

ソフトウェア開発の現状

Summer Seminar in Operations Research
2017/09/07

SRATECH Lab 株式会社
代表取締役社長
広島大学工学部 特任教授

組込系ソフトウェアの規模について

◆ システムにおけるソフトウェア

- F-22 ラプター: 1.7 M LOC
- F-35 ジョイント・ストライク・ファイター: 5.7 M LOC
- ボーイング 787 ドリームライナー: 6.5 M LOC
- エアバス A380: 20 M LOC
- 高級車: 100 M LOC (70~100 ECU)



ソフトウェアに関する問題事例

製品開発に対する安全要求

ソフトウェアにおける安全要求の実装

ソフトウェア安全状態の分析方法

ソフトウェアに対するセキュリティ要求

問題事例：携帯電話

◆ 発売日

2009年12月18日

⇒ 販売一時停止(2010年1月27日発表)

◆ 原因

内臓ソフトに一部不具合を確認

⇒ 緊急通報電話番号「110」「118」「119」等へ
接続ができない

「184」や「186」を付加した場合、正常に接続

※ 関係者:3桁以下の番号認識のテスト漏れ
による現象では！



テストケース設計：**境界値分析が不十分**



docomo STYLE series L-02B

出典:japan.internet.com

問題事例：自動車

◆ 2010/05:レクサス「LS」でリコール

- 原因:車速に応じてステアリングとギア比を変化させる電子制御に不具合
ハンドルとタイヤの動きが一時的に連動しなくなるトラブル

◆ 2012/10:世界で**743万台**をリコール

- 14車種:ヴィッツ、ベルタ、ラクティス、イスト、オーリス、ルミオン等
- 原因:パワーウィンドスイッチの不具合
 - ▶ 部品提供会社の負担金:**170億円以上**

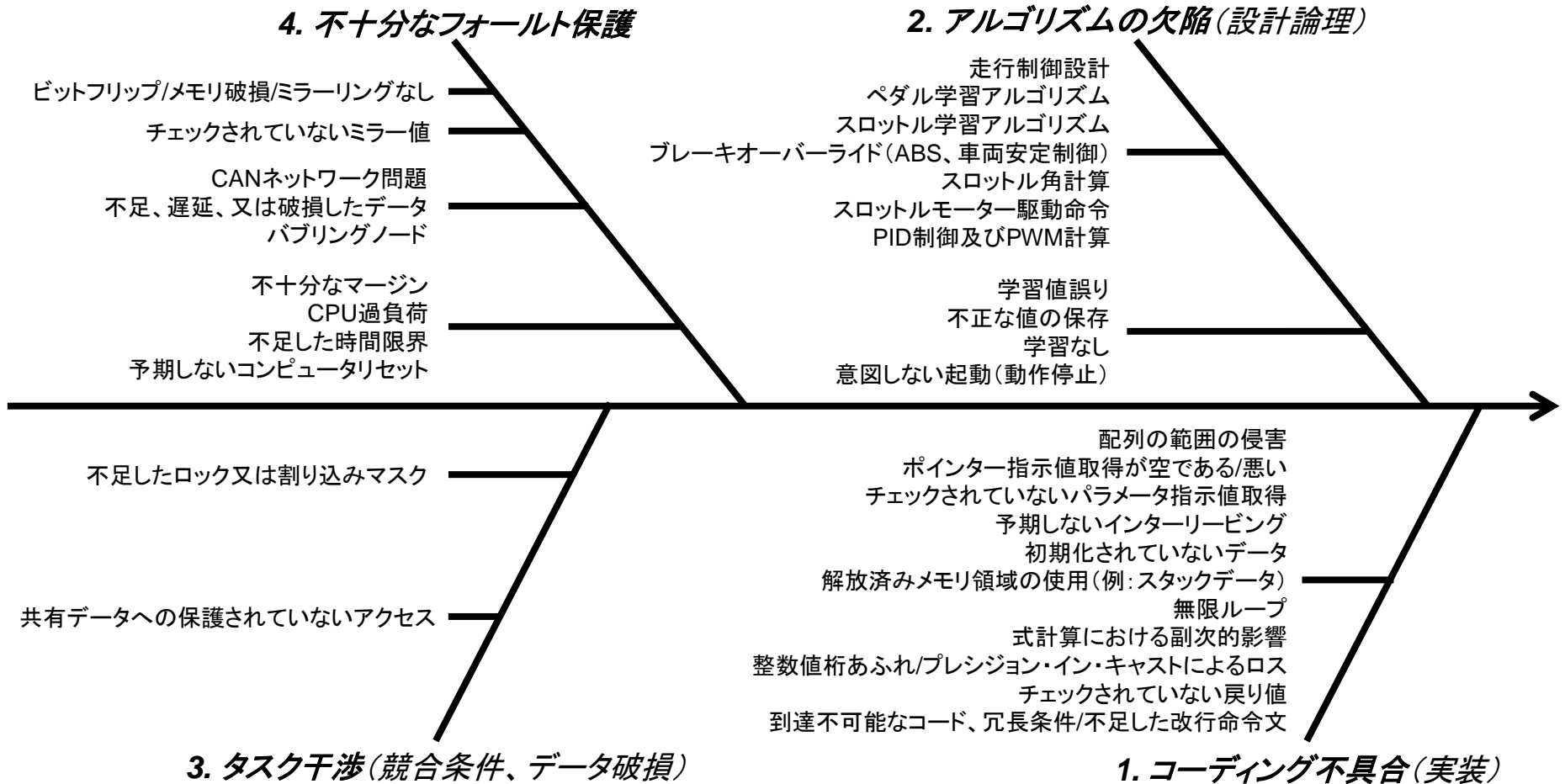
◆ 2014/04:世界で**600万台以上**をリコール

- 27車種:ポルテ、カローラ系、オーリス等、アイドリングストップ装着車
- 原因:始動装置のスタータ駆動用リレーの通電設定が不適切(最悪:火災)
 - ▶ トヨタとして**約700億円の減益**

◆ 他社車でも多数のリコール

問題事例：自動車(意図しない急加速)

※ 下記の石川ダイアグラム(フィッシュボーン、特性要因図)は、米国運輸省国家道路交通安全局「トヨタ車の意図しない急加速(UA)に関する調査報告書」より引用



意図しない急加速：見込まれるソフトウェア原因の特性要因図

ソフトウェアに関する問題事例

製品開発に対する安全要求

ソフトウェアにおける安全要求の実装

ソフトウェア安全状態の分析方法

ソフトウェアに対するセキュリティ要求

安全(Safety)の定義

- ◆ ISO/IEC Guide 51 (JIS Z 8051)
 - 受容できないリスクから免れていること
 - 原文: "Freedom from unacceptable risk"
- ◆ IEC 61508-4 (JIS C 0508-4)
 - 受容できないリスクから免れていること
 - 原文: "Freedom from unacceptable risk"

上記の定義は共に「リスクゼロの状態」を意味しているのではない
「受容できないリスク以外のリスクがあっても安全である」という意味である

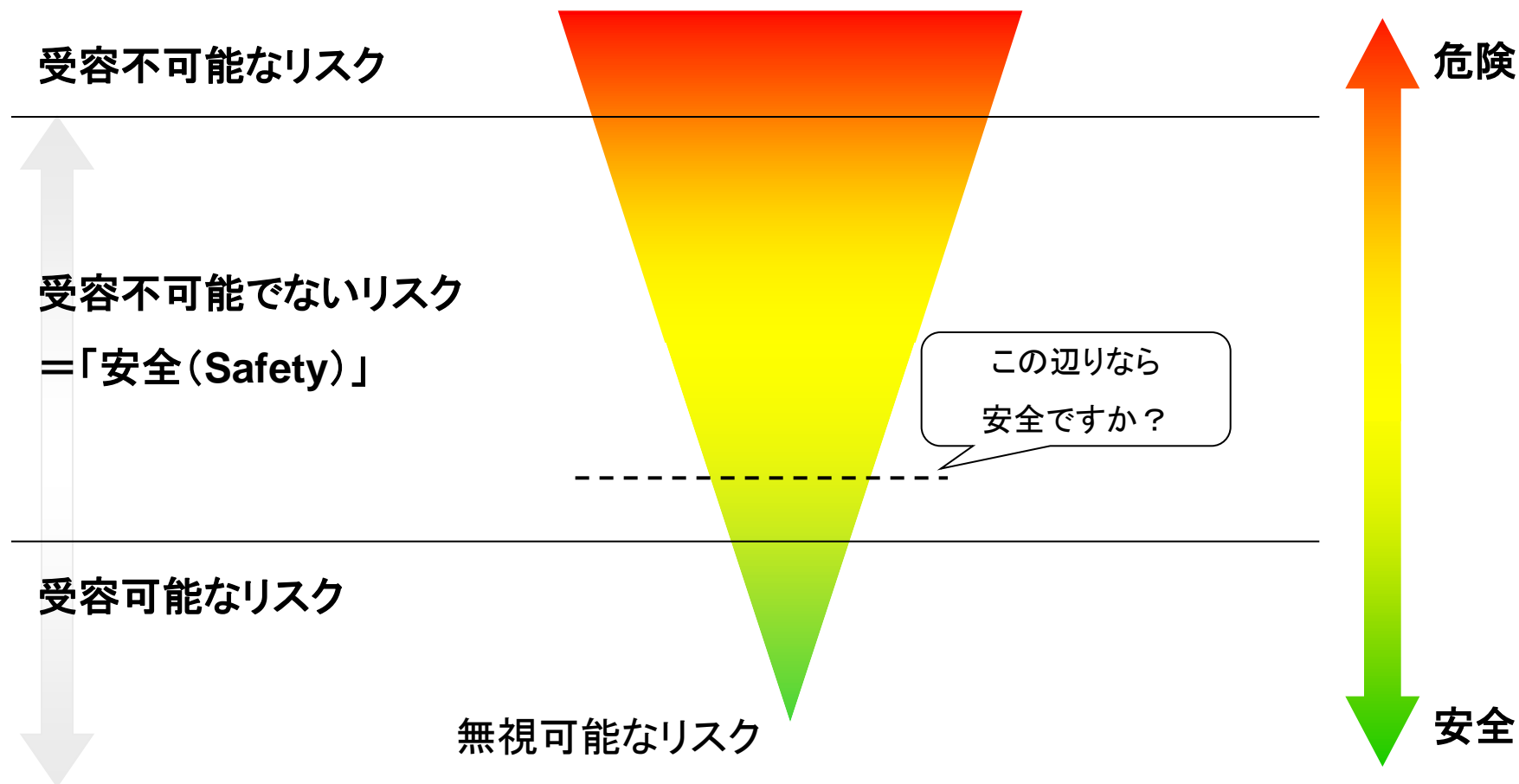


つまり...

「リスクが受容可能な状態まで抑えられている状態」を安全と定義している
「リスク」については数量的な概念を用いてコントロールしていく

リスク/危険とは？

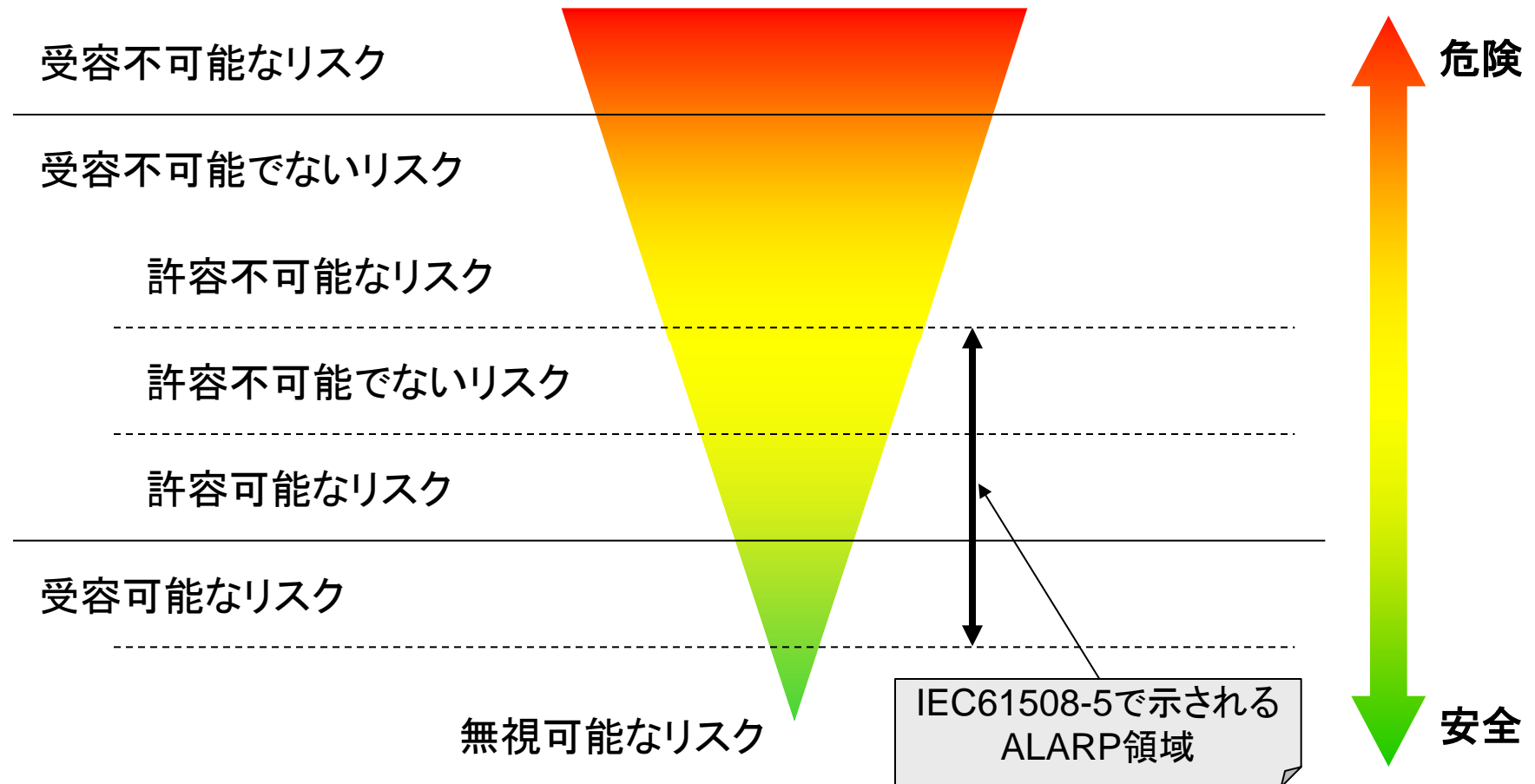
◆ どの程度のリスクなら「安全」か？



ALARPとは？

◆ ALARP: As Low As Reasonably Practicable

- 合理的に実施可能な限りリスクを下げる



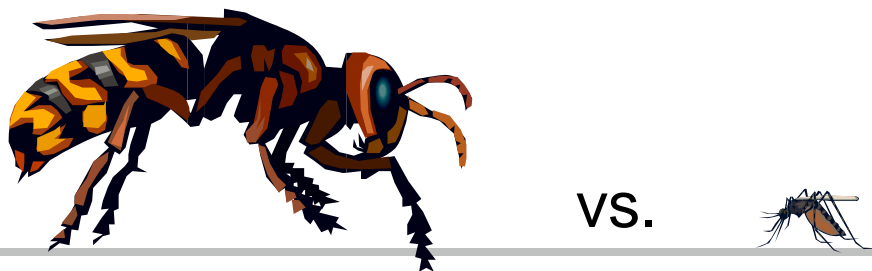
許容可能なリスクと受容可能なリスク

◆ 許容可能なリスク

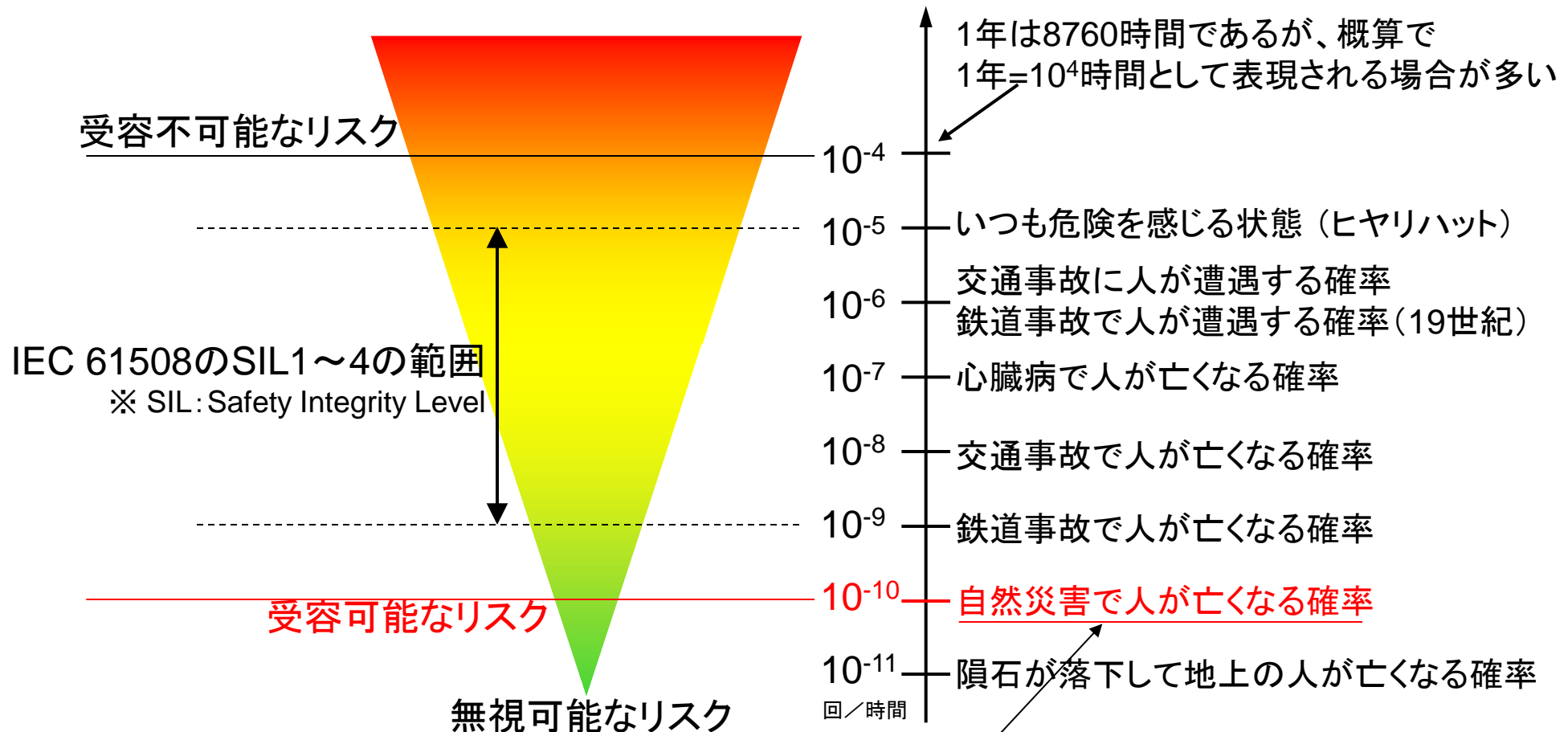
- 特定の条件下であれば、曝されても大きな問題とならない程度のリスク
 - ▶ 特定の条件となるまで、十分なリスク低減が必要
- 例：
 - ▶ 一匹の蜂に刺されたが、屈強な体であるため少し腫れただけで済んだ
 - ▶ 蜂が一匹である、体が屈強であるといった特定の条件下でのみ許容される

◆ 受容可能なリスク

- 一般に誰がどのような条件で曝されても大きな問題とならない程度のリスク
- 例：
 - ▶ 一匹の蚊に刺されたが、痒いだけで済んだ
 - ▶ 大きな問題にならないが伝染病などに感染するかもしれないというリスクは残る

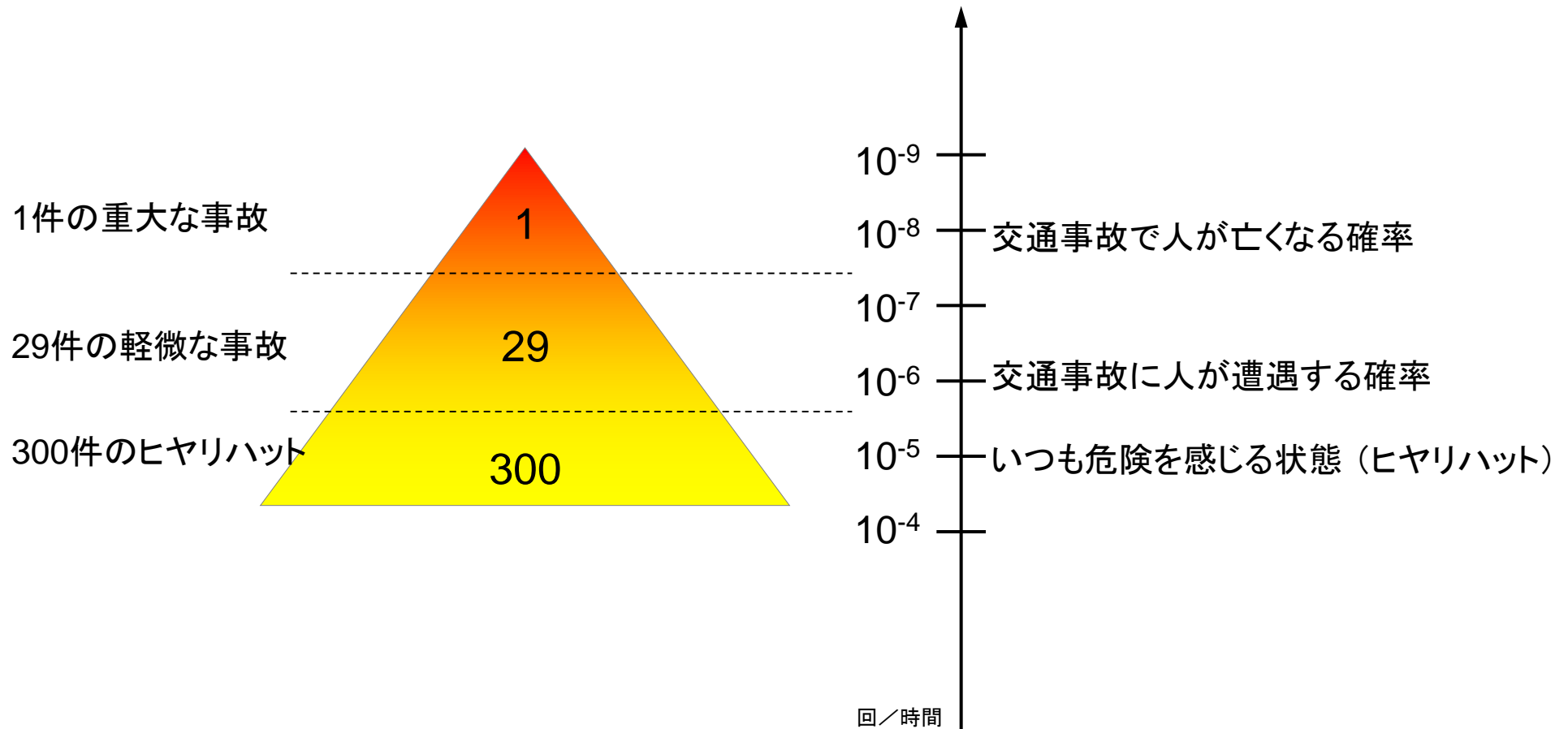


事故発生確率



自然災害による死亡確率が受容可能なレベルの基準になった(誕生秘話)

参考：ハインリッヒの法則との比較



交通死亡事故に至る確率も大凡ハインリッヒの法則と一致する

日米欧の安全文化の違い

日本の考え方	欧米の考え方
<ul style="list-style-type: none">・ 災害は、努力すれば2度と起こらないようにできる	<ul style="list-style-type: none">・ 災害は、努力しても、技術レベルに合わせて必ず起こる
<ul style="list-style-type: none">・ 災害の主原因は人である・ 技術対策よりも人の対策	<ul style="list-style-type: none">・ 災害防止は技術の問題・ 人の対策よりも技術対策
<ul style="list-style-type: none">・ 管理体制、教育訓練と規制の強化で安全を確保	<ul style="list-style-type: none">・ 人は必ず間違いを犯す・ 技術力向上がなければだめ
<ul style="list-style-type: none">・ 安全衛生法で対人および設備の安全化を目指す・ 災害が発生するたびに規制を強化	<ul style="list-style-type: none">・ 設備の安全化とともに、事故が起きても重大災害にならない技術を開発・ 災害低減化に関する技術力向上の努力
<ul style="list-style-type: none">・ 安全はただである	<ul style="list-style-type: none">・ 安全はコストがかかる
<ul style="list-style-type: none">・ 安全にコストを掛けにくい・ 目に見える具体的危険には最低限のコストで対応	<ul style="list-style-type: none">・ 安全にコストを掛ける・ 危険源を洗い出し、リスクを評価し、評価に応じたコストを掛ける
<ul style="list-style-type: none">・ 見つけた危険をなくす技術	<ul style="list-style-type: none">・ 論理的に安全を立証する技術
<ul style="list-style-type: none">・ 頻度(発生件数)重視	<ul style="list-style-type: none">・ 重大度(重大災害)重視

ソフトウェアに関する問題事例

製品開発に対する安全要求

ソフトウェアにおける安全要求の実装

ソフトウェア安全状態の分析方法

ソフトウェアに対するセキュリティ要求

ソフトウェアにおける分離設計の達成とは？

◆ 独立性/非干渉性を達成していることの論証は

■ 空間域

- ▶ あるモジュールが使用するデータを、**別モジュールが変更していないか？**
- ▶ 特に、**非安全関連モジュールが変更していないか？**

■ 時間域

- ▶ あるモジュールは、
 - 利用可能な**CPU実行時間を占有し過ぎること**
 - ある種の**共有資源をロック**して、別モジュールの実行を阻むこと
- により、別モジュールが正確に機能しなくなる原因になっていないか？

を実証すること

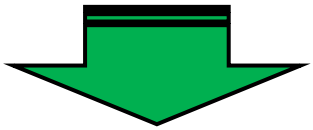
分離設計を阻害する要因

- ◆ **空間域及び時間域を共有するモジュール間**において、理論上、**独立性/非干渉性**を達成した設計を検討すること
⇒ **通常動作時**、及び**故障条件下での動作**の両方を検討
考え得る全ての原因を**識別**
- ランダム・アクセス・メモリ(RAM)の共用
- 周辺機器の共用
- CPU時間の共用(二つ以上のモジュールを**同一CPUによって実行**する場合)
- 設計全体を達成するために必要となるモジュール間通信
- あるモジュールの故障(オーバーフロー, ゼロ除算例外, 誤ったポイント計算等)が, その結果として, 別モジュールの故障を引き起こす可能性

空間域での独立性/非干渉性の達成

- ◆ 異なるASIL間(QM-ASIL間含む)でハードウェアメモリ保護の使用
 - メモリ配置(恒久的記憶装置含む)の分離(**バンク分割**等)
- ◆ ハードウェアメモリ保護が利用可能であることを前提に, 各モジュールが自身の仮想メモリ空間を有する自身のプロセスにおいて実行することを可能にするOSの使用
- ◆ ハードウェアメモリ保護が利用できない場合, 他モジュールに属す**データ**が**上書き**されているソフトウェアモジュール間から**明示的**あるいは**暗黙的**な**メモリ参照しない**ことの実証
 - 厳密な設計解析
 - ソースコード解析
 - オブジェクトコード解析

空間域での独立性/非干渉性の達成(続き)

- ◆ データの**完全度の十分性**が**検証不可**の場合、**低いASIL** (QM含む) **モジュール**から**高いASILモジュール**へデータを**受け渡さない**こと
 - データの受渡が必要な場合は、**共有メモリではなく、メッセージ**あるいは**パイプ**のような**片方向インタフェース**で実現すること
- 
- 通信機構は、送信側及び受信側の双方において**故障することがない**か、**データ送信が停止**した場合又は**遅延**した場合に、両モジュールの**実行を停止するメカニズムの実装**が必要

時間域での独立性/非干渉性の達成

- ◆ 決定論的スケジューリング手法の適用
 - **実行時間分析**で裏付けられた、各モジュールのワースト実行時間を割り当てた**タイミング要求事項**を**満足することの静的検証**
 - 時間駆動アーキテクチャ
- ◆ **優先順位**が**逆転**することを**回避する手段**を有するリアルタイム実行部によって実現する、**厳密な優先順位**に基づく**スケジューリング**
- ◆ 割り当てた実行時間あるいは期限を超過した場合に、当該**モジュールの実行を終了するタイムフェンス**(**確定時間**)
 - 潜在危険分析において、モジュール実行の終了が**危険側故障に繋がらないことを検証**すること
 - 本技法は、**非安全関連系への適用が最適**であることを示すこと

時間域での独立性/非干渉性の達成(続き)

- ◆ **タイムスライス**によって, どのプロセスも**CPU時間**が**欠乏することが無い**ことを保証するOS
 - 安全関連系が満足しなければならない, **厳しいリアルタイム要求が無い**ことを示すこと
 - スケジューリング・アルゴリズムが, どのモジュールに対しても**不当な遅延を招かない**ことを示すこと
- ◆ **共有資源**に**アクセス**するために必要な時間を考慮すること

インデックス

ソフトウェアに関する問題事例

製品開発に対する安全要求

ソフトウェアにおける安全要求の実装

ソフトウェア安全状態の分析方法

ソフトウェアに対するセキュリティ要求

ソフトウェア安全分析:故障モードの考え方

◆ 帰納的分析手法としてFMEA(Failure Mode and Effect Analysis)を適用？

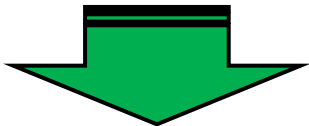
- ソフトウェアには故障モードが存在しない



- 故障モードの代わりにHAZOP(HAZard and OPerability Study)のガイドワー
ドを適用した分析
- FMEAの代わりにHAZOPを適用した分析

◆ 演繹的分析手法としてFTA(Fault Tree Analysis)を適用？

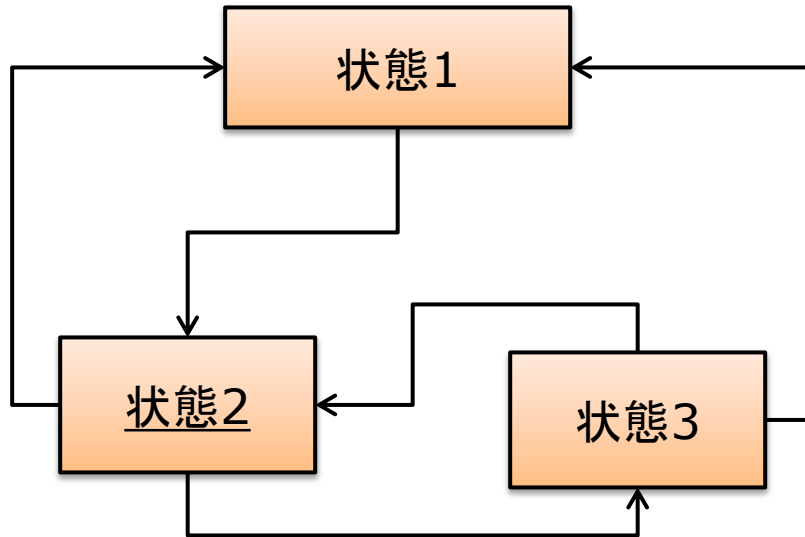
- HAZOPにより導出された危険事象をトップ事象として検証
⇒ ソフトウェア単体で実施しても効果が限定的



- ソフトウェアにおける安全分析結果をシステムレベルのFTAに反映し、システム全体で検証

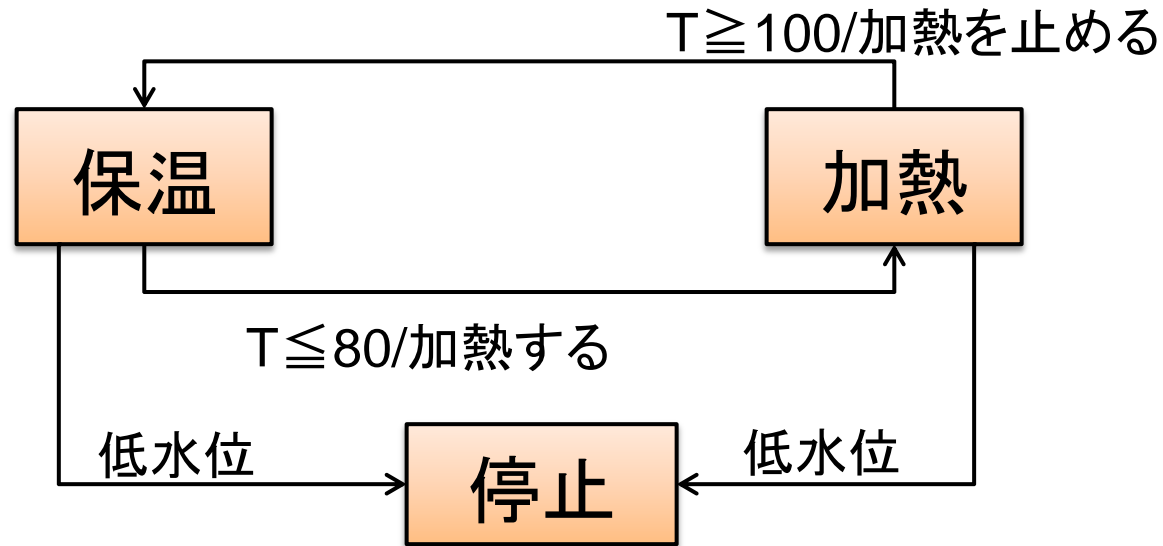
安全に関わる要求の洗出しについて

- ◆ 状態遷移図に対して、HAZOPのガイドワードを適用して、意図した仕様からのズレを洗い出すことで、安全要求の導出を実施



ガイドワード		遷移に対する解釈
No	否定	事象が未発生
More	量的変化	N/A
Less		N/A
As well as	質的变化	事象を誤検出
Part of		不完全な遷移
Reverse	置換	事象が未伝達
Other than		別事象を誤認

状態遷移図：電気ポットの例



備考:期待しない事象は無視するものとする

状態遷移図：HAZOP解析例

状態遷移: 加熱 → 保温

事象: $T \geq 100$

ガイドワード	原因と状況	結果と対策
No	沸騰しても100度を発生しない	加熱が続く <u>低水位が作動すれば空だきは防げる</u>
As well as	沸騰を誤検出	沸騰に至らないまま、80度になれば、 加熱再開
Part of	加熱を止めることができずに、保温状態へ	保温状態で加熱が続く <u>低水位が作動すれば空だきは防げる</u>
Reverse	沸騰しても事象が伝わらない	加熱が続く <u>低水位が作動すれば空だきは防げる</u>
Other than	低水位を100度と誤認	保温状態に移る その後、加熱状態に戻ってから <u>低水位が作動しない危険がある</u> 対策として低水位を2回検出する

インデックス

ソフトウェアに関する問題事例

製品開発に対する安全要求

ソフトウェアにおける安全要求の実装

ソフトウェア安全状態の分析方法

ソフトウェアに対するセキュリティ要求

セキュリティの脆弱性に対する賠償請求例

- ◆ X社がECサイトを運営開始(2009/4/15)



- ◆ 外部からの不正アクセスにより、最大7316件のクレジットカード情報が漏洩



- ◆ X社は謝罪、対応、調査等の費用、売上減少による損害を、Y社(システム開発会社)に対して、委託契約の債務不履行に基づき損害賠償を請求



- ◆ 東京地裁は約2262万円の損害賠償金を支払うことを命じ、確定

自動車セキュリティの脆弱性事例の変遷

◆ 実車両に対する脆弱性の事例

- 2010年：販売されている車両に対する脅威(**直接**)
 - ▶ 急ブレーキ/ブレーキの無効化
 - ▶ 運転手の意思とは関係無いハンドルの操舵
 - ▶ ワイパー/ライト/ロックの意図しない動作/無効化
- 2011年：販売されている車両に対する脅威(**間接**)
 - ▶ 遠隔(自動車の外部ネットワーク)から車載LAN(CAN)への侵入
- 2013年：車載機のプログラムが**書き換えられる**事例
- 2015年：**無線通信を介した攻撃事例**が多発
 - ▶ BMW Connected Driveで**携帯電話基地局のなりすまし攻撃**
HTTP⇒HTTPS(暗号化)で通信するように対策

ハッキング対策で初リコール: クライスラー

【ニューヨーク=杉本貴司】米FCAUS(旧クライスラー)は24日、米国内でハッキング対策のため140万台をリコール(回収・無償修理)すると発表した。[ソフトウェア](#)を更新して、[ハッカー](#)が無線を通じて車の操縦を乗っ取るような事態を防ぐ。米国ではネットにつながる「賢い車」が普及する一方、安全性への関心も急速に高まっている。ハッキング対策を目的とした初の大規模リコールとなる。

米国では著名ハッカーがネットにつながる車の安全性に警鐘をならす目的で、米専門誌と共同でクライスラー車に乗っ取る実験を行い、ネット上で公開したことが話題となっている。クライスラーが使う専用無線回線「Uコネクト」から車の頭脳であるコンピューターに侵入して、外部からエンジンを切ったり、ワイパーを動かしたりした。遠隔操作でハンドルを動かしたり、加減速させたりできるという。

クライスラーはこのような「遠隔操作による犯罪行為」を未然に防ぐ目的で、すでに改良ソフトウェアを配信していたが、社会的な関心が高まっていることから自主的リコールに切り替えた。同社によると現時点でハッキングによるけがなどはないとしている。

対象となるのは[SUV](#)(多目的スポーツ車)「ジープ・グランドチェロキー」や主力セダン「クライスラー300」など、Uコネクトの機能があり8.4インチスクリーンを搭載した車種。

ハッカーによる乗っ取り映像の公開を受けて、米議員は対策法案を議会に提出している。米自動車3社は各社とも専用回線を持っており、今後はクライスラーと同様の対応に追われる可能性がある。ただ、ソフトを更新してもハッカー対策はいたちごっこになることが多く、「賢い車」の開発に力を入れる世界の自動車大手にとって新たな課題となりそうだ。

ご清聴有難う
ございました!!