第19回ZIPCユーザーズカンファレンス向け原稿

FUJITSU
shaping tomorrow with you

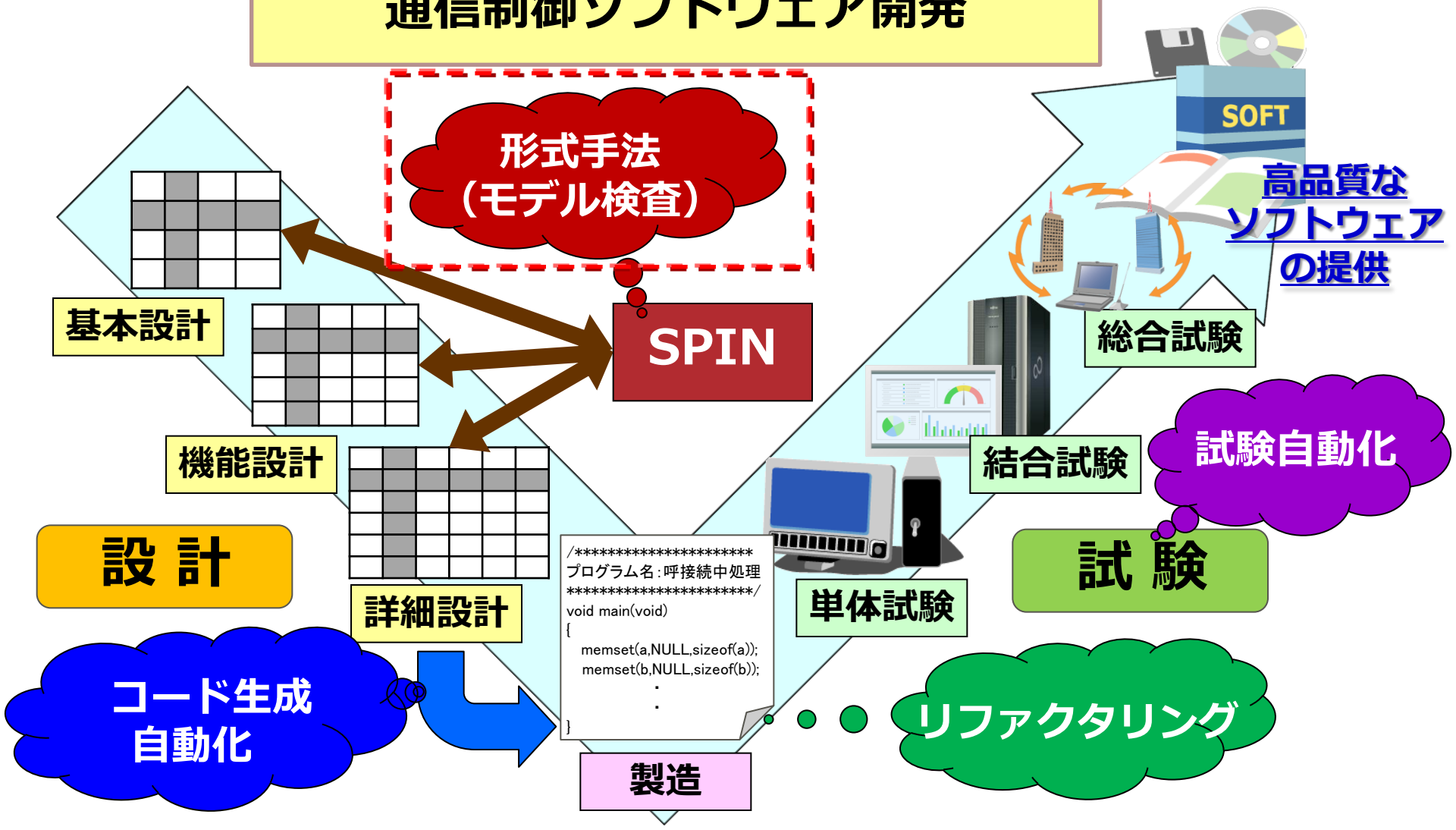
ZIPC状態遷移表の モデル検査適用への取り組み

2014年10月10日
富士通株式会社
共通開発本部）ソフトウェア方式統括部
安岡　大知

富士通の通信制御ソフトウェアに対する取り組み

FUJITSU

膨大な既存資産をベースとした
通信制御ソフトウェア開発



これまでの紹介事例との関係

FUJITSU

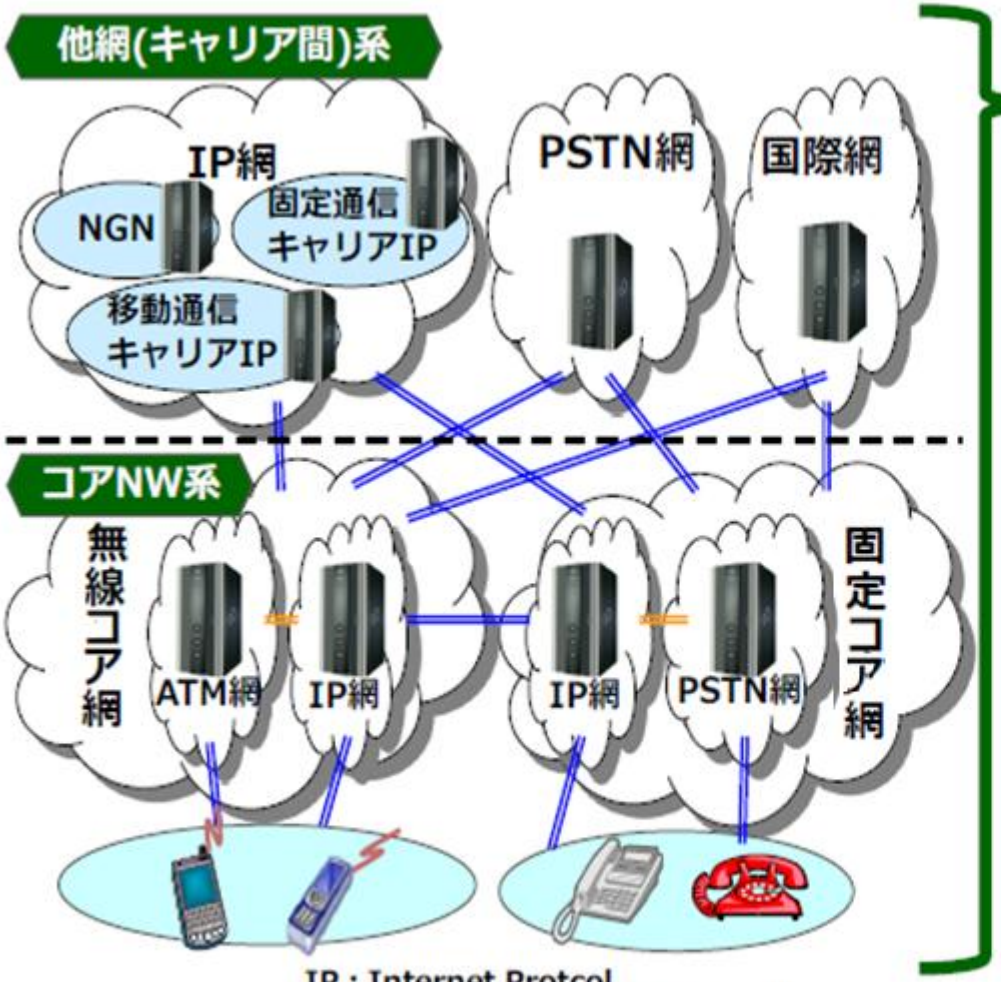
2011年	2012年	2013年	2014年
★ZIPCによるコード生成自動化への取組み (第17回ZIPCユーザーズカンファレンス)			
★ZIPCを活用したモデル検査ツール開発への取組み (第17回ZIPCユーザーズカンファレンス)			
		★ZIPC状態遷移表を使用した ソフトウェア品質向上への取組み (ZIPC Watchers Vol.16)	
			★ZIPC状態遷移表の モデル検査適用への取組み (第19回ZIPCユーザーズカンファレンス)

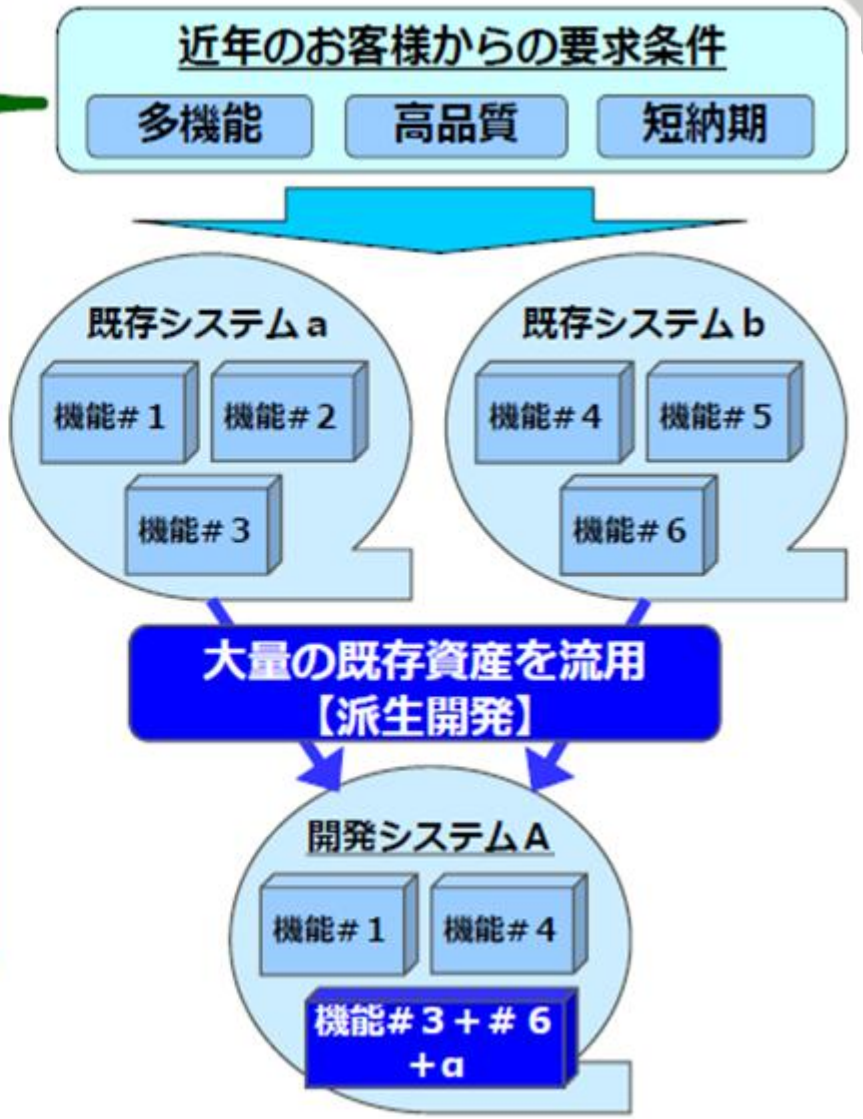
本日の発表の流れ

FUJITSU

1．通信制御ソフトウェア開発の概要
2．状態遷移設計における課題
3．形式手法・モデル検査について
4．SPINのプロジェクト適用の課題
[解決策紹介]
5．SPIN検査支援ツール
6．状態爆発の防止
7．SPIN適用難易度の把握
8．プロジェクト適用状況
9．プロジェクト適用事例
１０．最後に

1. 通信制御ソフトウェア開発の概要(1/2)

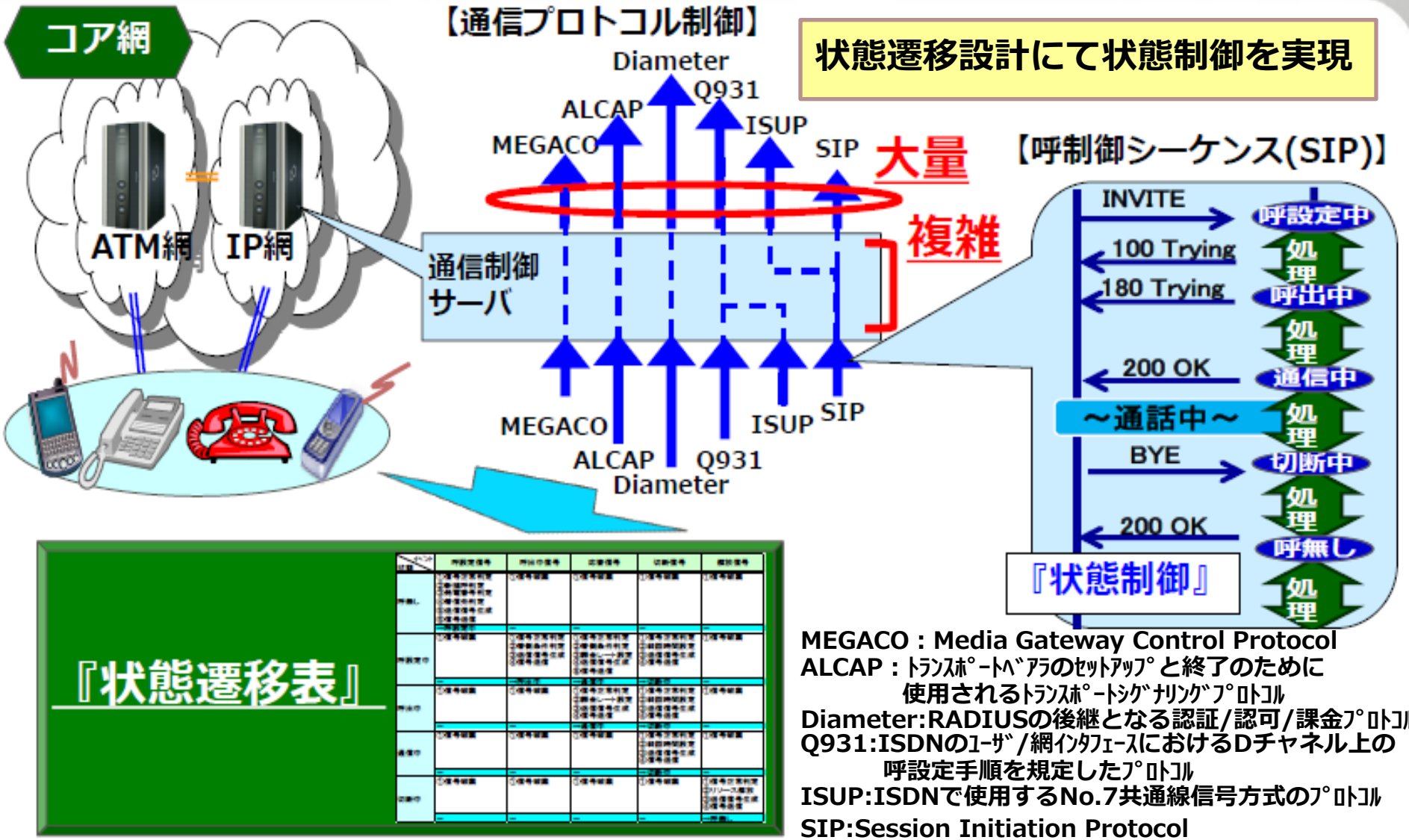




お客様からの要求に派生開発で対応

1. 通信制御ソフトウェア開発の概要(2/2)

FUJITSU



MEGACO : Media Gateway Control Protocol
ALCAP : ﾄﾗﾝｽﾎﾟｰﾄﾍﾞｱﾗのｾｯﾄｱｯﾌﾟと終了のために使用されるﾄﾗﾝｽﾎﾟｰﾄｼｸﾞﾅﾘﾝｸﾞﾌﾟﾛﾄｺﾙ
Diameter:RADIUSの後継となる認証/認可/課金ﾌﾟﾛﾄｺﾙ
Q931:ISDNのﾕｰｻﾞ/網ｲﾝﾀﾌｪｰｽにおけるDチャネル上の呼設定手順を規定したﾌﾟﾛﾄｺﾙ
ISUP:ISDNで使用するNo.7共通線信号方式のﾌﾟﾛﾄｺﾙ
SIP:Session Initiation Protocol

大量/複雑な状態制御の網羅性を状態遷移表で担保

2. 状態遷移設計における課題

FUJITSU

■ 状態遷移表
設計工程で多くの時間をかけ作成や調査した状態遷移表
状態制御の網羅性を担保し易い反面、規模が大きくなりやすい
(状態/イベント組合わせが数万以上の場合も)

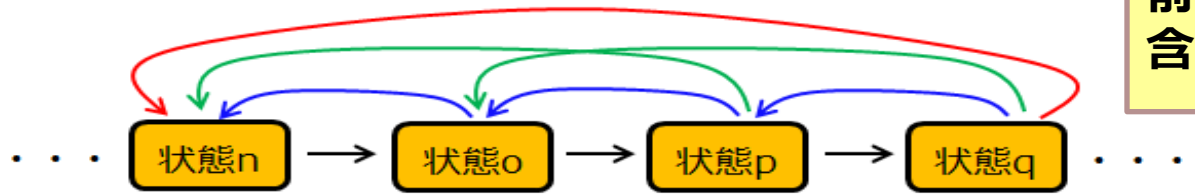
状態＼イベント	呼設定信号	呼出中信号	応答信号	切断信号	解放信号
呼無し	①信号正常判定 ②新規呼判定 ③発電番号判定 ④着信先判定 ⑤送信信号生成 ⑥信号送信	①信号破棄	①信号破棄	①信号破棄	①信号破棄
	→呼設定中	—	—	—	—
呼設定中	①信号破棄	①信号正常判定 ②着側条件判定 ③送信信号生成 ④信号送信	①信号正常判定 ②着側条件判定 ③課金レート設定 ④送信信号生成 ⑤信号送信	①信号正常判定 ②終話時間設定 ③送信信号生成 ④信号送信	①信号破棄
	—	→呼出中	→通信中	→切断中	—
呼出中	①信号破棄	①信号破棄	①信号正常判定 ②課金レート設定 ③送信信号生成 ④信号送信	①信号正常判定 ②終話時間設定 ③送信信号生成 ④信号送信	①信号破棄
	—	—	→通信中	→切断中	—
通信中	①信号破棄	①信号破棄	①信号破棄	①信号正常判定 ②終話時間設定 ③送信信号生成 ④信号送信	①信号破棄
	—	—	—	→切断中	—
切断中	①信号破棄	①信号破棄	①信号破棄	①信号破棄	①信号正常判定 ②リソース解放 ③送信信号生成 ④信号送信
	—	—	—	—	→呼無し

- 作成状態遷移表の検証漏れ
- 状態遷移表変更時の影響調査漏れ

人手での調査・検証による全遷移ルートの確認は限界

⇒ 対象の絞り込み(**調査・検証漏れ発生**)

前状態に戻る遷移ﾙｰﾄを含めた確認ｹｰｽ数は膨大



課題：膨大な遷移ルートの効率的/網羅的検証

3-1. 形式手法との出会い

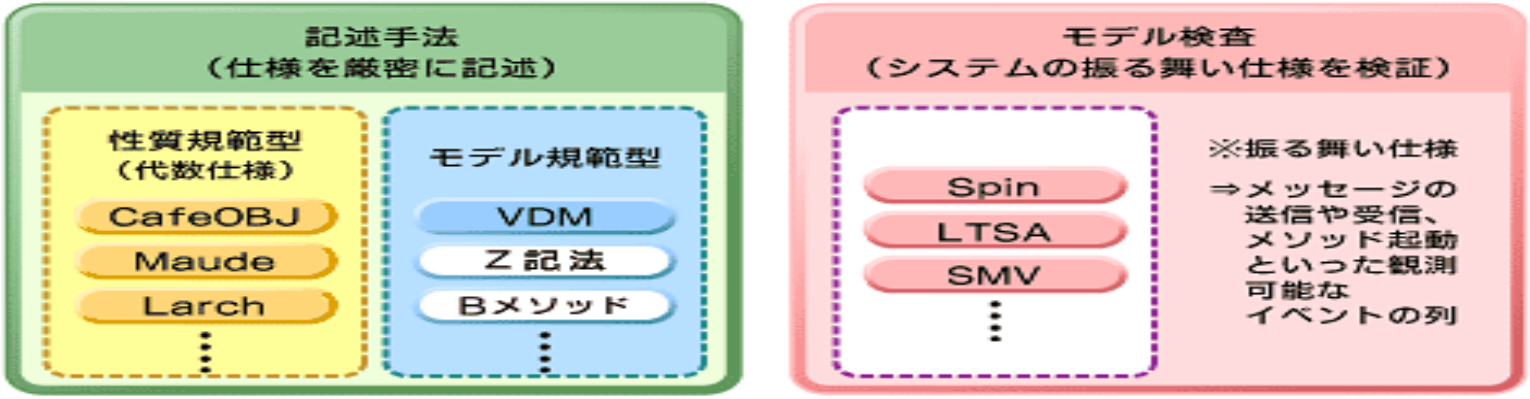
FUJITSU

■ 問題発生と改修コスト

設計工程以降に検出した問題解決コストは後工程に行くほど大きくなる
(ソフト開発データより)

膨大な遷移ルートの効率的/網羅的検証を上流工程で実施して品質確保
⇨ コスト削減

■ 形式手法



「仕様」を正確、詳細に記述・検証することを目的とした手法。
・仕様を正確に記述するための記述手法
・システムの振る舞い仕様を検証する検査手法（モデル検査）

形式手法 モデル検査の評価を実施

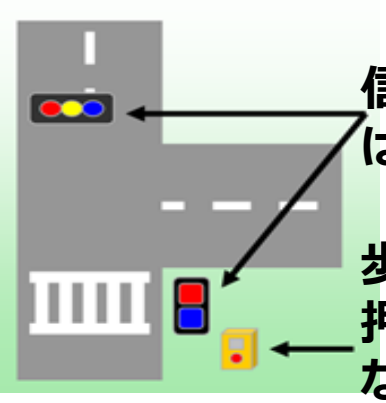
3-2. モデル検査の導入

FUJITSU

■ 形式手法 モデル検査

➢ モデル検査：設計工程の検証手法の1つ

「今」、「将来」など時間的な概念を含んだ性質をアルゴリズム的に検証



信号は同時に青になることはないか

歩行者信号のボタンを押せばいつか必ず青になるか

➢ 通信制御ソフトウェア

大量のイベント/状態制御を行うため、開発生産物は状態遷移表がメイン

各種検査器で状態遷移表との親和性、安定度、速度、完成度の高さからSPINを選定

<SPIN>
状態遷移表と検査観点を入力すると全遷移ルートが検証観点に合致しているか検査器にて自動検証を実施

解決策：SPINによる状態遷移表の効率的/網羅的検証

単純にモデル検査SPINを適用すれば、
状態遷移設計の品質向上・コスト削減に
効果を発揮するのか？

効果が出るまでに様々な準備が必要と考えます。
以降で、我々の取り組みをご紹介します。

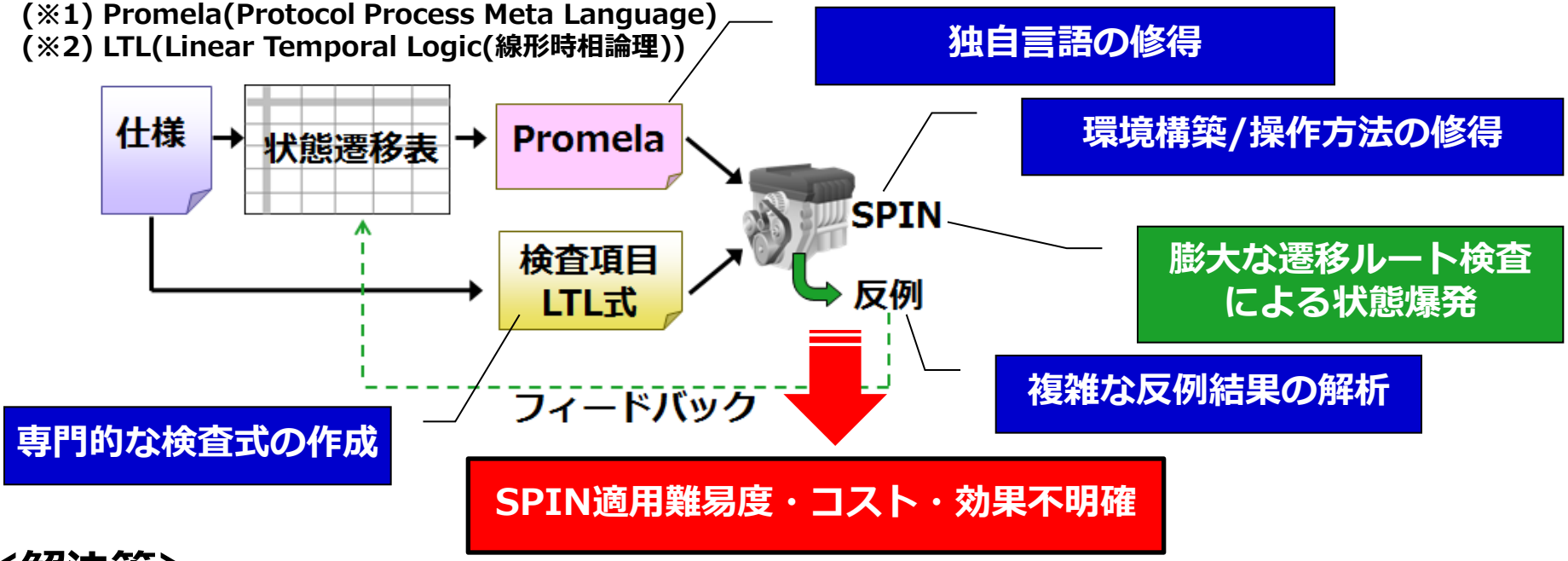
4．SPINのプロジェクト適用の課題

FUJITSU

■ モデル検査SPIN(Simple Promela Interpreter)

1．仕様から、状態遷移表作成・検査項目抽出
2．状態遷移表をPromela言語/検査項目をLTL式で記載。SPINにインプット/実行
3．実行結果として、状態遷移表が検査項目に合致しなかった場合、反例を出力
4．反例解析し、不具合であれば状態遷移表へフィードバック

(※1) Promela(Protocol Process Meta Language)
(※2) LTL(Linear Temporal Logic(線形時相論理))



＜解決策＞

①SPIN検査支援ツールの作成・導入

②状態爆発の防止

③SPIN適用難易度・コスト・効果の把握

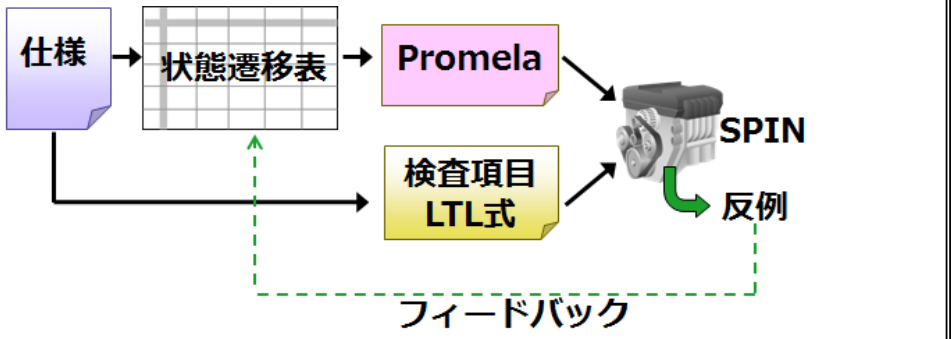
＜解決策＞

①SPIN検査支援ツールの作成・導入

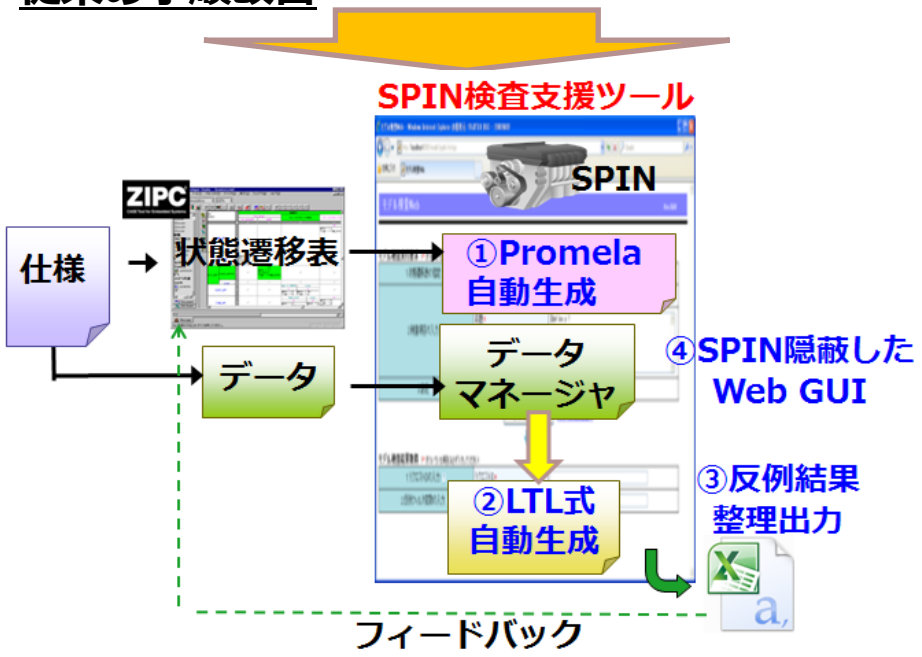
②状態爆発の防止

③SPIN適用難易度・コスト・効果の把握

5．SPIN検査支援ツールの作成・導入



従来の手順改善



支援ツール導入による効果

■ **専門言語の修得不要**

ZIPC状態遷移表からPromela生成

■ **数学的な知識の修得不要**

ソフトウェアの各種データ定義情報から
状態遷移時のデータ操作のLTL式生成

■ **反例結果の効率的な解析**

反例ログを状態/イベント毎に表形式整理
(可読性向上)

■ **SPIN環境構築/SPIN操作の隠蔽**

Webサーバへ構築/モデル検査を一画面
操作可能なGUI作成
(簡易なGUI操作によるモデル検査)

SPIN専門ノウハウ／スキル修得不要

Promela自動生成にZIPCを用いた理由

状態遷移表のPromela言語化はモデル検査適用の大きな障壁

Promela言語自動生成の検討

- **従来の状態遷移表**
  - Microsoft Office Excel使用

状態遷移表記述レベル/記述法が
プロジェクト毎に異なる。

⇨ 変換ルール複雑

- **解決策**
  - **ZIPCの採用**

ZIPC状態遷移表

- 状態遷移
- 条件分岐
- タスク間イベント発行
- タスク内イベント発行
- 四則演算
- 変数定義

状態遷移表記述レベル/記述法統一
出力フォーマット：XML

⇨ 変換ルール簡素

ZIPC状態遷移表からPromela言語の自動生成を実現

＜解決策＞

①SPIN検査支援ツールの作成・導入

②状態爆発の防止

③SPIN適用難易度・コスト・効果の把握

6-1. 状態爆発の防止における課題

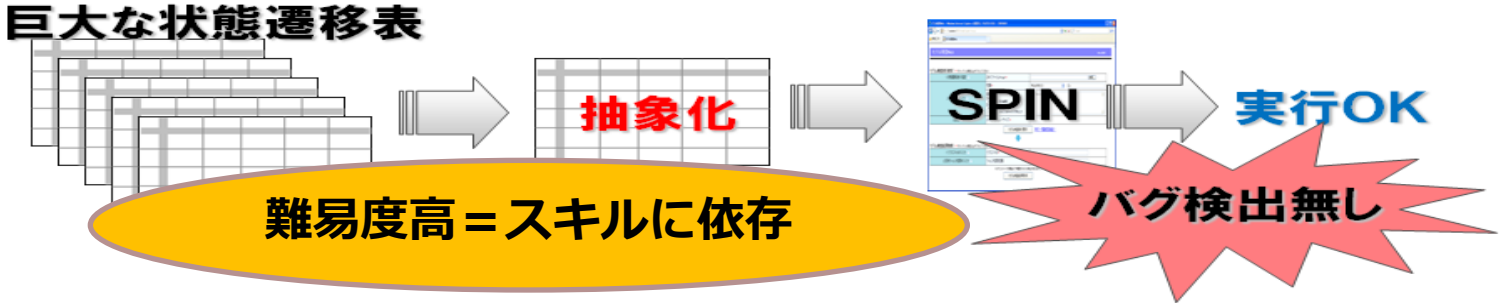
FUJITSU

■ 状態爆発

検査において探索すべき状態数が膨大な場合、コンピュータの記憶容量の限界によって検査器が動作しなくなる。

➢ 状態遷移表の抽象化による状態爆発の防止

・状態/イベント/データの粒度を粗くする
・不要なイベント/状態/データを削除する



後工程に問題流出
(SPIN適用における効果なし)

→

やみくもに
抽象化するのはダメ！！

課題：問題検出するための抽象化手順(検査ターゲットの決定)

6-2. 状態遷移表の検査ターゲットの決定

■ 各種通信制御ソフトウェアの状態遷移設計に関する流出問題分析

＜流出原因の特徴＞

状態切替時のデータ操作誤り
(過渡状態で競合/準正常/異常発生)

以降の状態遷移ルートで論理矛盾発生

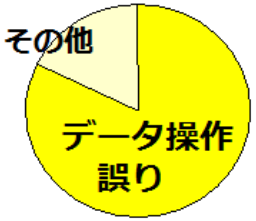
状態AからB遷移中のA切替

遷移の流れ	A	B	A
データ1	ON	ON	ON
データ2	ON	ON	ON
データ3	ON	ON	OFF

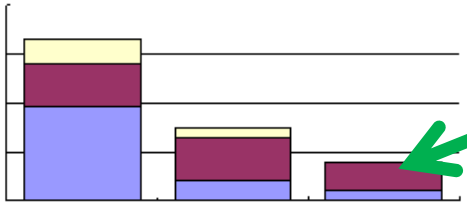
同じ状態Aでデータ内容が異なる

＜あるプロジェクト例＞

トータル流出問題



流出問題発生時期



・データ二重設定/二重解放
・未設定データの参照
(データ操作に関する問題が発生し続けている)

＜検査ターゲット＝データ操作＞

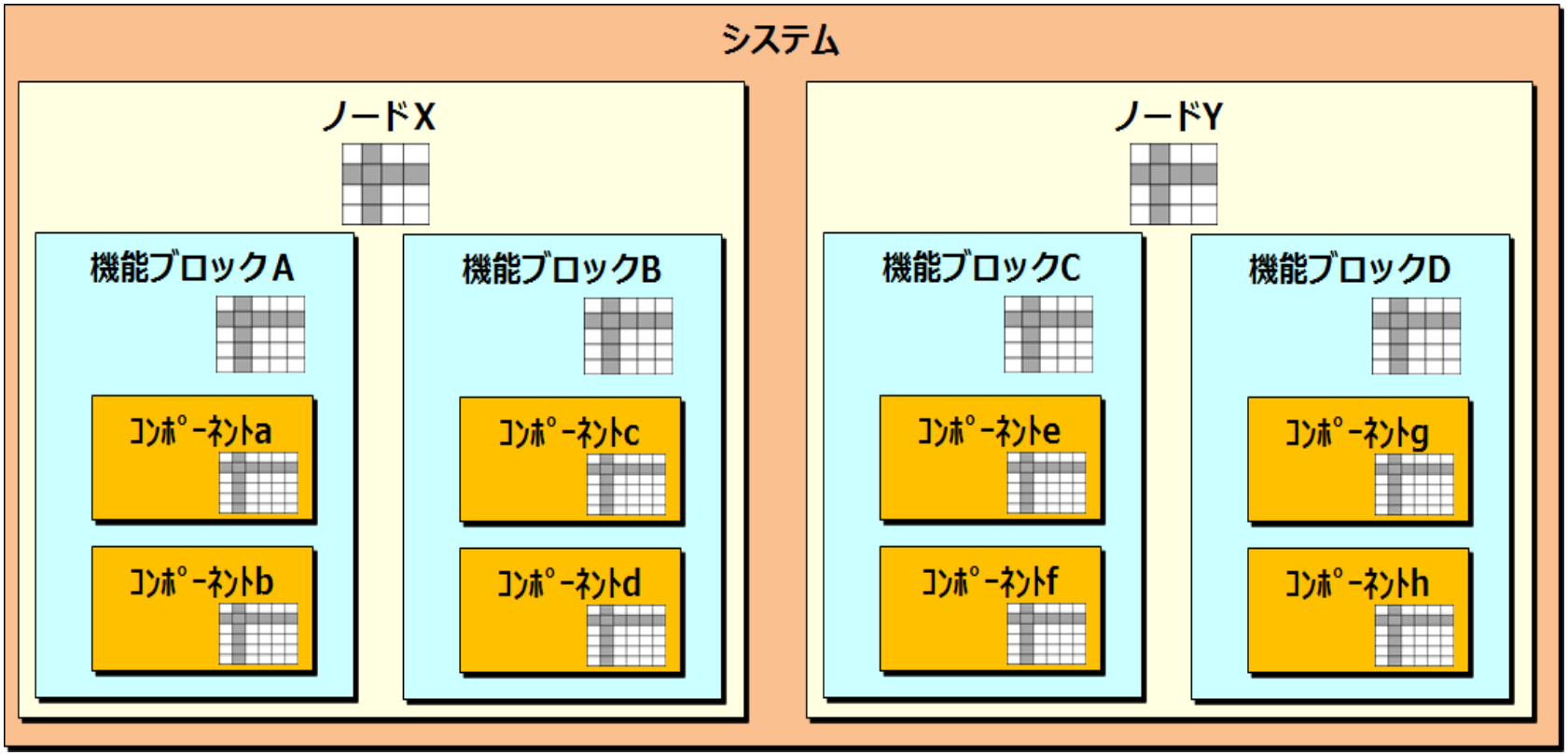
検査ターゲット決定により状態遷移表の粒度を定める。
→状態爆発の防止と問題検出の両立

解決策：データ操作に着目したモデル検査による状態爆発の防止

6-3. 状態爆発の防止例①

データ操作に着目したモデル検査にて状態爆発のすべてを回避することはできません。
検査ターゲットの決定に加えて実施する状態爆発の防止例をご紹介します。

＜防止例①＞
システム状態遷移表を分割したモデル検査



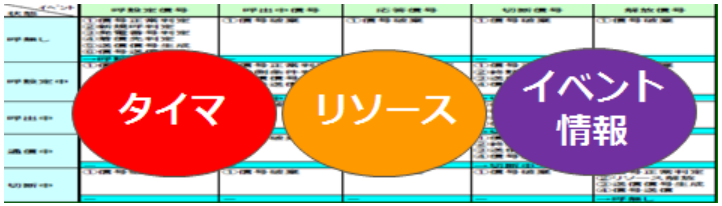
システムの全データ操作を一度に検査するのではなく、ノード/機能ﾌﾞﾛｯｸ/
コンポーネントといった小さな纏まりに分割してそれぞれを検査実施

6-4. 状態爆発の防止例②

FUJITSU

<防止例②>
ノード/機能ブロック/コンポーネントといった小さな纏まりの状態遷移表を分割したモデル検査

タイマ/リソース/イベント情報の操作が記述された状態遷移表



タイマ/リソース/イベント情報の操作毎に状態遷移表を分割

**全てのデータ操作を一度に検査するのではなく、
データ種別毎に分割してそれぞれを検査実施**

6-5. 状態爆発の防止例③

FUJITSU

＜防止例③＞

ソフトウェアメトリクスを活用したモデル検査

－あるプロジェクト例－

```
/************************
プログラム名：呼接続中処理
************************/
void main(void){
  memset(a,NULL,sizeof(a));
  memset(b,NULL,sizeof(b));
                .
                .
}
```

ソースコードメトリクス分析結果

ファイル	関数	結合度
AAA.C	aaa_1_func()	7000
AAA.C	aaa_2_func()	6500
BBB.C	bbb_1_func()	5000
AAA.C	aaa_3_func()	4500
CCC.C	ccc_1_func()	3000
CCC.C	ccc_2_func()	2000
・・・	・・・	・・・
BBB.C	bbb_2_func()	1
AAA.C	aaa_4_func()	1

メトリクス結果にて「ある閾値」以上の結合度を持つ関数内のデータ操作は、閾値以下の関数内のデータ操作と比べて問題流出頻度が高い傾向があった。

(結合度：モジュール間の結び付きや依存の強さ)

＜結合度が高い関数内のデータ操作＞

- 問題潜在の可能性が高い
- 流用／派生開発時に問題混入しやすい

既存状態遷移表において、全てのデータ操作を検査するのではなく、ある閾値以上の結合度の関数内で扱うデータ操作に絞って検査実施

＜対策＞

①SPIN検査支援ツールの作成・導入

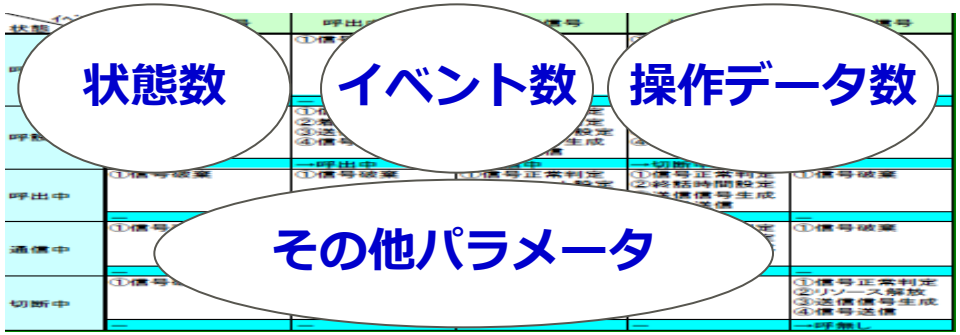
②状態爆発の防止

③SPIN適用難易度・コスト・効果の把握

7．SPIN適用難易度・コスト・効果の把握

FUJITSU

■ SPIN適用難易度・コスト・効果チェックツールの作成



各種パラメータとモデル検査サーバの処理性能、過去のSPIN検査実績を元に、検査対象の状態遷移表の複雑度を点数化するツール作成。
SPIN適用難易度/状態爆発対策要否/費用対効果を点数で判断可能

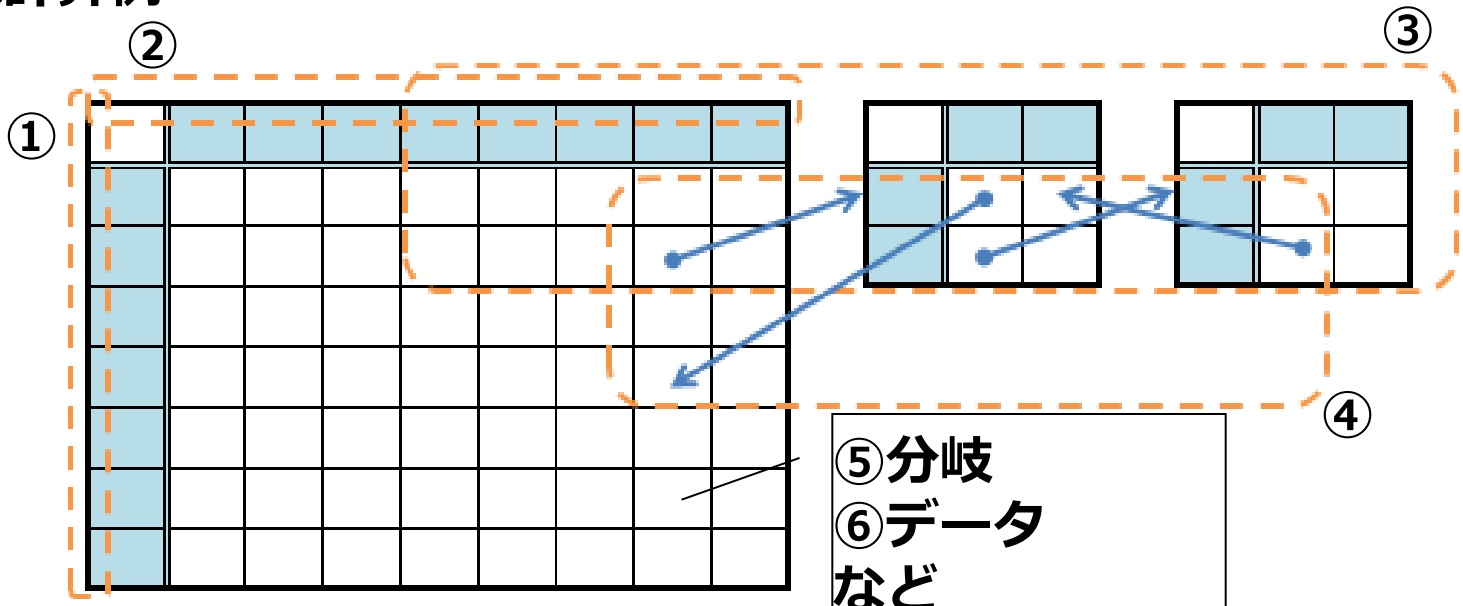
点数	適用難易度	状態爆発対策	導入コスト	導入効果
30点以下	低	不要	低	低
31点～65点	中	不要	中	有
66点以上	高	必須	高	有

プロジェクト適用前の定量的なSPIN適用難易度/費用対効果の把握

参考：SPIN適用難易度・コスト・効果ﾁｪｯｸﾂｰﾙ

FUJITSU

■ 点数計算例



①イベント数
②状態数
③階層数
④状態遷移表間の遷移数
⑤アクションセル内分岐数
⑥アクションセル内データ数
など

・カテゴリ毎の数をツールに入力
・カテゴリ入力数と閾値を比較し、
「大=10点」「中=5点」「小=3点」を決定
・各カテゴリの重み付けを踏まえた総和出力

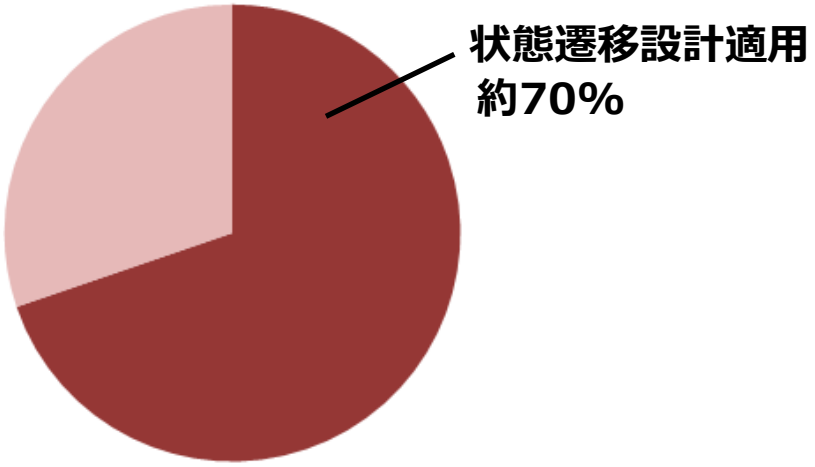
状態遷移表の各パラメータを元に点数を算出

8．プロジェクト適用状況

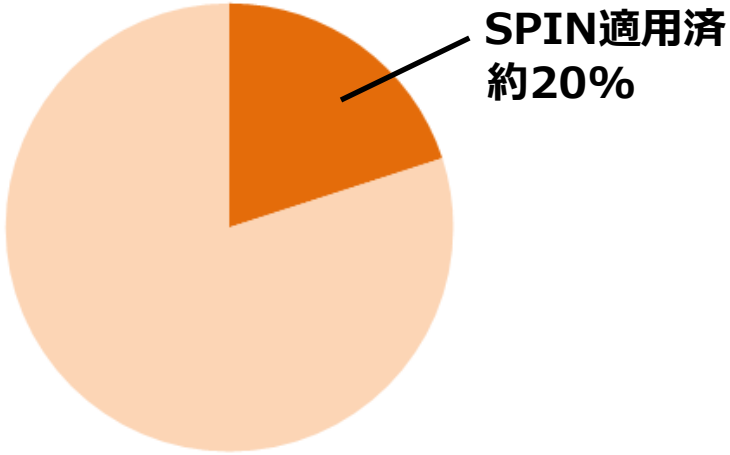
FUJITSU

■ プロジェクト適用状況

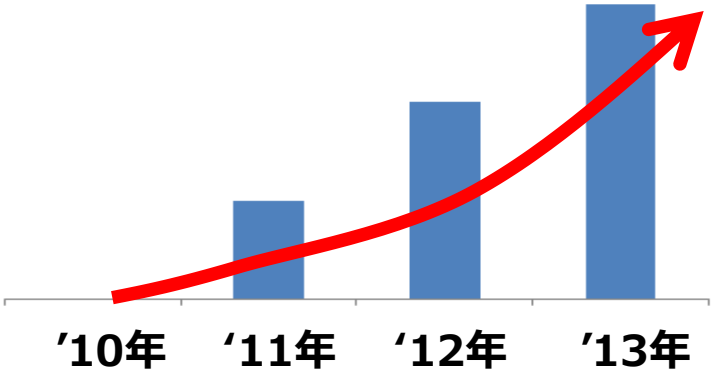
通信制御ソフトプロジェクト



状態遷移設計適用プロジェクト



SPIN適用済プロジェクト推移



- 既存状態遷移表の品質強化，エンハンス開発，新規開発に適用

  →適用プロジェクトの潜在問題や新規混入問題検出に有効を確認

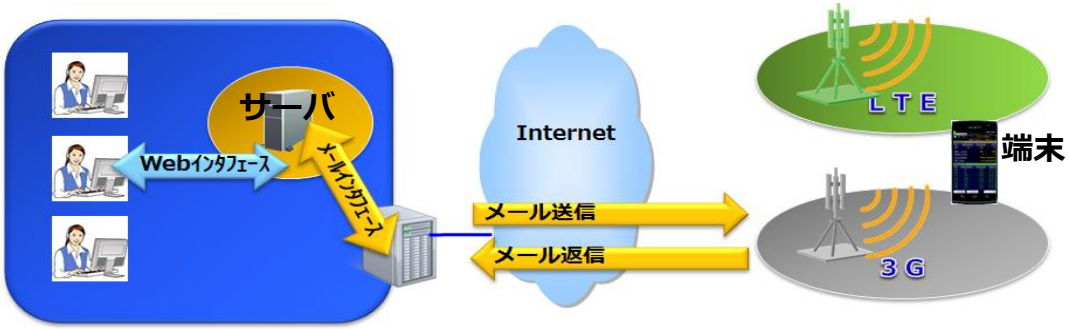
SPIN適用プロジェクト拡大中

9．実プロジェクト適用事例(1/2)

FUJITSU

■ プロジェクト特徴

ｻｰﾊﾞ/端末のﾒｰﾙｲﾝﾀﾌｪｰｽによる状態遷移中に各種ｶﾞｰﾄﾞﾀｲﾏ設定し、制御に不具合があってもタイマT.O.契機でリカバリ制御実施



■ SPIN適用難易度チェックツール結果

得点＝５０（状態爆発対策不要）

■ SPIN検査例

タイマ設定/停止の妥当性(二重設定, 二重停止等の有無)検査実施

■ 検出問題例

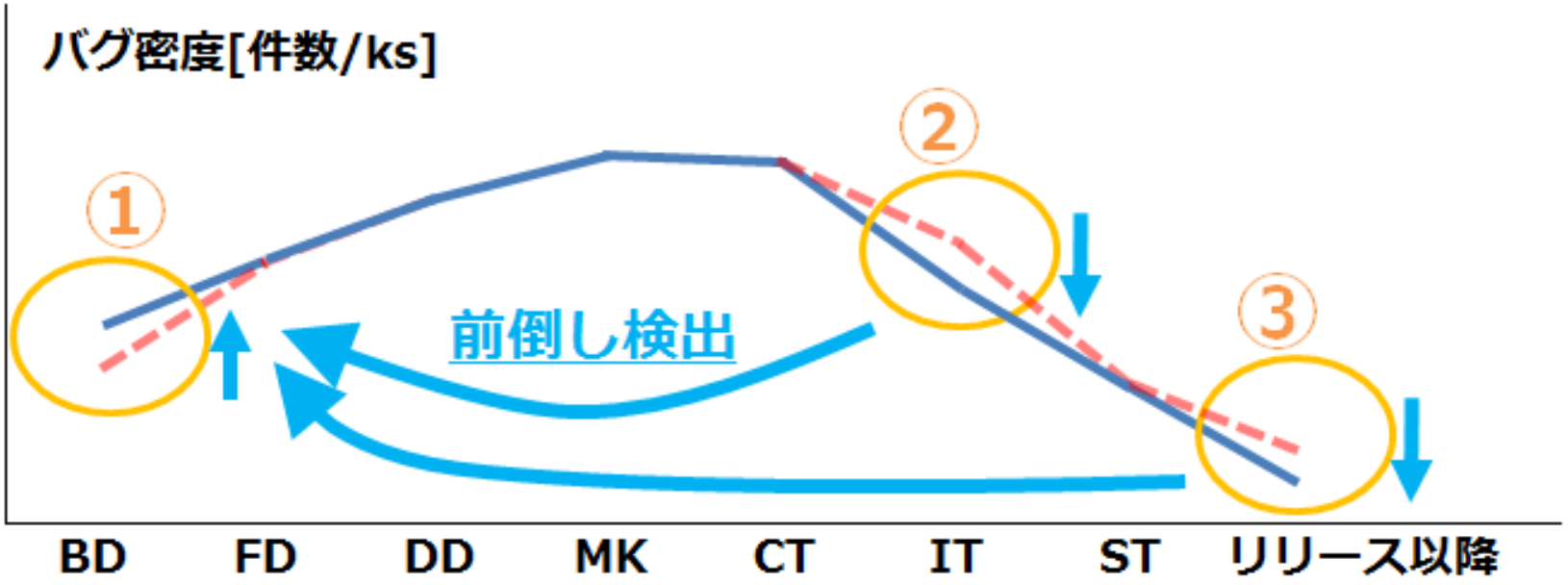
①ある遷移ルートでタイマ設定後、停止前に再度設定するケースを検出
→設定すべきタイマ誤りを検出

②ある遷移ルートでタイマ停止後、再度停止するケースを検出
→検査ターゲットであるタイマから状態遷移誤りを検出
（タイマ停止後の状態遷移誤りにより停止済のタイマを再度停止）

9．プロジェクト適用事例(2/2)

FUJITSU

■ プロジェクト適用効果



SPIN検証による実績(青線)

SPIN検証しなかった場合(赤線)

①BD工程で問題潜在
②IT試験流出
③リリース以降流出

設計工程でのSPIN検査適用による問題解決コスト削減

１０．まとめ

FUJITSU

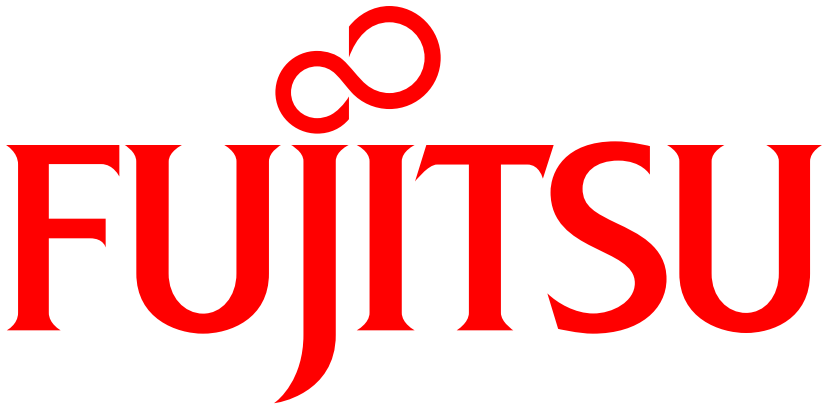
①モデル検査の適用
専門スキルを不要とする検査手順確立／状態爆発への対策が必要
(Promela言語の自動生成は必須(ZIPC活用にて現実的な仕様で実現))

②状態爆発への対策
明確なモデル検査ターゲットを元にした抽象化の実施が必要

③抽象化において注意した方がよいポイント
・作業者スキルになるべく依存しない抽象化手順の確立
・抽象化後の状態遷移表は誰が見ても理解できる必要がある
・本来の目的を見失わない
(状態爆発の対策が目的ではなく、品質向上が目的)

上記クリア後の効果(品質向上／コスト削減)は高いと考えます

④適用者の所感
一度仕組みを作ると、繰り返して状態遷移設計の効率的／網羅的検証に活用が可能であるため、リグレッション検証やアジャイル開発に効果が高いと思われるため、今後、使用用途拡大に取組む予定。



shaping tomorrow with you