様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成２４年　５月３１日現在

機関番号：21602
研究種目：若手研究（B）
研究期間：2009　～　2011
課題番号：21700038
研究課題名（和文）
セキュリティー強化型コンポーネントフレームワーク
研究課題名（英文）
Security Enhanced Component Framework
研究代表者
吉岡 廉太郎　（YOSHIOKA RENTARO）
会津大学・コンピュータ理工学部・准教授
研究者番号：00360008

研究成果の概要（和文）：ソフトウェアシステムの開発時において、各コンポーネントの入出力データの性質、想定される値の範囲、動作パターンなどを開発時に定義できるコンポーネント開発フレームワークを開発した。また、定義された情報に基づく実行時の監視と例外処理を可能とする実行環境の試験的実装を行った。

研究成果の概要（英文）：A component development framework that allows to specify a component's features of input/output values, their ranges, and behavior of has been developed. In addition, an experimental implementation of an execution environment to monitor the component at runtime and perform exception handling has been developed.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
2009 年度	1400000	420000	1820000
2010 年度	1100000	330000	1430000
2011 年度	900000	270000	1170000
年度			
年度			
総　計	3400000	1020000	4420000

研究分野：ソフトウェア工学、プログラミング技術、セキュリティー
科研費の分科・細目：情報科学・ソフトウェア
キーワード：自己説明型コンポーネント、ビジュアル言語

１．研究開始当初の背景

情報システムが社会の重要な位置を占めるようになり、その安全性と信頼性に対する要求は年々高まるばかりである。そんな中、内部に作り込まれたバグや外部からの攻撃によりシステムが誤動作（想定外の動作）することを防ぐ画一的な方法は現在のところなく、アプリケーション（開発プロジェクト）毎に対応しているのが現状である。本研究の目的は、システムを構成する各ソフトウェアコンポーネントに対して、想定される入出力データの性質や動作パターンを開発時に定義することにより、実行時にそれらを監視・制御することを可能とするコンポーネントフレームワークと実行環境を開発することである。これにより、外部または内部の要因によりシステムが誤動作した場合、安全に例外処理に移ることが可能になる。本研究の特徴は、開発用フレームワーク自体に、システムの安全性・信頼性を向上する機能を埋め込むことにあり、セキュリティーの専門知識を持たない開発者でも一定レベルのセキュリティーの確保されたシステムの開発を可能にすることである。また、そのような開発者

でも容易に入出力データと動作パターンを定義できるようする自己説明型言語を備えることももう一つの特徴である。

現在、システムの安全性を確保することを補助するツールには大きく分けて二つの種類がある。一つは、開発時に対策を作り込む種類で、もう一つは実行時に監視・制限する種類である。一つ目の、開発時に対策を作り込む一般的な方法は、いわゆるセキュアーなプログラミング言語(例えば、をJIF、Cyclone、Flow Camelなど）を用いることである。これらの言語は、厳密な型チェック、変数に対する階層的アクセス制限、詳細なメモリ管理、データの流れの管理などが可能で、プログラミング上のミスを抑制する狙いがある。これらは非常に細かい制御が可能である一方、設計の完全性に対する知識と高いスキルが開発者に要求され、生産性が悪いのが欠点である。他には、SQLインジェクションやstring-format-attackのような特定の攻撃を防ぐ機能をもった開発用ライブラリーも存在する。これらは特定の攻撃に対する防衛策としては有効だが、開発時点で分かっている攻撃にしか対応できず、対象とするアプリケーション分野も狭いのが欠点である。

二つ目の、実行時に監視・制限する方法は、SELinuxに代表される種類がもっとも強力な機能を提供している。これは、ファイルへのアクセスとプログラムの実行の権限を細かく制限するもので、明示的に許可されたもの以外はすべてブロックされる。非常に強力な仕組みではある一方、アプリケーションの動作に詳しくないと設定ができず、非常に煩雑であるのが欠点である。その他には、ウィルス対策ツールに代表されるものもあるが、対処療法的であり、未知の攻撃やシステム内部のバグに起因する誤動作には効果は全くない。

２．研究の目的

前項で述べた通り、現在活用できるツールはどれも専門知識が必要であり、手間もかかるために、一般的な開発の中で使われることは少ない。そこで、本研究では、セキュリティーの専門知識を必要とせず、一般的な開発作業のなかでコンポーネントの入出力と動作の制限を定義できる手法を提供することが目的である。具体的には、これらの機能をコンポーネント指向開発のフレームワークとして提供することで、より多くのシステム開発において活用できるようにする。

３．研究の方法

コンポーネントの入出力データとコンポーネントの動作を定義する自己説明型言語を設計する。これには、これまで研究者らが開発してきたcyberFilmを土台とする。CyberFilmは、ソフトウェアで行う計算処理を高度に自己説明的な表現で記述することを可能にする記述言語である。その中ですでに開発済みのアルゴリズム定義用言語の分類およびアイコン等を活用しながら入出力データの種類や性質を表現する言語を設計する。一方、入出力データの性質については、入出力の時間、周期性などの時間的性質など、cyberFilmではまだ扱っていないものについては新たな分類、定義を行う。また、コンポーネントの動作を定義する部分については新たに設計する。

一方、実行環境の開発では、cyberFilmの既存の実行環境に機能を追加する方向で開発を行う。完全ではないがすでに、cyberFilmで記述された計算からプログラムを生成し、実行する環境がある。それに、入出力データや動作の制約が守られていることを確認する機能を新たに追加する。さらに、制約が破られた際の例外処理を担う機構を考案し、機能を追加する。

４．研究成果

入出力定義用言語、動作定義用言語、定義用エディタ、実行環境、検証の5項目に分けて説明する。

入出力定義用言語

監視対象とする入出力データについて、その性質を定義するための言語を開発した。まず、対象とする入出力を次のようにした。監視の対象とするのは、標準入出力やファイルに対して書き込みやそれらからの読みを行う明示的な入出力関数と、関数の呼び出しに用いる引数と戻り値を含む暗黙的な入出力関数とした。データの性質については、意味種類、変数型、単位、最大値、最小値、初期値とした。この内、意味種類と単位は値が表すものごとに対応した分類から選択する性質である（発表論文②⑤⑥）。その具体例は図1の通りである。

記号	意味種類	単位
	距離	cm,m,km…
	速度	km/h,m/s,kts…
¥	金額	円,$, ユーロ…
	温度	℃,°F,K…

図 1　データの性質を表す記号

意味種類を開発者に指定させることで、その値の使用や操作、取り得る値の範囲について明らかにし意識させる狙いがある。また、意味種類や単位を用いて引数や戻り値を他変数に代入する際や計算式で利用する際に整合性の確認を行うことができる。例えば、「重量」の値に「速度」の値を代入しようとした時には、編集時にその操作を禁じる対応が、実行時であれば例外を発するなどの対応をする。これにより、プログラム内で用いられるデータの性質が明確になり、想定外の入出力データによる誤作動や、プログラムミスによるバグを防止できる。

定義された入出力データの性質を用いた監視は、その変数を用いた関数呼び出しの段階で実施される。具体的には、中間コードとして利用するプログラミング言語のライブラリーに対して、用意したラッパー関数と置き換えることで監視用のロジックを動作させる。変数毎に定義されるデータ定義は図 2 の通りである。

```
変数名 : xd
変数 ID : v1001
種類　: width
単位　: nm
変数型 : int
最小値 : 0
最大値 : 100
```

図 2　変数のデータ定義

この図では、xd という変数をユーザが宣言した場合に定義される内容を示している。変数 ID は、変数固有の識別子で、実変数とデータ定義を関連付けるのに用いられる。種類は、変数が表す値の意味を表す。この例の場合は、幅である。その他、この変数で表す値の単位はナノメートル、最小値は 0、最大値は 100 ということが定義されている。

動作定義用言語

実行時の振る舞いを監視するため、その振る舞いを定義するための言語を開発した。まず、監視の対象を関数呼び出しおよびコードブロックとした。関数呼び出しは、特定の関数を呼び出す前と後で、引数や戻り値の検査に加え、あらかじめ定義された環境条件も検査する。コードブロックについては、そのブロックの直前と直後に環境条件について検査する。ここでいう環境条件とは、プログラムの状態を表す値であり、今回は、実行開始時刻、実行終了時刻、大域変数の値をまず対象とした。さらに、関数呼び出しの検査では、特定の関数に渡される引数の累積値、特定の関数の戻り値の累積値を検査することもできる。また、これらの関数やコードブロックの呼び出し順序、タイミング（時刻と時刻間隔）、頻度も定義できる。これをまとめたのが図 3 である。

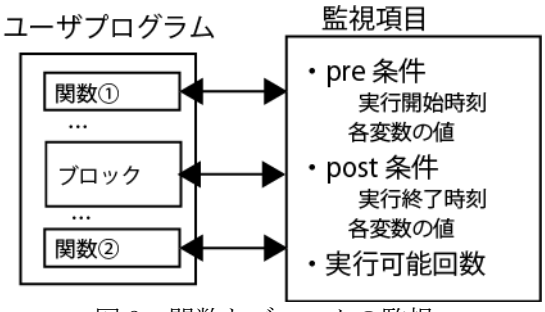

図 3　関数とブロックの監視

目的の関数あるいはコードブロックそれぞれに対して、監視項目に列挙した内容を定義できる。pre 条件は、ブロックを実行する前に行う検査で、実行が可能な時刻や時間帯を指定したり、特定の変数の値を条件にしたりできる。post 条件では、同様の内容について、実行終了後に検査を行う。その他に、累積的な実行可能回数を定義でき、定められた回数以上の実行を阻止することができる。

ここでいうコードブロックは、cyberFilm でのシーン（何らかの意味を持つ一まとまりの計算）を暗に指している。本研究では、対象とするフレームワークを cyberFilm に限定しているため、このブロックの指定はユーザにとっても自然なことであると考えることができる。しかし、他のフレームワークに適応する場合、このブロックをユーザが任意に指定するには困難があるのは確かである。

定義用エディタ

細かい定義にかかるユーザの負荷を低減することは、本研究の重要な課題であり目的である。軽減することができなければ、正確で

完全な情報をユーザから得ることができなくなり、監視と検査の精度が落ちることになる。そこで 2 つの方法で対処する。一つは、指定すべきデータの性質を明確化し、ユーザが意識すべき事柄をできるだけ明確にする。このことにより、確認漏れや指定ミスを抑止する効果を狙う。もう一つは、各性質には規定の初期値を自動的に設定するようにし、ユーザは変更が必要な箇所のみを設定するようにした。

実行環境

ユーザが指定した制約にもとづいて監視と制御を行う方法として、監視対象の関数を検査用コードが埋め込まれたラッパー関数と置き換える方法を検討した。例えば、数値計算で良く用いられる sin 関数の例を考える。Java 言語では Math クラスの sin 関数を用いるが、その場合の置き換えは次の図のようになる。

変換前： Math.sin(変数)
↓
変換後： asc.sin(変数 ID)

図 4　関数の置き換え

変換後の関数では Asc クラスのインスタンスである asc を用いて、その sin 関数を呼び出している。Asc クラスは、実行時の入出力データや動作を監視する監視モジュールである。このモジュールの概略は図 5 の通りである。

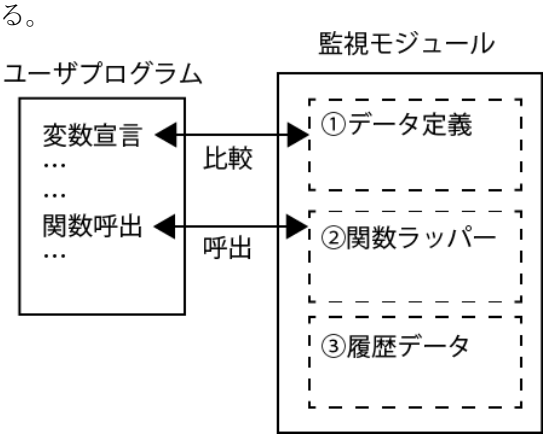

図 5　実行時監視の方法

本モジュールには変数のデータ定義、関数呼び出し用の関数ラッパー、入出力データの累積値やコードブロックの実行時刻を記録する履歴データで構成される。監視が行う手順は大まかに次の通りである。まず、関数呼出がユーザプログラムで発生すると、監視モジュールの対応する関数ラッパーに制御が移る。関数ラッパーの中では、入力データの確認、実関数の呼出、出力データの確認、戻り値を返して終了の各処理がこの順に行われる。図 4 の変換後の関数からも分かる通り、関数には、実変数ではなく、変数 ID が渡される。監視モジュールでは、変数 ID をもとに該当するデータ定義を取り出し、元のユーザプログラムの変数宣言等から変数値を取得し、これらを比較することにより処理の正当性を確認する。さらに、必要な場合は、入出力データの累積値計算に用いるために履歴データに入力データと出力データを記録する。

監視を実行し、ユーザが定めた制約に違反するデータや動作があった場合は例外処理が実行される。例外処理には 3 つの種類があり、定義時にユーザが選択できる。これらは、プログラムの実行を終了する、データを再設定して実行を続ける、例外を記録し実行を継続する、という 3 つである。データの再設定は、入出力データの値が想定の範囲を逸脱した場合に発生する。データ定義の際、ユーザは、例外発生時に再設定する値を指定できる。

実装した実行環境のアプリケーション画面を図 6 と図 7 に示す。

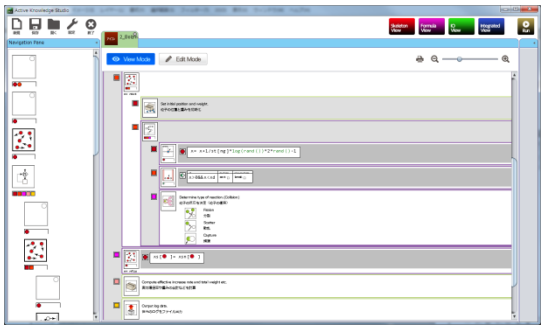

図 6　定義・実行機能を実装した環境

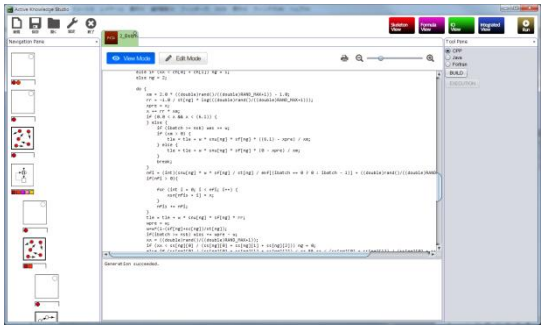

図 7　生成された中間コードと実行環境

本研究では、cyberFilm のフレームワークに今回の提案を適用するため、cyberFilm のプログラミング環境に機能を実装した。図 6 の画面は cyberFilm で計算を定義する画面であるが、入出力データと動作の定義も同様に見

ることができる。また、図 7 は cyberFilm の表記から中間コードを生成し、それを実行する画面である。ここで出力された中間コードは、すでに関数の変換が行われ、実行すると監視モジュールによる監視が動作する。

検証
熱拡散計算プログラム（発表論文⑦）と粒子の衝突反応計算プログラム（発表論文①）の 2 つについて、上述の方法と試験実装を用いて監視条件の定義と実際の実行を試した。その結果、今回実現した仕組みによって入出力データと動作の定義が可能で、実行時に監視と検査ができることを確認した。本手法を用いて cyberFilm を用いたコンポーネント開発フレームワークに対して、高いレベルのセキュリティーを付加できることを確認できた。今後は、cyberFilm 以外のフレームワークへの応用も検討したい。

一方、今回は性能を評価するまでは至らず、大量の監視対象がある場合の性能への影響を評価することが課題である。また、試験実装では対応変数の意味種類や単位、動作の種類が限定的であった。同様の手法でバリエーションを増やし、対応関数を増やすことは基本的には容易であるが、対応範囲の拡張とそれに伴う言語と監視手法の検証も必要である。

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 7 件）

① 菱沼信彦、猪狩潤、吉岡廉太郎、A Method to Annotate Programs with High-Level Knowledge of Computation 、 World Academy of Science, Engineering and Technology、62 巻、 469-476、2012、査読有り

② 渡部有隆、吉岡廉太郎、ニコライミレンコフ、Programming in pictures: a way toward reliable software 、 Proceedings of 10th International Conference on Software Methodologies, Tools, and Techniques 、 183-197、2011、査読有り

③ 渡部有隆、吉岡廉太郎、ニコライミレンコフ、Programming in pictures within Filmification Modeling Environment 、 Proceeding of IEEE Symposium on Visual Languages and Human-Centric Computing、267-268、2011、査読有り

④ 吉岡廉太郎、渡部有隆、ニコライミレンコフ、Open Set of Algorithmic Characters 、 Proceedings of 10th WSEAS International Conference on Applied Computer Science (ACS'10)、327-334、2010、査読有り

⑤ 渡部有隆、吉岡廉太郎、ニコライミレンコフ、Embedded Clarity in Filmification of Methods 、 New Trends in Software Methodologies, Tools and Techniques、217 巻、70-82、2010、査読有り

⑥ 柴田崇由、中村和之、佐藤崇信、吉岡廉太郎、A Knowledge Sharing System for Software Developers 、 Proceedings of 5th International Conference on Software and Data Technologies、499-503、2010、査読有り

⑦ 吉岡廉太郎、渡部有隆、ニコライミレンコフ、角山茂章、遠藤寛、透明性にもとづく信頼性の高いソフトウェアの実現に向けて、日本原子力学会「2011 年春の年会」予稿集、428、2010、査読なし

〔学会発表〕（計 4 件）

① 吉岡廉太郎、A Method to Annotate Programs with High-Level Knowledge of Computation, 2012 International Conference on Computer Science and Applications、2012、クアラルンプール、マレーシア

② 渡部有隆、Programming in Pictures within Filmification Modeling Environment, 2011 IEEE Symposium on Visual Languages and Human-Centric Computing、2012、ピッツバーグ、米国

③ 吉岡廉太郎、Open Set of Algorithmic Characters 、 10th WSEAS International Conference on Applied Computer Science (ACS'10)、2010、安比

④ 吉岡廉太郎、A Knowledge Sharing System for Software Developers、5th International Conference on Software and Data Technologies、2010、アテネ、ギリシャ

〔図書〕（計 0 件）

〔産業財産権〕
○出願状況（計 0 件）

○取得状況（計０件）

〔その他〕
なし。

６．研究組織
(1)研究代表者
吉岡 廉太郎（ YOSHIOKA RENTARO ）
会津大学・コンピュータ理工学部・准教授
研究者番号：00360008
(2)研究分担者
なし

(3)連携研究者
なし