Orchestrating a brighter world NEC

第1回 STAMP Workshop in Japan

制御システム以外の組込みシステムのソフトウェア障害に対するSTAMP適用の試み

Trial which applied STAMP to software fault of embedded system different from control system

日本電気通信システム株式会社

NEC Communication Systems, Ltd.

○羽田 裕　難波 秀之　佐野 忠

○Yutaka Hada, Hideyuki Namba, Tadashi Sano

氏名：羽田 裕（Yutaka Hada）

■ 現職

NEC通信システム ・品質推進本部・ソフトウェア技術センターに勤務.

- 所属組織は，社内の開発部門に対して「製品実現における設計や検証の技術・手法の提供」など，ソフトウェア生産技術に関する専門的な共通サービスの提供を機能とする.
- 自身は，技術移転・教育，新技術開発・獲得を担当.

■ 社内での業務経験

1984年に入社．局用電子交換機の評価・検査，携帯電話ソフトウェア開発，等に従事．2009年から現職.

■ 社外における活動

- 情報処理推進機構（IPA/SEC）IPA/SEC連携委員，未然防止知識WG
- 情報処理学会（IPSJ）組込みシステム研究会会員
- ソフトウェア技術者協会（SEA）会員，など

講演概要

- 弊社は，従来から通信事業者や企業・官公庁向けを始めとした各種ネットワークシステムのソフトウェアを開発しています.
- しかし，近年サービス多様化に伴うネットワークの追加・統合によりシステムの複雑さが増大し，ソフトウェア・リリース後の障害の分析・処置も困難さを増しています.
- このような状況を改善するため，実際に発生した障害を分析対象として，STAMPを試行的に適用しました.
- その結果，適用のねらいとした，障害の処置内容に対する十分性の確認について可能性を見出すことができました.
- 本発表では，トライアルの概要をご紹介するとともに，今後の展望についてお話します.

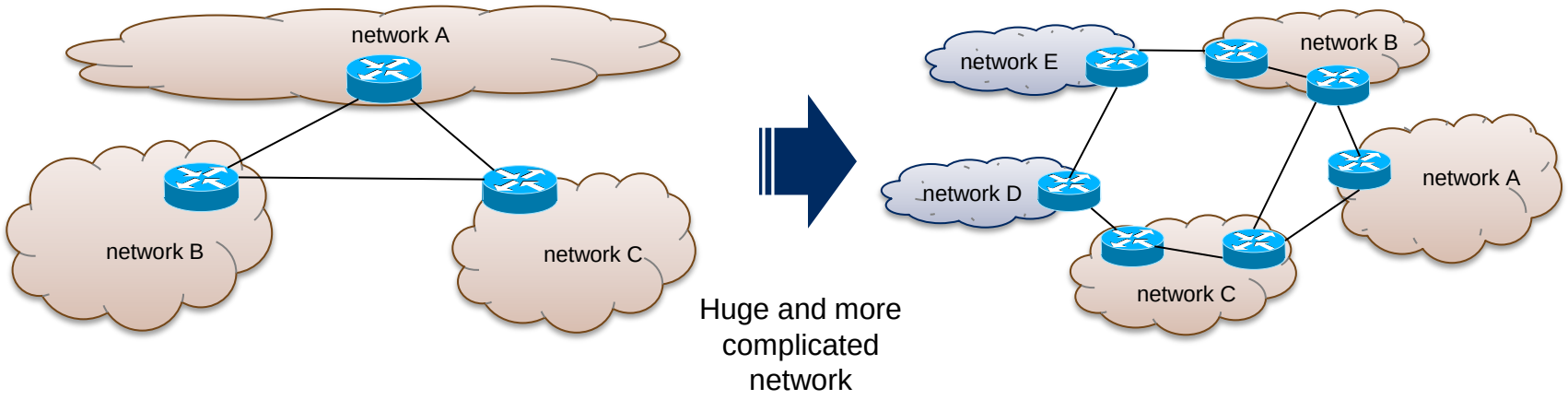
本日ご紹介する内容

1. 複雑化・増大する通信ネットワークシステム
2. 現状のソフトウェア障害の分析
3. トライアルのねらいとアプローチ
4. トライアル結果
5. 考察
6. 今後に向けて
7. まとめ

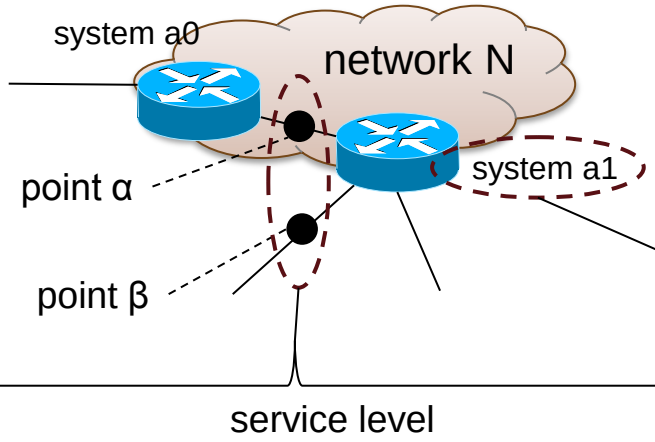
1．複雑化・増大する通信ネットワークシステム

- サービス多様化に伴うネットワークの追加・統合によりシステムの複雑さが増大
  - SoS (System of Systems) における，システム統合による創発性 (Emergent Properties) の 出現と類似の状況.
  - ソフトウェアリリース後の障害分析・処置の困難さが増している.
    - 障害原因の特定に時間がかかる.
      発生条件が識別できない，障害を再現できない.
    - 処置の妥当性の判断が困難.
      処置の影響が他に伝搬するケースも.

STAMP Trial

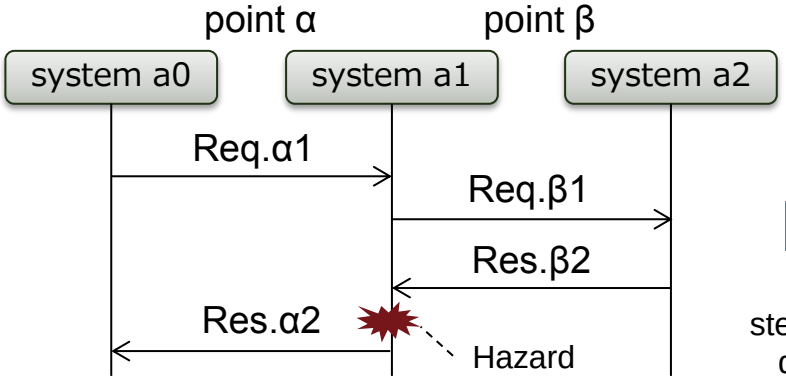


- 典型的にはシーケンス図を使い，段階的に詳細化して分析
  - 他ネットワークとのインタフェース規定点からネットワークシステム内部へと，障害シーケンスを段階的に表現していく．
    - 通信レイヤ，レイヤごとのプロトコル，通信フォーマット，…



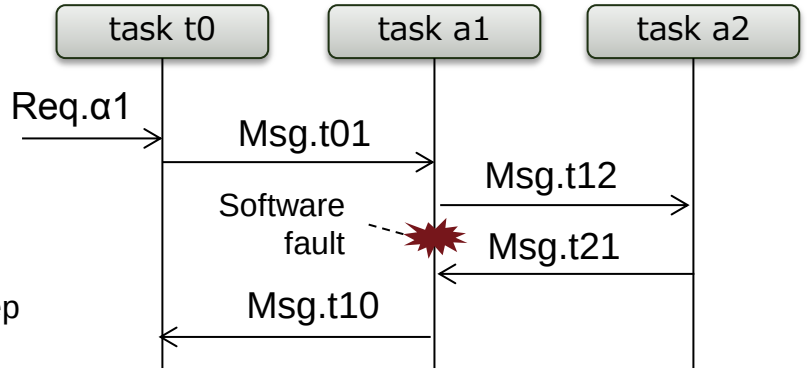
- 複雑なトランザクション管理（複数の状態遷移モデルの複合体）．
  - ✓ 障害原因の特定に時間がかかる．
- 再発防止は，主に類似の原因メカニズム探索．
  - ✓ 他の原因で，同じ障害が発生する場合も．

service level



step-by-step detailing

inside system a1



3．トライアルのねらいとアプローチ（1/3）

- ねらいは「ソフトウェア障害の処置内容に対する十分性の確認」
  - 他の原因による同じ障害の発生を抑止したい.
  - 再現・分析が困難な場合，原因特定の手掛かりとしたい.
  - 評価項目を次のように設定.

    • 実際に発生した障害の原因が特定できること.
    • 障害に至る他の有意な潜在的要因も抽出できること.

- 第三者が分析者となり，STAMPを適用して自社の実際に発生したソフトウェア障害を分析
  - 開発者ではない，筆者らがSTAMPを使って分析.
  - 障害は，システムのフィーチャーを阻害するレベルのもの.
  - 分析結果として典型的な報告（シーケンス図）が含まれ，筆者らが設計文書を入手可能なもの.
  - 分析にあたって，事前に輪講を実施． (An STPA Primer Version 1)

3．トライアルのねらいとアプローチ（2/3）

● トライアルのアプローチ

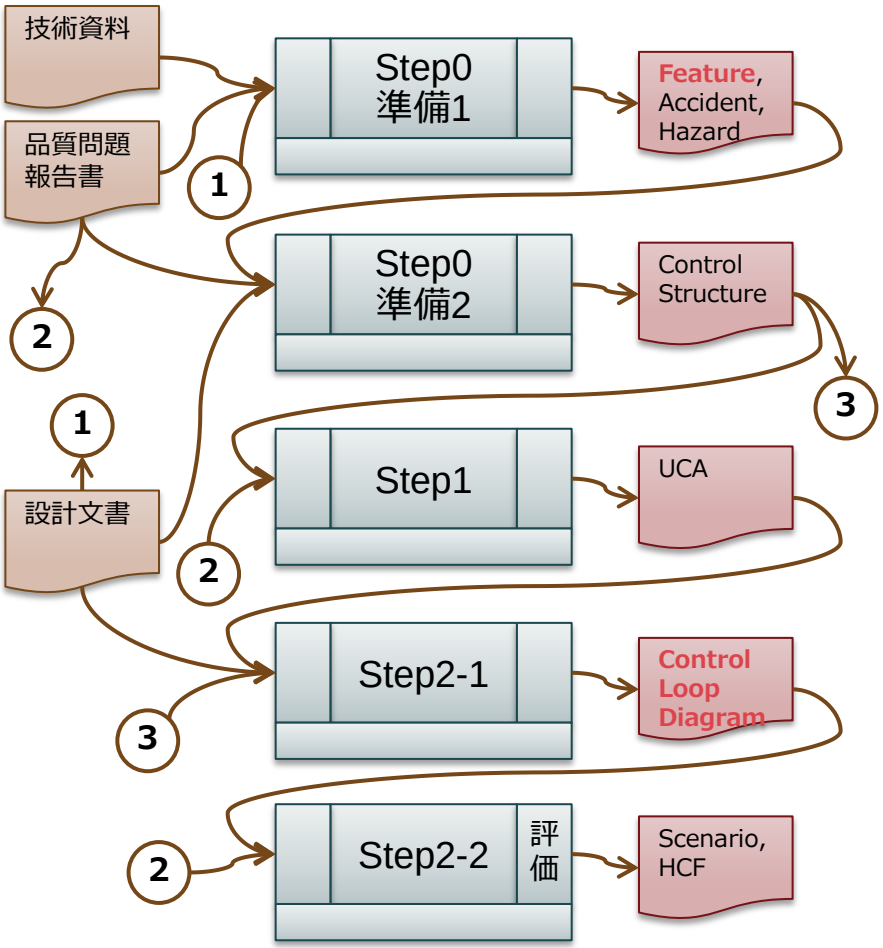


Figure 3.1: Process Network Diagram for Trial

Table 3.1: Trial Process

プロセス	作業内容
Step0 準備1：フィーチャー，事故，ハザードの識別	1. **対象システムのフィーチャーを識別する．障害がフィーチャーを阻害するものであることを確認する．** 2. フィーチャーから「ステークホルダーが許容できない損失」を事故として識別する． 3. 事故から，ハザード（事故が潜在するシステム状態）を識別する． 4. 分析対象の障害に対応するハザードを識別する．
Step0 準備2：コントロールストラクチャの作成	1. **フィーチャーにフォーカスして**，分析対象の障害に対応するハザードの制御関係を表すモデル（コントロールストラクチャ）を作成する．
Step1：UCA (Unsafe Control Action)の抽出	1. コントロールストラクチャの各コンポーネントの制御アクションに4つのガイドワードを適用して，UCAを抽出する．
Step2-1：Control Loop図の作成	1. 実際に発生したUCAについて，UCAが発生する原因を分析するため，**ソフトウェア機構を考慮したControl Loop図**を作成する．
Step2-2：HCF (Hazard Causal Factor)の特定	1. Contorol Loop図を見ながら，実際に発生したUCAの発生原因（HCF）を特定する．

3．トライアルのねらいとアプローチ（3/3）

- Control Loop図にアクチュエータ，センサーに相当するものとして，対象ソフトウェア上の機構を配置
  - 制御するコンポーネントと制御されるコンポーネントの関係からアクチュエータとセンサーを次のように扱う．

Case 1：2つのコンポーネントが同一コンテキストで動作する場合

アクチュエータとセンサーを省略． e.g. Fuction Call, …

Case 2：2つのコンポーネントが別なコンテキストで動作する場合

各コンポーネントに相当するソフトウェアコンポーネント間で，実際に使用されているソフトウェア機構を配置する．

e.g. Message Queue, External Variable, ….

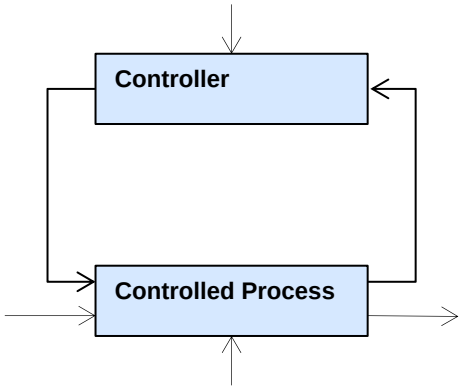


Figure 3.2: Case 1

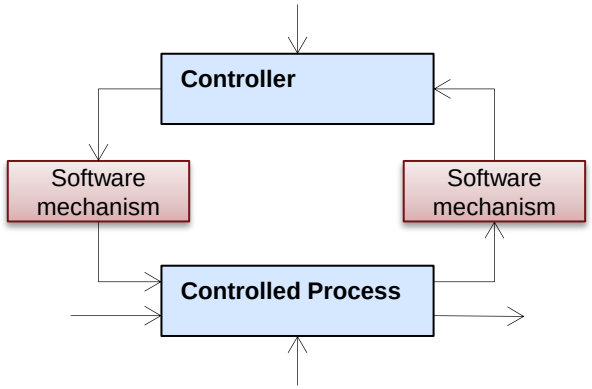


Figure 3.3: Case 2

４．トライアル結果（1/7）　分析対象のソフトウェア障害

- 分析対象のソフトウェア障害（観察された現象）
  - セッション制御プロセスが，Req送信契機のオーダや周期処理の要求を受け付けない.

- 対象システムのフィーチャー，事故とハザード
  - 通信制御システムのフィーチャー (Feature)
    - セッション毎のトラフィック規制.
  - 事故 (Accident)

Table 4.1: Accident

防止するシステムレベルの事故
A-1：トラフィック規制機能の停止（セッション接続不可）
A-2：誤ったトラフィック規制（経済的損失）

  - ハザード (Hazard)
    - 対象のソフトウェア障害に対応するハザード を特定 (H-1, H-2).

Table 4.2: Hazard

ハザード	関連する事故
H-1：セッション毎のポリシー情報（通信量，帯域）の設定不可	A-1
H-2：セッション毎のポリシー情報の更新不可	A-1，A-2
H-3：セッション毎のポリシー情報の破壊	A-1，A-2
H-4：通信帯域制御要求の不可	A-2

4．トライアル結果（3/7）　Step0 準備2：Control structure

- 対象のソフトウェア障害に対応するハザードのコントロールストラクチャ
  - 保守者がポリシー情報更新要求を発行しても，対向システムへポリシー情報更新要求が送信されない．(H-1, H-2)

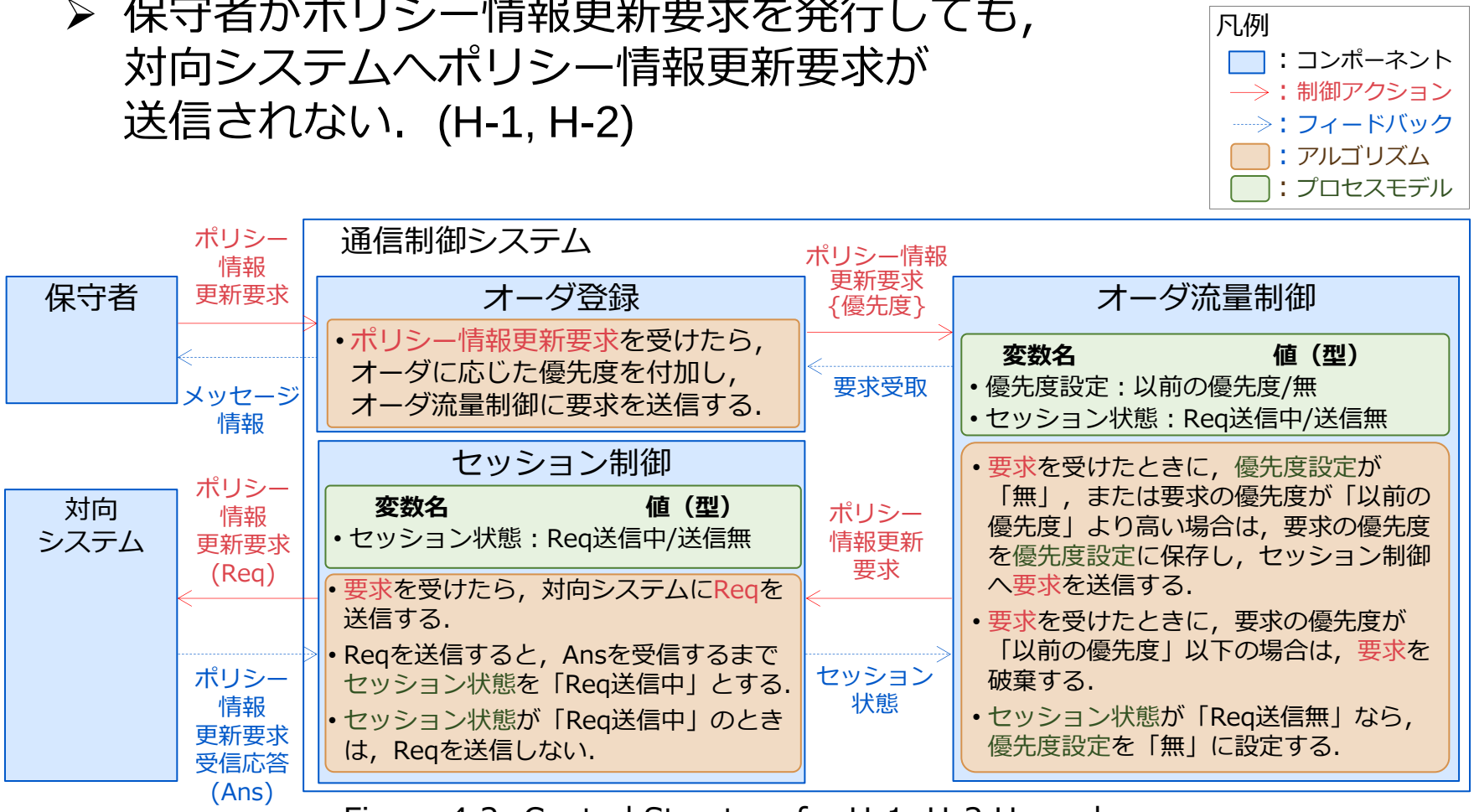


Figure 4.2: Control Structure for H-1, H-2 Hazard

● 対応ハザードにつながる非安全な制御アクション (UCA)

Table 4.3: UCA for H-1, H-2 Hazard

制御アクションを発行するコンポーネント	制御アクション		ガイドワード (a) 必要な状況で発行されない	ガイドワード (b) 誤った内容で発行される	ガイドワード (c) 発行のタイミングが早すぎる／遅すぎる	ガイドワード (d) 発行が長すぎる／短すぎる	コメント
保守者	1	ポリシー情報更新要求	• ポリシー情報更新が必要な状況で発行されない.	• ポリシー情報を更新してはいけない状況で発行される.	• 必要なタイミングを逸して要求する.		運用ルール不明のため分析対象外.
オーダ登録	2	ポリシー情報更新要求	• ポリシー情報更新要求を受けたが，オーダ流量制御へ要求が発行されない.	• ポリシー情報更新要求を受けていないのに，オーダ流量制御へ要求を発行する. • オーダ流量制御向け要求への変換を誤る.（セッションID，優先度など）	• オーダ流量制御の処理能力以上の間隔で連続して要求を発行する.	• オーダ流量制御へ同じ要求を何回も発行する.	
オーダ流量制御	3	ポリシー情報更新要求	• ポリシー情報更新要求を受けたが，セッション制御に送り出さない.　（UCA3-a）	• ポリシー情報更新要求を受けていないのに，セッション制御へ要求を発行する. • セッション制御向け要求への変換を誤る.	• セッション制御の処理能力以上の間隔で連続して要求を発行する.	• セッション制御へ同じ要求を何回も発行する.	UCA3-aが，実際に発生したUCA.
セッション制御	4	ポリシー情報更新要求 (Req)	• ポリシー情報更新要求を受けているがReqを発行しない.	• ポリシー情報更新要求を受けていないのにReqを発行する. • 要求からReqへの変換を誤る.	• 対向システムの処理能力以上の間隔で連続してReqを発行する.	• 対向システムへ同じReqを何回も発行する.	

- 実際に発生したUCA (UCA3-a) の発生原因を分析するための Control Loop図

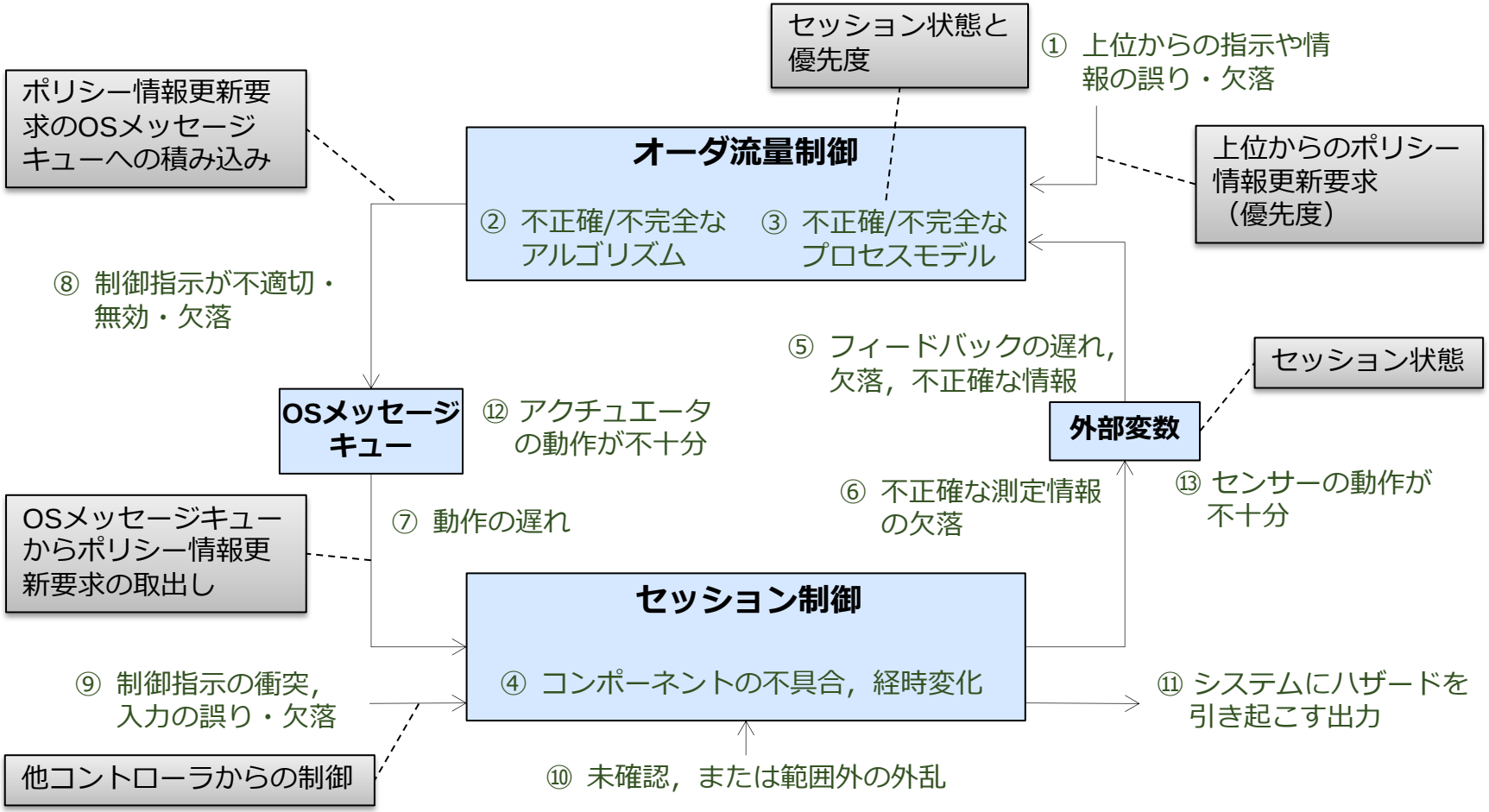


Figure 4.3: Control Loop Diagram for UCA3-a

- Control Loop図から特定したUCA3-aにつながるシナリオと誘発要因
  - 実際のソフトウェア障害の原因を特定（Table 4.4 : HCF No.2）.

Table 4.4: Scenario and HCF for UCA3-a

（UCA3-a）オータ流量制御がポリシー情報更新要求を受けたが，（ポリシー情報更新要求を）セッション制御に送り出さない			
Scenario	Hazard Causal Factors		Note
1. オータ流量制御がポリシー情報更新要求をOSメッセージキューに積み込まない.	1	① 上位からのポリシー情報更新要求内のパラメータに誤りがある.	
	2	③ 「優先度設定」の値が不正確な値を保持している.	実際の原因
	3	③ 「セッション状態」の値が不正確な値を保持している.	
	4	⑫ OSメッセージキューがいっぱい.	
2. セッション制御がOSメッセージキューからオータを取り出さない.	5	⑦ ポリシー情報更新要求が遅れて到達する.	
	6	⑨ 他コントローラが，ポリシー情報更新要求を取り出さない設定をしている.	
3. セッション制御（スレッド）が停止する.	7	⑨ コントロールアクションの衝突．プロセス入力の喪失または誤り.	
	8	⑩ 未確認，または範囲外の障害.	

4．トライアル結果（7/7）　評価

- 実際に発生した障害の原因を特定し，他の有意な潜在的要因も特定できた
  - 実際の障害原因をHCFで特定．(Table 4.4 : HCF No.2)
  - 他の有意な潜在的要因も特定．
    - 後日，同システムで該当原因の障害が発生．(Table 4.4 : HCF No.3)

- 障害発生時に，実際の障害原因だけでなく，他の潜在的要因についても検討が可能となり，発生した障害の他の要因による再発防止が期待できる．
- 障害の再現が難しい場合に，原因特定の手掛かりとすることができる．

5．考察（1/3）

- 障害の事後分析では，フィーチャーの識別が分析のカギ
  - 障害で観察された現象をアクシデントとすると，作成するコントロールストラクチャは発散.
    - 初期に識別したアクシデント（＝観察された現象）

      「セッション制御プロセスが，Req送信契機のオーダや周期処理の要求を受け付けない」
  - フィーチャーを識別してからアクシデントとハザードを識別すると，作成するコントロールストラクチャは収束.
    - 障害によっては，サブシステムに分割した後，該当サブシステムのフィーチャー識別が適切かもしれない.
  - 開発者が分析する場合，フィーチャーの識別に課題がありそう.
    - ソフトウェア開発者は，フィーチャーを意識しないこともある．UMLユースケース図の作成（適切な粒度のユースケース）は一つの対策.
    - テスト技術者を分析者に加えることも有効かもしれない.

      ISO/IEC/IEEE 29119-2 Test Design & Implementation Process
      (TD1 : Identify Feature Sets)

5．考察（2/3）

- コントロールストラクチャは，障害を表現してからフィーチャーで洗練する
  - 当初，観察された現象に出現するほとんどの要素を記載.
    - 品質問題報告書のシーケンス図のライフラインとメッセージを，コンポーネントと制御アクション/フィードバックに見立ててコントロールストラクチャを作成．複雑すぎるコントロールストラクチャ.
  - フィーチャーにフォーカスし，コントロールストラクチャを洗練.
    - 余分な要素を排除したコントロールストラクチャ.
- ソフトウェア機構は，コントロールストラクチャではなくControl Loop図に表現する
  - コントロールストラクチャに，ソフトウェア機構を出現させると複雑になりすぎる.
  - HCFを特定する際にソフトウェア機構を想定する.
    - 分析者にソフトウェア開発経験があれば，ソフトウェア機構がシナリオの導出を促す．　e.g. Message Queue, Thread, …

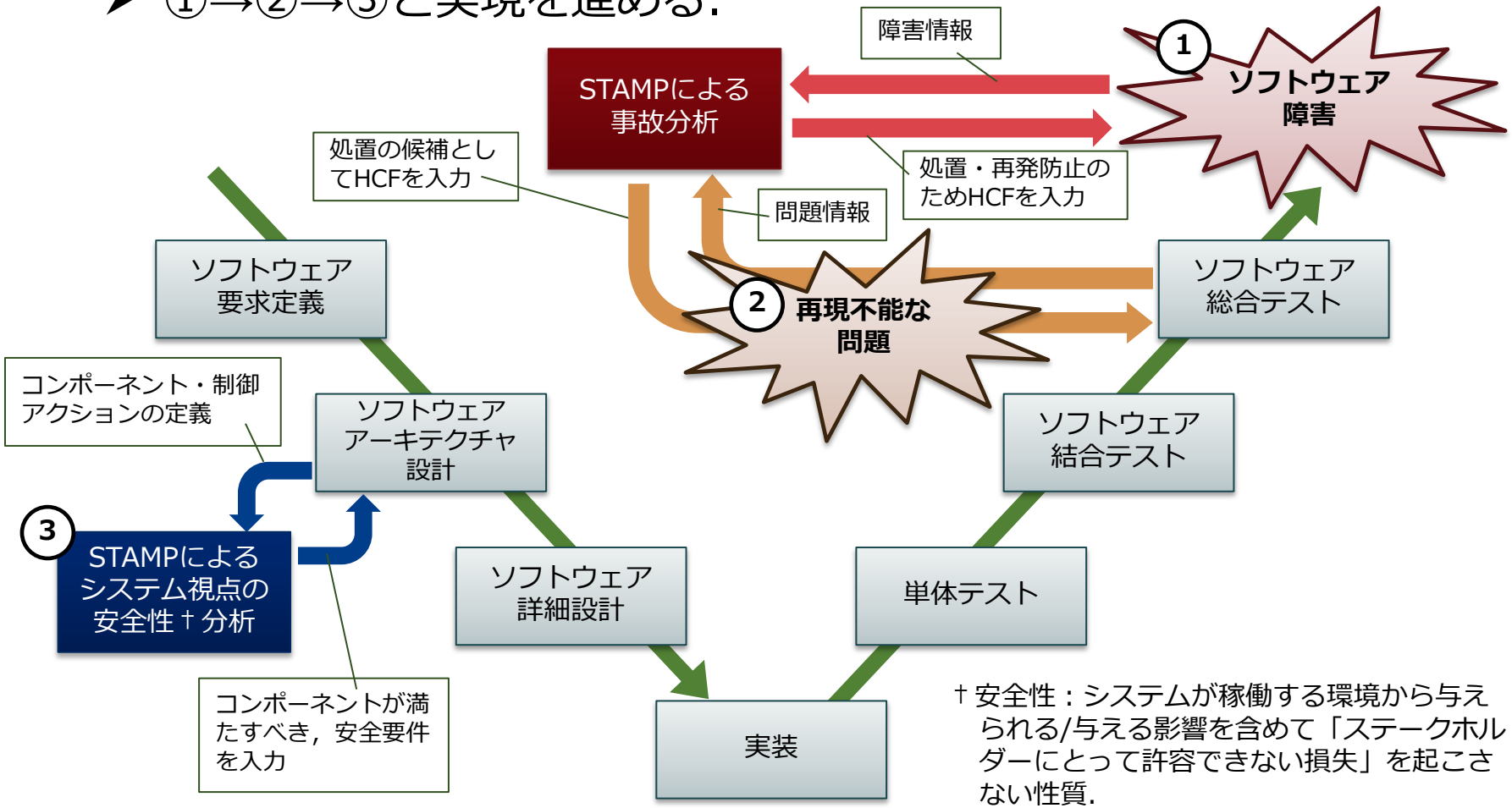
5．考察（3/3）

- 特定のモデリング技法の知見がなくとも，分析者が認識を共有できる
  - シーケンス図の知見はなくともよい．一意性や理解容易性に不安のある独自表現を加えたシーケンス図も不要．
    - 基本的には，シーケンス図にアルゴリズムやプロセスモデルのような情報は表現されない．シーケンス図を分析に適用した場合は，それらの情報収集および分析について，別な手段が必要となる．
    - 複数の関心事を一つのモデルで表現することは，その場しのぎにはなるがゆくゆくは禍根となることが多い．

      e.g. 独自モデリングを多用した設計文書による一意性や理解性の低下，…
- 分析の成果物を，分析の客観的な証跡として利用できる
  - 原因特定に至る分析の経過，範囲を客観的に確認できる．
    - 障害に対して検討した要因が明示的．

6．今後に向けて

- 考察の妥当性を検証し，障害の事後分析だけでなく，開発プロセスのより早い時期の適用を目指す
  - ①→②→③と実現を進める.



† 安全性：システムが稼働する環境から与えられる/与える影響を含めて「ステークホルダーにとって許容できない損失」を起こさない性質.

7．まとめ

- 「処置内容に対する十分性の確認」を目的に，自社の実際に発生した障害を分析対象としてSTAMPを試行的に適用しました.
- 適用では，STAMPの一般的な分析手順に加え，対象システムのフィーチャーの識別やソフトウェア構造に特有なControl Loop図の作成を実施しました.
- 結果，実際の障害原因と他の有意な潜在的要因を特定し，当初の目的に対する見通しを得ることができました.
- 今後は，いくつか追加試行を実施して考察の妥当性を検証し，障害の事後分析だけでなく，開発プロセスのより早い時期の適用を目指していきます.

ご清聴ありがとうございました

参考文献

[1] 落水浩一郎，システムオブシステムズとディペンダビリティ，ソフトウェア技術者フォーラム，ソフトウェア技術者協会 (2016/11/24).

[2] Nancy G. Leveson(著)，松原友夫(監訳)，セーフウェア，翔泳社 (2009/10/29).

[3] Nancy G. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety (Engneering Systems), The MIT Press (2012/1/13).

[4] An STPA Primer, Available at: <http://psas.scripts.mit.edu/home/wp-content/uploads/2015/06/STPA-Primer-v1.pdf> [Accessed 2016/12/1].

[5] IPA/SEC，STAMP手法による調査報告書，IPA (2015/8).

[6] IPA/SEC，はじめてのSTAMP/STPA～システム思考に基づく新しい安全解析手法～，IPA (2016/4).

[7] ISO/IEC/IEEE 29119-2:Software and systems engineering-Software testing-Part 2:Test processes (2013/9/1).

[8] IPA/SEC，【改訂版】組込みソフトウェア向け開発プロセスガイド，翔泳社 (2007).

[9] 向山輝，STAMP/STPA～IoT時代の新しい安全性解析手法～，ET/IoT2016・IPAブースプレゼン，IPA/SEC (2016/11/17).

Orchestrating a brighter world

未来に向かい、人が生きる、豊かに生きるために欠かせないもの。
それは「安全」「安心」「効率」「公平」という価値が実現された社会です。

NECは、ネットワーク技術とコンピューティング技術をあわせ持つ
類のないインテグレーターとしてリーダーシップを発揮し、
卓越した技術とさまざまな知見やアイデアを融合することで、
世界の国々や地域の人々と協奏しながら、
明るく希望に満ちた暮らしと社会を実現し、未来につなげていきます。


Orchestrating a brighter world
NEC