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　３０　年　　６　月　２８　日現在

機関番号：　３２６１９
研究種目：　基盤研究(C)（一般）
研究期間：　2015～2017
課題番号：　１５Ｋ００１０６

研究課題名（和文）大規模アジャイル開発のための超柔軟なアーキテクチャ構成の研究

研究課題名（英文）Research on Super-flexible Software Architecture Modelling Mechanism for Agile Development

研究代表者

野田　夏子（Noda, Natsuko）

芝浦工業大学・デザイン工学部・准教授

研究者番号：６０７０７７０１

交付決定額（研究期間全体）：（直接経費）　　2,100,000円

研究成果の概要（和文）：近年の変化の激しいビジネス環境や技術環境に対応するために、ソフトウェアプロダクトライン等の大規模なソフトウェア開発を対象にして、様々な変更に対してソフトウェアアーキテクチャの変更を必要とせずアジャイルに開発を行えるための、超柔軟なソフトウェアアーキテクチャの構成メカニズムを構築した。
アジャイル開発とソフトウェアアーキテクチャの関係を整理することにより、ソフトウェアアーキテクチャの柔軟化のためにアスペクト指向技術が有効であることを確認した。その上で、アスペクト指向によるソフトウェアアーキテクチャ構成メカニズムを構築し、その有効性をプロダクトラインアーキテクチャの事例で確認した。

研究成果の概要（英文）：We developed a modeling mechanism for super-flexible software architecture, which realize agility in large-scale software development such as software product lines without requiring any changes on software architecture when various types of changes on specification and/or environments are needed.
After surveys on the relationship between agile development and software architecture, we confirmed that aspect-oriented technologies are effective to make software architecture flexible. Based on this confirmation and knowledge, we developed an aspect-oriented modeling mechanism for software architecture. We confirmed its effectiveness in some cases of product line architecture developments.

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　アスペクト指向　モデリング　ソフトウェアプロダクトライン

１．研究開始当初の背景

近年、ビジネス環境や技術環境はますます激しく変化しており、ソフトウェアに対する要求もめまぐるしく変化している。こうした変化に迅速に対応するためにアジャイル開発が普及し主流になりつつある。また一方で多様な要求に効果的に対応するために、ソフトウェア単品を独立して開発するのではなく製品ファミリーを体系的に開発するプロダクトライン開発の適用が広がっているが、従来比較的小規模なソフトウェアを対象に適用されてきたアジャイル開発が、プロダクトライン開発のような規模の大きな開発にも適用されるようになりつつある。ソフトウェアアーキテクチャはソフトウェアの特性を決定づけるものであり、どのような開発スタイルであってもその時点での要求に合致したアーキテクチャを適用することは必須である。しかし、規模が大きくなるほど、堅牢なソフトウェアアーキテクチャはしばしば変化への迅速な対応と衝突するため、アジャイル開発においてどのようにアーキテクチャを扱うかが大きな課題となっている。従来アーキテクチャは、例えばクライアント・サーバアーキテクチャや３層アーキテクチャといった骨格構造を決定し、その構造上にアプリケーションの機能を配置するという構成をとってきた。しかしスケールの変化等により特性に関する要求が変化すると、それに応じて根底にある骨格構造の変更が必要となるため、迅速なアーキテクチャの変更が困難である。アジャイル開発における迅速な変化に対応するには、全く新しい柔軟なアーキテクチャ構成論が必要とされており、アーキテクチャ構成論とその構成メカニズムを明確化することが求められている。

２．研究の目的

本研究の目的は、大規模ソフトウェアのアジャイル開発を実現するための、超柔軟なソフトウェアアーキテクチャ構成論及びそのための構成メカニズムを提案、評価することである。ソフトウェアアーキテクチャは重要であるが、決定した骨格構造の上に機能を配置する従来のソフトウェアアーキテクチャ構成論では近年適用が広がっているアジャイル開発における迅速な変更に対応できず、アジリティの実現が妨げられる。本研究では、これらの課題を解決するために、ソフトウェアを様々な関心事毎に分離して構築する技術である、アスペクト指向技術に注目する。この技術をソフトウェアアーキテクチャ設計に応用し、骨格構造と機能の依存関係を逆転するソフトウェアアーキテクチャ構成論と構成メカニズムを提案、評価する。

３．研究の方法

本研究は以下のように、問題分析を踏まえてアーキテクチャ構成論を明らかにした上で、アーキテクチャ構成メカニズムの構築、評価を繰り返しながらインクリメンタルに進める。

第１ステップ：大規模ソフトウェアのアーキテクチャに関する事例や文献及びアジャイル開発に関する事例や文献の調査、分析を行い、超柔軟なアーキテクチャ構成論への要件の整理を行う。整理した要件を満たすアーキテクチャ構成論を構築する。構成論を実現する構成メカニズムの構築のために、提案済みのアスペクト指向モデリングメカニズムの改良、修正を行う。

第２ステップ：アーキテクチャ構成メカニズムを体系化する。これを企業事例に適用して評価する。評価結果に基づき、メカニズムをリファインすることを繰り返す。さらに、構成論とメカニズムに基づくアーキテクチャ設計手法を整理する。

４．研究成果

①　アジャイル開発とアーキテクチャ設計の関係の整理

アジャイル開発におけるアーキテクチャ構成論を明確にするために、近年多く行なわれているアーキテクチャを中心とした開発形態とアジャイル開発の関係を整理し、そこからアジャイル開発とアーキテクチャ設計の関係の整理を行った。

まずモデル駆動開発である。モデル駆動開発とは、モデルを変換することによりソフトウェアを開発するアプローチである。変換によりソフトウェアを開発するため、要求の変化に対しても、モデルを変更することにより最終的なソフトウェアへと変化を反映させることができる。モデル駆動開発の出現によってモデルをアジャイル開発に組み込むことが可能になった。しかし、課題になるのがモデル変換である。要求の変化が大き過ぎてソフトウェアの根本の構造も変化するような場合、既存のモデル変換では対応できないことも起こり得る。設計したモデルをモデル変換で変換するという枠組みで考えるのではなく、モデル変換自体も設計対象として考える必要がある。

次にプロダクトライン開発である。短いサイクルを繰り返すことで変化の都度に対応するアジャイル開発に対し、プロダクトライン開発ではある程度変化を見越して備えることで、それぞれ異なる部分を含んだプロダクトをひとつひとつ開発するより短い期間で開発する。プロダクトライン開発は「重い」開発の典型であり、「軽い」開発であるアジャイル開発の対極であるが、異なる方法で変化への対応という同じ目的にアプローチしているとも捉えられる。また両者は、変化が大き過ぎてソフトウェア構造が大きく変化する場合には、変化への追随が難しくなるという課題を共有する。

これらの開発形態とアジャイル開発の関係の分析から、アジャイル開発を実現するアーキテクチャ設計においては、データと機能、

時間と空間の二軸において、アプリケーションと骨格構造の間に成立する関係を、依存関係と捉えず、対等な 2 つの構造間の関係として抽出することが有用であることがわかった。これらの関係の実現方法として、アスペクト指向の適用が可能であることを確認した。

② アーキテクチャ構成メカニズムの構築とプロダクトラインアーキテクチャ事例による確認

アーキテクチャ柔軟化のポイントとしてアスペクト指向の適用が可能であることを確認したため、具体的なアーキテクチャ構成メカニズムを構成した。

アーキテクチャ構成メカニズムは、アスペクト指向技術を適用するが、このメカニズムにおいては、従来多く提案されているアスペクト指向言語とは異なり、すべてのアスペクトは対等であり、他のアスペクトに依存して定義することはしない。例えばアスペクト指向プログミラング言語として有名な ApsectJ では、通常のクラス構造に対して、そのクラス定義に依存してクラス間を横断する関心事をアスペクトとして定義する。しかし、本アスペクト指向メカニズムでは、関心事毎にアスペクトに分離し、アスペクトは完全に独立している。このようなアスペクトをアーキテクチャの構成要素する、本アーキテクチャ構成メカニズムを構築した。

本アーキテクチャ構成メカニズムは、アスペクトとアスペクト間の関連付けルールから成る。アスペクトは、複数のクラスから構成され、クラス間の関係の定義は通常のオブジェクト指向と同様のクラス図によって定義される。各クラスは状態遷移を持ち、通常のオブジェクト指向におけるステートマシン図によりモデル化される。このようにしてモデル化されるアスペクトは、各アスペクトで完結しており、他のアスペクトに依存して定義される部分は一切ない。アスペクトを関連付けるルールは、a)あるアスペクトで状態遷移が起こると別のアスペクトにイベントとして伝達するというルールと，b)あるアスペクトにおける状態遷移の条件として別のアスペクトにおけるオブジェクトの状態を参照するというルールの、2 種類がある。このようにしてアーキテクチャを構成することにより、アーキテクチャの各構成要素を完全に独立させることができるため、必要に応じて容易に構成要素を入れ替えることができる。これによって、アーキテクチャの柔軟化が実現できる。

この構成メカニズムを用いることにより、プロダクトラインアーキテクチャの事例を設計し、プロダクトライン開発で求められる製品導出の柔軟性を確保できることを確かめた。

このひとつの例を説明する。これは、自動車の室内ライト制御のプロダクトラインである。このプロダクトラインにおいては，ドアの開閉により室内ライトの点灯消灯を行うことは必須の機能である。また、この室内ライトの制御を行うにあたって、バッテリーセーバーの機能を持つものと持たないものがある。バッテリーセーバー機能を持つものは、バッテリー残量がある値を下回っている場合には、室内ライトが点灯しても一定時間後に自動で消灯する。このバッテリーセーバーに関する可変性は、機能の可変性であると同時に、この機能を実現するためにはハードウェアとしてバッテリーに関するセンサが装備されていなければならず、動作環境としての可変性にもなっている。これらの可変性が、アプリケーション層及びハードウェア制御層に実装される。このプロダクトラインアーキテクチャを本メカニズムを用いて設計すると、図 1 のような構造として実現できる。

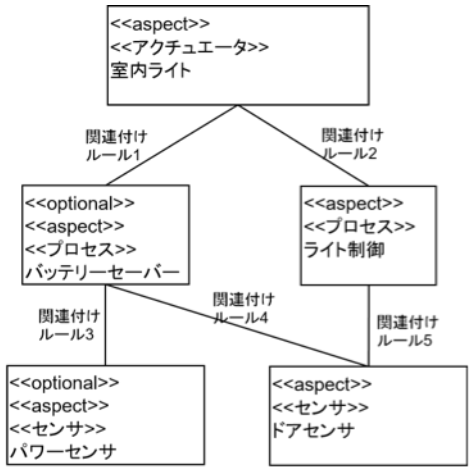

図 1 本メカニズムによるプロダクトラインアーキテクチャ

本事例のようなセンサ、アクチュエータの制御を行うプロダクトラインでは、階層的レイヤによるアーキテクチャが採用されることが多い。これにより、プロダクトラインの様々な可変性をカテゴライズして捉え、システムの構成と対応づけて管理することにより、可変性の実現が容易になるからである。しかし、単純な階層的レイヤでは、各階層間の関係は単純化されるが、上位の階層は下位の階層が提供するインタフェースに依存した構造となる。しかし、上位層と下位層は常に 1：1 に対応させられるとは限らない。アプリケーションが用いるデータを複数のセンサを利用して値を推測したり、同じセンサからのデータでも加工の方法が異ったりするなど、その関係性は一般に複雑である。このような関係性に絡む可変性が出現すると、層内のモジュールの単純な入れ替えでは可変性を実現できなくなり、アーキテクチャ自体の変更が必要になってしまう。

本メカニズムによるアーキテクチャでは、基本的な構造は階層的レイヤと類似となるが、依存関係が全く異なっている、アプリケーシ

ョンが下位のセンサに依存する構造は全く出現していない。従って，仮に別のセンサを利用する際にも各アプリケーション自体の修正は必要ではなく、両者を関連付けるルールの変更によって関連付けを行うことができる。
一般にプロダクトラインアーキテクチャにおいては、可変性に対応して、アーキテクチャ中に可変点とバリアントを定義して、製品系列への適用を図る。製品毎の可変性の違いに対応して、対応するバリアントを自由に組み合わせて製品が実現されることが望ましい。上記の例では、様々なアプリケーションやセンサの選択肢がバリアントとなる。しかしながら、レイヤ構造のようにバリアントがお互いに複雑な依存関係を持つと、そうした自由な組合せが困難となる。
本メカニズムを利用して構成されるアーキテクチャでは、それらの依存関係はルールに集約されるため、最も重要なコア資産であるアプリケーションやセンサそのものは、きわめて独立性高く実現することができる。

③　アーキテクチャ構成メカニズムの動的意味の明確化

本メカニズムにより、アーキテクチャを柔軟化した場合に、全く同じソフトウェアが実現できるかどうかは、本メカニズムが持つ動的意味に依存する。そこで、本メカニズムの動的意味の定義を行った。
動的意味の概要は以下の通りである。
・各アスペクト中のオブジェクトの動作意味は、通常のオブジェクト指向の動作意味に準じており、各オブジェクトはイベントキューを持ち、発生したイベントはそこに蓄えられ、run-to-completion に従って動作する。
・状態遷移が起こったことをイベントとして他のアスペクトに導入するルールが与えられたときは、イベントに対応するアクションを付け加える。実行時には、遷移が発火するとアクションにより起こったイベントがイベントキューに追加される。
・あるアスペクトにおける状態遷移の条件として別のアスペクトにおけるオブジェクトの状態を参照するというルールが与えられたときは、記載された条件を遷移のガードに加える。他にガード条件がある場合にはアンドをとる。
このように、いったんルールが与えられるとイベントがどのアスペクト中で発生したのか、参照するガード条件の状態がどのアスペクト中のものであるのか、ということとは関係なく。アスペクトが関連付けられる。
この動的意味により、UML に基づく従来のオブジェクト指向によるモジュール化と状態遷移による各モジュールの振る舞いの定義と同等のモデルが得られ、本メカニズムによるアーキテクチャは従来のオブジェクト指向の枠組みに自然にマップすることができるので、既存パラダイムのモデルやコードへの変換・生成を自然な方法で実現できる可能性が高い。一方で、複数のオブジェクトが並行動作するとしても、それぞれが全く対等に動作するのか、動作に優先順位があるのかは，規定の範囲外としている。これは UML にならい、意味論の可変点と考えるためである。今後、利用局面に応じて意味論を選択できる仕組みを検討する予定である。

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 5 件）

① 杉田 郁人、野田 夏子、アスペクト指向モデリング手法のためのモデル駆動開発環境の提案、研究報告ソフトウェア工学、査読無、Vol. 2017-SE-195、No. 14、pp.1-8

② Shintaro Hosoai, Natsuko Noda, Tomoji Kishi, System Product Line Engineering for Small Appliances with Driver Derivation, Proc.of 23rd Asia-Pacific Software Engineering Conference, 査読有, 2016, pp.389-392

③ 阿久津 由嗣、野田 夏子、Pepper に体操させるアプリケーション開発のためのライブラリの提案、組込みシステムシンポジウム 2016 論文集、査読有、2016、pp.64-72

④ 野田 夏子、岸 知二、柔軟なプロダクトラインアーキテクチャ設計に関する一考察、研究報告ソフトウェア工学、査読無、Vol. 2016-SE-191、No. 27、pp.1-3

⑤ 野田 夏子、アジャイル時代のアーキテクチャ設計、ウィンターワークショップ 2016・イン・逗子 論文集、査読有、2016、pp.20-21

〔学会発表〕（計 1 件）

① 杉田 郁人、野田 夏子、アスペクト指向モデル駆動開発環境について、第 24 回ソフトウェア工学の基礎ワークショップ、ポスター発表、2016

〔図書〕（計 1 件）

① 岸 知二、野田 夏子、近代科学社、ソフトウェア工学、2017、304

〔産業財産権〕

○出願状況（計　0 件）

○取得状況（計　0 件）

〔その他〕
ホームページ等　なし

６．研究組織
(1)研究代表者
野田　夏子（NODA, Natsuko）
芝浦工業大学・デザイン工学部・准教授
研究者番号：60707701

(2)研究分担者
なし

(3)連携研究者
岸　知二（KISHI, Tomoji）
早稲田大学・理工学術院・教授
研究者番号：30422661

(4)研究協力者
なし