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

平成　２９　年　　６　月　２７　日現在

機関番号：　１７１０２
研究種目：　挑戦的萌芽研究
研究期間：　2013～2016
課題番号：　２５５４００２５

研究課題名（和文）滑らかな設計抽象化に関する研究

研究課題名（英文）A Study on Fluid Design Abstraction

研究代表者

鵜林　尚靖（Ubayashi, Naoyasu）

九州大学・システム情報科学研究院・教授

研究者番号：８０３７２７６２

交付決定額（研究期間全体）：（直接経費）　　2,900,000円

研究成果の概要（和文）：抽象化はソフトウェア開発において重要な役割を果たすが，どの関心事を設計としてモデル化し，どの関心事をコードとして記述すべきかを判断することは容易ではない．また，開発の進行と共に設計とコードの間のバランスを再考をせざるを得ず，その結果として，両者間の抽象度が変化する．本研究では，適切な抽象化を支援するための概念「滑らかな設計抽象化」，開発支援環境 iArch および抽象化のためのソフトウェアパターン集を提案した．「滑らかな設計抽象化」とは，設計モデリングとプログラム実装の間を滑らかに移動した収束点として適切な抽象度が得られるという考え方である．

研究成果の概要（英文）：Abstraction plays an important role in software development. However, it is not easy to decide which concern should be described in a design model and which concern should be written in code. An abstraction level of a design specification may change as a result of reconsidering the balance between design and code. We proposed the notion of fluid design abstraction, its support tool iArch, and software patterns for abstraction. An appropriate abstraction level can be captured by the convergence of fluid moving between design modeling and program implementation.

研究分野：　ソフトウェア工学

キーワード：　ソフトウェアアーキテクチャ　インタフェース　モジュール　抽象化　整合性検証

１．研究開始当初の背景

ソフトウェア開発において「抽象化」は最重要課題の一つである．ソフトウェア工学の世界的権威である英国インペリアルカレッジの Kramer, J.は「明快でエレガントな設計やプログラミングができるエンジニアとそうでないエンジニアがいるのは何故であろうか？」と問題提起をした上で，「その答えは抽象化能力にある」と主張している[1]．応募者は大学に赴任する前は企業に長く在籍しエンジニアとして様々なソフトウェア開発に携わってきた．一人のエンジニアとして一番悩み，結局のところ在籍時に解答を見つけることが出来なかった問題がある．「設計はどこまで詳しく記載すれば良いのか」が分からなかったのである．抽象的な設計はお絵描きになりコードには直接つながらない．一方，現在主流のモデル駆動開発（MDD: Model-Driven Development）のように設計をコードに近いレベルにまで詳細化すれば両者の関係は明確になるが，残念ながらそのような設計はあまり役に立たない．設計は「適切に」抽象的でなければ意味がないからである．しかし，その「適切な」レベルはどうやって見つけたら良いのであろうか？ソフトウェア工学研究における難問の一つである．また，現状の MDD では残念ながら抽象化機構にまで議論が及んでいない．

２．研究の目的

本研究は，第１章で述べた背景の下，ソフトウェア開発における「抽象化」を支援するための理論とその理論に基づいた開発支援環境を提供することを目的に実施した．なお，抽象化の範囲として，本研究では，ソフトウェア設計とコードの間の抽象化を取り扱うことにした．コードに対して，ソフトウェア設計をどのレベルまで抽象的に書けば良いのか（本研究では「抽象度」と呼ぶ）を決定するとともに，設計とコードの間のトレーサビリティを保持する技術を開発した．

基本方針として，抽象度を記述するための言語機構，そのためのコンパイラ，適切な抽象度を探索するためのソフトウェアパターンを開発することに焦点を当てた．コンパイラはソフトウェア開発者が日々利用する道具であり，その中に「抽象化」のための支援機構を取り入れるのが，研究内容の普及という観点で優れていると判断したからである．通常，コンパイラというと，コードを対象にするが，本研究ではコードと共に設計モデル（UML で表記）もコンパイルの対象とした．これにより，ソフトウェア設計とコードの間の抽象化を言語機構の中で処理することが可能となった．

[1] Kramer, J.: Is Abstraction the Key to Computing?, *Communications of the ACM*, vol.50 issue 4, pp.36-42, 2007.

３．研究の方法

(1) 基盤となる既存研究

本研究では，応募者が提唱するインタフェース機構 *Archface* [2]を発展させ，反復的に設計とコード間のバランスを再考し「滑らかに」抽象度を決定するための技術を構築した．この反復の収束点が「適切な」抽象化となる．設計はソフトウェアアーキテクチャの抽象的な記述であり，コードはその設計を実装したものである．設計とコードは関心事の分離の観点から明確に切り分けるべきであるが，現実的には容易ではない．実際，抽象度はソフトウェア開発の進行と共に変化しがちである．エンジニアは万能ではないので，設計に関わる関心事を開発の初期段階ですべて把握できるとは限らない．どうしても，設計と実装の間で行ったり来たりせざるを得ない状況が発生する．

ソフトウェア設計とアーキテクチャに関する重要な研究領域の一つとして，Taylor, R. N.らは，設計と実装の間を流動的に移動できるための支援の必要性を主張している[3]．流動的に移動することにより設計とコードの間のバランスを再検討する必要性が生じ，その結果として設計の抽象度も流動的に変化する．すなわち，どの関心事を設計が分担に，どの関心事をプログラム記述に任せるべきかを常に判断する必要性が生じる．

(2) 本研究の基本アイデア

本研究で提案する「滑らかな設計抽象化」は「設計と実装の間を流動的に移動し，その収束の結果として適切な抽象度が獲得される」という考え方である．抽象化の視点からのソフトウェアリファクタリングと言ってもよく，従来にない斬新性がある．この場合のリファクタリングは，従来のコードあるいは設計モデルのみのリファクタリングとは異なり，それらを含め一つのソフトウェアを構成する成果物（設計およびコード）にまたがったリファクタリングである．

「滑らかな設計抽象化」という用語を使用した理由は，設計および実装フェーズにおける抽象化は絶対的に固定したものではなく，設計とコードの最適な組み合わせを柔軟に探索すべき（抽象度は連続かつ滑らかに変化させていくべき）だという考えに基づいている．しかし，「滑らかな設計抽象化」のための技術は世の中には存在せず，この実現には大きなチャレンジ性があり，鍵となるのが新

[2] 基盤研究(B): 高信頼ソフトウェアアーキテクチャ構築に関する研究（2011-2013）研究代表者（鵜林尚靖）

[3] Taylor, R. N. and Hoek, A.: Software Design and Architecture --The once and future focus of software engineering, Proceedings of 2007 Future of Software Engineering (FOSE 2007), pp.226-243, 2007.

しいインタフェース機構 *Archface* である．

(3) 提案内容

「滑らかな設計抽象化」を実現するための機構は，1) 抽象度を設定するための *Archface*，2) 抽象度を測定するためのメトリクス，3) 設定した抽象度を保持しつつ設計とコードの整合性を検証するための機構，の 3 本柱で構成される．

①抽象度を設定するための *Archface*

Archface では構造的設計点（クラスやメソッドなど）の他に振る舞い的設計点（メッセージ送受信など）を公開する．この公開設計点の個数が，(2)で述べるように，設計とコード間の抽象度を決定する．

Archface ではプログラムモジュールだけでなく設計モデルもモジュールと考えるため，設計モデルとコードの両方に対して機能する型システムを構築する必要がある．モジュールとしての設計モデルがそのインタフェースすなわち *Archface* に適合しているとは，設計モデルの中に公開した設計点が含まれていることを意味する．一方，クラスなどのプログラムモジュールが *Archface* に適合するとは，設計点に対応するプログラム点がコード中に存在する場合である．なお，設計モデルとしては産業界で広く利用されている UML モデルを，プログラミング言語としては Java をサポートの対象とした．ただし，理論をシンプルにさせるため，クラス図（構造的側面のモデル）とシーケンス図（振る舞い的側面のモデル）のみを考察の対象とした．設計モデルに対する型システムは従来にないアイデアである．

②抽象度を測定するためのメトリクス

抽象度については，プログラム点に占める公開設計点の比率によって算出することにした．このメトリクスの閾値（設計はどの程度コードよりも抽象的であるべきかの指針となる値）は，「適切な」抽象度か否かを判断する際に影響を与える．すなわち，設計と実装の間の流動的移動を収束させるための条件となる．

③設定した抽象度を保持しつつ設計とコードの整合性を検証するための機構

本研究では，設計とコードの間に *Archface* を導入し，型検査により両役割の間の対応をとることを目指した．「設計と *Archface*」「コードと *Archface*」の両方について型検査をパスすれば，*Archface* で設定した抽象度を保ちつつ，設計とコード間のトレーサビリティを確保することが可能となる．抽象化を型検査に帰着させるという従来にない斬新なアイデアと言える．最終的には，設計モデルとコードの双方に対して適用可能な *Archface* コンパイラとして実現した．

本研究では，上記の 3 つの課題を解決するための理論を構築すると共に，*Archface* コンパイラを核とする「抽象化」支援開発環境 *iArch* とソフトウェアパターンを提供することができた．

４．研究成果

平成 25 年度は *Archface* のための型システムについての研究を行い，平成 26 年度は基本アイデアを数理的に形式化した．平成 27 年度と 28 年度は，我々のアイデアをツール化した *iArch* の機能を充実させた[4]．*iArch* は平成 28 年秋にオープンソースソフトウェアとして公開した．*Archface* コンパイラに関する基礎理論については，ソフトウェア工学のトップカンファレンスである ASE2014 に論文採録された．

以下，主な研究成果について述べる．

(1) *Archface* コンパイラの理論構築

本研究は究極的には「設計モデルもプログラムモジュールもすべて一つのソフトウェアを構成するモジュールである」という新たなモジュラリティ哲学の構築を狙ったものであり，*Archface* コンパイラの理論構築により，その一端を実現することができた．開発者は設計モジュール（設計モデル）群とプログラムモジュール群（クラスなど）を開発し，それをコンパイラにかけると，両者の抽象度を保ちつつ，実行オブジェクトが生成される．

(2) 統合開発環境 *iArch* のオープンソース化

Archface コンパイラを核に「抽象化」という側面から，設計モデリング，プログラミングを支援するオープンソース統合開発環境 *iArch* を開発した．*iArch* は，1)モデルおよびプログラムエディタ，2)設計モデルからの *Archface* 生成機能，3) *Archface* チェッカ（モデルおよびコードに対する型検査），4)抽象度の表示機能をサポートする．*iArch* は Eclipse プラグインとして提供される．

(3) 抽象化のためのパターン集の作成

第 1 章および第 2 章で述べたように，適切に抽象化された設計を行うのは容易ではなく，設計とコードの間を行ったり来たりしながら，何度も両者の関係を見直していく必要がある．

---

[4] *iArch* のプロトタイプは前述の基盤研究(B)で開発し，本研究および後述の基盤研究(A)でオープンソースソフトウェアとして公開できるレベルにした．なお，本研究では型検査の理論構築に注力した．

本研究では，この問題を解決するための方法として，抽象化のためのソフトウェアパターン集（リファクタリングカタログ）を作成した．このパターン集は，開発者が抽象化の観点から設計モデルやソースコードを見直す際のノウハウを整理したものであり，大きく分けて 2 つのカテゴリから構成される．1 つは，「設計モデルからソースコードへの関心事の移動(MoveM2C: Move concerns from Model to Code)」であり，たとえば，実装の詳細に関わる関心事が設計モデルに含まれている場合に，モデルからその関心事を削除し，ソースコードのみにその関心事を含める場合に使用する．この場合，設計モデルの抽象度は上がる．もう 1 つのカテゴリは，「ソースコードから設計モデルへの関心事の移動(MoveC2M: Move concerns from Code to Model)」である．これは，ソースコードに含まれる関心事で設計モデルにも反映すべきものを抽出するためのものである．設計モデルを読んでも実装のイメージが掴めないときに有効なパターンである．この場合，設計モデルが詳細化されるので抽象度は下がる．本パターン集では，これら 2 つのカテゴリを細分化し 16 個のカタログとして作成した．

(4) 基盤研究(A)とのシナジー効果

本研究と並行して，平成 26 年度より「不確かさを包容するモデル駆動開発機構に関する研究」（基盤研究(A)）をスタートさせており，本研究の成果を「不確かさ」に関わる研究に応用した．不確かさが扱えるように *Archface* を拡張した *Archface-U*, *iArch* を拡張した *iArch-U* を開発した．*Archface-U* および *iArch-U* はそれぞれオリジナルの *Archface*, *iArch* の機能を含んでおり，上位互換となっている．

本研究の代表者は，インタフェース機構はソフトウェア開発に関する重要概念を言語処理機構として扱うのに非常に適していると考えている．基盤研究(A)では「不確かさ」を *Archface* に取り込んだが，その他の重要概念も *Archface* にモジュラーに導入して行けるのではないかと考えている．*Archface* を今後の研究の土台として位置付けて行きたい．

(5) 研究成果の公開

平成 28 年度には，*iArch-U*（オリジナルの *iArch* の機能を含む）をオープンソースソフトウェアとして GitHub から正式に公開すると共に，英語版の公式 Web サイトを開設し，研究成果を世界に向けて発信した．

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 8 件）

① Keisuke Watanabe, Naoyasu Ubayashi, Takuya Fukamachi, Shunya Nakamura, Hokuto Muraoka, and Yasutaka Kamei, iArch-U: Interface-Centric Integrated Uncertainty-aware Development Environment, Proceedings of the 9th International Workshop on Modelling in Software Engineering (MiSE 2017) (Workshop at ICSE 2017) , 査読有, 2017, to appear

② 深町 拓也, 亀井 靖高, 鵜林 尚靖, ICSE2015 参加報告, 情報処理, Vol.56, No.9, 査読無, 2015, pp.930-931

③ Takuya Fukamachi, Naoyasu Ubayashi, Shintaro Hosoai, and Yasutaka Kamei, Modularity for Uncertainty, Proceedings of the 7th International Workshop on Modelling in Software Engineering (MiSE 2015) (Workshop at ICSE 2015) , 査読有, 2015, pp. 7-12

④ Takuya Fukamachi, Naoyasu Ubayashi, Shintaro Hosoai, and Yasutaka Kamei, Conquering Uncertainty in Java Programming, Proceedings of the 37th International Conference on Software Engineering (ICSE 2015),Poster, 査読有, 2015, pp.823-824

⑤ Peiyuan Li, Naoyasu Ubayashi, Di Ai, Yu Ning Li, Shintaro Hosoai, and Yasutaka Kamei, Sketch-based Gradual Model-Driven Development, Proceedings of International Workshop on Innovative Software Development Methodologies and Practices (InnoSWDev) (Workshop at FSE 2014), 査読有, 2014, pp.100-105

⑥ Naoyasu Ubayashi, Di Ai, Peiyuan Li, Yu Ning Li, Shintaro Hosoai, and Yasutaka Kamei, Abstraction-aware Verifying Compiler for Yet Another MDD, Proceedings of the 29th International Conference on Automated Software Engineering (ASE 2014), 査読有, 2014, pp.557-562

⑦ Di Ai, Naoyasu Ubayashi, Peiyuan Li, Daisuke Yamamoto, Yu Ning Li, Shintaro Hosoai, and Yasutaka Kamei, iArch: An IDE for Supporting Fluid Abstraction, Proceedings of Companion Publication of the 13th International Conference on Modularity (Modularity'14), 査読有, 2014, pp.13-16

⑧ Di Ai, Naoyasu Ubayashi, Peiyuan Li, Shintaro Hosoai, and Yasutaka Kamei, iArch: An IDE for Supporting Abstraction-aware Design Traceability, Proceedings of the 2nd

International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2014), 査読有, 2014, pp.442-447

〔学会発表〕(計 11 件)

① Shunya Nakamura, Takuya Fukamachi, Naoyasu Ubayashi, Yasutaka Kamei, and Shintaro Hosoai, An Uncertainty-Aware Model Checker Embracing Known Unknowns, Poster, 35th International Conference on Conceptual Modeling (ER 2016), 2016 年 11 月 14 日～2016 年 11 月 17 日, 岐阜

② 深町 拓也, 鵜林 尚靖, 細合 晋太郎, 亀井 靖高, Git 連携による不確かさマネジメントシステム, 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 2016(SES 2016), 2016 年 08 月 31 日～2016 年 09 月 02 日, 東京

③ 中村 隼也, 渡辺 啓介, 深町 拓也, 鵜林 尚靖, 細合 晋太郎, 亀井 靖高, 不確かさを包容する統合開発環境 iArch-U, 情報処理学会ソフトウェア工学研究会, 2016 年 07 月 13 日～2016 年 07 月 15 日, 札幌

④ Naoyasu Ubayashi, Modular Programming and Reasoning for Dealing with Uncertainty in CPS, NII Shonan Meeting on Architecture-Centric Modeling, Analysis, and Verification of Cyber-Physical Systems, 2016 年 03 月 21 日～2016 年 03 月 24 日, 葉山

⑤ Naoyasu Ubayashi, Modularity for Uncertainties in Adaptive Software Systems, 2015 Shonan Workshop on Engineering Adaptive Software Systems(EASSY 2015), 2015 年 09 月 07 日～2015 年 09 月 10 日, 葉山

⑥ 郭 衆小, 鵜林 尚靖, 艾 迪, 李 沛源, 李 宇寧, 深町 拓也, 細合 晋太郎, 亀井靖高, 抽象化を考慮したデータフロートレーサビリティ, 電子情報通信学会ソフトウェアサイエンス研究, 2015 年 03 月 09 日～2015 年 03 月 10 日, 那覇

⑦ 艾 迪, 鵜林 尚靖, 李 沛源, 李 宇寧, 細合 晋太郎, 亀井 靖高, 設計抽象化のためのリファクタリング支援, 日本ソフトウェア科学会 第 21 回ソフトウェア工学の基礎ワークショップ(FOSE 2014), 2014 年 12 月 11 日～2014 年 12 月 13 日, 霧島

⑧ 艾 迪, 鵜林 尚靖, 李 沛源, 李 宇寧, 細合 晋太郎, 亀井 靖高, 設計抽象化のためのリファクタリングパターン, 情報処理学会ソフトウェア工学研究会, 2014 年 07 月 09 日～2014 年 07 月 11 日, 富良野

⑨ 艾 迪, 李 沛源, 細合 晋太郎, 亀井 靖高, 鵜林 尚靖, iArch: 滑らかな設計抽象化を支援する IDE [ライブ論文], 日本ソフトウェア科学会 第 20 回ソフトウェア工学の基礎ワークショップ (FOSE 2013), 2013 年 11 月 28 日～2013 年 11 月 30 日, 加賀

⑩ 鵜林 尚靖, モデル駆動開発とドメイン特化言語, 情報処理学会 組込みソフトウェアシンポジウム 2013 (ESS2013) チュートリアル, 2013 年 10 月 16 日, 東京

⑪ 鵜林 尚靖, 艾 迪, 細合 晋太郎, 亀井靖高, 滑らかな設計抽象化, 電子情報通信学会ソフトウェアサイエンス研究会, 2013 年 07 月 25 日～2013 年 07 月 26 日, 札幌

〔図書〕(計 1 件)

① 平山 雅之, 鵜林 尚靖, ソフトウェア工学, オーム社, 2017, 214 頁

〔産業財産権〕

○出願状況 (計 0 件)

名称：
発明者：
権利者：
種類：
番号：
出願年月日：
国内外の別：

○取得状況 (計 0 件)

名称：
発明者：
権利者：
種類：
番号：
取得年月日：
国内外の別：

〔その他〕

① 研究室ホームページ
http://posl.ait.kyushu-u.ac.jp/

② iArch-U 公式 Web サイト
http://posl.github.io/iArch/

６．研究組織

(1)研究代表者
鵜林 尚靖 (UBAYASHI, Naoyasu)
九州大学・
大学院システム情報科学研究院・教授
研究者番号： ８０３７２７６２

(2)研究分担者
なし

(3)連携研究者
なし

(4)研究協力者
なし