

モデル検査・エミュレーション技術の融合による
組込み機器のためのテスト実行環境の研究

－ テスト実行環境の構築と製品適用によるテスト効率化の評価 －

黒岩 丈瑠

目次

第 1 章	序論	7
1.1	研究の背景	7
1.2	研究の目的	13
1.3	本論文の構成	13
第 2 章	SPLE における回帰テストの現状と課題	15
2.1	問題 –回帰テスト工数の爆発的增加–	15
2.2	従来研究 –テスト支援技術概説–	21
2.3	本研究で解決する課題	28
第 3 章	組込み機器のための統合テスト実行環境	31
3.1	本研究のアプローチと新規性	31
3.2	統合テスト実行環境概要	33
第 4 章	モデル検査技術とエミュレーション技術の融合	43
4.1	モデル検査ツールによる実行テストの実現	43
4.2	実験	50
4.3	考察	54
第 5 章	エミュレーション技術を活用したテスト関連作業の効率化	56
5.1	実機–エミュレータ間協調・エミュレータ–エミュレータ間協調の実現	57
5.2	エミュレータ入出力/通信の可視化	60
5.3	実験	62
5.4	考察	68
第 6 章	テストケースの部品化による実行可能なテストケースの資産化支援	72
6.1	テスト手順・テスト構成への部品化	73
6.2	実行可能テストケースの自動生成	74

6.3	ビル空調システム製品向けテストケース管理リポジトリ・実行可能テストケース生成ユニット	78
6.4	実験	84
6.5	考察	86
第7章	製品適用によるテスト効率化の評価	89
7.1	統合テスト実行環境の構成	89
7.2	実験	91
7.3	考察	92
第8章	結論	95
8.1	本研究の成果	95
8.2	本研究の展望	97
	謝辞	100
	参考文献	102
	公表済み研究成果	110

目次

1.1	ビル空調システムの設置形態（オフィスビルの場合）	8
1.2	マーブル化したビル空調システム製品の例：管理コントローラ	11
1.3	ビル空調システム製品の SPLE で回帰テスト工数の爆発的増加が発生した流れ	12
1.4	あるビル空調システム製品の開発工数の変移（概算）	12
2.1	ビル空調システム製品における回帰テストの手順	19
2.2	ビル空調システム製品の従来のテスト環境	21
2.3	統一された仕組みで扱えることの効果	29
2.4	本研究の提案：統合テスト実行環境	30
3.1	ビル空調システム製品の構成（タイプ 1）の模式図	34
3.2	ビル空調システム製品の構成（タイプ 2）の模式図	35
3.3	統合テスト実行環境の構成・機能	37
3.4	テスト手順とテスト構成の分割管理の効果	39
3.5	ビル空調システム製品全体を 1 台の PC で動作させるイメージ図	40
3.6	テスト対象機器を実機，対向機を空調機器エミュレータで動作させる構成	40
3.7	テスト結果可視化ユニットが提供する可視化	42
4.1	テスト実行ユニット及び周辺の構成	44
4.2	SPIN のモデル検査実行フロー	45
4.3	操作・結果判定・通信におけるモデル検査と実行テストとのギャップ	46
4.4	Kripke 構造によるシステムの非決定的振る舞いの表現	47
4.5	SPIN による非決定的振る舞いを網羅するテストの進め方	47
4.6	実行テストにおける非決定的振る舞いを網羅するテストの進め方	48
4.7	モデル検査プログラム-実機/空調機器エミュレータ接続手法の実装	49
4.8	モデル検査プログラムから実行コードを初期化する手法の実装	50

4.9	[不具合 1] の再発防止テスト実行時の通信ログ	53
5.1	ビル空調プロトコル通信コマンドの送信手順（送信元：空調機器エミュレータ，送信先：実機）	58
5.2	エミュレータで上で動作する組込みソフトウェアの実行速度制御	60
5.3	テスト結果可視化ユニットが提供する通信可視化 GUI	62
5.4	ACET の構成	63
5.5	ACET におけるテスト結果可視化ユニット（入出力の表示）	64
5.6	テスト作業フロー	66
5.7	不具合となった機能衝突	70
6.1	複数機器/ソフトウェアが協調するシステムにおける機能の階層構造	74
6.2	テスト手順・テスト構成からのテストケース自動生成	75
6.3	手順テンプレートの模式図	76
6.4	構成データモデルの模式図	77
6.5	手順テンプレート・構成データモデルの組合せで生成される実行可能なテストケースの模式図	78
6.6	テストケース管理リポジトリ・実行可能テストケース生成ユニットの構成	79
6.7	手順テンプレート・構成データモデルの作成支援	80
6.8	手順テンプレート用作成支援ツール	80
6.9	構成データモデル用作成支援ツール	81
6.10	管理データベース数低減の割合の推移	87
6.11	機能衝突を起こす手順テンプレートの作成を支援するツール（イメージ）	88
7.1	統合テスト実行環境のソフトウェア構成	90
7.2	統合テスト実行環境のテスト実行画面	91
7.3	従来テスト環境と統合テスト実行環境との累積作業時間の比較	93
7.4	テストケースの自動切替機能（イメージ）	94
8.1	統合テスト実行環境の事業化活動風景	98

表目次

1.1	SPLE を適用したビル空調システム製品開発プロセス	9
1.2	ビル空調システム製品における国別機種展開開発の例	9
1.3	管理コントローラのソフトウェアモジュール数内訳	11
2.1	回帰テスト項目の増加理由の詳細	16
2.2	後工程への流出不具合事例	18
2.3	ビル空調システム製品の回帰テストにかかる作業量増加事例	20
3.1	ビル空調システム製品（タイプ1, タイプ2）の仕様	36
4.1	テスト実行ユニットを用いたテストによる不具合再発有無確認可否判断 結果	51
5.1	テスト結果可視化ユニットが提供する GUI 部品の例	61
5.2	室内機組込みソフトウェアを動作させる Softgun プロセス数に対する CPU 負荷測定結果	64
5.3	ACET の適用可否に関する判断結果	67
5.4	作業時間計測結果	67
5.5	不具合分析作業に関する実験結果	68
6.1	ビル空調システム製品の再発防止テスト項目一覧（抜粋）	84
6.2	手順テンプレート・構成データモデル作成に関する実験結果	85
7.1	作業時間計測結果	92

コード目次

4.1	不具合 1 の再発防止テストに対応するテストケース (概要)	52
4.2	不具合 2 の再発防止テストに対応するテストケース (概要)	53
4.3	不具合 3 の再発防止テストに対応するテストケース (概要)	54
6.1	手順テンプレートの記述例 (抜粋)	81
6.2	構成データモデルの記述例	82
6.3	実行可能なテストケースの記述例 (抜粋)	83
7.1	統合テスト実行環境における構成データモデルの記述例	90

第 1 章

序論

1.1 研究の背景

今，製品開発が限界を迎えている。

「製品」とは，本論文では，複数の機器ないし複数のソフトウェアが協調して機能を実現する，組込み機器製品を指す。このような製品の例として，複数の室外機や室内機が協調して空調機能を実現する，ビル空調システム製品がある。ビル空調システムとはオフィスビルや商業施設といったビルの空調を行うシステムであり，室外機や室内機は組込みソフトウェアによって制御されている。

図 1.1 に，ビル空調システムのオフィスビルにおける設置形態の一例を示す。ビルの屋上などに室外機が，オフィスフロアなどの空調を行う場所に室内機が，オフィスフロアの壁や柱といったユーザが操作可能な位置にリモコンが，そしてビル管理室などビルの管理者が操作可能な位置に管理コントローラが，それぞれ設置される。このうち室外機と室内機は，ビルの規模/用途に応じた空調能力を有するモデルが設置される。これらの機器は組込みソフトウェアで制御され，また互いに電源線及び通信線で接続され，通信によって複数の機器が協調し，空調機能を実現する。具体的には例えば，リモコンによってユーザより冷房ないし暖房を ON とするような操作が与えられると，室外機と室内機との間で冷媒ガス^{*1}のやり取りを冷媒ガス配管を通じて行い，室外機・室内機それぞれにおいて冷媒ガスに対する熱交換を行い，冷房ないし暖房を実現する。このようにビル空調システムの機能はユーザの操作や外気温の変化などのイベントをトリガとするものが多いため，各機能の実行順序及びタイミングは非決定的であるという特徴がある。接続台数については，オフィスビル向け製品ではおおよそ，室外機を最大 50 台，室内機を最大 50 台，リモコンを最大 100 台，管理コントローラを最大 4 台，1 つのシステムに接続できるようにして

^{*1} 冷蔵庫・エアコンなどの熱交換を必要とする機器において，熱を移動させるために用いられる熱媒体

いる。



図 1.1 ビル空調システムの設置形態（オフィスビルの場合）

「限界を迎えている」とは、テストにかかる工数が爆発的に増加していることを指す。本節では以降、この問題が発生した背景を、上述のビル空調システム製品の事例を用いて述べる。

ビル空調システム製品の開発では、様々なビルの規模/用途に応じるため、空調能力別の機種展開開発が必要となる。この機種展開開発に対応するため、2000年頃より、Software Product Line Engineering (SPLE) [1] を適用した開発プロセス（表 1.1 参照）を回していた。

しかし 2000 年代後半頃、ビル空調システム製品の主な展開先であった国内市場が伸び悩みを見せ始めた [2] ために海外市場への展開を図るようになると、以下に示す 2 つの開発が発生し、同期間でより多くの SPLE 開発を回さなければならなくなった。

（開発 1）国別機種展開開発

国毎の事情を鑑みた機能を搭載する開発であり、例としては表 1.2 に挙げるようなものがある。あるビル空調システム製品では 2018 年現在で 28 か国へのグローバル展開を実施しており、このような開発が 28 か国分、発生する状況となってしまっている。

*2 品質管理システムの要求事項を示した国際規格

表 1.1 SPLE を適用したビル空調システム製品開発プロセス

工程	実施内容（特徴的な内容のみ記載）
要件定義	ベース機種（ベースとする従来機種）、追加機能を決定
外部設計	ソフトウェア構成の、ベース機種からの差分を具体化
内部設計	－
コーディング	－
単体試験	－
機能試験	回帰テストとして競合テスト（追加機能と従来機能を衝突させるテスト）を実施，追加機能が従来機能に影響していないかを確認
システム試験	回帰テストとして組合せテスト（様々な機器構成/ソフトウェア構成でテスト）を実施，追加機能が従来機能に影響していないかを確認
品質管理試験	回帰テストとして再発防止テスト（ISO9001*2認証の取得/維持に必要な，過去不具合が再発していないことを示すテスト）を実施
出荷判定	ソースコードやテストケースなど，開発の成果物を開発責任者が承認
資産化	承認された成果物を，次の開発で再利用できるように管理する

表 1.2 ビル空調システム製品における国別機種展開開発の例

項目	内容
法規制対応	欧州向けに，F ガス規制*3の強化にともない，水方式*4機種を開発する．国内向けには，省エネ法*5の改定にともない，省エネ性能を強化した機種を開発する．
セキュリティ対応	国別に，各国のセキュリティ基準（欧州の GDPR*6，米国の NIST SP800-171*7など）を満足する，セキュリティ機能を開発する．
電源事情対応	国によって電源電圧や停電頻度 [3] が異なるため，国別に電源回路制御ソフトウェアや停電復旧処理ソフトウェアを開発する．

（開発 2） 他社製品取り込み開発

*3 温室効果のある冷媒ガスの放出を規制する目的で，空調機器の定期点検などを義務付ける欧州の法律

*4 冷媒ガスを使用しない，水による空調

*5 エネルギーの使用の合理化等に関する法律

*6 EU における個人データ保護に関する法律

*7 米国政府機関が調達する製品や技術などを開発・製造する企業に対して，一定のセキュリティ基準に準拠するように求めるガイドライン

各国の顧客要求への迅速な対応を目的に、他社製品（機器/ソフトウェア）を取り込む開発である。他社の機器を取り込む例としては、その国においてマーケットシェアの高いフレームワークに接続するためのゲートウェイ装置を、ビル空調システムに接続可能とする開発がある。他社のソフトウェアを取り込む例としては、管理コントローラの OS やファイルシステムなどの汎用的な機能をオープンソースソフトウェア（以降、OSS と称す）で実装する開発、要求分析の効率化を目的に製品販売地域に設けた開発拠点が作成したアプリケーションソフトウェアを取り込む開発、などがある。

SPLC を適用した開発プロセス（表 1.1 参照）のうち回帰テスト（競合テスト・組合せテスト・再発防止テスト）が、この開発頻度増加の影響を特に受け、工数が爆発的に増加した。理由を以下の 5 点にまとめる。

- 開発頻度が増加すると当然、システムは多機能化する。すなわち、競合テストで衝突させる従来機能が増加する。
- 新機種追加頻度の増加及び上述の多機能化のため、組合せテストの対象となる機器の組合せ及びソフトウェアの組合せが増加する。
- ビル空調システム製品では、長期にわたる旧機種接続対応を余儀なくされる（ビル空調機器の製品寿命は一般的に 10 年以上と長く、導入・交換も機器ごとに行われるため）。再発防止テストは接続可能な機種全ての不具合について実施する必要があるため、新機種が追加される度に、再発防止テストの項目数が増加していく。
- 社外製品はブラックボックス（詳細な仕様やソースコードが非公開）であることが多く、上流工程での評価（仕様レビュー、ソースコードレビューなど）を実施できない。結果、様々な評価を下流工程（テスト）で行わざるを得なくなる。
- ブラックボックスな機器やソフトウェアがシステムに多数/複雑に入り込むと（本論文では以降、この現象をマーブル化と称す）、競合テスト及び組合せテストのテストケースが複雑化し、テストケース設計にも工数がかかるようになる。マーブル化したビル空調システム製品の例として、管理コントローラを挙げる。図 1.2 に示したソフトウェア構成のうち色がついた部位がブラックボックスな社外製品であり、多数組み込まれていることが分かる。具体的には表 1.3 に示したソフトウェアモジュール数内訳の通り、汎用的な機能を実現するプラットフォーム部においては半数以上のモジュールが、社外製品となっている。

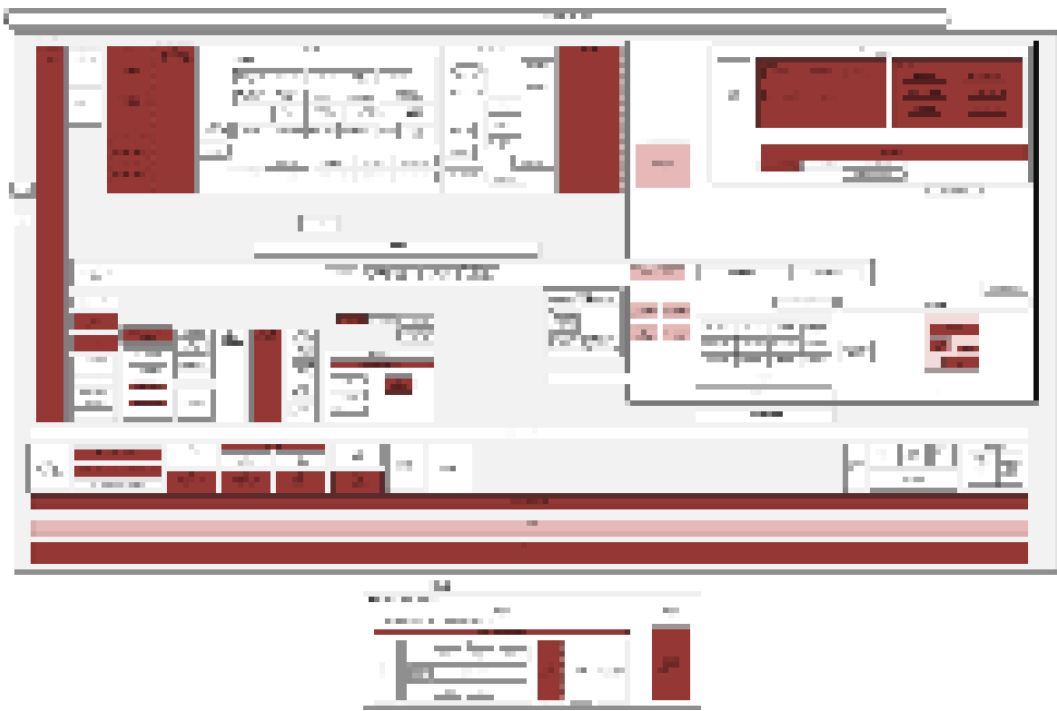


図 1.2 マーブル化したビル空調システム製品の例：管理コントローラ

表 1.3 管理コントローラのソフトウェアモジュール数内訳

部位	モジュール数（社内製）	モジュール数（社外製）
プラットフォーム	29	32
アプリケーション	150	10
合計	179	42

以上に述べた，SPLE を適用したビル空調システム製品開発において回帰テスト工数の爆発的増加が発生した流れを，図 1.3 にまとめる．実際，あるビル空調システム製品においては 2000 年から 2018 年にかけて開発工数及び設計工数/テスト工数が占める割合（概算）が，図 1.4 のように変移している．従来テストを実施していた専門のテスト技術者に加え経験の浅い作業者也投入し対応を継続しているが，グローバル展開にともなう国別機種展開開発及び他社製品取り込み開発の増加は止まる見込みがなく，このままではさらに対応が困難になっていくと考えられる．

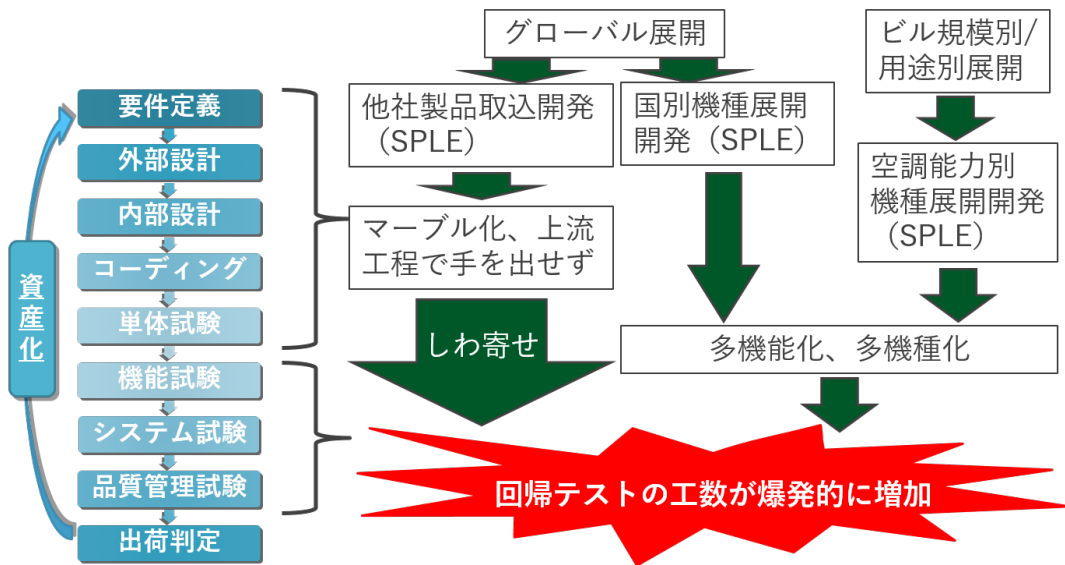


図 1.3 ビル空調システム製品の SPLE で回歸テスト工数の爆発的増加が発生した流れ

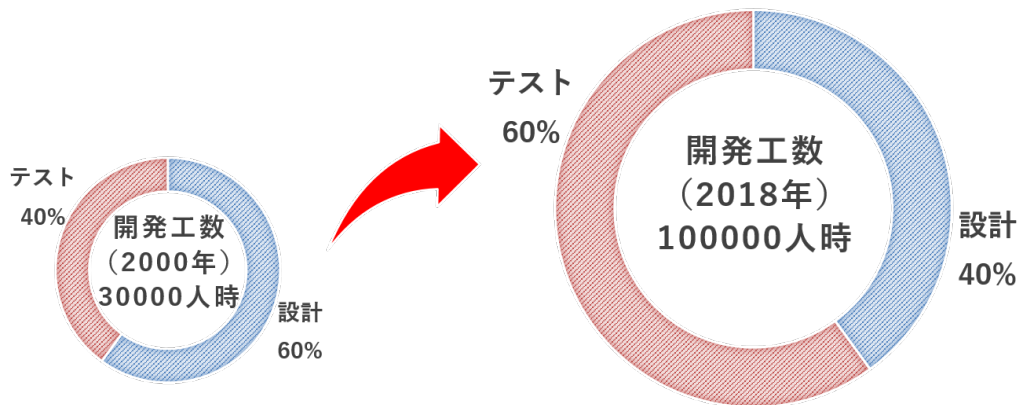


図 1.4 あるビル空調システム製品の開発工数の変移 (概算)

近年、IoT (Internet of Things)・CPS (Cyber Physical System) といったキーワードに代表されるように、複数の機器ないし複数のソフトウェアが協調する組込み機器製品は多数存在し、また高機能化が進んでいる。このため上述したような問題はビル空調システム製品に限らず、多くの組込み機器製品の開発で発生していると考えられる。この問題が深刻化する前に、一刻も早く対策を打つ必要がある。

1.2 研究の目的

本研究の目的は、SPLE を適用した組込み機器製品開発において爆発的に増加している、回帰テスト工数の低減である。本研究はこの目的を達成するため、以下を実現するテスト実行環境を提案するものである。

1. 上流工程でのテストケースと下流工程でのテストケースを共通化し、SPLE のコア資産として蓄積・再利用することを可能とすることで、SPLE を適用した組込み機器製品の回帰テストプロセスを統一的に支援する
2. 回帰テストの対象である組込み機器の管理作業及び準備作業、また回帰テストで抽出された不具合の分析作業にかかる工数を低減する
3. 回帰テストの実行及び結果確認を自動化する。併せて、自動テストのためのテストケースの資産化を支援する

本論文では初めに、SPLE を適用した組込み機器製品開発の回帰テストに関する問題を、ビル空調システム製品の事例とともに示す。続いて、それらの問題の解決に必要な技術について、従来研究から論じる。これらをふまえた本研究の提案手法として、回帰テスト工数を低減する、テスト実行環境について議論する。そしてそのテスト実行環境を、実際のビル空調システム製品開発で行われている競合テスト・組合せテスト・再発防止テストに適用して、その性能評価を行う。

以上をもって、既存のテスト支援技術では実現されていない、SPLE を適用した組込み機器製品開発における回帰テストに対する統一的な仕組みの下での効率化が実現されるようになることを明らかにする。最後にこの、本研究が明らかにした点を踏まえ、今後の研究と発展可能性について述べる。

1.3 本論文の構成

本論文は、8章から構成される。

第1章では、本研究の背景及び目的について記す。

第2章では、SPLE を適用した組込み機器製品開発の回帰テストで発生している問題の詳細をビル空調システム製品の事例を用いて示すとともに、問題解決に関連する従来研究について述べる。

第3章では、本研究の着眼点とアプローチについて説明し、提案するテスト実行環境の概要について述べる。

第4章では、提案するテスト実行環境のベースとなるモデル検査技術とエミュレーション技術を融合する方式について述べた上で、その方式の実現性を明らかにする。

第5章では、回帰テスト対象機器の管理作業及び準備作業、そして不具合分析作業をエミュレーション技術を活用して低減する方式と、その評価について述べる。

第6章では、回帰テスト自動化のために実行可能な形式で記述されたテストケースに対する資産化の支援を目的とした、テストケース記述方式及びテストケース生成方式について説明する。

第7章では、第4章から第6章で示した方式によって構築したビル空調システム製品向け統合テスト実行環境の実装、そして統合テスト実行環境を実際のビル空調システム製品開発で行われている回帰テストに適用する評価について述べる。

第8章では、本研究の成果と、今後の研究の展開について述べる。

第 2 章

SPLE における回帰テストの現状と課題

本章は 3 つの節で構成される。まず 2.1 節にて、SPLE における回帰テストで発生している問題を、ビル空調システム製品開発における具体的な事例を用いて説明し、その解決に有効な技術について議論する。次の 2.2 節は 2.1 節で述べた問題を踏まえ、関連技術がどのように発展してきたのかを従来研究を示しながら明らかにする。そして 2.3 節において、本研究が解決すべき課題を定義し、その解決に必要な技術を示す。

2.1 問題 –回帰テスト工数の爆発的増加–

SPLE を適用したビル空調システム製品の開発において、回帰テストの工数が爆発的に増加している背景については 1.1 節で述べた。本節では、回帰テスト工数を増加させている要因は以下の 4 点に大別されると考え、これらの要因を分析し、問題解決に有効な技術を明らかにする。

1. テスト項目の増加
2. テスト漏れによる手戻りの増加
3. テストにかかる作業量の増加
4. テスト技術者のスキル不足

2.1.1 テスト項目の増加

SPLE を適用したビル空調システム製品開発の回帰テストには、表 1.1 に挙げた競合テスト・組合せテスト・再発防止テストがある。1.1 節にて概要を述べたこれらのテスト項目が増加している理由の詳細化を通じ、解決に有効な技術を抽出した。詳細化の結果を表 2.1 にまとめる。

表 2.1 回帰テスト項目の増加理由の詳細

テスト種別名	項目増加理由
競合テスト	図 1.2 に示した管理コントローラのように、マールブル化が進みブラックボックスな部位を多く含んだソフトウェアでは、機能衝突に関する検証を上流工程（設計段階）で行うことが困難である。このため全ての従来機能と衝突させるテストが必要となり、開発が多くなり多機能化が進んだ分、テスト項目が増加している。
組合せテスト	空調能力別・国別の機種展開開発によって扱う機種数が増大したために、システムが取り得る構成が指数的に増加している。これにともなって問題がないことを確認すべき構成も指数的に増加、テスト項目もその分、指数的に増加している。
再発防止テスト	ISO9001 認証の取得・維持には、是正処置（発生した不具合の再発防止を図る処置）が求められる。是正処置が行われていることの確認方法は製品開発者に委ねられているが、ブラックボックスな部位については、再発防止テストで確認するしかない（設計レビューなどでは確認できない）。1.1 節でも述べたように旧機種接続対応が長期にわたるため、機種展開開発・他社製品取り込み開発が増加するとブラックボックスな部位を含んだ機種もその分増加し、再発防止テストで確認すべき項目も増加していく。

表 2.1 の内容からまず、問題解決に有効な技術として、テストケースの資産化を支援する技術が考えられる。テストケースとは本論文では、テストを実施するための入力と、その入力に対し望まれる出力をまとめたものを指す。資産とは、広義には企業に将来的に収益をもたらすことが期待される経済的価値を指すが、本論文では以降、「SPLE 資産としてテストケースを再利用可能とすること」を「テストケースの資産化」と称す。競合テスト・組合せテスト・再発防止テストは SPLE を適用したビル空調システム製品開発では

全ての開発で実施され、そして従来機能・旧機種もシステムに残り続けるため、その開発で追加される機能に特化したものを除いて、ほとんど同じテスト項目を開発の度に実施することとなる。よってテストケースを再利用可能とできれば、テストケース設計にかかる工数を大幅に低減できる見込みがある。しかしテストケースが再利用可能となったとしても、膨大なテストケースの管理作業には多くの工数がかかることが考えられ、機種展開開発が増加すればテストケースの追加作成作業も発生する。これら作業を効率化し、テストケース資産化を支援する技術が求められる。

また、ブラックボックスな部位の評価を実行コードに対するテスト（以降、本論文では実行テストと称す）だけでなく、上流工程で可能とする技術を確立できれば、フロントローディングによって各テストの項目数を低減できる。競合テスト・組合せテストについては影響範囲を根拠をもって狭めることができればテスト項目数を低減できるため、影響範囲の特定を自動化ないし支援する技術による対応も考えられる。

2.1.2 テスト漏れによる手戻りの増加

ビル空調システム製品において、各機能の実行順序及びタイミングが非決定的であることは、1.1 節で述べた。この非決定的振る舞いは、本質的には下記に示す内容となる。

- 機器間のメッセージ交換順序及びタイミングが、機能実行の度に変化する。
- 機器が有するある共通資源に複数の機能が同時にアクセスする場合、資源を獲得する順序及びタイミングが、実行時に初めて決定される。
- 複数の機器が協調のためにある機器に同時にアクセスする場合、アクセス対象機器と協調可能となる順序及びタイミングが、実行時に初めて決定される。

つまり、1 度機能を実行するテストを行うだけでは、不具合があったとしても、必ずしもその不具合を発見できない。以上の問題が顕在化した事例として、ビル空調システム製品の回帰テストにおいて後工程に流出した不具合を 3 件、表 2.2 に挙げる。このようなテスト漏れによる後工程への不具合流出が発生すると、手戻りが発生し、工数が増加する。

この問題の解決には、上述の非決定的な順序及びタイミングについて、漏れの無いテストケースを構築・実行する技術が求められる。

表 2.2 後工程への流出不具合事例

No	事例	概要	流出理由
[不具合 1]	メッセージ順序制約の見落とし	ビル空調管理システムの通信プロトコルは通信メッセージの到達順序を保証しないにもかかわらず、ある機能は3つの通信メッセージを定められた順序で受信しないと作動しなかった。	実機ではメッセージの到達順序を任意に入れ替えることができないために、事例の状況をテスト時に発生させることが難しかったため。
[不具合 2]	他機能による状態変更の見落とし	機能 A と機能 B が A → B の順番で実行されると、機能 B が参照するある状態変数に意図しない値が設定されており、機能 B が実行されなかった。	機能 A が当該状態変数を変更していたが、機能 A と機能 B とが異なるベンダで開発されており、ベンダ間で変更される状態に関する情報が共有されていなかったため。
[不具合 3]	機器モードによる制御状態変化の見落とし	機器 D1 の機能 C を、別の、モードが異なる2つの機器 (D2/D3) 両方から制御すると、D1 の状態が不安定になった。	機能 C に影響を与える機器 D2/D3 のモードの組合せをテスト技術者が把握することが難しく、モードの異なる機器 D2/D3 を組み合わせたテストが後工程までテストされなかったため。

2.1.3 テストにかかる作業量の増加

ビル空調システムの機器は組み込みソフトウェアで制御されるため、テストでは、組み込みソフトウェアを動作させる実機を扱う必要がある。結果、ビル空調システム製品の回帰テストは、図 2.1 に示すような作業手順にて実施される。

この各手順において、国別機種展開開発の増加・他社製品取り込み開発の増加にともなう作業量の増加が発生している。事例を表 2.3 にまとめる。

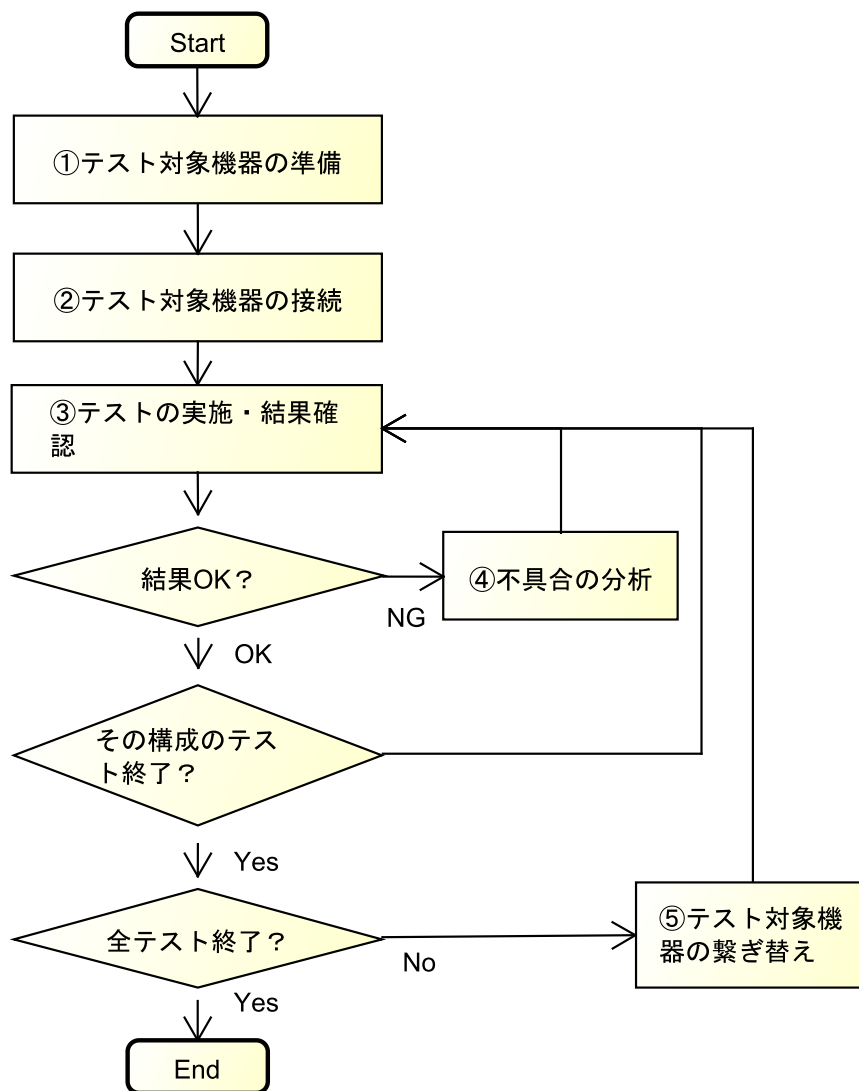


図 2.1 ビル空調システム製品における回帰テストの手順

表 2.3 に示した事例は、実行テストを実機レスで実施可能とすることで解決を図れる。実機レスとできれば、実機の保管にかかる作業、テスト実施時の実機の接続・繋ぎ替え作業は不要となり、テストの実施・結果確認・不具合分析を行うための入出力手段がハードウェアコスト低減によって制限される問題も回避できる。加えて、テストの実施・結果確認（③）にかかる問題については、テストの自動化によっても対応が見込める。

表 2.3 ビル空調システム製品の回帰テストにかかる作業量増加事例

作業手順	事例
テスト対象機器の準備 (①)	<p>組合せテストを実施するには、システムに接続される全機種の実機を最大接続台数分保管しておく必要がある。開発の増加にともない機種数が大きく増加し、この保管にかかる作業（定期的な棚卸・動作確認など）も増加している。他社製品について、故障しても再調達できない可能性を鑑み余分に確保することを考えると、作業量はさらに増加する。</p>
テスト対象機器の接続 (②)、テスト対象機器の繋ぎ替え (⑤)	<p>システムが取り得る構成が増加した分、機器の接続・繋ぎ替えにかかる作業量が増加している。ビル空調システムの開発現場では、頻繁に使用される機器をあらかじめ並べておくことでこの時間を低減するテスト環境 (図 2.2) を構築したが、並べられていない機器を含めた構成を構築する場合など、完全な解決とはなっていない。またその大きさ (1500mm×1700mm) ゆえ、設置スペースの確保が課題となっている。</p>
テストの実施・結果確認 (③)	<p>ビル空調システムの実機はハードウェアコスト低減のため、入出力手段 (スイッチ, LCD, デバッグ通信用端子など) を十分に備えておらず、テストの実施・結果確認において非効率な作業を強いられることが多い。テスト用に入出力を拡張する基板を別途用意する場合もあるが、その場合は拡張基板の開発コストが課題となる。</p>
不具合の分析 (④)	<p>テストの実施・結果確認 (③) の問題と同様。</p>

2.1.4 テスト技術者のスキル不足

1.1 節で述べたように、ビル空調システム製品の回帰テストは従来は高い専門性を有するテスト技術者が行っていたが、テスト工数の増加によってそういったテスト技術者だけでは対応が困難となり、経験の浅い技術者も投入して対応せざるを得ない状況となっている。経験の浅い技術者はどうしても、図 2.1 に挙げた手順を高い専門性を有するテスト技術者ほどには効率良く実施できないため、テスト工数はさらに増加することとなる。かと言って、逼迫した製品開発現場において、テスト技術者のスキルを向上させるための教育

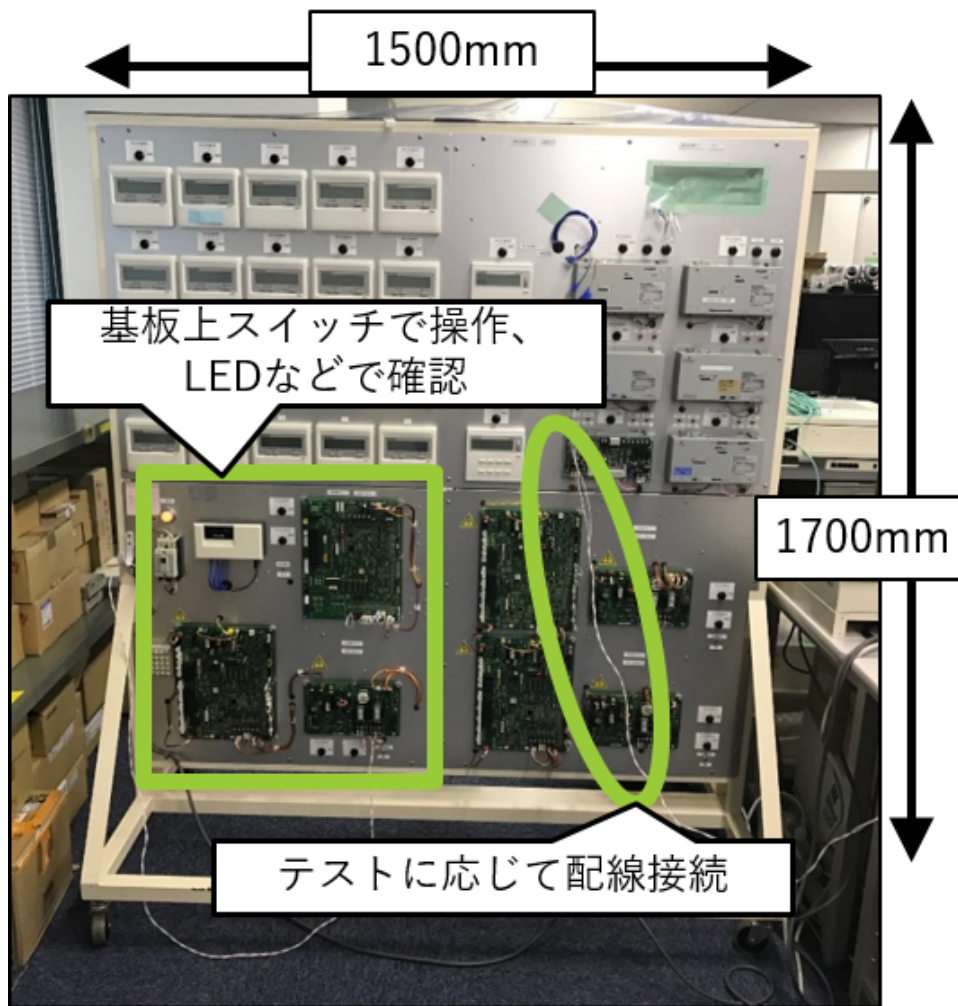


図 2.2 ビル空調システム製品の従来のテスト環境

時間・教育コストを確保することは困難である。

この問題解決にも、テスト技術者の教育の必要を低減することが可能な、テスト自動化技術が有効と考えられる。

2.2 従来研究 –テスト支援技術概説–

テスト工数の増加に対応するためのテスト支援技術については、多くの研究がなされている。本節では、SPLE で開発される製品を対象としたテスト支援技術、また前節で問題解決に有効とした技術がどのように発展してきたのかを、従来研究を示しながら明らかに

する。すなわち、以下の技術について概説する。

1. SPLE を対象としたテスト支援技術
2. テストケース資産化を支援する技術
3. 変更影響範囲の特定を支援する技術
4. ブラックボックステストを実行テスト以外で可能とする技術
5. システムの非決定的振る舞いに対するテストを支援する技術
6. 実行テスト環境の実機レス化技術
7. テスト自動化技術

2.2.1 SPLE を対象としたテスト支援技術

SPLE では、再利用可能なコア資産（プラットフォーム）と機種固有のアプリケーションとが明確に分離される。Eriksson（2005）[4] が示した SPLE の指針では、コア資産について以下のように述べられている。

- コア資産は機種共通の仕様を反映するもので、新機種要求を取り入れるものではない。
- コア資産と機種固有部分の独立性を高めることで、コア資産のメンテナンスによる影響分析を全機種について実施しなくてもよいようにする。

SPLE の適用効果は、Polzer ら（2009）[5] により確認されている。組み込みシステムの振る舞い部分をコア資産として分離し、設定パラメータ部分をラピッドプロトタイピングにて開発したところ、開発工数を大幅に削減できたとしている。

SPLE におけるテストは、コア資産拡張時のテストと、アプリケーション追加時のテストに分類される。Jin ら（2008）[6] は前者をドメインテスト、後者をアプリケーションテストと呼称しており、以降本論文でも、この呼称に倣う。ドメインテスト及びアプリケーションテストのテストケースはユースケースから継承され、テスト対象の変更点に応じて追加され、ソースコードと対応付けられ、ソースコードと同様にコア資産として管理されることが主張されており [7]、テストケースが SPLE において重要な資産となることが伺える。Neto（2012）[8] はユースケースとテストケースとの関連付けに着目し、ユースケースに対応するテストケースを資産化するツールを開発している。

SPLE のテストについては多くの研究がなされており、関心の高さが伺える一方で、様々な課題が抽出されている。Engström & Runeson（2011）[9] の調査からは、発表された手法に対する効果の確認が残課題となっていることが伺える。また Lee ら（2012）

[10]によれば、ドメインテストの仕様、テストケースの選択、テストコードの再利用、などに研究の余地があるとされている。Machadoら(2014)[11]も、SPLEのテストについての研究がテストケースの選択手法と実際のテスト手法という2つの観点でなされていることに言及しつつ、実製品開発に基づいた研究成果が少ないと述べている。

これらの残課題のうちドメインテストの仕様については、Ali(2010)[12]が、ユースケースをカテゴリ化した情報から設計するアプローチ・動的振る舞いを記述できるように拡張したアクティビティ図から設計するアプローチを提案している。テストケースの選択については、Pauloら(2010)[13]がソース構造のグラフ化により、Runeson(2012)[14]がテストの深さ・バージョン・派生物の3軸でテストを整理することにより、選択を支援する方式を提案している。一方でHenardら(2014)[15]によるT-wise法を適用し組合せパターン数を低減する手法や、Vidacsら(2015)[16]によるソースコード上のカバレッジを極大化するように組合せを選択するアルゴリズムによって、選択を自動化する方式の提案もなされている。

2.2.2 テストケース資産化を支援する技術

前述のように、テストケースはSPLEにとって重要な資産となり得る。しかしシステムが多機種化・多機能化しその分テストケースが増加すると、再利用は難しくなる。肥大化したドキュメントは、読みやすさが損なわれるとともに、作成/統合/編集にかかるコストが増加するためである[17]。

この再利用における課題への対応として、テストケースを実行可能な言語で記述し、これを資産化するアプローチが考えられる。実行可能な言語であれば、読解の必要性は減り、また作成/統合/編集を機械的な処理によって実施可能となることが見込めるためである。テストケースを記述可能かつ実行可能な言語としては、VDM[18]・B Method[19]・Z Notation[20]に代表される形式仕様言語(Formal Specification Language)がある。形式仕様言語は30年来研究が続けられており多数の成果が発表されている分野であるとHarmanら(2014)[21]は述べており、技術としてはほぼ確立されているものと推測される。

2.2.3 変更影響範囲の特定を支援する技術

変更影響範囲の特定を支援する技術としては、ソフトウェア構造を可視化する技術が多数提案されている[22]。

テストと関連させた可視化手法もあり、例えばLiuら(2012)[23]は組み込みプログラムのテストケース候補をコールグラフ形式で見える化するとともに、実施したテストケー

スおよびその結果をコールグラフに重ねて描画することでテストの網羅性を見える化するツールを提案している。Solid* Toolset[24]では、依存関係にある機能の組合せの可視化が実現されている。機能ブロック群を円状に配置し、機能ブロック間に引いた線の濃淡で機能間の依存の深さを示している。他には、様々な可視化手法を一つのツールにまとめることで、ユーザに使用方法を委ねるアプローチを採っている研究 [25] もある。機能ブロック群の配置を縦に並べる／横に並べる／円状に配置する、と言ったように変更できたり、その機能のライン数や複雑さといった特徴量を機能ブロックの形状で示したりしている。

2.2.4 ブラックボックステストを実行テスト以外で可能とする技術

1章で述べたように、ビル空調システムにおいてブラックボックスな部位となるものの一つに OSS がある。OSS に焦点をあてたブラックボックステスト支援技術は従来より提案されており、野村ら (2016) [26] が主張するように、品質評価が OSS の課題であることは認識されていると考えられる。ただ、例えば Xu ら (2016) [27] が 4 種類のブラックボックステスト手法を採り上げ 2 つの OSS に適用した結果をまとめているが、あくまでテスト対象は実行コードであったように、提案内容は実行テストを支援する手法であることが多い。OSS に対する実行テスト以外のアプローチとしては、Yahav ら (2014) [28] が OSS 自体ではなく OSS コミュニティに関するデータからその OSS のリスクを洗い出す方法を述べているが、これは OSS コミュニティがオープンであることに依存しており、開発に関わる情報も含めてブラックボックスである他社製品には適用できない。

一方、OSS に限定されない実行テスト以外の評価技術として、実行コードではなくシステムのモデルを検査するモデル検査技術 [29] がある。ブラックボックスな部位を取り込んだシステムのモデルを構築し、これをモデル検査で評価できれば、SPLE の上流工程でもブラックボックスな部位を評価することが可能となる。このモデル検査技術は次節の表題にも深く関連するため、従来研究の概説は次節にて行う。

2.2.5 システムの非決定的振る舞いに対するテストを支援する技術

前節で触れたモデル検査技術によれば、非決定性を含むシステムをモデル化し、モデルに含まれる全ての状態遷移を網羅的に探索し、モデルが満たすべき性質に違反していないかどうかを自動で検証することができる。そのため実行テストでは困難な、非決定的な順序及びタイミングで実行されるビル空調システムの機能に対する漏れの無い検証を、自動で行えるようになることが期待できる。実際、MC/DC カバレッジ [30] を網羅するテストケースをモデル検査によって生成した先行研究 [31] も発表されている。組み込みシステムも検査対象とできることが、形式手法 FD-DEVS[32] で記述した組み込みシステムの

モデルを検査する手法 [33] などにより明らかとなっている。

このモデル検査を実現するツールは 90 年代後半頃より研究されており、代表的なものに SPIN[34], NuSMV[35], UPPAAL[36] などがある。SPIN は専用の形式仕様言語である Promela で記述された検査モデルと線形時相論理式 (LTL 式) を入力とし, LTL 式に対する反例を自動抽出するツールである。SPIN を活用した研究例としては, Petrinet[37] を拡張した記法でモデル化された組み込みシステムのモデル化を Promela に落とし込んで検査する手法 [38] や, ゴールモデルを Promela に変換することにより要求仕様検証を SPIN で自動化する手法 [39] などが提案されている。NuSMV はシンボリック手法を用いたモデル検査ツールであり, 活用例には, IoT システムアーキテクチャモデルを NuSMV 用の SMV コードに変換し検査する手法 [40] などがある。また, 状態遷移テストについて仕様上の二つの状態からなる各順序対すべてを網羅する全状態ペアカバレッジ, 状態遷移の順序対すべてを網羅する全遷移ペアカバレッジを提案する論文 [41] が公表されているが, これらのカバレッジを網羅するテストケースを自動生成するツールとして, NuSMV が用いられている。

上述のようにモデル検査技術は長年の研究対象となっているが, 近年でも, モデル検査ツールをより容易に使用できるようにする研究や, 新たな対象をモデル検査可能とするツールの研究が多くなされている。

モデル検査ツールの検査対象であるモデルの構築, 検査結果 (反例など) の読解, 反例に至った原因 (不具合) の特定には, モデル検査ツール及び検査対象システムに対する専門的知識や経験が必要であった。これを受けて, 状態遷移表から Promela で記述された検査用モデルを自動生成するとともに, 反例をチャート形式に加工し可読性を向上させる SPIN 用ツール [42] が提案されている。状態遷移表が作りこまれていさえすれば, 専門的知識や経験が乏しくても, モデル検査の実施・反例の解析を容易に実施できる。不具合が発生したソフトウェアシステムのモデル化を半自動化し, SPIN にて不具合再現を試みる手法 [43] も構築されている。開発者が不具合レポートの内容に基づいて検査したいパラメータを決定すると, POM フレームワーク [44] にてソースコードを分析し, そのパラメータを検査するような Promela モデルが生成される。また不具合特定についても, SPIN の実行履歴から正常動作と LTL 式に違反した動作の分岐点を探索し不具合原因候補として自動抽出する手法 [45] が提案されている。NuSMV については, 使い勝手を向上させるフレームワーク NuSeen[46] が研究されている。このフレームワークでは, モデル記述言語用エディタ, モデル作成支援チェックリスト, モデル・モジュール間依存関係・反例の可読性を向上させる UI など, NuSMV を使用する上で負担となる作業を効率化するツール群を提供している。

続いて, 近年研究成果が発表されている, 新たな対象をモデル検査可能とするツール

について述べる。DIVINE[47]は、C/C++ソースコードに対しモデル検査を実施するLLVMをベースに開発されたツールであり、メモリ不正アクセス等を検知することができる。KIND 2[48]は、リアクティブシステム向け同期型形式記述言語 Lustre[49]で記述されたモデルを検査可能としている。マルチエージェントシステムに対するモデル検査ツールとしては、MCMAS[50]が公開されており、NuSMVに対する処理性能・メモリ性能の優位性をうたっている。WebMC[51]は、Webブラウザが行う通信プロトコルを対象とするモデル検査ツールで、クロスサイトスクリプティングやクロスサイトリクエストフォージェリといったセキュリティリスク有無の判定に活用できるとしている。パラメトリックマルコフ連鎖で記述されるモデル、信頼水準の範囲、確率計算木論理(PCTL)をインプットとし、信頼水準の値に対する信頼区間を導出するFACT[52]や、マルコフ連鎖モデル・マルコフ決定過程モデルを検査するSTORM[53]といった、確率モデルを対象としたモデル検査ツールも発表されている。

以上に述べた通り、モデル検査は利便性向上も含めての技術確立が進んでいる。前節のブラックボックスな部位に対する手段ともなることと併せ、本研究の目的であるビル空調システムの回帰テストを効率化する技術として、有用性が見込める。

2.2.6 実行テスト環境の実機レス化技術

実行テスト環境を実機レス化するアプローチとして、エミュレーション技術によりパソコン上で組み込みソフトウェアを動作させる環境を構築し、これをテストすることが考えられる。エミュレーション技術の特徴は実機に搭載するソフトウェアと同一のソフトウェアを動作させることにあるが、製品開発の評価においてこの特徴は大きなメリットとなる。例えばビル空調システム開発では、デバッグ情報を仕込んだソフトウェアは、実機に搭載されるソフトウェアとは異なると判断される。よってデバッグ情報を仕込んだソフトウェア上でどれだけ評価を実施したとしても、表 1.1 に示した出荷判定において、承認を得ることは難しくなる。Chen ら (2015) [54] はこの実機との差異に関する問題への対策として、テスト対象のソフトウェアは実機上で動かし、シンボリック実行やテストケースの自動生成といった処理をパソコン上で行う構成の組み込みソフトウェアコンコーリックテスト環境を提案しているが、エミュレータを活用すればこの問題の回避が可能となる。

エミュレーション技術を活用した実行テスト環境は Seo ら (2007) [55] が提案しており、テストの自動実行・自動判定を実現している。そしてこのアプローチによるテスト手法およびテスト環境は、研究対象、及び製品として広まりつつある。

製品の例には、OKL4[56]や Wind River Hypervisor[57]に代表される組み込み向けハイパーバイザがある。しかしこれらハイパーバイザは、メモリ保護機能や CPU 特権モー

ド機能に対応したアーキテクチャ (x86 や ARM) を必要とするため、ビル空調システムの室外機や室内機を初めとする、ローエンドマイコンを採用する組み込み機器では使用できない場合がある。

ローエンドマイコンを採用する組み込み機器を事例とした研究には、オープンソースのプロセッサエミュレータである QEMU[58] および QEMU に対応したモバイル向けネットワークエミュレータ NEmu[59] を活用した事例が多い。原嶋ら (2012) [60] は QEMU をベースに、実際のボードの完成前に組み込み機器のテストを可能とする実機レステスト環境を構築している。Bardhi ら (2016) [61] や Brady ら (2017) [62] は NEmu を用い IoT デバイスのエミュレータを構築し、エミュレータ間の通信試験、さらに実機とエミュレータとの通信試験を実施できるようにしている。Jayashree ら (2017) [63] の論文では、エミュレータ上でソフトウェアプロトタイプの開発を行うことで、H/W 開発状況に影響されないソフトウェア開発が可能となるメリットに言及している。またこの論文では、エミュレータ上で開発したソフトウェアと FPGA 上のソフトウェアを、トランザクション・レベル・モデリング (TLM) アダプタを用い相互変換できるようにすることで、製品評価に使えるソフトウェアも開発できるとしている。Costin ら (2016) [64] は、組み込み製品の脆弱性有無を自動検証する環境の構築を目的とし、組み込み製品ソフトウェアを PC 上で動作させる手段として、QEMU を用いている。[65] では、シミュレーションモデルと QEMU ベースのエミュレータを組み合わせた、CPS の閉回路試験向けプラットフォームについて述べている。ビル空調システムの組み込みソフトウェアを、QEMU で動作させた先行研究も存在する。Ozmen ら (2017) [66] が、QEMU 上で動作する実際のビル空調システム制御ソフトウェアを ADEVS[67] によってコントロールすることで、ビル空調システムのデマンドレスポンス機能をシミュレーションする環境を実現している。

QEMU をベースとしたもの以外では、消費電力を導出可能な空調機器のエミュレータを作成し、スマートグリッドシステム評価を目的として使用している事例 [68] がある。また安価な PV パネルエミュレータを構築することで、太陽光発電アルゴリズムの評価にかかるコストの低減をねらう論文 [69] も発表されている。

またエミュレータ自身が不具合を含む可能性についても、対策が提案されている。例えば Yu ら (2016) [70] は、エミュレータの不具合を端にセキュリティ上の脆弱性が流出したことに触れ、エミュレータの動作と信頼できるデバイス (Golden Device) との動作を効率的に比較する、エミュレータのための試験環境 (VDTEST) を構築している。

これらの先行研究より、ビル空調システムの組み込みソフトウェアをエミュレーション技術によりパソコン上で動作させテストするための技術は、ほぼ確立されていると考えられる。

2.2.7 テスト自動化技術

テストを自動化する技術は、組み込みシステムを対象とするものに限定しても、これまでに挙げたモデル検査ツール・エミュレーション技術を活用したテスト環境など、多数の方式が提案されている。2.2.2 節に述べた形式仕様言語も実行可能であるため、テストケースを記述すれば、自動テストの入力として用いることができる。テストの自動化については、これらの確立された技術をターゲットに合わせて最適化する段階にあると推測される。

2.3 本研究で解決する課題

本章では、SPLE を適用したビル空調システム製品開発で問題となっている回帰テスト工数の爆発的増加に対応するための技術を事例をふまえて明らかにするとともに、それらの技術がどのように発展してきたのかを関連研究を示しながら概説した。

回帰テスト工数が増加する要因は複数あるが、それら要因に有効と考えられる多様なテスト支援技術が議論され、発展してきた。テスト項目数の増加に対しては、SPLE ではテストケースの資産化が重要であることが示され、また実行可能な形式仕様言語でテストケースを記述することで資産化の支援が可能となっている。他にも回帰テスト項目数を低減する技術として、ブラックボックスな部位の上流工程での評価を可能とするモデル検査技術、変更影響範囲の特定を自動化/支援する技術が確立されてきた。特にモデル検査技術は、非決定的な順序及びタイミングに対する網羅的な評価を実現し、テスト漏れにともなう手戻りの増加にも対応できる。テストにかかる作業量の増加については、これを低減する、エミュレーション技術を活用した実機レステスト環境が整備されてきた。また、これら技術によってテストを自動化することで、テスト技術者のスキル不足にも対応が可能となる。

これらの技術は、以下2つのテスト環境に集約されると考える。

上流工程向け

モデルに対するテストケースを実行可能な形式仕様言語で記述し入力することで、ブラックボックスな部位を含むシステムについても設計段階での自動評価を行える、モデル検査ツール

下流工程向け

エミュレーション技術によって組み込みソフトウェアをパソコン上で動作させることで、実行テストの実施・結果判定の自動化と不具合分析の支援とを可能とする、

実行テスト環境

そして SPLE を適用した製品開発への導入を考えれば、2つの技術は統一された仕組みで扱えることが望ましい。モデル検査向けのテストケースと実行テスト向けのテストケースが統一されていれば、テストケースの再利用性を高めることができる上、システムのモデル（上流工程の成果物）通りに実行コード（下流工程の成果物）が作成されているかを明らかとしやすくなるためである。（図 2.3 参照）

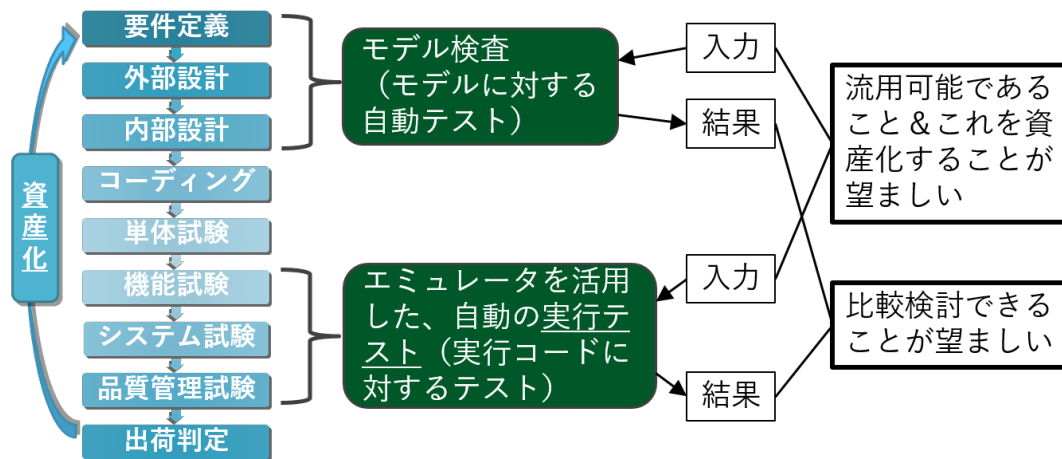


図 2.3 統一された仕組みで扱えることの効果

しかし先行研究を概観する限り、現状、2つの技術は独立している（テストケースが統一されていない）。設計支援技術に関しては、MDA[71]に代表される、モデルと実行コードとを統一的に扱えるモデル駆動開発が確立されてきており、また上流工程・下流工程で共通に扱えるレビューを支援する技術も確認される [72] が、テスト支援技術に関してはまだそのような動きは確認できない。運用で仕組みの統一を図ることもできるが、テスト技術者・開発責任者が双方の技術について知識・経験を得る必要があり、製品開発現場ではそのための余裕が与えられることはほぼ無いため、難しい。製品開発への導入のためにはやはり、技術的な解決が求められる。

本研究では以上の課題を考慮し、モデル検査技術とエミュレーション技術を融合し、SPLEにおける回帰テストを統一的な仕組みの下で支援することで、回帰テスト工数を低減する手法による解決を提案する。具体的には、上流工程向けのテストケース・下流工程向けのテストケースを共通化し、SPLE資産として再利用可能にすることを指す。そしてこの手法をビル空調システム向けの統合テスト実行環境（図 2.4）として構築し、実際のビル空調システム製品開発で行われている回帰テストに適用し、工数低減性能を評価する。

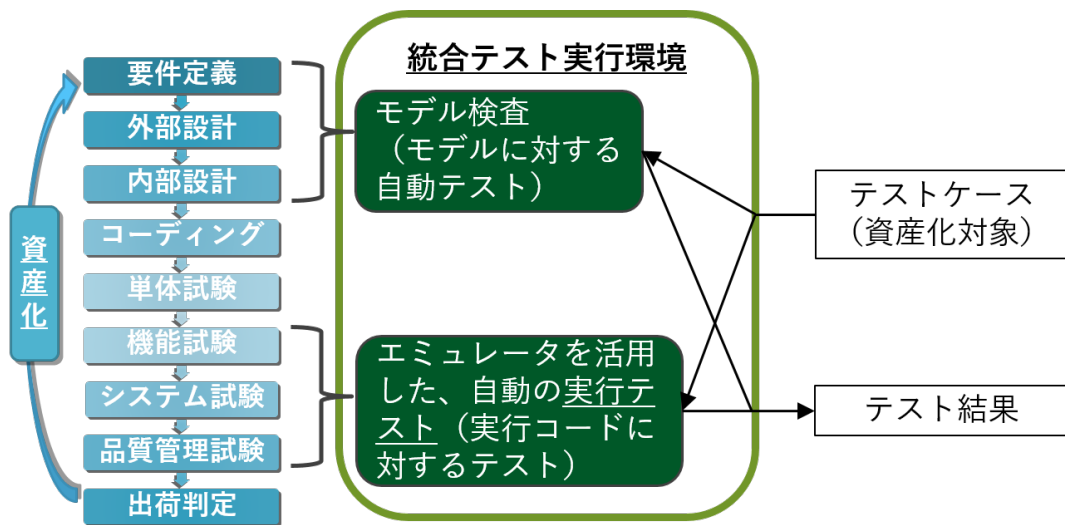


図 2.4 本研究の提案：統合テスト実行環境

次章では、本章で提案した手法の実現に向けたアプローチと、統合テスト実行環境の概要について説明する。

第 3 章

組込み機器のための統合テスト実行環境

第 2 章では本研究の課題について、ビル空調システム製品における事例及び先行研究を概観しつつ論じた。本章では、第 2 章でまとめた課題に対する本研究のアプローチと、本研究にて提案する統合テスト実行環境の概要を論じる。3.1 節では、SPLE を適用したビル空調システム製品開発における回帰テスト工数を低減するためのアプローチ、及びその新規性を述べる。そして 3.2 節では、ビル空調システム製品の仕様を照らしつつ、3.1 節に示したアプローチを考慮して設計した、統合テスト実行環境の基本的なアイデアを述べる。

3.1 本研究のアプローチと新規性

近年、ビル空調システム製品のグローバル展開を進めるために国別機種展開開発・ブラックボックスな他社製品取り込み開発が発生し、同期間で多数の SPLE プロセスを回さなければならなくなった結果、回帰テスト工数が爆発的に増加していることは第 1 章で述べた。続く第 2 章では、テスト支援技術が従来より積極的に議論され、上流工程向けには形式仕様言語で記述された実行可能なテストケースでシステムのモデルを評価するモデル検査ツール、下流工程向けにはエミュレーション技術を活用したテスト環境といった、回帰テスト工数の低減に有効と考えられる技術が確立されてきていることを示した。しかしこれら技術を SPLE に適用する上では、各技術が統一的な仕組みで扱えることが望ましい。そしてこの統一的な仕組みについては、従来、十分に議論されてこなかったと言える。

そこで本研究では、SPLE の回帰テスト工数を低減することを目的に、モデル検査技術

とエミュレーション技術とを融合し、上流工程向けのテスト環境と下流工程向けのテスト環境とを統一的な仕組みの下で扱える、統合テスト実行環境を提案する。そして実際のビル空調システム製品開発で行われている回帰テストに適用し、その評価を行い、提案する統合テスト環境が回帰テスト工数を低減可能であることを示す。

製品開発への適用にあたっては、製品開発現場の状況を考慮する必要がある。ビル空調システム製品における SPLE の関係者にはテスト技術者と開発責任者が居るが、2.3 節でも述べたように、彼らは新しい技術についての知識・経験を得るための余裕はほとんど得ることができない。したがって統合テスト実行環境では、図 2.2 に挙げた従来テスト環境からの作業手順変更が最小限となるようにする。作業手順を従来から変えざるを得ない機能については自動化を行い、知識・経験を習得する必要を低減する。

以上に述べたアプローチによって、ビル空調システム製品開発への適用が可能となると考える。上記の課題及び課題に対するアプローチを踏まえ、本研究の新規性を以下に列挙する。

1. モデル検査技術とエミュレーション技術との融合による、モデル検査と実行テストを統一的に扱える仕組みの実現（第 4 章）

上流工程向けテストケース・下流工程向けテストケースを共通化するため、システムのモデルを検査対象としてシステムの非決定的要素を自動評価するモデル検査を、組込み機器実機ないしエミュレータ上で動作する組込みソフトウェアに対して実施可能とする手法について議論した。そして議論した手法のビル空調システム向けプロトタイプを構築し、表 2.2 に挙げた不具合の再発防止テストを実施可能かどうかという観点で、本手法の実現性と効果を評価した。

2. エミュレーション技術による、回帰テスト対象機器の管理作業及び準備作業、及び不具合分析作業の効率化（第 5 章）

エミュレーション技術によって、図 1.1 に示したビル空調システム製品全体に対応する組込みソフトウェア群をパソコン上で動作させるとともに、エミュレータ上で動作する組込みソフトウェアと組込み機器実機との協調を可能とする仕組みを提案した。加えて、従来テスト環境ではハードウェア上の制約のためにテストで用いることができなかった入出力の操作・確認を可能にする機能を開発した。これらをビル空調システム向けの統合テスト実行環境の一部として実装し、ビル空調システムの SPLE における回帰テストについて得られる効果を実験的に考察した。

3. 形式仕様記述とテンプレートエンジンとによる、自動テストのための実行可能なテストケースの資産化支援（第 6 章）

自動テストのための実行可能なテストケースを、テスト手順・テスト構成に部品化

してそれぞれを形式的に記述し，テスト実行時にテンプレートエンジンを用いて任意に組み合わせ，多様なテストケースを自動生成する方式を提案した．この方式によってテストケースの資産化が支援されること，すなわちテストケースの再利用が促進されることを，ビル空調管理システム実製品の再発防止テストへの適用を通じ確認した．

3.2 統合テスト実行環境概要

本節ではビル空調システム製品の構成・仕様を分析し，提案する統合テスト実行環境が満たすべき性能を明らかにする．そしてその性能を実現するテスト環境の構成・機能について，概要を説明する．

3.2.1 ビル空調システム製品の構成・仕様

ビル空調システム製品は，他社製品（機器/ソフトウェア）を取り込めるようにするために，ベースをオープンな環境としている．オープンな環境とは，ビル空調システムを含む複数の機器が協調して機能を実現する組込みシステムにおいては，システムレベルのオープンネットワーク*1と，個々の機器レベルのオープンプラットフォーム*2の2つに大別できる．そこで本論文では以降，以下に定義する2つのタイプをもって，ビル空調システム製品の構成を示す．

- タイプ1（図3.1）

<定義>：オープンネットワーク上に，製造業者が自ら開発したホワイトボックスな機器と他社製のブラックボックスな機器とがともに接続され，複数の機器が協調して機能を実現するシステムの構成．

従来は実際の空調を行う室外機や室内機，リモコンなど，大半の機器を製造業者が自ら開発していた．しかし製品のグローバル展開が進むにつれ，まずは制御のインプットとなる温度・人の在不在などの検知に使用するセンサ，管理コントローラなどにおいて他社製を取り込むようになり，近年では室外機・室内機にも他社製品を含めるシステムも出始めている．

*1 処理手順や機能，外部インターフェース，構造などの仕様が公開されているネットワーク．ビル空調システムに関するオープンネットワークには，BACnet[®]・LONWORKS[®]・Modbus[®] などがある．

*2 ハードウェアやソフトウェアにおいて，技術仕様やプログラムのソースコードなどを公開したプラットフォーム．組込み機器における代表例には，リアルタイムオペレーティングシステム（組込み向けLinux[®]・FreeRTOS・μITRON）がある．

- タイプ 2 (図 3.2)

<定義>：オープンプラットフォーム上に、製造業者が自ら開発したホワイトボックスなソフトウェアモジュールと他社や製品販売地域開発拠点が開発したブラックボックスなソフトウェアモジュールとがともに搭載され、複数のソフトウェアが協調して機能を実現する、機器上ソフトウェアの構成。

ビル空調システム製品の中では、比較的大規模なソフトウェアを搭載する管理コントローラのソフトウェアが、この構成となっている（図 1.2 参照）。空調を初めとする製品の基本機能を実現するアプリケーションは製造業者が自ら開発し、販売地域に固有の拡張機能アプリケーションは製品販売地域開発拠点が開発していることが多い。またプロトコルスタック・データベースといった汎用的な機能を提供するミドルウェアには OSS を採用し、オペレーティングシステムやハードウェアドライバなどのプラットフォームを構成するソフトウェアはハードウェアメーカーから提供を受けるなど、ブラックボックスなソフトウェアモジュールを多数含んだ構成となっている。

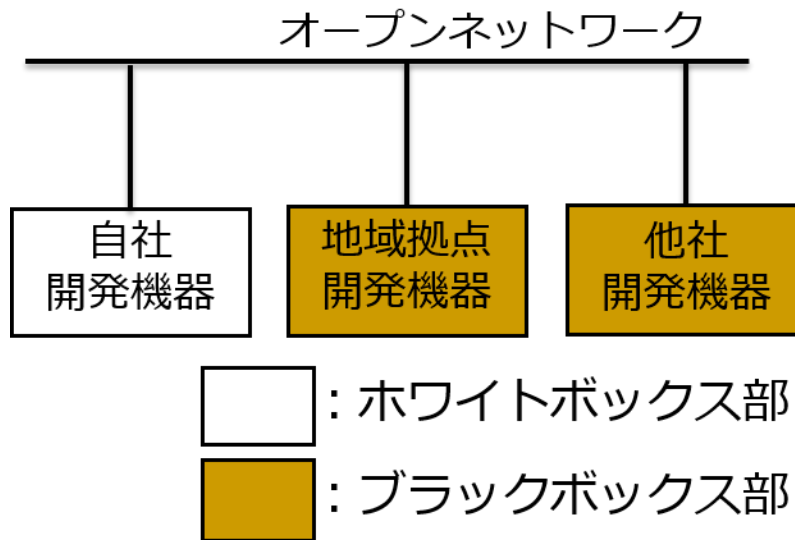


図 3.1 ビル空調システム製品の構成（タイプ 1）の模式図

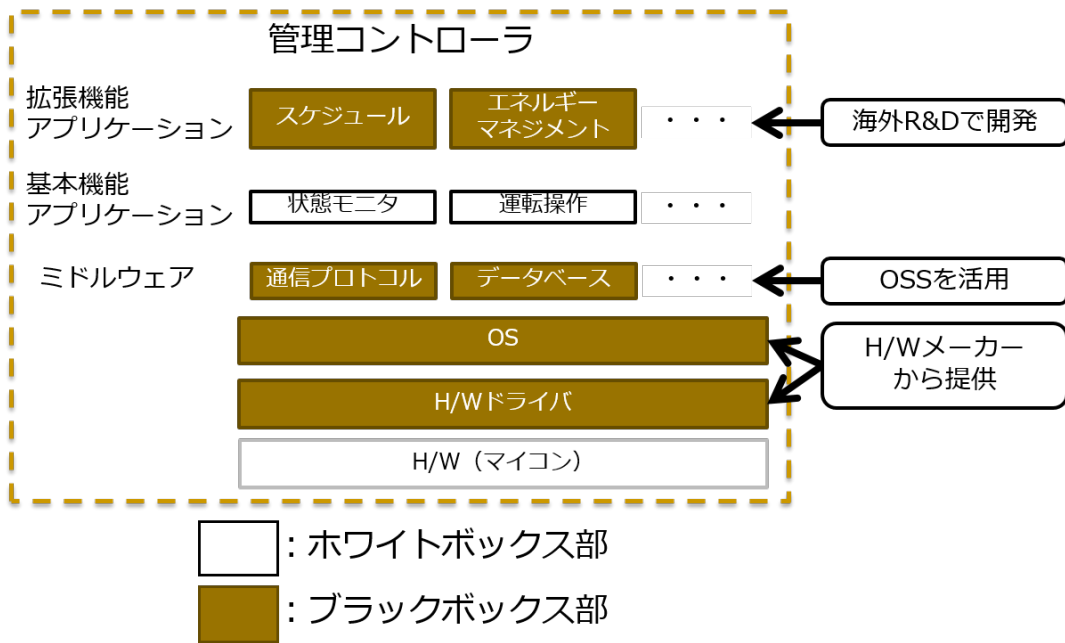


図 3.2 ビル空調システム製品の構成 (タイプ 2) の模式図

続いて、上述のタイプ 1 及びタイプ 2 の、回帰テストに関する仕様を表 3.1 にまとめる。統合テスト実行環境に組み入れるテスト手法は、この仕様を考慮し検討する。

表 3.1 ビル空調システム製品（タイプ 1, タイプ 2）の仕様

タイプ	項目	仕様	補足
タイプ 1	機器間通信速度	数 10kbps 程度	ノイズ耐性や回路コストを重視し、低速
	通信タイムアウト時間	数秒程度	低速通信であり通信タイミングが非決定的であるために、大きめ
	最大接続機器数	室外機・室内機：50 台程度，リモコン：100 台程度	ビルの規模/用途に応じて変化
タイプ 2	CPU 処理速度	数 10MHz 程度	コスト重視で低速な CPU が選択される
	入力インターフェース（操作可能）	スイッチ	ディップスイッチ，タクトスイッチなど
		タッチパネル	管理コントローラに搭載
	入力インターフェース（操作不可）	デジタル入力	在不在検知センサ入力などに使用
		アナログ入力	温度センサ入力などに使用
		通信入力	シリアル，パラレル，USB，有線 LAN，無線 LAN など
	出力インターフェース（確認可能）	LED	チップ LED，7 セグ LED など
		液晶	管理コントローラに搭載
	出力インターフェース（確認不可）	デジタル出力	シーケンス制御などに使用
		アナログ出力	アクチュエータ制御などに使用
通信出力		シリアル，パラレル，USB，有線 LAN，無線 LAN など	

3.2.2 統合テスト実行環境の構成・機能

本研究で実現を目指す統合テスト実行環境のコンセプトは、2.3節で述べた通りである。本節ではこのコンセプトを、3.1節で論じたアプローチ、3.2.1節に挙げたビル空調システム製品の構成・仕様を踏まえ実現する統合テスト実行環境の構成・機能について、概要を説明する。

統合テスト実行環境の構成及び機能を、図 3.3 に示す。本テスト環境は、テスト手順・テスト構成と、ビル空調システム製品を構成する機器の組込みソフトウェアとを、図 3.3 中①のリポジトリ（以降、テストケース管理リポジトリと称す）にて管理する。実行可能なテストケースはこのテスト手順及びテスト構成を任意に組み合わせ、実行可能テストケース生成ユニット（図 3.3 中②）にて自動生成される。そしてテストケースを、図 3.3 中③のプログラム（以降、テスト実行ユニットと称す）が自動実行する。また本テスト環境は、前述の組込みソフトウェアを PC 上で動作させるエミュレータ（図 3.3 中④）を備える。このエミュレータを本論文では以降、空調機器エミュレータと呼称する。空調機器エミュレータ上で動作する組込みソフトウェアの入出力、及び機器間の通信は、図 3.3 中⑤のテスト結果可視化ユニットによる操作・確認を可能とする。前述のテスト実行ユニットはゲートウェイ（図 3.3 中⑥）を介し、空調機器エミュレータ上で動作する組込みソフトウェアだけでなく、実機をテストすることもできる。

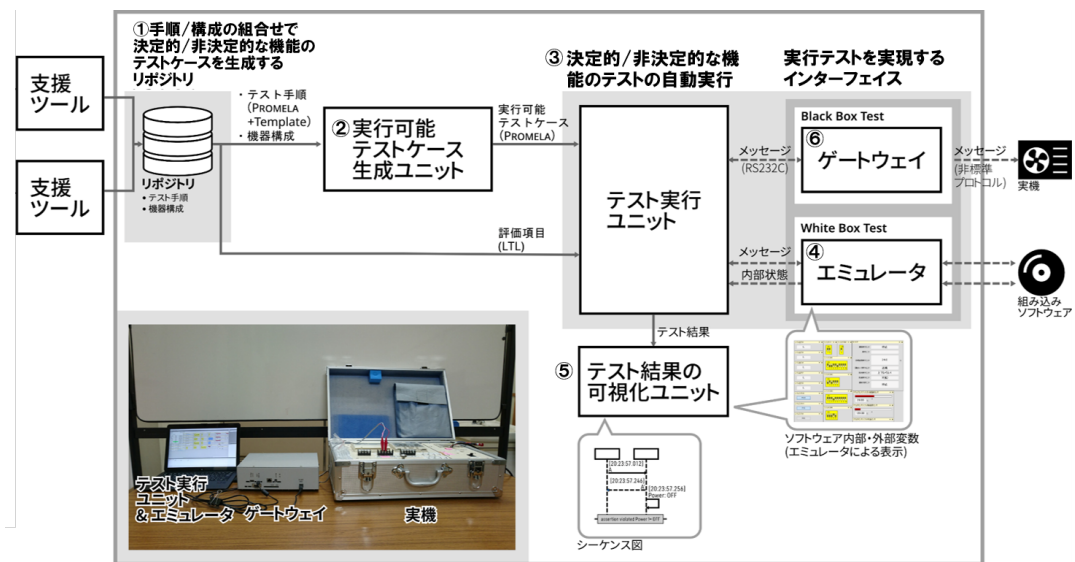


図 3.3 統合テスト実行環境の構成・機能

以降，①テストケース管理リポジトリ及び②実行可能テストケース生成ユニット，③テスト実行ユニット，④空調機器エミュレータ及び⑤テスト結果可視化ユニット，の順に各々のポイントを述べる。

テストケース管理リポジトリ・実行可能テストケース生成ユニット

本論文で提案するテストケース管理リポジトリは上述の通り，テスト手順・テスト構成と，ビル空調システム製品を構成する機器の組込みソフトウェアとを管理する。本論文では，テスト手順とはテスト対象である機能の事前条件・処理・事後条件や，各テストにおける機能の実行順序及び実行タイミング，それらが決定的か非決定的か，などの情報を指す。テスト構成とは，機器の種別（室外機/室内機/リモコン…）や，ビル空調システムにおいて各機器を特定するアドレス（以降，空調アドレスと称す）などの，テスト対象とするシステム構成を決定するための情報を指す。組込みソフトウェアはこのうちテスト構成に紐付けて管理し，テスト実施の際に，テスト構成に紐付けられたソフトウェア群が空調機器エミュレータで実行されるようにする。そして実行可能テストケース生成ユニットは，テスト実施時に，テスト手順・テスト構成の任意の組合せから，実行可能なテストケースを自動生成する。

テストケースを実行可能な形式とする狙いは，2.2節で述べた資産化の支援（読解の必要の低減，作成/統合/編集処理の機械化）と，テストの自動化による効率化である。加えて上述のようにテストケースをテスト手順・テスト構成に分割しこれを管理対象とすることで，2.1節で述べたテスト項目数の増加に対し管理データベース数を抑えることによっても，資産化の支援（テストケースの作成作業/管理作業の低減）を行う。具体的には例えば，組合せテストのための管理データベース数は，図3.4に示すように，（手順数×構成数）から（手順数＋構成数）に低減できると考える。

テスト実行ユニット

テスト実行ユニットは，実行可能テストケース生成ユニットが生成するテストケースに基づき，実機及び空調機器エミュレータ上で動作する組込みソフトウェアに対して自動実行テストを実施するプログラムである。自動実行テストは，モデル検査と同様の仕組みにて実施する。この機能が，2.3節に述べた，モデル検査と実行テストとを統一された仕組みで扱うというコンセプトへの対応となる。上流工程における評価と下流工程向けの評価を統一的な仕組みの下で支援することで，SPLEの回帰テスト工数を低減する。

テスト実行ユニットはモデル検査と自動実行テストを同様の仕組みで実施できるようにするため，既存のモデル検査ツールをベースとしつつ，以下に示すような実機とモデルとの差異を吸収する仕組みを備えている。

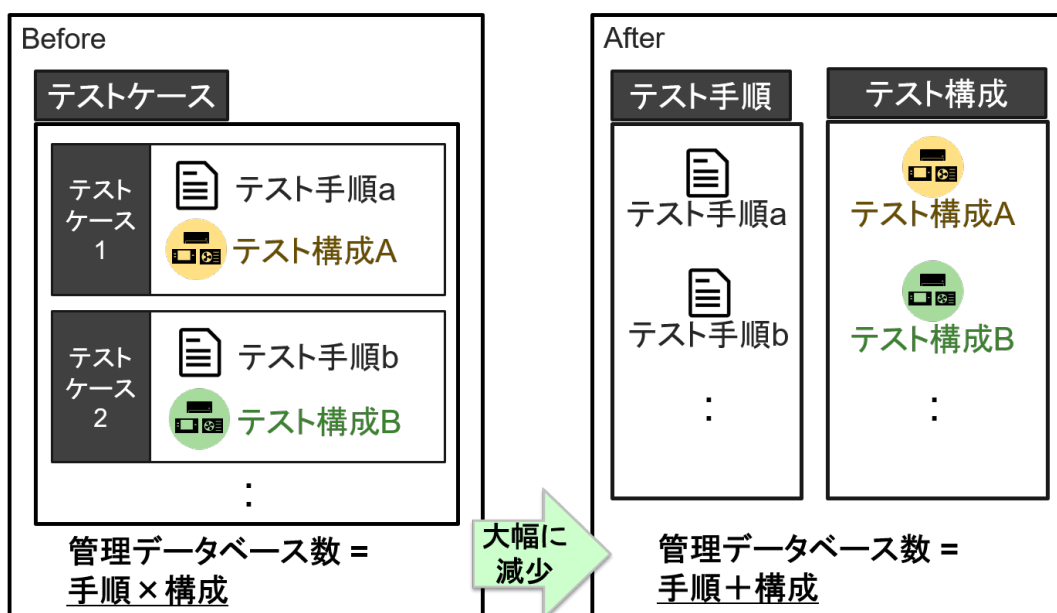


図 3.4 テスト手順とテスト構成の分割管理の効果

- (差異 1) モデル検査ではテストのための操作・確認をアルゴリズム的に処理するが、実行テストでは実機ないしエミュレータ上で動作する組込みソフトウェアの入出力を確認する必要がある
- (差異 2) モデル検査では状態遷移のタイミングは制御されないが、実行テストではテスト対象機器の状態が安定するまで待つなど、タイミングの制御が求められる
- (差異 3) モデル検査では一度検査した状態は次回以降通過しないようにテスト経路が生成されるが、実行テストではテスト対象機器を、テストの度に初期状態に戻すことが求められる

空調機器エミュレータ・テスト結果可視化ユニット

空調機器エミュレータのポイントは、ビル空調システム全体を 1 台の PC で動作させる構成 (図 3.5)、テスト対象機器を実機、対向機を空調機器エミュレータで動作させる構成 (図 3.6) の両方を取れるようにする所にある。前者の構成では、テスト対象組込みソフトウェアの内部パラメータをエミュレータによって確認することができるため、ホワイトボックステストを実施できる。後者の構成では、テスト対象組込みソフトウェアの実機搭載時の動作を確認する、ブラックボックステストを実施できるようにする狙いがある。

システム全体を 1 台の PC で動作可能とするため、空調機器エミュレータは以下の特長

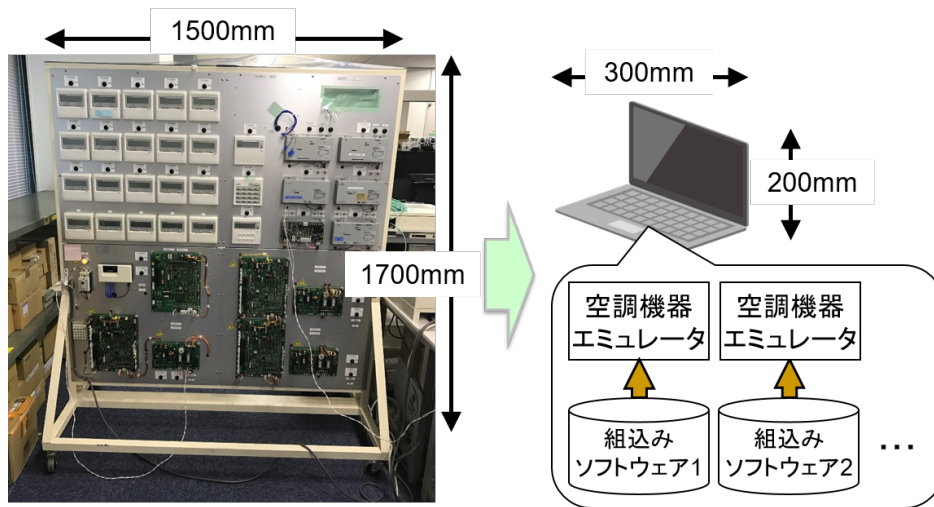


図 3.5 ビル空調システム製品全体を 1 台の PC で動作させるイメージ図

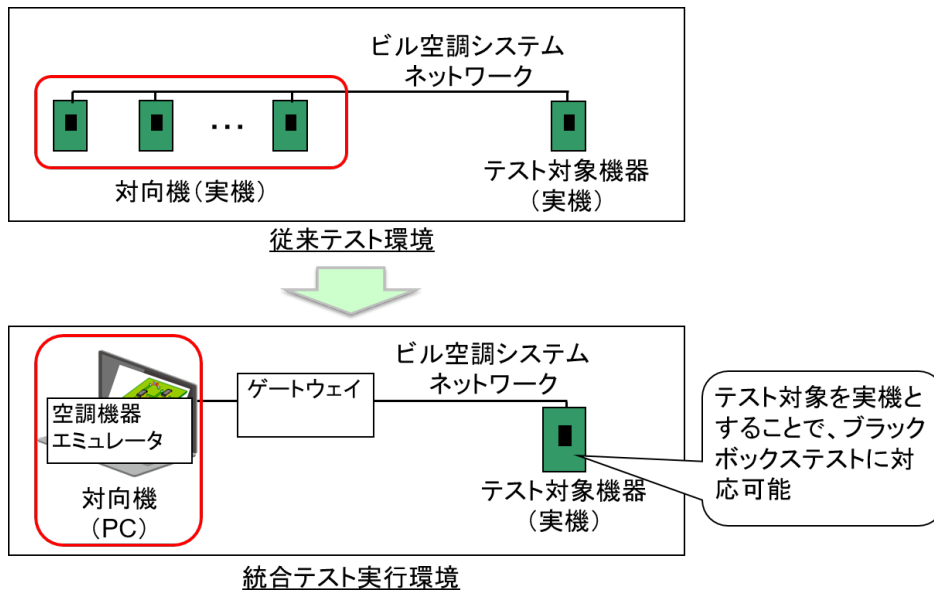


図 3.6 テスト対象機器を実機，対向機を空調機器エミュレータで動作させる構成

を備える。これらの特長によって、PC上で動作させた場合においても実機で動作させた場合と同様に、複数機器が協調して実現する機能が動くようにする。

- 1台のPC上で、複数の設備機器プログラムを同時に実行可能
- 実行速度を実機と同等に制御可能
- 機器間通信をPCが対応する通信プロトコルで模擬可能

この機器間通信を模擬するPCが対応する通信プロトコルと、ビル空調システム実機の機器間通信で使用されている通信プロトコル（以降、ビル空調プロトコルと称す）との相互変換を行う機器が、図3.3中⑥のゲートウェイである。一度PCが対応する通信プロトコルで模擬した機器間通信を、このゲートウェイで本来のプロトコルに戻すことによって、実機と空調機器エミュレータで動作する組込みソフトウェアとの間での通信を可能とする。これにより図3.6の、テスト対象機器を実機、対向機を空調機器エミュレータで動作させる構成が取れるようになる。

そして機器間通信及び空調機器エミュレータ上で動作する組込みソフトウェアの入出力は、テスト結果可視化ユニットによって可視化する。ここで言う可視化とは具体的には、実機ではハードウェア構成に制約され操作/確認できない入出力についても操作/確認を可能とし、テストの実施・結果確認・不具合分析にかかる工数の低減を狙うことである。併せて、実機で使用可能であった入出力についてはテスト技術者のユーザビリティに配慮し、従来のテスト環境に近い手順で操作/確認をできるようにする。テスト結果可視化ユニットが提供する可視化のイメージを、図3.7に示す。機器間通信については、ログを図式表現にて表示する。サーミスタの値など従来テスト環境上で視認できない入力については、数値化を行い視認できるようにする。ディップスイッチ等の従来テスト環境にも設けられていた入出力については、実機の部品に近い外観・操作感を有するGUI部品を提供する。

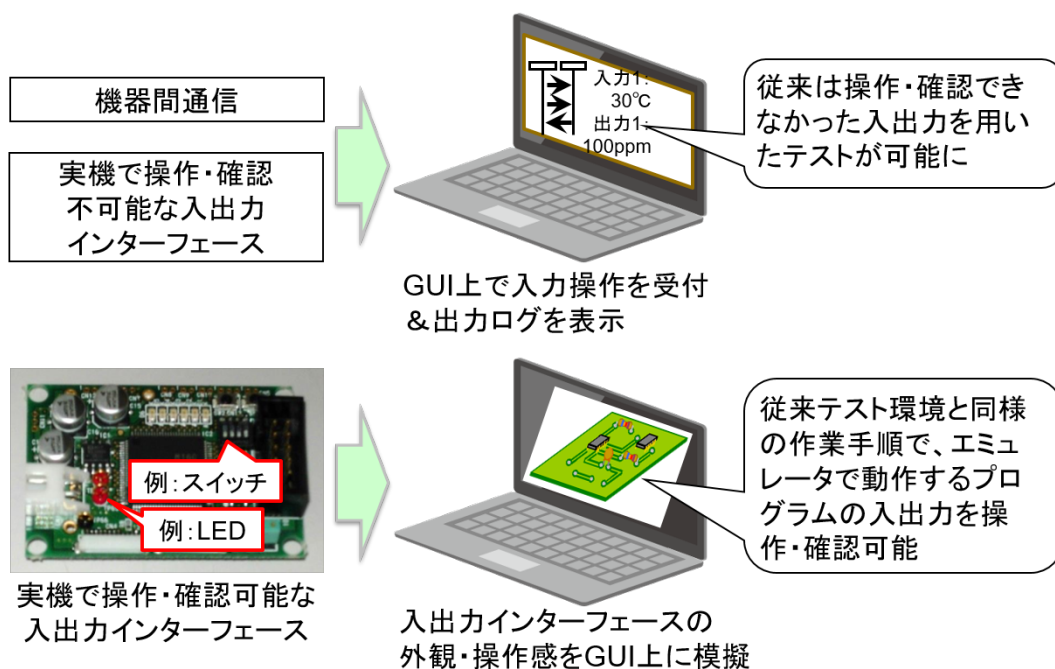


図 3.7 テスト結果可視化ユニットが提供する可視化

次章以降、本節で述べた統合テスト実行環境の各要素について、方式の詳細説明と検証を行う。第4章ではテスト実行ユニットがモデル検査と同様の仕組みで自動実行テストを実現する方式について詳細を述べ、表2.2に挙げた事例に適用し、得られた効果を確認する。第5章では空調機器エミュレータ・テスト結果可視化ユニットの実装と、ビル空調システム製品の回帰テストを対象とした実験的考察を述べる。第6章ではテストケース資産化を支援するテストケース記述方式・生成方式を詳細に説明するとともに、ビル空調システム製品の再発防止テストに適用した結果について検証する。第7章では第4章から第6章に述べた方式により構築した統合テスト実行環境が、ビル空調システム製品の回帰テストにもたらす効率化について評価する。

第4章

モデル検査技術とエミュレーション技術の融合

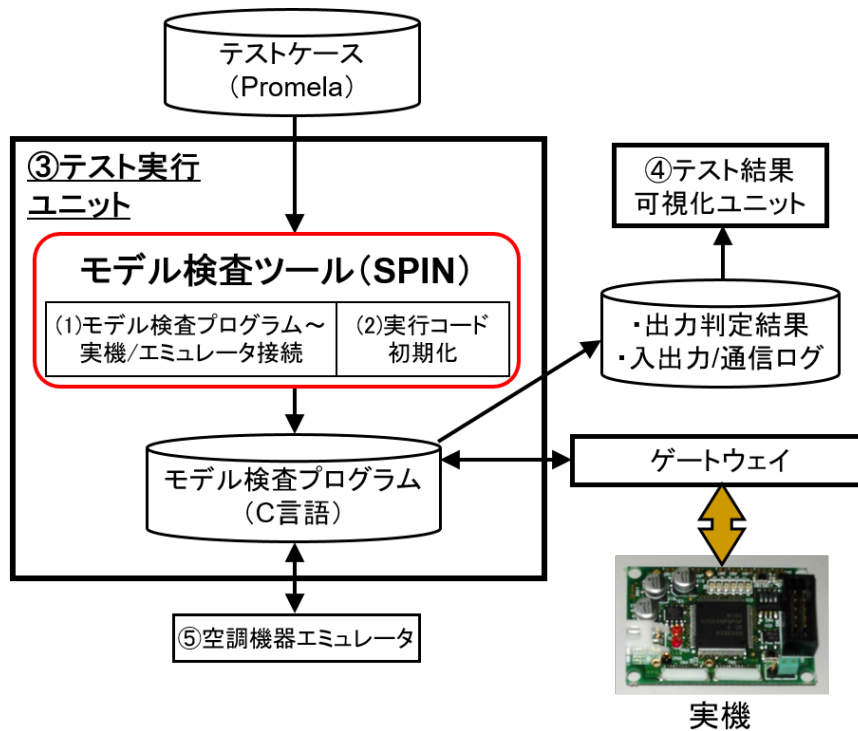
メッセージの交換順序及びタイミング，共通資源の獲得順序及びタイミングが非決定的であるビル空調システムの回帰テストを効率化するには，非決定的要素の自動評価が可能なモデル検査技術が有効と考えられることを，第2章で述べた．しかし SPLE への適用を考えたとき，上流工程向けの評価技術であるモデル検査と，下流工程向けの評価技術であるエミュレーション技術を活用した実行テスト環境は，統一的な仕組みで扱えること（テストケースを共通化し，SPLE 資産として再利用可能とすること）が望ましい．

本章ではこの課題を解決する，モデル検査技術とエミュレーション技術との融合によって，システムの非決定的振る舞いに対する実行テストをモデル検査の仕組みで実施可能とする方式を提案する．4.1 節では，提案方式の詳細を，統合テスト実行環境においてその実現をするテスト実行ユニット（図 3.3 中③）の構成と照らしつつ述べる．4.2 節ではこのテスト実行ユニットにて，表 2.2 に挙げたビル空調システムの後工程への流出不具合に対する再発防止テストを実施した結果を示す．4.3 節ではこの結果に対する考察を通じ，提案方式によって，ビル空調システムの非決定的要素に対する実行テストをモデル検査の仕組みで実施可能となったことを示す．

4.1 モデル検査ツールによる実行テストの実現

本節では，ビル空調システムの非決定的要素に対する実行テストをモデル検査の仕組みで実施可能とする方式の詳細を述べる．本方式は以下のアプローチに沿って構築しており，以降は，このアプローチ及び図 4.1 に示す本方式の実現であるテスト実行ユニットの構成に則して，方式の説明を行う．

- ビル空調システムの仕様（表 3.1 参照）に適合したモデル検査ツールを抽出し，テスト実行ユニットのベースとする
- 抽出したモデル検査ツールが実施するモデル検査と，SPLE を適用したビル空調システム開発で実施されている実行テストとのギャップを明らかにする
- 明らかにしたギャップを埋める機能を，モデル検査ツールに追加する



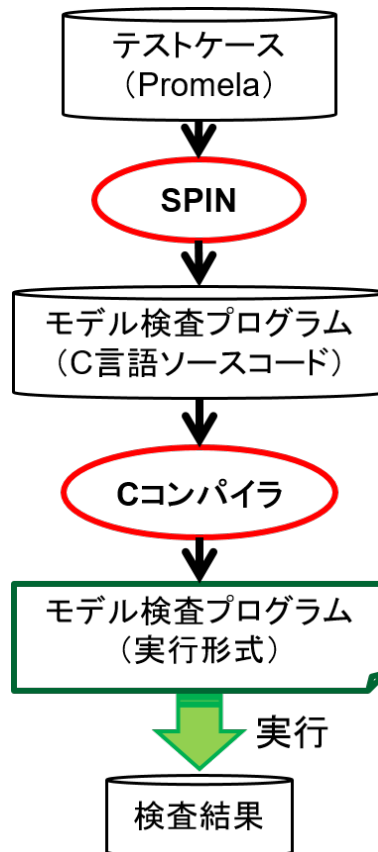


図 4.2 SPIN のモデル検査実行フロー

SPIN の入力は、Promela で記述されたシステムのモデル及びテストケースである。Promela では、システム内の各要素（3.2 節で述べたタイプ 1 においては機器，タイプ 2 においてはソフトウェアモジュール）をプロセスを表す「Proctype」に当てはめることで、システムをモデル化する。テストの際に行われる操作は Proctype 内メンバ変数への代入，結果判定は Proctype 内メンバ変数に対する LTL 式，機器間ないしソフトウェアモジュール間の通信はタイミングの制御がなされないプロセス間通信として記述される。実行テストでは当然ながら，操作・結果判定は実際の組み込みソフトウェアの入出力に対して行い，機器間ないしソフトウェアモジュール間の通信はビル空調プロトコルに則ったフォーマット・タイミングで実際の機器間ないしソフトウェアモジュール間で行われるため，この点が 1 つ，モデル検査と実行テストとのギャップとなる。このギャップを図 4.3 に，模式的に示す。

SPIN は Promela で記述されたテストケースが入力されると，実際のモデル検査を行

- (SPINの) モデル検査
各機器をプロセスとしてモデル化

```

inline OPERATE(){
status[Mode] = 4 /*Cooling*/
status[SettingTemperature] = 26
}

inline JUDGE(){
ltl p1 { []<> p }
}

inline COMMUNICATE(){
/*Send*/
mnet!frame;
/*Receive*/
mnet?<frame>;
}

```

テストケース (Promela) 抜粋

- 実行テスト

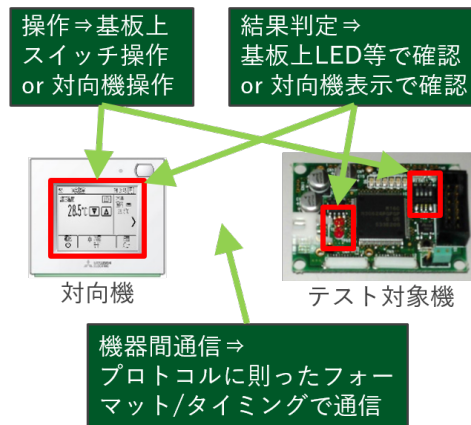


図 4.3 操作・結果判定・通信におけるモデル検査と実行テストとのギャップ

う C 言語のプログラム（以降，モデル検査プログラムと称す）を出力し，実行する．システムの非決定的振る舞いは図 4.4 に示すような Kripke 構造で表せるが，モデル検査プログラムはこの Kripke 構造に表れる状態の全てを網羅するように，テストを繰り返し実施する．ただし SPIN は「1 回遷移した状態はスキップする」という方針でテストを実施するため，対象システムによっては，2 回目以降のテスト実行経路は途中の状態からテストを始めるような経路となる（図 4.5 参照）．実行テストも，非決定的振る舞いを漏れなくテストしようとするならば Kripke 構造に表れる状態の全てを網羅するようにテストを繰り返し実施する点では変わりはないが，「前回のテストで設定された状態が実行コード上に残っていると，次のテストに影響し得る」ことを考慮し，「実行コードを初期状態に戻す処理（以降，初期化処理と称す）」をテストの度々実施している（図 4.6 参照）．この点が，モデル検査と実行テストとの 2 つ目のギャップである．

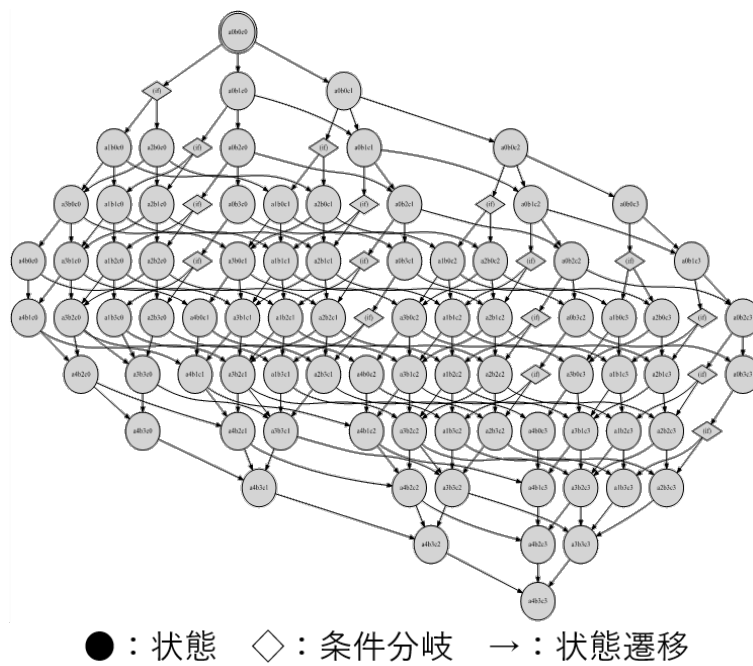


図 4.4 Kripke 構造によるシステムの非決定的振る舞いの表現

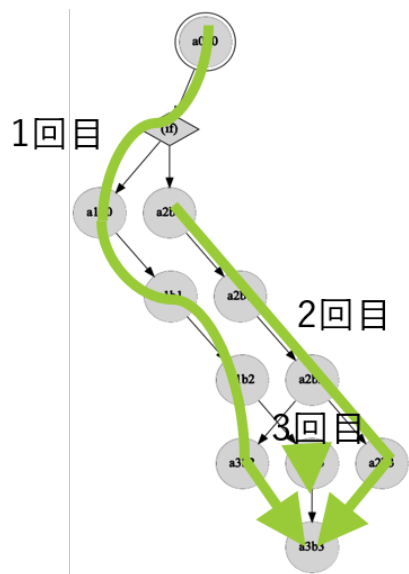


図 4.5 SPIN による非決定的振る舞いを網羅するテストの進め方



図 4.6 実行テストにおける非決定的振る舞いを網羅するテストの進め方

以上に述べた SPIN が実施するモデル検査と実行テストシステムに対して実施されているテストとの間の 2 点のギャップ，すなわち (1) 操作・結果判定・機器間ないしソフトウェアモジュール間通信処理の差異 (2) 非決定的振る舞いを網羅するテストの進め方の差異，を埋める手法を本論文では提案し，テスト実行ユニットにて実現する．以降ではこれらの手法，モデル検査プログラムと実機ないし空調機器エミュレータとを接続する手法 (図 4.1 中 (1))，モデル検査プログラムから実機ないし空調機器エミュレータを初期化する手法 (図 4.1 中 (2)) について述べる．

(1) モデル検査プログラム-実機/空調機器エミュレータ接続手法

提案する本手法では，テストケースに，モデル検査プログラムが実行する操作・結果判定・機器間通信ないしソフトウェアモジュール間通信を PC が対応する通信プロトコルで行わせる記述，またモデル検査プログラムがこれらの処理を実行するタイミングを実行コードと同等とする記述を埋め込んでおく．そして SPIN のモデル検査プログラム生成処理がこれらの記述を解釈できるようにしておくことで，モデル検査プログラムが実際にこれらの処理を実行するようになる．これにより，モデル検査プログラムが，実行テストにおける操作・結果判定・通信を実施することを可能にする．

実装 (図 4.7 参照) としては，実機/空調機器エミュレータと IP プロトコルで通信する API (入力操作，出力値要求，機器間通信コマンド送信/受信待ち，他) を提供する C++

DLL 形式の通信プログラムを用意し、この API を呼び出す処理を Promela に C 言語処理を埋め込む構文 (c_code) を用いてテストケース中に記載しておく。こうすることでモデル検査プログラム実行時にこの API が呼び出され、モデル検査プログラムと実機/空調機器エミュレータとの間で IP プロトコルによる通信が実施される。具体的には、実機に対しては、IP プロトコルとビル空調プロトコルを相互変換するゲートウェイを介することで通信を成立させる。空調機器エミュレータは、第 5 章に詳細を後述する IP プロトコルカプセル化機能によって、IP プロトコルと組み込みソフトウェアの入出力とを相互変換し、通信を成立させる。またコマンド受信待ち API などによって、モデル検査では制御されなかった通信タイミングも、実行コードに合わせての制御が可能となる。

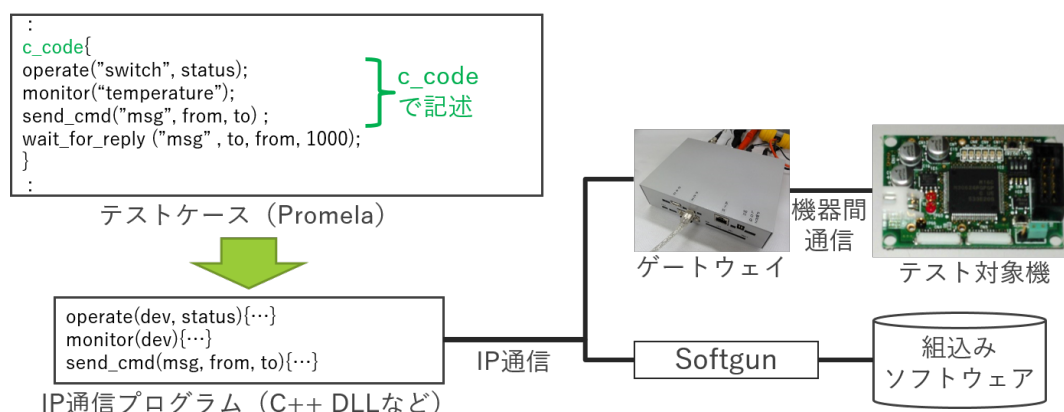


図 4.7 モデル検査プログラム-実機/空調機器エミュレータ接続手法の実装

(2) モデル検査プログラムからの実行コード初期化手法

上述したように、実行テストでは非決定的振る舞いを網羅するテストを行うような場合、初期化処理をテストの度に実施している。そこで Promela 形式のテストケースに初期化処理を記述できるようにし、SPIN に対しては、この記述が入力された際は既にその状態（初期状態）を取ったことがあっても毎回その状態に遷移するように、改修を加える。これにより、モデル検査プログラムが実行テストと同様に、実行コードに対する初期化処理をテストの度に実施できるようにする。

実装（図 4.8 参照）としては、実行コードを初期状態に戻すためのコマンドを JSON 形式で別途記述しておき、このコマンドを前述の実機/空調機器エミュレータと IP プロトコルで通信する API を用いて送信する処理を、テスト実施時に Promela の `init` ブロック（モデル検査プログラム実行時に最初に呼ばれる処理）に埋め込む。SPIN には、`init` ブ

ロックの内容を毎回実行するような改修を加えておく。こうすることでモデル検査プログラムは、実行コードを初期状態に戻すためのコマンドを、テストの度に送信する。

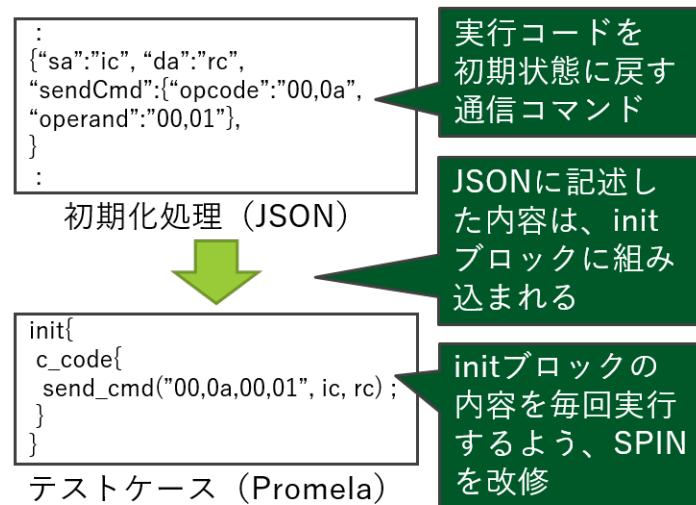


図 4.8 モデル検査プログラムから実行コードを初期化する手法の実装

以上の (1)(2) を備えたテスト実行ユニットによって、モデル検査と同様の仕組みで、実機ないし空調機器エミュレータの非決定的振る舞いを網羅する実行テストの自動実施を可能とする。

4.2 実験

前節で提案した方式に則し構築したテスト実行ユニットを用いての実験を通じ、提案方式によって、システムの非決定的振る舞いに対する実行テストをモデル検査の仕組みで実施できるようになるかどうかを評価した。本節では、この実験の内容と結果とを述べる。

4.2.1 実験内容

実験では、表 2.2 に挙げたビル空調システムの不具合事例 3 件に対応する再発防止テストを、テスト実行ユニットによって実施することを試みた。以下に実験手順を示す。

1. 被験者は、再発防止テストに対応する Promela 形式のテストケースを作成する。この際、テスト対象である不具合を発生させた空調機器は空調機器エミュレータまたは実機で動作させることを前提とする。

2. 被験者は、テスト実行ユニットにより作成したテストケースを実行し、判定結果、空調機器の入出力ログ及び通信ログを確認する。
3. 被験者は確認された内容より、実施したテストによって不具合の再発有無を確認できるかどうかを判断する。

4.2.2 実験結果

テスト実行ユニットを用いたテストによって、不具合の再発有無を確認できるかどうかを判断した結果を、表 4.1 にまとめる。

表 4.1 テスト実行ユニットを用いたテストによる不具合再発有無確認可否判断結果

テスト対象	再発有無確認可否判断結果	判断理由
[不具合 1]	確認可能	3つのメッセージを網羅的な順序で送信するテストケースを作成可能であり、実際に3つのメッセージが網羅的な順序で送信されていることを、ゲートウェイのログから確認できたため
[不具合 2]	確認可能	機能 A と機能 B を網羅的な順序で実行するテストケースを作成可能であり、どの順序で実行してもテスト対象機の状態が機能 A によるものになっていることを、実機画面で確認できたため
[不具合 3]	確認可能	空調機器エミュレータ上で動作する機器 D2 と機器 D3 の組込みソフトウェアに機能 C を実行させるテストケースを作成可能であり、各組込みソフトウェアが機能 C を実行していることを、空調機器エミュレータの入出力ログで確認できたため

上述した結果の補足を、以降に述べる。

ソースコード 4.1 は、[不具合 1] の再発防止テストに対応するテストケースとして被験者が作成した Promela の概要を示したものである。Promela の init ブロックで実行コードを初期状態に戻すためのコマンドの送信処理を記述するとともに、非決定的処理を記述する構文「do-od」の中に不具合をもたらした3つのメッセージの送信処理を、送信完了

後に機能の起動状態を確認するための通信処理を記述している。これにより、テストの度にテスト対象機（実機）を初期状態に戻しつつ、3つのメッセージの送信順序を網羅するテストが実施されるので、狙い通りの再発防止テストが実施可能と判断された。

ソースコード 4.1 不具合1の再発防止テストに対応するテストケース（概要）

```
1  :
2  init{
3    send_cmd('cmd_init');
4  }
5  :
6  do
7    :: flg1==0 -> send_cmd('cmdA'); flg1=1;
8    :: flg2==0 -> send_cmd('cmdC'); flg2=1;
9    :: flg3==0 -> send_cmd('cmdD'); flg3=1;
10 od
11 send_cmd('monitor_func')
12 wait_for_reply('monitor_func');
13 ltl p1 { []<> p }
14 :
```

この[不具合1]の再発防止テスト実行時の通信ログ（テスト2回分を抜粋したもの）を、図4.9に示す。テストケースでコマンドCとコマンドDの通信順序が非決定的であると記述した通り、テスト実行ユニットはコマンドCとコマンドDの通信順序を入れ替えてテストを行っていたことが確認できる。

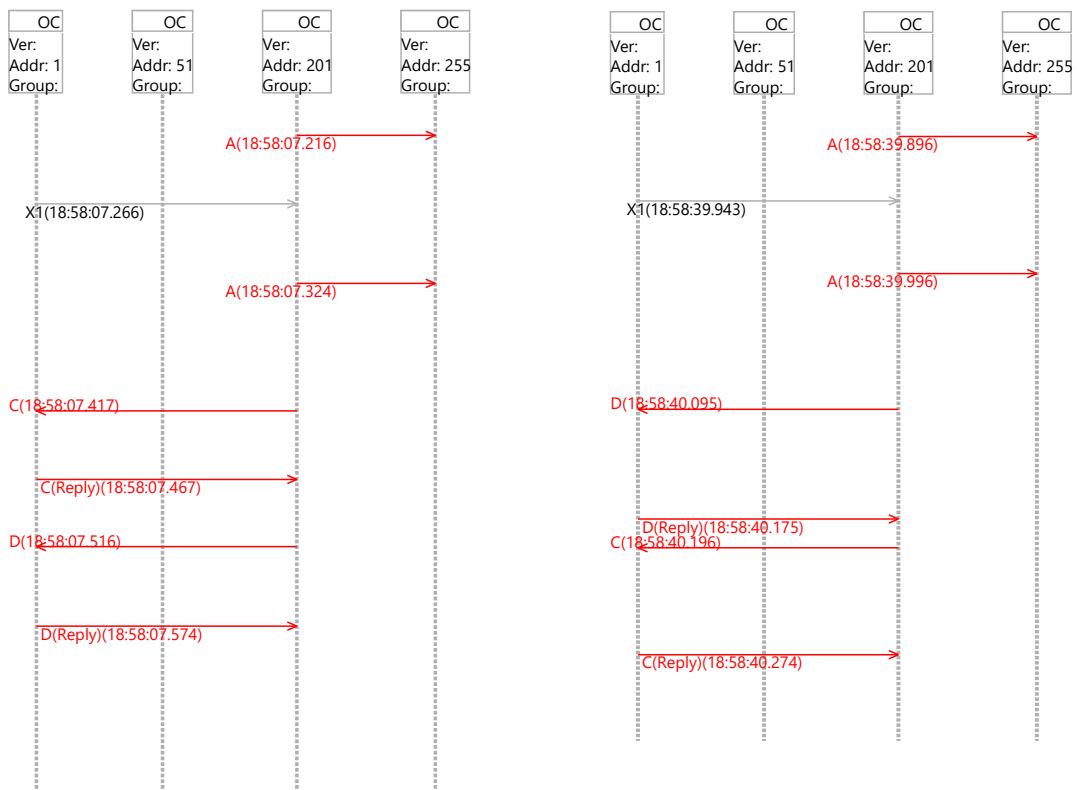


図 4.9 [不具合 1] の再発防止テスト実行時の通信ログ

ソースコード 4.2 は、[不具合 2] の再発防止テストに対応するテストケースとして被験者が作成した Promela の概要を示したものである。[不具合 1] の再発防止テストと同様、非決定的処理を記述する構文「do-od」で機能 A と機能 B を起動するコマンドを送信する処理を記述することで、機能 A と機能 B との実行順序を入れ替えてのテストが実施可能と判断された。

ソースコード 4.2 不具合 2 の再発防止テストに対応するテストケース（概要）

```

1  :
2  do
3  :: flg1==0 -> send_cmd('funcA'); flg1=1;
4  :: flg2==0 -> send_cmd('funcB'); flg2=1;
5  od
6  :

```

ソースコード 4.3 は, [不具合 3] の再発防止テストに対応するテストケースとして被験者が作成した Promela の概要を示したものである. このテストケースでは機器 D2 及び機器 D3 の組込みソフトウェアを動かしている空調機器エミュレータに, 機能 C を実行させるコマンドを送信している.

ソースコード 4.3 不具合 3 の再発防止テストに対応するテストケース (概要)

```
1  :  
2  send_cmd('funcC', D2);  
3  send_cmd('funcC', D3);  
4  :
```

4.3 考察

前節に示した実験結果に基づき, 本章で提案したモデル検査技術とエミュレーション技術を融合する手法が狙った効果を達成できるか, すなわちシステムの非決定的振る舞いに対する実行テストがモデル検査の仕組みで実施できるようになったかという視点で, 以下考察を述べる.

提案手法の実現であるテスト実行ユニットによって, 再発防止テスト 3 件について, ビル空調システムの回帰テストがモデル検査の仕組みで実施可能となることを確認した. 具体的には,

- 4.1 節に述べた (1) モデル検査プログラム-実機/空調機器エミュレータ接続手法によって, [不具合 1] 及び [不具合 2] の再発防止テストではモデル検査プログラムと実機との間で通信を成立させられたことを, [不具合 3] の再発防止テストではモデル検査プログラムと空調機器エミュレータとの間で通信を成立させられたことを確認した.
- 同じく (2) モデル検査プログラムからの実行コード初期化手法によって, モデル検査プログラムから実機ないし空調機器エミュレータの上で動作している実行コードを, テストの度に初期化できることを確認した.

また不具合 3 件の要因 (通信順序の考慮不足/機能の衝突/機器のモードの衝突) は, 2.1 節にて述べたビル空調システム製品の非決定的振る舞いの本質に, そのまま対応している. 実験においてこれらの再発有無が確認できたことから, 本提案手法はビル空調システ

ム製品の非決定的振る舞いに対する実行テストを，モデル検査と同じ仕組みで実施可能にすると判断した．そしてモデル検査と実行テストとを同じ仕組みで実施可能とできたことから，SPLE を適用したビル空調システムの開発において，本提案手法は上流工程向けテストケース・下流工程向けテストケースを共通化し，SPLE 資産としての再利用を可能にすると考える．

第5章

エミュレーション技術を活用したテスト関連作業の効率化

SPLE の回帰テストにおいて、テスト対象機器の準備及び接続作業、テストの実施・結果確認作業、不具合分析作業の作業量が増加していることを、ビル空調システム製品の事例とともに 2.1 節で述べた。しかし組込みソフトウェアを PC 上で動作させるエミュレーション技術を活用し、実機レステスト環境を構築できれば、これらの作業を効率化できる見込みがある。本章ではこの、エミュレーション技術を活用してテスト対象機器の準備及び接続作業、テストの実施・結果確認作業、不具合分析作業を効率化する方式について論じる。

本章は 4 つの節から構成される。5.1 節では、空調機器エミュレータ（図 3.3 中④）によって、テスト対象機器の準備及び接続作業を不要とするためにシステム全体を 1 台の PC で動作させる構成（図 3.5）、ハードウェアとのインタラクションを確認するテストなどにも適用可能とするためにテスト対象機器を実機・対向機 PC で動作させる構成（図 3.6）を取れるようにするための手法について述べる。5.2 節では、テストの実施・結果確認作業及び不具合分析作業の効率化を目的とした、空調機器エミュレータ上で動作する組込みソフトウェアの入出力を可視化する手法を、テスト結果可視化ユニット（図 3.3 中⑤）の構成と照らしつつ説明する。5.3 節ではこれらの手法を評価する。すなわち、空調機器エミュレータ及びテスト結果可視化ユニットの実装と、これらをビル空調システム製品開発で実際に行われている回帰テストに適用した結果を示す。この結果を受けて 5.4 節では、提案手法が効果的であった状況について考察し、提案手法がテスト対象機器の準備及び接続作業、テストの実施・結果確認作業、不具合分析作業の効率化に寄与することを示す。

5.1 実機-エミュレータ間協調・エミュレータ-エミュレータ間協調の実現

本節では、システム全体を1台のPCで動作させる構成(図3.5)、テスト対象機器を実機・対向機をPCで動作させる構成(図3.6)を取れるようにすることを目的とした、実機-エミュレータ間の協調及びエミュレータ同士での協調を可能とする手法について説明する。

ビル空調システムのように複数の組込み機器が協調可能であるということは、言い換えれば、各機器が同じ通信プロトコルに対応しているということである。このため通信プロトコルは機器共通の仕様として定義され、そのプロトコルで定められた通信フォーマットや通信タイミングを遵守するよう、各機器の通信機能(ハードウェア及びソフトウェア)は開発される。しかし通信を行う一方ないし双方をエミュレータで動作させた場合、その入出力はソフトウェアやPCが有するハードウェアで模擬され、またプログラムの実行速度はPCのCPU処理速度などに依存し変化する可能性があるため、前述の通信フォーマットや通信タイミングが実機とは変わってしまうことが考えられる。以降ではこの問題を解決する、空調機器エミュレータの入出力及び通信を実機及び他の空調機器エミュレータと送受する手法、空調機器エミュレータ上で動作する組込みソフトウェアの実行速度を実機同等に調整する手法について説明する。

5.1.1 エミュレータ入出力/通信送受手法

本研究では、空調機器エミュレータの入出力及び通信を実機及び他の空調機器エミュレータと送受可能とするために、これらをPCが対応する通信プロトコル(TCP/IPなど)でカプセル化する手法を提案する。詳細を以下に述べる。

準備として、空調機器エミュレータ上で動作する各組込みソフトウェアが保持する空調アドレスに、PCが対応する通信プロトコルにおけるアドレス(TCP/IPの場合、IPアドレスとポート番号)を対応付けておく。同様に、ゲートウェイに割り付けられているPCが対応する通信プロトコルにおけるアドレスと、実機で動作する各組込みソフトウェアが保持する空調アドレスも対応付けておく。そうして作成した、実機及び空調機器エミュレータ上で動作するすべての組込みソフトウェアについてのこの対応関係を集約したテーブル(以降、アドレス変換テーブルと称す)を、空調機器エミュレータが参照可能なように記憶しておく。

テスト実施時には、1つの組込みソフトウェアを1つの空調機器エミュレータプロセス

上で実行する。すなわち PC 上で動作させる組込みソフトウェアの数だけ、空調機器エミュレータのプロセスが起動することになる。カプセル化は各実機及び空調機器エミュレータ上で動作する組込みソフトウェアが通信処理を開始して以降、前述のアドレス変換テーブルを用いて実行され、空調機器エミュレータの入出力及び通信は実機及び他の空調機器エミュレータと送受可能となる。

ここより具体例として、PC に対応する通信プロトコルとして TCP/IP を採択し、通信の送信元が空調機器エミュレータ、送信先が実機である場合の、ビル空調プロトコルコマンドの送信手順を以下 (S1)~(S3) に示す (図 5.1 参照)。

- (S1): ソフトウェアで模擬されたビル空調プロトコル通信ポートから組込みソフトウェアが発する信号を、ビル空調プロトコルのフォーマットに変換
- (S2): (S1) にて得たビル空調プロトコルコマンドを透過的に IP ペイロードに乗せ (IP プロトコルでのカプセル化)、コマンドの宛先を示す空調アドレスにアドレス変換テーブルにて対応付けられていた IP アドレス (この場合、ゲートウェイの IP アドレス) とポート番号とを得て宛先とし、IP パケットを送信
- (S3): (S2) のパケットを受信したゲートウェイは、IP ペイロード、すなわちビル空調プロトコルコマンドを抽出し、実機に送信

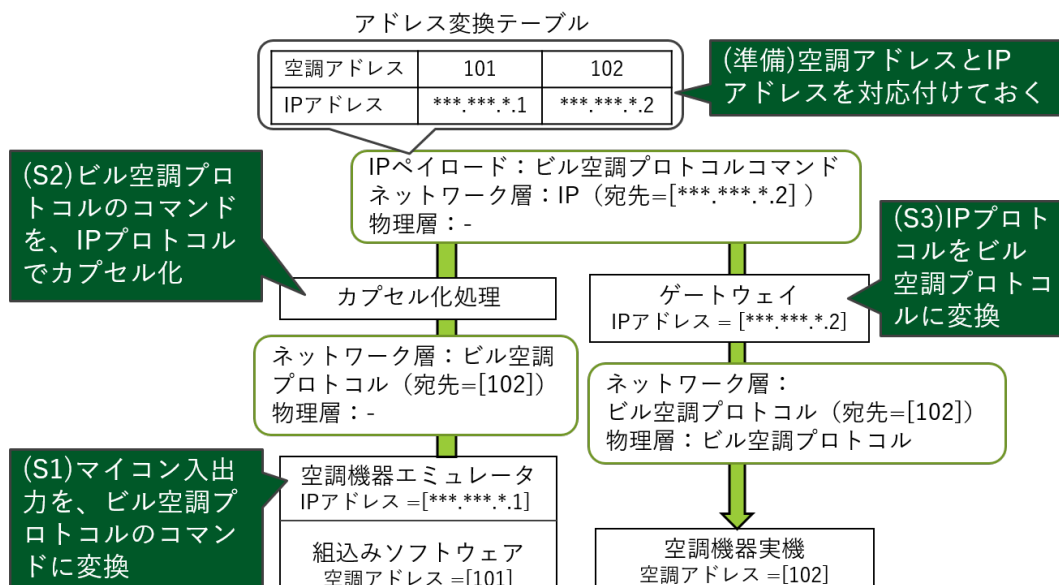


図 5.1 ビル空調プロトコル通信コマンドの送信手順 (送信元：空調機器エミュレータ、送信先：実機)

通信の送信元が実機、送信先が空調機器エミュレータである場合の手順は、以下の (T1) ~ (T2) となる。

(T1): 実機より送信されたビル空調プロトコルコマンドを透過的に IP ペイロードに乗せ、またビル空調プロトコルコマンドが含む宛先を示す空調アドレスをアドレス変換テーブルにより IP アドレスとポート番号とに変換し、これを宛先とする IP パケットを送信

(T2): (T1) のパケットを受信した空調機器エミュレータは、ソフトウェアで模擬されたマイコンのビル空調プロトコル通信ポートより組み込みソフトウェアが受け付ける信号に変換し、入力

通信する双方が空調機器エミュレータである場合は、以下の (U1) ~ (U3) に示す手順となる。

(U1): (S1) に同じ

(U2): (S2) に同じ

(U3): (T2) に同じ

5.1.2 空調機器エミュレータ上組み込みソフトウェアの実行速度調整手法

空調機器エミュレータが行う通信のタイミングを実機と合わせるにあたって、本研究では、空調機器エミュレータ上で動作する組み込みソフトウェアの実行速度を実機同等に調整する方針を採った。そしてこの調整には、OSS の CPU エミュレータ Softgun[73] で採用されている方式を適用することとした。

Softgun が行うエミュレータ上で動作する組み込みソフトウェアの実行速度を実機同等に調整する方式を、図 5.2 に示す。まず、組み込みソフトウェアがエミュレータ上で起動されて以降、実行された命令数をカウントしておく。この命令数と、あらかじめ与えておいた実機の CPU 速度から推定した実機で同経過時間内に実行されるであろう命令数とを 10ms 毎に比較し、前者の命令数が多ければ次の比較時までエミュレータを停止し、後者の命令数が多ければ、エミュレータを再生する。

この手法では、主記憶の参照などによる IPC (Instruction per Cycle) の変動にともなう誤差が残る。また表 3.1 で示した通りビル空調システムに接続される機器の CPU 処理速度は数 10MHz 程度であり、一般的な PC の CPU 処理速度より低速であるため、比較周期 (10ms) の間に誤差が生じることが予想されるが、この誤差は時間やエミュレートする組み込みソフトウェアの数によって累積しない。同じく表 3.1 で示した通りビル空調シス

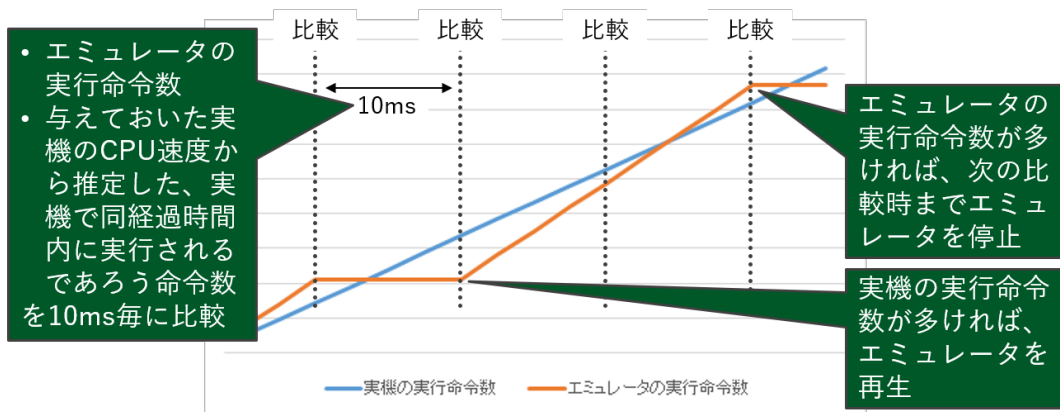


図 5.2 エミュレータ上で動作する組込みソフトウェアの実行速度制御

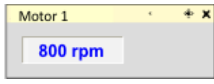
テムの通信タイムアウト時間は数秒程度ありこのオーダーの誤差は許容されるため、この許容誤差より十分に小さい上述の誤差は、本研究で構築するビル空調システム向けの統合テスト実行環境においては問題とならないと判断した。

5.2 エミュレータ入出力/通信の可視化

本節では 3.2 節で提案した、実機ではハードウェア構成に制約され操作/確認できない入出力及び通信を操作/確認を可能とすることで、テストの実施・結果確認・不具合分析にかかる工数を低減する、テスト結果可視化ユニットの詳細を説明する。

テスト結果可視化ユニットは 3.2 節に述べたように、サーミスタの抵抗値など実機上で操作/確認できない入出力については数値化し操作/確認を可能とする GUI 部品を、ディップスイッチなど実機上で操作/確認できる入出力については実機に近い外観・操作感を有する GUI 部品を提供する。この GUI 部品の例を数点、表 5.1 に示す。例えばサーミスタに対応する GUI 部品には、入力された温度の値を実機におけるサーミスタ計測回路の出力値に変換し、空調機器エミュレータ上で動作する組込みソフトウェアの所定のアナログ入力ポートに入力する機能を持たせる。この機能によって、所定の温度をこの GUI 部品に入力することで空調機器エミュレータ上で動作する組込みソフトウェアにサーミスタがその温度を検知した際の振る舞いをさせることができるため、当該温度に関するテストの実施が容易となる。

表 5.1 テスト結果可視化ユニットが提供する GUI 部品の例

入出力種別	I/O	外観	補足
サーミスタ	I		温度の値をスライダーまたはテキストボックスで入力
デジタル入力	I		Hi/Low をトグルボタンで切替
ディップスイッチ	I		各スイッチの ON/OFF を、ディップスイッチの外観を模擬したインターフェースにて切替
LED	O		各 LED の点灯/消灯を、実機に搭載の LED と同系統の色を使って表示. この例は、赤色 LED と緑色 LED に対応したもの
7セグ	O		7セグの表示状態を、実機に搭載の7セグと同様の外観にて表示
アクチュエータ	O		アクチュエータの動作状態を数値で表示. この例では、モータの回転数を表示

またこれらの GUI 部品は、空調機器エミュレータで動作中の組込みソフトウェアに対するリアルタイム表示に加えて、過去ログの表示に対応させる。過去ログ表示においては以下機能を持たせることで、テスト技術者が俯瞰性の高いデータによってテスト結果の確認・不具合分析を行えるようにする。

- 時系列グラフ形式での表示
- 入力/出力の変化，各入出力の対象システムにおける異常値，をキーとする検索

一方の通信については、異常通信を俯瞰的に観測できるよう、シーケンス図形式で可視化する GUI (図 5.3) を提供する。

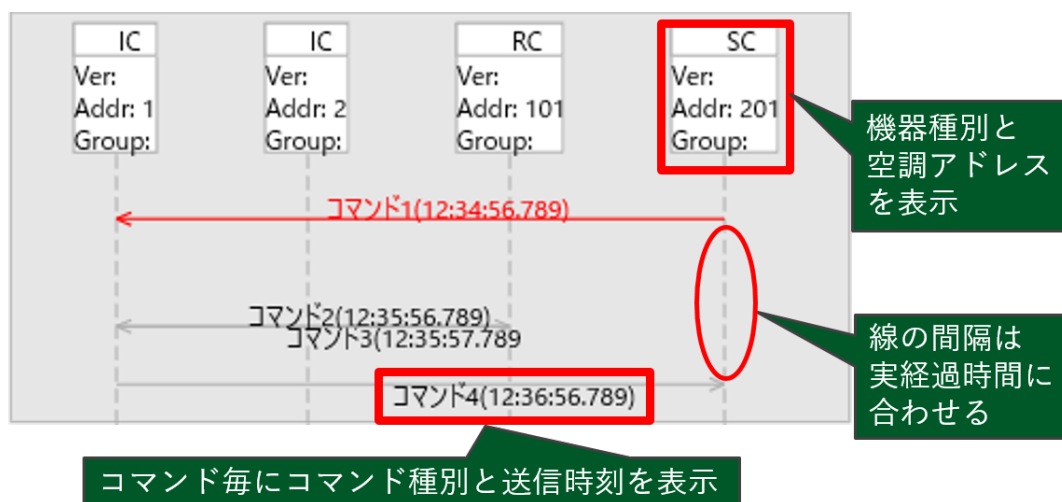


図 5.3 テスト結果可視化ユニットが提供する通信可視化 GUI

5.3 実験

5.1 節及び 5.2 節に述べた手法を評価するため、ビル空調システムをテスト対象とするプロトタイプを実装し、ビル空調システムの製品開発で実際に行われている回帰テストに適用する実験を行った。本節ではこのプロトタイプ（ACET：Air Conditioning Emulator Terminal）について、また実験の内容及び結果について述べる。

5.3.1 ACET (Air Conditioning Emulator Terminal)

ACET のソフトウェア構成を、図 5.4 に示す。ビル空調システムに接続される各機器の組込みソフトウェアを動作させる空調機器エミュレータは、5.1 節に挙げた手法（組込みソフトウェア毎にプロセスを立ち上げ、実行速度を実機同等に調整）に適合する Softgun（図 5.4 中 (C)）をベースとする。Softgun をベースとした理由には他に、以下の 2 点がある。

- 複数の組込みソフトウェアを 1 台の PC でエミュレート可能 (1 ソフト 1 プロセス)
- ソースコードが公開されており、製品開発現場でも改修が可能

空調機器エミュレータの入出力及び通信をカプセル化する PC が対応するプロトコルには、IP プロトコルを選択する。IP プロトコルによるカプセル化は空調機器エミュレータ

に内蔵した入出力/通信カプセル化機能（図 5.4 中 (D)）にて，IP パケットの送受は通信パケット分配器（図 5.4 中 (B)）にて行う。

テスト結果可視化ユニット（図 5.4 中 (A)）が提供する各入出力に対応する GUI 部品（表 5.1）は，機器毎にタブにまとめて表示する画面（図 5.5）として実装した。通信に対応するシーケンス図形式の GUI は，図 5.3 に示した外観を有する画面として実装した。

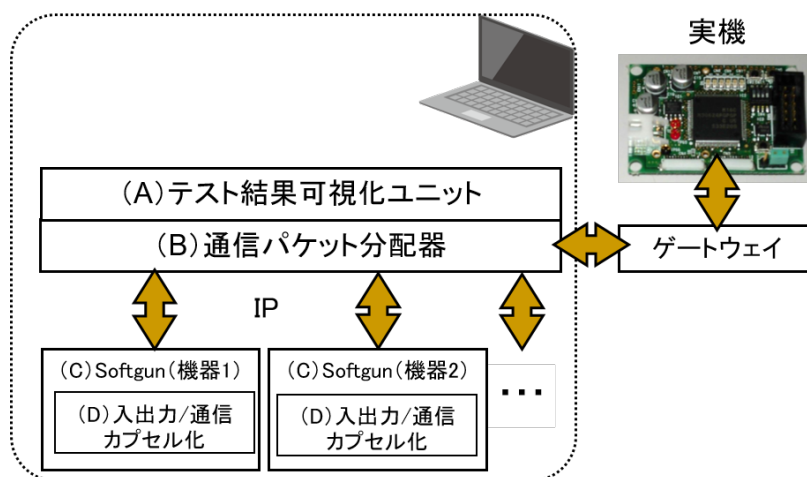


図 5.4 ACET の構成

5.3.2 性能確認

以上に述べた ACET をビル空調システム製品の回帰テストに適用する実験に先駆け，ACET がビル空調システム製品の全体を 1 台の PC で動作させることが可能かどうかを，以下の手順にて確認した。

- 市販 PC (CPU : デュアル, 2.9GHz, 8 コア 16 スレッド, メモリ : 32GB, Ethernet : 100BASE-TX) を用い，室内機が最大 50 台接続されるビル空調システム製品の，室内機の組込みソフトウェア（実機では 16MHz, 1 コアのマイコンで動作）を Softgun で動作させる
- 室内機の組込みソフトウェアを動作させる Softgun プロセスを，10 台分，20 台分…と 10 台刻みで増やしながら，CPU 負荷を計測
- CPU 負荷が 100 % となった時点で終了

結果を表 5.2 に示す。PC1 台の上で，60 台分の組込みソフトウェアを動作させられることを確認できた。当該ビル空調システム製品では 50 台の室内機に対し室外機 5~10 台



図 5.5 ACET におけるテスト結果可視化ユニット（入出力の表示）

程度を用意する構成が標準的であるため、当該ビル空調システム製品の最大構成を動作させるには、本確認で用いたような標準的なスペックの PC1 台で必要十分であることを確認できた。併せて、より CPU クロック周波数が大きい PC で同様の性能確認を実施したところ、より多台数分の組込みソフトウェアを動作させることが可能であることを確認できたため、今後ビル空調システムがより大規模な構成を取ることになっても、PC のスペックアップで対応可能と判断した。

表 5.2 室内機組込みソフトウェアを動作させる Softgun プロセス数に対する CPU 負荷測定結果

室内機台数（台）	10	20	30	40	50	60	70
CPU 使用率（％）	8.0	14.7	25.8	41.7	59.1	82.3	100

5.3.3 実験内容

ビル空調システムに接続される空調室外機製品 2 機種について、空調メーカーにおけるテスト技術者 3 名（うち当該室外機担当 2 名（被験者 X, 被験者 Y），他機種担当 1 名（被験者 Z））に、従来のテスト環境（図 2.2 参照）と ACET とを用いてテストを実施してもらった。以下に実験手順を示す。

1. 被験者に、ACET 機能を説明する。また、以降の作業に詰まった場合は補足説明を行うことを伝える。
2. 被験者 X は、室外機 2 機種 of テスト 444 項目（1 機種目 239 項目、2 機種目 205 項目）につき、ACET によって実施できるか否かを判断し、判断結果とその理由をコメントとして残す。
3. 被験者は、実験手順 2 で被験者 X が ACET で実施できると判断したテストを、従来のテスト環境を用い、図 5.6 に示す作業フローに則り実施する。また、同じテストを ACET を用いて、同じく図 5.6 に示す作業フローに則り実施する。この際、各作業項目について作業時間を計測しておく。
4. 被験者は、テスト中に見つかった不具合について、従来のテスト環境により原因特定作業を実施する。この際、作業内容と作業時間を記録する。
5. 被験者は、ACET により、実験手順 4 で記録した作業内容に沿って再度原因特定作業を実施する。この作業時間も計測する。
6. 各被験者について、作業中/作業後のコメントを記録する。

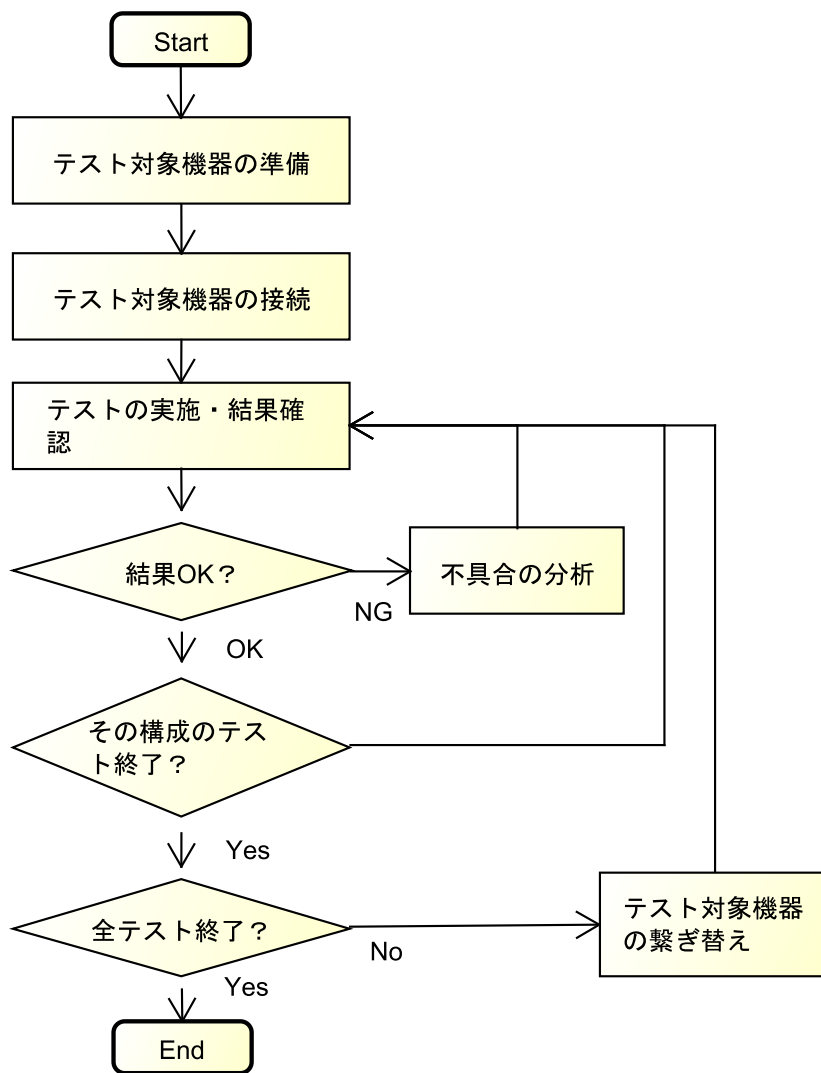


図 5.6 テスト作業フロー

5.3.4 実験結果

前述の手順により、以下に示す結果を得た。

表 5.3 に、被験者 X による ACET の適用可否に関する判断結果を示す。実験手順 2 では、444 項目のテストのうち 379 項目（約 85%）が、ACET によって実施可能と判断された。

表 5.3 ACET の適用可否に関する判断結果

適用可否に関する判断結果	テスト項目数	被験者 X によるコメント
適用可	379 (85.4%)	
適用不可	65 (14.6%)	ACET では操作・結果確認を実施できない

よって実験手順 3 で実施したテスト項目は 379 項目（1 機種目 201 項目，2 機種目 178 項目）となった。これらテスト項目に対する，被験者 3 名の作業時間（テスト対象機器の準備作業，テスト対象機器の接続・繋ぎ替え作業，テストの実施・結果確認作業）を表 5.4 にまとめる。

表 5.4 作業時間計測結果

テスト対象	作業項目	作業時間（時間）					
		従来テスト環境			ACET		
		X	Y	Z	X	Y	Z
室外機 1 (201 項目)	テスト対象機器の準備	0.5	0.4	0.8	0.1	0.1	0.1
	テストの実施，結果確認， テスト対象機器の接続・ 繋ぎ替え	10.9	10.2	15.7	6.0	5.1	6.6
	小計	11.4	10.6	16.5	6.1	5.2	6.7
室外機 2 (178 項目)	テスト対象機器の準備	0.5	0.4	0.6	0.1	0.1	0.1
	テストの実施，結果確認， テスト対象機器の接続・ 繋ぎ替え	9.8	9.0	11.9	5.1	4.7	5.9
	小計	10.3	9.4	12.5	5.2	4.8	6.0
合計		21.7	20.0	29.0	11.3	10.0	12.7

以下，主要な結果を説明する。

- テストの実施，結果確認，テスト対象機器の接続・繋ぎ替えにかかる時間は，機種・被験者によらず 40%～50% 程度短縮された。最も時間が短縮されたのは室外機 1

を対象とした被験者 Z の作業で、15.7 時間から 6.6 時間と、9.1 時間（約 58%）の短縮となった。

- 作業時間の合計は、当該室外機担当である被験者 X・被験者 Y では 21.7 時間から 11.3 時間と 10.4 時間（約 48%）、20 時間から 10 時間と 10 時間（約 50%）の短縮であり、被験者 Z では 29 時間から 12.7 時間と 16.3 時間（約 56%）短縮された。

テスト中に見つかった不具合の件数、および不具合分析にかけた作業時間を、表 5.5 に示す。ACET は従来のテスト環境に対し、過不足無く不具合を抽出することができた。各不具合の原因特定作業については、表 5.5 中「不具合 1 の分析作業時間」で被験者 Z が 3.1 時間から 1.4 時間と 1.7 時間（約 55%）の短縮を実現した一方、「不具合 2 の分析作業時間」では被験者 X・Y が 85% 程度の短縮となり、被験者 Z は従来テスト環境でも ACET でも原因を特定できなかった。

表 5.5 不具合分析作業に関する実験結果

項目	従来テスト環境			ACET			補足
	X	Y	Z	X	Y	Z	
不具合件数	2	2	2	2	2	2	両者は同じ不具合を検出
不具合 1 の分析作業時間（時間）	1.4	1.0	3.1	1.2	0.8	1.4	原因：室外機と室内機との間に流れる通信コマンドの値誤り
不具合 2 の分析作業時間（時間）	2.1	2.3	-	0.3	0.3	-	原因：同一の入出力を参照・操作する 2 つの機能の衝突

5.4 考察

前節に示した実験結果に基づき、本章で提案したエミュレーション技術を活用したテスト実行環境が狙った効果を達成できるか、すなわち、テスト対象機器の準備及び接続、テストの実施、結果確認、不具合の分析にかかる作業を効率化するかという視点で、以下考察を述べる。

(1) 提案手法の適用可能範囲について

表 5.3 に挙げた通り、ACET は室外機の回帰テストの 85% に適用可と判断された。ビル空調システムにおける回帰テストは、機種や機能が異なっても、抽象的にはおおよそ同

じ内容である（例えば，入力を操作して，通信を実施させて，出力を確認する，という点は機種や機能が異なっても変わらない）．よって本提案手法は，ビル空調システム製品の回帰テストの多くに適用可能と考えられる．また適用不可と判断された理由を確認したところ，ハードウェアのテスト，別基板上の入出力を用いておりテスト結果可視化ユニットが対応できていなかったテスト，の2種に大別できた．このうち後者は，基板間通信に5.1節に述べた手法を適用することで解決可能と考えられる．つまり，適用範囲をより拡大できる見込みがある．

(2) テスト対象機器の準備及び接続作業，テストの実施・結果確認作業の効率化について

本実験において，テスト対象機器の準備時間，テストの実施，結果確認，テスト対象機器の接続・繋ぎ替えにかかる時間は，表5.4に示したように全ての機種・被験者で短縮となった．中でも被験者Zについては，従来テスト環境における作業時間合計は被験者X・Yと比較して差があったところ，ACETにおける作業時間合計では大きな差が見られなくなるまでに作業時間を短縮できていた．この被験者Zの作業詳細を確認すると，被験者Zは実機の接続方法を習得していなかったために，従来テスト環境では機器の繋ぎ替えに多くの時間を費やしていた．ACETの空調機器エミュレータではこの繋ぎ替えが不要となるため，大幅な作業時間短縮効果に繋がった．

以上よりACETは，テスト対象機器の準備及び接続，テストの実施，結果確認にかかる作業を効率化できたと判断する．特に，専門性を有しないテスト技術者による作業に対して，大きな効果を発揮すると思われる．

(3) 不具合分析作業の効率化について

表5.5に挙げた通り，本実験において，ACETは従来テスト環境と同じ合否判定結果を導出できたため，空調機器エミュレータの動作と実機の動作とに不整合は無かったと考える．一方，不具合原因特定の成否および原因特定作業にかかる時間の短縮効果は，不具合によって差異があった．これは，不具合原因の種別によるものと推定する．以下，その論拠を述べる．

1つ目の不具合は，室外機と室内機との間に流れる通信コマンドの値誤りが原因で，空調運転状態の異常をもたらすものであった．空調運転状態の異常・通信コマンドの値は，従来テスト環境では基板上LEDおよびスニファで，ACETではテスト結果可視化ユニットで確認できた．このため被験者X・Yは，従来テスト環境による作業とACETによる作業を同様に進められたが，これには，テスト結果可視化ユニットが基板上LEDと同様の見た目であることも作用していたと推測される．一方で被験者Zは，「空調運転状態の

異常をもたらす通信コマンドの値」を知らなかったため、従来テスト環境では、空調運転状態の異常をもたらす原因の候補を調査し、その候補を一つ一つ潰していく作業を行うこととなった。対して ACET では、5.2 節に述べた入出力の異常値をキーとして過去ログを検索する機能によって通信コマンドの値誤りを早い段階で発見できたため、作業時間が大幅に短縮された。

2 つ目の不具合は、同一の入出力を参照・操作する 2 つの機能の衝突が原因であった。具体的には、2 つの機能がそれぞれ任意のタイミングで動作する仕様となっており、一方の機能で操作された出力に対して返ってきた入力を、他方の機能が自機能のための入力として参照してしまい、意図しない動作をもたらすタイミングがあった (図 5.7)。この入出力は室外機の基板内で処理されるものであったため、実機においては、原因特定のために基板パターン上にロジアナを繋いで観測する、デバッガによりプログラムをステップ実行する、という作業が必要となった。また特定のタイミングでのみ発生するため、不具合が再現するまで実機を何度も操作する作業も発生した。一方で ACET は、当該入出力を PC 画面上で確認できるために不具合の再現が容易であった。被験者 X・被験者 Y はこの点に気付き、結果、短時間で不具合特定を実現できた。しかし被験者 Z は、「ある 1 つの入力を 2 つの機能が参照する」仕様を知らなかったため、調査対象を絞り込めずに終わった。

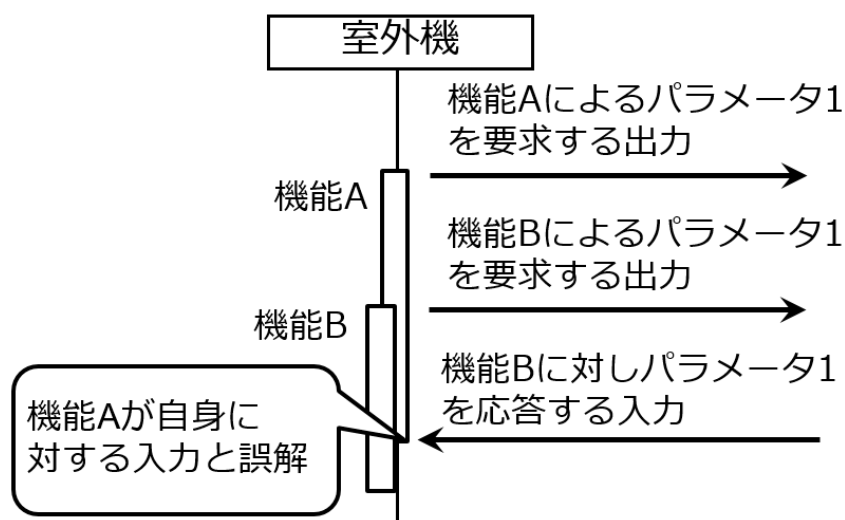


図 5.7 不具合となった機能衝突

よって ACET は、以下の通り、不具合原因特定作業を効率化できると判断する。

- 特定に専門性を要する不具合 (影響が表れるデータを視認できない、タイミングに

依存する，など)については，専門性を有するテスト技術者の作業時間を短縮する。

- 特定に専門性を要しない不具合（影響が表れるデータを視認できる，など）については，専門性の差異を吸収する。

(4) 他システムに対する効果について

以上，ビル空調システム製品の回帰テストに対する提案手法の効果を考察したが，他の組込み機器製品の回帰テストにおいても同様の効果を期待できる。ACET が効率化した作業（テスト対象機器の準備・繋ぎ替え，テストの実施・結果確認，不具合の分析）は他の組込み機器製品のテストでも同様に行われる作業であるし，分析作業の効率化が確認できた不具合（通信コマンドの値誤り・機能の衝突）もビル空調システム固有のものではなく，他の組込み機器製品でも起こり得るためである。

第 6 章

テストケースの部品化による実行可能なテストケースの資産化支援

SPLE 資産としてテストケースの再利用を可能にするテストケースの資産化が，SPLE を適用したビル空調システム製品開発の回帰テストにおいて，テストケース設計にかかる工数を低減できる見込みがあることは，第 2 章で述べた。しかし製品のグローバル展開にともない回帰テスト項目数が膨大となっており，テストケースが再利用可能となっても，追加作成作業・管理作業を効率化する支援が必要であることを併せて述べた。本章では，実行可能なテストケースをテスト手順・テスト構成に部品化することで，テストケースの資産化を支援する手法について論じる。

6.1 節では，膨大なテストケースの追加作成作業・管理作業の効率化のために，実行可能なテストケースをテスト手順・テスト構成に部品化する考え方について述べる。6.2 節では，部品化したテスト手順・テスト構成をテスト実施時に任意に組み合わせ，実行可能なテストケースを生成する手法について述べる。6.3 節では，前述の手法をビル空調システム製品開発へ適用するために検討した，テストケース管理リポジトリ（図 3.3 中①）及び実行可能テストケース生成ユニット（図 3.3 中②）を説明する。6.4 節では，このテストケース管理リポジトリ及び実行可能テストケース生成ユニットを，ビル空調システム製品の再発防止テストに適用した結果を示す。この結果について 6.5 節で考察し，提案手法がビル空調システム製品開発の回帰テストにおいて，テストケース設計作業の効率化に寄与することを示す。

6.1 テスト手順・テスト構成への部品化

資産化するテストケースを実行可能な形式とすることで、読解の必要の低減及び作成・統合・編集処理の機械化による資産化の支援と、テストの自動化による効率化が見込める。またテストケースをテスト手順・テスト構成に部品化することによって、管理データベース数を低減し、テストケースの追加作成作業・管理作業を低減することによっても、テストケースの資産化を支援できると考える。本節ではこの、実行可能なテストケースをテスト手順・テスト構成に分割する考え方について論じる。

本論文では、テスト手順・テスト構成はそれぞれ以下を記述するものと定義する。

テスト手順

- 機能の階層構造
- 機能の事前条件・処理・事後条件
- 機能の実行順序/タイミング，及びその順序/タイミングが決定的か非決定的か
- 機能の評価項目（テストの際の合否判定基準）

テスト構成

- 機器/ソフトウェアの種別（ビル空調システムの場合，室外機/室内機/リモコン…）
- 機器種別毎の接続台数/ソフトウェア種別毎の搭載モジュール数
- 各機器/ソフトウェアを特定するアドレス（ビル空調システムの場合，空調アドレス）

このうち、テスト手順について以下補足する。ビル空調システムを初めとする複数の機器ないし複数のソフトウェアが協調して機能を実現するシステムでは一般的に、機能は、機器間ないしソフトウェア間の通信セッションが複数実行されることで実現される。そしてこの機能が複数合わさることで、システムのユースケースは成立している。つまり本研究が対象とするシステムでは、ユースケース-機能-セッションを階層関係で表現することができると考えられる（図 6.1 参照）。ユースケース-機能-セッションは各々が、それぞれの階層における事前条件・処理・事後条件を有し、また実行順序/実行タイミングが決定的か非決定的かの性質を有する。例えば機能の実行順序が非決定的であるとは、あるユースケースにおいて、そのユースケースを構成する複数の機能が、ランダムな順序で実行されるということである。同様に、セッションの実行順序が非決定的であるとは、ある機能において、その機能を構成する複数のセッションが、ランダムな順序で実行されると

いうことである。

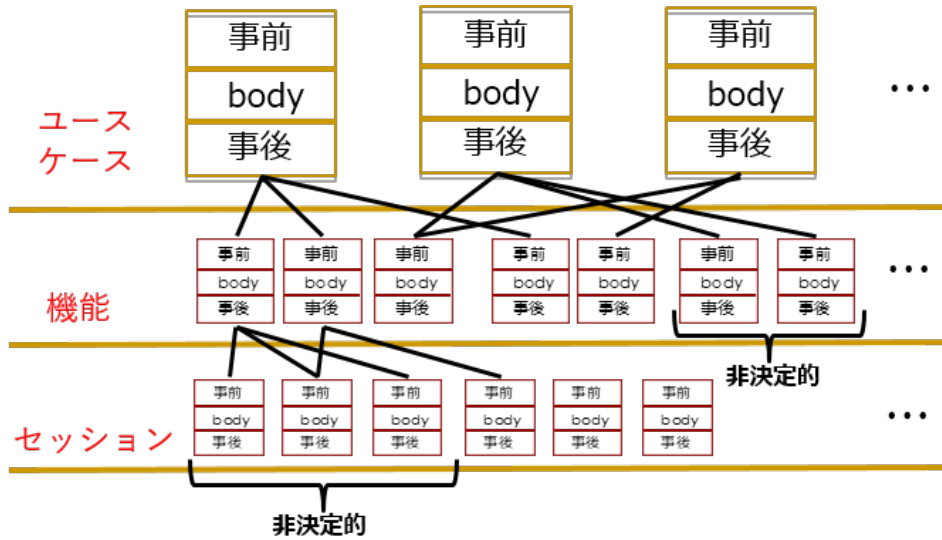


図 6.1 複数機器/ソフトウェアが協調するシステムにおける機能の階層構造

テスト手順・テスト構成を分割するこの考え方は、論理学における、シンタックス（形式や構文）とセマンティクス（意味）を区別する方針を参考にしている。すなわち、テスト手順に記述したユースケース-機能-セッションの構造や順序に対し、テスト構成に記述したユースケース-機能-セッションを実際に行う機器ないしソフトウェアの情報によって意味を与える、という考え方である。この分割の考え方は、ソフトウェア開発では従来、システム仕様の記述などで用いられている。例えば独立法人情報処理推進機構による機能要件の合意形成ガイド [74] では、「機能要件」を満たす情報システムの要素として「システム振る舞い」「データモデル」を分けて挙げている。この「システム振る舞い」が本研究におけるテスト手順、「データモデル」がテスト構成に、それぞれ対応していると考えられる。

6.2 実行可能テストケースの自動生成

本節では、前節の考え方に則り分割して記述したテスト手順・テスト構成から、実行可能なテストケースを自動生成する手法について説明する。

図 6.2 に、本手法の模式図を示す。本手法は、テンプレートエンジンの仕組みに基づいている。テンプレートエンジンとは一般的に、テンプレートと呼ばれる雛形にデータモデルを合成し成果ドキュメントを出力するソフトウェアまたはソフトウェアコンポーネント

である。すなわち、テスト手順をテンプレート、テスト構成をデータモデルと捉え、テンプレートエンジンたる実行可能テストケース生成ユニットが実行可能なテストケースを自動生成する。テスト手順はセッション単位で作成し、機能およびユースケースに対応するテスト手順は、このセッションに対応するテスト手順を複数組み合わせることによって生成する。

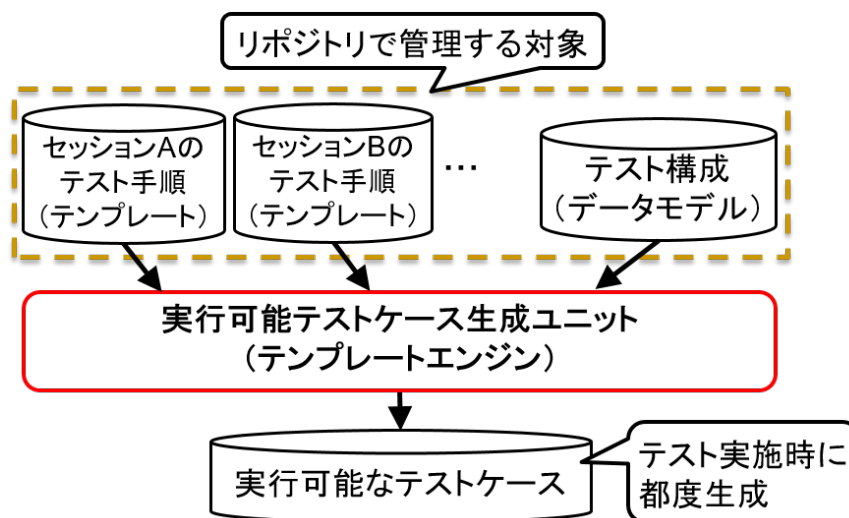


図 6.2 テスト手順・テスト構成からのテストケース自動生成

続いて、各要素の記述手法を具体的に示す。

実行可能なテストケースは、テスト実行ユニットのベースであるモデル検査ツール SPIN 専用の形式仕様言語 Promela (Process Meta Language) にて記述する。4.1 節でも述べた通り、Promela はその名前が示すように元来通信プロトコルの検証のために設計された実行可能な言語であり、またシステムの非決定的振る舞いを記述可能であることから、本論文が提案する統合テスト実行環境の対象である、ビル空調システムを初めとする複数の機器ないしソフトウェアが協調して機能を実現する組込み機器のテストケース記述に適していると判断した。

これにともないテスト手順は、Promela のテンプレートとして記述する。以降、テスト手順を記述した Promela のテンプレートを手順テンプレートと称す。図 6.3 に、手順テンプレートの内容を模式的に示す。手順テンプレートでは、Promela においてプロセスを表す「Proctype」を使用して、セッションを記述する。1 つの Proctype は、そのセッションにおいて 1 つの機器 (タイプ 1 の場合) ないし 1 つのソフトウェア (タイプ 2 の場合) が実行する処理を表す。各 Proctype の中には、その機器ないしソフトウェアの種別を表

すデータを含める。そして 4.1 節でも述べたように、テストの際に行う操作は Proctype メンバ変数への値代入として、機器間ないしソフトウェア間で行われる通信は Proctype 間通信として記述する。この通信の順序/タイミングが非決定的であるならば、通信処理を、Promela において非決定的処理を記述する構文「do-od」の中に記述する。評価項目は、C 言語コードにより LTL 式を記述する Promela の構文「c_expr」に、機器間通信のメッセージやソフトウェアの内部変数を判定する LTL 式を埋め込むことで記述する。そして各 LTL 式に、関係する機器ないしソフトウェアの種別（手順テンプレートにおける定義に合わせておく）を埋め込んでおく。

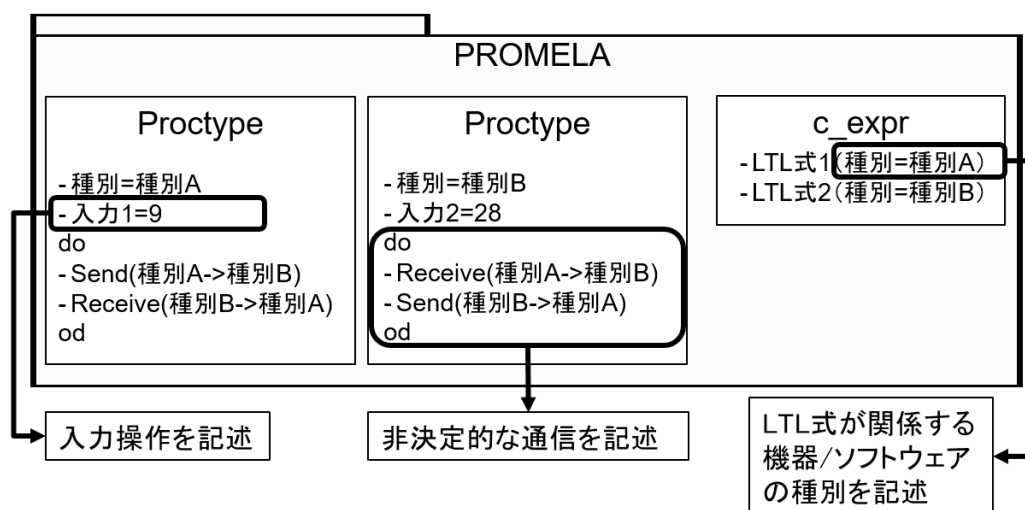


図 6.3 手順テンプレートの模式図

テスト構成は、記述されている情報を容易に抽出できるように、情報を形式的かつ階層的に記述可能なフォーマットで記述する。例えば、代表的なマークアップ言語である XML がこれに該当する。以降、このテスト構成を記述したデータを、構成データモデルと称す。図 6.4 に、構成データモデルの内容を模式的に示す。構成データモデルには、機器ないしソフトウェアの種別（手順テンプレートにおける定義に合わせておく）・システムにおけるアドレスを、機器の接続台数ないしソフトウェアの搭載モジュール数分記述しておく。図 6.4 の例は、システムにアドレスが 10 である種別 A の機器が 1 台、アドレスが 1, 2, 3 である種別 B の機器が 3 台接続されていることを示している。

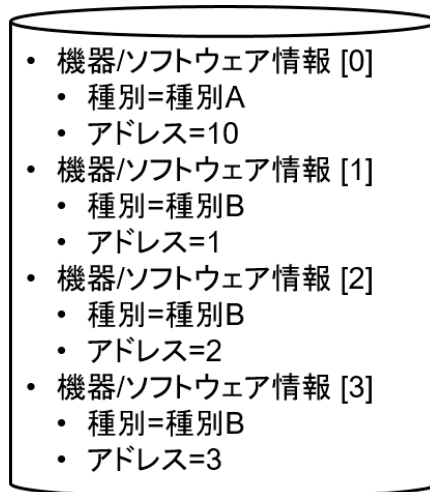


図 6.4 構成データモデルの模式図

実行可能テストケース生成ユニットは、以上に示した手順テンプレートに構成データモデルを組み合わせ、実行可能なテストケースを生成する。組み合わせの手順を下記の(V1)~(V6)に示す。

- (V1): (機能ないしユースケースに対応するテスト手順を作成するため、複数のテスト手順を組み合わせる場合) 各テスト手順中の Proctype について、同じ種別の機器ないしソフトウェアが含まれている Proctype を抽出し、内容をマージする
- (V2): 手順テンプレートに記述された Proctype 及び LTL 式が含む種別と同じ種別の機器ないしソフトウェアが、構成データモデルにいくつ記述されているかを抽出する。
- (V3): 抽出された回数だけその Proctype 及び LTL 式を複製し、複製した各々の Proctype 及び LTL 式に、構成データモデルに記述されていたアドレスを割り当てる。
- (V4): Proctype については、埋め込まれている全ての Proctype 間通信メッセージについて、通信宛先を示す種別と同じ種別の機器ないしソフトウェアが、構成データモデルにいくつ記述されているかを抽出する。
- (V5): 抽出された回数だけその通信メッセージを複製し、複製した各通信メッセージの宛先に、構成データモデルに記述されていたアドレスを割り当てる。
- (V6): (V2)~(V5) を、手順テンプレートに記述された全ての Proctype 及び全ての LTL 式に対して繰り返す。

上述の手順により生成した，図 6.3 の手順テンプレートで記述された機能を，図 6.4 の構成データモデルで記述されたシステム構成でテストするための実行可能なテストケースの模式図を図 6.5 に示す。

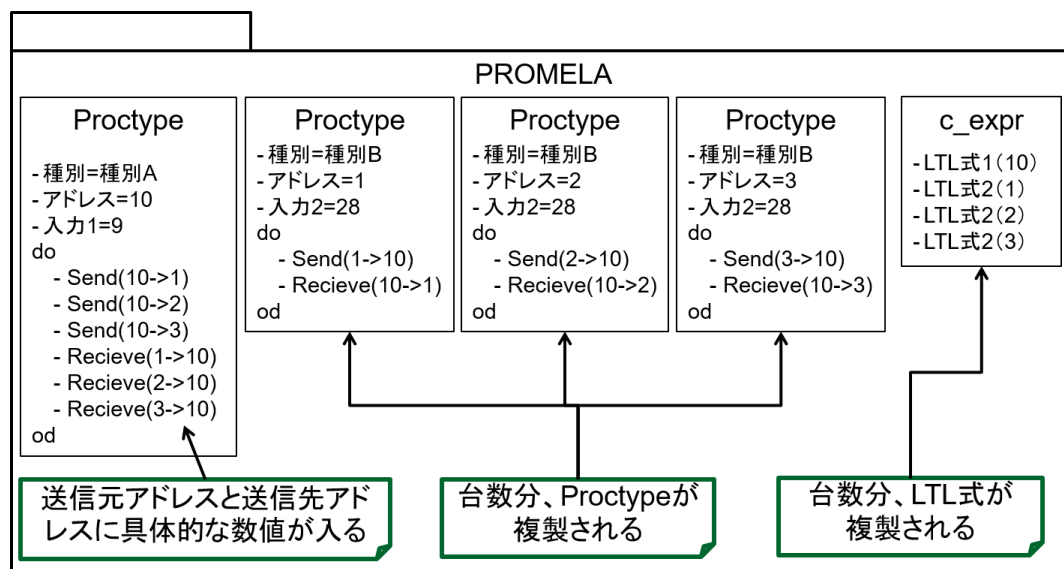


図 6.5 手順テンプレート・構成データモデルの組合せで生成される実行可能なテストケースの模式図

6.3 ビル空調システム製品向けテストケース管理リポジトリ・実行可能テストケース生成ユニット

本章でこれまでに述べた手法を評価するために，ビル空調システム製品向けのテストケース管理リポジトリ・実行可能テストケース生成ユニット（図 6.6）を構築した。実行可能テストケース生成ユニットにはテンプレートエンジン pongo2[75] を組み入れ，手順テンプレートは Promela に pongo2 の記述を加えた形式とする。構成データモデルは XML 形式とする。これだけではテスト技術者が Promela・pongo2・XML について習得する必要が発生し，3.1 節に挙げたテスト技術者が知識・経験を習得する必要を低減する本研究のアプローチにそぐわないため，手順テンプレート・構成データモデルの作成を支援するツールを用意した。

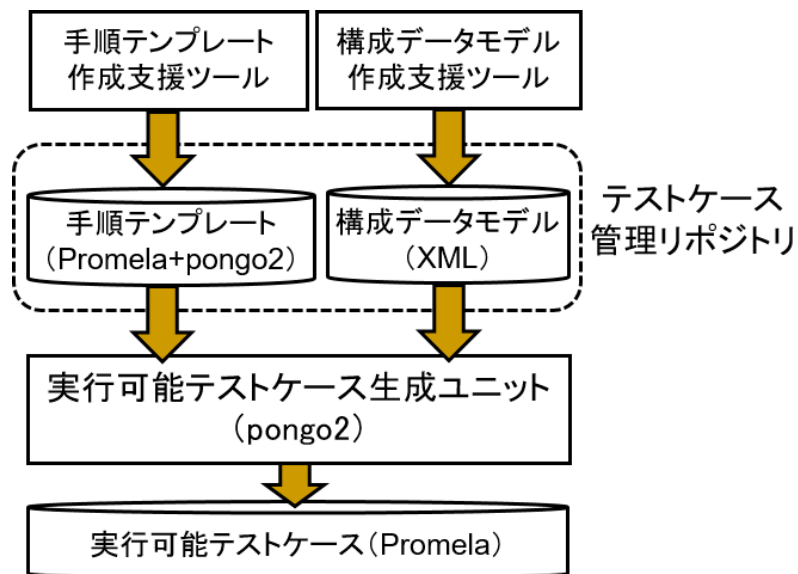


図 6.6 テストケース管理リポジトリ・実行可能テストケース生成ユニットの構成

図 6.7 に、ツールによる手順テンプレート・構成データモデル作成支援の概要を示す。Promela 形式の手順テンプレートについては、システムの構成要素である機器ないしソフトウェアの種別を一覧から選択させることで Proctype の雛型を、機器ないしソフトウェアの入力を一覧から選択させ操作のタイミングおよび値を設定することで Proctype のメンバ変数への代入する処理を自動生成することによって作成を支援する。構成データモデルについては、機器ないしソフトウェアの種別を示すブロックを一覧から選択して必要な数を配置し、アドレスを割り付けるとともに接続関係をブロック間の線で表現する、というユーザインターフェースを考える。また作成済の手順テンプレート及び構成データモデルを読み込み、作成時に設定した情報をグラフィカルに表示できるようにしておくことで、確認作業も支援する。

図 6.8 は、手順テンプレートの作成を支援するツールの外観である。GUI 上で各空調機器の入力を一覧から選択し、入力操作のタイミングおよび値を設定することで、手順テンプレートを出力する。

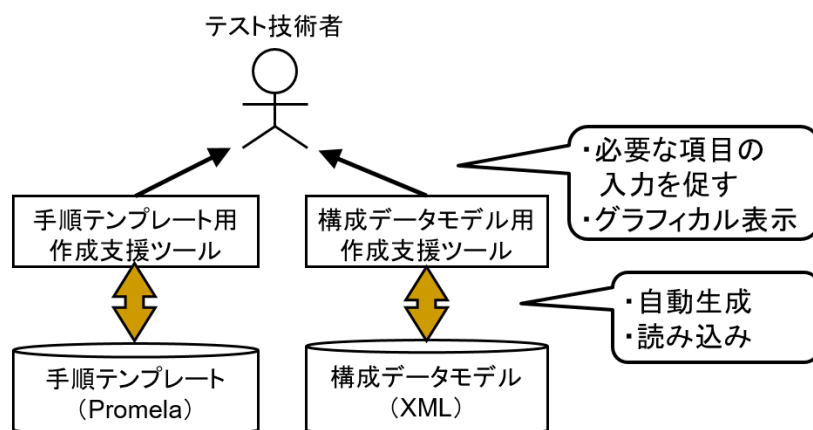


図 6.7 手順テンプレート・構成データモデルの作成支援

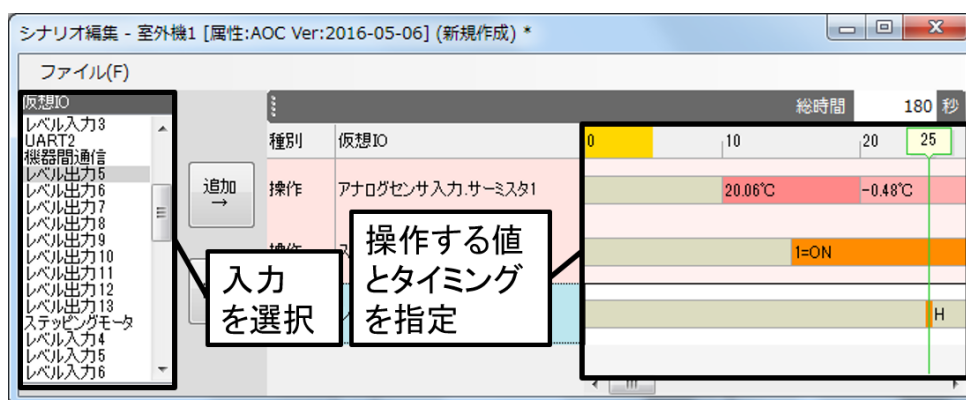


図 6.8 手順テンプレート用作成支援ツール

図 6.9 は、構成データモデルの作成を支援するツールの外観である。ビル空調システムの接続構成を可視化し、マウス操作主体で編集可能にすることで、構成データモデルの作成を容易にしている。

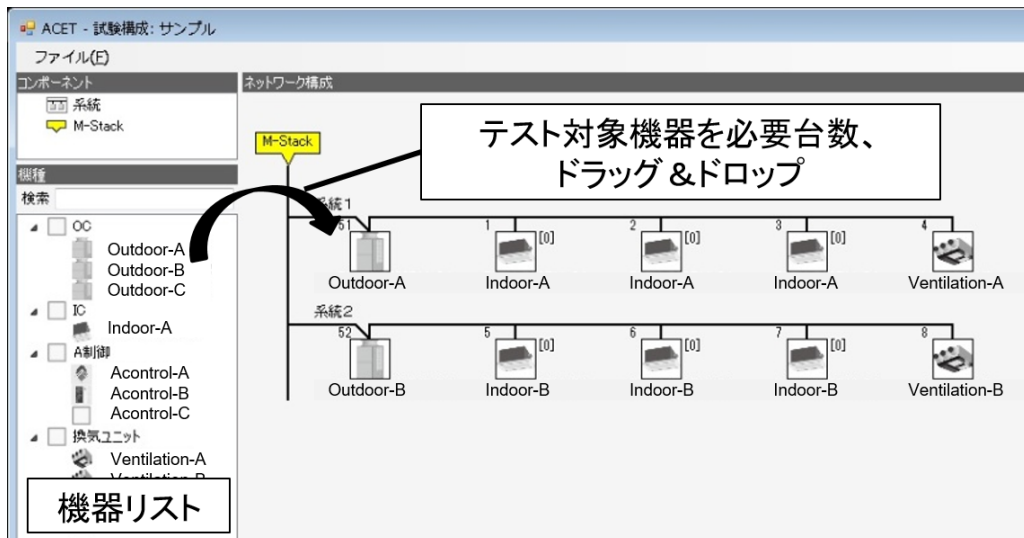


図 6.9 構成データモデル用作成支援ツール

このビル空調システム製品向けのテストケース管理リポジトリで管理される手順テンプレート・構成データモデルの例を、以降に挙げる。

室外機と室内機との協調による機能の 1 つについて記述した手順テンプレート（抜粋）を、ソースコード 6.1 に示す。テスト開始後室内機の「Mode」が「Cooling」に、「SettingTemperature」が「26」に設定され、その後室外機と室内機との間で通信がなされることで機能が実現される。記述している情報は 6.1 節に述べた通りで、室外機に対応する Proctype（8 行目）を宣言し、室内機に対応する Proctype（11 行目）を宣言し、室内機への設定を示す値の代入（5～6 行目）を記述している。評価項目としては、メッセージ `error_message` を受信した時に真、そうでない時に偽を返す `is_received` 関数による、「全ての室内機から、メッセージ `error_message` が決して受信されないこと」を示す LTL 式を記述している（1～6 行目）。また、構成データモデルの情報（機器種別毎の台数や各機器のアドレス）を組み合わせられるようにするため、`pongo2` のプレースホルダを埋め込んでいる。18 行目の `OC.address` が室外機の空調アドレスを、4 行目と 21 行目の `IC.address` が室内機の空調アドレスを組み合わせる部分である。そして 17～19 行目の「for-endfor」ブロックによって室外機プロセス（proctype OC）を室外機の台数分、20～22 行目の「for-endfor」ブロックによって室内機プロセス（proctype IC）を室内機の台数分、起動するようにしている。

ソースコード 6.1 手順テンプレートの記述例 (抜粋)

```
1 □
2 (!c_expr{
3   {% for IC in ICs %}
4   is_received({{IC.address}}, error_message)
5   {% endfor %}
6 })
7 :
8 proctype OC(int address){ /*Outdoor Unit*/
9   func1_OC(address)
10 }
11 proctype IC(int address){ /*Indoor Unit*/
12   status[Mode] = 4 /*Cooling*/
13   status[SettingTemperature] = 26
14   func1_IC(address)
15 }
16 Init{
17   {% for OC in OCs %}
18   run OC( {{OC.address}} )
19   {% endfor %}
20   {% for IC in ICs %}
21   run IC( {{IC.address}} )
22   {% endfor %}
23 }
24 :
```

続いて、空調アドレスが 51 である室外機 1 台、空調アドレスが 1 と 2 である室内機 2 台の構成について記述した構成データモデルを、ソースコード 6.2 に示す。記述している情報は 6.1 節に述べた通りで、TestInfo タグが 1 つの構成に、DeviceInfo タグが 1 つの機器ないしソフトウェアに対応しており、種別は ModelName 属性と Ver 属性によって記述し、空調アドレスは Address 属性によって記述する。ソースコード 6.2 の例は、空調アドレスが「51」である 1 台の室外機（バージョン 1.0）と、空調アドレスが「1」及び「2」である 2 台の室内機（バージョン 1.0）からなる構成を示す記述となっている。

ソースコード 6.2 構成データモデルの記述例

```

1 <TestInfo>
2   <DeviceInfo Modelname="OC" Ver="1.0" Address="51" />
3   <DeviceInfo Modelname="IC" Ver="1.0" Address="1" />
4   <DeviceInfo Modelname="IC" Ver="1.0" Address="2" />
5 </TestInfo>

```

以上の手順テンプレート・構成データモデルから、6.1節で(V1)~(V6)として示した手順を pongo2 を活用し実現した実行可能テストケース生成ユニットにより、Promela 形式の実行可能なテストケースを生成する。ソースコード 6.1 の手順テンプレート・ソースコード 6.2 の構成データモデルより自動生成した実行可能なテストケースの抜粋を、ソースコード 6.3 に示す。手順テンプレートに対し、3 行目、6 行目、18~20 行目に、構成データモデルの情報が挿入されている。

ソースコード 6.3 実行可能なテストケースの記述例 (抜粋)

```

1 □
2 (!c_expr{
3   is_received(1, error_message)
4 })□
5 (!c_expr{
6   is_received(2, error_message)
7 })
8 :
9 proctype OC(int address){ /*Outdoor Unit*/
10  func1_OC(address)
11 }
12 proctype IC(int address){ /*Indoor Unit*/
13  status[Mode] = 4 /*Cooling*/
14  status[SettingTemperature] = 26
15  func1_IC(address)
16 }
17 Init{
18  run OC( 51 )
19  run IC( 1 )
20  run IC( 2 )
21 }

```

6.4 実験

本節では、前節に述べたテストケース管理リポジトリ・実行可能テストケース生成ユニットを、ビル空調システム製品開発で実施される再発防止テストに適用した実験及びその結果を示す。

6.4.1 実験手順

テストケース管理リポジトリ・実行可能テストケース生成ユニットにより、ビル空調システムに接続される室外機製品 2 機種種の再発防止テストに対応するテストケースを、空調メーカーにおけるテスト技術者 2 名（当該室外機担当である被験者 X，被験者 Y）に作成してもらった。この再発防止テストは 1 機種目の室外機（以降，室外機 1 と称す）については 112 項目，2 機種目の室外機（以降，室外機 2 と称す）については 104 項目あり，それぞれが室外機に共通のテスト項目 72 項目を含んでいる。また，再発防止テストの内容は表 6.1 に示すような，再発防止テスト項目一覧にまとめられている。

表 6.1 ビル空調システム製品の再発防止テスト項目一覧（抜粋）

対象機種	後工程流出不具合	原因	再発防止テスト内容
管理コントローラ	再起動後，管理コントローラと室内機の状態が不一致となる	許可状態変化の送信漏れ	再起動後，許可状態変化を送信することを確認
室外機	計量値が誤った値となる	通信コマンドフォーマット誤り	通信コマンドフォーマットが誤っていないことを確認

以下，実験手順を示す。

1. 被験者に，テストケース管理リポジトリ・実行可能テストケース生成ユニット・手順テンプレート作成支援ツール・構成データモデル作成支援ツールの機能を説明する。
2. 上述の再発防止テスト項目一覧を元に，被験者 X に，室外機 1 の再発防止テスト 112 項目について，手順テンプレート作成支援ツール・構成データモデル作成支援ツールも用いて手順テンプレート・構成データモデル・評価テンプレートを作成してもらい，作成作業にかかる時間を計測する。被験者 X が，作成不可と判断した場

合、もしくは自身の成果物を流用できると判断した場合、そのテスト項目については作成作業をスキップし、スキップした理由をコメントに残してもらう。

3. 被験者 Y に室外機 2 の再発防止テスト 104 項目につき、同様に手順テンプレート・構成データモデル・評価テンプレートを作成してもらい、作成作業にかかる時間を計測する。被験者 Y が、作成不可と判断した場合、自身ないし被験者 X の成果物を流用できると判断した場合、そのテスト項目については作成作業をスキップし、スキップとした理由をコメントに残してもらう。
4. 作業完了後、被験者 X・被験者 Y に、本作業にかかる任意のコメントを残してもらう。
5. 作成された手順テンプレート・構成データモデルを実行可能テストケース生成ユニットに入力し、実行可能なテストケースを生成する。

6.4.2 実験結果

前述の手順により、以下に示す結果を得た。

2 人の被験者によって、合計 216 項目のテストのうち 190 項目（約 88%）に対応可能な、手順テンプレート・構成データモデルが作成された。表 6.2 に、各被験者が作成した手順テンプレートの数、構成データモデルの数、また作成に費やした作業時間を示す。

表 6.2 手順テンプレート・構成データモデル作成に関する実験結果

項目	被験者 X	被験者 Y	被験者コメント
手順テンプレート作成数	91	28	(被験者 Y) 室外機に共通のテスト項目については、被験者 X の成果物を流用できると考えた。
構成データモデル作成数	5	5	(被験者 Y) 作成作業はしたが、内容については被験者 X の成果物とほぼ同一であり、時間は殆どかからなかった。
作成作業時間(時間)	11.9	3.1	(被験者 X) テスト項目一覧の内容確認に、予想以上に時間がかかった。

6.5 考察

前節に示した実験結果に基づき、本章で提案したテストケース資産化支援手法が狙った効果を達成できるか、すなわち、SPLE 資産としてテストケースの再利用を可能にしテストケース設計にかかる作業を効率化できるか、また、管理データベース数を低減しテストケースの追加作成作業・管理作業を低減することができるかという視点で、以下考察を述べる。

(1) テストケースの再利用について

実験対象とした室外機の再発防止テストには室外機に共通のテスト項目（72 項目）が含まれており、そのテスト項目については、機種をまたいでテストケースを再利用できる期待があった。しかし実験の結果、被験者 X も被験者 Y も、この 72 項目のうち 13 項目（約 18%）はハードウェア動作を確認するテストであり、本手法によるテストケース作成は不可と判断していた。一方で残りの 59 項目（約 82%）については、被験者 Y は、59 項目全てについて被験者 X が作成した手順テンプレートを再利用する判断をした。また構成データモデルについても、内容の大半が再利用可能であった（構成データモデル作成支援ツール上で室外機を差し替えただけで作業が完了した）旨のコメントが得られた。以上より本提案手法は、ビル空調システムに接続される室外機の再発防止テストにおいて、機種に共通なテスト項目の約 82% につき、テストケースの再利用を可能にしたと判断する。

またこの再利用によって、被験者 Y のテストケース設計にかかる工数が 59 項目分、低減されたと考えられる。この効果は室外機に共通のテスト項目によるものであるため、3 機種目以降の室外機の再発防止テストについても、同程度の工数低減を達成できるものと考察する。

(2) 管理データベース数の低減について

本実験において、190 項目の再発防止テストに対応する実行可能なテストケースを、手順テンプレート・構成データモデル合わせ 129 個のデータより作成することができた。つまり管理データベース数を、約 32% 低減することができた。詳細には、被験者 X は室外機 1 の 112 項目のうち作成不可と判断した 13 項目を除いた 99 項目のテストケースに、96 個のデータ作成で対応した（約 3% の管理データベース数低減）。被験者 Y は室外機 2 の 104 項目から作成不可と判断した 13 項目を除いた 91 項目のテストケースに、33 個のデータ作成で対応した（約 64% の管理データベース数低減）。上述したテストケース再利用についての考察より、3 機種目以降の作成データ数も 2 機種目同等に抑えられる見込み

がある。仮に、3機種目以降も91項目のテストを33個のデータ作成で対応できるとすれば、管理データベース数低減の割合は図6.10のように推移する。当該室外機は年間5機種程度の開発があるため、1年で管理データベース数をおおよそ半減できる試算となる。

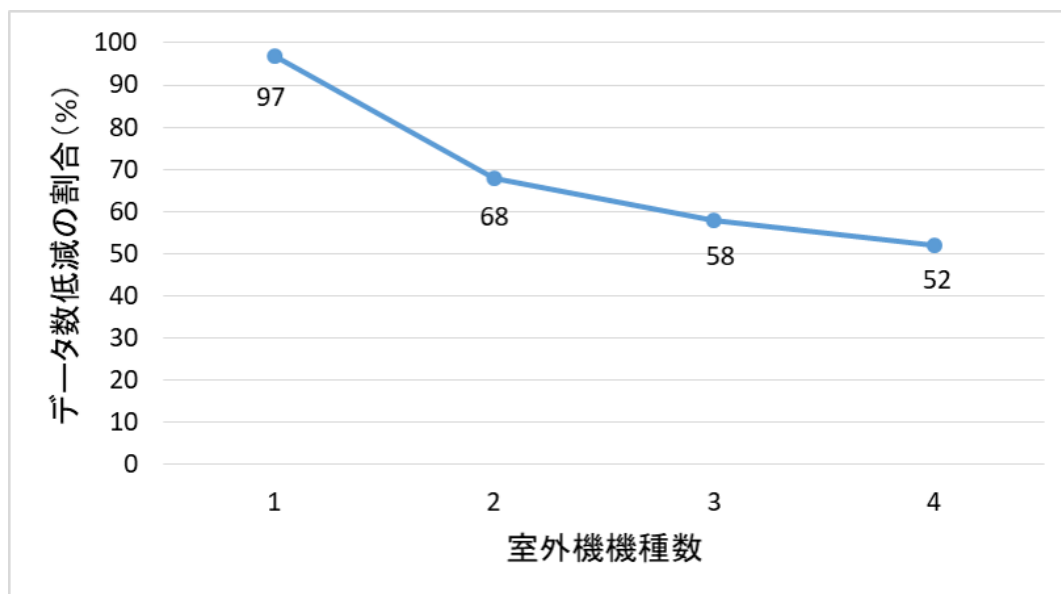


図 6.10 管理データベース数低減の割合の推移

一方で、より効果を大きくするための課題も抽出された。今回対象とした再発防止テストは、複数機能の衝突時の挙動を確認する、競合テストに該当するテストも含まれていた。しかし被験者はこれらのテストの手順テンプレートを機能毎に分割して作成するのではなく、1つの手順テンプレートとして作成していた。機能毎の手順テンプレートとなっていれば、より再利用率が高まり、管理データベース数も低減できていたと考えられる。

この、例えば競合テストのテストケース作成において部品化（機能毎の手順テンプレート作成）を促すためには、例えば図6.11に示すようなツールを提供する対応が考えられる。図6.11のツールは、作成済の手順テンプレートについて、登場する機器の種別及び機能/セッションを可視化するものである。このツールによってテスト技術者は、既に作成されている手順テンプレートが、自身が作成しようとしている手順テンプレートにおいて再利用可能かどうかを判断することが容易になると推測する。

(3) 他機種・他製品に対する効果について

以上ではビル空調システムの室外機における実験結果を考察したが、ビル空調システムに接続される他の空調機器、ないし他の組込み機器製品の回帰テストにおいても同様の効

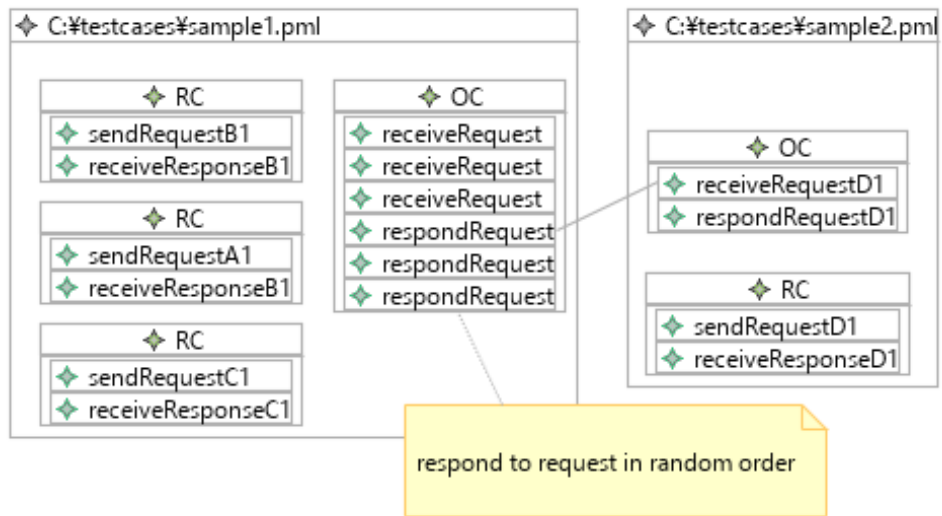


図 6.11 機能衝突を起こす手順テンプレートの作成を支援するツール（イメージ）

果を期待できる。他の空調機器については、室外機と同程度、同じ機器種別においては共通の機能が含まれていることが確認されたためである。そしてこれは SPLE を適用した機種展開開発を実施しているためと考えられるので、同じく SPLE を適用した他の組込み機器製品においても同程度の共通機能があり、その機能については、本提案手法にてテストケースの再利用・管理データベース数の低減を可能とすることが見込める。

第7章

製品適用によるテスト効率化の評価

本章では、これまでに述べた手法によって SPLE を適用したビル空調システム製品開発の回帰テストを効率化する統合テスト実行環境、及びその製品適用による効率化評価について述べる。7.1 節では、統合テスト実行環境の構成について述べる。7.2 節ではこの統合テスト実行環境によって、前章の実験で作成した実行可能なテストケースを実行した結果を示す。7.3 節ではこの結果に対する考察を通じ、提案方式によってビル空調システム製品の回帰テストが効率化されることを示す。

7.1 統合テスト実行環境の構成

統合テスト実行環境のソフトウェア構成を、図 7.1 に示す。統合テスト実行環境は第 4 章に示した空調機器エミュレータないし実機に対する実行テストをモデル検査の仕組みで実施可能なテスト実行ユニット、第 5 章に示した ACET、第 6 章に述べたテストケース管理リポジトリ・実行可能テストケース生成ユニットに、テストを実行するための GUI 画面であるテスト実行画面を追加している。

テスト実行画面の外観を、図 7.2 に示す。テスト技術者はこの画面を用い、手順テンプレートと構成データモデルを指定する。また各機器ないし各ソフトウェアについて、空調機器エミュレータで起動するか実機で起動するかを切り替えることができる。統合テスト実行環境はこの画面で指定された内容に応じて、実行可能なテストケースの生成、空調機器エミュレータの起動、空調機器実機の接続確認を行ったのち、テストを実行する。

加えて本環境では、組込みソフトウェアを空調機器エミュレータで実行する処理を構成データモデルに則して自動で行えるようにするため、構成データモデルのタグ・要素を追加した。具体的には、空調機器エミュレータと実機のどちらで起動するかを示すフラグを IsVirtual タグの Virtual 属性に、組込みソフトウェアのファイルパスを Program タグの

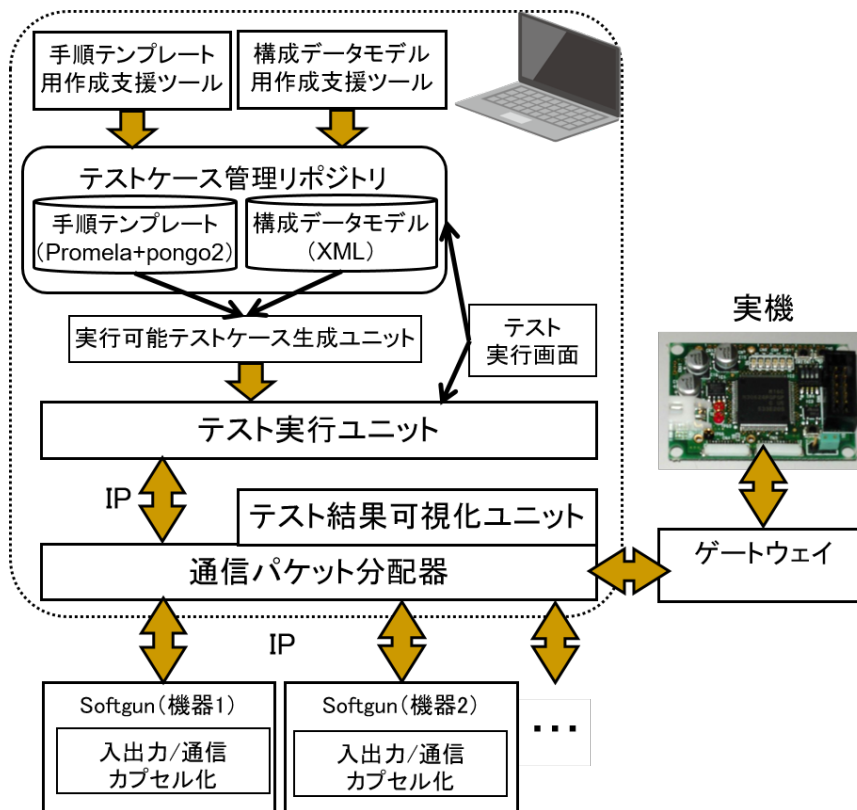


図 7.1 統合テスト実行環境のソフトウェア構成

Path 属性に記述することとした。ソースコード 7.1 に示した例では、アドレス 51 の室外機の組込みソフトウェア (A.mot) を空調機器エミュレータで、アドレス 1 の室内機を実機で動作させている。

ソースコード 7.1 統合テスト実行環境における構成データモデルの記述例

```

1 <TestInfo>
2   <DeviceInfo Modelname="Outdoor-A" Ver="1.0" Address="51">
3     <IsVirtual Virtual="False" />
4     <Program Path="C:\outdoor\mot\A.mot" />
5   </DeviceInfo>
6   <DeviceInfo Modelname="Indoor-B" Ver="1.0" Address="1">
7     <IsVirtual Virtual="True" />
8   </DeviceInfo>
9 </TestInfo>

```

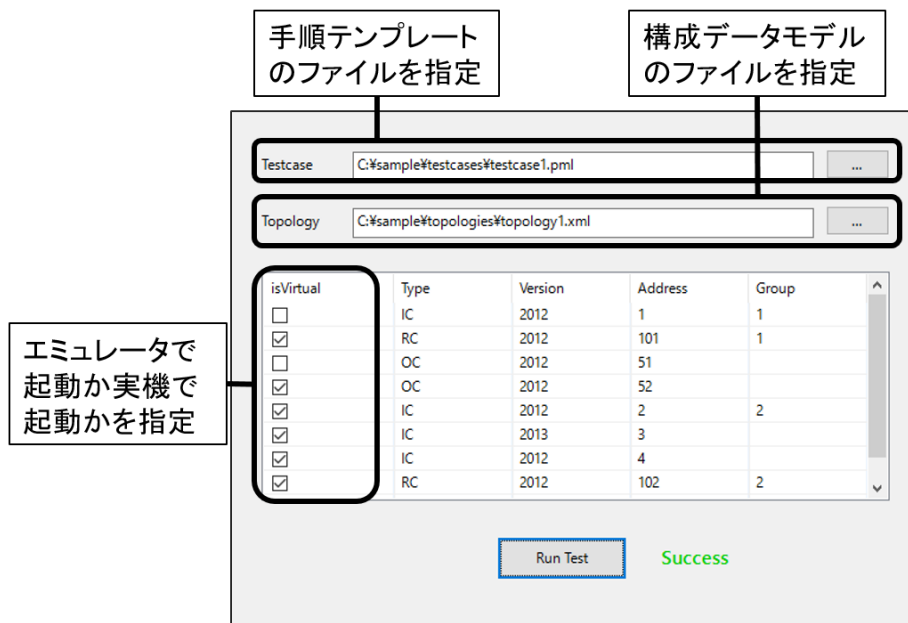


図 7.2 統合テスト実行環境のテスト実行画面

7.2 実験

前節で説明した統合テスト実行環境によって、前章の実験で実行可能なテストケースを作成したビル空調システムの再発防止テストに適用し、統合テスト実行環境がもたらすテスト効率化効果を評価する実験を行った。本節では、この実験の内容と結果とを述べる。

7.2.1 実験手順

統合テスト実行環境によるテスト効率化の効果を測定するため、空調メーカーにおけるテスト技術者 1 名を被験者として実験を行った。実験手順は以下の通りである。

1. 統合テスト実行環境の機能について、被験者に説明する。また、以降の作業に詰まった場合は補足説明を行うことを伝える。
2. 被験者は、ビル空調システムの室外機 2 機種種の再発防止テストのうち、前章の実験で実行可能なテストケースを作成した 190 項目（1 機種目 99 項目、2 機種目 91 項目）を、統合テスト実行環境を用いて図 5.6 の作業フローに則り実施する（ただし不具合の分析は除く）。この作業時間を計測する。

3. 被験者は、同じ 190 項目のテストを、従来テスト環境（図 2.2）を用いて図 5.6 の作業フローに則り実施する（ただし不具合の分析は除く）。この作業時間も計測する。

7.2.2 実験結果

190 項目のテストに費やした作業時間は、従来テスト環境では 13.2 時間であったところ統合テスト実行環境では 3.2 時間と、10 時間（約 76%）の短縮となった。作業時間の内訳を、表 7.1 にまとめる。

表 7.1 作業時間計測結果

テスト対象	作業項目	作業時間（時間）	
		従来テスト環境	統合テスト実行環境
室外機 1 の 再発防止テスト （99 項目）	テスト対象機器の準備	0.5	0.1
	テストの実施，結果確認，テスト対象機器の接続・繋ぎ替え	6.3	1.5
	小計	6.8	1.6
室外機 2 の 再発防止テスト （91 項目）	テスト対象機器の準備	0.4	0.1
	テストの実施，結果確認，テスト対象機器の接続・繋ぎ替え	6.0	1.5
	小計	6.4	1.6
合計		13.2	3.2

7.3 考察

前節に示した実験結果に基づき、本章で提案したモデル検査技術とエミュレーション技術を融合した統合テスト実行環境が狙った効果を達成できるか、すなわち組込み機器製品の回帰テスト実行を自動化し工数を低減できるかという視点で、以下考察を述べる。

本実験では統合テスト実行環境の適用によって、テスト工数を 10 時間、約 76% 短縮できた。被験者の作業詳細を確認した結果、やはり自動化の効果が大きいと判断した。従来テスト環境では必須であったテストの実施・結果確認・テスト対象機器の接続・繋ぎ替えにかかる作業が自動化され対応する作業時間が不要となったことは勿論、従来テスト環境

で少数見受けられた被験者の操作ミスによるテストのやり直しが、自動化によって解消されたことも、効率化に寄与していると考えられた。

この自動化に至る前作業として、前章の実験で行った手順テンプレート・構成データモデルの作成作業があるため、以降はこの作業時間をふまえて考察する。前章の表 6.2 及び表 7.1 に挙げた結果より、1 機種目のテストに対する手順テンプレート・構成データモデル作成作業時間と統合テスト実行環境によるテスト実行時間との和は $11.9 + 1.6 = 13.5$ (時間)、2 機種目は $3.1 + 1.6 = 4.7$ (時間) である。そして 3 機種目以降は、2 機種目と同等の作業で手順テンプレート・構成データモデルを作成可能となる見込みを 6.5 節で得たことから、2 機種目と同等の作業時間を見積もる。これをふまえて、従来テスト環境による作業時間の累積と統合テスト実行環境による作業時間の累積をプロットすると図 7.3 に示す通りとなり、おおよそ 5 機種目以降で、統合テスト実行環境によって効率化効果が得られると推測できる。6.5 節でも述べた通り、当該室外機はおおよそ年間 5 機種程度の開発があるため、1 年ほどで効率化効果が得られることになる。

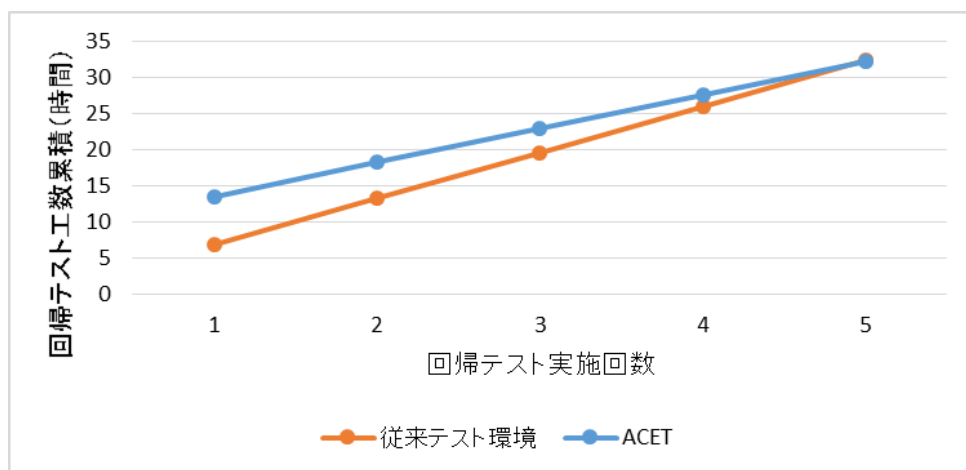


図 7.3 従来テスト環境と統合テスト実行環境との累積作業時間の比較

一方で、さらなる効率化が図れそうな場面も確認された。統合テスト実行環境では、テスト毎にテスト実行画面 (図 7.2) によって手順テンプレート及び構成データモデルを選択する手順としたが、被験者はこのファイル選択作業に少なくない時間を取られていた。この時間は、複数のテストケースを自動で切り替えつつ実行する機能によって低減できると考えられる。例えば、図 7.4 のように 1 つの手順テンプレートに対して組み合わせる複数の構成データモデルのリストを作成しておき、以降はそのリストを選択するだけで実行可能なテストケースの自動生成・実行を連続して行うようにすれば、テスト技術者がテス

ト環境前に貼り付く必要が低減され、さらなる効率化が見込める。

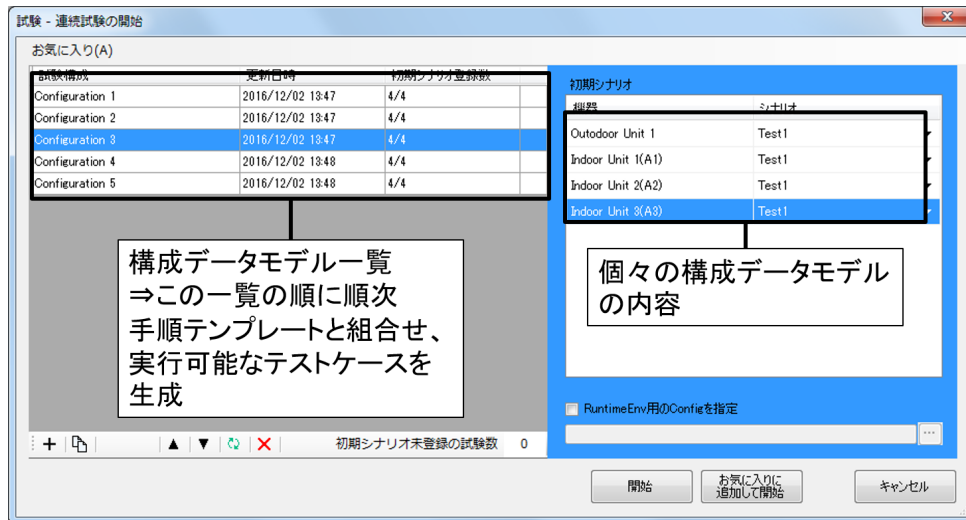


図 7.4 テストケースの自動切替機能（イメージ）

以上、ビル空調システム製品の室外機を事例としてモデル検査技術とエミュレーション技術を融合した統合テスト実行環境の効果を考察した。統合テスト実行環境はテスト対象機器の準備・接続，テストの実施・結果確認作業の自動化を実現し，作業時間短縮を実現する見込みを得た。テスト自動化による効率化は製品の機能によらないと考えられるため，ビル空調システムに接続される他の空調機器，また他の組込み機器製品においても同様の効果を期待できる。

第 8 章

結論

本章では結論として、本研究の成果を前章までの概要を交えつつ述べるとともに、本研究の将来の展望についてまとめる。

8.1 本研究の成果

機種に共通の機能をコア資産として開発し、機種固有の機能は派生開発にて実現する SPLE (Software Product Line Engineering) が、複数の機器ないし複数のソフトウェアが協調する組込み機器の開発に多く適用されている。一方で SPLE を適用した製品開発では、製品品質担保のために、新規機能とコア資産との整合性を確認する膨大な回帰テストを行うことが派生開発の度に求められ、開発効率が大きく阻害されていた。例えば、室外機・室内機・リモコン等が協調してオフィス等の空調を実現するビル空調システム製品開発でも SPLE を適用していたが、製品のグローバル展開にともなって国別機種展開開発・ブラックボックスな他社製品を取り込む開発が増加し、同期間で多くの SPLE プロセスを回さなければならなくなった結果、この 10 年ほどで回帰テスト工数が爆発的に増加した。

本研究では以上に述べた問題の解決を目的として、SPLE を適用した組込み機器製品開発の回帰テストを統一的な仕組みの下で支援する、統合テスト実行環境を構築した。また、上述のビル空調システム製品開発で実際に行われている回帰テストに適用し、効率化に対する評価を実施した。

複数の機器ないし複数のソフトウェアが協調する組込み機器製品では、多種多様な機器ないしソフトウェアを接続可能として様々な要求に対応することを目的に、オープンなネットワークないしプラットフォームの上にシステムが構築され、また通信順序や共有資源の獲得順序が非決定的となっている。このような組込み機器のテスト効率化を目的とし

た技術は従来より、上流工程向けの技術としてはシステムのモデルに対して非決定的振る舞いの網羅的評価を自動実施するモデル検査技術などが、下流工程向けの技術としては組込みソフトウェアを PC 上で動作させるエミュレーション技術などが確立されてきた。しかし SPLE においてはテストケース資産化（再利用を可能とすること）の観点から、上流工程での評価と下流工程での評価を統一的な仕組みの下で実施できることが望ましい。

そこで本研究では、モデル検査技術とエミュレーション技術との融合によって、上流工程向けテストケースと下流工程向けテストケースを共通化し SPLE 資産として再利用することを可能とする仕組みを検討した。すなわち、実機ないしエミュレータ上で動作する組込みソフトウェアを対象とする実行テストをモデル検査の仕組みで実現可能とするために、モデル検査と実行テストとのギャップを埋める、モデル検査ツールと実機ないしエミュレータとを接続する手法、テスト毎にモデル検査ツールから組込みソフトウェアの状態を初期化する手法を構築した。そしてこの手法をビル空調システム製品の再発防止テストに適用し、本手法によって、実行テストをモデル検査の仕組みで実現可能とできることを確認した。

またエミュレーション技術を活用し、実機を用いる従来テスト環境で工数がかかっていたテスト対象機器の管理作業及び準備作業、不具合分析作業を効率化する手法を提案した。具体的には、複数の組込みソフトウェアを動作させる複数のエミュレータプロセス同士で、またエミュレータと実機との間で通信を成立させ、エミュレータ上で動作する組込みソフトウェアと実機との協調を可能とする仕組みを構築した。加えて、従来テスト環境ではハードウェア上の制約のためにテストで用いることができなかった入出力の操作・確認を可能にする、テスト結果可視化ユニットを開発した。これらをビル空調システム向けの統合テスト実行環境の一部として実装し、ビル空調システムの SPLE における回帰テストについて得られる効果を実験的に考察した。

テスト項目数が爆発的に増加している状況においては、テストケースの作成作業・管理作業工数が大きくなり、資産化自体が困難となる問題についても、アプローチを行った。モデル検査の入力となる形式仕様言語で記述された実行可能なテストケースを、テスト対象である機能の階層構造・機能の実行順序及びその非決定性などを記述したテスト手順と、テスト対象とするシステム構成を記述したテスト構成とに分割して記述し、テスト実施時にこれらを任意に組み合わせてテストケースを自動生成する方式を提案した。そしてこの方式によってテストケースの資産化が支援されること、すなわちテストケースの再利用が促進されるとともに、管理データベース数が低減されることを、ビル空調システム室外機製品の再発防止テストへの適用を通じ確認した。

そしてこれらの手法を組み込んだ統合テスト実行環境を構築し、ビル空調システム製品の SPLE で実際に行われている回帰テストに適用する実験を通じて、効率化性能を評

価した。結果、室外機製品の再発防止テストにおいて、上述のテスト手順・テスト構成作成時間を鑑みても、5機種分の機種展開開発以降で効率化効果を得られることを確認できた。また室内機・リモコンといった他の機種の開発においても、同等の効果が得られる見込みを得た。

以上に述べた本研究の提案手法及び構築した統合テスト実行環境によって、SPLEを適用した組込み機器製品開発において問題となっていた、爆発的に増加している回帰テストの工数を大きく低減することを可能にしたと考える。

8.2 本研究の展望

SPLEを適用した組込み機器製品開発において爆発的に増加している回帰テスト工数のさらなる低減に向けた、今後の展望として、以下の3点が挙げられる。

(1) テスト結果可視化ユニットの拡張

本論文では、組込み機器製品のテストではハードウェア構成によって操作手段・確認手段が限られている課題を解決するテスト結果可視化ユニットを提案したが、システム及びマイコンの多機能化傾向を受け、使用可能な入出力は今後増えていくと予想される。第5章の実験で、ACETがある入出力に未対応であったことが理由で適用できなかったテストが確認されていることから、テスト結果可視化ユニットが対応する入出力の拡張が必要である。本研究で構築した代表的な入出力を可視化する手法を基に、入出力の種別や値域からGUI部品を自動生成するといった、より多様な入出力を簡易に操作・確認可能とする手法を検討し、統合テスト実行環境が適用可能なテストをより増やしていく。

(2) テスト手順部品化の支援

第6章で述べた手順テンプレート・構成データモデル作成に関する実験により、テストケース資産化をさらに支援するためには、手順テンプレートを機能毎/セッション毎に部品化することを促す支援が必要であることが分かった。開発頻度が上がり多機能化が進んでいる組込み機器製品の開発では、この部品化の効果はより高まっていくと予想されるため、部品化支援手法の早期確立が今後の課題となる。支援手法の案として、6.5節に、作成済の手順テンプレートについて登場する機器の種別及び機能/セッションを可視化するツールの提供を提案した。他にも仕様書の曖昧さを排し機能間の並列関係を可視化する先行研究[76]もあり、これらを本研究で提案したテスト実行環境に取り込むなどの方法を検討する必要があるだろう。

(3) 広がり続ける組み込みシステム製品への適用検討

本論文で提案した手法は，製品分野によらず，複数の機器ないしソフトウェアが連携する組み込み機器製品に広く適用可能と判断している．統合テスト実行環境をビル空調システム以外にも適用する，事業化に向けての活動を実施中である（図 8.1）．

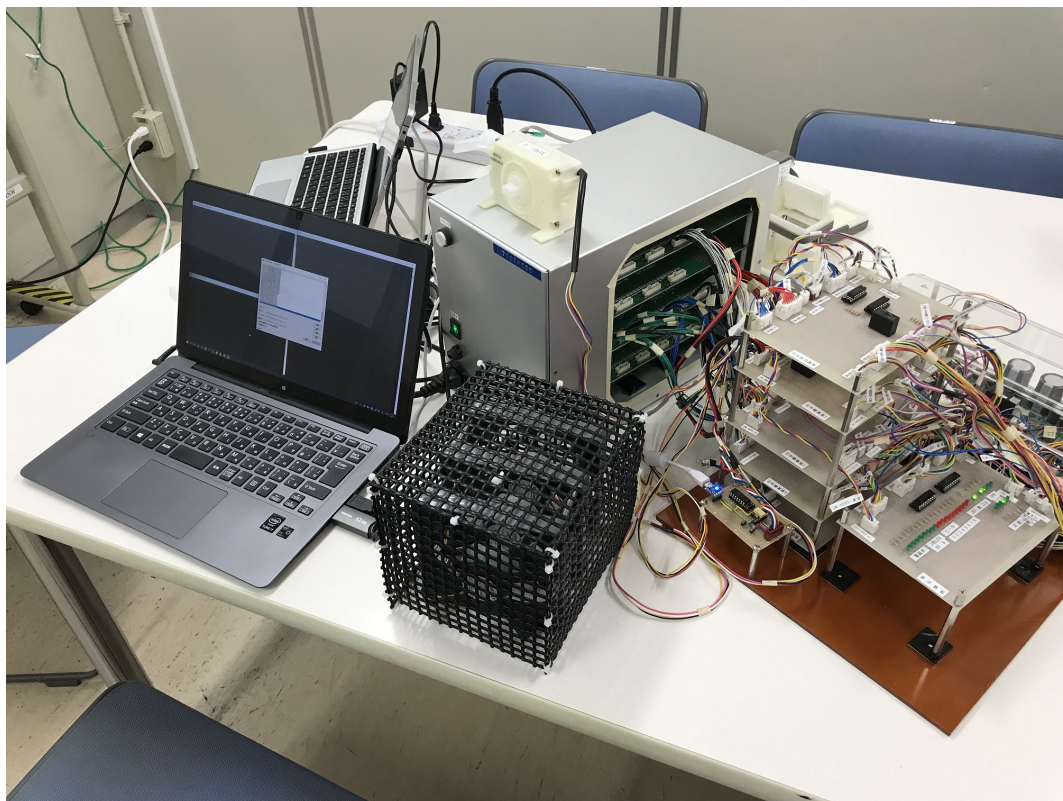


図 8.1 統合テスト実行環境の事業化活動風景

そして複数の機器ないしソフトウェアが連携する組み込み機器製品は今後，より広い分野の事業に展開されると考える．例として，ビルの空調・照明などを管理するビル設備システムについては Minoli ら（2017）[77] が，今後 IoT 技術の導入によってより広範囲の対象を管理していくと述べている．渡邊（2010）[78] も，従来のビル設備システムによる運営管理は点検，故障管理などが中心であったが，地球環境保護に対する意識の向上などから，近年ではビルの運用フェーズにおけるエネルギー管理に対応する必要性が生じていると主張している．異なる領域のシステムと組み合わせられるようにする要求も高まっている．組み合わせる手段として CPS や IoT，具体的にはインターネットやクラウドを活用する動きも始まっており [79]，組み込み機器同士がこれらを介して協調するシステ

ムが構築されつつある。

そしてこういったシステムでは、他社製品の取り込みを初めとする SPLE が、より高頻度で実施されるようになると予想される。本研究では組込み機器同士がローカルなネットワーク上で協調するシステムを対象として統合テスト実行環境を構築・評価したが、今後の研究では、前述したようなインターネットやクラウドでつながる製品についても、本研究で提案した手法の適用・評価が課題となると考えられる。

謝辞

本論文の執筆にあたり、九州工業大学大学院情報工学研究院の久代紀之教授に、深く感謝致します。久代教授には本研究についてのみならず、社会人ドクターとしての研究への取り組み方など、ご自身の経験もふまえ様々なご指導ご鞭撻を頂きました。本研究の内容につきましては特に、アカデミックな価値は何か？製品開発に対し貢献できることは何か？という、私の研究を進める中でいずれかに偏りがちになるこの2つの視点の両立について常に意識を促して頂き、また道を示して頂いたと感じております。改めて、心より御礼申し上げます。

またご多忙であらうところ、本研究の報告・査読・審査の日程調整にご対応を頂き、それらの場で貴重なご意見を賜りました九州工業大学大学院情報工学研究院の平田耕一教授、吉田隆一教授、光來健一教授に、深く御礼申し上げます。中々大学に足を運ぶことができない社会人ドクターという立場もあり、機会は多くなかったのですが、それだけに頂いたご教示は強く心に残っております。研究初期に本研究が前提とする範囲の明確化が重要であるということをご指導頂き、このポイントを早い段階から意識し、研究に臨むことができるようになりました。感謝申し上げます。

本研究を遂行するにあたり、共同研究先であり私のもう一つの所属先である三菱電機株式会社の皆様には、大変なご助力を頂きました。私の研究期間中に三菱電機照明株式会社にご異動となられました鈴木繁樹様には、私が社会人ドクターとしての研究を開始する際、また本研究の初期においてたいへん多くのご相談に乗って頂きました。深く感謝しております。IoT 事業推進センターの坂本忠昭様、冷熱システム製作所の石阪太一様、住環境研究開発センターの小泉吉秋様、関戸研司様には、本研究活動を進めるにあたり、私の三菱電機株式会社社員としての業務における調整など、ご尽力頂いていたと存じます。誠にありがとうございます。住環境研究開発センターの伊藤正俊君、中野裕梨さん、福田亜実さんには、共同研究業務の担当者として本研究のサポートをして頂きました。ありがとうございました。静岡製作所の服部真司様、長峯基様、東川傑様、後藤裕二様には、製品開発現場で実施されているテストに関する資料のご提供・テスト機材の貸与を頂くとともに、有益なご助言を頂き、本当にありがとうございました。この論文を新たな出発とし

て、三菱電機株式会社の事業に貢献できるように今後さらに精進し、皆様へのご恩返しとしたいと存じます。

そして久代研究室の皆様、ありがとうございました。中でも青山裕介君には、本研究の評価環境構築における実現手法の検討、そして私の研究成果と協調するテスト設計の課題に対しての研究活動など、非常に密に連携をして頂いたと感じております。他の皆様は実験の被験者としてや会社業務の紹介など、限られた時間での交流でしたが、様々な研究に触れることができ、また新鮮なご意見を頂くことができました。ありがとうございます。

また、本研究は JSPS 科研費 JP16K00100 の助成を受けたものです。ご支援、大変感謝致します。

最後に、私の生活を常日頃より支えてくれている家族に感謝の意を表し、謝辞と致します。

参考文献

- [1] Klaus Pohl, Günter Böckle, and Frank J van Der Linden. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.
- [2] 日本冷凍空調工業会. 業務用（パッケージ）エアコンの国内出荷台数の推移. <https://www.jraia.or.jp/statistic/s.com.aircon.html>.
- [3] 東京電力. 停電回数の国際比較. <http://210.250.6.22/corporateinfo/illustrated/electricity-supply/1253674.6280.html>.
- [4] Magnus Eriksson. An Introduction To Software Product Line Development. *Proceedings of Ume's Seventh Student Conference in Computing Science*, pp. 26–37, 2005.
- [5] Andreas Polzer, Stefan Kowalewski, and Goetz Botterweck. Applying software product line techniques in model-based embedded systems engineering. *Proceedings of the 2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software, MOMPES 2009*, pp. 2–10, 2009.
- [6] Li Jin-Hua, Li Qiong, and Li Jing. The W-model for testing software product lines. *Proceedings - International Symposium on Computer Science and Computational Technology, ISCCT 2008*, Vol. 1, pp. 690–693, 2008.
- [7] Yguaratã Cerqueira Cavalcanti, Ivan do Carmo Machado, Paulo Anselmo da Mota, Silveira Neto, Luanna Lopes Lobato, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. Towards metamodel support for variability and traceability in software product lines. *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems - VaMoS '11*, pp. 49–57, 2011.
- [8] Crescencio Rodrigues Lima Neto, Ivan Do Carmo Machado, Paulo Anselmo Da Mota Silveira Neto, Eduardo Santana De Almeida, and Silvio Romero De Lemos Meira. A Software Product Lines System Test Case Tool and Its Initial Evalu-

- ation. *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*, pp. 25–32, 2012.
- [9] E Engström and P Runeson. Software product line testing - A systematic mapping study. *Information and Software Technology*, Vol. 53, No. 1, pp. 2–13, 2011.
- [10] Jihyun Lee, Sungwon Kang, and Danhyung Lee. A survey on software product line testing. *Proceedings of the 16th International Software Product Line Conference on - SPLC '12 -volume 1*, p. 31, 2012.
- [11] Ivan Do Carmo Machado, John D. McGregor, Yguaratã Cerqueira Cavalcanti, and Eduardo Santana De Almeida. On strategies for testing software product lines: A systematic literature review. *Information and Software Technology*, Vol. 56, No. 10, pp. 1183–1199, 2014.
- [12] M. Mohamed Ali and R. Moawad. An approach for requirements based software product line testing. *2010 The 7th International Conference on Informatics and Systems (INFOS)*, pp. 1–10, 2010.
- [13] Paulo Anselmo Da Mota Silveira Neto, Ivan Do Carmo Machado, Yguarata Cerqueira Cavalcanti, Eduardo Santana De Almeida, Vinicius Cardoso Garcia, and Silvio Romero De Lemos Meira. A Regression Testing Approach for Software Product Lines Architectures. *2010 IV Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*., pp. 41–50, 2010.
- [14] Per Runeson and Emelie Engstrom. Software product line testing - A 3D regression testing problem. *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012*, pp. 742–746, 2012.
- [15] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, Patrick Heymans, and Yves Le Traon. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines. *IEEE Transactions on Software Engineering*, Vol. 40, No. 7, pp. 650–670, 2014.
- [16] László Vidács, Ferenc Horváth, József Mihalicza, Béla Vancsics, and Árpád Beszédes. Supporting software product line testing by optimizing code configuration coverage. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 1–7, 2015.
- [17] 守島浩. 大規模マニュアル開発の効率的な運営手法の検討. 情報処理学会研究報告, Vol. 2015, No. 2, pp. 1–7, 2015.
- [18] Cliff B Jones. *Systematic software development using VDM*, Vol. 2. Prentice Hall

- Englewood Cliffs, 1990.
- [19] J.-R. Abrial. *The B Book - Assigning Programs to Meanings*. Cambridge University Press, August 1996.
 - [20] J. M. Spivey and J. R. Abrial. The Z Notation. pp. VIII, 392 S. : Ill., 1991.
 - [21] Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. A Comprehensive Survey of Trends in Oracles for Software Testing. pp. 1–32.
 - [22] Mojtaba Shahin, Peng Liang, and Muhammad Ali Babar. A systematic review of software architecture visualization techniques. *The Journal of Systems & Software*, Vol. 94, pp. 161–185, 2014.
 - [23] Chien-Hung Liu, Shu-Ling Chen, and Tien-Chi Huang. A Model-Based Testing Tool for Embedded Software. *2012 Sixth International Conference on Genetic and Evolutionary Computing*, pp. 180–183, 2012.
 - [24] Dennie Reniers, Lucian Voinea, Ozan Ersoy, and Alexandru Telea. The Solid * Toolset for Software Visual Analytics of Program Structure and Metrics Comprehension : From Research Prototype to Product. *Science of Computer Programming*, Vol. 79, pp. 224–240, 2014.
 - [25] Rita Francese, Michele Risi, Giuseppe Scanniello, and Genoveffa Tortora. Proposing and Assessing a Software Visualization Approach Based on Polymetric Views. *Journal of Visual Languages & Computing*, Vol. 34, pp. 11–24, 2016.
 - [26] 野村佳秀, 木村功作, 福寄雅洋, 谷田英生. 『オープンソースソフトウェア工学』シリーズ 企業における oss 活用の実際. *コンピュータ ソフトウェア*, Vol. 33, No. 3, pp. 50–65, 2016.
 - [27] S. Xu, L. Chen, C. Wang, and O. Rud. A comparative study on black-box testing with open source applications. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 527–532, May 2016.
 - [28] Inbal Yahav, Ron S. Kenett, and Xiaoying Bai. Risk Based Testing of Open Source Software (OSS). *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, pp. 638–643, 2014.
 - [29] Em Clarke, Ea Emerson, and Ap Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 8, No. 2, pp. 244–263, 1986.
 - [30] John Joseph Chilenski and Steven P. Miller. Applicability of modified condi-

- tion/decision coverage to software testing. *Software Engineering Journal*, Vol. 9, No. 5, p. 193, 1994.
- [31] S. Rayadurgam and M.P.E. Heimdahl. Coverage based test-case generation using model checkers. *Proceedings. Eighth Annual IEEE International Conference and Workshop On the Engineering of Computer Based Systems-ECBS 2001*, pp. 83–91, 2001.
- [32] Moon Ho Hwang and Bernard P Zeigler. A modular verification framework based on finite & deterministic devs. *SIMULATION SERIES*, Vol. 38, No. 1, p. 57, 2006.
- [33] Hae Young Lee. Towards model checking of simulation models for embedded system development. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, pp. 452–453, 2013.
- [34] Gerard J Holzmann. The Model Checker SPIN. *Ieee Transactions on Software Engineering*, Vol. 23, No. 5, pp. 279–295, 1997.
- [35] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *International Conference on Computer Aided Verification*, pp. 359–364. Springer, 2002.
- [36] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A Tutorial on UPPAAL. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 3185, No. October, pp. 200–236, 2004.
- [37] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
- [38] O.R. Ribeiro, J.M. Fernandes, and L.F. Pinto. Model Checking Embedded Systems with PROMELA. *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, pp. 378–385, 2005.
- [39] 本田大雅, 小島英春, 中川博之, 土屋達弘. コンテキストに注目したゴールモデルのモデル検査に関する研究. ソフトウェアエンジニアリングシンポジウム 2018 論文集, pp. 229–235, aug 2018.
- [40] Shinpei Ogata, Yoshitaka Aoki, Hiroyuki Nakagawa, and Kazuki Kobayashi. IoT Prototyping and Evaluation of Support Method of Model Checking using Modeling Notation of IoT System Architecture. pp. 1–6, 2017.
- [41] Cassia De Souza Carvalho and Tatsuhiro Tsuchiya. Coverage criteria for state

- transition testing and model checker-based test case generation. *Proceedings - 2014 2nd International Symposium on Computing and Networking, CANDAR 2014*, pp. 596–598, 2015.
- [42] 高田沙都子, 森奈実子, 村田由香里. モデル検査自動化ツールの開発 ~ 検査自動化と反例解析効率化 ~. 情報処理学会第 74 回 全国大会, pp. 233–234, 2012.
- [43] Hideto Ogawa, Makoto Ichii, Fumihiro Kumeno, and Toshiaki Aoki. Experimental Fault Analysis Process Implemented Using Model Extraction and Model Checking. *2015 IEEE 39th Annual Computer Software and Applications Conference*, pp. 95–104, 2015.
- [44] Makoto Ichii, Tomoyuki Myojin, Yuichiroh Nakagawa, Masaki Chikahisa, and Hideto Ogawa. A rule-based automated approach for extracting models from source code. *Proceedings - Working Conference on Reverse Engineering, WCRE*, pp. 308–317, 2012.
- [45] 鷺見毅, 和田大輝, 晏リヨウ, 武山文信. モデル検査における不具合原因特定手法. 情報処理学会研究報告, Vol. 2015, No. 40, pp. 1–6, 2015.
- [46] Paolo Arcaini, Angelo Gargantini, and Elvinia Riccobene. NuSeen: A Tool Framework for the NuSMV Model Checker. *Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation, ICST 2017*, pp. 476–483, 2017.
- [47] Vladimír Štill, Petr Ročkal, and Jiří Barnat. Model checking of C and C++ with DIVINE 4. In *Automated Technology for Verification and Analysis (ATVA 2017)*, Vol. 10482 of *LNCS*, pp. 201–207. Springer, 2017.
- [48] Adrien Champion, Alain Mebsout, Christoph Stickel, and Cesare Tinelli. The KIND 2 Model Checker. *International Conference on Computer Aided Verification*, pp. 510–517, 2016.
- [49] Nicholas Halbwachs, Paul Caspi, Pascal Raymond, and Daniel Pilaud. The synchronous data flow programming language lustre. *Proceedings of the IEEE*, Vol. 79, No. 9, pp. 1305–1320, 1991.
- [50] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, Vol. 19, No. 1, pp. 9–30, 2017.
- [51] Victor Ferman, Dieter Hutter, and Raul Monroy. WebMC for Browser Based Protocol Verification. *Computación y Sistemas*, Vol. tbd., pp. 1–15, 2017.
- [52] Radu Calinescu, Kenneth Johnson, and Colin Paterson. FACT: A Probabilistic

- Model Checker for Formal Verification with Confidence Intervals. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 540–546. 2016.
- [53] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is Coming: A Modern Probabilistic Model Checker. 2017.
- [54] Ting Chen, Xiao Song Zhang, Xiao Li Ji, Cong Zhu, Yang Bai, and Yue Wu. Test generation for embedded executables via concolic execution in a real environment. *IEEE Transactions on Reliability*, Vol. 64, No. 1, pp. 284–296, 2015.
- [55] Jooyoung Seo, Ahyoung Sung, Byoungju Choi, and Sungbong Kang. Automating embedded software testing on an emulated target board. *Proceedings - International Conference on Software Engineering*, p. 9, 2007.
- [56] G. Heiser and B. Leslie. The okl4 microvisor: Convergence point of microkernels and hypervisors. *Proceedings of the first ACM asia-pacific workshop on Workshop on systems*, pp. 19–24, 2010.
- [57] Wind River Systems Inc. Wind river hypervisor. <http://www.windriver.com/products/hypervisor>, 2010.
- [58] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, pp. 41–46, 2005.
- [59] Vincent Autefage and Damien Magoni. NEmu: A distributed testbed for the virtualization of dynamic, fixed and mobile networks. *Computer Communications*, Vol. 80, pp. 33–44, 2016.
- [60] 原嶋秀次, 蔭山佳輝, 河込和宏. 仮想化技術による実機レステスト環境の構築. 東芝レビュー, Vol. 67, No. 8, pp. 31–34, 2012.
- [61] B Bardhi, A Claudi, L Spalazzi, G Taccari, and L Taccari. Virtualization on embedded boards as enabling technology for the cloud of things. In *Internet of Things*, pp. 103–124. Elsevier, 2016.
- [62] Shane Brady, Adriana Hava, Philip Perry, John Murphy, Damien Magoni, and A Omar Portillo-Dominguez. Towards an emulated IoT test environment for anomaly detection using NEMU. In *Global Internet of Things Summit (GIoTS), 2017*, pp. 1–6, 2017.
- [63] G S Jayashree and B S Rekha. Overview of Virtual Prototyping Techniques for System Development and Validation. *Imperial Journal of Interdisciplinary Research*, Vol. 3, No. 4, pp. 2143–2148, 2017.
- [64] Andrei Costin, Apostolis Zarras, and Aurélien Francillon. Automated Dynamic

- Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 437–448, 2016.
- [65] P. R. Oliveira, M. Meireles, C. Maia, L. M. Pinho, G. Gouveia, and J. Esteves. Emulation-in-the-loop for simulation and testing of real-time critical cps. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pp. 258–263, May 2018.
- [66] Ozgur Ozmen, James Nutaro, Jibonananda Sanyal, and Mohammed Olama. Simulation-based Testing of Control Software. Technical report, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States), 2017.
- [67] Jim Nutaro. Adevs (a discrete event system simulator). *Arizona Center for Integrative Modeling & Simulation (ACIMS), University of Arizona, Tucson*. Available at <http://www.ece.arizona.edu/nutaro/index.php>, 1999.
- [68] Y. Aoki, H. Ito, C. Ninagawa, and J. Morikawa. Smart grid real-time pricing optimization control with simulated annealing algorithm for office building air-conditioning facilities. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1308–1313, Feb 2018.
- [69] A. Chalh, S. Motahhir, A. El Hammoumi, A. El Ghzizal, and A. Derouich. A low-cost PV Emulator for testing MPPT algorithm. *IOP Conference Series: Earth and Environmental Science*, Vol. 161, No. 1, p. 012018, 2018.
- [70] Tingting Yu, Xiao Qu, and Myra B. Cohen. VDTTest: an automated framework to support testing for virtual devices. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pp. 583–594, 2016.
- [71] Anneke G Kleppe, Jos Warmer, Jos B Warmer, and Wim Bast. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.
- [72] 青山裕介, 黒岩丈瑠, 久代紀之. テストケース自動生成のための自然言語の形式変換アルゴリズム. 情報科学技術フォーラム 講演論文集, 第 18 巻, pp. 33–38, 2019. (選奨論文セッション).
- [73] Jochen Karrer. Softgun. <http://softgun.sourceforge.net/>.
- [74] 情報処理推進機構独立法人. 機能要件の合意形成ガイド ver1.0, 2010.
- [75] Florian Schlachter. pongo2. <https://github.com/flosch/pongo2>.
- [76] 青山裕介, 黒岩丈瑠, 久代紀之. モデル検査の実行順序制約の図式表現と試験ケースの自動生成. 電子情報通信学会論文誌 D, Vol. 101, No. 3, pp. 502–511, 2018.
- [77] Daniel Minoli, Kazem Sohraby, and Benedict Occhiogrosso. IoT Considerations,

Requirements, and Architectures for Smart Buildings-Energy Optimization and Next-Generation Building Management Systems. *IEEE Internet of Things Journal*, Vol. 4, No. 1, pp. 269–283, 2017.

- [78] 渡邊剛. ビルマネジメントシステム. 電気設備学会誌, Vol. 30, No. 3, pp. 40–44, 2010.
- [79] 岩野和生, 高島洋典. サイバーフィジカルシステムと iot (モノのインターネット) 実世界と情報を結びつける. 情報管理, Vol. 57, No. 11, pp. 826–834, 2015.

公表済み研究成果

1 審査のある原著論文：3 報

- 青山裕介, 黒岩丈瑠, 久代紀之, “テストケース生成のためのシステム仕様書の論理記述変換アルゴリズム,” 情報処理学会論文誌, Vol.61, No.3, 2020.
- 黒岩丈瑠, 青山裕介, 久代紀之, “エミュレーション技術の活用による IoT システムのテスト効率化,” 電気学会論文誌, Vol.140, No.1, 2020.
- 青山裕介, 黒岩丈瑠, 久代紀之, “モデル検査の実行順序制約の図式表現と試験ケースの自動生成,” 電子情報通信学会論文誌, Vol.J101-D, No.3, pp. 502-511, Mar. 2018. DOI:10.14923/transinfj.2017PDP0010

2 国際会議発表論文：6 報

- Yusuke Aoyama, Noriyuki Kushiro, Takeru Kuroiwa, “Test Case Generation Algorithms and Tools for Specifications in Natural Language,” International Conference on Consumer Electronics, Las Vegas, Jan 2020.
- Takeru Kuroiwa, Yusuke Aoyama, Noriyuki Kushiro, “Automatic Testing Environment for Virtual Network Embedded Systems,” International Conference on Consumer Electronics, Las Vegas, Jan 2020.
- Takeru Kuroiwa, Yusuke Aoyama, Noriyuki Kushiro, “A Hybrid Testing Environment between Execution Test and Model Checking for IoT Systems,” International Conference on Consumer Electronics, Las Vegas, pp.1-2, Jan 2019.
- Yusuke Aoyama, Takeru Kuroiwa, Noriyuki Kushiro, “Hybrid Testing Environment of Execution Testing and Model Checking for Product Line Approach,” 25th Asia-Pacific Software Engineering Conference (APSEC), pp. 693–694, Dec 2018.
- Takeru Kuroiwa, Yusuke Aoyama, Noriyuki Kushiro, “Testing Environment

for CPS by cooperating model checking with execution test,” *Procedia Computer Science*, Vol. 96, pp. 1341–1350, 2016. *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016*.

- Takeru Kuroiwa, Noriyuki Kushiro, “Testing Environment for Embedded Software Product Lines,” 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), pp.1-7, 2015.

3 査読のある国内会議論文：2 報

- 青山裕介, 黒岩丈瑠, 久代紀之, “CPS のためのモデル検査・実行テスト統合試験環境の構築 - モデル検査器を用いた試験ケースの資産化 -,” 第 15 回情報科学技術フォーラム講演論文集, 第 4 分冊, pp. 43–50, 2016.
- 青山裕介, 黒岩丈瑠, 久代紀之, “既存ソフトウェア部品を用いたソフトウェア開発におけるソースコード理解支援ツール,” 第 14 回情報科学技術フォーラム講演論文集, 第 4 分冊, pp. 111–118, 2015.

4 その他学会発表：7 報

- 青山裕介, 黒岩丈瑠, 久代紀之, “テストケース自動生成のための自然言語の形式変換アルゴリズム,” 第 18 回情報科学技術フォーラム講演論文集, 第 1 分冊, pp. 33–38, 2019. (選奨論文セッション).
- 青山裕介, 黒岩丈瑠, 久代紀之, “順序入れ替えテストケースの蓄積を実現するテスト実行環境,” 2018 年度 (第 32 回) 人工知能学会全国大会論文集, pp. 1-4, 2018.
- 十川雄司, 青山裕介, 黒岩丈瑠, 久代紀之, “LTL 式による動作ログからの不具合要因特定支援ツールの構築,” 情報処理学会/ソフトウェア工学研究会 ウィンターワークショップ 2018・イン・宮島 論文集, Vol. 2018, pp. 16–17, 2018.
- 青山裕介, 黒岩丈瑠, 久代紀之, “自然言語仕様からの機能間の並列・順序動作の抽出と左記テスト環境,” 第 16 回情報科学技術フォーラム講演論文集, pp. 35-40, 2017.
- 中野裕梨, 黒岩丈瑠, 青山裕介, 久代紀之, “設備機器の網羅的な通信試験におけるモデル検査の活用,” 第 14 回情報科学技術フォーラム講演論文集, 第 4 分冊, pp. 449-450, Aug. 2015.
- 青山裕介, 黒岩丈瑠, 久代紀之, “オープンソースソフトウェアを用いたシステム開発支援ツール,” 研究報告ソフトウェア工学 (SE), pp. 1-8, Nov. 2014.

- 黒岩丈瑠, 久代紀之, “設備機器システムのテストにおける CPU エミュレータの活用,” 第 13 回情報科学技術フォーラム講演論文集, 第 4 分冊, pp. 373-376, 2014.