様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　３ 年　 ６ 月 １８ 日現在

機関番号：３２６１９
研究種目：基盤研究(C)（一般）
研究期間：2018～2020
課題番号：１８Ｋ１１５５２

研究課題名（和文）IoT/CPS時代の安全で役立つサービス構築のためのモデル駆動開発手法の研究

研究課題名（英文）Model-driven Development Method for Building Safe and Useful Services in IoT / CPS Systems

研究代表者
松浦　佐江子（Matsuura, Saeko）

芝浦工業大学・システム理工学部・教授

研究者番号：１０３４８９０６

交付決定額（研究期間全体）：（直接経費）　　3,400,000円

研究成果の概要（和文）：IoT時代の到来を迎え、多くの人、組織、コンピュータ・システムがつながることにより、サービスの大規模・複雑化が加速し、その品質及び安全性の保証が危惧されている。多様な利用者の特性やハードウェア構成が最終的なサービスの質に大きな影響を及ぼす。本研究では、開発早期に利用者の満足度やビジネスへの効果を検証しながら、漸増的にサービスを構築することにより、高品質なサービスを効率よく実現するモデル駆動開発手法をベースに、複数システム連携のインタラクションをモデル化し、利用者のゴールの満足度や、システムの安全性を検証することにより、安全で役立つサービスのモデル駆動開発手法を開発した。

研究成果の学術的意義や社会的意義
コンピュータを活用したサービスの範囲が、IoT/CPSにより拡大し、社会、人間、コンピュータからなるサービスアーキテクチャがサービスの質にもたらす影響が大きくなっている。安全性の保証や利用者の特性に応じた満足度を考慮したサービス構築のための要求定義・検証方法を確立し、適切に定義された要求仕様からプログラムを自動生成できるようにすることと適切な要求仕様を定義できる人材を育成することが、IoT時代の高品質なソフトウェアを効率よく開発するための重要な課題である。本研究の成果は高品質なソフトウェアを開発する有効な手段であるモデル駆動開発を多くの開発者が適切な要求定義・検証方法として学習する手段となる。

研究成果の概要（英文）：With the advent of the IoT era, the connection of many people, organizations, and computer systems is accelerating the scale and complexity of services, and there are concerns about guaranteeing their quality and safety. The characteristics of various users and the hardware configuration have a great influence on the quality of the final service. This research is based on a model-driven development method that efficiently realizes high-quality services by gradually constructing services while verifying user satisfaction and business effects at an early stage of system development. By modeling the interaction of system cooperation and verifying the satisfaction of the user's goal and the safety of the system, we have developed a model-driven development method for safe and useful services.

研究分野：ソフトウェア工学

キーワード：モデル駆動開発　検証　UML　状態遷移モデル　モデル検査

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

1.　研究開始当初の背景

ソフトウェア開発において、サービスは「コンピュータを使用して社会における様々な問題を解決する手段の提供」であり、ソフトウェアは「サービスを実現するために、コンピュータを利用したシステムがどのように動作すればよいかの指示を記述したコンピュータへの指示書」である。IoT（Internet of Things）時代の到来を迎え、多くの人、組織、コンピュータ・システムがつながることにより、サービスの大規模・複雑化が加速し、その品質及び安全性の保証が危惧されている。多様な利用者の特性やハードウェア構成が最終的なサービスの価値に大きな影響を及ぼすことから、開発早期に利用者の満足度やビジネスへの効果を検証することが重要になっている。こうした問題を解決するためには、開発早期の要求分析段階において、最終的なサービスの価値を検証することが重要である。ソフトウェア工学ではコンピュータを活用したサービスを構築するための、モデリング技術、検証技術、開発プロセス等のソフトウェア開発技術の研究の実績があり、モデル駆動開発は要求仕様から実装までのトレーサビリティを保証する有効な手段である。コンピュータを活用したサービスの範囲が、IoT や CPS(Cyber Physical System）により拡大し、社会、人間、コンピュータからなるサービスアーキテクチャがサービスの質にもたらす影響が大きくなっている。安全性分析手法の STAMP/STPA の着目点である安全性を保証するサービス構築や、利用者の特性に応じた満足度を考慮したサービス構築のための要求定義・検証方法を確立し、適切に定義された要求仕様からプログラムを自動生成できるようにすることと、適切な要求仕様を定義できる人材を育成することが、IoT 時代の高品質なソフトウェアを効率よく開発するための重要な課題である。

2.　研究の目的

本研究では、サービスの機能要求を標準モデリング言語 UML（Unified Modeling Language）の振舞い・構造モデルで捉える。これをユースケース部品と呼ぶ。複数システムの連携によるサービスはユースケース部品を組み合わせたインタラクションとしてモデル化する。これをワークフローと呼ぶ。サービスの機能要求に対して、サービスの目標、アーキテクチャ、利用者の特性、STAMP/STPA の安全性といった非機能要求を状態モデル等で捉え、その充足性をワークフローに対して検証することを通じて、サブシステムの役割を明確にし、漸増的にユースケース部品を開発する。サービス工学では、サービスの構築のための「観測」が重要な手段であり、観測結果が、ユースケース部品とそのインタラクションを適切にモデル化する重要な要因となる。しかし、コンピュータを用いたサービスにおいては、ユースケースのモデル化の観点から必要な観測ができていないことがあり、ユースケース部品を定義する際の妨げとなる。人や社会の関わりを研究するサービス工学の方法に、コンピュータの制約を満たす高品質なサービスの実現性を保証する UML モデル駆動開発手法を融合することで、安全で役立つサービスを開発する方法を確立することを目的とする。

3.　研究の方法

われわれは、UML 要求分析モデルから、形式手法の 1 つであるモデル検査用モデルへの自動変換により、モデル検査ツールを用いて、要求分析モデルの到達可能性や安全性を検証する方法と、検証すべき性質としての非機能要求のモデル化方法を開発した。この方法を拡張し、複数の連携するシステムの機能要求をシステムや利用者間のインタラクションのモデルにより定義する。非機能要求を定義したモデルにより、利用者の満足度や、サービスの内容の変化を分析する。複雑なインタラクションはレビューレベルで検証を行うことが困難であるため、モデル検査用のモデルに変換して、網羅的な検証を行う。IoT/CPS 時代における複数のシステムの連携したサービスの場合、サブシステムの基本となるユースケースに基づき、まずは、サービスの基本形をワークフローで定義する。その定義に対して、安全性の観点や、利用者の満足度の観点から、不足あるいは過剰なサービスを検討する。使用するハードウェアの種類、性能、利用者の作業プロセスや、特性に依存して、基本形自体も変化することがあるため、この段階において、検証結果に基づき、ステークホルダーの合意を得る必要がある。この際、利用者の作業プロセスの観察の観点をモデル要素より抽出し、観察の適切さの向上を図る。本研究では、課題 1「ワークフローに対する非機能要求の検証を踏まえてユースケース部品を開発する方法」、課題 2「ワークフローのモデル検査による検証方法」、課題 3「ワークフローの検証およびユースケース部品の分析観点を用いて現場における要求獲得を効率的に行う方法」について取り組んだ。

4.　研究成果

近年、人々の生活や仕事では IT を利用したサービスや製品の利用が必要不可欠である。これらのサービスや製品を提供するシステムは、IoT に見られるように複数のシステムやハードウェアの連携によって開発されることが多い。このようなシステムを開発する際、要求分析ではシステムの連携状況やハードウェア構成、動作環境を考慮し、与えられた初期要求のサービス手順を明らかにすることが望ましい。われわれは、このサービス手順をシステムの境界となるユーザやサブシステム間の情報のやり取りに着目したユースケース分析に基づき、これらのユースケースの連携が初期要求のゴールを満たすサービス手順をワークフローとしてモデル化する。ワークフローによって、初期要求のゴールを満たすことを確認した後、各サブシステムのユースケースを定義し、要求仕様としての要求分析モデルを確定する。本研究では、半形式的な UML を用いてワークフロー、データモデル、ユースケースを定義し、これらをまとめて要求分析モデルと呼ぶ。初期段階の要求分析モデルの品質が全体の開発に影響を与えることから、要求分析モデルは自然言語による初期要求を全て満たす仕様書として十分妥当である必要がある。課題 1 では、モ

デル駆動開発により、要求仕様から実装までのトレーサビリティを保証できるモデルの構築方法を定義した。課題 2 では、この複数システムの連携モデルを基に、並列動作の検証が可能なモデル検査ツール UPPAAL のモデルに変換して、デッドロックなしに動作可能かを検証する方法を提案した。この検証により、定義したワークフローが、複数システムの妥当な連携を表していることは保証できるが、要求の過不足に関しては、確認が不十分である。そこで、課題 3 では各連携システムの状態遷移という観点から要求分析モデルのクロスチェックを行い、振る舞いの過不足を評価する方法を提案した。

① 課題 1

ユースケースとはシステムが「＊＊できる」という機能の単位であり、ユースケース分析により、システムの機能的要求を明らかにすることができる。本研究では UML を用いたモデル駆動要求分析手法として、ユースケースモデルをシステムの要件を定義する基本要素として捉え、既存の自然言語によるユースケース記述の問題点を解決する要求分析モデルを定義した。図 1 はその概要を示している。ユースケース図の各ユースケースのユースケース記述はアクティビティ図で定義する。アクティビティ図は開始ノードから終了ノードまでのアクションノードを連ねたフローにより基本フローを定義する。基本フローに対する代替・例外フローはデシジョン・マージノードを用いて定義し、ガード条件として分岐の発生理由を明記する。事前条件、事後条件はそれぞれ開始・終了ノードにコメントとして定義する。このように、役割毎に要素に分けて定義することで可読性や検証可能性が向上する。アクションの対象となるデータは、クラス図で定義されたクラスを用いて、アクティビティ図内にオブジェクトノードとして明記する。事前条件等の条件もクラスとその属性に基づき定義する。クラス図はシステムのデータモデルを定義するものであることから、システムの持つデータと結びつけて振舞いを分析することができる。

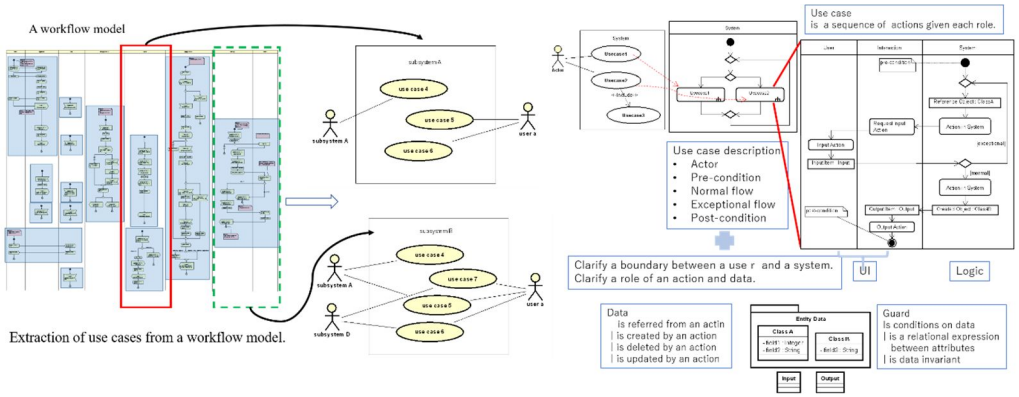

図 1　UML 要求分析モデル

さらに、アクターとシステムのやり取りは、事前条件の下、ユーザの入力に対して、事後条件満たす結果を導出する必要がある。アクティビティ図をアクター、システム、その間のインタラクションの 3 つのパーティションに分け、アクションやオブジェクトノードの主体が明確になるようにした。アクターパーティションはアクターのシステムに対する入力行為やそのデータ、インタラクションパーティションはシステムがアクターへ提供する問い合わせや出力の行為とそのデータを、システムはユースケースを実現するために、入力から出力を生成するためにシステムが行うべき行為と関連するデータを定義する。境界部分を分離することで、アクションの役割を限定できるため、統一性のある単語で振舞いを記述することができる。また、インタフェースとロジックを分離して分析することができるという利点があり、要求分析モデルから顧客の妥当性確認のためのプロトタイプを生成することができる。ここで、ユースケース間の関係は、サブアクティビティを 1 つのユースケースとしてアクティビティ図で定義することで利用シナリオが定義できる。このように、ユースケースは機能要求を定義する基本的な部品となるが、ハードウェアアーキテクチャ、ユーザ特性、安全性の要求といった非機能要求は、ユースケースの事前条件、事後条件、その組み合わせに大きな影響を与えることが多い。われわれは、これまでに、漸増的に要求仕様を分析・検証するプロセスを提案した。しかし、複数の独立したシステムの連携における並列動作を UML のモデルだけから検証することは困難である。

② 課題 2

UML において、アクターは利用者の役割、他のシステム、ハードウェアを表す。複数のアクターがシステムに関わる場合、そのシステムが提供するサービスは、異なる役割をもつ複数の利用者に対して、複数の外部システムやハードウェア機器によって提供される。これらのアクター間におけるインタラクションにより、システムのゴールを満たすように、すべてのシナリオを定義しなければならない。そこで、こうしたインタラクションをつぎのようにアクティビティ図で定義し、これをワークフローと呼ぶ。ワークフローは、システムを構成するすべてのアクターの部分集合間のインタラクションからなる、システムのゴールを満たすための 1 つ以上の利用シナリオの定義である。ワークフローの定義で明らかにすべきことは、つぎの 2 点である。

1)　すべてのサブシステムと利用者がシステムのゴールを満たすための要求された振舞いを各パーティション内のアクション系列として定義する。

2)　これらの振舞いが成り立つように各サブシステム間のインタラクションにおけるデー

タの授受を明らかにする。
　データの授受とは各サブシステムがその目的を達成するために必要な入力を受け取ること、他のサブシステムへの入力となるデータを出力することである。1)はユースケース記述と同様に、アクション系列で定義するが、2）はインタラクションパーティションと同様にサブシステム間の境界でのデータの授受であることから、宛先と内容を定めた通信と捉えて、サブシステム間のシグナル送受信ノードにより記述する。通信の宛先と内容を限定するために、宛先を UML のステレオタイプを用いて定義し、内容はノードのラベルに記述する。ノードには宛先（例えば<<中継所>>）と「荷物を送る」「荷物を受け取る」のように送受信内容を明記する。サブシステム固有のアクションは通常のアクションノードで定義し、詳細は、そのサブシステムのユースケース記述として、図１で示したようにアクティビティ図で定義する。これにより、シグナル送受信で授受されるデータを用いて各パーティションのアクション系列を詳細化することで、連携システムの振舞いを定義することができる。ワークフローが、システムのゴールを満たしていることを確認できたら、ワークフローの各パーティション毎にユースケース図を定義する。図１に示すように、各パーティションがサブシステムに対応し、枠内のフローが、各ユースケースに相当する。ユースケースの開始ノードには事前条件、終了ノードには事後条件を明記する。これにより、ユースケースの組み合わせとワークフローを対応付けることができる。
　UML のモデル要素のラベルは自然言語で記述する。これは、多くの一般的な開発者がモデルを理解しやすいという意味で利点である。しかし、モデリングにおいて、これは諸刃の剣であり、モデルが理解しやすい反面、曖昧性や不整合を生む要因となる。さらに、アクティビティ図は、振舞いを表現する汎用的なモデル図であり、フローチャートのようにアルゴリズムの定義も行うことができる。UML にはワークフローを定義するための特定の記法はない。さらに、１つのワークフローに書かれた複数のシナリオがどのような順序で動作するのかを、正確には記述できないという問題がある。そこで、上述のとおりサブシステム間のデータの授受のアクションの記述方法を、シグナル送受信ノードとステレオタイプにより、レビュー時に読みやすいように規定する。しかし、このように規定した並列動作において不整合がないかを、目視で確認することは困難である。また、サブシステム固有のアクションが、他のサブシステムとのデータの授受を行う場合には、その固有のユースケースを定義したアクティビティ図の振舞いを展開した状態で確認を行わなければならないため、さらに難しくなる。
　サブシステム間の連携では、正しい相手と正しいデータの授受を適切なタイミングで行うことで、システム全体が停止することなく、すべてのサブシステムが、ゴールを満たす振る舞いを完結できることが保証できればよい。UML でワークフローを定義することの利点を活かして、この保証を実現するために、並列動作を網羅的に検査可能なモデル検査ツール UPPAAL のモデルへ変換して、UML モデルの振舞いの検証を行う。UPPAAL モデルはパラメータをもつテンプレートから生成される複数のプロセスから構成される。各プロセスはシステムの振舞いのインスタンスであり、チャネルを通じて同期をとり、並列動作を行う。UML のワークフローのパーティション内の連結したフローを１つのテンプレートとし、シグナル送受信で定義したサブシステム間のデータ授受をチャネルによる同期に対応付けることで、UML モデルを並列動作のシミュレート可能な UPPAAL モデルに変換する。
　本研究では、すべてのサブシステム間の連携が正しく行われているかを検証するため、UML で定めた振舞いモデルにおけるデータ授受の振舞いのサブシステム間における関係に着目する。UML のワークフローの各パーティションには各サブシステムの期待される振舞いを複数のアクション系列で定義した。その振舞いを実現するために必要なサブシステム間のデータの授受がシグナル送受信を通じて行われているはずである。そこで、これらの複数のアクション系列が途中で停止することなく、すべての作業を完結できることで、この振舞いモデルの妥当性が保証できる。そこで、各パーティション内のアクション系列をテンプレートに変換する。各ノードはロケーションに、シグナル送受信ノードは、ノードのラベルにより識別可能なチャネルに変換する。シグナル送受信ノードに記述されたステレオタイプとラベルの目的語が同じ場合に、１つのチャネルに対応付ける。ここでのワークフローの振舞いモデルの妥当性とは、変換した UPPAAL モデルにおいて、デッドロックが生じないということである。そこで、UPPAAL における検証式は A[](not deadlock)となる。これにより、すべてのパスに対して、実行が停止することがないかを確認することができる。
　前述の変換規則に従い、UML モデルを UPPAAL モデルに自動変換するツールを Java と UML モデリングツール astah の API を使用して実装した。本ツールにより、UML で定義したワークフローは実行が停止することなくすべての作業が行えることを UPPAAL モデルの検査器を使用して確認することができる。UPPAAL で得られた反例を基に、元の UML モデルを修正することができる。
　本学の学部 3 年生 PBL 実習の課題である「自律走行車両型ロボットによる荷物の自動搬送システム」に本手法を適用した。このシステムは 2 台の自律走行車両型ロボットが、荷物の収集と配送の役割を担い、宅配受付所から収集した荷物を、中継所を経由して、指定された受取人宅まで指定された搬送路上を走行して、搬送するもので、複数システムの連携によりサービスが成り立っている。サブシステム間の連携により、各サブシステムの実現に必要なデータの授受が行われるかを確認することができた。
③　課題 3

課題 2 の検証方法により、定義したワークフローが、複数システムの妥当な連携を表していることは保証できるが、要求の過不足に関しては、確認が不十分である。アクティビティ図を用いたワークフローやユースケース記述の定義は、サービス手順を表すことに適している。しかし、サービス手順としての観点だけでは、定義したワークフローにおいてサブシステムが行うべき振る舞いを過不足なく抽出できているかは確認できていない。ワークフローはアクティビティ図を用いてサブシステムの連携やその振る舞いを各境界での情報の授受アクションに着目して、手順の観点からアクションのフローにより規定するものである。要求されている機能の手順は、制御構造に従って理解しやすいが、アクションによりシステムの状態がどのように変化したかは明示されないため、システムが取るべき状態の全てを把握することは難しい。一方、状態遷移モデルは、外部または内部のイベントによりシステムの状態が遷移することで振る舞いを定義する。状態名を明記することでシステムの要件が網羅的に検討されているかはわかりやすいが、システムの機能の実行手順や他のサブシステムとの関係を確認することが難しい。そこで、サブシステム間の連携を含んだワークフローの定義にはアクティビティ図が適している。逆に、サブシステムが満たすべき要件をシステムの取るべき状態という観点から整理する状態遷移モデルは、そのテストケースとなりうる。そこで、本研究では、初期要求に基づき分析者が定義した要求分析モデルから状態遷移モデルを自動抽出し、同じ初期要求から分析者とは異なる評価者が作成した状態遷移モデルと自動的に比較することで不整合個所を決定し、分析者にフィードバックする。これにより、分析者は要求分析モデルの妥当性を評価し、改善する。前者の状態遷移モデルを「抽出モデル」後者を「評価用モデル」と呼ぶ。

状態を区別する観点として、初めに、ワークフローの記述構造に着目する。デシジョン・マージノードは遷移の制御ノードであり分岐条件によって遷移が異なるため、その前後で状態が分かれていると識別できる。次の観点として、ワークフローのアクションに着目する。アクションは明示的に状態を表さないが、アクションによって状態が何らかの変化をすると考えられる。このようなアクションは他のシステムからデータを受け取るアクション、時間経過を示すアクション、システムのもつ属性を変化させるアクション、サブシステム独自のアクションから構成される。他のシステムからデータを受け取るアクションはシグナル受信ノードとして記述されており、時間経過を示すアクションはタイマーとして記述されている。これらの前後で状態を区別し、区別した状態間の遷移のトリガーとして抽出する。システムのもつ属性を変化させるアクションも状態を前後で区別し、区別した先の状態の入場動作として抽出する。サブシステム独自のアクションは、次の状態への遷移のアクションとして抽出する。シグナル送信アクションは、他のシステムとの連携を示すアクションであるが、自身から行うアクションであるため、区別された状態の入場動作として抽出する。また、ワークフローに記述されている事前条件・事後条件は、クラスに定義したデータによる条件であり、システムの状態を明確に表している。これらはノートとしてイニシャルノードとファイナルノードに紐づいているため、イニシャルノードとファイナルノードを初期状態と終了状態として抽出する。抽出モデルは、定義した対応規則によりアクションによる状態の変化を抽出しているため、状態の階層は持たない。また、ワークフローは手順を示しているので状態の移り変わりは特定できるが、状態そのものがどういう状態なのかは特定できない。抽出モデルが、評価用モデルで示されるシステムの取るべき状態を網羅しているかを確認するために、評価用モデルでは、対象システムの特徴に合わせて適切にシステムの取るべき状態を記述する必要がある。ワークフローにはサブシステム間の連携シナリオが定義されているので、サブシステムの取るべき状態は、初めに連携する相手毎の状態を決定し、その状態内でどのような仕事のシナリオを実行するべきかという観点から記述することとする。これにより評価用モデルの状態は、着目する観点毎の階層に分かれて定義される可能性がある。また、評価用モデルに記述する語句が評価者によって自由に記述されると状態遷移モデルの内容比較が困難になるため、ワークフローのアクション一覧を提供し、語句の統一を図る。これにより評価者の記述したい内容がアクション一覧に含まれているならば、そのアクションを用いて記述することで、内容比較が容易になる。評価用モデルは、「連携する相手毎に、どのような仕事のシナリオを実行するべきか」という観点からモデル化するので、状態の階層構造を持つことが多い。一方、抽出モデルでは、アクションや分岐構造毎に状態が細分化されるので、各状態を評価用モデルの状態に区分することで、抽出モデルの状態を評価用モデルの粒度に合わせて区分した時の状態数・区分した状態間の遷移の向きと本数・状態の動作、状態遷移のイベント・ガード・アクションの記述内容により階層毎に比較を行い、不整合の原因を特定する。状態の区分は、抽出モデルの状態を状態に記述されている動作・状態遷移に記述されているイベント・ガード・アクション・区分できていない状態の遷移前・遷移後の状態等により行う。

サブシステムのワークフローからの抽出モデルの生成・抽出モデルと評価用モデルの比較の機能を持つツールを UML モデリングツール astah のプラグインとして実装した。課題 2 と同じ事例に対して、適用実験を行い、過分析・不要な状態/フロー/処理の存在、必要な状態/フローの欠損、サービス手順の誤り等を発見することができた。

５．主な発表論文等

〔雑誌論文〕　計4件（うち査読付論文　4件／うち国際共著　0件／うちオープンアクセス　4件）

１．著者名 Hikaru Morita, Saeko Matsuura	４．巻 11
２．論文標題 Validation Method To Improve Behavioral Flows On UML Requirements Analysis Model By Cross-Checking With State Transition Model	５．発行年 2021年
３．雑誌名 10th International Conference on Software Engineering and Applications (SEAS 2021)	６．最初と最後の頁 1-15
掲載論文のDOI（デジタルオブジェクト識別子） 10.5121/csit.2021.110201	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

１．著者名 Saeko Matsuura, Sae Ikeda and Kasumi Yokota	４．巻 1
２．論文標題 Automatic Verification of Behavior of UML Requirements Specifications using Model Checking	５．発行年 2020年
３．雑誌名 Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development	６．最初と最後の頁 158-166
掲載論文のDOI（デジタルオブジェクト識別子） 10.5220/0009339001580166	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

１．著者名 Kauto Yoshino and Saeko Matsuura	４．巻 1
２．論文標題 Requirements Traceability Management Support Tool for UML Models	５．発行年 2020年
３．雑誌名 Proceedings of the 2020 9th International Conference on Software and Computer Applications	６．最初と最後の頁 163-166
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3384544.3384586	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

１．著者名 Saeko Matsuura, Shinpei Ogata and Yoshitaka Aoki	４．巻 1
２．論文標題 Goal-Satisfaction Verification to Combination of Use Case Component,	５．発行年 2018年
３．雑誌名 Conference: 13th International Conference on Evaluation of Novel Approaches to Software Engineering	６．最初と最後の頁 343-350
掲載論文のDOI（デジタルオブジェクト識別子） 10.5220/0006785003430350	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

〔学会発表〕　計19件（うち招待講演　0件／うち国際学会　2件）

1．発表者名
Hikaru Morita, Saeko Matsuura

2．発表標題
Validation Support Tool to Cross-check the Behavioral Flows on a Requirements Analysis Model using the State Transition Model

3．学会等名
7th International Conference on Computational Science and Computational Intelligence（CSCI2020）（国際学会）

4．発表年
2020年

1．発表者名
阿部 優樹　石倉 大輔　金子 恵大　齊藤 健太　長島 陸　鍋山 達也　南 孝輝　松浦 佐江子

2．発表標題
品質モデルを用いたソフトウェアモデリング教育における要求仕様改善実験

3．学会等名
ソフトウェアエンジニアリングシンポジウム2020

4．発表年
2020年

1．発表者名
森田光，松浦佐江子

2．発表標題
要求分析モデルと状態遷移モデルのクロスチェックによる要求検証ツールの開発

3．学会等名
ソフトウェアエンジニアリングシンポジウム2020.

4．発表年
2020年

1．発表者名
森田光，松浦佐江子

2．発表標題
要求分析モデルと状態遷移モデルのクロスチェックによる要求検証ツールの開発

3．学会等名
第27回 ソフトウェア工学の基礎ワークショップ FOSE2020，ソフトウェア工学の基礎XX VII，pp. 143-144

4．発表年
2020年

１．発表者名
吉野魁人，松浦佐江子

２．発表標題
要求定義後トレーサビリティの確保に向けたUMLモデルによるユースケース記述変換手法

３．学会等名
電子情報通信学会知能ソフトウェア工学研究会

４．発表年
2020年

１．発表者名
森田 光，松浦 佐江子

２．発表標題
要求分析モデルからの状態遷移抽出による振る舞いフローの妥当性確認支援

３．学会等名
第82回情報処理学会全国大会（学生奨励賞）

４．発表年
2019年

１．発表者名
森田 光，松浦 佐江子

２．発表標題
要求分析モデルからの状態遷移抽出による振る舞いフローの妥当性確認支援

３．学会等名
日本ソフトウェア科学会第26回ソフトウェア工学の基礎ワークショップ

４．発表年
2019年

１．発表者名
森田 光，松浦 佐江子

２．発表標題
要求分析モデルからの状態遷移抽出による振る舞いフローの妥当性確認支援

３．学会等名
情報処理学会ソフトウェアエンジニアリングシンポジウム2019

４．発表年
2019年

１．発表者名
上田達也,大久保美涼,鈴木航,鈴木亘,山下京之介,秋山孝平,松浦佐江子

２．発表標題
品質モデルを用いたソフトウェアモデリング教育における要求仕様改善実験

３．学会等名
情報処理学会ソフトウェアエンジニアリングシンポジウム2019

４．発表年
2019年

１．発表者名
吉野 魁人，松浦 佐江子

２．発表標題
要求定義後トレーサビリティ確保に向けたUMLモデルによるユースケース記述変換手法

３．学会等名
情報処理学会ソフトウェアエンジニアリングシンポジウム2019

４．発表年
2019年

１．発表者名
Saeko Matsuura

２．発表標題
User Characteristics Validation for User Scenario based on Use-Case Models in Requirements Analysis,

３．学会等名
SOFTENG2019（国際学会）

４．発表年
2019年

１．発表者名
吉野魁人，松浦佐江子

２．発表標題
要求分析と基本設計間のトレーサビリティ確保のためのユースケース記述変換ツール

３．学会等名
第81回情報処理学会全国大会（学生奨励賞）

４．発表年
2019年

１．発表者名
松隈暖，松浦佐江子

２．発表標題
トレーサビリティ確保のためのXtendを用いたUMLモデルコンパイラ

３．学会等名
第81回情報処理学会全国大会

４．発表年
2019年

１．発表者名
滝本竜海，松浦佐江子

２．発表標題
ソースコードからのUMLモデル自動生成によるソフトウェア概略の理解支援

３．学会等名
第81回情報処理学会全国大会（学生奨励賞）

４．発表年
2019年

１．発表者名
森田光，松浦佐江子

２．発表標題
要求分析モデルからの状態遷移抽出による振る舞いフローの妥当性確認支援

３．学会等名
第81回情報処理学会全国大会

４．発表年
2019年

１．発表者名
吉野魁人，松浦佐江子

２．発表標題
要求分析と基本設計間のトレーサビリティ確保のためのユースケース記述変換ツール

３．学会等名
日本ソフトウェア科学会FOSE 2018

４．発表年
2018年

１．発表者名
吉野魁人，松浦佐江子

２．発表標題
要求分析と基本設計間のトレーサビリティ確保のためのユースケース変換ツール

３．学会等名
情報処理学会ソフトウェアエンジニアリングシンポジウム

４．発表年
2018年

１．発表者名
池田彩恵，松浦佐江子

２．発表標題
要求分析段階におけるモデル検査技術を用いた設計制約検証の自動化

３．学会等名
第80回情報処理学会全国大会（学生奨励賞）

４．発表年
2018年

１．発表者名
吉野魁人，松浦佐江子

２．発表標題
要求分析と設計間のトレーサビリティ確保のためのUML 変換ツール

３．学会等名
第80回情報処理学会全国大会

４．発表年
2018年

〔図書〕　計1件

１．著者名	４．発行年
渡辺晴美，今村誠，久住憲嗣，石田繁巳，大川猛，小倉信彦，汐月哲夫，菅谷みどり，松浦佐江子，松原豊，三輪昌史，元木誠	2020年
２．出版社 コロナ社	５．総ページ数 176
３．書名 つながる！基礎技術　IoT入門　- コンピュータ・ネットワーク・データの基礎から開発まで -	

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関