論文 / 著書情報
Article / Book Information

題目(和文)	プロダクトライン開発における可変性モデル化手法とシステム構成導出への応用の研究
Title(English)	
著者(和文)	新原敦介
Author(English)	Daisuke Shimbara
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第11910号, 授与年月日:2021年3月26日, 学位の種別:課程博士, 審査員:佐伯 元司,権藤 克彦,渡部 卓雄,西崎 真也,小林 隆志,林 晋平
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11910号, Conferred date:2021/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Tokyo Institute of Technology
Department of Computer Science

令和 2 年度 学位論文

プロダクトライン開発における
可変性モデル化手法と
システム構成導出への応用の研究

東京工業大学
大学院情報理工学研究科 計算工学専攻

新　原　　敦　介

令和 3 年 2 月

概要

本研究では，単独の機器で機能を提供するのではなく，複数のサブシステムが接続され連携して機能を提供する複合システムを対象とした可変性モデルの提案を行い，また，それを用いたテストケースを網羅するシステム構成導出手法の提案を行った．

複合システムでは，サブシステムが独立して派生開発されているため，各々の可変性モデルを持つ．そして，これらのサブシステムが組み合わさり機能を提供する複合システムには，2 つの異なるレベルの可変性が生じるため，1 つの単純な可変性モデルで表現が困難という課題が生じる．また，複合システムには，複数の種類のサブシステムが複数台接続されるため，サブシステムを横断する制約や要求において，台数の可変性を示すインスタンス多重度に関する表現が曖昧となる課題がある．そこで，本研究では，複合システムの可変性に関するモデルリング記法として，異なる概念を分離して表現するために，システム可変性モデルと構造可変性モデルを定義し，関係記述によってつなげる記法の提案を行った．関係記述においては，複合システムの制約や要求に対して生じるインスタンス多重度を識別し，適切な集約演算を行うためのピンサームーブメントアプローチを提案した．この可変性モデリング手法を，モデリングツール CT-CVL として実装した．

提案した可変性モデリング手法に関して，多くのサブシステムで構成される業務用空調機にて例証した．また，複雑な可変性記述の例として，クラウド事業者の 4 つの事例を記述することで，提案する可変性モデルの記法が既存の可変性モデリング手法と同等の記述能力を持つことを確認した．さらに，別の自作 PC システムの例を用いて，可変性モデルを記述する被験者実験を実施し評価を行った．この実験では，モデリング経験のある参加者が，提案する可変性モデルの記法とピンサームーブメントアプローチを用いて，システムの可変性モデルを正しく記述できることを確認した．この被験者実験において，開発した CT-CVL ツールは，被験者がインスタンス多重度を特定するのに貢献した．

さらに，複合システムのシステムテストにおいて，長期にわたる派生開発によって蓄積されたテストケースを実施するためには，テストケースごとにどのシステム構成でテストすべきかを導出することが困難という課題がある．そして，テストを実行する際に，システム構成の切り替えは，機器の移動や接続の切り替えや初期設定のやり直しなどによって，非常に工数が高い．そのため，複数のテストケースを実施する際に，システム構成の切り替え回数を極力少なくしたいという要求がある．そこで，本研究では，提案した複合システム向けの可変性モデルを用いて，テストケースごとに実行可能なシステム構成を導出し，テストケース全てを網羅する少ない数のシステム構成の組を導出する手法を提案した．また，モデリングツール CT-CVL にシステム構成導出機能を実装した．

提案したテストケースを網羅するシステム構成導出手法を評価するために，POS レジシステムの例を用いて被験者実験を行った．その結果，提案手法を用いない方法では，テストケースを網羅するシステム構成を導出できなかったのに対し，提案手法を用いればテストケースを網羅する少ない数のシステム構成が導出できることを確認できた．

Abstract

Modern systems contain parts that are themselves systems. Such complex systems thus have sets of subsystems that have their own variability. These subsystems contribute to the functionality of a composite system. Such composite systems have a very high degree of variability. Therefore, a modeling technique for variability of an entire composite system is required to express two different levels of variability: variability of the composite system as a whole and variability of subsystems. If these levels are described together, the model becomes hard to understand. When the variability model of the composite system is described separately, each variability model is represented by a tree structure and these models are combined in a further tree structure. For each node in a variability model, a quantity is assigned to express the multiplicity of its instances per one instance of its parent node. Quantities of the whole system may refer to the number of subsystem instances in the system. From the viewpoint of the entire system, constraints and requirements written in natural language are often ambiguous regarding the quantities of subsystems. Such ambiguous constraints and requirements may lead to misunderstandings or conflicts in a configuration of the composite system.

A separate notion is proposed for variability of a composite system: one model considers the composite system as an undivided entity, while the other considers it as a combination of subsystems. Moreover, a domain-specific notation is proposed to express relationships among the variability properties of systems, to solve the ambiguity of quantities and establish the total validity. This notation adapts an approach, named Pincer Movement, which can then be used to automatically deduce the quantities for the constraints and requirements. As an application of the proposed notation, a method to derive system configurations covering all test cases is developed for testing by relating the variability model obtained by the proposed notation.

The descriptive capability of the proposed notation was validated with four examples of cloud providers. In addition, the proposed method and description tool were validated through a simple experiment on describing variability models with real practitioners.

目次

図目次

表目次

第 1 章

はじめに

1.1 本研究の背景

近年のソフトウェア開発において，派生開発が多くを占めるようになった [2]．派生開発とは，新規開発の対となる概念であり，ベースとなる既存ソフトウェアに変更・追加・削除を行い新たなソフトウェアを開発するものである．ソフトウェアの派生開発において，体系的に再利用を行うソフトウェアプロダクトラインエンジニアリング (Software Product Line Engineering (SPLE)) [3] [4] がある．SPLE は，派生開発によって生じる類似ソフトウェア群を製品系列として捉え，それら全体の開発においてコストを低減し品質の向上を目指す技術とプロセスを含んだ概念である．SPLE では，再利用を前提とした共通部品や可変部品を開発しコア資産として蓄積する．そして，それらを組み合わせて個別のソフトウェアを構成する．そのため，ソフトウェアのどの機能が共通部でどの機能が可変部となるかを分析する可変性分析が重要である．

ここで，近年のシステムは，単独の機器で機能を提供するのではなく，複数の機器 (サブシステム) 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．複合システムでは，サブシステムが独立して派生開発されており，サブシステムごとに可変性分析が行われる．そして，複合システム全体では，サブシステムが提供する個別の機能を組み合わせて，1 つのシステムとして機能を提供する．複合システムでは，それを構成するサブシステムを置き換えられるという可変性がある．そして，サブシステムが置き換わることで，複合システム全体としての提供機能に可変性が生じる．そのため，複合システムの可変性には，サブシステムごとの可変性，複合システムを構成するサブシステムの組み合わせに関する可変性，複合システム全体としての可変性の 3 種類が生じる．複合システム全体の効率的な再利用開発には，この 3 種類の可変性を適切に分析して表現する方法が必要となる．

さらに，複合システム全体の可変性において，サブシステムの台数 (インスタンス数) を適切に識別し，取り扱う必要がある．例えば，1 つの複合システムにおいて，複数台のサブシステムが結合してシステムを構成し，複合システム上のある機能を実現するのに，サブシステムが当該機能を提供する．この時，すべてのサブシステムがその機能を提供する必要があるのか，もしくは，少なくとも 1 台のサブシステムがその機能を提供すればよいのかという曖昧さが生じる．複合システムにおいて，このような曖昧さがなくなるように可変性を表現する必要がある．

複合システムのシステムテストでは，実施するテストケースごとに，テスト対象機能がシステム上に実現するようにサブシステムの構成を選択し，システムを構築してテストを実行する．そのため，テストケースごとに，どのシステム構成でテストができるのかを導出する必要がある．そして，複合システムのテストケースは，派生開発を続けたことで大量に蓄積されており，派生開発が行われるごとに後方互換性などを検証するために，過去テストケースの多くを実施しなければならない．テストを実行する際に生じるシステム構成の切り替えは，機器の搬出・搬入やケーブルでの接続，そして初期設定と機器の立ち上げ作業などが生じ，テスト作業そのものより工数が大きい．そのため，複数のテストケースを実施する際に，システム構成

の切り替え回数を極力少なくしたいという要求がある．

筆者が業務で経験した複合システムの開発では，このテストケースを実行するためのシステム構成の導出は，有識者による暗黙知を用いての経験的な手法が用いられていた．サブシステム間の排他関係や，サブシステムの組み合わせによって複合システムに表出する機能に関して，暗黙知となっており，体系的な導出手法は整備されていなかった．ヒアリングした範囲で，この複合システムのシステム構成導出には，有識者であっても多くの時間がかかり大きな問題となっていた．

1.2　本研究の目的

本研究では，複合システムの可変性モデリング手法とそのモデリング手法を応用したシステム構成の導出手法を提案する．複合システム全体では，3 種類の可変性が生じるため，それぞれを分けて記載し関係記述でつなぐモデリング手法を提案する．また，複合システムの可変性モデリングにおいて，システム内に生じるインスタンス数の可変性を適切に識別して取り扱う手法も提案する．そして，そのモデリング手法で得られた可変性モデルと派生開発によって蓄積されたテストケースを関連付けることで，テストケース向けのシステム構成を導出する手法を提案する．システム構成の切り替えコストを下げるために，より少ない数のシステム構成で全てのテストケースを網羅するような，システム構成の組を導出する．提案するシステム構成導出手法では，可変性モデル上に有識者の暗黙知を集約し明示化を行う．また，テストケースの関連付けを行った後は，自動的にシステム構成の組を得られる手法となるため，工数を削減できる．

1.3　本論文の構成

本論文の構成を以下に示す．本論文は全 6 章から成る．2 章以降の概要は以下の通りである．**「第 2 章 背景」**にて，ソフトウェアプロダクトラインの可変性モデルについて述べ，本研究で扱う複合システムにおける可変性モデルの課題と，システムテストにおけるシステム構成導出の課題について述べる．**「第 3 章 関連研究」**では，複合システムにおける可変性モデルやシステム構成導出に関連する研究を紹介する．**「第 4 章 複合システムの可変性モデルとピンサームーブメントアプローチ」**において，複合システム向けの可変性モデルとして，システム可変性モデルと構造可変性モデルに分割する記法とそれらの間をつなぐ関係記述を提案する．また，関係記述において，可変性モデル内に生じるインスタンス数を適切に識別して取り扱うためのピンサームーブメントアプローチを提案する．そして，これらの提案に関して被験者実験を用いた評価を行う．**「第 5 章 複合システムのシステム構成の導出」**では，4 章にて提案した可変性モデルの記法を用いて，テストケースごとにテスト実施可能なシステム構成導出方法を提案し，さらに複数のテストケースを網羅するように，より少ない数のシステム構成の組を導出する手法を提案する．そして，これらの提案に関して被験者実験を用いた評価を行う．最後に，**「第 6 章 おわりに」**で本論文をまとめる．

第 2 章

背景

本章では，本研究の背景について述べる．まず初めに，ソフトウェアプロダクトラインの概要について述べ，可変性分析・モデリング手法とその利用方法について述べる．次に，本研究で対象とする複合システムに関して述べる．最後に，本研究の課題として，複合システムにおける可変性モデリング及びシステムテストに関しての問題点を整理する．

2.1 ソフトウェアプロダクトラインエンジニアリングと可変性モデル

本節では，まず，ソフトウェアプロダクトラインエンジニアリングの概要について述べる．次に，可変性分析における代表的なモデリング手法について紹介する．最後に，可変性モデルの基本的な利用方法について述べる．

2.1.1 ソフトウェアプロダクトラインエンジニアリング

近年のソフトウェア開発において，派生開発が多くを占めるようになった．派生開発とは，新規開発の対となる概念であり，ベースとなる既存ソフトウェアに対して変更・追加・削除を行い新たなソフトウェアを開発するものである．派生開発は，過去製品のソフトウェアに機能を追加して新製品を開発する場合や，同時に高機能版・廉価版といった複数種類の製品を展開する場合などに行われる．新規開発に対して，差分のみの開発で済むため，開発量は少なくなる．独立行政法人情報処理推進機構が提供する組込みソフトウェア開発データ白書 2019 [2] によると，開発プロジェクトの 93% が改良 (派生) 開発であった．

派生開発は開発量の削減に効果的である一方で，不用意に派生開発を重ねると技術的負債を重ねることになる．派生開発を重ね巨大なソフトウェアとなった際に，全体を把握することが困難となり，アーキテクチャを無視した局所的な変更を行う場合がある．そのような変更が重なってしまった場合，大局的なアーキテクチャが破壊され，修正を行うことが困難となり，保守性が低下する．結果として，派生開発によって開発量を削減したつもりが，不具合修正などの工数が肥大化し，派生開発のメリットを享受できなくなる可能性がある．

ソフトウェアの派生開発において，体系的に再利用を行うソフトウェアプロダクトラインエンジニアリング (Software Product Line Engineering (SPLE)) [3, 4] がある．SPLE は，派生開発によって生じる類似ソフトウェア群を製品系列として捉え，それら全体の開発においてコストを低減し品質の向上を目指す技術とプロセスを含んだ方法論である．SPLE において，ソフトウェア開発は，ドメイン開発とアプリケーション開発に分けられる．ドメイン開発では，類似ソフトウェア群が属するドメインにおいて，再利用を前提とした共通部品や可変部品を開発しコア資産として蓄積する．アプリケーション開発では，コア資産である共通部品や可変部品を組み合わせ，個別のソフトウェアを構成する．そのため，ソフトウェアのどの機能が共通

部でどの機能が可変部となるかを分析することが需要となる．SPLE では，ソフトウェアの差異を整理し表現する分析手法として可変性分析 (Variability Analysis) を用いる．可変性分析は，Feature-oriented domain analysis(FODA) [5] で提唱され，それ以外でも様々な手法が提案されている．可変性分析を行った結果は，可変性モデルと呼ばれる．

2.1.2 可変性モデル

本論文では，可変性と可変性モデルを以下のように定義する．

Definition 1（可変性） 製品系列において，製品ごとに可変である性質．

Definition 2（可変性モデル） ソフトウェアのどの機能や特徴において，どの部分がどのように可変性を持つのか，そしてどの部分に可変性がないのかを，特定の記法に則り表現したもの．

本節では，代表的な可変性モデルの表現に関して，フィーチャモデル (Feature model) [5]，Product Map [6]，Orthogonal Variability Model(OVM) [4]，Common Variability Language(CVL) [7] を紹介する．

フィーチャモデルは，可変性分析を提唱した FODA [5] にて提案された表現方法である．フィーチャモデルでは，フィーチャ (Feature) と呼ばれる分析対象の機能や特徴を表すノードを木構造に整理する．可変性分析を行う対象全体を表すルートノードを最上位に置き，上位のノードを下位のノードに分解する形で可変性を表現する．木構造で分解していく点で，ゴール指向要求分析手法 [8] と類似している．図 2.1 に，フィーチャモデルの例を示す．フィーチャモデルにおいて，その分解の種類は，以下の 4 種類となる．

> **必須 (Mandatory) 分解** 上位のノードの実現に対して下位のノードの実現が必須である．下位ノードの上位との接続箇所に黒丸で表現される．
> **選択可能 (Optional) 分解** 上位ノードの実現に対して下位ノードの実現は影響しないため，選択可能である．下位ノードの上位との接続箇所に白抜きの丸で表現される．
> **オルタナティブ (Alternative) 分解** 上位ノードの実現のためには，下位のいずれか 1 つのノードが実現される必要がある．上位ノードから下位ノードへ分解されるリンク群の間に黒塗りの円弧で表現される．
> **オア (Or) 分解** 上位ノードの実現のためには，下位のいずれか 1 つ以上のノードが実現される必要がある上位ノードから下位ノードへ分解されるリンク群の間に白抜きの円弧で表現される．

フィーチャモデルでは，分析対象全体を表すルートノードが常に実現されることを保ちながら，4 つの分解を用いて，下位のノードへと詳細化する．フィーチャモデルでは，4 種の分解の他，ノード間の依存関係 (implies) や排他関係を表現できる．この図 2.1 の例は，スーパーマーケットなどにあるレジスターの可変性モデルを表現している．まずルートノードに「Cash Register」があり，これは「支払い (Payment)」と「通貨 (Currency)」にそれぞれ必須分解されている．支払いに関しては，「現金 (Cash)」の取り扱いが必須で分解されており，現金での支払い対応は必須と分かる．「クレジットカード (Credit card)」と「バンクカード (Bank card)」の支払いに関してはオア分解されており，1 つ以上選択可能である．一方の通貨では，1 つのレジスターでは 1 種類の通貨しか扱えないため，「日本円 (JPY)」か「ノルウェークローネ (NOK)」のどちらかを選択するオルタナティブ分解となっている．このようにして，分析対象のソフトウェアの可変性を木構造で表現するのがフィーチャモデルである．フィーチャモデルは，様々な可変性表現の基となっており，Chen らがまとめた可変性モデルのサーベイ論文 [9] では，様々な可変性モデルがこのフィーチャモデルのコンセプトを継承していることを示している．

次に，別の可変性表現としてプロダクトマップ [6] がある．プロダクトマップは，表に基づく可変性表現

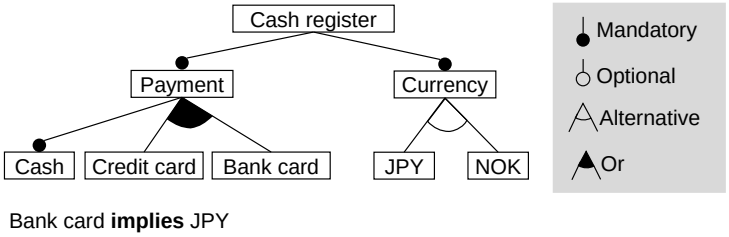

図 2.1: フィーチャモデルの例

表 2.1: プロダクトマップの例

Product	Payment			Currency	
	Cash	Credit card	Bank card	JPY	NOK
product A	✓		✓	✓	
product B	✓	✓		✓	
product C	✓	✓	✓		✓

✓: Selected

であり，列方向に選択可能なフィーチャを並べ，行方向に製品系列に属する製品を並べる．そして製品ごとに実現するフィーチャを交差するセルに記載する．プロダクトマップの例を，表 2.1 に示す．表 2.1 では，「productA」は，支払い機能として「現金 (Cash)」と「銀行カード (Bank card)」に対応し，「日本円 (JPY)」を扱う製品であることを示している．この表では，クレジットカードと銀行カードの併用が可能であることや，通貨の日本円とノルウェークローネが排他であることなどの，フィーチャの間の関係は表現できない．プロダクトマップは，製品ごとにどのフィーチャを実現しているかのみを示すため，製品カタログに頻繁に用いられており，製品群の比較に優れている．

次に紹介する可変性表現は OVM [4] は，複数の可変性を独立して表現する点が特徴である．OVM では，可変する箇所を可変点 Variation Point(VP) として白抜きの三角形の上部に黒塗りの三角形を重ねた形状で表す．そして，可変点にぶら下がる形で選択する対象である可変部 Variant(V) を白抜きの四角形とその左上に黒塗りの四角形を重ねた形で表す．可変点には，必須可変点 (Mandatory VP) と選択可能な可変点 (Optional VP) があり，前者は必ず生じる可変点，後者は場合によっては生じない可変点である．可変点にぶら下がる可変部は，必須 (Mandatory) 分解と選択可能 (Optional) 分解 と選択数指定 (Alternative Choice) 分解がある．この必須分解と選択可能分解に関しては，フィーチャモデルの分解と同じである．選択数指定は，弧で示された可変部の中から，最小数から最大数の間で選択できるという可変性を表現している．最小数 1・最大数 1 で示された選択数指定分解は，フィーチャモデルのオルタナティブ分解と同じ意味である．また，VP や V の間の関係性として要求 (Requires) や排他 (Excludes) を表現できる．OVM の例を，図 2.2 に示す．支払い機能を選択する VP「Payment」において，可変部が 3 つぶら下がっている．「現金 (Cash)」は必須であり，「クレジットカード (Credit card)」と「銀行カード (Bank card)」による支払い機能は，少なくとも 1 つを実現する必要がある．通貨の VP「Currency」においては，「日本円 (JPN)」と「ノルウェークローネ (NOK)」の可変部のうち，どちらか片方を選ぶ形で可変である．図 2.2 は，意味的に図 2.1 と同等である．OVM では，可変点 (VP) が独立してそれぞれに存在し，その間の関係を表現する．

最後に，ドメインに非依存で可変性の表現と可変性からの選択結果の表現を定義した CVL [10] について述べる．CVL は，Haugen らが提案した可変性とその選択結果を表現するモデルであり，フィーチャモデルと同様に一つの木構造で分析対象全体の可変性を表現する．CVL では，可変性モデルの木構造のことを

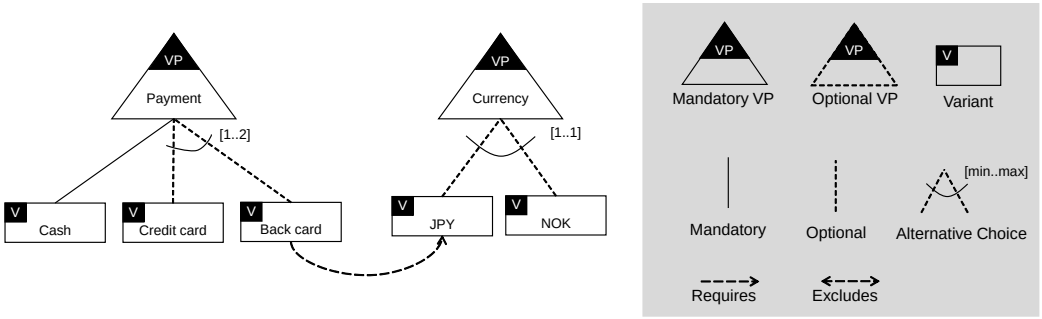

図 2.2: OVM の例

VSpec model と呼び，木構造を構成するノードを VSpec と呼ぶ．CVL には，可変性を示す VSpec model に対応して，可変箇所を具体的に選択した結果を表現するリゾリューションモデル (Resolution model) もある (Definition 7 参照)．VSpec ノードには，Choice, VClassifier, Variable, CVSpec の 4 種類がある．Choice ノードは，フィーチャモデルにおけるフィーチャに該当し，角を丸くした四角形で表す．Choice ノードは実線や点線で分解構造を示しており，実線は必須分解，点線は選択可能分解を表す．また，VSpec ノードの分解には，OVM における選択数指定分解 (Alternative Choice) と同じ意味の，下位のノードからいくつ選択可能かを示すグループ多重度 (Group multiplicity) 分解という方法がある．

Definition 3（グループ多重度） 可変性モデルにおいて，上位のノード 1 つに対して，そのノードからぶら下がるノードの許容される選択数の下限値と上限値で表される量的な可変性．

グループ多重度は，CVL では VSpec ノードの下部に三角形を付け，隣に下限数・上限数を付記することで表現する．VClassifier ノードは，そのノードより下位のノード群がインスタンスをいくつ作成可能かという可変性を示し，四角形で表現される．VClassifier ノードの四角形の中に，許容するインスタンス数の範囲として下限数・上限数を指定できる．Variable ノードは，変数の可変性を表し楕円で表現する．VSpec ノードによる木構造の分解関係以外の関係として，平行四辺形で記載する制約 (Constraint) がある．

本論文では，可変性モデルに関して，インスタンス数に関する可変性表現を持ち，可変性の選択結果の表現モデルもあるため，CVL を用いる．本論文において，インスタンス数に関する可変性を，インスタンス多重度 (Instance multiplicity) と呼び，以下のように定義する．

Definition 4（インスタンス多重度） 可変性モデルにおいて，その一部にインスタンスの多重性があり，許容されるインスタンス数の下限値と上限値で表される量的な可変性．

CVL において，インスタンス多重度は VClassifier ノードで表され，上位のノードの 1 インスタンスに対して，サブツリー構造の許容されるインスタンス数の下限値と上限値が記載される．

図 2.3 に，CVL による可変性モデルの例を示す．図 2.3 も，意味的に図 2.1 と類似している．図 2.1 では，「支払い方法 (Payment)」ノードの下に，必須分解である「現金 (Cash)」とオルタナティブ分解の「クレジットカード (Credit card)」と「銀行カード (Bank card)」がぶら下がっていた．一方の，図 2.3 では，「支払い方法 (Payment)」ノードの下に，Choice ノード「現金 (Cash)」と Choice ノード「カード (Card)」がぶら下がり，その「カード (Card)」の下にグループ多重度分解で「クレジットカード (Credit card)」と「銀行カード (Bank card)」がある．つまり，フィーチャモデルでは，1 つのノードに複数の分解方法を混在して表現できる一方で，CVL では各ノードの分解方法は 1 つに限定されており，支払い方法 (Payment) ノードの下にカード (Card) ノードを挿入して表現している．図 2.1 との差異として，ルートノードから必須分解で VClassifier ノード「バーコードリーダー (Barcode reader)」がある．このノードは，レジスターにおけるバーコードリー

ダーのインスタンス多重度を示しており，バーコードリーダーの搭載数が 1 つもしくは 2 つであることを意味する．そしてそれぞれのバーコードリーダーについては，「ハンディタイプ (Handy)」と「埋め込みタイプ (Embedded)」が選択可能であり，グループ多重度分解で示されている．図 2.3 の例では，Variable や CVSpec ノードは現れていないが，本論文の以降で用いる．

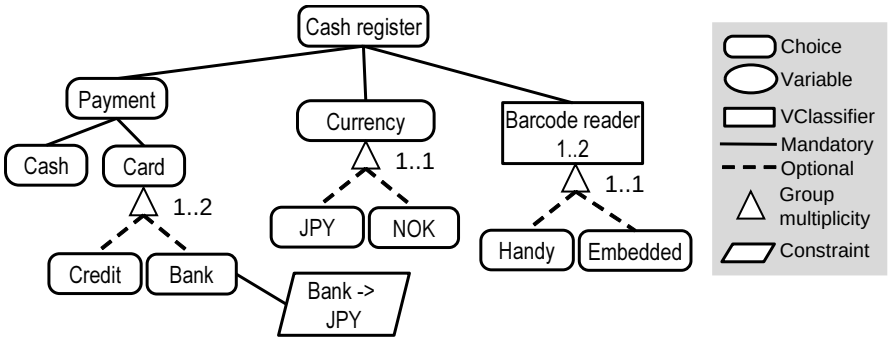

図 2.3: CVL の例

2.1.3 可変性の利用方法

本節では，基本的な可変性モデルの利用方法について述べる．可変性モデルは，ソフトウェアの特徴を示すフィーチャがどのように可変であるかを表現しており，他のソフトウェア開発資産と連動することで価値を提供する．可変性が他のソフトウェア開発資産と連動する関係について，Antuetil ら [1] は，図 2.4 のような 2 プロセス× 2 スペースの 4 象限で表現している．ドメイン開発 (Domain engineering) とアプリケーション開発 (Application engineering) のプロセスは，2.1.1 節で述べた通り，製品ドメイン全体の開発と個別製品の開発を表している．プロブレムスペース (Problem space) とソリューションスペース (Solution space) は，可変性を扱う空間と，可変性と連動する他のソフトウェア開発資産を扱う空間を表している．

図 2.4 では，可変性モデルとソースコードの連携の例を示している．ドメイン開発のプロブレムスペースにおいて，可変性分析を行った結果のフィーチャツリーが可変性モデル (Variability model) として示されている．また，ドメイン開発のソリューションスペースでは，可変性モデルのフィーチャに対応づく形で，コードの変換ルール (Transformation rules) やベースとなるコード (Code) が定義される．アプリケーション開発に移りプロブレムスペースでは，ドメイン開発で分析した可変性モデルから製品として実現すべきフィーチャを決定し，可変性が定まったモデル (Feature selection) を作成する．その結果，決定したフィーチャに対応する変換ルールに基づいて，ソリューションスペースにおいて製品のコードが生成される．この例において，可変性モデルを作成し，そのモデルと可変箇所を内包したソースコードを関連付けておくことで，可変性モデルから具体的な製品に向けてフィーチャを束縛することで，製品向けのコードを生成することができる．

本論文では，可変性モデルの可変部の選択操作を束縛 (Bind) と呼ぶ．

Definition 5（束縛 (Bind)） 可変性モデルの中に生じる可変箇所において，可変である選択肢の中から選択するものと選択しないものを決定すること．

可変性モデルは，生じうるバリエーションを包括的に定義するために，採否を選べる選択可能分解や，子要素から複数を選ぶ分解を記述する．そして，システムの開発が進むにつれ，生じていた可変性は意思決定されて束縛される．束縛する操作を，Czarnecki ら [11] は「Specialization」と呼んでいる．

可変性モデルの束縛において，可変性モデルの中に生じる可変性の一部のみを束縛し，他の可変性が残っているモデルがある．その場合の残ったモデルのことを束縛可変性モデルと呼ぶ．

Definition 6（束縛可変性モデル） 可変性モデルにおいて，可変性がある箇所の一部が束縛されているが，他に可変性のある箇所が残っている可変性モデル．

そして，可変性モデルにおいて，すべての可変性が束縛されたものを，本論文では CVL の名称に合わせてリゾリューションモデルと呼ぶ．これは，Antuetil ら [1] が Feature selection と称しているものと同義である．

Definition 7（リゾリューション (Resolution) モデル） 可変性モデルもしくは束縛可変性モデルにおいて，可変性のある箇所の全てが束縛され，他に可変性が残っていない状態のモデル．

ここで，アプリケーション開発におけるプロブレムスペースには，リゾリューションモデルが記載される．リゾリューションモデルは，可変性のコンフィグレーション (Configuration) とも呼ばれる．また，2.1.2 節で述べたプロダクトマップは，このリゾリューションモデルを表現している．

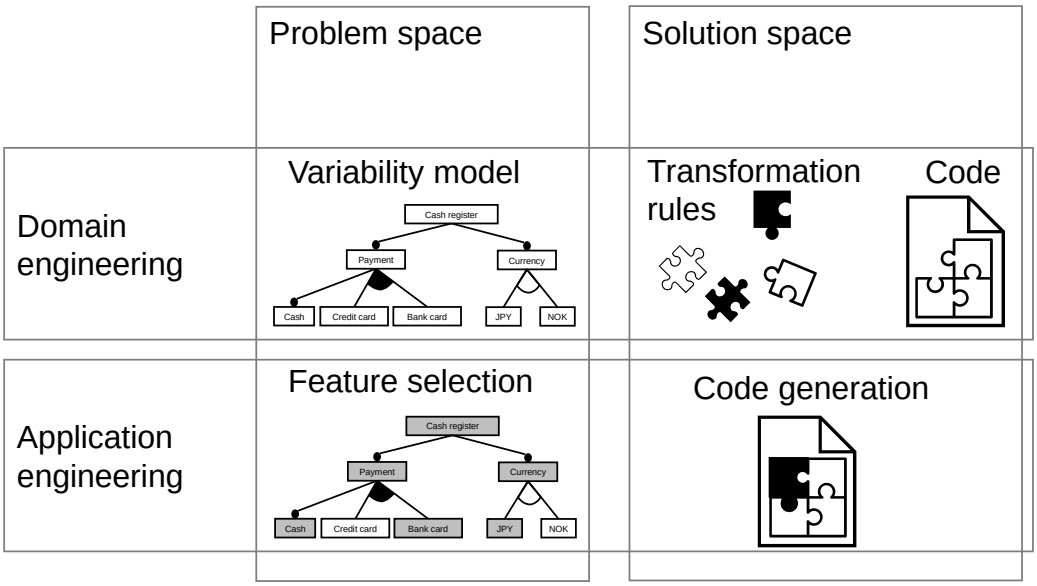

図 2.4: SPL における 2 つのスペースと 2 つの開発 [1]

図 2.4 の例ではソリューションスペースにソースコードを用いていたが，これは，他の実装資産や，要求や設計に関する文書資産にも適用可能である．ソフトウェアの設計資産と可変性の連動に関して，設計文書としてアクティビティ図を生成する手法 [12] がある．この手法では，ドメイン開発におけるソリューションスペースにて，可変箇所を色分けして記載したアクティビティ図テンプレートを定義し，可変性モデルを束縛することでアクティビティ図のインスタンスが得られる．最も有名な可変性分析ツールの一つである pure systems 社の pure::variant も，この 4 象限をツールチュートリアル [13] にて参照し，ソリューションスペースにおいて要件管理ツール DOORS [14] などの外部ツールとの連動を謳っている [15]．このように，可変性モデルは様々な開発資産と連動して用いられるため，どのような開発資産と連動し何を目的とするかで，可変性モデルの表現方法は変わってくる．

2.2 複合システム

本節では，本論文で対象とする複合システムについて述べ，複合システムの例として架空の業務用空調機を紹介する．その後，複合システムにおけるシステムテストの特徴について述べる．

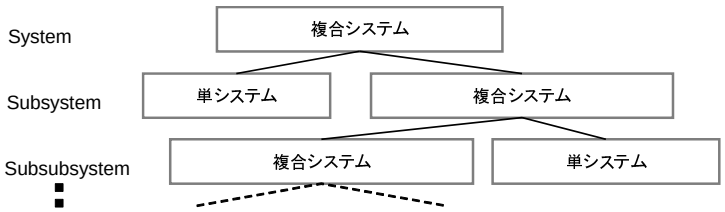

図 2.5: 複合システムの多層構造

2.2.1 複合システムの定義

近年のシステムは，単独の機器で機能を提供するのではなく，複数の機器 (サブシステム) 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．本論文では，複合システムを以下のように定義する．

Definition 8（複合システム） サブシステムが独立してプロダクトライン開発され，それらが組み合わさって機能を提供するシステム．

複合システムと類似の概念として，システム・オブ・システムズ (System-of-systems) がある．Maier [16] は，システム・オブ・システムズに関して，以下の 2 点を特徴として述べている．

> **運用面での独立性**：サブシステムそのものも，独立して運用が可能である
> **管理面での独立性**：サブシステムが独立して管理されている

本論文で扱う複合システムでは，サブシステムの管理面での独立性のみを持ち，運用面での独立性を問わない．例えば，自作 PC は，ソケットやバスの規格にのっとり，CPU やメモリ，マザーボードやビデオカードがそれぞれにバリエーションをもって展開されており，ユーザはそれらを組み合わせて PC システムを構築する．この時，ビデオカードをサブシステムとした場合，ビデオカード単独で運用が可能ではないため，自作 PC はシステム・オブ・システムズではない．一方で，ビデオカードやマザーボードは異なる製造元が独立して開発しており，それらが組み合わさって機能を提供するため，複合システムになる．

複合システムでは，サブシステムごとに異なる開発チーム・組織で開発されることがある．自社内で複数のチームを構築し，それぞれのチームごとにサブシステムを開発するケースもあれば，サブシステムの開発を外注や OEM で開発するケースもある．また，サブシステムを社外の既存製品から調達するケースもあり得る．そして，複合システム全体の開発を行う組織は，サブシステムの開発とは分離していることも多い．そのため，複合システム全体のチームはサブシステム開発について知見が少なかったり，その逆にサブシステム開発に係る開発者がシステム全体の視点を持たなかったりすることがある．

複合システムのサブシステムは，さらに下位の機器 (サブサブシステム) で構成されることもある．そのため，図 2.5 に示すように，複合システムは多層構造を取りうる．ここで，システム全体から見て，1 段目の構成要素をサブシステム，サブシステムがさらに分解された構成要素をサブサブシステムとする．また，独立して開発される子要素を持つものを複合システムとし，これ以上独立して開発されないものを単システムとする．

2.2.2 複合システムの例

複合システムの例として，ビル等に導入する業務用空調機がある [17]．業務用空調機では，室外機はビルの屋上などに設置され，冷媒を圧縮して冷媒管に送り出し，室内機は天井に埋め込まれて，届いた圧縮冷媒を用いて冷房や暖房といった空調機能を提供する (図 2.6)．1 つの業務用空調機では，複数台の室外機と複数

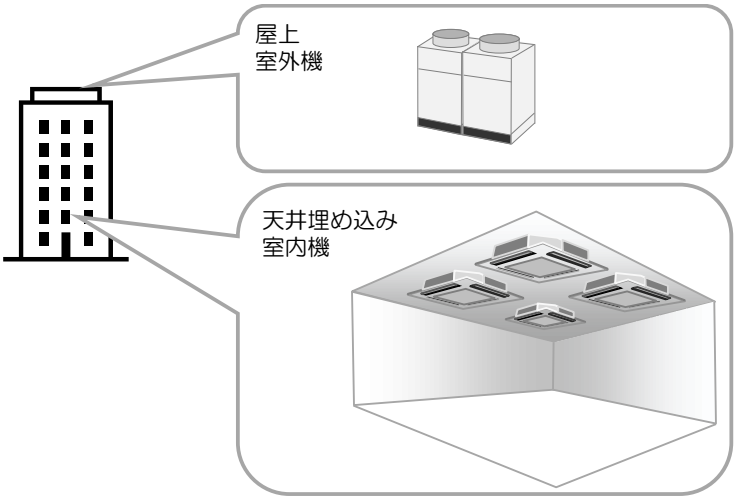

図 2.6: 業務用空調機の例

台の室内機が組み合わさりシステムを構成する．室内機の数は，空調機能を提供する部屋数や広さによって変動し，室外機の数は室内機が使用する冷媒の提供能力に依存する．室外機・室内機のそれぞれで多品種が開発され，顧客の要求に応じて必要な機器を選択し組み合わせてシステムを構築する．業務用空調機は，単独の製造元が開発する，複数種類の機器による複合システムである．

本研究における課題を述べるにあたり，架空の業務用空調機の例を導入する．業務用空調機のサブシステムである室内機・室外機は個別に多品種開発され，各々の可変性を持つ．室外機のサブシステム可変性モデルを図 2.7 に示す．室内機の可変性モデルを図 2.8 に示す．

ここで，サブシステム可変性モデルとは，サブシステムの持つ可変性を表現したモデルを意味する．

Definition 9（サブシステム可変性モデル）　複合システムを構成するサブシステムそれぞれの可変性を表現したモデル．

このサブシステム可変性モデルでは，それぞれのサブシステムのスコープで分析されるため，複合システム全体に関する視点は持たない．この架空の業務用空調機の例では，サブシステムは全て単システムを想定しており，さらなるサブサブシステムを持たない．

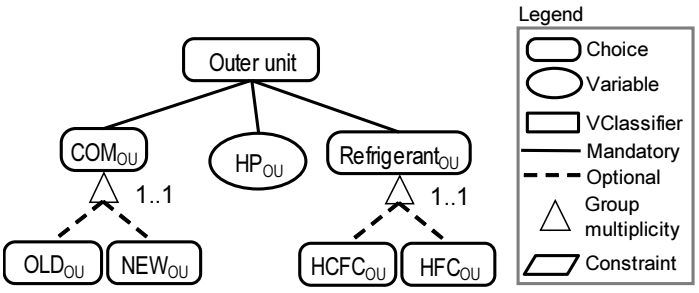

図 2.7: 室外機のサブシステム可変性モデル

図 2.7 に示す室外機のサブシステム可変性モデルでは，室外機には「通信プロトコル (COM)」の可変性があり，そのグループ多重度分解として「OLD$_{OU}$」と「NEW$_{OU}$」がある．これらの Choice ノードは，グループ多重度が「1..1」とあるため，どちらか一方を選択する可変性である．これは室外機と室内機の間の通信が定義された後に新たな機能が追加されたため，既存のプロトコルを旧プロトコルとし，新たに機能追

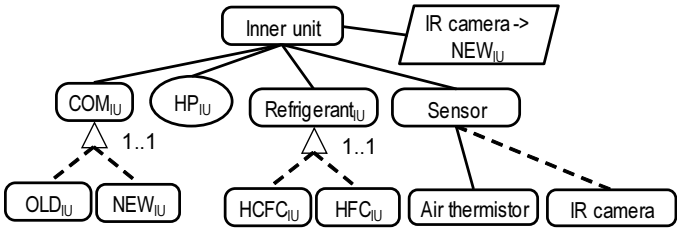

図 2.8: 室内機のサブシステム可変性モデル

加されたものを新プロトコルと定めた結果，室外機の各製品では新プロトコル対応のものと非対応のものが生じたという可変性を表している．下付きで記載の「OU」は，室外機を示す Outer Unit の略記である．「HP」は，室外機が提供する冷媒の処理能力を表す馬力 (Horsepower) のことで，数値の可変性を意味する Variable ノードで表現されている．室外機それぞれに固有の冷媒提供能力値を持つ．また，室外機は使用する冷媒の種類の可変性を持つ．Choice ノード「Refrigerant$_{OU}$」は，「HCFC(hydro-chloro-fluoro-carbon)$_{OU}$」と「HFC(hydro-fluoro-carbon)$_{OU}$」の 2 つの Choice ノードを子要素として持ち，グループ多重度が 1..1 とあるため，どちらか一方を選択する可変性である．「HFC」は，環境性能が高い後発の冷媒となっている．

図 2.8 に示す室内機のサブシステム可変性モデルは，「COM」と「HP」と「Refrigerant」に関して室外機の可変性モデルに類似している．室外機の HP は冷媒提供能力の値であったが，室内機は冷媒を消費する能力値で空調機能の性能を示す値となる．下付きで記載の「IU」は，室内機を示す Inner Unit の略記である．室外機にない可変性として，Choice ノード「Sensor」には，必須分解の Choice ノード「Air thermistor」と選択分解の Choice ノード「IR(infrared) camera」がある．前者は室内温度センサであり，後者は赤外線カメラセンサである．

各サブシステムの中では，ノード間の制約を記述できる．例えば，室内機において，赤外線カメラセンサの情報をやり取りするためには，通信は新プロトコルに対応する必要があるという制約がある．この制約は，室内機のサブシステム可変性に閉じた制約であり，ほとんどの可変性記述言語でもこのような閉じた制約はサポートしている．本論文で用いる CVL では，平行四辺形の中に Basic constraint Language(BCL) を記載することで，このような制約を表現できる．図 2.8 では，平行四辺形に「IR camera -> NEW$_{IU}$」と書くことでこの制約を表現している．CVL では，制約を BCL だけではなく，Object Constraint Language (OCL) [18] で書くことも可能である．

サブシステム可変性モデルの可変性を束縛していくと，具体的なサブシステム製品になる．例えば，表 2.2 に，室外機のサブシステム可変性モデルを束縛して，得られるリゾリューションモデルを表現する，室外機製品のプロダクトマップを示す．表 2.2 に示す 9 製品それぞれは，自身が実装している通信プロトコル (COM)，使用している冷媒 (Refrigerant)，そして冷媒提供能力値を可変性から選択している．表 2.3 は，同様にして室内機の製品に関するプロダクトマップを示している．サブシステム製品は，サブシステム可変性モデルによって表現された可変性の，生じうるすべての組み合わせを網羅しなくて良い．

業務用空調機を構成するには，表 2.2 と表 2.3 から，製品を選択し接続する必要がある．そして，室外機と室内機の構成は，システム全体の要求を満たす必要がある．例えば，簡単な業務用空調機システムとして，「200HP 相当の空調機能を提供するシステム」を考える．この時, 2 台の「Outer-STD」と 4 台の「Inner-HIGH」を選択し組み合わせることで，提供冷媒能力が 200HP で冷媒消費能力が 200HP となるため，前記の「200HP 相当の空調機能を提供する」というシステム要求を満たした構成となる．このような，サブシステムのリゾリューションモデルである製品を組み合わせた複合システムの具体的な構成のことをシステム構成と呼ぶ．

Definition 10（システム構成 (System configuration)） 複合システムにおけるサブシステムのリゾリューションモデルを選択し，複合システムとして組み合わせる構成のこと．

表 2.2: 室外機のプロダクトマップ

Product name	COM		HP	Refrigerant	
	OLD	NEW		HCFC	HFC
Outer-LOW	X		70	X	
Outer-STD	X		100	X	
Outer-HIGH	X		150	X	
OuterEX-STD		X	100	X	
OuterEX-HIGH		X	150	X	
Outer-STD-Eco	X		100		X
Outer-HIGH-Eco	X		150		X
OuterEX-STD-Eco		X	100		X
OuterEX-HIGH-Eco		X	150		X

表 2.3: 室内機のプロダクトマップ

Product name	COM		HP	Refrigerant		IR camera
	OLD	NEW		HCFC	HFC	
Inner-LOW	X		30	X		
Inner-HIGH	X		50	X		
InnerEX-LOW		X	40	X		
InnerEX-HIGH		X	80	X		X
Inner-HIGH-Eco	X		50		X	
InnerEX-LOW-Eco		X	40		X	X
InnerEX-HIGH-Eco		X	80		X	X

全ての状況に対応可能な業務用空調機のシステム構成というものは存在しないため，製造業者は，様々な状況に応じてシステム構成をカスタマイズして販売する．製造業者は，市場の観点から生じうる要求を整理し，状況に応じて要求から取捨選択を行い，採択された要求を満たす最適なシステム構成を導出する．例えば，この架空の業務用空調機では，次の 2 つの要求がある．

Requirement 1 フロンガス規制に準拠する．
Requirement 2 空調システムは，室内の人間を検知し，直接風が当たらないようにする．

要求定義プロセスにおいて，要求は状況によって取捨選択される．そのため，上記の 2 つの要求は常に満たされるわけではない．一方で，システム全体として常に満たし続けねばならない必須の制約というものも存在する．架空の業務用空調機の例では，以下の 3 つの制約がある．

Constraint 1 室内機と室外機は同じ冷媒を利用する必要がある．
Constraint 2 室内機と室外機は，相互に通信できる必要がある．
Constraint 3 冷媒消費能力の値 (HP) は，冷媒提供能力の値 (HP) を超えてはならない．

Constraint 2 に関して補足すると，室内機と室外機はそれぞれに，「OLD」と「NEW」が選択肢となっている通信プロトコルに関する可変性を持つ．この時，それぞれ同じ通信プロトコルを選択していれば，室内機と

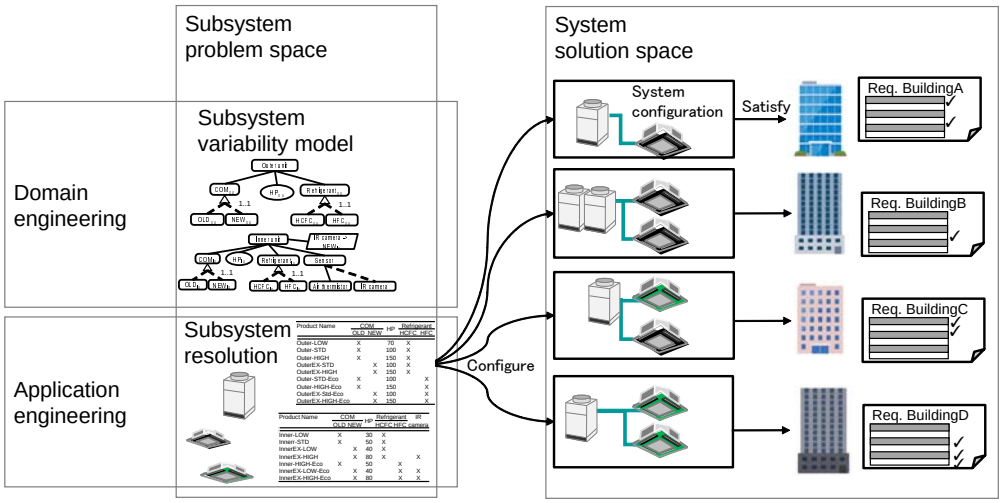

図 2.9: 複合システム開発

室外機は通信可能となる．そして，室外機が「NEW」の新通信プロトコルを実現している時，室外機は通信において後方互換性を有し，「OLD」の旧通信プロトコルを持つ室内機と通信が可能である．しかし，室内機が「NEW」の新通信プロトコルを実現している時は，後方互換性はなく，「NEW」の新通信プロトコルを持つ室外機としか通信できない．

2.2.3　複合システム開発

本節では，複合システム開発について述べる．ここで，複合システム開発は，個々の複合システムのみの開発という意味ではなく，各サブシステムの派生開発や，それらを様々に組み合わせた複合システム全体の開発を示す．図 2.9 に，複合システム開発の概要を示す．図 2.9 の左側は，サブシステムの可変性モデルと具体化された製品を示すリゾリューションモデル (プロダクトマップ表現) が示されている．2.2.2 節で示した業務用空調機のサブシステム可変性モデル (図 2.7, 図 2.8) が上部のドメイン開発において定義され，それらが束縛された形で，下部のアプリケーション開発では，各サブシステムのリゾリューションである製品群 (表 2.2, 表 2.3) がある．これらのサブシステムはシステムと独立して開発され，それぞれの可変性分析が行われている．一方の右側では，複合システムの開発形態を示している．複合システムの開発では，複合システムの要求を満たすようにサブシステムの製品を選択して構成する．つまり，業務用空調機を納品する先のビルごとに，複合システムの要求が異なり，それぞれに要求を満たすシステム構成が必要である．

ここで，ビルごとに要求が異なるということは，複合システム全体として提供する機能に関して可変性が生じていることになる．そして，その要求ごとにシステム構成が異なるため，システム構成としてのサブシステムの組み合わせ方にも可変性がある．そのため，複合システムの可変性に関しては，以下の 2 点の可変性を表現できる必要がある．

- 複合システム全体としての機能はどのような可変性があるのか，またそれらは，どのサブシステムの機能を組み合わせると実現できるのか？
- 各複合システムは，どの種類のサブシステムを何台組み合わせて構成されるのか？

また，複合システムの開発において，ビルごとに生じるような要求を，複合システムが満たしているのかのシステムテストが行われる．そのため，複合システムの要求のそれぞれに対して，テストケースを設計し，それらを実施可能なシステム構成を導出する必要がある．

複合システムのシステムテストは，以下の特徴を持つ．

- 複合システム向けのシステムテストは，サブシステムにおいて新たに製品開発された際に実施され，後方互換すべき既存サブシステム製品が多量に存在する．
- 複合システム向けのシステムテストでは，サブシステムの新製品の開発の時点で，大量のテストケースが蓄積されており，新製品が組み込まれたシステムにおいてもデグレートが起きないことを確認する必要がある．
- 複合システム向けのシステムテストの実行時，システム構成の切り替えには非常に多くの工数を必要とする．

上記の特徴の 1 点目は，架空の業務用空調機の例において，表 2.2 にある OuterEX-HIGH-Eco という室外機製品が新たに開発されたと仮定したときに，表 2.2 の他の室外機製品と表 2.3 に並ぶ室内機製品すべてが過去に開発済みとなるという意味である．この仮定では，新製品に対して，既存サブシステム製品が 15 種あることになる．筆者が業務で経験した複合システムでは，このような既存の後方互換すべきサブシステムは，数十から数百に上ることがあった．そして，架空の業務用空調機では，複数台の室外機と複数台の室内機によってシステムが構成される．そのため，生じうるシステム構成の種類は，室外機で複数種類の製品が複数台，室内機で複数種類の製品が複数台と組み合わせていくと，理論上の組み合わせ数は無限となる．

2 つ目の特徴として，すでに既存サブシステム製品が多量に存在し，それらを組み合わせたシステム構成で複合システムがリリースされているため，それらをテストしてきたテストケースが大量に蓄積されている．これらのテストケースは，新たに製品開発されたサブシステムを含むシステム構成においてもデグレードが起きないことを検証するために，実施する必要がある．しかし，これらの蓄積されたテストケースの全てを実施するのは工数の問題で困難である．また，蓄積された全てのテストケースの中には新規開発されたサブシステムと関係がないものも含まれている．一方で，新規開発されたサブシステムが持つ機能に，関連するテストケースは全て実施しなければならない．そのため，多量にあるテストケースセットの中から，実施しなければならないテストケースの抽出を行わなければならない．このテストケースの抽出も，筆者が業務で経験した複合システムでは，有識者による手作業で実施されており，難易度が高く，作業にかかる工数も多くなっていた．

そして，テストケースごとにテスト対象の機能が存在し，その機能が実現されているシステム構成が必要となる．例えば，架空の業務用空調機において，室内の人を検出して風が当たらない機能をテストするテストケースでは，室内機には赤外線カメラを持つ製品を選択し，室外機には新プロトコル通信を持つ製品が必要となる．このように，テストケースごとに，テストを実施可能なシステム構成は異なる．さらに，実際の開発現場では，サブシステム製品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，テストケースに対応したシステム構成を導出する作業は，暗黙知を多く蓄積した開発者が長い時間をかけて実施している．

2 点目の特徴の説明において，デグレードが起きないことを検証するために大量にテストケースを実行する必要があり，テストケースごとに実施可能なシステム構成は異なることを述べた．この時，3 つ目の特徴として，システムテスト実行時にテストケースを消化していく際に，システム構成を変更するという作業が生じる．そして，システム構成を変更する作業では，該当するサブシステムを入れ替えて配線作業を行うため，非常に多くの物理的な工数が生じる．そのため，システムテスト実施中に，システム構成の変更数を最小にすることがコスト削減につながる．より少ないシステム構成にて，必要なテストケースセットを網羅して実施できる必要がある．

筆者が業務で経験した複合システムの開発では，この複合システムの要求を満たすシステム構成の導出は，有識者による暗黙知を用いての経験的な手法が用いられていた．サブシステム間の排他関係や，サブシステムの組み合わせによって複合システムに表出する機能に関して，暗黙知となっており，体系的な導出手法は

整備されていなかった．ヒアリングした範囲で，この複合システムのシステム構成導出には，有識者であっても多くの時間がかかっていた．

2.3 本研究の課題

本節では本研究における課題について述べる．2.2.3 節で述べたように，複合システムの要求やシステム構成に関する，複合システム向けの可変性表現が必要となる．また，テストケースごとサブシステムを選択してテスト実施可能なシステム構成を得る必要があり，その導出作業にはその複合システムに熟練した開発者が多くの時間をかけて行うため，実施すべき大量のテストケースに対して，少ないシステム構成数でそれらを実施したいという要求がある．

そこで，本研究では以下の 2 つを研究課題とする．

課題 1 複合システム向けの可変性モデルが必要
課題 2 複合システムのシステム構成導出が困難

本研究では，これらの課題を解決するために，複合システムの可変性モデリング手法と，そのモデリング手法を用いたシステムテスト向けのシステム構成の導出方法を提案する．本節の以降では，これらの課題それぞれについて述べる．

2.3.1 課題 1:複合システム向けの可変性モデル

本節では，本研究で扱う複合システム向けの可変性モデルの課題について述べる．2.2.3 節で述べたように，複合システムの開発において，要求を満たす複合システムを開発では，要求を実現するためにサブシステム製品を組み合わせる．この時，どの製品を組み合わせるかというシステム構成の導出を体系的に行うためには，複合システム向けの可変性が必要となる．ここで，複合システム向けの可変性に関して，さらに以下の 2 つの課題に分割する．

課題 1-1 複合システム向け可変性を単一モデルで表現することの困難性
課題 1-2 複合システム可変性における多重度の曖昧さ

まず，課題 1-1 の，「複合システム向け可変性を単一モデルで表現することの困難性」に関して述べる．複合システムは，組み合わせるサブシステムそれぞれの可変性がすでに存在しており，これらの可変性を 1 つのモデルに取り込んだ形で複合システムの可変性を表現できれば，管理しやすいため利便性の点で優位である．しかし，複合システムの可変性は，サブシステムそれぞれに生じる可変性と分けて表現すべきである．サブシステム開発はそれぞれ独立に行われており，サブシステムの可変性は事前に定義されている．そのため，複合システムの可変性においてサブシステムの可変性を内包する形で記述してしまうと，既に定義されたサブシステムの可変性まで変更されてしまうリスクが生じる．サブシステムの可変性を複合システムの可変性と分離しておくことで，関心ごとの分離が実現でき，サブシステム可変性の再利用性が高まる．

複合システムの可変性は，サブシステム可変性モデルから分離した上で，さらに 2 つの可変性に分けた方が良い．2.2.3 節で述べたように，複合システムの可変性には，複合システムとして表出する機能の可変性と，サブシステムの組み合わせ方の可変性がある．本論文では前者をシステム可変性，後者を構造可変性と呼ぶ．

Definition 11（システム可変性） 複合システム全体としての機能や特徴に関する可変性．複合システム全体の要求や制約も表現する．

複合システム全体の要求は，サブシステムの機能の組み合わせによって達成される．複合システムを構成

する際には，サブシステムの間で排他関係 *(exclude)* や要求関係 *(require)* といった制約が生じることがある．サブシステムそれぞれが複合システム全体を想定せずに，各々の可変性を定義するため，これらの制約はサブシステム可変性には表れてこない．一方で，複合システムとしてのシステム可変性でこれらの制約を表現する必要がある．

各サブシステムは独立して開発されており，サブシステムの可変性は独自の分析がされている．そして，筆者の経験上，サブシステムの種類ごとに開発部署は異なることが多い．サブシステムの種類とは，架空の業務用空調の例の，室内機と室外機が該当する．さらに，複合システム全体の構成を管理するシステム開発部署が存在する．そのため，複合システムとサブシステムは役割が独立しており，それらをまとめて記述することは非現実的である．システム部署は，2.2.2 で述べた複合システム全体の要求や制約を可変性として表現することに注力すべきである．そして，複合システムのシステム可変性は，サブシステムが組み合わさって機能を提供するため，サブシステム可変性モデル (業務用空調機における図 2.7 と図 2.8) と関係が生じる．

Definition 12（構造可変性） どのサブシステムがどれだけの数組み合わさって複合システムを構成できるのかを表す可変性.

この構造可変性は，サブシステムの種類と数の可変性を示す．架空の業務用空調機の例でいえば，室内機の種類と数の選択と室外機の種類と数の選択をすることでシステムを構成できる．この構造可変性の室内機や室外機の台数に関する可変性は，インスタンス多重度で表すことができる．

システム可変性と構造可変性は，両者ともに複合システム全体の可変性を表現しているが，視点が異なる．システム可変性は，システム全体としての機能性に関する可変性といえる．その一方で，構造可変性は，サブシステムの組み合わせに関する可変性である．また，例えば構造可変性が束縛されると，複合システムにおける具体的なサブシステム製品が選択された状態になり，その結果実現されるシステム全体の機能が定まる．複合システムの構造を独立して記述することで，サブシステムがいかに組み合わさり複合システムを構成するかが分かりやすくなり，誤りを生じにくくなる．また，複合システムが全体として持つ特徴や機能をサブシステムから独立してシステム可変性として表現することで，サブシステムを隠蔽し，検討するスコープを限定することができる．これらの側面は，1 つのモデルに集約して記述することは可能ではあるが，それは混乱を招く可能性がある．これらの可変性は，1 つのモデルに統合することは可能であるが，その場合，透明性が低くなり，誤りが生じやすくなる．

次に，「複合システム可変性における多重度の曖昧さ」が生じるという課題 1-2 について述べる．架空の業務用空調機の例では，複合システムは，少なくとも一台以上の室内機と少なくとも一台以上の室外機から構成される．この「少なくとも一台」という表現は，室内機と室外機それぞれに対してのインスタンス多重度を表している．インスタンス多重度は，2.1.2 節で述べた CVL において，VClassifier ノード (四角形) で表現されるもので，そのノードからぶら下がるサブツリー構造のインスタンスの多重性を意味する．可変性モデルにおけるインスタンス数の許容範囲は，CVL では VClassifier ノード内に上限数と下限数で記述される．

そして，複合システム全体に関して，自然言語で書かれた制約や要求は，インスタンス多重度に対して曖昧さを生じることがある．そのような曖昧さは，誤解につながったり，システム構成の際の矛盾を生じたりする．このような曖昧さを解決するためには，インスタンス多重度を適切に識別し取り扱う必要がある．

2.2.2 節で述べた Constraint 3 (冷媒消費能力の値は，冷媒提供能力の値を超えてはならない) には，2 つの能力値の比較が記述されている．冷媒消費能力の値と冷媒提供能力の値は，それぞれ複合システム全体として持つ能力値であり，サブシステムの組み合わせによって変化するため，両者ともにシステム可変性である．そしてこの 2 つの値の比較自体は非常にシンプルだが，冷媒消費能力の値と冷媒提供能力の値そのものには注意が必要である．例えば，表 2.2 と表 2.3 から，室外機に Outer-LOW(HP: 70) を 1 台と Outer-HIGH(HP: 150) を 1 台，室内機に Inner-HIGH(HP: 50) を 4 台繋げた複合システムを構成したとする．その時，冷媒消費能力の値と冷媒提供能力の値に関して，生じうる解釈の例として以下に 3 つ挙げる．

解釈 1 冷媒提供能力の値は室内機の HP の合計値であり，冷媒消費能力の値は室外機の HP の合計値である．

解釈 2 冷媒提供能力の値は室外機の HP の最大値であり，冷媒消費能力の値は室内機の HP の合計値である．

解釈 3 冷媒提供能力の値は室外機の HP の合計値であり，冷媒消費能力の値は室内機の HP の合計値である．

これら 3 つの解釈には，2 つの曖昧さを内包している．1 つ目は，冷媒提供能力の値と冷媒消費能力の値がそれぞれ，室内機と室外機の HP のどちらと関係するかの曖昧さである．解釈 1 においては，冷媒提供能力の値は室内機と関係し，冷媒消費能力の値は室外機と関係している．解釈 2 と 3 に関しては逆で，冷媒提供能力の値は室外機と関係し，冷媒消費能力の値は室内機と関係している．解釈 1 で計算すると，冷媒提供能力の値は 200 となり，冷媒消費能力の値の値は 220 となる．これは，冷媒消費能力の値の値が冷媒提供能力の値を上回り，Constraint 3 を満たさない．

2 つ目は，複合システムにおける冷媒提供能力の値と冷媒消費能力の値に対して，サブシステムにある複数の HP の値をどう集計するのかの曖昧さである．解釈 1 と解釈 3 では，サブシステムの HP の値を合計しているのに対して，解釈 2 では冷媒提供能力の値に室外機の HP の最大値を用いている．解釈 2 で計算すると，冷媒提供能力の値は 150 となり，冷媒消費能力の値の値は 200 となり，これも Constraint3 を満たさない．

ここでは，この 3 つの解釈の中で正しい解釈は 3 番であったとする．冷媒提供能力の値は室外機の HP の合計値を意味しており，冷媒消費能力の値は室内機の HP の合計値を意味する．この解釈において，室外機に Outer-LOW(HP: 70) を 1 台と Outer-HIGH(HP: 150) を 1 台，室内機に Inner-HIGH(HP: 50) を 4 台繋げたシステム構成は，冷媒提供能力の値は 220 となり，冷媒消費能力の値の値は 200 となるため，Constraint3 を満たす．

つまり，冷媒消費能力と冷媒提供能力の例では，以下の 2 つの曖昧さが生じる．

- システム可変性が，どのサブシステム可変性と関係するのかが曖昧
- システム可変性がサブシステム可変性と関係する時，その間にあるインスタンス多重度と，その集約方法が曖昧

複合システムにおけるシステム可変性は，サブシステムの可変性とは独立して分析され，システム全体としての制約や要求を表現する．その一方で，サブシステム可変性の組み合わせによって，それらのシステム可変性は影響を受ける．つまり，複合システムにおけるシステム可変性 (上記の例では，冷媒消費能力の値と冷媒提供能力の値) は，構造可変性に示すサブシステムの台数を表すインスタンス多重度と関連して，そのサブシステムが持つ可変性を集約したものとなる．複合システムにおけるシステム可変性において，2 つの曖昧さを排除するためには，関連するサブシステム可変性を適切に選択し，その間に生じるインスタンス多重度を集約することが必要である．そして，そのインスタンス多重度を記述する構成可変性に基づく制約が生じるため，できるだけ誤りなく表現する方法が必要となる．

2.3.2 課題 2:複合システムのシステム構成導出

2.2.3 節で述べた複合システム向けシステムテストの特徴から，以下のように課題を整理する．

課題 2-1 実施すべきテストケースの抽出が必要

課題 2-2 テストケースごとに実施可能なシステム構成の導出が必要

課題 2-3 より少ないシステム構成で必要なテストケースセットの網羅が必要

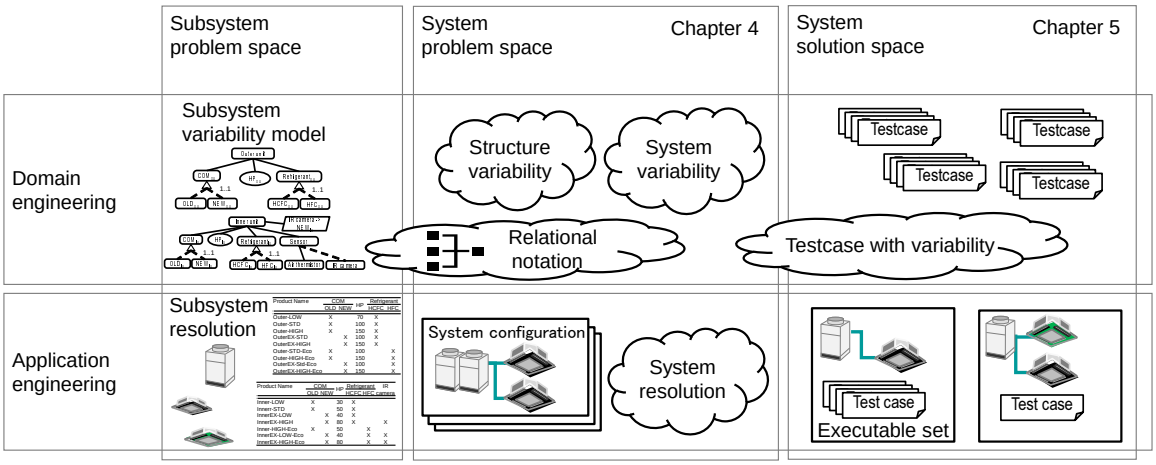

図 2.10: 解決方針

まず，課題 2-1 の「実施すべきテストケースの抽出が必要」について述べる．2.2.3 節で述べたように，派生開発を重ねた結果，テストケースが大量に蓄積されており，後方互換性の確認のために，それらのテストケースを実施して検証する必要が生じる．しかしながら，大量にあるテストケースを全て実施するのは，現実的な工数に収まらないため，不可能である．新しく追加した機能やサブシステムが持つ特性に関連するテストケースを抽出してテストをしなければならないという課題がある．

次に，課題 2-2 の「テストケースごとに実施可能なシステム構成の導出が必要」という課題について述べる．2.2.3 節で述べたように，テストケースごとにテスト対象の機能が存在し，その機能が実現されているシステム構成が必要となる．このシステム構成の導出には，図 2.9 における，左側のサブシステムのプロブレムスペースと，右側のシステムのソリューションスペースをつなぐ必要がある．左側のサブシステムのプロブレムスペースでは，サブシステムの可変性モデルとリゾリューションモデルが定義される．一方の右側では，複合システムの視点で，要求を確認するテストケースごとに，それを実現するシステム構成を用意している．この間をつなぐ方法として，複合システムの可変性モデルが利用可能である．

最後に，課題 2-3 の「より少ないシステム構成で必要なテストケースセットの網羅が必要」について述べる．システムテスト実行時にテストケースを消化していく際に，システム構成を変更するという作業が生じる．そして，システム構成を変更する作業では，該当するサブシステムを入れ替えて配線作業を行うため，非常に多くの物理的な工数が生じる．そのため，システムテスト実施中に，システム構成の変更数を最小にすることがコスト削減につながる．より少ないシステム構成にて，必要なテストケースセットを網羅して実施できる必要がある．

本研究では，特に課題 1 と，課題 2 のうちの課題 2-2 と課題 2-3 に関して取り組む．本研究では課題 1 に対して，複合システム向けの可変性モデルを提案する．その可変性モデルを用いて，実施したいテストケースをその可変性モデル上で表現し，複合システムの可変性を束縛することで，テストケースに対応するシステム構成を得る手法を提案し，さらにより少ないシステム構成数でテストケースを網羅する手法を提案する．課題 2-1 の，テストケースの抽出に関しては，可変性モデルにおいてテストケースとしてフィーチャを選択する手法 [19–22] が提案されているため，既存手法を用いる．これらの関連手法の詳細は，3 章にて述べる．

図 2.10 に課題の解決方針を示す．サブシステムのドメイン開発プロブレムスペースにおいて各サブシステムのサブシステム可変性モデルがあり，アプリケーション開発プロブレムスペースにサブシステムリゾリューションモデルがある．まず，課題 1 に対して，複合システム全体に対するプロブレムスペースを導入し，構造可変性とシステム可変性のモデル表現を提案する．特に課題 1-2 の「複合システム可変性における多重度の曖昧さ」に関連して，インスタンス多重度を伴う関係表現を提案する．そして，提案する複合シス

テム全体に対する可変性を束縛すると，システム構成が得られる．続いて，課題 2 に対して，複合システム全体に対するソリューションスペースにて，テストケースをプロブレムスペースにおける可変性表現と紐づける表現を導入する．そして，この紐づけによって，プロブレムスペースにおける可変性の束縛に伴い，テストケースとそれを実施可能なシステム構成の組を得る方法を提案する．課題 1 に対する複合システムのプロブレムスペースの可変性モデルについては，4 章で述べる．課題 2 に対する複合システムのソリューションスペースのテストケースに対応するシステム構成導出は，5 章で述べる．

第 3 章

関連研究

本章では，まず，複合システム向けの可変性モデルに関する関連研究について述べる．次に可変性とテストに関する関連研究について述べ，最後にシステム構成の導出の関連研究について述べる．

3.1 可変性モデルの表現方法

本節では，2.3 節で述べた課題に対応して，可変性モデルが持つべき要件を整理する．次に，関連する可変性モデルに関してグラフィカルベースの表現とテキストベースの表現を紹介する．また，可変性を扱う商用ツールについても述べる．

3.1.1 可変性モデルの要件

2.3 節の課題 1 で述べたように，複合システム向けの可変性には，システム可変性と構造可変性が生じる．これらの可変性を複数の可変性モデルに分けて記載した場合，可変性モデルの表記方法は，他の可変性モデルへの参照や関連を記述できる必要がある．また，例えば構造可変性が束縛されると，複合システムにおける具体的なサブシステム製品が選択された状態になり，その結果実現されるシステム全体の機能も定まる．そのため，一方のモデルでの可変性の束縛が他方のモデルに伝搬できる必要がある．そして，具体的なサブシステム製品の表現は，サブシステム可変性が束縛されたリゾリューションモデルによって行われる．

構造可変性において，サブシステムの台数の可変性を表現するため，可変性モデルの表記方法はインスタンス多重度を表現できる必要がある．また，課題 1-2 で述べたように，可変性表現の中でインスタンス多重度の集約関係を記述する必要がある．

まとめると，可変性モデルに対する要件として，以下の 4 点がある．

- 複数の可変性モデルによる表現ができる．また，その間の関係表現と束縛情報の伝搬ができる．
- 可変性モデル内でインスタンス多重度を表現できる．
- 可変性モデルの関係表現においてインスタンス多重度の識別と集約表現ができる．
- 可変性モデルに対応した束縛後のリゾリューションモデルが表現できる．

3.1.2 グラフィカルベースの可変性モデル

本節では，グラフィカルベースの可変性モデルに関して述べる．グラフィカルベースとは，グラフ構造や木構造などを用いて図として可変性を表現するものである．2.1.2 節で述べた Feature Model や OVM や CVL は，グラフィカルベースの代表的な可変性モデル表現である．

FODA [5] で提案された Feature Model ではインスタンス多重度をサポートしていないが，Czarnecki らは，

Feature Model を拡張し，Cardinality と呼ばれるインスタンス多重度の表現を加えた [23]．また，Czarnecki らは，Multi-level staged configuration [11] を提案した．これは，可変性に複数の階層を用意し，上位層では抽象化された可変性モデルを記述し，下位層では具体的なフィーチャを記述し，異なる階層の可変性の間に関連を張ることができる．この関連を用いて，いずれかの階層での可変性の束縛が，他の階層に伝搬することが可能となる．さらに本手法では，束縛情報の伝搬に関して Positive(束縛時に選択したことを伝搬) と Negative(束縛時に選択しなかったことを伝搬) と Complete(束縛時に選択と選択しなかったことの両方を伝搬) の 3 種類の関係を定義した．Czarnecki らは，可変性モデル内の制約記述言語として，OCL を用いることも提案している [24]．OCL には，集約演算子があるため，インスタンス多重度の集約は表現可能である．Czarnecki らの手法は，可変性モデルと束縛後のリゾリューションモデルの表記は同じ文法を用いて記述する．

Reiser らは，コンポーネント図と連動したフィーチャ分析を提案した [25] ．この手法では，コンポーネント図によってシステム全体の構造を示し，その図の要素であるコンポーネントそれぞれに可変性を分析し Feature Model を記述する．フィーチャ間の関係に関しては 5 パターンを定義しており，これらはシステムとサブシステムの間の，可変性の束縛情報伝搬に用いられる．Reiser らは，さらに，Multi-level feature tree [26] を提案している．これは，フィーチャモデルの間の参照関係を定義しており，参照するモデルは参照先モデルから束縛されたものとなる．また，参照するモデルは，束縛だけではなくフィーチャの追加などの改変も許容する．

Chavarriaga らは，可変性の束縛情報伝搬について，Multi-step configuration process [27] を提案した．伝搬のための可変性の間の関係に関して，Force と Prohibit の 2 種類があり，これらの関係は，Czarnecki らの Positive と Negative の関係と同じある．さらに，本手法では，可変性の間の関係である Force と Prohibit を用いて，束縛情報が伝搬した際に，伝搬した先のフィーチャに Selected と De-selected の 2 種類のマークを付与する．フィーチャは異なる伝搬元から複数の束縛情報を受け取る可能性があり，この Selected と De-selected を両方マークされたフィーチャを検出すると，可変性の矛盾を検出できる．本手法では，異なる抽象レベルの可変性モデルを定義し，その間の束縛情報が伝搬できる．

Hubaux らは，Multi-view feature based configuration [28] を提案した．この多視点アプローチでは，ステークホルダごとに，それぞれの視点に適する形でフィーチャモデルをグレイアウトしたり，枝刈りしたり，折りたたんだりして表示する．仮想的に複数のフィーチャモデルを提示できるが，ベースとなるモデルは単一である．

Janeta らは，フィーチャモデルとアーキテクチャモデルを統合する形式的なアプローチ [29] を提案した．このアプローチは，フィーチャのコンポーネントの間に Decision propagation link を定義する．これはシステムの構造を示すコンポーネントとシステムのフィーチャの関係を表現できる．

2.1.2 節で述べた CVL は，ドメイン非依存の GUI を用いた言語であり，可変性の明示と解明を行うことができる．可変性に関しては，VSpec と呼ばれる木構造表現で行い，その束縛後のリゾリューションモデルも提供している．VSpec モデルとリゾリューションモデルは異なるメタモデルによって定義されている．また，VSpec には，インスタンス多重度を示す VClassifier ノードや，他の VSpec モデルを参照する CVSpec ノードが存在する．Haugen らは，CVL から発展した Better Variability Results(BVR) [30] も提案している．BVR には，VType という構造可変性を表現可能な拡張がある．

Krueger らは，サブシステムフィーチャモデルとシステムフィーチャモデルを統合する System-of-systems の可変性のためのオントロジー [31] を提案した．また，Tekinedagon らは，そのオントロジーを拡張してフィーチャ駆動開発プロセス [32] を提案した．これらの手法は，サブシステムの可変性を独立して分析し，システム全体の可変性分析ではそれらを統合するアプローチをとる．しかし，これらはオントロジーやプロセスの提案であり，可変性の表現方法や，インスタンス多重度に関する言及はない．

3.1.3 テキストベースの可変性モデル

Rosenmüller らは，Velvet [33] という，多次元可変性モデリング手法を提案した．この Velvet では，サブシステムのインスタンスごとに識別名を付与することで，システム内の複数のサブシステムインスタンスを表現できる．しかし，それぞれのインスタンスは異なる識別名を必要としており，サブシステムのインスタンス数が可変であったとしても，その束縛結果しか表現できない．そのため，インスタンス数の許容範囲などの可変性は表現できず，インスタンス多重度は直接的にサポートしない．Velvet では，また，ベースとなるフィーチャモデルと，そこから可変性が絞られた束縛後のフィーチャモデルの間の継承関係を定義している．Velvet では，可変性モデルと束縛後のリゾリューションモデルの表記は同じ文法を用いて記述する．

Classen らは，インスタンス多重度の可変性を許容する数値で表現可能な TVL [34] を提案した．本手法では，親ノードに対する子ノードのインスタンス多重度に関して，SUM/MAX/MIN の集約演算子を付与できる．これらの集約は，木構造における直接の親子ノード間にしか付与できない．

Abele らも，インスタンス多重度を記述できる VSL [35] を提案した．この手法では，サブシステムのインスタンス多重度を許容できる数値で表現可能であり，さらにインスタンスに他のインスタンス特別するための識別名も付与できる．加えて，フィーチャモデル間の束縛情報の伝搬も提供している．この伝搬には，Selection と Inclusion という 2 つのタイプを持つ ConfigLink という関係を用いる．Selection ConfigLink は，ConfigLink のソースフィーチャが選択として束縛された際に，ターゲットフィーチャから木構造のルートノードとなるフィーチャまでのすべてのフィーチャが選択されるという伝搬方法を示す．Inclusion ConfigLink は，ConfigLink のソースフィーチャが選択として束縛された際に，ターゲットフィーチャのみが選択されるという伝搬方法を示す．そのため，Inclusion ConfigLink で伝搬された場合，そのターゲットフィーチャのより上位の可変性によって除外されている可能性がある．Selection や Inclusion が，選択として束縛した情報を他のフィーチャモデルへ伝搬するリンクだったのに対して，除外で束縛した際の伝搬を表すリンク De-selection と Exclusion も VSL では提供している．これらの違いは，Selection や Inclusion の違いと同様である．VSL における ConfigLink では，インスタンス多重度に関する集約の表現はない．VSL では，可変性モデルと，束縛後のリゾリューションモデルで別の表記法を導入している．

Sausa らは，可変性におけるインスタンス多重度に着目して，Relative cardinality [36] を提案した．インスタンス多重度が，直接つながっている子ノードに対するインスタンスの数の許容数を示すのに対して，Relative cardinality は直接つながらないような祖先のノードに対するインスタンス数の許容数も記述できる．これは，例えば可変性モデルにおいて，ノード A，ノード B，ノード C という 3 つが直列に分解されていた場合，ノード A は最大で 10 個のノード B を保持できるというのが，インスタンス多重度の表現になる．ここでさらに，ノード B は最大で 10 個のノード C を保持できるというインスタンス多重度の可変性があった場合，ノード A1 つに対して，ノード C は原理的に 100 個のノードが生じる．Relative cardinality では，この時に，ノード A1 つに対してノード C は最大で 50 個までとするといった可変性に対する制約を記述できる．この Relative cardinality を用いた記法では，1 つの可変性モデルの中での多重度の関係を記述しており，複数のモデルを連携させることは想定していない．

3.1.4 商用ツールにおける可変性モデル

可変性管理で最も有名な市販の商用ツールとして，pure-systems 社 [37] が提供する pure::variants がある．このツールでは，可変性表現としてフィーチャモデルを記述し，それを束縛したリゾリューションモデルの表現もサポートする．リゾリューションモデルに関してはプロダクトマップのビューも提供している．フィーチャモデルには，下位のフィーチャモデルを参照する階層構造にも対応している．つまり，図 2.4 における

左側のプロブレムスペースの表現を広くサポートする．そして，図 2.4 の右側のソリューションスペースに対応して，外部の様々なツールと連携できる．例えば，制御設計ツールである simulink，要件管理ツール Doors，テスト管理ツール Clearcase などと連携できる．

ptc 社 [38] は，開発ライフサイクルマネジメントツールセットの中の Integrity modeler にて，SYSML [39] によるソフトウェアの構造的な設計表現と，それに紐づく OVM による可変性表現の連携機能を提供する．これは，構造可変性を SYSML で表現し，その構造部分に限定されたサブシステムにおける可変性を OVM で表現できる．

3.1.5　要件に対する関連研究の評価

本節では，これまで紹介した可変性モデルの関連研究と 3.1.1 節で述べた要件の対応関係について述べる．要件 4 種を列に並べ，各関連研究を行として，表 3.1 にまとめた．

要件を最も満たしているのは，Haugen らの CVL と Czarnecki らによる Multi-level staged configuration である．これらは，本研究で課題として挙げているシステム可変性・構造可変性とその間の関係性を記述できる可能性がある．リゾリューションモデルに関して，Multi-level staged configuration では，可変性モデルの可変部に生じる選択肢を消して別の可変性モデルとして表現する．それに対して，CVL では専用の束縛結果を表すモデルを導入し，そのモデルがベースとなった可変性モデルを参照しており，CVL の方が可変性モデルとリゾリューションモデルの連携について表現しやすい．そのため，本研究では CVL を用いることとする．

3.2　可変性とテストに関する研究

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 [19] が提案されている．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 [20, 21] も提案されている．そして，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 [22] が提案されている．これらの関連研究では，テストすべき結果として，可変性モデルが束縛された結果であるリゾリューションモデルが得られる．つまり，これらの手法は，図 2.4 における左側のプロブレムスペースに閉じた提案となっている．

図 2.4 におけるソリューションスペースにおいて，テスト向けのシナリオの導出 [40] を行う手法も提案されている．この手法では，ドメイン開発において可変性を含んだ形でアクティビティ図によってテストのシナリオを記述する．可変性を含む箇所に関しては，アクティビティ図の分岐を利用し拡張している．そして，アプリケーション開発において，プロブレムスペースにおける可変性の束縛に連動する形で，アクティビティ図内で分岐表現された可変性が決定し，実行可能なテストシナリオが導出される．

3.3　システム構成導出に関する研究

本論文では，複合システム向けのシステムテストにおいて，少ないシステム構成数で，すべてのテストケースを実施することを目的としている．Scheidemann は，分散組み込みシステムのプロダクトラインにおける，システムテスト向けの代表構成を導出する手法 [41] を提案している．この手法では，生じうる構成と要求の組に対して，その構成で要求が検証可能かどうかを定義したデータセットを入力とし，少ない構成数ですべての要求が満たされるような構成を，貪欲法を用いて導出している．この関連研究では，システムを構成するサブシステムに関する可変性やシステムに関する可変性は用いず，要求と検証可能なシステム構成があら

表 3.1: 可変性表現と要件の対応

関連研究	複数のモデルと束縛伝搬	インスタンス多重度表現	インスタンス多重度の識別と集約表現	リゾリューションモデル表現
Multi-level staged configuration [23] [11] [24]	抽象度の異なる可変性モデルを階層化．伝搬を定義	Cardinality を導入	識別無し OCL 利用	可変性モデルと同じ表記法
Multi-level feature tree [25]	抽象度の異なる可変性モデルを階層化．伝搬を定義.	言及無し	言及無し	可変性モデルと同じ表記法
Multi-step configuration process [27]	抽象度の異なる可変性モデルを階層化．伝搬を定義.	言及無し	言及無し	可変性モデルと同じ表記法
Multi-view feature based configuration [28]	見た目を変えるのみで，単一可変性モデル	言及無し	言及無し	可変性モデルと同じ表記法
Decision propagation link [29]	コンポーネント図と可変性モデル連携	言及無し	言及無し	可変性モデルと同じ表記法
CVL [7] [30]	抽象度の異なる可変性モデルを階層化．伝搬はない.	表現可能	識別無し OCL 利用	リゾリューションモデル定義
Krueger et.al, [31] Tekinedagon et.al, [32]	抽象度の異なる可変性モデルを階層化．伝搬はない.	言及無し	言及無し	言及無し
Velvet [33]	抽象度の異なる可変性モデルを階層化．伝搬を定義.	識別名で記述	言及無し	可変性モデルと同じ表記法
TVL [34]	言及無し	表現可能	識別無し SUM/MAX/MIN	可変性モデルと同じ表記法
VSL [35]	抽象度の異なる可変性モデルを階層化．伝搬を定義.	表現可能	言及無し	リゾリューションモデル定義
Relative cardinality [36]	単一可変性モデル	表現可能	識別無し Relative cardinality で MAX 表現	可変性モデルと同じ表記法

かじめ判明している．本論文では，この関連研究ではあらかじめ与えられる「その構成で要求を検証可能かどうか」に関して可変性を用いた手法を提案する．

システム構成導出に関して，アーキテクチャ設計において，非機能要求を達成する構成を導出する研究がある．Tsadimas ら [42] は，エンタープライズインフォメーションシステムのアーキテクチャ設計において，他のステークホルダによって既に定義されているコンポーネントを効率的な方法で統合する手法を提案した．この手法では，非機能要求を中心に据えたモデルベースアプローチをとっており，ファンクショナルビューとトポロジービューとネットワークインフラストラクチャビューと非機能要求ビューの 4 つのモデルを記述し，それらが連動することで構成を得る．ファンクショナルビューや非機能要求ビューにおいてシステム可変性を表現し，トポロジービューとネットワークインフラストラクチャービューにて構造可変性を表現できる．しかしながら，この手法は，エンタープライズインフォメーションシステムを前提としており，サーバ・ワークステーションが連携してサービスを提供するシステムのみが対象である．非機能要求も，トラッフィクやロードなどエンタープライズシステムに特化している．

風戸ら [43] は，ソフトウェアの実行基盤を構成する個々の実装技術と，ソフトウェア全体が満たす品質特性の因果関係をベイジアンネットワークでモデル化し，実装技術の選択を支援する手法を提案した．この手法では，ベイジアンネットワークにおいて，実装技術を選択するノード，設計上の決定事項を示すノード，満たすべき品質特性を示すノードの 3 つのレイヤに分けてモデル化し，実装技術を選択した場合の条件付き確率の伝搬によって，品質特性への影響を可視化する．つまり，実装技術を選択するノードが，サブシステムの選択を表す可変性とみなすことができ，品質特性はシステム可変性とみなすことができる．サブシステムの台数に関する可変性も，ベイジアンネットワーク上の別のノードとして定義できる可能性がある．しかし，サブシステムの可変性と構造の可変性とシステムの可変性を全てベイジアンネットワーク上にモデル化すると，可読性が低くなる．また，この手法では，システム構成の導出を支援する確率表示にとどまる．

第 4 章

複合システムの可変性モデルとピンサームーブメントアプローチ

本章[*1]2.3.1 節で述べた課題を解決するために，複合システムの可変性モデルを提案する．まず，課題 1-1 を解決するために，複合システム向け可変性モデルとして，可変性表現の分割した記法を提案する．次に，課題 1-2 を解決するために，多重度の識別のためのピンサームーブメントアプローチについて述べる．そして提案する分割記法とピンサームーブメントアプローチを実現するツールについて述べる．続いて，提案手法に関して，ツールを用いて評価実験を行った結果について紹介する．最後に，実験結果及び提案手法に関する議論を述べる．本章では，2.2.2 節で述べた仮想の業務用空調機を例として用いる．

4.1 複合システム向け可変性モデルの分割記法

複合システム向けの可変性モデルは，サブシステムそれぞれの可変性モデルとは別に，2 つの視点が存在しており複雑になる．1 つの視点は，複合システムの中にどのサブシステムが何台含まれるかという視点である．業務用空調機でいうと，室内機と室外機のそれぞれが何台つながることで複合システムを構成するかという可変性である．もう一つの視点は，複合システム全体として提供する機能に関する可変性である．業務用空調機では，複合システム全体としての冷媒の提供能力値が，その可変性に該当する．

複合システムの可変性は，複合システムを 1 つの分割されていない実体としての可変性と，台数の可変性が付与された各々の可変性を持つサブシステムの構成としての可変性の二つに分けられる．そこで本論文では，複合システムの可変性を，システム可変性モデルと構造可変性モデル[*2]の 2 つに分けた分割記法を提案する．これらの可変性モデルは，可変性が束縛される際に，一貫性をもって連携する．

本節の以降では，サブシステム可変性モデルについて述べた後，システム可変性モデルと構造可変性モデルのそれぞれについて述べ，これらが連携する全体像について述べる．

4.1.1 サブシステム可変性モデル

提案するシステム可変性モデルと構造可変性モデルは，複合システムを構成するサブシステムそれぞれのサブシステム可変性モデル (Definition 9) を参照する．そのため，システム可変性モデルと構造可変性モデルの前に，サブシステム可変性モデルから作成する必要がある．

複合システムを構成するサブシステムは，それぞれが独立して派生開発され，個別の可変性を持つ．そし

[*1] 本章の内容は，掲載予定 (採録決定済み) の論文 [44] に基づく．

[*2] 筆者の以前の発表 [45] から発展させたモデルであり，その論文においては System feature model と Structure model と称していた．

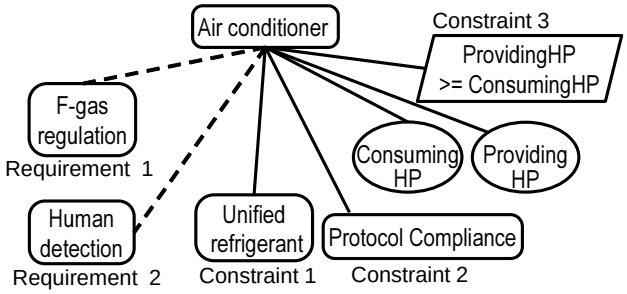

図 4.1: システム可変性モデル

て，2.3.1 節の課題 1-1 で述べたように，複合システムの可変性とは分けて記述する必要がある．そのため，サブシステム可変性モデルは，複合システム全体の視点を持たず，表現対象のサブシステムに閉じた表現となる．

提案する可変性モデルにおいてサブシステム可変性モデルは，4.2.2 節で後述するピンサームーブメントアプローチによって，可変性の分解関係における上位から下位のノードへの探索を行うため，木構造表現が求められる．また，サブシステム可変性モデル内に、インスタンス数の可変性を表現できる必要がある．そのため，本提案におけるサブシステム可変性モデルは，サブシステムそれぞれの視点で CVL の記法に則り分析し作成する．

2.2.2 節の業務用空調機の複合機の例では，室外機のサブシステム可変性モデルを図 2.7 に，室内機の可変性モデルを図 2.8 に示した．

4.1.2　システム可変性モデル

システム可変性モデルは，Definition 11 について CVL を用いてモデル化したもので，以下のように定義する．

Definition 13（システム可変性モデル (System variability model)）　複合システムを 1 つの実体として捉え，複合システムが提供する機能に関しての可変性を表現するモデル．

システム可変性モデルは，複合システム全体の視点で可変性を CVL の記法に則り分析し作成する．そのため，システム可変性モデルでは，複合システム全体の要求や制約を表現できる．システム可変性モデルに生じるノードは，複合システム全体の視点で分析され，サブシステム固有の可変性は記述しない．その一方で，システム可変性モデルにおける可変性の束縛は，サブシステム可変性モデルの束縛結果の組み合わせによって定まるため，システム可変性モデルの各ノードはサブシステム可変性モデルと関連する．サブシステムの機能がそのまま複合システム全体に表出する場合は，システム可変性モデルとして改めて記述する．

例えば，2.2.2 節で述べた複合システム全体の要求や制約について述べる．これらの要求はや制約は，複合システム全体の可変性とみなすことができる．

複合システムにおける潜在的な要求は，多くのステークホルダから獲得され，複合システム開発において個別のシステムごとに要求を取捨選択する．そのため，複合システムにおける要求は，選択可能分解として表現するのが適している．2.2.2 節で述べた 2 つの要求は，図 4.1 の左側のように表現できる．Requirement 1 は，「フロンガス規制に準拠する」という要求であり，使用する冷媒に関連する記述である．フロンガス規制は，仕向け地によって準拠必須であったり，不要であったりする．そのため，図 4.1 では，「フロンガス規制 (F-gas regulation)」という Choice ノードが上位ノードから選択可能分解を示すコネクタでつながっている．サブシステム可変性モデルには，使用する冷媒タイプを直接的に表現した可変性があるが，一方でシ

表 4.1: 通信プロトコル (COM) の関係

Subsystem		System
Outer unit	Inner unit	Protocol compliance
OLD_{OU}	OLD_{IU}	Yes (old protocol)
NEW_{OU}	OLD_{IU}	Yes (old protocol)
NEW_{OU}	NEW_{IU}	Yes (new protocol)
OLD_{OU}	NEW_{IU}	No

ステム可変性モデルでは，冷媒に関する表現ではなく，規制への準拠に関する可変性表現となっている．これにより，複合システム全体の視点で見たとき，冷媒の種類が何であるかは意識しない分析が可能となる．Requirement 2 は，「空調システムは，室内の人間を検知し，直接風が当たらないようにする」という要求である．図上では，「人体検知 (Human Detection)」というノードが，選択可能分解で示されている．これは，複合システムが，空調機能を提供する室内において人の存在を検出できるかどうかの可変性を表している．

要求と対照的に，サブシステムを横断するような複合システムにおける制約は，常に満たされなければならない．制約に違反した場合，不正なシステム構成につながってしまうからである．そのため，システム可変性モデルにおいて，制約は必須分解によって表現される．2.2.2 節で述べた 3 つの制約は，図 4.1 の右側のように表現できる．Constraint 1 は，「室内機と室外機は同じ冷媒を利用する必要がある」という制約である．図 4.1 では，必須分解された Choice ノード「冷媒の統一 (Unified refrigerant)」が導入され，すべての室外機と室内機で同じタイプの冷媒を使用しなければならないことを示している．Constraint 2 は，「室内機と室外機は，相互に通信できる必要がある」という通信プロトコルに関する制約である．室外機と室内機の両方のサブシステム可変性モデルには，「通信プロトコル (COM)」の Choice ノードの下にオルタナティブ分解で「OLD」と「NEW」が存在する．一方で複合システムの視点では，通信プロトコルが何であるかは関係なく，室外機と室内機が通信可能であればよい．そこで，必須分解された Choice ノード「Protocol compliance」が導入される．この Choice ノードは，システム全体の視点から定義されており，室外機と室内機のそれぞれの COM 可変性と関係する．表 4.1 に，通信に係るシステム可変性モデルと，室外機・室内機のサブシステム可変性モデルの関係を示す．複合システムに含まれる室内機全てが旧プロトコルであった場合，室外機側が新プロトコルであったとしても通信の後方互換性により旧プロトコルとして動作するため，システム内で通信が可能である．逆に，室内機側が新プロトコルであった場合，室内機側には後方互換性がないため，室外機の旧プロトコルと通信ができない．この場合は，制約を満たさない構成となる．図 4.1 の「冷媒提供能力値 (ProvidingHP)」と「冷媒消費能力値 (ConsumingHP)」は，それぞれ複合システム内で提供される冷媒の能力値 (馬力) の合計と消費される冷媒の能力値 (馬力) の合計を示している．これらの能力値を示す可変性は，Variable ノードで表現されており，数値の可変性を示す．そして図 4 の平行四辺形で示される制約式「ProvidingHP >= ConsumingHP」によって，これらの Variable ノードに言及し，Constraint 3 の「冷媒消費能力の値 (HP) は，冷媒提供能力の値 (HP) を超えてはならない」を表現している．

図 4.1 で示した要求や制約を表すノードは，複合システム全体の視点で記述されている．その一方でこれらのノードの具体的な意味は，サブシステム可変性モデルのノードとの関係によって定義されている．例えば表 4.1 の通信プロトコルに関する関係のように，サブシステム可変性モデルにおける束縛の組み合わせによって，システム可変性モデルの束縛が定まる．この関係に関する詳細は，4.2 節にて述べる．

システム可変性モデルは，システム全体の視点での可変性の分析だけではなく，開発中のシステムのスコープ定義や，可変性を内包したシステムアーキテクチャの検討，派生製品の新機能の検討，システムテス

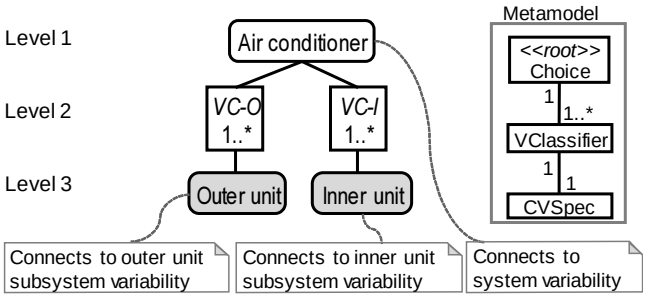

図 4.2: 構造可変性モデル

トのためのテストケース生成などにも利用できる．

4.1.3　構造可変性モデル

この構造可変性モデルは，Definition 12 について CVL を用いてモデル化したもので，以下のように定義する．

Definition 14（構造可変性モデル (Structure variability model)）　複合システムにおいて，どの種類のサブシステムがどれだけのインスタンス数で構成されるかを表す可変性モデル．

構造可変性モデルは，以下の三つのレベルを持つ木構造で示される．

- 木構造のルートとなる Choice ノード
- ルートの子要素で，インスタンス多重度を示す VClassifier ノード
- ルートの孫要素で葉ノードとなる CVSpec ノード

孫要素である CVSpec ノードは，サブシステム可変性モデルへの参照を意味し，その親に当たる VClassifier ノードはそれぞれのサブシステムに関する許容されるインスタンス数を示す．

構造可変性モデルにおいて，孫要素にある複数の CVSpec ノードが，同じサブシステム可変性モデルを参照することはできない．単に，インスタンスが複数生じることを表す場合は，子要素の VClassifier ノードに表されるインスタンス多重度で表現する．同じようなサブシステムが違う意味で接続されて複合システムを構成する場合は，異なるサブシステムとしてサブシステム可変性モデルのクローンを作成する．これは 4.2.2 節で後述するピンサームーブメントアプローチによって，可変性の分解関係における上位から下位のノードへの探索を一意に行うための制約である．

図 4.2 は，仮想の業務用空調機の構造可変性モデルを示している．構造可変性モデルのルートノードは，システム可変性モデルのルートノードと同じ「Air conditioner」である．これは，異なる視点ではあるが，両者はともに複合システム全体を示すため，同じ名前となっている．そのため，このルートノードは異なる名前でも良い．サブシステムは，室外機 (Outer unit) と室内機 (Inner unit) である．これらのサブシステムは，図 2.7 や図 2.8 のように，それぞれに個別のサブシステム可変性モデルを持っている．そして，業務用空調機は，1 つ以上の室外機と 1 つ以上の室内機によって構成される．この「1 つ以上の」というインスタンス多重度は，それぞれの VClassifer ノードに記載の上限値と下限値によって示される．

構造可変性モデルは，複合システムを構成するサブシステムの組み合わせ方に関して，サブシステムの種類とインスタンス数の可変性のみを表現し，サブシステム可変性モデルで示される特徴に基づくサブシステムの組み合わせ方は扱わない．例えば，室外機と室内機の両方のサブシステム可変性モデルには，「通信プロトコル (COM)」の Choice ノードの下にオルタナティブ分解で「OLD」と「NEW」が存在する．そして，

「OLD] を持つ室外機と「NEW」を持つ室内機が接続された場合，複合システムとして通信できない組み合わせのため，サブシステムの組み合わせとして不適当である．このようなサブシステム可変性モデルで示される特徴に基づくサブシステムの組み合わせの制約は，システム可変性モデルとそれらとサブシステム可変性モデルの間の関係を定義することで表現する．通信に関する例では，表 4.1 に示すように，通信可能な状況を表すシステム可変性モデルの Choice ノード「Protocol Compliance」を導入し，そのノードが満たされるサブシステム可変性モデルの組み合わせを表現している．

システム可変性モデルも構造可変性モデルも，文法としては同じ CVL を用いているため，これらを 1 つのモデルに結合することは原理的には可能である．システム可変性モデルは，サブシステムから完全に独立し，システム全体の視点で自由にモデリングできる．一方で，構造可変性モデルは，厳格に 3 レベルに則る必要があり，複合システムを構成するサブシステムの種類と許容されるインスタンス多重度を示す．これらの可変性を 1 つのモデルに結合してしまうと，作成者がこれらの可変性モデルの区別がつかなかったり，3 レベルルールに違反して矛盾が生じたりと，誤りを生じやすくなる．

4.1.4 複合システム向け可変性モデルの全体像

複合システム向けの可変性モデルは，システム可変性モデルと構造可変性モデル，そして複合システムを構成するサブシステムのサブシステム可変性モデルがある．複合システムの具体的な製品は，これら 3 種類の可変性が完全に束縛されている必要がある．図 4.3 に，これらの可変性モデルの束縛とその関係性を示す．図 4.3 には，「サブシステム (Individual subsystem)」と「構造」と「システム」の 3 つのレーンがある．それぞれのレーンの上部は可変性を示すドメインエンジニアリング領域であり，下部は具体的な製品を示すアプリケーションエンジニアリング領域である．上部のモデルが束縛され下部に示されるという構造となっている．

サブシステムレーンの上部は，図 2.7 と図 2.8 に対応しており，下部は表 2.2 と表 2.3 に対応している．サブシステムの可変性が束縛されリゾリューションモデルになると，具体的なサブシステム製品を示す．

構造可変性モデルとシステム可変性モデルも束縛される．構造のレーンの上部では図 4.2 の構造可変性モデルが記載されている．この可変性モデルの束縛は，複合システムの構成要素として，具体的なサブシステム製品を選択することを意味する．本論文では，この構造可変性モデルに対するリゾリューションモデルのことを，システム構成モデルと呼ぶ．図 4.3 の例では，2 台の Outer-STD と 4 台の Inner-HIGH が組み合わさって，一つの複合システムを構成していることを示している．

三つ目のシステムレーンでは，上部にシステム可変性モデル (図 4.1) があり，この可変性を束縛することで，システムリゾリューションモデルが得られる．システムリゾリューションモデルは，システム可変性モデルの束縛であると同時に，構造可変性モデルの束縛であるシステム構成モデルによって示されたサブシステム製品の組み合わせによって実現されるものを示している．図 4.3 のシステムリゾリューションモデルの例では，冷媒提供能力値 (ProvidingHP) が 200 という値になっている．これは，室外機 Outer-STD が冷媒提供能力値 100 を持ち，この室外機が 2 台構成となることで実現されている．また，同様にして冷媒消費能力値 (ConsumingHP) も 200 に束縛されているが，こちらは 50 の値を持つ Inner-HIGH が 4 台構成で実現している．そして，この冷媒提供能力値と冷媒消費能力値の合計値は，2.2.2 節で述べた Constraint 3 を満たしており，このシステム構成は有効なものである．

ここまでで述べたように，複合システムの可変性モデリングには，システム可変性と構造可変性とサブシステム可変性が含まれる．これら 3 つの可変性モデルは個別に作成されるが，図 4.3 で示したように，これらのモデルの要素間に関係が生じる．この関係に関しては，後で提案する関係記述によって形式的に定義される．

この節では，複合システムの例として，サブシステムがこれ以上分解されないシンプルな可変性を用いて説明した．一方で，図 2.5 に示した通り，サブシステムもさらにサブサブシステムに分解できるような多段

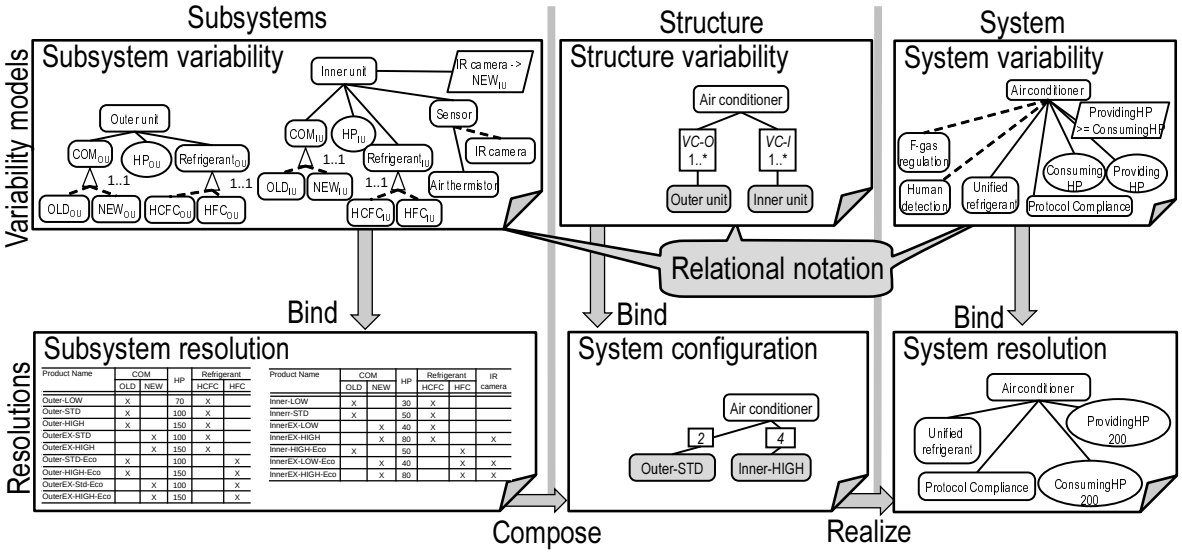

図 4.3: 複合システム向け可変性モデルの束縛

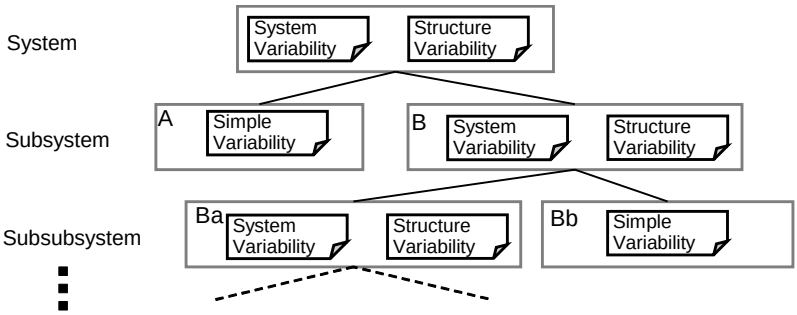

図 4.4: 多段構造におけるシステム可変性モデルと構造可変性モデル

構造がありうる．多段構造となった場合の例を，図 4.4 に示す．最上位の複合システムにおける構造可変性モデルは，サブシステムである *A* と *B* が何台接続されるかの可変性を示す．そして，最上位の複合システムにおけるシステム可変性モデルは，サブシステム *A* の可変性モデルと，複合システムでもあるサブシステム *B* のシステム可変性モデルを，組み合わせた結果生じる可変性を記述する．複合システムでもあるサブシステムの可変性モデルでは，より上位に対して自身の機能を示すためにシステム可変性モデルを提供し，自身の下に位置づけられている構成要素がどう組み合わさるのかを示す構造可変性モデルを上位に対して隠ぺいする．

この多段構成によって，複雑な複合システムも表現可能になる．例えば，サーバ・クライアント型の複合システムが，さらに複数接続されて大きな複合システムを形成する場合がある．このような場合，サーバとクライアントはそれぞれに，図 4.4 におけるサブサブシステム (Subsubsystem) の層にて可変性モデルを記述し，サーバ・クライアントが組み合わさった複合システムはサブシステム (Subsystem) の層に記述することで，全体としての大きな複合システムを表現できる．

4.2 モデル間の関係記述とピンサームーブメントアプローチ

4.1.2 節で述べたように，システム可変性モデルの束縛は，複合システムを構成するサブシステムのリゾリューションモデルの組み合わせによって定まる．そのため，これらのシステム可変性モデルとサブシステム可変性モデルをつなぐ記述が必要である．そして，2.3.1 節で述べたように，複合システムの要求や制約をシステム可変性モデルに記載する場合，以下の曖昧さが生じる．

- システム可変性が，どのサブシステム可変性と関係するのかが曖昧
- システム可変性がサブシステム可変性と関係する時，その間にあるインスタンス多重度と，その集約方法が曖昧

本節では，曖昧さを解決するために，複合システム向けの可変性モデルにおいて，システム可変性モデルとサブシステム可変性モデル間の関係記述を定義し，インスタンス多重度を識別して適切な対処を行う手法を提案する。この手法をここではピンサームーブメントアプローチと呼ぶ。「システム可変性が，どのサブシステム可変性と関係するのかが曖昧」に関しては，提案する関係記述において可変性の分析者が，システム可変性モデルの各ノードに対して関係するサブシステム可変性モデルのノードを選択することで解決する．システム可変性モデルのノードに対してサブシステム可変性モデルのノードを選択するため，関係記述においてシステム可変性モデル側をソースノード，サブシステム可変性モデル側をターゲットノードとし，記述する関係は方向を持つ．「システム可変性がサブシステム可変性と関係する時，その間にあるインスタンス多重度と，その集約方法が曖昧」に関しては，ピンサームーブメントアプローチによって自動的に集約するべきインスタンス多重度が識別され，分析者が集約方法を付与することで解決する．

2.3.1 節で述べたシステム全体の冷媒提供能力に関していうと，まず，図 4.1 の「Providing HP」がソースノードとなる．関係するサブシステム可変性モデルの候補として，室内機の馬力を示す「HP_{IU}」や室外機の馬力を示す「HP_{OU}」などがあり得る．システム可変性モデルの分析者は，システム全体の冷媒提供能力という意味を考慮し，室外機側の「HP_{OU}」を選択しターゲットノードとする．これにより，システム可変性モデルの「Providing HP」は，室外機側のサブシステム可変性と関係することが明示化される．そして，システム可変性モデルのターゲットノード「Providing HP」はシステム全体で 1 インスタンスしか生じない．一方の，室外機サブシステム可変性モデルにおけるターゲットノード「HP_{OU}」は，室外機 1 台に対して 1 インスタンス生じており，システム全体では室外機の台数分のインスタンス数が生じる．そのため関係記述には，この室外機の台数を表すインスタンス多重度を識別し，そのインスタンス多重度に対してどのような集約方法が必要かを指定する必要がある．この例においては，VClassifier ノード「*VC-O*」が室外機の台数を表し，このインスタンス多重度において集約方法として合計を付与する．その結果，「システム全体の冷媒提供能力」を示す「Providing HP」は，室外機のサブシステム可変性モデルの「HP_{OU}」の，VClassifier ノード「*VC-O*」が示す室外機の台数分の合計であると明示でき，曖昧さがなくなる．

このシステム全体での冷媒提供能力「Providing HP」についての関係は，比較的シンプルであった．ここで，サブシステム可変性モデルが複雑になると例として，室外機のサブシステム可変性モデルを変更した例を図 4.5 に示す．2.2.2 節の図 2.7 との違いは，「HP_{OU}」が示す冷媒提供能力のノードが VClassifier ノード「*VC-Compressor*」にぶら下がっていることである．VClassifier ノード「*VC-Compressor*」は，室外機 1 台の中に，1 台もしくは 2 台のコンプレッサが存在可能であり，それぞれに馬力を示す Variable ノード「HP_{OU}」を持てることを表している．この場合，図 4.1 の「Providing HP」のターゲットノードは「HP_{OU}」と同様である．そして，ソースノードが表現しているのはシステム全体であるのに対して，ターゲットノードは複数生じる室外機の中に，さらに複数生じるコンプレッサの 1 台に関する馬力「HP_{OU}」を示している．そのため，識別すべきインスタンス多重度は VClassifier ノード「*VC-O*」と VClassifier ノード「*VC-Compressor*」

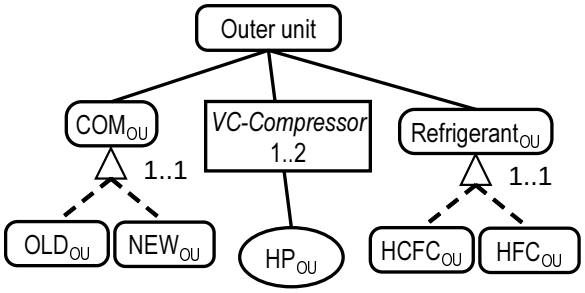

図 4.5: 室外機サブシステム可変性モデルの別の例

の 2 つになり，それぞれのインスタンス多重度に対して合計の集約方法を付与することで，関係が正確に記述できる．このインスタンス多重度の識別で誤った VClassifier ノードを識別したり，意図しない集約したりすると，適切なシステム可変性モデルの表現ができない．このような関係を，誤りのないように記述する記法が必要となる．

関係記述は，構造可変性モデルと関連するサブシステム可変性モデルとシステム可変性モデルの，束縛時の関係性を表す記述である．関係記述の形式的な定義に関しては 4.2.4 節にて後述する．この関係記述では，どのサブシステムの組み合わせの時にシステム可変性モデル上の要求や制約が満たされるかを表現する．さらに，この関係記述は，システム可変性モデルを介して，サブシステムを横断する互換性も表現できる．

関係記述を得るための手順を以下に示す．

(1) 分析者による，ソースノードに対するターゲットノード選択.
(2) ピンサームーブメントアプローチによる，インスタンス多重度識別.
(3) 分析者による，識別したインスタンス多重度への集約演算子付与.

本節の以降では，上記手順の詳細について述べ，その後に関係記述の言語定義を導入する．

4.2.1　ターゲットノード選択

複合システムの可変性における関係は，システム可変性モデルからサブシステム可変性モデルへと張られる．それは，どのサブシステムの可変性束縛の組み合わせによってシステム可変性モデルにおける可変性が定まるかを関係で表現するためである．そのため，システム可変性モデルの分析者が，システム可変性モデルの各ノードに対して，サブシステム可変性モデルのノードをターゲットノードとして手動で指定する．言い換えると，システム可変性モデルの各ノードの実現のためには，どのサブシステム可変性モデルのノードが関連するかを選ぶものである．システム可変性モデル側のソースノードに対して，複数のターゲットノードを指定できる．

図 4.6 は，例として，システム可変性モデルにおける「ProvidingHP」ノードに関する関係を示す．左側の木構造は，図 4.1 に示すシステム可変性モデルからの抜粋である．右上の木構造は，図 4.2 に示す構造可変性モデルである．最後に右下の構造は，図 2.7 に示す室外機のサブシステム可変性モデルの抜粋である．図 4.6 では，「ProvidingHP」は，システム可変性モデル内のソースノードであり，室外機のサブシステム可変性モデルにおける「HP_{OU}」ノードと関係があることを黒の矢印で示している．これは，「ProvidingHP」が，「HP_{OU}」の値の合計値を意味するからであり，ターゲットノードとして識別されていることを示す．

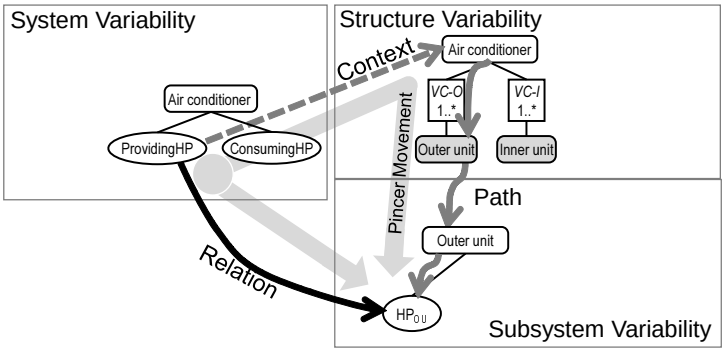

図 4.6: 「ProvidingHP」におけるピンサームーブメント

4.2.2 ピンサームーブメントアプローチによるインスタンス多重度識別

インスタンス多重度を識別するピンサームーブメントアプローチについて述べる．ピンサームーブメントのために，コンテキスト (Context) とパス (Path) という概念を導入する．関係記述において，ソースノードからターゲットノードを直接指し示すのに加え，コンテキストとパスを経由してターゲットノードを指し示すことで，挟み込むようにインスタンス多重度を識別するため，ピンサームーブメントアプローチと呼称する．

コンテキストを以下のように定義する．

Definition 15（コンテキスト (Context)） システム可変性モデル M におけるノード n に対して，構造可変性モデルあるいはサブシステム可変性モデル M' において同じインスタンスを示すノード n' を指すポインタを，n の M' におけるコンテキストと呼び，ノード n と n' の対 $C = \langle n, n' \rangle$ で表す．

例えば，システム可変性モデルのルートノードのコンテキストは，構造可変性モデルのルートノードと対となる．これは，両ルートノードが，システム全体を一つとして捉えており，同じインスタンスを示している．そして，システム可変性モデルにおける Choice ノードと Variable ノードは，そのノードに分解される際にインスタンス多重度を生じさせないため，親ノードに付与されたコンテキストを自動的に継承する．一方で，システム可変性モデルにおける VClassifier ノードは，そのノードに分解された際に，そのノード以下のサブツリーのインスタンス数の可変性を表し，インスタンス多重度が生じる．そのため，VClassifier ノードのコンテキストは，構造可変性モデルもしくはサブシステム可変性モデルの中の対応するインスタンス数の可変性を示す VClassifier ノードを対とする．

図 4.6 の「ProvidingHP」の例では，「ProvidingHP」のコンテキストは灰色の点線矢印で示される．システム可変性モデルの「ProvidingHP」ノードとルートノードの間に VClassifier ノードはないため，構造可変性モデルのルートノードを対とする．図 4.1 のシステム可変性モデルには VClassifier ノードがないため，システム可変性モデルのすべてのノードのコンテキストは，構造可変性モデルのルートノードと対となる．これはシステム可変性モデルの全てのノードが，システム全体を一つのインスタンスと捉えて可変性を表現していることを意味する．

システム可変性モデルの中に VClassifier ノードが存在する場合の例を，図 4.7 で示す．システム可変性モデルには，VClassifier ノード「*VC-A*」と VClassifier ノード「*VC-B*」がある．まずノード X のコンテキストは，ルートノードからコンテキストの対を継承して構造可変性モデルのルートノードを対として，$\langle X, SoS \rangle$ となる．ここで，VClassifier ノード「*VC-A*」のコンテキストは，以下の 3 通りから選択できる．

$\langle$*VC-A*, $SoS\rangle$ VClassifier ノード「*VC-A*」で示されるサブツリーのインスタンス数の可変性は，複合シ

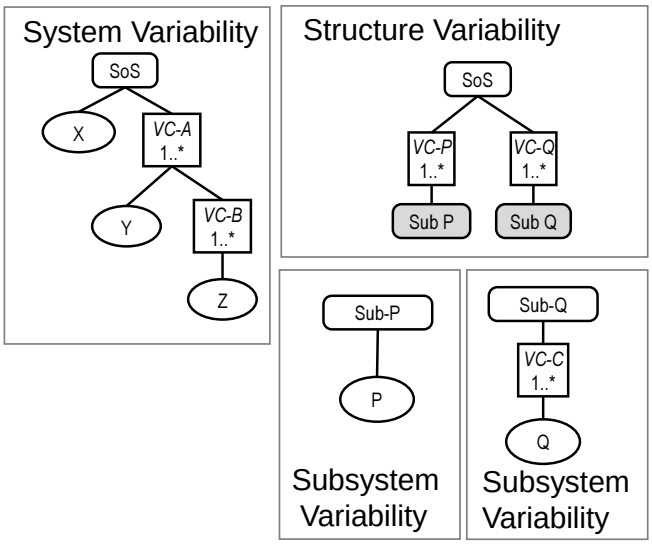

図 4.7: コンテキストとパスの複雑な例

ステムにおけるサブシステムのインスタンス数の可変性とは関係が生じないため，構造可変性モデルのルートノードと対となる．

⟨*VC-A*, *VC-P*⟩ VClassifier ノード「*VC-A*」で示されるサブツリーのインスタンス数の可変性は，サブシステム「*Sub-P*」のインスタンスと対応するため，VClassifier ノード「*VC-P*」と対となる．

⟨*VC-A*, *VC-Q*⟩ VClassifier ノード「*VC-A*」で示されるサブツリーのインスタンス数の可変性は，サブシステム「*Sub-Q*」のインスタンスと対応するため，VClassifier ノード「*VC-Q*」と対となる．

このコンテキストの選択は，システム可変性モデルを記述する際の，各ノードの意図を考慮して可変性の分析者が付与する．生じうるコンテキストのパターンは、表 4.2 の 2 列目と 3 列目に示す．

次に，パスを以下のように定義する．

Definition 16（パス (Path)）　システム可変性モデルのソースノード n_0 において，そのコンテキストを示す構造可変性モデルもしくはサブシステム可変性モデルのノード n_1 からターゲットノード n_m に至るエッジの列「$\langle\langle n_1, n_2\rangle, ..., \langle n_{m-1}, n_m\rangle\rangle$」をパスと呼ぶ．各エッジは構造可変性モデルおよびサブシステム可変性モデルの木構造の分解を示す枝である．パスのエッジ列が構造可変性モデルの葉ノードに到達した場合，その葉ノードはサブシステム可変性モデルへの参照を示す CVSpec ノードとなっているため，そのサブシステム可変性モデルのルートノードへと繋がり，ターゲットノード n_m まで辿る．

パスは，コンテキストとターゲットノードが決定されると，一意に定まる．これは，構造可変性モデルとサブシステム可変性モデルが木構造であることと，構造可変性モデルにあるサブシステムへの参照が，サブシステムごとに 1 つまでに限定されていることで実現できる．システム可変性モデルのノードに対するコンテキストは，構造可変性モデルあるいはサブシステム可変性モデル上の同じインスタンスを表すノードを指し示し，ターゲットノードは関係として表したいサブシステム可変性モデルのノードを意味する．つまり，構造可変性モデルの葉ノードである CVSpec ノードを参照先のサブシステム可変性モデルのルートノードと便宜的に接続したとして，ターゲットノードはコンテキストが指し示すノードのサブツリー内に存在するため，その間のパスは一意に定まる．ここで，コンテキストが指し示すノードのサブツリー内にターゲットノードが存在しない場合，システム可変性モデルのノードが表現したいインスタンスとは関係ないノードをターゲットとしていることになるため，システム可変性モデルを修正する必要が生じる．

パスの上に現れる VClassifier ノードは，その関係における識別すべきインスタンス多重度を示す．つまり，コンテキストとは，システム可変性モデルに生じるインスタンス数の可変性を，サブシステムのインスタンス数の可変性やサブシステム可変性モデル内に生じるインスタンス数の可変性と対応付けるものである．そしてパスを辿ることで，そのマッピングにおいて，システム可変性モデルとサブシステム可変性モデルのノードの間の関係を定義した際に，両ノードの間にインスタンス数の可変性があるかを検出できる．

図 4.6 の「ProvidingHP」の例では，構造可変性モデルのルートノードからパスが始まり，サブシステム可変性モデルの「HP_{OU}」までたどり着くパスを灰色の実線矢印で示している．結果として，VClassifier ノード「*VC-O*」が識別できる．この例を言い換えると，ソースノードである「ProvidingHP」は，複合システムに対して 1 ノードで変数の可変性を表しているのに対して，ターゲットノード「HP_{OU}」は，複合システムに対して室外機サブシステムの台数分のノードが生じている．そして，その室外機サブシステムの台数の可変性は，VClassifier ノード「*VC-O*」が表している．

図 4.7 の例における，コンテキストのシチュエーションごとの識別される VClassifier ノードを表 4.2 に示す．例えば No.1 から No.6 は，コンテキストが全て構造可変性モデルのルートノードを対とするため，システム可変性モデルのすべてのノードは，複数のサブシステムがシステムを構成した全体を 1 つと捉えて可変性を表現していることになる．そのため，ノード P へのパス上には VClassifier ノード「*VC-P*」があり，ノード Q へのパス上には VClassifier ノード「*VC-Q*」と VClassifier ノード「*VC-C*」がある．No.19 から No.24 の行は，コンテキストとして $\langle$*VC-A*, *VC-Q*$\rangle$ と $\langle$*VC-B*, *VC-C*$\rangle$ が指定されたケースを示している．この時，ノード Y は，サブシステム「*Sub-Q*」のインスタンスと対応付いているため，ノード Q に対して，VClassifier ノード「*VC-Q*」は必要とせず，VClassifier ノード「*VC-C*」のみが識別される．ノード Z は，サブシステム可変性モデルの VClassifier ノード「*VC-C*」のインスタンスと対応するため，ノード Q に対して識別するべき VClassifier ノードは生じない．No.21 と No.23 は，コンテキスト $\langle$*VC-A*, *VC-Q*$\rangle$ と $\langle$*VC-B*, *VC-C*$\rangle$ において，ノード Y やノード Z はノード P と関係を定義できないことを示している．これは，コンテキストによって VClassifier ノード「*VC-A*」のインスタンスが VClassifier ノード「*VC-Q*」のインスタンスと対応することを示しているため，その対応に違反した関係になるからである．

4.2.3 集約演算子付与

関係記述を記述するためには，パス上にある VClassifier ノードを識別したのち，その VClassifier ノードにて生じるインスタンス多重度をどう集約させるのかを決める必要がある．図 4.6 の「ProvidingHP」の例では，「ProvidingHP」は，複合システム全体における冷媒提供能力値を意味しており，それはシステムを構成するすべての室外機の持つ冷媒提供能力値を足し合わせた数である．そのため，ターゲットノード「HP_{OU}」に対して合計を行った結果を「ProvidingHP」に代入するという集約を関係記述上に表現する必要がある．

CVL には，フィーチャを示す Choice ノードと，インスタンス多重度を示す VClassifier ノードと，変数を表す Variable ノードの 3 種類ある．ソースノードが Variable でターゲットノードも Variable であった場合，識別した VClassifier ノードには算術的な集約演算子 (count, sum, max など) が選択できる．つまり，ソースノード側の Variable が 1 つのインスタンスに対して，識別した VClassifier ノードの分のターゲット Variable ノードのインスタンスが生じているため，1 対多の関係になっている．互いが変数の可変性のため，合計をとったり，最大値をとったりという関係を示すために集約演算子を付与する．ソースノードが Choice でターゲットノードも Choice であった場合，Choice ノードは選択するかしないかの 2 択の可変性なので，存在限量子で集約できる．表 4.3 に，ソースノードとターゲットノードのノード種類の組み合わせに応じた集約演算子に関してまとめる．

図 4.6 の「ProvidingHP」の例では，VClassifier ノード「*VC-O*」が識別されており，ソースノードもターゲットノードも Variable である．そのため，室外機のサブシステム可変性モデルの Variable「HP_{OU}」の値を

表 4.2: 図 4.7 のコンテキストと識別される VClassifier ノード

No.	Context situation *VC-A*	Context situation *VC-B*	Source node	Target node	Identified VClassifier
1	⟨*VC-A*, *SoS*⟩	⟨*VC-B*, *SoS*⟩	X	P	*VC-P*
2			X	Q	*VC-Q*, *VC-C*
3			Y	P	*VC-P*
4			Y	Q	*VC-Q*, *VC-C*
5			Z	P	*VC-P*
6			Z	Q	*VC-Q*, *VC-C*
7	⟨*VC-A*, *VC-P*⟩	⟨*VC-B*, *VC-P*⟩	X	P	*VC-P*
8			X	Q	*VC-Q*, *VC-C*
9			Y	P	
10			Y	Q	N/A
11			Z	P	
12			Z	Q	N/A
13	⟨*VC-A*, *VC-Q*⟩	⟨*VC-B*, *VC-Q*⟩	X	P	*VC-P*
14			X	Q	*VC-Q*, *VC-C*
15			Y	P	N/A
16			Y	Q	*VC-C*
17			Z	P	N/A
18			Z	Q	*VC-C*
19	⟨*VC-A*, *VC-Q*⟩	⟨*VC-B*, *VC-C*⟩	X	P	*VC-P*
20			X	Q	*VC-Q*, *VC-C*
21			Y	P	N/A
22			Y	Q	*VC-C*
23			Z	P	N/A
24			Z	Q	

合計する SUM 演算子を *VC-O* に付与する．

ソースノードが Choice ノードもしくは Variable ノードの場合，パス上に現れるサブシステムのインスタンス多重度を関係記述は集約する．一方で，表 4.3 の下側 3 行で示される，ソースノードが VClassifier ノードの場合は，関係記述は集約を意味しない．システム可変性モデルにおいて VClassifier ノードが生じたとき，それは複合システム全体の視点においてインスタンス多重度が生じていることを意味する．そのため，システム可変性モデルの VClassifier ノードは，ターゲットノードに Choice ノードや Variable ノードを持たない．そして，システム可変性モデルの VClassifier ノードがターゲットノードに VClassifier ノードを持った場合，それはインスタンス多重度の観点で対応が取れたことを意味する．この対応関係に関してはすでにコンテキストとして導入済みである．

表 4.3: ノード種類ごとの集約演算子

System node type	Subsystem node type	Aggregation operator	Comment
Choice	Choice	∀	Indicates that all subsystems are required
		∃	Indicates that at least one subsystem is required
	VClassifier	N/A	
	Variable	N/A	
Variable	Choice	COUNT	Number of choices selected
	VClassifier	COUNT	Number of instances that exist
	Variable	SUM	Sum of subsystem variables
		MAX	Maximum number of subsystem variables
		MIN	Minimum number of subsystem variables
		AVERAGE	Average number of subsystem variables
VClassifier	Choice	N/A	
	VClassifier	CONTEXT	Indicates correspondence of quantity
	Variable	N/A	

4.2.4 関係記述の定義言語

図 4.8 に，提案する関係記述の言語を Backus-Nour Form(BNF) で示す．この文法における記号に関して以下にまとめる．

⟨*systemVSpec*⟩ システム可変性モデルにおけるソースノード．

⟨*choiceClause*⟩ 関係記述を示す節の 1 つ．ターゲットノードが Choice ノードの場合に用いる．複数のターゲットノードが生じる場合は，∧ や ∨ の論理演算子で組み合わせ出来る．

⟨*variableClause*⟩ 関係記述を示す節の 1 つ．ターゲットノードが Variable ノードの場合に用いる．複数のターゲットノードが生じる場合は，SUM や MAX などの集約演算子で組み合わせることができる．

⟨*vclassifierClause*⟩ 関係記述を示す節の 1 つ．ターゲットノードが VClassifier ノードの場合に用いる．

⟨*subsystemChoice*⟩ ターゲットノードとしてのサブシステム可変性モデルにおける Choice ノード．

⟨*subsystemVSpec*⟩ ターゲットノードとしてのサブシステム可変性モデルにおける種類は問わないノード．

⟨*relationalNotation*⟩ ::= ⟨*systemVSpec*⟩ = ⟨*clause*⟩*

⟨*clause*⟩ ::= ⟨*choiceClause*⟩ | ⟨*variableClause*⟩ | ⟨*vclassifierClause*⟩

⟨*choiceClause*⟩ ::= ⟨*subsystemChoice*⟩
| ⟨*quantifier*⟩ $_{\langle vclassifier\rangle}$ (⟨*choiceClause*⟩)
| ⟨*choiceClause*⟩ ∧ ⟨*choiceClause*⟩
| ⟨*choiceClause*⟩ ∨ ⟨*choiceClause*⟩

⟨*variableClause*⟩ ::= ⟨*subsystemVSpec*⟩
| ⟨*aggregation*⟩ $_{\langle vclassifier\rangle}$ (⟨*variableClause*⟩)
| ⟨*aggregation*⟩ (⟨*variableClause*⟩+)

⟨*vclassifierClause*⟩ ::= CONTEXT (⟨*subsystemVclassifier*⟩)

⟨*quantifier*⟩ ::= ∀ | ∃

⟨*aggregation*⟩ ::= COUNT | SUM | MAX | MIN | AVERAGE

図 4.8: 関係記述の文法

⟨*vclassifier*⟩　パス上に識別された VClassifier ノード．表 4.3 に示す集約演算子が付与される．

本節では関係記述の文法を紹介したが，この文法は我々が開発したモデリングツールのメタモデルとして用いられており，モデリングツール自体は木構造をグラフィカルに操作する UI を提供する．そのため，モデリングツールのユーザは，この文法に則った関係記述をテキストで記述する必要はない．ソースノードとターゲットノードをユーザが指定すると，ツールが自動でインスタンス多重度を識別し，付与するべき集約演算子の候補を提示する．ユーザは提示された演算子の中から，意味的に適切なものを選ぶだけで関係記述が記述できる．モデリングツールに関しては，4.3 節にて詳細を述べる．この論文では，BNF によるテキストの表記を用いることで，可変性モデルの関係記述を書き下すことが容易になるため用いている．4.2.5 節で後述する仮想の業務用空調機における関係記述の例や，4.4 節で述べる評価実験にて，上記 BNF に基づいたテキスト表現を用いる．

図 4.6 の「ProvidingHP」の例では，定義言語を用いると以下のような関係記述になる．

$$\text{ProvidingHP} = \text{SUM}_{VC\text{-}O}(\text{HP}_{\text{OU}})$$

提案する関係記述は，標準化された記法である OCL [18] でも記述可能である．OCL は汎用的な記法であるのに対して，提案する関係記述はインスタンス多重度を識別することや，その影響を記述することに特化している．提案する関係記述はドメイン特有であり，様々な種類の可変性モデルを利用し，ユーザに適切なガイダンスと関係表現の正確さを提供できる．例えば，「ProvidingHP」の例を OCL で書くと以下のようになる．

context ac:AirConditioner **inv**:

ProvidingHP = ac.*)*VC-O* −>collect(ou:OuterUnit | ou.HP*)$_{\text{OU}}$−>sum()

この記述の「context」は，構造可変性モデルのルートノードを指し示しており，提案する関係記述のコンテキストと役割は同一である．*VC-O* は，OCL における *Bag* とみなすことができる．そして，「ac」と「ou」は，それぞれに業務用空調機のインスタンス，サブシステム室外機インスタンスの代表を示している．したがって，OCL は本研究の目的のために十分な記述機能を備えているが，関係記述よりも長く，わかりにくい表現になる．また，識別すべきインスタンス多重度は，ユーザが手動で識別する必要が生じるため，誤りを生じやすい．

表 4.4: 業務用空調機の関係記述

	Source	Target	VClassifier	Relational definition
Constraint 1	Unified Refrigerant	$HCFC_{OU}$	*VC-O*	$(\forall_{VC\text{-}O}(HCFC_{OU}) \wedge$
		HFC_{OU}	*VC-O*	$\forall_{VC\text{-}I}(HCFC_{IU})) \vee$
		$HCFC_{IU}$	*VC-I*	$(\forall_{VC\text{-}O}(HFC_{OU}) \wedge$
		HFC_{IU}	*VC-I*	$\forall_{VC\text{-}I}(HFC_{IU}))$
Constraint 2	Protocol Compliance	NEW_{OU}	*VC-O*	$\forall_{VC\text{-}O}(NEW_{OU}) \vee$
		OLD_{OU}	*VC-O*	$(\forall_{VC\text{-}O}(OLD_{OU}) \wedge$
		OLD_{IU}	*VC-I*	$\forall_{VC\text{-}I}(OLD_{IU}))$
Constraint 3	ProvidingHP	HP_{OU}	*VC-O*	$\mathsf{SUM}_{VC\text{-}O}(HP_{OU})$
	ConsumingHP	HP_{IU}	*VC-I*	$\mathsf{SUM}_{VC\text{-}I}(HP_{IU})$
Requirement 1	F-gas Regulation	HFC_{OU}	*VC-O*	$\forall_{VC\text{-}O}(HFC_{OU}) \wedge$
		HFC_{IU}	*VC-I*	$\forall_{VC\text{-}I}(HFC_{IU})$
Requirement 2	Human Detection	NEW_{OU}	*VC-O*	$\forall_{VC\text{-}O}(NEW_{OU}) \wedge$
		IR camera	*VC-I*	$\forall_{VC\text{-}I}$(IR camera)

4.2.5 業務用空調機における関係記述

本節では，2.2.2 節で述べた業務用空調機の例における要求と制約のすべてについて，関係記述を適用した結果について述べる．その結果を，表 4.4 に示す．最初の列は，2.2.2 節で述べた要求と制約を示す．2 列目は，図 4.1 で示すシステム可変性モデルにおけるソースノードを記述する．3 列目は，示したい関係性の意味によって探索されて指定されたターゲットノードを示す．4.2 節で述べたように，コンテキストからターゲットノードを辿るパス上に存在する VClassifier ノードを，インスタンス多重度として識別する．4 列目は，その識別した VClassifier ノードを示している．そして，最後の列「Relational definition」において，識別した VClassifier ノードに対する集約演算子を付与して，複数のターゲットノードがある場合は結合して関係記述を完成させる．

表 4.4 において，最初の列の 1 行に対して 2 列目が複数行になっているのは，システム全体の制約や要求を，システム可変性モデルのノード複数で表現する場合を示している．2 列目 1 行に対して，3 列目以降が複数行になっているのは，システム可変性モデルのソースノードに対して，サブシステム可変性モデル側のターゲットノードが複数生じていることを示す．最後の列「Relational definition」に関しては，複数生じたターゲットノードのそれぞれにインスタンス多重度とその集約演算子を付与し，⟨*choiceClause*⟩ や ⟨*variableClause*⟩ で定義されている方法で組み合わせた結果を示している．最後の列「Relational definition」は，3 列目と 4 列目で示している複数行に対して，貫通して記述されているものの，横方向にターゲットノードを対応させて記述してある．

システム可変性モデルは，システム全体の視点で分析されているため，サブシステム可変性モデルに用いられる表現とは異なる表現が記載される．Constraint 1 と Requirement 1 は，システム全体で用いられる冷媒に関するものである．Constraint 1 は，使用する冷媒の種類が統一されていないと空調機能が動作しないという基本的な制約を表現しており，サブシステム可変性モデルに記載されている HCFC または HFC のいずれかが統一して使用されることを示す．次に，Requirement 1 の F ガス規制は，規制に準拠するためには環境

負荷を低減できる HFC を使用する必要があることを示している．そのため，Constraint 1 はシステムを動作させるために必須であるのに対し，Requirement 1 は，規制に準拠するかしないかによって選択可能となる．サブシステム可変性モデルでは，具体的な冷媒名で可変性が記載されるのに対して，システム可変性モデルでは，システム全体の視点で抽象的な記述になっている．

提案する関係記述では，サブシステム間の互換性も表現できる．表 4.4 の 2 行目は，通信プロトコルに関連する Constraint 2 を示している．2.2.2 節で述べた通り，室外機側の新プロトコル側のみ後方互換性を持ち，室内機における旧プロトコルを実装する機器と通信が可能である．図 4.1 の Choice ノード「Protocol compliance」は，必須分解されており，関係記述の列を見ると 2 つのケースがある．まず一つ目のケースでは，室外機が新プロトコルを実装している場合，複合システムは室内機がどちらの通信プロトコルを持ったとしても通信が可能であることを示す．2 つ目のケースは，室外が機旧プロトコルを実装している場合，適切に通信するためには，室内機は全て旧プロトコルを実装する必要があることを示す．これらのケースは，選言標準形にて結合される．このようにして，後方互換性などの複雑な条件が，関係記述によって表現可能となる．

表 4.4 の 3 行目は，Constraint 3 を表している．4.2.1 節と 4.2.3 節で述べたように，システム可変性モデルの Variable ノード「冷媒提供能力値 (ProvidingHP)」は室外機サブシステム可変性モデルの Variable「HP_{OU}」の値を合計する関係である．同様にして，「冷媒消費能力値 (ConsumingHP)」は室内機サブシステム可変性モデルの Variable「HP_{IU}」の値を合計する関係である．Constraint 3 は，これらのノードを不等号式に用いて，冷媒消費能力値は冷媒提供能力値を超えてはならないという制約となる．この不等号式は，図 4.1 の平行四辺形によって記述される．システム可変性モデルにおける数量の定義を関係記述によって行うことで，数量の比較は CVL が提供する Constraint ノード (平行四辺形) によって，システム可変性モデルに閉じて記述できる．

サブシステム可変性モデルだけでは見えてこない関係性を，システム可変性モデルで可視化することができる．表 4.4 の最終行は，Requirement 2 を表している．これは，業務用空調機が人間検出機能を選択的に保有できるという要求である．室内機のサブシステム可変性モデルには，赤外線カメラを示す Choice ノード「IR camera」があり，これが人間検出機能の根幹をなす．そして，室内機のサブシステム可変性モデルには，そのモデル内の制約として，「IR camera」は「新プロトコル」を必要とする記述がある．一方で，室外機のサブシステム可変性モデルには，人間検出機能に係る直接的な記述はない．ここで，赤外線カメラのデータを通信するためには新プロトコルが必要となるため，室外機に人間検出機能に係る表現がない場合でも，室外機側のサブシステム可変性モデルの新プロトコル「NEW_{IU}」がターゲットノードとして表出する．

このようにして関係記述を適用することによって，システム可変性モデルとサブシステム可変性モデルの間をつなぎ，システム全体の要求や制約を表現することができる．特に，識別した VClassifier ノードに適切な集約演算子を付与することで，要求や制約を正確に表現できる．

4.3　可変性モデリングツール Configuration Tool using CVL(CT-CVL))

複合システム向け可変性として，分割したモデルを記述しその間の関係記述を定義するためのツール「Configuration Tool with CVL (CT-CVL)」を開発した．

図 4.9 に，CT-CVL のモデリング機能の概要を示す．モデリングツールには，サブシステム可変性モデルのエディタと，サブシステムリゾリューションモデルのエディタと，構造可変性モデルのエディタと，システム可変性モデルのエディタがある．システム可変性モデルエディタには，関係記述の定義機能も含まれる．CT-CVL は，Eclipse のプラグインとして動作する．

まず，サブシステム可変性エディタによってサブシステム可変性モデルを作成する．次に，サブシステムリゾリューションエディタは，そのサブシステム可変性モデルをベースとして，リゾリューションモデルを

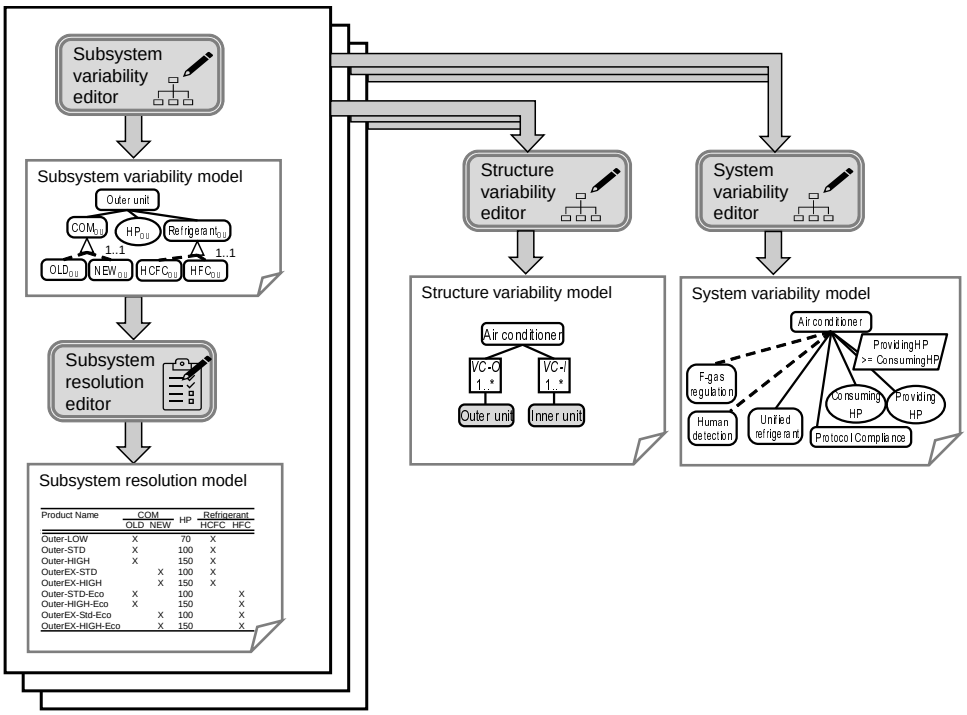

図 4.9: CT-CVL のモデリング機能概要

作成できる．これらのサブシステム可変性モデルとリゾリューションモデルは，サブシステムの種類ごと作成され，それらは構造可変性エディタとシステム可変性エディタが参照する．構造可変性エディタは構造可変性モデルを編集し，システム可変性エディタは，システム可変性モデルを編集できる．

以降では，それぞれのエディタに関して述べる．

4.3.1 サブシステム可変性エディタ

図 4.10 に，サブシステム可変性エディタのスクリーンショットを示す．左上部はファイルエクスプローラで Eclipse の機能である．中央上部がメインとなるエディタ部であり，右側にモデル編集用のパレットが用意されている．下部はエディタ部で選択中の要素のプロパティ表示部であり，左下はエディタ部のアウトラインを示している．

このエディタにおいて新規にサブシステム可変性モデルを作成すると，自動でルートノードが作成される．ルートノードは名前の変更がプロパティ表示部にて可能である．可変性を分解してノードを追加するには，パレットから VSpecChild を選択し，付与したいノードを選択すると，図 4.11 のように，どのノード種を追加するか選択できる．ノードの削除は，エディタ部のノードを右クリックして表示されるメニューから行う．必須分解と選択可能分解の指定は，プロパティ部の「Is Implied By Parent」属性を用い，この値が true であれば必須分解，false であれば選択分解を意味する．グループ多重度分解を指定する際は，ノードを追加した後に，パレットから「VSpecGroupMultiplicity」を選択しノードに付加する．また，エディタ部の何もないエリアを右クリックし，CustomLayout を選択すると，木構造を自動で再配置する．

平行四辺形で示される CVL モデルの中で閉じた制約は，パレットから OpaqueConstraint を選択する．この制約ノードは，Choice ノードや Variable ノードと異なり，木構造の外側に位置づけられる．そのため，どのノードに付与するかは，パレット部の OpaqueConstraintLink を用いてノードに結合することで表現する．作成した制約ノードをエディタ部で選択し，プロパティ部に表示される「Constraint」プロパティの値の右側のボタン「…」を押すと，図 4.12 のような OCL 編集ダイアログが表示される．このダイアログは，記入し

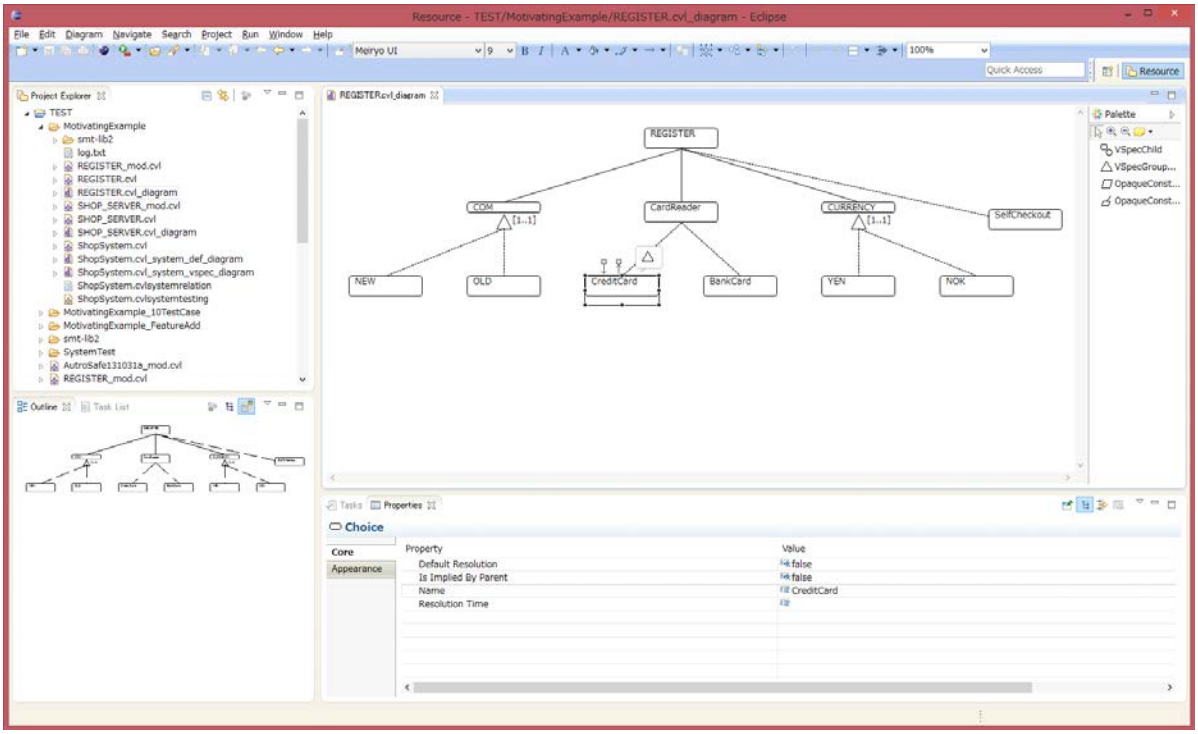

図 4.10: サブシステム可変性エディタ

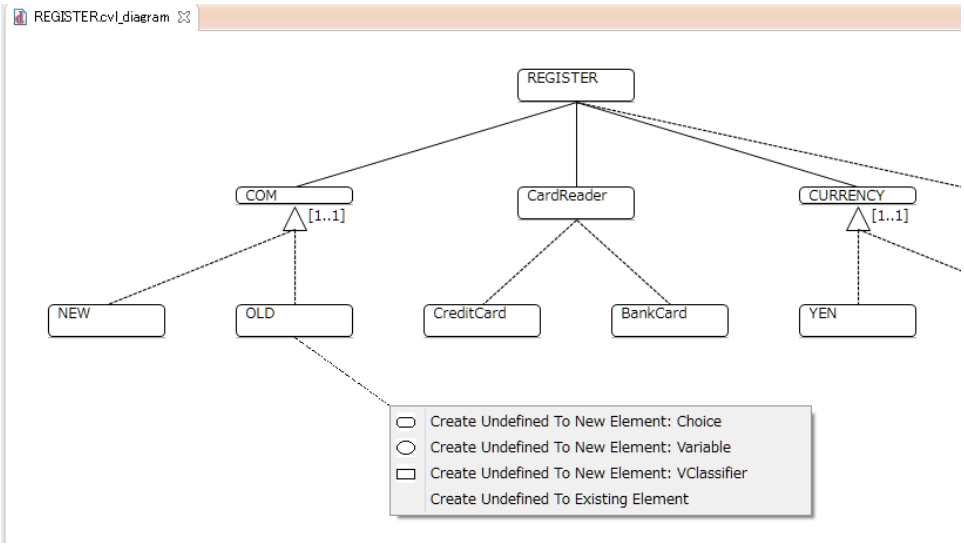

図 4.11: ノードの追加

た OCL の文法チェック機能と，エディタ部に作成したノード名などのシンボルを補完する機能 (Ctrl+Space を押下) を持つ．

4.3.2　リゾリューションエディタ

左上部はファイルエクスプローラから，作成したサブシステム可変性モデルを右クリックして，サブシステムリゾリューションエディタを開くことができる．図 4.13 に，サブシステム可変性エディタのスクリーンショットを示す．中央上部がメインとなるエディタ部であり，下部はエディタ部で選択中の要素のプロパ

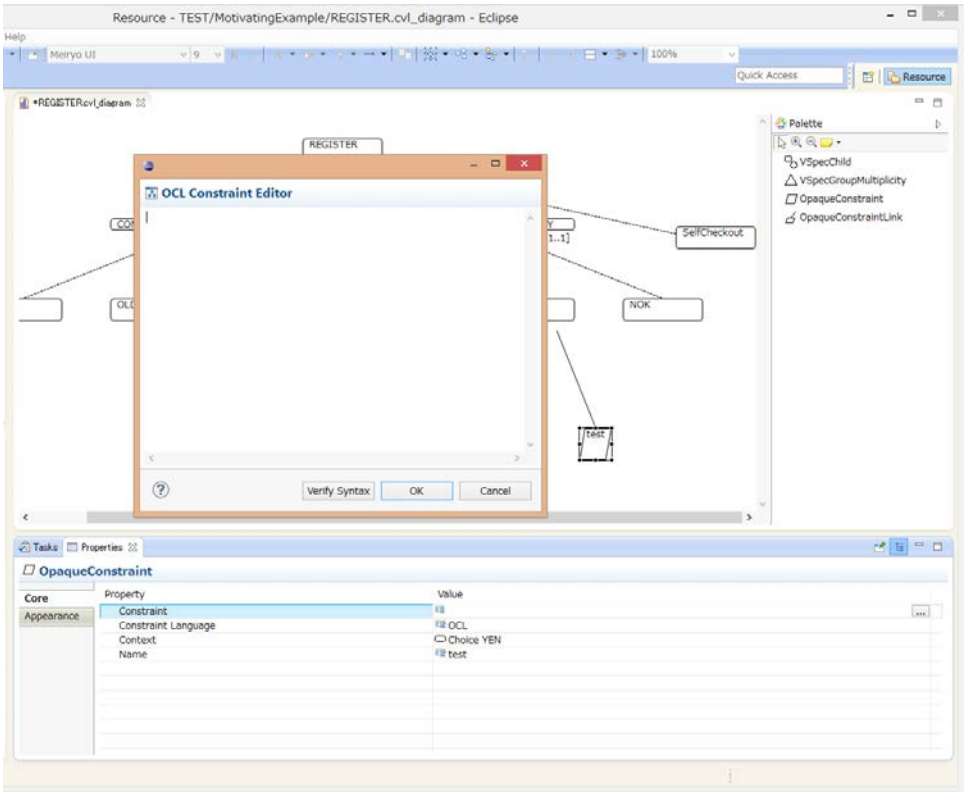

図 4.12: モデルに閉じた制約記述

ティ表示部である．エディタ部は，Eclipse のツリーコンポーネントを用いてリゾリューションモデルを操作する．サブシステム可変性エディタが可変性を表す木構造を 1 つ作成するのに対して，リゾリューションエディタでは，リゾリューションモデルを複数作成するため，同じ木構造を同一画面内で複数表現しやすいツリーコンポーネントを用いている．

サブシステムのリゾリューションモデルを作成するには，Configuration Unit を右クリックして表示される「Create Resolution」を実行する．そうすると，Configuration Unit の下に，サブシステム可変性エディタで定義したサブシステム可変性モデルと同じ構造を持つ新たなサブツリーが生成される．このサブツリーのルート部分のノードの Name プロパティには，このリゾリューションを説明する製品名などを付与する．このサブツリーにおいて，サブシステム可変性モデルで定義した可変性に対する束縛を記述できる．例えば Choice ノードに対して，それが選択可能分解であれば，Decision プロパティに true か false を付与して束縛を行う．Variable ノードに対しては，具体的な値を属性に付与できる．VClassifier ノードは，そのノードにぶら下がる可変性モデルのインスタンス数の可変性を表現する．そのため，この VClassifer ノードが示す可変性の束縛は，実際にそのインスタンスを作成することである．リゾリューションエディタでは，VClassifier ノードに対応するノードとして「Virtual VClassifier」というノードが生成されているため，このノードを右クリックし「Create Instance」を行うと，その「Virtual VClassifier」以下にインスタンスが追加される．

このリゾリューションエディタにおいて，Configuration Unit を右クリックして表示される「Check all product」を実行すると，サブシステム可変性モデルで定義した可変性を逸脱したリゾリューションが無いか自動検証ができる．グループ多重度の可変性に対して多く選択しすぎた場合や，VClassifier ノードに対するインスタンス数の超過や，制約として記述した OCL に違反していないかの検証を行う．

サブシステムリゾリューションエディタのエディタ部の下には，「Selection」と「Table」というタブが表示されている．先ほど述べたサブシステムリゾリューションエディタの編集画面は「Selection」であり，

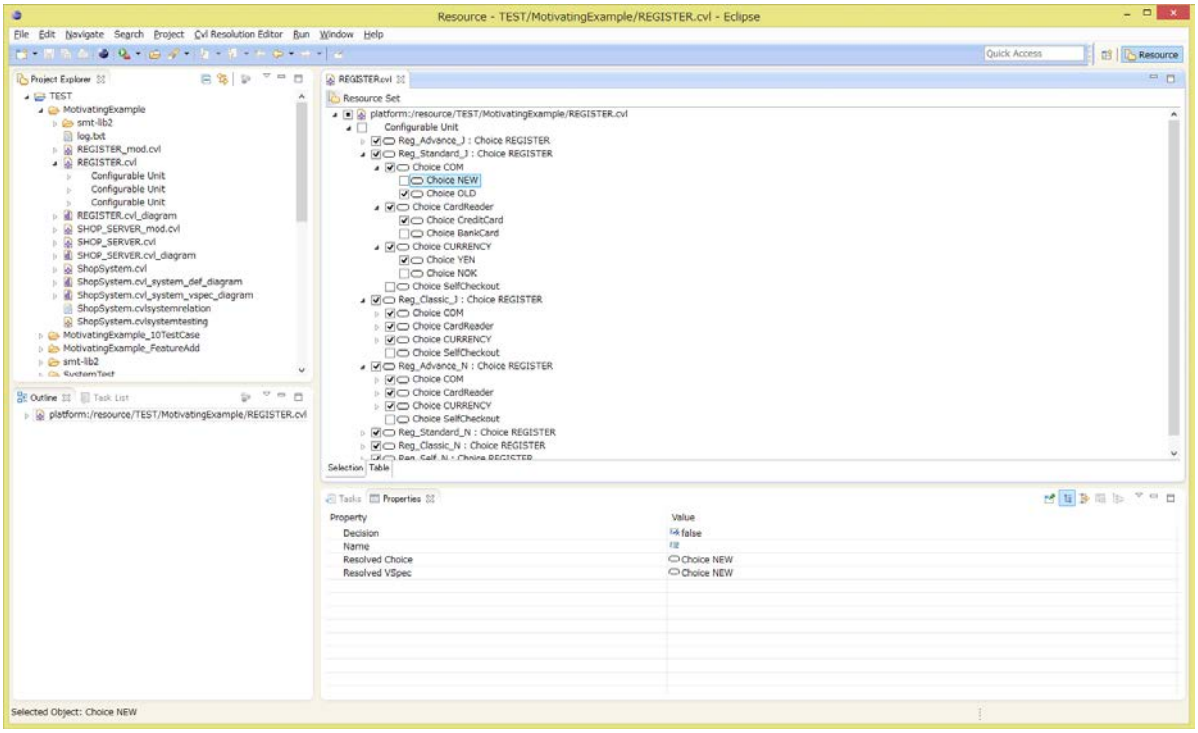

図 4.13: サブシステムリゾリューションエディタ

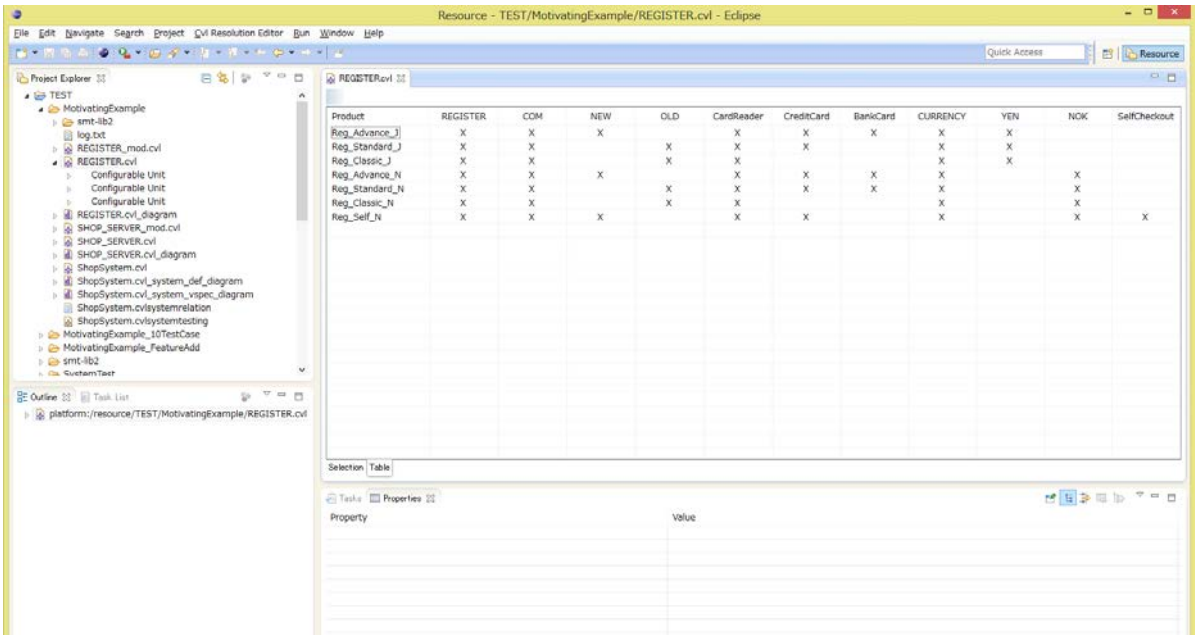

図 4.14: プロダクトマップビュー

「Table」を選ぶと図 4.14 に示すようなプロダクトマップビューが得られる．このビューと編集ビューを切り替えながら，必要なサブシステムリゾリューションモデルを構築する．

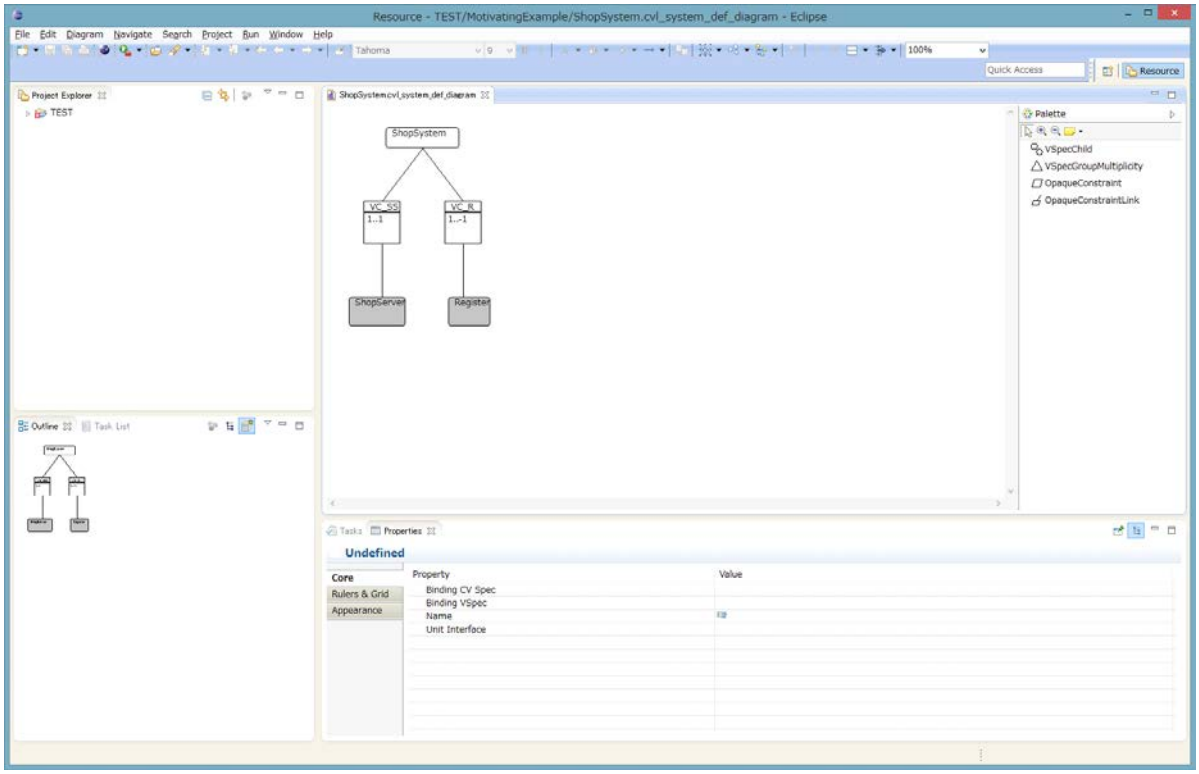

図 4.15: 構造可変性エディタ

4.3.3　構造可変性エディタ

構造可変性エディタのスクリーンショットを，図 4.15 に示す．基本的に操作はサブシステム可変性エディタと同様であるが，サブシステム可変性モデルを参照する CVSpec ノードを追加できることと，4.1.3 節で述べたように 3 つのレベルという強い記述上の制約があることが違いである．3 レベル目の CVSpec ノードにおいて，プロパティ部で参照先のサブシステム可変性モデルを指定することで，複合システムの構造を記述できる．

4.3.4　システム可変性エディタ

システム可変性エディタのスクリーンショットを，図 4.16 に示す．このエディタも，操作はサブシステム可変性エディタと同様である．サブシステム可変性エディタとの違いは，関係記述をシステム可変性モデルのノードに付与できることである．

まず，システム可変性モデルの中に VClassifier ノードがある場合には，そのノードにコンテキストを設定する．VClassifier ノードを選択し，プロパティ部に表示される Context プロパティの入力欄の右側のボタンをクリックすると，図 4.17 のダイアログが表示される．このダイアログには，構造可変性モデルとサブシステム可変性モデルに生じたすべての VClassifier ノードが一覧で並ぶため，選択して OK ボタンを押すことでコンテキストを設定できる．

システム可変性モデルの Choice ノードでは，プロパティ部に表示される Context プロパティの入力欄の右側のボタンをクリックすると，図 4.18 のダイアログが表示される．ダイアログ上部には，サブシステム可変性モデルのノードがリストアップされる．このノードを複数選択して「AddRelation」ボタンを押すと，関係記述のターゲットノードとして選択され左下に表示される．さらに，それぞれのターゲットノードに関して，

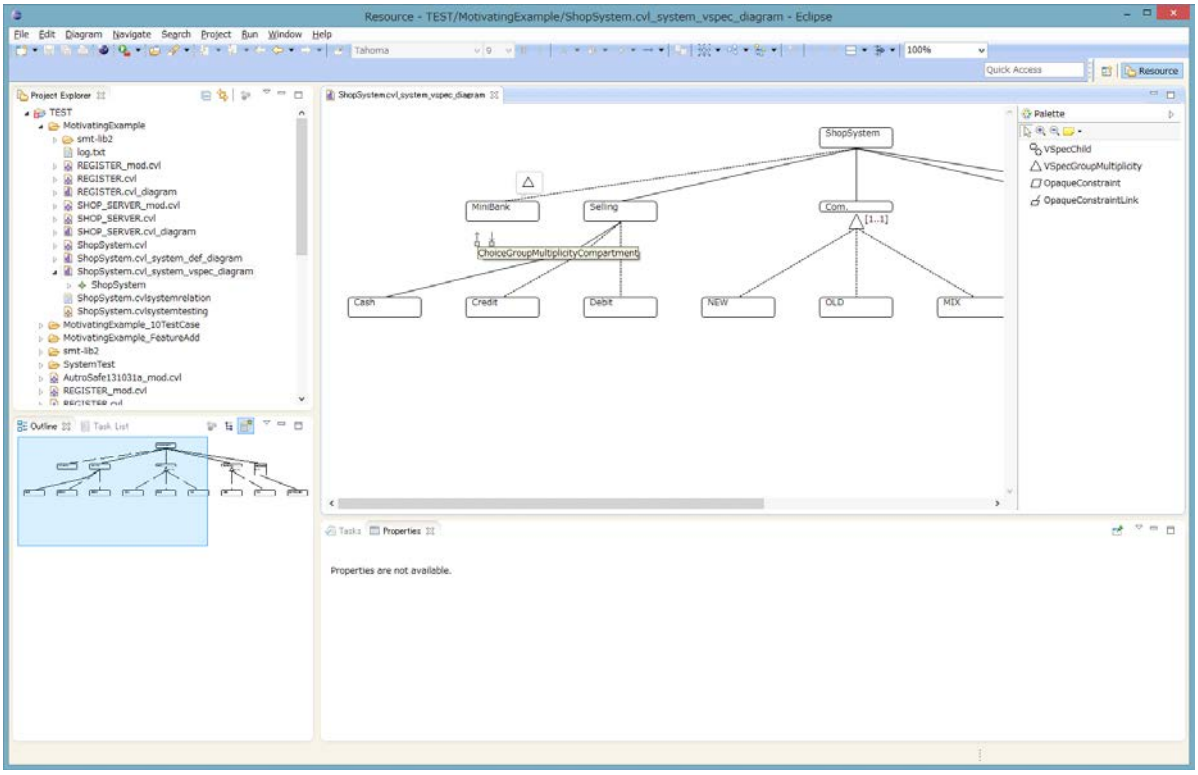

図 4.16: システム可変性エディタ

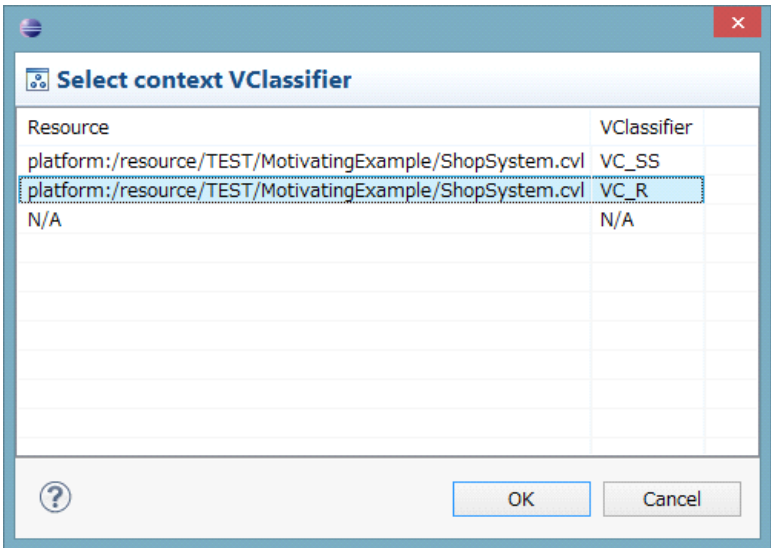

図 4.17: コンテキストの設定

自動で識別されたパス上の VClassifier ノードが表示されるため，適切な集約演算子を付与して関係記述を設定できる．

4.4　評価実験

提案する複合システム向け可変性モデルの分割記法とピンサームーブメントアプローチを用いた関係記述に関して，評価を行った．提案手法は，複数の可変性モデルとそれらの関係を記述するため複雑に見える．

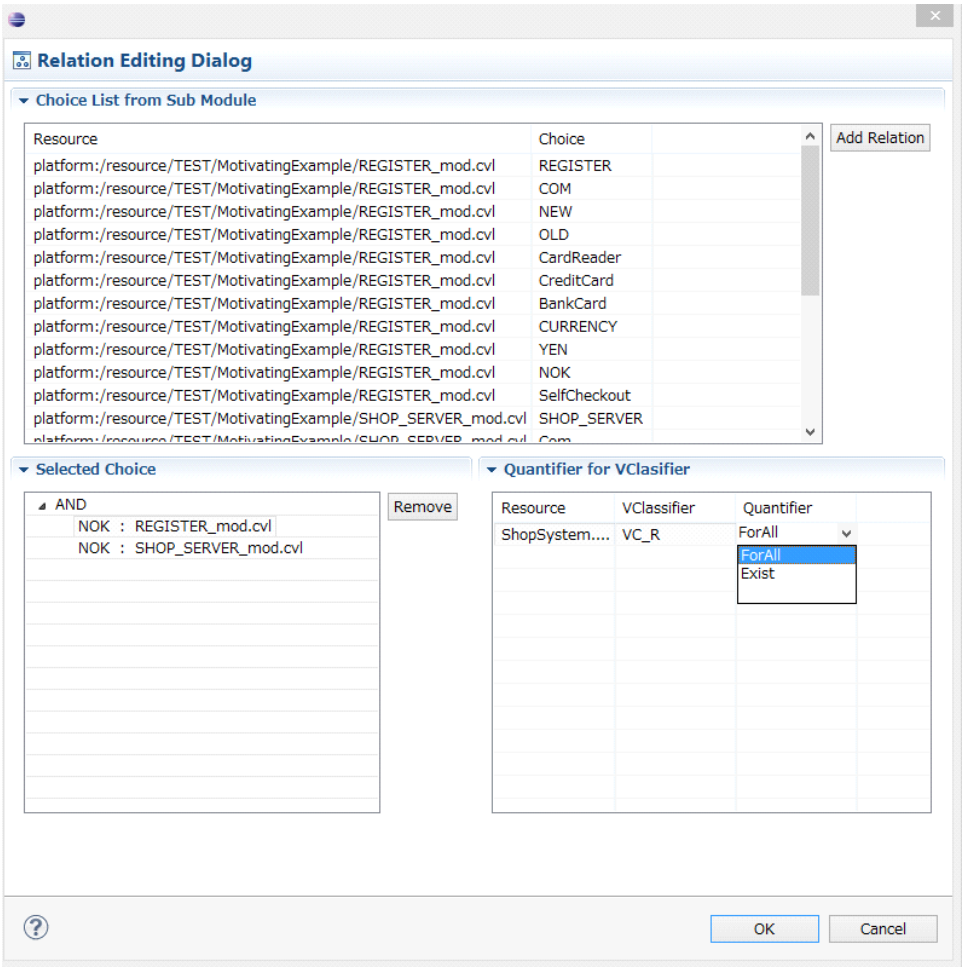

図 4.18: 関係記述の設定

そこで，この提案手法によって複合システムの可変性を表現するのに十分な記述能力があるのか，そして提案手法は実際に人が記述できるのかを評価する．評価に向けたリサーチクエスチョンを以下に定義する．

RQ1 提案された分割記法と関係記述は，既存のモデリング手法と比較して記述能力は十分か？
RQ2 提案手法による記法で，システム全体の要求や制約を記述するのは可能か？

RQ1 に対して，本論文では，既存の可変性モデリング手法の事例が我々の提案手法にて記述できるかを検証する．RQ2 に対して，自作 PC の例を用いて，モデリングの被験者実験を行い評価する．

4.4.1 RQ1: 既存のモデリング手法との比較

3.1.3 節で述べたように，インスタンス多重度を表現可能な Relative cardinality [36] が提案されている．Relative cardinality の記法は，1 つの可変性モデルで記述することを前提としており，複合システムの可変性を直接的にはサポートしない．Relative cardinality では，直近の親ノードに対するインスタンス多重度だけではなく，直接つながらないような祖先のノードに対するインスタンス多重度を記述できる．Sausa らは，提案する可変性モデル記法に関して，4 つのクラウドプロバイダの事例をもって評価を行った．例えば，Relative cardinality の記法におけるクラウドプロバイダ OpenShift を簡易化したものを図 4.19 に示す．

この記述は，OpenShift インスタンス 1 つに対して，1 から 100 までの Cartridge インスタンスを保持でき

```
OpenShift {
    Cartridge [1..100] {
        Gear < OpenShift > [1..100]
    }
}
```

図 4.19: Relative cardinality 記法による Open Shift

て，その Cartridge インスタンス 1 つに対して複数の Gear インスタンスが生じることを示している．さらに，この Gear インスタンスの数は，OpenShift インスタンス 1 つに対して，全部で 1 から 100 までの Gear インスタンスを保持できるという制約がある．このように Gear から見て，OpenShift は直接の親としてつながらない関係だが，そのノードに対する自身のインスタンス多重度を記述できる．この距離を置いたインスタンス多重度の関係記法を Relative cardinality という．

Sausa らの評価において，4 つのクラウドプロバイダに生じたフィーチャの合計数は 181 個で，Relative cardinality を記述したものは 11 個であった．また，合計で 40 個の制約が課されており，そのうちの 27 個が Relative cardinality に関するものであった．それゆえ，これらのフィーチャや制約を，本論文で提案する手法で記述可能かを検証する．

筆者は，4 つのクラウドプロバイダの可変性を提案する記法によって記述し，全てのフィーチャと制約が正しく記述できることを確認した．Relative cardinality の記法ではクラウドプロバイダごとに 1 つの可変性モデルによって表現していたところ，提案手法では 4 つのプロバイダに対して合計で 21 個の可変性モデルになった．これらの可変性モデルのノード数は 241 個，制約は 34 個になった．本論文で提案する手法では，関心ごとの分離が可能なため，ノード数が冗長化し増える傾向がある．

Relative cardinality は，本論文の提案手法における COUNT と SUM の演算子に対応している．図 4.20 に，上記の Relative cardinality の記法におけるクラウドプロバイダ OpenShift を簡易化したものを，提案手法で記述した例を示す．Cartridge はサブシステムとして定義され，そのサブシステム可変性モデルにおいて，Choice ノード「Gear」が VClassifier ノード「*VC-G* にぶら下がっている．構造可変性モデルでは，OpenShift という複合システムは，1 から 100 個の Cartridge インスタンスを保有できることを VClassifier ノード「*VC-C*」で示している．最後に，システム可変性モデルでは，Variable ノード「NumOfGear」を導入する．関係記述に関して，Variable ノード「NumOfGear」はターゲットノードとしてサブシステム可変性モデルの Choice ノード「Gear」を参照し，コンテキストは OpenShift 全体を意味するルートノードを示す．そのため，パスは，「*VC-C*」と「*VC-G*」を通る．これらの VClassifier ノードに対して以下のように，適切な集約演算子を付与することで Variable ノード「NumOfGear」を適切に定義できる．

$$\mathrm{NumOfGear} = \mathsf{SUM}_{VC\text{-}C}(\mathsf{COUNT}_{VC\text{-}G}(\mathrm{Gear}))$$

この関係記述では，この Variable ノード「NumOfGear」は，OpenShift システム全体の「Gear」の数を表すことを示している．このようにして Relative cardinality は，提案手法における関係記述と CVL の制約記述によって表現できる．

これらの結果から，RQ1 に関して，提案した分割記法と関係記述は，少なくとも既存の可変性モデリング手法と同等の記述能力を有することが確認できた．

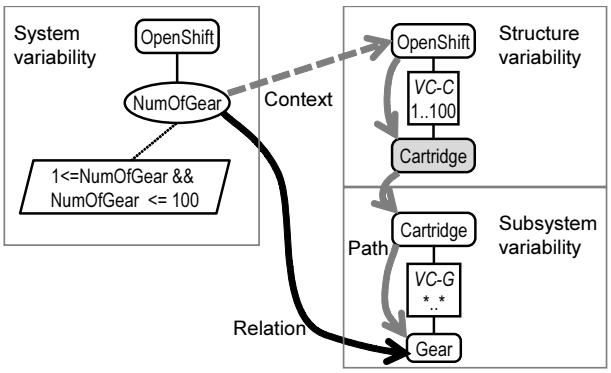

図 4.20: OpenShift example using the proposed notation.

4.4.2 RQ2: 要求と制約のモデリング実験

提案する分割記法と関係記述の実用性を評価するため，被験者に自然言語で記載されたシステム全体の要求と制約を与え，それに対応する可変性モデル (構造可変性モデル，システム可変性モデル，関係記述) を記述する実験を行った．もし，全ての要求と制約が適切にモデル化できていれば，RQ2 に対して肯定的な回答となる．

実験には，被験者として 6 名が参加した．6 名は全員，可変性モデルを記述した経験はないが，実務としてソフトウェア工学に関わっており，経験年数は 0 年から 12 年にわたる．彼らには最初に，CVL と提案手法のチュートリアルが提供され，可変性モデルを記述する作業の前に，提案手法について質問する時間を設けた．このチュートリアルには 2.2.2 節で示す架空の業務用空調機を用いており，4.1 節で述べた分割記述した可変性モデルや，4.2 節の関係記述の結果も含まれている．その後，被験者には 4.3 節で述べたモデリングツールを用いて，構造可変性モデル，システム可変性モデルと関係記述を実際に記述してもらった．この作業中，被験者たちはツールの使い方のみサポートが提供されている．また，それぞれの被験者たちは個別に実験を行った．

この実験には，様々なパーツを組み合わせて作成する架空の自作 PC システムを用いた．サブシステムには，ケース，CPU，メモリ，電源，ビデオカードがある．図 2.7 や図 2.8 のようなサブシステム可変性モデルは，あらかじめ筆者が作成し，被験者に提供された．そして，自然言語で記載された 2 つの制約 (C1 と C2) と 4 つの要求 (R1 から R4) を自作 PC システム向けに設定した．それらの特性は以下の通りである．

C1 業務用空調機の例における HP の値の比較のように，サブシステムにおける変数の値の合計値の比較を行う制約．

C2 業務用空調機の例における通信の後方互換性のように，サブシステム間の互換性の制約．

R1 and R2 適切な集約演算子を選択する必要のある要求．自作 PC システムは，複数のビデオカードを内包でき，ビデオカードごとに複数のデジタルインターフェースを持てる．この時，R1 と R2 は似た要求であるが，R1 は「$\exists_{Videocard}(\exists_{DigitalInterface}(\text{Target}))$」で，R2 は「$\forall_{Videocard}(\forall_{DigitalInterface}(\text{Target}))$」のようになる．

R3 クラウドプロバイダの例における Gear の数のように，サブシステム内のインスタンス数を合計する必要のある要求．

R4 システム可変性モデルの中で，サブシステムのインスタンス数に言及する要求．

実験の目的は，ツールを使って被験者がこれらの制約や要件を適切にモデル化できるかどうかを検証するこ

表 4.5: モデリング実験の結果

Years of software experience	Minutes for tutorial	Minutes for task	C1	C2	R1	R2	R3	R4
0	45	110	✓	×	✓	×	✓	✓
1	45	40	✓	✓	✓	✓	✓	✓
3	45	120	✓	✓	✓	∗1	✓	✓
8	45	65	✓	✓	✓	✓	✓	✓
10	45	65	✓	✓	✓	∗2	✓	✓
12	40	95	✓	✓	✓	✓	✓	✓

✓: Correct, × : Wrong, ∗ : Almost correct

とである．

表 4.5 に，実験結果[*3]を示す．各行は，被験者を表している．最初の列は，被験者の業務経験年数を表している．2 列目と 3 列目はそれぞれ，チュートリアルと質問に要した時間と，実験を行った時間を示している．残りの列は制約と要求が適切に記述できたかの結果を示している．

全ての被験者が，構造可変性モデルは適切に記述できた．モデリングにかかった時間は最大で 2 時間であり，実用的な範囲に収まっている．一方で，システム可変性モデルに表現する制約と要求に関しては，被験者によって結果が異なっている．

経験年数 1 年，8 年，12 年の 3 人の被験者は，すべての制約と要求を正しく記述できた．一方で，経験年数 3 年の被験者は，R2 (∗1) に関して自然言語の読解時に誤解が生じていたことが分かった．要求の記述として「すべてのデジタルインターフェースがターゲット機能を持っている」という記載を，その被験者は「すべてのビデオカードがターゲット機能を持ったデジタルインターフェースを少なくとも 1 つ持っている」と誤解していた．筆者は，この被験者に対して実験後にインタビューを行い，誤解を解いた状態であれば，適切にモデリングができたであろうことを確認してある．この誤解に関しては，2.3.1 節で述べた課題 1-2 の「複合システム可変性における多重度の曖昧さ」を実証しているといえる．経験年数 10 年の被験者は，ほぼすべての要求と制約を適切に記述でできたが，R2 (∗2) に関して余分な記述を追加していた．この被験者は，R2 向けの Choice ノードには適切な関係記述として，「$\forall_{Videocard}(\forall_{DigitalInterface}(\text{Target}))$」を付与できていた．しかし，この被験者は，その Choice ノードに余計な VClassifier ノードをぶら下げており，それは不必要なものであった．このような小さな問題があっても，経験年数 3 年と 10 年の被験者 2 名はほとんどのモデル化を適切に行うことができた．しかし，経験年数 0 年の被験者に限っては，C2 と R2 の両方のモデル化を適切に行うことができなかった．その被験者にインタビューしたところ，失敗の原因は，モデル化の難しさ（モデル化経験がないことに起因する）にあることが判明した．

一般的に，モデル化に不慣れな人がモデルを適切に記述できないのは当然のことである．よって，RQ2 に対する回答として，他のモデリング手法と同様に複雑な表現になっているが，多少のモデリング経験者であれば提案手法を用いて記述が可能である．

4.5　考察

本節では，提案手法と既存の可変性モデリングアプローチの比較について述べた後，妥当性への脅威について議論する．

[*3] 実験に用いたチュートリアル･ツール･結果ファイルは以下のサイトで公開している．https://doi.org/10.6084/m9.figshare.13370714

表 4.6: モデリングアプローチの比較

	Style	Separate model	Identify quantities	Aggregation in relations
Relative cardinality [36]	Text	–	–	–
Multi-level configuration [11, 23]	Graphical	✓	–	–
Proposed method	Graphical	✓	✓	✓

✓: Available, –: Not available

4.5.1 モデリングアプローチの比較

提案手法について，既存研究と比較した．比較に際し，2 つの既存研究を抽出している．1 つ目は，4.4.1 節にてインスタンス多重度の記述能力で議論を行った Relative cardinality [36] である．筆者の知る限りにおいて，直接インスタンス多重度を取り扱う既存研究は Relative cardinality のみである．比較を公正に行うため，Multi-level configuration [11, 23] も比較対象に加える．このアプローチは，構造可変性，システム可変性，サブシステム可変性を分けた上でその間の関係を表現できる．また，このアプローチはインスタンス多重度の制約に関して直接的な表現を持っていないが，OCL を制約記述言語としてサポートしているため，複合システムにおけるインスタンス多重度を扱うことができる．表 4.6 に，比較結果のサマリを示す．比較は以下の 3 項目で行っている．

- 複数のモデルに分割して他のモデルを参照することは可能か？これにより，可変性モデルの再利用性を向上させることができる．
- 可変性モデルの間でのインスタンス多重度の識別は可能か？これにより，各可変性モデルを記述や修正した場合に，整合性をとりやすくなる．
- 可変性モデルの間に生じるインスタンス多重度に対して，集約するための演算子を付与できるか？これにより，複雑な関係を記述できる．

Relative cardinality では，単一の可変性モデルで表現される．また，可変性モデルを記述した場合，ノードの間に生じるインスタンス多重度を識別することはできない．関係性の記述に関して，インスタンス数の上限・下限値を表現することは可能だが，最大値・最小値などの複雑な表現は未対応である．

Multi-level configuration は，他の可変性モデルへの参照を示す機能を持つノードを提供している．このアプローチでは，ノード間に生じるインスタンス多重度を識別する方法はない．関係の表現に関しては OCL が導入されているため，本論文で提案した集約演算の全てを行うことが可能であるが，システムとサブシステムで別々に記述された可変性モデル間の関係を表現することはできない．これは可変性モデルを 1 つのモデルに統合して記述することで解決可能ではあるが，その場合は可変性モデルが大きくなり複雑になる．

本論文で提案する手法では，構造可変性，システム可変性，サブシステム可変性を分けて記述する専用の記法を提供している．また，簡単にインスタンス多重度を識別して集約演算を行うためのピンサームーブメントアプローチを用いている．そして，関係記述において，様々な集約演算に対応している．

既存のアプローチはどれも全ての項目を満たしてはいない一方で，本論文における提案手法は全ての項目を満たした．

4.5.2　妥当性への脅威

内的妥当性への脅威として，4.4.2 節で述べた実験には 2 つの脅威が存在する．まず，サブシステム可変性モデルがあらかじめ筆者が作成し被験者に提供され，それをもって構造可変性モデルとシステム可変性モデルを記述する実験を行っている．もし，被験者がサブシステム可変性モデルから作成した場合，実験結果に影響した可能性がある．本研究では，サブシステムそれぞれが，システム全体の開発から独立して行われていることを前提としているため，この実験方法を選択している．実験の目的は，全ての可変性モデリングの可能性を判断するのではなく，複数のサブシステム可変性モデルが結合されたシステム全体の可変性モデルを記述することの実現性を判断することである．次の脅威は，可変性モデルを記述する際に，本研究で開発したツールを用いたことである．ツールの UI はやや複雑になっており，記述結果に強く影響を及ぼす．具体的には，可変性モデル間の関係記述を入力する際に，複数のダイアログの入力方法の説明が必要であった．

RQ1 では，4 つのクラウドプロバイダの可変性モデルを確認した．RQ2 では，チュートリアルとして業務用空調機の例を紹介し，提案ツールを用いて自作 PC システムを記述する実験を行った．外的妥当性の脅威としては，実際の複雑なシステムに対する提案手法の有効性が検証されていないことが挙げられる．他のケーススタディ研究と同様に，生じうる全てのケースを網羅するように実験を一般化することができない可能性がある．また，被験者が同じ会社のソフトウェア技術者であることも外的妥当性の脅威であり，1 社の従業員のみを対象とした実験では偏りが生じてしまう可能性がある．今後は，様々なシステムを用いた実験をする必要がある．

第 5 章

複合システムのシステム構成の導出

本章[*1]では，2.3.2 節で述べた以下の課題を解決するための，複合システムにおけるテストケース網羅のためのシステム構成導出手法について述べる．

課題 2-2 テストケースごとに実施可能なシステム構成の導出が必要
課題 2-3 より少ないシステム構成で必要なテストケースセットの網羅が必要

まず，本章で用いる仮想の Point of sale（POS) レジシステムについて述べる．次に，システム構成導出手法について述べ，4.3 節で述べたツール CT-CVL 上でのシステム構成の導出に関して述べる．最後に，提案するシステム構成導出手法についての評価結果について述べる．

5.1 複合システム：POS レジシステム

本節では，システム構成導出の方法を説明する上で用いる例として，スーパーマーケット向けの仮想の POS レジシステムを導入する．本例は筆者が作成している．まず，扱う POS レジシステムの仕様とテストケースについて述べる．その後，4 章で述べた複合システム向けの可変性モデルの分割記法と関係記述を用いて POS レジシステムの可変性を記述する．

5.1.1 POS レジシステムとテストケース

POS レジシステムは，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 5.1)．つまり，POS レジシステムにおけるサブシステムにはレジ (Register) と店舗サーバ (Shop server) と，そして集中サーバ (Center server) がある．サブシステムの製品リストと各仕様を表 5.1.2 に示す．サブシステムの機能として，業務用

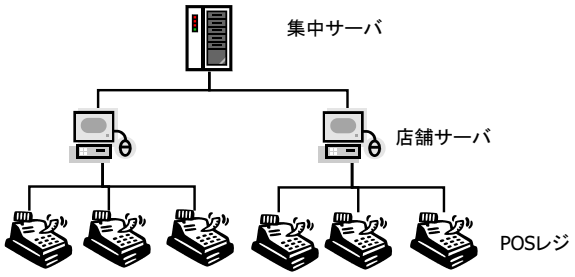

図 5.1: POS レジシステム

[*1] 本章の内容は，筆者の以前の論文 [46] と発表 [45] に基づく．

表 5.1: POS レジシステムのサブシステム製品

Subsystem	Product	Specification
Center server	CENTER_No.1	JPY
	CENTER_US	USD
Shop server	SHOP_PERFECT	New protocol, Func A, JPY
	SHOP_PRO	New protocol, JPY
	SHOP_STANDARD	Func A, JPY
	SHOP_CLASSIC	JPY
	SHOP_PERFECT_US	New protocol, Func A, USD
	SHOP_PRO_US	New protocol, USD
	SHOP_STANDARD_US	Func A, USD
	SHOP_CLASSIC_US	USD
Register	REGISTER_PRO	New protocol, Func B, JPY
	REGISTER_STANDARD	New protocol, Func C, JPY
	REGISTER_CLASSIC	Func C, JPY
	REGISTER_PRO_US	New protocol, Func B, USD
	REGISTER_STANDARD_US	New protocol, Func C, USD
	REGISTER_CLASSIC_US	Func C, USD

空調機と同様に通信に関して新プロトコル (New protocol) がある．また，店舗サーバ側には機能 A(Func A) があり，レジ側には機能 B(Func B) と機能 C(Func C) がある．

POS レジシステムの製品の組み合わ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　レジと店舗サーバの間でやり取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，店舗サーバが新プロトコル対応の機器であれば，接続される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　店舗サーバが機能 A を持つ場合，その店舗サーバに接続するレジは，機能 B を持たなければならないという制約がある．

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 つを挙げる．

TC1　日本円を扱うシステムにおいて，店舗サーバとレジの接続が新プロトコルで通信を行う状況で，店舗サーバの機能 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，店舗サーバと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，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，店舗サーバとそれに接続するレジの中で，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，一つの店舗サーバとそれに接続するレジの中で機能 B が動作し，別の店舗サーバとそれに接続するレジの中で機能 C が動作することを確認する．なお，同じ店舗サーバ

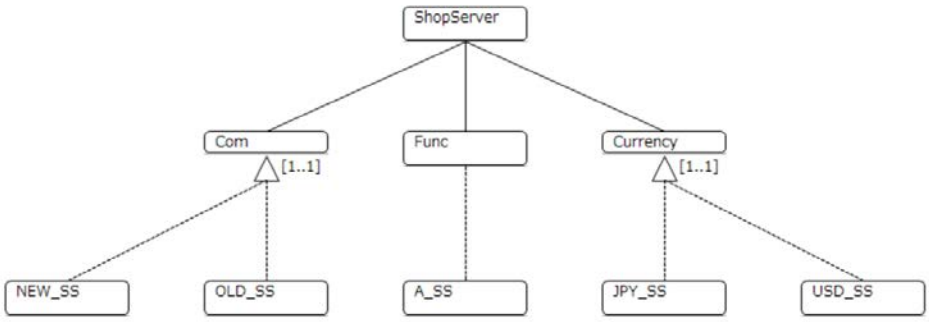

図 5.2: 店舗サーバのサブシステム可変性モデル

Product	ShopServer	Com	NEW_SS	OLD_SS	Func	A_SS	Currency	JPY_SS	USD_SS
SHOP_PERFECT	X	X	X		X	X	X	X	
SHOP_PRO	X	X	X		X		X	X	
SHOP_STANDARD	X	X		X	X	X	X	X	
SHOP_CLASSIC	X	X		X	X		X	X	
SHOP_PERFECT_US	X	X	X		X	X	X		X
SHOP_PRO_US	X	X	X		X		X		X
SHOP_STANDARD_US	X	X		X	X	X	X		X
SHOP_CLASSIC_US	X	X		X	X		X		X

図 5.3: 店舗サーバのサブシステムリゾリューションモデル

に接続するレジでは機能 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，店舗サーバとレジの接続が新プロトコルで通信を行う状況で，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，店舗サーバとレジの組み合わせを複数必要としている．TC5 は，4 つのフィーチャを指定するテスト項目となっている．

5.1.2 サブシステム可変性モデル

本節では，5.1.1 節で述べた POS レジシステムのサブシステム可変性モデルを導入する．集中サーバに関しては，日本円 (JPY) と US ドル (USD) のどちらかを選択する可変性のみで，それぞれに 1 製品があるだけのため，モデル表現は割愛する．サブシステム可変性モデルは，表の仕様情報を読み解き作成する．

図 5.2 に店舗サーバのサブシステム可変性モデル，図 5.3 に店舗サーバのサブシステムリゾリューションモデル (プロダクトマップビュー) を示す．これらの図は，4.3 節で述べた CT-CVL にて記述したスクリーンショットを用いている．店舗サーバは，2.2.2 節で述べた業務用空調機と同様にして，通信機能において新・旧プロトコルを持ち，グループ多重度分解にてどちらかを選ぶ可変性を持つ．また，選択可能分解で，機能 A を持つ．最後に，取り扱う通貨として日本円 (JPY) と US ドル (USD) があり，グループ多重度分解にてどちらかを選ぶ可変性を持つ．各葉ノード名の末尾の「_SS」は，店舗サーバ (ShopServer) のサブシステム可変性モデルのノードであることを示す．

図 5.4 にレジのサブシステム可変性モデル，図 5.5 にレジのサブシステムリゾリューションモデル (プロダクトマップビュー) を示す．レジは，店舗サーバと通信を行うため新・旧プロトコルを持ち，グループ多重度分解にてどちらかを選ぶ可変性を持つ．また，レジは，機能 B と機能 C を持ち，グループ多重度分解にてどちらかを選ぶ可変性を持つ．最後に，取り扱う通貨として，日本円 (JPY) と US ドル (USD) があり，グループ多重度分解にてどちらかを選ぶ可変性を持つ．各葉ノードの「_R」は，レジ (Register) のサブシステム可変性モデルのノードであることを示す．

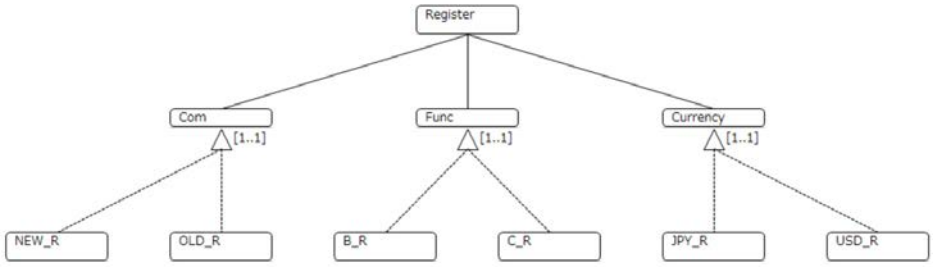

図 5.4: レジのサブシステム可変性モデル

Product	Register	Com	NEW_R	OLD_R	Func	B_R	C_R	Currency	JPY_R	USD_R
REGISTER_PRO	X	X	X		X	X		X	X	
REGISTER_STANDA...	X	X	X		X		X	X	X	
REGISTER_CLASSIC	X	X		X	X		X	X	X	
REGISTER_PRO_US	X	X	X		X	X		X		X
REGISTER_STANDA...	X	X	X		X		X	X		X
REGISTER_CLASSI...	X	X		X	X		X	X		X

図 5.5: レジのサブシステムリゾリューションモデル

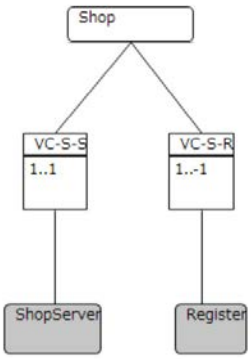

図 5.6: 店舗システムの構造可変性モデル

5.1.3　構造可変性モデルとシステム可変性モデル

集中サーバを含まない，各店舗の店舗サーバ 1 台とレジ複数台の組み合わさった複合システムを，店舗システムと呼ぶ．本節では，この店舗システムの構造可変性モデルとシステム可変性モデルについて述べる．図 5.6 に，店舗システムの構造可変性モデルを示す．店舗システムには，1 台の店舗サーバと，1 台以上のレジによって構成されることが示されている．

図 5.7 に，店舗システムのシステム可変性モデルを示す．まず，店舗システムの可変性として，通信プロトコルがある．通信プロトコルの可変性を表す「COM」ノードには，グループ多重度分解として，「New」，「Old」，「Mix」の 3 つの Choice ノードがあり，このうちのいずれか 1 つを選択可能である．ここで，図 5.7 の Choice ノード名に付加されている「_SYS」は，店舗システムのノードであることを示している．2.2.2 節で述べた業務用空調機の通信プロトコルでは，室外機の新プロトコルは後方互換性があり，旧プロトコルを持つ室内機と接続された際は旧プロトコルで通信が可能であった．ここで，店舗システムは，業務用空調機の例とは異なり，構成する店舗サーバとレジは，両者ともに後方互換性を持つ．また，5.1.1 節にて，POS レジシステムは通信プロトコルの混在というシステム仕様があると述べた．新プロトコルを持つ店舗サーバが，新プロトコルを持つレジと旧プロトコルを持つレジが接続された場合，店舗サーバは接続先ごとにプロトコ

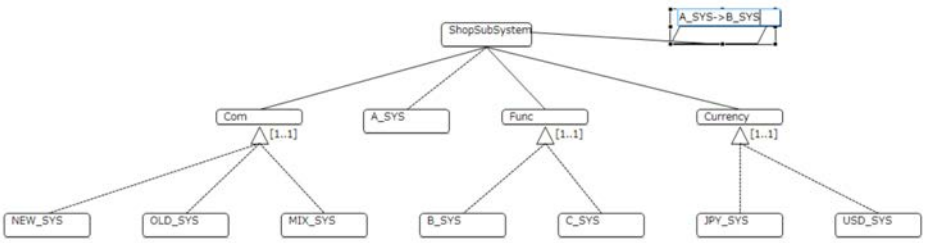

図 5.7: 店舗システムのシステム可変性モデル

表 5.2: 店舗システムの通信プロトコル (COM) の関係

	Subsystem		System
Shop server	Register	Register	
OLD	OLD	OLD	OLD
OLD	OLD	NEW	OLD
OLD	NEW	NEW	OLD
NEW	OLD	OLD	OLD
NEW	OLD	NEW	MIX
NEW	NEW	NEW	NEW

ルを変えることができる．つまりこの状況において，店舗システム内には，新プロトコルの通信と旧プロトコルの通信が混在する．このケースにおいて，先に述べた Choice ノード「Mix」が選択される．表 5.2 にて，店舗サーバ・レジの通信プロトコルと，店舗システムの通信プロトコルの可変性の関係を示す．この表では，サブシステムであるレジが複数台生じたときに，その通信プロトコルが異なるケースを表現するために，レジを表す列を 2 つ導入している．

次に，店舗サーバのサブシステム可変性モデルにおける機能 A と，レジのサブシステム可変性モデルにおける機能 B・機能 C に対応して，店舗システムの可変性は機能 A・機能 B・機能 C を持つ．5.1.1 節にて，店舗サーバが機能 A を持つ場合，そのテンポサーバに接続レジは機能 B を持たなければならないという制約があると述べた．そのため，平行四辺形で示す CVL の制約記述方法を用いて，機能 A が機能 B を要求する制約を記述する．これは，サブシステムを横断して生じる制約をシステム可変性モデルに持ち上げて表現が可能になった例となる．

最後に，店舗システムのシステム可変性モデルでは通貨の可変性を持つ．5.1.1 節にて述べたように，通貨は排他性があるので，店舗システム内のサブシステム全てで同じ通貨を用いる必要があり，日本円と US ドルのどちらかを選択する表現となっている．

これらのシステム可変性モデルの各ノードの関係記述を，表 5.3 に示す．

本節では，POS レジシステムにおける，店舗サーバとレジを組み合わせた店舗システムの構造可変性モデルとシステム可変性モデルについて述べた．一方で，POS レジシステムには，集中サーバを導入し，複数の店舗システムを管理することもできる．この場合，全体の POS レジシステムに対して，集中サーバと店舗システムがサブシステムとして機能し，店舗サーバとレジがサブサブシステムとなる．ここで，TC1 から TC5 のテストケースを見ると，TC4 以外は，店舗システムを対象としたテストケースとなっている．また，TC4 に関しても，複数の店舗システムを繋げるだけでテストが可能であり，集中サーバを含めた機能の組み合わ

表 5.3: 店舗システムの関係記述

Source	Target	VClassifier	Relational definition
NEW_{SYS}	NEW_{SS}	*VC-S-S*	$\forall_{VC\text{-}S\text{-}S}(NEW_{SS}) \wedge$
	OLD_{R}	*VC-S-R*	$\forall_{VC\text{-}S\text{-}R}(NEW_{R})$
MIX_{SYS}	NEW_{SS}	*VC-S-S*	$\forall_{VC\text{-}S\text{-}S}(NEW_{SS}) \wedge$
	OLD_{R}	*VC-S-R*	$\exists_{VC\text{-}S\text{-}R}(NEW_{R}) \wedge$
	OLD_{R}	*VC-S-R*	$\exists_{VC\text{-}S\text{-}R}(OLD_{R})$
OLD_{SYS}	NEW_{SS}	*VC-S-S*	$\forall_{VC\text{-}S\text{-}S}(OLD_{SS}) \vee$
	OLD_{R}	*VC-S-R*	$\forall_{VC\text{-}S\text{-}R}(OLD_{R})$
A_{SYS}	A_{SS}	*VC-S-S*	$\forall_{VC\text{-}S\text{-}S}(A_{SS})$
B_{SYS}	B_{R}	*VC-S-R*	$\forall_{VC\text{-}S\text{-}R}(B_{R})$
C_{SYS}	C_{R}	*VC-S-R*	$\forall_{VC\text{-}S\text{-}R}(C_{R})$
JPY_{SYS}	JPY_{SS}	*VC-S-S*	$\forall_{VC\text{-}S\text{-}S}(JPY_{SS}) \wedge$
	JPY_{R}	*VC-S-R*	$\forall_{VC\text{-}S\text{-}R}(JPY_{R})$
USD_{SYS}	USD_{SS}	*VC-S-S*	$\forall_{VC\text{-}S\text{-}S}(USD_{SS}) \wedge$
	USD_{R}	*VC-S-R*	$\forall_{VC\text{-}S\text{-}R}(USD_{R})$

せをテストするものではない．集中サーバの接続に関しても，集中サーバの通貨と店舗システムの通貨を統一するだけで良い．そのため，テストケースを実施可能なシステム構成の導出に関しては，店舗システムに焦点を当て，集中サーバに関してはスコープ外とする．

5.2　テストケース網羅のためのシステム構成導出

本節では，2.3.2 節で述べた課題 2-2 と課題 2-3 を解決するためのシステム構成導出方法について述べる．入力は，サブシステム可変性モデル，サブシステムリゾリューションモデル，構造可変性モデル，システム可変性モデル，そして実施したいテストケースである．基本的なアプローチとして，テストケースごとに実施可能なシステム構成導出に関して，細かく問題を分割してそれぞれを充足可能性問題に落とし込む手法をとる．そして，テストケースごとに得られたシステム構成に対して，他のテストケースを実施可能か検証し，その結果に対して充足可能性問題にて全てのテストケースを網羅する組み合わせを得ることで，より少ない数のシステム構成でテストケースを網羅する．システム構成導出に関して，提案する導出手法を以下のような手順に分解する．

1. テストケース作成者は，実施するテストケースごとに，システム可変性モデルにおいて対応する可変性を束縛したシステムテストケースモデルを作成する．
2. 提案手法では自動で，システムテストケースモデルごとに，関係記述を用いて，サブシステム可変性モデルにおいて対応する可変性を束縛したサブシステムテストケースモデルを導出する．
3. 提案手法では自動で，サブシステムテストケースモデルごとに，そのモデルが満たすサブシステムリゾリューションモデルを抽出する．
4. 提案手法では自動で，得られたサブシステムリゾリューションモデルを，構造可変性モデルを参照して組み合わせることで，候補となるシステム構成を得る．
5. 提案手法では自動で，システム構成の候補ごとに，その組み合わせによってシステム可変性モデルの

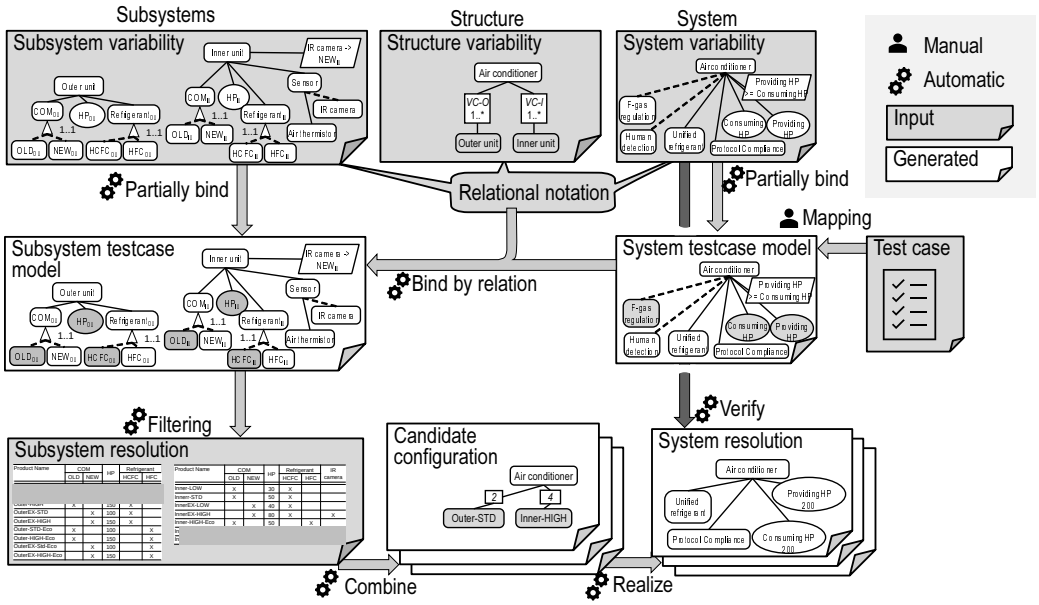

図 5.8: システム構成導出の概要

束縛結果としてのシステムリゾリューションモデルを導出する．

6. 提案手法では自動で，得られたシステムリゾリューションモデルが，システム可変性モデルにおいて記述された制約に違反していないか検証し，違反したリゾリューションは取り除く．
7. 提案手法では自動で，残ったシステムリゾリューションモデルが，他のシステムテストケースモデルを実施可能か検証し対応関係を記録する．
8. 提案手法では自動で，記録したシステムテストケースモデルとシステムリゾリューションモデルの対応関係を充足可能性問題に変換し，テストケースを網羅する少ない数のシステム構成を導出する．

手順 1 から手順 6 までで課題 2-2 を解決し，以降の手順にて課題 2-3 を解決する．手順 1 は，テストケースを読み解き，システム可変性モデル上の対応するノードを選択して可変性を束縛する作業のため，手動での作業を想定しており，他の手順は，システマチックに行えるため自動化が可能である．手順 1 から手順 6 までの概要を，図 5.8 に示す．

本節の以降では，5.1 節で述べた仮想の POS レジの例を用いて，手順 1 から 8 までを説明する．

5.2.1 手順 1: システムテストケースモデルの作成

複合システム向けのテストケースを，システム可変性モデルのフィーチャと紐づける表現を作成する．図 5.8 の右側に示すテストケースを入力として，テストケースと関係するシステム可変性モデルに現れているノードを束縛し，束縛可変性モデルを作成しシステムテストケースモデルとする．

5.1 節で述べたテストケースの TC1 から TC5 に関して，店舗システムのシステム可変性モデルとの対応とその束縛結果について，表 5.4 にまとめる．各行にテストケースを示し，列にはシステム可変性モデルの葉ノードを示す．中間ノードに関しては，基本的に必須分解されており可変性は生じないため省略する．テストケースの記述ごとに，テスト実施に必要なシステム可変性モデルのノードに対して，チェックマーク (✓) を付与する．また，テストケースを実施するためには不要な場合は空欄とする．アスタリスク (∗) は，そのテストケースの実行に関して，そのシステム可変性モデルのノードは関係がないことを意味する．言い換えると，アスタリスクが付与された箇所は Choice ノードで示された可変性に対して束縛を行わず，システムテ

表 5.4: 店舗システムのシステムテストケースモデル

	COM				Func			
Testcase	OLD_{SYS}	NEW_{SYS}	MIX_{SYS}	A_{SYS}	B_{SYS}	C_{SYS}	JPY_{SYS}	USD_{SYS}
TC1	*	✓	*	✓	*	*	✓	*
TC2	*	*	✓	*	*	✓	✓	*
TC3	*	*	*	✓	✓	*	✓	*
TC4-1	*	*	*	*	✓	*	✓	*
TC4-2	*	*	*	*	*	✓	✓	*
TC5	*	✓	*	✓	✓	*	✓	*

ストケースモデルにおいて可変性が残る．

例えば，TC1 の記述は，「日本円を扱うシステムにおいて，店舗サーバとレジの接続が新プロトコルで通信を行う状況で，店舗サーバの機能 A が正常に動作することを確認する．」とある．そのため，JPY_{SYS} と NEW_{SYS} と A_{SYS} にチェックマークがあり，他はアスタリスクとなる．

TC4 以外のテストケースは，1 つの店舗システムを対象とするテスト記述であったのに対して，TC4 では集中サーバを介して複数の店舗システムが結合された POS レジシステムを対象としている．この TC4 に関しては，記述を見ると複数の店舗システムを組み合わせるだけの記述である．そのため，店舗システムに関するシステムテストケースモデルとして 2 つに分解し，「TC4-1」と「TC4-2」のそれぞれで，システム可変性モデルとの関係を表現している．

POS レジシステムの例は，実装するかしないかを選択可能な Choice ノードのみで構成されている．CVL には，それ以外のノードとしてインスタンス数の上限値と下限値によってインスタンス多重度の可変性を表す VClassifier ノードと，2.2.2 節の業務用空調機の例における馬力数を表すような変数の可変性である Variable ノードがある．これらのノードに関してもテストケースと関連がある場合は，束縛を行う．VClassifier ノードの場合は，システム可変性モデルにおいて既にインスタンス数の上限値と下限値が記載されている．VClassifier ノードをテストケースと関連付けてシステムテストケースモデルで記載する場合，この範囲を超えないように，新たな上限値と下限値によって束縛することができる．また，具体的なインスタンス数を指定することも可能である．その場合は，上限値と下限値の両方に，その指定したい値を付与する．Variable ノードも同様にして，テストケースを実行するために必要な値の範囲を上限値と下限値を与えることによって束縛する．

本手順に関して，システム可変性モデルを束縛してシステムテストケースモデルを作成すると述べた．システム可変性モデルは，複合システム全体の視点で分析された可変性を表している一方で，ここでのテストケースはシステム全体に対して検証すべき機能などを記述している．そのため，原則的には，テストケースに記載の内容は，システム可変性モデルにおいて対応が取れるものとなる．しかしながら，実際の開発現場では，システム可変性モデルを作成したのち，テストケースを検討した結果，新たなシステム可変性モデルのノードが生じるケースがある．そのため，システム可変性モデルは，各テストケース記述からシステムテストケースモデルの作成を行いながら，加筆することが望ましい．

システム可変性モデルに対して，その可変性モデルの中の各ノードが持つ可変性の選択結果を組み合わせることで，システムリゾリューションモデルが有限数で生成可能である．そのため，それらのリゾリューションモデルをテストケースとして定義するというアプローチが考えられる．このアプローチに関して，3.2 節の関連研究において，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 [20, 21] を紹介した．これらの手法を用いることで，テストケースの生成や抽出が可能

となるため，2.3.2 節で述べた課題 2-1 を解決できる．本節では，2.2.3 節で述べたように派生開発において蓄積されたテストケースが存在することを前提とする．

5.2.2 手順 2: サブシステムテストケースモデルの導出

それぞれのシステムテストケースモデルから，関係記述を用いて，サブシステム可変性モデルを束縛したサブシステムテストケースモデルを導出する．サブシステムテストケースの目的は，サブシステムの製品を示すリゾリューションモデルの中から，システムを構成するサブシステムの候補を抽出することである．サブシステムテストケースモデルは，サブシステム可変性モデルを部分的に束縛した，束縛可変性モデルで表される．サブシステム可変性モデルは，すべてのサブシステムリゾリューションモデルを包含するように可変性を記述するのに対し，サブシステムテストケースモデルは，可変性を部分的に束縛しているため，束縛に合致するサブシステムリゾリューションモデルを抽出できる．

複合システムは複数種類のサブシステムで構成されるため，1 つのシステムテストケースモデルに対して，サブシステムの種類ごとにサブシステムテストケースモデルを導出する必要がある．また，サブシステムの種類が同じでも，システムテストケースモデルで表されたテストを実施するためには，異なるサブシステムリゾリューションを複数必要とするケースが生じる．そのため，システムテストケースモデル 1 つに対して，複数のサブシステムテストケースモデルが導出される．そこで本手順では，このシステムテストケースモデルと複数のサブシステムテストケースモデルの関係を表すデータ構造を導入する．

サブシステムテストケースモデルの導出を述べるにあたり，データ構造として図 5.9 にクラス図とオブジェクト図を示す．図 5.9 上部のクラス図には，コンポジットパターンを用いた SubsystemTestcaseEntity と SubsystemTestcaseSet と SubsystemTestcase クラスがある．SubsystemTestcase クラスは，サブシステムテストケースモデルを保持する．サブシステムテストケースモデルは，VSpec ノードの木構造をとり，そのルートノードは Choice ノードが必須である．そのため，SubsystemTestcase クラスは Choice ノードをルートノードとして保持している．システムテストケースモデル 1 つに対して，関係記述に OR 表現がある場合など，複数のサブシステムが生じることを表現する必要がある．そこで，この複数のサブシステムに関して，サブシステムテストケースモデルがコンポジットパターンによる入れ子構造をとることで表現する．

図 5.9 下部のオブジェクト図に，このコンポジットパターンを用いたオブジェクト構造を示す．まず，システムテストケースモデル 1 つに対してサブシステムの種類ごとに，SubsystemTestcaseSet の tc インスタンスを用意する．tc インスタンスは複数の SubsystemTestcaseSet のインスタンスを持つ．この最初の入れ子をレベル A とし，これらは OR 結合を意味する．図中では，このレベルは「A1:SubsystemTestcaseSet」と「A2:SubsystemTestcaseSet」の二つのインスタンスが存在する．次にレベル A の SubsystemTestcaseSet インスタンスは，複数の SubsystemTestcase インスタンスを持つ．これは，AND 結合を意味し，レベル B とする．レベル B の SubsystemTestcase インスタンスはそれぞれ，サブシステム可変性モデルを部分的に束縛した木構造を示す束縛可変性モデルのルートノード (図中の C1 から C4) を 1 つずつ持つ．図中では割愛しているが，C1 から C4 のルートノードは，それぞれに子ノードや孫ノードといった木構造がぶら下がっており，サブシステム可変性モデルから部分的に束縛されたモデルを保持する．つまり，この図の例では，システムテストケースモデルで表されるシステムテストのテストを実施するためには，「$(C1 \wedge C2) \vee (C3 \wedge C4)$」のサブシステムテストケースモデルの組み合わせが必要であることを示している．そして，この C1 から C4 はそれぞれに，束縛に合致するサブシステムリゾリューションモデルを抽出することができる．よって，図 5.9 下部のオブジェクト図は，「C1 を満たすサブシステムリゾリューションモデルと C2 を満たすサブシステムリゾリューションモデルによって構成された複合システムか，もしくは，C3 を満たすサブシステムリゾリューションモデルと C4 を満たすサブシステムリゾリューションモデルによって構成された複合システムによって，システムテストケースモデルで示すテストを実施可能である」ということを意味する．

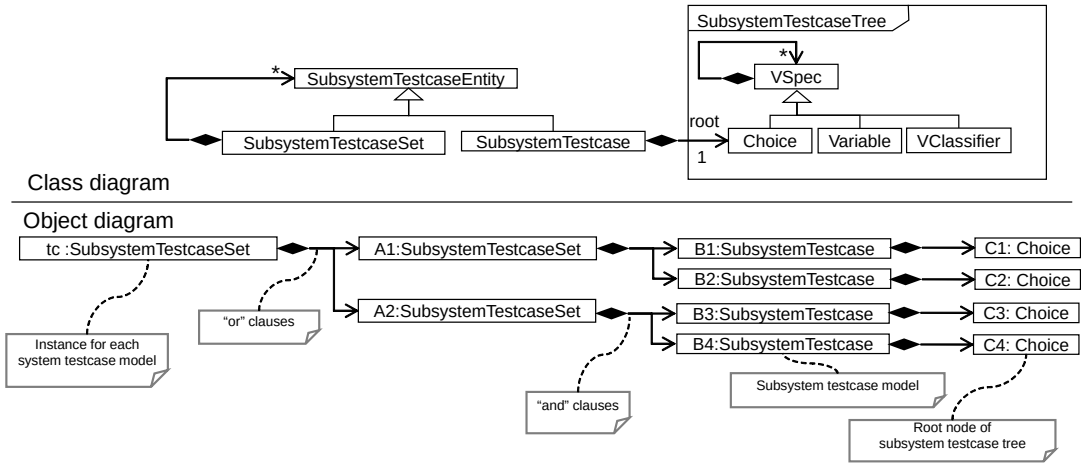

図 5.9: サブシステムテストケースモデル導出に係るデータモデル

このデータモデルを用いたサブシステムテストケースモデルの導出の方法の疑似コードを，図 5.10 に示す．1 行目は，SubsystemTestcaseSet の tc インスタンスの定義であり，図 5.9 下部のオブジェクト図の tc インスタンスを示す．このインスタンスに，最終的に得られるサブシステムテストケースを格納する．2 行目と 3 行目は，入力となるシステムテストケースモデルと，システムテストケースモデルの各ノードに付与された関係記述である．4 行目は，この処理全体のループで，システムテストケースモデルの全てのノードに対して繰り返しを行う．そしてこの繰り返しに関して，6 行目の条件分岐によって，システムテストケースモデル内の Choice ノードのみを対象とする．8 行目にて，対象とするシステムテストケースモデルのノードに対応する関係記述を取得し，9 行目にて，そのノードの結果を格納する SubsystemTestcaseSet のインスタンス nodeResult を作成する．10 行目から 27 行目までは，関係記述の繰り返し処理である．システム可変性モデルの Choice ノードに付与される関係記述は，論理和と論理積を用いた選言標準形でサブシステム可変性モデルのノードとの関係を表現している．そのため，10 行目の foreach 文において論理和で結合されているそれぞれの節の繰り返し処理を行い，14 行目の foreach 文において論理積で結合されている節の繰り返し処理を行う．論理和で結合されている節は，システムテストケースモデルのノードが必要とするサブシステム可変性モデルのノードが複数生じており，そのどちらかが成り立てばよいことを示しているため，図 5.9 下部のオブジェクト図におけるレベル A の aNew インスタンスを作成して，結果を格納する nodeResult インスタンスに加える (12,13 行目)．14 行目からの論理積で結合されている節の繰り返し処理では，サブシステムのインスタンス多重度を示す VClassifier ノードに付与された集約演算子によって処理が分かれる．$\forall$ の場合 (16 行目)，aNew インスタンス以下に含まれるサブシステム可変性モデルの対応するノードの可変性を選択として束縛する (18 行目)．$\exists$ の場合 (20 行目)，少なくとも 1 つのサブシステムが，関係記述で示されるサブシステム可変性モデルの対応するノードの可変性を選択として束縛されている必要がある．そのため，束縛されているサブシステムが存在し，該当する可変性の束縛を行わないサブシステムも許容することを表現する．具体的には，図 5.9 下部のオブジェクト図におけるレベル B において bNew インスタンスを，すでにある同じレベルのインスタンスからコピーして作成し，そのインスタンスに対して該当する可変性を選択として束縛する (22-24 行目)．これにより，作成したレベル B のインスタンスで該当する可変性が束縛されたサブシステムテストケースモデルが作成され，コピー元はほかのサブシステムを許容するテストケースモデルとして表現できる．システムテストケースモデルの各ノードに対して，サブシステムテストケースモデルを作成したら，それらを統合する (28 行目)．この統合では，例えば「$A1 \vee A2$」という結果と「$A3 \vee A4$」という結果を統合する場合に，OR で結合された同士はそれぞれ独立してサブシステムテストケースモデルを表現しているため，「$(A1 \wedge A3) \vee (A1 \wedge A4) \vee (A2 \wedge A3) \vee (A2 \wedge A4)$」というように分配されて統合する．ま

```
1  SubsystemTestcaseSet tc = new SubsystemTestcaseSet(); // サブシステムテストケースモデルの格納先
2  VSpec systemTestcaseRoot;     // システムテストケースモデル
3  map<VSpec, RelationalNotation> notations; // 関係記述
4  foreach( VSpec systemTestcaseNode in systemTestcaseRoot.getAll() ) // システムテストケースモデルのトラ
       バース
5  {
6    if ( systemTestcaseNode typeof Choice )
7    {
8      RelationalNotation relation = notations.get(systemTestcaseNode);
9      SubsystemTestcaseSet nodeResult = new SubsystemTestcaseSet(); // システムテストケースモデルのノー
           ドごとの結果
10     foreach( orClause in relation.getOrClauses() ) // 選言標準形なのでOR節を取得
11     {
12       SubsystemTestcaseSet aNew = createSubsystemTestcaseModel(); // レベルAのインスタンスを作成
13       nodeResult.add(aNew);
14       foreach( andClause in orClause.getAndClauses() ) // AND節を取得
15       {
16         if(andClause.getQuantifier = Type.ALL)
17         {
18           aNew.findNode(andClause.getTarget).setSelected(); //aNew以下の該当ノード全てを選択に束縛
19         }
20         if(andClause.getQuantifier = Type.EXISTS)
21         {
22           SubsystemTestcaseSet bNew = aNew.getFirst().clone(); // Bレベルのインスタンスを作成
23           bNew.findNode(andClause.getTarget).setSelected(); //bNew以下の該当ノードを選択に束縛
24           aNew.add(bNew);
25         }
26       }
27     }
28     tc.merge(nodeResult); // ノードごとの結果を全体結果に統合
29   }
30 }
```

図 5.10: システムテストケースモデルごとのサブシステムテストケースモデル導出

た，∧ で結合されるサブシステムテストケースモデル間で包含関係がある場合は，1 つのサブシステムテストケースモデルに集約する．

例えば, TC2 は, 表 5.4 にまとめたように, テストを実施するために「$\mathrm{MIX_{SYS}}$」と「$\mathrm{C_{SYS}}$」と「$\mathrm{JPY_{SYS}}$」の 3 つの可変性が束縛されている．ここで, 表 5.3 から,「$\mathrm{MIX_{SYS}}$」の関係記述は,「$\forall_{VC\text{-}S\text{-}S}(\mathrm{NEW_{SS}}) \wedge \exists_{VC\text{-}S\text{-}R}(\mathrm{NEW_R}) \wedge \exists_{VC\text{-}S\text{-}R}(\mathrm{OLD_R})$」である．また,「$\mathrm{C_{SYS}}$」の関係記述は,「$\forall_{VC\text{-}S\text{-}R}(\mathrm{C_R})$」である．次に,「$\mathrm{JPY_{SYS}}$」の関係記述は,「$\forall_{VC\text{-}S\text{-}S}(\mathrm{JPY_{SS}}) \wedge \forall_{VC\text{-}S\text{-}R}(\mathrm{JPY_R})$」である．店舗サーバ側への関係記述の言及は,「$\forall_{VC\text{-}S\text{-}S}(\mathrm{NEW_{SS}})$」と「$\forall_{VC\text{-}S\text{-}S}(\mathrm{JPY_{SS}})$」であり，これを解析すると，全ての店舗サーバが「$\mathrm{NEW_{SS}}$」と「$\mathrm{JPY_{SS}}$」の Choice ノードを実装していればよいとわかる．また，レジ側への関係記述の言及は,「$\exists_{VC\text{-}S\text{-}R}(\mathrm{NEW_R}) \wedge \exists_{VC\text{-}S\text{-}R}(\mathrm{OLD_R})$」と「$\forall_{VC\text{-}S\text{-}R}(\mathrm{C_R})$」と「$\forall_{VC\text{-}S\text{-}S}(\mathrm{JPY_R})$」である．これを読み解くと，レジは全て「$\mathrm{C_R}$」と「$\mathrm{JPY_R}$」を持っており，その中の少なくとも 1 台が「$\mathrm{NEW_R}$」を持ち，さらに少なくとも別の 1 台が「$\mathrm{OLD_R}$」を持つと分かる．

それぞれのテストケースに関して，店舗サーバとレジ向けにサブシステムテストケースモデルを導出すると，表 5.5 と表 5.6 のようになる．TC2 以外に関して，表 5.5 と表 5.6 に 1 行ずつしか生じておらず，それぞれが組み合わさる．例えば，TC1 に関しては，Shopserver-TC1 を満たす店舗サーバのサブシステムリゾリューションと Register-TC1 を満たすレジのサブシステムリゾリューションを組み合わせれば，TC1 のテストを実行可能なシステム構成が得られるという意味になる．

一方で，先に述べた TC2 の例のように，関係記述において ∃ が生じると，表 5.6 の Register-TC2-1 と Register-TC2-2 のように, サブシステムテストケースモデルが分割される．TC2 に関しては, Shopserver-TC2

表 5.5: 店舗サーバのサブシステムテストケースモデル

Subsystem testcase model	COM		Func	Currency	
	OLD_{SS}	NEW_{SS}	A_{SS}	JPY_{SS}	USD_{SS}
Shopserver-TC1	*	✓	✓	✓	*
Shopserver-TC2	*	✓	*	✓	*
Shopserver-TC3	*	*	✓	✓	*
Shopserver-TC4-1	*	*	*	✓	*
Shopserver-TC4-2	*	*	*	✓	*
Shopserver-TC5	*	✓	✓	✓	*

表 5.6: レジのサブシステムテストケースモデル

Subsystem testcase model	COM		Func		Currency	
	OLD_R	NEW_R	B_R	C_R	JPY_R	USD_R
Register-TC1	*	✓	*	*	✓	*
Register-TC2-1	*	✓	*	✓	✓	*
Register-TC2-2	✓	*	*	✓	✓	*
Register-TC3	*	*	✓	*	✓	*
Register-TC4-1	*	*	✓	*	✓	*
Register-TC4-2	*	*	*	✓	✓	*
Register-TC5	*	✓	✓	*	✓	*

を満たす店舗サーバのサブシステムリゾリューションと，Register-TC2-1 を満たすレジと Register-TC2-2 を満たすレジを組み合わせてシステム構成を作成する必要がある．

表 5.5 と表 5.6 で整理したサブシステムテストケースモデルに関して，システム全体のテストケースとの関係は表 5.7 のようになる．TC2 のテスト実行のために必要な構成は，Shopserver-TC2 を満たす店舗サーバ 1 台と，Register-TC2-1 と Register-TC2-2 を満たすレジをそれぞれ 1 台以上という意味になる．TC4 に関しては，店舗システムの構成を 2 つ作成することを示しているため 2 行に分かれている．

この手順では，システム可変性モデルの Choice ノードのみを用いて実施し，VClassifier ノードと Variable ノードに関して，対応するサブシステムテストケースモデルの可変性を束縛することはない．その理由は，システム可変性モデルの VClassifer ノードや Variable ノードに記載された関係記述を用いても，サブシステムテストケースモデルの可変性を束縛することができないからである．例えば，2.2.2 節の業務用空調機の例における馬力数を表すような変数の可変性である Variable ノードを考える．システム可変性モデルには，複合システムが提供する冷媒能力を示す Variable ノード「Providing HP」があるため，このノードに対してシステムテストケースモデルでは，上限値と下限値を指定する．一方で，室外機のサブシステム可変性モデルには，1 台の室外機で提供する冷媒能力を示す Variable ノード「HP」があり，先の Variable ノード「Providing HP」と関係記述によって関係性が定義されている．この関係記述は，複合システムを構成する室外機サブシステムの Variable ノード「HP」の合計が Variable ノード「Providing HP」という意味である．つまり，Variable

表 5.7: システムテストケースモデルとサブシステムテストケースモデルの関係

Testcase model	Shopserver subsystem testcase model	Register subsystem testcase model
TC1	Shopserver-TC1	Register-TC1
TC2	Shopserver-TC2	Register-TC2-1, Register-TC2-2
TC3	Shopserver-TC3	Register-TC3
TC4	Shopserver-TC4-1	Register-TC4-1
TC4	Shopserver-TC4-2	Register-TC4-2
TC5	Shopserver-TC5	Register-TC5

ノード「Providing HP」が束縛され，上限値・下限値による変数の範囲の束縛がされたとしても，複数台の室外機の合計値が束縛されるだけで，個別の室外機サブシステムを絞り込む情報は得られない．

5.2.3 手順 3: サブシステムリゾリューションモデルの抽出

この手順では，得られたサブシステムテストケースモデルごとに，そのモデルを満たすサブシステムリゾリューションを抽出する．手順 3 で得られたサブシステムテストケースモデルごとに，図 5.5 と図 5.5 にあるサブシステムリゾリューションが対応可能かどうかを確認する．

提案手法では，SMT ソルバを用いた充足可能性問題に変換して対応関係を確認している．サブシステムテストケースモデルを 1 つとり，各可変性ノードを変数として定義する．Choice ノードに関しては Bool の変数で定義を行い，チェックマークが入っており選択として束縛されているものを True で，空欄の非選択で束縛している箇所を False と具体値を与える．サブシステムテストケースモデルのノードで，「∗」で示される束縛していない箇所に関しては具体的な値を付与しない．一方でサブシステムリゾリューションモデルに関しても，各可変性ノードを変数として定義し，可変性の束縛結果に関して具体的な値を与える．そして，サブシステムテストケースモデルとサブシステムリゾリューションモデルにおいて，同じ可変性ノードを表す変数に関して等号でつないだ制約を加えて，これらの式が充足するかを確認する．この充足可能性問題に反例が見つからず，具体的な値の組が見つかった場合，そのサブシステムリゾリューションは，サブシステムテストケースモデルを満たすことが確認できる．

例えば，レジにおけるサブシステムリゾリューションモデル REGISTER_PRO とサブシステムテストケースモデル TC1 についての充足可能性問題に関して，SMT-LIB2 [47] で記述した例を図 5.11 に示す．この記述を解くと，充足する値の組み合わせを得られるため，サブシステムテストケース TC1 はレジのリゾリューションモデル REGISTER_PRO によってテスト可能と分かる．このような充足可能性問題を，各サブシステムテストケースモデルとサブシステムリゾリューションモデルの全ての組み合わせ実施することで，求めるサブシステムリゾリューションモデルが得られる．

表 5.8 と表 5.9 に，抽出したサブシステムリゾリューションモデルをサブシステムテストケースごとに示す．

5.2.4 手順 4: システム構成候補の作成

この手順では，手順 3 にて作成したサブシステムリゾリューションの候補を組み合わせて，システム構成の候補を作成する．図 5.6 に示す構造可変性モデルでは，店舗サーバ 1 台と複数台のレジによって店舗シス

```
; Definition
(declare-fun TEST_REGISTER_COM_OLD () Bool)
(declare-fun TEST_REGISTER_COM_NEW () Bool)
(declare-fun TEST_REGISTER_B () Bool)
(declare-fun TEST_REGISTER_CURRENCY_JPY () Bool)
(declare-fun TEST_REGISTER_CURRENCY_USD () Bool)
(declare-fun RESO_REGISTER_COM_OLD () Bool)
(declare-fun RESO_REGISTER_COM_NEW () Bool)
(declare-fun RESO_REGISTER_B () Bool)
(declare-fun RESO_REGISTER_CURRENCY_JPY () Bool)
(declare-fun RESO_REGISTER_CURRENCY_USD () Bool)

; Constraint
(assert
   (and true
      (= TEST_REGISTER_COM_OLD RESO_REGISTER_COM_OLD)
      (= TEST_REGISTER_COM_NEW RESO_REGISTER_COM_NEW)
      (= TEST_REGISTER_B RESO_REGISTER_B)
      (= TEST_REGISTER_CURRENCY_JPY RESO_REGISTER_CURRENCY_JPY)
      (= TEST_REGISTER_CURRENCY_USD RESO_REGISTER_CURRENCY_USD)
   )
)

; Register Testcase Register-TC1
(assert (= TEST_REGISTER_COM_NEW true))
(assert (= TEST_REGISTER_CURRENCY_JPY true))

; Register Resolution REGISTER_PRO
(assert (= RESO_REGISTER_COM_OLD false))
(assert (= RESO_REGISTER_COM_NEW true))
(assert (= RESO_REGISTER_B true))
(assert (= RESO_REGISTER_CURRENCY_JPY true))
(assert (= RESO_REGISTER_CURRENCY_USD false))
```

図 5.11: サブシステムリゾリューションの充足可能性問題の例

表 5.8: 店舗サーバのリゾリューションモデル候補

Subsystem testcase model	Subsystem resolution model
Shopserver-TC1	SHOP_PERFECT
Shopserver-TC2	SHOP_PERFECT, SHOP_PRO
Shopserver-TC3	SHOP_PERFECT, SHOP_STANDARD
Shopserver-TC4-1	SHOP_PERFECT, SHOP_PRO, SHOP_STANDARD
Shopserver-TC4-2	SHOP_PERFECT, SHOP_PRO, SHOP_STANDARD
Shopserver-TC5	SHOP_PERFECT

テムを構成する．そのため，その指定された台数を満たすように，表 5.8 と表 5.9 からリゾリューションモデルを選択し，生じうる組み合わせを網羅させる．システム構成を導出するためのサブシステムテストケースモデルの組み合わせには表 5.7 を用いる．

POS レジシステムのシステム構成候補の導出結果を表 5.10 に示す．この仮想の POS レジシステムでは，5 つのテストケースに対して 16 個のシステム構成を候補として導出した．表 5.10 の列は順に，構成候補の番号，テストケース，店舗サーバのサブシステムテストケース，レジのサブシステムテストケース，店舗サー

表 5.9: レジのリゾリューションモデル候補

Subsystem testcase model	Subsystem resolution model
Register-TC1	REGISTER_PRO, REGISTER_STANDARD
Register-TC2-1	REGISTER_STANDARD
Register-TC2-2	REGISTER_CLASSIC
Register-TC3	REGISTER_PRO
Register-TC4-1	REGISTER_PRO
Register-TC4-2	REGISTER_STANDARD, REGISTER_CLASSIC
Register-TC5	REGISTER_PRO

表 5.10: システム構成の候補

System configuration	Testcase	ShopServer subsystem testcase model	Register subsystem testcase model	ShopServer resolution model	Register resolution model
SC1	TC1	Shopserver-TC1	Register-TC1	SHOP_PERFECT	REGISTER_PRO
SC2	TC1	Shopserver-TC1	Register-TC1	SHOP_PERFECT	REGISTER_STANDARD
SC3	TC2	Shopserver-TC2	Register-TC2-1, Register-TC2-2	SHOP_PERFECT	REGISTER_STANDARD, REGISTER_CLASSIC
SC4	TC2	Shopserver-TC2	Register-TC2-1, Register-TC2-2	SHOP_PRO	REGISTER_STANDARD, REGISTER_CLASSIC
SC5	TC3	Shopserver-TC3	Register-TC3	SHOP_PERFECT	REGISTER_PRO
SC6	TC3	Shopserver-TC3	Register-TC3	SHOP_STANDARD	REGISTER_PRO
SC7	TC4	Shopserver-TC4-1	Register-TC4-1	SHOP_PERFECT	REGISTER_PRO
SC8	TC4	Shopserver-TC4-1	Register-TC4-1	SHOP_PRO	REGISTER_PRO
SC9	TC4	Shopserver-TC4-1	Register-TC4-1	SHOP_STANDARD	REGISTER_PRO
SC10	TC4	Shopserver-TC4-2	Register-TC4-2	SHOP_PERFECT	REGISTER_STANDARD
SC11	TC4	Shopserver-TC4-2	Register-TC4-2	SHOP_PRO	REGISTER_STANDARD
SC12	TC4	Shopserver-TC4-2	Register-TC4-2	SHOP_STANDARD	REGISTER_STANDARD
SC13	TC4	Shopserver-TC4-2	Register-TC4-2	SHOP_PERFECT	REGISTER_CLASSIC
SC14	TC4	Shopserver-TC4-2	Register-TC4-2	SHOP_PRO	REGISTER_CLASSIC
SC15	TC4	Shopserver-TC4-2	Register-TC4-2	SHOP_STANDARD	REGISTER_CLASSIC
SC16	TC5	Shopserver-TC5	Register-TC5	SHOP_PERFECT	REGISTER_PRO

バのリゾリューション，レジのリゾリューションを表している．店舗サーバのリゾリューションとレジのリゾリューションの列に記載されているサブシステム製品が組み合わさって，テストケースを実行可能な構成候補として導出できている．

表 5.10 には，サブシステムリゾリューションが選択されているが，それぞれの台数に関する記載はないが，この手順においてはそれぞれの台数のバリエーションも作成する．ただし，それぞれの台数に上限を付与しないと無限の組み合わせが生じてしまうため，上限設定値を与えて組み合わせ数を制限する．この上限値には，固定値を与える方法や，テスト実施時に準備可能な台数を与える方法がある．

5.2.5 手順 5: システム構成候補ごとのシステムリゾリューションモデル作成

この手順では，表 5.10 で得られたシステム構成の候補に対して，指定されたサブシステムリゾリューションが持つ束縛された可変性と関係記述を利用して，システム可変性モデル上はどのような実現となっているかを求める．その結果を，表 5.11 に示す．例えば，システム構成 SC1 では，店舗サーバ SHOP_PERFECT1 台とレジ REGISTER_PRO が複数台という構成である．この組み合わせでは，両者は新プロトコルに対応しており，システム上は新プロトコルの通信となる．また，店舗サーバ SHOP_PERFECT は機能 A を実装し，レジ REGISTER_PRO は機能 B を実装しているため，表 5.11 のようになる．

表 5.11: システム構成候補によるシステムリゾリューションモデル

	COM				Func			
System configuration	OLD_{SYS}	NEW_{SYS}	MIX_{SYS}	A_{SYS}	B_{SYS}	C_{SYS}	JPY_{SYS}	USD_{SYS}
SC1		✓		✓	✓		✓	
SC2		✓		✓			✓	
SC3			✓	✓		✓	✓	
SC4			✓			✓	✓	
SC5		✓		✓	✓		✓	
SC6	✓			✓	✓		✓	
SC7		✓		✓	✓		✓	
SC8		✓			✓		✓	
SC9	✓			✓	✓		✓	
SC10		✓		✓			✓	
SC11		✓				✓	✓	
SC12	✓					✓	✓	
SC13	✓			✓		✓	✓	
SC14	✓					✓	✓	
SC15	✓					✓	✓	
SC16		✓		✓	✓		✓	

5.2.6　手順 6: システムリゾリューションモデルの検証

表 5.11 に記載されているシステム構成候補ごとのシステム可変性モデルの実現は，システム構成候補として識別したサブシステムに対して生じうる組み合わせを作成したものである．そのため，システム可変性モデルに記載された制約を満たすかどうかは手順 5 の時点では不明である．そこで手順 6 では，システム可変性モデルに記載の制約を，各システム構成の候補が満たすかどうかを検証する．

例えば，POS レジシステムのシステム可変性モデルでは，図 5.7 に示すように，機能 A は機能 B を必要とする制約が記載されている．この制約を満たすためには，機能 A が実装される場合，機能 B が実装されている必要がある．ここで，表 5.11 を見ると，システム構成の SC2，SC3，SC10，SC13 はこの制約に違反していることが分かるため，これらの構成を除外する．

制約を満たさないシステム構成を除外した結果が，各テストケースを実行可能なシステム構成を表すことになる．よって，課題 2-2 の「テストケースごとに実施可能なシステム構成を導出方法が必要」を解決できている．

さらに本手順では，システムテストケースモデルの Variable ノードや VClassifer ノードに関しても，検証を行う．手順 2 の「サブシステムテストケースモデルの導出」から本手順まで，システムテストケースモデルの Choice ノードに焦点を当てて，システム構成候補を導出し，システム可変性モデルと突き合わせて検証する方法を述べた．例えば，2.2.2 節の業務用空調機の例において，システム可変性モデルには，複合システムが提供する冷媒能力を示す Variable ノード「Providing HP」がある．このノードに対してシステムテストケースモデルでは，上限値と下限値を指定することでテストを実施するために必要な総馬力の表現を行う．本手順では，得られたシステムリゾリューション上に記述されている複合システム全体で提供する冷媒能力

表 5.12: システム構成とテストケースの対応

System configuration	TC1	TC2	TC3	TC4-1	TC4-2	TC5
SC1, SC5, SC7, SC16	✓		✓	✓		✓
SC4		✓			✓	
SC6, SC9			✓	✓		
SC8				✓		
SC11					✓	
SC12					✓	
SC14					✓	
SC15					✓	

値，つまり複合システムを構成するサブシステムそれぞれが持つ冷媒能力値の合計値が，その上限値・下限値の間にあることを検証する．このようにして，Variablen ノードや VClassifer ノードの検証が可能となる．

5.2.7 手順 7: テストケースとシステムリゾリューションモデルの対応付け

手順 6 までで課題 2-2 を解決したので，次は課題 2-3 の「より少ないシステム構成で必要なテストケースセットを網羅する方法が必要」を解決する．まず，導出されているシステム構成と，その構成によって実現されるシステムリゾリューションモデルのそれぞれで，他のテストケースの実行可能性について確認する．表 5.11 に示されるシステム構成候補の実現するシステム可変性モデルと，表 5.4 に示されるテストケースごとに必要なシステム可変性モデルを突き合わせて行う．

また，本手順において，同じシステムリゾリューションモデルを持つシステム構成が導出されている場合は統合する．例えば，表 5.11 において，システム構成の SC1 と SC5 と SC7 と SC16 が同じシステムリゾリューションモデルを示しており，SC6 と SC9 も同様のため，それらを統合する．これは，サブシステムテストケースモデルを満たすように，システム構成をそれぞれ異なるものとして導出したが，それらが複合システムに表出する本質としての機能が同一であることを示している．

POS レジシステムにおけるシステム構成候補と，その構成における実施可能なテストケースの対応に関して表 5.12 に示す．

5.2.8 手順 8: システム構成の導出

最後の手順では，手順 7 で得られたシステム構成とテストケースの対応表である表 5.12 に対して充足可能性問題を用いて，より少ない数のシステム構成でテストケースを網羅する組み合わせを選択する．まず，「システム構成の数が 1 であった場合で，全てのテストケースを網羅するようなシステム構成がある」という充足可能性問題に変換し，それをソルバによって解く．この時，テストケースを網羅するようなシステム構成があれば，それを結果として得る．Unsat で解がなかった場合は，「システム構成の数が 2 であった場合で，〜」と，システム構成数を増やして同様の充足可能性問題に変換し解く．この手順において，入力したテストケースそれぞれに対応するシステム構成が候補としてあるため，この充足可能性問題は最大でもテストケース数分繰り返すことで，解が得られる．この方法を用いることで，表 5.12 のようなシステム構成とテストケースの対応表の中では，最小数のシステム構成でテストケースを網羅する組み合わせを導出できる．

実際に表 5.12 にこの手順を適用すると，SC1 と SC4 の組み合わせによってテストケースが全て網羅され

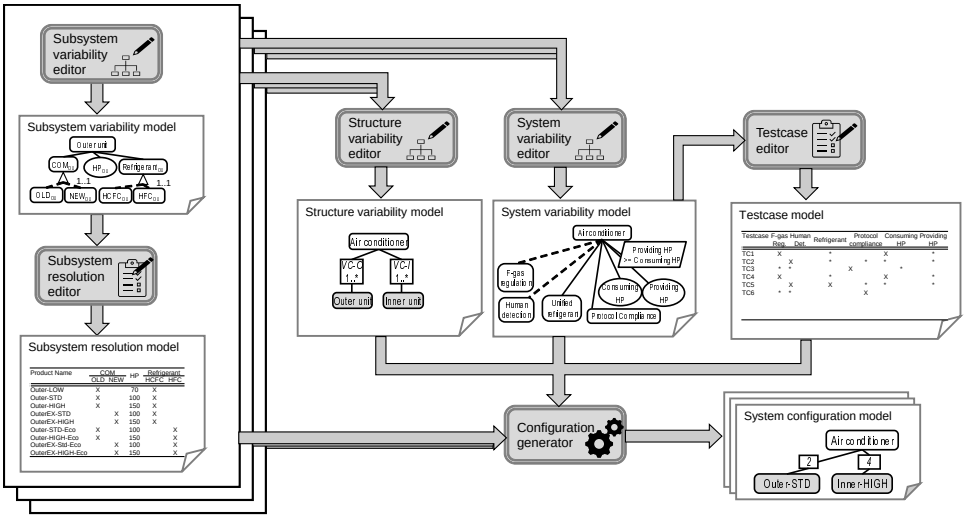

図 5.12: CT-CVL のシステム構成導出機能概要

る．SC1 は，店舗サーバ SHOP_PERFECT と REGISTER_PRO の組み合わせであり，SC4 は，店舗サーバ SHOP_PRO とレジ REGISTER_STANDARD とレジ REGISTER_CLASSIC の組み合わせである．

5.3　Configuration Tool using CVL(CT-CVL) のシステム構成導出機能

4.3 節にて紹介した可変性モデリングツール CT-CVL に，5.2 節で述べたテストケースの記述とシステム構成導出を行う機能を開発した．図 5.12 に，CT-CVL のモデリング機能の概要を示す．図 4.9 と比較して，可変性モデリング機能に加えて，テストケースを分析してシステム可変性モデル上に可変性の束縛として表現するテストケースエディタと，システム構成導出部が加わっている．可変性モデルを満たすリゾリューションの導出や制約の充足判定において，CT-CVL は充足可能性問題を解く SMT ソルバである CVC4 [48] を用いている．

5.3.1　テストケースエディタ

図 5.13 に，テストケースエディタのスクリーンショットを示す．4.3.2 節で述べたリゾリューションエディタとほぼ同様のインターフェースを持ち，ツリーコンポーネント上での操作を行う．リゾリューションエディタとの違いは，リゾリューションエディタが可変性を残らず束縛したモデルを記述するのに対し，テストケースエディタでは表現したいテストケースに関連する箇所のみを束縛する．つまりテストケースエディタにて作成するのはシステム可変性モデルに対する束縛可変性モデルとなる．そこで，例えば Choice ノードには可変性の束縛を示す True と False 以外に，可変性をそのまま残すという意味の NotCare を選択できる．この画面にて，右クリックメニューから「Solve TestSet」を選択すると，自動システム構成導出の処理が始まる．

5.3.2　システム構成の導出

図 5.14 に，システム構成導出結果のスクリーンショットを示す．左側に導出したシステム構成がツリーコンポーネント上に表示される．「Result」が 1 つのシステム構成を示しており，その下位には，構造可変性モデルのリゾリューションモデルとしてシステム構成が示される．

画面の右側には，左側の画面でシステム構成を選択すると，対応して実施可能となるテストケースが木構

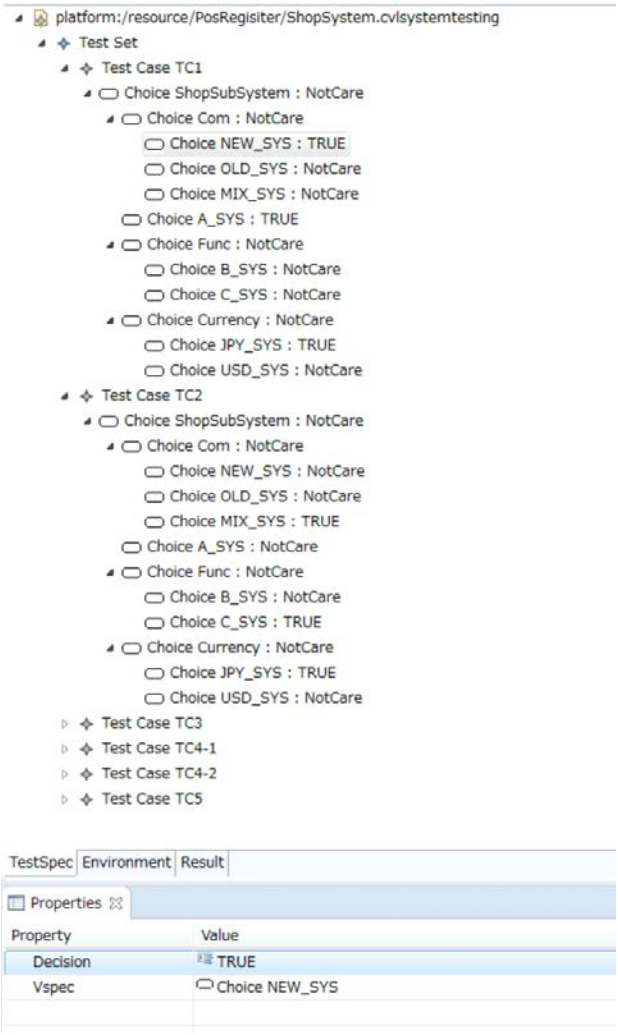

図 5.13: テストケースエディタ

造で表示される．こちらの画面はテストケースエディタで記入したものが選択的に表示されている．

本ツールのシステム構成導出機能は，現在 Choice ノードと VClassifier ノードのみを対象に充足可能性問題を解く実装となっており，Variable ノードには対応していない．

4.3 節にて紹介した CT-CVL の可変性モデリング機能は，図 2.10 におけるサブシステムとシステムのプロブレムスペースに対応し，本章で述べたテストケースエディタとシステム構成の導出機能はシステムのソリューションスペースに対応する．ここで，2.1.3 節の四象限の説明にて，ソリューションスペースでは応用として様々なソフトウェア資産に適用可能と述べた．そして，3.1.4 節で述べた商用ツール pure::variants [37] は，様々な外部ツールと連携して種々のソフトウェア資産に対応可能である．その一方で，CT-CVL はソリューションスペースにおいてテストケースとシステム構成のみを扱っているため，応用範囲は限定的である．CT-CVL では，CVL を提案した研究グループが開発した CVL Tool[*2] [49] と同じ CVL メタモデルを利用しているため，同様のメタモデルを利用した CVL ツールと連携可能である．

[*2] 現在は，ツール及びメタモデルは公開停止．

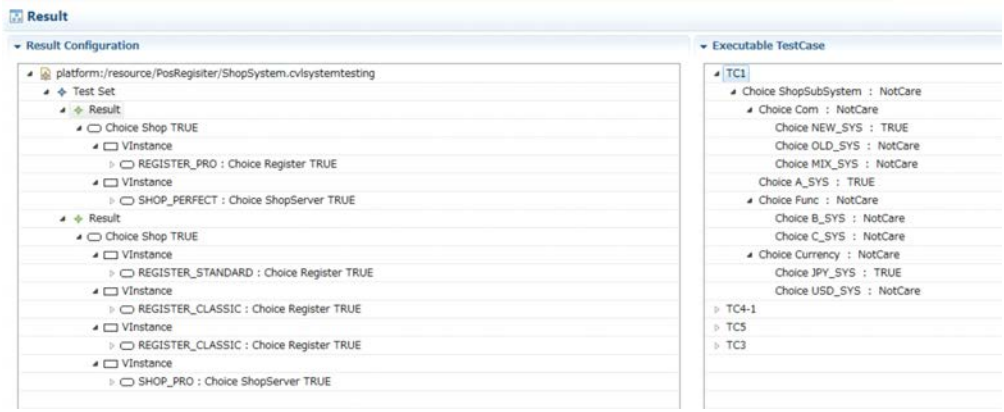

図 5.14: システム構成の導出結果

5.4　評価実験

本節では，複合システムにおいて，提案したテストケースを網羅するシステム構成の導出手法の評価を行う．本評価におけるリサーチクエスチョンを以下に示す．

RQ3　テストケースを網羅する，より少ない数のシステム構成を導出可能か？
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？
RQ5　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？

本節の以降では，まず，評価方法について述べる．そして結果を示した後に，考察を述べる．

5.4.1　評価方法

5.1.1 節で述べた 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に関して被験者実験による評価を行った．提案手法で作業が必要となるのは，5.3 節のツールを用いた場合，以下の作業となる．

1. サブシステム可変性分析
2. 構造可変性分析
3. システム可変性分析と関係記述の定義
4. システムテストケースモデルの作成

この評価実験では，複合システムの可変性モデルに関してはスコープ外とするため，上記作業における 1〜3 に関しては筆者が行い，4 のシステムテストケースモデルは筆者と実験参加者が行った．この作業を行う実験参加者は，可変性モデルを記述した経験はないが，実務としてソフトウェア工学に関わっている 2 名 (経験 6 年，経験 11 年) に依頼した．こちらの 2 名に関しては，提案手法の説明を事前に行い，システムテストケースモデルを作成した．また対照評価として，同じ POS レジシステムの事例に対して，提案手法を用いる参加者とは別に 3 名（インターン学生，経験 3 年，経験 9 年）に参加してもらい，5.1.1 節の仕様とテスト項目のみを渡し，手作業によるシステム構成の導出を行った．

RQ3 に関しては，まず，提案手法によってシステム構成が導出できたかどうかを確認する．次に，手作業で導出したシステム構成の数と，提案手法で導出した数の比較によって評価する．続いて，RQ4 に関しては，作業時間の比較を行う．RQ5 に関しては，属人性の評価として，手作業で実施した際のシステム構成結果のばらつき具合と，提案手法によって導出したシステム構成のばらつき具合の比較を定性的に行う．

表 5.13: 提案手法を用いた所要時間

作業	実施者	所要時間
1. サブシステム 可変性分析 2. 構造可変性分析 3. システム可変性分析と 関係記述の定義	筆者	60 分
4. システム	筆者	30 分
テストケース	経験 6 年	35 分
モデルの作成	経験 11 年	45 分
システム構成導出	CT-CVL	1 分

表 5.14: 手作業による所要時間

作業	実施者	所要時間
システム 構成導出	インターン	90 分
	経験 3 年	40 分
	経験 9 年	90 分

5.4.2 結果

提案手法を用いた評価実験では，筆者が作業の 1〜3 を行ったため，適切に可変性モデルを記述できた．そして，筆者及び実験参加者 2 名に関して，システムテストケースモデルの作成作業は適切に行えたことを確認できた．その結果，表 5.12 に示すシステム構成の中から SC1 と SC4 の組み合わせを導出でき，1 つのシステム構成によって全てのテストケースを網羅できている．提案手法を用いた実験の所要時間を表 5.13 に示す．ただし，経験 6 年の参加者に関しては，実験開始直後に，サブシステムテストケースモデルの作成を始めていた．実験前に行った提案手法の説明に，サブシステム可変性モデルも含めた説明していたため，実験の作業にサブシステムに関する作業が必要と誤解していた．そのため，再度，実験内容の説明を行った後，システムテストケースモデルの作成を行った．表 5.13 の経験 6 年目の作業時間は，当初のサブシステムに関する作業時間も含んでいる．

また，対称評価として，手作業にて行った結果，適切にテストケースを網羅するシステム構成の導出ができなかった．手作業にて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，正しくテストケースを網羅していたが 3 つのシステム構成を導出した．経験 3 年の方が導出したテストケースを網羅するシステム構成は 1 つだが，無関係なサブシステムリゾリューションが加えられていた．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．それぞれに結果に関してヒアリングを行った結果，5.1.1 節に記載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．システム構成導出にかかった所要時間は，インターンと経験 9 年の方が 90 分，経験 3 年の方が 40 分となった (表 5.14).

5.5 考察

本節では，被験者実験の考察について述べ，その後，提案手法に関して実際の複合システム開発への適用に関する考察を述べる．

5.5.1 評価実験の考察

本節では，リサーチクエスチョンの RQ3 から RQ5 に沿って，実験について考察を述べ，妥当性への脅威についてまとめる．

RQ3 テスト項目を網羅する，より少ない数のシステム構成を導出可能か？

対照実験の手作業によるシステム構成の導出では，3 名中 2 名がテストケース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．提案手法では，テストケースそれぞれにおいて実施可能なシステム構成を網羅的に抽出し，それらの中での最小の構成数で全てのテストケースが網羅できる組み合わせを抽出するため，より少ない数の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 つを得ることができた．

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？

対照実験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．提案手法を用いた実験では，可変性モデルの作成に 60 分かかっているが，これはテスト工程より前に行う工程のため，比較を行う時間としてはシステムテストケースモデルの作成のみを考える．システムテストケースモデルの作成では 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，現実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．

RQ5 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？

対照実験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判明した．一方の提案手法によるケーススタディでは 3 人が同様のシステムテストケースモデルを作成できたため，属人性を低減している．ただし，1 名が提案手法を誤解してシステムテストケースモデルを作成できなかったため，提案手法そのものの習熟が必要となる．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，それぞれ不要な機器を含むケースと，テストケースを網羅していないケースとなる．

5.5.2　妥当性への脅威

内的妥当性への脅威として，提案手法を用いた実験では，複合システム向けの可変性モデルの作成は筆者が行っており，提案手法への習熟度が高いことが挙げられる．さらに協力者が行ったシステムテストケースモデルの作成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，事例として用いた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全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．

提案手法では，実施したいテストケースをシステム可変性モデル上にマッピングする形でシステムテストケースモデルを作成する．内的妥当性への脅威として，テストケースの求める複合システムの要件が，システム可変性モデル上に表現されていない可能性がある．これは，システム可変性モデルを分析する際に，サブシステムの組み合わせによって生じる複合システム全体の特徴を漏れなく抽出できていた場合，システム可変性モデル上に表現されていない要件はシステム構成に影響しないため，問題とならない．しかし，実際の運用上は，サブシステムの組み合わせによって生じる特徴を漏れなく抽出するのは困難である可能性があるため，こういった事例が生じた場合はテストケースが求める要件をシステム可変性モデルに加筆し対応する．この脅威に関しては，複合システムの可変性分析とテストケースの分析が相互に行き来し，インクリメンタルに行うことで解決できる．

テストケースを網羅するシステム構成の導出に関して，特定のテストケースにおいて見つかったシステム構成が他のテストケースをカバーすることを期待して，充足可能性問題に変換して解くことにより，その中

表 5.15: 実際の複合システムの可変性規模

サブシステムの種類	サブシステムリゾリューション数	可変数	生じうる組み合わせ数
サブシステム 1	205	21	41,287,680
サブシステム 2	217	49	1,321,656,957,051,863,040
サブシステム 3	40	42	4,398,046,511,104
サブシステム 4	63	10	27,648

での最小数のシステム構成でテストケースを網羅させている．一方で，複数のテストケースを実施可能なように，システム構成を合成していくことで，さらに少ない数のシステム構成でテストケースを網羅できる可能性があり，内的妥当性への脅威となる．しかし，このシステム構成の合成に関しては，構造可変性モデルによって強く制限されることがある．例えば，今回の POS レジシステムでは，店舗システムにおいて 1 台の店舗サーバのみがシステム構成に存在する．システム構成の合成を考えたとき，店舗サーバの機種が異なるシステム構成は，合成することができない．店舗サーバの機種が同一だったとしても，それぞれのシステム構成に属していたレジがどのように合成されるかを定義しなければならない．そのため，システム構成の合成は困難である．

提案手法では，実施したいテストケースを入力としてシステム構成導出を行っており，入力となるテストケースの生成方法やその十分性についてはスコープ外としている．3.2 節で述べたように，フィーチャモデルからテスト設計を行う既存手法 [19–22] がある．これらの手法と組み合わせることによって，テストケースを自動生成したり，入力としたテストケースと自動生成したテストケースを突き合わせて十分性を評価したりできる．

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挙げられる．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．

5.5.3 実際の複合システム開発への適用に向けて

5.5.2 節の内的妥当性への脅威で言及があったように，実際の複合システムの仕様を用いて，システム構成導出手法について評価をする必要がある．そこで，実際の複合システムの仕様を用いて試行を行った．

使用した複合製品に関しては，製品仕様情報のため詳細は公開できないが，その可変性の規模を表 5.15 に示す．サブシステムは 4 種類あり，この複合システムにおいて，サブシステム 1 とサブシステム 2 が組み合わさり主要な機能を提供する．サブシステム 1 に対して，複数のサブシステム 2 が接続される形になる．サブシステム 3 は，サブシステム 2 に接続されるコントローラ装置であり，サブシステム 4 は全体を集中的に管理する装置になる．それぞれのサブシステムの種類に対して，製品の数を「サブシステムリゾリューション数」の列に示す．「可変数」の列は，可変性モデルにおいて束縛できる可変箇所の数を示す．Choice ノードの選択可能分解や，グループ多重度分解が行われた箇所を 1 つと数える．また，Variable ノードを 1 つと数えている．例えば，図 5.4 の POS レジシステムにおけるレジのサブシステム可変性モデルでいうと，「COM」「Func」「Currency」のそれぞれでグループ多重度分解があるため，可変数は 3 となる．本章で用いた例に比べ，実際の複合システムでは，可変数だけで見ても非常に大きい．そして，それぞれの可変箇所での選択肢の数は，2 から 14 まで生じており，理論的にそれらすべての組み合わせ数を算出すると，「生じうる組み合わせ数」の列のようになる．この数はあくまで可変箇所の組み合わせなので，実際には製品上生じない組み合わせが多数生じる．

このような実際の複合システムに関して，サブシステム 1 とサブシステム 2 の可変性のみに限定して，5.3

節で述べたツール CT-CVL に入力し，テストケースからシステム構成導出を試行した．その結果，手順 4 のシステム構成候補の作成にて，処理が終了せずシステム構成の導出に失敗した．生じうるサブシステムリゾリューションの組み合わせを作成しているため，サブシステムリゾリューションの数の組み合わせ数が膨大になったことが原因であった．今後は，サブシステムリゾリューションの選択に関して，何らかの指針をもとにフィルタリングを行う機能や，実際にテストで用意できる機器を上限とするような拡張が求められる．

第 6 章

おわりに

6.1 結論

本研究では，単独の機器で機能を提供するのではなく，複数のサブシステムが接続され連携して機能を提供する複合システムを対象とした可変性モデルの提案を行い，また，それを用いたテストケースを網羅するシステム構成導出手法の提案を行った．

複合システムでは，サブシステムが独立して派生開発されているため，各々の可変性モデルを持つ．そして，これらのサブシステムが組み合わさり機能を提供する複合システムには，2 つの異なるレベルの可変性が生じるため，1 つの単純な可変性モデルで表現が困難という課題が生じる．また，複合システムには，複数の種類のサブシステムが複数台接続されるため，サブシステムを横断する制約や要求において，台数や量などのインスタンス多重度に関する表現が曖昧となる課題がある．そこで，本研究では，複合システムの可変性に関するモデルリング記法として，異なる概念を分離して表現するために，システム可変性モデルと構造可変性モデルを定義し，関係記述によってつなげる記法の提案を行った．関係記述においては，複合システムの制約や要求に対して生じるインスタンス多重度を識別し，適切な集約演算を行うためのピンサームーブメントアプローチを提案した．この可変性モデリング手法を，モデリングツール CT-CVL として実装した．

提案する可変性モデリング手法に関して，多くのサブシステムで構成される業務用空調機にて例証した．また，複雑な可変性記述の例として，クラウド事業者の 4 つの事例を記述することで，提案する可変性モデルの記法が既存の可変性モデリング手法と同等の記述能力を持つことを確認した．さらに，別の自作 PC システムの例を用いて，可変性モデルを記述する被験者実験を実施し評価を行った．この実験では，モデリング経験のある参加者が，提案する可変性モデルの記法とピンサームーブメントアプローチを用いて，システムの可変性モデルを正しく記述することができることを確認した．本被験者実験において，開発した CT-CVL ツールは，被験者がインスタンス多重度を特定するのに貢献した．

さらに，複合システムのシステムテストにおいて，派生開発を続けたことにより蓄積したテストケースを実施するためには，テストケースごとにどのシステム構成でテストすべきか導き出すことが困難という課題がある．そして，テストを実行する際に，システム構成の切り替えは，機器の移動や接続の切り替え，そして初期設定のやり直しなど非常に工数が高い．そのため，複数のテストケースを実施する際に，システム構成の切り替え回数を極力少なくしたいという要求がある．そこで，本研究では，提案した複合システム向けの可変性モデルを用いて，テストケースごとに実行可能なシステム構成を導出し，テストケース全てを網羅する少ない数のシステム構成の組を導出する手法を提案した．また，モデリングツール CT-CVL にシステム構成導出機能を実装した．

提案したテストケースを網羅するシステム構成導出手法を評価するために，POS レジシステムの例を用いて被験者実験を行った．対照実験として提案手法を用いない実験も同時に行った．その結果，提案手法を用いない方法では，適切にテストケースを網羅するシステム構成を導出できなかったのに対し，提案手法を用

いればテストケースを網羅する少ない数のシステム構成が導出できることを確認できた．

6.2　将来への展望

筆者は研究所に勤務しており，システム開発の中でもソフトウェアよりの開発現場に，ソフトウェア工学の技術を導入し，ソフトウェアの開発工数削減や品質向上を行う業務を行っている．特に特定の製品に限定されることなく，複数の製品の開発現場を見る機会に恵まれている．近年，様々な開発現場で課題となっているのが，複合システムの開発において，どのテストケースをどのシステム構成で，そしてどのタイミングでテストをすれば，複合システム全体として品質を確認したことになるかが不明であるというものである．ここで，タイミングというのは，システム開発の全体プロセスの中で，日々走る回帰テストや，新しいサブシステム製品が組み込まれたとき，そして不具合が検出されたときなど，タイミングごとの必要に応じたテストを行うという観点である．本論文で提案する範囲は，可変性モデルを定義してシステム構成を導出するというものだが，今後はシステム開発全体の品質検証に対して，スコープを広げていく必要がある．

本研究で実施した評価に関して，用いた事例は関連研究において記述されたクラウドプロバイダと仮想の業務用空調・自作 PC システム・POS レジシステムである．これらの事例は，実際の開発現場で生じている課題を解くために，実際の製品を参考に簡略化して作成されたものである．そのため，これらの事例だけではなく，実際の製品情報を用いた検証が行われるべきである．実際の製品情報を用いた場合，既存のサブシステム製品の数が多く，提案手法で導出するのが困難な可能性がある．今後，実際の製品への適用に向けた手法の改良が求められる．

本研究で提案する複合システム向けの可変性モデルにおいて，サブシステムの種類とインスタンス数を表現する構造可変性モデルを導入している．また，可変性モデル間に生じるインスタンス多重度を識別して集約演算子を付与するピンサームーブメントアプローチを提案している．しかしながら，提案するアプローチでは組み合わせのみを扱っており，サブシステムの接続におけるスター型やバス型などのネットワークトポロジの違いや，室内機をフロアに割り振るデプロイメントや，アドレス空間の管理などはスコープ外としている．今後は，構造可変性モデルにおいて，これらの概念を含む形で拡張する必要がある．この時，構造可変性モデルは有効非巡回グラフとなる可能性があるため，構造可変性モデルが木構造であることを前提としているピンサームーブメントアプローチも拡張が必要となる．

謝辞

本研究を進めるにあたり終始あたたかいご指導と激励を賜りました東京工業大学の佐伯元司教授には，心から感謝の意を表します．東京工業大学の林晋平准教授と Østfold University の Øystein Haugen 教授には研究に関して多大なるご指導をいただきました．深く感謝いたします．博士課程入学前に，トップエスイーにて多大なるご指導をいただきました早稲田大学の鷲崎弘宜教授に，深く感謝いたします．

博士課程に進学するにあたり，快く送り出してくださった日立製作所 横浜研究所の上長と職場の方々に感謝いたします．また，実験に協力くださった日立製作所 横浜研究所の皆様に深く感謝いたします．

最後に，これまで私をあたたかく応援してくれた妻 陽子，娘 美咲に心から感謝します．

参考文献

[1] Nicolas Anquetil, Birgit Grammel, Ismênia Galvão, Joost Noppen, Shakil Khan, Hugo Arboleda, Awais Rashid, and Alessandro Garcia. Traceability for model driven, software product line engineering. In *Proc. ECMDA Traceability Workshop*, pp. 77–86. SINTEF, 2008.

[2] 独立行政法人情報処理推進機構社会基盤センター. 組込みソフトウェア開発データ白書 2019. 2019.

[3] Paul C. Clements and Linda Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison-Wesley, August 2001.

[4] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering : Foundations, Principles and Techniques*. Springer, September 2005.

[5] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Carnegie-Mellon University, Software Engineering Institute, 1990.

[6] Joachim Bayer, Oliver Flege, Peter Knauber, Roland Laqua, Dirk Muthig, Klaus Schmid, Tanya Widen, and Jean Marc DeBaud. PULSE: A methodology to develop software product lines. *Proc. 1999 Symposium on Software Reusability*, pp. 122–131, 1999.

[7] Øystein Haugen, Andrzej Wasowski, and Krzysztof Czarnecki. CVL: Common variability language. In *Proc. 16th International Software Product Line Conference - Volume 2*, pp. 266–267, 2012.

[8] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proc. Fifth IEEE International Symposium on Requirements Engineering*, p. 249, USA, 2001. IEEE Computer Society.

[9] Lianping Chen and Muhammad Ali Babar. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, Vol. 53, No. 4, pp. 344–362, 2011.

[10] Object Management Group. Common Variability Language (CVL), 2012.

[11] Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. Staged configuration through specialization and multilevel configuration of feature models. *Software Process: Improvement and Practice*, Vol. 10, No. 2, pp. 143–169, 2005.

[12] Krzysztof Czarnecki and Michał Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In *Generative Programming and Component Engineering*, pp. 422–437. Springer, 2005.

[13] Danilo Beuche and Mark Dalgarno. Software product line engineering with feature models. https://www.pure-systems.com/fileadmin/downloads/pure-variants/tutorials/SPLWithFeatureModelling.pdf.

[14] Rational DOORS. https://www.ibm.com/support/knowledgecenter/SSYQBZ/doors_family_welcome.html.

[15] Danilo Beuche. Modeling and building software product lines with pure::variants. In *ACM International Conference Proceeding Series*, Vol. 2, p. 255, 2012.

[16] Mark W. Maier. Architecting Principles for Systems-of-Systems. *INCOSE International Symposium*, Vol. 6,

No. 1, pp. 565–573, 1996.

[17] 日立グローバルライフソリューションズ株式会社 業務用空調機器. https://www.hitachi-gls.co.jp/products/ac.html.

[18] Object Management Group. Object Constraint Language (OCL). Version 2.3.1, 2012. http://www.omg.org/spec/OCL/2.3.1/.

[19] Erika Mir Olimpiew and Hassan Gomaa. Model-based test design for software product lines. In *Proc. 13th International Software Product Line conference*, pp. 173–178, 2008.

[20] Aymeric Hervieu, Benoit Baudry, and Arnaud Gotlieb. PACOGEN: Automatic generation of pairwise test configurations from feature models. In *Proc. 22nd International Symposium on Software Reliability Engineering*, pp. 120–129. IEEE, 2011.

[21] Sebastian Oster, et al. MoSo-PoLiTe: tool support for pairwise and model-based software product line testing. In *Proc. Fifth International Workshop on Variability Modelling of Software-intensive Systems*, pp. 79–82. ACM, 2011.

[22] Takashi Kitamura, Ngoc Thi Bich Do, Hitoshi Ohsaki, Ling Fang, and Shunsuke Yatabe. Test-case design by feature trees. In *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*, pp. 458–473. Springer, 2012.

[23] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process Improvement and Practice*, Vol. 10, No. 1, pp. 7–29, 2005.

[24] Krzysztof Czarnecki and Chang Hwan Peter Kim. Cardinality-based feature modeling and constraints: A progress report. In *Proc. International Workshop on Software Factories*, pp. 16–20, 2005.

[25] Mark Oliver Reiser, Ramin Tavakoli Kolagari, and Matthias Weber. Compositional variability - concepts and patterns. In *Proc. 42nd Hawaii International International Conference on Systems Science*, pp. 1–10, 2009.

[26] Mark Oliver Reiser and Matthias Weber. Managing highly complex product families with multi-level feature trees. In *Proc. 14th IEEE International Requirements Engineering Conference*, pp. 146–155, 2006.

[27] Jaime Chavarriaga and Carlos Noguera. Propagating decisions to detect and explain conflicts in a multi-step configuration process. In *Proc. ACM/IEEE 17th International. Conference on Model Driven Engineering Languages and Systems*, pp. 337–352, 2014.

[28] Arnaud Hubaux, Patrick Heymans, Pierre-Yves Schobbens, and Dirk Deridder. Towards multi-view feature-based configuration. In *Requirements Engineering: Foundation for Software Quality*, pp. 106–112, 2010.

[29] Mikoláš Janota and Goetz Botterweck. Formal approach to integrating feature and architecture models. In *Proc. 11th International Conference on Fundamental Approaches to Software Engineering*, pp. 31–45, 2008.

[30] Øystein Haugen and Ommund Øgård. BVR – better variability results. In *Proc. 8th System Analysis and Modeling Conference*, pp. 1–15, 2014.

[31] Charles Krueger and Paul Clements. Enterprise feature ontology for feature-based product line engineering and operations. In *Proc. 21st International Systems and Software Product Line Conference - Volume A*, pp. 227–236, 2017.

[32] Bedir Tekinerdogan, Sami Duman, Hakan Caner, and Bülent Durak. Customizing a feature ontology for product line engineering within a system-of-systems context. In *Proc. 2019 International Symposium on Systems Engineering*, pp. 1–6, 2019.

[33] Marko Rosenmüller, Norbert Siegmund, Thomas Thüm, and Gunter Saake. Multi-dimensional variability modeling. In *Proc. 5th International Workshop on Variability Modeling of Software-Intensive Systems*, pp.

11–20, 2011.

[34] Andreas Classen, Quentin Boucher, and Patrick Heymans. A text-based approach to feature modelling: Syntax and semantics of TVL. *Science of Computer Programming*, Vol. 76, No. 12, pp. 1130–1143, 2011.

[35] Andreas Abele, Yiannis Papadopoulos, David Servat, Martin Torngren, and Matthias Weber. The CVM framework - a prototype tool for compositional variability management. In *Proc. 4th International Workshop on Variability Modelling of Software-Intensive Systems*, Vol. 37 of *ICB-Research Report*, pp. 101–105. Universitat Duisburg-Essen, 2010.

[36] Gustavo Sousa, Walter Rudametkin, and Laurence Duchien. Extending feature models with relative cardinalities. In *Proc. 20th International Systems and Software Product Line Conference*, pp. 79–88, 2016.

[37] pure-systems gmbh. https://www.pure-systems.com/.

[38] ptc. https://www.ptc.com/ja.

[39] SYSML. http://www.omgsysml.org/.

[40] Andreas Reuys, Sacha Reis, Erik Kamsties, and Klaus Pohl. *The ScenTED Method for Testing Software Product Lines*, pp. 479–520. Springer, 2006.

[41] Karthrin D. Scheidemann. Optimizing the selection of representative configurations in verification of evolving product lines of distributed embedded systems. In *Proc. 10th International Software Product Line Conference*, pp. 75–84, 2006.

[42] Anargyros Tsadimas, Mara Nikolaidou, and Dimosthenis Anagnostopoulos. Handling non-functional requirements in information system architecture design. In *Proc. Fourth International Conference on Software Engineering Advances*, pp. 59–64, 2009.

[43] 風戸広史, 林晋平, 小林隆志, 佐伯元司. ベイジアンネットワークを用いたソフトウェア実装技術の選択支援. 情報処理学会論文誌, Vol. 51, No. 9, pp. 1765–1776, 2010.

[44] Daisuke Shimbara, Shinpei Hayashi, Motoshi Saeki, and Øystein Haugen. Handling quantity in variability models for system-of-systems. *International Journal of Software Engineering and Knowledge Engineering*. (to appear).

[45] Daisuke Shimbara and Øystein Haugen. Generating configurations for system testing with common variability language. In *Proc. 17th International SDL Forum*, pp. 221–237, 2015.

[46] 新原敦介, 小川秀人, 鷲崎弘宜. 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. 情報処理学会論文誌, Vol. 55, No. 2, pp. 922–938, 2014.

[47] Clark Barrett, Aaron Stump, and Cesare Tinelli. The SMT-LIB Standard: Version 2.0, 2010. http://www.SMT-LIB.org.

[48] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In *Proc. 23rd International Conference on Computer Aided Verification*, pp. 171–177, 2011.

[49] Andreas Svendsen, Xiaorui Zhang, Franck Fleurey, Øystein Haugen, Gøran K. Olsen, and Birger Møller-Pedersen. CVL tool - modeling variability in SPLs. In *Proc. 14th International Conference on Software Product Lines. Workshop Proceedings (Volume 2 : Workshops, Industrial Track, Doctoral Symposium, Demonstrations and Tools)*, p. 299, 2010.