情報処理学会論文誌 Vol.55 No.2 922–938 (Feb. 2014)

フィーチャ分析と充足可能性判定を用いたシステムテストに向けたシステム構成導出

新原 敦介[1,a)] 小川 秀人[1] 鷲崎 弘宜[2]

受付日 2013年5月17日, 採録日 2013年11月1日

概要：複数機器を組み合わせて機能を提供するシステムにおいて，各機器の仕様は定義されているが，機器の組合せによって表出するシステム仕様や，システム仕様と各機器の仕様との関係は暗黙知となっていることがある．またそれらのシステムテストでは，テスト項目ごとに適切な機器を組み合わせてシステム構成を導出しテストを実施する必要がある．ここで，機器の組合せ作業は，前記の暗黙知を必要とし，多くの時間がかかり，テスト項目に対して漏れのリスクが生じる．本報告では，各機器だけではなく，機器が組み合わさり生じる仕様に対してもフィーチャ分析を行い，その結果を用いてテスト項目を分析し，充足可能性判定を利用してテスト項目を網羅するシステム構成群を導出する方法を提案する．また，ケーススタディとして，提案手法を仮想の多店舗向け POS レジシステムに適用した．ケーススタディにおいて，従来のシステム構成導出に比べ提案手法では，テスト項目を網羅する妥当なシステム構成の導出が少ない時間で可能であった．

キーワード：ソフトウェア工学，テスト工程，フィーチャ分析

Selection Method for System Configuration for System Testing Using Feature Analysis and SAT Solver

Daisuke Shimbara[1,a)] Hideto Ogawa[1] Hironori Washizaki[2]

Received: May 17, 2013, Accepted: November 1, 2013

Abstract: In a system composed of multiple products, a specification of each product is defined well. Otherwise a specification of the system and relational definitions between the specification of products and the specification of the system become implicit knowledge in mature developers frequently. When a new product is developed and the product is tested, the implicit knowledge is needed for selecting peripheral products for composing system. This paper proposes models of the specification of the system and the relation definition and proposes method of the selection using feature analysis and SAT solver. The proposed method is applied to virtual POS register systems as a case study. The result of the case study indicates the proposed method can select suitable peripherals in shorter time than selecting by hand.

Keywords: software engineering, software testing, feature analysis

1. はじめに

単独の機器で機能を提供するのではなく，複数の機器が接続され連携して機能を提供する複合システムが多くなっている．このような複合システムを構成する機器のシステムテストは，テストの内容ごとに他の既存機器を適切に選択し，テスト対象の機器と接続してテストを実施する必要がある．テスト内容に対して不適切な機器の選択がされた場合，目的の機能が複合システム上に実現されず，テストできない可能性がある．本論文では，複合システムを構成する機器の選択結果をシステム構成とする．

[1] 株式会社日立製作所横浜研究所
Hitachi, Ltd., Yokohama Research Laboratory, Yokohama, Kanagawa 244–0817, Japan
[2] 早稲田大学
Waseda University, Shinjuku, Tokyo 169–8555, Japan
[a)] daisuke.shimbara.gk@hitachi.com

さらに，1つのシステム構成で多くのテスト項目をテスト可能なように機器を選択できれば，テスト実施時にシステム構成の組み換え工数を削減できる．そのため，少ないシステム構成数でテスト項目を網羅するという要求がある．

適切に機器を選択してシステム構成を導出する作業には，各機器の仕様だけでなく，機器が組み合わさった際に生じる様々な仕様や制約の知識が必要となる．しかし，実際の開発現場では，機器が組み合わさった際に生じる仕様や制約は，部分的には書類化されていても，その多くは暗黙知となってしまっている．そのため，システム構成を導出する作業は，暗黙知を持つ開発者に属人性が高く，効率的なシステム構成群を導出することが難しく，導出したシステム構成群がすべてのテスト項目を網羅していない可能性が生じている．

本研究では，暗黙知を形式知化して属人性を減らすために，機器に対するフィーチャ分析だけではなく，機器が組み合わさり複合システムとなった際に表出するフィーチャと，複合システムに表出したフィーチャと機器のフィーチャの関係性の分析を行い，その結果を用いてテスト項目を分析し，充足可能性判定を利用してテスト項目を網羅するシステム構成群を導出する方法を提案する．

本論文では以下を研究課題とする．

RQ1 テスト項目からシステム構成を導出する際，暗黙知などの属人性を低減可能か？

RQ2 テスト項目を網羅するシステム構成を導出可能か？

RQ3 テスト項目を網羅するために導出するシステム構成の数が現実的に少ない数か？

RQ4 テスト項目からシステム構成を導出する際，現実的な時間で実施可能か？

本論文の貢献は以下となる．

- 研究課題を満たす方法として，フィーチャ分析を用い，その結果と充足性判定を利用してテスト項目を網羅するシステム構成群を導出する方法を提案した．
- 提案する手法を実現するツールの実装を行った．
- 実装したツールを用いて，動機付けの例に対してケーススタディを行った．

本論文では，ソフトウェアプロダクトライン [1] におけるコア資産開発を対象とせず，機器の製品開発中のシステムテストにおいて，ソフトウェアプロダクトラインの技術であるフィーチャ分析を用いた手法を提案している．

2. 背景

2.1 複合システム

複数の機器が組み合わさって初めて機能を提供する複合システムには，業務用空調機や POS レジシステムなどがある．業務用空調機では，屋上にある室外機と屋内の室内エアコンがあり，制御情報のための通信線と冷媒管で接続されて空調機能を提供している．また，POS レジも，店舗内に設置したサーバと商品情報を共有し，売上情報を管理する複合システムである．

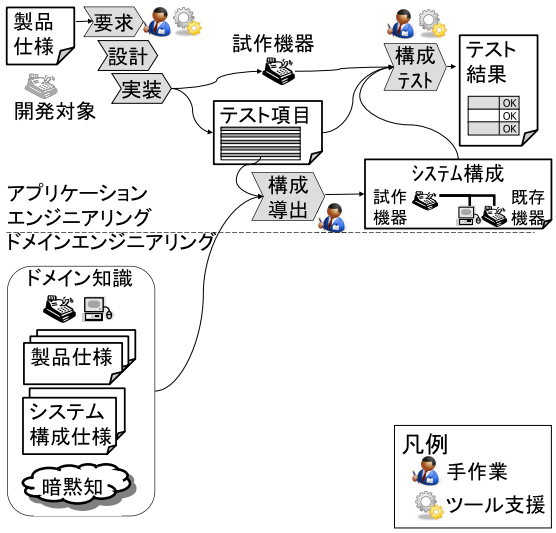


図 1 複合システムにおける既存の構成テスト

Fig. 1 Existing testing process for a system composed of multiple products.

これらの複合システムを構成する機器は，プロダクトライン開発が行われ，細かいオプションの有無の差など様々なバリエーションが生じる．これらの細かい差異のある機器は，相互に接続できなければならず，複合システムを構成する機器の理論上の組合せ数は膨大となる．

このような機器の開発では，システムテストにおいて，他の機器と接続して複合システムを構成しテストを行う必要がある．これを本論文では構成テストと呼ぶ．典型的な構成テストのプロセスを図 1 に示す．

開発対象機器が要求・設計・実装プロセスを経て，テストできる状態になると，開発者は構成テストを行う．要求・設計・実装プロセスでは，製品仕様に対応して構成テストで確認する項目であるテスト項目も作成する．構成テストでは，テスト項目ごとに，多数存在する既存機器の中から適切な機器を選択し，システム構成を導出して構成テストを実施する．間違った機器を組み合わせて構成テストを実施しようとしても，テスト項目が目的とする機能を実現できないことがあるため，適切な構成導出は重要となる．

複合システムを構成する機器を選択するには，開発対象機器の仕様情報だけではなく，他の機器の仕様や機器の組合せに関する仕様を熟知している必要があるため，ドメインエンジニアリングが重要となる．しかし，開発対象機器の仕様は充実したドキュメントがあっても，既存機器のすべてを網羅した機器の組合せに関しては，熟練した開発者の暗黙知となっていることが多く，十分なドキュメントが存在しないことがある．

さらに，構成テストにおいて，テスト項目を実際に実施していく際，システム構成の組み換えが発生する．この作

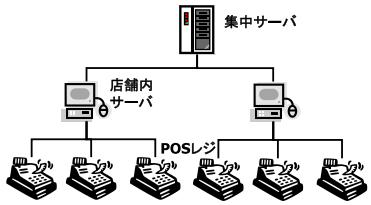


図 2 多店舗展開向け POS レジシステム
Fig. 2 POS register system for chain stores.

表 1 機器とそれらの仕様
Table 1 Products and their specification.

機器カテゴリ	機器名	対応機能
集中サーバ	CENTER_No.1	日本円
	CENTER_US	アメリカドル
店舗サーバ	SHOP_PERFECT	新プロトコル, 機能A, 日本円
	SHOP_PRO	新プロトコル, 日本円
	SHOP_STANDARD	機能A, 日本円
	SHOP_CLASSIC	日本円
	SHOP_PERFECT_US	新プロトコル, 機能A, アメリカドル
	SHOP_PRO_US	新プロトコル, アメリカドル
	SHOP_STANDARD_US	機能A, アメリカドル
	SHOP_CLASSIC_US	アメリカドル
POSレジ	REGISTER_PRO	新プロトコル, 機能B, 日本円
	REGISTER_STANDARD	新プロトコル, 機能C, 日本円
	REGISTER_CLASSIC	機能C, 日本円
	REGISTER_PRO_US	新プロトコル, 機能B, アメリカドル
	REGISTER_STANDARD_US	新プロトコル, 機能C アメリカドル
	REGISTER_CLASSIC_US	機能C, アメリカドル

業は，機器の運搬や結線作業に時間がかかり，構成テストの工数を圧迫してしまう．そのため，少ないシステム構成で，より多くのテスト項目を効率的に実施したいという要求がある．しかし先にも述べたように，既存機器のすべてを網羅した機器の組合せに関しては暗黙知が多く，効率的なシステム構成の導出が困難となっている．

2.2 動機付けの例：多店舗向け POS レジシステム

本論文では複合システムの構成テスト支援の動機付けの例として，多店舗展開を行うスーパマーケット向けの，現実の複合システムを参考にした仮想の POS レジシステムを用いる．本例は筆者が作成している．

POS レジは，各店舗に複数台設置され，店舗ごとに店舗内のレジを統括する店舗内サーバが設置される．また，店舗をまたいで情報を統括するための集中サーバが設置される（**図 2**）．ここで，集中サーバ，店舗サーバ，POS レジなどの複合システムを構成する機器の種類を本論文では機器カテゴリと称する．機器カテゴリごとの機器とそれらの仕様を**表 1** に示す．多店舗向け POS レジシステムは，これらの機器を組み合わせて複合システムとして機能を提供する．

動機付けの例で用いる多店舗向け POS レジシステムの機器の組合せによって生じるシステム仕様として，以下が定義されている．

通信プロトコルの混在：POS レジと店舗内サーバの間でやりとりする情報が増加し，新プロトコルを導入した経緯がある．この新プロトコルは，既存の古いプロトコルへの後方互換性を持つ．特に，店舗内サーバが新プロトコル対応の機器であれば，接続される POS レジが新プロトコルの機器と旧プロトコルの機器が混在していても，動作しなければならない．

通貨の排他性：この POS レジは，日本円のみを取り扱っていたが，US ドルを取り扱う機器を開発した．日本円対応の機器と US ドル対応の機器の混在は不可能となる．

機能 A の機能 B への依存：店舗内サーバが機能 A を持つ場合，その店舗内サーバに接続する POS レジは，機能 B を持たなければならないという制約がある．

機能 A どうしの排他：店舗内サーバが機能 A を持つ場合，この機能 A は，独立してすべての機器を統括する機能を有するため，その上位の集中サーバに接続する他の店舗内サーバは機能 A を持ってはならないという制約がある．

POS レジシステムにおける構成テストについて生じる特徴的なテスト項目の例として次の 5 つをあげる．

TC1 日本円を扱うシステムにおいて，店舗内サーバと POS レジの接続が新プロトコルで通信を行う状況で，店舗内サーバの機能 A が正常に動作することを確認する．

TC2 日本円を扱うシステムにおいて，店舗内サーバと POS レジの接続が新プロトコルと旧プロトコルの混在で通信を行う状況で，POS レジの機能 C が正常に動作することを確認する．

TC3 日本円を扱うシステムにおいて，1 つの店舗内サーバとそれに接続する POS レジの中で，同時に機能 A と機能 B が正常に動作することを確認する．

TC4 日本円を扱うシステムにおいて，1 つの店舗内サーバとそれに接続する POS レジの中で機能 B が動作し，別の店舗内サーバとそれに接続する POS レジの中で機能 C が動作することを確認する．なお，同じ店舗内サーバに接続する POS レジでは機能 B と機能 C は排他の関係にあり，動作してはならない．

TC5 日本円を扱うシステムにおいて，店舗内サーバ “SHOP_PERFECT” とそれに接続する POS レジの中で，同時に機能 A と機能 B が正常に動作することを確認する．

TC2 では，通信プロトコルの混在に関する仕様をテストし，TC3 では，機能 A から機能 B への依存の仕様に対応する．TC4 では，店舗内サーバと POS レジの組合せを複数必要としている．TC5 は，特定の機器を指定し，その機器を含むシステム構成におけるテスト項目となっている．

2.3 関連研究

フィーチャモデルを用いたテスト設計手法として，ユー

スケースモデルとフィーチャモデルを組み合わせてディシジョンテーブルを作成しテスト仕様とする手法が提案されている [2]．また，フィーチャモデルとペアワイズ法を組み合わせてテストすべきフィーチャセットを自動で導出する手法も提案されている [3], [4]．フィーチャツリーから充足可能性判定を用いてテストケースを設計する手法 [5] が提案されており，これは本論文で提案するアプローチと類似している．しかしこれらの手法は，ソフトウェアプロダクトラインにより開発されている単独の機器に対するテスト項目と機器の選択を支援しているが（**図 3** の A），本論文で扱う複数種類の機器が組み合わさる複合システムにおけるシステム構成の選択（図 3 の B）については論じていない．筆者は複数の機器を組み合わせた複合システムのテストを行うための組合せを導出する手法に関する研究を見つけられていない．

テスト項目を網羅させるために All-Pair 法を用いるアプローチが提案されている [6]．また，テスト項目数の最小化を整数計画問題として扱い，様々な最小化基準で行うためのフレームワークが提案されている [7]．これらは，直接的には複合システムのテストを目的としていないが，フィーチャモデルなどを因子や基準として与えることで支援できる可能性がある．本論文で提案する方法では，フィーチャモデルと複数種類の機器を組み合わせた複雑なシステム構成の導出に特化する．

PC 上で動作するソフトウェアを対象とするシステムテスト技法として，ソフトウェアが利用するリソースに着目してシステム構成を設計する方法が提案されている [8]．この手法は，共有リソースの衝突を検知することを目的としており，本論文で扱う目的とは合わない．

筆者らは，業務用空調機器を対象にフィーチャ分析を用いたシステム構成の導出方法を提案済みである [9], [10]．

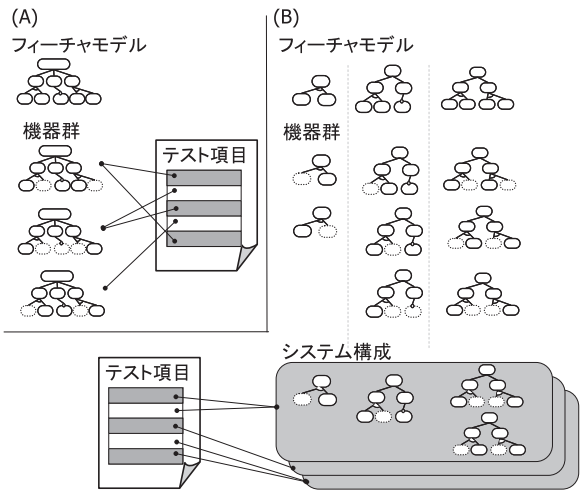


図 3 複合システムにおけるテスト項目とシステム構成

Fig. 3 Test cases and system configurations in for a system composed of multiple products.

この提案では，機器をまたがるフィーチャの表現として，プロダクトマップ [11] の表現を拡張する．しかし，機器をまたがったフィーチャが，複合システム上で，どう表出するかを表現できておらず，2.2 節で述べた通信プロトコルの混在などが扱えないという問題があった．本論文では，提案済みのこの手法を拡張した手法を提案する．

3. システム構成群の導出方法の提案

本研究で提案する，フィーチャ分析と充足可能性判定を用いた，テスト項目を網羅するシステム構成群を導出する方法について述べる．本章では提案手法のプロセスを中心に述べ，充足可能性判定に関わる部分の詳細は付録に記述する．テスト項目を網羅するシステム構成群を導出する方法は，ドメインエンジニアリング領域とアプリケーションエンジニアリング領域に分かれる（**図 4**）．

提案手法は，以下のプロセスで構成される．

(1) フィーチャ分析：ドメイン知識の分析を人手で行いフィーチャモデルとして整理する．

(2) フィーチャ登録：フィーチャモデルをツールへ入力する．

(3) サブシステム候補作成：ツールは，入力されたフィーチャモデルから，導出できるサブシステム候補を作成する．

(4) テスト項目分析：テスト項目とフィーチャモデルの間の関連を分析し表としてツールに入力する．

(5) システム構成導出：ツールは，サブシステム候補とテスト項目の分析結果の表からテスト項目を網羅するシステム構成群を導出する．

以降では各プロセスの詳細を示す．

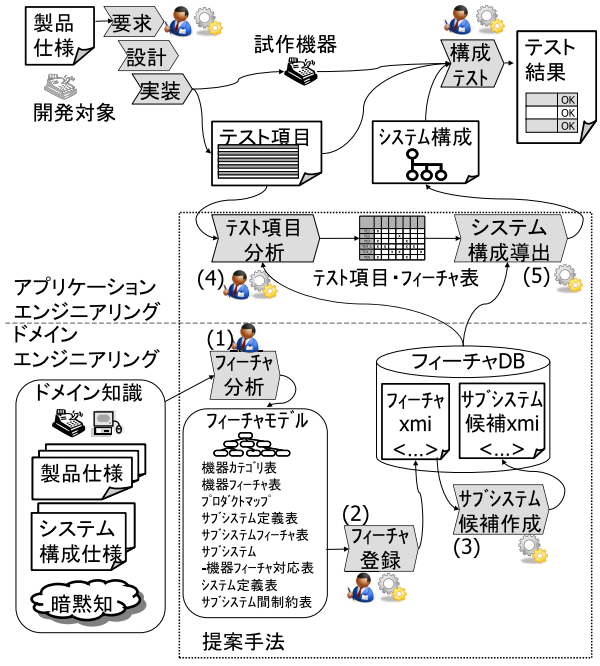


図 4 提案手法の概要

Fig. 4 Overview of the proposed approach.

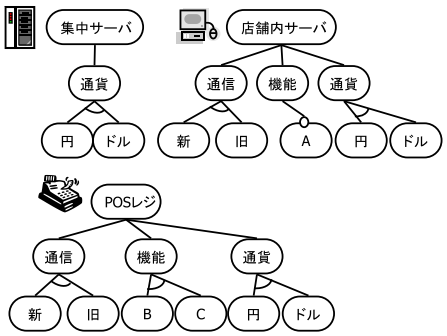


図 5 各機器カテゴリの機器フィーチャツリー
Fig. 5 Product feature tree for each category.

3.1 フィーチャ分析

ドメインエンジニアリング領域では，ドキュメント化されていない暗黙知や，複数のドキュメントに散らばったドメイン知識をフィーチャモデルとして整理する．フィーチャ分析の方法は，おおむね文献 [12] に記載のドメイン分析手法を用いる．提案手法では，このドメイン分析手法によるフィーチャ分析を，機器に閉じた特徴を表す機器フィーチャと，機器を組み合わせたサブシステムに生じるサブシステムフィーチャに対して行い，複合システムは，このサブシステムを組み合わせて表現する．

このプロセスにおいてフィーチャモデルに不備が生じると，提案手法が正しく実施できないというリスクがあるため，このプロセスには熟練した開発者に参加してもらい十分に時間をかける必要がある．ただし，このプロセスの成果はそのドメインにおいて再利用可能な情報となるため，初回だけ十分な時間をかければよい．

フィーチャ分析プロセスは以下の手順で実施する．

a. 機器フィーチャとそれらの制約・可変性の分析
b. 機器が持つ機器フィーチャの分析
c. サブシステムフィーチャとそれらの制約・可変性の分析
d. サブシステムフィーチャと機器フィーチャの関係を分析
e. システム全体のサブシステムフィーチャ制約を分析

それぞれの手順を，2.2 節の例を用いながら説明する．まず a では，各製品の仕様を入力として，各機器カテゴリについてフィーチャツリーを用いて機器の特徴を表すフィーチャとその可変性や制約について分析を行う．集中サーバ，店舗内サーバおよび POS レジのそれぞれのフィーチャ分析の結果を**図 5** に示す．

次の b では，各機器カテゴリに含まれる既存機器の製品仕様を入力として，図 5 のどのフィーチャを実現しているかを，プロダクトマップを用いて分析する．プロダクトマップでは，行が機器を表し，列に機器フィーチャを並べて，各機器が持つ機器フィーチャのマスに X を記載している．集中サーバ，店舗サーバ，POS レジのそれぞれのプロダクトマップを**表 2**，**表 3**，**表 4** に示す．

c のサブシステムフィーチャとそれらの制約・可変性の分析では，a で分析した機器フィーチャツリーおよび機器が組み合わさって生じる仕様を入力として，サブシステムフィーチャツリーを作成する．

表 2 集中サーバのプロダクトマップ
Table 2 Product map of center server.

機器名	通貨	
	円	ドル
CENTER_No.1	X	
CENTER_US		X

表 3 店舗サーバのプロダクトマップ
Table 3 Product map of shop server.

機器名	通信		機能	通貨	
	新	旧	A	円	ドル
SHOP_PERFECT	X		X	X	
SHOP_PRO	X			X	
SHOP_STANDARD		X	X	X	
SHOP_CLASSIC		X		X	
SHOP_PERFECT_US	X		X		X
SHOP_PRO_US	X				X
SHOP_STANDARD_US		X	X		X
SHOP_CLASSIC_US		X			X

表 4 POS レジのプロダクトマップ
Table 4 Product map of POS register.

機器名	通信		機能		通貨	
	新	旧	B	C	円	ドル
REGISTER_PRO	X		X		X	
REGISTER_STANDARD	X			X	X	
REGISTER_CLASSIC		X		X	X	
REGISTER_PRO_US	X		X			X
REGISTER_STANDARD_US	X			X		X
REGISTER_CLASSIC_US		X		X		X

サブシステムは複数の機器から構成され，システムは複数のサブシステムによって構成される．サブシステムは，どの機器カテゴリに属する機器を持つか，またその機器の台数に関する多重度によって，サブシステムの型を定義する．また，システムは，どのサブシステムの型をどれだけ持つかを多重度によって，システムの型として定義する．

POS レジシステムの例では，店舗内サーバ 1 台とそれに接続する複数台の POS レジで構成されるサブシステムと，集中サーバが 1 台で構成されるサブシステムが定義される．また，システムは，前者のサブシステムを複数と後者のサブシステムを 1 つ持つ．

サブシステムフィーチャは，サブシステムを構成する複数の機器が持つ機器フィーチャの組合せによって定まる仕様を表す．

サブシステムの例として，店舗内サーバ 1 台とそれに接続する複数台の POS レジで構成されるサブシステムを考える．このとき，このサブシステムにおける機器の組合せによる仕様は，通信プロトコルの混在・通貨の排他性・機能 A から機能 B への依存の 3 つがある．これらは，図 5 のような各機器に閉じたフィーチャでは表現できない．そのため，サブシステムを構成する店舗内サーバのフィーチャツリーと POS レジのフィーチャツリーを合成し，機器の

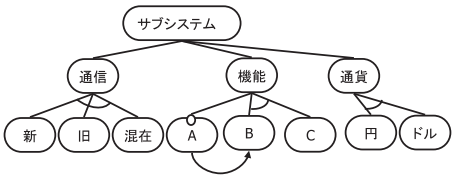


図 6 サブシステムフィーチャツリー
Fig. 6 Subsystem feature tree.

表 5 サブシステムにおける通信フィーチャの例
Table 5 Example of communication feature in subsystem.

	機器フィーチャ						サブシステムフィーチャ		
	店舗内サーバ		POSレジ		POSレジ				
	新	旧	新	旧	新	旧	新	旧	新旧混在
A	X		X		X		X		
B	X		X			X			X
C	X			X		X		X	
D		X	X		X			X	
E		X	X			X		X	
F		X		X		X		X	

組合せとして生じている仕様をフィーチャとして加えて，**図 6** のようなサブシステムフィーチャツリーとして分析する．図 6 では，通信プロトコルの混在の仕様は通信の下の混在フィーチャとして，機能 A の機能 B への依存は機能 A から機能 B への共起の制約として表現され，通貨の排他はオルタナティブとして表現できている．

d では，機器フィーチャツリーとサブシステムフィーチャツリーを入力として，それらの関係を分析する．この関係性を定義することで，機器を選択してサブシステムを構成した際に，機器フィーチャからどのサブシステムフィーチャが表出しているかを導き出すことができる．

サブシステムフィーチャは，単独もしくは複数の機器フィーチャで定義される．単独の場合，機器フィーチャがそのままサブシステムフィーチャとなる．複数の機器フィーチャの合成によってサブシステムフィーチャの有無が定まるケースは，通信プロトコルの混在の例が該当する．この通信プロトコルのサブシステムフィーチャと機器フィーチャのすべての組合せを**表 5** にまとめる．

表 5 の A の行のように，サブシステムを構成する機器のすべてが新プロトコルの機器フィーチャを持っていれば，そのサブシステムでは，新プロトコルのサブシステムフィーチャが現れる．新プロトコル対応の店舗内サーバは，通信相手が旧プロトコルの機器であった場合は，旧プロトコルとして通信し，通信相手が新プロトコルであった場合は，新プロトコルで通信可能のため，B の行のように，店舗内サーバが新プロトコルに対応する機器で，POS レジに新プロトコル対応機器と，旧プロトコルの機器が混じっていた場合，サブシステム上は新旧混在プロトコルとしてサブシステムフィーチャが表出する．複数の機器フィーチャの合成で定義されるサブシステムフィーチャは，それぞれに，表 5 に相当する表を定義する必要がある．

最後の e のシステム全体におけるサブシステムフィーチャ制約の分析では，サブシステムフィーチャツリーを入力として，システム全体に影響するサブシステムフィーチャの制約を分析し表で整理する．

表 6 サブシステム間の制約表
Table 6 Constraints between subsystem features.

		通信			機能			通貨	
		新	旧	混	A	B	C	円	ドル
通信	新	--	--	--	--	--	--	--	--
	旧	--	--	--	--	--	--	--	--
	混	--	--	--	--	--	--	--	--
機能	A	--	--	--	Ex	--	--	--	--
	B	--	--	--	--	--	--	--	--
	C	--	--	--	--	--	--	--	--
通貨	円	--	--	--	--	--	--	--	Ex
	ドル	--	--	--	--	--	--	Ex	--

2.2 節の例では，機能 A どうしの排他や通貨の排他性が該当する．これらは，システムに含まれるサブシステム間で生じるサブシステムフィーチャの共起（Co）や排他（Ex）として分析できる．図 6 に生じるサブシステムフィーチャを行と列にとり，マス目に制約を記述して**表 6** のように分析する．

表 6 の行「A」で列「A」のマスの Ex は，システム内に機能 A を持つサブシステムが存在するとき，他のサブシステムは機能 A を持ってはならないという制約を表し，行「ドル」で列「円」では，システム内の円とドルの排他を表す．

3.2 フィーチャ登録

フィーチャ登録では，3.1 節で分析したフィーチャをユーザがツールに入力し，ドメインエンジニアリングの結果として蓄積する．フィーチャモデルを登録するためのユーザインタフェースは，網羅性のある記述が可能なことから表形式を採用する．入力する表を以下に示す．

機器カテゴリ表 機器カテゴリを定義する．

機器フィーチャ表 機器カテゴリごとに，機器フィーチャツリー相当の情報を表として入力する．図 5 に相当．

プロダクトマップ 表 2・表 3・表 4 のように，機器それぞれを行として，機器が持つフィーチャを入力する．

サブシステム定義表 サブシステムを構成する機器カテゴリとそれらの機器数を多重度として定義する．

サブシステムフィーチャ表 サブシステムフィーチャツリーに相当する情報を入力する．図 6 の情報に相当．

サブシステム–機器フィーチャ対応表 サブシステムフィーチャと機器フィーチャの関係を入力する．表 5 に相当．

システム定義表 システムを構成するサブシステムとそれらの数を多重度として入力する．

サブシステム間制約表 システム内のサブシステムフィーチャ制約を入力する．表 6 に相当．

ツールにこれらのフィーチャ情報を入力すると，ツールは内部でデータモデルを構築する．データモデルの詳細

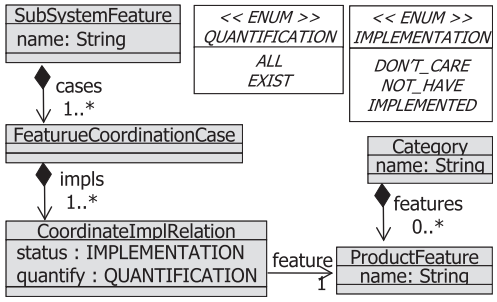


図 7 サブシステムフィーチャと機器フィーチャ対応

Fig. 7 Relations between subsystem feature and product feature.

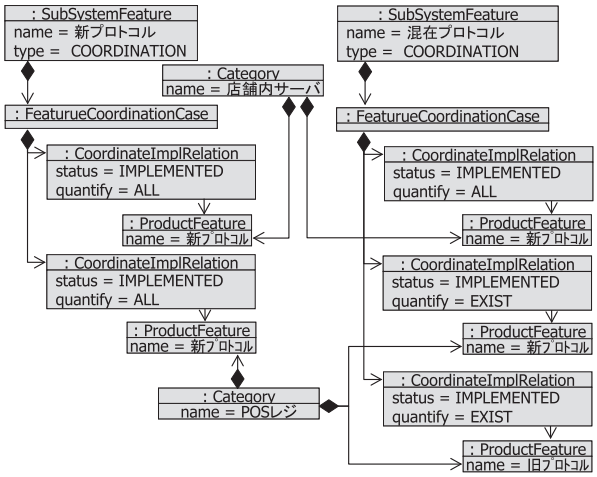


図 8 通信プロトコルにおけるオブジェクト図

Fig. 8 Object diagram for communication protocol.

は付録に記述する．特にサブシステムフィーチャと機器フィーチャの対応は，**図 7** に示すクラス図によって実現している．SubSystemFeature と ProductFeature の間の関係を，FeatureCoordinationCase と CoordinateImplRelation クラスで表現する．それぞれの FeatureCoordinationCase インスタンスは，どのような機器フィーチャの組合せで実現されているかを表す CoordinationImplRelation クラスのインスタンスを保持する．CoordinationImplRelation クラスは機器フィーチャとの関連クラスとなる．そのクラスの status プロパティは，feature 関連で参照している機器フィーチャの有無を列挙型 IMPLEMENTATION で表現する．また，quantify プロパティは限量子を意味する．参照している機器フィーチャが，サブシステム内の同じ機器カテゴリに属する機器のすべてに存在したときにサブシステムフィーチャが実現できることを示す場合，ALL を指定する．その機器フィーチャを満たす機器がサブシステム内に 1 つでも存在するときにサブシステムフィーチャが有効となる場合は，EXIST を指定する．

例における通信プロトコルのサブシステムフィーチャである新プロトコルおよび混在プロトコルを，オブジェクト図で表すと**図 8** のようになる．サブシステムフィーチャである新プロトコルは，店舗内サーバに属する機器フィーチャの新プロトコルと POS レジに属する機器フィーチャの新プロトコルのみで構成されるサブシステムにおいて実現できる．また，サブシステムフィーチャである混在プロトコルは，店舗内サーバに属する機器フィーチャは新プロトコルのみである点は同様で，POS レジに属する機器フィーチャの新プロトコルと旧プロトコルの両者を持つサブシステムにおいて実現できる．

ツールはフィーチャのデータモデルを構築後に，それらを用いて，フィーチャ間の制約や，各機器のフィーチャ実装情報に対して，充足可能性判定を行い，無矛盾であることを検証する．

表 7 サブシステム候補

Table 7 Candidates of subsystem.

サブシステム	通信			機能		
	新	旧	新旧混在	A	B	C
Sub a	X			X	X	
Sub b	X			X		X
Sub c	X				X	
Sub d	X					X
Sub e			X	X	X	
Sub f			X	X		X
Sub g			X		X	
Sub h			X			X
Sub i		X		X	X	
Sub j		X		X		X
Sub k		X			X	
Sub l		X				X

3.3 サブシステム候補作成

3.2 節までドメインエンジニアリングにおいて手作業によるプロセスは終わり，ツールによって入力されたフィーチャモデルから自動でサブシステム候補の作成を行う．

まず，ツールは図 6 にあるようなサブシステムフィーチャの可変性のすべてを掛け合わせて，**表 7** のように生じうるサブシステム候補を作る．表 7 の例は簡単化するために，通貨に関しては円を選択し，通信のオルタナティブ 3 パターンと，機能 A のオプショナルで 2 パターンと，機能 B と機能 C のオルタナティブで 2 パターンとを掛け合わせて 12 のサブシステム候補が生成されている．

表 7 の各行に対して，ツールは蓄積したフィーチャモデルの制約を用いて充足可能性判定を行い，サブシステムの候補を絞り込む．たとえば，Sub b，Sub f，Sub j の行は，サブシステムフィーチャである機能 A から機能 B への依存制約を満たさないため除外する．

また，ツールは図 7 のようなサブシステムフィーチャと機器フィーチャの対応関係を用いて，サブシステムフィーチャを満たすための機器の組合せが存在するかを検証する．Sub e と Sub g の行は，旧通信かつ機能 B を機器フィーチャとして持つ POS レジを必要とするが，表 4 に存在しない．これらのサブシステム候補は，実現するための機器の組合せが存在しないため除外する．

表 8 テスト項目–サブシステムフィーチャ表
Table 8 Relations between test case and subsystem.

	通信			機能			通貨	
	新	旧	混	A	B	C	円	ドル
TC1	X	--	--	X	--	--	X	--
TC2	--	--	X	--	--	X	X	--
TC3	--	--	--	X	X	--	X	--
TC4_1	--	--	--	--	X	--	X	--
TC4_2	--	--	--	--	--	X	X	--
TC5	--	--	--	X	X	--	X	--

サブシステム候補の作成と絞り込みは，テスト項目の入力後に行えば，本節で説明するような可変点を掛け合わせたすべての組合せの候補を必要とせず，テスト項目を満たすサブシステム候補だけを導出することができる．しかし，本節で述べたようなテスト項目入力前のドメインエンジニアリング領域においてサブシステム候補の作成と絞り込みの方法をとれば，製品開発ごとに行うテスト項目入力後の充足可能性判定の規模を小さくすることができるため，サブシステム候補の作成を前もって行うこととした．

3.4 テスト項目分析

本節では，フィーチャモデルを利用してテスト項目を分析するプロセスについて述べる．テスト項目の分析は，テスト項目が対象とするシステムの構造を分析する手順と，各テスト項目がどのフィーチャに関連するかを分析する手順の 2 つに分かれる．

テスト項目が対象とするシステムの構造を分析する手順では，テスト項目の記述を読み解き，テストを実施するためにはどのサブシステムがどれだけ必要であるかを分析する．例における TC4 では，「1 つの店舗内サーバとそれに接続する POS レジの中で機能 B が動作し，別の店舗内サーバとそれに接続する POS レジの中で機能 C が動作することを確認する」とあり，店舗内サーバと POS レジが接続されたサブシステムが 2 つ必要であると判明する．例における他のテスト項目では，店舗内サーバと POS レジが接続されたサブシステムは 1 つで実施できる．

次に，テスト項目とフィーチャの関連を分析する．テスト項目の分析に用いるフィーチャモデルの情報は，サブシステムフィーチャと機器フィーチャの 2 種類となるため，それぞれに対応して 2 種類のテスト項目–フィーチャ表を作成する．

まず，サブシステムフィーチャとテスト項目について分析を行いテスト項目–サブシステムフィーチャ表を作成する．この表は，行がテスト項目で，列は対象とするシステムに含まれるすべてのサブシステムフィーチャとなる．

たとえば，TC1 には「日本円」「新プロトコルで通信」「機能 A」といった記述が存在することから，これらを示すサブシステムフィーチャに該当することを示すマスに X を記入する．同様にして TC1 から TC5 を入力すると，**表 8** のようになる．TC4 は，店舗内サーバ 1 台と複数台の POS レジで構成されるサブシステムが 2 つ必要となるため，表 8 では 2 行に分ける．

表 9 テスト項目–機器フィーチャ表
Table 9 Relations between test case and product feature.

	フィーチャ												機器名						
	店舗内サーバ					POSレジ							店舗内サーバ				POSレジ		
	通信		機能	通貨		通信		機能		通貨									
	新	旧	A	円	ドル	新	旧	B	C	円	ドル		SHOP_PERFECT	SHOP_PRO	SHOP_STANDARD	SHOP_CLASSIC	REGISTER_PRO	REGISTER_STANDARD	REGISTER_CLASSIC
TC5	--	--	--	--	--	--	--	--	--	--	--		X	--	--	--	--	--	--

次に，テスト項目と機器フィーチャについて分析を行い，テスト項目–機器フィーチャ表を作成する．この表も，行にテスト項目で，列に機器フィーチャを並べる．また，テスト項目–機器フィーチャ表は，テスト項目が直接具体的な機器を指定している場合に対応するために，機器フィーチャの列だけではなく，機器名を持つ列も用意する．ユーザは，テスト項目の記述を確認し，機器フィーチャや具体的な機器に言及する記述があれば，そのマスに X を入力する．

TC5 は，テスト項目の記述の中に機器名である「SHOP_PERFECT」を指定している．これをテスト項目–機器フィーチャ表に入力した結果を**表 9** に示す．他のテスト項目 4 つは，機器フィーチャや機器名に関する言及がないため，テスト項目–機器フィーチャ表には入力しない．

このプロセスでは，フィーチャとテスト項目の対応を導出して分析を行うが，自然言語で記載されたテスト項目はフィーチャと対応をとるのが困難なケースが生じる可能性がある．その原因は，テスト項目とフィーチャの用語の不統一などからくるケースと，テスト項目が可変点であるフィーチャと関係ないケースが考えられる．前者は，テスト項目の再検討を行いフィーチャと用語の統一を行う．後者のケースでは，すべてのフィーチャと対応しないことをツールに入力すれば，導出した任意のシステムでそのテストが実施可能であると判定できる．

3.5 システム構成の導出

本節では，これまでの登録された情報を用いてツールが自動でシステム構成を導出するプロセスについて述べる．

まずツールは，表 8 と表 9 に登録されたテスト項目に対して，それを実現するサブシステムを表 7 のサブシステム候補の中から見つけ出し，それをテスト項目に紐づける．これは，サブシステム候補の持つサブシステムフィーチャや機器と，テスト項目において記述されたサブシステムフィーチャや機器の指定を論理式として記述し充足可能性判定をソルバで解くことで導出する．

テスト項目とサブシステム候補を紐づけた結果は，**表 10** のようになる．たとえば TC1 は，新プロトコルで機能 A

表 10 テスト項目とサブシステム候補の対応表
Table 10 Relations between test case and subsystem candidate.

	Sub a	Sub c	Sub d	Sub h	Sub i	Sub k	Sub l
TC1	X						
TC2				X			
TC3	X				X		
TC4_1	X	X			X	X	
TC4_2			X	X			X
TC5	X				X		

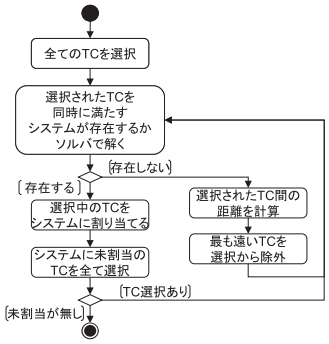


図 9 テスト項目を網羅するシステム構成導出
Fig. 9 Generation of system configurations.

を実現しているサブシステムが必要なので，それを満たすサブシステム候補は，表 7 では Sub a と Sub b があり，Sub b は制約違反が発生しているため，Sub a のみが選択できる．

次にツールは，より少ないシステム構成の数で，登録されたテスト項目を網羅するように，**図 9** のように貪欲法を用いてシステム構成を導出する．システム構成が割り当てられていないテスト項目すべてを選択し，1 つのシステム構成でそれらのテスト項目が実施可能か充足可能性判定を用いて検証する．

実施可能の場合，それをシステム構成として登録する．実施不可能であれば，選択されているテスト項目を評価して，テスト項目を 1 つ選択から除外し，残ったテスト項目が 1 つのシステム構成で実施可能かを検証する．除外するテスト項目は，テスト項目間の距離を表 10 の各行を符号化し各行の間のハミング距離を求めることで計算し，他のテスト項目への距離の総和が最も大きいものを選択する．除外されたテスト項目は，それらを集約して，また別のシステム構成で実施可能かを検証する同じ処理を繰り返す．

表 10 の例で実際にすべてのテスト項目を実施可能なシステム構成を導出すると，TC1 から TC5 のすべてを実施可能なシステム構成として，Sub a と Sub h を組み合わせたシステム構成が導き出される．これは蓄積されているすべての制約と矛盾が生じない．

4. ツールの開発

本研究では，提案手法を実現するツールの開発を行った．

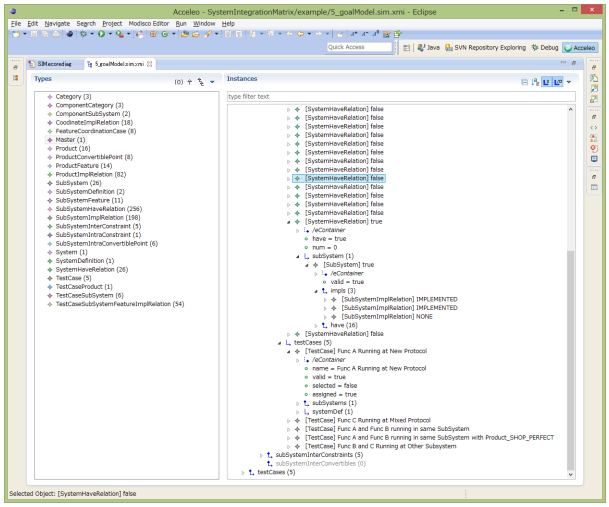


図 10 ツールのスクリーンショット
Fig. 10 Screen shot of the tool.

開発言語は Java [13] を用いて Eclipse [14] のプラグインとして開発を行った．データモデルは，EMF [15] を用いてクラス図を記述し，その機能を用いてコード生成を行っている．

提案手法で利用している充足可能性判定を解くにあたり，SMT ソルバである CVC4 [16] を利用した．提案手法の範囲では，SAT ソルバで十分だが，今後，複合システムに対する接続数の最大値などの制約を扱えるように拡張したいため，整数を扱える SMT ソルバを選択した．入力形式は，多くの SMT ソルバが対応している SMT-LIB2 [17] を選択した．SMT ソルバに与える SMT-LIB2 形式を生成するために，テンプレートを記述することでモデルからテキストファイルを生成できる Acceleo [18] を用いた．

ツールの表によるユーザインタフェースは，未実装となっている．現在は，EMF のモデルインスタンスを編集するためのプラグインである MoDisco [19] を用いて XMI を直接編集し，フィーチャやテスト項目の登録を行っている．MoDisco による XMI の編集画面を**図 10** に示す．

5. ケーススタディ

5.1 対象と結果

2.2 節の仮想の多店舗向け POS レジシステムの事例を，提案手法を用いてシステム構成の導出を行った．提案手法で作業が必要となるのは，ドメインエンジニアリング領域の 3.1 節フィーチャ分析，3.2 節フィーチャ登録と，アプリケーションエンジニアリング領域の 3.4 節テスト項目分析となる．このケーススタディでは，ドメイン知識が必要となる 3.1 節と 3.2 節は筆者が行い，後半の 3.4 節は，筆者と組込みソフトウェアの開発技術研究に従事する協力者 2 名（経験 6 年，経験 11 年）がそれぞれ行った．

また対照ケーススタディとして，同じ POS レジシステムの事例に対して，提案手法の協力者とは別の協力者 3 名

表 11 提案手法の適用結果
Table 11 Result of proposed approach.

プロセス		項目	結果
3.1 フィーチャ分析 3.2 フィーチャ登録 (執筆者実施)		作業時間	60分
		機器カテゴリ数	3
		機器数	16
		機器フィーチャ数	14
		機器フィーチャ制約・可変点数	8
		サブシステムフィーチャ数	11
		サブシステム内制約・可変点数	7
		サブシステム間制約数	5
3.3サブシステム絞込み (ツール自動)		計算時間	15秒
		サブシステム候補数	26
		有効サブシステム候補数	14
3.4 テスト項目分析		テスト項目数	5
	執筆者	作業時間	30分
	経験6年	作業時間	35分
	経験11年	作業時間	45分
3.5 システム構成導出 (ツール自動)		計算時間	40秒
		導出したシステム構成数	1

表 12 手作業による結果
Table 12 Result by hand.

協力者	項目	結果
インターン	作業時間	90分
	導出したシステム構成数	3
経験3年	作業時間	40分
	導出したシステム構成数	1
経験9年	作業時間	90分
	導出したシステム構成数	1

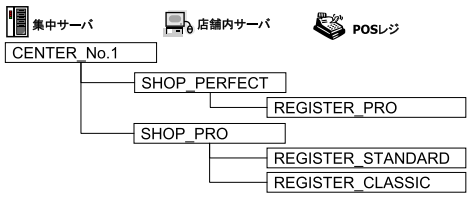


図 11 提案手法で導出したシステム構成
Fig. 11 Generated system configuration by the proposed approach.

（インターン学生，経験 3 年，経験 9 年）に本論文の 2.2 節の仕様とテスト項目を渡し，手作業によるシステム構成の導出を行った．

提案手法を用いたケーススタディの結果を**表 11**，手作業で行った結果を**表 12** に示す．提案手法において使用した環境は，CPU: Core i3 1.8 GHz，Memory: 4 GB，OS: Windows8 64 bit のタブレット PC を用いている．

提案手法を用いたケーススタディでは，筆者および協力者 2 名の行ったテスト項目分析の結果は同様のものとなり，その結果を用いてツールを実行して，テスト項目を網羅するシステム構成 1 つを導出することができた．ただし，経験 6 年目の協力者に関しては実験開始時に行った提案手法の説明では，手法を誤解しテスト項目とサブシステム候補の関連付けを行っていた．そのため，再度，提案手法の説明を行った後，テスト項目分析を行った．表 11 の経験 6 年目の協力者の作業時間は，当初のサブシステム候補の関連付けの作業時間も含んでいる．

提案手法とツールで導出したシステム構成を，**図 11** に示す．図 11 が示すシステム構成では，SHOP_PERFECT につながるサブシステムにおいて，新プロトコルの下で機能 A・機能 B が実現し，SHOP_PRO につながるサブシステムにおいて，新旧混在プロトコルの下で機能 C が実現されている．そのため，図 11 が示すシステム構成において，TC1 から TC5 までのテスト項目を網羅している．

手作業で導出されたシステム構成を調査すると，インターンの導出したシステム構成は，正しくテスト項目を網羅していたが 3 構成を導出した．経験 3 年の方が導出したテスト項目を網羅するシステム構成が 1 つだが，冗長な機器が加えられていた．また，経験 9 年の方の導出したシステム構成では，TC2 を実施できないシステム構成であった．それぞれにヒアリングを行った結果，両者ともに 2.2 節の自然言語の記述を誤解していたことが原因と判明した．

5.2 考察

本節では，1 章で述べた研究課題の RQ1 から RQ4 に沿って，ケーススタディについて考察を述べる．

RQ1 テスト項目からシステム構成を導出する際，暗黙知などの属人性を低減可能か？

対照ケーススタディの手作業によるシステム構成の導出では，三者三様の結果となり属人性が高いことが分かる．一方の提案手法によるケーススタディでは 3 名が同様のテスト項目分析を行えたため，属人性を低減させている．ただし，1 名が提案手法を誤解してテスト項目分析を行えなかったため，提案手法そのものの習熟が必要となる．

RQ2 テスト項目を網羅するシステム構成を導出可能か？

対照ケーススタディの手作業によるシステム構成の導出では，3 名中 2 名がテスト項目を網羅するシステム構成群を導出しているが，1 名は網羅できていなかった．

提案手法においては，図 9 に示すように「未割当てのテスト項目がない」ことが終了条件になっており，テスト項目を網羅するシステム構成を導出できる．実際に，提案手法によって，テスト項目を網羅するシステム構成 1 つを得た．

RQ3 テスト項目を網羅するために導出するシステム構成の数が現実的に少ない数か？

手作業では，1 名がシステム構成を 3 つ導出した．他 2 名は 1 つずつ導出し，それぞれ冗長な機器を含むケースと，テスト項目を網羅していないケースとなる．

提案手法では，1 つのシステム構成が TC1 から TC5 に対応するものとして導出されたため，実用的であるといえる．

提案手法では，計算量爆発を避けるため，与えられたテスト項目に対して理想的なシステム構成の最小数を求める方法をとっていない．提案手法では貪欲法を用いることによって，より多くのテスト項目をカバーするシステム構成を得る．

RQ4 テスト項目からシステム構成を導出する際，実用的な時間で実施可能か？

対照ケーススタディの手作業によるシステム構成の導出では，1 名が 40 分で，他 2 名が 90 分となった．

提案手法を用いたケーススタディでは，フィーチャ分析とフィーチャ登録で 60 分かかっているが，このプロセスはドメインエンジニアリング領域であり，その結果も再利用できるため，比較を行う時間としてはテスト項目分析のみを考える．テスト項目分析では 3 名が作業を行い，それぞれ 30 分，35 分，45 分かかっている．これらは，手作業による時間に対して短い傾向で，実用的な範囲内にあると考える．また，ツールによる計算時間は，分析や入力する時間と比べると無視できるほど小さかった．

5.3 妥当性への脅威

内的妥当性への脅威として，提案手法を用いたケーススタディでは，ドメインエンジニアリング領域は執筆者が行っており，提案手法への習熟度が高いことがあげられる．さらに協力者が行ったテスト項目分析では提案手法の誤解が発生しており，提案手法の習熟方法が課題となる．

また，ケーススタディの複合システムにおける仕様やテスト項目は，現実に存在する複合システムのすべての仕様を網羅できている保証がなく，そのフィーチャモデルの規模も小さいため，内的妥当性への脅威となる．今後，様々な複合システムの仕様を調査し，提案手法の表現可能性や，フィーチャモデルの規模のスケーラビリティを検証する必要がある．

外的妥当性への脅威として，他の実際の複合システムにおいて，提案手法が有効であるかを検証していない点があげられる．この脅威に対しても，今後，様々な実際の複合システムへの調査・試行が必要となる．

6. おわりに

6.1 結論

本研究では，機器に対するフィーチャ分析だけではなく，機器が組み合わさり複合システムとなった際に表出するフィーチャを分析し，複合システムに表出したフィーチャと機器のフィーチャの関係性の分析を行い，その結果を用いてテスト項目を分析し，充足可能性判定を利用してテスト項目を網羅するシステム構成群を導出する方法を提案した．また，提案手法を実現するためのツールを実装し，動機付けの例において提案手法のケーススタディを行った．我々の提案する方法によると，ドメイン知識に乏しい開発者であっても，蓄積されたフィーチャ情報を利用して，テスト項目を分析することで，テスト項目を網羅し，現実的に実施できる数のシステム構成群を得ることが可能となる．

6.2 将来への展望

本論文の執筆段階では，基本機能を優先したため，本研究を完成させるためには，以下の点を拡充する必要がある．

- 提案する表形式のユーザインタフェースの開発
- 実際の複合システムへの試行と評価

また，機器の選択後にそれらの機器の接続数や，識別のためのアドレスの概念の拡張を検討している．

参考文献

[1] クラウス・ポール，ギュンター・ベックレ，フランク・ヴァン・デル・リンデン：ソフトウェアプロダクトラインエンジニアリング—ソフトウェア製品系列開発の基礎と概念から技法まで，エスアイビー・アクセス (2009).

[2] Olimpiew, E.M. and Gomaa, H.: Model-based Test Design for Software Product Line, *SPLiT'08*, *5th Software Product Line Testing Workshop in SPLC2008* (2008).

[3] Hervieu, A., Baudry, B. and Gotlieb, A.: PACOGEN: Automatic Generation of Pairwise Test Configurations from Feature Models, *ISSRE '11*, *Proc. 2011 IEEE 22nd International Symposium on Software Reliability Engineering* (2011).

[4] Oster, S., Zorcic, I., Markert, F. and Lochau, M.: Moso-PoLite – Tool Support for Pairwise and Model-Based Software product Line Testing, *VaMoS '11*, *Proc. 5th Workshop on Variability Modeling of Software-Intensive Systems* (2011).

[5] Kitamura, T., Do, N.T.B., Ohsaki, H., Fang, L. and Yatabe, S.: Test-Case Design by Feature Trees, *Proc. ISoLA 2012*, LNCS 7609, pp.458–473, Springer (2012).

[6] 川上真澄，小川秀人，加藤正恭：デジタル家電に対する All-Pair テストの改善と適用，*Japan Symposium on Software Testing* (2008).

[7] Hsu, H. and Orso, A.: MINTS: A General Framework and Tool for Supporting Test-suite Minimization, *ICSE'09* (2009).

[8] 西 康晴，飯塚悦功：ソフトウェアを対象にした構成テストの設計，電子情報通信学会論文誌，Vol.J84-D-I, No.11, pp.1542–1552 (2001).

[9] 新原敦介，渡辺浩之，角至真一，川上真澄，小川秀人：システムテストのためのフィーチャ分析を用いた機器構成の選択方法，情報処理学会研究報告ソフトウェア工学 (SE)，2010-SE-170, Vol.21, pp.1–7 (2010).

[10] Shimbara, D., Watanabe, H., Kakushi, S., Kawakami, M. and Ogawa, H.: Feature-analysis-based Selection Method for System Configuration for System Testing, *Product Line Approaches in Software Engineering 2012 3rd International Workshop* (*PLEASE2012*) (2012).

[11] Bayer, J., Flege, O., Knauber, P., et al.: PuLSE: A Methodology to Develop Software Product Lines, *Symposium on Software Reusability*, Los Angeles, CA, USA (1999).

[12] Kang, K.C., Cohen, S., Hess, J., et al.: Feature Oriented Domain Analysis (FODA) Feasibility Study, *Software Engineering Institute* (1990).

[13] Java（ORACLE 社の登録商標），入手先 ⟨http://java.com/⟩.

[14] Eclipse（Eclipse Foundation, Inc. の商標），入手先 ⟨http://www.eclipse.org/⟩.

[15] Eclipse Modeling Framework, available from ⟨http://www.eclipse.org/modeling/emf/⟩.

[16] CVC4, available from ⟨http://cvc4.cs.nyu.edu/web/⟩.

[17] SMT-LIB2, available from ⟨http://www.smtlib.org/⟩.

[18] Acceleo, available from ⟨http://www.acceleo.org/pages/home/en⟩.
[19] MoDisco, available from ⟨http://www.eclipse.org/MoDisco/⟩.

付　　録

A.1 データモデルと充足可能性判定

3 章では提案手法のプロセスと入力する情報について，例とともに述べた．本付録では提案手法のうち，充足可能性判定を用いる部分に関して，内部で用いるデータモデルの定義と，ツールが生成する充足可能性判定式について述べる．

A.1.1 データモデル

本付録ではデータモデルをクラス図で表現する．データモデルは，機器階層とサブシステム階層とシステム階層の 3 つの階層に分かれ，テスト項目を表すモデルは，この 3 つの階層を横断する形で表現される．

A.1.1.1 機器階層のモデル

機器階層におけるモデルを**図 A·1** と**図 A·2** のクラス図で表す．図 A·1 は機器フィーチャとその制約や可変性を表現し，図 A·2 は機器フィーチャを持つ機器そのものを表現する．本論文では，クラス図に関して，機器そのものや，機器を集約する実体を表すクラスを白抜きで表現し，機器のフィーチャやフィーチャ間の関係のように，実体をともなわないクラスを灰色で表現している．また，クラスのプロパティとして現れる列挙型は，イタリックで図中に表現している．

図 A·1 における，Category クラスは，機器カテゴリを表し，Category クラスのインスタンスは，例では集中サーバ，店舗内サーバ，POS レジに該当する．Category はそれぞれに，異なるフィーチャの集合 ProductFeature を持つ．先の例では，新プロトコルや通貨の円対応などが該当する．

ここで，フィーチャ分析では木構造で表現していたが，ツール実装上は簡略化して木構造の葉の要素のみを集合として持つようにしている．集中サーバに現れる通貨のようなオルタナティブフィーチャ（選択肢の中から 1 つを選択）である円とドルや，店舗内サーバのオプショナルフィーチャ（そのフィーチャの有無を選択）である機能 A は，ProductFeatureConvertiblePoint クラスを用いて表現する．オルタナティブフィーチャの場合は，同クラスから feature 関連が該当する複数の機器フィーチャに張られ，プロパティ convertible は ENUM 型のオルタナティブが選択される．オプショナルフィーチャの場合は，同クラスから単一の feature 関連が該当する機器フィーチャに張られる．また，機器フィーチャの共起や排他関係は

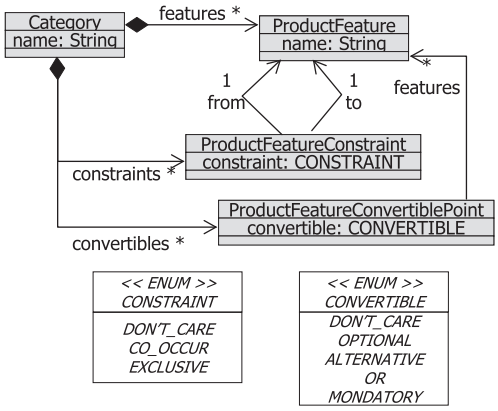


図 A·1 機器フィーチャのクラス図
Fig. A·1 Class diagram of product feature.

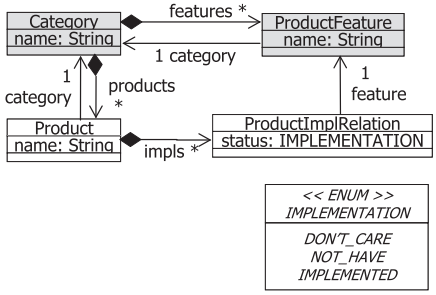


図 A·2 機器のクラス図
Fig. A·2 Class diagram of product.

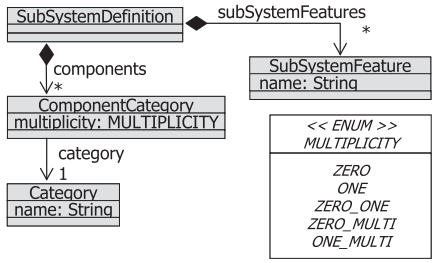


図 A·3 サブシステムフィーチャのクラス図
Fig. A·3 Class diagram of subsystem feature.

ProductFeatureConstraint クラスによって表現する．同クラスは機器フィーチャに対して from と to の関連を持ち，制約発生元と制約発生先を意味し，プロパティ constraint によって，共起（CO_OCCUR）もしくは排他（EXCLUSIVE）を指定する．

図 A·2 では，実際の機器を表すクラスとして Product を定義している．Product は，その機器がどのフィーチャを持っているかを表す ProductImplRelation クラスを保持し，この関連クラスが機器フィーチャへの関連を持つ．実際に機器が，機器フィーチャを持っているとき，この関連クラスのプロパティ status には IMPLEMENTED が設定され，機器フィーチャを持っていない場合は NOT_HAVE が設定される．ここで，列挙型 IMPLEMENTATION の中で，DON'T_CARE はテスト項目を表すモデルで使用するため後述する．

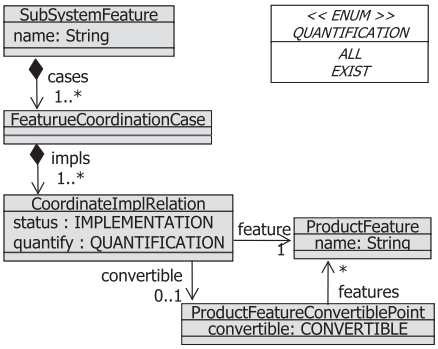


図 A·4 サブシステムフィーチャと機器フィーチャ
Fig. A·4 Relation of subsystem feature and product feature.

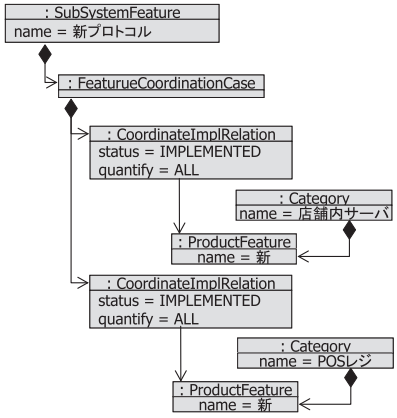


図 A·5 新プロトコルのオブジェクト図
Fig. A·5 Object diagram of new protocol.

A.1.1.2 サブシステム階層のモデル

サブシステム階層におけるモデルを，**図 A·3**，**図 A·4**，**図 A·7**，**図 A·8** で示す．図 A·3 はサブシステムの型と，サブシステムフィーチャを表し，図 A·4 はサブシステムフィーチャと機器フィーチャとの関係を表している．図 A·7 は，サブシステム階層の中で，サブシステムフィーチャ間の共起や排他制約，可変性を表現するモデルを表す．図 A·8 は，これらのサブシステムフィーチャを持つサブシステムそのものを表現するモデルを表す．

図 A·3 の SubSystemDefinition クラスは，サブシステムの型を表す．同クラスが保持する ComponentCategory クラスは，サブシステムの型のクラスと機器カテゴリを表すクラスの間を取り持つ関連クラスで，多重度を表す列挙型 MULTIPLICITY をプロパティとして持つ．

本論文の例では，機器カテゴリの店舗内サーバが 1 台と，POS レジが複数台集まり，1 つのサブシステムを構築するため，このサブシステムの型を表す SubSystemDefinition のインスタンスは，MULTIPLICITY を ONE とし店舗内サーバの Category を指す ComponentCategory インスタンスと，1 台以上複数台を表す ONE_MULTI を持ち POS レジの Category を指す ComponentCategory を持つ．

図 A·4 では，SubSystemFeature と ProductFeature の間の関係を，FeatureCoordinationCase と CoordinateImplRelation クラスで表現する．これらのクラスは，複数の機器フィーチャの合成によってサブシステムフィーチャの有無が定まる場合を表現している．

通信プロトコルの例では，FeatureCoordinationCase クラスの 1 つのインスタンスは，表 5 の 1 行に相当する．たとえば，サブシステムフィーチャの新プロトコルを表す SubSystemFeature クラスのインスタンスは，表 5 における A の行を表す FeatureCoordinationCase クラスのインスタンス 1 つを持ち，サブシステムフィーチャの旧プロトコルを表す SubSystemFeature クラスのインスタンスは，表 5 の C から F の行に該当し，FeatureCoordinationCase クラスのインスタンスを 4 つ保持する．

それぞれの FeatureCoordinationCase インスタンスは，

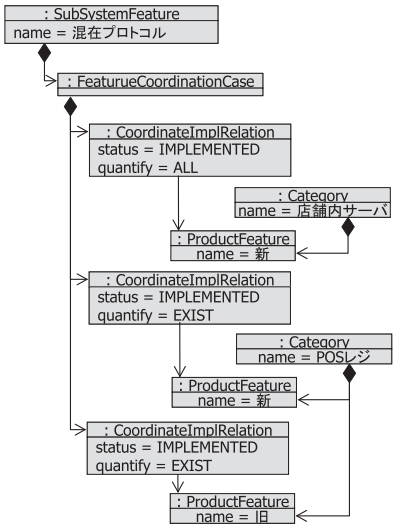


図 A·6 混在プロトコルのオブジェクト図
Fig. A·6 Object diagram of mix protocol.

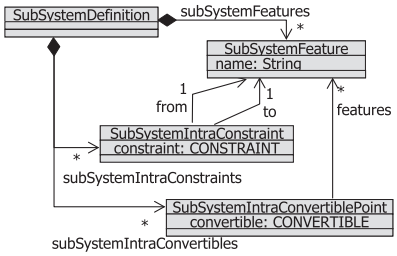


図 A·7 サブシステムフィーチャ制約のクラス図
Fig. A·7 Class diagram of feature constraint of subsystem.

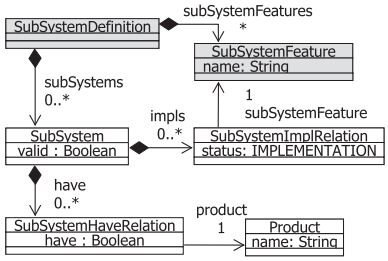


図 A·8 サブシステムのクラス図
Fig. A·8 Class diagram of subsystem feature.

どのような機器フィーチャの組合せで実現されているかを表す CoordinationImplRelation クラスのインスタンスを保持し，同クラスは機器フィーチャとの関連クラスとなる．CoordinationImplRelation の status プロパティは，feature 関連で参照している機器フィーチャの有無を列挙型 IMPLEMENTATION で表現する．また，quantify プロパティは限量子を意味する．参照している機器フィーチャがサブシステム内の同じ機器カテゴリに属する機器のすべてに存在したときにサブシステムフィーチャが実現できることを示す ALL を指定し，その機器フィーチャを満たす機器がサブシステム内に 1 つでも存在するときにサブシステムフィーチャが有効となる EXIST を指定する．

表 5 における A と B の行であるサブシステムフィーチャの新と新旧混在について，図 A·4 のクラス定義を用いてオブジェクト図で表現すると**図 A·5** と**図 A·6** のようになる．

図 A·7 は，サブシステム階層の中で，サブシステムフィーチャ間の共起や排他制約，可変性を表現するモデルを表す．このサブシステムフィーチャ間の制約や可変性は，図 A·1 で示す機器フィーチャ間の制約や可変性と同じ構造を持つ．このサブシステム階層での制約によって，例における通貨の排他の例や，機能 A の機能 B への共起を表現することができるようになる．

ここで，クラス名に Intra が含まれるのはサブシステム内を意味する．サブシステム間のサブシステムフィーチャの関係は，Inter を含むクラスとして後述する．

図 A·8 は，これらのサブシステムフィーチャを持つサブシステムそのものを表現するモデルを示す．SubSystem クラスのインスタンスは，サブシステムの型を定義する SubSystemDefiition クラスから保持されており，実際の機器を表すクラスである Product との間に，SubSystemHaveRelation という関連クラスを持つ．このクラスによって，サブシステムがどの機器を持つのかを表現する．参照先の Product クラスインスタンスがサブシステムに含まれる場合は，SubSystemHaveRelation クラスの have プロパティに真が設定され，そうでない場合は偽となる．

また，SubSystem クラスは，サブシステムフィーチャを表す SubSystemFeature クラスとの間に，関連クラス SubSystemImplRelation を持ち，どのフィーチャを実現しているかを表現する．サブシステムとサブシステムフィーチャの間の関連クラス SubSystemImplRelation は，機器と機器フィーチャの間の関連クラス ProductImplRelation と同様の構造を持つ．

サブシステムを表す SubSystem クラスは，valid プロパティを持ち，そのサブシステムの有効性を表現している．これは，テスト項目を網羅するシステム構成群を導出する方法内で利用する．サブシステムフィーチャの可変点すべての組合せによってサブシステムの候補を作成した後，機器との組合せや共起や排他制約によって生じえないサブシステムであると判明した際にこのプロパティを偽に設定し，サブシステムが有効ではないことを表す．

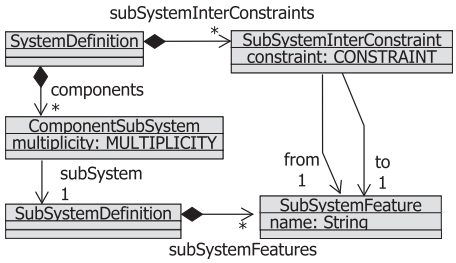


図 A·9 システムのフィーチャ制約のクラス図
Fig. A·9 Class diagram of feature constraint of system.

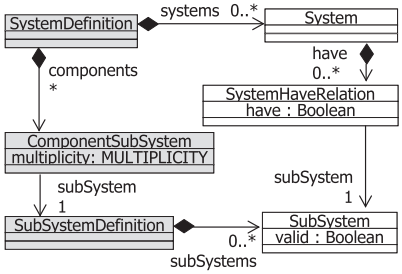


図 A·10 システムのクラス図
Fig. A·10 Class diagram of system.

A.1.1.3 システム階層のモデル

システム階層のモデルを，**図 A·9** と**図 A·10** で示す．

図 A·9 は，システムの型を表す SystemDefinition クラスと，そのシステムの型に属するサブシステム間に生じるサブシステムフィーチャの共起や排他などの制約を表す SubSystemInterConstraint クラスを示している．

SystemDefinition クラスは，SubSystemDefinition クラスとの間の関連クラス ComponentSubSystem を持ち，システムが，どのサブシステムの型の組合せで構成されているかの型を表現している．関連クラス ComponentSubSystem は多重度を表す multiplicity プロパティを列挙型 MULTIPLICITY で持つ．この関係は，サブシステムと機器の間の関連クラス ComponentCategory と同様となる．

SubSystemInterConstraint クラスは，本論文の例にある機能 A どうしの排他の例を表現するのに用いられる．この例において，SubSystemInterConstraint クラスのインスタンスが生成され，プロパティ constraint は EXCLUSIVE に設定され，サブシステムフィーチャへの関連である from と to は両方ともに機能 A を表す SubSystemFeature インスタンスを参照する．この制約は，サブシステム間に対してのみ機能するため，サブシステムの内部でのサブシステムフィーチャ間の制約である SubSystemIntraConstraint とは別に定義する．

図 A·10 は，システムの実体を表すための System クラ

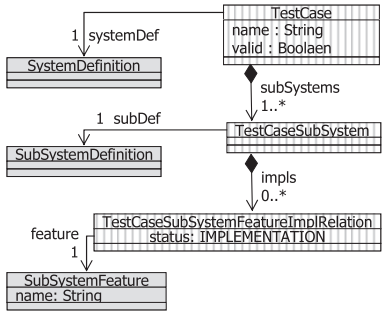


図 A·11 システム階層とサブシステム階層のテスト項目
Fig. A·11 Test case at system layer and subsystem layer.

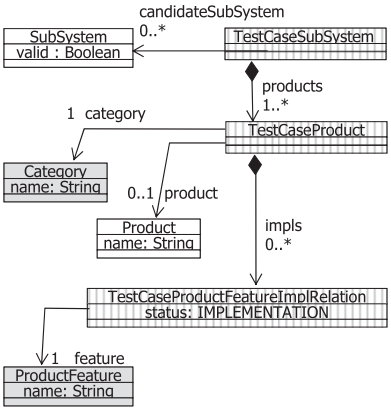


図 A·12 機器階層のテスト項目
Fig. A·12 Test case at product layer.

スと，そのシステムに含まれるサブシステムを示すための関連クラス SystemHaveRelation を示す．これは，サブシステムのクラス SubSystem と機器のクラス Product の間の SubSystemHaveRelation 関連クラスと同様の関係を表している．

A.1.1.4 テスト項目のモデル

ここでは，モデルを参照して，テスト項目を表現するためのモデルを，ストライプの網掛けのクラスとして，**図 A·11** と**図 A·12** に示す．テスト項目のモデルは，機器やサブシステムの階層を横断しており，図 A·11 ではシステム階層とサブシステム階層との関連を，図 A·12 で機器階層との関連を示す．

まず，テスト項目を表すのは TestCase クラスである．テスト項目ごとに TestCase インスタンスを生成する．本論文で述べたテスト項目の例では，TC1 から TC5 までのインスタンスを生成する．テスト項目は，テスト対象とするシステムがどのようなシステムであるかを指定する必要があるため，TestCase クラスはシステムの型を表す SystemDefinition クラスへの参照を持つ．

TestCase クラスは，対象とするシステム内のどのようなサブシステムについてテストをするべきかを表現するために，複数の TestCaseSubSystem クラスのインスタンスを持つ．TestCaseSubSystem は，そのテスト対象となるサブシステムが，どのサブシステムの型を持つかを意味するため SubSystemDefinition への参照を持つ．本論文で述べたテスト項目の例では，TC1，TC2，TC3，TC5 のテスト項目は，1 つのサブシステムに閉じている内容なので，TestCaseSubSystem クラスのインスタンスは 1 つとなる．TC4 の例では，店舗内サーバと POS レジの組合せが 1 つのサブシステムを意味し，機能 B を持つサブシステムと機能 C を持つサブシステムの両方がシステム内に存在して動作することを確認したいため，TestCaseSubSystem インスタンスは，2 つ生成する必要がある．

TestCaseSubSystem クラスは，そのサブシステムが持つべきフィーチャを表すための，TestCaseSubSystemFeatureImplRelation 関連クラスを持ち，そのクラスは SubSystemFeature を参照する．この関連クラスはプロパティとして列挙型 IMPLEMENTATION の status を持ち，参照しているサブシステムフィーチャの存在の有無を表現する．ここで，テスト項目を表すモデルでは，この関連クラスが参照しているサブシステムフィーチャの有無がテスト項目としてはどちらでもよい場合，列挙型 IMPLEMENTATION の DON'T_CARE を用いる．

ここまでのモデルで，TC1，TC2，TC3，TC4 のテスト項目を表現することができる．たとえば TC2 では，1 つの TestCaseSubSystem インスタンスを含む TestCase インスタンスを生成し，その TestCaseSubSystem インスタンスは，サブシステムフィーチャとして日本円，新旧混在の通信，機能 C に対して status を IMPLEMENTED にする TestCaseSubSystemFeatureImplRelation 関連クラスと，そのほかのサブシステムフィーチャに対して DON'T_CARE にする関連クラスを保持する．

図 A·12 は機器階層のテスト項目を示す．TC5 のように特定の機器を含むようなテストケースや，機器フィーチャを指定しているテスト項目を表現するためのクラスとなる．テスト対象のサブシステムを表現する TestCaseSubSystem は，そのサブシステムに含まれるテスト対象の機器を表すクラスとして TestCaseProduct を持つ．TestCaseProduct クラスは，どの機器カテゴリを表すかを示すため，Category クラスへの参照を持つ．また，TestCaseProduct は，特定の機器を表現する場合と，特定の機器を指定せずにその機器が持つべきフィーチャを指定するケースの 2 つの表現を持つ．

TC5 は前者の場合に該当し，TestCaseProduct が持つ Product クラスへの関連によって SHOP_SERVER_PERFECT という機器を参照して表現する．後者の場合は，TestCaseProduct から機器フィーチャを表す ProductFeature への関連クラスとして TestCaseProductFeatureImpleRelation クラスのインスタンスを保持させて表現する．この関連クラスは，TestCaseSubSystemFeatureImplRelation がサブシステムフィーチャへの関連であったと同様の構造を持ち，列挙

型 IMPLEMENTATION の DON'T_CARE 要素によって機器フィーチャの有無のどちらでもよいケースも表現できる．

A.1.2 充足可能性判定式

本章では，充足可能性判定に用いる式を OCL で記述する．式は，入力されたフィーチャ情報の妥当性を検証する式と，機器を組み合わせたサブシステム候補に対してその妥当性を検証する式と，サブシステム候補を組み合わせたシステムがテスト項目を網羅するかを検証する式で構成される．

A.1.2.1 フィーチャモデルの充足可能性判定

本節では 3.2 節のフィーチャ登録で入力されたフィーチャ情報や機器情報に矛盾がないかを検証する目的で充足可能性判定で妥当性を検証する．式はフィーチャ間の関係であるオルタナティブとオアと共起と排他となる．

オルタナティブフィーチャ

```
context Category inv:
 self.convertibles->forAll( c | c.convertible = ALTERNATIVE
 implies
 self.products->forAll(p |
 p.impls->sellect( status = IMPLEMENTED)
        ->collect( feature)
        ->select( f | c.features->includes(f) ) ->size() = 1 ) ) )
```

オアフィーチャ

```
context Category inv:
 self.convertibles->forAll( c | c.convertible = OR
 implies
 self.products->forAll(p |
 p.impls->sellect( status = IMPLEMENTED)
        ->collect(feature)
        ->select( f | c.features->includes(f) ) ->size() >= 1 ) ) )
```

共起制約

```
context Category inv:
 self.constraints->forAll(c | c.constraint = CO_OCCUR
 implies
 self.products->forAll( p |
 p.impls->select( status = IMPLEMENTED)
        ->collect( feature)
        ->includes( c.from )
   implies
   p.impls->select( i | i.status = IMPLEMENTED)
        ->collect(feature)
        ->includes( c.to ) ) )
```

排他制約

```
context Category inv:
  self.constraints->forAll(c | c.constraint = CO_OCCUR
  implies
  self.products->forAll( p |
  p.impls->select(status = IMPLEMENTED)
        ->collect(feature)
        ->includes( c.from )
    implies
    p.impls->select(status = IMPLEMENTED)
        ->collect(feature)
        ->excludes( c.to ) ) )
```

本節に記載の式は機器フィーチャモデル向けであるが，サブシステムフィーチャモデルにおける関係や制約も同様の式で妥当性を判定する．

A.1.2.2 サブシステム候補の充足可能性判定

すべてのサブシステム候補に対して，そのサブシステム候補が実現するサブシステムフィーチャが，そのサブシステム候補が持つ機器群によって実現できるのかを検証する．

```
Context SubSystemDefinition inv:
  self.subSystems->forAll( ss |
  ss.impls -> select( status = IMPLEMENTED)
          -> collect (subSystemFeature)
          -> forAll( ssf |
  ssf.cases->exists( case |
  case.impls-> select( status = IMPLEMENTED
                          and quantify = ALL)
           -> forAll( cir |
  ss.have   ->select( have = true)
          -> collect( product)
          -> select (cat = cir.feature.cat)
          -> forAll( p |
  p.impls -> select( status = IMPLEMENTED)
          ->collect( feature)
          ->includes(cir.feature) ) ) ) ) )
```

上記の不変式は，サブシステムフィーチャ ssf を満たすためには，サブシステム ss に含まれるすべての機器 p に，ssf に関連する機器フィーチャ cir.feature が実装されていることが必要なケースを表している．

一方で，サブシステム ss に含まれる機器が，1 つでも cir.feature が実装されていればサブシステム ssf を実現できる不変式は，上記とほぼ同様の不変式となり，quantify を EXIST にして，機器 p にかかわる forAll を exists にする．

これらの式の充足可能性判定を行うことにより，サブシステム候補を満たす機器の選択を行う．また，これらの式が満たされない場合，そのサブシステム候補を実現する機器の組合せが存在しないことを意味する．

A.1.2.3 テスト項目とサブシステム候補の充足可能性判定

テスト項目を表す TestCase クラスは，サブシステムに対応するテスト項目情報 TestCaseSubSystem クラスのインスタンスを複数保持する．この TestCaseSubSystem ご

とに，サブシステム候補との充足可能性判定を行い，そのテスト項目がどのサブシステム候補で実現できるかを検証する．

まず，以下の不変式によって，サブシステムフィーチャによるテスト項目とサブシステム候補の充足判定を行う．

```
Context TestCase inv:
  self.subSystems ->forAll( tcss |
  tcss.candidateSubSystem -> select(valid = true)
                          -> forAll( ss |
  ss.impls-> select( status = IMPLEMENTED)
        -> collect( subSystemFeature)
        -> includesAll(
             tcss.impls->select(status=IMPLEMENTED)
                  ->collect(feature) ) ) )
```

次に，テスト項目における機器フィーチャの情報とサブシステム候補の充足可能性判定のため，以下の不変式を検証する．

```
Context TestCase inv:
  self.subSystems ->forAll( tcss |
  tcss.candidateSubSystem -> select(valid = true)
                          -> forAll( ss |
  tcss.products ->forAll(tp |
    ss.have->select(have=true)
        ->collect(product)
        ->exists(sp |
    sp.impls-> select(status=IMPLEMENTED)
      -> collect(feature)
      ->includesAll(
    tp.impls-> select(status=IMPLEMENTED)
      -> collect(feature) ) ) ) ) )
```

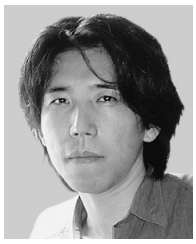
最後に，テスト項目 TestCase において，含むべき機器が直接 TestCaseProduct クラスの関連 product に指定されているケースの不変式を以下に示す．

```
Context TestCase inv:
  self.subSystems -> forAll( tcss |
  tcss.candidateSubSystem -> select(valid = true)
                          -> forAll( ss |
  tcss.products -> forAll(tp |
  ss.have->select( have=true )
        ->collect(product)
        ->includes( tp.product) ) ) )
```

これらの不変式を充足するサブシステム候補がテスト項目を実現可能なものとして識別できる．この識別後に，3.5 節で述べたように貪欲法でサブシステム候補を組み合わせてシステムを構成する．

新原 敦介 （正会員）

2004 年東京工業大学工学部情報工学科卒業．2006 年同大学大学院情報理工学研究科計算工学専攻修士課程修了．同年株式会社日立製作所中央研究所入社．現在，同社横浜研究所研究員．ソフトウェア開発技術の研究と製品開発適用に従事．



小川 秀人 （正会員）

1994 年名古屋大学工学部情報工学科卒業．1996 年同大学大学院工学研究科博士前期課程修了．同年株式会社日立製作所入社．同社システム開発研究所，中央研究所を経て，現在，横浜研究所主任研究員．入社後一貫して，ソフトウェア開発技術の研究と製品開発適用に従事．特に，ソフトウェアテスティング，形式手法，開発プロセスに興味を持つ．

鷲崎 弘宜 （正会員）

早稲田大学グローバルソフトウェアエンジニアリング研究所所長，同大学基幹理工学部情報理工学科准教授，国立情報学研究所客員准教授．1999 年早稲田大学理工学部情報学科卒業，2001 年同大学大学院理工学研究科修士課程修了，2003 年博士課程修了，博士（情報科学）．設計，再利用，品質保証，マネジメントを中心としたソフトウェアエンジニアリングの研究，教育，社会展開に従事．対外活動に SEMAT Japan Chapter Chair, IEEE Computer Society Japan Chapter Secretary, ISO/IEC/JTC1/SC7/WG20 国内委員会主査，日本科学技術連盟研究会副委員長，日本ソフトウェア科学会論文誌編集委員ほか．