人工知能学会研究会資料
SIG-KST-030-05(2017-03-03)

要求変更による手戻りを考慮した混合アジャイル開発プロジェクトの設計に関する研究

Design of Mixed Agile Project Considering Rework by Requirement Change

満行泰河[1]　後藤拓矢[2*]　稗方和夫[2]　Bryan Moser[2,3]
Taiga Mitsuyuki[1], Takuya Goto[2], Kazuo Hiekata[2], and Bryan Moser[2,3]

[1] 東京大学大学院工学系研究科
[1] Graduate School of Engineering, The University of Tokyo
[2] 東京大学大学院新領域創成科学研究科
[2] Graduate School of Frontier Sciences, The University of Tokyo
[3] マサチューセッツ工科大学
[3] Massachusetts Institute of Technology

Abstract: Many IT venders need to adopt a mixture of waterfall and agile development, which is called mixed agile. In order to manage the mixed agile projects, project managers have to design the agility of the development process considering rework caused by requirement change. This study proposes a design method for mixed agile project. To be concrete, a new model to represent the agility for simulation of mixed agile process is proposed and a simulator which evaluates the mixed agile project based on the agility model is developed. A case study shows a demonstration of project design using the proposed method. The sensitivity analysis in changing the product's structure and the process's agility is conducted and the result shows the usefulness and availability of the proposed method.

1　はじめに

情報システム開発では、要求変更に対応するため導入されてきたアジャイル開発の欠点を補うため、ウォーターフォール開発とアジャイル開発の特徴を合わせた混合アジャイル開発によるプロジェクト設計が求められている。本研究では、要求変更によって発生する手戻りを考慮した、混合アジャイル開発によるプロジェクト設計手法を提案することを目的とする。

アジャイル開発とウォーターフォール開発を組み合わせた混合アジャイル開発の設計は Boehm ら [1] によるサブシステムごとの定性的なリスク評価に基づいて行う研究があるが、開発プロセスとしての混合アジャイル開発がモデル化されていないため具体的なプロジェクト設計が行えない。本研究では、混合アジャイル開発のプロセスにおけるアジリティをモデル化し、既存研究 [2][3] で行われているようなシミュレーション手法を適用することで、要求変更による手戻りを考慮した、品質と開発期間に基づく混合アジャイル開発プロジェクトの設計手法を提案する。なお、本研究では 1 つのイテレーションですべてのサブシステムを対象に設計、実装、テストと進行するプロセスをウォーターフォール開発、サブシステムごとにイテレーションを持ち、設計、実装、テストを行うプロセスをアジャイル開発、1 つのイテレーションで複数のサブシステムを対象とし、複数のイテレーションをもつプロセスを混合アジャイル開発とする。

*連絡先：東京大学大学院新領域創成科学研究科人間環境学専攻
〒 277-8563 千葉県柏市柏の葉 5-1-5 環境棟 274 号室
E-mail: tgoto@s.h.k.u-tokyo.ac.jp

2　提案手法

2.1　提案手法概要

提案手法の概要を図 1 に示す。提案手法ではまず、プロジェクトに固有の設定値を設定パラメータとして、意思決定の対象である混合アジャイル開発のアジリティを意思決定パラメータとして設定する。設定したパラメータに基づいてプロジェクトのモデル化を行い、プロセスシミュレーションの入力とする。プロセスシミュ

レーションでは要求変更による手戻りを考慮して、品質として未検出エラー数と、開発期間およびそのばらつきを出力する。この結果をもとにプロジェクトモデルを選択し、プロジェクトの設計を行う。

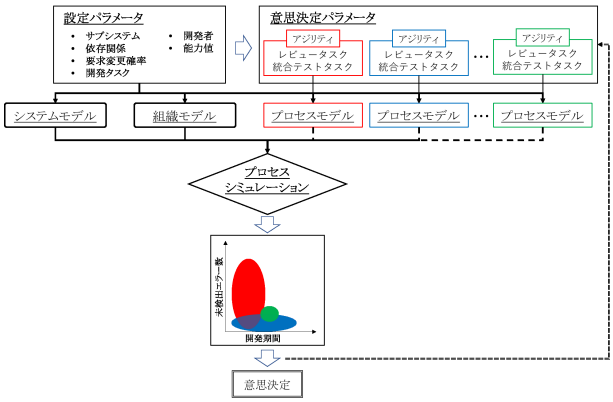

図 1: 提案手法の概要.

2.2 開発プロセスのモデル化

開発プロセスのモデル化では、プロジェクトのモデル化 と開発プロセスにおける手戻りのモデル化を行う。

2.2.1 プロジェクトモデル

プロジェクトモデルは、開発するシステムを表すシステムモデル、開発者の構成と能力を表す組織モデル、開発プロジェクトのプロセスを表すプロセスモデルから構成される。

システムモデルは、開発するシステムを表すモデルで、構成するサブシステムとその依存関係の情報を持つ。それぞれのサブシステムは要求変更確率を属性に持つ。組織モデルは、開発に携わる人員の集合である組織の情報を表すモデルで、人員の構成とその能力値を属性に持つ。具体的には、能力値はタスクの実行能力と、エラーの発生率、エラー検出率に関する値として表現される。プロセスモデルは、開発プロセスにおけるタスクとその依存関係を表すモデルである。プロセスモデルに関して、アジリティのモデル化とタスクのモデル化を行った。

まず、アジリティのモデル化について述べる。一般にアジャイル開発では、開発プロセスのアジリティは、要求変更に柔軟に対応する能力の「flexibility」と、動く製品を迅速に顧客に提供する能力の「leanness」を合わせたものと定義される [4]。提案手法では、「flexibility」はレビューの数に相当し、「leanness」は統合テストのタイミングに相当するとしてモデル化し、レビューが多いほど、統合テストがより前段階にあるほどアジリティが大きいと判断する。また、本研究では要求変更による手戻りを考慮するため、「flexibility」の要素を重視してプロジェクト設計を行う。

次にタスクのモデル化について述べる。タスクは開発プ ロセスを構成する工程を表し、必要な工数、タイプ、対象のサブシステム、先行するサブシステムの集合を属性に持つ。タイプは「要件定義」、「設計」、「実装」、「単体テスト」、「統合テスト」、「レビュー」に分類する。

アジリティとタスクのモデル化により、ウォーターフォール開発、アジャイル開発、混合アジャイル開発の開発プロセスの例として図 2、3、4 のように作成できる。また、レビュータスクの数からウォーターフォール開発、混合アジャイル開発、アジャイル開発の順にアジリティが大きいと判断され、一般的な定性的な特徴に合致していることが分かる。

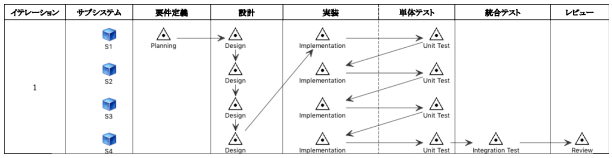

図 2: ウォーターフォール開発プロセスの例.

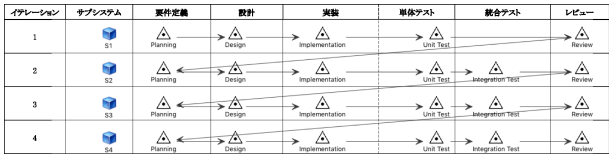

図 3: アジャイル開発プロセスの例.

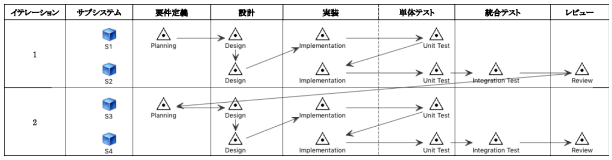

図 4: 混合アジャイル開発プロセスの例.

2.2.2 手戻りモデル

手戻りには、要求変更によって発生する仕様変更に対応するための再開発の手戻りと、検出されたエラーの手直しのためのバグフィックスの手戻りがあり、開発プロセスにおける手戻りはサブシステム間の依存関係によって伝播すると考えられる [5]。提案手法では、再開発の手戻りを開発タスクと統合テストの手戻り工数を実行することで表し、手戻りの伝播は仕様変更の伝播によってモデル化する。バグフィックスの手戻りは、

エラーの検出とバグフィックスを未検出エラー数の推定手法として一般的なソフトウェア信頼度成長モデル (SRGM)[6] を用いてシミュレートし、手戻りの伝播はエラーの伝播によってモデル化する。

2.3 開発プロセスシミュレーション

開発プロセスシミュレーションは、満行ら [2] の研究を参考にした離散イベントシミュレーション (DES) によって行う。また、シミュレーションにおいて実行しているタスクのタイプによって以下の処理を加える。

- 実装：ランダムにエラーを発生させる。
- 単体テスト・統合テスト：SRGM に基づきエラーを減らす。
- レビュー：ランダムに要求変更を発生させる。

3 ケーススタディ

ケーススタディでは、実プロジェクトを想定し、提案手法を用いて混合アジャイル開発プロジェクトの設計を行えることを検証する。モデルにはスマートナビゲーション研究会で提案されている船舶の運航支援システムのための情報共有プラットフォームの開発プロジェクト [7] を参考にした。Master DB はすべての DB を統合する DB で、モジュール独立性を高めることで手戻りを抑える効果が期待できる。本研究ではアジリティの調整する効果だけを考慮するために、システムモデルでは図 5 に示すように Master DB を開発しないモジュール独立性が低いケースと、Master DB を開発するモジュール独立性が高いケースに分けて考える。

それぞれのケースにアジリティが異なる 3 つの開発プロセスを設計し、それぞれの開発プロセスに基づくプロセスモデルをアジリティが小さいものから、モジュール独立性が低い場合には W_{2-a}, W_{2-b}, W_{2-c}、モジュール独立性が高い場合には W'_{2-a}, W'_{2-b}, W'_{2-c} とした。

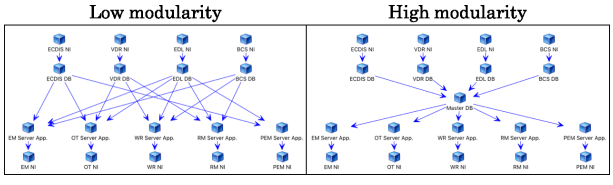

図 5: ケーススタディのサブシステム間の依存関係.

作成したプロジェクトモデルに対して 1000 回ずつシミュレーションを行った。モジュール独立性が低い場合の結果の未検出エラーと開発期間の平均値および標準偏差をまとめたものを図 6 に示す。未検出エラー数、開発期間に関しては W_{2-a}, W_{2-b}, W_{2-c} の順で大きくなっている。また、開発期間に関しては W_{2-b}, W_{2-c} が同程度で、開発期間のばらつきは W_{2-a} より小さくなっている。したがって、未検出エラー数、開発期間の平均値が小さいことを重視するプロジェクトでは W_{2-a} を選択し、開発期間のばらつきが小さいことを重視するプロジェクトでは W_{2-b} を選択すると考えられる。

モジュール独立性が高い場合の結果と未検出エラーと開発期間の平均値および標準偏差をまとめたものを図 7 に示す。未検出エラー数、開発期間に関しては W'_{2-a}, W'_{2-b}, W'_{2-c} の順で大きくなり、開発期間のばらつきに関しては W'_{2-b}, W'_{2-c} が同程度で、W'_{2-a} より小さくなっている。したがって、モジュール独立性が低い場合と同様に、未検出エラー数、開発期間の平均値が小さいことを重視するプロジェクトでは W'_{2-a} を選択し、開発期間のばらつきが小さいことを重視するプロジェクトでは W'_{2-b} を選択すると考えられる。

以上より、提案手法を用いて実際に要求変更による手戻りを考慮した混合アジャイル開発プロジェクトの設計が可能であることが示された。

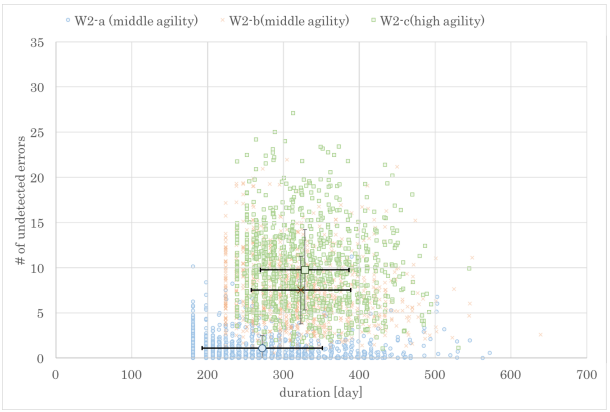

図 6: ケーススタディの結果 (モジュール独立性:低).

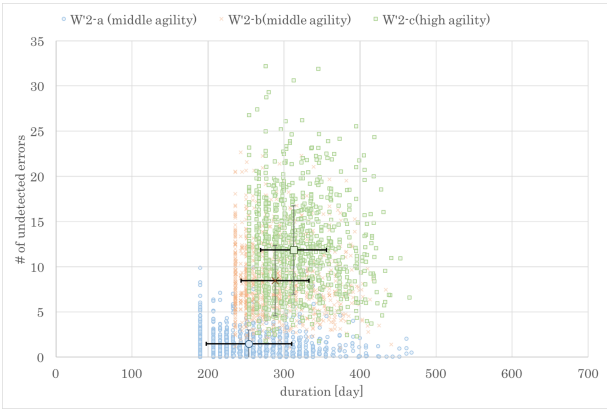

図 7: ケーススタディの結果 (モジュール独立性:高).

4　考察

要求変更確率の変化に対するアジリティの感度解析によってアジリティを調整することの有用性を検証する。また、要求変更による手戻りの影響を抑える方法として、モジュール独立性を高めることと、アジリティを大きくすることでどちらの感度が高いかを検証する。モデルにはケーススタディのモデルにウォーターフォール開発によるプロセスモデルを加えたものを用いる。

結果を図 8、9 に示す。平均未検出エラー数に関しては要求変更確率によらず、アジリティが小さい開発プロセスが適切であるといえる。一方、平均開発期間に関しては要求変更確率が高くなるにつれて、適切な開発プロセスのアジリティも大きくなるといえる。したがって、要求変更確率によって適切なアジリティが異なることが分かる。また、要求変更確率が高い場合には、モジュール独立性によらず、アジリティの感度が高く、アジリティが小さい場合には、少し大きくすることで未検出エラー数をあまり増加させずに開発期間を短縮できるため、モジュール独立性によらず、アジリティを大きくすること効果が高いことが示された。

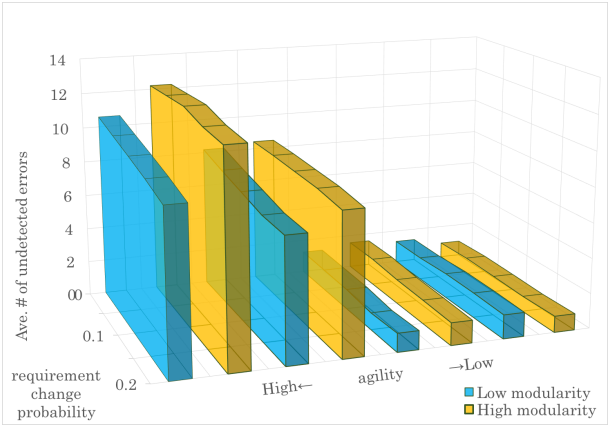

図 8: 要求変更確率の変化に対する未検出エラー数の感度.

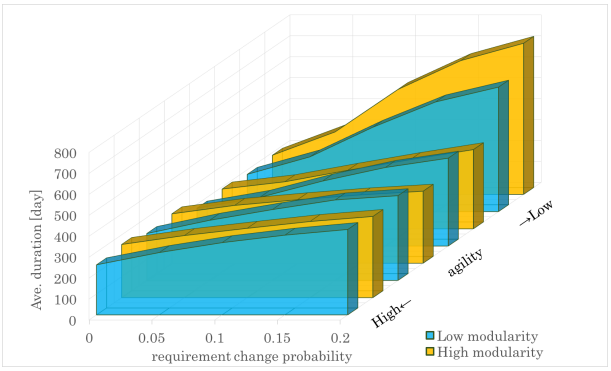

図 9: 要求変更確率の変化に対する開発期間の感度.

5　むすび

要求変更によって発生する手戻りを考慮した、情報システムの混合アジャイル開発プロジェクトの設計手法を提案した。ケーススタディにおいて提案手法が実際のプロジェクト設計に対して適用が可能であることを示した。また、要求変更確率に対するアジリティの感度解析によって、アジリティを調整することが有用であることを示した。

今後は、プロジェクトの管理手法と組み合わせることで、より現場で活用できる動的な混合アジャイル開発のプロジェクト管理に応用することが期待できる。

参考文献

[1] Boehm, B., Turner, R.: Using Risk to Balance Agile and Plan-Driven Methods, *Computer*, Vol.36, No.6, pp.57–66 (2003).

[2] 満行泰河, 稗方和夫, 松原洸也, 大和裕幸, Moser, B.: 船舶建造プロセスシミュレーションを用いた生産設備の導入に関する研究, 日本船舶海洋工学会論文集, Vol.24, pp.293–300 (2017).

[3] Turnu, I., Melis, M., Cau, A., Setzu, A., Concas, G., Mannaro, K.: Modeling and simulation of open source development using an agile practice, *Journal of Systems Architecture*, Vol.52, No.11, pp.610–618 (2006).

[4] Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N. B.: A decade of agile methodologies: Towards explaining agile software development, *Journal of Systems and Software*, Vol.85, No.6, pp.1213–1221 (2012).

[5] Clarkson, P. J., Simons, C., Eckert, C.: Predicting change propagation in complex design, *Journal of Mechanical Design*, Vol.126, No.5, pp.788–797 (2004).

[6] Yamada, S., Ohba, M., Osaki, S.: S-Shaped Reliability Growth Modeling for Software Error Detection, *IEEE Transactions on reliability*, Vol.32, No.5, pp.475–484 (1983).

[7] 諸野普: 日本舶用工業会スマートナビゲーションシステム研究会の活動概要-安全で効率的な運航をサポートするための実海域データの情報共有プラットフォーム開発, マリンエンジニアリング, Vol.49, No.5, pp.617–621 (2014).