ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

アーキテクチャとパターン – WW2009・セッションテーマ紹介

鹿糠秀行[†1]　羽生田栄一[†2]　鷲崎弘宜[†3]

本稿では，ソフトウェア開発におけるアーキテクチャ技術とパターン技術のそれぞれの現状や両者の関係の展望を得るために「アーキテクチャとパターン」セッションが実施予定の議論内容を説明する．

Topics of the Session on Software Architecture and Patterns

Hideyuki Kanuka,[†1] Eiiti Hanyuda[†2] and Hironori Washizaki[†3]

This paper explains what discussions will take place in the session on software architecture and patterns to get insight into the directions of future software architecture/patterns efforts.

1. はじめに

ソフトウェアパターンは，ソフトウェア開発における特定の状況のもとで繰り返される問題について，考慮すべき事柄と解決策を抽象化してまとめあげた記述であり[1)]，ソフトウェアのプロダクトやプロセス等の構造について何らかの指針を与えることが多い．一方，アーキテクチャはシステムの基本的構造の記述である．一般にアーキテクチャは，機能要求や非機能要求に応じた決定の組み合わせで成立し，その過程で既知の有効なパターンを参照する，あるいは，新パターン候補が得られるなど両者は密接に関係する．

両者はそれぞれ活発に研究/実践されているが，近年の開発環境の急速な進化および社会環境の変化によりそれぞれ議論し尽くされておらず，また，両者の関係は未整理な部分も多い．そこで本セッションでは，参加者のポジションペーパ発表を起点として各技術に関連する経験や提案を概観し，続いて，アーキテクチャとパターンのそれぞれの課題や展望，両者の関係や周辺技術について幅広く議論する．

本稿では以降において，これまでの議論経緯と本セッションにおける議論のテーマ案を紹介する．

2. これまでの議論

情報処理学会ソフトウェア工学研究会では，2003 年にパターンワーキンググループ[2)] が結成され，以降，パターンやパターン活用支援技術に関する議論を重ねてきた．ウィンターワークショップにおいては，2004 年よりワーキンググループが中心となって「パターン」セッションおよび「アーキテクチャとパターン」セッションを設置し，各議論の成果を公開してきた．これまでの成果を以下にまとめる．

- 石垣島 2004: ソフトウェア要求獲得を実験し，建築でのパターンランゲージがもたらす要求獲得支援効果が，ソフトウェア開発についても得られることを明らかにした[3)]．これは，パターン活動プロセス[4)] を構成する利用活動の評価/適用工程について得られた知見である．
- 伊豆 2005: 参加者の経験からパターンを発掘した結果，パターンの必要条件として対立するフォースの組が不可欠なことや，マイニング手法によってパターンランゲージへの発展を支援できる可能性を明らかにした[5)]．これらは，パターン活動プロセスを構成する抽出活動の発見/記述工程について得られた知見である．
- 鴨川 2006: ライターズワークショップ[6)] の実施により，パターンランゲージが備えるべき特性として，範囲設定の妥当さ，および，パターン間の関連の明確さを明らかにした[7)]．さらに，パターン活動支援技術の特性を幾つか明らかとした．
- 那覇 2007: 種々の要素技術の討議の上で，ソフトウェア開発におけるアーキテクチャとパターンの関係を整理することを試みた[8)]．特に，アーキテクチャ設計に注目した場合に，同関係の 1 つの見方としてパターンがコンテキストと実現手段を結びつけるものであり，かつ，プロセスであると位置づけ可能なことを整理した．
- 道後 2008: 各技術に関連する経験や提案を概観し，続いて，アーキテクチャ進化とパターンの課題や展望，関係について幅広く議論するとともに，

†1 株式会社日立製作所 Hitachi, Ltd.
†2 株式会社豆蔵 Mamezou, Co., Ltd.
†3 早稲田大学 Waseda University

アーキテクチャ進化のメタモデル記述実験やプレゼンパターンのパターンランゲージ化実験を行い，進化やプレゼンパターンの可能性を確認した[9]．

3. 議論テーマ

以上 5 年間の継続的議論は主にパターンの抽出/利用の観点から実施されたものであり，2007–2008 年の議論についてもパターンとアーキテクチャの関係について詳細に議論し尽くされていない．特に，両者について機能/非機能要求や品質，設計，モデリング技法などが密接に関係するため，それらを軸としたさらなる議論が必要である．具体的には下記を検討する．

3.1 要素技術および特性

ポジションペーパ発表に基づくアーキテクチャとパターンそれぞれの特性や活用技術，周辺技術（例えばモデリング，要求工学など）との関係を議論したい．

3.2 アーキテクチャとパターンに関するモデリング実験

アーキテクチャとパターンそれぞれの特性や，例えばコンポーネントやリファクタリングなどの周辺技術との関係をモデリングしたい．モデリングを通じて，これまで継続的に議論してきた内容の再整理を行い，成果と課題を共有することで，当該分野における今後の研究方向性を討論したい．

本セッションの限られた時間内でモデリングを遂行するために，これまでの議論内容や各発表ポジションペーパの内容に加えて，既存のリファレンスモデルを材料にして考えたい．リファレンスモデルとしては，アーキテクチャに関してその特性を表現した IEEE1471-2000[10] の概念モデル，パターンに関してはデザインパターンとリファクタリングの関係を中心にそれらの特性を表現した文献[11] のオントロジーなどを想定している．

3.3 アーキテクチャの設計と評価の実験

アーキテクチャの設計および評価にあたり，課せられた品質要求に基づいて既知のアーキテクチャパターンやアーキテクチャスタイルを参照あるいは識別し，設計・評価に活用することが多い（例えば CMU/SEI の ADD や ATAM[12]）．そこで，具体的なアーキテクチャレビューあるいは設計の実験を交えて，アーキテクチャ設計・評価におけるパターンの役割や有効性，今後の技術展望を議論したい．

3.4 非機能要件（NFR）の分析とデザインパターンへの橋渡し

NFR が相互にいかに依存しているかを具体的な例題を利用してワークショップ形式で検討し，どのような形式や手法で NFR の相互関係を管理するのが妥当・現実的なのかを探ってみたい．IPA/SEC NFR-WG で提案の依存関係確認表を提示し，それをたたき台にそれをアーキテクチャ設計にどのように利用可能か，より改善した表現形式は考えられないかと議論したい．ここでの議論を通して，現在，NFR-WG で研究中の，レファレンスアーキテクチャを使った NFR とアーキテクチャ設計の橋渡しに活かすための課題と展望に関して示唆を得られれば幸いである．

4. おわりに

本セッションにおける議論の成果は，パターンワーキンググループの Web サイトやソフトウェア工学研究会を通じて公開し，さらなる議論へと繋げていく予定である．今後，さらなる議論，研究および実践を通して，得られた知見が検証され，ソフトウェア開発活動および組織活動一般においてアーキテクチャおよびパターン技術が活用されることを期待する．

参考文献

1) 深澤 編, 鷲崎, 丸山, 山本, 久保 著: ソフトウェアパターン, 近代科学社, 2007.
2) http://patterns-wg.fuka.info.waseda.ac.jp/
3) 松下ほか: ウィンターワークショップ・イン・石垣島参加報告, 情処研報 2004-SE-145, 2004.
4) 鷲崎ほか: ソフトウェアパターン研究の発展経緯と最近の動向, 情処研報 2004-SE-147, 2005.
5) 紫合ほか: ウィンターワークショップ 2005・イン・伊豆参加報告，情処研報 2005-SE-148, 2005.
6) 羽生田 監修, パターンワーキンググループ 著: ソフトウェアパターン入門, SRC, 2005.
7) 満田ほか: ウィンターワークショップ 2006・イン・鴨川参加報告, 情処研報 2006-SE-152, 2006.
8) 松塚ほか: ウィンターワークショップ 2007・イン・那覇開催報告, 情処研報 2007-SE-156, 2007.
9) 阿萬ほか: ウィンターワークショップ 2008・イン・道後開催報告, 情処研報 2008-SE-160, 2008.
10) IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software Intensive Systems Description
11) J.Garzas and M.Piattini: An ontology for microarchitectural design knowledge, IEEE Software, Vol 22, pp28-33, 2005.
12) L. Bass, et al.: Software Architecture in Practice, Second Edition, Addison-Wesley, 2003.

ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

ソフトウェアトラブル事例分析のためのプロセスモデリング

鹿　糠　秀　行[†1]　村田　大二郎[†1]　三　部　良　太[†1]

ソフトウェアトラブルプロセスモデル (STPM) を提案し，STPM を利用したソフトウェアに関わる欠陥の混入過程と欠陥によるトラブルの発生過程の可視化について検討する．

Process Modeling for Analyzing Software Trouble Cases

Hideyuki Kanuka,[†1] Daijiro Murata[†1] and Ryota Mibe[†1]

We propose Software Trouble Process Model (STPM) for visualizing a process of software defect introduction and a process of trouble due to the defect.

1. はじめに

我々の生活を取り巻くソフトウェアの数が増大する一方で，社会的にも大きく影響を及ぼすようなソフトウェアの欠陥に起因したトラブルの発生が顕著になってきている．

ソフトウェア欠陥予防[1)] は，ソフトウェアに起因するトラブルの発生をできる限り予防するために，その発生原因を分析し再発防止策を実施する活動である．トラブルを引き起こす欠陥は，テストなどによって未然に検出し除去するだけでは十分でない．欠陥を混入させる人的なエラーの分析や予測を行い，エラーの再発防止策を実施することで，想定しうるエラーによる欠陥の混入を防止する必要がある．

我々は，ソフトウェア欠陥予防にあたり，ソフトウェアトラブル事例を分析し，欠陥などトラブルの原因に関わる事実を識別し，欠陥の混入過程と欠陥によるトラブルの発生過程を可視化するためのソフトウェアトラブルプロセスモデル (STPM) を提案する．

2. STPM について

STPM は，ソフトウェアに関わる欠陥などのトラブルの原因に関わる概念とその関係を抽出し，UML のクラス図で表現したものである．STPM を**図 1** に示す．

トラブルの直接原因として欠陥（バグ）を考え，その混入や見落しについては，プロジェクト，作業単位，エラー，アーティファクト，欠陥，これらの関係から表現できるようにする．トラブルの発生については，状況，アーティファクト，欠陥，障害，トラブル，これらの関係から表現できるようにする．

STPM に基づいてトラブル事例を分析し，UML のオブジェクト図を作成することによって，定める観点でトラブルの原因に関わる事実を識別し，それらの関係を可視化できる．

なお，エラー，欠陥，障害など，これらトラブルの原因に関わる概念の定義が多数存在している[2)]．STPM においては，それらの定義を各クラス間の関係から明確に表現できている．

各トラブル事例の欠陥混入過程とトラブル発生過程は，STPM を利用して UML のシーケンス図で表現する．**図 2** に例を示すように，欠陥混入過程とトラブル発生過程を可視化でき，それら過程における原因と結果の因果関係を把握しやすくなる．

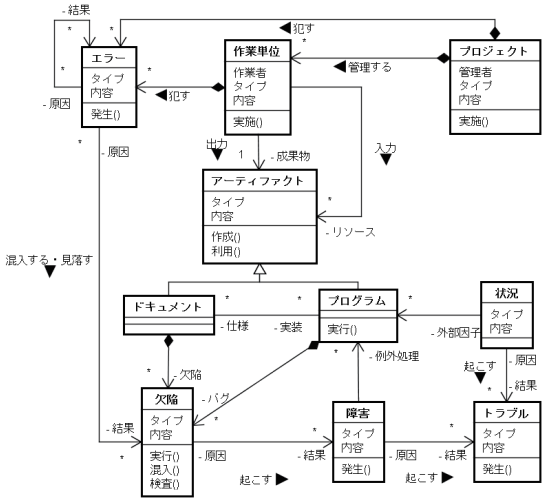

図 1　ソフトウェアトラブルプロセスモデル (STPM)

†1 (株) 日立製作所　Hitachi, Ltd.

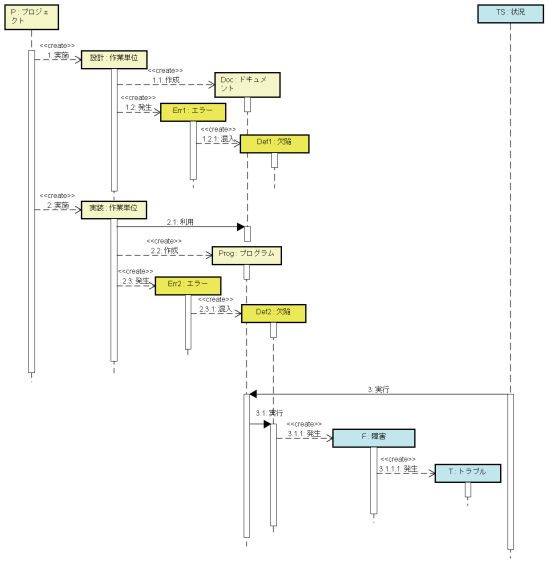

図 2 STPM による欠陥混入過程とトラブル発生過程の可視化例

STPM の各クラスについて，以下の節で説明する．

2.1 プロジェクト

プロジェクトは，ある管理者の下で実施されるソフトウェア開発プロジェクトを表現するクラスである．タイプという属性は，その値として新規開発や保守などプロジェクトの種類を表現する．

2.2 作 業 単 位

作業単位は，ある作業者が複数のリソースを入力として一つの成果物を出力するという役割を持つ作業を表現するクラスである．ここでは，この作業単位の組み合わせからプロジェクトが構成されると考える．タイプという属性は，プロジェクトの作業工程を表現する．

2.3 アーティファクト

アーティファクトは，作業単位で利用されるリソースと，作成される成果物を表現するクラスである．アーティファクトは，プログラムやドキュメントを表現する．ここでは，作業単位で使われる手順書なども，ドキュメントとして考える．タイプという属性は，アーティファクトの種類を表現する．

2.4 エ ラ ー

エラーは，プロジェクトの管理者や作業単位の作業者が犯す人的なエラーを表現するクラスであり，欠陥の混入原因である．テストなどの作業単位で検出すべきエラーを見落した場合も，エラーとして表現する．タイプという属性は，エラーの種類を表現する．

2.5 欠 陥

欠陥は，文書やプログラムのバグを表現するクラスである．タイプという属性は，欠陥の種類を表現する．

2.6 状 況

状況は，トラブルの原因であり，プログラム中のバグを実行させる外部因子を表現するクラスである．タイプという属性は，状況の種類を表現する．

2.7 障 害

障害は，プログラム中のバグが実行されたときに発生する異常な状態を表現するクラスである．障害が起こった場合には，引き続き例外処理プログラムが実行されるか，またはトラブルが発生すると考える．タイプという属性は，障害の種類を表現する．

2.8 トラブル

トラブルは，ソフトウェア外部に及ぼす悪影響を表現する．タイプという属性は，トラブルの種類を表現する．

3. STPM の応用に関する検討

ソフトウェア欠陥予防においては，ソフトウェアトラブル事例を分析するだけでなく，分析した事例を第三者が理解できる形式で知識として蓄積し活用することが重要であると考える．

そのような知識事例として，科学技術振興機構の失敗知識データベース[3]，文献 4) の問題事象と対策事例など，その対象とする分野や表現形式は様々であるが，一般に公開されているものがある．

一方，組織内部のトラブル事例を，一般に公開することは難しいが，組織固有の知識として限られた中で蓄積し活用することは欠陥防止に有用であると考える．

我々は，組織が有するソフトウェアトラブル事例を STPM に基づいて分析し一般化を行い，パターンとして知識化し活用することを考えている．

4. おわりに

STPM を提案した．今後，応用としてトラブル事例を STPM に基づいて分析し，パターンの発掘を試みる．

参 考 文 献

1) Marc McDonald and Robert Musson and Ross Smith: The practical guide to defect prevention, Microsoft Press, 2007.
2) SQuBOK 策定部会編: ソフトウェア品質知識体系ガイド-SQuBOK Guide, オーム社, 2007.
3) JST 失敗知識データベース: http://shippai.jst.go.jp/fkd/
4) 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター: IT プロジェクトの「見える化」〜下流工程編, 日経 BP 社, 2006.

ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

リファクタリングの観点に基づく粗粒度コンポーネント分割の考察

正畠博政[†1]　鹿糠秀行[†1]
神　祐介[†1]　寺濱幸徳[†1]

本稿では，粗粒度コンポーネント分割の支援方法に関してリファクタリングの観点から考察する．

A Consideration on a Support Method for Coarse-graded Component Partitioning in Terms of Refactoring

Hiromasa Shobatake,[†1] Hideyuki Kanuka,[†1] Yusuke Jin[†1]
and Yukinori Terahama[†1]

In this paper, we consider a support method for coarse-graded component partitioning in terms of refactoring.

1. はじめに

大規模システムの開発においては，開発早期に，開発対象を論理的に大きな単位の粗粒度コンポーネント（以下，コンポーネント）に分割し，分割した単位で開発を進めることで，生産性と保守性の向上を図る．

我々は，コンポーネント分割における観点としてコンポーネント間の結合度に着目し，結合度を強める不正依存関係とその改善によるコンポーネント分割支援方法を提案した[1)]．

本稿では，分割支援方法をさらに有用にするために，リファクタリングに基づく新たな観点の導入を検討する．

2. 議　　論

分割作業においては，段階的に追加・変更される要件に対してイテレーティブに，成果物であるコンポーネント体系を洗練させていく．この洗練させる作業は，コンポーネント体系の検証と改善であり，リファクタリングの作業と合致する．つまり,「不吉な匂い」を検出し，これを回避する「改善策」を適用する．例えば，コンポーネントの結合度を強めるコンポーネント要素 (コンポーネントが扱うデータを保持するテーブルや，コンポーネント自身の機能を公開するためのサービスインタフェース (以下，サービス I/F)) 間の相互依存は「不吉な匂い」の一種と考えられる．それに対して，相互依存を回避させるためにコンポーネント要素を変更・移動させるといった「改善策」の適用を考えることができる．

ここでは，リファクタリングの「不吉な匂い」と「改善策」のうち，リファクタリングの各種文献を元にコンポーネント分割における「不吉な匂い」を抽出し，その検出方法を考えて分類する．コンポーネント分割において，検出可能な「不吉な匂い」の種類を増やすことは，分割対象とするコンポーネント体系の改善必要性を判断する材料を増やし，より望ましいコンポーネント体系へと洗練させるための前提となる．

2.1 コンポーネント分割向けの「不吉な匂い」の抽出

リファクタリングの文献[2),3)] から，コンポーネントの分割にも該当する「不吉な匂い」を抽出した．抽出結果を**表 1** に示す．

抽出にあたっては，リファクタリングの文献[2),3)] に登場する要素を，コンポーネントの要素に置き換え，コンポーネントの要素に対しても「不吉な匂い」が成立するものを抽出した．具体的には，クラスとパッケージをコンポーネントに，メソッドをサービス I/F に，属性をテーブルにそれぞれ置き換えた．また，粗粒度のコンポーネントでは想定していない要素を扱うものは除外した．例えば,「継承」を扱うものや，スイッチ文の使用といったプログラム特有の処理記述方法を扱うものがある．なお，抽出にあたって，文献中の用語は粗粒度のコンポーネントに適した言葉に置き換えた．

2.2 検出方法の観点に基づく「不吉な匂い」の分類

前節で示した表 1 の「不吉な匂い」を検出する具体

†1 （株）日立製作所　Hitachi, Ltd.

表 1 粗粒度コンポーネント分割時の「不吉な匂い」

No	「不吉な匂い」	概要	出典
1	サービス I/F の迂回	コンポーネントのサービス I/F を迂回して，内部のデータに直接アクセスしている	文献[2],[3]
2	相互依存	コンポーネント同士が双方向の関連をもつ	文献[2]
3	関連の過多	関連の数が多い	文献[2]
4	重複したサービス I/F	2 つ以上のコンポーネントに同様の処理を行うサービス I/F が存在する	文献[2]
5	仲介人	処理を他のコンポーネントのサービス I/F に委譲しているだけのコンポーネントが存在する	文献[2]
6	属性，操作の横恋慕	あるサービス I/F が他のコンポーネントのサービス I/F を頻繁に呼び出す	文献[2]
7	怠け者コンポーネント	役割の非常に小さなコンポーネントが存在する	文献[2]
8	変更の分散	コンポーネントにおける 1 つの変更が，あちこちのコンポーネントに影響を及ぼす	文献[2]
9	変更の発散	1 つのコンポーネントに対する複数の変更要求が，それぞれ異なるサービス I/F の変更を伴う	文献[2]
10	小さすぎるコンポーネント	コンポーネントが保持するデータの数が，他のコンポーネントと比較して少ない	文献[3]
11	大きすぎるコンポーネント	コンポーネントが保持するデータの数が，他のコンポーネントと比較して多い	文献[2],[3]
12	多すぎるコンポーネント	コンポーネントの数が管理できないくらい多い	文献[3]
13	コンポーネントの循環	3 つ以上のコンポーネントが互いに依存している	文献[3]

表 2 検出方法の観点からの「不吉な匂い」の分類

No	「不吉な匂い」の検出方法の観点から分類			「不吉な匂い」（表 1のNo）
1	コンポーネント要素間(※)の依存関係の情報から検出できる	現在の依存関係の情報から検出できる	検出の基準は数値によらない (ex.サービス I/F を経由せずに，他コンポーネントのテーブルにアクセス)	1,2,13
2			検出の基準となる数値の設定が必要 (ex.要素間の関連の数)	3
3		過去の依存関係の情報も必要	依存関係の変更履歴と，検出の基準となる数値の設定が必要	8,9
4	コンポーネント	サービス I/F の処理内容の情報も必要		4,5
5	要素間(※)の依存関係の情報だけでは検出できない	処理シーケンスの情報（時系列の情報）も必要		6
6	コンポーネント要素間(※)の依存関係の情報は，検出には必要ない	コンポーネントの要素の数から検出できる	検出の基準値の設定が必要 (ex.データの数，コンポーネントの数)	7,10,11,12

(※)‥サービス I/F 間，サービス I/F‐テーブル間，テーブル間

的な方法を考えて，検出方法の観点に基づく分類表を**表 2** に示す．

その結果，抽出した「不吉な匂い」の中には，分割支援方法[1] に示したコンポーネント要素間の依存関係を利用して，検出できるものが含まれていることがわかった．それ以外の「不吉な匂い」についても，検出方法の拡張により，概ね対応できる見通しを得た．

表 2 の No.1 に対応する「不吉な匂い」は，分割支援方法[1] に示した，要素間の不正依存関係の有無を検証する方法により検出可能である．

表 2 の No.2,3 に対応する「不吉な匂い」の検出には，依存関係を示す情報のうち，不正依存関係の有無を示す以外の情報も必要である．No.2 は要素間の依存関係の数のように依存関係の程度を表す数値も用いることにより検出可能である．No.3 は，変更内容とくくりつけられた依存関係の変更履歴や，依存関係が変更される要素の範囲および変更回数を定量的に表す値を用いることにより検出可能である．これらの情報は，いずれも依存関係に関する情報であるため，分割支援方法[1] の検出方法の拡張で対応可能と考える．

表 2 の No.4,5 に対応する「不吉な匂い」の検出には，依存関係以外の情報も必要である．例えば，重複するサービス I/F の判定にはサービス I/F の処理内容の情報が，サービス I/F 間の呼び出しの頻度を判定するためには呼び出し回数を示すような時系列の情報が必要になると考える．また，表 2 の No.6 に対応する「不吉な匂い」の検出には，依存関係の情報は用いない．このため，表 2 の No.4,5,6 に対しては，分割支援方法[1] とは異なる検出方法が必要となるかもしれないが，上記したような必要な情報を追加することによって対応可能であると考える．

3. おわりに

粗粒度コンポーネント分割支援方法に，リファクタリングの観点の導入を検討した．

特にコンポーネント分割にも応用可能な「不吉な匂い」をリファクタリングの各種文献から抽出し，その検出方法を考えて分類した．その結果，依存関係に着目するという分割支援方法[1] の基本方針は変えることなく，検出方法の拡張によって，検出対象の「不吉な匂い」の範囲を広げられることがわかった．

「改善策」についても，表 2 の No.1 に該当する「不吉な匂い」の「改善策」として，分割支援方法[1] に示した不正依存関係の「改善策」を適用可能である．これは，「不吉な匂い」の場合と同様に，文献[2),3)] 中の「改善策」の要素を，コンポーネントの要素に置き換えることで考えた．その他の「不吉な匂い」に対する「改善策」の検討については，今後の課題とする．

今後の方向として，本検討の内容を取り込んだコンポーネント分割支援ツールの開発，およびツールの評価実施による支援効果の検証を考えている．

参考文献

1) 鹿糠 秀行, 神 祐介, 鈴木 滋, 寺濱 幸徳, “粗粒度コンポーネント分割のための依存関係について”, ウインターワークショップ 2008 in 道後 論文集 pp87-88, 2008.
2) Martin Fowler , “Refactoring: Improving the Design of Existing Code”, Addison-Wesley Professional, 1999.
3) Martin Lippert and Stefan Roock , “Refactoring in Large Software Projects”, 2005.

ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

デザインパターン構造のモデル化

下　滝　亜　里[†1]　青　山　幹　雄[†2]

デザインパターンの構造のモデル化を議論する．

Modeling the Structure of Design Patterns

Asato Shimotaki[†1] and Mikio Aoyama[†2]

We discuss an approach for modeling the structure of design patterns.

1. はじめに

デザインパターン[3] の保守性・進化容易性を向上させるために，その構造のモデル化を議論する．

2. 課　　題

多くの人工物と同じく，個々のパターン記述も永続的な存在ではなく，環境に適応して保守・進化できる必要がある[1]．一般化・特殊化の度合いの観点から，以下の二つの適応の必要性を特定できる．

(1)　**大域適応:** パターンの執筆者は，既存のパターンの修正や改善を行う必要がある．

(2)　**局所適応:** パターンの利用者は，記述されたパターンを具体的な適用の状況に合わせて特化する必要がある[1]．

このような適応過程において，保持されなければならない特性に一貫性がある．つまり，修正や改善，特化は，パターン内の要素間，そしてパターン間の関係の妥当性を壊すことなく行わなければならない．

一貫性を保持する適応を容易にするには，個々のパターンの構成要素とその間の関係を明確にする構造モデルの構築，および適応過程において保持されるべき構造特性の特定が必要である．

構造モデルに対する要件には，上記の適応の粒度を表現できることが挙げられる．

3. アプローチ

図 1 に課題へのアプローチを示す．

(1)　従来のパターン記述から，パターンの構成要素と要素間の関係を抽出し，構造モデルを構築する．

(2)　Design Structure Matrix (DSM)[2] により，構成要素間の関係を表現する．

(3)　DSM を用いて，パターン構造の特性を特定する．

(4)　パターン適応時に構造特性の保持を検証する．

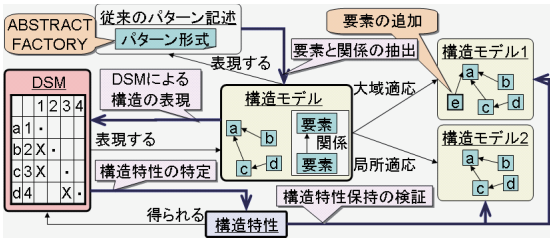

図 1　アプローチ

4. 例　　題

ABSTRACT FACTORY パターン記述[3] を例題として分析し，17 個の要素を抽出した．要素は，「動機」「適用可能性」「構成要素」「結果」の項目から抽出した．「実装」については，他のパターンへの参照があったため抽出の対象外とした．**図 2** に DSM による構造モデルの表現を示す．

5. 考　　察

5.1 要素のカテゴリ化

抽出した要素を以下の基本カテゴリに分類した．

(1)　**文脈:** 要求や要求変化を表す要素．

(2)　**非パターン解:** パターンでないプログラム構造上の解を構成する要素．

(3)　**パターン解:** パターンとなるプログラム構造上の解を構成する要素．

†1 南山大学 大学院 数理情報研究科, Graduate School of Mathematical Sciences and Information Engineering, Nanzan University

†2 南山大学 数理情報学部 情報通信学科, Faculty of Mathematical Sciences and Information Engineering, Nanzan University

パターン要素	GoF形式項目	構成要素		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
文脈	（暗黙）	部品の種類が複数存在	1	・																
文脈	適用可能性	部品の集合が複数存在	2		・															
文脈	適用可能性	複数存在する部品の集合から一つを選ぶ	3		X	・														
文脈	適用可能性	一群の関連する部品を常に使用する	4		X		・													
文脈	結果	使用する部品の集合を変更する	5			X		・												
文脈	結果	部品の種類の追加・削除	6	X					・											
非パターン解	動機（一般化）	特定の部品集合に関わる部品のプログラムをアプリケーション内に直接書く	7	X	X					・										
非パターン解	動機（一般化）	特定の部品集合に関わる部品のクラスをインスタンス化する	8	X	X						・									
パターン解	動機、構成要素	各種の部品のインタフェースを定義する抽象クラス（AbstractProduct）を作成する	9	X								・								
パターン解	動機、構成要素	各部品の具象クラス（ConcreteProduct）を作成する	10		X							X	・							
パターン解	動機、構成要素	各種の部品（AbstractProduct）を生成するオペレーションのインタフェースを宣言する抽象クラス（AbstractFactory）を作成する	11		X							X		・						
パターン解	動機、構成要素	各部品の具象クラス（ConcreteProduct）を生成するオペレーションを実装するクラス（ConcreteFactory）を作成する	12		X								X	X	・					
パターン解	動機、構成要素	AbstractFactoryクラスとAbstractProductクラスで宣言されたインタフェースのみを利用する	13									X		X		・				
非パターン解特性、問題	動機	使用する部品の集合の変更が困難	14					X		X	X						・			
パターン解特性	動機、結果	使用する部品の集合の変更が容易	15					X				X	X	X	X	X	X	・		
パターン解特性	結果	部品間の無矛盾性が促進	16				X							X	X	X			・	
パターン解特性	結果	部品の種類の追加・削除が困難	17						X					X						・

図 2 DSM による ABSTRACT FACTORY パターンの構造モデルの表現

(4) **非パターン解特性:** 非パターン解から得られる特性を表す要素.

(5) **パターン解特性:** パターン解から得られる特性を表す要素.

5.2 非パターン解特性の特化としての問題要素

従来のパターン定義やモデルは，問題要素を基本要素として扱う．本稿のモデルでは，問題要素は，非パターン解特性を特化した要素として表現できる．特化の理由は，非パターン解から得られる特性が必ずしもパターンが解決する問題特性と限らないためである．

5.3 要素間の関係

例題から作成した図 2 の DSM から，要素カテゴリ間に 9 つの関係が特定できた．**図 3** にパターン構造のメタモデルとしてこれらの関係を示す．

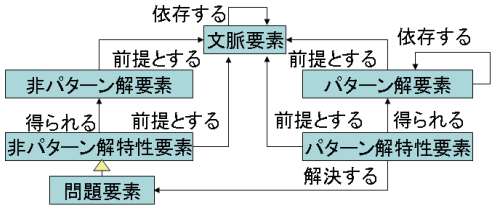

図 3 パターン構造のメタモデル

5.4 パターン構造の特性

メタモデルからは，パターン構造の適応過程では，単独の要素だけでなく，複数の要素が集団的に変更に関わることが分かった．

また，例題では，解特性要素の全てが，文脈要素と依存関係を持つことが分かった．

5.5 パターン適応の一貫性

構造モデルと構造特性を用いることで，構造の一貫性保持の観点から，パターンの適応過程を議論できる．たとえば，解特性の追加は，文脈要素の追加，もしくは既存の文脈要素との依存関係の追加を意味する．したがって，文脈要素との関係を持たない解特性の追加は，構造の一貫性を保持しないと考えられる．

6. 今後の課題

モデル改善の今後の課題として以下を検討する．

(1) カテゴリの種類と粒度の妥当性の検証.

(2) 要素の関係付けの妥当性の検証.

(3) 他のパターン構造のモデル化.

(4) パターン構造間での共通特性の特定によるパターン構造のメタモデルの改善.

(5) パターン間の関係のモデル化.

7. ま と め

本稿では，ABSTRACT FACTORY パターンを例題とし，パターンの構成要素および関係を抽出し，構造のモデルとメタモデルの構築を行った．

参 考 文 献

1) C. Alexander, et. al., A Pattern Language: Towns, Buildings, Construction, Oxford University Press, 1977.

2) C. Y. Baldwin and K. B. Clark, Design Rules: The Power of Modularity, The MIT Press, 2000.

3) E. Gamma, et al., Design Patterns, Addison-Wesley, 1995.

ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

クラスタリング技法を用いたソフトウェアパターン分類

久保淳人[†1]　鷲崎弘宜[†1]　深澤良彰[†1]

ソフトウェアパターンとは，ソフトウェア開発における特定の文脈上で繰り返し発生する問題と，問題に対する実証済の解法として得られるソフトウェア構造，および，解法を導く制約条件などの記述である．ソフトウェアパターンの利用促進のために既存のソフトウェアパターンの収集，分類を行うことが望ましいが，ソフトウェアパターン数の増大に伴って人手での分類が困難になりつつある．本稿では，文書クラスタリング手法を用いたソフトウェアパターンの分類手法を提案する．

Classification of Software Patterns using Cluster Analysis

Atsuto Kubo,[†1] Hironori Washizaki[†1]
and Yoshiaki Fukazawa[†1]

In this paper, we propose a classification algorithm of software patterns using cluster analysis.

1. はじめに

ソフトウェアパターン (以下，単にパターンと表記) とは，ソフトウェア開発における特定の文脈上で繰り返し発生する問題と，問題に対する実証済の解法として得られるソフトウェア構造，および，解法を導く制約条件などの記述である[8]．パターンの利用とは，既に存在するパターンを選択し，理解し，適用し，評価する一連の活動である．ソフトウェア開発の多くの局面について，2200 個以上のパターンが発表されている[6]．しかし，多数のパターン集合から技術者が状況に適したパターンを発見・選択することは困難である．技術者が適切なパターンに到達するために，パターンの選択支援が重要である．著者らは，これまで特に選択支援手法を「分類」「関連分析」「検索」の 3 種に分類した[10]．本稿では，これらのうち特に分類によるパターン選択支援を取り扱う．

パターンの分類による選択支援については，非常に多くの取り組みがあり，多くのパターンカタログ[⋆1]では何らかの形でパターン分類が行われる．多くのパターン分類は主に人手により[1),4),5)]，その後計算機を用いた分類支援手法[2),3),7)] が提案された．

大規模なパターン集合を扱う場合，手動による分類では分類者の負担が大きい．逆に，詳細かつ妥当なパターン分類を自動的に実施するのは困難である．この課題に対して，筆者らは大規模で粗い自動パターン分類手法として，文書クラスタリングによるパターン分類手法を検討した[13]．本稿では，上記での議論をふまえて，文書クラスタリングを用いたパターン分類手法を提案する．

2. 提案手法

提案手法の目標は，大規模なパターン集合を，手動による分類が容易な程度まで細分化することである．また，パターンから特定の項目を抽出してクラスタリングを行うことで，クラスタリングにおける分類視点に対応づける．

手法の特徴は下記の 2 点である．

要素別クラスタリング　パターン文書を構成する「文脈」「問題」「解法」といった各項目それぞれを用いてクラスタリングを行う．パターン文書を項目に分解して抽出する手法は 12) にて提案された．

分割最適化クラスタリング手法　凝集型の階層的クラスタリング手法を用いた予備実験[13)] では，初期に形成されたパターンクラスタが徐々に他のパターンを取り込み，上手くクラスタに分割できない現象[9)] が観察されたため，分割最適化を行う k-means 法を用いる．また，選択した種類の項目を一文書と見なした TF-IDF 法[11)] による評価関数を用いる．各クラスタにおいて，他パターンとの距離の合計が最も小さいパターンがク

†1 早稲田大学
Waseda University
⋆1 互いに関連するパターン集合

表 1 クラスター分析結果

グループ	所属パターン
1	Abstract Factory, Prototype, Factory Method, Builder
2	Composite, State, Strategy, Decorator, Adapter, Bridge
3	Chain of Responsiblity, Flyweight, Interpreter, Proxy, Visitor, Template Method, Singleton, Iterator
4	Memento, Command, Observer, Mediator, Facade

ラスタ代表パターンとなる．k-means 法では以下の手順で分析を行う．

(1) 指定された数の初期クラスタを作成し，各パターンをランダムに割り振る．
(2) 各クラスタで代表パターンを選出する．
(3) クラスタの各パターンについて，現在所属しているクラスタよりも近いクラスタが存在すれば，そのクラスタに移動する．
(4) パターンの移動が無くなるまで 2〜3 を繰り返す．

パターン分析者は，分類の観点として特定の項目を選択し，クラスタ数を指定する必要がある．分析システムは，上で述べた手法を用いて分析を行う．

Gamma らのデザインパターンについて，実際に分析を行った結果を**表 1** に示す．項目として「問題」を選択し，クラスタ数として 4 を指定した．なお，k-means 法の特徴として，決定的なアルゴリズムではなく，結果が初期のランダムなクラスタ割り振りに影響を受ける．表に示した結果は複数回実験を行った場合での，比較的良くクラスタ化された一例である．

3. 議　　論

ワークショップでは，以下の点について議論を行いたい．(1) パターン分類を一部自動化するとして，手動でのパターン分類との境界はどの程度の規模にするのが妥当か (2) 手法の拡張可能性，特に類義語の取り扱い．

参 考 文 献

1) Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M.: *Pattern Oriented Software Architecture: A System of Patterns*, Wiley, New York (1996).
2) Cunningham, W. et al.: PatternShare. `http://patternshare.org/`.
3) Frauenberger, C., Stockman, T. and Bourguet, M.-L.: Pattern design in the context space, *Pattern Languages of Programming 2007* (2007).
4) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley (1994).
5) Hafiz, M., Adamczyk, P. and Johnson, R.E.: Organizing Security Patterns, *IEEE Software*, Vol.24, No.4, pp.52–60 (2007).
6) Henninger, S. and Correa, V.: Software Pattern Communities: Current Practices and Challenges, *Proceedings of the 14th Pattern Languages of Programs* (2007).
7) Vrsalovic, V. and Eterovic, Y.: Human Based Pattern Classification. `http://patterns.ing.puc.cl/`.
8) 鷲崎弘宜，深澤良彰：ソフトウェアパターン研究の現在と未来，情報処理学会第 141 回ソフトウェア工学研究会，Vol.2003, No.55 (2003).
9) Witten, I. H. and Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Pub (2005).
10) 久保淳人，鷲崎弘宜，深澤良彰：ソフトウェアパターン選択支援の現状と展望，ソフトウェア開発のパターンとアーキテクチャ(ソフトウェアエンジニアリングシンポジウム 2008 併設ワークショップ) (2008).
11) 徳永健伸：情報検索と言語処理，東京大学出版会 (1999).
12) 久保淳人，鷲崎弘宜，高須淳宏，深澤良彰：文書中のパターン間の類似度による関連分析，日本データベース学会論文誌 DBSJ Letters， Vol.3, No.3, pp.13–16 (2004).
13) 久保淳人，鷲崎弘宜，深澤良彰：クラスタリングを用いたソフトウェアパターンの分類支援手法，ウィンターワークショップ 2008・イン・道後 論文集，SIGSE, pp.85–86 (2008).

ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

リファレンスアーキテクチャに含まれるアーキテクチャパターンと非機能要求間の因果関係分析の予備調査

大　嶽　隆　児[†1]

典型的なユーザー課題に対する包括的なアーキテクチャのベストプラクティスであるリファレンスアーキテクチャ(RA) には，対象のドメインに想定される様々な非機能要求 (NFR) を解決するアーキテクチャパターンが暗黙的に含まれている．しかし，アーキテクチャパターンと NFR の因果関係を理解せずに RA をソリューションアーキテクチャに適用すると期待した品質を達成できないリスクがある．このペーパーは，IPA SEC NFR とアーキテクチャ WG の予備調査について紹介する．

A preliminary study to analyze causal relationships between architecture patterns and non-functional requirements included in a reference architecture

Ryuhji Ohtake[†1]

Reference architecture (RA), which is a consistent set of architectural best practices for typical user problems, implicitly contains architecture patterns to solve various non-functional requirements for a specific domain. However, there is a risk that quality goals cannot be achieved when adopting RA to solution architecture without understanding causal relationships of architecture patterns and NFRs. This paper aims to introduce a preliminary study in IPA/SEC NFR and Architecture WG.

1. はじめに

IPA SEC エンタプライズ系 “要求工学・設計開発技術研究部会” では，“非機能要求 (以下，NFR) とアーキテクチャWG” を 2006 年 4 月より発足し，NFR とアーキテクチャに関するスタディーを始めている．2006 年度には，NFR の記述とアーキテクチャの評価に関わる課題，および，既存の技術や手法の概要を「2006 年度活動報告書」[1] で示した．2007 年度には，NFR の記述方法をガイドにまとめ，事例プロジェクトの要件定義書を題材にして時間効率性と回複性の記述を実験し，ガイドの妥当性分析・評価も合わせて「非機能要求記述ガイド (以下，NFR 記述ガイド」[2] として公開した．2008 年度では，1) NFR 記述ガイドを他の品質特性に拡張する，2) 品質特性を良くするアーキテクチャパターンを調査・分析する，2 つのスタディーを行っている．このポジショニングペーパーは，2) で取り組んでいるリファレンスアーキテクチャ(以降，RA) に含まれている NFR とアーキテクチャパターン分析の予備調査について紹介する．

†1 日本アイビーエム (株) 非会員
ohtakro@jp.ibm.com

2. 本　　論

2.1 問題提起

短期間で高品質の IT システムを構築するニーズの高まりにより，RA に基づくソリューションアーキテクチャ(以下，SA) の設計が一般的になっている．RA は，ベストプラクティスやノウハウを一般化し，対象のドメインに存在する典型的なユーザー課題に対する包括的な解決策として提示されている．そのため，RA には，必要なアプリケーション機能やサービスが網羅されているだけでなく，対象のドメインに想定される様々な NFR を解決するアーキテクチャパターンが暗黙的に含まれている．

アーキテクチャパターンは，制約あるコンテキスト (環境や状況) の中で，IT システムの品質特性を良くする解決策を静的な構造や動的な振る舞いで表現したものである．アーキテクチャパターンが意図している主目的の品質特性を良くする仕掛け (メカニズム) には前提条件になるパラメーターとしてセンシティビティポイント[3] があり，これが満たされないと意図した品質特性が良くならない．また，主目的の品質特性を良くする仕掛けは，他の品質特性を悪くするトレードオ

フポイント[3] になっていることが多い．

しかし，多くの RA は，内包しているアーキテクチャパターンと NFR の因果関係を明示していないため，アーキテクチャパターンのセンシティビティポイントやトレードオフポイントを分析せずに，安易に RA を SA に適用すると期待した品質が得られないリスクがある．

2.2 スタディーの手法

RA に含まれているアーキテクチャパターンと NFR の因果関係を明らかにするため，代表的な RA の一つである Patterns for e-business (以下，p4eb)[4] を事例として取り上げ，次のようなスタディーを行っている．

1. 品質特性別に，品質に効果のあるアーキテクチャパターンを調査・分類する
2. 品質を良くする仕掛けとセンシティビティポイント，および，その原理を識別する
3. 他の品質を悪くする副作用 (トレードオフポイント) を識別する
4. 波及効果，相乗効果などを分析する

2.3 Patterns for e-business の概要

p4eb は，抽象度や粒度が異なる複合パターン，ビジネスパターン，統合パターン，アプリケーションパターン，ランタイム (実行環境) パターンで階層的な体系で RA を構成している．

複合パターン (EC, e-Marketplace, Portals, Account Access) は，複数のビジネスパターン (Self-Service[U2B], Collaboration[U2U], Information Aggregation[U2D], Extended Enterprise[B2B]) を組み合わせたものである．統合パターン (Access Integration, Application Integration) は，ビジネスパターンのフロントエンドやバックエンドの可変性に対応するシステム間接続のパターンである．ビジネスパターンと統合パターンには，戦略的な意図や既存システムの制約に対応する複数のアプリケーションパターンが定義され，それぞれの下位に実行環境のバリエーションとして複数のランタイムパターンが定義されている．更に，特定のアプリケーションパターンから独立して，高可用性とハイパフォーマンスの２つ NFR に特化したランタイムパターンが定義されている．

2.4 ファインディング

ビジネスパターンのアプリケーションパターンを調査・分析した結果，以下のことが確認された．

1. アプリケーションパターンの解説には，統合パターンのアプリケーションパターンを示唆するパターン言語が含まれている．
2. p4eb は，次の 4 つの表現方法がある．
   (ア)トポロジーで表現する
   Point-to-Point, Hub-and-Spoke
   (イ)ステレオタイプやアイコンで表現する
   ステレオタイプ：同期，非同期
   データアイコン：Read-write Data, Read Only Data, Transient Data
   (ウ)構成要素のロール名で表現する
   Router, Decomposition, Agent
3. アプリケーションパターンの解説には，独立して定義されている高可用性とハイパフォーマンスのランタイムパターン以外にも，セキュリティや保守性など他の品質特性に関連する基本的なアーキテクチャパターンとして Quality Attribute Design Primitive[5] や Architectural Primitive[6] が含まれている．
4. ISO/IEC9126 ソフトウェア品質特性[1] の分類は，ソフトウェアだけを対象にしているため，分散システムを対象にした p4eb のアーキテクチャパターンの品質特性を網羅していない．たとえば，性能拡張性 (Scalability)，基盤保守容易性 (Serviceability)，システム管理容易性 (System Manageability) などが存在しない．

3. ディスカッション

N-Tier の分散システムを前提とした p4eb に含まれる品質特性に関連するアーキテクチャパターンにソフトウェアのデザインパターンと同じメタファーが見いだせるか，または，両者には大きな乖離があるか，などをウィンターワークショップで議論する．

参 考 文 献

1) 要求工学・設計開発技術研究部会　非機能要求とアーキテクチャWG，2006 年度活動報告書，IPA SEC，2006
2) 要求工学・設計開発技術研究部会　非機能要求とアーキテクチャWG，非機能要求記述ガイド，IPA SEC，2007
3) Rick Kazman et al., ATAM: Method for Architecture Evaluation, CMU/SEI-2000-TR-0004, 2000
4) Jonathan Adams et al., Patterns for e-business — A Strategy for Reuse, Mc Press, 2001
5) Lenn Bass et al., Quality Attribute Design Primitives, CMU/SEI-2000-TN-017, 2000
6) Uwe Zdun, Modeling Architectural Patterns Using Architectural Primitives, OOPSLA'05, 2005

ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

要求マトリクスからアーキテクチャパタン言語へ —Alexander を辿る—

羽生田　栄一[†1]

IPA/SEC 非機能要求 (NFR) とアーキテクチャWG での活動の変遷を，Alexander の『形の合成に関するノート』から『パターンランゲージ』への発想の転換と対比させながら振り返り，NFR を含む要求からアーキテクチャへの橋渡しをする際の示唆を得ることを本 WS での議論を通して試みたい．

Following Alexander from Req-matrix to Architecture pattern languages

Hanyuda Eiiti[†1]

IPA/SEC NFR and Architecture WG has been studying the way of formulating NFRs and of mapping those to IT architectures. We've realized that NFR-WG activities has followed the C. Alexander's thought process from "Notes on the synthesis of form" to "A pattern language". We try to compare our activity's work products(NFR description form, NFR interdependency graph and matorix, reference architecture patterns) to Alexander's requirement-matrix and a Pattern language.

1. はじめに

IPA SEC エンタプライズ系 "要求工学・設計開発技術研究部会" では，"非機能要求 (NFR) とアーキテクチャWG" を 2006 年 4 月立ち上げ，NFR とアーキテクチャを研究している．この経緯を追うと，NFR の定義や分類に始まり，各 NFR の記述方法を検討し，NFR を含む要求項目の間の依存関係に対するグラフ及び表形式での表現を定義し，そこからアーキテクチャを導出する・ないしはヒントを得る，という流れで行われ，今は NFR からアーキテクチャへのトレースがテーマである．

本論では，ソフトウェアパターンのアイデアの源泉である C. Alexander の『形の合成に関するノート』[3] とその問題点の克服のために考え出された『パターンランゲージ』[4] という思考の流れと照らし合わせながら，NFR-WG での議論を振り返り整理してみたい．

2. Alexander の進んだ道

2.1 『形の合成に関するノート』と要求マトリクス

Alexander の博士論文であり,「デザインのプロセス，すなわち機能に対応する新しい物理的秩序，組織，形を表現している物を発明するプロセス」について述べている．満足せねばらない複数の要求を持ち，要求の間には相互作用があり，それが要求を満足させるのを困難にする．Alexander は次のステップを提示する：

(1)　構造に影響を与える要求をすべて見つける
(2)　独立した要求のペアをマトリクスに記述する
(3)　凝集度の高いパーツ集合と，互いに緩く結合し

	1.	2.	3.	4.	5	.....	30
1. 所有者駐車スペース	○.	●.	●.	○.	○	.....	○
2. 配達用駐車スペース	●.	○.	●.	●.	○	.....	○
3. 集団との接点	●.	●.	○.	○.	○	.....	○
4. 公共設備用スペース	○.	●.	○.	○.	○	.....	○
5. 休息スペース・遊び場	○.	○.	○.	○.	○	.....	●
6. 住居私用出入口	●.	●.	●.	○.	○	.....	●
7. 洗濯施設や外部倉庫	○.	○.	●.	○.	●	.....	○
8. ほこり・風を防ぐ仕切り	○.	○.	●.	○.	●	.....	○
…							
30. 安全快適な歩行路面							

図 1　要求ペアの相互依存性●と独立性○ (7) より)

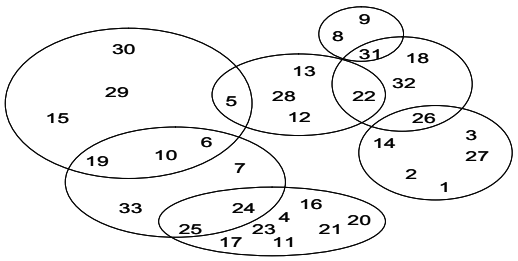

図 2　マトリクスを高凝集度の集合群に分解 (7) より)

†1 株式会社豆蔵．hanyuda@mamezou.com

たパーツ集合にマトリクスを分解する．これらが適切なパーツとなる．

(4)　要求の集合のそれぞれに対しパーツを与える

(5)　これらのパーツが新たな物理システムを構成するように配置する

2.2　パタンランゲージおよび時を超えた建設の道

しかしながら要求マトリクスをインドの農村デザインに試験的に適用した際 140 項目の要求の相互作用分析に数ヶ月要している．実際の問題ならさらに膨大な時間を要するだろう．またステップ (3) の凝集パタンの識別はデザイナのセンスに多分に負うものと思われる．それに気づいた Alexander はあらかじめ妥当な要求セットとパーツとが凝集されたパターンのカタログとそれらの結合の可能性を『パタンランゲージ』として提示するに至る．要求と解法はパタンに構造化され，無名の質を保持するとし，その基本戦略は 5) に示されている．

3. SEC NFR-WG の進んできた道

3.1　要求とくに NFR の記述方法

Alexander には特に言及がないが NFR-WG では 2) で提示したタグ付の XML フォーマットを採用しユースケース・コントロールケースに基づく記述を提案した．

3.2　NFR の関係性俯瞰図と依存関係確認表

NFR 記述に依存関係があることに気づき，グラフと表の 2 形式で NFR 間相互作用の整理を提案した[2)]．特に「依存関係確認表」は要求マトリクスに対応する．ただし正確に言うと，依存関係確認表は ISO9126 品質副特性とその詳細化レベルでの要求項目であり，Alexander らの要求マトリクスが物理構成要素とそれの満たすべき要求という形の記述を基本としているのとは異なる．そのため，この表の解析から凝集モジュールを識別するというプロセスにはなっていない．

3.3　アーキテクチャパターンランゲージへ向けて

NFR-WG では今期，コントロールケースや NFR の記述をもとに妥当なアーキテクチャをデザインするプロセスの検討を開始した．その際，アーキテクチャパターン[9)] を選択しそれらを組合わせることが基本となるが，一方で基本的な組合わせが既に行われた状態のレファレンスアーキテクチャをその適切性条件とともにカタログ化しておくというアプローチとの併用を考えている．これはアーキテクチャのパターンランゲージ化という点で Alexander の思考プロセスを辿っていることになる．

4. 議論と今後

4.1　Alexander の思考プロセスとの共通点

『形の合成』における Alexander の要求記述とその依存関係から設計要素を凝集させるというアプローチの困難さは NFR-WG でも痛感させられた．考えるべき因子の数とトレードオフ関係の錯綜が尋常ではない．

我々はパターンランゲージと呼んでいないがレファレンスアーキテクチャを典型状況ごとに用意しそこからの変異を NFR の分析に基づいてカバーすることを検討している．その際「ランゲージ化」が参考なりそうだ．

4.2　相　違　点

NFR-WG の依存関係確認表は NFR 特性のみに限定しており，アーキテクチャ要素やシステムスコープまで含めた記述間の依存関係を表わすことで，レファレンスアーキテクチャの解像度の粗さが補えるのではないか検討していきたい．

4.3　今後の課題

実際に，利害関係者・機能・NFR とアーキテクチャの相互関係をパターンランゲージ化していく具体的な検討を通して設計プロセスのあり方を考えていきたい．

参　考　文　献

1) 要求工学・設計開発技術研究部会 NFR とアーキテクチャWG, 2006 年度活動報告書,IPA SEC

2) 同上, 非機能要求記述ガイド, IPA SEC, 2007

3) C. Alexander，形の合成に関するノート，鹿島出版会，1978

4) 同上，パタンランゲージ，鹿島出版会，1984

5) 同上，時を超えた建設の道，鹿島出版会，1993

6) 同上，The Nature of Order, Book 1，Center for Environmental Structure，2003

7) J.C.Jones，デザインの手法，丸善，1973

8) Brian Nixon et al., Non-Functional Requirements in Software Engineering, Kluwer Academic, 1999

9) M. Fowler, エンタープライズアプリケーションアーキテクチャパターン, 翔泳社, 2005

10) Jonathan Adams et al., Web システムのデザインパターン—すぐに使える e‐business アプリケーション構築の定石集，翔泳社，2003

表 1　NFR-WG の依存関係確認表（要求マトリクス）

実現手段 ＼ ソフトゴール			効率性				信頼性		
			時間効率性		資源効率性		回復性		
			TAT、応答時間を減らす）	スループット	利用量（を減らす）	利用率？	RPO 復旧時点指標	RTO 復旧時間指標	RLO 復旧状態指標
運用	負荷量削減	スケジューリングによる負荷分散、ユーザ制限	+	+	+				
	性能チューニング		+	+	+				
	保守・点検	より詳細な記述が必要？					+	+	+
ソフトウェア	圧縮		−	−	+				
	索引		+	+	−				
	バックアップ		−	−	−		+	+	+
	監視		−	−	−		+	+	+
	再起動								
	ミドルウェア適用		−	−	−				
	リトライ						?	?	?
	故障診断		−	−	−		?	?	?
ハードウェア	並列処理	マルチプロセッサ、マルチコアプロセッサ	+	+					
	分散処理	クラスタコンピューティング、ロードバランサ	+	+					
	待機予備						+	+	+
	多数決						+	+	+
	ディスクアレイ	RAID（ミラーリング、パ	+	+	−		+		+
		RAID（ストライピング）	+	+	−		−	−	−

ウインターワークショップ 2009・イン・宮崎　論文集　（平成 21 年 1 月）

セキュリティパターンのモデル化と応用に向けて

鷲　崎　弘　宜[†1]

セキュリティパターンとパターン活用技術研究の現状を概観した上で，その基盤を与える様々なモデル化の方法，および，応用の可能性を検討する．

Modeling Security Patterns and Application

Hironori Washizaki[†1]

We overview the current state of security patterns and researches on patterns utilization technology. Moreover, we discuss the way of modeling security patterns and the possibility of application regarding the modeling results.

1. はじめに

計算機やサービス間の相互運用性や接続性が高まると同時に，情報システムの広範あるいは双方向の利用が広まる中で，ソフトウェアシステムの開発および運用においてセキュリティを確保することの重要性が増している．しかしながら，セキュリティは他の品質特性と異なる側面が多く（例えば脆弱性や攻撃への対策が主であること，開発やシステムの全体にわたって考慮する必要があることなど），セキュリティの専門家や過去の対策経験者でなければその確保は難しい．

そこで，熟練者のノウハウや過去の成功事例を習得し再利用する方法としてセキュリティパターンが注目されている．セキュリティパターンとは，特定の文脈において頻出するセキュリティ上の問題に対するよく知られた解決策をまとめあげた記述である[?]．これまでに，分析や設計，実装などのさまざまな工程について多数のセキュリティパターンが抽出され，その報告数は近年増加の傾向にある[2]．

2. セキュリティパターン研究

セキュリティパターンの蓄積と利用機会の増大に呼応して，セキュリティパターンの活用技術に関する研究が世界的に取り組まれ始めている[3]．例えば筆者らは，セキュリティパターンの分類や適用について研究を進めている[4),5]．セキュリティパターン研究の多くは計算機を用いたパターンの自動処理を扱うため，もともとのセキュリティパターンについて注目する側面を何らかの形でモデル化している．

ここで，要求やソフトウェアシステムの分析，設計，実装，開発にあたってセキュリティ上の共通の言語は存在しないのが現状であるため，扱うセキュリティパターンをどのようにモデル化するかが，セキュリティパターン研究・技術の性質に大きく影響を与える．

3. セキュリティパターンのモデル化

セキュリティパターンの活用にあたり，そのモデル化の方法の適切さ・特性が基盤となるとの考えに基づき，筆者らは複数のモデリング言語を同一のセキュリティパターンの表現に適用してその特性を比較する研究を進めている（詳細は文献[6]に掲載予定である）．具体的には，代表的なセキュリティパターンとして Role-Based Access Control（RBAC）パターン[1] を取り上げて，同パターンの注目する側面のモデル化にあたり UML・ミスユースケース，UML のセキュリティ拡張である UMLsec や SecureUML，i*/Tropos/Secure Tropos 等のゴール指向モデル，さらには Problem Frames 等をそれぞれ別個に適用し，その結果を主に表現力や応用の観点から比較考察している．

例えば，RBAC そのものは様々なバリエーションを含めて文献[7] にまとめられており，そこでは，RBAC が適用されている場合にユーザやロール，パーミッション設定およびセッションの間で満たされるべき性質が述語論理によりモデル化されている．同様に文献[8] では，形式言語により性質を厳密に規定している．これらの規定やモデル化は特定の性質・制約の記述にとどまり，周囲の環境やソフトウェアシステム，それらの

†1 早稲田大学 Waseda University

構造や振る舞いとの関係をモデル化したものではない．一方，ソフトウェアシステムのモデル化について代表的な UML（さらには SysML）を RBAC パターンの与える解決構造のモデル化に適用できるが，その表現力は限定的であり，性質・制約の表現に他の記述言語（例えば OCL など）を必要とする．このように，セキュリティパターンのモデル表現とソフトウェア・システムモデルの融合が重要な課題となる．

4. 議　　論

上述のような現状の比較を概観した上で，ワークショップにおいては次のような未整理な点を議論したい．

- 適用や検出，検証といった特定用途の観点からのモデリング言語の適性: 例えば筆者らは，アクタ間の依存関係が複雑な場合のモデリングに特に有効なゴール指向モデル i*[9] を用いて，RBAC パターンをモデル化し，さらにはモデル変換言語を用いて変換により部分的に適用が可能であることを確認しつつある[5]．
  また，セキュリティはアーキテクチャの設計にあたり効率性や使用性といった他の品質としばしばトレードオフの関係にあり，モデリング言語はその関係を表現できると同時に，解消に向けた応用へと役立てられることも望まれる．この点で，ゴール指向モデルは有効に機能する可能性がある．さらに既存のアーキテクチャ設計・評価手法との接続も検討すべき事項である．
- セキュリティパターンそのものの種別や特性に応じたモデリング言語の適性: セキュリティパターンは Problem Pattern（問題・ドメインより）と Solution Pattern（解決より）に大別できる．このとき，その種別に応じて最適なモデリング言語・表現は異なる可能性がある．例えば，ゴール指向モデルは Solution Pattern には不向きな可能性がある．
- プロセスとの関係: Secure Tropos 等の一部の方法論は特定のモデリング言語に加えて，プロセスを内包しているが，他のモデリング言語の適用・併用可能性も含めて検討の余地がある．

参 考 文 献

1) M. Schumacher, E.B. Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, ”Security Patterns: Integrating security and systems engineering,” Wiley 2006.
2) T. Heyman, et al., ”An Analysis of the Security Patterns Landscape,” Proc. 29th International Conference on Software Engineering Workshops (ICSEW), 2007.
3) N. Yoshioka, H. Washizaki and K. Maruyama, ”A survey on security patterns,” Progress in Informatics, No.5, pp.35–47, 2008.
4) E.B. Fernandez, H. Washizaki, N. Yoshioka, A. Kubo, Y. Fukazawa, ”Classifying Security Patterns,” Proc. Asia-Pacific Web Conference (APWeb), 2008.
5) Y. Yu, H. Kaiya, H. Washizaki, Y. Xiong, Z. Hu, N. Yoshioka, ”Enforcing a security pattern in stakeholder goal models,” Proc. 4th ACM Workshop on Quality of Protection (QoP), 2008.
6) T. Tun, et al.: ”Modelling Security Patterns: A Survey,” Chapter in ed. H. Mouratidis , ”Software Engineering for Secure Systems: Industrial and Research Perspectives, ” IGI Global, 2009. (to be published)
7) R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman ”Role-Based Access Control Models,” IEEE Computer, Vol.29, No.2, pp.38–47, 1996.
8) D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli. Proposed NIST standard for role-based access control . ACM Transactions on Information and System Security (TISSEC), Vol. 4, No. 3, pp. 224 - 274, Aug. 2001.
9) E.S.K. Yu. Modelling strategic relationships for process reengineering, University of Toronto, 1996.