き起こされる結果 (波及事故) を評価し、回復に要するコストや発生確率を基準として「致命度」を算出する。表に示した例では、回復コストと発生確率をかけ合わせて致命度を評価しています。

観察による補強 (可能な場合)

実際の業務現場 (例：窓口業務、ログデータ等) を観察し、発生確率を経験的に補正することで、分析の信頼性を高める。

サブプロセス	エラー内容	波及事故	回復コスト	発生確率	致命度
マイナンバーカードを読み取る	読み取ったあと放置する	紛失	5	2	10
ログインする	パスワードを間違える	ログイン失敗	2	3	6
手続を選ぶ	間違った手続を選択する	正しい手続ができない	3	3	9

発生確率：例えば0〜3の4段階とし、0：起こり得ない。1：ほとんどない。2：たまにある。3：頻繁にありうる、等で判定する。

FMEAはもともと航空宇宙、自動車、医療機器の設計品質保証で活用されてきた手法ですが、ユーザー行動に起因する操作ミスや手順ミスにも適用可能です。特に、次のような状況での活用を検討すると良いでしょう。

- 使用時の誤操作が業務停止に直結する場合
- マニュアルやUI設計の見直しポイントを明確にしたい場合
- 現場で蓄積されたトラブル事例を定量的に整理したい場合

8.7 HAZOP（HAZard and OPerability Studies）

FTAやFMEAと並びよく使われるリスク解析方法です。HAZOPでは「ない」「反した」「よりも早く」といった「設計意図・通常の利用方法」からのずれを「ガイドワード」として用意してブレインストーミング等を行い、分析を行う手法です。プロセスを分析の対象とするノードに分解し、ノードごとに分析を行います。

- ノード1：ログイン認証
- ノード2：申請フォーム入力
- ノード3：添付書類アップロード
- ノード4：内容確認
- ノード5：送信・完了

元々は化学プラントの安全設計のために開発された手法なので、ユーザビリティの解析に使うためには、ガイドワード等を改良したものを使います。

ガイドワード	エラー	責任	発生事故
抜かす	ステップを抜かす	使用者・使用状況	手続が完了できない
繰り返す	フォームでクリックを繰り返す	両方	情報が重複して送信されてしまう
忘れる	提出を忘れる	使用者・使用状況	手続が完了できない
急いで	急いで作業する	使用者・使用状況	重要な情報を見落とす
ゆっくり	ゆっくり操作する	使用者・使用状況	タイムアウトしてしまう
中断する	申請を中断する	両方	データが保存できない

注記）ガイドワードには「必要以上に」「足りない」「異なる順序で」「あえて」等、本ガイドブックを参考にしながら独自のものを追加すると良いでしょう。

エラーは、使用者側で発生するものと、システム上の仕様や検討不足、障害等で発生しうるものがあります（例えばタイムアウトエラーや、データの不整合等）。

また、以下のような要因によっても発生させやすいエラーが変わってきますので留意してください。

- 利用する主体の能力や知識の違い
- 利用状況や文脈の違い

8.8 確率論的リスク評価（Probabilistic Risk Assessment: PRA）

原子力発電所や空港の管制システムのような、複数のシステムやオペレーターが互いに連携しているシステムでは、ここまでに紹介した方法だけでは十分な評価・対策が難しい場合があります。また、偶発的な事象は先にも述べた通り、すべてを予測・対策することは現実的ではありません。

こうした状況に対応する手法のひとつが、確率論的評価（Probabilistic Risk Assessment: PRA）です。PRAは、リスク事象の発生確率およびそれに伴う影響を定量的に評価し、リスクに関する意思決定を合理的に支援することを目的としたアプローチです。PRAは特に、原子力施設における安全評価の分野で長年活用されており、複雑なシステムの構成要素がどのように相互作用してリスクに至るかを把握する上ではある程度有効です。

ただ、以下のような特徴を持つソフトウェア開発プロジェクトでは、PRAは導入に一定の制約がある手法であり、本ガイドブックでは深く踏み込みません。

- 統計的・定量的な情報が十分に蓄積されていない
- 対象システムが動的かつ複雑であり、因果関係が明確でない
- 未知のリスクや設計初期段階での不確実性に対応したい
- 人的要因やソフトウェアの設計ミス等定量化が困難な要素を扱いたい
- デザインプロセスへの柔軟な統合が求められる

PRAは「原因と結果のチェーン」を重視しているため、過去の事故データやイベントツリー分析、フォールトツリー分析を使ってリスクを定量的に評価していきます。このため、構成が変化しない情報システムや確率的なイベントを取扱うには強力な手法ですが、時間による変化や外的要因を分析するのは苦手で、ネットワークと連動し、日々更新されるソフトウェアや複雑なヒューマンファクター（人的要因）のリスク評価をPRAだけで実施するのには限界があります。

工場の機械の故障率のようにデータの蓄積がすでにある場合は、PRAは導入しやすい手法です。また、ヒューマンファクター（人的要因）を解析する場合はHRA等の他の手法と併用していくことが効果的です。本ガイドブックではこれ以上詳しくは取り上げませんが、PRAの考え方や手法を知っておくこと自体は有用です。

ソフトウェア開発の段階では、フォールトツリー分析、FMEA、HAZOP等のこれまで紹介したような定性的な分析手法が用いられる他、STAMP/STPA [9] といったシステム理論に基づく手法も開発されています。後続の章をあわせて参考にしてください。

8.9 人間信頼性アセスメント（Human Reliability Assessment: HRA）

人間信頼性アセスメント（HRA）は、機器やサービスの利用時に発生しうる使用エラーに着目し、人間の行動に影響を与える行動形成要因（ストレス、作業環境、記憶力の限界等）を考慮しながら、事故の発生確率を評価し、リスクの低減策を導くことを目的とした評価活動のことです。

従来の確率論的リスク評価（PRA［8.8］）では、人の行動を確率的に扱うことが前提であり、複雑な認知判断や環境要因の影響までは十分に捉えきれないという課題があります。HRAはこうした限界を補い、人的要因をより深く理解しながら、安全設計に反映させるための枠組みです。HRAはいくつかのアプローチに分けられます。

- 第1世代のアプローチ
  主に作業手順に従った行動の逸脱確率を、経験的データに基づいて推定する手法です。代表的なものに、THERP（Technique for Human Error Rate Prediction）やSLIM（Success Likelihood Index Method）があります。
- 第2世代のアプローチ
  人間を意思決定主体としてとらえ、作業コンテキストや認知過程を重視するモデルです。心理学や認知科学の知見を取り入れた手法として、CREAM（Cognitive Reliability and Error Analysis Method）やATHEANA（A Technique for Human Event Analysis）が知られています。
- 第3世代のアプローチ
  Safety-IIやレジリエンスエンジニアリングの影響を受け、「エラーの防止」から「適応能力の向上」へと視点を転換するアプローチです。FRAM（Functional Resonance Analysis Method）やSPAR-H（Standardized Plant Analysis Risk-Human）が知られています。

HRAの発展は、人間の認知処理に対する理解の深化と密接に関連しています。1980年代の認知科学の成熟、そして1990年代の分散認知理論の登場により、ヒューマンファクター（人的要因）、使用エラーの理解は大きく変化してきました。いずれの手法も、次のような基本プロセスに基づいています。

- 分析対象の明確化
- エラータイプの特定
- 適切な分析手法の選択と定量・定性的評価
- 発生確率の算出（データ収集）または推定
- 妥当性の検証と改善策の導出

HRAを適切に実施するには、PRA同様、分析対象としたい作業や手続における過誤データの蓄積が必要になってきますし、組織的な人材育成、安全文化の醸成が前提として決定的に重要になってきます。

8.10 HRAとFTAの併用による使用エラーの分析例

HRAは人的エラーの発生確率や背景要因を評価する手法であり、FTA（Fault Tree Analysis）はリスク要因の論理的構造を視覚化する手法です。両者を組み合わせることで、リスクの定量評価と因果関係の構造化を同時に行うことができます。以下に、病院における薬剤投与業務を例に、HRAとFTAの併用による分析の手順を示します。

8.10.1 使用エラーの分析と確率評価（HRA）

病院の薬剤投与システムにおいて、以下のような使用エラーが想定されるとします（※確率は例示です）。これらのエラータイプ（発生要因ベース）には、[6.3] で紹介したラスムッセンのSRK（Skill-Rule-Knowledge）モデルや、J・リーズンのGEMS（Generic Error-Modelling System）を基盤として用います。

- スリップ（誤選択）：看護師が正しい薬剤を認識しているが、隣の薬剤を手に取る（確率：0.1）（注記）。
- ラプス（忘却）：看護師が指示通りの時間に薬を投与し忘れる、投与手順の一部を飛ばす（確率：0.05）。
- ミステイク（判断ミス）：看護師が薬の投与量を間違える（確率：0.02）。

注記）ここで示す確率はこのプロジェクトで得られたデータだと仮定するもので、実際に得られたデータではありません。実際の分析では、各医療機関の具体的インシデントデータを収集・分析する必要があります。

FTAによるリスク分析

トップイベントを薬剤投与ミスによる患者への健康被害として、ツリー図を作成し、複数のエラーが同時に発生した場合のリスクを視覚化します（詳しくは [8.5.1 FTAを用いた分析例] を参考にしてください）。

改善策の提案

リスク分析結果を元に対策を立案します。

- 薬のラベルを色分けしてスリップを防止する。
- 投与量を自動計算するシステムを導入してミステイクを防止する。
- タスク管理システムで投与時間をリマインドしてラプスを防止する。

ここで示した改善策は技術的な改善策ですが、他にも、設計の基本原則（3章）に基づいた改善や、組織的な改善（10章）をあわせて検討すると良いでしょう。

9

システム思考に基づく安全設計のアプローチ

現代的なシステムは複雑かつ多層的な構造を持ち、他のシステムやサービスとの相互接続性が高まっています。例えば、複数のウェブサービスが連携したり、新たな機能が継ぎ足されて運用されたりする中で、個々の設計者が想定しなかった相互作用によって事故や障害が引き起こされるリスクが増加しています。

従来の安全工学（Safety-I）は、「事故の防止」に焦点を当て、個別の故障や人的エラーを排除することで安全性を確保しようとしてきました。しかし、ホルナゲルらが提唱する Safety-II のパラダイム（Hollnagel, 2006）では、「うまくいくことの確保」に重点を置き、システムの適応能力と回復力（レジリエンス）を重視します。

その背景には、現代的な情報システムが抱える「複雑性」と「密結合性」があります。特に以下のような状況では、従来型の因果分析や故障前提のアプローチでは対応が困難であり、これまでに紹介してきた「個別のエラーや故障の原因と結果」に注目するリスク分析手法では不十分です。

- 複数のコンポーネントや組織が関与し、全体像が見えにくいシステム
- ヒューマンファクターや組織的要因が影響するシステム
- ソフトウェアや非物理的要素が中核を担う構造
- 単一のエラーではなく、連携や判断のミスマッチによって事故が起こるケース

このような状況に対応するため、制御理論とシステム思考に基づいた新しい安全モデルと手法が登場しています。本章では以下のアプローチを紹介します：

- STAMP（Systems-Theoretic Accident Model and Processes）
  システム全体の制御構造と制約を分析する事故モデル
- STPA（System-Theoretic Process Analysis）
  STAMP に基づいて、事故やハザードの予防を目的とした安全解析手法
- CAST（Causal Analysis based on STAMP）
  インシデントや事故が発生した後に、構造的要因を明らかにする事後分析手法

これらの手法を用いることで、ハードウェアやソフトウェアだけでなく、人間、組織、マニュアル、方針といった多様な要素が相互に関与するシステムを対象として、より現実的かつ構造的な安全分析を目指すことができます。

9.1 STAMP/STPA

STAMP/STPAは、レブソンが提唱したシステム思考と制御理論に基づいた安全分析手法（Leveson, 2012）です。システムの構造や制御の失敗に着目し、ヒューマンファクター（人的要因）や組織要因も含めて全体的な安全性を評価できることが特徴です。特に、ソフトウェアの影響が大きい現代の複雑なシステムにおいて、STPAは有効な分析手段として注目されてきました。

9.1.1 STAMP/STPAのキーコンセプト

STAMP/STPAは、事故の発生を個別要素の故障ではなく、システム全体における安全制約（Safety Constraints）の不備として捉える枠組みです。ここでいう「システム」とは、情報システムそのものに限らず、関係組織、運用者、規則や社会的要請を含んだ全体構造を意味します。STAMPでは、これらの構成要素が相互作用する構造の中で、安全制約をどのように維持するかを分析対象とします。システム全体の構造を制御の視点からモデル化する際には、主に階層的な安全制御構造を用いた整理を目指します。

これは、政策決定層、組織運用層、現場実行層等、役割ごとの制御主体（コントローラー）の関係性（コントロールアクションとフィードバックの流れ：後述）を可視化しやすくするための表現形式です。例えば、デジタル庁の中でマイナンバーカードに関わる部門を例にすれば、各部門がそれぞれ異なる役割を持ちながら、共通の安全目標に向けて制御とフィードバックを行う構造を形成しています。

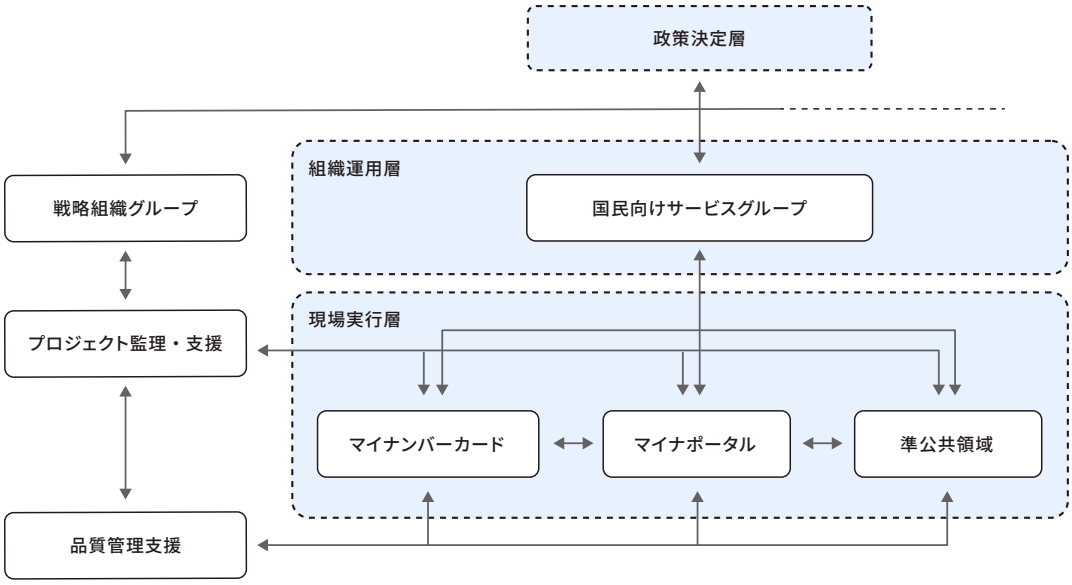

図9.1 組織内での各コントローラーの関係(例示であり、実際のデジタル庁の組織構造を示すものではない)。

もちろん実際のシステムはネットワーク型の構造であったり、複数の制御主体（コントローラー）が横断的に相互作用する等、階層に限定されない構造を持つことも一般的です。STAMP/STPAではそのような複雑な関係性も柔軟にモデル化することができます。関係者間での柔軟な思考や創発を促すことを念頭に置きながら、システムの特性に応じて制御構造の定義に取り組むことが求められます。

さてここまで「コントローラー」とは何かを具体的に説明してきませんでしたが、以下の要件を満たしているものがコントローラーです。

- なんらかの制御アルゴリズムを持っている（STAMPの文脈では特に、安全のための制約、判断）
- 他のコントローラー（システム）に影響を与えることができる（コントロールアクションの出力）
- システムの状態を観測することができる（フィードバックの受信）
- コントロール対象となるプロセスに対応したプロセスモデル（どんな状態にあるときになにができるのか、状態変数とプロセスの遷移）を持つ

STAMPを安全の設計や解析に応用するための手法がSTPAです。「十分な情報かつ適切なタイミングで」（適切性及び適時性）各コントローラーがコントロールアクションを行い、その結果がコントローラーにフィードバックされるフィードバックループにおいて、コントローラー内のプロセスモデルとシステムの状態が一致するときに、ある目標（安全制約）を満たすシステムの状態が維持できると考えます。

9.1.2 動的・複雑な状況への適用

STAMP/STPAは、次のような特徴によって、従来手法では扱いにくかった要因を分析対象に含めることができます。

- 人間の主観的判断や、行動の動的な変化を前提にモデル化できる
- 未経験の状況にも対応可能である
- 手順書の不備やコミュニケーションエラーといった構造的・組織的問題を分析できる

これは、従来のFTAやFMEAのような因果・故障ベースの手法では直接的に取り扱えなかった領域にまで踏み込めるという点で、大きな利点です。

9.1.3 STAMP/STPAを用いた使用エラーの可視化

STAMP/STPAでは、次の4つの観点から非安全なコントロールアクション（UCA: Unsafe Control Action）を特定し、それらがシステム上の制約に違反していないかを検討します。

- コントロールアクションが実行されたことが不適切（例：誤入力）
- コントロールアクションが実行されなかったことが不適切（例：必要な操作を行わない）
- 実行のタイミングが早すぎる／遅すぎる
- 順序の誤りや文脈誤認によって、誤った判断がなされる

これらのUCAに注目することで、例えば「誤入力」「確認漏れ」「判断の混乱」等の使用エラーを、システムの構造やフィードバックループとの関係で可視化することができます。

9.1.4 システムの実際の挙動とプロセスモデルのずれを分析する

STAMPでは、事故の多くは「意図的な違反」よりも「合理的に見えた行動が結果として不安全だった」という構造に起因すると考えます。これは、コントローラーが保持するプロセスモデル（＝対象の理解や予測）が、現実のシステム挙動とずれていることで生じます。

STAMPにおけるプロセスモデルの不整合分析は、認知工学におけるメンタルモデル研究と類似の課題を扱いますが、理論的基盤は異なります。プロセスモデルは制御理論に基づく「制御対象の内部表現」であり、UX設計のメンタルモデルは「ユーザーの概念的理解」を指します。ただし、両者とも「現実との認識のずれ」が安全性や使用性に影響する点で、実践的な親和性があります。このため、STPAの分析は以下のような設計手法と併用することで効果を高められます。

- ヒューリスティック評価と併用する
- シナリオベースで使用状況の文脈を踏まえた分析を行う

9.1.5 STAMP/STPAの導入が有効なケース

以下のような場合には、STAMP/STPAの導入を検討してみると良いでしょう。

- 新規に大規模または複雑なシステム開発を行う場合
- 複数のコンポーネント・サービスが相互作用する構造を持つ場合
- 設計の初期段階で、要件検討や制約の可視化を行いたい場合
- ヒューマンファクターやソフトウェアのバグがリスクの大半を占める場合

STAMP/STPAは定性的な分析手法になるため、発生確率や影響の大きさを数値的に扱うことは困難です。費用対効果等を算出する必要がある場合は、STAMP/STPAで特定したリスクに対して、PRAやHRAを併用するなどして確率的な評価を行う必要がでてきます。

一方で、STAMP/STPAは定性的な検討が多い人間中心設計とは親和性が高い手法です。開発の初期企画段階からSTAMP/STPAを導入することで、安全制約を考慮に入れたデザイン及び開発を行うことができます。また、本手法はセキュリティインシデントを防ぐための手法としても活用できます。

本セクションでは、基本プロセスとデザインプロセスへの導入のヒントを紹介します。本手法について詳しく知りたい方は、IPAが発行しているガイドブックや民間の解説書を参考にしてください。IPA版ガイドブックには、日本の利用者向けに適応した手順、分析例等が網羅されています。

9.2 STPAの基本プロセス

ここからは、STAMPに基づいて事故やハザードの予防を目的とした安全解析手法であるSTPAの基本的なプロセスを見ていきます。主に以下のステップで構成されています。

- 分析目的の明確化と安全目標の設定 [9.2.1]
- コントロールストラクチャ（Control Structure）の可視化 [9.2.2]
- 安全でないコントロールアクションを抽出する [9.2.3]
- 安全でないコントロールアクションが引き起こされる要因やプロセスを特定する [9.2.4]
- 安全制約を満たすための制御構造とプロセスをモデル化し設計に反映する [9.2.4]

9.2.1 システムの全体像を明らかにし、安全目標を定める

STPAの最初のステップは、分析の目的を明確化することです。これにより、システムが本来あるべき「安全な状態」を定義し、分析の指針を整えることができます。このステップでは、次の3つの要素を定義します。

- 損失（Loss）：利害関係者にとって価値のあるものの喪失
  例：人命、財産、環境、社会的信頼等
- ハザード（Hazard）：特定の環境条件下で損失につながる可能性のあるシステムの状態
  例：車両が歩行者に接近している状態
- 安全制約（Safety Constraint）：ハザードの発生を防ぐためにシステムが維持すべき制約条件
  例：車両は歩行者から2m以上の距離を保つ

具体的には、対象システムの実態や利用状況について、ドメインエキスパートや運用者、利用者等のステークホルダーから情報を収集し、ハザードや損失を洗い出していきます。ここでの「システム」は、単なる情報機器やサービスにとどまらず、人、組織、手続き、文化、運用ルール等を含めた社会的システム全体を意味します。例えば以下のような点も、システムの要素として考慮する必要があります。

- オペレーターの慣れや熟練度、認知特性
- 組織文化や教育水準
- マニュアルや運用ルールの変化
- 緊急時対応の柔軟性

残念ながら（どんな手法においてもそうですが）「なにが安全なのか」「何を回避するべきなのか」を定義しきる、簡単で完全な手法はありません。本ガイドブックの前半でも示したように、ユーザビリティは使っているときの文脈や環境、利用者の知識等によって影響を受けます（ハザードランプの示す意味が文脈によって変わる例を思い出してください）。また、ここでいう「システム」には、[9.1.1] でも示したように、安全性を担保する組織や人も「システムの構成要素」として含んでいます。組織文化やオペレーターの「慣れ」や経年変化等によっても安全は脅かされうるのです。

例えば空港の航空管制を例に考えてみましょう。航空管制は、出域管制（離陸）、入域管制（着陸）、タワー管制（滑走路管理）といった複数の制御主体（コントローラー）が協調して運用されています。出域管制は離陸機の安全を、入域管制は到着機の安全を受け持つと（ここでは試しに）定義するとします。このときに各管制が最も注

目するのは「どの飛行機が離陸し又は着陸しようとしているのか」という情報です。しかし「安全を確保する」ときにはそれだけでは不十分です。「離着陸中の航空機を安全に離着陸させる」という基本的目標だけでなく以下のような、普段のオペレーションの枠外の事象に対しても、どのように安全制約を維持するか明らかにしておく必要があります。

- 滑走路に迷い込んだ工事車両
- ヘリコプターの緊急着陸
- 渡り鳥の群れの飛来

想定外の事態が起きても安全が維持できるようにシステム全体の安全目標を定めるのが本ステップの目標です。そのためには、各コントローラー間での相互作用をモデル化し、なにかエラーが起きたときに、危険につながる要因を見定め、対策をする必要があります。具体的には、以下のような原因が安全性を脅かすリスク要因となり得ると考えます。

- 不適切なアクションや手順（不完全なプロセスモデルや設計ミス）
- 不正確な認知や、利用者のメンタルモデルと実際のプロセスモデルの不一致
- コントローラー間の連携不全
- フィードバックの遅延や欠如

これらを踏まえて、安全制約とは「○○してはならない」「常に○○を確認する必要がある」といった形で、具体的かつ制御的に定義されるべきものです。安全制約を定めるには、こうした安全性に影響を与える要素や文脈等の全体像を十分に洗い出し、関係者間で、どのような目標に向かって設計を行うのか合意していく必要があります。

自動運転システムの安全目標の例を見てみましょう。自動運転車両では、以下のような包括的かつ現実的な安全目標が考えられます。

- 交通状況に関係なく安全に運転を維持する
- 人が乗り降りしているときにも、事故や挟まりが起きないよう制御する
- 万一事故を起こした場合も、乗員・被害者・歩行者の安全を最優先する
- 災害時・緊急事態下では、乗員及び周囲の人の安全を確保する動作に切り替わる

これらの目標は、「事故を起こさない」ことだけでなく、「事故やハザードが発生しても被害を最小限に抑える」という視点から設定されるべきです（＝フェイルセーフやリカバリの観点を含む）。また、以下のようなハザードの管理体制上の曖昧さを見逃さず分析することも重要です。

- コントローラーの誰も責任を負っていないハザード
- 複数のコントローラーの相互作用によって発生するハザード
- コントロールアクションが重複したり、コンフリクトを起こす場合の対応の不備

「なにをもって安全とするか」を可能な限り包括的かつ明確に定義し、以降の構造分析や対策立案の軸にする本ステップは、STPAで非常に重要なステップです。

9.2.2 制御構造の可視化

STPAの次のステップは、システムの制御構造（Control Structure: CS）を明示的にモデル化することです。これは、前ステップで明らかにした安全目標やハザードに関わるコントローラー（制御主体）や被制御要素、その相互作用を図式化する作業にあたります。この作業によって、システムがどのように制御されているか、制御がどこで破綻する可能性があるかを分析する基盤が整います。制御構造を可視化するには、次のような5つの基本構成要素を必ず盛り込みます。

- コントローラー（Controller）
  コントロールアクションを発する主体。人間、ソフトウェア、組織等が該当します。
- コントロール対象プロセス（Controlled Process）
  コントローラーの指示によって振る舞いが変化する対象。車両、システムUI、現場オペレーター等。
- コントロールアクション（Control Action）
  コントローラーが出す命令・操作指示（例：「ドアを閉める」「エアバッグ展開」等）
- フィードバック（Feedback）
  コントロール対象プロセスからコントローラーに返る情報（例：「ドアが閉じた」「エンジン異常あり」等）
- コントロールアルゴリズム（Control Algorithm）
  プロセスモデルと入力に基づいて、どのアクションをいつ実行するか判断する内部ロジック

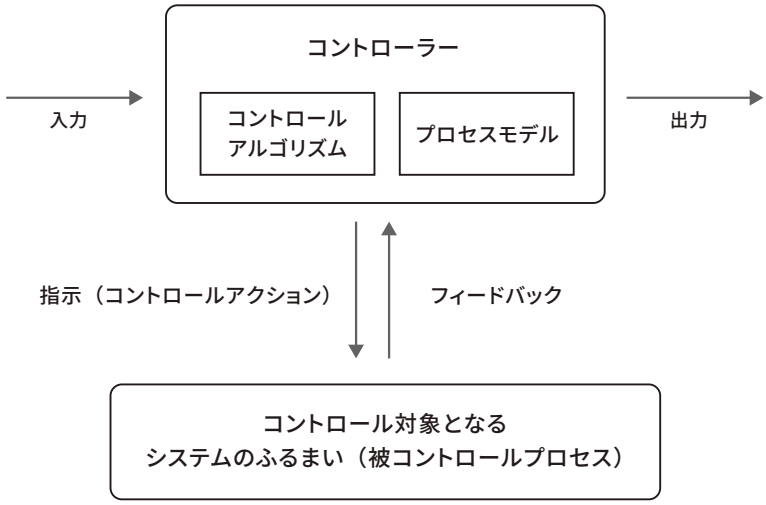

図9.2 コントローラーとコントロール対象プロセスの相互作用を示す基本的なモデル。

また、各コントローラーは内部に（コントロール対象プロセスに対する）プロセスモデル（Process Model）を持ち、プロセスモデルに沿ってシステムの状態を把握し、制御判断を行います。本章冒頭のキーコンセプトの紹介で述べたように、制御構造は一般に階層構造で描かれることが多く、例えば次のようなレベルを分けて記述します。

- 政策レベル（例：運営方針の決定者）
- 運用レベル（例：現場責任者・管制）
- 実行レベル（例：システム操作、端末制御）

ただし、実際のシステムはネットワーク型や分散型の構造を取ることもあります。柔軟に制御構造を表現して大丈夫です。制御構造図を作成することで、以下のようなリスク因子の検出がやりやすくなります。

- コントロールアクションが出されていない、または遅延している箇所
- 情報フィードバックが遮断または不足している箇所
- プロセスモデルに誤りや齟齬がある箇所
- 複数のコントローラーの指示が競合している箇所

これにより、システム全体として「なぜ安全制約が維持されないのか」を構造的に捉える土台を形成できます。具体例もみておきましょう。自動運転者での速度制御を対象に検討します。

- コントローラー：自動運転ソフトウェア、監視者
- コントロール対象プロセス：車両のブレーキ、アクセル、ステアリング
- フィードバック：センサーからのデータ（車速、車間距離、現在位置等）

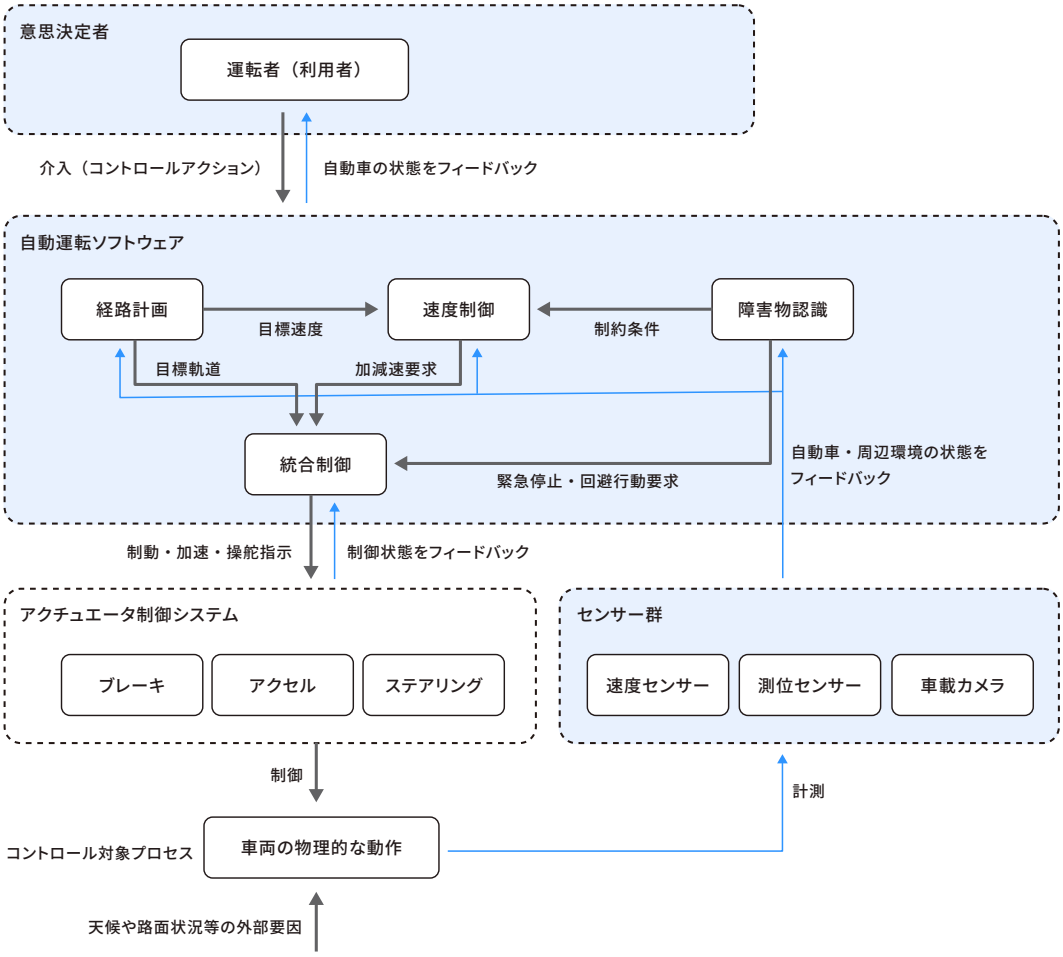

図9.3 自動運転を例とした制御構造図(CS図)の例。

制御構造図の作成は、単なる図解作業ではなく、関係者間で「安全性とは何か」「制御はどう行われているか」を共有・可視化する重要なコミュニケーションツールにもなります。この図に基づいて、次のステップでは「非安全なコントロールアクション（UCA）」の抽出へと進みます。

9.2.3 非安全なコントロールアクションの抽出

STPAの第2ステップでは、コントローラーが実行するすべてのコントロールアクション（Control Action: CA）を特定し、それらの中から安全を脅かす可能性のあるもの、すなわち非安全なコントロールアクション（Unsafe Control Action: UCA）を抽出します。これは、従来の使用エラーと対応づけながらも、「アクションの不在」「不適切なタイミング」「誤った順序」等、制御の文脈に沿った原因分析を可能にする重要なプロセスです。STPAでは、各コントロールアクションについて、以下の4つの観点から「そのアクションがどのように非安全になりうるか」を検討します。

- ① 必要なアクションが実行されない
  例：危険な状況下でブレーキが作動しない。
- ② 不適切なアクションが実行される
  例：通常走行中に誤って緊急ブレーキが作動する。
- ③ タイミングが早すぎる／遅すぎる、または順序が誤っている
  例：停止線を越えた後にブレーキがかかる、信号よりも早く発進する。
- ④ アクションが必要な時間よりも長く／短く続く
  例：機首を下げ続けたまま戻らない、加熱処理が途中で停止する。

このような形式で分類することで、操作の不在・過剰・不適切なタイミング・時間といった観点でリスクを網羅的に洗い出すことができます。例えば、自動運転車のコントロールアクションを想定すると、次のような組み合わせが考えられます。

コントロールアクション（CA）	非安全な実行例（UCA）
アクセルを踏む	危険な状況でアクセルを踏んでしまう（②）
ブレーキをかける	必要なときにブレーキが作動しない（①）
方向を変更する	変更のタイミングが遅れ、衝突する（③）
アラートを表示する	表示が早すぎ／遅すぎ／表示されない（①〜③）

非安全なコントロールアクションの洗い出しには、HAZOP [8.7] で紹介したガイドワードを応用することで、網羅的な検討がしやすくなります。例として以下のようなキーワードが使えます。

- Not executed（実行されない）
- Too early / Too late（早すぎる／遅すぎる）
- Wrong direction / Wrong object（誤対象／誤方向）
- Too long / Too short（時間過剰／不足）

こうした工夫で分析者の見落としや、熟練度等によるばらつきをある程度防ぎ、形式知としての安全分析を効率化・底上げすることができます。

このステップでは、コントロールアクションそのものが「危険であるかどうか」だけでなく、そのタイミング、実行状況、文脈を含めて評価することが重要です。特に設計初期の段階や、複数のシステムが連携する状況では、予想外のUCAが潜在していることが少なくありません。STPAの真価は、このような「制御の失敗」に注目し、エラーや故障がなくても事故が起こる構造的原因を明らかにできる点にあります。

9.2.4 原因分析と設計の改善

前節 [9.2.3] で特定した非安全なコントロールアクション（UCAs）をもとに、ここではそれがなぜ発生し、どのように損失につながるかを具体的に検討します。この段階では、UCAに紐づくハザードの原因要因（Hazard Causal Factor: HCF）、そしてそれがどのような過程で損害につながるのか（Hazard Scenario: HS）を特定し、安全制約を強化するための設計改善へとつなげていきます。

HCFは、UCAが発生してしまう背景にある構造的、認知的、プロセス的な原因要因、組織運用等、幅広い範囲で発生する制御上のギャップを意味します。例えば次のような事象が該当します。

- コントローラーのプロセスモデルが実状と食い違っている
- 入力情報が誤っている、または不足している
- コントロールアルゴリズムが適切でない
- フィードバックが遅延／欠落している

このような原因が、コントロールアクションの誤りや判断ミスを引き起こすことで、結果としてハザードを誘発します。注意すべきは、人間の行動は静的なアルゴリズムではなく、状況やフィードバックに応じて動的に変化するという点です。システムと対話するオペレーターは、システムの反応を見て判断を変えたり、ルールを補完的に解釈することがあります。ここで、オペレーターに正常性バイアスが働いたり、システムイメージを適切に認知できていなかったりすると、以下のようなHCFが現れます。

- 「空港の滑走路にはヘリコプターは入ってこない」と信じていた
- 「鳥が接近すれば警告灯が出る」と思い込んでいた

このような誤認知や知識のギャップは、人的エラーとして再訓練や教育で対処されるだけではなく、設計上取り除くべき構造的要因とみなすことが大切です。使用エラーが発生した場合に安全を保つための制御は（人の訓練だけではなく）システム側でなるべく保障しておく必要があります。

次に、ハザードシナリオ（HS）とは、UCAやHCFを介して、損失が実際に引き起こされるプロセスを記述したものです。これは一種の「安全面でのユーザーシナリオ」であり、構造化された時系列記述により、対策すべき局面を明確にします。例（自動運転システム）で考えてみましょう。

- UCA：ブレーキが遅れて作動した
- HCF：雨天によりセンサー視界が遮られ、歩行者を誤認した
- HS：歩行者を検知できず→ブレーキ信号が遅延→減速が間に合わず接触事故を起こした

このように、HSはHCFを複数含み得る「損害への因果ルート」として、対策優先度や設計的な弱点を把握する手がかりになります。ハザードシナリオの例を見てみましょう。

損害シナリオ（HS）	背景にある HCF
センサー誤検出により歩行者を見落とす	誤った入力情報に基づくプロセスモデル
通信エラーで自動運転がコントロールアクションを送信できない	フィードバックの欠落／制御経路の脆弱性
制御の遅延により危険回避が間に合わない	リアルタイム処理不足、制御設計の非現実性

HCFを元に特定されたリスクを軽減させるため、設計を見直します。

- 安全制約の再評価・強化
  冗長化、フェイルセーフの導入等を検討します。
  例：センサーの二重化、予備系の切り替え設計

- 制御構造・フィードバック経路等のコントロールプロセスの改良
  情報伝達の信頼性向上、監視強化等を検討します。
  例：ソフトウェアの状態監視とリカバリプロセスの追加

- 人間の動的な変化に配慮した設計
  メンタルモデルの明示等のUXデザイン、UIデザインの再検討を行います。
  例：フィードバック情報の可視化、リハーサル機能の導入

- 組織・責任の可視化とルール整備
  責任の空白や競合をなくす構造を設計に埋め込みます。
  例：複数部署が関わる場合の安全チェックポイントの明示化

HCFは「なぜUCAが起きたのか」を構造的に説明する鍵であり、HSは「それがどう損害に至るのか」を記述する道具です。HCF・HSの分析は、STPAを人間中心の安全設計に結びつける重要な橋渡しとなります。人や組織のエラーを「訓練不足」で済ませず、構造的にエラーが起きない仕組みとして設計へ落とし込むことが、STPAの本質的な価値です。特に、ユーザビリティ設計の場面では、人の認知的限界やメンタルモデルのずれに起因するHCFを早期に検出できる点が、大きな強みとなります。

9.3 CASTによるインシデントの原因分析と再発防止

CAST（Causal Analysis based on STAMP）は、STPAと同様にSTAMPのシステム理論に基づいていますが、事故やインシデントが発生した「事後」にその原因構造を分析し、組織的・設計的な再発防止策を導出するための手法です。「誰が何を間違えたか」といった責任追及型の分析ではなく、「なぜ合理的な判断が結果として不安全になったのか」「どのような情報や組織構造がそれを許容したのか」を構造的に問います。

9.3.1 CASTの基本プロセス

CASTでは、以下のようなステップで分析を行います。

- 発生した損失とそのハザードを定義する
  例：患者への誤投薬、サーバーダウン、列車の遅延 等。分析対象システムの境界と発生した事故の詳細を明確化する。
- 安全制約を明らかにする
  事故防止のために本来維持されるべきだった制約を特定する。
- 安全制御構造を可視化・文書化する
  制約がどう破られたか、関与したコントローラーの制御構造を明らかにする。
  人的／システム的コントローラー、コントロールアクション、フィードバックの有無など。
- 各コントローラーのプロセスモデルと認知状況を分析し、安全制約違反（欠陥）を特定する
  人がコントローラーであれば「合理的に思えた判断、その要因は何だったか」、「情報に欠落や誤認はなかったか」。
- 組織的・設計的な背景要因（HCF）を抽出する
  物理プロセス、技術・運用、組織・管理、政府・規制の各レベルで要因を特定する。
- 改善策・再発防止策を導出する

大枠の手順はSTPAで紹介したものと同じなので、ここでは詳説しませんが、ユーザビリティ確保の観点から重要なポイントを紹介します。

9.3.2 CASTの観点をユーザビリティテストで援用する

CASTの観点は、ユーザビリティテストの補完的手法としても利用可能です。

- UIが誤認識を生み出す原因となっていないか
- 複数の利用者（例：看護師と医師）の間で、期待されるアクションに齟齬がなかったか？
- 操作ログや記録は、正しく利用者の行動を支援していたか？

電子カルテの操作ミスを例に、簡易に分析した例を示します。

ステップ	内容
損失定義	患者に誤薬を投与してしまった
安全制約	患者の処方薬は照合後にのみ実行されるべき
コントローラー	看護師、処方システム、照合アプリ
プロセスモデルの齟齬	看護師が「照合済」と誤認した（アラート UI が見落とされやすかった）
フィードバックの不備	処方の確認結果が画面上に明示されなかった
改善案	UI 再設計、アラート方式の変更、操作ログの可視化

繰り返しになりますが、CAST は、インシデントが発生した際に「誰かが間違えたから」ではなく、なぜそのような合理的判断が不安全につながったのかを、構造的・認知的に可視化して再発防止につなげるための枠組みです。以下のような観点を踏まえて分析を行いましょう。

- その判断や行動は、当事者にとってどのように合理的に見えたのか？
- どの情報が不足／誤っていたために誤認が起きたのか？
- フィードバックループは途切れていなかったか？（コミュニケーションミスマッチ）
- 組織として、そのコントロールアクションをサポートできていたか？
- 手順・権限・責任は適切に分担されていたか？

STAMP に基づく安全設計では、STPA による事前の予防的分析と、CAST による事後の因果構造分析を組み合わせることで、設計と運用の間に連続性を持たせ、より強固な安全文化の構築を目指すことができます。CAST で明らかになった以下のような事象は、STPA のフィードバックとして再利用することもできます。

- 安全制約の不備：新しい制約の追加
- 認知・プロセスモデルの誤り：人的要因をより深く反映したモデル化
- コントローラー間の責任のあいまいさ：制御構造の再設計

STPA や CAST で明らかになった非安全な制御構造や要因を一度限りの対応で終わらせず、システムや組織そのものが学習し、より適応的な安全性を構築していくためには、「ダブルループ学習」の考え方が重要になることも覚えておきましょう。これは、事故やインシデントの発生時に、単に行動（例：手順や UI）を修正するだけではなく、その行動の背後にある前提・ルール・価値判断・安全制約のあり方自体を見直すことを意味します。

例えば、操作ミスが繰り返される場合、それをオペレーターの教育不足とみなすのではなく、「なぜそのような判断が合理的に見えたのか」「判断を支える情報設計や組織構造が適切だったのか」といった、システム全体の設計思想そのものに立ち返る視点をもたらします。

注記）ダブルループ学習（Double-Loop Learning）とは、組織が失敗や問題から学ぶ際、単に行動や対応を修正するだけでなく、根本的な目標・ルール・価値観そのものを見直す学習プロセスを指します。

10

リスク情報を活用した意思決定

これまで紹介してきたようなリスク評価や解析の手法（STPA、FTA、FMEA等）は、それ自体が目的ではなく、意思決定に役立てて初めて意味を持つものです。いかに優れたリスク分析を行っても、それが設計や運用の判断に反映されなければ、安全性の向上にはつながりません。リスク情報を活用して合理的かつ組織的な意思決定を行うプロセス、すなわち「リスク情報に基づく意思決定（Risk-Informed Decision-Making: RIDM）」が重要になってきます。RIDMの実現にあたっては、以下のような要素を明確に定義しておく必要があります。

- 提供しようとしている製品・サービスの特性とリスク特性の把握（パフォーマンス観察・評価）
- 安全な状態とはなにかを組織内で定義し、共有する
- 安全目標の達成に向けて、どのような情報を、どのように活用して判断を下すかという意思決定の流れ（合意形成のための組織的プロトコル）を整備する
- 人的・財政的リソースを確保する

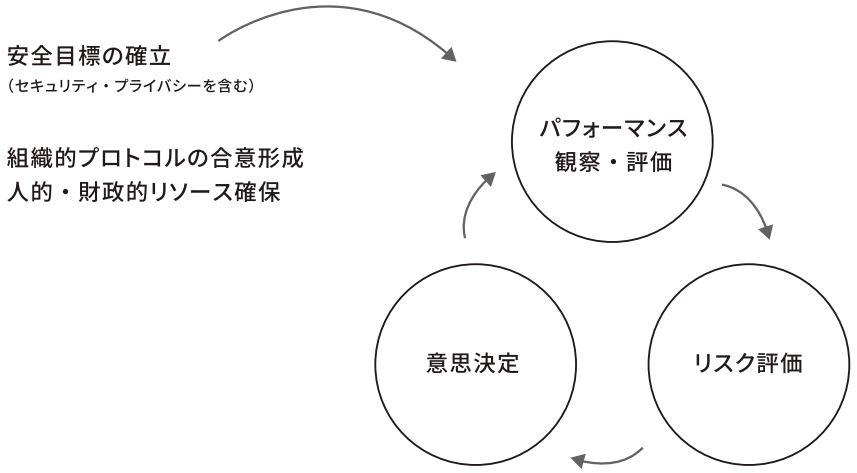

図10.1 RIDMのプロセス。安全目標に対してのパフォーマンスを観測・評価し、把握されたリスク特性に基づいた意思決定、実行のサイクルを繰り返す。

近年の安全工学においては、単に「事故やインシデントを防止する」ことを目標とするのではなく、発生を前提とした「安全のコントロール」（Safety-II）へと発想がシフトしています。つまり、失敗をゼロにすることよりも、失敗が起きても被害を最小限に抑えるよう、システムの構造を設計することが重要とされています。第９章の冒頭でも述べたように、使用エラーやハザードを排除するのではなく、「うまく対処できる」適応能力と回復力（レジリエンス）を重視したシステムづくりが求められているのです。

この「安全のコントロール」（Safety-II）は、システムが複雑化し、相互接続性が高まる中では核心的な考え方です。単純なシステムでは従来の「失敗ゼロ」アプローチが有効な場合もありますが、複雑な社会技術システムでは、それが現実的でなくなっています。

なお、完全に「失敗を前提とする」わけではなく、重大な事故につながる可能性のある失敗に対しては依然として予防的アプローチを取る必要もあります。つまり、すべての失敗に対して同じく「対処能力」を重視するのではなく、リスクの重大性に応じたバランスが重要なのです。

こうしたパラダイムの変化は、「ヒューマンファクター（人的要素）」や「使用エラー」の見方も変えました。従来のようにヒューマンファクター（人的要素）や使用エラーを単に「排除すべき厄介な問題」として扱うのではなく、人間の適応能力がシステムの安全性に貢献する側面も認識されるようになっています。

10.1 RIDMの基本プロセス

RIDMは、意思決定においてリスク評価結果を単なる参考情報にとどめず、合理的な判断の中心要素として位置づけるプロセスです。RIDMの特徴は、単なる「安全第一」ではなく、「目的を達成しながらリスクをどのように受容・制御するか」を評価し、複数の選択肢の中から、最も妥当で実行可能な解を選ぶ点にあります。RIDMは以下のようなステップで実施されます。このプロセスは、安全工学に限らず、UI設計や運用業務の変更、サプライチェーンリスク等にも応用可能です。

- 課題の特定とスコープ設定
- 関連するリスク評価情報の収集
- 設計・運用案（選択肢）の策定
- 選択肢ごとのリスク比較と評価
- 価値判断と合意形成に基づく意思決定
- 決定根拠の文書化とフィードバックの活用

10.2 リスク評価情報の可視化と説明責任

リスク情報に基づいて意思決定を行うにあたり、評価内容が誰にとっても理解できる形で表現されているか、そしてその判断の根拠が将来にわたって追跡可能かどうかは極めて重要です。いかに定量的なリスク評価がなされていても、それが一部の専門家だけに理解されるような形では、意思決定の正当性を社会的に説明することは困難です。リスク情報の「可視化」と「説明責任（Accountability）」は、RIDMを行政機関で実効性あるものとするための土台となります。

10.2.1 リスク評価情報の可視化

リスク評価情報の可視化において考慮すべきポイントは、次の3つです。

- 関係者の立場・理解レベルに応じた可視化を行うこと
  管理者向けにはハザード全体の位置づけを示すマップ（例：リスクマトリクス）、技術担当者には因果関係の流れやUCA・HCFの構造（例：STPAの制御構造図）等、興味・関心に応じた可視化を行います

- リスクの不確実性の明示
  リスク評価は確率的であり、常に限界があることを示す必要があります。不確実性に関するメタ情報（例：評価に使ったデータの質）等を整理しておくようにします

- 複数の代替案を比較しやすい形で提示する
  表形式、FTAやFMEAの視覚化結果等を活用すると良いでしょう。UIが議論になる場合は、プロトタイプを作成することも有用な選択肢のひとつです

10.2.2 説明責任

リスクに関する意思決定は、将来的にその正当性を問われる場面（例：事故後の調査、社会的説明、法的責任の所在確認等）に直面します。そのためには、以下の情報を継続的に記録し、トレーサビリティを保つことが求められます。

- 評価時点での仮定・前提条件の記録
- 使用したデータ・手法・モデルの明記
- 判断における選択肢と理由づけの文書化
- 意思決定に関与した関係者と合意形成の過程

このようにして、意思決定の背後にある思考プロセスを外部化・文書化することで、後から検証可能な状態を保ちます。これは、単に「安全である」と主張するのではなく、「どのように安全と判断したか」を透明にするために大切な観点です。記録管理については、「DS-680.2 ウェブコンテンツガイドライン」（デジタル庁）［16 公開情報の品質確保］及び［18 記録管理］も援用できるものですから、参考にしてください。

10.3 安全文化と学習サイクルへの統合

リスクに基づいた意思決定を一過性の対応で終わらせないためには、組織全体で安全に対する共通理解と継続的な学習の仕組み＝「安全文化」を醸成する必要があります。ここでいう安全文化とは、「安全は現場任せではなく、組織全体の価値判断として根づくべきものである」という認識に立ち、日常的な意思決定にリスク評価を組み込む態度や制度のことを指します。

前節までに紹介したCASTによる事故後の因果構造分析は、単なる事後対応にとどまらず、STPAを用いた設計段階の安全制約にフィードバックをかけることで、システム全体の設計そのものを改善する契機になります。

このように、発生したインシデントを通じて組織が自身の「安全観」や「制約の定義」「責任構造」まで問い直す学習は、ダブルループ学習（Double-Loop Learning）と呼ばれ、現代的な安全マネジメントの重要な構成要素です。

- シングルループ学習
  手順や行動を修正する（例：手順書の見直し）
- ダブルループ学習
  手順の背後にある前提や価値判断を問い直す（例：手順そのものの必要性、目的の再定義）

この学習ループを回すためには、事故が起きたときの調査や報告を、責任追及ではなく、構造的な知見の共有と位置づける組織文化が欠かせません。

10.3.1 安全文化のための制度的支援

安全文化の醸成には、単に制度や仕組みを整えるだけでは不十分です。最も本質的なのは、誰もが「安心して懸念を表明できる」「エラーや改善提案を共有しても非難されない」組織的な土壌＝心理的安全性（Psychological Safety）を確保することです。

心理的安全性とは、「この場で自分の考えを発言しても拒絶されたり評価を下げられたりしない」と信じられる状態のことであり、効果的なチームの最重要要因として繰り返し示されています。この心理的安全の醸成を図るには、組織・チーム内での関係性の質を高めなければなりません。ダニエル・キムが提唱した「成功循環モデル（Core Theory of Success）」は、組織の成果を「結果」から変えようとするのではなく、「関係の質」から見直すことが、持続的な成果につながると説くものです。

要素	安全文化における意味
関係の質	上下・部署間を問わず、誰もが安全に発言・共有できる関係性（心理的安全性）
思考の質	「この行動の背景には何があるか？」という構造的・非懲罰的な視点で考える態度
行動の質	使用エラーやインシデントをオープンに報告・記録・改善提案する行動の実践
結果の質	リスクの早期発見、安全設計の改善、組織としての信頼性の向上

関係の質（＝心理的安全）が整っていなければ、いくら「STPAを導入する」「CASTを使って調査する」と宣言しても、現場からの報告や知見が集まらず、表面的な分析に終わる危険があります。逆に、心理的安全性が高く、誰もが「言っていい」「聞いてもらえる」と感じている状態であれば、現場起点のインサイトが集まり、思考・行動の質が高まり、設計の質・結果の質が向上し、さらに良好な関係性が醸成されるという好循環が回り出します。こうした観点から、安全文化を支える制度的支援は、以下のように組織内の各階層で個別及び越境的に設計される必要があります。

関係の質を高める仕組み

定期的な「失敗共有カフェ」やラウンドテーブルの開催。上下関係のないレビュー形式（ローテーション司会、匿名投稿制度等）の整備等、安全に関する懸念を誰でも持ち寄れるレビュー会議の場を、ヒエラルキーを越えて定期的に設けます。発言の場に役職・経験年数の影響を与えないようにファシリテーションやグラウンドルールを設けるのもポイント。1on1を定期的に実施することも関係の質の向上につながります。

思考の質を高める問いかけ

CASTやSTPAの観点を「問いのテンプレート」として組み込む（例：「なぜそうしたのか？」ではなく「その時どう見えたか？」等）。問いには、クローズドな質問（「はい/いいえ」で答えられるもの）とオープンな質問（具体的な列挙や叙述が求められる質問）、具体的な質問と抽象的な質問等をシーンに応じて使い分けると良いでしょう。

行動の質を変える制度

報告した人を責めず、報告そのものを評価する仕組みの制度化。潜在的なリスクの「見える化」を推奨します。なお、行動の質を変えるには、多様な立場の声を汲み取るフィードバック構造を作り、各ステークホルダーがそれぞれの認識・判断・行動に別々のプロトコルを持っていると十分に自覚することが重要なポイントです。横断的な議論の場（シナリオワークショップ等）を設けること等も有効でしょう。

結果の質を活かす循環

報告から得た教訓をUIや業務マニュアルに反映し、定期的に「変更理由」を周知する。社内WikiやLT（ライトニングトーク）の実施等も有効です。

全体の振り返り

安全文化の維持・醸成のプロセスの循環が組織内でどのように行われたか、振り返りの機会を設けることも大切です。心理的安全が損なわれた職場では、リスク情報が隠蔽され、表層的なリスク管理に終始しがちです。逆に、「言っても大丈夫」という文化があることで、初期兆候の早期発見、ユーザー行動の理解、設計上の課題発掘が可能になります。

10.3.2 安全文化の醸成のためのチェック項目（サンプル）

1	インシデント報告やヒヤリハット共有を評価しているか	☐
2	使用エラーは「状況に対する合理的な行動の結果」と捉える視点が共有されているか	☐
3	役職や立場に関係なく意見を出せる仕組みがあるか	☐
4	雑談・振り返り・朝会等の中でも、リスクや懸念を共有できる機会があるか	☐
5	リスクや懸念の共有に対し、即時に感謝や承認のフィードバックが与えられるか	☐
6	多様な関係者が、横断的に議論できる場が整備されているか	☐
7	STPA/CAST 等のリスク分析手法を活用し、定期的な見直しと議論が行われているか	☐
8	「安心して話せる」状態を継続的に可視化しているか	☐
9	RIDM のプロセスや根拠が記録され、共有可能になっているか	☐
10	過去のエラーやインシデントがまとめられ、教育や設計の検討に活用されているか	☐

このチェックリストはチェック形式を用いていますが、成熟度等のスケール（尺度）を用いても良いでしょう。

10.4 合意形成

現代のシステムでは、複雑なルールと評価基準、技術仕様、ガイドラインが錯綜し、設計・開発・運用に関わる多様なステークホルダーが共通理解を持つこと自体が難しくなっています。さらに「機能改修単位で発注が行われ、実装する受託事業者が異なる」「複数のプロジェクトにまたがって設計・運用される」情報システムも多くなっています。このため、横断的かつ整合性のある監査基準、言い換えれば組織内外での合意形成のプロセスの設計が求められています。

情報システムの品質評価の透明性や公平性が欠けると、現場では「規制への一時対応」に終始してしまい、本質的なユーザビリティや安全性が置き去りになるリスクがあります。というのも、例えば成果物の品質確認を行う人による対応や判断の差がある、あるいは評価基準が透明性を欠くものになっている場合、対応コストが見通せず大きな負担になってしまいます（プロジェクトが「炎上」しているときは特にそうです）。対応を迫られる事業者側も規制への目先の対応を優先してしまい、真にユーザビリティやアクセシビリティを確保するための取り組みをおざなりにしてしまう危険があります。また、課題設定や評価対象のフレーミングの瑕疵が、限られた関係者内での閉じたコミュニケーションだけでは認識されない場合もありえます。

こうした形骸化を防ぐには、「リスク情報に基づく意思決定（RIDM）」と「心理的安全性を基盤とするオープンな評価プロセス」を制度設計に組み込む必要があります。生成AIや動的に変化するAPI連携の普及等により、システムの挙動そのものがブラックボックス化する傾向が強まっている中で、利用者・行政機関・設計者間の信頼を築くには、「どのようにしてリスクが特定・制御・再評価されているか」を可視化し、相互に評価・合意できる枠組み（＝監査: Oversight）が欠かせません。

このため、リスクの管理と合意形成のためには、以下のような観点を基本として、監査の枠組みを構築していく必要があります（注記）。

客観性

評価指標が主観ではなく、明文化されたルールとパフォーマンスに基づいているか。特に政府の情報システムの開発・運用には、様々な事業者・府省庁が関係してきます。違うステークホルダーでも同じ言葉を同じ意味で使える状態、共通の目標を共有できている状態をすぐに作れることは、情報システムの品質を継続的に向上させるための土台になり、開発の失敗やインシデントのリスク低減につながります。

予見性

評価対象者（事業者・開発者）が、必要な資源や対応策を事前に把握可能であるか。監査主体によって評価に大きな違いがあると規制への目先の対応が横行するようになり、本来は中長期的に取り組むべき課題に取り組めなくなってしまいます。開発の目標設定や設計の基準を定め、適切な開発プロセスを整備できるよう、品質の基準を明確化することは、開発プロセスや工数の見通しを事業者や調達担当者が明確にする材料になります。

適時性

問題のある箇所を早期に特定し、修正できるプロセスが整備されているか。問題が発覚しても、修正するのに来年度まで待たなければならない状態では、情報システムを信頼して利用することができません。課題は発見されるだけではなく、それが必要なタイミングで是正される必要があります。

公平性

「現実の利用状況における実際のパフォーマンス」に焦点をあてているか。監査がチェックリスト偏重になると規制が複雑かつ膨大になったり、個別にチェック事項を満たすことが優先されてしまい、システム全体の安全性に注意が払われなくなったりする懸念があります。

チェックリストに基づいた評点制度は、一見すると客観的で公平に見えますが、「アクセシビリティ試験は機械チェックで100点を取れば良い」（実際に障害者が利用できるかは確認していない）というように、本来の目的から外れて逆に客観性を欠いた理解や運用を招くことがありますので、注意が必要です。

注記）これらの観点は、米国の原子炉監査プロセス（ROP: Reactor Oversight Process）を参考にしています。

デジタル庁が提供しているツール群で、客観性と予見性を担保するための補助ツールにあたるのがデジタル庁デザインシステムやUIチェックリスト、各種ガイドラインです。これらのツール群を参照することにより、調達担当者及び事業者は、どのような目標をどのように達成すればいいのか一定のレベルで予測し、リソースの確保やWBSの策定等のプロジェクト計画を一定の品質で立案できるようになるよう配慮されています。

さらに、本ガイドブックで紹介したCASTやSTPA等のシステム思考ベースの手法は、「なぜ失敗が起きたか」ではなく「なぜその行動が合理的だったか」を問う姿勢を提供し、非懲罰的かつ構造的な監査の文化を醸成するうえでも有用です。

10.4.1 安全に関する合意形成のためにやるべきこと

国民生活や企業活動に大きな影響を与えうる多くの行政サービスには複数の行政機関や業界団体、企業、専門家、メディア等が関与しており、ステークホルダーの関係性も複雑です。これまで繰り返し述べてきたように、「法令に従って設計されたから安心」といった考え方はすでに限界を迎えています。むしろ、リスクが潜在することを前提に、それらがどのように検出・制御・再設計されているかを、社会に対して説明し、共に評価する枠組みが求められています。

このため、専門知識を持たない行政の担当者や、様々なステークホルダーが容易に取り扱える評価モデルを共通言語として持ち、プロジェクト全体がチェックされるプロセスが正常に機能することで、社会的信頼が醸成されるよう努めることが重要になってきます。

- システムにどのようなハザードが潜在しうるかが構造的に可視化されていること
- 想定される使用エラーや設計バイアスについて具体的に説明可能であること
- 監査者・事業者・利用者のあいだで、評価基準とプロセスの共通理解が形成されていること

本ガイドブックで紹介してきた「スリップ」「ミステイク」「ラプス」といった分類や「安全制約」等の概念は、専門知識のないステークホルダーでも直観的に理解しやすく、合意形成の認識の土台として活用しやすいツールです。

10.4.2 ユーザビリティ・安全に関するプロジェクト内での取組

最後に、組織内での取組について見ていきましょう。まずはPJMOの立場で、どんな準備をするべきかを説明します。ユーザビリティ・安全の説明責任を果たすためには、単に「UIチェックリスト」（デジタル庁）等を用いて取組状況をチェックするだけでは十分ではありません。設計・運用の対象となる（広義の意味での）システムが「実際の利用状況において、どのようなパフォーマンスを発揮しているか」を評価軸とするパフォーマンスベースのアプローチと、「どんなリスクを想定していて、どんな対策を行っているのか」を内外に説明できる状態を整えること（RIDM）が必要不可欠です。

目的の明確化と共有

プロジェクトの安全・品質・UXに関する目標が関係者に共有されていること。目標は抽象的でなく、「誰が・どこで・なにを・どう使う」かを具体的に踏まえて設定されること。

リスクとベネフィットの可視化

使用エラーや不確実性による影響（損失）の可能性と、それに対する対策コストのバランスを確認・文書化し、組織的に合意すること。このプロセスには、ヒヤリハットの蓄積やSTPAによる潜在的リスクの構造的把握が含まれる。幅広い関係者からリスクに関する視点が提供され、フレーミングの罠（問題の枠組みフレームの設定の仕方によって、判断や認識が偏ってしまう認知バイアス）に陥らないようにすること。

パフォーマンスベースの評価

実使用状況に即した検証（例：シナリオベース評価、クリック数、エラー率、達成時間）。評価者間のばらつきを最小化するためのプロトコル（手続き・手順）が整備され、計測できるようになっていること。検証者間での教育・研修が行われ、定期的に指標や、環境の変化等を反映した活動が行われているか確認できること。システムのパフォーマンスに関する評価は、「設計文書」だけでなく、「ユーザビリティテストの記録」「インシデントログ」「UIプロトタイプ」「A/Bテストの記録」「コールセンターへの問合せ」等の現物・エビデンスを元にした観察と対話によって運用されていくことが重要です。

心理的安全性を前提とした評価環境・文化の醸成

「問題を指摘する側」も「指摘される側」も無闇に攻撃されない、対話型のフィードバックの仕組み。CAST等を活用し、「なぜ起きたか」ではなく「なぜそうすることが合理的だったか」を共に問い直す文化が、実際のシステムの運用に関わる外部事業者等を含めて形成されるように取組むことが大切です。

10.4.3 ユーザビリティ・安全に関する横断的・組織的な意思決定

これまでも繰り返し説明してきたように、プロジェクトの評価やタイミングが属人的にならないように工夫し、PJMOが「何に取り組めば良いのか」「どんな準備が必要か」予見でき、十分に準備の時間や工数を確保するよう組織的に支援していく必要があります（リソースが十分に確保されていない状態や、やるべきことが詰め込まれすぎた状態をできる限り回避しなければなりません）。

プロジェクトのリスクやハザードを整理し、それに対する対策コストのバランスを確認して、対策を立てていくためには、各プロジェクトだけではできないことが出てきます。例えば以下のような問題です。

- 安全の基準を定義するのに、十分な情報がまだPJ内にない
- リスク・ハザードの定義に、他のプロジェクトのリスク情報が必要である
- ある取組の優先順位が、プロジェクト単体で決められない（組織的決断が必要である）
- 誰に、どのようなコミュニケーションをどのタイミングで取るべきかが定まらない
- 複数のステークホルダーが安全確保において重要な役割を担っている

こうした課題を乗り越えていくには、普段からプロセスの標準化や、十分に予見可能な対策の実施に向けた情報・基準の管理が欠かせませんし、適切な記録管理が行われること、適切にスケジューリングされること等も大切です。開発者・PJMO・利用者代表等多様なステークホルダーと随時コンテキストの共有が行われ、合意形成のプロセスが段階的に踏まれていかないと、全員にとって負担が大きすぎることになるからです。

しかし、異なる階層・組織間でのコミュニケーションコストを低減させ、潜在リスクを早期に洗い出すためには［10.3.1 安全文化のための制度的支援］で示したような組織的な基盤が欠かせません。ぜひ、組織全体で、安全・安心なシステムの構築に向けて取り組んでいただきたいと思います。

10.5 まとめ（安全とユーザビリティ、サービスを両立させるために）

本ガイドブックでは、情報システムにおける使用エラーとユーザビリティの課題を出発点に、ヒューマンファクター、認知科学、システム工学、安全設計、リスク評価の各分野にまたがる知見を踏まえて、実践的かつ構造的なアプローチを紹介してきました。

ATSモデルやGEMSモデルを通じて、なぜ人は誤るのか、どこで支援すればよいのかを可視化する方法を紹介しました。また、FMEAやFTA、HAZOPといった定量・定性的な分析手法に加え、STAMP/STPA等のシステム思考に基づいた手法によって、従来では見過ごされがちだった相互作用や組織文化によるリスクへの対応も可能になることを示しました。

そして、本章ではRisk-Informed Decision Making（RIDM）や安全文化の醸成といった視点を導入し、「評価しただけで終わらない」「リスクを可視化し、判断に活かす」という実務上の要求に応える設計プロセスを紹介しました。ここでは、心理的安全性を担保しながら、関係者が対話を通じて学び続けることが、事故の未然防止だけでなく、信頼されるサービス設計に不可欠であることを強調しています。

従来、ユーザビリティと安全性やセキュリティ、プライバシーといった観点はしばしば別々に扱われがちでしたが、本書を通じて、各要素は本質的に密接に結びついており、「人が安心して使えること」こそが、「システム全体の信頼性を支える基盤」であることをご理解いただけたかと思います。

現代の情報システムは、かつてないほど多くの人と技術が複雑に絡み合い、設計や運用の判断が難しくなっています。その中で私たちは、「エラーが起きないように人を管理する」という旧来型の発想から、「エラーが起きることを前提に、どう安全に制御するか」というレジリエンス（回復力）を軸とした発想へと転換する必要に迫られています。本ガイドブックが、そうした思考と実践の橋渡しを担い、より多くの機器・サービスが人にとって使いやすく、安全で、信頼できるものとなる一助になれば幸いです。

11

付 録

11.1 リンク集

ユーザビリティに関するツールや情報は、多くがインターネットで公開されています。

利用・配布

- 公共データ利用規約（第1.0版）
- コンテンツの利用に係るPDL1.0に関する重要情報
- デジタル庁　コピーライトポリシー｜デジタル庁

UXデザイン

- 図 User Experience Design, Peter Morville
- Jakob Nielsen and Thomas K. Landauer. 1993. A mathematical model of the finding of usability problems. In Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93). Association for Computing Machinery, New York, NY, USA, 206–213.

STAMP

- IPA 独立行政法人 情報処理推進機構 複雑化したシステムの安全性確保（STAMP）| 社会・産業のデジタル変革 | IPA 独立行政法人 情報処理推進機構

11.2 参考文献

1. The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. 1979. The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research. U.S. Department of Health and Human Services. https://www.hhs.gov/ohrp/regulations-and-policy/belmont-report/index.html
2. Martha C. Nussbaum. 2011. Creating Capabilities: The Human Development Approach. Harvard University Press, Cambridge. ISBN: 978-0674072350.
3. マーサ・ヌスバウム. 2025. ケイパビリティ・アプローチとは何か：生活の豊かさを測る. 池本幸生 and 栗林寛幸 訳. 勁草書房. ISBN: 978-4326154944.（Nussbaum, 2011 の邦訳）
4. 平沢尚毅 and 福住伸一. 2023. 顧客経験を指向するインタラクション――自律システムの社会実装に向けた人間工学国際標準. 日本経済評論社. ISBN: 978-4818826328.
5. 福住伸一 and 平沢尚毅. 2024. 詳説 ユーザビリティのための産業共通様式 CIF: Common Industry Format for usability. 近代科学社Digital. ISBN: 978-4764960787.
6. Jakob Nielsen and Thomas K. Landauer. 1993. A mathematical model of the finding of usability problems. In Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93). Association for Computing Machinery, 206–213. DOI: https://doi.org/10.1145/169059
7. John Brooke. 1996. SUS: A "quick and dirty" usability scale. In Usability Evaluation in Industry, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland (Eds.). Taylor & Francis, London, 189–194. DOI: https://doi.org/10.1201/9781498710411-35
8. 甘利昭一郎, 大橋敦, 中口俊哉, 金井Pak雅子, 金太一, 五十嵐歩, 笹川恵美, 細野美奈子, 藤原道隆, 吉岡直紀, and 小山博史. 2025. 言語的妥当性を担保した日本語版System Usability Scaleの作成. VR医学 23, 1. 日本VR医学会. DOI: https://doi.org/10.7876/jmvr.23.1
9. Charles E. Osgood, George J. Suci, and Percy H. Tannenbaum. 1957. The Measurement of Meaning. University of Illinois Press, Urbana, IL. ISBN: 978-0252745393.
10. Alan D. Swain and Henry E. Guttmann. 1983. Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications. NUREG/CR-1278. U.S. Nuclear Regulatory Commission. DOI: https://www.nrc.gov/docs/ML2008/ML20085K326.html
11. Jens Rasmussen. 1983. Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. IEEE Transactions on Systems, Man, and Cybernetics SMC-13, 3 (May-June 1983), 257–266. DOI: https://doi.org/10.1109/TSMC.1983.6313160
12. Donald A. Norman and Daniel G. Bobrow. 1976. On the role of active memory processes in perception and cognition. In The Structure of Human Memory, C.N. Cofer (Ed.). W.H. Freeman, 114–132. ISBN: 978-0716707158.
13. David E. Rumelhart and Andrew Ortony. 1977. The representation of knowledge in memory. In Schooling and the Acquisition of Knowledge, R.C. Anderson, R.J. Spiro, and W.E. Montague (Eds.). Lawrence Erlbaum Associates, 99–135. DOI: https://doi.org/10.4324/9781315271644
14. Donald A. Norman. 1981. Categorization of action slips. Psychological Review 88, 1 (January 1981), 1–15. DOI: https://doi.org/10.1037/0033-295X.88.1.1
15. Donald A. Norman. 2013. The Design of Everyday Things: Revised and Expanded Edition. Basic Books, New York. ISBN: 978-0465050659.
16. D. A. ノーマン. 2015. 誰のためのデザイン？増補・改訂版――認知科学者のデザイン原論. 岡本明, 安村通晃, 伊賀聡一郎, and 野島久雄 訳. 新曜社, 東京. ISBN: 9784788514348.（Norman, 2013 の邦訳）
17. James Reason. 1987. Generic error-modelling system (GEMS): a cognitive framework for locating common human error forms. In New Technology and Human Error, J. Rasmussen, K. Duncan, and J. Leplat (Eds.). John Wiley & Sons, 63–83. ISBN: 978-0471910442.
18. Herbert W. Heinrich. 1931. Industrial Accident Prevention: A Scientific Approach. McGraw-Hill.
19. H. W. ハインリッヒ, D. ピーターセン, and N. ルース. 1982. ハインリッヒ産業災害防止論. 総合安全工学研究所 訳, 井上威恭 監修. 海文堂出版, 東京. ISBN: 978-4-303-58051-3.（Heinrich et al., 1980 第5版の邦訳）

20. James Reason. 1990. Human Error. Cambridge University Press. DOI: https://doi.org/10.1017/CBO9781139062367

21. Frank H. Hawkins. 1993. Human Factors in Flight, 2nd ed. Ashgate Publishing. ISBN: 978-1857421354.

22. Erik Hollnagel. 2006. Resilience: the challenge of the unstable. In Resilience Engineering: Concepts and Precepts, E. Hollnagel, D.D. Woods, and N. Leveson (Eds.). Ashgate Publishing, 9–17. ISBN: 978-0754649045.

23. 小松原明哲 . 2019. ヒューマンエラー , 第 3 版 . 丸善出版 . ISBN: 978-4621304358.

24. Nancy G. Leveson. 2012. Engineering a Safer World: Systems Thinking Applied to Safety. MIT Press. DOI: https://doi.org/10.7551/mitpress/8179.001.0001 ISBN electronic: 9780262298247

25. 兼本茂 and 福島祐子 (監訳) . 2022. システム理論による安全工学――想定外に気づくための思考法 STAMP. 共立出版 . ISBN: 9784320072039. (Leveson, 2012 の邦訳)

索引

索引

索引

索引