発表資料

組込みシステムにおけるロバスト性の検証を目的としたテストケース設計手法の提案

国立研究開発法人　宇宙航空研究開発機構（JAXA）

研究開発部門　第三研究ユニット

◎小口一浩（発表者）　梅田浩貴　植田泰士　片平真史

oguchi.kazuhiro@jaxa.jp

JaSST'21 Tokyo　2021年3月15日

本プレゼンのターゲット

- 組み込みシステムテスト設計者として
  ロバスト性を検証するテストの作成に課題をお持ちの方

- 過去の欠陥情報（不具合情報）の活用に
  課題をお持ちの方

- Goal Structuring Notation（GSN）を使った分析を
  業務への適用を検討している方

初めに

- 本発表のアジェンダ
  - 背景説明(テスト対象の特性、課題)
  - 提案手法の概要＆適用事例
  - 提案手法の適用効果の確認
  - まとめ

- 本発表のターゲット
  組み込みシステムなどシステム用のテストを
  設計する際に、ロバスト性のテストにおける業務課題
  をお持ちの方

発表の概要

課題

組込みシステムのロバスト性を検証するためのテスト条件を特定する際、以下の課題がある

課題①：過去の欠陥情報からタスクがハードウェアへ負荷を発生させるまでの因果関係を特定し、
かつ特定された結果から、検証対象システム固有のテスト条件を識別する事が困難
- 主な理由１：3種類以上のテスト条件を表やチェックリストで特定する事が難しい
- 主な理由２：過去の欠陥情報からテスト対象の組込みシステムに対し、テスト条件を想起する具体的な手順がない

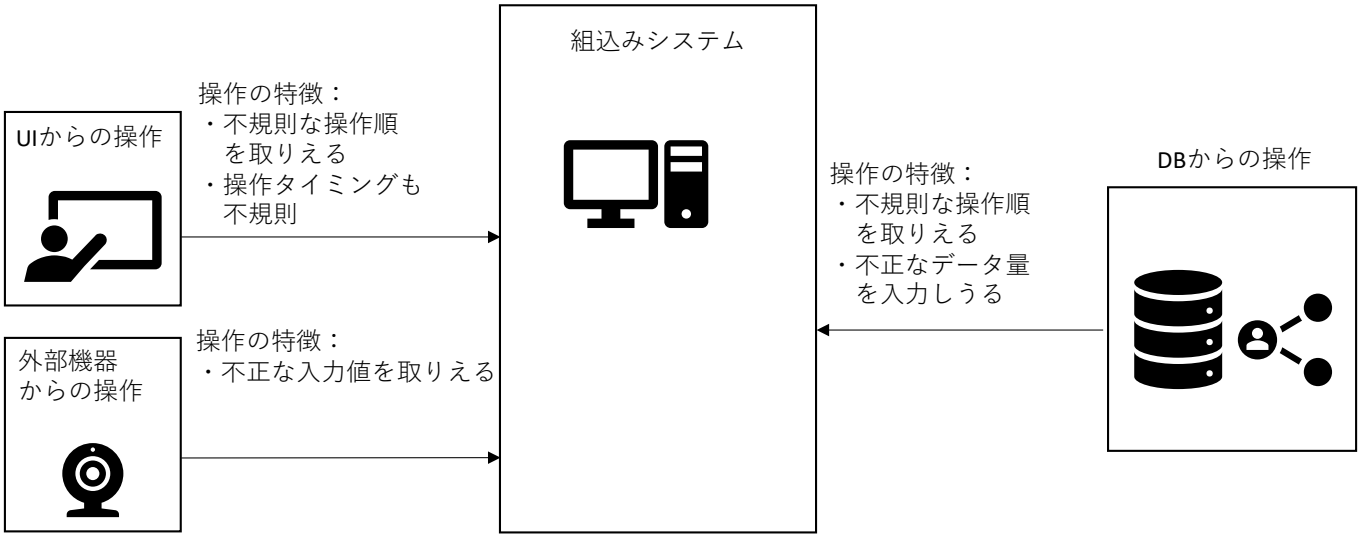
課題②：過去の欠陥情報から、各動作に対する制約の適用順を特定する事が困難
- 主な理由：2つの要素が関係する条件、時間経過に関する条件の表現が困難である

効果

- 提案手法は、複数条件の因果関係の膨大な母集合に対し、欠陥情報を用いた選別を行う事で外部環境やSW処理等の発生可能性のある条件の因果関係の識別に成功した
- 提案手法は、ガイドワード適用の促進により時間経過に関する制約の種類及び処理の分岐条件を可視化する事で、制約の適用順の特定に成功した

組み込みシステムの特徴

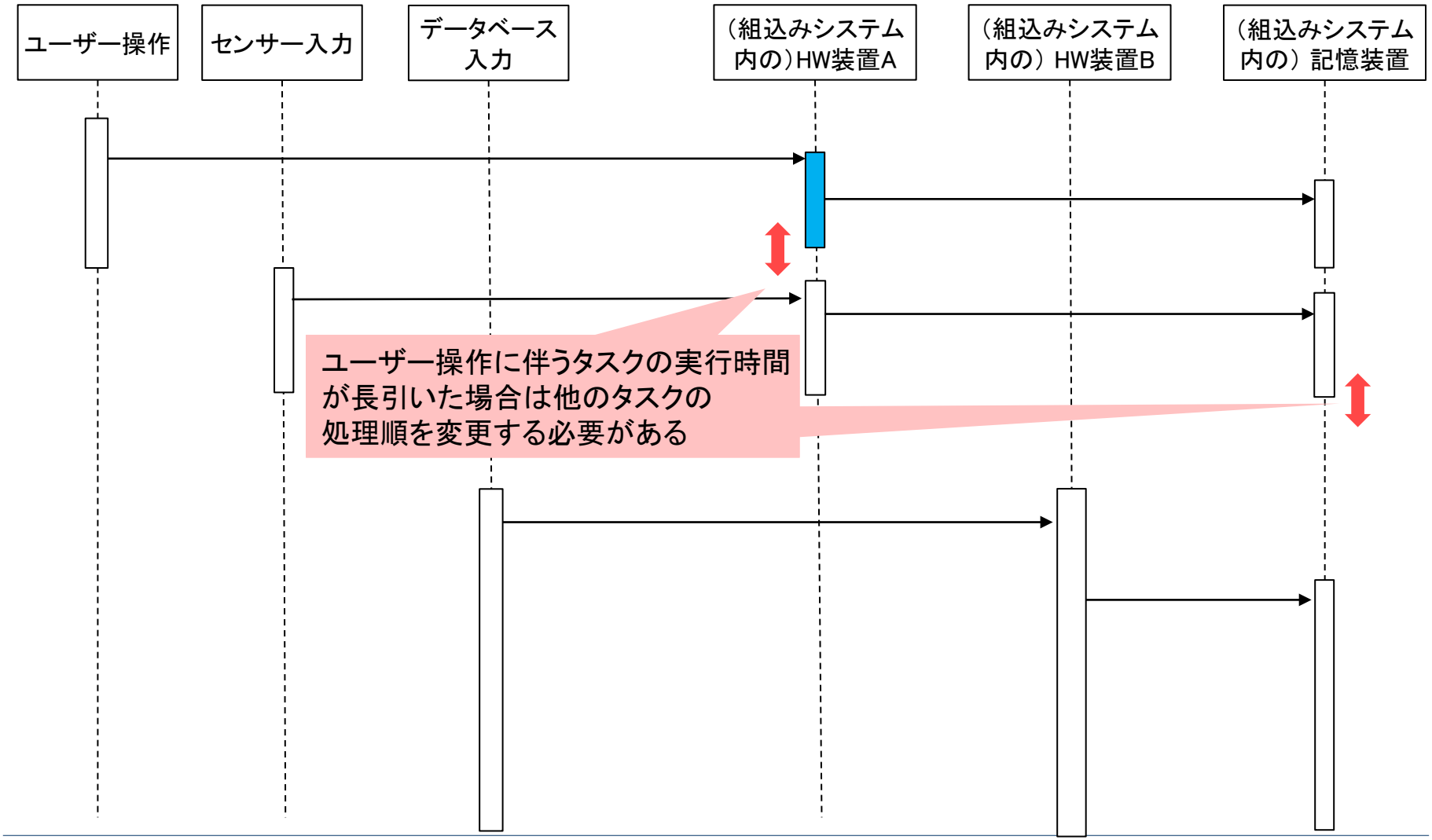
- 特徴1： 実行タスクが、システム内のハードウェアを長時間占有する事がある（例：高負荷タスク）
  - 処理データのサイズや熱負荷等が大きい時、メモリやMPUは制限時間を超えて処理を続ける事がある
  - 電源系、熱系、ネットワークなどシステムの置かれた環境が異常な状態の時、初期化や起動処理が制限時間を超えて続く事がある
- 特徴2： 多種多様な外部インターフェースから様々なタイミングで、タスクは実行される
  - ユーザー操作、外部機器からの操作、ネットワーク経由の操作等がなされる
- システム検証においては、『高負荷タスクによりタスクの実行順序が変わっても、情報の一貫性が保たれる事』を確認する必要がある（＝ロバスト性の検証）
  - 高負荷タスクが制限時間を超えて実行された結果、次のタスクで用いるハードウェアが解放されない場合、他のタスクを先に実行し次のタスクの実行順を先送りする必要がある



組込みシステムにおけるロバスト性の検証

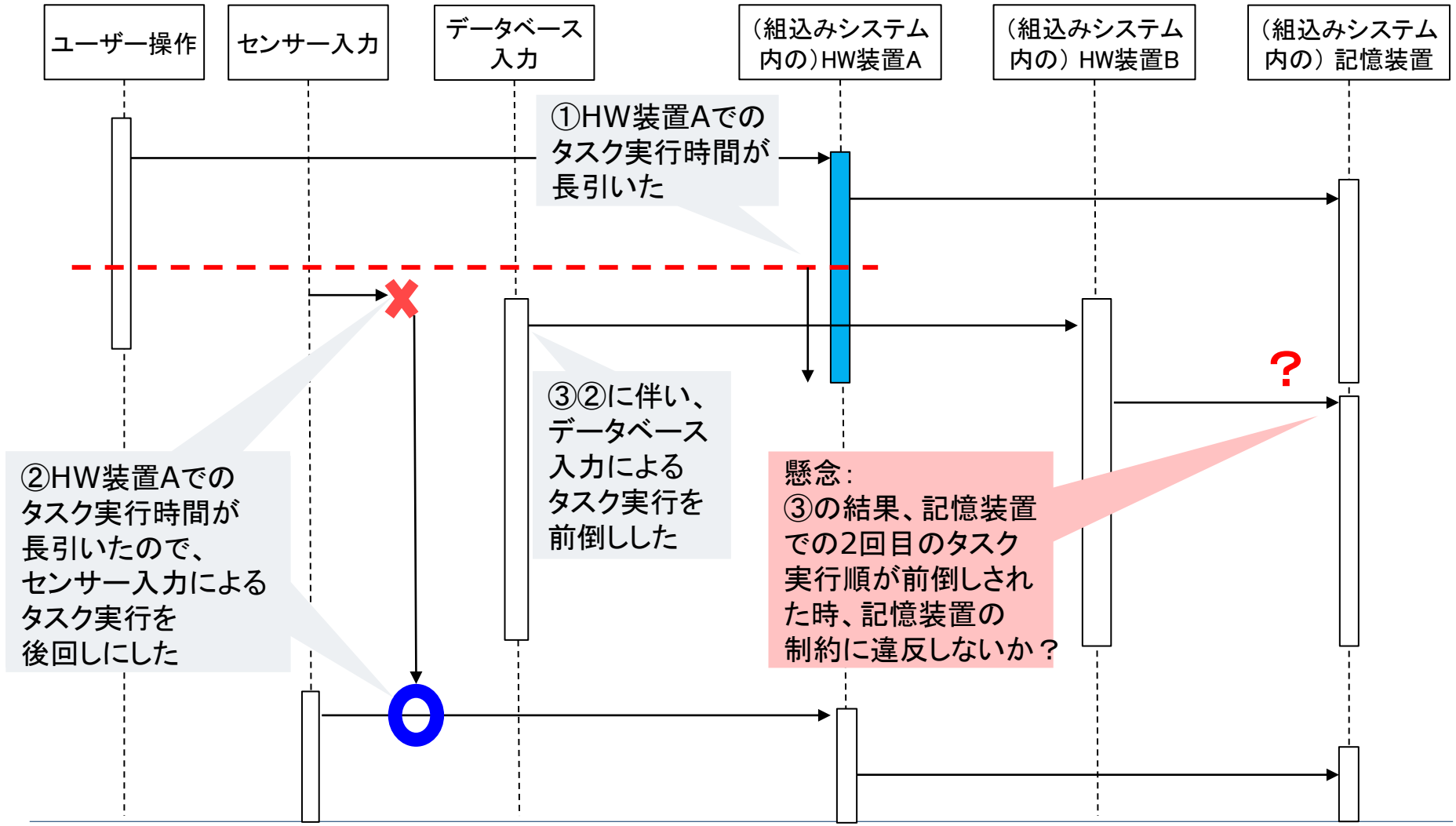
- 組込みシステムがタスクを実行する時、ハードウェアを占有する時間に幅があるため（※）、タスクの実行順序が代替フローへ変更される事がある

（※）タスク処理遅延、異なる外部インターフェースからの操作タイミングのズレ、誤った過去データの蓄積など



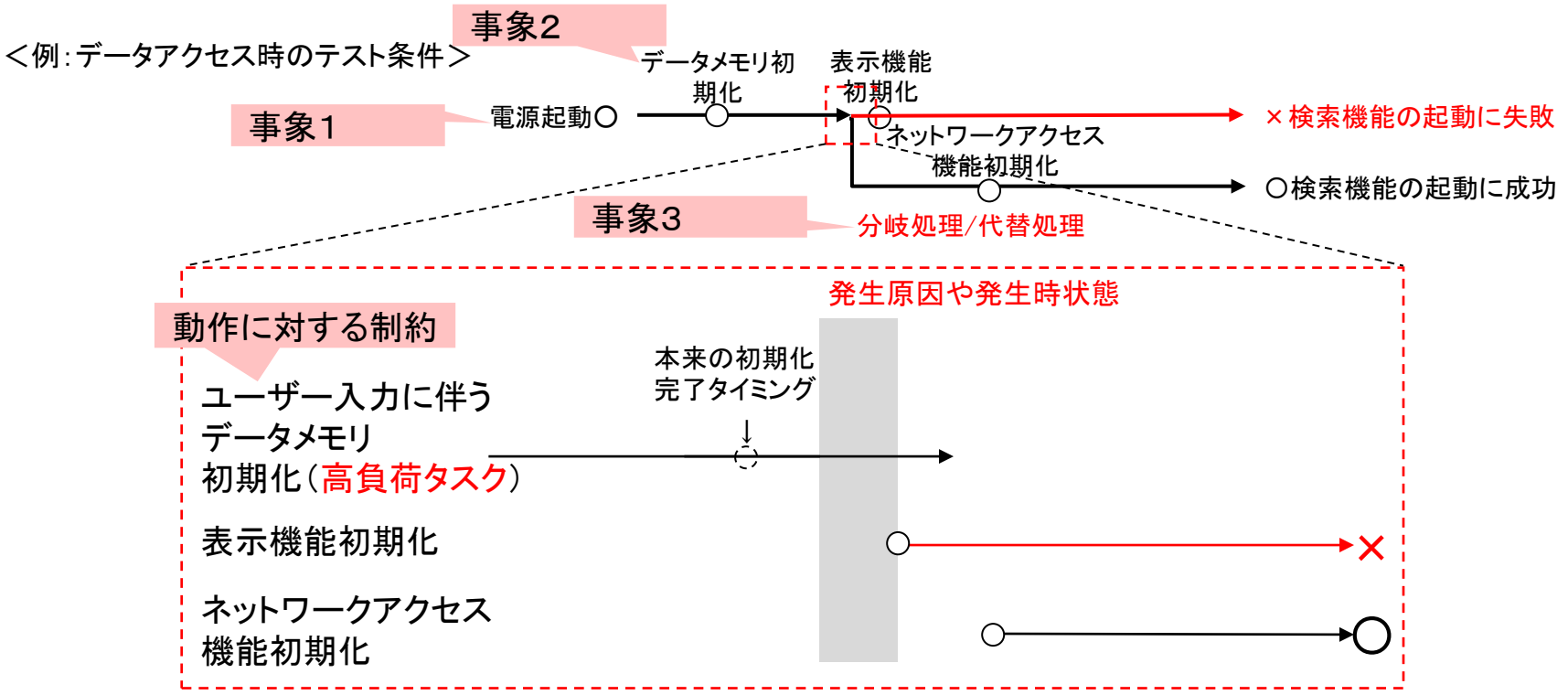
ロバスト性を検証するために特定すべきテスト条件

- 高負荷タスクによりタスクの実行順序が変更された際、タスクが占有するハードウェアの制約が破綻し、情報の一貫性が保たれない懸念があり、ユースケーステストでの検証が必要である
- テストを行うためには、インターフェース・高負荷タスク・高負荷タスクが動かすハードウェアのそれぞれの因果関係と守られるべき制約とをテスト条件として特定する必要がある



テスト条件を特定する際の課題

- インターフェース・高負荷タスク・高負荷タスクが動かすハードウェアのそれぞれに対し、
  タスク間の因果関係や各タスクへの制約を特定する時、過去の欠陥情報からテスト条件を読み取る事が多い
- 課題①： 過去の欠陥情報から、タスク間やタスクとハードウェア装置の負荷との因果関係として、
  検証対象システムに対するテスト条件を識別する事が困難
  （テスト条件は、欠陥情報から、以下3つの事象と事象間の因果関係とを特定する事で得られる）
  - 事象1：各外部インターフェースからどんな入出力があったか
  - 事象2：高負荷タスクがどのハードウェア装置を動かすか
  - 事象3：高負荷タスクに伴い、どのタイミングで代替処理が生じるか
- 課題②： 過去の欠陥情報から、各動作に対する制約の適用順を特定する事が困難



【参考】既存のテスト設計方法における課題

従来使われる「表」や「単一観点」を使った発生条件の識別

テスト設計のフォーマット①

テスト対象	テスト観点 もしくは 品質特性	テスト観点 もしくは 品質特性
○○機能	○	
××機能		○

■説明
テスト対象とテスト観点/品質特性を因果関係で表している。
例：○○機能と「入力」　[テスト対象×テスト観点]
　　××機能と「正確性」　[テスト対象×品質特性]
■工夫の必要なポイント1
3つ以上の要素間の因果関係に対し、要素の品質特性を表形式で対応付ける事が難しい
- {環境、システム、タスク}の各要素の品質特性の分類軸は、テスト観点によって異なる

因果関係によっては動作が変化する場合の考慮が困難
- 入力センサーの故障の仕方によって振る舞いが異なる　など

テスト設計のフォーマット②

大項目	中項目	小項目
○○機能	起動	
	終了	

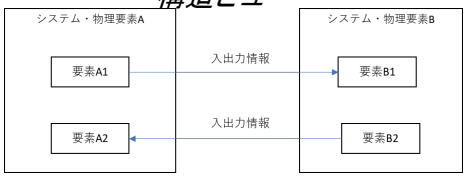
■説明
テスト項目を大/中/小などの項目で項目を分割している。
（仕様書の項番がそのまま大/中/小の項目に使われる場合が多い）
■工夫の必要なポイント2
- 2つの要素が関係する条件、時間経過に関する条件の表現が困難
- 協調動作/並列動作、復旧処理 などの因果関係を表で表現する事が難しい
- タイミングのズレ　などが引き起こす処理順の変更を表で表現する事が難しい

提案手法の概要

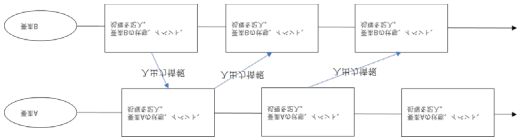
①欠陥情報から発生条件の抽出

【工夫点1】
入出力関係と時系列の視点で
欠陥の発生条件を抽出

構造ビュー

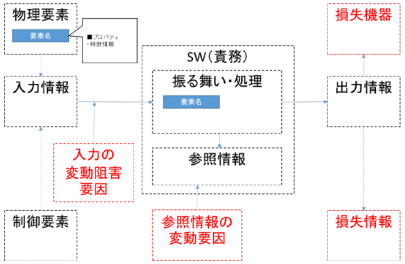


時系列ビュー



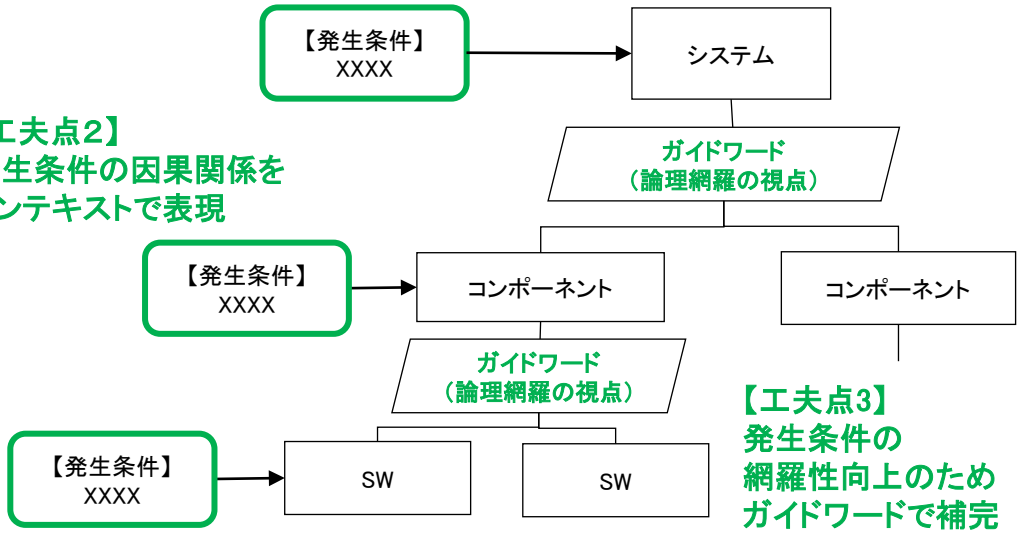
各位置づけで特徴情報を抽出

構造ビュー



②リスクシナリオ発生条件の導出

GSN

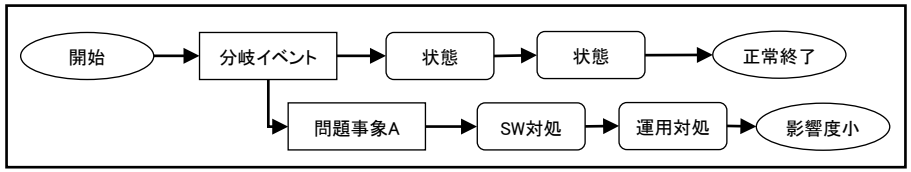


【工夫点2】
発生条件の因果関係を
コンテキストで表現

【工夫点3】
発生条件の
網羅性向上のため
ガイドワードで補完

③リスクシナリオの作成

因果関係と発生条件をシナリオとして表現する



④テスト条件の詳細化

【工夫点4】
時間経過に関する制約の可視化
のため
ガイドワードで補完

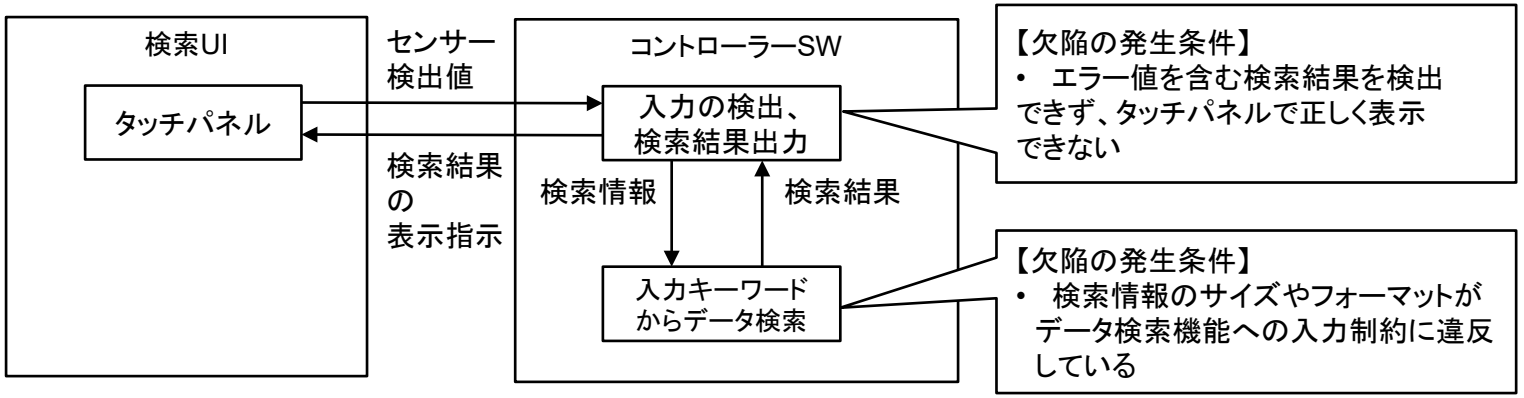
分類	属性	ガイドワード
入力	入力値	異常値、特殊値、なし(NULL)
	入力値域	最大、最小、範囲外、範囲内
	初期値	ゼロ、デフォルト値、未更新
アルゴリズム	時刻変化	変化なし、急激な変化
	積算	飛び値、未積算、一部積算

適用事例

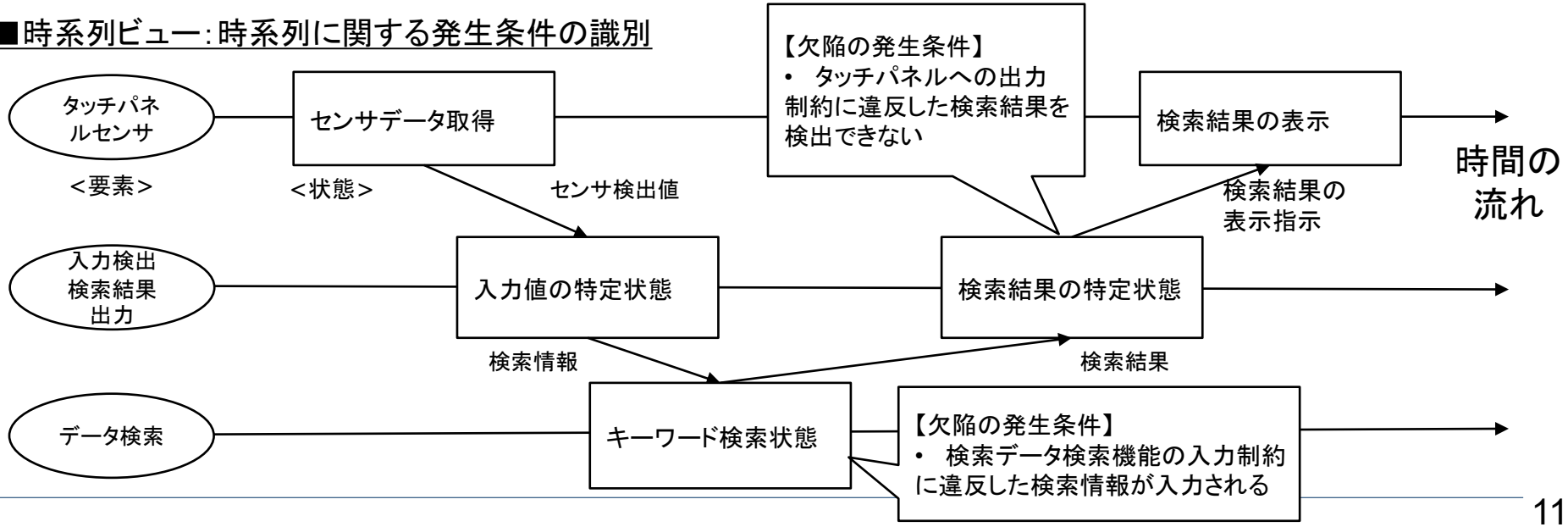
工程①-1：欠陥情報から発生条件の抽出

【工夫点1】
データ入出力関係と時系列の視点（ビュー）を使い、
欠陥の発生条件を識別する

■構造ビュー：複数の要素が関わる発生条件の識別



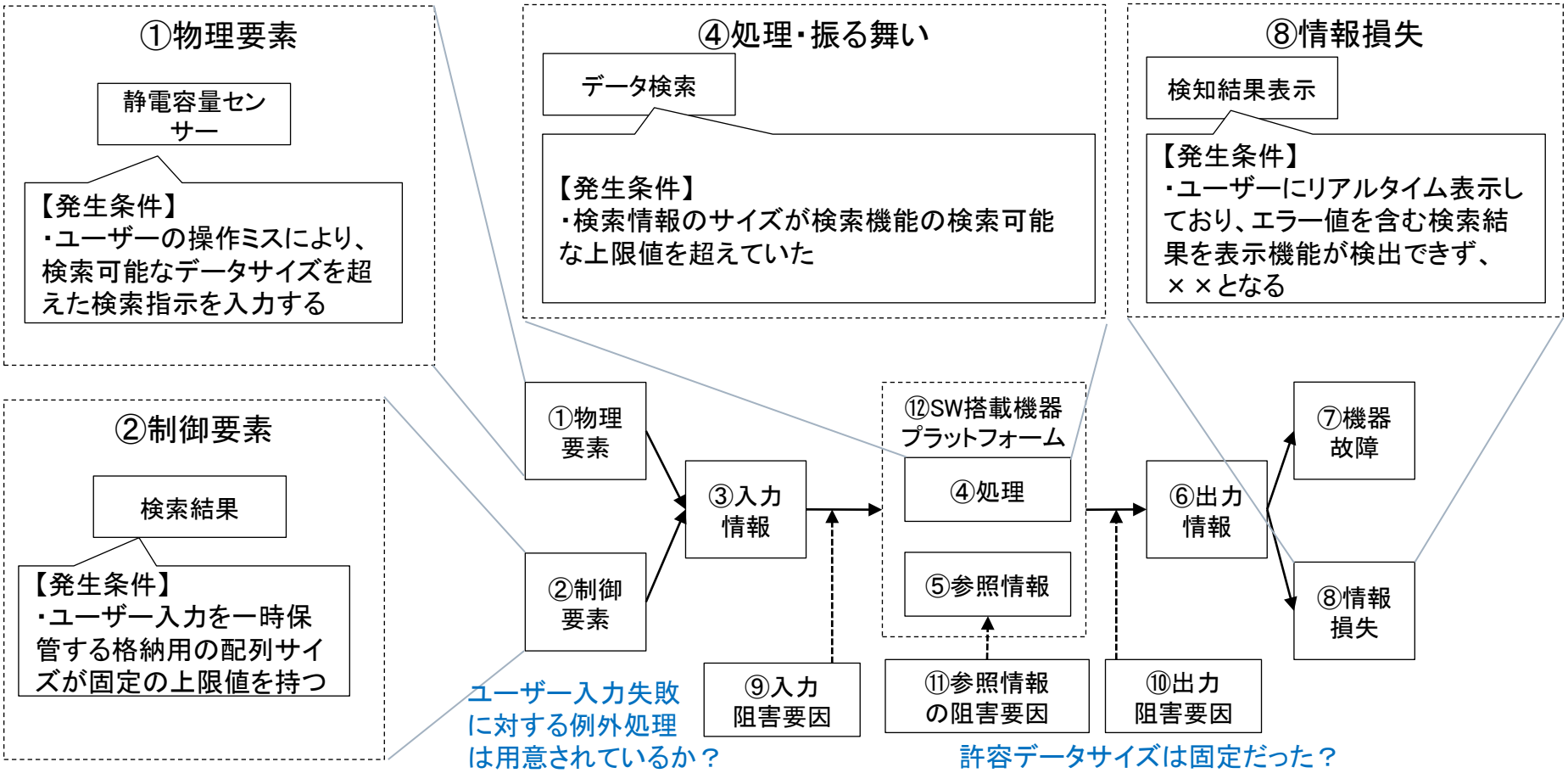
■時系列ビュー：時系列に関する発生条件の識別



適用事例

工程①-2：欠陥情報から発生条件の抽出

【工夫点2】
「構造ビュー」＋「要素の位置づけのガイドワード」で発生条件の漏れがないか確認する



【参考】提案手法により明確化したテスト条件の因果関係例

- 過去の欠陥から読み取った事象間の因果関係の事例は以下となる

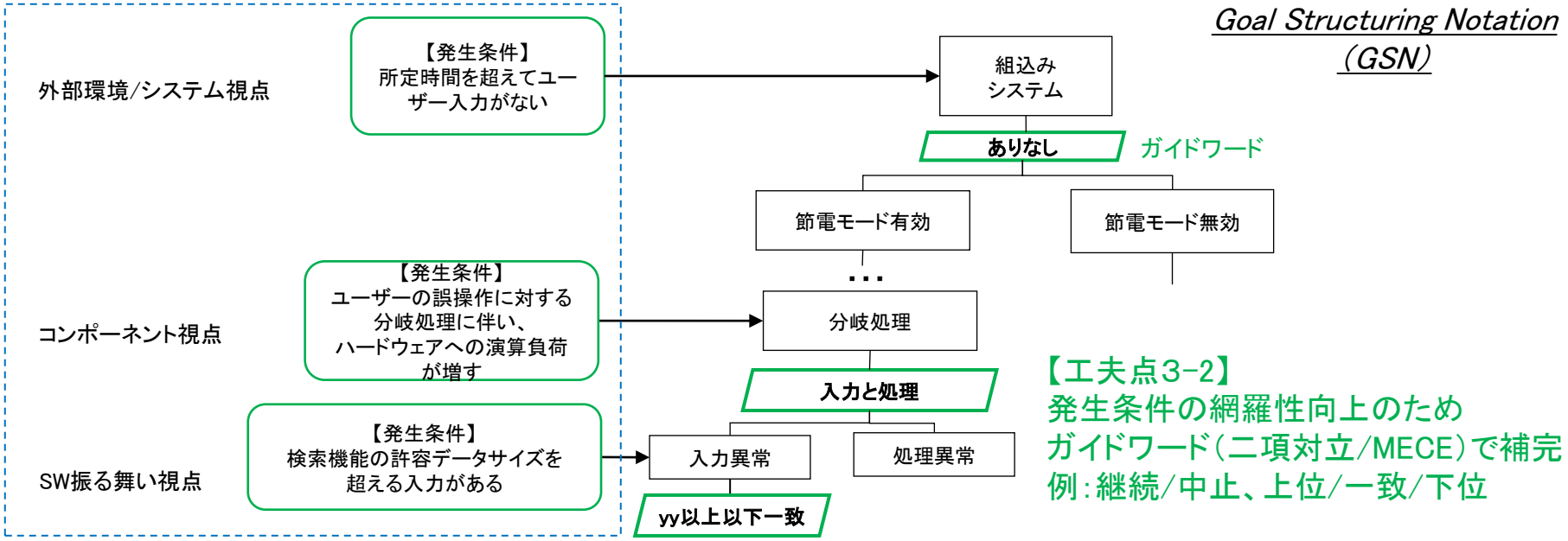
No.	発生した高負荷タスク	各外部インターフェースからどんな入出力があったか？	高負荷タスクがどのハードウェア装置を動かすか？	高負荷タスクに伴い、どのタイミングで代替処理が発生するべきか？
1	過重なデータサイズのロードに伴うデータ保持配列でのバッファオーバーフロー	ユーザーからの過剰な個数のデータ入力	データ保持配列	バッファオーバーフロー検出時
2	外部入力からの書き込みアクセスと、外部表示のための読み出し時のアクセスの連続実行に伴うキャッシュメモリの計算負荷過重	・ユーザーからの表示切替入力の繰り返し	キャッシュメモリ	タイムアウト時など
3	ユーザーからのタッチパネル入力処理とユーザーへの表示出力処理との繰り返しに伴うUI表示機能の計算負荷過重	・表示画面の繰り返し更新	UI表示機能	表示処理とユーザー入力処理とが競合した時
4	電源起動に伴う複数機能での初期化の繰り返しに伴うキャッシュメモリの計算負荷過重	・電源起動	・初期化機能 ・キャッシュメモリ	タイムアウト時など

適用事例

工程②:リスクシナリオの発生条件の導出

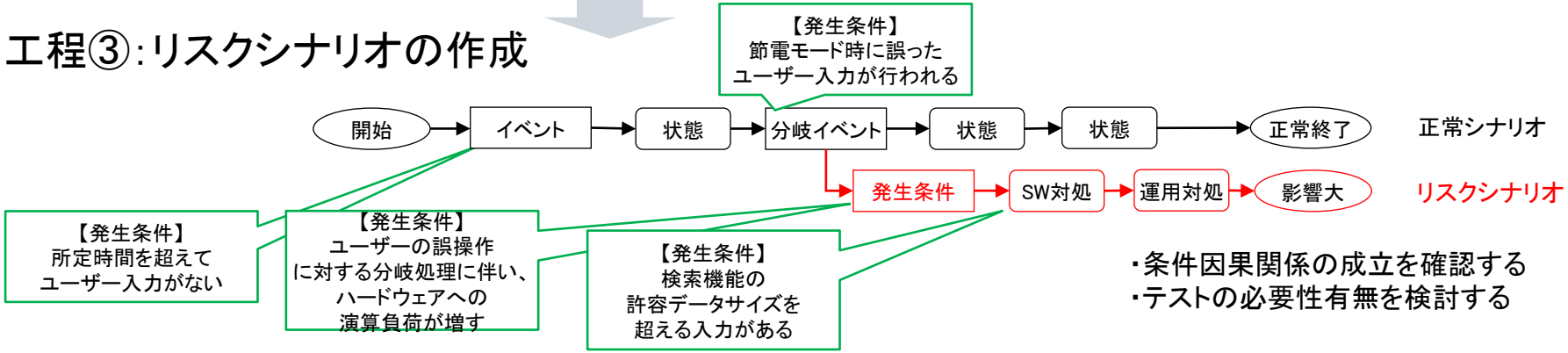
【工夫点3-1】ツリービューを使い、多数の欠陥情報から個別に抽出した発生条件を集約し、
今回のテスト対象の仕様を考慮/変更し整理する

Goal Structuring Notation (GSN)



【工夫点3-2】
発生条件の網羅性向上のため
ガイドワード(二項対立/MECE)で補完
例:継続/中止、上位/一致/下位

工程③:リスクシナリオの作成

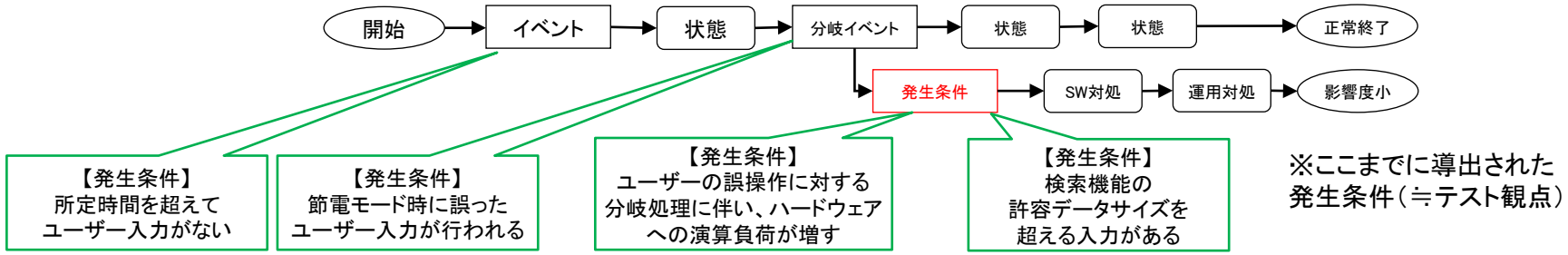


・条件因果関係の成立を確認する
・テストの必要性有無を検討する

適用事例

工程④:テスト条件の詳細化

【工夫点4】
時間経過に関する制約の可視化のため、異常パターンのガイドワードで補完



分類	属性	ガイドワード
入力	入力値	異常値、特殊値、なし(NULL)
	入力値域	最大、最小、範囲外、範囲内
	初期値	ゼロ、デフォルト値、未更新
アルゴリズム	時刻変化	変化なし、急激な変化
	積算	飛び値、未積算、一部積算

■最終的に導出されるリスクシナリオテストの例

“節電モード中に外部から規格外入力が行われ”、“分岐処理に伴いテーブルの保持データが初期化された時”、“テーブルデータの読み出しが初期化と同時に実行されることはないか？”

【実証実験】提案手法の有効性評価

❑ 有効性確認方法：

【実施方法】
1．被験者は、手法を用いず試験ケースを導出する。
2．その後、被験者は手法の手順に従い、過去の欠陥情報を参照して試験ケースを追加する。

【評価指標】

- 手法の適用によって追加されたテストケース数の中で、システムのロバスト性を評価する事のできるテストケース

【実施条件】

- 分析対象の特性が有効性評価に影響しない様、実施条件を設定した（下表）

適用先	分析対象	被験者（分析者 ）	テストケース設計順序
No.1	ユースケース1	第三者検証組織（1名）	No.1～No.5に対し従来手法（※）でテストケースを設計 ↓ No.1～No.5に対し提案手法でテストケースを設計
No.2	ユースケース2		
No.3	ユースケース3		
No.4	ユースケース4		
No.5	ユースケース5		

（※） 従来手法＝「表」や「単一観点」（例：チェックリスト）を使ったテストケースの識別

【実証実験】実験により手法効果を確認した結果

- 結果①：提案手法の適用により、システムのロバスト性を評価するテストケースの追加に成功した
- 結果②：追加したテストを実施し、実際の製品のシステムに対し欠陥を検出する事が出来た

		既存のテストケース		提案手法	
テスト対象 ユースケース	ユースケース概要	テストケース数	検出した欠陥数	追加テストケース	検出した欠陥数
ユースケース１	ユーザーが組込みシステム経由でサーバー上のサービスを起動する	0	0	+6	0
ユースケース２	サーバー経由でリモートから組込みシステムを操作する	41	4	+16	+3
ユースケース３	サーバーから組込みシステムへ記憶データをインポートする	5	0	+14	+1
ユースケース４	組込みシステムが外部機器を操作する	81	3	+17	0
ユースケース5	ユーザーが組込みシステムの記憶データを編集する	71	2	+17	0

【参考】テストケース数全体に対する追加テストケース数の内訳

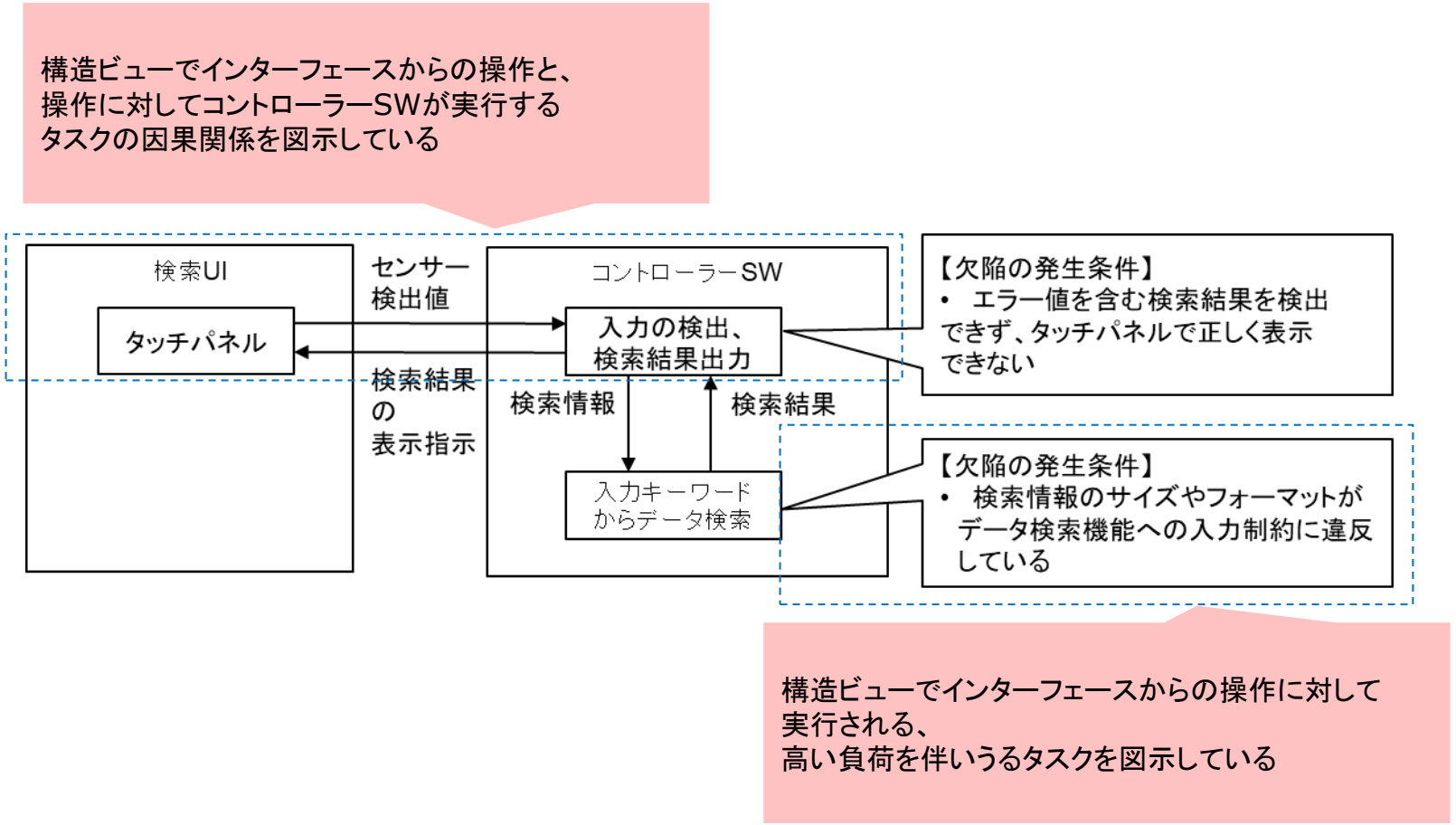
- 確認した内訳：既存テストケース数に対し見つかった検出欠陥数の割合は、追加テストケース数に対し見つかった検出欠陥数の割合と近く、妥当な結果となった

			既存のテストケース		提案手法	
テスト対象ユースケース	ユースケース概要	分類	テストケース数	検出した欠陥数	追加テストケース	検出した欠陥数
ユースケース１	ユーザーが組込みシステム経由でサーバー上のサービスを起動する	システムのロバスト性を評価したテストケース	0	0	+6	0
		それ以外	26	0	0	0
ユースケース２	サーバー経由でリモートから組込みシステムを操作する	システムのロバスト性を評価したテストケース	41	4	+16	+3
		それ以外	89	5	+2	0
ユースケース３	サーバーから組込みシステムへ記憶データをインポートする	システムのロバスト性を評価したテストケース	5	0	+14	+1
		それ以外	7	3	0	0
ユースケース４	組込みシステムが外部機器を操作する	システムのロバスト性を評価したテストケース	81	3	+17	0
		それ以外	0	0	0	0
ユースケース5	ユーザーが組込みシステムの記憶データを編集する	システムのロバスト性を評価したテストケース	71	2	+17	0
		それ以外	0	0	0	0
合計数			320	17	72	4

既存手法でも提案手法でもテストケース数に対する検出欠陥数の割合は概ね近い値

【参考】事例の考察1

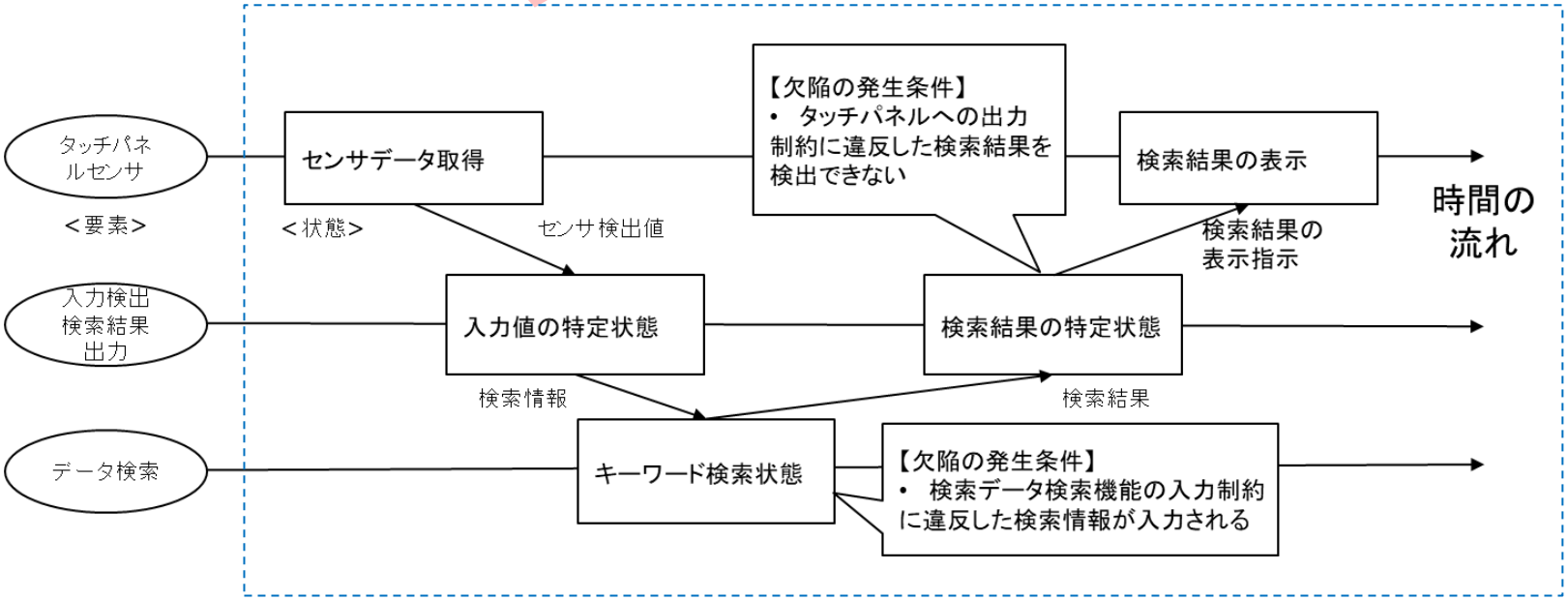
- 考察1：構造ビューから以下を読み取れるか？
  - インタフェースが実行したタスクが因果関係
  - 高い負荷を伴うタスクの発生条件



【参考】事例の考察2

- 考察2：時系列ビューから以下を読み取れるか？
  - インタフェースからの操作に対してタスクの実行される順番
  - 高い負荷を伴うタスクの発生タイミング

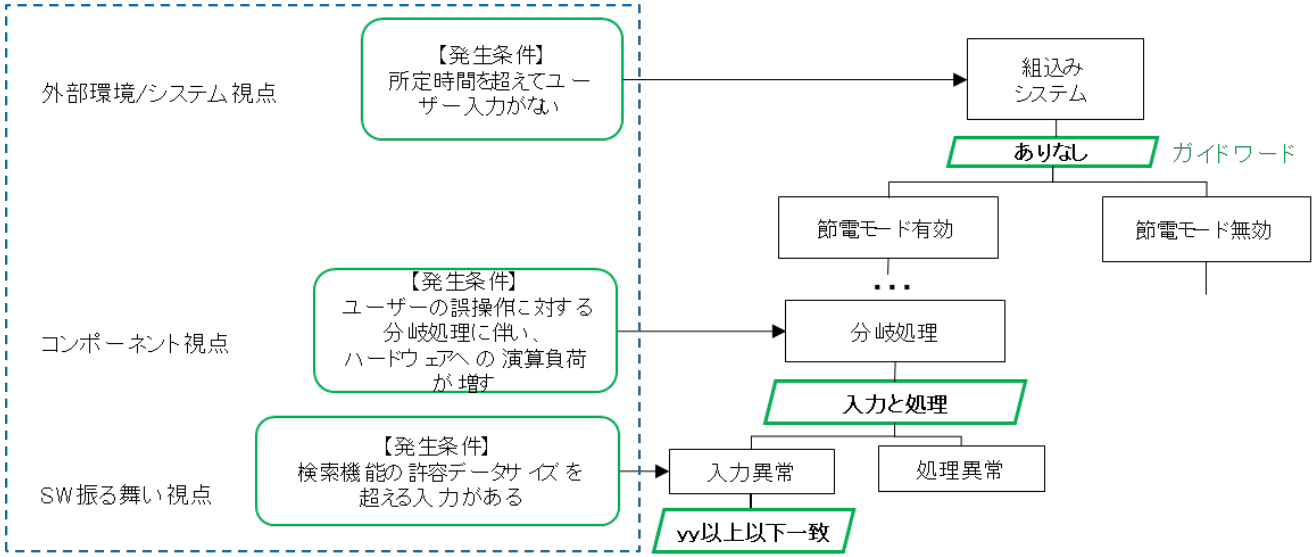
時系列ビューでインターフェースからの操作と、
操作に対してコントローラーSWが実行する
タスクの順番を図示している



【参考】事例の考察3

- 考察3：ツリービューから、以下を読み取れるか？
  ・タスクの実行順が変更される条件

ツリービューで
インターフェースからの操作
に対してタスクの実行順が
変更される条件を
組み合わせで図示している



まとめ

- 結論
  - 提案手法は、過去欠陥の発生条件の識別により、異常時のシナリオテスト設計を支援する事が出来る
  - 提案手法は、GSNを活用することで、動作環境やSW処理などの複数の処理の因果関係を検討する事ができる
  - 提案手法は、ガイドワード適用を促進することで時間経過に関する処理への制約を可視化する事ができる

- 今後の展開
  - 提案手法の適用範囲の拡大、費用対効果の計測
    → ご協力いただける方募集中！
  - ソフトウェアとハードウェアとの相互作用に伴い、
  - シナリオテスト設計からみた、欠陥情報の記載方法の改善や教育支援

ご清聴　ありがとうございました