Vol.2012-SE-177 No.4
2012/7/19

ET ロボットコンテストを題材とした プロセスが不明瞭な開発におけるパターンマイニングの提案

角谷　将司[†]　中野　聡之[†]　小澤 貴之[†]　和田 雅彦[†]　伊藤 弘毅[†]
鷲崎　弘宜[†]　深澤　良彰[†]

パターン・ランゲージは，プロセスのグラスボックス化を目的としている．しかし、従来のパターンのマイニング手法は，プロセスが明確であることを前提としているので，マイニングされたパターン・ランゲージが正当であるといえる．したがって，開発者自身も開発プロセスの全容を把握していないプロセスが不明瞭な開発においては，目的となるプロセス自体がそもそも不明瞭なので，抽出されたパターン・ランゲージが正当であるか判断できない．我々はこれらの問題を解決するために，開発プロセスとパターン・ランゲージを同時に抽出する新たなパターンマイニングの手法を提案する．我々は実際に新たなマイニング手法を用いて，開発プロセスが不明瞭になりがちな，本業の合間に取り組むコンテスト形式の「ET ロボットコンテスト」のパターン・ランゲージの抽出を行った．

A proposal of pattern mining in development with an ambiguity process used ET robot contest.

MASASHI KADOYA[†]　TOSHIYUKI NAKANO[†]　TAKANORI OZAWA[†]
MASAHIKO WADA[†]　HIROKI ITOH[†]　HIRONORI WASHIZAKI[†]
YOSIAKI FUKAZAWA[†]

Pattern Language aimed at to clear the process. However, it can be said that it is correct since the mining technique of the conventional pattern is premised on a process being clear. Therefore, we cannot judge whether the extracted pattern language is correct when the development process which development is ambiguity and not understood by developer because the process itself is not clear. We propose the technique of new pattern mining which extracts a development process and a pattern language simultaneously, in order to solve these problems. We extracted the pattern language of "ET robot contest" which tends to become ambiguous because that development of the contest is in the intervals of vocational.

1. はじめに

ソフトウェア開発におけるパターンとは，先人的に培われてきた「知の集合体」を記述した言語である．ここでいう「知の集合体」とは，一定の状況下における問題とその解決策のことを示す．つまり，ソフトウェア開発におけるパターンとは，どのような「状況」でどのような「問題」が発生し，それをどう「解決」すべきかを記述したものを言う[1]．これらパターンを用いることで、先人的な知識を伝授するだけでなく，組織やチーム内の自発的なコミュニケーションを促すことができる．

このような問題発見と問題解決の先代的な知恵を上記のように記述することを考案したのは，クリストファー・アレグザンダーという一人の建築家である．アレグザンダーは，住人が心地よいと感じる家や町をデザインするプロセスに参加するにはどうしたら良いか考え，その支援方法として編み出されたのがパターン・ランゲージである．現在ではソフトウェア開発だけでなく，学び・教育・変革行動など様々な分野に適用されている[2]．

また，パターンは独立するものではなく，大概他のパターンと繋がりを持つ．例えば，パターンの記述の中に他のパターンを参照する機会が頻繁に存在する．このような記述はパターン間の関係を強くさせ，プロセスを明確化させる．特に類似する関係や利用する関係にあるパターンは重要となり，その関係性を意識して記述されたパターンの集合体をパターン・ランゲージと呼ぶ．

パターンの運用にまつわる活動は，パターンの利用活動と抽出活動の２つに分けられる．前述したものは，利用活動に関するパターンの活動である．本研究では，特にパターンの抽出活動を構成するパターンマイニングに関して，効率の良いマイニング手法を考案する．

パターンの抽出活動は，ソフトウェアに関する知識から，パターンとして再利用可能な知識を特定の形式にしたがって記述し，集団的レビューによって洗練する活動である．パターンの抽出活動は一般的に４つのプロセスを順に実行することで達成される．それは，発見→記述→提出→洗練といった過程である．これらのプロセスを合わせてパターンマイニングと呼ぶ[3]．

代表的なものとして，ワークショップ形式，インタビュー形式，

† 早稲田大学
Waseda University
早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室

教育・講義形式，セルフマイニング形式などが挙げられる．いずれの手法も有用であり，幅広く採用されている．

しかし，これらの代表的なマイニング手法はいずれも開発プロセスが明確であることを前提としている．したがって，特に開発者自身も開発プロセスの全容を把握していない「不明瞭な開発」においては，適切なマイニングが行えないと考えられる．我々はこれらの問題を解決するために，開発プロセスを明確にしながらパターン・ランゲージを抽出する新たなパターンマイニングの手法を提案する．

2. 不明瞭な開発におけるパターンマイニング

2.1 従来のパターンマイニング

パターン・ランゲージはパターン間の繋がりを持たせることで，プロセスをグラスボックス化による再利用性の向上を目的としている[4]． よって，従来のパターンマイニングは開発プロセスが明確であるが故に，適切なパターン・ランゲージが抽出されたか判断できる．しかし，マイニング者や開発者自身も開発プロセスの全容を把握していない「不明瞭な開発」におけるパターンマイニングは，元来の明確化させるべき対象がそもそも不明瞭である．そのため，抽出されたパターン・ランゲージ間の依存関係の妥当性を判断できない．したがって，開発者自身もプロセスの全容を把握していない開発におけるパターン・ランゲージの抽出は，従来のパターンマイニングでは適切なマイニングが困難であると考えられる．

代表的なパターンマイニングの手法として，ワークショップやインタビュー，講義形式，セルフマイニングなどが挙げられる．以下にこれらマイニング手法の詳細を示す．

(1) **ワークショップ形式**
特定の形式に従って，共通のプロダクト・開発経験から集団によってマイニングする手法[5].

(2) **インタビュー形式**
パターンマイニングの経験を有するパターン技術者が，経験を有する一般技術者にインタビューすることでマイニングする手法.

(3) **教育・講義形式**
パターン技術者が一般技術者に対して，パターンマイニング手法を教育しながら，パターンを抽出するパターンマイニング手法.

(4) **セルフマイニング形式**
一般技術者に何らかの形で動機づけさせ，技術者自らが過去の経験を振り返ることによってパターンを抽出するマイニング手法.

これらマイニング手法において，特に開発プロセスが不明瞭な開発におけるマイニング手法として適切でないものは，講義形式とセルフマイニングであると考えられる．講義形式は，パターン技術者が技術者にパターン技術を教育すると共にパターンを抽出するマイニング手法である．しかし，教育者自身がプロセスを理解していない状況における教育は困難である．また，セルフマイニングは，名の通り技術者自身がマイニングを行う手法である．この手法は非常に属人性が強いため，特にプロセスを主とするパターン・ランゲージには適さない．一方，ワークショップは集団でマイニングを行うため，意見の多面性があり，プロセスを明快にする上では適当と考えられる．しかし，集団によるマイニングで得られたパターンは具体性に欠ける一面も持つ．したがって，パターン技術者が特に抽象度の高いと思われる事項を技術者にインタビューすることで，適切なマイニングを行える．

これら代表的なマイニング手法は前述した通り，いずれも「パターン・ランゲージ」の抽出は困難である．しかし，不明瞭な開発においても，ワークショップ形式やインタビュー形式は「パターン」を抽出することは可能であると考えられる．

2.2 プロセスが不明瞭な開発

開発プロセスが不明瞭になる傾向を持つ開発の例として，本開発前の提案書作成段階におけるプロトタイピングモデルの開発や，本業の合間に取り組むコンテスト形式の開発などが挙げられる．よって，組み込みシステム開発分野および同教育分野における若年層や初級エンジニアへの分析・設計モデリングの教育を目的としている「ET ロボットコンテスト」は，開発プロセスが不明瞭になる可能性が高いと考えらえる．我々は不明瞭な開発の例として「ET ロボットコンテスト」に着眼し，実際に新たなマイニング手法を用いてパターン・ランゲージを抽出することで，問題に対する解決を示す．また，開発プロセスが明確である場合のパターンの関係性を図 1 に，不明瞭な開発プロセスにおけるパターンの関係性を図 2 に示す．

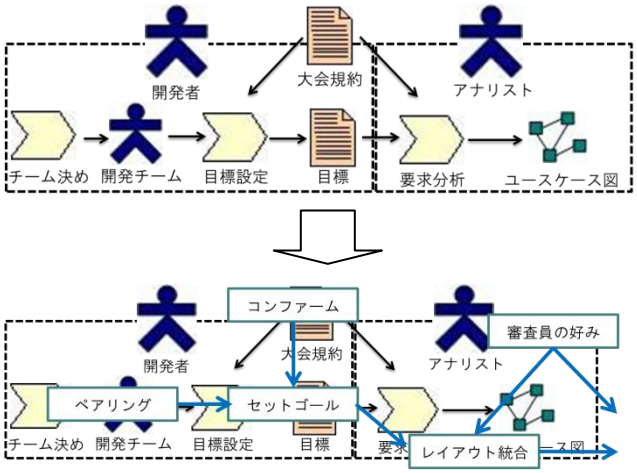
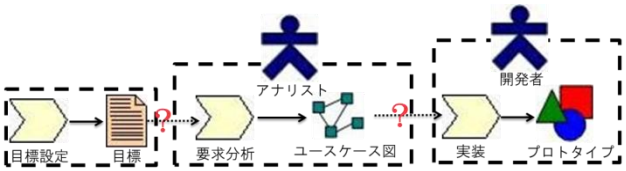


図 1 開発プロセスが明確である場合のパターンの関係性



早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室

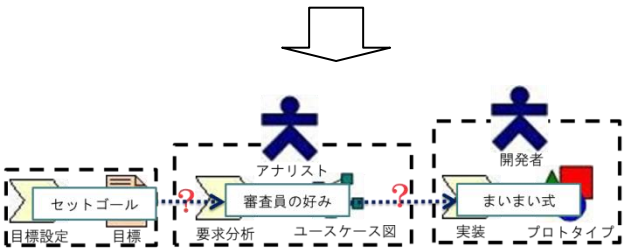


図 2　不明瞭な開発プロセスにおけるパターンの関係性

3.　開発プロセスを明確にするマイニング手法

本節では，開発プロセスとパターン・ランゲージを同時に抽出する新たなパターンマイニングの手法を提案する．3.1 節では本手法の特徴を，3.2 節では具体的なマイニング方法を記す．

3.1　新たなマイニング手法の特徴

本手法は，大きく二つの特徴を持つ．一つは，パターン・ランゲージの抽出と共に，プロトタイプとしての開発プロセスを同時に抽出する過程を導入することである．そして，抽出された開発プロセスは，再びワークショップやインタビューによって，より洗練された開発プロセスへと成長する．よって，パターン間の依存関係も明確になり，適切なパターン・ランゲージが抽出されると考えられる．また，この抽出の前段階として，抽出されたパターンを時系列，プロセス・設計・実装のドメインごとに配置する．配置されたパターンによって，開発プロセスを視覚的に予測することができ，プロトタイプとしての開発プロセスを抽出できる．その際，パターン間の関係性を持たせることでパターン・ランゲージの抽出を行い，パターン・ランゲージと開発プロセスの同時抽出を行うことが可能となる．

二つ目の特徴として，これらパターン・ランゲージと開発プロセスを同時に抽出する過程を繰り返すことである．このサイクルを繰り返すことによって，両者は相互に深く関係し，洗練されたパターン・ランゲージの抽出が期待できる．

3.2　新たなマイニング手法のガイドライン

本節では，以上の 2 つの特徴を踏まえ，具体的なマイニング方法について記す．まず，マイニングを行う前に事前情報を収集する．これはマイニング中にある程度の“あたり”をつけることを目的としている．そして，実際にワークショップとインタビューを行うことでパターンの抽出を行う．これらのマイニングによって抽出されたパターン・ランゲージは，誰でもレビューをできる環境にするため Wiki によって編集・構築を行う．次に，抽出されたパターンから開発プロセスを視覚的に把握するため，Wiki によって集められたパターン・ランゲージを時系列・ドメイン別に分け，さらにマトリクス化する．この手法により，視覚的に開発プロセスを予測でき，プロトタイプの開発プロセスを抽出できる．そして，抽出された開発プロセスとパターン・ランゲージは，再びワークショップやインタビューによって洗練される．

これら一連の詳細を以下の節で説明する．また，解決の全体像を図 3 に示す．

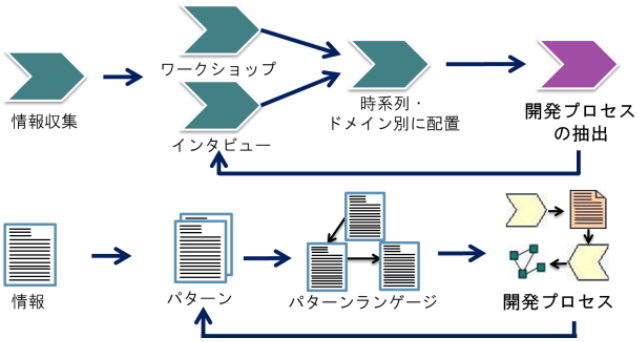


図 3　解決の全体像

図 3 における上図はプロセスアクティビティを示し，下図は実際に抽出される成果物のプロセスモデルである．

3.2.1　情報収集

パターンを抽出する際，ワークショップやインタビューによって得られた知見以外にも，事前情報としての知識を付ける必要がある．例えば，一つのプロセスの過程において抽出されたリソース（モデル，ログ，プロダクト）を知ることにより，手当たり次第なマイニングではなく，ある程度の抽出すべきパターンの“あたり”を付けることが可能となる．また，抽出する対象にある背景や関連する知識をマイニングする初期の段階から得ることによって，収集した情報を“道具”として利用することができる．

3.2.2　ワークショップ

本手法におけるワークショップは，大きく二つの機能を持つ．それは，パターン抽出としての機能とパターン洗練としての機能である．

初期の段階におけるワークショップでは，主にパターン抽出としての機能を持つ．まず，開発プロジェクトに関わった開発者にプロジェクトの成功した事例と失敗した事例（アンチパターン）を挙げるように促す．

成功例
・実装した箇所が期待通り機能した．
・走行速度を早く実装できた．
・レプリカのコースで練習できた．
・モデルで走行戦略を評価された．
・研究成果を盛り込めた．
・モデルファーストができた．
・RedMine を上手く活用できた．

早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室

・要求分析がうまくできた.
・イテレーターごとの Output が出せた.
・モデルに沿って書けた.

表 1　成功例の抽出結果

失敗例
・自作シーソーが劣化した.
・坂道完成が遅かった.
・ソースコードのマージが大変だった.
・モデル提出後にソースコード実装を行ったため，リスクがあった.
・C++との制約によるトレーサビリティが取れない問題があった.
・ジャイロの値が正確に読み込めないケースが存在した.
・最後はつめこみの開発になってしまった.
・ロボットが三台あるとよかった.
・ソナーセンサーが動作しない.

表 2　失敗例の抽出結果

これらの失敗・成功の背景や解決策を議論することで，パターンの抽出を行う．そして，得られたパターンをマイニング者は，再びワークショップで公表し，再び議論の対象とする．これがパターン洗練としての機能である．

3.2.3　インタビュー

本手法におけるインタビューは，主にワークショップで抽出された問題と解決を具体化するために用いる．特に設計部分に関しては，そのプロセスに関係した技術者のみが把握していることが多々ある．したがって，それら潜在的知識を抽出するために，インタビューは行われる．

3.2.4　Wiki 開発

パターン・ランゲージを抽出する過程において，Wiki を利用したマイニングを実施する．もともと Wiki の誕生はパターンと深く関係しており，パターンを記録し閲覧するためのツールとして開発されたものである[6]．Wiki を開発した Cunningham は，当初自身のためのツールだったが，HyperCard でつくられたコンテンツは，コピーを共有することによって，複数の利用者が同時編集を行うことが可能となった．また，Wiki の特徴として，互いのリンクを容易に実装できる．特に文章中に他パターンを参照することで，パターン間の繋がりを強くし関係性を深めることは，パターン・ランゲージを抽出する上で重要である．これらの理由から，誰もが閲覧でき，パターンを編集することを目的として，Wiki によるパターンマイニングの手法を採用した．

3.2.5　時系列・ドメイン別にパターンを配置

ワークショップ，インタビューによって抽出されたパターンを時系列・ドメイン（プロセス，実装，設計）別に配置していく．このプロセスにより，パターンを時系列・ドメイン別に配置することにより，視覚的に開発プロセスを予測することができ，パターン間の関係性を抽出することができる．以下に時系列・ドメイン別に配置されたパターンの具体例を図 4 に示す．

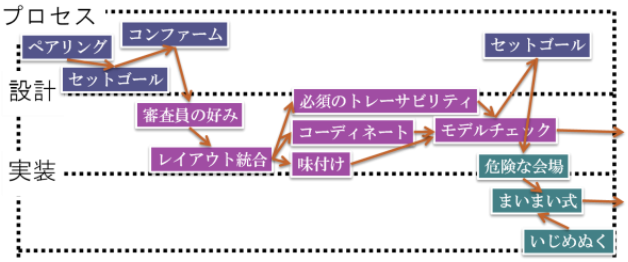


図 4　時系列・ドメイン別に配置されたパターンの具体例

3.2.6　パターンのマトリクス化

パターン間の繋がりをより視覚的に把握できるための工夫として，パターンの関係をマトリクス化するとよい．特にドメイン別に区分けされたパターンのマトリクスと，パターン同士の関係を示したマトリクスは有効である．ドメイン別に区分けされたパターンのマトリクスの具定例を表 3，パターン同士のマトリクスを示した関係の具体例を表 4 に示す．

	パターン ＼ 種類	コース	外乱光	走行体	チーム	開発	モデル評	レイアウ	走行	難所
1	コースメイク	○							○	○
2	コースシェア	○								
3	コースファースト	○				○			○	○
4	いじめぬく		○						○	
5	危険な会場		○						○	
6	瞬時に充電			○					○	
7	モータの性能差			○		○			○	
8	ペアリング				○	○				
9	セットゴール				○	○				
10	コンファーム	○			○	○	○			
11	たくさんの走行体					○			○	○
12	取捨選択					○				○

表 3　ドメイン別に区分けされたパターンのマトリクスの具定例

	パターン ＼ パターン	コースメイク	コースシェア	コースファー	いじめぬく	危険な会場	瞬時に充電	モータの性能
1	コースメイク		○	○				
2	コースシェア	○						
3	コースファースト	○						
4	いじめぬく					○		
5	危険な会場				○			
6	瞬時に充電							○
7	モータの性能差						○	

表 4　パターン同士の関係を示したマトリクスの具体例

早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室

3.2.7 開発プロセス

時系列・ドメイン別にパターンを配置した結果から，予測される開発プロセスを抽出する．その際，入力，アクティビティ，出力を明らかにするため SPEM(Software Process Engineering Metamodel)を利用する．SPEM はソフトウェアプロセスを定義するための OMG 承認の UML Profile であり，SPEM を用いることで，パターンの開発プロセスにおける役割を詳細化することができる[7]．以下に例として，図 2，表 3，表 4 から抽出された開発プロセスを図 5 に示す．

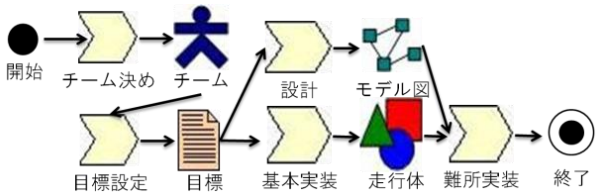


図 5　抽出された開発プロセス

4. 抽出されたパターン・ランゲージ

4.1 実際の抽出過程

本研究では，実際に ET ロボットコンテストを題材として，パターン・ランゲージを抽出している．抽出期間は，2011 年 10 月から 2012 年 1 月までの間に，3 回のワークショップと複数回におよぶインタビューによって，パターン・ランゲージを抽出した．

4.2 パターン・ランゲージの進化

実際に抽出したパターン・ランゲージと開発プロセスの関係を示す．初期段階におけるパターン・ランゲージは，繋がりも薄く，再利用性の低い成果物である．しかし，抽出とサイクルを繰り返すことで，パターンは洗練され，最終的に強い関係性を持つ，再利用性の高い成果物を得られたと考えられる．10 月のパターン・ランゲージと開発プロセスの関係を図 6 に，11 月を図 7 に，最終的に抽出された 1 月の両者の関係を図 8 に示す．

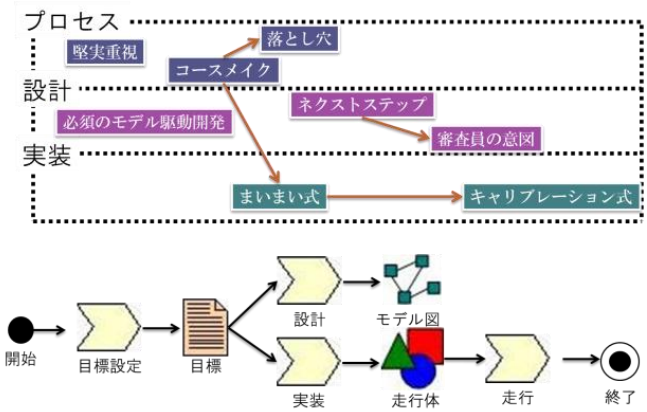


図 6　10 月のパターン・ランゲージと開発プロセスの関係

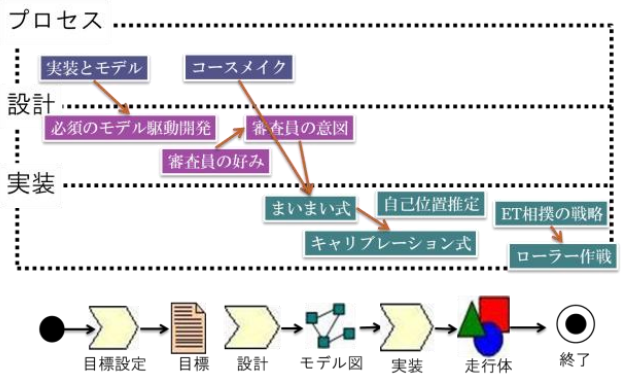


図 7　11 月のパターン・ランゲージと開発プロセスの関係

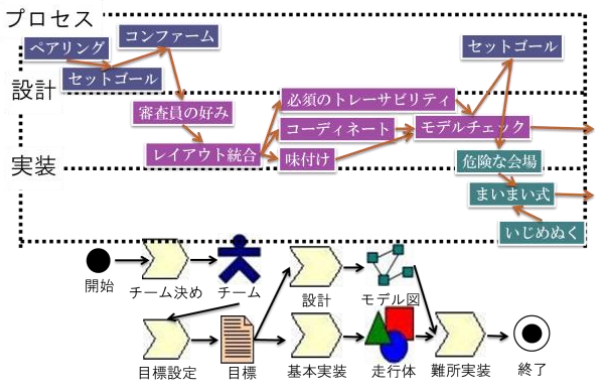


図 8　1 月のパターン・ランゲージと開発プロセスの関係

4.3 パターン・ランゲージの全体図

実際に新たなマイニング手法を用いて，ET ロボットコンテストのパターン・ランゲージを全部で 35 個抽出した．これら抽出されたパターン・ランゲージを「環境」「モデル」「実装」「プロセス」の 4 つに分類した．「環境」のパターン・ランゲージは主に，ロボットに起こる不具合やコースの設置に関するパターンを集めている．「モデル」のパターン・ランゲージでは，ET ロボットコンテストにおいて UML で記述されたモデルも評価対象となるので，評価されるモデルに成長させていくノウハウを集めたものである．「実装」のパターン・ランゲージは，主にロボットがコースを走るための攻略法を集めたものである．「プロセス」は，チームの構成やチーム単位で動く時の留意点をまとめたものである．また，抽出されたパターン・ランゲージは，「名前」「サムネイル」「状況」「問題」「フォース」「解決策」から成り立つ．実際の具体例を図 9 と実際に抽出されたパターン・ランゲージの関係性をドメイン別に振り分けたものを図 10 に示す．

たくさんの走行体

早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室

■状況

難所などの実装を並行開発で行いたいとき

■問題

並行開発を行う際，走行体が１つしかないと，開発したプログラムをテストしようと思っても，他のチームがテストをしていると，競合してしまい効率よく開発が行えない

■フォース

・並行開発を行う人員がいる
・走行体を複数所持している
・資産に余裕がある

■解決策

走行体を複数所持することで効率よく並行開発を行うことができた．ただし，走行体による駆動差があるため，細かなチューニングを行う場合は複数の走行体は必要ない．
また，複数のコードが生成されるので，定期的にマージを行う必要性がある

図 9　パターン・ランゲージ具体例

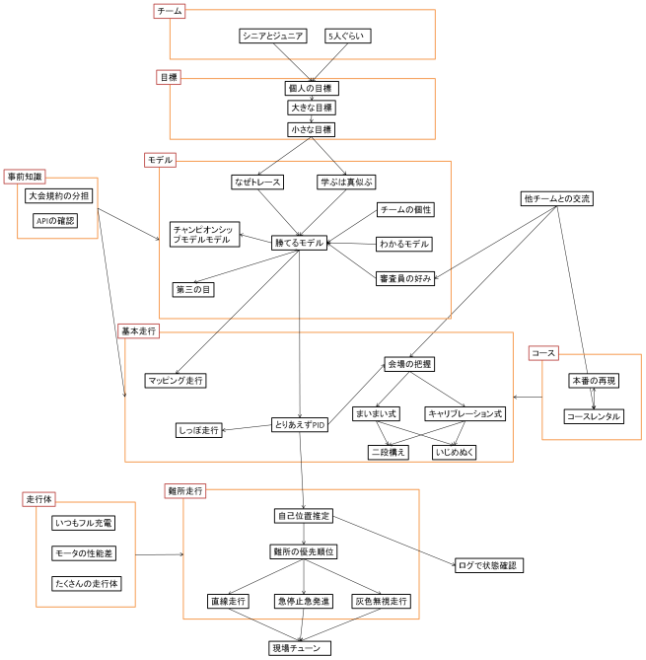


図 10　パターン・ランゲージの全体図

早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室

5. おわりに

従来のパターンマイニングは，開発プロセスが明確であることを前提としている．しかし，技術者自身もプロセスを理解していない開発におけるパターン・ランゲージの抽出は困難かつ，その整合性が確かでない．

本稿では，パターン・ランゲージの抽出と共に，プロトタイプとしての開発プロセスを同時に抽出する過程を導入し，その過程を繰り返す新たなパターンマイニングの手法を考案した．その結果，開発プロセスと抽出されたパターン・ランゲージは同期し，整合性の取れたパターンマイニングを行うことができる．また，何度も抽出する過程を繰り返すことで，効果的な抽象度のパターンを抽出できると考えられる．

また，我々は「ET ロボットコンテスト」を題材としているため，それに特化したパターン・ランゲージである．したがって，これらパターン・ランゲージを抽象化することで，コンテスト形式や組み込みソフトウェアの開発に対応できるパターン・ランゲージを生成することができると考えられる．

謝　　辞

本研究を進めるにあたり，パターン・ランゲージを生成する上で，ワークショップやインタビューにご協力頂いた鷲崎研究室・深澤研究室の皆様に深く感謝致します．

参 考 文 献

1) 鷲崎 弘宜，“ソフトウェアパターン概観”，情報処理，vol.52，pp.19-21，2011.
2) 井庭 崇，“パターンランゲージ 3.0”，情報処理，vol.52，pp.51-56，2011.
3) 鷲崎 弘宜，“ソフトウェアパターン・マイニングに関する一考察”，ウィンターワークショップ 2005，2005.
4) 湯村洋平，若松 孝次，井庭 崇，“プロジェクト推進のためのパターン・ランゲージとその進化”，情報処理学会，vol.68，2008.
5) Kathrin Lappe，“Sharing Requirements Engineering Experience Using Patterns”，IEEE Software，Vol.22，No.1，2005.
6) 鷲崎 弘宜，“ソフトウェアパターン概観”，情報処理，vol.52，pp.19-21，2011.
7) UML でソフトウェアプロセスを定義する -SPEM を使って- http://www.ogis-ri.co.jp/otc/hiroba/technical/UML_Profile/SPEM/

付　　録

最後に抽出されたパターン・ランゲージの概要を示す．

種類	No	部類	パターン名	概要
環境	1	コース系	コースの再現	本番のコースと近い環境が整っていないと、思わぬアクシデントに遭遇する。そのため、レプリカのコースを購入し、地面にスタイロフォームを入れる必要がある。
	2	コース系	コースレンタル	コースを購入する等の金銭的余裕がない場合は、他チームから練習台を借りる。
	3	外乱光系	光でいじめぬく	まいまい式がきちんと実装されているか把握できていない場合、プロジェクターや懐中電灯で外乱光を作り出す、もしくは場所を変更させて実装されているか確認する。
	4	外乱光系	会場の把握	会場によっては、外乱光の強いところが存在する。なので、会場の状況を把握する必要がある。
	5	外乱光系	二段構え	実際に走行させるとキャリブレーション式では対応できない場合がある。なので、キャリブレーション式とまいまい式の 2 つのプログラムを用意しておき、どちらにも対応しておく。
	6	走行体系	いつもフル充電	走行体は電圧による駆動差が出やすい。同じソースコードでも段差を登れない等の問題が発生する。特に本番はフル充電の状態で走行するので、フル充電の電池を確保できるような状況を常に維持するべきである。
	7	走行体系	モータの性能差	走行体が直線的に動かないのは、モータの駆動差によるものである。なので、なるべくモータの駆動差が少ないものを選ぶ、もしくは補正をかけることで直線的な走行を可能とする。
プロセス	8	チーム系	シニアとジュニア	チームに経験者と未経験者が混在していると、タスクの分割等、あらゆる問題が発生する。なので、経験者と未経験者チームで分けることにより、開発の効率だけでなく、未経験者の教育にもつながる。
	9	チーム系	個人の目標	チームを形成しても、一人一人の目標を確立しないと、大きな目標や日々の目標が明確にならない。また、個人の目標を明確にすることで、互いの理解を深める結果にも繋がる。
	10	チーム系	大きな目標	目標が明確に定まっていないと、チームのモチベーションや会議の内容も薄くなってしまう。モデル部門と競技部門において、明確な目標を設定することでモチベーションの維持だけでなく、マイルストーンの設定も容易にできる。
	11	チーム系	日々の目標	日々のスケジューリング（マイルストーン）を明確にするために、大きな目標を達成するための詳細を日々決定していく。
	12	チーム系	大会規約の分担	大会規約は大量に存在するので、すべてを個人が把握するには限界がある。規約が公式に発表されたら、それぞれの規約の担当を決めて、それらをシェアすることで効率よく開発を行える。
	13	チーム系	他チームとの交流	コースの提供、情報を共有・入手できるタイミングを逃さないようにするため、他チームとの交流を多く持つことで、コースレンタルや新たな技術の発見が期待できる
	14	開発	たくさんの走行体	並列開発を行うために、走行体を複数所持することで開発効率をあげることができる。
	15	開発	難所の優先順位	難所の開発を行う際、技術の差はあるもののすべての実装を完璧にこなすことは困難である。なので、コースの順番、ボーナスタイムの順に優先順位をつけて開発を行う。
	16	開発	現場チューン	光センサーの値は、練習と本番の環境では対応できなくなるため、サンプルコースを用意することで、最後にチューンをすることができる

早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室

	17	開発	API の確認	それぞれの値（光センサーやジャイロセンサー）を理解していないと、実装どころかモデル図もかけないため、モデルを書く前に入力の値の確認をする。
モデル	18	評価	チャンピオンシップモデル	予選突破からチャンピオンシップに向けてモデルの改善を行う必要がある。その場合、予選との差分、ネガティブな意見に対するアンサーを用意する等の対策を行う。
	19	評価	審査団の好み	どのようなモデルを評価されるか具体的に把握していないとき。ET ロボコンで開催されるワークショップで、モデル審査の評価対象をきちんと記録することが重要。去年度のワークショップに参加していれば、トレンドもわかるので尚よい。
	20	評価	学ぶは真似ぶ	「審査員の好み」同様、評価されるモデルが把握できていない場合は、とりあえず例年の評価されているモデルを真似することが定石である。
	21	評価	チームの個性	他チームに差を付けたいとき。差をつけるためには、ラインから外れたときの解決策の提案や、省電力・研究分野を取り組むとよい。
	22	評価	勝てるモデル	モデルはチャンピオンシップにおいても変更を余儀なくされる。その場合、トレーサビリティの取れているモデルを初期の段階から作成することで変更を用意にさせることができる。
	23	評価	第三の目	開発者自身では気づかない問題点やレビューをもらうために、第三者にモデルをチェックしてもらう
	24	評価	なぜトレース	「なぜ」その機能が必要なのか、「なぜ」その値を利用するのか、その「なぜ」を繰り返すことで、トレーサビリティの取れたモデルへと進化していく
	25	レイアウト	わかるモデル	モデルの内容以外にも、他者から見て理解しやすい内容になっているかも重要である。なので、単語の意味、定義、配色、配置にもきちんとこだわる必要がある。
実装	26	走行	とりあえず PID	滑らかな走行でないと、無駄な距離を走行してしまい、結果タイムも遅くなる。なので、PID 制御を用いる。
	27	走行	キャリブレーション式	外乱光対策に対して弱いが、スピードは期待できる。
	28	走行	まいまい式	外乱光対策の走行方法。スピードは期待できない。
	29	走行	しっぽ走行	倒立振子の値を常に考慮すると、スピードに限界がでてしまう。しっぽをはじめから下した状態で走行することで、安定性を増して走行することができ、スピードも出せる。
	30	走行	限界感度法	PID の値を調整するときに利用できる方法。詳しくは、http://monoist.atmarkit.co.jp/mn/articles/1007/26/news083.html
	31	走行	マッピング走行	ライントレースをしないで、曲がる位置、角度を記憶し、正確に実装することで、センサー等を用いずに走行する方法
	32	難所	急発進急停止	段差のある難所ポイントを上る際、急発進急停止することで段差を攻略できる。
	33	場面	自己位置推定	走行体がどこを走っているか把握できないとき。時間とジャイロセンサーを用いることで、どこを走っているか走行体自身が把握することができる。
	34	場面	ログで状態確認	走行体が難所によって、モードを変化されているか確認したいとき。走行体のログに値を与えることができるので、それを利用することでモードが切り替わったか判断できる。
	35	場面	灰色無視走行	しっぽ走行をしていると、灰色の検知がうまくできない。なので、自己位置推定とモータの性能差を組み合わせることで、灰色検地をおこなわずに攻略できる。

早稲田理工大学院　基幹理工学研究科
情報理工専攻　鷲崎研究室