システムグループ

重要インフラ等システム障害対策（製品・制御システム）

SEC 調査役 三原　幸博　SEC 研究員 松田　充弘　SEC 研究員 石田　茂　SEC 調査役 十山　圭介

交通機関や電気・水道の制御等、製品に組み込まれた機器の制御を行う「製品・制御システム」（組込みシステム）の障害情報の収集・分析と対策の検討を行い、その結果を普遍化した「教訓」として取りまとめ、「情報処理システム高信頼化教訓集（製品・制御システム編）」として公開した。

1 製品・制御システム分野におけるソフトウェア障害情報の収集・分析

近年、コンピュータを利用して制御や機能実現を図る機器や製品（以下、製品・制御システム）が増加している。これらの製品・制御システムの中には社会生活のインフラとして重要な役割を担うものも多く、それらには高い信頼性が求められる場合が少なくない。しかし、製品・制御システムは、実現する機能規模が肥大化すると共に複合化する傾向にあり、システム全体として信頼性を確保するための技術面での工夫や運用管理での工夫が求められている。

一方、グローバル化に伴う企業間競争の激化により、低価格・短納期の製品開発が主流となり、システム高信頼化のための技術やノウハウの伝承がうまく行われないといった問題も顕在化している。

システム高信頼化に関する技術やノウハウは個々の企業の中で自らの経験から習得した技術資産として認識されてはいるが、知としての共有と活用は必ずしも十分であるとは言い難い。近年、複雑化が進む製品・制御システムにあって、その信頼性やユーザにとっての安全性を考えた場合、システム高信頼化に関する技術やノウハウを企業・業界を越えて共有し活用しようとする意識の醸成とそれを可能にする仕組みの構築が重要になっている。

製品・制御システムのシステム信頼性に関する上記現状を鑑み、産業界におけるシステム高信頼の知見を集積し、将来に向けたシステム信頼性向上のための技術的な布石を打ち、その結果としてシステム信頼性に関する社会的な認識レベルを上げていくことを目的に、「製品・制御システム信頼性向上部会」とその傘下に２つの WG を設置し、産業界の有識者を交えた議論を進めた。（図１）

（1）未然防止知識 WG：製品・制御システムの障害を未然に防止するためのノウハウや知見を分析収集し、産業界で共通に利用できる資料を作成する。

（2）障害事例検証 WG：製品・制御システムに関するシステム障害に関する事例研究を通して、システム障害発生時の対処法や障害要因分析の手法を整理する。

2 障害対策事例の収集と教訓集・対策事例集作成

2.1 背景と狙い

産業界で実践されているシステムの品質上の問題を未然に防ぐための知識をもとに、製品・制御システムの障害を抽象化、一般化することによって、組込みシステム開発企業において幅広く利活用できるための『智恵』として教訓集を作成した※1。併せて教訓を実践するための対策の事例をまとめた。

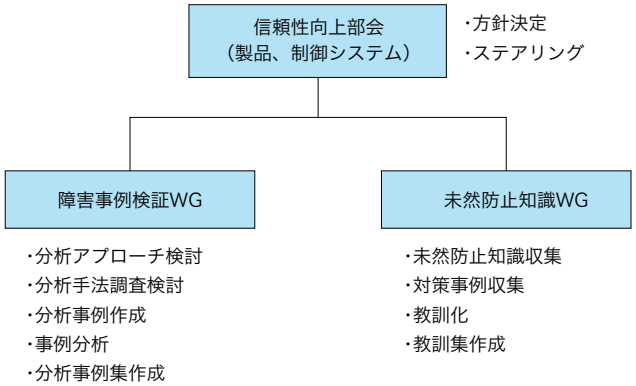

図１　製品・制御システムに関する部会・WG 構成

【脚注】

※１　http://www.ipa.go.jp/sec/reports/20140513_2.html

2.2 事例収集の方針

（1）概要

未然防止のための知識とは、他分野、他企業、他組織が起こした事故事例を抽象化、一般化することによって得られる、自分野、自組織において類似の問題を防ぐための知識である。また、未然防止のための知識には、内在的な障害を取り除くための品質向上のための知識と、内在的な障害が発現したとしても問題として外化するのを防ぐフォールトトレランスのための知識があるが、両者とも収集の対象とした。

表１　教訓の分類軸の候補

NO	分類軸候補
1	並行システム
2	複雑・組み合わせ多い
3	COTS 利用
4	ハードウェア利用
5	フォールトトレランス
6	設計思想
7	ヒューマンファクタ

表２　教訓一覧と対策が必要な工程との対応例

教訓番号	教訓タイトル	システム要求定義	システムアーキテクチャ設計	ソフトウェアアーキテクチャ設計	ソフトウェア詳細設計（変更設計）	実装（コーディング）	レビュー	システムテスト	教育	プロジェクトマネジメント	運用
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			○	○						
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、関連する条件をすべて洗い出して整理し不整合がないことを確認する			○	○						
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				○			○			
4	変数値域が広く、組み合わ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				○						
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		○	○			○	○	○		
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	○	○							○	○
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		○								
8	想定可能な例外を形式的に漏れなく分析する	○	○								
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			○							
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		○		○		○		○		
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			○		○			○		
12	歩留りのある製品の良品／不良品を検査する装置では、すべてが良品あるいは、不良品との検査結果は異常と判断すべきである	○									○
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			○	○			○	○	○	
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 ・時間帯による負荷変動について考慮する	○	○	○			○				
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				○			○			
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			○							
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				○						
18	ログファイルの断片化に注意する			○							

（2）想定利用者

未然防止知識の利用者としては、組込みシステム開発にかかわる以下の３者を想定している。

- ソフトウェア設計者
- ベンダ側システム設計者
- ユーザ側システム設計者

なお、システムを利用する特別な訓練を受けていないエンドユーザは、知識の利用が困難であると考え、想定利用者からは除外した。

（3）収集整理手順

組込みシステムの製品分野は多岐にわたるため、同一の企業内であっても他の分野の事例や知見では実感しにくいため、なるべく利用者の分野に近い事例に書き換える必要がある。そのため、下記の手順に従って、収集した事例や知見から肝となる部分を抽出し、別の事例に書き換えたものを知識として整理した。

step1：所定のシートでの情報提供／ヒアリング
step2：抽象化、他分野、別事例への書き換え
step3：対策の整理
step4：有識者によるレビューと補足

2.3 教訓の分類

（1）分類の観点

教訓の分類に関しては、教訓の数が未だ少ないこともあり、今回は、適用可能な開発プロセスで分類した。表１に示す分類軸の候補を含め。更なる分類方法を継続して検討していく。

（2）教訓と開発工程による分類

開発工程においてどのような対策を施せば、教訓が解決しようとしている問題を未然防止できるのかという観点から整理した。プロセスモデルは ESPR Ver.2.0:「組込みソフトウェア向け開発プロセスガイド」[1] のモデルを採用した。また、直接開発にかかわるシステム・エンジニアリング・プロセス（SYP）やソフトウェア・エンジニアリング・プロセス（SWP）だけでなく、サポート・プロセス（SUP）も対象とし、ESPR では定義されていない教育に関する工程も工程別未然防止知識の一部として採用した。また、組込みシステム開発において多く見られる差分開発特有の未然防止知識についても、切り出して記載した。

教訓と対策が必要な工程との対応例を表２に示す。

3 障害分析手法並びに分析事例集

製品・制御システムに関するシステム障害の未然防止には、障害を分析して直接原因を探り出して対策するだけでなく、根本原因の分析と（恒久）対策の立案が重要である。事例研究を通して、システム障害発生時の対処手順と障害要因分析の方法を整理し、手法の解説と事例を併せて公開した。

3.1 障害発生から分析・対策立案までの流れ

障害が発生した際に行う活動を図２に示す。次節で各活動の詳細について述べる。

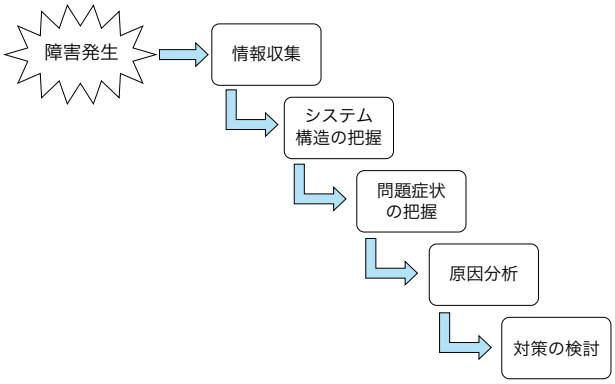

図２　障害分析・対策立案の流れ

3.2 分析の各ステップ詳細

（１）情報収集

障害の分析に際してまず行うのが情報収集である。障害の分析に必要な情報の収集は事態の収拾後速やかに行い、収集した情報は整理して文書にまとめる。システムの種類によって収集が必要な情報は異なり、収集できる情報が限られる場合もあるが、必要と思われる情報をできるだけ集めることが、以降の分析を円滑に進めるために肝要である。

（２）全体を把握する（システム構造の把握）

収集した情報を、障害発生の経緯に関する情報と、システムの動作・構造に関する情報に分類し、整理して対応付けることで、障害に至った問題症状の把握が可能になる。

とくにシステムの動作・構造に関する情報を整理して、システムの全体像を見渡せるような簡潔なシステム構造図を作成することで、サブシステム間の相互作用の関係が明確になる等、システムの構造と振る舞いの把握に繋がる。

（３）問題症状の把握（事象経過）

次に、時系列と相互作用を意識しながら事象を整理して問題症状を把握する。まず、ヒアリングなどで得た事象に関する情報を時系列に整理するとよい。このとき、観点を定めて事象を整理すると分析しやすくなる。また、（２）で作成したシステムの構造図を利用して、収集した情報に矛盾する点やあいまいな点がないことを確認する。

最後に、障害にかかわった人やシステムの構成要素の間の相互作用が分りやすくなるように事象を整理して一覧性の高い形式で図表にまとめるとよい。

（４）原因分析

原因分析は、以下の３つのステップで構成される。

①　原因個所の推定

問題症状を引き起こしたと考えられるシステム上の原因個所を推定する際にソフトウェア、ハードウェア、人、環境・想定条件の観点から原因を推定すると漏れや抜けを防ぎ易くなる。原因分析の担当者の経験なども貴重な情報源である。情報を得ることが難しい場合には、一定の仮定を置いて原因個所を推定する必要があるが、事実と区別できるように必ず記録する。

②　直接原因の特定

推定結果に基づいて直接原因を特定するために再試験によって障害の再現条件を調査するなどの作業が必要となる。ソフトウェア部分に問題があると推定される場合には、コードレビューやインスペクションを実施するとよい。

③　根本原因の特定

多くの場合は設計誤りや人為的ミスが直接原因となるが、そのような誤りやミスが行われた要因や見逃された要因を複数の視点に立って根本原因を分析するとよい。ここでは、開発担当者へのヒアリング結果等が貴重な情報源となる。

（５）対策の検討

特定された根本原因を取り除く対策を検討するが、すべての原因を取り除けない場合には影響を軽減する対策を検討する。開発プロセスや体制に関する現状や制約を踏まえた上で短期的 / 長期的対策を考えるとよい。

技術的課題か管理的な課題であるのかなど、複数の観点から検討することが重要である。

3.3　分析手法と分析事例

有用な分析手法のうち、調査した企業 の現場で利用されている８つの手法について、公開されている過去の障害事例に適用し、分析事例として整理した。その結果は、

「情報処理システム高信頼化教訓集（製品・制御システム編）」と併せて公開した。

[分析手法]

1　ブロック図
2　事故経過表
3　VTA（Variation Tree Analysis）
4　問題行動分析
5　PNA（プロセスネットワーク分析法）
6　発生源・検出漏れ分析
7　例外分析
8　なぜなぜ分析

[障害事例]

1　湘南モノレール事故
2　駒場ダム事故
3　アリアン５事故
4　カンタス航空事故

4 今後の課題

現場での教訓集の適用を促進するためのセミナー等の開催と、活用を意識した質・量両面からのブラッシュアップを進めていく。また、現場からの評価の収集にも努め、今後の活動にフィードバックしていく。

【参考文献】
[1] Embedded System Development Process Reference