IPA Better Life with IT

障害未然防止のための
設計知識の整理手法
ガ イ ド ブ ッ ク

組込みシステム 編

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

障害未然防止のための設計知識の整理手法ガイドブック

独立行政法人情報処理推進機構

障害未然防止のための設計知識の整理手法ガイドブック

目次

1 はじめに 1
1.1 背景・目的 1
1.2 本ガイドブックの位置付け 3
1.3 ソフトウェア障害の発生を未然防止する設計知識 4
1.4 設計知識の活用シーン 5
2 設計知識の体系化 7
2.1 設計知識モデル 7
2.1.1 知識として伝えるべき事 7
2.1.2 知識の文脈表現 9
2.1.3 設計知識の構造 9
2.2 知識の再利用モデル 10
2.2.1 キーワード抽出 11
2.2.2 分類情報の抽出 12
2.2.3 知識分類の体系イメージ 13
2.2.4 分類タグ 14
3 設計知識の整理手法 16
3.1 概要 16
3.2 設計知識の抽出 17
3.2.1 知識要素の抽出 17
3.2.2 知識要素の一般化 21
3.3 分類タグ設定 22
3.3.1 キーワード抽出 23
3.3.2 知識の分類 24
4 検索ツールのイメージ 27
4.1 検索キー 27
4.2 検索結果 28
5 設計知識の整理サンプル 30
参考文献 43

1 はじめに

本ガイドブックでは、いわゆる「過去トラDB（過去トラブルデータベース）」と呼ばれる障害情報データベースのソフトウェア障害情報の記録から、障害発生を未然防止するための設計知識を抽出し、有効活用できる形に整理する方法を提案する。障害情報記録から設計知識を抽出するための観点を示し、抽出した設計知識を構造的に整理することが出来れば、「過去トラDB」が再発防止や未然防止のために活用できるようになる。

ソフトウェア障害は、設計書やプログラムコードに作り込んだ不具合や回避しなければならない状況への対策不備をレビューやテストで発見できなかった場合に発生する。原因がうっかりミスの場合は別として、設計者の設計知識が乏しければ、障害を発生させないようにする処理を設計書に書くこともプログラムコードに書くこともできない。それを補うのが、設計書やテストケースのチーム内レビューや共同レビューであるが、レビューアも設計知識が乏しいと起こりうる障害の可能性を指摘することすら出来ない。ソフトウェア障害の未然防止に最も必要なことは、レビューやテストの実施以上に設計者やレビューアが適切な設計知識を有することである。

本資料で提案する手法は、ベテラン技術者の経験を若手技術者が自分のものとして利用できることを目指している。

障害の再発防止や未然防止の活動に「過去トラDB」の活用を考えているソフトウェア品質部門の方には、是非この方法をご自身の部門の障害情報記録で試していただきたい。

1.1 背景・目的

情報処理システムや組込みシステムを開発するベンダー企業は、過去の障害事例を一定の様式で記録し蓄積した障害情報データベースを、類似障害の再発防止や未然防止のために活用したいと考えている。しかし、実態として社内規定の中に障害情報データベースを参照するようにチェックリストを設けているにも関わらず類似障害を引き起こしている例もある。ベンダー側だけでなく、社会インフラシステムを運用している企業や組織においても同じような取り組みをしている。そのようなデータベースは、一般に「過去トラ」や「過去トラDB（データベース）」と呼ばれている。再発防止や未然防止のために内部で公開していても、書かれている内容が整理されていないため当事者以外の人にとっては理解しづらく、また蓄積した件数が次第に増えて来ると、さらに、有効情報の取り出しに手間がかかるため、次第に活用されなくなっていく。

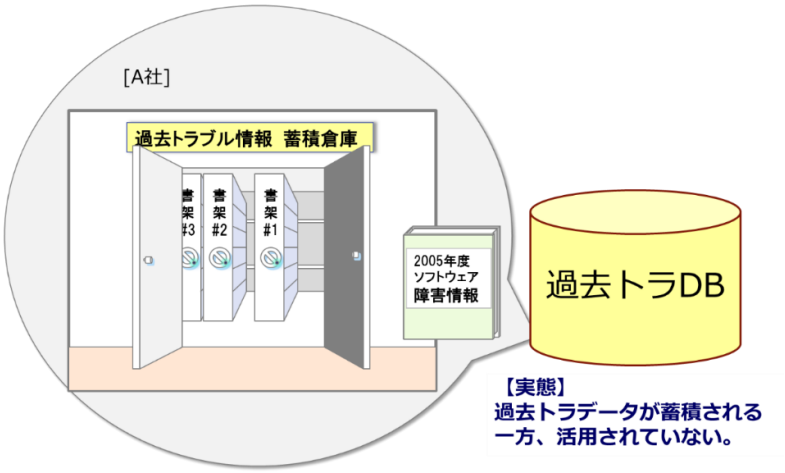


図1.1　活用されない過去トラデータベース

「過去トラDB」が有効活用されない理由は、障害を防止するためのノウハウが整理されておらず、表面的な障害の事象とどこにどんな対処をしたかぐらいの情報しか書かれていないためである。検索キーワードを入力してデータベースから関連する情報を取り出そうとしても、うまくヒットせずに取り出せないことがある。ヒットした場合でも、「過去トラDB」は、一般に障害事例の原因と対処が装置やサービス固有の具体的な表現で記載されているため、記載内容を見る利用者は、自身と無関係な障害事例として扱う傾向がある。

一方、ソフトウェアの高信頼化を実現する上で、業界が抱える課題の一つに、ベテラン技術者の豊富な経験やノウハウの伝承がある。ベテラン技術者の頭の中には、類似障害が起こらないように原因と対処が一般化された形で頭の中に残っていると考えられる（図1.2)。「過去トラDB」には、ベテラン技術者の豊富な経験やノウハウの断片が埋蔵されている。

したがって、「過去トラDB」を障害の再発防止や未然防止の活動に活用するためには、障害情報記録を後の人に伝承できるような形に一般化した設計知識に変換し整理する必要がある。

「過去トラDB」の情報を設計知識の形に整理する方法として、部品や材料等ハードウェアに関する方法は紹介されている（参考文献[1]）が、ソフトウェアに関する整理方法は、見当たらない。

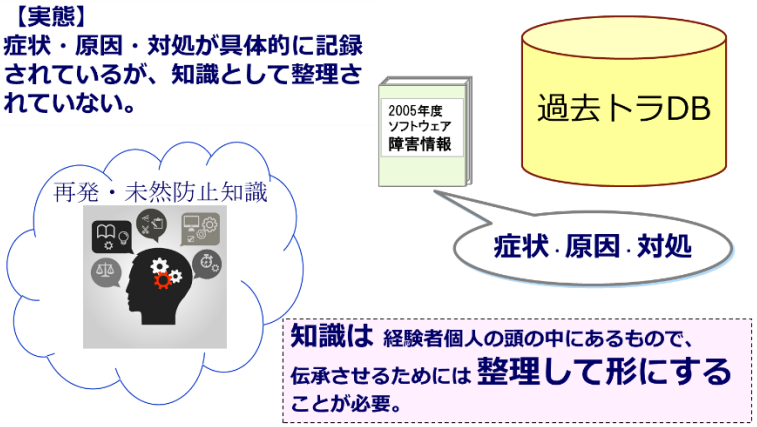


図1.2　経験者個人の頭の中では知識は一般化できている

1.2 本ガイドブックの位置付け

IPA/SEC では、ソフトウェア障害を未然に防止するためのノウハウを広く業界に伝えるために、次のドキュメント（教訓集シリーズ）を発行している。

- 「情報処理システム高信頼化教訓集（組込みシステム編）」（参考文献[2]）
- 「障害未然防止のための教訓化ガイドブック（組込みシステム編）」（参考文献[3]）
- 「現場で役立つ教訓活用のための実践ガイドブック（組込みシステム編）」（参考文献[4]）

教訓集シリーズは、障害を未然に防止するノウハウを障害事例から学び“教訓”として伝えている。障害事例には、要求分析に関するノウハウ、設計に関するノウハウ、テストに関するノウハウ等、ソフトウェア開発の中で必要なノウハウが幾つか含まれている。教訓は、その中から強く伝えたいノウハウを選び、インパクトのある言い回しで表現している。

一方、本ガイドブックは、ソフトウェア障害を未然に防止するためのノウハウを伝えるという点では同じであるが、伝えたいノウハウは設計ノウハウに絞り込み、それを障害の未然防止のための対策が可能なレベルの設計知識に変換している。

教訓集シリーズの「教訓」と本ガイドブックにおける「設計知識」の位置付けを図 1.3 に示す。障害を未然に防止するための「教訓」は、“～するべからず”のようなレベルで抽象化され、必ずしも具体的な対策までは言及しない。一方、「設計知識」は、製品ドメインに依存しない抽象度で対策まで言及する。

本ガイドブックの整理手法は、教訓集シリーズに紹介された障害事例を題材にして、そこから設計知識を抽出する。また、知識化に必要な“一般化”の考え方や、知識整理の体系化の考え方は、障害事例を教訓化する場合と同であり、教訓集シリーズの“一般化”や“観点マップ“に準じている。

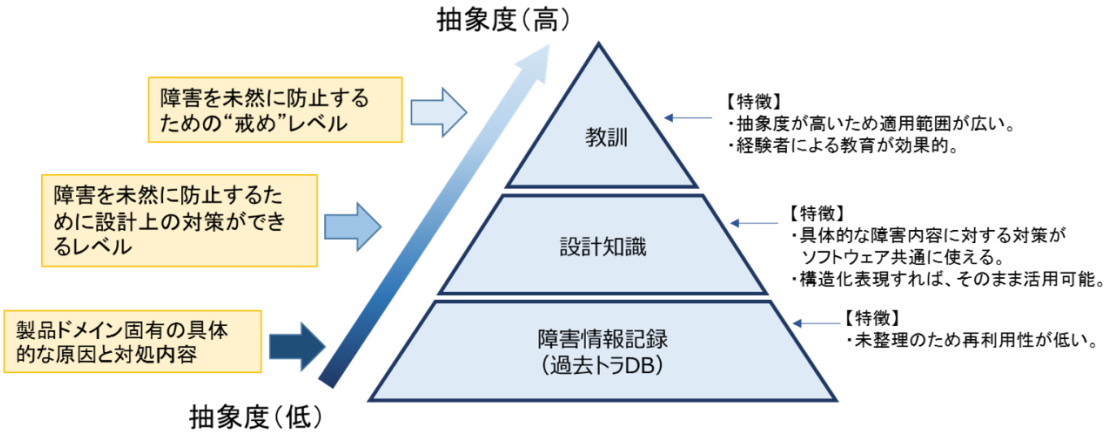


図 1.3　教訓と設計知識の位置付け

1.3 ソフトウェア障害の発生を未然防止する設計知識

近年のソフトウェアは大規模化・複雑化に伴って品質を担保することが困難になってきており、次のような開発手法の知識を活用して品質向上を図っている。

- リスク分析に関する知識（例：HAZOP、STAMP（参考文献[5]））
- レビュー手法に関する知識（例：ピアレビュー、インスペクション）
- テスト技法に関する知識（例：同値分割、境界値分析）

これらの知識は当然のことながら、ソフトウェア障害を未然防止するために活用されるが、管理面、技術面の手法や手段に関するものであり、これらだけでは、プログラムコード等に障害を回避する適切な処理を施すことはできない。

そのような対処に必要な知識は開発対象に対する設計知識であり、未然防止の観点では、「過去トラ DB」の障害情報記録に記載されている「直接原因」や「対処内容」から得ることができる。そこから得られる知識は、信頼性の適切な評価尺度として活用できる。

【用語の定義】
「知識」ある事柄などについて、知っている内容（デジタル大辞泉）
「設計」機械類の製作や建築・土木工事に際して、仕上がりの形や構造を図面などによって表すこと。（大辞林）
「設計知識」機能や処理を設計することが出来る知識。その機能や処理についての知識が無ければ設計出来ない。ここで言うところの機能や処理はソフトウェアで実現する機能や処理。（本ガイドブックにおける定義）

「情報処理システム高信頼化教訓集（組込みシステム編）2015 年度版」の障害事例には、ソフトウェアが意図しない動作をして障害に至った事例や、システム構築時のミスが原因で発生した障害事例、運用時や保守作業時にオペレータの操作ミスや判断ミスが原因で引

き起こされた障害事例が含まれている。ソフトウェア設計時に、機器やシステムの使用者が障害を誘発しないようにするための設計知識があれば、そのような障害を防止したり被害の拡散を回避したりできる。

一方で、設計知識の不足ではなく要求事項を文書で確認することを怠ったために、仕様を取り違えてしまった事例があり、このようなプロセスやプロジェクトマネジメントに起因する障害事例もある。

ソフトウェアで実現する機能は、装置やシステムの製品ドメインが異なっても共通に扱えるものが多くある。例えば、「起動処理」、「割り込み処理」、「バッチ処理」等があげられる。そのため、ソフトウェアの設計知識は、製品ドメイン固有の表現を避けることで、製品ドメインに依存しない共通の設計知識として活用できる。

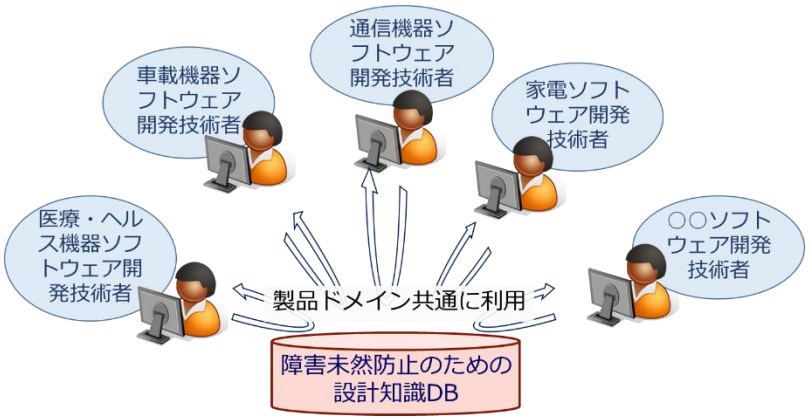


図 1.4 製品ドメインに依存しない共通の設計知識

1.4 設計知識の活用シーン

設計知識の整理方法を検討する上で、整理された設計知識がどのようなシーンで活用されるのかを想定し、そこから整理の方針を導く。

設計知識は、ソフトウェア設計担当者にとって要求事項を分析し必要な機能・処理の実現方法を検討する際には不可欠な知識である。何もないところから検討するのは大変なので過去の設計資料を参照したり、ベテラン技術者に聞くなどして設計知識を得る。設計知識が一般化表現で DB 化されていれば、設計担当者にとって当然役立つ。一方、レビューアも同じ設計知識 DB を参照できれば、設計担当者がその対象製品に製品ドメイン共通の設計知識をどのように実装するのか想像できるため、レビューに先立って適切な指摘事項を準備できる。また、設計知識はテストケース作成時やそのレビュー、障害対策等、ソフトウェア開発の基盤としても不可欠である。

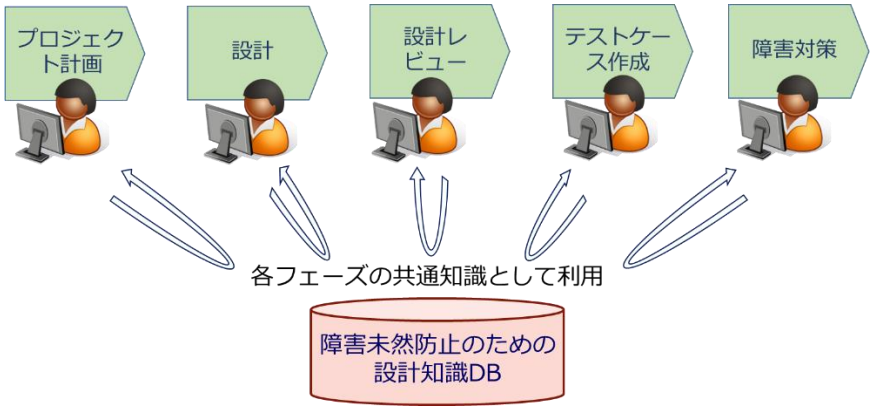


図 1.5 設計知識の活用シーン

設計知識が活用される場面は、図 1.5 に示すようにソフトウェア開発のプロジェクト計画時からフィールドに出荷された後の障害対策まで広い。それらの活用例を列挙する。

① プロジェクト計画書作成（プロジェクトリーダ）

例）類似機器で過去に発生した障害に関する防止知識を検索したい。

例）今回初めて使うデバイスに関する知識を探したい。

② 設計作業（担当者）

例）今回開発する機能で、設計時に考慮漏れし易い視点・観点を知りたい。

③ 設計レビュー（レビューア）

例）今回開発する機能で、レビューアがビジネス目線や経営目線で障害発生時の対策レベルを伝える際に、設計担当者と同じ設計知識を共有した上で指摘したい。

例）各機能について、チーム内部でピアレビューを実施する際に、考慮漏れし易い設計視点・観点を抽出し、その知識から連想される他の障害の有無等を相互に指摘し合いたい。

④ テストケース作成（テストケース作成者）

例）今回開発した機能で、考慮漏れし易い設計視点・観点を抽出してテストケース作成やレビューで確認したい。

⑤ 障害対策（対策方針検討者）

例）出荷前、出荷後の障害の症状から、考えられる原因と対策案を調べたい。

⑥ 各シーン共通

検索した結果得られた情報から関連する知識を連想させたい。

これらの活用を効果的に行えるように、設計知識の整理の仕方を考える。

2 設計知識の体系化

ソフトウェア障害の発生を未然防止する設計知識は、「過去トラ DB」の障害情報記録から「考慮不足」、「考慮漏れ」、「知識不足」に起因する事例を探し出し、それをじっくりと理解すれば暗黙知として得ることができる。しかしながら、たいていの障害情報記録は、障害を引き起こしたプログラムコードの事実とそれをどのように修正したのかの内容が淡々と記載されただけで、何故そのようなプログラムコードになってしまったかの背景（なぜ対応しなかったか、なぜ間違ったのか等）が記載されていない。また、一般に技術者は、自責を感じる原因でも仕様書不備等の問題として障害情報に記録する場合があるため、背景に「考慮不足」、「考慮漏れ」、「知識不足」の要因が含まれていても、それを判断することが難しい。しかし、頭と時間を使って、そこから不足していた設計知識を抽出して、その知識を形式知化することが出来れば、確実に効率良く設計ノウハウが習得できる。そこで知識が頭の中にスーッと入ってくる文脈とその知識文脈の構造化を考え、さらに効果的に知識習得を促せる工夫をする。

一方で、伝えなければならない知識が膨大になると、利便性を高める工夫が要る。１つは、その知識が何に役立つ知識なのか一目見て、もしくは一言聞いて分かる工夫、２つ目は、知識が体系的に整理できていることで、一目見て知識が頭の中に入ってくれば、知識の吸収性が高まる。知識が体系的に整理できれば、関心のある分野を選択して知識をデータベース等から検索できる。これらの工夫は、製品ドメインの個別事例から抽出した知識を出来るだけ共通的な知識として再利用を図るために行う。

2.1 設計知識モデル

ソフトウェア障害未然防止のための設計知識は、知識の利用者が、直感的に不具合の混入と障害発生のメカニズムの文脈を頭の中で組み立てられなければ、うまく人に伝わらない。そのためには、知識を知識要素で構成し要素間の関係を構造的に整理した知識のモデル表現が必要になる。

2.1.1 知識として伝えるべき事

ソフトウェア障害の発生を未然に防止するノウハウを設計知識としてどの様に伝えるべきかを考える。

伝えるべきことは、考慮漏れ等設計知識の不足に起因した障害情報記録の「直接原因」とその「対処内容」である（図 2.1）が、そのままでは頭に刺さらない。頭に刺さるようにするためには、「直接原因」を掘り下げて「障害発生のメカニズム」を調べ、設計知識の不足が障害を発生させる要因であることを伝える必要がある。

つまり、伝えるべきことは、経験の浅い技術者にとっては誰かに教わらなければ知り得ないことでもベテラン技術者であれば過去の経験から思いつき、あらかじめ設計に盛り込むことができるような処置である。障害は、図 2.2 に示すように「考慮が漏れていた設計視点・観点」と「問題を引き起こすトリガー（発生契機）」によって発生するシナリオとして表現できる。

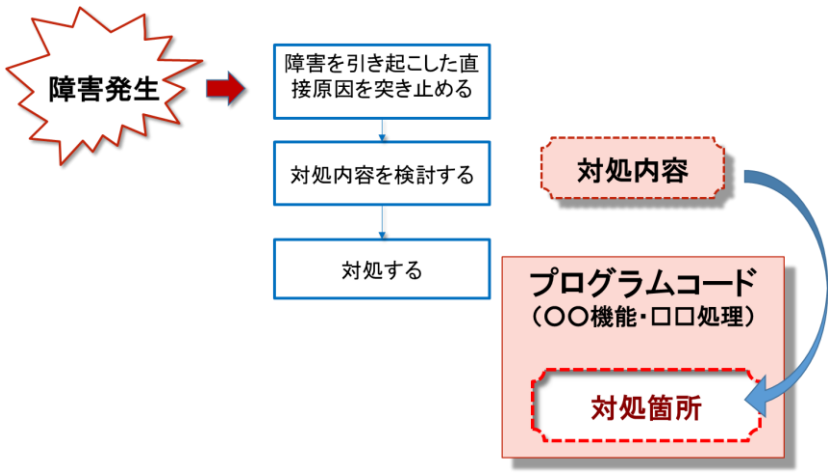
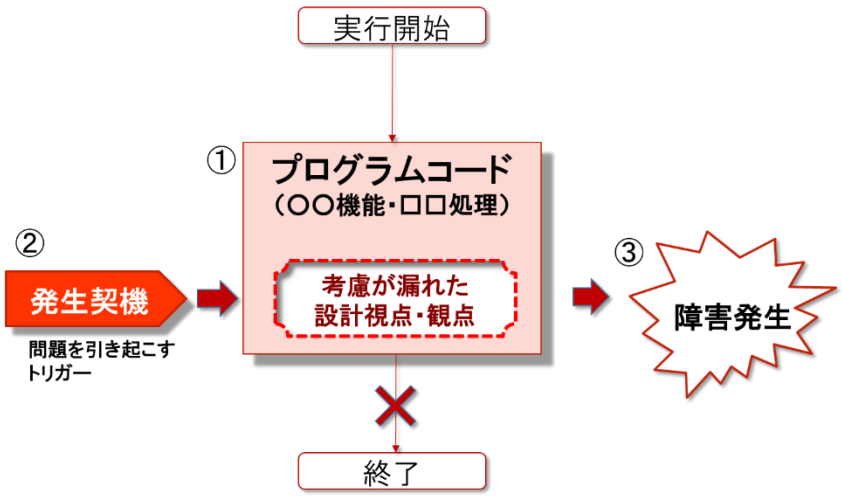


図 2.1　直接原因への対処



① 問題を引き起こす可能性のある事象に関して、考慮が必要な設計視点・観点が漏れたままプログラムコードを作成し実装する。

② 障害を引き起こすトリガーがかかる。

③ 障害が発生する。

図 2.2　考慮漏れに起因する障害発生のシナリオ

図 2.2 のように考慮が必要な設計視点･観点が欠落したままプログラムコードを書いたとしてもトリガーがなければ障害を引き起こすまでには至らない。

ソフトウェア要求仕様書に記載された機能や処理を作り込む（設計と実装）際に、その製品やシステムの知識や経験が不足していると、問題を引き起こす可能性のある事象等の「考慮漏れ」が起こる。考慮が漏れた観点や視点で作り込んだソフトウェアに対し、考慮漏れによる不具合箇所に問題を引き起こすトリガーとなる事象が発生すると、障害を引き起こしてしまう。

プログラムコードに不具合を含んでしまう考慮漏れ要因とその不具合によって障害が発生するトリガーが何かを頭の中に明確にイメージすることができると、考慮が漏れていた設計視点・観点が明確になり、その障害を起こさないようにするための知識として頭の中に入って来る。

2.1.2 知識の文脈表現

前項 2.1.1 では、図 2.1「直接原因のへの対処」と図 2.2「考慮漏れに起因する障害発生のシナリオ」から、設計知識を構造的に整理するための知識要素（候補）とその関係を表現した。知識要素（候補）は、図 2.1 の「対処内容」や「対処箇所（機能・処理)」、図 2.2 の「考慮が漏れた設計視点・観点」、「発生契機」等である。知識要素間の関係を考慮して、これらの知識要素を繋げて意味の通る日本語に変換すると、伝わり易い設計知識の文脈ができる。

ソフトウェアを設計する段階で障害の発生を未然に防止する方法として、「○○○が（何が)」×「△△△すると（どうしたら)」×「◇◇◇する（どうなる)」の文脈で障害発生のシナリオをパターン化すると、不具合混入原因の発想を促す効果があることが報告されている（参考文献[6][7])。この文脈は、頭の中に整理されやすい文脈でもあると考え、この文脈を参考にして、障害未然防止のための設計知識の文脈を次のように考えた。

設計知識の文脈：
「○○○の機能や処理を考えるときに、
▽▽▽の考慮が漏れていると、
△△△が起こった契機で◇◇◇◇の障害が発生する。
その障害の発生を防ぐためには□□□□の処理を作り込んでおく。」

2.1.3 設計知識の構造

障害の発生を未然に防止するための設計知識をパターン化して頭の中に整理し易くするために、2.1.2 項で考えた設計知識の文脈から知識要素を抜き出し、それらを構造的に表現する（図 2.3)。図 2.3 の知識要素(1)～(4)は、不具合を混入させて障害が発生するシナリオの構成要素を表す。知識要素(5)は、(1)～(4)の要素を文章に組み立てた「何が」、「どうして」、

「どうなる」で、その障害が発生するメカニズムを表す。 (6)は対策を表す。(5)と(6)は文章で簡潔に説明する。

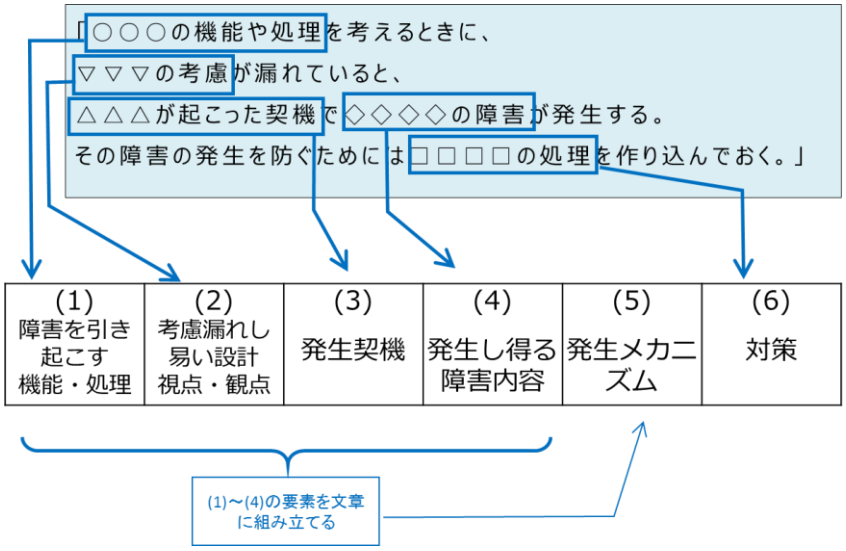


図 2.3 設計知識の構造

上記の(1)～(4)の要素は、これらの要素だけで直観的に障害発生のシナリオがイメージできるように要素の表現を選ぶ。設計知識を製品ドメインに依存しない機能ドメインの知識として広めるために、各要素はできるだけ一般化した用語で簡潔に表現する。一般化は、得られた知見の適用を広く求めるために行う作業で、１）まず適用する製品・システム・組織の範囲を想定し、２）次にその範囲で共通的な性質に知見を置換・転換する。一般化の考え方は、本資料に関連する「障害未然防止のための教訓化ガイドブック（組込みシステム編)」（参考文献[3]）に記載される「一般化」に基づいている。

このように(1)～(4)の要素を一般化することで、(5)の発生メカニズムを丁寧に見なくてもそこに書かれている説明が想像でき、また(6)の対策も推測することができる。

2.2 知識の再利用モデル

前節では、「過去トラ DB」が利用されないという課題を解決するために、「過去トラ DB」の中の障害事例から設計知識を抽出することを提案した。しかしながら知識が増えてくると、件数が膨大になると「過去トラ DB」が有効活用されなくなることと同様の問題が発生する。そのため、設計知識の再利用を促すための工夫をする。

工夫のポイントは、

（１） 知識体系がイメージできること。（知識の分類）

（２） 知識が DB 化されたときに探し易いこと。（探し易いタグの設定）

（３） 探した知識情報を一目見てどんな知識であるのか分かり易いこと。（適切なキーワ

ードの設定）

これら工夫のポイントに共通していることは、設計知識の更なる抽象化である（図 2.4）。前項 2.1.3、図 2.3 の構造に整理した設計知識から再利用性を高める情報を抽出する。

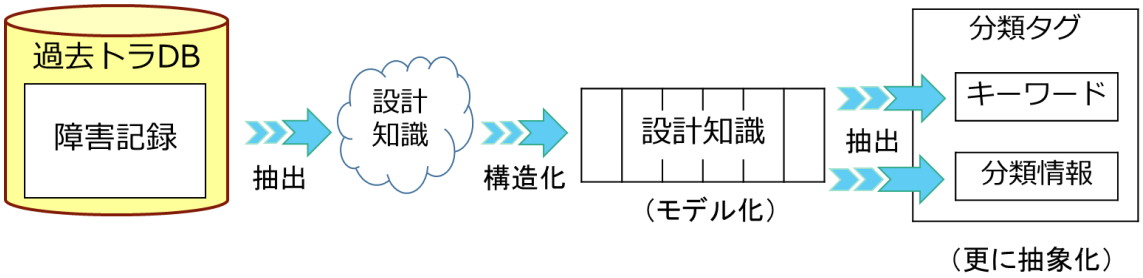


図 2.4　設計知識の再利用性を高める更なる抽象化

2.2.1　キーワード抽出

障害を未然に防止する設計知識の構造（図 2.3）は、頭の中に整理し易く、知識を個別に理解する上で効果的である。一方で伝えたい知識、習得したい知識が増えてくると、知識の吸収に時間がかかるため一目見て若しくは一言聞いてその知識が何に役立つ知識なのか分かるような工夫が要る。

図 2.5 に示すように、障害が発生する観点で、設計知識の文脈から「何が」「どうなる」をキーワードとして抜き出せば、何に役立つ知識なのか直観的に理解できる。「何が」は、図 2.3 の知識構造の(1)障害を引き起こす機能・処理から、「どうなる」は、(4)発生し得る障害内容から抽出し、抽象化する。

キーワード抽出の考え方は本資料に関連する「障害未然防止のための教訓化ガイドブック（組込みシステム編）」（参考文献[3]）に掲載された「直接原因観点マップ」に基づいている。

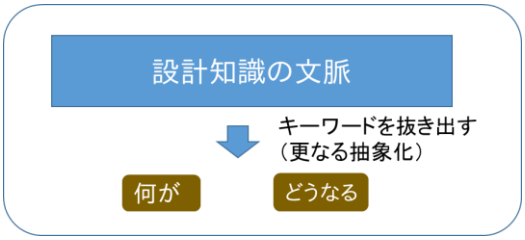


図 2.5　キーワード抽出

人の頭は、理解した知識が頭の中で増えてくると、類似の知識を連想することが出来る。つまり、ソフトウェア障害の発生とその障害の解決を多く経験したベテラン技術者は、開発に携わったことの無い別のシステムのレビューに参加した場合でも、様々な知識を持っ

ているため、効果的な指摘を行うことができる。これは、頭の中に蓄積している特定の障害事例から得た知識を利用して、同じ機能を持つ他のシステムでも同様の障害が発生することを予測していると言える。連想を促す仕組みは、ベテラン技術者の頭の中で、具体的な知識を抽象的に整理できているからだと考えられる。

2.2.2 分類情報の抽出

知識が体系的に整理できれば、関心のある分野を選択して知識をデータベース等から検索できる。整理のポイントは、ソフトウェア技術者の頭の中でどんな切り口で整理したいかである。知識が頭の中に整理されていると思考時に参照できるため、応用力が高まる。

このような観点で設計知識を

- 機能・処理
- デバイス・機器
- 混入プロセス

で分類する（図 2.6）。

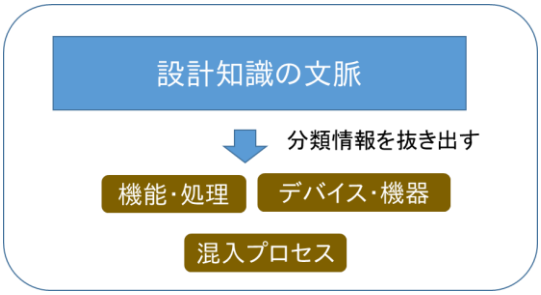


図 2.6　分類情報抽出

設計知識は、再利用性を高めるためにできるだけソフトウェア共通に使われる機能名や処理名で分類する。一方で、特定のデバイスや機器に特化した知識もあり得るため、デバイスや機器の分類情報も設ける。障害事例の中には、知識不足を補うよりもプロセスを補う（例：自動化できることはツール利用を規定する等）ほうがはるかに効果的な場合もあり、障害原因の混入プロセスによる分類も可能にする。

分類情報の「機能・処理」、「デバイス・機器」抽出は、前項のキーワードと同様に、本資料に関連する「障害未然防止のための教訓化ガイドブック（組込みシステム編）」（参考文献[3]）に掲載された「直接原因観点マップ」の“機能”や“デバイス”等の考えに基づく。

再利用が促されるポイントは、知識を整理する際の抽象化と具体化のバランスなので、分類タグには、両方の要素を含めている。

2.2.3　　知識分類の体系イメージ

キーワード抽出（2.2.1 項）とデータベース検索キーに使う分類情報を抽出すること（2.2.2 項）で、障害を未然防止するための設計知識を体系的に捉えることができる。体系を考える上で参照した設計知識の分類モデル（参考）（図 2.7）とその体系イメージを図 2.8 に示す。

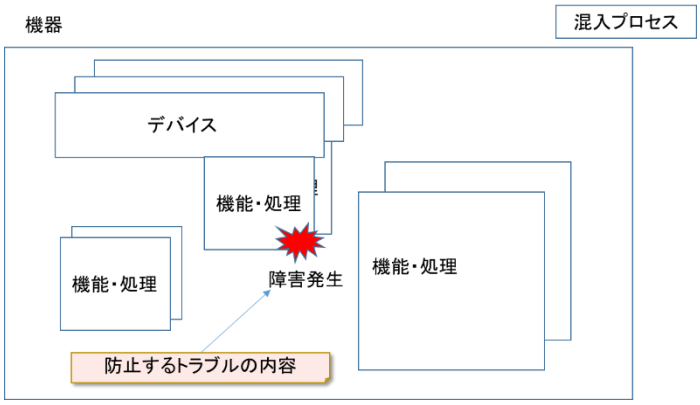


図 2.7　設計知識の分類モデル（参考）

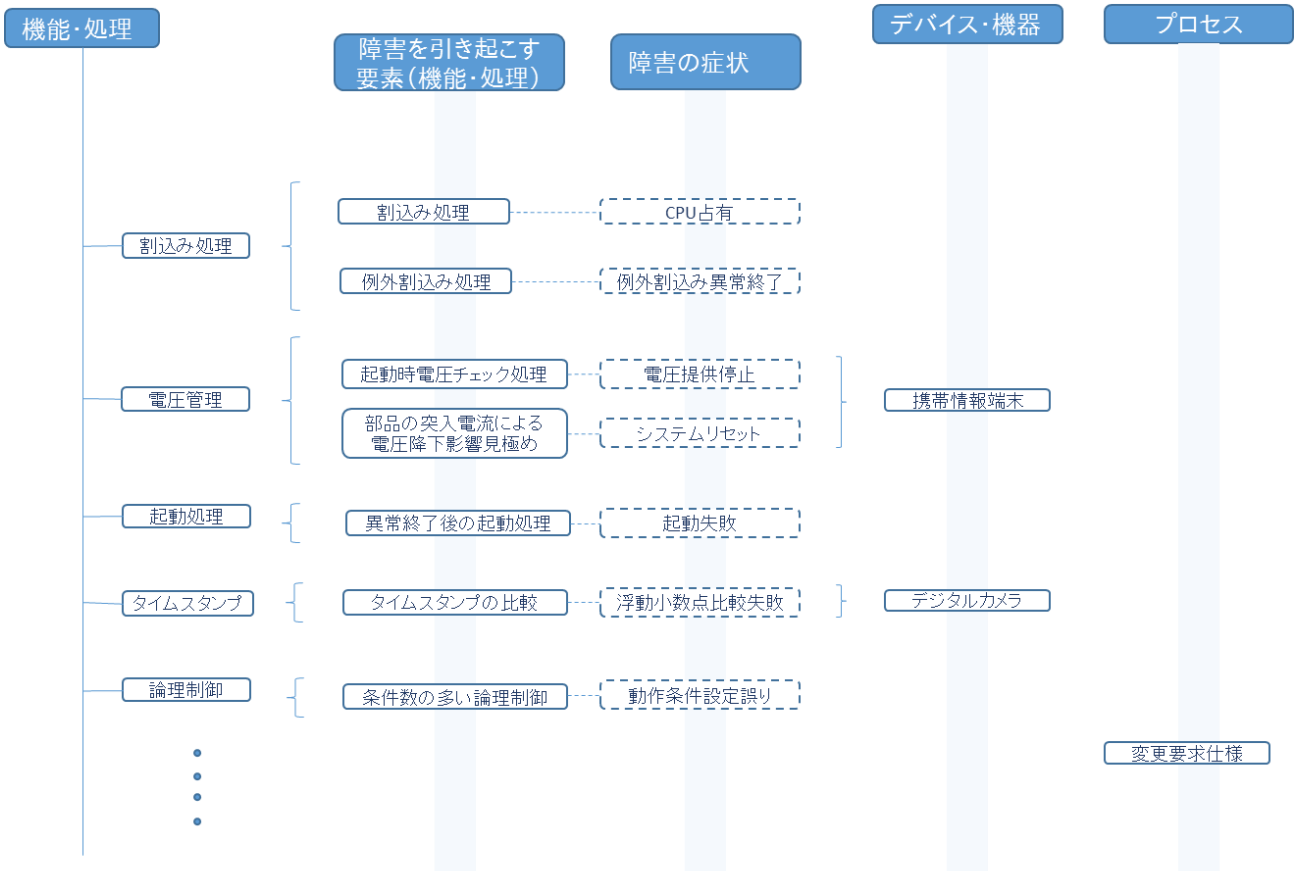


図 2.8　知識分類の体系イメージ

図 2.8 は、何に役立つ知識かを直観的に示す「何が（障害を引き起こす要素（機能・処理））」と「障害の症状」の組み合わせを、「機能・処理」で分類している。
「何が」に相当する「障害を引き起こす要素（機能・処理）」は、障害を起こし易い条件が含まれる場合があるため、分類は、抽象化した「機能・処理」で行う。しかしながら、この 3 つの要素が、抽象的な知識として表現されていても、どこかの製品ドメインを連想する場合もある。例えば、割込み処理に関する「何が（障害を引き起こす要素（機能・処理）」と「障害の症状」の組み合わせは、製品ドメインを連想することは無いが、電圧管理に関する組み合わせでは、電圧管理が必要な機器に適用されることが想像できる。このような場合は、「デバイス・機器」で分類できるようにする。デバイスと機器は、知識として整理する観点で、頭の中に記憶し易い方を選択する。
右端に「プロセス」があるのは、障害事例から抽出した知識の中で、ソフトウェアの機能や処理に対する知識不足ではなく、仕様を文書化しなかったことが原因で発生したものがあったため、例外的に整理の視点に加えている。

2.2.4 分類タグ

前項 2.2.3 で知識の体系イメージを捉えることができたので、知識体系の構成要素として下記の 4 種の分類タグを導出した。

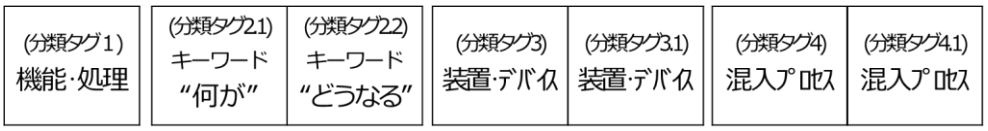


図 2.9 分類タグ

（分類タグ１）機能・処理
（分類タグ２）キーワード“何が”、“どうなる”
（分類タグ３）装置・デバイス
（分類タグ４）混入プロセス

分類タグは、構造化表現された設計知識に付けるもので、知識の体系化、知識の検索のし易さ、別の知識の発想のし易さを考慮する。想定される検索イメージや検索結果の表示イメージは、第 4 章「検索ツールのイメージ」に示す。
タグは検索キーにヒットさせることを考慮すると、キーの粒度は人によって異なるため、階層的に設定できるようにする。（分類タグ 1）機能・処理と（分類タグ 2.1）キーワード“何が”は、階層構造になっている。（分類タグ 3）と（分類タグ 3.1）は、装置・デバイスの階層、（分類タグ 4）と（分類タグ 4.1）は、混入プロセスの階層としている。階層化は、

分類タグのキーワード「何が」「どうなる」の要素とともに、他の知識の連想を促す効果もある。設計知識 DB に蓄積される設計知識が増えて、分類タグの抽象度が適度なレベルになってゆけば、知識体系イメージがより明確になるため、知識の網羅性が高まる。

分類タグは、設計知識の文脈から知識を抽象化する要素を抽出して設定するため、製品ドメインに依存しない共通の知識として利用範囲が拡大する。知識の再利用を図るために、知識そのものの抽象度を上げるのではなく、分類タグを抽象化する。

3 設計知識の整理手法

3.1 概要

第２章、2.1 節及び 2.2 節で説明した「設計知識モデル」と「知識の再利用モデル」に基づいて、設計知識の整理手法を詳述する。設計知識の整理手法は、ソフトウェアで実現する機能を設計する際に、その機能が引き起こし易い障害を対策したり、別の機能やシステム全体で発生する障害をその機能で回避したりするための設計知識を、過去の障害情報記録（過去トラ DB）から抽出し、障害の未然防止に活用できるように整理するものである。

■整理手順

整理手順は、「過去トラ DB」の障害情報記録から設計知識を抽出し、分類タグを設定する流れになる（表 3.1）。

表 3.1 設計知識の整理手順

整理手順	入力	出力	参照先
1　設計知識の抽出	障害情報記録 （過去トラ DB）	設計知識	3.2 節
2　分類タグの設定	設計知識	分類タグ	3.3 節

■入出力情報と整理手順イメージ

表 3.1 の整理手順を図示すると図 3.1 のイメージになる。「過去トラ DB」より抽出した設計知識を構造化、一般化する。さらにキーワードや分類情報を含む分類タグを設定し再利用性を高める。

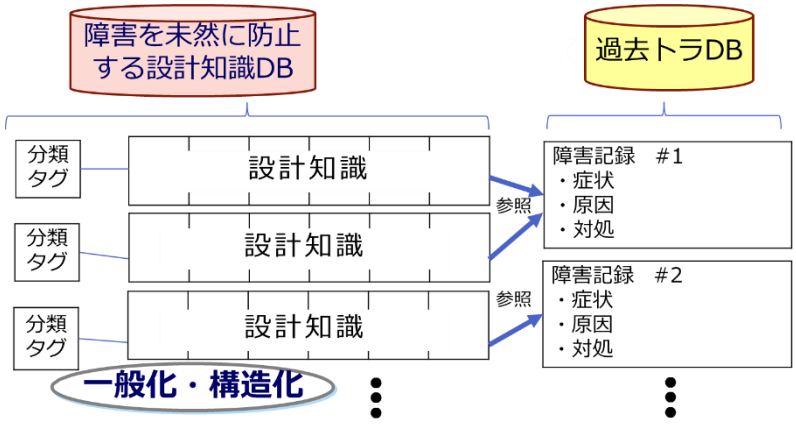


図 3.1 過去の障害情報記録（障害事例）と設計知識の関係

- 設計知識は、整理して設計知識 DB に格納し、検索ツール等で取り出せるようにする。
- 「過去トラ DB」とリンクさせ、必要に応じて元の障害情報記録を参照する。

障害情報記録は、障害報告の副産物として、「過去トラ DB」に次第に溜まってゆくが、障害を未然に防止するための設計知識は、障害情報記録から抽出する作業を経なければ溜まってゆかない。これには、かなりの労力が必要になる。そのため現実的には障害情報記録から、再発しがちな事例や、リスク管理の観点で重要な事例など、優先順位の高い事例を選択し、そこから設計知識を抽出して整理する考え方を取らざるを得ない。

手順の詳細は 3.2 節以降で説明する。

3.2 設計知識の抽出

設計知識の抽出は、(1) 知識要素の抽出、(2) 知識要素の一般化の 2 つのタスクで構成する。

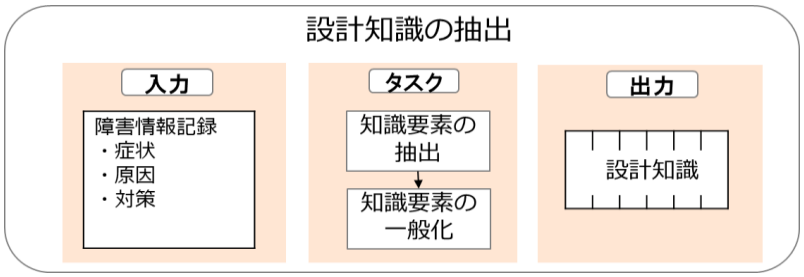


図 3.2　設計知識の抽出

設計知識は、「過去トラ DB」の障害情報記録（図 3.1 参照）から、知識要素を抽出し、前述 2.1.3 項「設計知識の構造」の構造に整理する。

3.2.1　知識要素の抽出

障害情報記録から障害の発生を未然に防止できる設計知識の知識要素を抽出し、構造的に整理する。

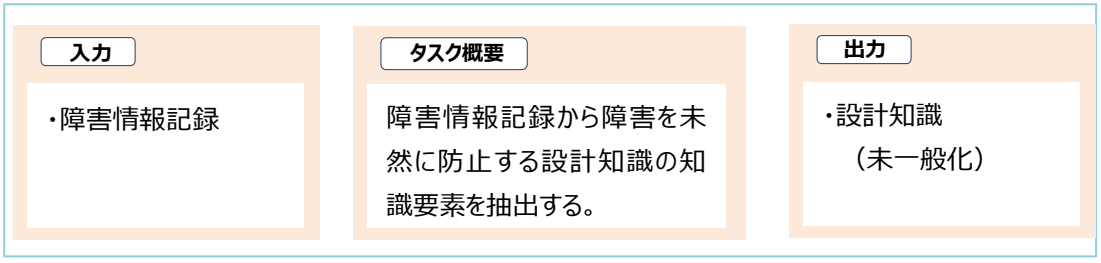


図 3.3　設計要素の抽出

■入力

- 障害情報記録

  障害情報記録には、一般に「障害の症状（状況）」、「直接原因」「対処」が含まれ

ている（図 3.4）。

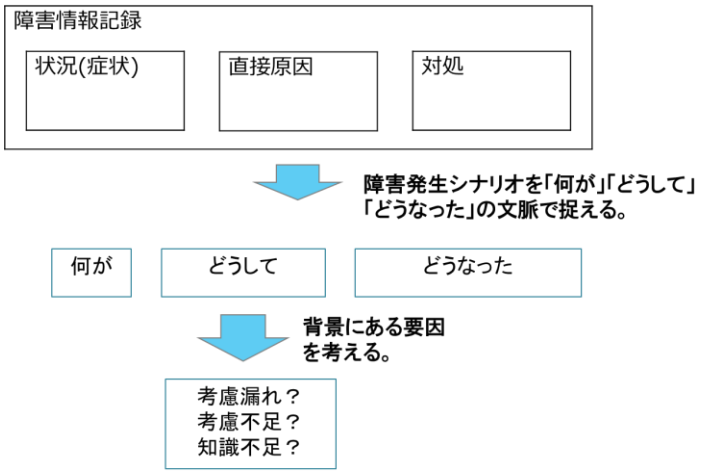


図 3.4　障害情報記録

障害情報記録に記載される障害事例の中には、複数の原因が複合的に絡み合って障害に至る場合もある。その場合は、複数の設計知識を抽出することができる。

■タスク

（１）考慮漏れ要因の調査

障害情報記録の状況（症状）、直接原因、対処から障害発生シナリオ「何が」「どうして」「どうなった」の文脈を捉える。捉えた文脈の「どうして」の背景要因に設計時の考慮漏れ、考慮不足、知識不足は無かったかどうかを、次のガイドに従って調べる。

> 【ガイド１】
> 経験を積んだベテラン技術者であれば、その障害を未然に防止できたかどうか考えてみる。ベテラン技術者は仕様書等に明示されていなくても、経験によって必要な対策を施すことができる。

考慮漏れ要因が想定できない場合は、その障害情報記録を設計知識抽出の対象から外す。

（２）障害発生シナリオの確認

考慮漏れ、考慮不足、知識不足等の背景要因が考えられるなら、次のガイドに従って、障害発生シナリオを作ってみる。

> 【ガイド２】
> ○○○の[機能／処理／作業／データ]において、
> ▽▽▽[の考慮が漏れていると／のことを知らないと]、
> △△△[の場合に／に依存して／を契機に]、◇◇◇◇の障害が発生する。
> ※「括弧」内は、文脈に合うものを選ぶ

このガイドに従った障害発生シナリオが作成できれば、設計知識が抽出できる。

（３）知識要素の抽出

障害情報記録から障害未然防止のための設計知識を抽出する。抽出の観点は、設計時にどんな知識があればこの障害の原因に対して対策を講じることが出来たか、その知識が無かったために対策を講じなかったと考える。そのような知識を想像しながら、次の(1)～(6)の知識要素を見つける。

表 3.2　知識要素の抽出

抽出する知識要素	抽出方法	抽出のポイント
(1)障害を引き起こす機能・処理（技術要素）	障害情報記録の「症状」、「直接原因」、「対処」から直接原因に対処を施した個所を見つけ、そこから「障害を引き起こす機能・処理（技術要素）」を抽出する。	● 対策を施す対象が機能・処理ではなく、データ（領域、形式、サイズ等）の場合もある。また、プロセス上の問題の場合は、対策の対象が規定文書になることもある。
(2)設計時に考慮が漏れ易い設計視点・観点	障害情報記録の「直接原因」、「対処」から(3)の「発生契機」が引き起こす可能性のある事象や状態に対して、考慮が必要な設計視点・観点を抽出する。	● 考慮漏れの観点・視点が設計に行き着かない場合は、例外的に、設計時に限定せず、未然防止の観点で知っておくべき視点・観点を抽出する。例）要求定義時に漏れていた検討の観点・視点
(3)発生契機	障害情報記録の「直接原因」から障害を引き起こすトリガーとなった「発生契機」を抽出する。	● 「発生契機」が無ければ障害は発生しなかったと見なせる事象や状態を指す。
(4)発生し得る障害内容	障害情報記録の「症状」から、(1)「障害を引き起こす機能・処理（技術要素）」、(2)「設計時に考慮が漏れ易い設計視点・観点」、(3)「発生契機」によって、引き起こす障害の内容を抽出する。	● 障害情報記録の「症状」には、外から見える症状のみが記載されている場合は、障害情報記録の他の情報から抽出する。
(5)発生メカニズム	障害情報記録から抽出した(1)～(4)の要素を文章に組み立てる。	
(6)対策	障害情報記録の「対処」の内容を元に、(5)「発生メカニズム」に記された障害の直接原因への対策を抽出する。	

（４）知識文脈の確認

上記(1)～(4)と(6)の要素でガイド 4 に示す知識文脈が組み立てられるか確認する。綺麗な文章の形に整っていなくても、意味が通る文脈が頭の中で整理できれば良い。

【ガイド４】
「(1)の機能や処理を考えるときに、(2)の考慮が漏れていると、(3)が起こった契機で(4)の障害が発生する。その障害の発生を防ぐためには(6)の処理を作り込んでおく。」

■出力

- 障害を未然に防止する視点で構造化した設計知識

(1) 障害を引き起こす機能・処理	(2) 考慮漏れし易い設計視点・観点	(3) 発生契機	(4) 発生し得る障害内容	(5) 発生メカニズム	(6) 対策

「情報処理システム高信頼化教訓集（組込みシステム編）2015 年度版」（参考文献[2]）の障害事例（教訓 8、現象 2）を例に知識要素の抽出例を示す。

(1)【障害を引き起こす機能・処理】(例)

例外設計

(2)【考慮漏れし易い設計視点・観点】(例)

機能実行中の例外発生時の考慮漏れ

(3)【発生契機】(例)

機能実行中の例外発生

(4)【発生し得る障害内容】(例)

外部入出力を正しく参照 /操作できず機能が正常に動作せずシステム障害となる。外部入出力の種別及び機能により現象は特定できない。

(5)【発生メカニズム】（例）

機能実行中に例外が発生すると、入力処理の実施にあたって不定な入力値を使用することになり、システム障害が発生する。

(6)【対策】（例）

割り込みの起動時に正常割り込みであるかを判断してノイズによる割り込みであれば何もせずに割り込みプログラムを終了。

【知識の文脈】「何が、何々したら、どうなる」
（(1)に(2)の考慮がもれていると(3)が起こったときに(4)が発生する）

3.2.2　　知識要素の一般化

特定の障害情報記録から抽出した設計知識をソフトウェア共通の知識として伝えるために、知識の各要素は特定の製品ドメインを連想させる用語を避け出来るだけソフトウェア共通の一般化した表現を見つける。

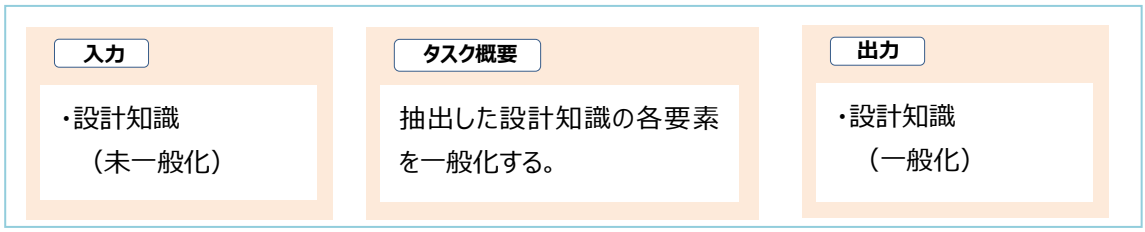


図 3.5　知識要素の一般化

■入力

- 障害を未然に防止する視点で構造化した設計知識

(1) 障害を引き起こす機能・処理	(2) 考慮漏れし易い設計視点・観点	(3) 発生契機	(4) 発生し得る障害内容	(5) 発生メカニズム	(6) 対策

■タスク

- 各要素の一般化。

（1）～（6）の各要素に製品ドメインに依存する機能や処理を表す用語や表現等がある場合は、ソフトウェア共通に通じる用語や表現に修正する。

※一般化表現に拘りすぎると、設計知識が頭に刺さらなくなる場合があるので注意する。この設計知識が幾分、特定の装置や機器を連想させる場合や、反対に過度に一般化してしまった場合は、3.3 節「分類タグの設定」、3.3.2 項「知識の分類」で、装置・デバイスに関する分類タグを考える際に調整できる。

■出力

- 障害を未然に防止する視点で構造化した設計知識（一般化表現）

(1) 障害を引き起こす機能・処理	(2) 考慮漏れし易い設計視点・観点	(3) 発生契機	(4) 発生し得る障害内容	(5) 発生メカニズム	(6) 対策

前項 3.2.1「知識要素の抽出」の出力を一般化表現した例を示す。

(1)【障害を引き起こす機能・処理】(例)

割り込み処理

(2)【考慮漏れし易い設計視点・観点】(例)

想定していない例外割り込み発生時の考慮漏れ

(3)【発生契機】(例)

ノイズ

(4)【発生し得る障害内容】(例)

機能実行中に例外割り込みが発生することでシステム異常・停止等の障害を引き起こす

(5)【発生メカニズム】（例）

割り込み要求端子にノイズが入ると、不定な値を使用して割り込み処理が起動される場合ある。その際、割り込み処理プログラムに、不正な割り込みを想定したコーディングがなされていなければ、システム障害が発生する。

(6)【対策】（例）

割り込み処理起動時に正常割り込みであるかを判断し、不正な割り込みであれば何もせずに割り込みプログラムを終了させる。

3.3 分類タグ設定

障害情報記録から抽出した設計知識が、知識 DB に蓄積されることを想定する。DB の検索や、DB から取出した設計知識の見せ方を工夫することで知識の再利用が促される。ここでは、2.2 節「知識の再利用モデル」の考えに基づいて、分類タグを設定する。分類タグの設定は、(1) キーワード抽出、(2) 知識の分類の 2 つのタスクで構成する。

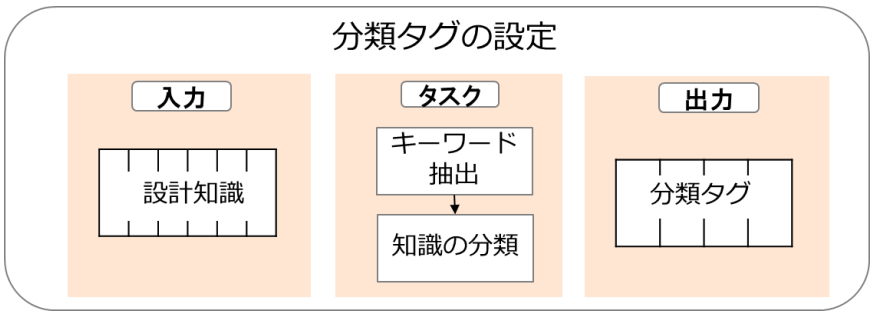


図 3.6　分類タグの設定

3.3.1　　キーワード抽出

一般化表現された設計知識を一目見てもしくは一言聞いて、その知識が何に役立つのか直観的に分かるように、キーワード“何が”“どうなる”に相当する要素を抽出する。

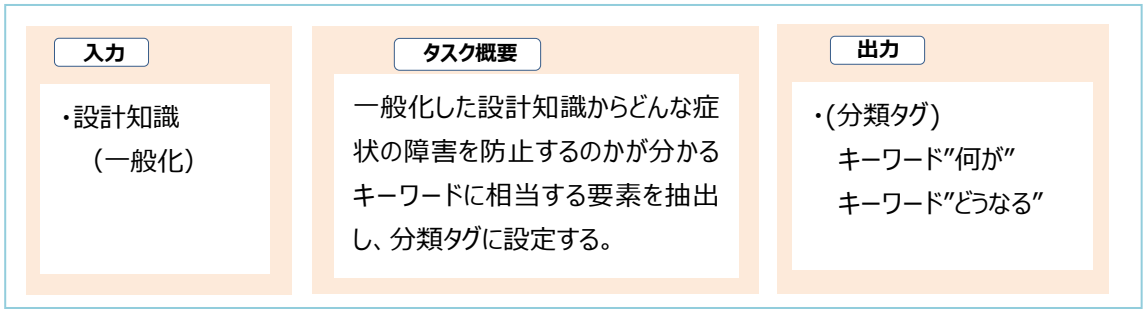


図 3.7　キーワード抽出

■入力

- 障害を未然に防止する視点で構造化した設計知識（一般化表現）

(1) 障害を引き起こす機能・処理	(2) 考慮漏れし易い設計視点・観点	(3) 発生契機	(4) 発生し得る障害内容	(5) 発生メカニズム	(6) 対策

■タスク

設計知識の要素からキーワード“何が”“どうなる”に相当する要素を抽出し、分類タグに設定する。

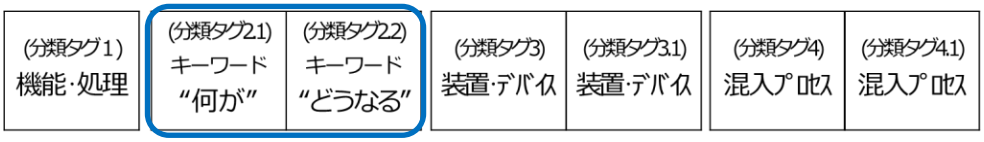
表 3.4　キーワード抽出

分類タグ	抽出方法	抽出のポイント
キーワード “何が”	設計知識の要素(1)「障害を引き起こす機能・処理」から「何が」に相当する要素を抽出する。	● キーワードは、「何が」「どうなる」のセットで考え、抽象度を上げて引き起こす障害が限定されず発想が広まるように言葉を選ぶ。
キーワード “どうなる”	設計知識の要素(4)「発生し得る障害内容」から「どうなる」に相当する要素を抽出する。	● どんな障害を防止するのか、防止する障害症状を設定する。 ● 最終状態を表す症状は、原因をイメージしづらくなる。例）「システム停止」、「運用中断」 ● 途中の状態を表す表現は使わない。 例）「組合せ爆発」

※キーワード抽出は、「障害未然防止のための教訓化ガイドブック（組込みシステム編）」（参考文献[3]）に掲載された「直接原因観点マップ」を参考にする。

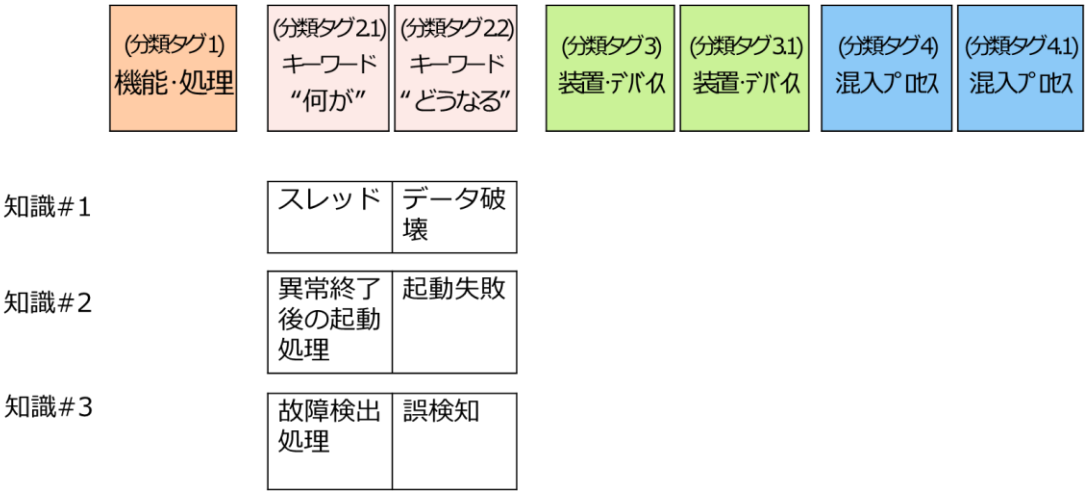
■出力

- 分類タグ



「情報処理システム高信頼化教訓集（組込みシステム編）2015 年度版」（参考文献[2]）の障害事例から抽出した分類タグ 2.1、2.2 の例を示す。詳細は第 4 章に記載している。

【例】



3.3.2 知識の分類

一般化表現された設計知識と“何が”、“どうなる”で表されるキーワードの両方を包含するイメージを意識して、設計知識を分類する。

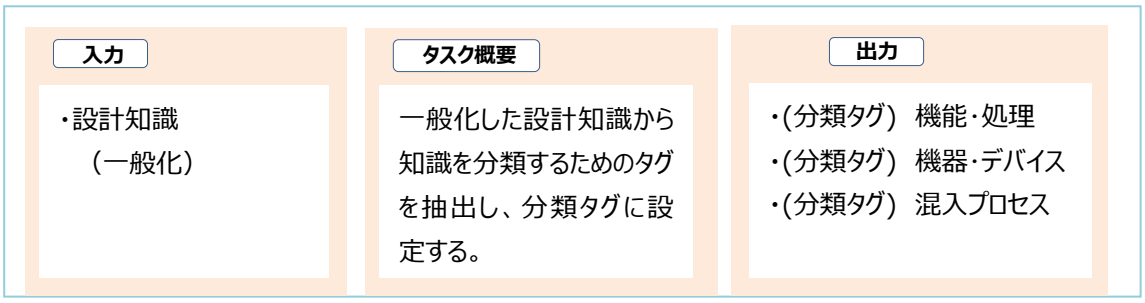


図 3.8　知識の分類

■入力

① 障害を未然に防止する視点で構造化した設計知識（一般化表現）

(1) 障害を引き起こす機能・処理	(2) 考慮漏れし易い設計視点・観点	(3) 発生契機	(4) 発生し得る障害内容	(5) 発生メカニズム	(6) 対策

② 分類タグ （キーワード）“何が”

(分類タグ2.1) キーワード “何が”

■タスク

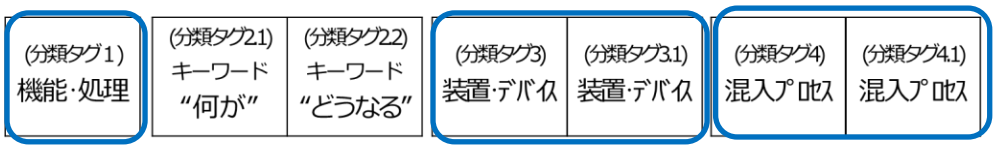
表 3.5 知識の分類

分類タグ	設定の仕方	設定のポイント
（分類タグ１） 機能・処理	設計知識の要素(1)「障害を引き起こす機能・処理」と分類タグ （キーワード）“何が”から、できるだけソフトウェア共通の機能・処理のグループを設定する。	● 設計知識を検索するための「検索キー」と考え、ソフトウェア共通の機能・処理をタグにする。 ● 検索した設計知識を表示する際の「見出し」として使うことも考える。 ● このタグは、原則、設定必須とする。
（分類タグ３） 装置・デバイス	設計知識の要素(2)「考慮漏れし易い設計視点・観点」に、特定の装置やデバイスに依存すると考えられる場合にはこのタグを設定する。	● 設計知識の検索を、過去に発生した事例を対象に、装置やデバイス横串で行いたい場合にはこのタグを設定する。 ● 設計知識が装置やシステムに依存しない、共通な知識として認識してほしい場合は、敢えてこのタグを設定しない。 ● 装置名称の設定については、設計知識DBの公開範囲を考慮して装置や製品が特定されるリスクに注意する。
（分類タグ４） 混入プロセス	設計知識の要素(2)「考慮漏れし易い設計視点・観点」に混入プロセスの視点・観点があれば、設定する。	● 不具合を混入させた要因が設計知識の欠如やうっかりミスではなく、仕様書を書かない等、プロセス遵守違反やプロセス不備による場合にこのタグを設定する。

※分類タグ１と分類タグ２の抽出は、「障害未然防止のための教訓化ガイドブック（組込みシステム編）」（参考文献[3]）に掲載された「直接原因観点マップ」を参考にする。

■出力

- 分類タグ



「情報処理システム高信頼化教訓集（組込みシステム編）2015 年度版」（参考文献[2]）の障害事例から抽出した分類タグの例を示す。詳細は第 4 章に記載している。

【例】

	(分類タグ1) 機能･処理	(分類タグ2.1) キーワード “何が”	(分類タグ2.2) キーワード “どうなる”	(分類タグ3) 装置･ﾃﾞﾊﾞｲｽ	(分類タグ3.1) 装置･ﾃﾞﾊﾞｲｽ	(分類タグ4) 混入ﾌﾟﾛｾｽ	(分類タグ4.1) 混入ﾌﾟﾛｾｽ
知識#1	並列処理	スレッド	データ破壊				
知識#2	起動処理	異常終了後の起動処理	起動失敗	業務システム	店舗用窓口システム		
知識#3	監視処理	故障検出処理	誤検知	センサー	故障検知センサー		
知識#4						要求定義	変更要求仕様

4 検索ツールのイメージ

第３章の整理手法に基づいて整理した設計知識が、図4.1のようにDB化されている場合、どの様に検索され、どの様に検索結果が取り出されるか、検索ツールのイメージを示す。この検索ツールの動作を机上トレースする、あるいは実際に設計知識DBと検索ツールを作るなどして、2.1節の「設計知識モデル」と2.2節の「知識の再利用モデル」が設計知識を伝える上で効果的かどうか、その評価がフィードバックされることを期待する。

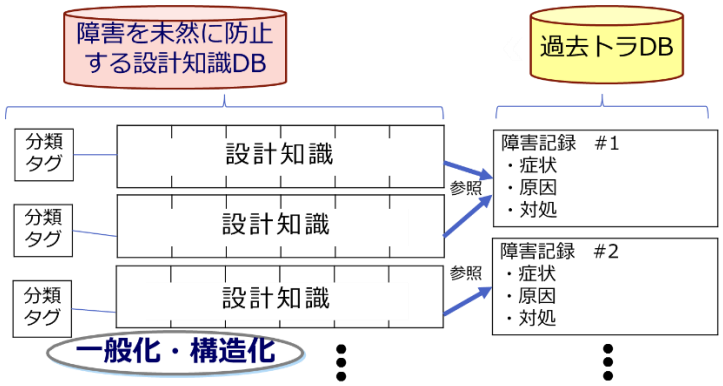
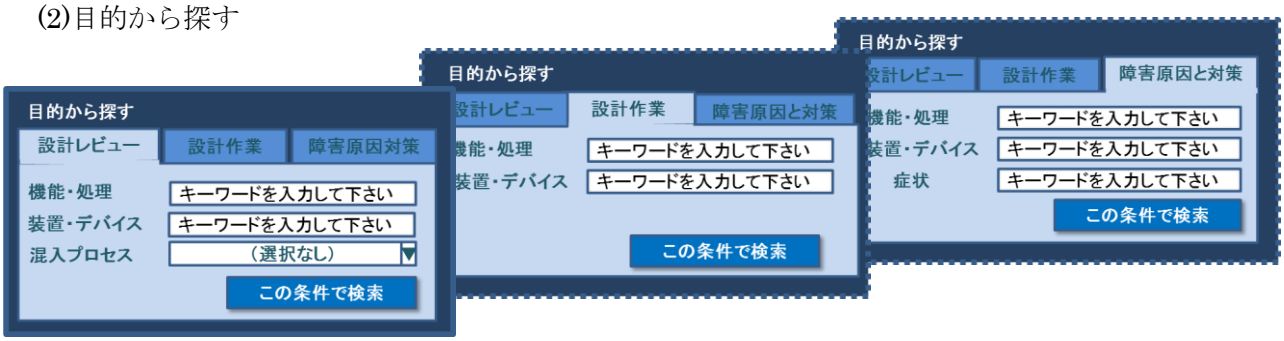


図 4.1　設計知識 DB

4.1 検索キー

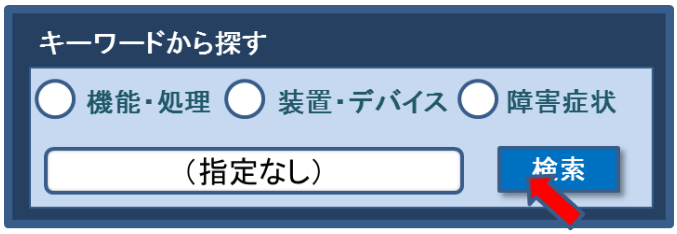
(1)キーワードから探す

(2)目的から探す



4.2 検索結果

(1)キーワードから探す



並べ替え： 登録日付順　知識レベル順　〇〇〇順

101件中 1～25件を表示 [1 | 2 | 3 | 4 | 5]次の25件▼

知識ID A-200213
●機能／処理 割り込み・例外割り込み ●装置／デバイス － ●障害症状 システム異常、停止 ●混入プロセス －

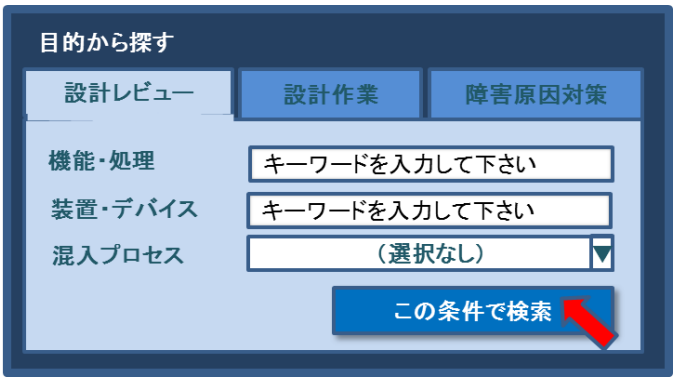
-
【発生契機】　ノイズ
【技術要素】　割り込み処理
【考慮漏れし易い設計視点・観点】　想定していない例外割り込みが発生したときの考慮漏れ
【発生し得る障害内容】　機能実行中に例外割り込みが発生することでシステム異常・停止等の障害を引き起こす
【発生メカニズム】　割り込み要求端子にノイズが入ると、不定な値を使用して割り込み処理が起動される場合がある。・・
【対策】 割り込み処理起動時に正常割り込みであるかを判断し、不正な割り込みであれば何もせずに割り込みプログラムを・・
（登録日）2016年9月1日

■トラブル事例：教訓集No8

知識ID A-500225
●機能／処理 デバイス管理・デバイス管理テーブル ●装置／デバイス － ●障害症状 デバイス接続エラー ●混入プロセス －

-
【発生契機】　最大接続数
【技術要素】　複数の接続先に接続可能なシステム／装置において、接続先管理テーブルを動的に作成する処理
【考慮漏れし易い設計視点・観点】　接続オプションにより、接続先管理テーブルに登録するデータサイズが変動する場合・・・

(2)目的から探す



並べ替え： 登録日付順　知識レベル順　〇〇〇順

101件中　1～25件を表示　[1 | 2 | 3 | 4 | 5]次の25件▼

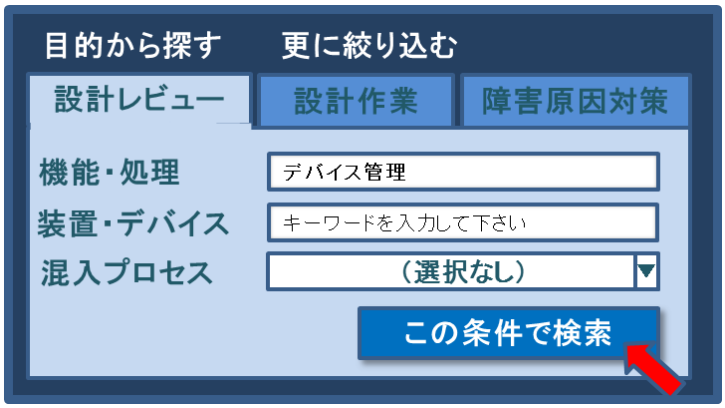
知識ID A-200213
●機能／処理 割り込み・例外割り込み ●装置／デバイス － ●障害症状 システム異常、停止 ●混入プロセス －

-
【発生契機】　ノイズ
【技術要素】　割り込み処理
【考慮漏れし易い設計視点・観点】　想定していない例外割り込みが発生したときの考慮漏れ
【発生し得る障害内容】　機能実行中に例外割り込みが発生することでシステム異常・停止等の障害を引き起こす
【発生メカニズム】　割り込み要求端子にノイズが入ると、不定な値を使用して割り込み処理が起動される場合がある。・・
【対策】 割り込み処理起動時に正常割り込みであるかを判断し、不正な割り込みであれば何もせずに割り込みプログラムを・・
（登録日）2016年9月1日

■トラブル事例：教訓集No8

知識ID A-500225
●機能／処理 デバイス管理・デバイス管理テーブル ●装置／デバイス － ●障害症状 デバイス接続エラー ●混入プロセス －

-
【発生契機】　最大接続数
【技術要素】　複数の接続先に接続可能なシステム／装置において、接続先管理テーブルを動的に作成する処理
【考慮漏れし易い設計視点・観点】　接続オプションにより、接続先管理テーブルに登録するデータサイズが変動する場合・・・



5 設計知識の整理サンプル

第 3 章の整理方法に従って、「情報処理システム高信頼化教訓集（組込みシステム編）2015 年度版」（参考文献[2]）の障害事例（教訓）から設計知識（表 5.2）を抽出し整理した。その対応表を表 5.1 に示す。

表 5.1　教訓番号と設計知識 Index 対応表

教訓番号	教訓タイトル	設計知識Index
1	複雑な条件式のロジック変更を行う場合は、デシジョンテーブル等による検証が有効である	27
2	条件が整理されていない状態で、トータルの条件数が100を超えるような機能、または10個以上の条件を有する機能を修正する場合、関連する条件を全て洗い出して整理し不整合がないことを確認する	28
3	複数機能モジュールを統合する場合、統合前の条件数の総和と統合後の条件数を比較し差がある場合は、条件の抜けがないか確認する。	29
4	変数値域が広く、組合せバリエーションが非常に多くなる場合には、値域を適切な大きさに分割した上で境界値テストを実施する	30
5	内蔵電池を使用する場合には、深放電時の起動シーケンスを考慮すること	19
6	フラッシュメモリを使用する場合には、書き込み寿命回数を考慮すること	18
7	消費電力の多い機能を追加する場合には、一時的な電圧降下による影響（リセット、フリーズ等）や電源の種類、電池の場合は残量を考慮すること	20
8	想定可能な例外を形式的に漏れなく分析する	13, 14
9	システムを二重化する場合は、同期すべきデータ領域を適切に設定する	25
10	制御対象のハードウェアが同一でも、運用条件が変わるときは、ハードウェア仕様を再確認する	15, 16
11	プロセス間、スレッド間でデータを共有（引き渡し）する場合は、排他・同期処理が正しく行われているか、あるいはデッドロックが発生していないかどうか注意する	31
12	歩留りのある製品の良品／不良品を検査する装置では、全てが良品あるいは、不良品との検査結果は異常と判断すべきである	7,8
13	既存ソフトウェアの性能改善を実施する際には、アイドリングタイムの発生、処理の同期ずれの発生等と影響を確認する	21
14	・大量のデータを通信経由で扱う場合、一連の処理の流れの中にボトルネックを作りこまないように注意する	24
15	納入したあと、お客様が運用するような業務システムでは、業務シーケンス中のあらゆる異常操作（リセット、電源断、放置も含め）、への対応を考える	1, 3
16	障害解析時の保守メンテ用ログ処理であっても、仕様書を作成し、影響評価を実施すること	9
17	判断処理は、必要条件だけでなく、制限すべき条件も漏れなく抽出する	6
18	ログファイルの断片化に注意する	10
19	人による変更作業ではミスが起きることを前提に、ツール活用などで不具合の作り込みや流出の防止に心がける	12
20	信頼性向上施策を採る場合は、故障発生確率と影響の定量評価を行い、対策は確実に実装する	4
21	高い信頼性対策が求められるシステムでは重大な影響を及ぼす事象の想定と復旧手順を十分に検討する	2
22	処理時間がクリティカルなシステムではツールを活用し、変数やその取りうる状態数とそれぞれの状況における動作処理に最大バラツキを意識し余裕を把握し設計する。	5
23	開発を伴わない保守案件でも，システム構成変更が発生する場合は，手順等作業内容の妥当性を確認できるようなプロセスを経る	17
24	物理量（時間、重量など）を扱う場合は単位、桁数を確認する。	11
25	顧客が要求していることの目的と背景に遡って、その意図を確認することが、要求仕様のあいまいさ排除に役立つ	32
26	遠隔地等物理的に離れた装置をネットワーク接続して稼働させるシステムでは、故障などの状態検知やメンテナンスも容易ではないため、システム的視点での状態把握を行う。	22
27	マルチベンダーシステムでは仕様に外れた想定外事象が発生することを前提とした自己防衛策を採る。	23
28	データベース等COTS製品のバージョン、動作仕様の相違等の情報が関係者にタイムリーに参照できるようにする	26

表 5.2　教訓集から抽出した設計知識（1/6）

	(分類タグ1) 機能・処理	(分類タグ2.1) キーワード “何が”	(分類タグ2.2) キーワード “どうなる”	(分類タグ3) 装置・デバイス	(分類タグ3.1) 装置・デバイス	(分類タグ4) 混入プロセス	(分類タグ4.1) 混入プロセス	① 〔障害を引き起こす機能・処理〕	② 〔考慮漏れし易い設計視点・観点〕
1	起動処理	異常終了後の起動処理	起動失敗	業務システム	店舗用窓口システム			業務システムの起動処理	前回異常終了していた場合のリカバリ処理
2	起動処理	異常終了後の起動処理	起動失敗					(システム異常発生時の)起動・終了シーケンス	直前にシステム異常終了したことを考慮したシステム起動シーケンス設計
3	集計処理	バッチ処理	再起動失敗	業務システム	店舗用窓口システム			バッチ処理	バッチ処理シーケンスが異常終了した後のリカバリ処理
4	監視処理	故障検出能力向上	誤検知	センサー	故障検知センサー	設計	変更設計	故障検知センサー	センサーの感度を必要以上にアップさせると、システムの信頼性を低下させる
5	定周期処理	定周期処理	処理時間の超過					クリティカルな定周期処理	変更設計によって、定周期に起動されるメインルーチンのWCET(最悪実行時間)が増加する可能性があること。また、メインルーチンの処理時間に影響を与える割り込みルーチンのWCET。

※分類タグ１：設定必須

分類タグ２：設定必須

分類タグ３：装置・デバイスの横串情報が不要な場合に空白としている。

分類タグ４：プロセス要因を強調したい場合のみ設定。通常は空白にしている。

③ 〔発生契機〕	④ 〔発生し得る障害内容〕	⑤ 〔発生メカニズム〕	⑥ 〔対策〕	
システムの異常終了	業務システムが正常に立ち上がらない。	前日の業務処理で集計データをサーバに送信中にバッチ処理が強制終了され、データ送信処理が未完了のまま終了した。 このような事態を考慮せず業務システムを設計していたため、業務システムを再起動した際データ再送などのリカバリ処理が行われず、システムが正常に立ち上がらなかった。	起動処理には前回異常終了を想定したリカバリ処理を組み込む。	教訓15
RAID故障	システムが起動したが、システム異常が解消されず、正常動作できない状態になる。	システム終了時の正常・異常情報を、RAIDに格納した、起動・終了シーケンスを組んでいたが、RAID故障で情報が喪失する場面を想定していなかった。	直前のシステム終了状態を不揮発メモリに残すなど、確実に確認できるような起動・終了シーケンスにする。	教訓21
バッチ処理の強制終了	バッチ処理が再起動できない。	前日の業務処理で集計データをサーバに送信中にバッチ処理が強制終了され、データ送信処理が未完了のまま終了した。 このような事態を考慮せず業務システムを設計していたため、業務システムを再起動した際データ再送などのリカバリ処理が行われず、システムが正常に立ち上がらなかった。	起動処理には前回異常終了を想定したリカバリ処理を組み込む。	教訓15
ノイズ	信頼性向上のために故障検知センサー感度を上げてしまったら、無視しても問題のないノイズまで検出してしまい、障害発生とみなしてしまった。	システムの機構部分の改造に伴いノイズの発生頻度が増えたように感じたため、信頼性向上が必要と判断し、故障検知センサーの感度をアップさせた。その際、故障発生確率の算出と故障発生時の影響の定量評価を行わず、設計者個人の判断で改造作業を行っていた。ノイズを検出した場合、故障と判断するかどうかのロジックが組み込まれていたが、不十分であったため、無視しても問題のないノイズを故障と判断してしまい障害とみなされてしまった。	故障検知センサーの感度アップは、故障発生確率と影響の定量評価に基づいて行う。	教訓20
動作シーケンスの組合せバリエーションの増大化、変化	変更設計により、動作シーケンスのWCETが伸びてしまうと、制御信号の発出タイミングやセンサー情報の読み取りタイミングにずれが生じる。その結果システムの不安定動作を引き起こす。	タイマー割り込みにより5ms定周期で起動されるメインルーチンの中で、他の割り込み処理を受け付けながら、5ms以内に処理シーケンスを終了する必要があるが、変更設計により、処理シーケンスのWCET(最悪実行時間)が5msを超過してしまう非常に稀なケースが発生した。	・定周期に起動されるメインルーチンのWCET(最悪実行時間)の見積もりを行い増減を確認する。 ・更に遅延を引き起こす割り込みルーチンの変数組合せバリエーションを考慮したWCETの増減を確認する。 ・メインルーチンと割り込みルーチンに共有変数がある場合は割り込み干渉の影響も確認する(相互排他問題)。	教訓22

表 5.2　教訓集から抽出した設計知識（2/6）

	(分類タグ1) 機能・処理	(分類タグ2.1) キーワード “何が”	(分類タグ2.2) キーワード “どうなる”	(分類タグ3) 装置・デバイス	(分類タグ3.1) 装置・デバイス	(分類タグ4) 混入プロセス	(分類タグ4.1) 混入プロセス	① 〔障害を引き起こす機能・処理〕	② 〔考慮漏れし易い設計視点・観点〕
6	判定処理	入場判定処理	入場可否判定不能	入退出ゲート管理システム				電子通行証の記録と施設状態を照合して入場判定する入退出ゲートの判定処理	複雑化した判定条件の組合せパターンは、抜け漏れが起こり易く、類似トラブル防止のための知識が蓄積されていること
7	判定処理	判定処理	ユーザ判断ミス誘発	検査システム	歩留まりのある量産製品の検査装置			検査システムにおける異常状態の仕様検討	ユーザ視点による異常状態の洗い出し
8	判定処理	判定処理	ユーザ判断ミス誘発	検査システム	歩留まりのある量産製品の検査装置			検査結果表示のユーザインタフェース	良品率100%の検査結果の場合に検査結果判定者はどのように感じるかのユーザエクスペリエンス
9	保守機能	ログ収集	データ消失					保守用処理の実装	保守用処理など顧客要求に直接かかわらない機能が異常になった場合の影響の確認
10	保守機能	ログ収集	I/O性能低下	業務システム	サーバーシステム			ログファイル	DISK上のファイルの断片化を考慮しないとI/O負荷が高騰する

③ 〔発生契機〕	④ 〔発生し得る障害内容〕	⑤ 〔発生メカニズム〕	⑥ 〔対策〕	
セキュリティ強化のための入退出詳細記録と判定条件の組合せパターンの増大化	複数施設に入場するために施設毎の入退出ゲートを通行する際、ある施設の入場可否判定が不能になり異常終了した。その結果システム全体が異常となり使用できなくなった。	入退室ゲートで利用者の通行証の入場可否を判定する処理で本来考慮すべき入場判定条件の一部が抜けていた。 これにより入場不可とすべき通行証を入場可として処理を進めたため、正常なデータ処理ができず入場判定処理が異常終了、それを契機に入退出ゲート管理システムが停止した。	・判定条件に抜けがないように、不変条件を論理式で記述するなど形式手法の適用を検討する。 ・過去の知識を蓄積・活用するために、判定条件を文書化するとともに蓄積された知識を活用・確認する場を設ける。	教訓17
検査システムにおける異常状態の判定	検査システムが異常のまま検査を継続する。 【本ケースでは、全て良品として検査を継続し、後に全量再検査となった】	全てが不良品の場合にはシステム異常としていたが、全てが良品の場合にもシステム異常としなければならないユーザ視点が抜けた案件。 本例の半導体検査では、通常一定の割合で良品／不良品と判定されるため、全て良品あるいは全て不良品となる場合は、検査システムが異常であることが多く、通常システムの確認が必要になる。	全不良、全良品発生時の検査システムの振る舞いを仕様に明記する。	教訓12
メンテナンスモードの設定	検査結果をマスクして、意図的に不良を検出させないメンテナンスモードの設定に気づかず、良品率100%の検査結果を正常と判定したまま検査を継続してしまい、後に全量再検査となる。	全てが不良品の場合にはシステム異常としていたが、全てが良品の場合にもシステム異常としなければならないユーザ視点が抜けた案件。 本例の半導体検査では、通常一定の割合で良品／不良品と判定されるため、全て良品あるいは全て不良品となる場合は、検査システムが異常であることが多く、通常システムの確認が必要になる。	全不良、全良品発生時の検査システムの振る舞いを仕様に明記する。	教訓12
異常時のログデータ出力	保守用処理の影響で業務処理が異常終了し、業務が停止する。	ある製品製造工程管理システムでは工程ごとの作業情報をログデータとして集計し次の工程以降で利用しているが、正常時と異常時のログデータを区別なく同じファイルに書き込んでいたため、異常発生時に正常時のデータが失われてしまった。 異常時のログデータ出力処理は、仕様書に明記されておらず影響評価も実施されていなかった。	保守用処理も仕様書を作成し、影響評価を実施する。	教訓16
保守ログ採取	ログファイル採取時によりシステムがスローダウンする。	日毎に作成し一定期間保持する可変長のログファイルが作成・削除を繰り返すことによりDISK上で徐々に断片化し、ログ採取時にDISK負荷高騰を発生させてオンラインプロセスが待たされた。	一日分のログファイルと保存用のログファイルパーテーションを分離し断片化を発生させにくいようにした。	教訓18

表 5.2　教訓集から抽出した設計知識（3/6）

	(分類タグ1) 機能・処理	(分類タグ2.1) キーワード “何が”	(分類タグ2.2) キーワード “どうなる”	(分類タグ3) 装置・デバイス	(分類タグ3.1) 装置・デバイス	(分類タグ4) 混入プロセス	(分類タグ4.1) 混入プロセス	① 〔障害を引き起こす機能・処理〕	② 〔考慮漏れし易い設計視点・観点〕
11	タイムスタンプ	タイムスタンプの比較	浮動少数点の比較失敗	デジタルカメラ				浮動小数点型(double)の変数で保存されている時刻情報	浮動少数点型の変数は少数点誤差が含まれるため、比較する場合には許容誤差の考慮が必要なこと
12	データ読み取り処理	データ読み取り処理	データ読み取り異常			設計	データ入力（変換）作業	データ設計・実装	データ入力（変換）作業やデータ確認を人が行う場合には、目視で差異を確認しにくいデータがあること
13	割り込み処理	例外割り込み処理	割り込み異常終了					割り込み処理	想定していない例外割り込み発生時の考慮もれ
14	割り込み処理	割り込み処理	CPU占有					割り込み処理	割り込みがバースト的に発生すると割 り込み処理が連続起動されてCPU処理が占有されること
15	デバイス管理	デバイス管理テーブル	不正メモリアクセス	通信デバイス	無線LANチップ			入出力装置を接続する場合の、入出力制御デバイスよる管理テーブルの作成メカニズム	チップ内部のメモリに管理テーブルを作成する場合の制約事項
16	デバイス管理	デバイス管理テーブル	不正メモリアクセス					複数の接続先に接続可能なシステム／装置において、接続先管理テーブルを動的に作成する処理	接続オプションにより、接続先管理テーブルに登録するデータサイズが変動する場合があること

③ 〔発生契機〕	④ 〔発生し得る障害内容〕	⑤ 〔発生メカニズム〕	⑥ 〔対策〕	
時刻情報の比較	浮動小数点型（double)に保存された時刻情報を取り出して有効桁数で丸めたものをlong型に保存し、これを元の少数点誤差を含んだ時刻情報を比較すると一致しないことが起こる。	撮影時刻を浮動小数点型（double)の変数にて、sec(秒)単位、小数点以下有効3桁で保存している画像ファイルがある。（例8.138999999[sec])。 画像ファイルの名称には、少数点誤差を丸めたmsec(秒)単位の時刻情報が設定されている（例_8139_)。画像ファイルに保存されている時刻情報をdouble型のまま単純に1000倍して取り出し、msec(秒)単位に変換した時刻情報（例8138）とファイル名に使用している時刻情報を比較したら不一致になった。	画像ファイルに保存されている時刻情報(sec単位（浮動小数点数))を1000倍した後、小数点以下1桁目を四捨五入し、msec単位（long型）に変換してから、long型同士でファイル名称の時刻情報と比較するように変更する。	教訓24
データ誤入力	データ処理が異常となり使用できない。	データベース上の管理情報を担当者が入力する際に、本来全角の「・」（なかてん）を、半角として入力した。この全角と半角の違いを担当者が見抜けず、管理情報として登録してしまった。 プログラムとして扱うデータは全角を前提としていたため、データ 読み取り（変換）処理でエラーとなり、業務に影響を及ぼした。	・属人的作業に依存せず、自動入力やチェックツールを最大限に活用する ・データを手入力しなくてはならない場合は、入力規則を定める。 ・手入力を要するシステムでは範囲外テストを実施する。	教訓19
ノイズ	機能実行中に例外割り込みが発生することでシステム異常・停止等の障害を引き起こす。	割 り込み要求端子にノイズが入力すると、不定な値を使用して割り込み処理が起動される場合ある。その際、割り込み処理プログラムに、不正な割り込みを想定したコーディングがなされていなければ、シ ステム障害が発生する。	割り込み処理起動時に正常割 り込みであるかを判断し、不正な割り込みであれば何も せずに割 り込みプログラムを終了させる。	教訓8
割り込み連続起動	CPUが占有されれるとリアルタイム制約のある他の処理に影響を及ぼす。	割り込みがバースト的に発生すると割 り込み処理が連続起動されてCPU処理が占有されるため、リアルタイム制約のある他の処理に影響を及ぼす。	時間制約を満足することができる場合は、割り込みをマスクしてポーリング方式に切り替える。	教訓8
最大数接続	入出力制御デバイスがチップ内部メモリに管理テーブルを作成できず、入出力制御デバイスのI/Oエラーとなり、装置がリブート。	内部メモリに管理テーブルを作成する方式としたときに、管理テーブルサイズが混在する最大数の入出力装置の接続において、管理テーブルサイズの大きい入出力装置を先に複数接続する。 入出力制御デバイスは、接続順に内部メモリへ管理テーブルを作成していくが、途中で内部メモリがいっぱいになり、入出力制御デバイスが管理テーブルを作成できなくなる。	入出力制御デバイスの外部メモリに管理テーブルを作成する方式とする。	教訓10
最大数接続	接続先管理テーブルの最大サイズ容量を超えてしまい、仕様上の最大数が接続できずに、接続エラーとなった。	内部メモリに管理テーブルを作成する方式としたときに、管理テーブルサイズが混在する最大数の入出力装置の接続において、管理テーブルサイズの大きい入出力装置を先に複数接続する。 入出力制御デバイスは、接続順に内部メモリへ管理テーブルを作成していくが、途中で内部メモリがいっぱいになり、入出力制御デバイスが管理テーブルを作成できなくなる。	入出力制御デバイスの外部メモリに管理テーブルを作成する方式とする。	教訓10

表 5.2　教訓集から抽出した設計知識（4/6）

	(分類タグ1) 機能・処理	(分類タグ2.1) キーワード “何が”	(分類タグ2.2) キーワード “どうなる”	(分類タグ3) 装置・デバイス	(分類タグ3.1) 装置・デバイス	(分類タグ4) 混入プロセス	(分類タグ4.1) 混入プロセス	〔障害を引き起こす①機能・処理〕	〔考慮漏れし易い②設計視点・観点〕
17	ストレージ管理	マウント／アンマウント	マウント失敗	オープン系システム	ファイルシステム領域とデータベース領域が共存するようなデータアクセス方式の異なるストレージが混在するシステム			システムを構成する各プログラムが、再起動時にマウントするストレージ領域とアクセスする領域	データベース構成をデータベース領域とファイルシステム領域に混在して作成したときの、システム再起動時の各プログラムの起動順序
18	フラッシュメモリ書き込み処理	フラッシュメモリ書き込み処理	書き込みエラー	携帯情報端末	フラッシュメモリ			フラッシュメモリ(書き込み寿命回数上限あり)	フラッシュメモリ上に配置するデータ設計・対策（配置されるデータ(お客様、AP等)を想定した設計・対策）
19	電圧管理	起動時電圧チェック処理	電圧提供停止	携帯情報端末				・起動電圧閾値の見積もり（電源ICの起動電圧閾値を参考に検討） ・電圧提供手段とその組合せ(充電池とAC充電器など) ・更に各電圧提供手段の状態（深放電状態など）・動作を加味した組合せ	・各電圧提供手段の状態・動作を加味した組合せ検討
20	電圧管理	部品の突入電流による電圧降下影響	システムリセット	携帯情報端末				・突入電流（始動電流） ・起動許可電圧閾値の見積もり（突入電流の影響見極め）	・突入電流による電圧降下影響（部品のバラツキも加味）を考慮した起動許可電圧閾値の決定 ・電池容量が少ない状態での突入電流の電圧低下の影響の見極め
21	データ通信処理	通信データの収集と蓄積	無効データへのアクセス			設計	設計意図	定周期で通信データの収集と蓄積を行う処理	収集データの有効／無効の判断を行う処理が何処に実装されているかの認識

③ 〔発生契機〕	④ 〔発生し得る障害内容〕	⑤ 〔発生メカニズム〕	⑥ 〔対策〕	
システム再起動	システム再起動により、データベース領域をマウントしたプログラムがデータベースにアクセスできない。	データベース領域とファイルシステム領域で構成されるストレージに対して、運用コマンドにてデータベースの一部をファイルシステム領域に追加。その後、システム再起動により最初にデータベース領域をマウントしてデータベースにアクセスするRDBMSが起動。しかし、その時点ではファイルシステム領域がマウントされておらずデータベース・アクセス不可となりRDBMSが運用モードにならない。	ファイルシステム領域に追加したデータベースの一部を、データベース領域に再登録。	教訓23
フラッシュメモリの領域への書き込み寿命回数を超える書き込みの発生	書き込み寿命回数を超えた領域が使用不可となり容量が減少する。その影響で以下が発生する。 ・起動不可 ・使用中に電源断、リセット、フリーズ等発生	・フラッシュメモリには書き込み寿命回数に上限がある ・装置のストレージとしてフラッシュメモリを採用。そこにはOSやお客様追加のAPなどが書き込みを行える状態。 ・お客様追加のAPが書き込み回数が多い場合、メモリの書き込み寿命回数の上限に達した領域が使用不可となりフラッシュメモリの使用可能容量が減少 ・結果、容量減少がOS起動にも支障を来し、起動不可、電源断、リセット、フリーズ等発生	フラッシュメモリなど使用する部品寿命を十分考慮し、運用（お客様のAP等）を加味したデータ配置・設計および対策	教訓6
以下の3つの条件を同時に満たす ・端末起動 ・電池の深放電状態 ・AC充電器接続による充電中	・端末が起動できず、充電も停止する。	・複数電圧提供手段がある（充電池とAC充電器など） ・充電池が深放電状態で起動電圧閾値以下の状態で、AC充電器接続による電圧提供して起動実行 ・起動時電圧チェック処理にて充電池＋AC充電器の電圧で計算し、問題なしで起動開始 ・しかし起動処理で充電を一度停止する仕組みがあり、その時に、電池ICの起動電圧閾値を下回る。 ・結果、電圧提供不十分で電源供共有が停止し、充電停止のままシステム停止	・起動時電圧チェック処理の設計では、各電圧提供手段とその状態・動作を加味した組合せ検討	教訓5
消費電力の多い機能・部品（3Gモデムなど）の起動による突入電流	突入電流による電流電圧低下により、以下の事象が発生する。 ・電源断（タイミングによりファイルシステム破壊） ・モジュールリセット ・装置のシステムリセット	・装置起動処理の中で、消費電力の多い機能・部品の起動開始し突入電流発生 ・突入電流による電圧低下にて電源断/モジュールリセットやシステムリセット発生 ※　電池容量が少ない場合は発生リスク高 ・結果、タイミングにより電源断によるファイルシステム破壊で起動不可など致命的問題発生	・あらかじめ突入電流による電圧降下（バラツキも加味）を考慮した装置設計（起動許可電圧閾値の決定など） ・電池容量が少ない状態での突入電流の電圧低下を加味した装置設計	教訓7
通信がアイドリング状態	蓄積データを処理する際に、無効なデータを有効なデータとして処理してしまう。	・通信データが存在しない時間帯（アイドリングタイム）に収集したデータ（NULLデータ）は、廃棄すべきであるが、そのまま蓄積データファイルに蓄積したため、無効なデータが含まれてしまった。 ・本来実装されていた「収集データの有効／無効を判断する処理」が、機能追加と処理速度の改善を同時に実施した際に削除されていた。設計意図を記録せず、変更点管理ができていなかった。	「通信機器のバッファから通信データを読み出す際、通信データの有無を確認する処理」追加し、通信データが無しの場合は、データ蓄積ファイルへの保存をしないようにソフトウェアを変更	教訓13

表 5.2　教訓集から抽出した設計知識（5/6）

	(分類タグ1) 機能・処理	(分類タグ2.1) キーワード “何が”	(分類タグ2.2) キーワード “どうなる”	(分類タグ3) 装置・デバイス	(分類タグ3.1) 装置・デバイス	(分類タグ4) 混入プロセス	(分類タグ4.1) 混入プロセス	① 〔障害を引き起こす機能・処理〕	② 〔考慮漏れし易い設計視点・観点〕
22	データ通信処理	生存監視処理	誤検知					生存監視	生存監視方法ではpingで応答があってもアプリで通信できない故障があること
23	データ通信処理	他社接続	不正メモリアクセス	業務システム				受信データ分析処理	規定されたフォーマット以外のデータを受信したときの設計漏れ
24	データ通信処理	データ通信処理	性能低下	業務システム	サーバーシステム			遅滞してはいけないデータ処理	ボトルネック箇所の推定と処理プログラミングの最適化
25	二重化	二重化	同期不全					二重化システムでの機能追加	二重化システムにおけるデータ同期の充分性
26	COTS	データベース	DISK満杯	業務システム	サーバーシステム	テスト		バージョンの違いによるログ出力	マイナーバージョンアップでも挙動が変化することがある
27	論理制御	条件数の多い論理制御	動作条件設定誤り					複数条件の組合せによる動作判定	条件の組合せパターンの可視化
28	論理制御	条件数の多い論理制御	動作条件設定誤り					複数条件の組合せによる動作判定	全体をモジュール分割し更に共通機能を抽出して共通モジュールに集約させることで、複雑なものが整理できること

③ 〔発生契機〕	④ 〔発生し得る障害内容〕	⑤ 〔発生メカニズム〕	⑥ 〔対策〕	
通信ドライバの異常	生存監視では正常なのに実際の通信が出来ない。	生存監視をping応答のみで実施していたが、相手装置の故障状況ではpingの応答があってもアプリケーションで通信できなくなった。異常と検出できないために送信バッファが満杯となってしまった。	生存監視を送信バッファの状態も見るように追加した。	教訓26
規定外フォーマットのデータ受信	規定外フォーマットのデータを処理しようとして不正メモリにアクセスしプログラム異常終了。	処理変更に伴い、処理を追加したが3つ目の処理が不要となったが処理を残したため規定外フォーマットの不正データを受信したときに該当のロジックが動作してしまった。	制御フィールドが規定外の値の場合は受信データを廃棄した。	教訓27
大量データ受信	一連のデータ通信処理の流れの中で、ボトルネックになり得る部分のプログラム処理が最適化されていないと、データ処理の遅滞が発生し、提供するサービスの断続的な停止を引き起こす。	複数の業務用携帯端末から送信されるデータをサーバーに蓄積し、複数拠点のクライアント端末が蓄積されたデータを解析処理するシステムで常時稼動と遅延の無いデータ授受が求められていたが、サーバー内処理プログラムの中に最適化されていない箇所（データ文字列をそのまま比較すればよいところを、一旦、数値に変換してから比較していたために処理時間の遅延を招いていた）があり、利用者の多い時間帯に、業務用携帯端末から送られてくるデータが一時記憶部（通信バッファ）に大量に滞留した。	・性能に関する非機能要求を整理する。 ・時間帯による負荷変動を考慮する。 ・ボトルネック箇所を推定し処理の最適化を行う。	教訓14
二重化システムでの切り替わり	データ同期が取れないまま切り替わることで、エラー発生や、切り替わり前と異なる動作をする。	機能拡張用の領域を使用して機能追加を行ってきたが、ある時点からその領域を超えたことに気づかなかった。二重化システムの単体の検証では正常動作するため問題を発見できず、二重化システムの切り替え検証で、データ同期範囲が不充分であることが判明した。	データ同期の範囲内であることを確認と、データ同期の閾値のチェックを行う。	教訓9
マイナーバージョンアップ	想定していないログが増えDISKを圧迫しシステム停止に至る	マイナーバージョンアップを実施したときに、ログ出力の条件が変更になっていたことに気が付かずログが増殖しDISKの空きを圧迫した。	ログの削除処理を追加した。	教訓28
条件の複雑化	複数条件の組合せによって実施動作を判定する制御ロジックにおいて、条件が複雑になってしまった場合は、条件組合せパターンの可視化を怠ると、制御ロジックの矛盾や不適合を引き起こすことが多くなる。	複雑化した複数条件の組合せによって動作を判定する制御ロジックにおいて、制御対象の構成変更に伴って条件の組合せパターンを変更する場合は、変更部分を可視化した上で検証しなければ、ロジックの矛盾や不適合を見逃してしまう。	複雑な条件組合せパターンの変更は、デシジョンテーブルで変更前後の違いを可視化して、ロジックの矛盾や不適合の有無を検証する。	教訓1
条件数の増大化	複数条件の組合せによって実施動作を判定する制御ロジックにおいて、条件数が増大化してしまった場合は、条件組合せパターンの可視化検証に限界が生じ、制御ロジックの矛盾や不適合を引き起こすことが多くなる。	複数条件の組合せによって動作を判定する制御ロジックにおいて、制御対象の構成変更に伴って条件の組合せパターンを変更した結果、特定のモジュールの動作条件数が増大してしまった場合は、条件組合せパターンの可視化検証に限界が生じるため、ロジックの矛盾や不適合を見逃してしまう。	動作条件数が増大してしまったモジュールを適切な規模のモジュールに分割し、条件の組合せパターンの可視化検証が可能なレベルに条件数を減少させる。	教訓2

表 5.2　教訓集から抽出した設計知識（6/6）

	(分類タグ1) 機能・処理	(分類タグ2.1) キーワード “何が”	(分類タグ2.2) キーワード “どうなる”	(分類タグ3) 装置・デバイス	(分類タグ3.1) 装置・デバイス	(分類タグ4) 混入プロセス	(分類タグ4.1) 混入プロセス	① 〔障害を引き起こす機能・処理〕	② 〔考慮漏れし易い設計視点・観点〕
29	学習機能	学習機能	学習条件設定誤り					自己学習機能による最適制御	学習機能のテストは、網羅度に限界があるため、小さな設計変更であっても、学習条件等の設計構造は可視化して確認する必要があること
30	論理制御	論理制御	算出条件設定漏れ					パラメータの組合せによる動作判定	パラメータ値域が広範囲でパラメータの組合せバリエーションが多くなり過ぎた場合、動作確認テストの網羅度に限界があること
31	並列処理	スレッド	データ破壊					マルチスレッドプログラミング	スレッドセーフ(関数が複数のスレッドから同時に利用されても正常に動作すること) ※IT用語辞典e-Words
32						要求定義	変更要求仕様	変更要求仕様の文書化	変更要求された仕様の背景事情や理由等の仕様書への記載と確認

※32 は、設計知識要因よりもプロセス要因の方がはるかに強いため、あえて例外的に分類タグ 1 を非設定にした。

③ 〔発生契機〕	④ 〔発生し得る障害内容〕	⑤ 〔発生メカニズム〕	⑥ 〔対策〕	
機能モジュールの統合	機能モジュールを統合した際に学習条件の設定漏れを見逃してしまうと、学習機能の低下によって制御失敗を引き起こす。	機能モジュールを統合した際に学習条件は構造ビジュアルツールによって目視確認したが、学習条件の設定漏れを見逃してしまった。	構造をビジュアル化して、統合前の学習条件数総和と統合後の学習条件数を総和の比較確認する。	教訓3
パラメータ値域の広範囲化、パラメータ組合せバリエーションの増大化	パラメータ組合せバリエーションに漏れや判定誤りが生じた結果、誤動作を引き起こす。	パラメータの組合せによる動作判定を行う処理において、パラメータ値域が広範囲でパラメータの組合せバリエーションが多くなると、テストを実施しても網羅度には限界があるため、レアケースで障害が発生する。	・モデル検査を導入する。 ・パラメータ組合せバリエーションが多い場合には、組合せ爆発が起こりやすいため、それを回避する方法として、パラメータ値域の範囲を同じ結果をもたらす集合（同値クラス）に分割し、各集合の代表値を用いたり、その同値クラスの境界値付近（境界値、境界値の前後）の値を持ちいたりして、パラメータの取りうる範囲を狭める。	教訓4
プログラム動作タイミング	スレッド間の処理競合が発生した場合、結果不正やプログラム異常が発生する場合がある。	スレッドセーフでない関数が存在する。 この関数がスレッド間で同時に動作すると関数内のメモリ等が競合し結果不正となる。	スレッドセーフでない関数を使用するときは排他制御を実施する。	教訓11
口頭による仕様変更	変更要求と異なった仕様でプログラムを変更してしまう。	簡単な変更であり口頭で指示をされたが、言葉に複数の解釈方法があり、指示内容を確認せず思い込みで変更した。 変更仕様の目的を正しく理解していれば誤った解釈はしないと考えられる。	変更要求はその背景事情や理由まで仕様書へ記載し、仕様の解釈誤りを排除する。	教訓25

参考文献

[1]「SSM による構造化知識マネジメント」，田村康彦　著，日科技連出版社, 2012 年 9 月
[2]「情報処理システム高信頼化教訓集（組込みシステム編）」2015 年度版，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, 2016 年 3 月
http://www.ipa.go.jp/sec/reports/20160331_2.html
[3]「障害未然防止のための教訓化ガイドブック（組込みシステム編）」，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, 2016 年 3 月
http://www.ipa.go.jp/sec/reports/20160331_3.html
[4]「現場で役立つ教訓活用のための実践ガイドブック（組込みシステム編）」，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，2016 年 3 月
http://www.ipa.go.jp/sec/reports/20160331_3.html
[5]「はじめての STAMP/STPA ～システム思考に基づく新しい安全性解析手法～」，独立行政法人　情報処理推進機構　技術本部　ソフトウェア高信頼化センター, 2016 年 4 月
http://www.ipa.go.jp/sec/reports/20160428.html
[6]「上流設計工程における未然防止プロセスの実用化に向けて―不具合モード発想力を高める秘訣―」，東芝ソシオシステムズ（株），JaSST'13 Tokyo, 2013 年 1 月
http://www.jasst.jp/symposium/jasst13tokyo/pdf/D2-3.pdf
[7]「上流設計工程における 未然防止プロセスの提案，―未然防止リストの活用と欠陥の発想―」，東芝ソシオシステムズ（株），JaSST' 12 Tokyo, 2012 年 1 月
http://jasst.jp/symposium/jasst12tokyo/pdf/D2-3.pdf

執筆者

【未然防止知識 WG】

主査	久住　憲嗣	国立大学法人九州大学
	内平　直志	国立大学法人北陸先端科学技術大学院大学
	石川　学	横河電機株式会社
	石原　鉄也	矢崎総業株式会社
	岩橋　正実	三菱電機株式会社
	植武　信弘	株式会社日立産業制御ソリューションズ
	木村　裕之	日本電気株式会社
	鈴村　延保	アイシン・コムクルーズ株式会社
	高木　徳生	オムロンソーシアルソリューションズ株式会社
	羽田　裕	日本電気通信システム株式会社
	細谷　伊知郎	トヨタ自動車株式会社

（50 音順）

三原　幸博	独立行政法人情報処理推進機構
十山　圭介	独立行政法人情報処理推進機構
石井　正悟	独立行政法人情報処理推進機構
松田　充弘	独立行政法人情報処理推進機構

監修

製品・制御システム高信頼化部会