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　４ 年　 ６ 月 １８ 日現在

機関番号：１７５０１
研究種目：基盤研究(C)（一般）
研究期間：2017～2021
課題番号：１７Ｋ００１１５

研究課題名（和文）環境と密に繋がるソフトウェアの開発を支援する新たなプログラミングの基盤技術

研究課題名（英文）Development of programming languages and practices for software that intensively interacts with its surrounding environment

研究代表者
紙名　哲生（KAMINA, Tetsuo）

大分大学・理工学部・准教授

研究者番号：９０４３１８８２

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）：ソフトウェアとハードウェアの融合が進む現在，ソフトウェアは外界から取得される多様な情報に応じて適応的に動作することが求められる．そこには複雑なデータの流れとそれに応じた振る舞いの動的な変更があり，このことは既存技術によるソフトウェアの開発を難しくする．本研究では，この問題を，プログラミング言語による記述を変えることによって根源的に解決することを目指し，データの流れを宣言的に記述できるリアクティブプログラミング（RP）言語と，文脈に依存した動作をモジュールとして分離できる文脈指向プログラミング（COP）言語それぞれの機構を深化させた．また，両者を統合したシンプルな計算体系を実現した．

研究成果の学術的意義や社会的意義
本研究の応用分野としては，CPSやIoTをはじめ，環境とのやりとりを密に行う様々な分野があげられる．これらのソフトウェア開発の難しさを，プログラミング言語による記述を変えることによって解決することにより，複雑なコーディングにコストをかけなくてすむようになるため，ソフトウェアの生産性や安全性の向上が期待できるという点で社会的に意義がある．また，異なる起源を持つCOPとRPを，同じ目的のもとに統合させる点は，本研究の大きな特色である．この統合はこれまでなされておらず新規性があり，またそれを通じて両者の考え方をともに深化できるという点において，プログラミング言語分野の研究に貢献する．

研究成果の概要（英文）：Currently, the boundary between software and hardware is becoming more and more ambiguous, and a software system is required to behave adaptively according to information obtained by several sensor devices. Such a software system consists of complex data flows and adaptive behavior changes based on them. Those complexity and adaptive changes make software development difficult. This research aims to tackle this problem by proposing a new programming language. Specifically, we proposed a set of important language mechanisms for both reactive programming (RP, where data flows are declaratively specified) and context oriented programming (COP, where context-dependent behaviors are modularized). Furthermore, we realized a simple core calculus that integrates both basic features of COP and RP.

研究分野：プログラミング言語

キーワード：文脈指向プログラミング　リアクティブプログラミング

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

ソフトウェアとハードウェアの融合が進む現在，ソフトウェアは外界から取得される多様な情報に応じて適応的に動作することが求められる．例えば，実世界に浸透した組み込みシステムなどが構成する情報を，サイバー空間の強力な計算能力とシームレスに結びつけるサイバーフィジカルシステム（CPS）やモノのインターネット（IoT）は今後の社会を支える重要なインフラになると予想される．このようなインフラをいかに効率よく開発し，高品質で安全なシステムを世に送り出すかは，社会にとって重要な問題であり，産業界でも注目されている．そこには，内外からの様々な値に起源を発する複雑なデータの流れが存在し，そこを流れるデータに応じてモノを動かすソフトウェアの振る舞いも動的に変更される．このことは，既存技術によるソフトウェアの開発を難しくする．

こうした問題を解決する一つの方法は，陳腐化した現在主流のプログラミング言語に代わる新たなプログラミング言語や実践方法を探ることである．実際，ソフトウェアの振る舞いを動的に切り替えることや，データフローをプログラミングすることに関して，新たなプログラミングの実践方法が今熱心に研究されている．まず，動的にモジュールを切り替えることで環境が著しく変化していく状況に対応する文脈指向プログラミング（COP）が研究者の注目を集めている．COP は従来の固定的なモジュールを覆し，環境や状況の変化に応じてモジュールの振る舞いを動的に変更できる言語要素を実現した．また，近年注目されるリアクティブプログラミング（RP）は，データフローや外部からのイベント列を直接扱え，センサ等から随時更新されるデータに対して応答を決めていく CPS や IoT にもよく馴染むプログラミングの方法である．

これらの研究は未だ発展途上にあり，CPS や IoT などの今後の環境適応型ソフトウェアの開発を支援するには，より発展的なプログラミングの実践方法を追求する必要がある．例えば COP の研究は振る舞いの動的な変更の仕組みにばかりフォーカスしており，実測されるデータと振る舞い変更の関連については考えられておらず，実用的でない．また，RP で表現できるデータフローは固定的で，環境適応型の制御に適用するには限界がある．言い換えると，COP にはデータフローの観点が，RP には動的変更の観点が抜けている．これらは互いに関連するが，両者の特徴や利点を併せ持つようなプログラミング言語はこれまで実現されてきていない．

２．研究の目的

本研究では，RP のデータフローの考え方と COP の振る舞い変更のアイデアを統合した，環境適応型ソフトウェアの開発を支援する新たなプログラミングの実践方法を明らかにする．具体的には、外部環境の観測から得られるデータフローに基づいたモジュールの振る舞いの自動的な変更と，データフロー自体の動的な変更を直接記述できる言語をプログラミング言語として実現する．そのための基礎理論を構築する．それと同時に，両者の考え方をお互いに深化させるための，新たな言語要素の提案や，プログラミング実践のためのツールの提案を行う．

３．研究の方法

この目的を実現するためには，（１）COP が RP から受ける影響，（２）RP が COP から受ける影響，（３）両者の統合方法，（４）プログラミングの実践方法をそれぞれ解明する必要がある．そのため，本研究では以下の研究項目を実施する．

① RP の言語要素をはじめ，環境適応に必要なさらなる機能をもつ COP 言語を設計する．
② COP の言語要素をはじめ，環境適応に必要なさらなる機能をもつ RP 言語を設計する．
③ COP と RP の統合を説明できる新しいプログラミング言語体系を明らかにする．
④ これらのプログラミングの方法を円滑に実践するための，新しいプログラミング実践や支援の在り方を明らかにする．

本研究では，これらの研究項目を，できる限り共通の言語体系のもとで進めるため，いずれも，Java に基づく COP 言語と，同じく Java に基づく RP 言語を研究の前提にする．具体的には，それぞれ ServalCJ [1]と SignalJ [2]を用いる．理論体系の構築には，やはり Java の基本的な構文と意味論を与える Featherweight Java [3]を前提にし，それに COP と RP それぞれの要素を追加した体系を考える．④については，プログラミング支援が必要となる実際の例として，デバッグのためのツールを考える．

４．研究成果

（１）上記の①について，COP 言語の深化にかかわる一連の成果を得た．COP 言語では，環境や状況の変化に応じてモジュールが動的に切り替わるプログラムを記述することができる．この動的に切り替えられるモジュールを層，層を動的に切り替えることを層活性と呼ぶ．層活性を記述する方法には様々あるが，そのうちの一つに，ある層が活性化される条件を宣言的に記述するという方法がある．以下は，画面の表示方法としてランドスケープとポートレートの方法を実装したそれぞれの層 `Landscape` と `Portrait` の層活性を，COP 言語 ServalCJ を用いて宣言的に記述する方法で記述したものである．

```
contextgroup Orientation(Sensor sensor) {
  activate Landscape if(sensor.value > THRESHOLD);
  activate Portrait when !Landscape;
}
```

しかし，従来の ServalCJ（や他の COP 言語）においては，if や when が評価されるタイミングに問題がある．通常これらは，Lanscape か Portrait どちらかの振る舞いを呼び出す場面において，必要に応じて評価されるオンデマンド方式をとっている．しかしながら，実際には画面描画を切り替えるタイミングと層の振る舞いを呼び出すタイミングは一致しておらず，画面描画を切り替えるべきであるにも関わらず画面が切り替わらないという問題が生じる．

本研究ではこの問題を，RP 言語の SignalJ におけるシグナルを if や when の条件式に採用することにより解決した．SignalJ ではシグナルの値計算にプル（オンデマンド）方式を，シグナルの値計算の副作用の実行にプッシュ方式を採用しており，本研究では層活性をシグナルの値更新の副作用として実現した．例えば，上記の層活性の sensor.value が，SignalJ のシグナルの記法を採用して，以下のように宣言されているとする．

```
class Sensor { signal int value; … }
```

コンパイラは，if の条件式にシグナルを見つけると，その値変化の副作用として，if の条件式を評価し，それが真であれば層 Landscape を活性化させるような振る舞いを登録する．これにより，状況や環境が変わった際に即座に振る舞いを切り替えることが可能になり，上述した問題は起こらなくなる．

一方で，この拡張により引き起こされる問題もある．この層活性方式は，層の振る舞いに対して非同期的に，いつでも層活性が起こることを意味する．これは，層のインタフェースの拡張（つまり，層に参加するクラスに新しいメソッドを追加するなどの拡張を行うこと）を安全に行うことを難しくする．なぜなら，層が，その層内にしか存在しないメソッドを呼び出す直前に不活性化されると，そのメソッド呼び出しは失敗するからである．この問題を解決するため，メソッド探索に，それの呼び出しが記述された静的なスコープを用いる新たなメソッド探索の方法を実現した．その意味論を形式的に記述し，型安全性（つまり，メソッド探索が失敗しないこと）を証明した．

（2）上記の②について，RP 言語の深化にかかわる一連の成果を得た．RP 言語では，変数同士の依存関係（データフロー）を宣言的に記述できる．例えば，RP 言語 SignalJ で以下のようにセンサとモータの連携を記述したとする．

```
signal double sensorValue;
signal double powerDifference = f(sensorValue);
powerDifference.subscribe(x -> actuateMotor(x));
```

ここで，sensorValue の値が更新されると，powerDifference の値が再計算され，powerDifference に登録された副作用 x->actuateMotor(x) が実行される．SignalJ ではシグナルの値更新にはプル方式を，副作用の実行にはプッシュ方式を採用している．上の例では，sensorValue の値が更新されるとそのデータフローの下流にある副作用が全て起動される．その副作用の中でシグナルの値が使われると（上の例では，副作用の引数 x にはそのシグナル自身（つまり powerDifference の値）が代入される），その時点で最新の値が再計算されて，最新の powerDifference の値を用いてモータの制御が行われることになる．

言い換えると，SignalJ には複数の計算戦略（プルとプッシュ）と複数の機構（データフローとイベント（によって引き起こされる副作用））が，一つの言語機構シグナルの中に「調和して」存在している．この「調和」がプログラマの期待通りの振る舞いをするのかどうか検証することは重要である．本研究では，そのような検証を行うために SignalJ の意味論を形式化し，副作用と値更新の実行タイミングや計算の進行に関する様々な性質を証明した．

さらに，RP の言語機構を環境適応の場面で用いることを考えると，想定されるアプリケーションは IoT などの時系列データを多用するものになる．通常，RP 言語では観測された現在の値に応じた現在の振る舞いを宣言的に記述することはできても，過去の値変化の履歴を含めたそれらの時系列情報を扱うことはできない．

本研究では，RP における新たな言語機構として，シグナルの値変化を永続化する永続シグナルを提案した．永続シグナルを持つプログラミング言語の処理系は，実行時に時系列データベースを提供する．各永続シグナルは，その時系列データベースのテーブルに暗黙的に紐づけられ，値の依存関係はデータベースのビューに対応させられる．関係データベースに基づくオープンソースの時系列データベース TimescaleDB を用いて実際に処理系を実装し，マイクロベンチマーキングにより評価を行ってこの方式の実現可能性を示した．

（３）上記の③について，COP と RP の統合を説明できる新しいプログラミング言語体系 TinyCORP を提案した．COP と RP を統合する際に起こる問題としてあげられるのが，シグナルと層活性の循環である．この循環とは，次のようなものである．まず，上述の（１）のように，層活性の条件にシグナルが使われる．次に，そのシグナルの値は層活性に依存しており，層活性によって値が変わり得るものとする．すると，シグナルと層活性が相互に依存するという依存の循環が生じる．このような循環があると，シグナルの値と層活性がともに定まらない不定の状況に陥る可能性がある．しかしながら，実際問題としてこのような循環の関係は現実に存在するため，これを許しつつ，不定の状況に陥ることを防ぐ計算の仕組みを実現することが必要である．

TinyCORP では，通常の（メソッド呼び出しやシグナルの値を得るなどの）計算をスモールステップの簡約規則で与えると同時に，層活性の判断をそれとは別のビッグステップの計算で与えている．とくに，ビッグステップの計算中にスモールステップの簡約が起こらないように形式化した．これにより，層活性の判断中に層活性が変わらないことを保証し，層活性とシグナルの相互依存の循環を許しつつ，不定の状況に陥らない計算の仕組みを実現した．

（４）上記の④について，RP におけるデバッグの支援のため，時間変化する値の更新履歴を時系列に沿って表示・可視化する新たなデバッグのためのツールを提案した．RP のプログラムを理解するには，シグナルの依存関係を理解することが重要であり，それを可視化するデバッギングツールがこれまでに提案されている[4]．一方，RP の別の見方として，個々のシグナルを，常に値が流れ込んでくるストリームであるとみなすことがある．この見方では，ストリーム上の個々の値が別のストリームに伝搬していく様子を一覧できるような可視化が必要になる．本研究では，図１に示すような，ストリームを跨る値の伝播を時系列（横軸）に沿って一覧することにより，RP のプログラムのデバッグを支援するツールを実現した．

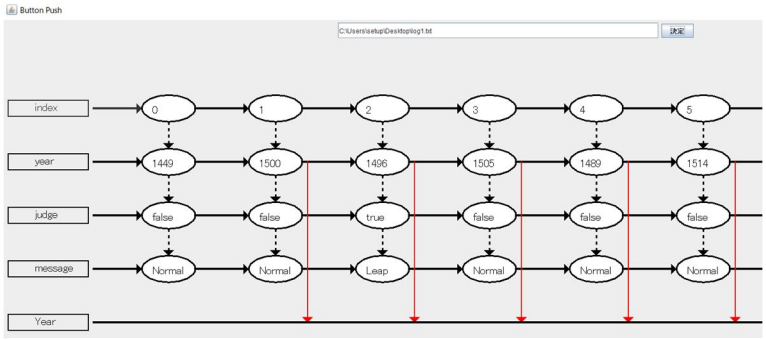

図１．シグナル間の値伝搬を時系列に示したダイアグラム

＜引用文献＞

[1] T. Kamina, T. Aotani, and H. Masuhara, Generalized Layer Activation Mechanism for Context-Oriented Programming, *LNCS Transactions on Modularity and Composition*, Vol.9800 of LNCS, pp.123-166, 2016.

[2] T. Kamina, Introducing Lightweight Reactive Values to Java, In *SPLASH'16 Companion: Conference on Systems, Programming, Languages, and Applications: Software for Humanity Proceedings*, pp.27-28, 2016.

[3] A. Igarashi, B.C. Pierce, and P. Wadler, Featherweight Java: a minimal core calculus for Java and GJ, *ACM Transactions on Programming Languages and Systems*, Vol.23, No.3, pp.396-450, 2001.

[4] G. Salvaneschi and M. Mezini. Debugging for reactive programming. In *ICSE'16*, pp.796-807, 2016.

５．主な発表論文等

〔雑誌論文〕　計7件（うち査読付論文　7件／うち国際共著　0件／うちオープンアクセス　3件）

１．著者名 Tetsuo Kamina, Tomoyuki Aotani, Hidehiko Masuhara	４．巻 1
２．論文標題 Managing Persistent Signals using Signal Classes	５．発行年 2020年
３．雑誌名 7th Workshop on Reactive and Event-based Languages & Systems (REBLS 2020)	６．最初と最後の頁 1,7
掲載論文のDOI（デジタルオブジェクト識別子） なし	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 Tetsuo Kamina, Tomoyuki Aotani	４．巻 1
２．論文標題 TinyCORP: A Calculus for Context-Oriented Reactive Programming	５．発行年 2019年
３．雑誌名 COP '19: Proceedings of the Workshop on Context-oriented Programming	６．最初と最後の頁 1,8
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3340671.3343356	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Tetsuo Kamina, Tomoyuki Aotani	４．巻 1
２．論文標題 An Approach for Persistent Time-Varying Values	５．発行年 2019年
３．雑誌名 Onward! 2019: Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software	６．最初と最後の頁 17,31
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3359591.3359730	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Takumi Hikosaka, Tetsuo Kamina, Katsuhisa Maruyama	４．巻 1
２．論文標題 Visualizing Reactive Execution History using Propagation Traces	５．発行年 2018年
３．雑誌名 5th Workshop on Reactive and Event-based Languages & Systems	６．最初と最後の頁 1-6
掲載論文のDOI（デジタルオブジェクト識別子） なし	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 Kamina Tetsuo、Aotani Tomoyuki、Masuhara Hidehiko、Igarashi Atsushi	4．巻 156
2．論文標題 Method safety mechanism for asynchronous layer deactivation	5．発行年 2018年
3．雑誌名 Science of Computer Programming	6．最初と最後の頁 104～120
掲載論文のDOI（デジタルオブジェクト識別子） 10.1016/j.scico.2018.01.006	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Kamina Tetsuo、Aotani Tomoyuki	4．巻 2
2．論文標題 Harmonizing Signals and Events with a Lightweight Extension to Java	5．発行年 2018年
3．雑誌名 The Art, Science, and Engineering of Programming	6．最初と最後の頁 5:1～5:29
掲載論文のDOI（デジタルオブジェクト識別子） 10.22152/programming-journal.org/2018/2/5	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 Kamina Tetsuo、Aotani Tomoyuki、Masuhara Hidehiko	4．巻 9
2．論文標題 Push-based reactive layer activation in context-oriented programming	5．発行年 2017年
3．雑誌名 Proceedings of the 9th International Workshop on Context-Oriented Programming (COP'17)	6．最初と最後の頁 17～21
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3117802.3117805	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

〔学会発表〕　計4件（うち招待講演　0件／うち国際学会　1件）

1．発表者名 紙名哲生，青谷知幸
2．発表標題 リアクティブプログラミングにおけるシグナルの永続化
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2019(SES2019)
4．発表年 2019年

１．発表者名
Tetsuo Kamina, Tomoyuki Aotani

２．発表標題
Harmonizing Signals and Events with a Lightweight Extension to Java

３．学会等名
<Programming> 2018（国際学会）

４．発表年
2018年

１．発表者名
Tetsuo Kamina, Tomoyuki Aotani

２．発表標題
Harmonizing Signals and Events with a Lightweight Extension to Java

３．学会等名
第21回プログラミングおよびプログラミング言語ワークショップ（PPL2019）

４．発表年
2019年

１．発表者名
紙名哲生，青谷知幸

２．発表標題
イベントとシグナルの統合：プログラミング言語のアプローチ

３．学会等名
第24回ソフトウェア工学の基礎ワークショップ(FOSE2017)

４．発表年
2017年

〔図書〕　計1件

１．著者名	４．発行年
Guido Salvaneschi, Wolfgang De Meuter, Patrick Th Eugster, Francisco Sant ' Anna, Lukasz S Ziarek, Tetsuo Kamina, Hidehiko Masuhara	2019年
２．出版社	５．総ページ数
Association for Computing Machinery	41
３．書名	
REBLS 2019: Proceedings of the 6th ACM SIGPLAN International Workshop on Reactive and Event-Based Languages and Systems	

〔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