1309

自発的なソフトウェア進化を促すプロジェクト要因の考察

Investigating Project Factors Boosting Spontaneous Software Evolution

○中村 匡秀[*1*3]，松本 健一[*2]
Masahide NAKAMURA[*1]，Kenichi MATSUMOTO
[*1] 神戸大学 Kobe University
[*2] 奈良先端科学技術大学院大学 Nara Institute of Science and Technology
[*3] 理化学研究所・革新知能統合研究センター RIKEN AIP

This paper addresses an issue of spontaneous software evolution, which is a modern form of software evolution observed in social coding platforms such as GitHub. In the spontaneous software evolution, feature additions and modifications of software are achieved by spontaneous proposals from individual developers, not by the request from the project manager. In this paper, we investigate project factors that can boost such spontaneous software evolution. Specifically, we first introduce a governance framework of the spontaneous evolution, inspired by a smart city execution model, and then consider relevant factors that can motivate developers to propose actions. Finally, we discuss necessary features for the governance framework from viewpoints of (1) motivation of developers, (2) individual sense of value, (3) skill and competency, and (4) characteristics, quantity, and deadline.

Key Words : software evolution, project management, motivation, governance framework, team building, smart city

1. 緒　　言

ソフトウェア進化とは，一旦出荷されたソフトウェアに対する変更を受け入れる仕組みや活動を指す(1)．近年のネットワークの発達・普及によって，ソフトウェアの配布・流通が容易化されたことにより，OS やアプリをはじめ，様々なソフトウェアは出荷後も常時アップデートされることはもはや珍しくない．ソフトウェア進化の概念により，「ソフトウェアは，新たなユーザの要求や利用環境の変化に応じて，機能追加や不具合修正が繰り返されるものである」と考えられるようになった．

これまで多くのソフトウェア開発では，プロジェクト・マネージャ(PM)を責任者とする階層型組織に基づく開発手法が採られている．すなわち，PM が成果を定義し，仕事を設計し，プロジェクトを管理する役割を果たし，開発者は PM の指示に従って開発するという，いわば上意下達によって，ソフトウェアの開発が進められている．

その一方で，近年ソーシャルコーディングという新しいソフトウェア開発方式が注目を集めている(2)．ソーシャルコーディングは，多数の開発者が協調し，集合知によってソフトウェアを開発する手法である．その代表的なプラットフォーム GitHub(4)では，ブランチ機能により，開発者は，他の開発者の作業に影響を与えることなく，機能追加や不具合修正のためのソースコード変更を独自のアイデアや判断で行うことができる．更に，プルリクエスト機能により，その変更内容を当該ソースコードに反映することを，ソースコードの管理者（プロダクトオーナー）に提案することができる．プルリクエストを受け取ったプロダクトオーナーは変更内容をレビューし，ソースコードに反映するかどうかその採否を判断する．

ソーシャルコーディングは，オープンソースソフトウェア開発ですでに広く普及しており，一部の先進的な企業でも採用され始めている．従来のソフトウェア開発との大きな違いは，ソフトウェアの変更，すなわち，ソフトウェア進化のアイデアが，上意下達ではなく開発者自身から自発的に提案されることである．このような新たなソフトウェア進化の形態を，我々は「自発的ソフトウェア進化」と呼ぶ．

自発的ソフトウェア進化は，開発者自身のアイデアや解決法に基づいてソフトウェアが変更されるため，プロダクトオーナーが想像もしなかった斬新な機能が提案されたり，把握しきれていなかった不具合の修正が行われたりする．そのため，従来の上意下達のやり方に比べて，より創造的で価値の高いプロダクトが生まれる可能性が高い．実際，オープンソースソフトウェアの中には，Linux，Apache，MySQL 等，自発的進化とみなせるバージョンアップを繰り返し，デファクトスタンダードとなっているものがある．

しかしながら，自発的ソフトウェア進化をどのように広く，安定的・持続的に進めていくかは自明ではない．より具体的には，自発的進化をどのように統制（ガバナンス）するのか，開発者の協調によって生まれる新たな作業（ソーシャル・オーバーヘッド）をいかに低減するのか，コミュニティの自発性をどのように維持していくのか（持続可能性）といった技術的課題がある．GitHub をはじめとするソーシャルコーディングプラットフォームでは，現状これらの課題の解決はプロダクトオーナーの裁量・手腕に任されている．実際，GitHub のプロジェクトのうち，着実に進化が進んでいるプロジェクトと誰もメンテナンスしなくなって廃れてしまうプロジェクトを比べると，後者の方が圧倒的に多い．

自発的ソフトウェア進化をより工学的に実施するために，我々は科学研究費・基盤研究(A)「自発的ソフトウェア進化の加速に向けた基礎技術の開発」に取り組んでいる[5]．本稿では特に，自発的進化の統制（ガバナンス）の観点から，自発的進化を促すために本質的に必要なプロジェクト要因は何なのかを分析することを試みる．また，それらの要因を用いてプロジェクトの状態を定義し，自発的ソフトウェアが起こるためのメカニズムを考察する．

2. 自発的ソフトウェア進化のガバナンスフレームワーク

自発的ソフトウェア進化は，開発者個人のアイデアによってトリガされる．そこでは，従来の上意下達の指揮系統が薄れ，全ての開発者が平等に自身の創造的なアイデアを提案することができる．このことから，自発的ソフトウェア進化を前提とするプロジェクトの体制は，企業に見られる階層的な組織構造ではなく，都市や街に見られるフラットなコミュニティ構造を想定する方が自然である．

そこで我々は，スマートシティの持続的な進化モデル[6]を，自発的ソフトウェア進化の統制に利用できないか検討を進めている．Figure 1 にその概要を示す．スマートシティでは，まず都市のデータを収集して都市の現在の状態を観測する（(1) State Observation）．次にデータを分析し，現在の状態がその都市の「あるべき姿」にどれだけ遠い／近いのかを理解する（(2) State Interpretation）．続いて，現状を「あるべき姿」に近づけるための方策・アクションを提案し（(3) Action Proposal），最後にアクションを選択して実行する（(4) Action Execution）．(1)から(4)のサイクルを，IoT や ICT 等の最新技術を駆使して繰り返し，都市を漸進的に理想状態に近づけていく．ここで，本モデルは抽象的なスマートシティの実行モデルのひな形を定めているにすぎず，「あるべき姿」や実行可能なアクションは，それぞれの都市の課題に基づいて具体化される．

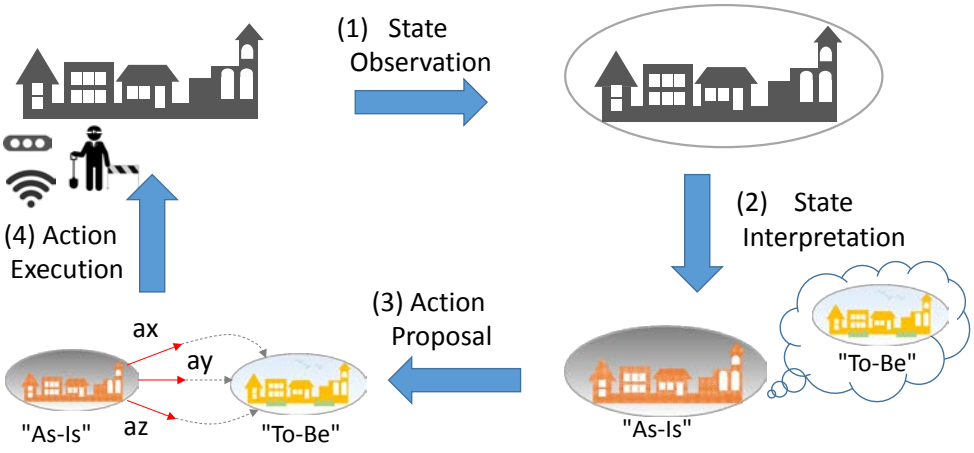

Figure 1: Smart City Service Execution Model [5]

このスマートシティ実行モデルを,自発的ソフトウェア進化を前提とするソフトウェアプロジェクトの統制（ガバナンス）に当てはめることを考える．具体的には，以下のような対応付けを行う．

【都市】プロジェクトに対応付ける

【市長（ガバナー)】 プロダクトオーナーに対応付ける

【市民】 プロジェクトの開発メンバに対応付ける

【都市のあるべき姿】 プロジェクトの成果物が要求を満たしている状態とする．プロダクトオーナーによって定義される

【状態観測】 プロジェクトから必要なデータを計測・取得する

【状態理解】 データで示される現状態が「あるべき姿」とどれだけ離れているかを開発者が理解する

【行動提案】 開発者がコードを変更し，プルリクエストを行う

【行動実行】 プロダクトオーナーがプルリクエストをレビューし，ふさわしい変更をコミットする

対応付けによって得られたモデルを，「自発的ソフトウェア進化のためのガバナンスフレームワーク」と呼ぶ．フレームワークと呼ぶ理由は，上記のモデルが実行の流れを規定しているだけで，具体的なプロジェクトに適用する際には，各プロセスの実行の方法を具体的に決めなければならないからである．

ガバナンスフレームワークに基づいて自発的ソフトウェア進化のサイクルを回していくためには,まず適用するプロジェクトの種類を決定する必要がある．本稿では特に，企業でのチーム・ソフトウェア開発を前提としたうえで，ガバナンスフレームワークにおける【行動提案】を促す要因に焦点を絞って考察する．

3. 自発的ソフトウェア進化を促す要因の考察

3・1 開発者への動機付け

自発的ソフトウェア進化は，開発者の自発性を原動力とする．そのため，開発者に強烈なモチベーションを持たせる仕組みが必要である．行動提案において，各々の開発者はプロジェクトの「あるべき姿」に向けて，自らのアイデアをコードに起こし，プルリクエストを行う必要がある．また，提案したコードの変更は，必ずしも採用されるとは限らず，提案が却下された場合にはショックが大きいであろう．このような状況で，開発者自らが進化を提案するには，相当なモチベーションが必要である．

一般的に人を動機づける方法として，外的モチベーションと内的モチベーションが存在する．外的モチベーションとは，外部からその人に働きかけることで動機付けを行うことである．例えば，報酬やボーナス，評価，懲罰，強制など，いわゆるアメとムチに相当する．外的モチベーションは，汎用性が高くその効果も強力であるメリットがあるが，その人の自主性を引き出せないというデメリットがある．内的モチベーションは，その人自身の内面的な要因から動機づけられることである．興味や関心から湧き出す意欲，満足感，達成感，充実感をえたいという欲望である．内的モチベーションは持続性が高く自己成長がつながるというメリットがあるが，モチベーションが発揮されるかどうかは個人頼みであるというデメリットがある．

自発的ソフトウェア進化に対して，どのような動機付けの方法が有効であるかについては，過去の様々な知見が参考になる．Daniel Pink は著書[7]の中で，「モダンな経営システムは，より内的な動機付けに基づく必要がある」と述べている．その根拠の一つとして，彼は，機能的固着を測る認知能力テストである「ロウソク問題」において，金銭的なインセンティブが問題解決に負に働いた実証実験[8]を取り上げている．さらに，内的モチベーションの重要な源泉は，次の 3 つの要素であると説いている：

(1) Autonomy(自主性) ：自分のやるべきことは自分で決めたいという欲求

(2) Mastery(成長)：何か大切なことについて上達したいという要望

(3) Purpose(目的)：自分自身よりも大きな何かのためにやりたいという切望

また，Francino はソフトウェア技術者を動機づける 13 の方法を公開している[9]：(1) 聴いてあげる，(2) 新しい技術，(3)トレーニング，(4)認知してあげる，(5)成長の機会，(6)イノベーションの機会，(7)フレックス・タイム，(8)リモートワーク，(9)決定権，(10)単純化と合理化，(11)才能で囲む，(12) ゲーミフィケーション，(13)リスペクト．

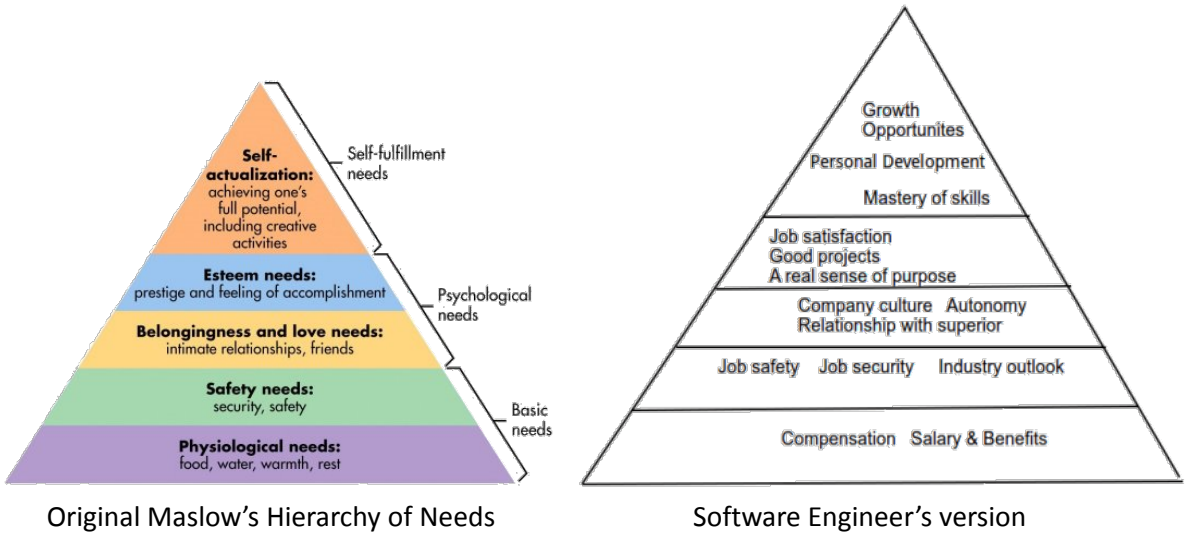

Figure 2: Software Engineer's Hierarchy of Needs [9]

さらに，あるソフトウェア開発チームのブログ[10]では，ソフトウェア技術者の欲求を「マズローの階層的欲求」に当てはめて図示したものを掲載している（Figure 2 参照）．

興味深いことに，これらの知見において共通しているのは，ソフトウェア技術者が，報酬や評価といった外的モチベーションよりも，自己の成長や仕事の達成感等の内的モチベーションを大事にしていることである（もちろん，最低限の給与は保障されているという前提であるが）．よって，ガバナンスフレームワークでは，開発者の内的モチベーションを維持・向上する仕組みをいかに実現するかがカギとなる．

3・2 開発者個人の性格・価値観

内的モチベーションは，開発者個人の性格や価値観に大きく左右される．したがって，自発的ソフトウェア進化を前提とするプロジェクトに対して，個人が大事にしているモチベーションの源泉を明らかにし，プロジェクトで共有することは重要である．個人ごとに異なるモチベーションをチームで共有するプラクティスとして，Management 3.0[11]の Moving Motivators が知られている．

Figure 3 に Moving Motivators で用いられるカードを示す．これらのカードには以下の 10 種類のモチベーションの源泉が書かれている：

- ACCEPTANCE(受容)　「私の周りの人が、私がやっていることや私らしさを認めてくれる」
- CURIOUSITY(好奇心)　「調査したり、考えたりしたいことがたくさんある」
- FREEDOM(自由)　「私は自分自身の仕事と責任で他人から独立している」
- STATUS(地位)　「私の職位は、優れていて、いっしょに働いている人たちから認められている」
- GOAL(ゴール)　「私の人生の目的が、私がしている仕事に反映されている」
- HONOR(名誉)　「私個人の価値が私の働き方に反映されていることを誇りに思う」
- MASTERY(熟達)　「私の仕事は難しいが、まだ能力の範囲内にある」
- ORDER(秩序)　「安定した環境のための十分な規則と方針がある」
- POWER(権力)　「私の周りに起きることを自分で左右できる裁量がある」
- RELATEDNESS(関係性)　「私は仕事でかかわる人々と良好な社会的つながりを持っている」

Moving Motivators では，チームでテーマを決め，個人が特に大事にしているモチベーションのカードを選び，モチベーションの紹介とその理由を説明する．これによって，各メンバがどのようなモチベーションのもとに仕事に取り組むかをチームで共有できる．こうして得られた個人のモチベーションに対する価値観は，ガバナンスフレームワークにおけるタスクの割り振りをサポートする情報として活用できる．

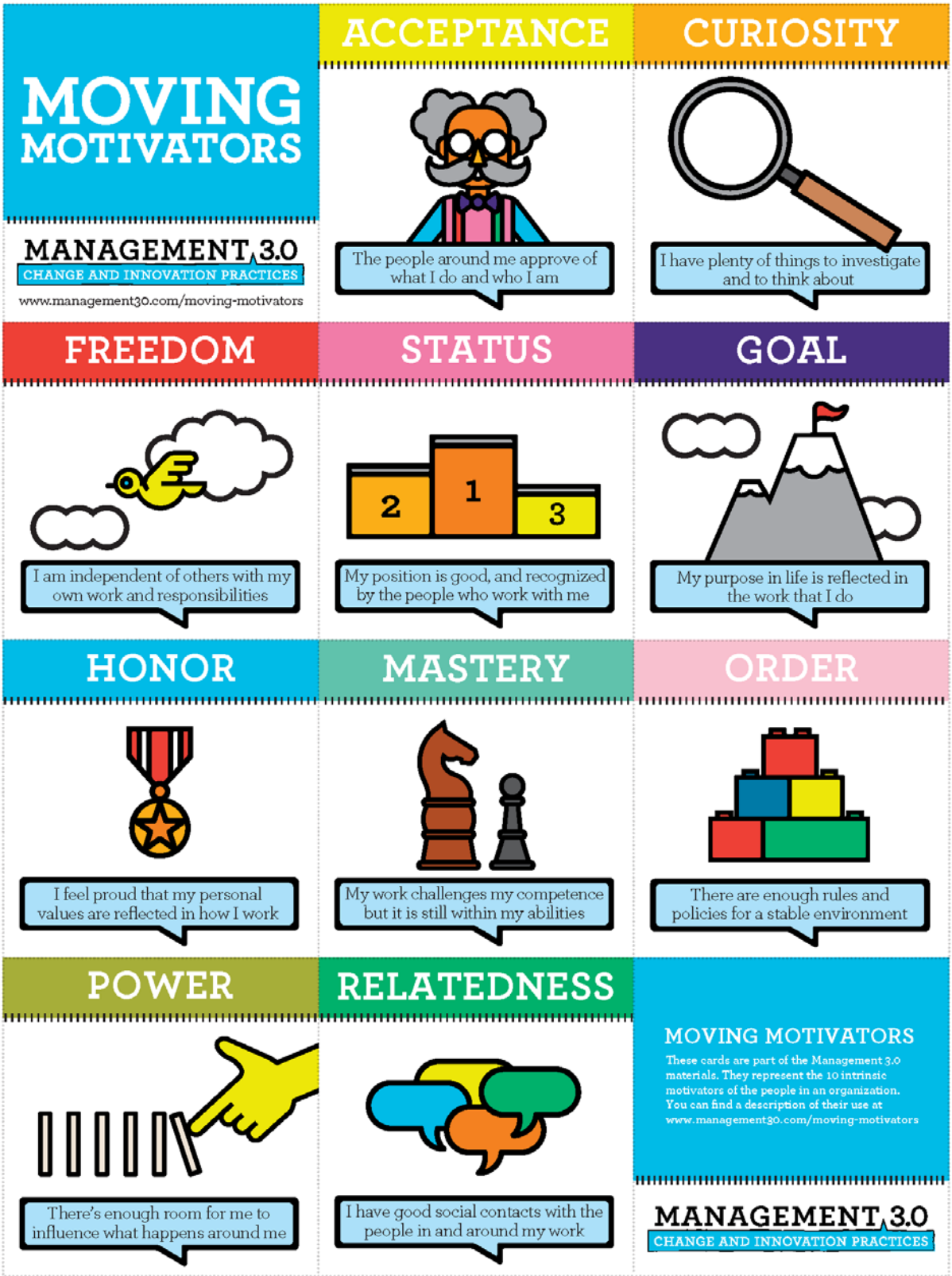

Figure 3: Moving Motivators Cards
(Downloaded from https://management30.com/practice/moving-motivators/)

3・3 開発者個人の能力・スキル

ソフトウェア進化に必要なタスクが自分の能力やスキルを大きく超えていた場合，開発者は自発的な行動提案を控えてしまう可能性がある．その結果，できる人はどんどんタスクをこなすが，できない人は何もできないという状況が生まれる．能力・スキルの向上には教育や自己学習の機会が必要である．ガバナンスフレームワークでは，教育や自己学習の機会もソフトウェア進化のためのタスクと認めることで，内的モチベーションに寄与するのではないだろうか．

また，逆にタスクが簡単すぎる・単調である場合にも，モチベーションが下がることが懸念される．「こんな単純なことは自分がやる仕事ではない」とやらない状況が想像できる．人間だれしも，自分の能力よりも少し高い挑戦的なタスクにやりがいを感じるところがある．自分の能力・スキルに合ったタスクを，開発者がうまく見つけられるようにサポートする仕組みが必要である．

3・4 タスクの性質・量・締切り

達成すべきソフトウェア進化に必要なタスクの性質（難しさ），量，締切りまでの時間も，開発者個人のモチベーションに影響を与える．短期間に難しい仕事が多数集中する場合には，メンバ間でタスクの割り振りが偏るであろうし，逆に暇すぎても全員のモチベーションが上がらないだろう．ガバナンスフレームワークには，チームの開発能力・キャパシティに応じたタスク消化をサポートする機能が求められる．

4. 結　　語

本稿では，GitHub 等のソーシャルコーディング・プラットフォームに見られる新しいソフトウェア進化の形態「自発的ソフトウェア進化」を取り上げ，自発的進化を促すプロジェクトの要因について考察した．具体的には，スマートシティの実行モデルを採り入れた自発的進化のガバナンスフレームワークを述べ，行動提案に寄与する主要な要因について述べた．開発者への動機付け，開発者個人の性格・価値観，開発者個人の能力・スキル，タスクの性質・量・〆切の観点から，ガバナンスフレームワークに必要な機能・仕組みについて考察した．

今後の課題としては，考察した機能の具体的な設計と実装，行動提案以外のプロセスへのアプローチが挙げられる．

謝　　辞

この研究の一部は，科学技術研究費（基盤研究 A 17H00731，基盤研究 B 16H02908，18H03242，18H03342），および，立石科学技術振興財団の研究助成を受けて行われている．

文　　献

(1) 大森 隆行, 丸山 勝久, 林 晋平, 沢田 篤史, “ソフトウェア進化研究の分類と動向”, コンピュータソフトウェア, Vol.29, No.3 (2012), pp. 3_3-3_28.

(2) Hans van Vliet, "Software Engineering: Principles and Practice," *Wiley*, ISBN 978-0470031469 (2013).

(3) F. Thung, T. F. Bissyandé, D. Lo and L. Jiang, "Network Structure of Social Coding in GitHub," *2013 17th European Conference on Software Maintenance and Reengineering*, Genova (2013), pp. 323-326.

(4) GitHub, http://github.com

(5) 松本 健一，門田暁人，中村匡秀，森崎修司，角田雅照，大平雅雄，玉田春昭，戸田航史，伊原彰紀，畑秀明，“自発的ソフトウェア進化の加速に向けた基礎技術の開発”，文部科学省・科学研究費・基盤研究(A), 課題番号 17H00731 (2017).

(6) M. Nakamura and L. D. Bousquet, "Constructing Execution and Life-Cycle Models for Smart City Services with Self-Aware IoT," *2015 IEEE International Conference on Autonomic Computing*, Grenoble (2015), pp. 289-294.

(7) D. H. Pink “Drive: The Surprising Truth about What Motivates Us”, Riverhead Books, ISBN-978-1594488849 (2009).

(8) Glucksberg, S. “The influence of strength of drive on functional fixedness and perceptual recognition”, *Journal of Experimental Psychology,* Vol. 63, No. 1 (1962), pp.36-41.
(9) Y. Francino, “13 ways to motivate software engineers”, https://techbeacon.com/13-ways-motivate-software-engineers
(10) J. Fung, “How to retain your software engineers?”, https://www.recruityourninja.com/retain-software-engineers/
(11) Jurgen Appelo, “Management 3.0 – Leading Agile Developers, Developing Agile Leaders”, Addison-Wesley Professional, ISBN-978-0321712479 (2010).