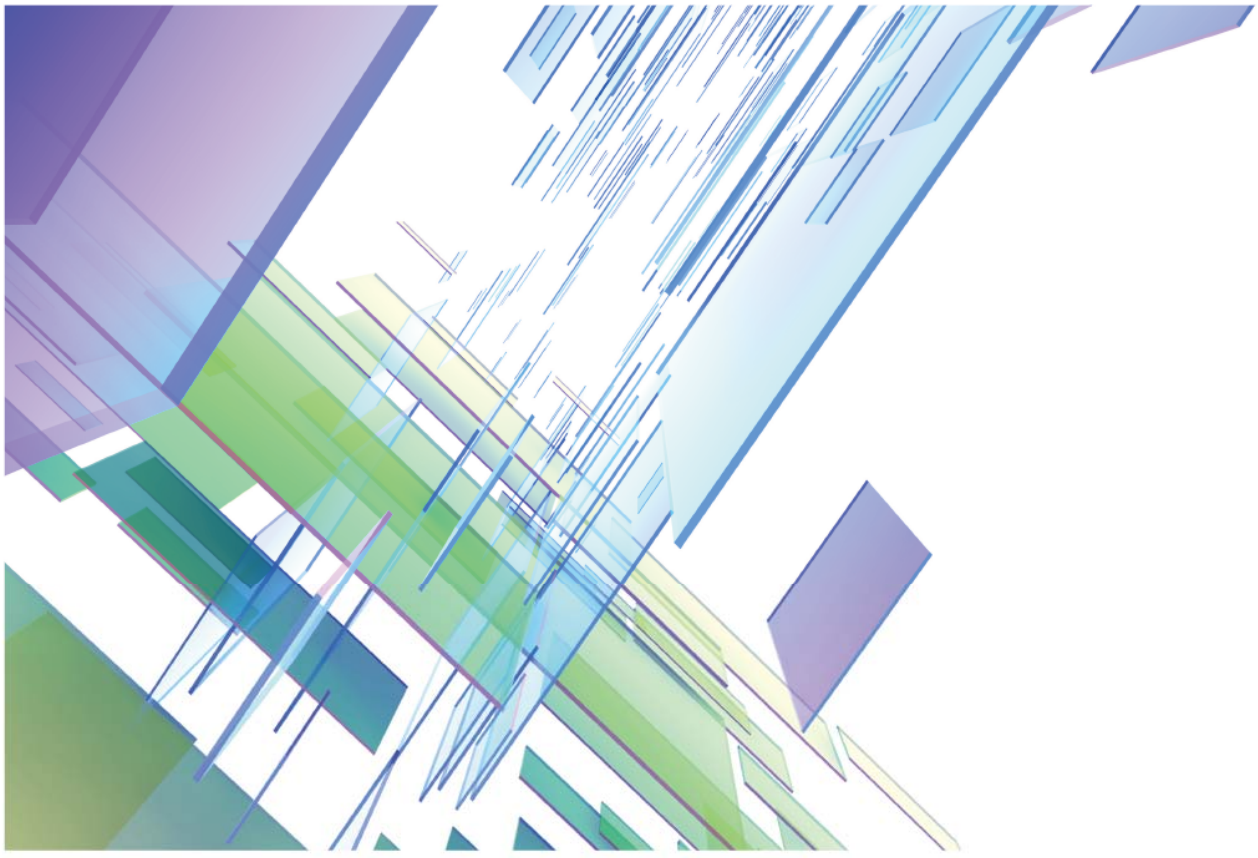
IPA Better Life with IT

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（IPA）
技術本部 ソフトウェア高信頼化センター（SEC）

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

目次

1 はじめに

1.1 目的

本ガイドは、組込みシステムの実際の開発現場において行われている障害対応の際に作成される再発防止策に基づいてどのように未然防止の知見、ノウハウを教訓として導出してゆけばよいか、組織活動として教訓をいかに定着させてゆけばよいかを実践する際のガイドとして作成したものである。

なお、本ガイドを活用する際には「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）[1]（以下“教訓集”と称する）ならびに、「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）」[2]なども参考にされたい。

1.2 対象読者

本ガイドの利用者としては組込みシステム開発に関わる以下の読者を想定する。

・ソフトウェア設計者
・システム設計者
・品質管理者
・技術者教育担当者

1.3 用語定義

本ガイドの中で重要と思われる用語について定義する。

再発防止：製品・システムで実際に発生した問題（故障・不具合）に関する根本的な原因を追究し、その原因に対して当該製品・システムで今後同じ問題が発生しないように対策を講じること。
（参考）問題の原因又は原因の影響を除去して、再発しないようにする処置（JIS Q 9024:2003）

未然防止：製品・システムで得られた再発防止の知見に基づくリスク要因を、他製品やシステムへ適用し類似の障害発生を予防する取組み
（参考）起こり得る不適合又はその他の望ましくない起こり得る状況の原因を除去するための処置（ISO9000「予防処置」）

教訓：製品ドメインに限定、特化することなく、当事者以外の人、他製品・システムにも役立たせることができる経験知識・ノウハウ

原因分析：障害発生時、その情報を収集しシステム構造把握、問題構造把握を行い、障害を引き起こした直接原因を分析するまでの一連の活動

真因分析：原因分析によって明らかにされた直接原因に対し、その原因を発生させるに至った真因を抽出するための分析

障害の原因：障害を発生させた要因

障害の真因：障害の原因のうちの本質的な原因

障害の誘因：障害発生を引き起こす背景となる要因

知識継承：送り手の頭の中にある暗黙知を、仕事や世代等の異なる受け手が受け取れる形式知に変換し、具体的行動として実行できる内容とすること

1.4 本ガイドの構成

本ガイドは前掲の 1.1 節の目的に鑑み、以下のような構成となっている。

２章　教訓化ための概念モデル
３章　教訓化の定着に向けたプロセスと組織活動
４章　実践的アプローチ
５章　未然防止に向けた企業内事例

第２章では、得られた再発防止の知見を未然防止の教訓として一般化してゆく上で基本となる、概念モデルや考え方を解説している。

第３章では、障害発生ごとの対応プロセスと、未然防止の教訓を定常的に活用するための組織活動はいかにあるべきかについて概説している。

第４章では、教訓化作業を企業内で実践する際の具体的な手順や留意点を、体験型ワークショップ事例も含め自社内で教訓化を行う上での手順と要点を記述している。

第５章では、実際に企業内で行われている未然防止に向けた取組み事例を紹介している。

2 教訓化のための概念モデル

過去の障害に基づく再発防止の知見から類似要因による障害発生を未然に防止するための教訓を得ながら、組織としての経験知を高めてゆく一連の取組みは、当事者の知識を共有し他者に受け継いでゆく知識継承活動である。本章では本ガイドで拠り所としている知識継承のバックグランドとなる概念モデルについて説明する。

2.1 知識継承のモデル

知識継承の古典的概念として SECI モデル[3]がある。これは、「共同化(socialization)」「表出化(externalization)」「連結化(combination)」「内面化(internalization)」の４つのモードによって暗黙知と形式知の絶え間ない変換を行いながら知識を創造するモデルである。

この SECI モデルに基づき、障害事例情報を媒介に送り手の知識を「表出化」（暗黙知を第三者にもわかりやすい情報に抽出・変換）し、受け手へ「内面化」（受け手が実践行動できるように形式知を変換）するような知識継承モデル（図. 2-1）が提唱されている。[4]

障害対応を実際に体験し、その知見を伝えたいと考える当事者（送り手）が、自らの経験で得られた知識を事例として表出化する。次いで社内教育などの場でその知見を学ぶべき受け手は、その事例を活用して知識を内面化し、再発防止、未然防止の取組みを実際に行動に移すことができるようにする。本ガイドでは基本的にこの概念モデルを念頭に置いた教訓化の取組みを説明している。

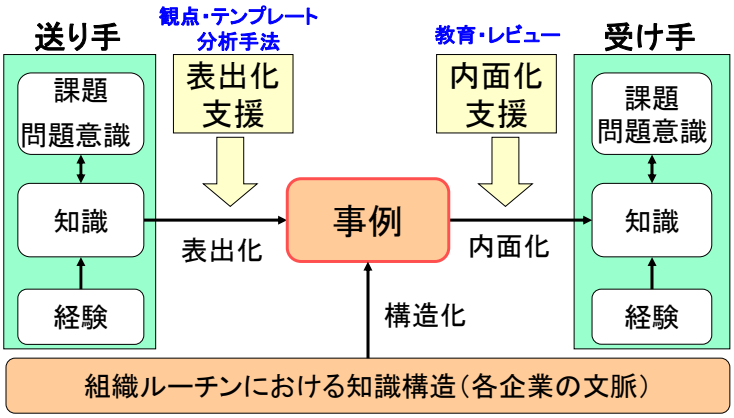


図. 2-1　本ガイドにおける知識継承モデル

2.2 事例抽出・蓄積の工夫

教訓事例は通常の開発プロジェクトとは異なり、障害発生時の対応の中で作成されることになる。今日多くの企業では、そうした障害情報と対応結果の情報をデータベース（過去トラデータベースなどと称される）として管理している。記録される情報としては、発生した経緯、原因とその再発防止策、必要に応じた水平展開情報などがある。

こうした情報は事象の記録として必要だが、往々にして一過的であり、障害の対応が終了するとそれに関する情報は顧みられず、以降有効に活用されることが少ないと言われている。製品開発プロジェクトでは、開発終了時にポストプロジェクトレビュー(Post-project Review)といった振り返り分析を行い、実施したプロジェクト活動の中から学ぶべき内容を確認し、将来のプロジェクトに活かそうとしているのとは対照的である。

これは、過去トラデータベースは障害対応の経過を管理しエビデンスとして保管するという目的に主眼が置かれていて、後々に活用できるようなものとして設計、運用されていないことにも一因があり、活用のためには後述する一般化を前提とした経験知の抽出・蓄積の工夫が要る。例えば、一般化のための観点を体系化し、その観点をカテゴライズしたキーワードを予め用意し、テンプレート化することも一つの方法である。本ガイドではそうした工夫の一例として 3.2.7 項に観点マップを掲載した。

2.3 継承手段

表出化された先人の知識を継承する手段として、ケースメソッド、ベストプラクティス、失敗知識データベース、ストーリーテリングが知られている。

①ケースメソッド

事例を用いて教育する手法としては、ケースメソッドが普及している。一般的にケースメソッドにおける「ケース」とは、経営に関する出来事や状況を記述したものとなるが、ここでは障害対応に関する一連の取組みとそこから得られる知見が対象となる。

ケースメソッドは、ケースを用いた擬似的な経験を通して、必要な能力を身につける上で有益な方法と言われており、本ガイドでも 4 章で紹介する体験型ワークショップで採用している。

②ベストプラクティス

ベストプラクティスとはある結果を得るための最も効果的な方法、最良の事例という意味であり、開発プロジェクトや IT 導入における成功要因などで多く使われている。

ベストプラクティスを収集する際には、成功の定義並びに成功要因と成功のための KPI (Key Performance Index)の抽出が重要になる。

③失敗知識データベース

不具合事例集や不具合データベースを使った知識継承も従来から行われてきた方法で、失敗を構造化しその構造に基づき失敗事例を蓄積・分類・活用する「失敗学」が知られている。[5]　これは失敗を「事象」「経過」「原因」「対処」「総括」「知識化」の６項目で整理し、「失敗まんだら」として体系化したものである。

④ストーリーテリング

ストーリーテリングも事例を利用した組織学習の手法である。組織におけるストーリーテリングにはその目的別に７つのパターン（行動を引き出す、みずからの人となりを伝える、価値観を伝達する、コラボレーションを育む、噂を管理する、知識を共有する、人々を未来に導く）が知られている。

3 教訓化の定着に向けたプロセスと組織活動

組込みシステムを開発する多くの企業では、予見されるシステム障害の発生に備えた対応を定め、適切な対処を行うための仕組みを保有しているが、大規模で複雑なシステムが相互連携するIoT(Internet of Things)時代の今日、発生するシステム障害を予見することは容易ではない。

組込みシステムの中で動作する変数とその組合せ、それらに対応する動作処理パターンは多数存在し、また人間の誤認識や操作ミスをゼロにすることも簡単ではなく、それら要因の組合せを予め全て網羅し尽くすことも困難である。従って障害の経験に基づく再発防止、未然防止の知見を教訓として体系化し、組織や社会の中で体得するという学習サイクルをいかに運用していくかが重要になる。

3.1 障害対応時のプロセス概要

本節では個別の障害発生から再発防止策を作成するとともに、そこから得られる未然防止の知見を組織の形式知として蓄積し、3.2 節の継続的な組織学習のサイクルにつなげてゆくための活動プロセス（図. 3-1）について、ステップごとに要点を解説した。なお、本プロセスは 3.2 節図.3-5 のルーチンワーク以降に発生する障害発生から教訓作成の部分に位置付けられる。

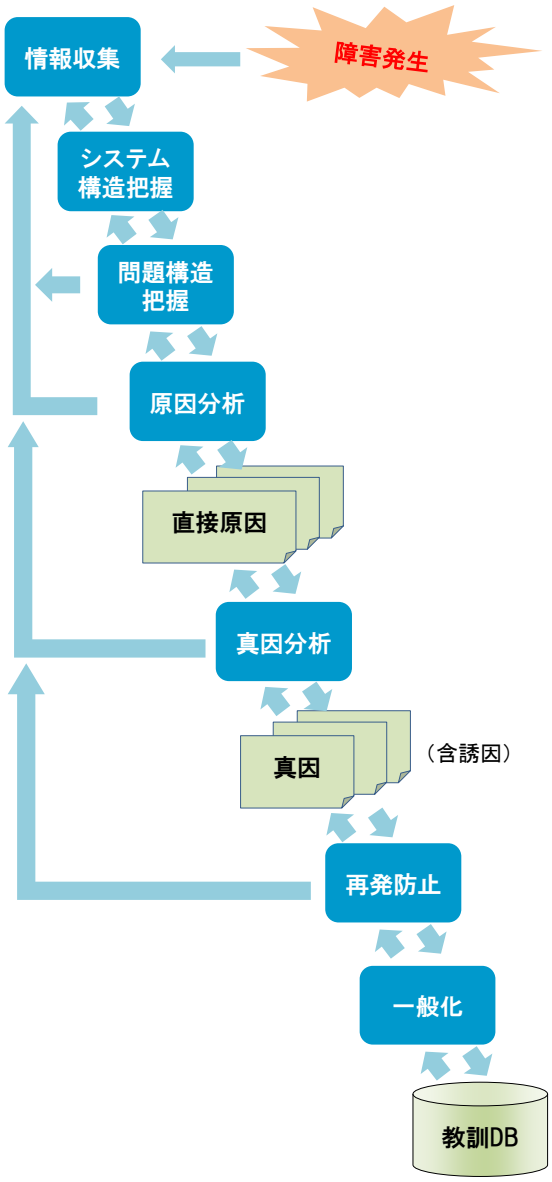


図. 3-1　障害対応プロセス

3.1.1 情報収集

障害発生時には速やかな情報収集が必要である。障害の深刻度や影響もシステムによって様々で、例えばメモリの間欠故障や２重化システムの片系故障などではただちにシステム全体の停止には至らないと予測されるなど、障害時の対処もそのシステムの特性や障害内容によって異なってくる。

障害の分析に必要となる収集すべき情報例として以下の 5 つが挙げられるが、システムごとに使用環境条件も同じではなく、必要な情報も異なる。当事者へのヒアリングや情報収集、あるいは現象を再現するための環境を構築してヒートラン（一定条件下での長時間にわたる連続動作）の実施を必要とすることもある。

- 障害の発生状況（関係者に対するヒアリングを通じて）
- システムの稼働ログ情報
- 障害の再現手順
- システムの構成
- 外部環境

3.1.2 システム構造把握

収集した情報に基づいて障害原因を究明してゆく上においては、障害を起こした対象システムの内部構造の理解が前提となる。この作業は当該システムに詳しい開発・設計部門が中心となり、収集情報に基づいて事象のシーケンスを追うことになる。しかし、既存の設計資料はハードウェア、ソフトウェアの各コンポーネントやユニット毎となっていたり、外部業者からの購入物であったりと、全体を把握するための俯瞰的情報が常に揃っているわけではない。

そこで関係すると思われる設計や製造資料、業者提示情報などを整理して、システムの全体像を見渡せるようなシステム構造図を当事者で作成することになる。対象システムが複数のサブシステムから構成されていることも多いが、障害状況を見渡し、原因部位を推定するための全体図を用意することは、サブシステム間の関係が明確になる等、対象システムの構造把握に役立つ。

3.1.3 問題構造の把握

時系列に発生した状況をシステム動作と突き合わせながら、問題となった症状を把握するために、「人」「モノ」「情報」について構造と動きの側面で理解してゆく。

組込みシステムではシステムの構成物が機械構造物や電子制御装置、ソフトウェアなどの複合的要素から成り、熱、圧力などの物理的現象を取り扱うことになる。このため全体構造を理解する上では、機械／電気／電子／ソフトウェアの各担当者、物理現象やシステム特性に詳しい専門家による総合的な視点が欠かせない。原因を探ってゆく上で情報やデータに不足がないか、どこにそうした情報があるかを常に考えながら探索しなくてはならないからである。

3.1.4 原因分析

障害を引き起こした直接原因を特定するためには、前掲したような収集情報に基づき制御ロジックやプログラムコードをトレースする、再現試験を行うなどさまざまな方法が用いられる。重要インフラシステムでは障害時には復旧を優先しなくてはならない場合も多いと考えられるため、直接原因を特定し、復旧できた後に真因分析の段階で「なぜなぜ分析」を適用することになる。

また、マルチベンダー環境での障害原因の切り分けが必要な場合や、ハードウェア・ソフトウェアの購入品について海外を含めたサプライヤーに詳細データを求めるなど社外に遡及するようなケースでは、当面の回避策検討が急務である。特に COTS（Commercial Off-The-Shelf）製品は内部仕様が公開されないブラックボックスであることを前提とせざるをえないため、導入に際しては、バージョンやパッチの管理や不適合発生時の回避策も可能な限り設計時点で事前検討しておかなくてはならない。

3.1.5 真因分析

直接原因を生じさせるに至った真因として近年では人的過誤等ヒューマンファクターに起因する重大障害が多く散見されている。しかしながら、これらの障害を個人の責任に帰すだけでは単なる精神論となってしまう。再発防止策を具体的アクションに落とし込めるよう真因が的確に抽出されているか、問題事象の原因－結果の因果関係に無理や飛躍はないか、などの観点で分析すべきである。

そうした意味からも真因分析時に追求の観点をあらかじめ定めておくことが有益であり、例えば「不具合を作りこんだ要因」、「不具合を流出させた要因」などで分析することは一般的に行われている。

3.1.6 再発防止

障害を発生させた真因が適切に抽出できていれば、再発防止策を導出することは比較的容易にできるかもしれないが、実際には適切な真因を抽出できるまで何回か分析と再発防止策導出の手順を繰り返すことが多い。また障害を作りこんだ直接原因に比べると、流出させてしまった真因に関する再発防止策は、レビューを確実に行うというようなマネジメント的対策になりがちである。

再発防止策がレビューやチェックシートによるチェックとなることは間違いではないものの、安易にそうした対処にしてしまうと、形骸化を招き実質的に再発防止効果がない対策になってしまう。また重要システムの障害では当該システムを利用して事業を行う顧客企業の納得を得ることが重要であり、再発防止策の策定においては、事業者・利用者の視点が求められる。

3.1.7 一般化

再発防止までの活動で得られた知見を、障害が発生した当該製品・システム以外にも適用することは、未然防止の基本的な取り組みであるが、再発防止で得られた知見のままでは適用できる製品・システム・組織は限定的である。これは、

- 使用している動作原理、技術（要素技術、開発技術）
- 商流・ビジネスモデル
- 開発プロセス
- 組織風土・不文律

などが異なるためであり、同じ会社であっても事業部門や工場ごとに上記の要素が大きく異なっているということは珍しいことではない。そこでさらに広く他の製品・システム・組織に知見を適用し、障害の発生を未然に防止するためには、得られた知見を異なる製品・システム・組織に共通的に使用できるものとする必要がある。

このような行為は、一般化や上位概念化などと呼ばれる。あるいは、得られた知見を適用したい製品・システム・組織に共通する性質に抽象するために抽象化と呼ばれることもある。ソフトウェアおよびシステムズエンジニアリングの分野では、類似の用語として「generalization：両方のエンティティのインスタンスが同じ実体的なまたは抽象的なものを表す分類法（ISO/IEC/IEEE 24765)」がある。

いずれにせよ、得られた知見の適用を広く求める場合には、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・転換する作業を行わなければならない。この作業をここでは一般化と呼び、一般化と後述する未然防止の教訓作成を図.3-2 のような真因と真因に対する対策の本質を理解した上での置換・転換と定義する。

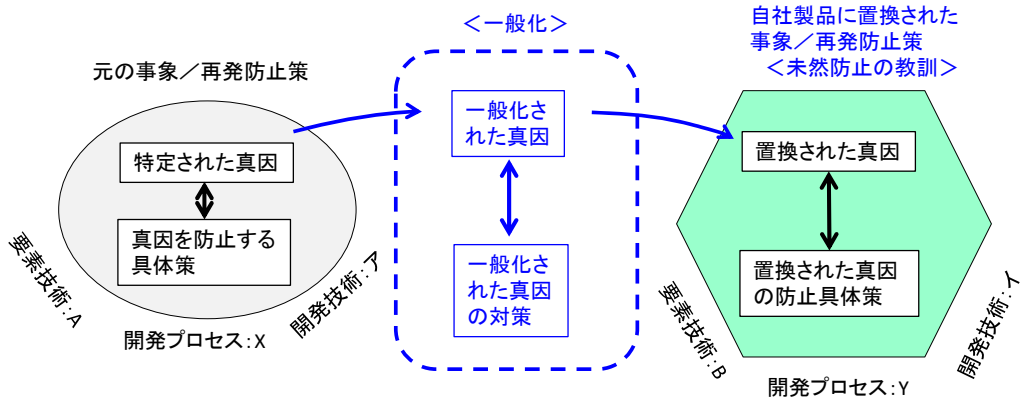


図. 3-2　一般化

この置換・転換の際に想定する製品・システム・組織の範囲（カテゴライズ）については図.3-3、3-4 の観点マップの分類を参考にすることができる。

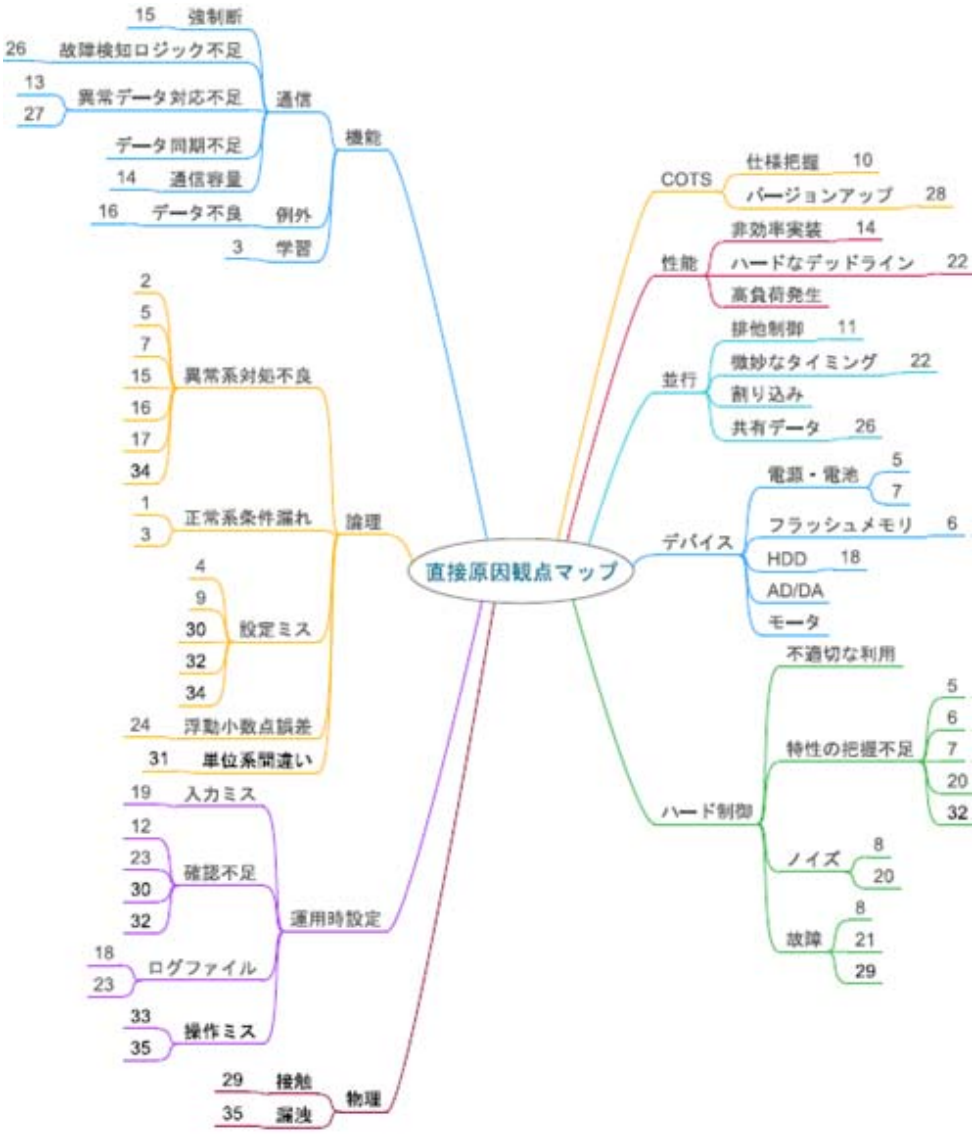


図. 3-3　直接原因観点マップ

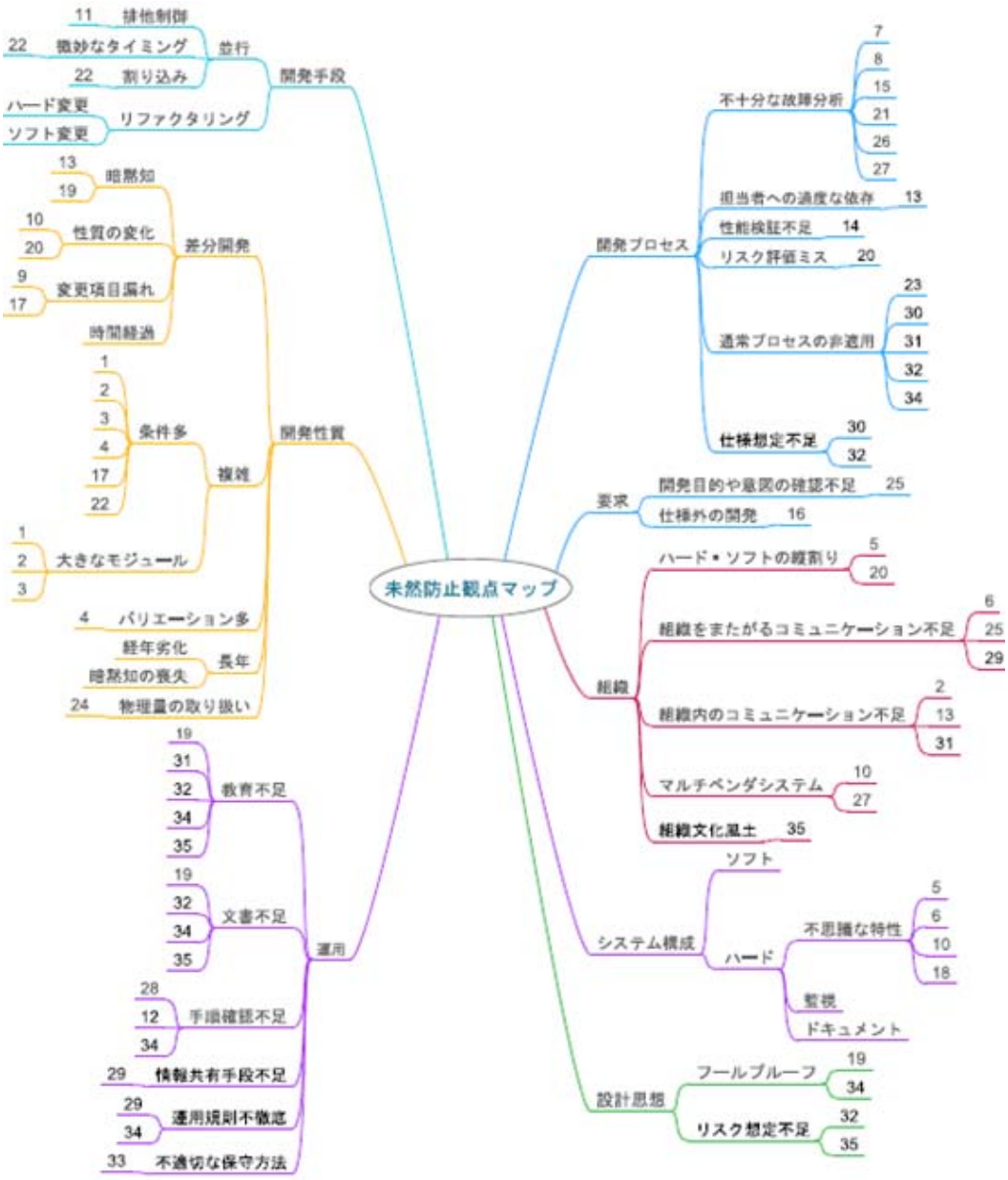


図. 3-4　未然防止観点マップ

3.2 継続的な組織学習サイクル

図. 3-5はこの組織学習の継続的な運用イメージを示したものである。フィールドで発生する障害対応の教訓を、日々の開発業務や運用保守といったいわゆるルーチンワークの中で活用・評価し、また新たな教訓が得られた際は教訓データベースに登録する。教訓の中で自部門製品・技術に対しても有益と思われるものは、定期的に行われる部門内の情報共有ミーティングまたは、教育・研修などの場において適宜選択し、活用することができる。

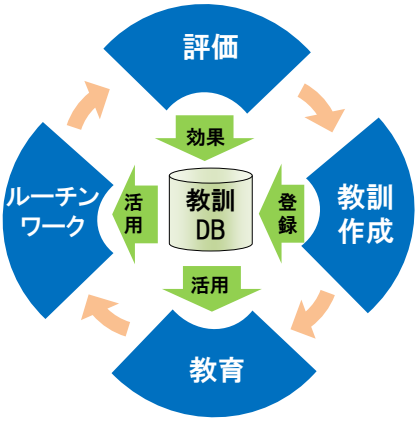


図. 3-5　組織学習サイクル

年度や期の節目ごとに行われる品質活動の定期フォロー等のタイミングで、個々の開発プロジェクトや品質保証業務においてどのような効果があったかが評価され、ルーチンワークでの活用促進に向けたフォローが推進されることになる。

こうした活動が実際の製品開発の中で意味を持ち、絵に書いた餅とならないよう継続運用されてゆくためには、

- 開発に携わる実務者に気づきを与える
- 組織のルーチンワークの中に定着させる

ことが、車の両輪として定常的に行われなくてはならない。

3.2.1 気づきを与える

気づきを与えるにはどのようにすればよいのだろうか。一般的な取組みとして、経験の浅い若手技術者向けに座学による研修を実施することが広く行われている。（図.3-5 の教育に相当）。一方さらに踏み込んだ取組みも行われている。例えば、失敗の模擬体験を行うための環境を整えグループワークによる集合教育を行って参加者に失敗を体験してもらい、なぜそうなったのかを考えてもらう。

またあらかじめ誤りを混入させた模擬システム環境を準備し、実習形式で参加者が間違い探しをするというような研修を行っている企業もある。さらに、4 章でも触れているが、リスクを感じるための感度を向上させることも有効な手段と考えられ、実際そうした集合研修を実施している企業もある。

この感度を上げる試みということでは、従来製造現場で繰り返し行われている、KYT（危険予知トレーニング）がある。KYT では、現場作業や路上通行における危険要因を参加者同士で話し合うというような、誰にでも理解できるように比較的単純な例が使用される。これを応用し、障害要因と再発防止から得られる教訓エッセンスを KYT のようなやり方で訓練することは一定の効果が得られるものと期待できる。いずれにしても何らかのきっかけを与えて、自分自身で考えてもらうということが大切である。

3.2.2 定着化に向けて

何かを始めることは容易であっても、それを定常的な活動として継続させることは難しい。未然防止のための教訓化を定着させてゆくことも例外ではなく、これを定常的な活動としてゆくためには、様々な工夫を取り入れた不断の努力が求められる。

多くの現場で行われている取組みの一例として、プロセスへのフィードバックがある。同種の不適合を未然に防止するために開発プロセスの該当工程に対策を反映する、工程自体を見直す、あるいは開発中における問題の作りこみや流出を予防するために、開発レビューのチェックリストに追記して見落としがないかを確認し、ツールによる動作確認を義務付けることもある。但し、よく言われるようにチェックリストが肥大化すると、作業が機械的になり、本来の目的を果たさなくなるため、定期的な見直しなどを行い形骸化防止に努めなくてはならない。

また、過去の経験の臨場感が失われないよう図.3-5に示した組織学習サイクルを確実に回すことが大切であり、このためにも経営層による働きかけが重要となってくることは言うまでもない。

4 実践的アプローチ

4.1 体験型ワークショップの意義と狙い

前章で示した一連の手順を具体的に自社、自組織でどのように進めるかを汎用的に示すということも簡単ではない。そこで、より実例に近い障害例を教訓集に基づいて作成し、実際に企業内で行われるような障害対応を意識しながら解説することを考え、実際にこのコンセプトに基づいてワークショップ形式による実践アプローチを行った。本ガイドはそうした実践的トライアル活動の内容に即した説明となっている。

一般に体験型ワークショップとは、問題解決を図る、知識を学ぶなどの目的を達成するためのトレーニング方法で、2.3 節で示したケースメソッドの 1 つとして位置付けられる。座学教育と異なり、参加者が自発的に作業を行い、他参加者等周囲とのコミュニケーションを行いながら体験学習を行うものである。

本章ではこの体験型ワークショップのためにあらかじめ作成したケーススタディを用い、そのプロセスと結果から得られたポイントなどを併せて記述した。読者企業において類似の社内教育を行う上で参考となれば幸いである。

4.2 未然防止のための体験型ワークショップ概要

体験型ワークショップ研修では、3.1 節の障害時対応手順を 3 段階のステップにまとめて実施した。

Step1）実際の障害事象に基づいてその内容を理解するステップで、情報収集から問題構造把握のアクティビティに該当する。

Step2）なぜなぜ分析を用いて障害の原因分析を行い、再発防止策を立案する。ここは原因分析を行って再発防止策を得るまでに相当する部分である。

Step3）この事例から得られる知見の本質を理解し自身の製品・システムに置き換えて、同様の障害発生を未然に防止するための教訓を作り、当事者以外に伝えられる内容となっているかどうかを確認するまでの一連の作業を行う。

以下の図. 4-1 にワークショップにおける各手順の時間配分を目安として示した。

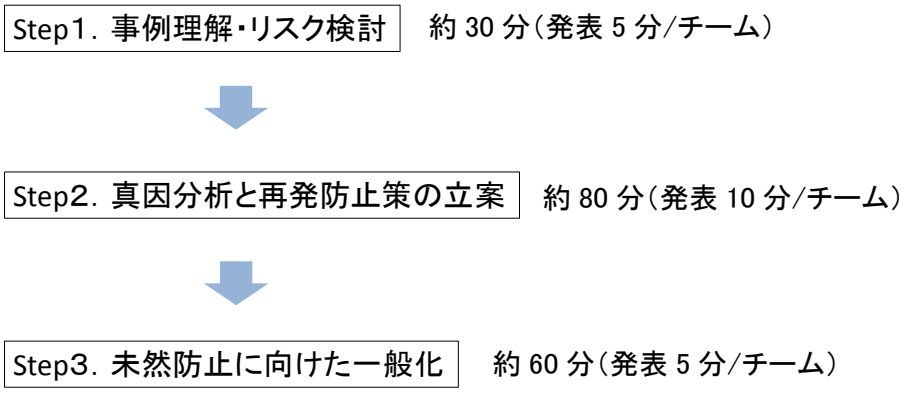


図. 4-1　手順と時間配分

4.2.1 Step1：事例理解とリスク検討

事前準備したケーススタディ資料を読み内容の理解を行う。但し、実際にグループワークで行う都合上、教訓集に記載されている元々の事例情報に対し、事象の推移や対象システムの詳細などを創作・付加している。（登場人物名、システム名称は全て架空のものである）

また Step1 の最初では KYT の要領にならい想定されるリスク検討を行っている。この手順は実際の障害対応では復旧を優先するために実施されないことがほとんどであり、ワークショップでも必須ではないが、ここではケース事例内容をよく理解するとともに、集合教育のアイスブレークとしての意味もあり取り入れている。本来 KYT は現場の災害や事故予防のために行われるものだが、所与の状況の中でリスク発生の可能性を自ら考えることはシステム開発におけるリスク感度を磨く上でも有益と思われる。

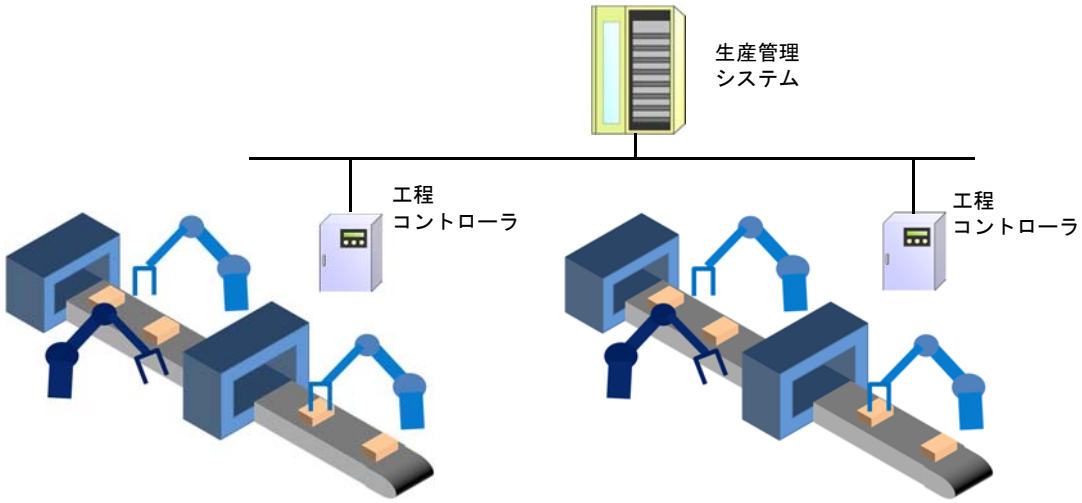
以下に今回トライアルで使用したケースを示した。このケースはワークのステップ進行に従って受講者に段階的に情報提示してゆくことを念頭に、意図的に前半部分と後半部分に分割されている。

ケースＡ（前半部分）

【背景・経緯】

システムの概要：

■■■産業は金属加工成型製品の製造メーカで、その生産工場ではロボットも活用した全自動に近い生産ラインが稼働している。



生産ラインは加工や検査など多くの工程から成り、溶剤処理・熱処理工程においてはその薬剤投入量や温度管理、次工程までの搬送時間管理が重要になるため、温度センサー、位置や速度検知のための多数のパルスセンサーが各部位に設置されている。

各製造工程単位に製造機器の制御を管理する工程コントローラが置かれ、全ての工程コントローラは全体を制御する生産管理システムによって監視制御されている。

生産管理システム、工程コントローラは●●システムズから購入し、工程コントローラ上で動作する制御ソフトウェアも●●システムズに委託し開発した。

工程コントローラの制御ソフトウェア：

工程コントローラの制御ソフトウェアは、生産管理システムからの製造指示に基づき、センサー入力情報を受けながらロボットの動作やコンベアのモータ回転制御等をリアルタイムに処理する必要がある。

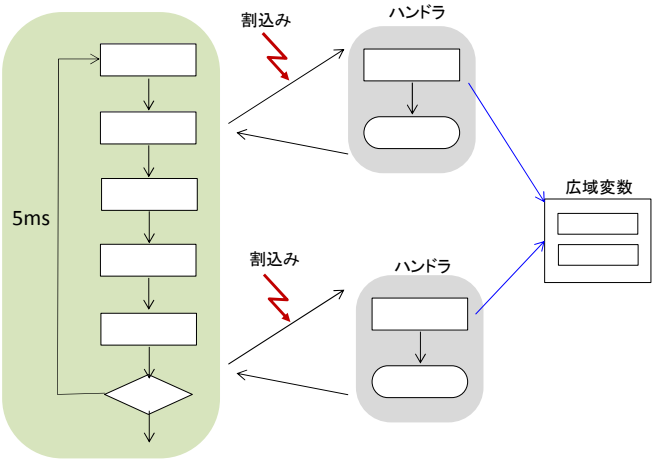
そのためメインルーチンは 5ms 周期で動作し、このインターバル中に割り込み処理を受け

付けるとともに、5ms 周期の時間を用いて PI 制御に必要な P 要素、I 要素の時間カウントも行っている。

このメインルーチンでは多くの広域変数を使用し、割込み処理の中でもそうした広域変数を操作する可能性もあるため、排他処理を行い意図しない書き換えが行われないよう注意深い制御が求められる。

また、メインルーチンには多くの条件分岐があり、その中で割込み処理を行う上では 5ms の周回時間内で完了しなくてはならないので、最悪実行時間（WCET: Worst-case execution time）を十分考慮することが重要になる。

この■■■産業向け工程コントローラの制御ソフトウェアを立上げの当初から開発してきた●●システムズの高橋主任は、5ms 周期含め上述したような重要ポイントは制御ソフトウェア設計仕様書に記述してきた。（基本設計書もあるが、もっぱら詳細仕様が記述されているこの制御ソフトウェア設計仕様書に記述し基本設計書はあまりメンテされていない）



開発プロセス：

●●システムズではシステム開発を行う場合、新規開発／改良開発の別に準拠すべき開発プロセスが定められており、設計、製造、試験のマイルストーンでデザインレビューを実施することが必須となっている。

最初に開発した当時は、フルスペックの新規開発プロセスを適用して開発したが、以降はもっぱら改良開発プロセスが適用されてきた。これらプロセスは●●システムズが過去の経験に基づいて体系化してきたもので、実適用の都度、開発案件の特性に応じたテーラリングをすることも認められている。近年は改良開発がほとんどで、改造部分のレビューでも高橋主任の個人的活動に依存する傾向があった。

ソフトウェア開発の経緯と変化：
■■■産業では 5、6 年ほど前から多品種少量生産の比率が増え始め、これに伴い生産ラインの改造や新規の設備機器が導入されてきた。また製造工程作業のバリエーションも増えるに伴い工程コントローラに実装される機能も増加し、プログラム量もかなり肥大化してきた。

高橋主任はその都度、変更の要点を制御ソフトウェア設計仕様書に反映し、またソースプログラムの中にも注記（コメント）として記述してきた。但し、記述にはクセがあり高橋主任以外の人にはわかりにくい箇所もあった。

【マイナーチェンジ対応】
ある時、■■■産業からある次期新製品製造に向けた試作をしたいので、ある特定の生産ラインの工程コントローラの動作仕様を部分改造したいという引き合いを受け、●●システムズは高橋主任をリーダーに任命した。

■■■産業からの要求は工程コントローラの制御ソフトウェア中の定数値を変更し、変数も追加し材料の予熱や成型、加熱のそれぞれの経過時間を変更しながら、サンプルを製造するというものであった。

高橋主任はちょうど他案件対応で非常に多忙な状態であったが、既存の処理ルーチンを部分修正すれば可能と考えた。開発プロセスも今回は試作のためのマイナーチェンジであり、改造開発のプロセスで進めることになった。

一方高橋主任自身はほとんど時間を割けそうになく、協力会社に依頼した。その結果、田中さんがアサインされることになった。

田中さんは C 言語や Java には詳しいが、実際の組込みシステムでの経験はあまり多くなく、Web 系システムの実務経験の方が多かった。が、高橋主任は今回の改造程度であればなんとかなるだろうし、自分が最終チェックすれば大丈夫だと考えた。

田中さんは制御ソフトウェア設計仕様書を見ながら改造対象ソースプログラムを確認し始めた。ソース中の注記記述は意味がよくわからない箇所も多く、高橋主任に不明箇所について確認しながら進めた。

5ms 周期の周回動作をキープするために実行時間に対する制限事項があり、高橋主任からも説明してもらったが、既存の処理ルーチンにはできるだけ手を加えずに踏襲するのが最も安全だろうと考えた。そこで今回試作用のサブルーチンを既存の処理ルーチンを参考にしながら新たに作り、メインルーチンから呼び出すようにした。

改造開発プロセスの設計レビューでは変更箇所を確認し、模擬試験環境でのテストを行った上で最終レビューを行うことになっている。設計レビューで高橋主任は田中さんのプログラムを目視で確認し、「やや冗長な感じがする。本当はメインルーチンを変更すればもう少し簡素化できると思うが、まぁいいか･･･」と思い、いくつかコーディングの軽微なミスを指摘するにとどめた。

その後、田中さんは指摘内容を反映し模擬試験環境でテストを繰り返し、動作に異常がないことが確認され、作業要領書も作成し最終レビュー合格となった。この１週間後に現場の工程コントローラのマイナーチェンジ改造が開始された。

【設問】

以上までの状況についてどのようなリスクが考えられるか、リーダーの高橋主任の立場で考えてみましょう。（←ここは高橋主任の上長もしくは●●システムズの責任者の立場としてもよいかもしれない）

以下に本ケースにて抽出が期待されるリスク例を示す。

【リスク解答例】

・田中さんは組込みシステムの経験、■■■産業の当該システム知識が不十分なため、システム動作上の制限や留意事項を考慮せず改良作業を行ってしまう。

・制御ソフトウェア設計仕様書やソースコードの注記は高橋主任以外には説明を受けてもわかりにくい内容であり、深く理解しないまま改造してしまう。

・設計レビュー時、高橋主任は目視で確認しているため、重大な問題を見逃してしまう。

・■■■産業のシステムを高橋主任以外のエンジニアが理解していないため、代替することができない体制になっている。

・制御ソフトウェア設計仕様書には記述しきれていない重要情報がある。（ベテランの高橋主任の暗黙知になっている）

・既存の処理ルーチンをそのまま踏襲しているため、流用した結果新たな潜在バグが顕在化する。

・時間が経過するにつれ■■■産業のマイナーチェンジ仕様の追加・変更が発生し、当初想定していたような改良開発の範囲を超える内容になってしまう。

・長年の改良開発の中で開発プロセスのテーラリングが形骸化している可能性があり、ルール通りにプロセス適用せずに開発してしまう。

4.2.2 Step2：真因分析と再発防止策の立案

実際に発生した問題事象情報（後半部分）を提示し、なぜなぜ分析を行って真因を追求し再発防止策を作成する。ケーススタディ資料には分析に必要な背景状況等に関する情報が不足しているかもしれない。そうした際には、前提条件を設定し、その上で分析を進める。（この際、そうした想定事項を記録しておくとよい）

なお、なぜなぜ分析は企業毎、組織毎に固有のやり方があることが多いため、固有のやり方がある場合はそれに従って実施する。

ケースＡ（後半部分）

【発生した事象】

■■■産業の試作を行う生産ラインで高橋主任と田中氏は、■■■産業の現場社員の立会の元、工程コントローラのソフトウェアモジュールの更新作業を行った。更新作業は順調に進み、加熱時間などを段階的に変えてサンプルを作成することになった。

初日のサンプル製造は何の問題もなく進捗したため、もう大丈夫だろうと思った高橋主任は以降の現場対応は田中さんに任せ自身は自社に戻った。

２日目もパラメータを変化させながらサンプル製造を続けたが、３日目から試作ラインの検査装置で不良と判断される試作品が散発的に製造される事象が発生するようになった。田中さんは工程機器の不良か、検査機器の問題だろうと思い、高橋主任にも特に報告もしなかった。

試作を開始してから１週間後に、■■■産業から「原因がよくわからないが試作品の不良が目立つ。このまま量産に持ってゆくのは不安であり、新製品の生産に支障が出るようだと、当社の事業インパクトが大きいので原因究明に●●システムズも協力してほしい」との一報が入った。

【内部で発生した事象】

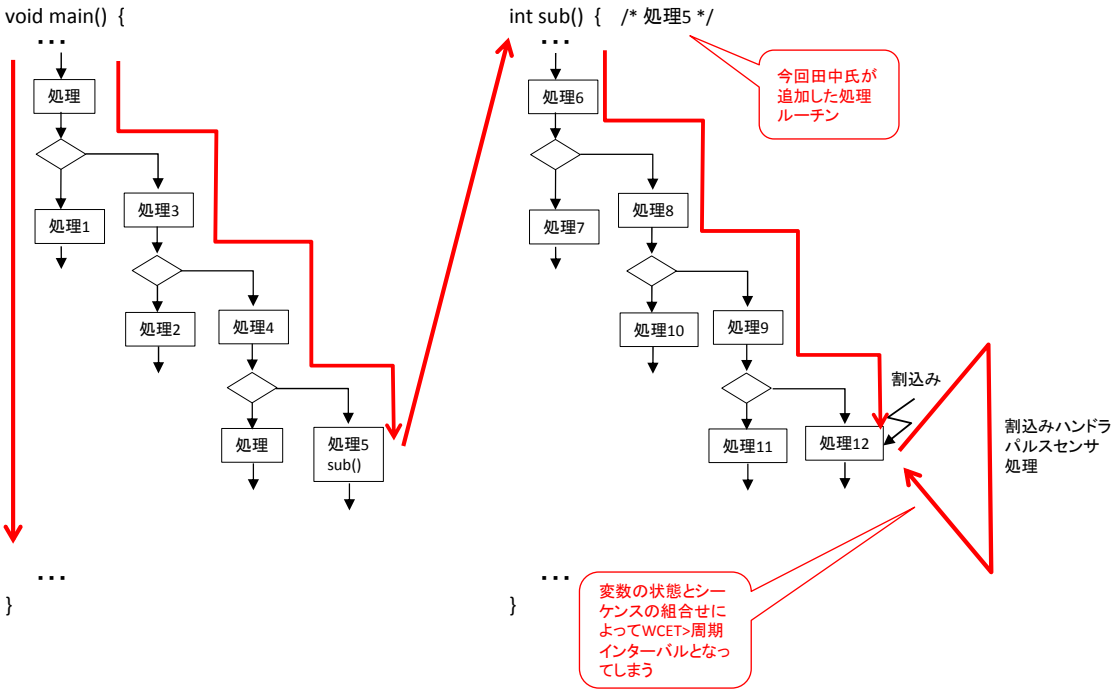
調査したところ、更新した工程コントローラのソフトウェアプログラムに問題があることがわかった。

元々のメインルーチンは 5ms の周回ループの中で複数の処理を行いながら溶剤処理・熱処理工程の処理をコントロールしているが、今回田中さんが追加した試作用ルーチンによって、熱処理時間の算出にズレが生じ、この結果当該工程の熱処理時間に影響が出たため、不良品が製造されてしまったことがわかった。

【原因となる要因】

・今回付加された試作対応のルーチンは、加熱等の制御パラメータ値を変えながらサンプル製造するがその組合せによっては通過するルーチン経路が長くなり、その延長で割込み処理を受けるとメインルーチンのタイムインターバル 5ms を超えてしまうケースが存在することがツールを用いた検証で判明した。

・インターバルを超えると熱処理に必要な PI 制御計算値が不適切になるが、目視による確認では容易に検知できないものであった。ツール活用は開発プロセスの中で明記されてもいなかった。



・田中さんは組込みシステムのリアルタイム処理の経験が少なく、5ms 周回動作を担保するために必要な最悪実行時間の重要性は頭では理解していたつもりだったが、実経験による注意深い考察を行うことに不慣れだった。

・本来この処理部分で割込みを受け付ける必要はなく、元の処理を参考にした際にこの割込み処理を排他する記述の追記をすべきであったが、高橋主任も見逃してしまった。

・今回改造開発プロセスのテーラリングでは負荷テストはしなくてよいと判断された。しかし今回問題となった事象を模擬試験環境で再現することは難しく、負荷テストをしていれば検出できた可能性があったもののそのチャンスを逃してしまった。

・ベテランの高橋主任であれば■■■産業から指摘される前のもっと早い段階で異変に気がついた可能性があったが、現場作業開始後の２日目から高橋主任が不在となったことが発見を遅らせてしまった。

【なぜなぜ分析例】

本例におけるなぜなぜ分析をサンプルとして図. 4-2 に例示した。図中の㋐ ･･･㋕は、真因を示している。

この例では分析をしやすくするための工夫として大きく要因を「作りこんだ要因」、「流出させた要因」、「影響拡大させた要因」の別にジャンル分けしている。

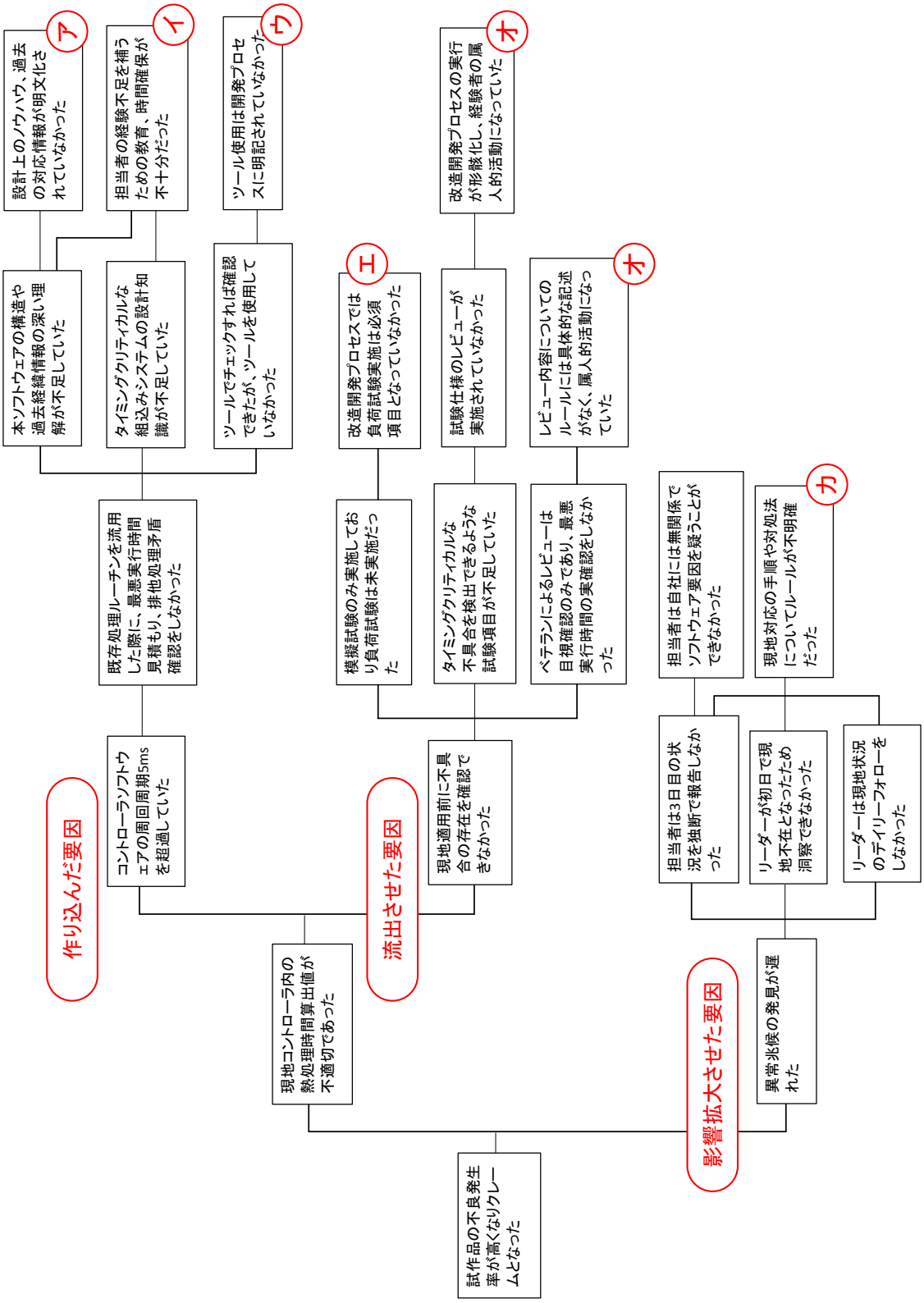

試作品の不良発生率が高くなりクレームとなった
影響拡大させた要因
現地コントローラ内の熱処理時間算出値が不適切であった
異常兆候の発見が遅れた
作り込んだ要因
流出させた要因
コントローラソフトウェアの周回周期5msを超過していた
既存処理ルーチンを流用した際に、最悪実行時間見積もり、排他処理矛盾確認をしなかった
本ソフトウェアの構造や過去経緯情報の深い理解が不足していた
設計上のノウハウ、過去の対応情報が明文化されていなかった
ア
タイミングクリティカルな組込みシステムの設計知識が不足していた
担当者の経験不足を補うための教育、時間確保が不十分だった
イ
ツールでチェックすれば確認できたが、ツールを使用していなかった
ツール使用は開発プロセスに明記されていなかった
ウ
現地適用前に不具合の存在を確認できなかった
模擬試験のみ実施しており負荷試験は未実施だった
改造開発プロセスでは負荷試験実施は必須項目となっていなかった
エ
タイミングクリティカルな不具合を検出できるような試験項目が不足していた
試験仕様のレビューが実施されていなかった
改造開発プロセスの実行が形骸化し、経験者の属人的活動になっていた
オ
ベテランによるレビューは目視確認のみであり、最悪実行時間の実確認をしなかった
レビュー内容についてのルールには具体的な記述がなく、属人的活動になっていた
オ
リーダーは現地状況のデイリーフォローをしなかった
リーダーが初日で現地不在となったため洞察できなかった
担当者は3日目の状況を独断で報告しなかった
担当者は自社には無関係でソフトウェア要因を疑うことができなかった
現地対応の手順や対処法についてルールが不明確だった
力

図. 4-2　なぜなぜ分析例

前掲のケースの各真因に対する再発防止策を以下に記した。

【直接原因への対策】

タイムインターバルを最悪でも超えることがないように処理を修正し、他の関連箇所についても同様の確認作業を実施。

【真因への恒久対策】

開発：

D1 メインループの処理時間がクリティカルなシステムでは、変数やその取りうる状態数とそれぞれの状況における機能動作処理時間にバラツキが出るため、開発プロセスのソフトウェアアーキテクチャ工程での検討項目に WCET（最悪実行時間：Worst- case execution time）見積もりを含める。(ア) (イ)

D2 単体テスト工程では解析ツールを利用した確認作業を必須とし、ソフトウェアアーキテクチャ設計のレビューにおいてもレビュアーによる目視確認の他、解析ツールによる動作結果の確認を行うこと。(ウ) (オ)

D3 改造開発プロセスにおいても、手を加えたことにより発現する潜在的な不具合がないか確認できるように、システムテストでは負荷試験を行うよう変更し、実施しない場合はその理由を明確にして承認を得るルールに変更する。(エ)

管理：

M1 経験の浅い実務者をアサインする際には、時間確保して現状を教える、不明点確認をするなどを行い、コーディング前の設計・変更設計レビューで必ずレビューし、結果エビデンスを残す。(イ)

M2 排他処理、等時性、参照透過性等組込み系システムで重要となる基本概念をケーススタディ、チーム定例ミーティングを通じ理解習得を促進させる。(ア) (イ)

M3 現地対応の作業手順、対応体制などを定めた現地作業要領書に、1)不測事態時の連絡体制、2)日次レポート事項、を追記し、作業前の最終レビューで内容確認の上、プロジェクト関係者に周知する (カ)

4.2.3 Step3：未然防止に向けた一般化

教訓とは 1.3 節の定義を再掲すると「製品ドメインに限定、特化することなく、当事者以外の人、他製品・システムにも役立たせることができる経験知識・ノウハウ」となる。作成された再発防止策に基づいて類似障害を未然に防止するための教訓を得ていくために、3.1.7 項で示すような一般化を行い、自社に適用することになる。教訓を導出するための確立されたプロセスというものは存在していないが、ここでは試みとして導出を容易にするためのガイドとして以下のような手順で例示した。

❶検討の観点

前掲のように再発防止策は障害を「作りこんだ要因」と「流出させた要因」の両面から案出されているので、下記のような観点に基づいて、何を教訓として抽出するかを自社・自部門の製品・システムで発生してきた要因、対策と照らし合わせながら考える（図. 4-3）。

・有用性･･･自社製品・技術に置き換えた場合にも有益な教訓内容が含まれているか
・類似性･･･自社で発生した事象や再発防止の対策に類似する内容か
・可搬性･･･異なる動作原理、技術であっても本質的な部分は置換できるか
・頻度･･･自社、自部門でも同様の事象、要因が多く発生しているか

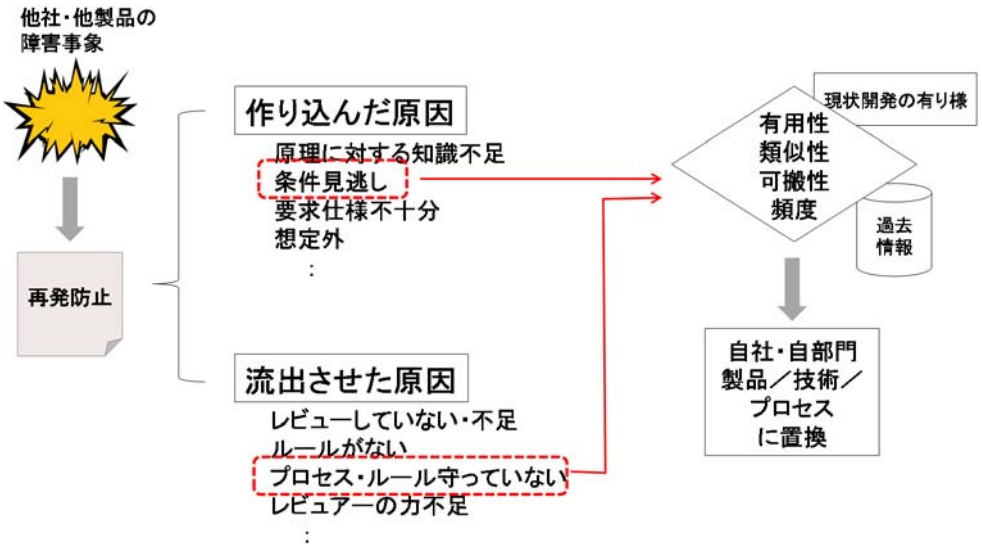


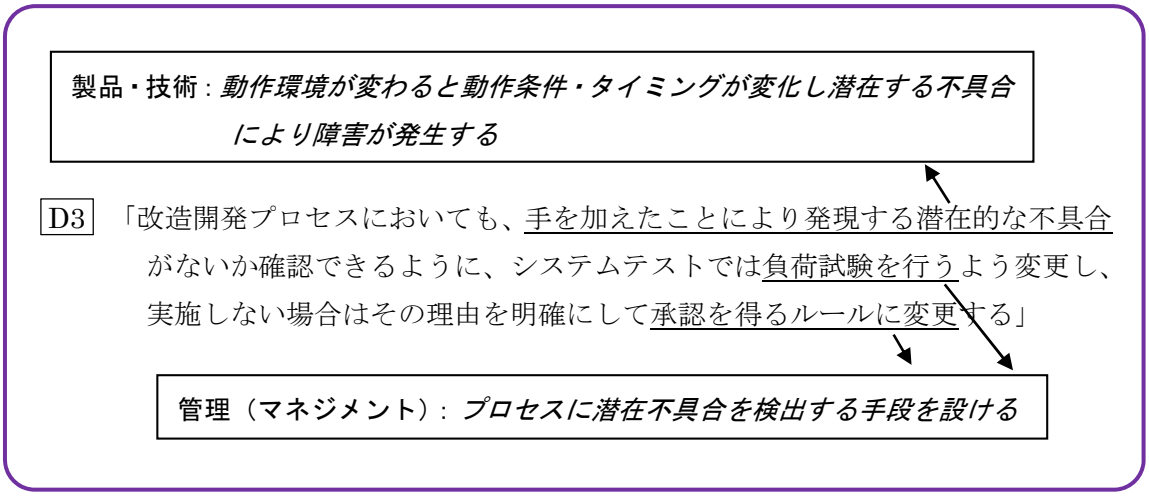
図. 4-3　抽出の観点

❷一般化

Step2 で作成した再発防止策の中から前記した観点に基づいて有益であると判断された事項を一般化する。

例えば本章で示したケース A の真因の一つに、「改良開発プロセスでは負荷試験が必須項目となっていなかった」(エ) がある。この真因に対する再発防止の恒久対策が D3 である。

ここで一般化する際の工夫として「製品・技術」と「管理（マネジメント)」という 2 つの側面を取り入れている。これは障害を「作りこんだ要因」と「流出させた要因」に対応したものでもあり、D3 に対して、次のような一般化（斜体部）がなされている。



これは前出の図 3-2 中の一般化された真因と一般化された真因の対策に該当する。これを自部門製品の経験に照らして置換し導出した教訓は例えば以下のようになる。

「製品・技術」：性能悪化に伴い不具合が誘発されることがある。そのような不具合が他にもないか、基本設計段階で洗い出す。

「管理（マネジメント)」：性能評価を実施しているか（特にシステム限界試験）をシステム詳細設計レビューの確認項目に含める。

❸気づきの与え方

次に作成された教訓を誰にどう伝えたらよいか、単なる精神論とならないよう具体的な手段を検討する。効果的な気づきを与えるためには、具体的に伝えたい対象者を想定の上、その対象者の技術や力量などを念頭に置いた言葉、表現手段を用いるなどの配慮が求められる。ここでは対象者を「実担当者」「プロジェクトマネージャ」と設定した。前掲の対策例 D3 から得られた教訓に関する気づきの与え方を下記に示した。

「実担当者」：開発プロジェクトマネージャ会議や個別開発デザインレビュー（基本設計 DR では必須）等の場で、ノウハウリストを確認する機会を設ける。

「プロジェクトマネージャ」：プロジェクト計画書作成時のチェックシートに、ノウハウリストを確認する時点（基本設計 DR 時）を、開発プロセス定義書中に追記する。

❹自部門・他部門への伝え方

最後に異なる技術を用いて製品を開発している自部門や他部門のエンジニアにこの教訓を伝えるとするならば、どのような伝え方をするかを考えてみる。ここでも実際のアクションにつながるよう処置を具体的に記すことが必要であり、そのためには社内で実際に行われている仕組みや会議、帳票の名称や用語を用いて行動に移せる内容とすることが大切である。以下は前例の場合の処置例を示したものである。

「部門標準のノウハウマスターに新たに追加されたノウハウについて、TQM 部会等で共有する」

4.3 実施する際の留意事項

本節では実際に行った体験型ワークショップの経験なども含め、企業の開発現場で実際に未然防止の教訓を導出する際の留意事項を記した。

【教訓の導出】

・概して一般化は難しいため、発想時に何かの”観点”を与えるとよい。本ガイドでは一般化の際に、「製品・技術」「管理（マネジメント）」の観点を設け、気づきを与えるというステップを取り入れるとともに、気づきを与える相手を特定するような工夫もしている。

【ファシリテーション】

・できるだけ同一テーマで演習を行い、チーム発表時にはチーム同士で質疑応答が行われるような問掛け（例えば、「そちらのチームでは、このような問題は話し合われましたか？」、「今のようなことは、そちらでご担当されている製品開発や技術にもありますか？」など）を行うなどのファシリテーションを行う。

・開発プロセスとの対応、対比を行うとイメージがつかみやすくなる。下図のような一般的なV字プロセスやESPR[6]で定義されているような開発プロセスを提示して、障害はどの工程で作りこまれたのか、どのプロセスにおけるリスクなのかなどを考えてもらうような働きかけを行う。

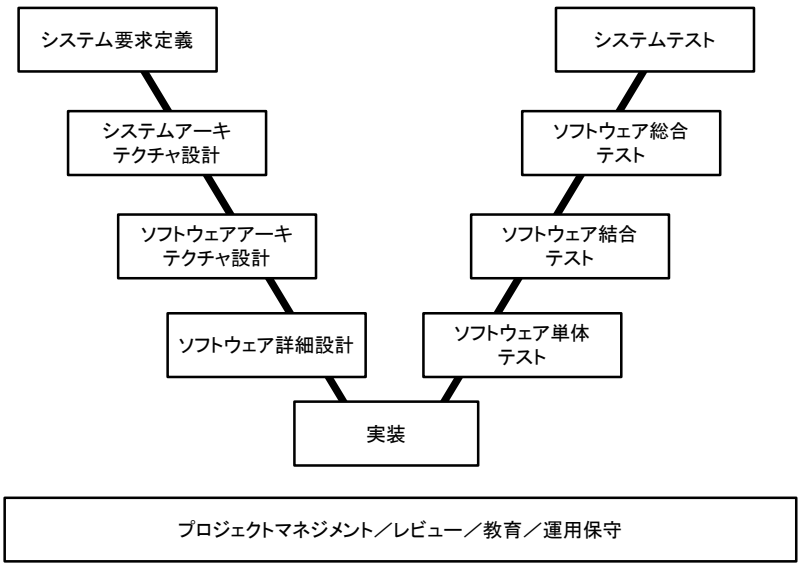


図. 4-4　開発プロセス例

・作業ステップごとに作業を進める際に、ある１チームだけ短時間で中間発表してもらい、

その内容を聞いた後で再びチーム作業を続行すると、その後のディスカッションが活性化することもあり、そうした工夫を臨機応変に取り入れると効果がある。

【用語・名称】

・社内で実施する際には、真因、再発防止策は社内用語や名称など、できるだけ具体的に表現するのがよい。

・ケーススタディ資料も、自社・自部門の実例、仕組みに基づいて準備する。開発プロセスや名称、帳票フォーマットなども社内専用のものを前提に置き換えるとよい。

【時間配分】

・集合研修にかけられる時間やどこに力点を置いた研修とするかによって、全体の作業時間配分、誘導などは変化をつけるとよい。障害時になぜなぜ分析を行う企業は多く、再発防止の作成までは比較的容易にできる反面、未然防止の教訓を考えることは不慣れな場合が多いことを考慮すると、要因分析までは模範解答を早めに提示し教訓づくりに多めに時間を割くなどの時間コントロールをするのが望ましいこともある。

4.4 ワークショップ体験者の声

実際に体験型ワークショップに参加された受講者の感想を以下に抜粋した。集合研修では各社の具体的な製品、技術で進めることは難しいが、一定の評価を得ることは出来たものと思われる。また再発防止策から未然防止の教訓を得ていく上では、自社の製品技術、開発のあり方に即した効果的な誘導を行うことがポイントとなると思われる。

・実例を想定した演習内容であり、ディスカッションをおこない有意義だった。
・障害発生リスクの考え方を学ぶことができたので、社内展開したい。
・一方的に聞くセミナーではなく、演習の中でサンプルケースを見て作業を行うことにより、自社に置き換えて考える事ができた。
・再発防止策から未然防止策を導き、他者へ伝えるのは工夫が必要だと感じた。
・自社の問題を考えるきっかけとなった。各製品の環境に合った方法を考えていくことが大切である。

5 未然防止に向けた企業内事例

5.1 ケーススタディによる教育事例

本事例は、実際に自社で発生した失敗事例を題材としたケーススタディ資料を使い、KYT の要領でリスク検討を行う教育である。教育形式は、学習者 4～5 名程度を 1 グループとし、全体を 5 グループ程度に分けて実施するワークショップである。グループ内あるいはグループ間でリスクを議論・共有することで、学習者が特定するリスクの多様性の広がりや理解の深まりを狙っている。

始めに座学でリスクマネジメントの必要性、リスクの定義や基本的な特定方法、リスクの分類例などを学習した後、個人演習とグループ演習を行う。最後にケーススタディ資料の題材となった失敗事例を紹介し、マネジメントやソフトウェアエンジニアリング上のポイントを解説する。

個人演習では、学習者が個々にケーススタディ資料を読んでリスクを特定する。次のグループ演習では、個々に特定したリスクをグループ内で共有し、そこから 1 つリスクを選択して対応策を議論する。選択したリスクおよびその対応策をグループ発表し、グループ間で共有する。

分量としては、1 つのケーススタディ資料を使った教育時間が半日のコースである。グループ演習では、グループ内あるいはグループ間の議論が活性化するよう促す講師のファシリテーションをポイントとしている。

章	学習項目	内容
1	はじめに	なぜリスクマネジメントが必要か？
2	リスクって何でしょう？	リスクとは
		リスク抽出方法
		リスク抽出以外のプロセス
		リスクの分類例
3	ケーススタディ	事例からのリスク抽出
4	実際の事例紹介	失敗事例，リスク抽出例，事例に関連するソフトウェア・エンジニアリング技法

図. 5-1　ケーススタディ教育項目

失敗事例からのケーススタディ資料の作成では、事例の抽象化・一般化とともに、次の 5 項目をポイントとしている。

- プロジェクトの進行に合わせたショートショート形式の読み物とする
- 物語はプロジェクト開始～プロジェクト終了（製品リリース）までとする
- プロジェクトのプロフィール（組織やチームの役割、開発対象製品、組織やチーム/プロジェクトの位置付けなどは、物語の冒頭で説明する
- プロジェクトの進行に従って要因作業の説明、モノローグでマネージャや開発者が該当作業に対する判断や対応を記載する
- 物語には結末は記載しない

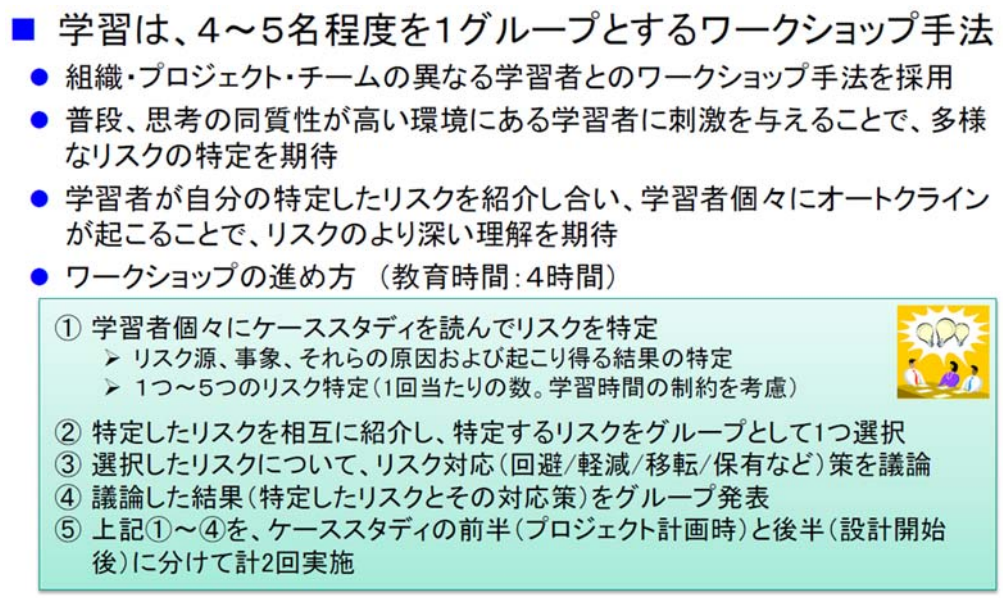


図. 5-2　ケーススタディ教育のポイント

5.2 不具合モード発想の仕組み化事例

本事例は、過去の障害事例の経験を未然防止の知見として体系的に蓄え、その情報に基づいて不具合モードの発想力を高める取組みを行っている事例である。図. 5-3 は設計時に不具合の作り込みを防ぐために導入されている未然防止プロセスである。「観点リスト」や「未然防止リスト」等を活用し、どのような不具合モードがあるかを発想する。

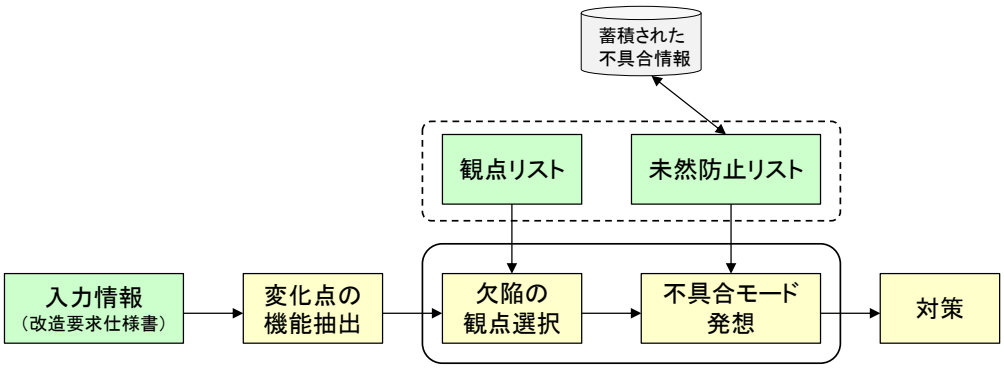


図. 5-3　未然防止プロセス

この「観点リスト」は図. 5-4 のようなマインドマップ形式で整理したもので、不具合を発生させるメカニズムを従来の経験に基づいて体系化している。

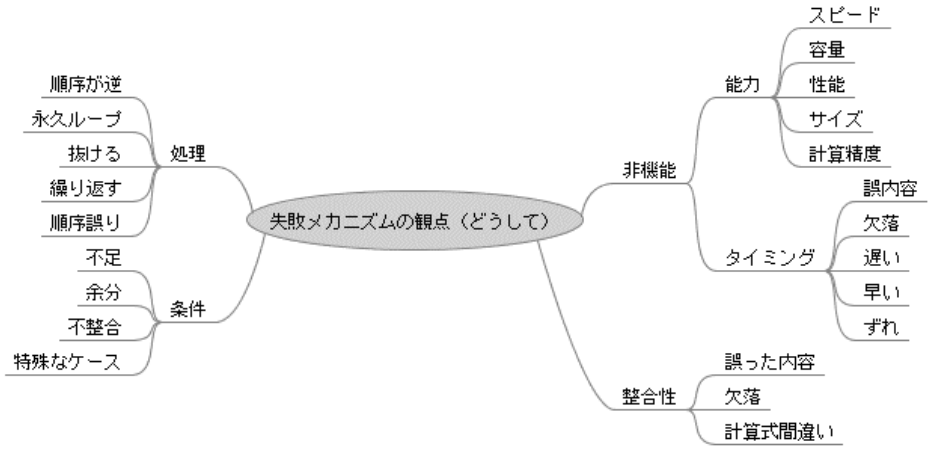


図. 5-4　観点リスト（抜粋）

また「未然防止リスト」は過去に発生した不具合事例を１件１葉で記録している情報デ

ータ群である。

これらのリストを用い、設計の上流工程でどのように不具合モードを発想するかを以下に示す。ここでは、ある対象機能における不具合について「何が」「どうして」「どうなる」のかを考えるが、この際、図. 5-5 のように発想のトリガとなる観点を前掲した観点リストの中から選択・抽出し、対象機能やその不具合と関係のある要素を対応づける。

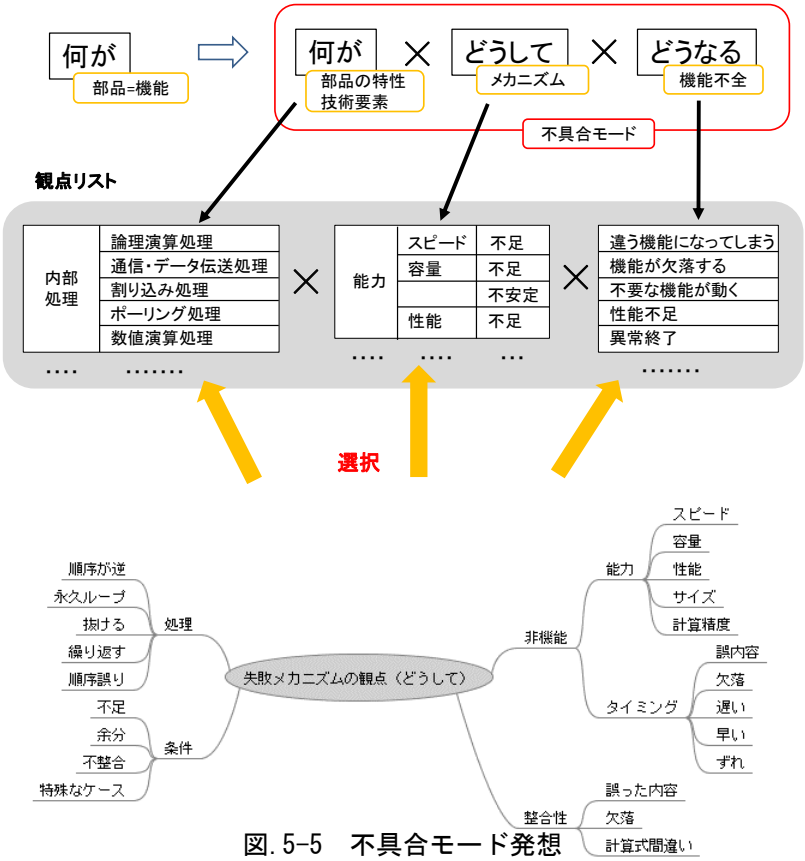


図. 5-5　不具合モード発想

前掲図の例では「何が」には「各種内部処理」が、「どうして」には「各種能力の不足や不安定さ」が、「どうなる」には「機能が欠落する」といった観点要素が抽出され対応づけられている。この観点に基づいて過去に経験した同様の不具合例を未然防止リストから選択し、同じような不具合が発生しないかを考えるが、この際「どうして」「どうなる」のそれぞれにおいて発想の起点を設けることにより、実施者の発想が大きくブレないようにすることができる。以下の図. 5-6a、b にそうした発想起点の例を示した。

（1）不具合モード発想手順

・技術要素・環境起点（何が）
⇒失敗メカニズム（どうして）⇒機能不全（どうなる）

①日付・変換処理（何が） ⇒ 整合性・誤った内容（どうして）
『確認日を西暦年4桁を2桁として扱い、
未確認期間を正しく計算できず、
集計除外とすべき日を間違える』

②リトライ処理（何が） ⇒ 処理・抜ける（どうして）
『データ読取りエラーのリトライ読取り時、
適用除外期間チェック処理が抜け、
期間Nであっても除外にならない』

図. 5-6a　不具合モード発想例

（2）不具合モード発想手順

・機能不全起点（どうなる） ⇒ 何が ⇒ どうして

①アラーム出力が出ない
内部I/F ⇒ 参照先・不整合
『上限チェック有効／無効パラメータの
参照先アドレスを間違えて無効となり
アラームが出力されない』

②システムが停止してしまう
制御量計算 ⇒ 条件・不足
『位置計算結果が負となる条件を
考慮していなかったため
処理がアボートし停止してしまう』

図. 5-6b　不具合モード発想例

参考文献

[1]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（平成 28 年 3 月）

[2]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（平成 28 年 3 月）

[3]野中郁次郎、知識創造企業、東洋経済新聞社、1996 年

[4]内平直志（2010). 研究開発プロジェクトマネジメントの知識継承. Ph.D. Thesis, 北陸先端科学技術大学院大学

[5]畑村洋太郎（2000). 『失敗学のすすめ』. 講談社.

[6] ESPR :Embedded System development Process Reference
（SECBOOKS : ESPR Ver.2.0 :【改訂版】組込みソフトウェア向け　開発プロセスガイド）
http://www.ipa.go.jp/sec/publish/tn07-005.html

[7] ESMR :Embedded System development Management Reference
（SECBOOKS : ESMR Ver.1.0 : 組込みソフトウェア向けプロジェクトマネジメントガイド[計画書編]）
http://www.ipa.go.jp/sec/publish/tn05-010.html

[8] ESDR :Embedded System development Design Reference
（SECBOOKS : 組込みソフトウェア向け設計ガイド［事例編］）
http://www.ipa.go.jp/sec/publish/tn12-003.html

[9] ESTR :Embedded system development Testing Reference
（SECBOOKS : 組込みソフトウェア開発における品質向上の勧め［テスト編～事例集～］）
http://www.ipa.go.jp/sec/publish/tn12-004.html

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	三菱電機メカトロニクスソフトウエア株式会社
	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
	土山　欽也	日本電気株式会社
	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
	松田　充弘	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
	石田　茂	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