Vol.2022-SE-210 No.6
2022/3/11

可変性を持つシステムの確率的モデル検査手法

内藤裕暉[1]　岸知二[1]

概要：近年，可変性を持つシステムが増えており，SPL (Software Product Line)開発が重要視されている．従来のモデル検査手法の研究では，MTS や FTS を検証するツールとして SNIP などのファミリベースのモデル検査を行うことのできる特定のツールを用いた研究が多いが，汎用性に問題がある．一般的に広く用いられるモデル検査ツールで検証が行えるようになると有用性が増すが，製品群の製品をひとつひとつ検証しなければならず，複数製品の検証に対応していないという問題がある．本研究では可変性を持つシステムに対して，汎用的なモデル検査ツールを用いて，複数の製品を同時に検証できる手法を提案した．フィーチャの出現確率である FIP を用いて，バリアントの選択を確率的に行う遷移を FTS に付与することで，確率システムとして確率的モデル検査を行い，プロパティがどの程度の製品において成立するのか検証した．これにより SPL の検証を汎用的なモデル検査ツールで製品をひとつひとつ検証しなくてもモデル検査を行えるようになった．

キーワード：SPL，可変性，確率的モデル検査，FTS

Stochastic model checking method for variability-intensive system

HIROKI NAITO[†1]　TOMOJI KISHI[†1]

1. 背景

近年, 可変性を持つシステムが増えており, SPL (Software Product Line)開発が重要視されている．SPL とは，共通のフィーチャを持つソフトウェアプロダクトの製品群を指し，SPL 中の製品すべてが持つ特徴を共通性，一部の製品が持つ特徴を可変性と呼ぶ. また, 体系的な再利用に基づき SPL を効率的に開発する手法を SPL 開発と呼ぶ．具体的には，共通性と可変性を予め整理して，コア資産を作って，開発するアプローチである．可変性を持つシステムの特徴（フィーチャ）を記述するモデルにはフィーチャモデルなどがある．

ソフトウェア開発において，ソフトウェアの重要なプロパティを検証することは重要である．しかし，SPL 開発における可変性を持つシステムの検証は，潜在的な製品数がバリアントの組み合わせの数だけ存在する問題がある．そのため，単一製品の検証手法を SPL の検証に適用させることが難しい．そこで，可変性を考慮したモデル検査手法が提案されている．これらのモデル検査手法では，可変性を持つシステムのふるまいを表現するモデルである MTS (Modal Transition Systems) や FTS (Featured Transition Systems) などが用いられている．

従来の可変性を考慮したモデル検査手法の研究では，MTS や FTS を検証するツールとして SNIP などのファミリベースのモデル検査を行うことのできる特定のツールを用いた研究が多いが，汎用性に問題がある．一般的に広く用いられるモデル検査ツールで検証が行えるようになると有用性が増すが，製品群の製品をひとつひとつ検証しなければならず，複数製品の検証に対応していないという問題がある．

そこで，本研究では可変性を持つシステムに対して，汎用的なモデル検査ツールを用いて，複数の製品を同時に検証できる手法を提案する．フィーチャの出現確率である FIP を用いて, バリアントの選択を確率的に行う遷移を FTS に付与することで，製品をひとつひとつ検証しなくてもモデル検査を行えるようにする．具体的に，確率的モデル検査を行い，プロパティがどの程度の製品において成立するのか検証する．

2. 従来研究

2.1 Featured Transition Systems (FTS) のモデル検査[3][4]

FTS は，SPL の可変性を含むふるまいを表すモデルである．各遷移にフィーチャ式がラベルとして与えられたものである．ラベルのフィーチャ式はフィーチャモデルで定義されたフィーチャと関連付けられているため，存在しない製品の TS が導出されなくなる．

FTS の構成：($S, Act, trans, I, \gamma$)

- S：状態
- Act：アクション名
- $trans$：遷移（$S \times Act \times S$）
- I：初期状態
- γ：trans に対するフィーチャ式

1 早稲田大学
Waseda University

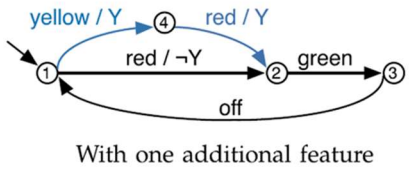

図 1 FTS の例[4]

可変性を持つシステムの FTS の例を図 1 に示す．“ ① → ④ ”と“ ④ → ② ”の遷移には“Y”がラベル付けされ，同時に“ ① → ② ”の遷移には“¬Y ”がラベル付けされる．これにより，FTS から導出される製品はすべて，①からの遷移が一つだけとなり，不必要な遷移を削除することができる．

Classen らは SNIP という専用のモデル検査ツールを用いた FTS のモデル検査手法を提案している． SNIP での検証で用いられる特別なアルゴリズムによって，FTS に対して全製品のプロパティ検証ができるようになり，プロパティを満たさない製品を出力できる．しかし，一般的な単一システムを検証するモデル検査ツールではこの検証方法を適応できない．

2.2 Featured Inclusion Probability (FIP) [6][7]

フィーチャモデルの特徴を捉える指標の一つに FIP がある．FIP は全製品数に対する各フィーチャが出現する製品数の確率を表している．また楊[7]はフィーチャモデルを命題論理式に変換することで FIP を高速に求める手法を提案している．本研究では，楊の提案する命題論理式を利用した手法でフィーチャやフィーチャの組み合わせの出現確率を求める．

2.3 確率的モデル検査 [8]

確率的モデル検査は，システムの実行上で特定のイベントが発生する確率を求める技術である．システムの振る舞いの記述には例えば離散時間マルコフ連鎖やマルコフ決定プロセスなどが用いられ，性質の記述には確率計算機論理などが用いられる．これにより性質記述で定義されるイベントが指定した確率範囲で発生するかどうかを求めたり，特定のイベントが発生する確率を求めたりすることができる．

汎用的なモデル検査ツールには，PRISM, PLASMA Lab, UPPAAL などがある．本研究では，PRISM というモデル検査ツールを用いて確率的モデル検査を行う．PRISM は確率モデルチェッカーであり，ランダムまたは確率的な動作を示すシステムの正式なモデリングと分析のためのツールである．プロパティ仕様言語には，時相論理 PCTL, CSL, LTL, PCTL*のほか、定量仕様などの拡張機能が組み込まれている．

3. 提案手法

3.1 着眼点

FTS の構成要素である遷移にラベル付けされるフィーチャ式に着目する．FTS の遷移にラベル付けされるフィーチャ式のフィーチャは，SPL のフィーチャモデルと関連付けられている．そこで，フィーチャの出現確率である FIP を用いることで確率を含むモデルを作成し確率的モデル検査を行うことができる．それにより SPL のどの程度の製品がプロパティを満たすのか定量的な検証を行うことができる．

3.2 拡張 FTS

拡張 FTS は，FTS にフィーチャの出現確率を付与したモデルである．SPL の製品を導出するためのフィーチャの選択を確率的に行うことができるように拡張している．

元の FTS の初期状態の前に，FTS のラベルに出現するフィーチャの数だけ状態を追加し，確率に応じてフィーチャが選択・非選択されるようにする．

拡張 FTS の構成：(S, Act, $trans$, I, γ, F, P, $trans^F$)

- S, Act, $trans$, I, γ は FTS と同じ
- F：γ に用いられるフィーチャ
- $P(F)$：F の出現確率
- $trans^F$：F の選択・非選択の遷移

FTS の初期状態 S の前に，すべてのフィーチャについて，フィーチャ f が出現する（$f = 1$ になる）遷移と f が出現しない（f = 0 になる）遷移を加える．その際，f が出現する遷移が出現確率 $P(f)$で遷移するように確率を付与する．拡張 FTS の基本形を図 2 に示す．

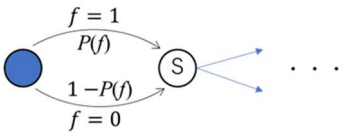

図 2 拡張 FTS の基本形

なお，この方法では FTS 上のパスの出現確率を正確に得ることはできずあくまで近似となる．明らかに排他的なフィーチャなどについては，それを反映した遷移を構成することでより正確な検証に近づくことが期待される．拡張 FTS の例を図 3 に示す．図 3(a)は基本形に沿った拡張 FTS であり，図 3(b)は明らかに排他的なフィーチャを反映した拡張 FTS である．

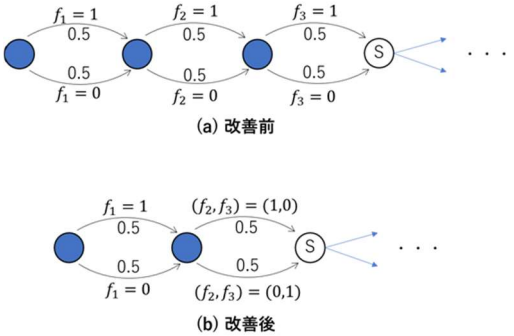

図 3 拡張 FTS の例

f_1, f_2, f_3はフィーチャを表し，(f_2, f_3) は排他の関係のフィーチャの組み合わせとする．排他であるf_2とf_3について，起こりうる状況は $f_2 \wedge \neg f_3$，あるいは $\neg f_2 \wedge f_3$の二通りである．それぞれの生起確率は，楊の手法を応用すれば求めることができる．それにより図 3 (a)のf_2, f_3のフィーチャ選択の遷移は，図 3 (b)のようにまとめることができる．

3.3 提案手法

提案手法の全体像を図 4 に示す．

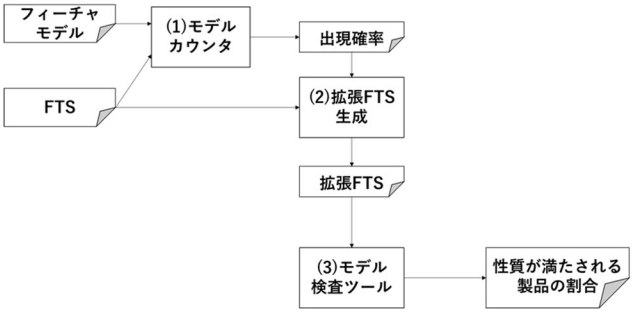

図 4　提案手法の全体像

提案手法の手順は以下の通りである．

(1) フィーチャモデルと FTS からモデルカウンタによってフィーチャの出現確率（FIP）を求める．
(2) (1)で求めた FIP を FTS にマッピングし，拡張 FTS を作成する．
(3) モデル検査ツールを用いて確率的モデル検査を行う．

3.3.1 （1）フィーチャの出現確率を求める

FTS の遷移のラベルのフィーチャ式に使われるフィーチャについて，楊の FIP を求める手法によって出現確率を求める．

フィーチャモデルによって表されるフィーチャ選択の制約を連言標準形で表し，モデルカウンタを実行して，全製品数を求める．FTS からは，ラベルのフィーチャ式に使われるフィーチャを抽出し，モデルカウンタによって抽出されたフィーチャを含む製品数を求める．求めた製品数により，フィーチャの出現確率を計算する．

3.3.2 （2）拡張 FTS を作成する

3.2 のように，手順(1)で得られたフィーチャの出現確率を FTS に付与し，拡張 FTS を作成する．

3.3.3 （3）確率的モデル検査を行う

汎用的な確率的モデル検査ツールに拡張FTS と検証したい性質を入力し，性質が満たされる製品の割合を出力する．

本研究では，モデル検査ツールとして PRISM を用いる．

確率的モデル検査によって行った検証の結果として，SPL の全製品の中でプロパティを満たす製品の割合を得ることができる．

4. 評価実験

4.1 評価方法

提案手法による確率的モデル検査で求められる「性質が満たされる割合」を，全製品の中で検証したい性質が満たされる製品の正確な割合と比較し，どの程度正しいかを確かめる．具体的に以下の二つを比較する．

① 提案手法により作成した拡張 FTS を汎用性の高い単一システムを検証するモデル検査ツールを用いて検証したい性質の確率的モデル検査を行い出力される確率．
② 対象となる可変性を持つシステムの FTS から導出され得る全製品を①と同様のモデル検査ツールを用いて個々にモデル検査を行い，真と判断された製品の割合．

評価に用いた検証項目は，モデル検査における典型的な以下の三つである．

- デッドロック：どの状態においても遷移可能な状態がある．
- 到達可能性：初期状態からいつか任意の状態に到達可能である．
- 遷移可能性：いつか任意の遷移が起こる．

本研究では，状態遷移図に確率を付与することのできる一般的なモデル検査ツールとして PRISM を用いる．

4.2 実験対象

本研究では，VendingMachine システムを対象に実験を行う．VendingMachine システムの FTS を図 5 に示す．

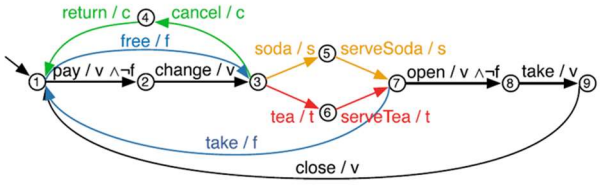

図 5　VendingMachine システムの FTS[4]

FTS のラベルのフィーチャ式に使われるフィーチャ数は 5，状態の数は 9，遷移の数は 13，製品数は 12 である．き本形の拡張 FTS と改善した拡張 FTS の検証を行う．

4.3 フィーチャの出現確率

フィーチャ s，t は同じ親フィーチャに対してどちらか 1 つ以上選ばれる関係である．そのため，手順(1)で求めた FTS のラベルのフィーチャ式に使われる 5 つのフィーチャの出現確率と，(s, t) の組み合わせを考慮した出現確率を表 1 に示す．

表 1　拡張 FTS に使われる確率(左：改善前，右：改善後)

フィーチャ	出現確率	フィーチャ		出現確率
v	1	v		1
f	0.5	f		0.5
c	0.5	c		0.5
s	0.667	(s, t)	(0, 0)	0
t	0.667		(1, 0)	0.333
			(0, 1)	0.333
			(1, 1)	0.333

4.4 拡張 FTS

手順(2)で作成した拡張 FTS1 の拡張部分の図式表現を図 6 に示す．また，改善した拡張 FTS2 の拡張部分の図式表現を図 7 に示す．

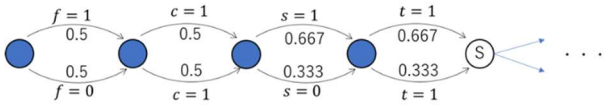

図 6　拡張 FTS1 の拡張部分

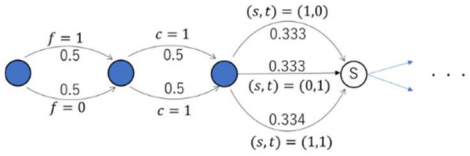

図 7　拡張 FTS2 の拡張部分

4.5 検証結果

実験対象のシステムに対して，提案手法によって作成した拡張 FTS1 と拡張 FTS2 を確率的モデル検査によって検証を行い，検証したい性質が満たされる確率を求めた．また，全 12 製品に対して個別にモデル検査を行い，検証したい性質の真偽を求め真となる割合を計算した．

検証項目は 4.1 で示した「デッドロック」,「到達可能性」,「遷移可能性」の 3 項目である．

4.5.1 拡張 FTS の確率的モデル検査の検証結果

提案手法である拡張FTSの確率的モデル検査の検証結果について説明する．

拡張FTS1と拡張FTS2のデッドロックの検証結果を表2に示す．拡張 FTS1 でどの状態でも遷移可能である確率は 0.945 となった．また拡張 FTS2 では 1 となった．

表 2 拡張 FTS のデッドロックの検証結果

	拡張 FTS1	拡張 FTS2
Deadlock	0.945	1

拡張 FTS1 と拡張 FTS2 の各状態への到達可能性の検証結果を表 3 に示す．

拡張 FTS1 と拡張 FTS2 の各遷移の遷移可能性の検証結果を表 4 に示す．

表 3　拡張 FTS の到達可能性の検証結果

State	拡張 FTS1	拡張 FTS2
1	1	1
2	0.5	0.5
3	1	1
4	0.5	0.5
5	0.667	0.667
6	0.667	0.667
7	0.889111	1
8	0.444556	0.5
9	0.444556	0.5

表 4　拡張 FTS の遷移可能性の検証結果

Trans	拡張 FTS1	拡張 FTS2
1	0.5	0.5
2	0.5	0.5
3	0.5	0.5
4	0.5	0.5
5	0.5	0.5
6	0.667	0.667
7	0.667	0.667
8	0.667	0.667
9	0.667	0.667
10	0.444556	0.5
11	0.444556	0.5
12	0.444556	0.5
13	0.444556	0.5

4.5.2 全 12 製品のモデル検査の検証結果

可変性を持つシステムの全 12 製品を個々にモデル検査の検証結果と性質が満たされる割合について説明する．

全 12 製品のデッドロックの検証結果を表 5 に示す．全 12 製品のうち全てにおいてどの状態でも遷移可能であることが満たされたため，性質を満たす製品の割合は 1 となった．

全 12 製品の各状態への到達可能性の真偽と性質が満たされる割合を表 6 に示す．

全 12 製品の各遷移の遷移可能性の真偽と性質が満たされる割合を表 7 に示す．

4.6 比較と考察

デッドロックに関して，拡張 FTS1 の検証結果は実際の製品の割合より約 0.055 低かった．拡張 FTS2 の検証結果は実際の製品の割合と一致した．

到達可能性に関して，拡張 FTS1 の検証結果は，State1~6 は実際の製品の割合と一致していたが，State7 は約 0.111，State8, 9 は約 0.055 低かった．拡張 FTS2 の検証結果は実際の製品の割合と一致した．

遷移可能性に関して，拡張 FTS1 の検証結果は，Trans1~9 は実際の製品の割合と一致していたが，Trans10~13 は約 0.055 低かった．拡張 FTS2 の検証結果は実際の製品の割合と一致した．

表 5　全 12 製品のデッドロックの検証結果

	prod1	prod2	prod3	prod4	prod5	prod6	prod7	prod8	prod9	prod10	prod11	prod12	割合
Deadlock	true	true	true	true	true	true	true	true	true	true	true	true	1

表 6　全 12 製品の到達可能性の検証結果

State	prod1	prod2	prod3	prod4	prod5	prod6	prod7	prod8	prod9	prod10	prod11	prod12	割合
1	true	true	true	True	true	true	true	true	true	true	true	true	**1**
2	true	true	true	True	true	true	false	false	false	false	false	false	**0.5**
3	true	true	true	True	true	true	true	true	true	true	true	true	**1**
4	false	true	true	True	false	false	true	true	true	false	false	false	**0.5**
5	true	true	false	True	true	false	true	true	false	false	true	true	**0.667**
6	true	true	true	False	false	true	true	false	true	true	true	false	**0.667**
7	true	true	true	True	true	true	true	true	true	true	true	true	**1**
8	true	true	true	True	true	true	false	false	false	false	false	false	**0.5**
9	true	true	true	True	true	true	false	false	false	false	false	false	**0.5**

表 7　全 12 製品の遷移可能性の検証結果

Trans	prod1	prod2	prod3	prod4	prod5	prod6	prod7	prod8	prod9	prod10	prod11	prod12	割合
1	true	true	true	true	true	true	false	false	false	false	false	false	**0.5**
2	true	true	true	true	true	true	false	false	false	false	false	false	**0.5**
3	false	false	false	false	false	false	true	true	true	true	true	true	**0.5**
4	false	true	true	true	false	false	true	true	true	false	false	false	**0.5**
5	false	true	true	true	false	false	true	true	true	false	false	false	**0.5**
6	true	true	false	true	true	false	true	true	false	false	true	true	**0.667**
7	true	true	false	true	true	false	true	true	false	false	true	true	**0.667**
8	true	true	true	false	false	true	true	false	true	true	true	false	**0.667**
9	true	true	true	false	false	true	true	false	true	true	true	false	**0.667**
10	false	false	false	false	false	false	true	true	true	true	true	true	**0.5**
11	true	true	true	true	true	true	false	false	false	false	false	false	**0.5**
12	true	true	true	true	true	true	false	false	false	false	false	false	**0.5**
13	true	true	true	true	true	true	false	false	false	false	false	false	**0.5**

拡張 FTS1 について，全製品を検証している先行研究とは違い，本研究では確率的モデル検査を行っているため，検証結果が必ずしも実際の値と一致しないことは前述した通りである．今回検証結果が実際の製品の割合と一致しなかった理由は，先行研究ではフィーチャの組み合わせを考慮したフィーチャの選択を特別なモデル検査ツール内でのアルゴリズムも用いて行っているが，今回は一般的なモデル検査ツールを使用し FIP を用いて確率的にフィーチャを選択しているため，実際には存在しない製品のフィーチャの組み合わせが含まれているからだと考えられる．本研究では存在しない製品を排除するアルゴリズムに関しては議論しないが，提案手法により作成した拡張 FTS1 の検証結果は実際の値と近似しているといえ，提案手法の妥当性が確かめられた．

拡張 FTS2 について，確率的モデル検査を行った検証結果が，実際に性質が満たされる製品の割合と一致したことから，拡張 FTS2 は排他的なフィーチャの組み合わせを正確に表現できたといえる．しかし，排他的なフィーチャの組み合わせを正確に表現できた理由は，今回の実験対象はフィーチャ間の関係が明らかであったからであり，一般的

には排他的なフィーチャを厳密に把握することは難しいため，排他的なフィーチャを正確に表現することはできないと考えられる．

この評価実験で，一般的なモデル検査ツールで可変性モデルを一回で検証できることが確認できた．確率的モデル検査の検証では全製品数や性質を満たす製品数などの具体的な数字が出力されるわけではないが，全製品を検証しないことで計算量が抑えられると考えられる．

5. 結論

本研究では，ファミリベースのモデル検査を一般的な単一システムを検証するモデル検査ツールを用いて行う手法を提案した．FIP を導出する手法を拡張し，FTS にフィーチャの出現確率をマッピングすることで確率的モデル検査を可能にした．確率的モデル検査では，性質を満たす割合を求めることができ，評価実験によって適切に検証が行えることを確認した．以上より，提案手法によって可変性を持つシステムを汎用的なモデル検査ツール用いて一回の実行で検証することができるようになった．

6. 今後の課題

本研究で提案している拡張FTSはフィーチャの数に対して状態数を増やしているため，可変性が膨大に増えた場合にはスケーラビリティの問題が発生する．今回明らかに排他なフィーチャに対しての処理方法を提案したが，フィーチャの関係が複雑なシステムになると対応できないため検証に時間がかかるなどの問題が発生する可能性がある．確率的モデルに対してシミュレーションを行う検証方法を応用し，シミュレーションのサンプル数などを検討することで，検証にかかる時間や製品数を削減することが必要である．

参考文献

[1] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. “A Classification and Survey of Analysis Strategies for Software Product Lines. ” ACM Comput. Surv. Vol. 47, No. 6, pp. 1-45, 2014.

[2] Harsh Beohar, Mahsa Varshosaz, Mohammad Reza Mousavi, “Basic behavioral models for software product lines”, Science of Computer Programming, Vol 123, pp. 42-60, 2016.

[3] Classen, A., Cordy, M., Heymans, P. et al. “Model checking software product lines with SNIP”. Int J Softw Tools Technol Transfer, Vol 14, pp.589–612 (2012).

[4] A. Legay, et al. , "Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking" in IEEE Transactions on Software Engineering, Vol. 39, No. 08, pp. 1069-1089, 2013.

[5] K. Czarnecki, S. She and A. Wasowski, "Sample Spaces and Feature Models: There and Back Again, " 2008 12th International Software Product Line Conference, pp. 22-31, 2008.

[6]Ruben Heradio , “Supporting the Statistical Analysis of Variability Models”, 41st International Conference on Software Engineering (ICSE), pp. 843-853, 2019.

[7]楊宇昕, 「モデルカウンターを使った FM の特徴把握方法の提案」, 早稲田大学修士論文, 2020.

[8]Marta Kwiatkowska, Gethin Norman and David Parker, “Stochastic Model Checking”, LNCS 4486, pp.220-270, 2007.

[9]「prismmodelchecker」, http://www.prismmodelchecker.org/, 2021/8/18

[10]A. Classen, P. Heymans, P. Schobbens, A. Legay, J. Raskin, "Model checking lots of systems: efficient verification of temporal properties in software product lines," 2010 ACM/IEEE 32nd International Conference on Software Engineering, pp. 335-344, 2010.

[11]K. G. Larsen, B. Thomsen, "A modal process logic," Proceedings. Third Annual Symposium on Logic in Computer Science, pp. 203-210, 1988.

[12]Maurice H. ter Beek, Franco Mazzanti, “VMC: recent advances and challenges ahead,” In Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools - Volume 2 (SPLC '14), Association for Computing Machinery, pp.70–77. 2014.

[13]Xavier Devroey, Gilles Perrouin, Maxime Cordy, Pierre-Yves Schobbens, Axel Legay, Patrick Heymans, “Towards statistical prioritization for software product lines testing,” In Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS '14). Association for Computing Machinery, Article 10, pp.1–7, 2014.

[14]Dimovski A.S., Wąsowski A., “Variability-Specific Abstraction Refinement for Family-Based Model Checking,” Fundamental Approaches to Software Engineering. (FASE 2017), vol 10202, pp.406-423, 2017.

[15]Ter Beek, M. H., Damiani, F., Gnesi, S., Mazzanti, F., Paolini, L., “On the expressiveness of modal transition systems with variability constraints,” Science of Computer Programming, Vol.169, pp.1–17, 2019.

[16]野田 夏子, 岸 知二, 「プロダクトライン開発における検証」, コンピュータ ソフトウェア, Vol.30, pp.3_3-3_17, 2013.