ISSN 1884-0760

GRACE TECHNICAL REPORTS

Proceedings of the 1st Asian Conference on Pattern Languages of Programs (AsianPLoP 2010)

Volume II

Hironori WASHIZAKI Nobukazu YOSHIOKA
(editors)

GRACE-TR-2010-01 March 16, 2010

CENTER FOR GLOBAL RESEARCH IN
ADVANCED SOFTWARE SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF INFORMATICS
2-1-2 HITOTSUBASHI, CHIYODA-KU, TOKYO, JAPAN

WWW page: http://grace-center.jp/

The GRACE technical reports are published as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Proceedings

AsianPLoP 2010

1st Asian Conference on Pattern Languages of Programs

Volume II

Tokyo, Japan, March 16-17, 2010, Co-located with

The GRACE International Symposium on Advanced Software Engineering

Edited by

Hironori Washizaki and Nobukazu Yoshioka

Sponsored by

GRACE Center of the National Institute of Informatics (NII)

IPSJ/SIGSE Patterns Working Group

ACM Japan Chapter

Supported by

eXtreme Programming Japan Users Group (XPJUG)

AsianPLoP is a PLoP® Conference sanctioned by the Hillside Group.

PLoP is a registered trademark of The Hillside Group.

Conference Committee

Conference Chair

Eiichi Hanyuda, Mamezou, Co. Ltd.

Program Co-Chairs

Hironori Washizaki, Waseda University / National Institute of Informatics GRACE Center

Nobukazu Yoshioka, National Institute of Informatics

Program Committee

Eduardo B. Fernandez, Florida Atlantic University

Joseph W. Yoder, The Refactory Inc and Joe Yoder Enterprises

Norihiro Yoshida, Osaka University

Masao Tomono, KameNet Inc.

Koido Ryo, eXtreme Programming Japan User's Groups

Terunobu 'Terry' Fujino, InArcadia, Ltd.

Kiminobu Kodama, Information System Research Institute

Kenji Hiranabe, Eiwa System Management, Inc.

Masaru Amano, Eiwa System Management, Inc.

Takeshi Kakeda, Eiwa System Management, Inc.

Kenichiroh Ohta, IBM Japan

Yuriko Sawatani, IBM Japan

Akira Sakakibara, IBM Japan

Takao Okubo, Fujitsu Limited, Japan

Okita Naoyuki, Yokogawa Electric Corporation

Akio Kawai, Object Design Laboratory, Inc.

Yann-Gaël Guéhéneuc, Canada Research Chair on Software Patterns and Patterns of Software, École Polytechnique de Montréal

Yuji Yamano, OGIS International, Inc.

Yoichi Hasegawa, Technoport

Takashi Iba, Keio University

Takashi Kobayashi, Nagoya University

Dinesha K V, IIIT Bangalore

Raj Datta, MindTree Ltd.

Eric Platon, Cirius Technologies

Foutse Khomh, DIRO, Universite de Montreal, QC

Tsukasa Takemura, NSD CO., LTD.

Message from PC Co-Chairs

Welcome to the 1st Asian Conference on Pattern Languages of Programs, AsianPLoP 2010. AsianPLoP takes place at the first time, as a premier event for pattern authors and users to gather, discuss and learn more about patterns and software development in the Asia region as well as other regions.

The purpose of AsianPLoP is to promote development of patterns, pattern languages, technologies and experiences of patterns primarily about software; however, these for domains outside software are also welcome.

In AsianPLoP 2010, various patterns, pattern languages and related techniques will be discussed. Topics include software design, services, security, interaction, pedagogy and organizational change. Most of papers will be workshopped in the traditional PLoP Writer's Workshop format. We received 16 paper submissions. After the rigorous shepherding processes, 13 papers have been accepted for Writer's Workshops and 3 papers for Writing Groups. Moreover the 1st program incorporates one invited talk and one tutorial.

We thank program committee members. They reviewed and conducted shepherding processes for papers carefully and fairly. Moreover we thank our sponsors, supporters and the Hillside Group for their kind supports. We hope that the 1st conference of AsianPLoP is successful, and will contribute the development of this filed.

Hironori Washizaki and Nobukazu Yoshioka

Program Co-Chairs

Invited Talk

Title: A Timeless Way Of Communicating

Presenter: Joshua Kerievsky (Industrial Logic, Inc.)

Abstract

If you pick up the masterpiece, "A Pattern Language", by Christopher Alexander et. al, you will discover a book filled with engaging photographs, hand-drawn sketches, big bold, hard-to-miss text, memorable stories and scholarly notes for the academically minded. One can quickly "surf" this book by focusing only on pattern titles, images and headlines or one can dive deep into the book by reading the detailed text of each pattern. In short, A Pattern Language uses a timeless way of communicating, a form that engages people and provides numerous pathways for accessing the knowledge. As authors of software-related pattern languages, we must understand what it takes to make our own works endure. In this talk, we will analyze the form and content of real-world software patterns/pattern languages, looking for what makes them succeed or fail at engaging the reader and providing knowledge pathways. If you are interested in crafting great pattern languages, this talk will help you discover some essential ingredients.

Biography

Joshua Kerievsky is founder of Industrial Logic, Inc., an early pioneer and expert in Extreme Programming (XP), author of the best-selling, Jolt Cola Award-winning book Refactoring to Patterns , thought leader behind Industrial XP, a state-of-the-art synthesis of XP and Agile Project Management and an innovator of Agile eLearning, which helps organizations “Scale Agility Faster.” Joshua has over 20 years of experience in software development and loves coaching agile project communities, helping executives understand and manage technical debt, leading excellent workshops, and building software products (because it enables him to “walk the agile talk” as an entrepreneur, manager, customer and programmer).

Tutorial

Title: Pattern Writing: The Straight Scoop

Presenter: Joseph W. Yoder (The Refactory, Inc.)

Abstract

Writing Patterns can be a difficult task and getting started sometimes is the most difficult step. Pattern ideas start to emerge from experience practitioners but if you don't have the experience of writing patterns, it can be daunting on how to capture these experiences and start outlining your patterns. This tutorial will discuss ideas on How to Write Patterns. We will discuss Patterns for Writing Patterns and outline some different processes that beginning pattern writers can use to start the process of capturing their patterns. We will also examine some different pattern forms and workshop on some pattern writing.

Contents: Volume II

WW-1J: 品質とパターンランゲージ

「クオリティ」をうみだすプロセスの追求 II-9
中埜博（環境構造センター），笹川万国（ヨーロッパ環境構造センター）

環境デザインにおけるパターン・ランゲージ II-53
岡瑞起（東京大学），李明喜（デザインチーム），橋本康弘（東京大学），江渡浩一郎（産業技術総合研究所）

リアルイベント開催のためのパターンランゲージ II-63
江渡浩一郎（産業技術総合研究所），柴村しのぶ（Wiki ばな）

WW-2J: ファシリテーションと回顧

神輿と寄合のパタン・ランゲージ II-75
本橋 正成

ふりかえりパターンランゲージ(進行役編) II-89
懸田 剛（永和システムマネジメント）

WW-3J: 情報システム

情報システムのためのパタンランゲージに向けて II-101
児玉 公信（情報システム総研）

WW-4J: ビジネスプロセス

ビジネスプロセスパターンの宣言的記述 II-109
竹村 司（日本システムディベロップメント）

WG-1J: テスト

テストカバレッジに基づくテストコードの再構成パターン II-135
坂本 一憲（早稲田大学），和田 卓人（タワーズ・クエスト），鷲崎 弘宜，深澤 良彰（早稲田大学）

Contents: Volume I

WW-1E: Security and Design

A pattern for the WS-Trust standard for web services I-9
Ola Ajaj and Eduardo B. Fernandez (Florida Atlantic University)

A Worm misuse pattern I-21
Eduardo B. Fernandez (Florida Atlantic University), Nobukazu Yoshioka (National Institute of Informatics) and Hironori Washizaki (Waseda University)

Design Decision Topology Model for Pattern Relationship Analysis I-31
Kiran Kumar Vajja and Prabhakar TV (Indian Institute of Technology Kanpur)

WW-2E Pedagogy and Organization

Learning Patterns: A Pattern Language for Creative Learners II I-41
Takashi Iba (Keio University / MIT) and Toko Miyake (Keio University)

Metamorphosis - A Successful Organizational Change Management Pattern I-59
Madhup Jain, Ranjith Kutty and Raju Dani (MindTree Limited)

WW-3E: Human Computer Interaction

Analyzing the HCI Design Pattern Variety I-85
Christian Kruschitz and Martin Hitz (University of Klagenfurt)

WG-1E: Organization and Services

Research Organization Servicelization Patterns I-95
Yuriko Sawatani (IBM Research-Tokyo)

WG-2E: Network Systems

Adaptable Load Balancing I-99
Sung Kim and Youngsu Son (Samsung Electronics)

A Search for
a
Process getting “Quality”

Introduction

This short essay paper aims at two specific objects.

First is to deepen the understanding with regards to the “Quality without a name-so called QUWAN set forwarded by Prof.Christopher Alexander in the book”Timeless way of Building”. We are to set a hypothesis of this quality the same as in the two books using the same word for it ,”Zen and the art of motorcycle Maintenance” and “Zen and Archery”.

Secondly, identifying a drawing process of human face as a form of centering process which Alexander also set forth a process of Structure- Preserving Transformation Process and this process only assure you of the purpose quality.

It might be a surprising to you a little.

Hiroshi Nakano, Community planner

With the assistance of Bankoku Sasagawa, Rep.of ESRG

「クオリティ」を生成する
プロセスの追求

中埜博・笹川万国

2010・3・1

目次
「クオリティ」を生成するプロセスの追求

はじめに

「無名の質」

１ 「無名の質」とパタンランゲージ

２ 「無名の質とクオリティ」

３ 禅について

４ 禅の追求する「質」と「名付けられぬ質」

５ 有機的秩序と無名の質（QWAN）

６ 診断の原理

7 住民参加とワークショップ

「全体性（一体性）」

８ クオリティと一体性（全体性）

９ クオリティの実感

１０ 建物・環境からの「クオリティ」の喪失

１１ クオリティとパタンランゲージ

センタリング・プロセス

１２センタリング

１３センタリング・プロセス

１４センタリングの例（建築の場合）

１５「クオリティ」を産み出す「センタリング・プロセス」

１６この「プロセス」の実験にあたっての注意点

１７なぜ、このプロセスは「クオリティ」をうみだすのか？

結論

参考付録１

参考文献表

はじめに

本論文の趣旨は2つです。ひとつは、C.アレギザンダーの「時を超えた建設の道」の第2章「無名の質」(Quality without a name=QWAN)をより深く理解し、それとプロセスのかかわりを簡単に説明する実験のために書いたものです。

特に「無名の質」を「クオリティ」と呼んでいるロバート・パーシグの著作「禅とオートバイ修理技術」とオイゲン・ヘリゲルの「弓と禅」の”それ”との関連性を結びつけたのは著者の一解釈です。ふたつめは、パタンランゲージのめざすQWANを、「センタリングプロセス」というアレギザンダーの提案によって、だれにでも体験してもらおうというチャレンジです。そのチャレンジ方法が「似顔絵書き」であることは、読者にはショックであるかもしれません。

2008 英国のアレギザンダーの自宅裏庭にて 中埜

「クオリティ」をうみだすプロセスの追求

「無名の質」とパタン・ランゲージ

アレグザンダーは、「名付けられぬ質」の創造こそ、パタンランゲージによる建物、環境づくりの究極の目的と、第１巻「時を超えた建設の道」の第１章で明快に述べた。

この第１巻がパタン・ランゲージ・シリーズ３巻のうちでは、理論的解説、定義の本と位置づけられていたので、期待して読み始めたにもかかわらず面食らった人が多かったのではないかと思う。 パタン・ランゲージが、批判を受け入れ修正が可能な自由な発想に裏付けられ、機能的アイデアに満ちているのにもかかわらず、その目的である「質」の定義の曖昧さに驚いてしまったひとが多かったのだろう。またこの「質」という言い方も変わっている。

もちろんこの質を、「生き生きしたもの」「全体性をもつもの」「美しいもの」「正確なもの」などその他いろいろなわかりやすい言葉によって説明しようとする試みはなされてはいる。しかし、どの言葉も正確にこの質を特定できていないと結ぶ。そしてこの「質」をうみだすには、できるだけ生き生きした「パタン」を多用することと指示されているだけである。生き生きしたパタンとそうでは無いパタンと、どうやって区別すれば良いのかという疑問も残る。

最も、面食らうのは、この「生き生きした質」＝無名の質が、個人の「主観性」にゆだねられているところだろう。この名づけられぬ、定義できない実体が理解できない、あるいは、感じとることのできない者にとっては、ただ、自動的にパタンという、かたちのルールを、設計に適用していけば機械的に、そういう結果を持った環境・建築がうみだされるということなのだろうか。一体パタンとこの質の創造はどうつながっているのかについて、ほとんど、説明されていないのが問題なのである。

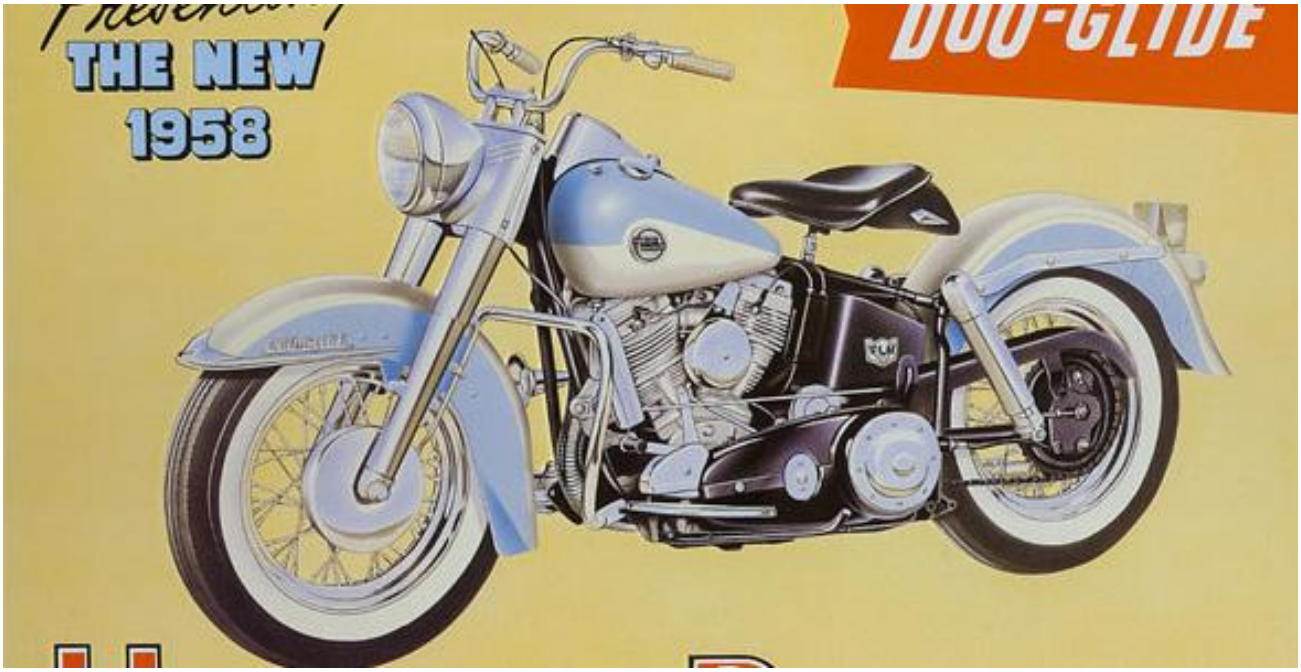
もう一つの問題は、「ランゲージ」ということばにもある。パタンを組み合わせて、ランゲージ化し、無限にストーリーをうみだしていく事と説明されている。これは、単語とセンテンスの関係と同じと例示しているが、「パタン」が単語と同じという論理的な説明もない。実は、パタンの合成は、目をとじて、頭の中でイメージ化することで、ストーリー化（ランゲージ）して実現するが、このストーリーづくりには、視覚的空間が絵本のように浮かんでいなければならない。ひとつのイメージにまとまっていくためには、パタンのボトムアッププロセスの理論化が必要である。このボトムアッププロセスを支配するのが、この「無名の質」なのである。（この説明は、センタリングの項目に書いてある） 同じパタンを使っても個人によってまったくちがうランゲージイメージがうかんでいるはずである。あくまで言葉としての合意がパタンランゲージの合意のことである。

実際の形はあくまで、現場（敷地あるいはそれが実現する場）でうまれてくる。

しかし、それにしてもこの質が共有されていなければ、プロセスが成立しない。ここに、「質」の理解を深めておかなければならない理由がある。

「無名の質とクオリティ」

この「無名の質」という言葉を、ロバート・パーシングの書いた「禅とオートバイ修理技術」（1974)という本で使われている「クオリティ」という言葉で言い換えてみたい。



「禅」について

パーシグの本は明らかに「弓と禅」という有名なオイゲン・ヘリゲルというドイツの戦前の哲学者の本のタイトルを踏襲している。ドイツの洋弓と弓道の違いを自ら体験して、その「道」という哲学がいかに日本文化に深くくみこまれているかを、西洋人の目と哲学で説明している興味深い本である。この本は、面白いことに　禅そのものには言及していない。この「弓道」という思想の奥義が禅に基礎を置いていると推定しているにすぎない。ヘリゲルは、鈴木大拙の「禅と日本文化」に深く影響をうけ、禅による「大いなる解放」という一種の神秘主義的魅力に牽引されて日本にきた哲学者ではあったが、日本の弓道が、禅とむすびつけるきっかけになるとは思ってもいなかったと述べている（著作の序言より）

禅の共通理解

そこで、この2つの本の禅に対する共通点を比較してみる。

共通なのは、自分と外世界の存在（バイク、自然、弓、矢、的、道場など）が、不可分に一体化しているという理解の仕方である。つまり、「世界は自己を含めて不可分なひとつであると言う理解」を、禅の本質の一つとみている。

それを、パーシグは「クオリティ」とよび、ヘリゲルは単純に”それ”といっている。そこには、両者とも、自分自身の「大いなる解放」の必要があるといっている。（外国人にとって、禅は、そういう悟りを得られる神秘的核心が有ると信じているようだ。）しかし、「弓と禅」はそう甘くない。その質を求めることが弓道の本質であるとはっきり書いている。「あの”それ”は肉眼には精神的であり、心眼には肉体的なのでしょうか一その両者でしょうか？それともどちらでもないのでしょうか。これらの凡て、すなわち弓と矢と的と私とが互いに絡み合っているので、もはや私はこれを、分離することができません。のみならずこれを、分離しようとする要求すら消え去ってしまいました」（p156弓と禅 1977版福村書店）

禅の追求する「質」と「名付けられぬ質」

禅の追求する質としての”それ”とパーシグの「クオリティ」の一体性の世界観が、「無名の質」と一致すれば、禅がここで共通に登場することも当然理解できる。また逆に禅ということばが、日本人にとっての、この質の理解と生成の方法を具体的に考える糸口となれるかもしれない。（つい最近のジム・コプリエンというソフトエンジニアーの講演の主要テーマであった。）

我々、日本人にとっても、禅はほとんど外国文化である。しかし。伝統として形を変え日本人の作法やマナーとして、禅は、ことばとしては語られないにかかわらず、「誰でもわかる、語る必要のない質」として存在はしている。ここで、私たちが、逆にラフカデオ・ハーンや、フランク・ロイド・ライトのように在日外国人と同じ目で、禅を見直してみると、「クオリティ」の理解に役立つ。しかし、この無名の質の説明はこれだけでは、不十分。（ネイチュアー・オブ・オーダーには、これは生命の構造であると定義している。）しかし、禅というキーワードによって、一前提条件を理解を深めておくのが重要。

この論文は最小限の知識で、クオリテイを作りだす試みだから、あらゆる説明は必要は無いと思う。

有機的秩序と無名の質（QWAN)

我々は、「オレゴン大学の実験」という3部作の3冊目こそが一番パタン・ランゲージの理論的、実践的な著作だと思っている。 この本の中の「有機的秩序」の生成原理が、一番「クオリティ」の理解に役立つ。

著者は「有機的秩序とは、部分の要求と全体の要求との間に完璧なる均衡が存在するときに達成される秩序のことである。1975」 これを、どうやって、実現するか？については、「この秩序の生成には、個別の部分を作り出す行為が、全体性をうみだすプロセスとなっていなければならない1975」としている。 ここで重要なのは、「クオリテイ」は、部分と全体の均衡状態と、ステップバイステップの動的「プロセス」の2つによって、産まれるという条件である。この条件は、アレギザンダーに言わせれば、自然の生成プロセスのことであり、生物でも、自然の動的成長すべてに発見できるとし、様々な例をあげている。（N.o.O 第1巻）自然発生的にできたまちの美しさの質の根源ともいっている。このように、有機的秩序とは、部分と全体の統合化をめざすことなのである。

もし、この仮定をうけいれるならば、禅が、自己という部分と、全体としての外世界の動的平衡状態になっているとき、”それ”が, 産まれるというヘリゲルの師の教えと一致する。また、パーシグの「クオリティ」も、精神と物質の2分論では、説明できない事を、前提としている。「「クオリティ」とは前知性的（感情的）な実在であって、「クオリティ」は、すべての母となり、あらゆる主観と客観の根源なのである上p91」と。この意味するところは、世界の一部である自己は、「クオリティ」という、知性的実在の説明では名付けることのできない上位の全体的実在であって、言語をもっては表わせない全体的実在としている。。（それは　老子の言う名づけようの無い「道」とぴったり同じ。アレギザンダーもここからの引用であった。）

だから、バーシグにとって、バイクの機械のシステムと自己と言う部分がその外世界とひとつの「クオリティ」のもとに、一体化している。（しかもこの「クオリティ」はギリシア哲学の確立までは、「徳」あるいは優れたものと呼ばれていた事と言われていたと指摘している。下p300）

（バイクの本でパーシグは、「クオリティ」の存在の発見に狂気し、次第に本当に精神の病気になってしまった。かれの「クオリティ」の発見が原因ではないと思うが）

診断の原理

この「クオリティ」と、QWANと”それ”が禅でいう世界の一体化世界観のことであるならば、それを、忘れてしまった近代人の意識のなかで、この再生・発見がどんな意味をもつだろうか？

これも、「オレゴン大学の実験」にそのヒントがある。

アレギザンダーは指摘する。近代人は、意識上の重い病気にかかっている。部分と全体との二元論の世界観に犯され、その知識がばらばらに断片化するという重い病気に犯されている。知識も、組織も、思考も、世界もすべてが、ばらばらな存在となっていると考えは、ついには、それをまとめる共感性も秩序もルールも、意識上で失ってしまったと。

しかし、生物の世界でも自然界でも人間社会以外の世界では、どこでも、秩序が存在する。それを、無視し破壊してきた人間は、今思い知らされている。自らの破滅すら導きかねない無秩序による地球破滅状況をうみだしてきている。人間社会は、この秩序を、法制度や、科学的あらゆる手段をこうじて秩序化しようと努力している。しかし、この秩序基準をどうやって、樹立し、破壊をくい止めることができるのだろうか。「オレゴン大学の実験」のなかでは、その問題を実際に観察して（＝参加のワークショップ手法によって）、自分の感情に「クオリティ」が働きかけてくるかという思考実験をすることで、その病気の重さの程度の判断を、まずしなければ成らないと提案している。これを「診断の原理」という。（「パタン」が存在するかということで診断基準としている。）

まず、ひとり、ひとりが自分の感情に忠実に成る事で。現実の世界の診断実験をすることからのスタートである。

住民参加とワークショップ

人間は　環境や、建物や世界のどこが病気に犯されていて悪いかを診断し、発見することができる能力を持っている。その診断基準は、　ちなみに、アレギザンダーの近著では、この質の存在を感じとることができる事を思考実験で証明する。その質があるか、あればそれはどれくらい強く、あるいは、どれくらい失われているかと。（これらは、デヴィッド・ボームの主張する「断片と全体」の著作の思想）「オレゴンの実験」ではまちづくりの問題であったので、コミュニティによる、集団住民参加のワークショップの実施の中て、これを具体的に診断マップ化する手法で、説明している。

アレギザンダーが1981年訪日時、　北海道札幌を訪れた時　禅寺の高僧との対話を行ったときの写真。

クオリティと一体性（全体性）

1974年に発刊された元哲学者であったパーシグの著作「禅とオートバイ修理技術」には、アレギザンダーの無名の質とほとんど同じ意味として、究極のオートバイ乗りは、この「クオリティ」を求めねばならないと書いた。彼はバイクの修理技術もバイク乗りのプロにとっては、一体として習得しておくべきものであると言い切る。 なぜなら、この技術が、バイクという機械に乗っている人間の満足、快適の源泉になるからだ。彼はそこで、バイクの取り扱い技術説明書自体がバイクの本質であるクオリティに触れていないことから、批判を始めている。バイクを一つのシステムと考えていくとバイクを構成するのは、それに乗る人間を含めて一体化して考えなければバイク乗りの満足と充実感が達成できないと彼は主張する。バイク乗りにとって、バイクツアーは、バイクという機械と人間が一体化して自然との一体感に触れることを楽しむものである。バイクの機械の調子、その排気音すべてが一体でかかわるものであるらしい。ここに、「クオリティ」を理解するヒントが有る。

クオリテイの実感

このバイク技術書をパタンランゲージに置き換え、無名の質をこの「クオリティ」と言い換えてみてもいいだろう。

一つの「クオリティ」存在証拠が彼の行った授業にでている。彼は2つの作文を皆に読み上げ、どちらの方がクオリテイが存在するかと問うた。明確なクオリティの定義を述べずに、ただ「どちらの作文のほうに、何か自分の存在を感じられるか？そういうクオリティの存在を感じるか？」と問いかけた、ある意味、実に危ない授業であった。しかし、驚いたことに、大多数が著者が質を感じる方と同一の作文に質を感じたのであった。この一体感を感じる「クオリティ」は、著者にとってのバイクの一体感と同じものであった。確かに言葉で定義できない質の存在を人間は感知することはできる。

では、なぜ今、わざわざこの質を、今、明文、定義化する必要があるのだろうか？

それは、人間が、建築の世界では特にだが、ここ100年、人間と一体化できない建物を大量につくりだすようになってしまったからだ。アレグザンダーの挑戦と、この失われたクオリティ、その存在意義すら顧みられなくなってしまったこの「クオリティ」追求のパ−シグの態度は同じなのである。

建物・環境からの「クオリティ」の喪失

「パタン・ランゲージ」という技術書を利用して作ったと自称する人々の作品が、あまりに混乱していることにアレグザンダーは気づいた。そこには「クオリティ」と呼べる秩序が存在していない。ほんとうに「人間との一体感」を伴った質を持ち合わせていない。アレグザンダーにとっては「パタン・ランゲージ」は、そもそもその「美しさ」を追求するものであった。それが究極の目的だったのだ。この「クオリティ」の存在こそ、古い歴史的な町のうみだす懐かしさや、快適さ、魅力の源泉なのである。パタンランゲージだけではまだ、それをうみだすのに不十分な事に気づいたのだ。

1980年「ネイチャー・オブ・オーダー（＝秩序の本質）」（以下N.o.O.とする）という4巻シリーズで、彼は、この「名付けられぬ質」の明快な定義付け（名付け）に取りくんだ。

クオリティとパタン・ランゲージ

N.o.O.によると、クオリティを確認し学ぶには2つの道が有る。

ひとつは、パタンランゲージを通じて、言い換えれば、15の幾何学的特性と呼ばれる形を通じて学ぶ方法である。アレグザンダーによれば、「クオリテイ」を静的に観察すると（止まった状態で観察すると）15の幾何学的特性の構造で分析できるという。

15の幾何学的特性はアレグザンダーの著作のパタンランゲージのパタンの図解ひとつひとつに発見できる。パタンの持つ有用性（機能性）は、この特性からうまれるとまでいう。この15の特性が、「全体性」をうみだす接着剤の役割をしている。だから、パタンのもつかたちで分類して学習することができる。

しかも、この特性は、パタン・ランゲージの「一体性」生成の保証となる。

（ひとつ注意：パタン・ランゲージは「全体性」の一部をくりかえし創造するルールであって、分析、診断、継承性には有効ではあるが、クオリティを確実に保証するモノではない。新しい時代の要請、つまり「クオリティ」を保証する、あたらしいパタン・ランゲージを創造して行く必要がある。）

一方、別の学習方法として、動的状態による学習がある。「クオリティ」は、プロセス構造の結果として把握できるという。

このクオリティの創造の動的プロセスを別名「センタリング・プロセス」と名付ける。このプロセスの結果「クオリティ」を産み出したかどうかを検定する。

今回のこの論文の目的はクオリティの構造の学習ではなく、その再生方法のプロセスを、できるだけシンプルに実験体験することだから、この学習方法に深入りすることを抜きに、まず、仮説として、承認してもらって話を進める。

※15の特性は巻末に参考資料として、パタンランゲージとの比較と2つの定義（機能とセンター）でまとめておいた。

センタリング

　プロセスの理解のための仮定として、「センター」と「センターリング」という言葉を再定義しておく。「センター」は簡単にいえば、15の特性の別名。パタンの源泉であり、クオリティを構造化する元型であり、重層構造化されることで、強い「クオリティ」をうみだす。「センタリング」とは、既存の構造にさらに「センター」を加えることで、重層化された構造に変容させること。基本的には、15の特性を持つトポロジカル（柔軟）な形で、既存の構造に働きかけ、重ね合わせて、複雑にならないように、シンプルではあるが、構造的には重く、人の心に強く呼びかけるものへと変化させる行為をいう。

センタリング・プロセス

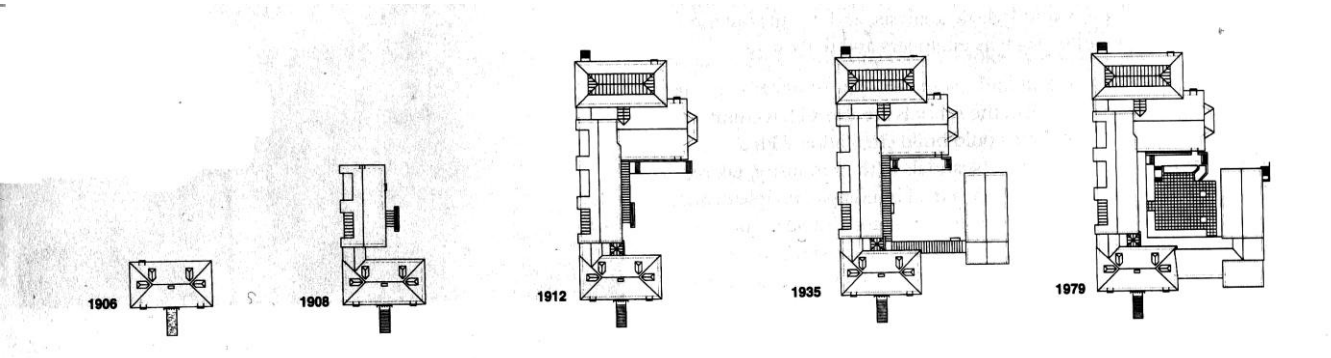
「クオリテイ」が何なのか、まだ解らないのに、それを、つくりだす。「15の特性」も「パタン」も「センター」も何も知らなくても、それができるプロセスを完成させたい、しかし、何百年か前の中世の時代にはそのプロセスにあたる何かが存在した。だから、あらゆるものに、強い「クオリティ」を持った建物や町並みは当たり前だった。近代は何を失ったのだろうか？今回はその簡易版として、「次に何をやるか？」をプロセスの一番のテーマとしたい。実は、「センタリング・プロセス」のエッセンスは、「つぎに、何をやるのがベストか？」なのである。

似顔絵かきのセンタリング

▲単純な円から始まる６段階の幾何学的変容
（“The Nature of Order – Book Two – The Process of Creating Life” より）

まず、上の図から説明しよう。この7つの図は一つの円が与えられた時、その円の構造に矛盾せず、むしろ円に対してもとの構造をより強化すると考えられる幾何特性（センター）を加えて変容させていくプロセスである。中心→部分シンメトリー→結合→境界→境界→中心といった幾何学的特性を重ね、付加していく。前の構造が次の構造を決定する。これは、円が持つ構造に着目し、授業で、グループ討論してから取り組んだ。結果はみな異なったが、変容のプロセスには驚く程何の問題もなく進めることができた。どの結果もとても美しく軽いクオリティを感じさせた。この変容プロセスが似顔絵書きの基礎となっている。このプロセスはまず、そこに何があるか？そこに何を加えると、その既存のセンターの特質を失う事無く強化できるか？それを、いかに、自然に、滑らかにおこなうか？である。

センターリングの例（建築の場合-バークレーキャンパス）



1. 教室　　2.サブシンメトリー（実習室増築）　　3.力強いセンター（展示室増築）　　4.対比、境界、中空（実習室増築）　　5.「正」の空間、境界不可分であること（図書館増築）

上記のセンターリングの例はバークレーキャンパスにおける、一つの教室（あだ名は箱船）の1906年から1979年までの、変遷過程である。（”How Buildings Learn”（1994)より）これに、センターリングとして、考えるとどんな特性が加えられていったかを我々が、注釈を加えたものである。このように、各年代を経て、センターを付加して行く事がセンターリングの基本である。センターを付加することで、前の段階の建物の機能を強化し、センターの特性は増加してゆく。しかも、その段階での全体性は損なわれていない。それ以前の全体性も保全されている。まさに、「次に、何をするのがベストか？」が問われる。

似顔絵書きについても同じことである。

「なにを次にするか？なにが、必要か？それによって、どんなセンターがうまれ、全体性はどう、強化されたか？どの特性（センター）を用いるか？」つねにその時点で必要とされるベストの「センター」を組み込んで行く。そういう「プロセス」に、ボトムアッププロセスの成功、つまり、最終的な「クオリティ」の生成が実現する。。これが「ネイチュアー　オブ

オーダー」の第2巻のエッセンスである。このプロセスは自然界の変容プロセスから学んだものである。

自然界の秩序生成プロセスからまずはその「クオリティ」の創造を仮定する。

「クオリティ」を、産み出す「センターリング・プロセス」

「センター」という言葉をアレギザンダーは使って、「クオリティ」という、全体性をうみだすプロセスを提案している。

センターとは、パタンの「かたち」の源泉である。また、ひとつの全体性（一体感）をもつ形の構造のことである。

世界がパタンという繰り返しに起こってくる問題解決のルールでできていることははっきりしている。そういう伝統的ルールを組み合わせる事で形をことばとして、表現し、建築や町をつくりだすこともできるし、実施してきた。このかたちを言葉であらわすことが、「パタンランゲージ」の意味である。そして、そういう古い建物や町のよさをくらべたり、解析することもできる。

しかし、いわゆるパタンランゲージによる建物やつくりだされたものが「クオリティ」を持つものに成るかの保証はまったくない。事実、多くのその後、作り出されてきたいわゆる、パタンを使った建物だと主張する新しい建物が「クオリテイ」をもたない例をアレギザンダーは沢山見るにいたり、がっかりする。パタンランゲージはそういう「クオリティ」をうみだすことが目的であったのに。この欠陥をうめるには、ボトムアップのプロセスの定義が必要なことに彼は気づいた。

つねに全体性（クオリティ）を産み出すボトムアップのプロセス―これこそ究極の目的である。「パタン」が重合化し、単純な形となることで、全体性を産み出す合成プロセスはないか？

（実は；この問いかけに対する彼の答えが「ネイチュアー・オブ・オーダー」という「パタンランゲージ」から25年後の著作のなかにある。）この著作の中で、パタンを重合化し、単純な幾何学的形に特化し、全体性をうみだす関係性を、「センター」（特性のこと）と名付けた。

パタンは「センター」によって、最終的には分類される。 センターは、元の構造を保全しながら、新しいセンターを産み出すプロセスもうみだす。これを。正式には、「中心構造保全変容プロセス」（Structure Preserving Transformational Process） と名付けている。

これが「センターリングプロセス」のことである。

このセンターリングプロセスによって、どうやって、「クオリティ」を生成できるのだろう。

「クオリティ」をうみだす15の接着剤

簡単に説明する。世界との一体性をつくりだす「センター」には15種類ある。これが15の特性。この15の特性は、接着剤のようにそこに存在する「センター」をくっつきあわせる役割を果たす。 この接着剤も、パタンと同じように名前がつけられている。（付録別表参照）この15の幾何学的特性は、つねに一つの全体性として存在し、べつの特性が付加されると、それでもその段階ごとに新しい全体性（センター）という平衡状態をうみだす。

このこみいった抽象的プロセスを、なんとか、わかりやすく体験する方法はないだろうか

と、探しあぐねてたところ、ひとつの絵の書き方の本に行き当たった。画家であり先生である松本キミ子さんが、だれでも、楽しみながら絵を完成させるおもしろさを探った結果、「キミ子方式」という絵の書き方の本を完成させた。この絵の書き方は、部分から全体をうみだし、各プロセスごとに完成（センターづくり）するセンターリング・プロセスと極似している。 これを使って説明できるのではないかと着目した。

ここでは、その理論説明ぬきに、ただその特性の使い方にのみ限定して説明していく。

ここには、アレグザンダーの結論として、「クオリティ」はひとつの構造であるという前提がいきている。「クオリティ」はどんなスケールでも存在する。しかし、その構造がしっかりしているか、特性を量的に多く持っているかによって強弱は存在するという。

私たちが、この「クオリティ」をうみだすプロセス実験をするにあたり、簡単にいくつかの結果としての現象をのべておく。

1。いまからつくりだしたい「名付けられぬ質」を一言で「クオリテイ」と呼ぶ。この意味は、言葉では説明しにくいが「全体性」を持ったひとつの柔軟な一体性を持った実体を意味する。

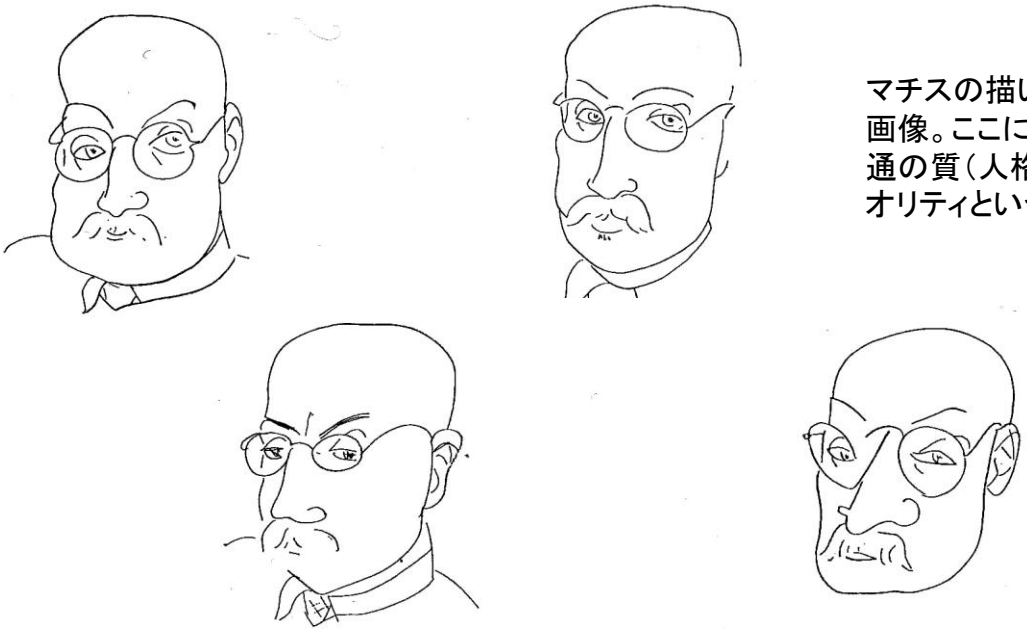
2。この「クオリティ」は、感情に深く呼びかける力を持っている。だから見ることができるような感情を観察主体にうみだす。

3。この「クオリティ」は、主観性と客感性の相互作用で生じるリアルな存在であるが、そこには強く感じる場合と弱く感じる場合とがある。

4。この「クオリティ」をうみだす実験に用いているプロセスは非常に単純化してあり、松本キヨ子氏のうみだした松本式絵の描き方のプロセスに大きく影響をうけ、利用させてもらっている。

5。この「柔らかな実体」という言葉は、ある特別な形を持ちながら、多様に変化し、つくりだす人の個性も加わり柔軟性を持っているという意味である。

6。「クオリティ」がうまれてくると、自分の書いた他人の顔を見るとき、自分自身を見ているような感情を受ける。



マチスの描いた自画像。ここにある共通の質（人格）をクオリティという

この「プロセス」の実験にあったての注意点

１。前ページのマチスのかいた顔のスケッチを見ると、どれも少しずつ違う。しかし全体としての顔は明らかにどれも、マチスである。どんなテクニックをマチスは使っているのだろう？いや、一体どうすれば、違った表情でも同じ人間の顔に見えるように描けるのだろう（感じるのだろう）？それには、次のような注意点が必要である。

2まず、このプロセスでは、各ステップで部分と部分の関係性が、どの段階でも決して狂わないように描くことが絶対条件である。（部分自体をよく観察する。見ることが大切）

３。そのステップの作業が、必ず前のステップで完成した部分の関係性とずれないように、さらに、次の大きな関係性を付加することを、強く指示する。

４。関係性の大きさ自体のサイズは、しだいに大きくなっていくこと。

５。部分として、一ステップで必ず一つの「全体性」と呼べる部分が完成している。（鼻、目，顔かたち）

６。部分としての完成度と、次のステップでの付加部分の完成度の正確さがクオリティの強度（構造性の強さ）を決定する。

７。部分の選定は、「センター」となっているかが基準である。顔の場合おおざっぱには3つのセンターでできている。（A：鼻とその周辺、B：口とその周辺、C：顔の外形線と頭髪）この3つを結びつける特性は各プロセスごとにメモした。特に不思議なのは、A：鼻が「センター」の「センター」で、第1ステップとなる。（体毛も鼻に向かって生えている。）

８。明らかに、顔形は部分の関係性の正確さによって、その「クオリテイ」を生じさせる。このクオリティによって、その人に似ているとか、人格が感じられる。

９。プロの似顔絵描きは「センター」を強調し、クオリティの一部を強調するテクニックに長けている。しかし、上記のプロセスの方が、クオリティは強い。たとえ本当に似ていなくても。

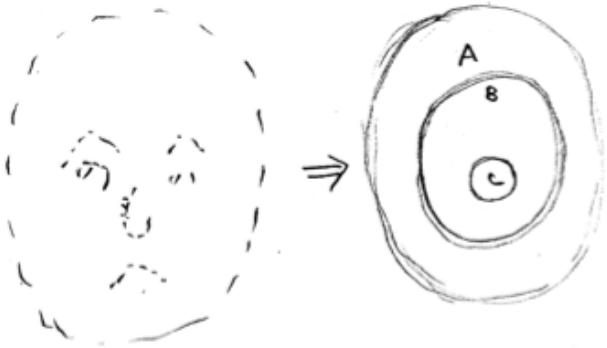
!0. この 「似顔絵かきのぷろせす」は、対象とその結果がはっきりしているので、実験には向いている。しかし、この実験の一番の障害は、描く側にある。似ているように描かなければならないと申し訳ない、自分にはこんな才能はない、など、描くことの本質には関係ない自我が邪魔をするのである。似顔絵の難しさは、テクニックではない。（テクニックは簡単に指示できる）この自我が入ってこないようにするのがファシリテーターの役割である。

「センタリング」で「似顔絵」を描くプロセス

「センタリング」を体験しよう！　⇒　「プロセス」を必ず守ること！

●用意するもの・・・画用紙、クレヨン

＜１＞頭の中で顔のラフなサイズを考えてみる



↑頭の中に描かれた曖昧なイメージ

↑だいたい上図 A, B, C のような
３つぐらいの大きさの感じがあることを
つかんでみる

スケールの段階性

大：中：小　＝　３：２：１（直径）ぐらいの差がある

＜２＞鼻からスタート

①まず、顔全体を見ずに、その真ん中の鼻、
そのまた先端の鼻の頭だけを見てみよう

②それだけでは少々捉えづらいので、左右の
小鼻も一緒に観察してみると、形が見えてく
る

③それでもまだ難しければ、鼻の穴も一緒にしてよく見てみよう

やっと見えてきた
鼻だけだよ
小鼻が丸いか、穴が大きいか、横に広いか、
見えにくいか、その部分のみに集中して
それだけを描き取ろう

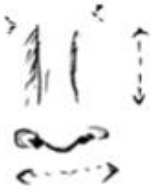
力強いセンターの発見

④さらに気をつけて見てみよう！
鼻筋や鼻の下の溝や笑いジワ・・・
いろんなものが見えてくる

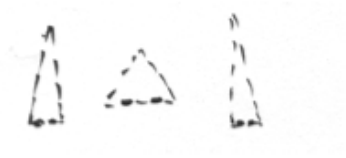
基本的にはラフなシンメトリーになっている
穴は方向のちがうシンメトリーをしている
→ ローカル（局所的に現れる）・シンメトリー

⑤次に鼻筋を描こう
まずは左側
下の小鼻の広がり幅をよく見て、
それに合わせて長さを決めて、
目の端が少し見えるところまで

次に右側
必ず鼻の長さに合わせて、
鼻だけを完成させる

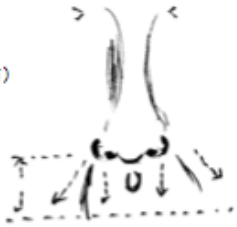


鼻というのは、ひとつのまとまった形に見えます
この形をはっきりと見分けなければなりません
（どんな形でもその人らしい形があるのです）



良い形
力強いセンター

⑥次に、鼻の下の溝の長さ、形をよく見て描いてみよう
さらに、笑いジワを見つけて形を正しく捉えてみよう
広がっているか、縮んでいるか、ほとんど見えないか（左右）
全体をよーく見ながら描いてみよう



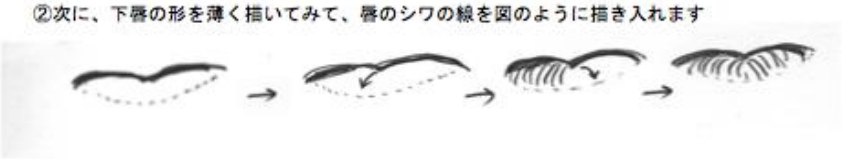
相互反復
ローカル・シンメトリー

これでまず、顔の大切な部分（第一段階）が完成します

<u>＜３＞次に、口に移ります</u>
①まず位置を決めよう
すぐ上の鼻が大きい役割を果たします
鼻筋、笑いジワ、鼻の下の溝との関係を見ながら、
薄く描いてみてから、濃く描きなおします
口の割れ目　→　真っすぐではありません

鼻と口が笑いジワと溝で
（深い）相互結合されます

②次に、下唇の形を薄く描いてみて、唇のシワの線を図のように描き入れます



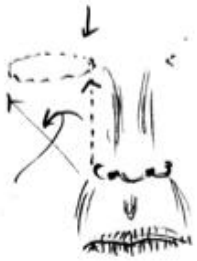
③次に、上唇の形を当たってから、同じように線を描き入れていきます

常に口の真ん中から描き始め、下、上の順で完成させます
ここで気がつくのは、口の割れ目は太い線で、唇はシワの線でできていることです。
そして必ず鼻の大きさ、シワの大きさに対応していなければなりません。

力強いセンター

<u>＜4＞目を描いてみよう</u>

左の目頭の位置から
①、②と、目の形をよーく見て線を描いていきます
①の線をどのあたりまで延ばすかが大切
小鼻の位置との角度に注意して決めましょう
Aの角度をよく見ながら①を描きましょう
①が②にかぶさっていきます



目の球は真ん中の黒い部分から描き（③）

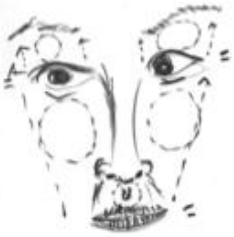
次にその周りを描く（④）

①を濃く描く
右目も左に合わせて同じ要領で描く

目の傾きで顔全体の流れ（方向）がうまれます
目と同時に眉毛も描きたくなりますが、目を描き終えるのが先です

良い形
ローカル・シンメトリー
共鳴

＜５＞まゆ毛
目、鼻筋、唇との関係すべてを比較して
形をつかんでください
内側から外側へ、毛を１本１本描いていくこと
この段階で、目尻のシワ、目の下の部分、
口の端の部分等を描き加えていきます
その人らしい繰り返しが出てきます



頬や目の間、目とまゆ毛の間等、
それまですき間だった部分が、次第に
正の（ポジティブな）形（空間）に
なってきます
共鳴

＜６＞顎
まず顎の先端ａから描き始め、ｂ，ｃの順に
形を読み取って描く。
全体としての輪郭を切り取るということが大切

正の空間
良い形

<u>＜７＞耳と首</u>

①左耳の一番手前に出ているところから
　外側へと描いていく
②次に右耳も同じ要領で

③首の両端の高いところは口の高さとおなじぐらい
　首は意外と太い

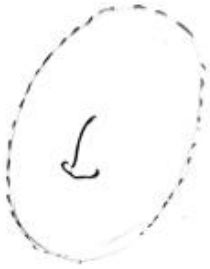
＜８＞髪の毛
つむじから毛を１本１本描いていく
端の方も、つむじから出ているように描く
帽子を描くような感じで
頭の形だけでその人とわかるように描く

深い相互結合

●以上のプロセスに従って実際に描いてみると、どんな人が描いても必ず個性が出てきます。そして、いつも普通に描いている絵とは違ったものになっているはずです。それは、描かれている人の個性が強く出てくるからです。この個性を「全体性」と言います。この個性は部分からはうまれてきません。全体として不可分な互いの関係からうまれてくるのです。この「個性」という「全体性」は、部分と部分の関係性（幾何学的特性）の組み合わせによってうみだされるのですが、これはプロセスによってコントロールされなければならないのです。

実験　:マチスの顔もこのプロセスで書いてみよう！

①大体の方向を決めます



②特長のある鼻を描きます（鼻先、小鼻、鼻筋・・・）

力強いセンター

ローカル・シンメトリー

③目の位置（目頭）を決めます

良い形

共鳴

④笑いジワと目の方向を決めます

共鳴

正の空間

⑤口と口髭を描きます

相互結合

⑥まゆ毛と顎の流れ（輪郭）

正の空間

⑦眼鏡と頭の形

不可分

なぜ、このプロセスは、「クオリティ」を産み出すのか？

今の段階で考えられる我々の仮説をのべる。

「クオリティ」とは、「ネイチュアーオブオーダー」の定義によれば、

・自分も含めて、対象物との「一体感」を産み出す「全体性」のことである。

・そこには、「我」が存在しない。個人のエゴとしての意図が消えている。（普遍化）

・15のセンター（特性）が発見できる。それは入れ子状態になって、お互いを支え合い強化しあって重複している。そのような形の構造になっている。

・「感情」に呼びかける何かの存在がある。（人格を感じる）

といった性質をもったものである。

そこで、改まってこの「似顔絵書き」のプロセスにはつぎのように、上記の特質をうみだしていることがわかる。

1。まず、このプロセスのマニュアルは部分から全体への描き方になっている。書きかたは、部分をえがいて、全体へとすすむ。常にそこに描いた部分を見ながらフイードバックして次ぎを描くプロセスになっている。。これはフィードバックではなく、じつはフィード・フォワードになっている。（実はボトムアップは、まず部分から全体へ進む）

その時点、時点で完成しており、その部分でもって、対象は、理解できる。そして結果的に完成したものからの自分の反応には、自分の書き方とその対象の観察によって得た情報との合併が起こっており、その書き手の個性ぬきには描き得ない一体性を持った絵となっている。、

2。その段階に集中することで、「自我」（結果を予想する希望）が自然に喪失する。また、そのように、ファシリテーターが誘導する。

3。各段階で、必ずなんらかの幾何学特性特性を導入して、書かせている。ただ、人間の顔は、その特性の依存関係でその個性が表現されているから当然のことである。顔の表現はセンターの組み合わせそのものだから。

4。結果としての、一体性（全体性）は、モデルによって、他人にも確かめられる。その個性が写し取られている感情的反応を確認、判断できる。

5。産み出されたものには「構造」があり、それは、われわれが抑圧している「本当の自分を解き放つ」（展開）結果となる。どんなプロセスもそれが究極の目的であるが、とくにこの絵の書き方の技術を開発した松本キミ子さんに負うものである。まず技術的開発が先にあると思える。

結論

「無名の質」が、かならず実現できる普遍的プロセスは果たして存在するのだろうか？
この似顔絵描きは、すでに完成した方法の後づけにすぎないのではないか？
この方法が、たまたま巧く行くのは、まったくべつの理由ではないか？

様々な反論があると思う。しかし、読者にはさらに、幾何学的特性について、その膨大な裏付け書類、データを。原本「ネイチャーオブオーダー」にあたって、検討、確証を得る事を望む。

一つ確かなことは、この「クオリティ」を察知する能力はほとんど誰にも有り、ほぼだれでも、一致する。特に職人社会が健全であった中世のものづくりは、この「クオリティ」の追求以外に目的がない。これが無ければ、ものづくりの進歩の基準が存在しなくなる。

そして、もうひとつ何にしても. このプロセス自体の実施過程が楽しい。「センターリング」は大変努力のいるプロセスだが、その楽しさは共通である。特に、住民参加を必要とすルプロジェクトには欠かせない条件だと思う。キミ子方式の絵の書き方にもそれは共通である。

このセンターリングこそボトムアップ・プロセスの基本であることが新時代の科学と哲学によって現在のアインシュタイン・ボーアパラダイムを超えるパラダイムに取り替えられ「クオりティ」創造にかかすことのできないものであることが証明される日がくることを、切に望むものである。

基本的な考えとして、クーンのいうパラダイムシフト（転換）のステップというのは、その理論を共有するグループ・コミュニティが実践的、実用的に理論を使用する時、簡単かつ単純に使える理論の方が、その便宜性ゆえに、多数の人がそれを使って問題の解決に当たるようになるのが第１ステップ。数として、その理論を使用する人が多数派となってそのパラダイムが、勝利したように見えるのが第２ステップ。しかし、その時点でも、理論闘争のレベルの勝負はついていない。それは、最後の裏付けとしてついてくるものであること。それが第３ステップだそうである。（科学革命の構造　トマス・クーン）

だから、まずプロセス」として、容易に「クオリティ」が産み出せる実践技術の開発とその実践を重ねることがいま我々がとりくまなければならない第一ステップであろう。

参考付録1

15の幾何学的特性のまとめとパタンによる例示

ここに、アレギザンダーの「ネイチュアー・オブ・オーダー」の著作から。15の幾何学特性（センター）の簡単なまとめを提供する。

まず、まとめはできるだけ、第1巻の本文にそって、できるだけ忠実に、わかりやすくいいかえた。簡略にしたのは我々の責任であるから、解釈間違いは我々の責任である。

まず、このまとめは、元の説明が、長大であるので、3点にしぼりこんだ。

まず、第1は、ひとつひとつの幾何学特性のもつ「機能」的側面にのみ着目しまとめた。しかしこの場合「機能」という意味を拡大して機能の関係性（感性も含める機能）にまで拡大解釈した。（「機能」の意味が異なるので括弧を付けた）（※）例えば、「境界」の存在によって「センター」が強化され、それが美しくなるといった言い方。

第2は、「センター」は、パタンランゲージのインフラ構造（源泉）であり、視覚的形体として「センターリング」の中で、必要な変容プロセスにもなる。その「センター」の変容プロセスをまとめた。

第3は、この上記のまとめを、既存のパタンの中からその関係性が一番表現されている例をあげた。パタンひとつひとつについて、上記の2つのまとめの説明は省略した。

また、似顔絵プロセスのなかで付加された特性の名前をプロセスごとにいれた。

（※）：機能とはいろいろな解釈が可能だがデカルトの機械主義的解釈によると、あるインプットに対してある条件下で、一定のアウトプットを常に提供するブラックボックスのことである。そのブラックボックス自体の関係性までは、デカルトは深く追求していない、15の特性は、各「機能」のブラックボックスを結びつける（接着する）可能性をもっている、（重複するものもある）その点を、「センター」とその機能を「センター」によって関係づけるとしたのが以下のまとめである。

参考資料（N.o.O.より）

★「15の幾何学的特性」と「253のパタン」の相関一覧

	15の幾何学的特性 ―「The Nature of Order Book One：Phenomenon of Life」（C.アレグザンダー著、CES出版刊）より ―	253のパタン ―「パタン・ランゲージ」（C.アレグザンダー他著、平田翰那訳、鹿島出版会刊）より ―
1	局所的に現れるシンメトリー	レース状の田園道路（5）、段階的な動線領域（98）、プロムナード（31）、連続住宅（38）、屋内空間の形（191）
2	スケールの段階性	半インチの見切り縁（240）、小割りの窓ガラス（239）、自立地域（1）、7000人のコミュニティ（12）、見分けやすい近隣（14）、段階的な屋外空間（114）、アルコーブ（179）、ベッド・アルコーブ（188）、天井高の変化（190）
3	力強いセンター	おも屋（99）、炉火（181）、中心部の共域（129）、南向きの屋外（105）、日のあたる場所（161）、隅のドア（196）、都市の魔力（10）、小さな広場（61）、小高い場所（62）、ほぼ中央の焦点（126）、木のある場所（171）、親密度の変化（127）
4	境　界	サブカルチャーの境界（13）、近隣の境界（15）、厚い縁どりの枠（225）、アルコーブ（179）、窓のある場所（180）、厚い壁（197）、部屋ざかいのクロゼット（198）、外廊（166）、アーケード（119）、戸外室（163）、コミュニティ活動の輪（45）
5	交互反復	フィンガー状の都市と田園（3）、平行道路（23）、歩行路と目標（120）、大物倉庫（145）、段状の斜面（169）、部屋ざかいのクロゼット（198）、人と車のネットワーク（52）
6	「正」の空間	正の屋外空間（106）、屋内空間の形（191）、あいだの家（48）、見えない駐車場（97）、敷地の修復（104）、一間バルコニー（167）、窓のある場所（180）、天井高の変化（190）
7	良い形	歩行路の形（121）、建物の正面（122）、柱のある場所（226）、屋根飾り（232）、屋内空間の形（191）
8	深い相互結合と両義性	フィンガー状の都市と田園（3）、工業の帯（42）、光の入る棟（107）、どの部屋も2面採光（159）、アーケード（119）、街路を見おろすテラス（140）、いっぱいに開く窓（236）、格子棚の散歩道（174）

9	対　比	モザイク状のサブカルチャー（8）、仕事場の分散（9）、生活空間にしたがう構造（205）、明暗のタピストリー（135）、明かりだまり（252）生活を見おろす窓（192）、炉火（187）
10	段階的変容	カスケード状の屋根（116）、補強柱の配分（213）、密度のリング（29）、中心をはずれた核（28）、親密性の変化（127）、入口での転換（112）、公共度の変化（36）
11	粗っぽさ	自然なドアと窓（221）、ほぼ中央の焦点（126）、野生の庭（172）、柔らかいタイルとレンガ（248）、禅窓（134）、見えがくれの庭（111）、重ね張りの外壁（234）、床・天井ヴォールト（219）
12	共　鳴	見分けやすい入口の集まり（102）、順に固める構造（208）、どの部屋も2面採光（159）、自分を語る小物（253）
13	中　空	聖地（24）、水への接近（25）、泳げる水（71）、車座（185）、入浴室（144）、開かずの間（204）
14	簡潔さと静謐さ	緑路（51）、池と小川（64）、低い窓台（222）、柔らかい内壁（235）、キャンバス屋根（244）
15	不可分であること	仕事場の分散（9）、どこにも老人（40）、自主管理の作業場とオフィス（80）、つながった建物（108）、通りぬけ街路（101）、戸外室（163）、車との接続（113）、つながった遊び場（68）、人前の居眠り（94）、柔らげた光（238）、いっぱいに開く窓（236）、大地へのなじみ（168）、さわれる花（245）、すきまだらけの舗石（247）

＊各特性に対応するパタンの並びは番号の若い順になっていないが、これは The Nature of Order Book One 第5章中の Functional Notes 内における登場順に準じたものである

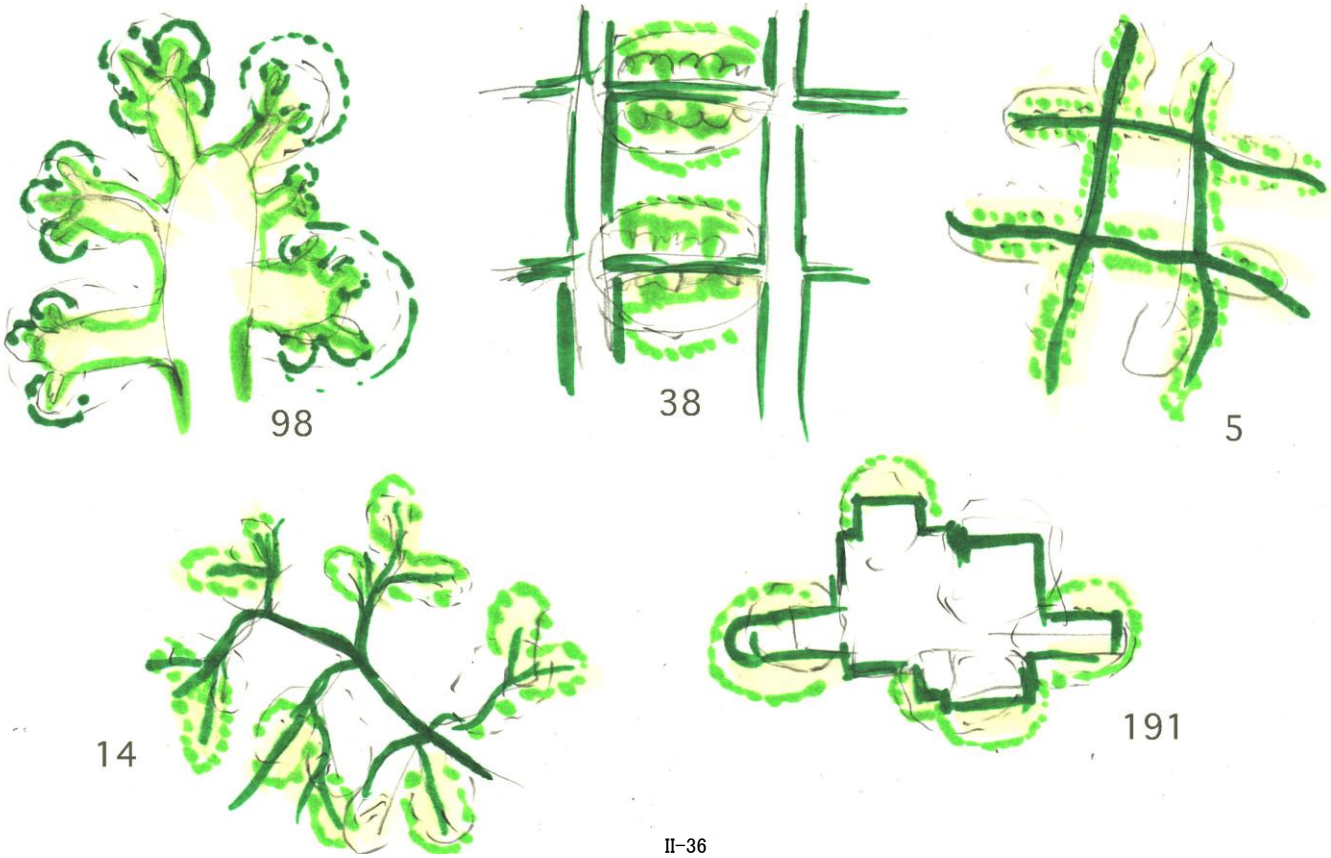
1

局所的なサブシンメトリー

・「機能」としては、最小限エネルギーで最小限作用原理の機能をうみだすこと。

・「センター」としては。サブシンメトリーの数の多いほど強化されるということ。

例　98　段階的な動線領域　38 連続住居



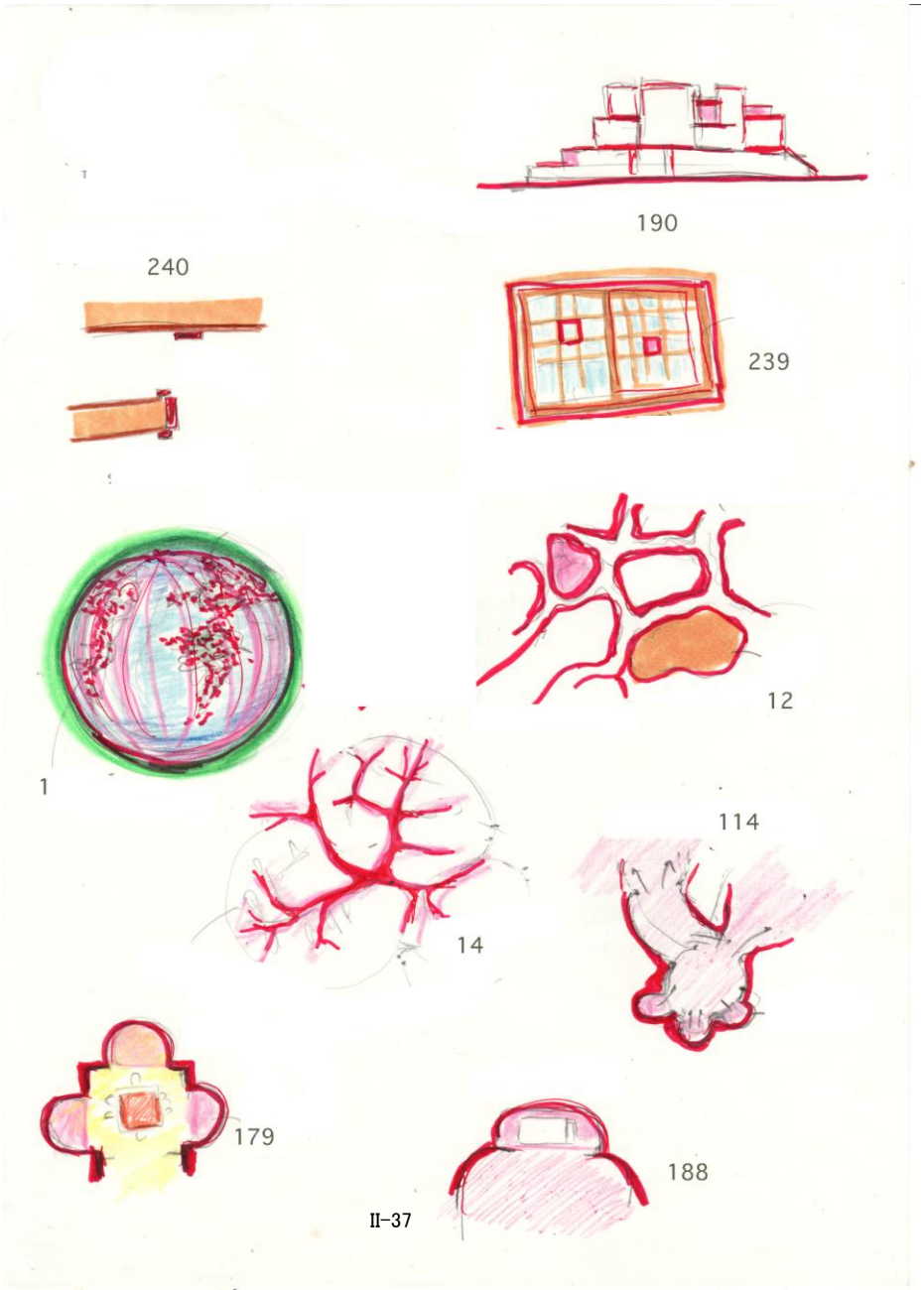
2

スケールの段階性

・「機能」としては、その機能分布に階層性が存在するということ。

・「センター」としては、その内部、外部に同時に補強するセンターをもつということ。

例 114 段階的な屋外空間　14 見分けやすい近隣　　2 町の分布



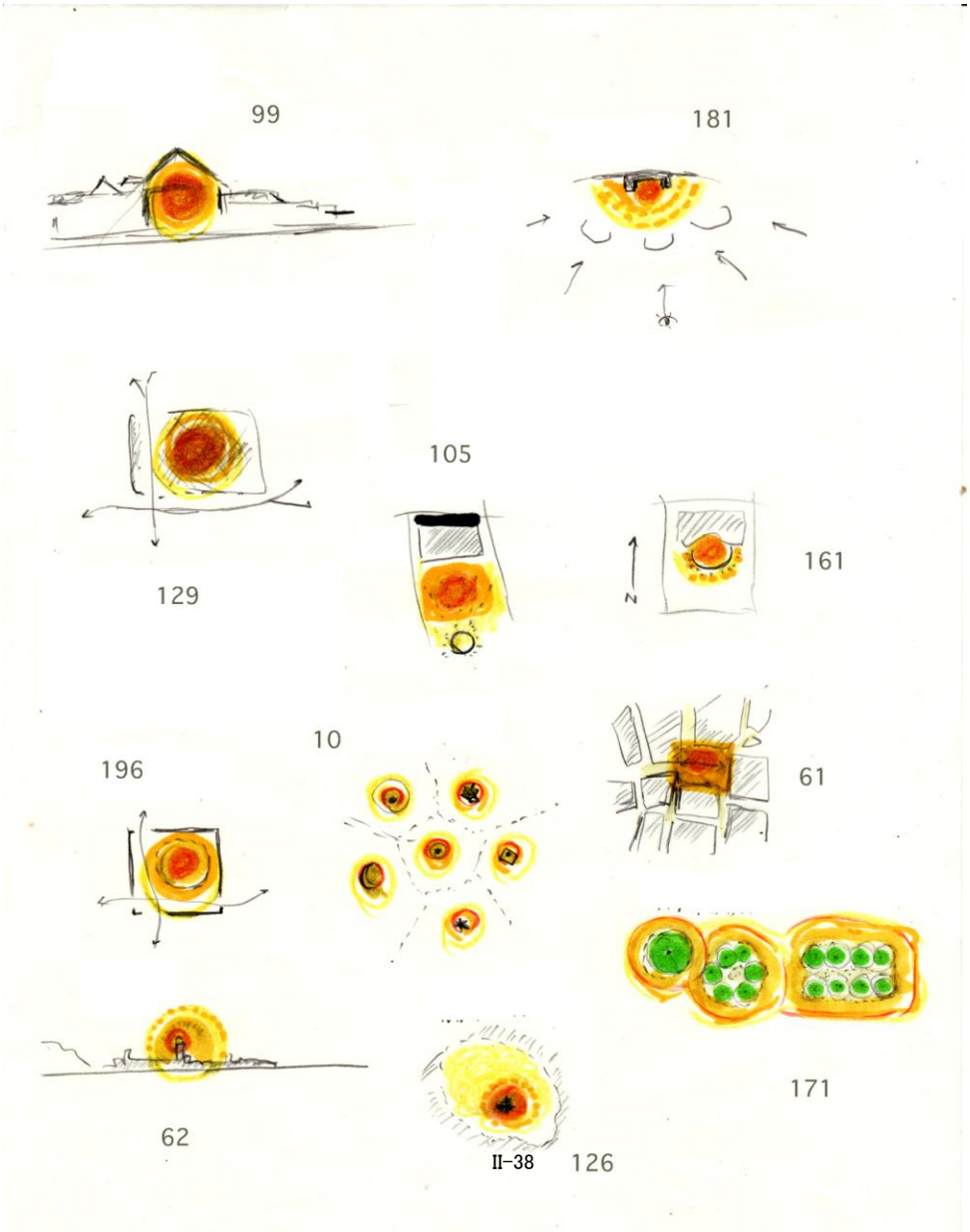
3

力強いセンター

・「機能」としては、各機能の構造化の純度が高い部分があること。。

・「センター」としては、そこに存在する「場」の力とよく合致していること。

例　99 おも屋 129 中心部の共域　62小高い場所



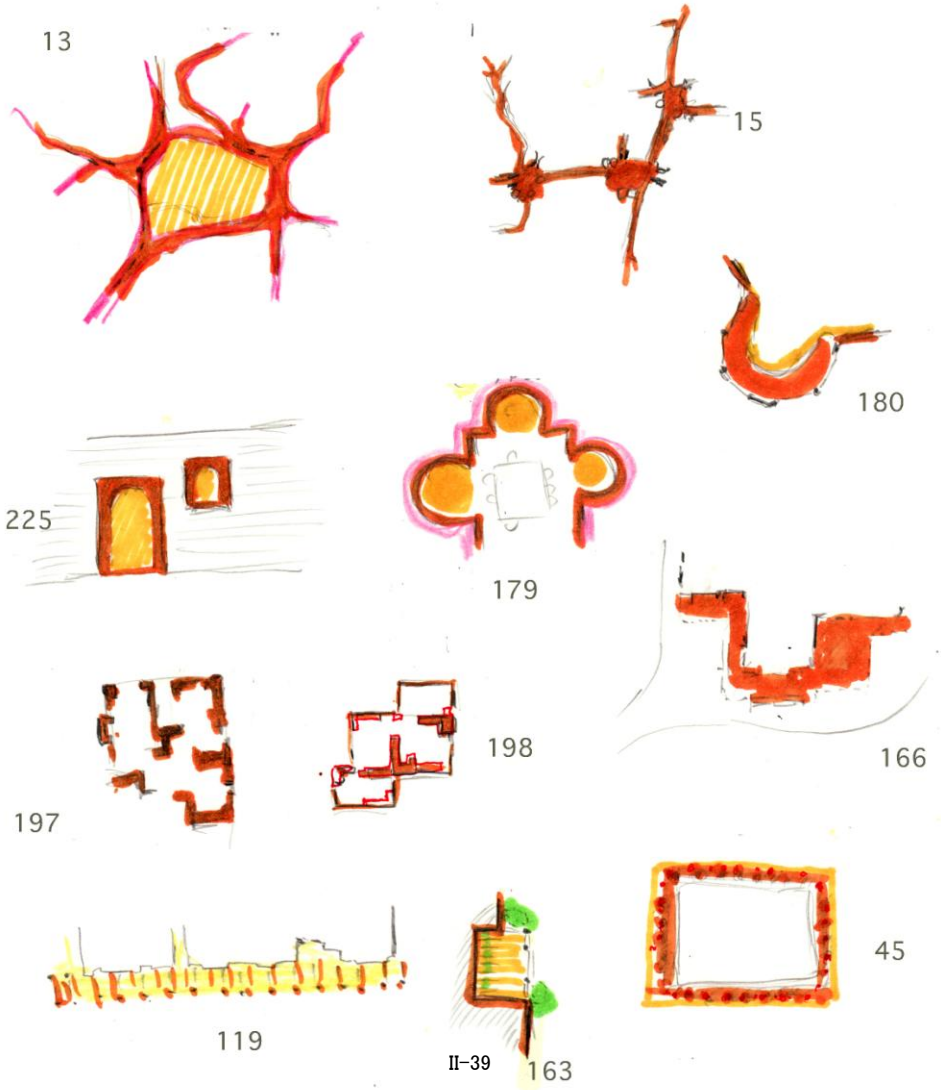
4

境界

・「機能」としては、システムとシステムの間には、その分離と。通過帯が存在する。

・「センター」としては、「センター」 の内、外を結びつけ、両方を強化する帯のような「センター」が中間に存在するという事。

例　13 サブカルチャーの存在 166 外廊



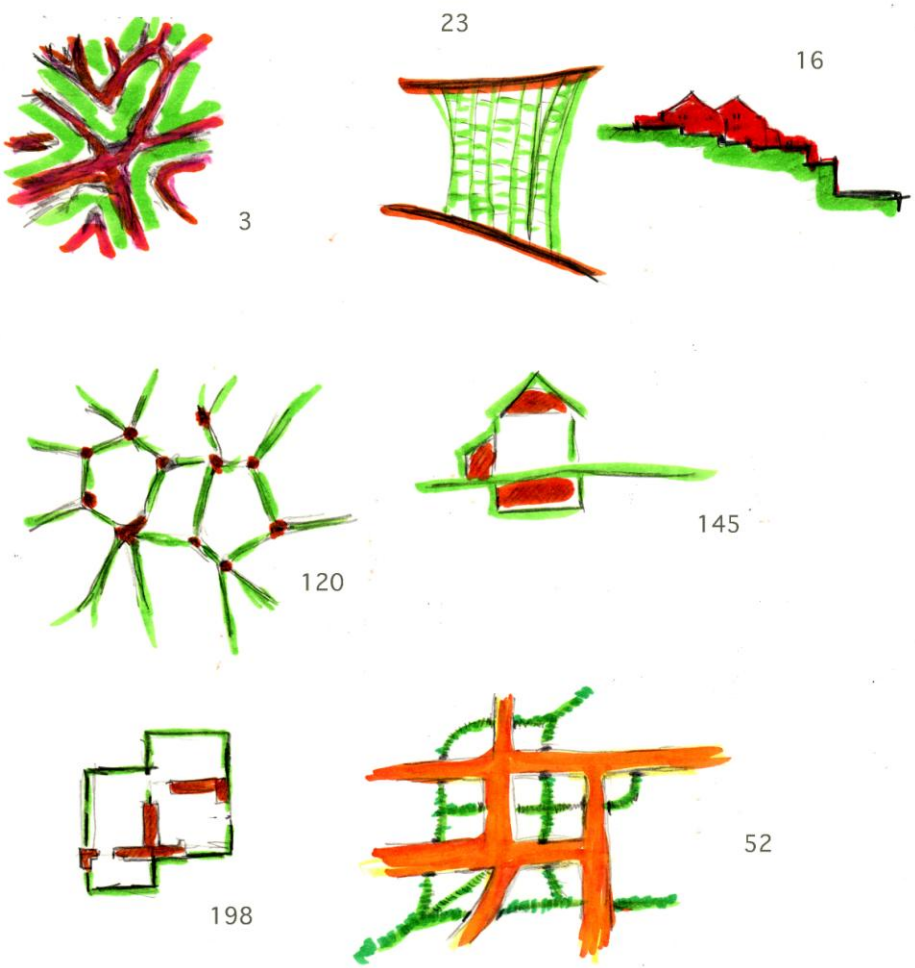
5

交互反復

・「機能」としては、相似的機能が交互に繰り返されること。

・「センター」としては、それが交互反復することによって、強化し合うということ。

例　3 フィンガー状の都市と田園　52 人と車のネットワーク



II-40

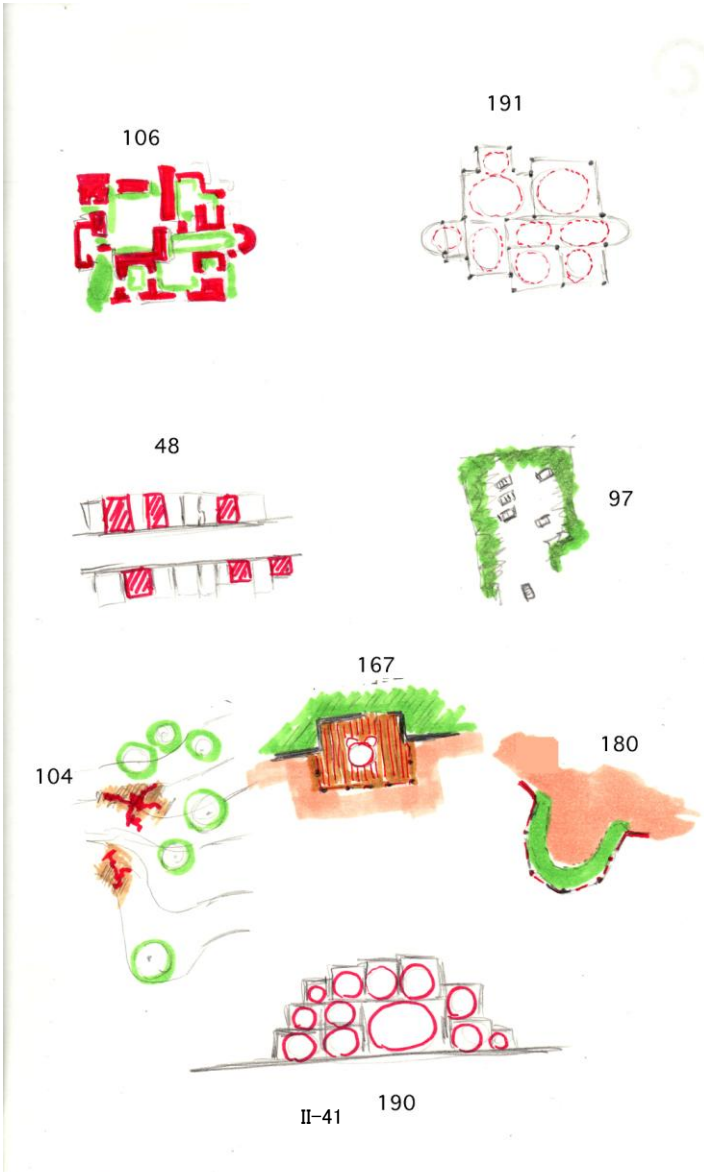
6

「正」の空間

・「機能」としては、機能同士の重なり合いも。機能となり、連続性を形成すること

・「センター」としては、隣接する「センター」が補強し合いながら、連続結合している状態

例　106 正の屋外空間　104 敷地の修復



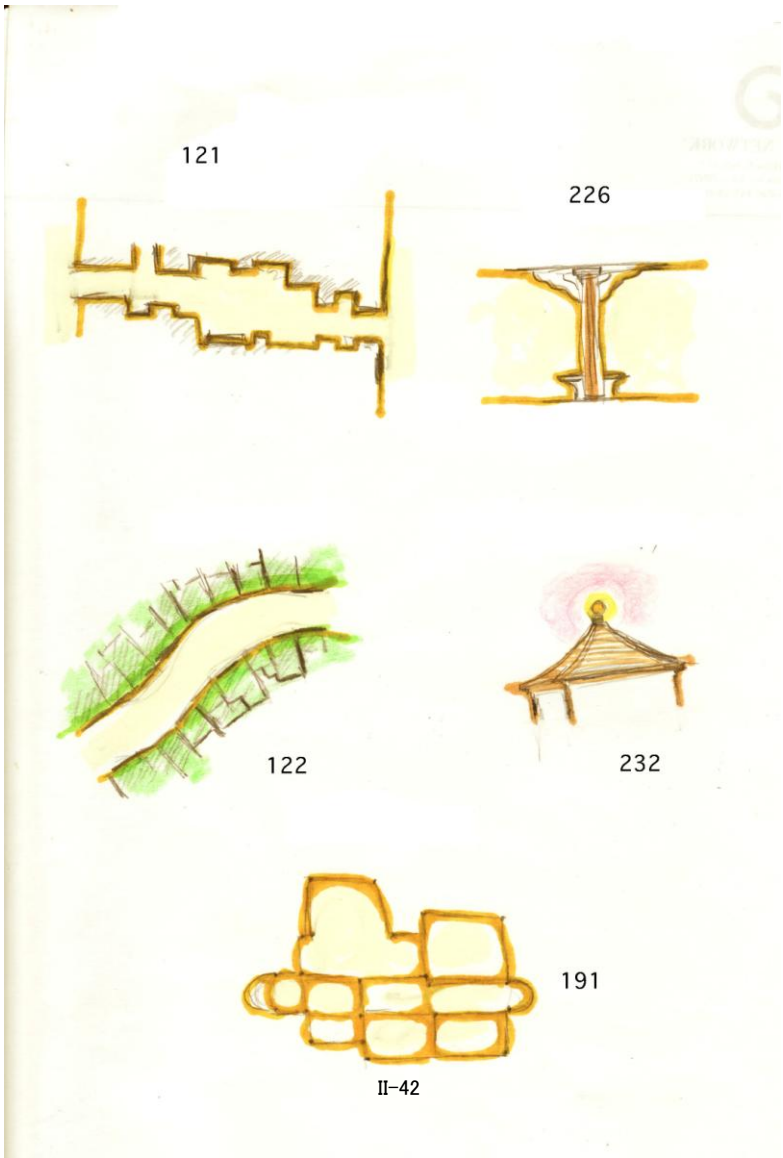
7

良い形

・「機能」としては、外部機能によって、内部に封じ込まれた機能の存在のこと。

・「センター」としては、外部の「センター」によって、一つの内部的に閉ざすような「センター」が存在すること

例 121 歩行路の形　122　建物の正面



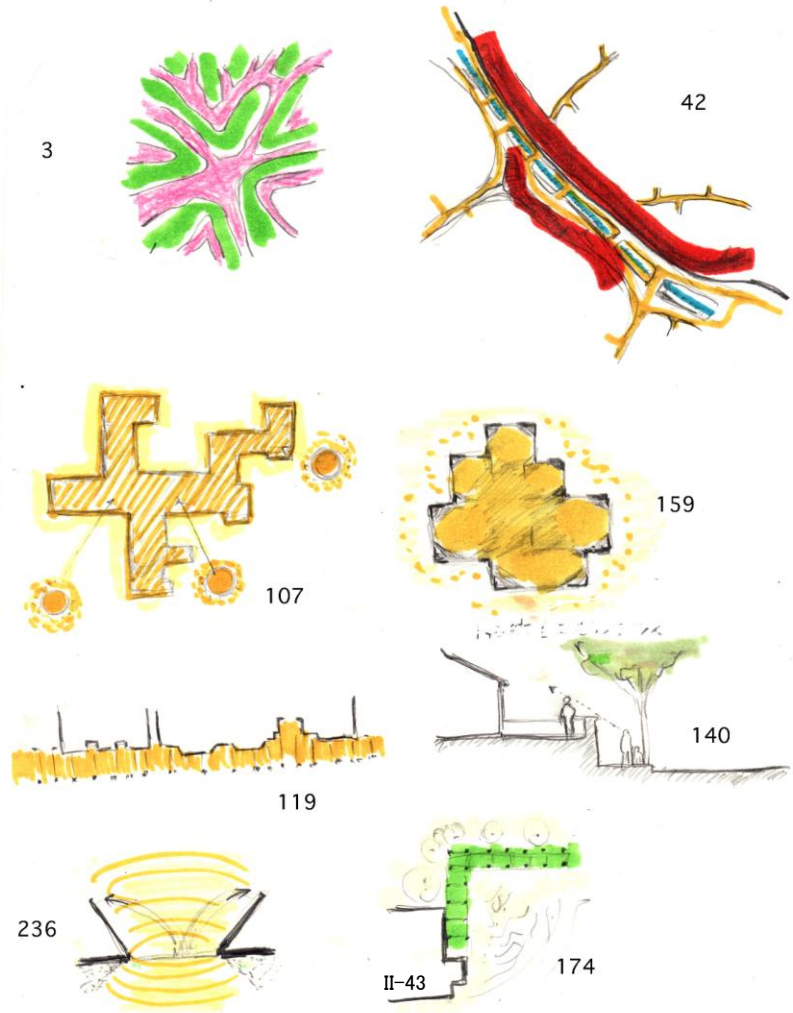
8

深い相互結合と両義性

・「機能」としては、各機能接合点において、両義的な機能で融合し合う機能のこと。

・「センター」としては、各「センター」の表面拡張により融合結合が起こる。その強い結合状態

例　119 アーケード　159 どの部屋も2面採光



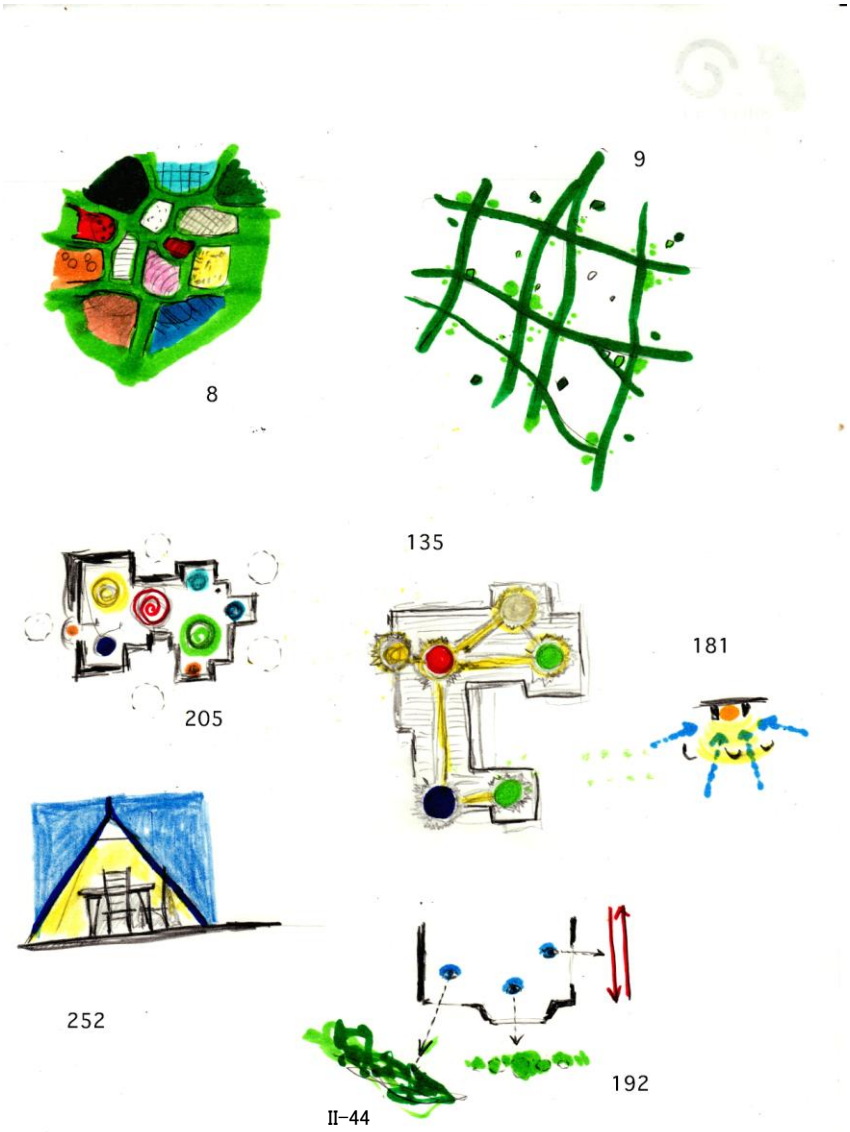
9

対比

・「機能」としては、対抗、対比する機能の存在によってとり強い機能が産まれること。

・「センター」としては、各「センター」の特質に際立った違いが有る「センター」の存在によって強化されるということ。

例　8 モザイク状のサブカルチャー　252 明かりだまり



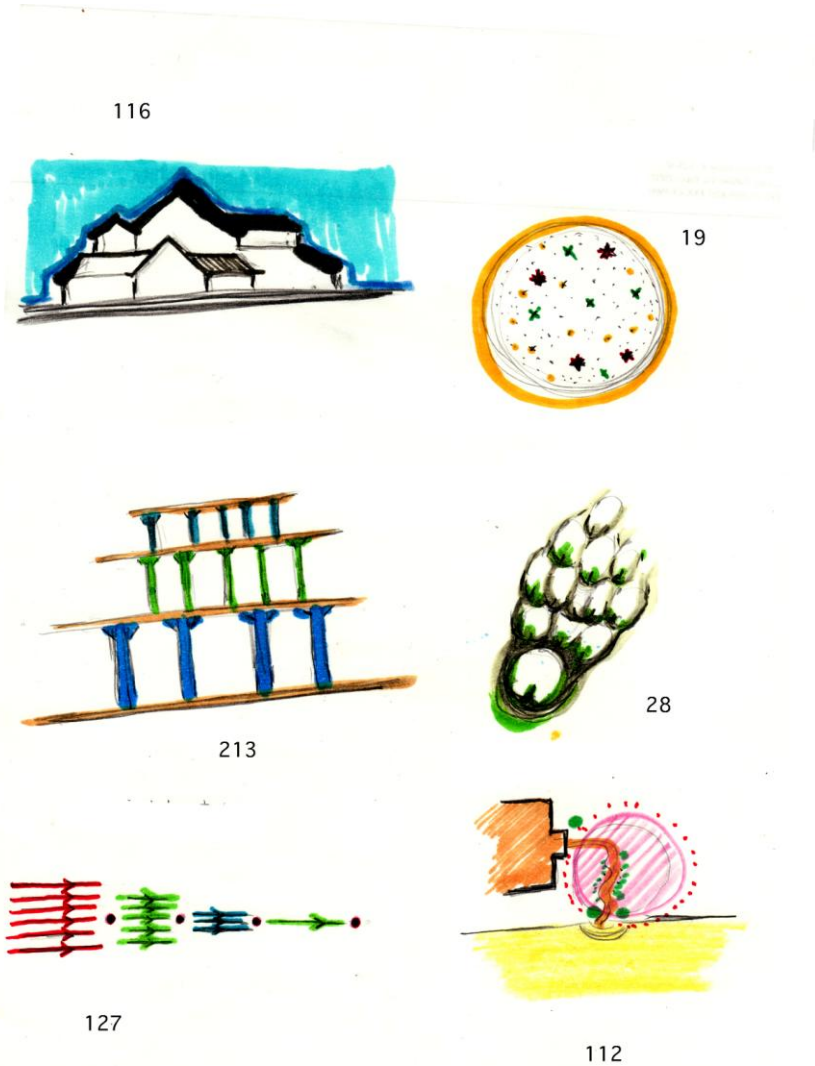
10

段階的変容

・「機能」としては、機能はその存在する「場」に応じて段階的に働く。

・「センター」としては、新「センター」にむかって「場」に影響をうけながら。
一連の変化をしていくとき補強．強化されること。

例　116 カスケード状の屋根　１２７　親密度の変化



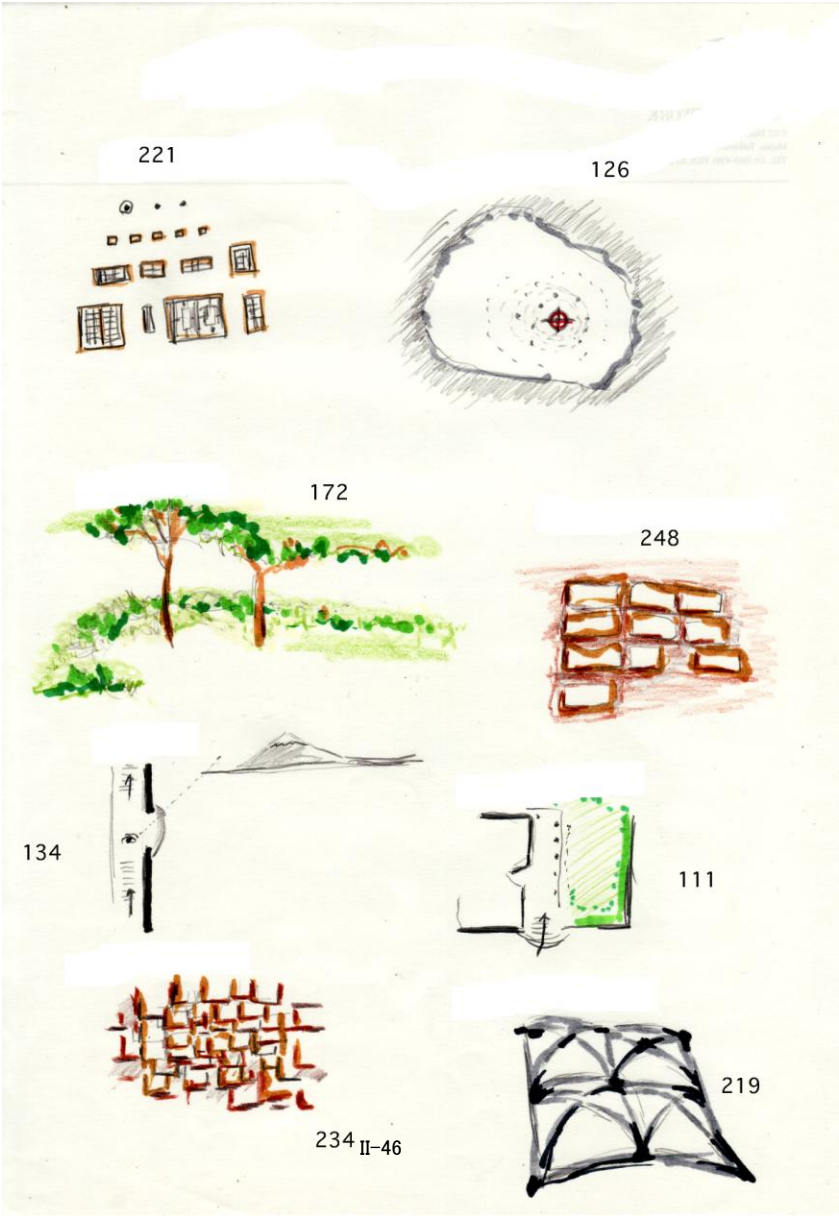
11

粗っぽさ

・「機能」としては、「場」に応じて、局所的に適応すること。

・「センター」としては、「場」や局所状況に応じて、乱調になること。

例　221 自然なドアと窓　172 野生の庭



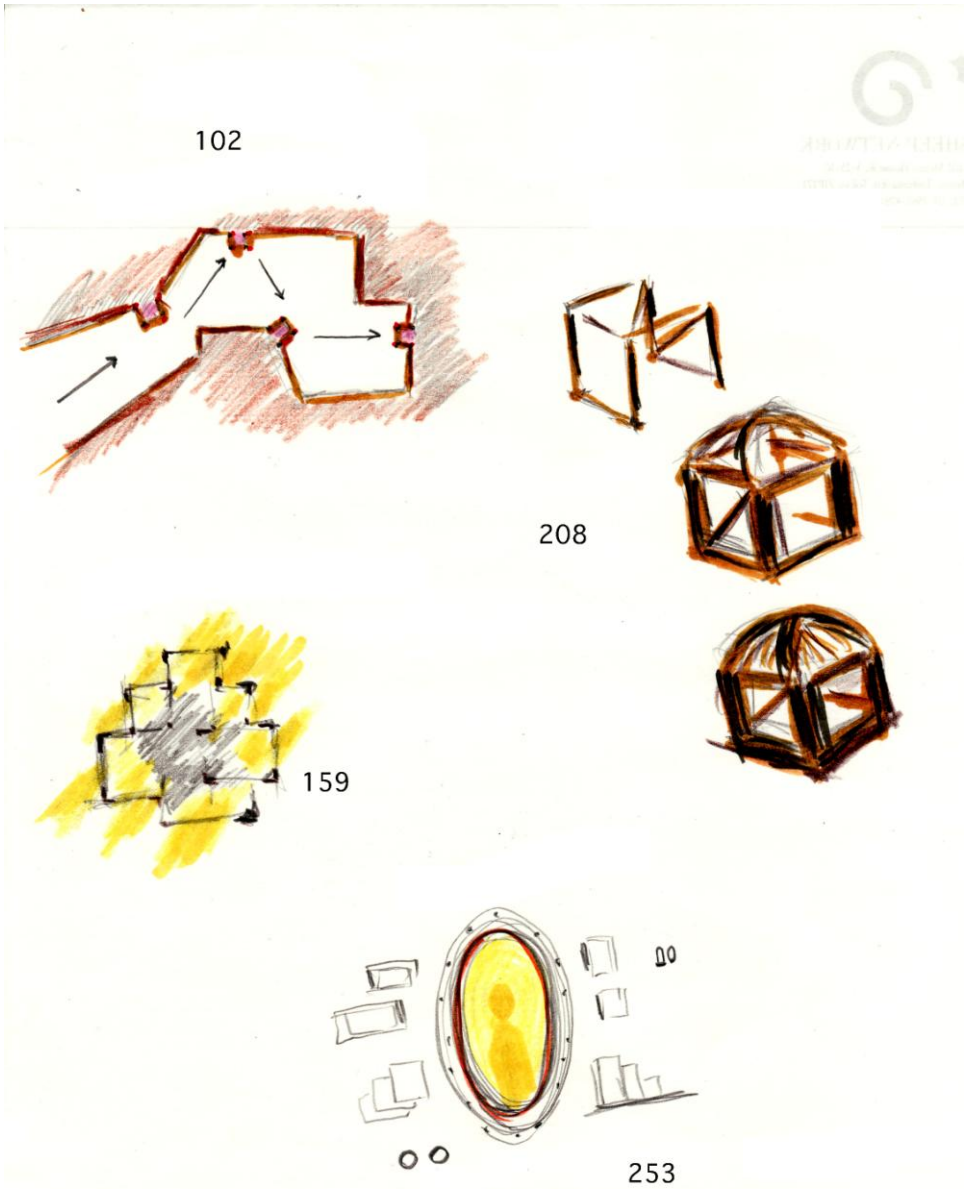
12

共鳴

・「機能」としては、一つの場所で異種の機能が、異種同形のような、性質を持ち合うこと。

・「センター」としては、より大きな「センター」に対して、一定の角度などの性質によって、対応する質をもっていること。

例　102 わかりやすい入り口の集まり　208 順に固める構造



II-47

13

簡潔さと静謐さ

・「機能」としては、複雑な条件にたいして、必要最小限のシンプルな機能で対応するのがベストということ。

・「センター」としては、内部のセンターは少しでも、少ないほうが良いということ。

例　51 緑路　64 池と小川

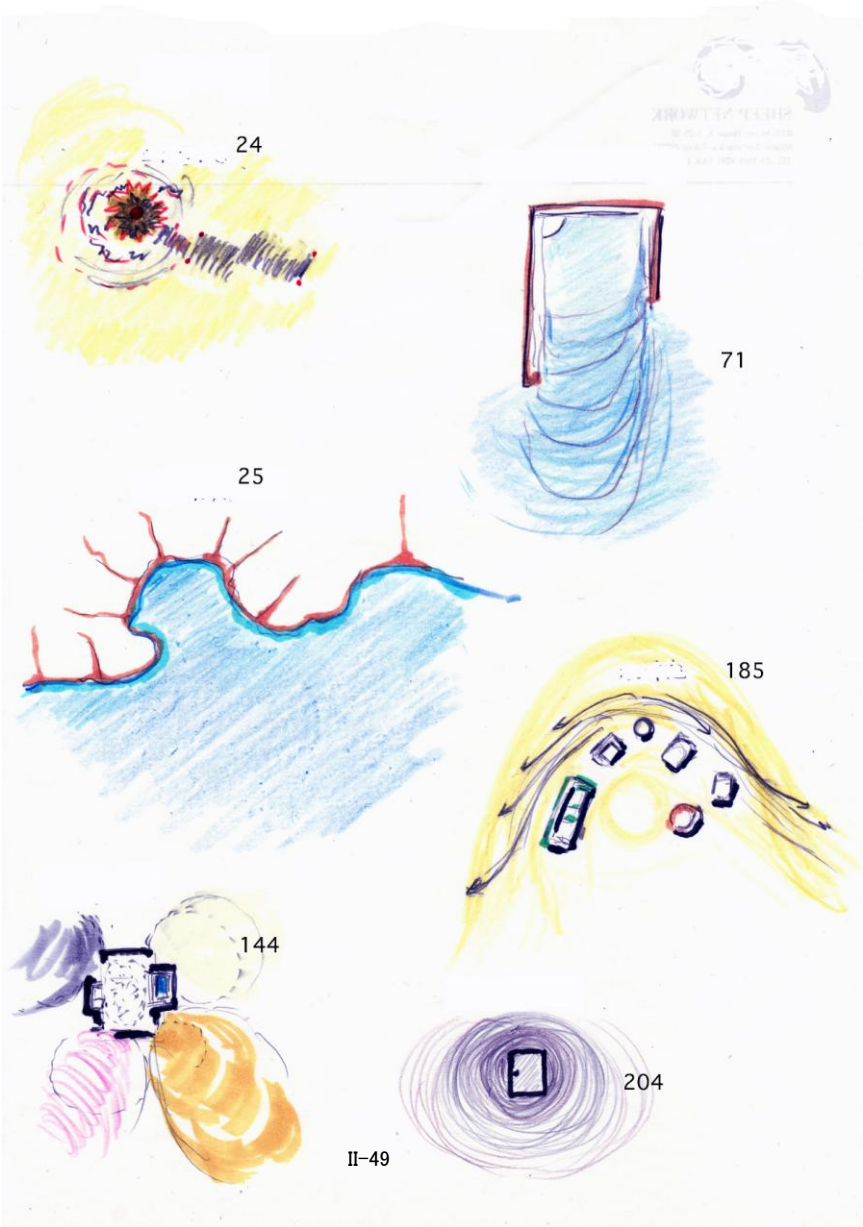
14

中空

・「機能」としては、小さな機能は、大きく、安定した機能の存在により強く、より個性的に機能できること。

・「センター」としては、「場」の存在内部に静かな「空」的存在がある場合のほうが、強く安定するということ。

例　25 水辺への接近　24 聖地



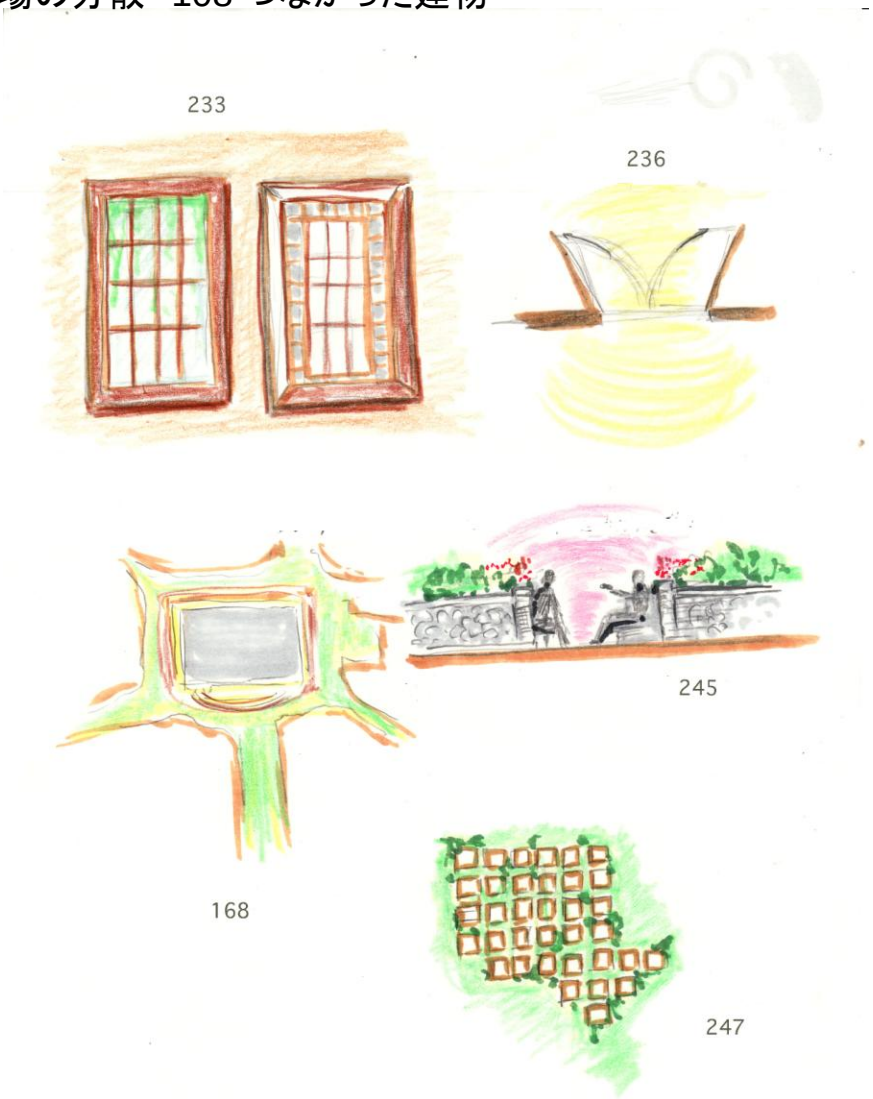
15

不可分であること

・「機能」としては、すべての機能は他機能の入れ子構造になって、一体化していること。

・「センター」は、周辺と滑らかに入れ子、回帰的に融合一体化して、識別困難なくらいに一体化すること。

例　9 仕事場の分散　108 つながった建物



参考文献表

1 Christopher Alexander, Book2 The Nature of Order-The process of Creating Life-、Book1 The -Phenomenon of Life ,CES, 2002

2 ロバート・M・パーシグ、禅とオートバイ修理技術（上、下）、五十嵐美克訳、日本版出版2008,ハヤカワノンフィクション文庫

3 オイゲン・ヘリゲル、弓と禅、稲富栄次郎他訳、1977, 福村出版

4 松本キミ子、宇宙のものみんな描いちゃおう、１９８７、太郎次郎社

5 クリストファー・アレギザンダー、オレゴン大学の実験、宮本雅明訳、鹿島出版会日本版出版1977,SD選書

６ クリストファー・アレギザンダー、時を超えた建設の道、平田翰那訳、鹿島出版会日本版出版1993

7 Stewart Brand, HOW BUILDINGS LEARN-What Happens After They're Built, Penguin Books 1994

8 トーマス・クーン、科学革命の構造、中山　茂訳、みすず書房日本版出版、1971

,

環境デザインにおけるパターン・ランゲージ
～pingpong 多摩美術大学図書館ワークショップを通じて～

岡瑞起* 李明喜† 橋本康弘‡ 江渡浩一郎§

pingpong プロジェクト

我々はモノの色や形といった意匠的なデザインではなく、人間の行為をより良い形で適えるためのデザインフレームワークを構築することを目的とする pingpong プロジェクト [1]を進めている。デザインフレームワークの構築にあたり、

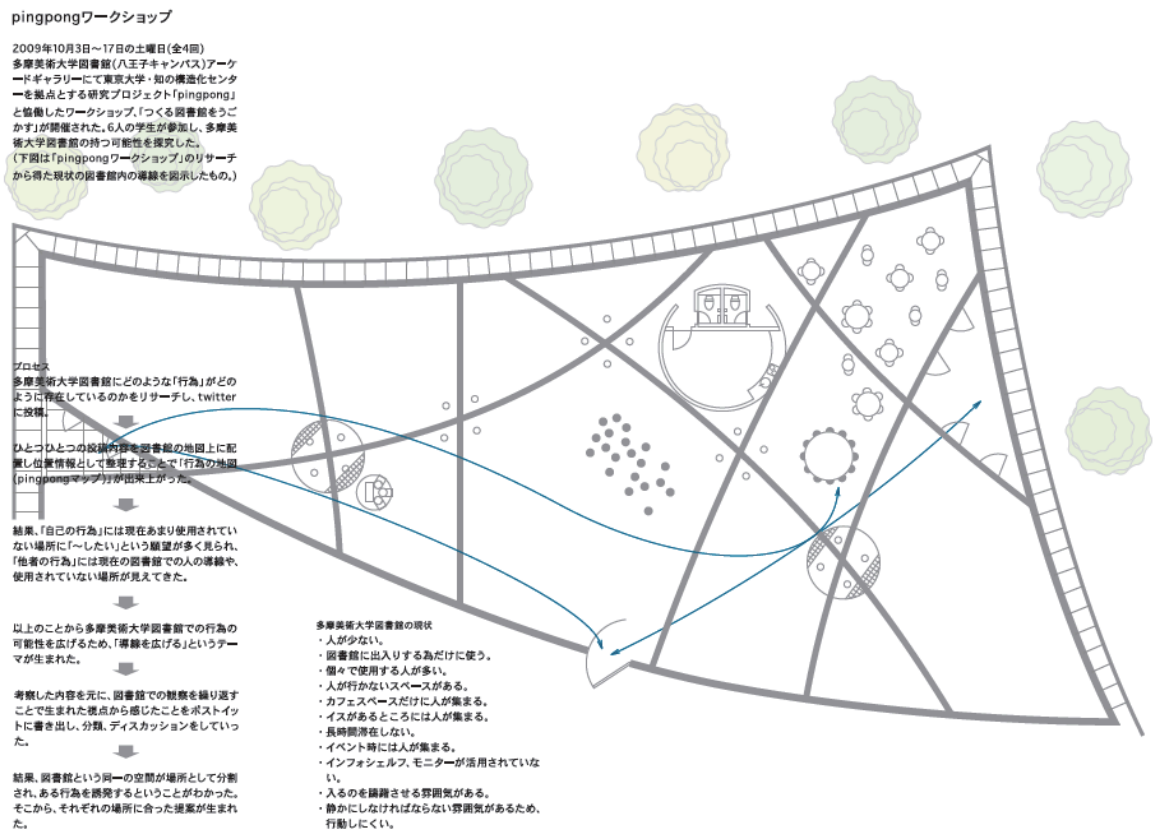


図 1: 多摩美術大学図書館利用者・現状の導線

*東京大学・知の構造化センター mizuki@cks.u-tokyo.ac.jp
†デザインチーム matt lee@mattoct.jp
‡東京大学大学院工学系研究科 hy@sys.t.u-tokyo.ac.jp
§産業技術総合研究所 k-eto@aist.go.jp
[1]東京大学知の構造化センター pingpong プロジェクト: http://www.cks.u-tokyo.ac.jp/project4.html

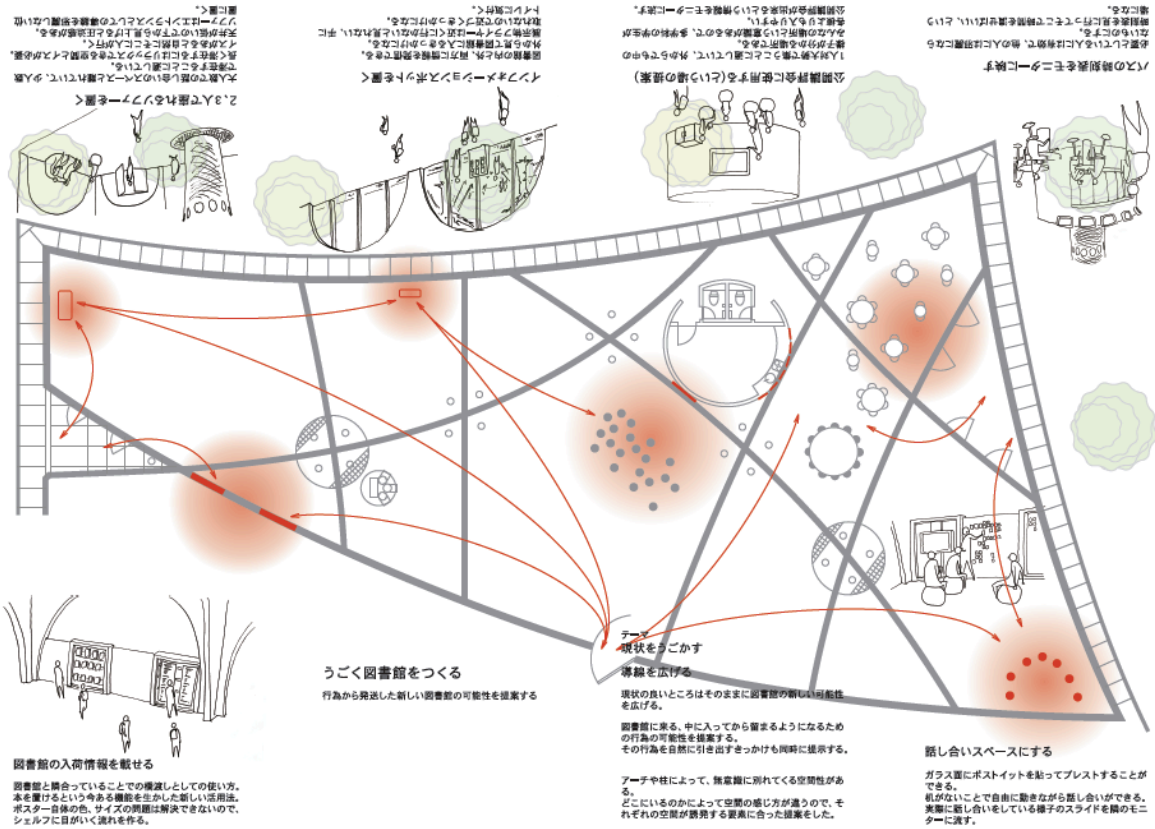


図 2: 図書館利用者の導線を広げる 6 つのデザイン提案

特に「人」「対象物」「時間」「空間」の四つのファクターから構成される「行為」からデザインを捉え、これら行為の関係性を捉えたデザインを行うための方法論およびツールを開発している [2]。

pingpong プロジェクトとパターン・ランゲージ

開発中の方法論・ツールを用いたデザインフレームワークの構築に向けた試みの第一歩として、多摩美術大学生を対象に、多摩美術大学八王子図書館 [3]のアーケードギャラリーをデザインするワークショップを行った。ワークショップでは「行為から考えるデザイン」をテーマに、アーケイドギャラリーを活性化するためのデザイン案を提出することを課題とした。pingpong プロジェクトメンバーがワークショップのファシリテーションを行うとともに、言語を通じて行為を扱う方法論・開発しているツールを提供した。

ワークショップでは、まずアーケードギャラリーで行われる行為を言語データとして収集し、開発中のツールを用いてデータを地図上にマップし可視化することを行った。このワークショップのプロセスにおいて注目すべき点が二つある。一つ目は、可視化ツールを用いることより図書館のアーケイドギャラリーで行われている行為の俯瞰図が見えるようになったことである。一つ一つの行為はもちろんのこと、俯瞰図が無い状態では今まで不可視だった集合としての行為、行為間の関係性を考慮した環境の理解や分析が可能となった。二つ目は、「いつ、誰が、どこで、何をしたのか」といった行為に関して収集されたデータを可視化ツールを用いることによりいつでも参照することが出来た点である。

[2]方法論・ツールに関する詳細は http://pingpong.ne.jp/を参照。
[3]多摩美術大学図書館ホームページ：http://library.tamabi.ac.jp/

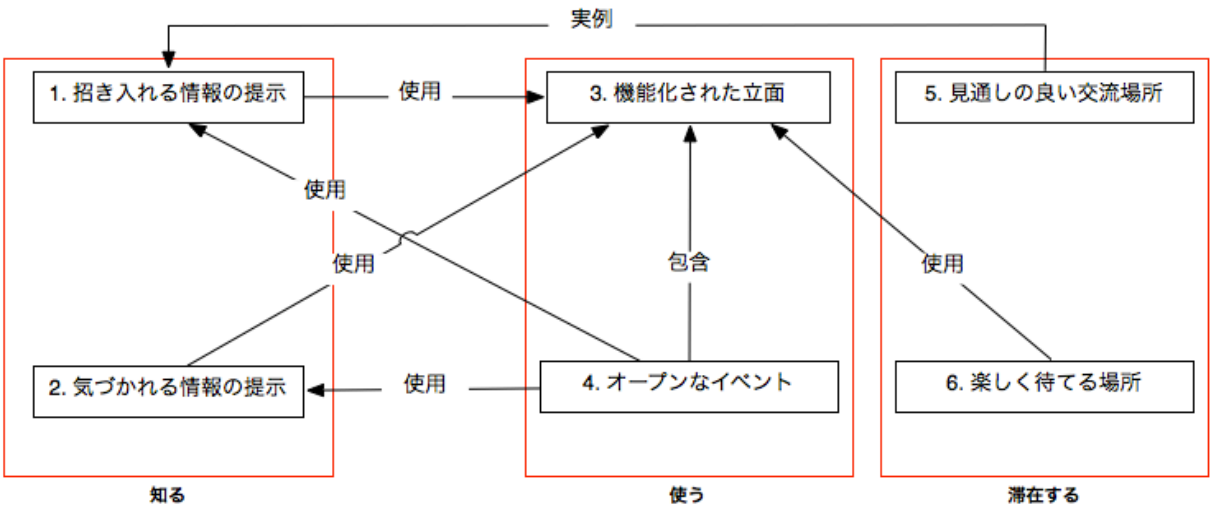


図 3: パターン間の関係図

このことは一見何でもないことのように思えるが、どのような行為、行為との関係を根拠としたデザインの提案なのかを、データとして振り返り、共有することができるということは、提案の根拠の理解を深めたり、意思決定のための合意形成を行うのに非常に有効であるという結果となった。

図 1 は、ワークショップ参加者の観察・理解・分析の結果として導き出されたアーケードギャラリー利用者の現状の導線を示している。また、図 2 は、その現状を踏まえ「利用者の導線を広げる」というテーマのもとに提出されたアーケードギャラリーのデザイン提案である。以下に記述するパターンは、図 2 に示される図書館のアーケードギャラリーのデザイン提案をもとに、著者らが環境デザインのパターン・ランゲージとしてまとめたものである。

環境デザインにおけるパターン・ランゲージ

舞台設定

パターンを使用する舞台設定として、次のようなことを想像してほしい。ある大学の図書館において、あなたがライブラリアンとして採用されたとする。あなたは既にその大学の図書館に関する建築の経緯について知っており、もちろん利用者としての経験があるとする。図書館長からあなたに図書館をより有効活用させるために、アーケイドギャラリーのデザインを対象として検討することを言い渡されたところである。その際の制約として、建築を変更しないということが課せられているものとする。

パターン地図

本稿におけるパターンは以下の６つから構成されている：

1. **招き入れる情報の提示**
2. **気づかれる情報の提示**
3. **機能化された立面**
4. **オープンなイベント**

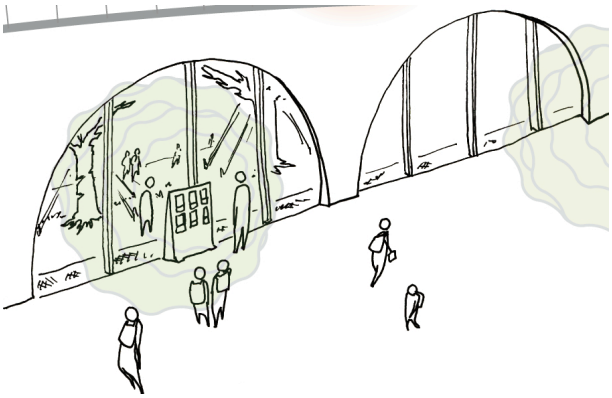
5. **見通しの良い交流場所**

6. **楽しく待てる場所**

各パターンは、「名前」「図」「状況」「問題」「解決策」「フォース」「結果」の７つの要素で記述する。また、各パターン間の関係を図 3 に示す。これらのパターンは大きく３つのフェーズから構成されている。図書館について「知る」フェーズ（パターン 1，2)、図書館内のイベントするなどして利用者が図書館を「使う」フェーズ（パターン 3，4)、そして最後に利用者が長く「滞在する」フェーズ（パターン 5，6）である。それぞれのパターン間の関係については、次章の各パターンの説明において詳しく述べる。

The Patterns

1. 招き入れる情報の提示



状況：　図書館内の情報を伝える手段として、図書館内には、館内を横断するような面積が広く、長さも非常に長いインフォシェルフが設備として用意されている。

問題：　図書館内の情報を外に伝える手段が乏しい。そのため、図書館へのアクセスが誘発されにくい。

解決策：　建物の壁（表層面）にサイネージやフライヤー等のインフォメーションスポットを設置することにより、建物の表層をコンテンツ化する。インフォーメンションスポットが図書館内の情報を外に伝える仕組みとして機能することになる。

フォース：　インフォシェルフには図書館に関する情報が提示されているが、このインフォシェルフへは図書館内に入らないとアクセスできないため、外からは図書館内で何をやっているのか情報が伝わらない。もちろん、図書館に入るとその情報にはアクセスできるのであるが、そもそも図書館に入るきっかけが無い現状がある。その結果、図書館への来場者が少なくなってしまっている。

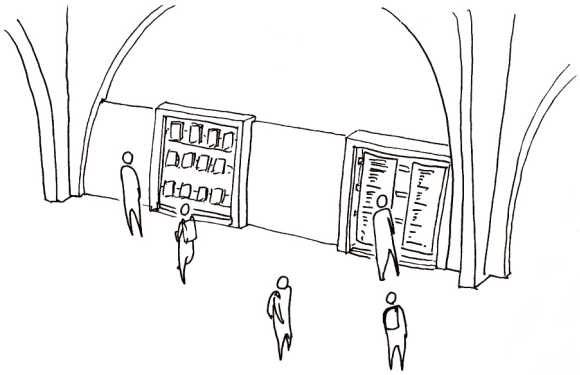
結果：　建築における表層のコンテンツ化を図ることにより、図書館の外に情報を発信できる仕組みが生まれるとともに、図書館へアクセスするきっかけとしても機能するようになる。

◇　◇　◇

図書館へのアクセスを誘発するためには、来訪者が図書館に関するコンテンツや図書館が持っているコンテンツを、外からでも閲覧できるようにすることが必要となる。一般的に、外から見ることのできるコンテンツを提供する手段は、情報の掲示場所などが限られることから、それほど多くはない。このパターンを活用するためには、多摩美術大学八王子図書館の場合のように、建物の壁一面がガラス張りになっていることや、図書館の周りの緩衝地帯が広いといった建築的な条件が必要となる。このような前提条件が揃って初めて、ガラス張りの壁に情報を掲示したり、壁に近づき提供されている情報にアクセスすることが可能となる。

ガラス張りの壁を情報提示に利用するといった、建築における表層のコンテンツ化は、壁という建築の立面を情報提示という機能化を通じて行っていることから、「3. 機能化された立面」パターンを包含しているパターンとして位置づけることができる。また、情報提示という観点から考えると、「2. 気づかれる情報の提示」パターンで提示する情報とどのように使い分けを行うのかを考慮する必要がある。

2. 気づかれる情報の提示



状況： 図書館のアーケードギャラリースペースを横断するようにガラス張りのインフォシェルフが存在する（上図・写真（右）を参照)。インフォシェルフは、アーケードギャラリーと図書館を分ける仕切りの壁としても機能している。

問題： 図書館利用者にとって情報に気づくきっかけがなく、情報を提供する媒体として充分に機能していない。

解決策： インフォシェルフに、本の入荷情報などこの場で必要とされる情報を載せ、頻繁に更新する。そのために、図書館のオペレーション側のスタッフを雇う。

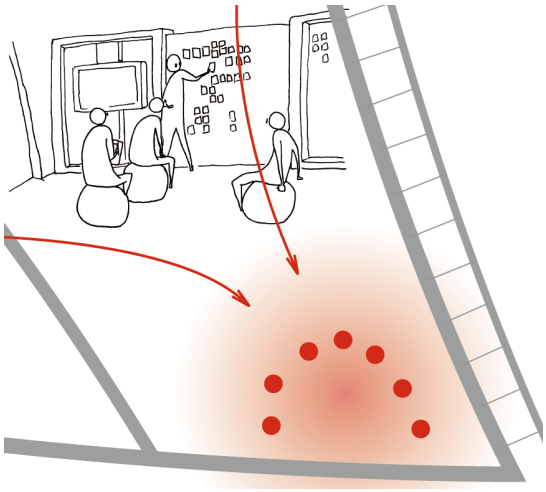
フォース： インフォシェルフは長さも長く、面積も広いため情報が更新されても気づきにくい。とはいえ、アーケイドギャラリーで行われている他の行為や、落ち着いた雰囲気を壊すような視覚的な刺激を与えることは避けたい。

結果： 更新頻度高く情報を提供することによって、空間の仕切り壁としてだけでなく、情報提供のための媒体として機能していることに利用者が気づくことができるようになる。さらに、アーケイドギャラリーへの入り口から、図書館への入り口まで大きな範囲に渡って存在するインフォシェルフを有効活用することによって、一度足を止めて図書館に入るというフローが新たに発生する。また、利用者にとって有益な情報を提供することによって、図書館の利用を誘発・活性化することにつながる。

◇　◇　◇

本パターンは、「1. 招き入れる情報の提示」と同様、壁という建築の立面を情報提示という機能化を通じて行っていることから、「3. 機能化された立面」パターンを使用しているパターンとして位置づけることができる。

3. 機能化された立面



状況： 二人以上の人数で構成されているグループが、日常の何気ない話から出てきた議題について話し合いができる場を必要としている。

問題： 座って話し合いをする場所は存在するが、ポストイット等のツールを使って視覚的に議題を共有できないので話し合いが深まらない。

解決策： 図書館のさまざまな空間に存在する建築における立面を機能化する。たとえば、立面に掛けてある既存のモニターに共有するべき情報を映し出し、モニターの前の空間を利用することにより視覚的に議題を共有し話し合いを行う。同様に、ガラスの立面も資料を貼りつけ、情報を共有しながら話し合いを行える空間として使う。

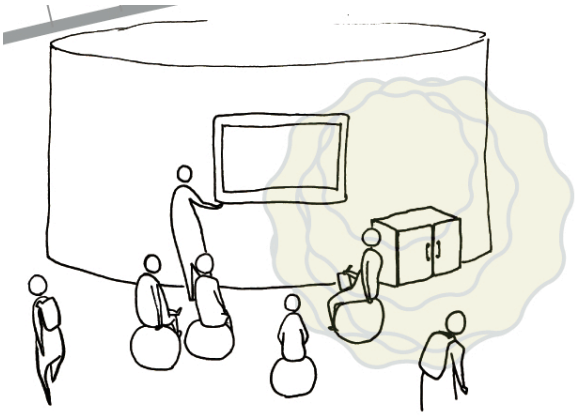
フォース： 話し言葉はその場で失われてしまい、人の記憶も大して当てにならないため、議題や意見をきちんと共有するために情報は即時的に視覚化される必要がある。

結果： 活用されていない立面を機能化することによって、二人以上から構成されるグループのための話し合いの空間を創り出すことが可能になった。これまで、このような立面の機能化が可能であるということに気づいていなかった利用者も、他のグループが話し合いを行っている様子を目にすることによって、同様に活用し始めるようになることが想像される。

◇　◇　◇

話し合いの空間を作るには、建築の立面の周りに使われていない場所が存在する必要がある。また、他の図書館の利用者の雑音が干渉したり、話し合いをしているグループ自体が周りに迷惑を及ぼす可能性がある。そのため、そのような他の利用者への干渉が少ない時間帯を選んだり、余裕があるスペースであることが大切となってくる。また、情報を共有するための視覚化として、パワーポイントといった電子資料を使う場合が多い。そのための道具として建物の立面にモニターがあると更に便利である。

4. オープンなイベント



状況： 図書館の出入口と接している場所にアーケイドギャラリーは存在する。出入口はアーケイドギャラリーの両サイドに二つ存在し、一つの出入口からもう一つの出入口が見えるほどオープンで広い空間となっている。しかし、その広くオープンな環境のため寂しい印象を与えている。

問題： イベント開催時は人が訪れ、アーケイドギャラリーに留まるため寂しい印象を与えないが、そもそも開催されるイベントの数が少ない。

解決策： 学科内で行っていた講評というイベントを公共の場に持ち出し、オープン化を行うことで、アーケイドギャラリーで頻繁にイベントを開催する。

フォース： 新しくイベントを企画し、実施することは、人的、時間的コストが高く困難である。また、アーケイドギャラリーでイベントを開催する際には、不特定多数の人が利用するという場所の性格上、どんな人が参加しても問題が無いという条件が必要になる。

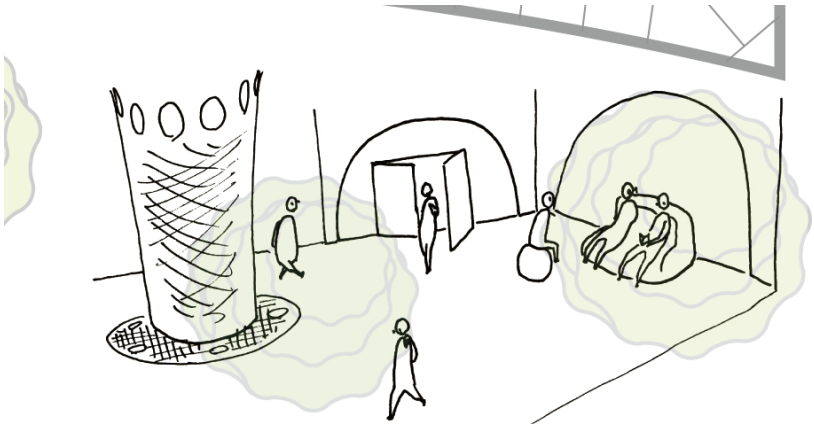
結果： イベントのオープン化の結果、他学科の学生の講評への参加が可能となる。結果として、ギャラリーが共有の場所として使わる頻度が増え全体が賑やかになる。

◇　◇　◇

イベントのオープン化に際して、イベントに関する情報を提示することが必要となる。そのために使うことが可能な他のパターンとして、「1. 招き入れる情報の提示」、「2. 気づかせる情報の提示」が挙げられる。また、本パターンは、イベントを行う際に立面に掛けてあるモニターを使用できる機能が用意されていることから、パターン「3. 機能化された立面」を包含している。

5. 見通しの良い交流場所

状況： 多摩美術大学八王子図書館アーケイドギャラリーの床は元の地形を活かした建築設計となっている。その結果、傾斜を持った床となっている。床が一番高くなっている場所は、特に何のインテリアも置かれておらず使われていない空間となっている。



問題：　アーケイドギャラリーにおいてゆったりと長時間話しができる空間がない。「4. オープンなイベント」で述べられている講評会のようなイベントを行う空間や、「6. 楽しく豊かな待つという行為」が提案されている空間において、テーブルやイスが用意されてはいるが、これらの空間はほとんどの場合、くつろぐための空間としては利用されていない。

解決策　空間全体が見渡せて、大人数で話している場所からは離れている場所に、２、３人で座ってくつろげるソファーを置く。

フォース：　ゆったりと気持ちよく滞在させるためには、大人数で話している場所からは離れた、開放的でゆとりのある空間である必要がある。さらに、座ったときに天井が高すぎないということも落ち着くためには必要な要素である。

結果　空間全体を見渡すことが可能で、天井が一番低い比較的落ち着ける場所にソファーが置かれることで、リラックスして長時間、会話ができる空間が生まれることが期待される。

◇　◇　◇

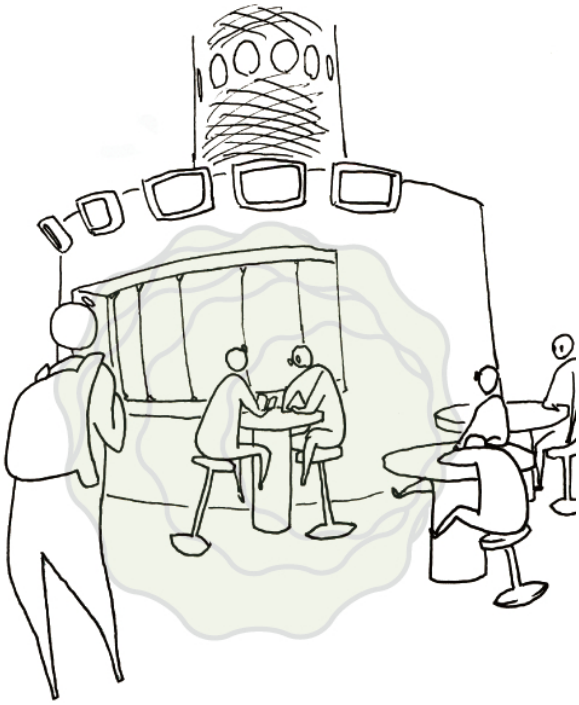
本パターンは、「1. 招き入れる情報の提示」パターンの実例の一つとして捉えることができる。壁がガラスで出来ているため、２、３人で座ってくつろげるためのソファーがあり、人々がリラックスしている様子を図書館の外からガラス越しに見ることができる環境にある。また、その様子が目に入ることにより、「1. 招き入れる情報の提示」パターンの一例として機能している、と捉えることができる。

6. 楽しく待てる場所

状況：　アーケードギャラリーへの２つの出入口のうち、１つの出入口に近い空間にテーブルやイスが置かれた空間が存在する。このアーケードギャラリーの出入口から数分歩いたところにバス停がある。そのため、この空間はバスを待つ場所として図書館の利用者に良く用いられている。

問題：　バスを待っている利用者は、図書館でぎりぎりまで楽しい時間を過ごしたいのだけどいつバスが来るかわからないので、早々に切り上げなければならない。

解決策：　バスを待っている利用者のために、時刻表や、発着状況についての情報を提供する。



フォース：　バスは交通状況や天候状況によって発着状況が変更されることが多い。そのためバスの時刻情報は役立たないことが多い。

結果：　バスの時刻表や発着状況の情報に常にアクセスすることが可能になり、待つという行為をストレス無く過ごせるようになる。その結果、待つという行為を楽しく豊かな行為を行える時間として使うことが可能になる。

◇　◇　◇

快適な待ち空間となるために、情報提供をするためのディスプレイ、風雨が凌げる屋根、座って待てるためのイス、荷物を置くことのできるテーブル、建物の出入口に近い、トイレに近いといった状況があると望ましい。

本パターンは、モニターを使用し、バスに関する情報提供をすることにより待つという行為の時間を楽しく豊かにしていることから「3. 機能化された立面」パターンを使用している。

結び

本稿におけるパターンは、多摩美術大学図書館のアートギャラリーがどのようにしたらさらに有効活用することができるかについて述べている。最終的なアウトプットとしてのパターンももちろん大切であるが、我々が注目したいことは、冒頭で述べた通りこれらのパターンが「人」「対象物」「時間」「空間」の四つのファクターから構成される行為データを収集し、それらを地図上にマップ・可視化するツールを通じて、一つ一つの行為だけでなく、集合としての行為、行為間の関係性を考慮して作成されたパターンである、ということである。たとえば、「6. 楽しく待てる場所」というパターンは、既にバスを待つ場所として使われているという行為とともに、他にもその場所で行われている「集まる」「話す」「座る」「机にモノを置く」といった行為を邪魔しないで更に良くするにはどうしたらよいのか、といった行為間の関係を考慮して作成されたパターンとなっている。

また、ツールを用いることにより、パターンの根拠となった行為をデータとして振り返ること、および、パターンが作成されたコンテキストを常に参照することが可能であるという点も重要である。これらパターンが作成された根拠となる生のデータは、図書館のアーケードギャラリーというある特定の環境をデザインするだけでなく、それ以外の環境デザインにおいてパターンを活用する際に、より効果的に利用する手助けとなるはずである。

リアルイベント開催のためのパターンランゲージ

江渡 浩一郎* 柴村 しのぶ†

1 はじめに

近年，ウェブ上での活動を背景としたイベントの開催がさかんに行われている．ブログでプログラミングなどの知見を公開していた人々が，日時と場所を決め，共に会って情報交換をする．そのようなイベントは，勉強会，読書会，研究会，ワークショップ，カンファレンス，シンポジウム，学会，ハッカソン，キャンプなどといった名称で呼ばれている．本稿ではそれらのイベントを，オンラインでのつながりと対比する形で**リアルイベント**と総称することにする．

このようなリアルイベントは近年増大傾向にある．リアルイベントについての情報を集積した「IT 勉強会カレンダー」[1] によれば，最近ではほぼ毎日どこかでリアルイベントが開かれている状況である．これは，インターネットとウェブの普及によって共通のテーマを持つ人々同士を見つけやすくなったことが背景にあると思われる．

学術分野では従来から研究会・学会などといった形で関係者が定期的に会合を持つことは一般的だったが，近年のリアルイベントの流行は，そのような学術分野の習慣が一般にまで広がったことと見ることもできるだろう．学術分野の組織に属する人たちだけが行ってきたリアルイベントを普通の人も行うようになってきたため，そのようなイベント開催のための知見の重要性も増していると言える．

筆者は，主催者や発起人，運営委員という形で，カンファレンスや勉強会といったリアルイベントに関わってきた．その経緯から，リアルイベントの開催に向けての知見をパターンランゲージの形でまとめることにした．パターンの記述形式は「シェファーディングのランゲージ」[2]，「絵本を読むときのパターン・ランゲージ」[3] を参考にした．

リアルイベントの開催準備の観点からは，参加人数の規模は重要な要素であり，規模の大小によって各々のパターンの適用可能性も変わってくると思われるため，その点はパターン毎に付記する．

1.1 舞台設定

あなたが，ある特定のテーマに関心があるとする．あなたはすでにそのテーマについて，ブログなどで情報発信を続けてきている．また，ブログや Wiki，SNS，メーリングリストなどといった場で，同じテーマに興味を持つ知り合いと情報交換を行い，オンラインコミュニティを形成してきたとする．そのような情報交換をもっと促進させ，自身の知見を深めるために，オンラインの枠を超えたリアルイベントを開催したいと考えたとしよう．

* 産業技術総合研究所 k-eto[at]aist.go.jp
† Wiki ばな shino[at]freedomcat.com

2　The Patterns

2.1　仲間を探す

あなたがリアルイベントを開催したいと思ったとき，いったいどこから取り組めばいいのだろう．オンラインでの取り組みならなんとなくわかるが，リアルイベントは考える要素が多すぎてどこから考えればいいのかわからなくなってしまう．

＊　＊　＊

リアルイベントの開催という大きな仕事を一人で取り組もうとすると，すぐに煮詰まってしまう．一人では決められないような問題が発生したときに，やる気をくじかれ，プロジェクト進行が停止してしまうのだ．

同じテーマに関心をもっている人同士とはいえ，リアルイベントは異なる意見を持つさまざまな人が集まってくる場だ．考えなくてはいけないことがたくさんでてきて，頭がごちゃごちゃになってしまうかもしれない．一度にたくさんの人に意見を聞けば，そのなかには互いに相容れない意見もあるだろう．それらをひとりで一手に引き受けてしまうと，一人の人間の処理能力を超えてしまう．

しかし，よく考えてみれば，あなたはすでにオンラインコミュニティの中で経験を積んできていて，さまざまな知り合いと出会っているのだった．その中には，信用してもよいと思える人もいる．

それゆえ：

共にリアルイベントを開催しようという意欲を持つ仲間を探そう．これまでオンラインコミュニティで知り合った人の中から信頼できそうな人を探し，声をかけてみよう．

一人では決められないことでも，複数の人で話しながらであれば不思議と決められる．決断が迫られる場面でも，一人じゃなくて複数人で決めれば心強い．

イベントのテーマや運営について共に考え，プロジェクトを進めていく仲間を，ここで共同開催者と呼ぶことにしよう．イベントの方針に悩むことがあったら，共同開催者の意見に優先して耳を傾けよう．リアルイベントではさまざまな関係者の利害関係が衝突することがある．その場合，全員の意見を同時に聞くことはできない．さまざまな条件を考慮しつつ共同開催者と議論し，自分と共同開催者が共に納得できる解を見つけることを重視しよう．

共同開催者という立場でなくても，イベントの開催経験を持つ人をアドバイザにつけるのも良いだろう．イベントの開催には失敗がつきものだ．アドバイザによる他のイベントの過去の経験を知ることができれば，そこから得られるものは大きいだろう．

直接イベントの経験が無くても，自分を精神的に支えてくれるメンターを得るのも有益だ．一歩距離をおいて見守ってくれるような人を味方につけると，精神的な負荷をおおいに軽減してくれる．

＊　＊　＊

このようにして仲間を見つけることができたら，次はイベントを具体的にしていこう．最初の一歩は，イベントの性格決めだ．それには≪名は体を表す≫について考えるとよい．

2.2　名は体を表す

無事仲間を探すことができたら，次にイベントの性格について考えてみよう．

＊　　＊　　＊

イベントの性格はどのようにして決めればいいのか．イベントの対象範囲を明解にしないと，先に進めない．

リアルイベントは，オンラインではできなかった人とのつながりを作ったり，情報交換をするために開催するのだから，できるだけたくさんの人に来てほしいし，興味を持ってほしい．

しかし，自分たちで開催するからには，自分たちで手におえる範囲に収める必要がある．あまりに幅を広げすぎれば自分たちが興味のない対象まで扱わなければいけなくなるかもしれない．人が多すぎればそれだけ準備も面倒になる．かといってあまりに対象をせばめすぎると，参加者が集まらないかもしれない．一方で，限られた参加者でより深く密度の濃い話をするために，あえて対象を狭めると良い場合もある．イベントの性格決めによって，イベントの参加者だけでなく，イベントの協力者の性質も変わってきてしまうだろう．

イベントの性格を一言で説明できるようにしたい．「○○を対象として○○のようにやるイベントです」ということを，部外者にも簡単にわかるようにしておきたい．

それゆえ：

イベント名を決めるというプロセスを通して，イベントの性格決めをしよう．

イベントの名前はとても大事だ．その名前自体がイベントの本質を表してしまう．イベント名が曖昧だと，そのイベントが何のためのものなのか部外者には伝わらない．自分たちが来てほしいと思っているような人にアピールしつつ，かつ可能性が開かれている名前を選ぶ必要がある．

実はこの課題にはすでに取り組み始めているはずだ．共同開催者にアプローチする際に，自分が開催したいと思っているイベントを「○○について話しあう（or 集まる）イベントを開催したい．一言で言えば○○という感じ」といった形で説明していることだろう．それが出発点となる．共同開催者と共に議論し，共に納得するようなイベント名を決めよう．

例えば，「Wiki について話し合う集まり」を開催したいと思ったとしよう．「Wiki 勉強会」という名前はありがちだが，これを聞いた人は Wiki について勉強する場だと思うだろう．「Wiki 研究会」であれば Wiki について研究する場だと思うだろう．筆者は，Wiki という曖昧な対象について議論するためには，研究成果というよりもむしろ実践した経験がある人を集めたいと考えた．そのイベントの名前として「Wiki ばな」[4] を選択した．「ばな」は「恋バナ」と同じ語法で，「話」の省略形である．このようなフランクなイベント名にすることによって，固くるしくなく話し合う場を産み出すことができた．同様の言葉として海外では「Meetup」という名称が使われている．こちらも，会って話をするイベントという意味だ．

イベントを続けて何回も行うつもりであれば，イベント全体を通した名前と，個々のイベント毎のタイトルの両方を考えよう．初回は，単に「オープニング」などといった無限定の名前でいい．次回以降，サブタイトルを詳細化していく．

イベント全体の名前が適切な名前ではなかった場合は，変更することを考える．例えば，Ruby に関するイベント「日本 Ruby 会議（RubyKaigi）」[5] は，2006 年の初回開催時には「日本 Ruby カンファレンス（Japan Ruby Conference）」[6] という名前だった．しかし，アメリカでは以前より「Ruby Conference」が毎年開催

されていて，英語圏の人にとっては紛らわしい名前だった．そのため，翌 2007 年より正式名称を「日本 Ruby 会議 (RubyKaigi)」[7] に変更した．これによって，海外の人にも日本独自の Ruby に関するイベントというイベントの性格がわかりやすくなった．それだけではなく，「カンファレンス」という固い印象を与える言葉から「Kaigi」という柔らかい印象を与える言葉に変更したことで，「Kaigi らしさ」とは何かを自らが考えるきっかけとなり，イベントに独自性を出すことにつながった．

＊　＊　＊

そうしてイベント名をつけたら，次は≪最初に外枠を決める≫に従って，イベントの概要を決めよう．

2.3　最初に外枠を決める

仲間を得られ，イベント名をつけたなら，徐々にイベントの詳細を決めていこう．

＊　＊　＊

イベントの詳細を決めるといっても，どこから決めていけばいいのだろうか．さまざまな制約条件が相互に絡みあっていて，どこから決めればいいのかわからない．

例えばスケジュールを決めようと思っても，もしその決めた日が話者にとって都合が悪かったらどうしよう．もちろん誰に話してもらうのかも決めていないため，スケジュール調整のしようもない．

とはいえ，何か決めないと始まらない．決めるとしたら，最初に決めると後からでは変えられなくなるようなところからだろう．

それゆえ：

イベントの外枠から決めていこう．一番明解な外枠は，日時と場所だ．

まず自分たちが参加できなければしょうがない．自分と共同開催者の都合のいい日時を決めよう．その日時で使える場所を候補の中から探す．場所はそれぞれ性質が異なる．大きな会場，小さな会場，高い，安い，いろいろある．自分が開催したいと思っているイベントは，そのうちのどの場所に対応するべきものなのかを考えよう．

安くて広くて使いやすい場所は，誰もが使いたいと思っている．そのような人気のある場所は，だいたい 1 年くらい前から予約が入る．だから，日時を決めたら，場所を決めて予約をとることを優先するのがいい．

逆に，参加人数の規模が小さい勉強会のようなものを開催したいと思っているのであれば，広い会場は必要ない．小規模な会議室であれば，比較的直前まで余裕があるはずだ．その場合は後述の《人選は誠実かつ大胆に》を優先するのがいいだろう．

日時と場所が決まると，後はその外枠に従って中身を決めていけばいいということになるので，決めることに対する悩みがぐっと減る．

時間と場所が決まったら，タイムテーブルについて考えておくといいだろう．会場を開ける時間と閉める時間が決まっていれば，おのずとイベントに使える時間も決まる．一人あたり何分で話すかを決めれば，全部で何人話せるかも決まる．そうしたら，その人数分の人選を行えばいいのだ．

このとき，タイムテーブルにはあらかじめ適度に休憩をいれておくようにしよう．人の集中力は 30〜40 分程度が限界なので，その程度の時間毎に区切って，5〜10 分の休憩を挟んでおくようにしよう．

話者の候補者に依頼をする際には,「この日にイベントをやろうと思っています」と具体的に日付を指定した上で依頼する方がやりやすい．日付，時間，場所，話す時間，他の候補者などといった情報が無いと，候補者も判断に迷う．できるだけ外枠をしっかり決めていればこそ，話者も安心して引き受けられるというものだ．

＊　＊　＊

もちろん，この方法は常に正しいわけではない．《人選は誠実かつ大胆に》にあるように，非常に重要と考える話者がいる場合には，その人のスケジュールを最初に確認して，その人の都合を優先する形でイベントの日程を決めよう．

2.4　人選は誠実かつ大胆に

イベントの日時と場所が決定したら，今度はいよいよ中身に移っていく．いったいどういうイベントにするのか．イベントを考えていて，一番楽しい瞬間だ．参加者数の規模が大きいリアルイベントの場合には，登壇者がイベントの中心となる．また，少人数のリアルイベントの場合には，参加者全員がイベントの重要な構成員となる．このように，イベントの規模の大小に関わらず，イベントにどのような人を招き，話してもらうかはリアルイベントの最も重要な要素だと言える．

＊　＊　＊

イベントで話してもらう人は，どのように選べばいいのだろうか．突然依頼したりしても，大丈夫なものなのだろうか．

いままではイベントに参加する側だった．プログラムの話者のリストを見て，面白い人がいるなと思ったことはある．でもいざそのイベントを自分で作る側に回ってみると，いったいどうやってそのような話者のリストを作ればいいんだろうと悩んでしまう．

でも，自分にはイベントをやりたいと思ったきっかけがあるんだった．すでにオンラインでの人とのつながりがあって，そこで知り合ったたくさんの人がいる．そういった人たちの話を直接会って聞きたいと思ったからこそ，リアルイベントを企画しようと考えたのだった．

それゆえ：

まずは自分に誠実に，自分が本当に話を聞きたいのは誰なのかを考えてみよう．肩書きや見栄ではなく，その人の話の中身に興味があるかどうかに集中しよう．そうして選んだ人が，すごく偉い人だったり，逆に一度も人前で話したことが無いような人だったとしても，臆することなく大胆に依頼してみよう．

自分がイベントをやりたいと思ったきっかけからスタートするといいだろう．「この人の話を聞きたい，だからイベントを開催したい」そのように思ったきっかけがきっとあるはずだ．そのような自分の心の動きを大切にしよう．

この人の話を聞きたいと思った人が，すごく有名な人だったりすることもあるだろう．そのときは，まず自分がどれだけその人の話を聞きたいのか自問自答してみよう．そして，共同開催者と相談してみよう．絶対にその人の話を聞きたいという確信が持てるのだったら，ぜひトライしてみよう．

逆に，その人が一度も人前で話したことが無いような人だったとしても，そこであきらめるのは早すぎる．誰だって最初は初心者だ．話がうまい下手は関係ない．本当にその人の話を聞きたいかどうかに集中しよう．その気持ちを誠実に伝えることができたら，きっと道は開ける．

もし適任者が思い浮ばなかったとしてもあきらめることはない．テーマ設定について適任者がいないかどうか，ウェブで探してみよう．共同開催者と一緒に探すといいだろう．

その分野には不慣れな場合，中心人物を一人選んだら，その人にその分野に他にどんな人がいるかを紹介してもらうという手がある．その分野の専門の人であれば，横のつながりでいろいろな人を知っているはずだ．しかし，これはある意味その人を共同開催者として仲間に引き込んでいるのと同じことなので，信頼できる人かどうか注意しよう．

金銭的な報酬がないイベントだとしても，イベントのテーマや目的がはっきりしていて，意識の高い参加者が多そうだと認められれば，出演者にとってはそこに無形の報酬としての価値を認められることもあるだろう．また，講演料は支払えないけど，懇親会は無料にするという条件でお願いするのも一案だ．

当然だが断わられることもある．落ち込むこともあるだろう．その場合は共同開催者に状況を聞いてもらったり，メンターに話を聞いてもらったりすると，気持ちが楽になる．

＊　＊　＊

出演を依頼した人，参加してくれる人，イベントに興味を持ってくれた人，全てに誠実に対応することを心掛けよう．たとえ一人一人に挨拶をすることはできなかったとしても，みんな興味を持ってくれているんだという意識を持って対応しよう．そうすることで，あなたが無報酬で動いていたのだとしても，報酬は思わぬ形でやってくるはずだ．

2.5　自分たちで作るイベント

リアルイベントを開催する際には，細々とした作業がたくさん発生する．リアルイベントでは日時と場所が決まっているため，当日のその会場でのみ発生する作業がたくさんある．具体的には，来場者の受付，会場内での誘導，資料の配布，会場設営・撤収，プロジェクタやネットワーク環境の整備などである．このような，リアルイベント開催にあたって発生する作業のことを，ここでは**スタッフワーク**と呼ぶことにする．また，このスタックワークを担当する人をスタッフと呼ぶ．自分と共同開催者だけで全てのスタッフワークをまかなうのは無理があるため，多くの場合には自分たち以外のスタッフが必要となる．

＊　＊　＊

さまざまなスタッフワークを手伝ってくれるスタッフはどうやって見つければいいのだろうか．

スタッフワークを頼めそうな知り合いもすぐには思いつかない．お金が無いので，さすがにイベントの専門業者に頼むわけにもいかない．

とはいえ，イベントの外枠は決定したし，人選には興味を持ってもらえる自信がある．ここまで来たら，きっと面白いイベントになるんじゃないかと思っている．

それゆえ：

参加者にイベント運営の役割の一部を担ってもらい，参加者が自主的に働けるフィールドを作ることで，参加者が自分達自身で作るイベントにしよう．

イベントの概要を決定して公開しているのであれば，そのイベントに興味を持ち，イベントを運営する側として参加したいと言う人もきっといるはずだ．イベントの規模や運営方針によって，必要とされるスタッフワークの量や質は異なる．動員数が多い大規模なイベントと社内勉強会のような小規模なイベントでは，必要

とされる準備作業には大きな違いがあるだろう．しかしいずれにしても，参加者にスタッフとしての役割を持ってもらい，自主的に動いてほしいのであれば，あらかじめ必要なスタッフワークを割り出しておくことが必要となる．小規模なイベントであったとしても，役割が明示されていないと動きずらい．

もし来場者が多く参加しているメーリングリストがあるならば，イベント概要とともに，必要とされるスタッフワークの一覧を掲載し，そのような役割を担当してもらうスタッフを公募してみよう．または，≪ウェブに公式ページを作る≫で公式ページがある場合には，そこで公募してもいい．

当日スタッフが担当する仕事としては，会場準備，会場撤収，受付，タイムキーパー，議事録係などがある．会場撤収時の片付けなどといった作業は，人がいればそれだけ楽になるし，参加者の連帯感を築くことができるので効果的だ．

この時，やりたくない仕事を押しつけるという意識であってはいけない．イベントを作るメンバの一員になってもらうという意識で対応することが重要だ．参加意識の強いスタッフの集うイベントは，メーリングリストだけでも，スタッフワークに関する事柄については，ひとつずつトピックと問題点が整理され，事前準備の議論が進む．それはあくまでも「自分たちでイベントを作る」という前提がある場合であり，そのようなメンバを募集するのだ．

スタッフが当日会場でどのように動くのかといった段取りに関する議論を，参加者からも見えるようなオープンな場で進めるといい．来場者も含まれているようなオープンなメーリングリストで，当日スタッフの配置などといった運営に関わる情報を流すと，スタッフはボランティアなんだということがみんなにも伝わる．そのような情報を得て参加する来場者は，スタッフの働きに対して配慮してくれるようになるかもしれない．

もちろん，個人情報など，スタッフだけが知るべき情報もある．そういった情報を共有するためのスタッフ間メーリングリストも別に作っておく．

当日スタッフは，事前にメーリングリストで役割分担を決めておき，当日の受付前の所定の時刻に手順の確認を行って，段取りを固める．

そして，現場でスタッフ自身が，イベントをすばらしいものにするに違いないと判断したならば，誰かに判断を仰がないで，自分で動いていいのだと伝えよう．その後に，どのような判断をしたのか伝えてもらうようにしよう．当日，スタッフではない一般の参加者にも，スタッフたちがすばらしい働きをしたことを伝え，一般の参加者がスタッフとして参加することによって作られたイベントであることを伝えよう．

イベント当日，準備が整い，受付が開始されたら，主催者は，スタッフや参加者が動くのを暖かく見守り，イベントを「見届ける」ことを注力しよう．最初から主催者が仕事をかかえた状態にしてしまうと，その仕事に気がとられて，問題が発生しても気づかなくなってしまうことがある．主催者は見守ることが大事な仕事だ．基本的に全ての当日に発生する仕事は担当するスタッフを決めて，主催者の手は常に空いた状態にしておこう．

当日のスタッフワークをうまく割り当てることができず，主催者自身がスタッフワークを担当してしまうと，本来必要な仕事を行えなくなる恐れがある．主催者は≪人選は誠実かつ大胆に≫で話者を選んだのに，せっかくのその講演を聞けなくなってしまうかもしれない．それは，設定したテーマや内容を次の展開に繋げる機会を失うことであり，非常に大きな損失である．そのため，主催者は講演者の話を聞くことが重要な仕事であることを認識しよう．当日はスタッフワークをしないという選択をし，全体を「見届ける」ことを優先するようにしよう．

もし当日のスタッフが自発的に動き，イベントを作り上げたのは自分たちであるという意識を持つようになれば，そのイベントは主催者にとって，最高の成功かもしれない．

＊　＊　＊

≪人選は誠実かつ大胆に≫し，≪オープンな募集≫ (後述) を行い，≪自分たちで作るイベント≫の準備ができたなら，当日は主催者であるあなたは，イベントで何が起こるのかを見届ける役目をするのがよい．

2.6　明朗会計

さて，無事にイベントを開催することができた．イベント終了後にやるべきことはなんだろうか．

＊　＊　＊

また次もイベントをやりたい．そのためにはイベント開催者として信用してもらえるようになりたい．どうすればいいだろうか．

イベント終了後はだいぶ疲れている．関係者の方々にお礼を言うのが精一杯だろう．できるだけ労力はかけたくない．

でも，イベントに参加してくれた人には，最低限の説明責任は果すべきだろう．何が最低限の説明責任にあたるのか．

それゆえ：

イベントの収支報告をウェブ上で公開しよう．

不明瞭なお金の動きが無いかどうかは，イベントを外部から見たときに気になることの一つだ．大きな会場を借りれば，ある一定以上の大きな金額のお金が動く．その場合には，もちろん参加者から参加費を徴収する必要があるが，参加者からすればそのお金が何に使われているのか気になるだろう．収支報告を公開すれば，お金の流れを明示できる．収支報告を明示していることは，イベント開催者への信頼につながる．

もし誰かの助けを借りることで会場代がかからなかったのだとしたら，そのことを明記すべきだろう．情報を公開することで初めて重要な協力者の存在がわかり，その貢献を次につなげることができるようになる．

主催者とは別に，金銭管理者を置くのもいいだろう．自分の名前ではなく，第三者の名前で収支報告が公開されていれば，より信頼感がある収支報告になる．

収支がプラスで利益が出ていた場合だとしても，金額を明らかにした上で，関係団体に寄付するなり，次回のイベント準備に使うなりといった用途を明示すればいい．

＊　＊　＊

イベント終了後はゆっくりと休もう．自分が何をしたいと思ってイベントを開催したのかを思い出し，イベント開催でその目標を達成できたかどうか，ゆっくりと考えてみよう．そして，次のイベント開催に向けて，英気を養おう．

3　パトレット

パターンにまで昇華できなかったが，小規模なパターンに近いものがいくつか見出された．ここにパトレットとして記録を残す．

3.1 ウェブに公式ページを作る

状況：イベントを開催することが決まった．
問題：まだイベントの周知が徹底していない．
フォース：
→イベント立ち上げ初期は誰もイベントの価値がわからない．
→しかし，イベントのテーマや関わる人々がわかると，それに興味を持つ人がさまざまなアクションを起こしてくれる．
解決：ウェブに公式ページを作る．
結果：イベントに興味を持つ人が実際にイベントに来てくれることにつながる．

3.2 事前に顔合わせをする

状況：ウェブや仲介者を通じて話者と知り合った．
問題：主催者と話者に面識がない．
フォース：
→事前に顔を合わせなくても，信頼できる話者であればいい話をしてもらえるだろう．
→しかし，話者によっては主催者の性格を事前に知っておきたいと思うだろう．
解決：事前に主催者と話者が直接会って打ち合わせをする．
結果：主催者のテーマに対する熱意を話者に伝えられるだろう．話者に安心してイベントで語ってもらえるようになる．

3.3 ポジションペーパを用意する

状況：集まった人同士は，知らない人同士なので，話がはずまない．
問題：知らない人同士がお互いの考えてることを手っ取り早く知りたい．
フォース：
→相手の名前やポジションがわからないと会話をはじめづらい．
→参加者は少人数なので，お互いに全員に紙を配布するといったことが可能．
解決：自分が今考えていることをポジションペーパ [8] という形式にまとめ，人数分用意し，互いに配布する．
結果：お互いが何を考えているのかわかるようになる．A4 一枚に語りたい内容がまとまっていれば，そのテーマについて初対面同士でも語り合えるようになる．これは勉強会などの比較的小規模な人数のリアルイベントを対象としているが，事前の顔合わせでも，ワークショップ形式のイベントでも，懇親会でも有効なパターンである．

3.4 大きな文字の名札

状況：リアルイベントには，さまざまな人が集まるため，お互いの顔を知らないという状況が発生しがち．
問題：会場で，名前を知らない人には声をかけずらい．
フォース：

→会場には初めて会う人がたくさんいる．
→しかし，集まっている人はオンラインで活躍している人が多いため，名前（ハンドル名）がわかれば知ってるという人がたくさんいる．
→名前（ハンドル名）さえわかれば話しかけらる．
→一般的な名札は文字が小さすぎて，近寄らないと文字が読めない．
解決：名札に大きな文字で名前を書いて，首からぶらさげる．例えば葉書 1 枚ぶんくらいの大きさの紙に大きく名前を書けば，遠くから見ても名前がすぐにわかるようになる．
結果：顔と名前が一致することにより，オンライン上での知り合いだった人同士がお互いの顔をわかるようになり，話がはずむようになる．

3.5 オープンな募集

状況：イベントの参加者を募集する．
問題：従来の募集方式では，どんな人が来るのかは主催者しかわからない．
フォース：
→来場者は，単に話者の話だけでなく，他の来場者としてどんな人が来るのかにも興味があるはずだ．
→イベントの性格から，誰がくるのかは公開してもいい情報である．
解決：オープンな場所（Wiki ページや**こくちーず**，**atnd** など）で参加者を募集する．
結果：そのイベントに誰がくるのかを，お互いにわかるようになる．参加者同士があらかじめ他にどんな人が来るのかを理解すると，ふだん会わないような人に会える期待感が増す．話者は聞き手を具体的に想像しやすくなり，想定聴者をつかめるため，話のレベルを合わせやすくなる．誰が参加するのかという情報そのものがコンテンツになって，人が人を呼ぶという現象につながり，熱意のある人が集まりやすくなる．もちろん参加を公開したくない人もいるだろうが，その場合は任意のハンドル名を認めるようにすればよい．

3.6 オンラインの感想

状況：イベント終了後，参加者がどんな感想を持ったのか，主催者は簡単には知ることができない．
問題：参加者の感想を聞く簡単な方法がない．
フォース：
→従来のアンケート用紙を配って感想を書いてもらうという方式では，アンケート用紙を配って回収するという手間がかかる．
→しかし，オンラインコミュニティを背景としたリアルイベントであれば，ほとんどの来場者はブログなどといった情報発信のための媒体を持っていることを期待できる．
解決：イベントの最中に，イベントの感想をブログに書いてもらうように呼びかける．「**帰って日記を書くまでが Wiki ばなです**」などという形でわかりやすく告知する．
結果：オンラインに感想が書かれることで，主催者は参加者のリアルな感想を知ることができ，モチベーション向上につながる．参加者にもよるが，アンケート用紙に鉛筆で書くよりも，ブログに書く方が書きやすいという人も多いだろう．また，アンケート用紙は主催者しか読めないが，ブログであれば誰でも読めるため，参加者が感想を書くモチベーションにもつながる．そのようにして公開された感想がイベントの価値を高め，次回以降のイベントへの布石となる．しかし，ブログのような公開の場だけだと，イベントに対する厳しい意見

を言いずらいかもしれない．そのため，感想通知用のメールアドレスなども告知しておいた方がよい．

4　おわりに

本稿では，リアルイベント開催に向けてのパターンランゲージを提案した．これまでリアルイベントを開催してきた経験のうち，特にうまくいったと思う部分をパターンとして抽出することができた．

筆者は，近年のリアルイベントの流行には大きな可能性があると考えている．実際に本稿を執筆した筆者らは，リアルイベントを通して知り合い，同じ問題について考えていることがわかり，本パターンランゲージを共に執筆するに至った．つまり，リアルイベントを通じて共著というコラボレーションに至ったことであり，その可能性を示すことができたのではないかと考えている．

これまで以上にリアルイベントの開催に力を注ぐ人が増え，異分野間の交流が進み，既存の枠を超えた接点が産み出されることを期待する．このパターンランゲージがそのような人たちの手助けになれば幸いである．

謝辞

これまで開催してきたイベントの関係者や参加者に，深く感謝の意を表します．

参考文献

[1] id:hanazukin: IT 勉強会カレンダー (2008). https://www.google.com/calendar/embed?src=fvijvohm91uifvd9hratehf65k@group.calendar.google.com

[2] Harrison, N.: The Language of the Shepherds: A Pattern Language for Shepherding, *Proceedings of the 6th Annual Conference on the Pattern Languages of Programs* (1999).

[3] 結城浩：絵本を読むときのパターン・ランゲージ (2001). http://www.hyuki.com/writing/ehonpat.html

[4] Wiki ばな：Wiki ばな (2004). http://wikibana.socoda.net/

[5] 日本 Ruby の会：RubyKaigi (2006). http://rubykaigi.org/

[6] 日本 Ruby の会：日本 Ruby カンファレンス 2006 (2006). http://jp.rubyist.net/RubyKaigi2006/

[7] 日本 Ruby の会：日本 Ruby 会議 2007 (2007). http://jp.rubyist.net/RubyKaigi2007/

[8] 江渡浩一郎：ポジションペーパ方式 (2004). http://eto.com/d/PositionPaper.html

神輿と寄合のパタン・ランゲージ

本橋 正成 (MOTOHASHI, Masanari)

+81-90-2303-2135
masanari *at* masanari.com

ABSTRACT

This paper describes a pattern language of MIKOSHI, which is a movable shrine, and YORIAI, which is a traditional Japanese meeting style, in today's local festivals in Tokyo. This paper also describes a pattern language of KUUKI, which is "atmosphere," when people communicate. These pattern languages are useful in community management, organization management, and software development.

1. はじめに

和を以て貴しと為す

十七条憲法

聖徳太子

20 世紀は、科学技術や工業化が西洋の文化に基づいて構築され発展し成果を上げてきた。特に情報技術やソフトウェア開発、製造などの分野でも経済的な成功を収めたと言えよう。しかしながら、それらの分野において細分化され全体が把握しにくくなり行き詰まりを感じるようになった。たとえば、品質や同調性に行き詰まりを感じていたソフトウェア開発手法において、東洋の文化や思想をルーツに持つ手法やプロセスにより成果を上げ一般的になった[“West meets East,” Coplien 他] 。同様に、企業経営やマネジメント、社会、組織、建築、工学などの様々な分野においても同様の傾向があることを実感している。

日本は、古来から海外より多くのことを学んでいる。特に、鎖国していた江戸時代が終わって西洋の文化や思想を学び、徹底的に取り入れて経済的な成果を上げ、日本の経済は大きな成果を出した。しかしながら、バブル経済の崩壊や金融危機などの情勢から閉塞感を感じている。ここで、自ら育んだ「和」を改めて考えてもよいのではないだろうか。

本稿では、東京のある町で行われている神輿（みこし）と寄合（よりあい）の二つの行事に注目した。伝統的な文化でありながら、現代の社会情勢や他地域の文化を取り込みながら変化しつつ町民によって行われている。今後、現代社会の大きな変化からの影響を受け、これらの知識や知恵が消えていく危惧を感じている。

現在の神輿と寄合についての知識や知恵をパタン・ランゲージとして表現した。次に、組織経営、コミュニケーション、ソフトウェアなどいくつかの領域や分野にとって有用である可能性と方向性について考察を加えた。様々な分野や領域でのヒントを得る基礎的な研究を目指している。

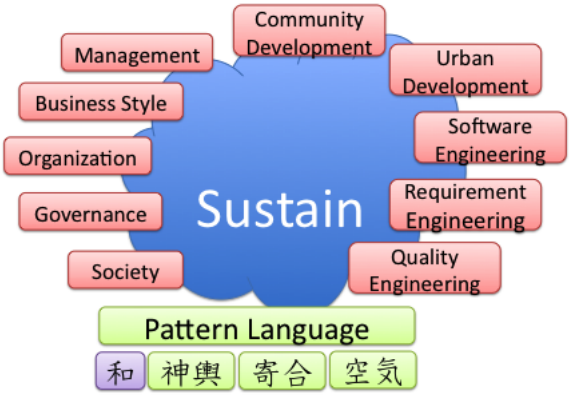


図 1 目的

なお、私自身が生まれ育った東京のある町の祭礼について、私の参加経験と氏子総代へのインタビューから記述している。他の地域で行われている祭礼の経験はなく調査も行っていないことに注意いただきたい。

2. 構成

本稿の章立てとその概要を紹介する。

3. 「祭礼と神輿」：とある町会における祭礼と神輿の全体像と歴史的な経緯ついて説明する。
4. 「寄合」：準備段階である寄合や会議についての紹介を行い、二つの形態の寄合についてパタン・ランゲージを示す。さらに、寄合の中でコアをなす場の空気についてのパタン・ランゲージを示した。
5. 「担ぎ手」：神輿の担ぎ手についての現状と、募集についての解説とパタン・ランゲージを示す。
6. 「神輿」：神輿の準備から、町を練り歩く行為から終了までを時系列に解説し、そのパタン・ランゲージを示す。

７．　「考察」：最後に、本稿についての考察を加える。

重要と思われるパタンについては「★」マークをつけ、特に重要なパタンには二つの★マークをつけたので、参考にしてほしい。

3. 祭礼と神輿

神輿、御輿（みこし、しんよ）は、神道の祭の際に、神霊が御旅所などへ渡御するに当たって一時的に鎮まるとされる輿である。輿であるから通常は担ぎ上げて移動するものを指して言うが、それを台車に乗せて曳くものなど、別形態のものを指すこともある。祭りによっては、御輿の巡行に、山車（だし）、鉾（ほこ）、だんじり、屋台が随行する場合もある [Wikipedia, 神輿]。

地元の神社において、神様への感謝の意味を込めて毎年九月に祭礼が開催され、その祭礼の中で神輿と山車が出される。神輿も宮神輿と子供神輿の二種類がある。宮神輿は、荘厳で重たいもので、町会の大きさに比べて立派であるため神輿友好会の方々も魅力を感じて担ぎにくる。一方、子供神輿は軽く、地元の小学生が主に担ぐ。今回の記事において、宮神輿の話題を主として扱いたい。

私の住んでいる地域にある神社において、二つの町会に住んでいる人たちがそれぞれ氏子会（うじこかい）を形成している。それぞれの町会ごとに氏子総代（うじこそうだい）がおり、毎年の祭礼準備の寄合（よりあい、会議）において、その年の祭礼で神輿を出すかどうかを氏子総代が決定する。たとえば、東京でも有名な三社祭のように不祥事が発生すると神輿を出さない決定がなされる。地元の神社においても、事件が発生した翌年に宮神輿を出さない決定がなされたこともあった。氏子総代の決定に従って、役員がそれぞれ調整を行う。

宮神輿の決定がされると、役員によっていくつかの実務的な調整がなされる。たとえば、

- 氏子総代によって割り当てられた予算の中から具体的な用途と費用配分についての調整
- 担ぐコースの調整と警察などに道路占有の調整
- 何カ所かの神輿友好会に日程を含む開催のお知らせと調整
- 神輿の担ぎ手たちにコースの途中で休憩を取り、その休憩場所でのお酒や食事が振る舞われる。休憩場所と担当、おもてなし内容の調整

などがあげられる。

20 年ほど前（1990 年）まで、誰でもその場にいれば神輿を自由に担ぐことができた。氏子以外の人々、たとえ組織的な暴力集団に属していても担ぐことができたが、背景にある暴力を示唆し威嚇しながら地元の氏子を押しのけて神輿を担ぐ行動をだんだんとするようになり、緊張が高まるようになった。ついに、20 年前に地元の氏子との間に二年連続で傷害事件が発生してしまい、重傷者が出てしまった。その結果、五年程度の宮神輿を休止した。その際、神輿同好会や組織的な暴力集団に声をかけた地元の人は、町会の氏子会にいづらくなり、参加しなくなった。そのような歴史と経緯から、和気あいあいとした寄合や会議であったものから、また違った形に移行した。

20 年前までは、神輿を担ぐ人（担ぎ手）は、神輿を担ぐために半纏（はんてん）を着る必要はなかった。事件が起こった翌年には半纏を着る約束をしたものの、そのルールさえも守られなかった。５年の休止を経て、神輿を担ぐ人（担ぎ手）は、必ず半纏を着る約束にした。神輿同好会の人は、その参加している神輿同好会の半纏を着るし、地元町会（氏子）はその半纏を着る。半纏によって、その担ぎ手が参加している団体がわかるようにした。

宮神輿の構造は地域や場所によって違うようだ。神輿は二本の縦棒が直接繋がっている親棒（おやぼう）がある。親棒の神輿より前の部分を「先棒（はなぼう）」と呼ぶ。この先棒は神輿の中心であり、この先棒を担ぐことはひとつのステータスとされている。宮神輿の場合、氏子が先棒を優先的に担げるように依頼している。

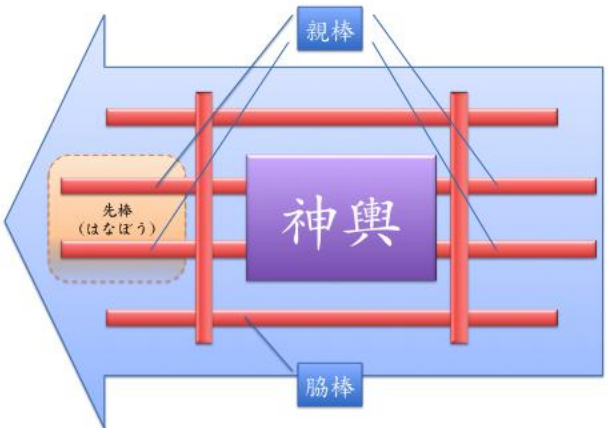


図 2 神輿の担ぎ棒

今回のパタン・ランゲージにおいて、運営側である氏子総代と役員の見地からのパタン、またひとりの担ぎ手としてのパタンの二種類の視点が入っている。

4. 寄合

4.1 寄合についての解説

神輿の準備は、「寄合（よりあい）」と言われる会議に従って決定される。寄合は、日本における町や村の協議機関であり、文字通り「よりあう」ことである。その町や村では商店街店主が参加する「商店街の寄合」、氏子役員が参加する「祭礼の寄合」、若い年代の人たちが集まる「青年部の寄合」のように話題や参加者によって異なる寄合が開催される。

寄合開催の日時は、毎年の行事である祭礼であるため、ほぼ例年通りということになる。参加者も大きく変わることはまれであるため、そのスケジュール観はおおよそ暗黙の中で合意されている。時期が近づくと、たいてい氏子総代があらかじめ根回しを行った後に「もうそろそろ」の一言で、長年に構築された阿吽（あうん）の呼吸により寄合が開催に向かっていく。そこでの決定事項や役割も、宮神輿の開催是非などすでに暗黙の中で合意されていることが多い。

宮本常一著「忘れられた日本人」で記述されている寄合は、時間を区切られることなく何日にもわたって続けられている。しかし、私の町会の寄合は、会場の都合などから時間が区切られている。事前に根回しで決められていたり、数時間が経過すると「担当者で決めて、後で報告します」ということも多い。また、参加者の中からしびれを切らして結論を急ぐことがある。このように時間感覚は現代的な要素が取り入れられたと実感している。しかしながら、その根底にある合意形成へ至る道は共通点を見いだせる。

4.2 親密な形態の寄合

寄合は、いくつかの形態が観察されている。日本企業における会議もおおよそのところで似ている。その共同体における実力者（神輿の場合は、二つの町会の氏子総代）によって根回しが事前に行われる。根回しは、寄合を開催する前に実力者によって合意形成をされる。その際の合意形成について記述する。

参加者が顔なじみで同じような考えや文脈を持つ場合の寄合について記述する。いわゆる議論 (discussion) のように正しい／間違っている、賛成／反対などのような対極の意見の優劣を決めるものではない。いくつかある意見を投票を行って、ひとつにしぼる選択的な会議形態とも異なっている。この形式の寄合が開催されると、寄合の議題、もしくは、どのような議題について話し合うかについても話される。たいていの場合、参加者がその問題に関する自分の意見を話していく。その際、反対意見も出されるし、議題から外れた意見も飛び出してくるが、その場の空気（雰囲気）でその意見についての取り扱いが決まってくる。その会議において、場の空気（全体的な雰囲気）が支配することが多い。この形式の寄合において反対意見は歓迎されるが、その場の空気の範囲の中でということになる。時間の制限や興味の限界に近づいてくると終わりそうな空気の中「これで行きましょう」と意見が出されて、たいていの場合はその空気に従い、他の参加者も「それでいいや」で終わる。

寄合を仕切るモデレータやファシリテータの役割は、町会長のような役職者とは限らない。参加者の中から自発的に役割が生まれ、ダイナミックに変わっていく。たいていは年長のものがその役割を与えたりもする。

<u>対話による根回しパタン ★</u>

<u>状況</u>

- 顔なじみの人たちで同じような文脈や地位を持っているごく少数の人たちが集まっている。
- 何を議論することかも決まっていない場合もある。

<u>問題</u>

宮神輿を出す・出さないなどの祭礼に関して事前に合意形成しておく。

<u>フォース</u>

- 数十年から数百年単位の歴史があるため短期的で直接的な方針ではなく、長期的に永続させる視点に基づいた方針が必要である。
- 伝統を大切にしているが、急激に変化している現代の潮流や文化などを取り込む必要がある。たとえば、団地やマンションのコミュニティ、コンビニエンス・ストアなどの影響や共同作業も検討される。
- 神事である祭礼や神輿の基本的な考えや、地域の人の性格や状況、過去の経緯を理解している必要がある。

解決

同じ文脈や立場を持つもの同士で徹底した対話を重ねることによって全体像の把握や方針の決定を行った。特に一部の参加者にとって快適ではない内容については、あらかじめ問答集を用意するなどの検討を行う。

否定的で非建設的な大人げない意見は、発言者が尊敬されなくなる可能性が高いため、基本的に回避する。ただ、そのような発言があった場合でも、その発言者の性格や状況を知っているため、周りの人は許容することが多い。

信頼された関係性が既に構築され、未来にわたって信頼関係を維持するために何らかの納得できる合意が得られるまで譲歩や止揚を含めて対話が続けられる。

結果

当事者（氏子総代）同士の認識の強化や共有がなされ全体方針が決定される。ただし、この決定された方針は、全員の意見を吸収したものではなく偏っている可能性があることに注意が必要である。

4.3 成熟度合い（守破離）が異なる寄合と、その関わり方

事前に根回しによって決定された方針が定義され、それに基づいた係の分担がされる。たとえば、「氏子の半纏を用意するのは洗濯屋にお願いできないだろうか」のように係へ依頼する。半纏の係は、半纏について任され自分の裁量の中で決定し行動する。具体的な中身までは指示されず、不明な手順についておおよそ「今まではこうしていた」という過去の経験や知識に基づいて判断される。

根回しで決定された方針に基づいて、祭礼についての寄合が氏子総代の声がけによって招集される。かつては新しく地域に参加した人は、自らの遠慮によってコントロールされていたが、この寄合は二十年前の事件を教訓により強化されるようになった。

この寄合においては、先ほどの形態の寄合とは趣向が異なり、新しく参加した人からの反対意見は歓迎されない。新参者の意見は、どんなに正しい場合でも原則として無視される。神輿に詳しい方が新しく参加し、その場を仕切ろうとした事例があったが、彼の意見は誰も聞かなくなり彼は参加しづらくなってしまった。つまり地域社会においても形稽古における「守破離（しゅはり）」に近いパタンが抽出できた。

形稽古（かたげいこ）は、芸道・武道・武術等における形を磨く為の稽古（けいこ）の中に守破離という言葉がある。守＝まずは決められた通りの動き、つまり形を忠実に守り、 破＝守で学んだ基本に自分なりの応用を加え、離＝形に囚われない自由な境地に至るというものである [Wikipedia, 形稽古]。 つまり形をしっかりと身に付けることではじめて、高度な応用や個性の発揮が可能になるということである。ただし、地域社会においては明確な指導者はいない点は、一般的な芸道や武道とは違う点である。

新しく地域社会に参加した場合、守破離の「守」の段階であるから、できる限り多くの人の意見を聞き、まずはその意見に従って地道に貢献することが求められる。その際、その地域でやり取りされているパタン（型）や共通言語の習得をしながら、その地域への貢献も行う。その後、守破離の「破」の段階に至り自分独自の応用を加え、離の段階で自分のものにしていく。周りのものは、新しい参加者をこのように時間をかけて成長させ、受け入れていく。

★ 共同体における守破離パタン ★

状況

長い地域共同体が過去から永続的に営まれている。その共同体において人の出入りはごく少なかった。

問題

今後も永続的に地域共同体を営むための共通認識を継続させる。

フォース

- その地域になじまない考え方や行動は大切であるが、そのままでは受け入れられないことがある。
- 長い間、地域社会に参加している者は、参加者の家族構成から状況を含めてほぼ全員が共有しているが、新しく参加した人については共有されていない。ただし、長くいれば参加しているだけでなく、共同体への貢献実績によって受け入れられる度合いが左右される。

解決

- 地域社会に新しく参加した人は、まずひたすら周りの人を見て聞いて、その地域に貢献し共通言語を獲得する。この段階にいる人の発言は、原則として聞き入れられない。
- その後、地域社会の全体を見通せ、自分自身の創意工夫ができるようになった頃には、自然と周りの人も発言を受け入れるようになる。
- その参加者の学習と成長は、形稽古における守破離と同様に分類される。

そのコミュニティが過去と未来にわたって永続化され、10 年から 100 年近い単位で代 (generation) を引き継いで構築されている。その結果、このような自分たちと異なっている考え方を持っている人や考えを受け入れているのではないだろうか。逆に、米国や新興分野においては、このような長い期間を前提にしたパタンは不適切であったのだろう。米国化した日本では、そのコミュニティや組織の期間を視野に入れ、適切なパタン言語を構築する必要がある。

4.4 場の空気

東洋、特に日本において場の空気が寄合や会議の進行や合意形成に大きな影響を及ぼす。場の空気とは、コミュニケーションの場において、対人関係や社会集団の状況における情緒的関係、力関係、利害関係など言語では明示的に表現されていない、もしくは表現が忌避されている関係性の諸要素のことなどを示す日本語の慣用句である [Wikipedia, 場の空気] とされている。

会議や寄合における汎用的なパタンとして場の空気を扱う。Jim Coplien 氏や Cristopher Alexander 氏が指摘している幾何学的模様 (geometory) の役割を場の空気が果たしている。その会議の話題がいかに提起され、話し合われていくのかを場の空気が支配し、場の空気を扱うことによって行われる。空気は、参加者が対話による多重で多様なフィードバックをかけることから生まれる。参加者は、多重なフィードバックによって、コミュニケーションの場における関係性やその背後にある意図を暗黙的に学習する。その背後に感じている関係性や予想される意図が空気の実体である。

空気は自然発生的に生まれてくる。ただし、その寄合や会議の種類によって、生まれる空気も同様に違うものになってくる。根回しをしているときなど信頼が既に構築されている場合は、柔らかで穏やかな空気が発生する。もう一方で経験者がその場を支配したい場合は、みんなから意見を集めながら、主として経験者の方向性と違う意見を無視するか、議題を変更する、場の空気に圧力をかけて黙らせるなどの手法がとられる。その場合、全体的な知を集めることはされず、一方的な寄合や会議になってしまう。

★ 場の空気を作り始めるパタン ★

状況

寄合や会議を始める段階で、すでに参加者が打ち解け雑談をしているなど、寄合や会議について熱心ではない日常的な状況である。

問題

参加者たちが真摯に寄合や会議に取り組むようにしたい。

フォース

- 参加者たちは、雑談や日常の事柄に気を取られているが、寄合や会議の空気を作りたい。
- 寄合や会議の空気をいったん引き締め、明確に寄合や会議を始めたい。

解決

「皆さん、お静かに」「開会宣言」「かけ声を掛け合う」などの何らかの儀式を行うことによって、明確に寄合や会議を始める。この儀式は「マイクをセットアップする」などの不明確な場合も可能である。神輿を担ぐときは、静粛にしてから拍子木を打ち、特に「ケ」から「ハレ」への移行を明示する。

柔らかな空気を作り出すパタン ★

状況

- 寄合や会議が開催され最初の段階で、参加者たちは空気を読むために努力している。
- あまり防御的になる必要がない、もしくは、参加者が十分に成熟度合いが高い場合である。

問題

参加者たちに率直な意見に聞いて、様々な人に偏らない空気を作り出したい。

フォース

- 新しく参加した人にとって、参加者が誰が何を考えているかわからない。逆に、昔からの参加者は、新しく参加した人が何を考えているがわからない。
- 新しく参加した人たちをもりたて、寄合や会議に多様性を生みたいと考えている。

解決

すでに立場がある、もしくは、声が大きい人が主体になって、一人一人にざっくばらんな意見を聞いていく。その結果、参加者がどんな意見を持っているのかが明確になってくる。

ただし、参加者が自分の意見を話しだした後の、空気を支配している人のリアクションによって結果は異なってくる。多様性を求める参加者にとってのアンチパタンとして、その意見を無視することや、圧を加えて変形してしまうことだ。たとえば、本来の目的探しの為にある「なぜ、なぜ、を五回繰り返すプラクティス」を空気に圧を加える目的に用い、相手が発言しにくくし、多様性を押さえ込む方向になる。

この新しく生み出された状況が、立場によって評価が異なることに注意が必要である。全体としてみると、お互いの空気の押し合いになるため、成功や失敗の判断は保留される。

空気に流されるパタン／空気に従わせるパタン ★

状況

寄合や会議において、新しく参加した。もしくは、参加者が増えた。その方々は、守破離の「守」と見なされる。

問題

新しく参入した人がその組織や共同体に早く馴染み、長い間、共同体と長い付き合いをしたい。

フォース

- 新規参加者は熱意はあるが、何をすべきか全体を見いだすことができない。
- 新規参入者は熱意はあるが、思いついたことを強く主張することは、遠慮の美徳から外されると見なされる。
- 既存の参加者は、新規参加者が何を考えているのかわからない。

解決

新規参入者は、まずはその場の空気を読むことから始める。誰がどのような主張をしたいのかを徹底的に理解し、想像することが重要である。例え、意見を求められても多くは語らず、簡単な感想や気づいた重要な点を述べるにとどめる。また、依頼された事項お願いされたことについて、自然体で受け答えするように努力する。

既に共同体に参加している人は、新規参加者のやる気が高いことを理解し、そのやる気（モチベーション）を削がないように配慮が必要である。具体的でわかりやすい指示を出すことによって、そのやる気を有効に活用し、コミュニティにとっても有効な働き手になる。もちろん、その働きについて適切なフィードバック「よくやっているね、ありがとう」を行うことによって仕事の品質向上と、新規参加者のやる気の存続を促す。

このパタンは、期間の短いプロジェクトや企業では有効な場合もあるが、逆に積極性がないと評価される可能性もある。まさしく空気を読んで実施したい。

空気をふくらませて支配するパタン

状況

寄合や会議、人が集まっている。その場合、意見や成熟度の一致度合いが少ない。

問題

寄合や会議、人が集まる場合において、意見の主体者が意図している結論に持っていきたい。

フォース

- 意見の主体者と賛同者が優位な方向性に結論を導きたいが、それとは違う意見を持っている人がいる。
- 意図している方向の正当性を主張したいが、認められない場合がある。

解決

戦略的に意図している方向に議論を持って行く方法がある。事前に根回しにより賛同者（取り巻き、応援者、follower とも呼ばれる）を得ておく。この賛同者は、ある特定の議題の意見だけでなく、その人に対する信用など全体としての賛同者であることが多い。賛同している人たちは、同じ文化や背景を持ち同じ方向性で語られることが多い。そのため、この事前に根回しについては、意識せずに習慣で行われている場合もある。

寄合や会議など人が集まっているときに、空気に何らかの圧力を加え膨らませる。たとえば、原理原則や正論を声を大きく主張し、その賛同者も同様に大声や威圧感を持って煽動活動を行う。発言は、意見の主体者でも賛同者でもよい。水を差された場合でも「前提」と主張したり、無意味に長い発言を行い他の人を混乱させ疲弊させることによって思考を停止させ、再び空気をふくらませ支配させる。

空気に圧力を加え膨らませる方法に、威圧を含めた沈黙も使われる場合がある。実質的な権力者や支配者が発言をさせないような雰囲気作りを行い、暗黙的な結論へ導くこともある。

結果

空気をふくらませ支配するパタンは、全体の状況を把握せずに意図する状況に基づいた原理原則や正論を使うことができるため、非現実的であったりモラルに反する結論に至る可能性がある。

なお、当パタンと次の水を差すパタンは、山本七平／小室直樹著「日本教の社会学」からヒントを得ている。

★ 水を差すパタン ★

状況

寄合や会議の中で空気が作られていくものの、原理主義的な方向になってきている。たとえば、全員が平等に担がなくてはならない、という正論に向けて全員が向いている。

問題

空気が膨らんだ状況になってきたので、現実的な方向に向けて引き戻したい。

フォース

- 原理には基づいているが、現実に即している方向に向かっていない。
- 全体の空気がおかしな方向に進んでいると感じているが、それを具体的に指示する勇気がない。

解決

- 事件が発生したなどの事実を示すことによって、空気を壊し、現実的な解に向けて話を進めることができた。
- 休憩を入れるなどミーティングを中断して、空気を壊し、その休憩時間中に別の根回しを行い、会議再開時に新しい空気を作り出す。納得性を高めるために再開したときにその状況を説明し、再度の意見を求める必要がある。
- 水を差すパタンは、新しく空気を作り出したいチャンスを作り出すときに有用である。しかし、働きは、そのときの空気を壊すだけで、その方向を正しく進める訳ではないので注意が必要である。

共有された場の空気を消滅させるパタン

状況

寄合・会議の開催、神輿を担ぐなど、コミュニティや集団でひとつのことを実施している。

問題

集団でしている一つのことを解体し、それぞれ個人や日常の生活に戻りたい。

フォース

- 寄合・会議や神輿など、みんなで一つのことを行ってきたが、一人一人に分解できていない。
- 場の空気が全体の進行や決定を行ってきたが、個人にそれらの責任を移行させたい。

解決

空気を作り出す宣言パタンと同様であるが、違った役割を持つ。明確に空気を切り替えることを宣言する。寄合が終わった後は、一本締めなど「明確に空気を切り替えるパタン」を使い、寄合や神輿などの「ハレ」から、場の空気が日常の状態の「ケ」の状態へ戻す。寄合の空気が亡くなることで、参加者の気分が変わる。その結果、少しの安堵感を感じた後、自分自身の作り出す日常に気持ちを切り替える。

---

5. 担ぎ手

5.1 担ぎ手の減少

氏子（町会住民）の神輿の担ぎ手は、減少傾向にある。いくつかの理由が考えられるようだ。

世界第二次大戦への政治による神道や宗教の利用、カルト宗教による大事件や反省などから、一般的な人は神事をさける方向にあった。ここ数年は、テレビ番組や自己啓発イベントによってスピリチュアル・ブームになっている。しかし、その矛先はメディアで取り上げられた地点や事柄に向かっていて、家族や地元を大切にしたり、鎮守の神社には向かわないと実感している。

東京地域以外の地方や地域から東京への人の流入が多く、相対的な人数としての地元民が減ってきている。地方から来た人の多くは、サラリーマンのような仕事をするために東京に来ていることが多く、東京地域の地元に帰化するために東京に来ている人は少ないようである。さらに、東京に引っ越してきた人が地元行事に参加することは敷居が高いなどの状況がある。東京に引っ越しをしてきても、祭礼や神輿を含む、商店街の餅つき大会のような行事への参加やコミットメントは減っている傾向にある。

祭礼や神輿の主役になる人の弱体化がある。地元で商店を営んでいる商店主とその家族が祭礼の主役になっている。しかしながら、スーパー・マーケットやコンビニエンス・ストアの台頭により、多くの商店では売り上げが激減し、昔からの商店は減少し絶滅の危機に瀕している。また、商店主の家族や子供たちも、売り上げが激減している商店より、企業に就職する傾向にあり、行事の参加者が全体的に高齢化が進んでいる。地元の行事を主体的に動かす人たちが減っている。

多くの日本人が感じているように価値観が移行している。昔、大切にされた価値観は、悪いことをすると罰が当たる、鎮守の神社仏閣は大切にする、人と人のつながりやご縁などであった。最近、主流になってきていた価値観は、回帰現象があるものの、直接的・短期的な収入に結びつく、娯楽性が高い、面倒くさくない、などに移行してきている。地元の行事は、現代風の価値観では享受できるメリットが見いだしにくいものであるため、参加人数が減ってきているようである。また、学生にとっても大学受験や留学、就職などで継続的な地元行事への参加が難しくなった。さらに、担ぎ手のプロフェッショナル化と神輿に関する事件の発生によって、地元民であったとしても祭礼や神輿に深入りすることを避ける傾向にある。

5.2 安全と募集のジレンマ

安全性確保と参加者募集のジレンマがある。事件の発生によって、神輿を担ぐためには、あらかじめ許可された半纏を着ることが強化された。その結果、半纏を入手していない地元民が祭礼開催当日に神輿を見て担ぎたいと思っても、半纏がないため神輿を担ぐことができない。その結果、新規に担ごうと思っている人の敷居を高くしてしまった。この半纏については、子供神輿については貸し出しの敷居が低く、この限りではない。

小さい町であり、住んでいる人は数十年にわたって住んでいる人たちであるため、ほとんどの人たちが顔見知りであった。逆に言うと、顔見知りではない人は他の町からやってきたことになる。そのことが危機管理に役に立っていた。見知らぬ人がいれば、声をかけるようにして、この安全性がより強化されてきた。しかしながら、その当時は大学生や会社員が殺される事件が相次ぎ、見知らぬ人とは関わらないような指導がなされた。そのようにして町の安全性がだんだんと低下してきたのである。

挨拶による安全・安心の拡大パタン

状況

コミュニティや組織が十分に小さいため、地元民の参加者全員の顔を覚えられる規模である。

問題

コミュニティや組織の独自性や安全性を確保したい。

フォース

独自性や安全性を確保したいが、コストをかけられない。

解決

さわやかに元気に挨拶をする。見知らぬ人でも少しずつ挨拶の環を広げていく。もし危険そうな状況や人であれば、警察に連絡する。

これらによって、地元民の顔を少しずつ覚えていけるようになる。挨拶を続けることによって、その人の体調や変更点に気がつきやすくなる。犯罪行為を未然に防ぐことが期待される。挨拶された側も気に留められているという安心感を得やすくなる。

ただし、人数が増えてきた場合や、人の流入／流出が多い場合、このパタンは有効ではないため、半纏などを使った「印（しるし、タグ、認識票）」が必要になる。

5.3 氏子の参加募集

現時点においては、これらの事情によって、祭礼や神輿に関わる人が少なくなっている。神輿については、神輿同好会の人の助けがなければ、氏子（町会住民）だけでは神輿を担ぐのに十分な人数が集まらない。つまり、神輿同好会の助けがあって、どうにか十分な人数が集まっている状況である。現時点では、この状況は変わらず祭礼や神輿についての再定義が行われている。祭礼や神輿に関わる地元の人（氏子）を増やすためには、これからの対応が必要である。

氏子の参加については、街角の掲示板に貼付けて祭礼と神輿についての日時の案内をしている。また、氏子が神輿を担ぐ場合は、役員共々担ぎやすいようにサポートするように周知徹底している。しかしながら、これらの試みは、今まで一度も神輿を担いだことのない人たちは知らず、成功しているとは言えない。

結局のところ、祭礼や神輿に参加してくれそうな地元民のひとりひとりに声をかけ、口説き落として参加してもらうことが主な方法である。たいていの場合、中身が見えないだけに初めて参加することは勇気がいるため躊躇することが多い。半纏の代理取得や、メンバーへの紹介の挨拶を行うなどの徹底したサポートを行っている。

ひとりひとりへの参加依頼パタン ★

状況

祭礼や神輿が行われるものの、参加者が今ひとつである。

問題

問題を起こさない信用できる参加者を増やす。

フォース

- 商店街のお客様など家族を含めて顔見知りになったが、まだ祭礼や神輿には参加していない。
- 祭礼や神輿に参加していないが、そのための時間や体力、家族の理解はある。
- 祭礼や神輿に参加したいが、そのための勇気や関心がない。

解決

可能性のありそうな人に声をかけ、参加を促すようにする。その際、前提や内容を説明し、経験や知識から注意点を事前に教えるなどのサポートする。このパタンは、参加人数を一気に増やすことに向かないが、少しずつ確実な参加者を増やすことができる。

5.4 神輿同好会の参加募集

日程が決定すると、神輿同好会の登録住所宛にお知らせして参加していただく。問題や事件を起こした同好会については案内しないなどの昨年までの実績を判断してから案内を出す。

信用ベースによる新規参加パタン

状況

神輿に新規に同好会が参加したい。もしくは、参加をお願いしたい同好会がある。

問題

トラブルを起こさず、新規に参加してほしい。

フォース

新しく参加している同好会の性格や実績について氏子の役員全員が知らない。

解決

新しく神輿を担ぎたい同好会の場合は、基本的に紹介ベースとなっている。今までの実績がある人から紹介された同好会は、「あの人の紹介だから間違いない」と許可される。一方、実績がない人が同好会を紹介した場合は、残念ながら許可されないようだ。このように新規に参加する場合も、信用ベースでの参入となっている。

このようにして安全と担ぎ手を確保した参加者の募集が行われ、現在のところ無事に祭礼が執り行われている。

半纏による参加者の判別パタン

状況

神輿を複数同好会と氏子により担ぐ（2009 年は 14 団体の神輿同好会）。

問題

複数の団体が参入で、暴力沙汰やトラブルを起こさずに神輿を担ぐ。

フォース

- 同好会の数を増やしたいが、安全も確保したい。

- 安全も確保したいが、同好会の組織員について顔を知らない。
- 誰がどの同好会に所属しているかを知らない。

解決

神輿を担ぐ人は、所属している同好会の半纏を着ることを義務づけた。開始時に、神輿同好会の責任者は、半纏を全員にお披露目して紹介を行う。その半纏を着ている人が暴力沙汰や泥酔などの問題を起こした場合は、その責任者に問い合わせ対応をお願いする体制にした。同好会の責任者にも、会員が問題を起こさないことを含め、当日の運営補助を依頼する。半纏を着ていながら問題を発生すると、自分だけでなくその同好会全体に迷惑がかかることが、安全を担保している。

安全は確保できたが、半纏を持っていない神輿を担ぎたい氏子が気軽に参加できなくなったデメリットが残されている。

---

6. 神輿

6.1 準備と宮出

神輿は、神輿に親棒（おやぼう）や脇棒（わきぼう）を麻ひもでしっかり締めるなどの準備を行われる。御神霊を神輿に移す神事（御魂入式）が行われる。

毎年、渡御のコースは検討され異なる。図 3 は、両町会を練り歩く典型的なコースの例である。最近は手配と人員の関係から全部のコースを担ぐことをせず、ひとつの町会を一年ごとに練り歩くことことが多い。本来ならば神社から担いで出て（宮出）神社に戻る（宮入）べきであろうが、ひとつの町会だけを練り歩くために事前に別の場所へ神輿を移動し、そこから担ぎ始めることが多い。

宮出（みやだし）は、神輿を神社から出すことである。神輿は馬（うま）と呼ばれる木製の台に乗せられて、担がれるのを待っている。人が三々五々に集まってくる。氏子総代や同好会の方の挨拶が執り行われる。簡単なルールやお約束事の説明がなされる。神輿の馬に拍子木を持った人が乗り、その人の合図とともに肩を入れて担ぎ始める。寄合における場の空気を作り始めるパタンを用いる。神輿の場合は、拍子木や一本締めを使って明確により強くモードを切り替えている。同パタンによって、その場の雰囲気や空気を「ケ」から「ハレ」にがらっと変える。事前に決めた地元のルートを練り歩き始める（渡御、とぎょ）。

図 3 典型的なコース例

6.2 神輿を担いで練り歩く（渡御）

白木の入れ物三宝（さんぽう）を持った人が先頭を歩き、通りがかったときに地元の人が紙に包んだお金を三宝に奉納する。なぜか、ここ数年は経験したことがないが、奉納した家の前で「神輿もみ（みこしもみ）」は神輿を左右に揺さぶり盛り上げる。

神輿の周りには担ぎ手と見物者が集まる。子供神輿は軽く一人で持ち上げられるが、今回取り上げている宮神輿はとても重たい。常に担ぎ手がどんどん入れ替わるようにして、元気が回復した担ぎ手を供給するようにしている。神輿の周りに担ぎ手も一緒に練り歩き、頃合いを見て「入る」。だいたい神輿の棒の一番後ろに人が入って一番先頭の人が抜けることが多いようだ。

★ 担ぎ手のライフサイクル・パタン ★

状況

神輿が担がれているとき、神輿の周りに担いでない見学者がいる。神輿を担ぐために必要な人数はそろっている。

問題

神輿を終わりの宮入までコースに従って担ぐ（渡御）。

フォース

- 神輿を担ぐことができる見学者が周りにいるにも関わらず、担いでいない。
- すでに神輿を担いでいる参加者がいるが、疲れている。
- たとえ、神輿の右前の一部の箇所が下がり水平やリズムを維持できていないなどの状況が不安定であるが、周りの見学者が参加しサポートしていない。

- 状態がダイナミックに変わっているが、担ぎ手の中には特に明確なリーダや管理者はいない。

解決

- すでに担いでいる参加者が疲れたと感じた場合、担いでいない見学者に声をかけて入れ替わる。
- まだ担いでいない見学者が神輿のある箇所が十分な人数がいないと感じた場合、自らの判断に従って担ぐ。

結果

担ぎ手と見学者のライフサイクルが構築された。

担ぎ手のライフサイクルにおいて、担ぎ手は間を見て神輿に入る、神輿の間を読みあわせる、高さを上げたりし、あえて間をずらし作り直す、神輿から抜けるときの間などがある。担ぎ手は、様々な間を支配し、支配されながら担ぐ。これらの間はパタンであり体を通じて学習される。

担ぎ手は、高さとリズムなどの間を読みながら担ぐ。担がれた神輿にリズムがつくと神輿自体の持つ物理的なエネルギーは大きいものになる。その結果、ひとりで独自のリズムを刻む、一人で独自の高さを維持することは困難になり、肩や体へのインパクトが大きい。たまに、ひとりで高い位置に神輿を「上げる」人もいるが、たいてい数秒で他の人の高さに戻ってしまう。リズムが崩れた場合は、担ぎ手の一人もしくは数人が声を上げてリズムや間を整える。たいてい「さー、さー、さー、さー」というかけ声が多いようだ。

時間はバラバラであるが、感覚として三十分弱ぐらいで一度の休憩「おもてなし」が入る。たいてい地元の婦人部が用意するお酒や食べ物が振る舞われる。たいていは地べたに座り歓談している。むろん酒も振る舞われるため、どんどんと盛り上がっていく。

6.3 宮入

だんだん神社に近づいていき、最後は宮入（みやいり）を迎える。ただし、宮入の直前は何度も何度も戻され、より盛り上げることが多い。最後は、馬に乗った人が拍子木で終わりの合図を行う（木を入れる）。

間（リズム）を壊して一体感を高めるパタン

状況

神輿の練り歩き（渡御）が宮入を目前にしている。

解決策

最高に盛り上げて、神輿を宮入させる。

フォース

- 宮入したいが、神輿を終わるのがさびしい。
- 担ぎ手の一体感はあるが、より一体感を増して宮入への欲求を高めたい。

解決

宮入できるように馬の上に拍子木を持って状況を把握している。神輿の棒をおさえて方向や速度をコントロールしている係に指示し神輿を押し戻してリズムを徹底的に壊す。担ぎ手は、それを受けてもう一度かけ声をかけて宮入に向けて進めていく。

拍子木を持った人は、満足するまで何度も何度も押し戻し、その度ごとに担ぎ手は体制を立て直していく。一体感が鍛えられ、より宮入への欲求を高めていく。クライマックスでは拍子木をたたいて、無事に宮入ができた。

神輿で練り歩いた後は一本締め（いよーっ、シャシャシャン、シャシャシャン、シャシャシャン、シャン）を行う。共有された場の空気を消滅させるパタンと同じ目的を持つパタンであり、「ハレ」から「ケ」に明確に空気を変え、祭りから日常に戻っていく。

7. 考察

今回は、神輿と寄合についての内省やインタビューを通じて観察とパタンへの抽象化を行ってきた。その作業の中で、いくつかの側面において考察を加えたい

7.1 社会的／文化的領域

アレグザンダーは、自立地域 (Independent regions) において「大都市地域は、それぞれが独自の文化を形成するほど小規模かつ自律的でない限り、安定しないだろう [pp.6, パタン・ランゲージ 環境設計の手引き, アレグザンダー他] 」と述べている。地域で自立的に行われている祭礼は、地域で実施されている行事のごく一部の側面を切り取っているにすぎない。しかしながら、アレグザンダーの主張する自立地域についてのプラクティカルな実例として考察できるだろう。

東京において、地域に住む他の人と関係を持つことが少なくなったと実感している。スーパーマーケットやコンビニエンス・ストア、大型ショッピング・モールの台頭により現時点で行事の主体となって動いている地元の商店街も収入と数が減ってきているため、地元の発言力や影響力も弱まっている状態である。宮神輿も地元の氏子だけでは既に担ぎきらず、他方からの同好会の方々の協力によって、どうにか成り立っている状態である。他の地域では人が集まらずに、宮神輿は出されずトラックに載せて回っていると聞いたこともある。自立性が減っている地域も多いのではないだろうか。

祭礼は、町に住んでいる人たちが集まって自立的に運営を行っているボトムアップ的な行為のひとつであろう。当たり前のことであるが、祭礼は法律や行政で規定されているものではない。

もし仮にアレグザンダーの言う自立地域に価値があるのであれば、挨拶による安全・安心の拡大パタンによって、町のつながりを暗黙的に増やし、ひとりひとりへの参加依頼パタンや信用ベースによる新規参加パタンが、自立地域の実現のごく一部に役立つかもしれないと期待している。自立地域の実現には、根本的で有用なパタンと全体の流れを変えるようなプラクティスが必要であるが、その実現如何に関わらず日常生活を豊かにしてくれると信じている。

7.2 組織経営やマネジメント領域

7.2.1 役割の組織と参加の組織

20 世紀は、科学技術と工業化の時代であった。事前に設計したものを同じものを同じように作ることが主流であった。そのためプロセスや規定をドキュメントとして事前に規定化し、それに従うことが求められた。この方法は、原子力発電所や飛行機の製造には有効である。その結果、ステレオタイプ的な捉え方ではあるが、多くの企業、特に大企業においては、トップダウンの組織とプロセスを作り、役割をベースに業務を行っている。社長や経営陣で決められたことを部門や課に分解して実施することが求められた。定型作業のみで現場に近い下の階層にいる人たちが、未熟（守破離の「守」）である場合は有効なモデルである。

この一般的なトップダウン・モデルに基づいた組織はトレードオフではありながらも、いくつもの決定的な欠陥があった。現場にいる人たちが成熟した場合は、内的動機付けの理由からも適切に組織が働かない。ひとつは横方向の情報伝達や情報共有および意思決定プロセスに存在し、もう一つは領域（業務領域、ドメイン）の分離である。“トップの位置で「事態を読み」、他のみんながこの「大戦略化」の指示にしたがうといったやり方では、もはやとうてい対処不可能なのだ [pp.10-11, 最強の組織、センゲ]。”

一方で、参加をベースにした組織もある。たとえば、最大 10 人程度のダイナミックなチームを作り、ひとかたまりの仕事を行う企業も増えている。たとえば、神輿が町を練り歩いている場にいる人たちを企業や組織の従事者もしくは登録している人と仮定し、担ぎ手をプロジェクトなどひとかたまりの仕事を行う関係者と考え、流動的でダイナミックな人員の配置を行う。担ぎ手のライフサイクル・パタンの知識がその組織の全体を構成する時のヒントのひとつになる。そのプロジェクトへの参加募集は、ひとりひとりへの参加依頼パタンや信用ベースによる新規参加パタンなども参考になろう。

現在、社会の変化は複雑で非連続に変化している。多くの業務において盲目で従うことより主体的に自分で考え行動することに価値が移行している。組織階層をトップダウンでひとりのトップが決めたビジョンを伝達されることによって組織するのではない。完全に個人の願望によって組織するのではない。スキルや個人の特性を大切にし、集団的に発現したリーダーシップと参加（メンバーシップ）によって組織することを考える時期に来ていると思う。

7.2.2 寄合の特徴と会議

現代の企業や組織では、会議は様々な様式があり、日々発展している。その中で、寄合のパタン・ランゲージを使用した寄合形式の会議が有用な場合と、他の形式が適切な場合を想定したい。なお、日本における一般的な会議は、賛成／反対 (pros/cons) を決める戦争のメタファで表現される討論形式より寄合形式に近いのではなかろうか。

寄合は、場の空気をコアにしている。場の空気は、その関係性の中で浮き出た場に基づいているため、役割や責任が個人に転嫁されにくい。全体としての責任が共有され分散されるため、個人の精神的な負担が少ないことが多い。逆に個人個人の精神的な負担が少ないが故に、空気が膨らみ、個人の意図とは違う方向に進んだり、急激で大げさな方向に進む危険性をはらんでいることは認識すべきである。空気を柔らかくするパタンや空気が膨らんだときは水を差すパタンを参照にしてほしい。

寄合における決定は、成熟度が高い「大人」であることが求められ評価される。全体の空気を読み現実的な側面も配慮しつつ個人への思いやりを考えて最終的な決定を行う。空気に基づいた決定時も「このようにしたらどうだろう」のように、あくまで質問や確認の形をとることが多い。逆に「このようにするべきだ」と決定した場合は、その発言者に責任が転嫁されやすいからである。

寄合は、じっくり時間をかけて言説（ディスコース）を作り上げられることに向いている。逆に言えば、短期間のプロジェクトや会議では寄合形式が向かないのではなかろうか。ブレイン・ストーミングから WBS (Work Breakdown Structure) に落としていくような形式は、短時間で行動レベルに落とすことができる。プロジェクトは時間制限があり少しでも時間を短くすべきと考える前提や文化に基づいているが、時間はたっぷりあり急がないことに価値を持つ文化もある。じっくりと納得いくまで共通認識を作り上げたい場合には寄合が向いている。このように継続的に漸進的な成長をしていく過程において寄合は力を発揮する。

実際の会議の形式自体を決定するときに、おいしいところを使ったり、場面によって使い分けることが重要であろう。また、現在の認識に基づいて寄合形式の特徴を記述したが、今後は修正しながら整理していきたいと考えている。

7.3 ソフトウェア領域

ソフトウェア開発においても、ヒントになるようなパタンがあると思っている。今回は、ピア・レビューのひとつの形態を紹介する。さらに要求工学やビジネス分析時点における方向性を示す。

7.3.1 寄合型ピア・レビュー

ピア・レビューは、正式なインスペクションをはじめウォークスルーなど、いくつもの手法がある。事前にチェックポイントを明確にリストアップし、それに基づいたピア・レビューは効果を上げている。しかし、複雑で不確定な社会的状況に対応したアジャイル開発プロセスやエゴレス開発においては、発見的な対話をベースにしたレビューがより注目されていくだろう。

寄合型ピア・レビューは、対話による根回しパタンがベースになる。その業務領域や専門知識について詳しくない初心者がいる場合は、共同体における守破離パタンを用い、状況に応じて対話を峻別する。空気に流されるパタン／空気に従わせるパタンを意識することになる。ただし、経験によると専門知識がない場合でも適切な指摘が行われた。

ピア・レビューは、場の空気を作り始めるパタンによって開始し、全体的には柔らかな空気を作り出すパタンによって運用した。場合によっては水を差すパタンによって軌道修正を行う。最後は、共有された場の空気を消滅させるパタンで完了する。

この寄合型ピア・レビューは、ある企業での継続的な 750 を越えるエゴレス開発の案件においても実績がある。その事例でのピア・レビューの終了条件は「相手を思いやり全員が納得すること」とし、全員が力を合わせ、言葉通りの「よりあう」ようにファシリテートした。この寄合型のピア・レビューは、あらゆるアウトプットとステークホルダーが対象で、要件定義／要求開発や仕様策定段階、テスト計画段階、本番移行手続などの対象で、内容と重要度に応じてユーザやマネジメントを含め行った。結果として、顧客満足度向上や工数削減、品質だけでなく教育や属人性排除の点においても高い成果を出すことができた。

寄合型ピア・レビューは、既存のプロセスとは違うため納得性の観点から導入のための説得が難しかった。特にドキュメントよりも人間関係を大切にするため、具体的な手順を記述しにくかった。当初は、正式なインスペクションのサブセットを手順書として用意し、周りの説得を行って導入した。

7.3.2 要求工学やビジネス分析に果たす役割

漸進的な実現プロセスにおいて要求に関しての寄合型ピア・レビューを行っている際に、新しい要求が獲得されたり修正／削除される事例が多かった。要求工学は、要求獲得、要求検証、要求管理、要求分析・仕様化の四つの分野があるといわれている。検証時に新しい要求が獲得されトリアージ（取捨選択）されることにより、要求検証と要求獲得のプロセスが曖昧になった。アジャイルや PDCA(Plan Do Check Action) サイクルのような循環をなし、漸進的なビジネス成長のコア・エンジンとして、寄合や空気に関するパタン・ランゲージが有用であった。

特に対話による根回しのパタンは、ビジネス分析知識体系や要求工学知識体系を用いた包括性や関係性を導き妥当解を得ることを補助するために有用であろう。勝ち負けという二者択一の議論ではなく、また否定的な意見を言わないというブレイン・ストーミングと違う。正しく使えば、対立したり矛盾する止揚や中庸を探し出すことに向いている。なお、Richard Nisbett 著 The Geography of Thought によると、東アジア地域の文化圏の特徴にある一般的な関係性や包括的に物事を見ることが前提となる。

ただし、本稿にあげたパタン・ランゲージだけでは、要求工学をやビジネス分析を行う現場でのプロセスを満たすためには抽象度が高すぎると考えている。現場をよく見つめた上で、具体的なパタン・ランゲージ発掘や創造が重要である。

7.4 場と関係性と時間のメモ

神輿の担ぎ手は、間（ま）の把握や理解がポイントである。間は、広辞苑第五版から抜粋すると「物と物の間」「リズムを生む為の休拍や句や句の感覚、転じて全体のリズム感」「ほどよい頃合い」「その場の様子」となっている。たとえば、参加している個人が筋肉があるかより、場との関連性に重きを置いている。一方、日本における会議や寄合の参加者は、場の空気がポイントである。場の空気は、コミュニケーションの場における関係性である。いずれも場との関係性を取り扱っている。さらに関係性は静的ではなく動的で時間の概念が欠かせない。場とのダイナミックな関係性が様々な分野の発展により重要なのかもしれない。

間は、物や音など身体に近い場との関係性であり、空気がコミュニケーションなど抽象的な場との関係性である。抽象的な空気は、間の例を体感することで理解を助けられると思っている。今後も研究と考察を進めていきたい。

8. おわりに

今回は、地元で行われている神輿を発端とし、準備段階である寄合、そのコアである空気に関するいくつかのパタン・ランゲージを抽出した。アイデアの一部ではあるが、簡単な考察を加えた。しかしながら、まだまだスタート地点に立とうとしたところであり、今後の発掘と検証が必要である。

神輿以外でも日常的な生活の片隅に価値あるパタンが潜んでいることを実感し、更なる発掘の必要性を感じている。和の再発見が様々な分野で役に立つことを期待している。

9. ACKNOWLEDGMENTS

今回のきっかけをいただき、相談にのっていただいた羽生田栄一氏に感謝を捧げたい。

「パターンライティングのためのパターンマイニング」で教えてくださった鷲崎弘宜氏と懸田剛氏に感謝している。パタンについて学ぶことができた。

コメントをいただいた野口隆史氏、大槻繁氏、綿引琢磨氏、玉牧陽一氏、梶田英邦氏に感謝したい。

Terunobu 'Terry' FUJINO 氏から本質的で的確なシェファーディングをしていただいた。多くのことを学んだだけでなく、コメントに感動し心が揺さぶられ涙したことが何度もある。心から感謝している。

二ヶ月に渡って神輿や寄合の話をしてくれた父に感謝している。いつまでも元気でいてほしい。夜な夜なマクドナルドやファミレスに通った私をサポートしてくれた家族に心から感謝している。

最後にあらゆる先人の皆様と、なにより本稿を読んでいるあなたに感謝したい。

:Author: 懸田 剛(Takeshi Kakeda)
:E-mail: takeshi@giantech.jp
:Created: 2010/01/06
:Updated: 2010/03/01

ふりかえりパターンランゲージ(進行役編)

Abstract

ふりかえり(Retrospective,Reflection)は、アジャイル開発において、最も基本的でかつ重要なプラクティスとなっている。ふりかえりは、チームにリズムを作り、継続的な改善をもたらし、チーム力を高めるアクティビティであるが、それまで「ふりかえり」を実施していないチームにとっては、迷い、戸惑う場面が多々あることも事実である。またふりかえりに慣れたチームに置いても、マンネリ化などに陥る危険性も多々ある。これらの諸処の問題のうち、本パターンランゲージは進行役についての問題に対する解決策を提示する。

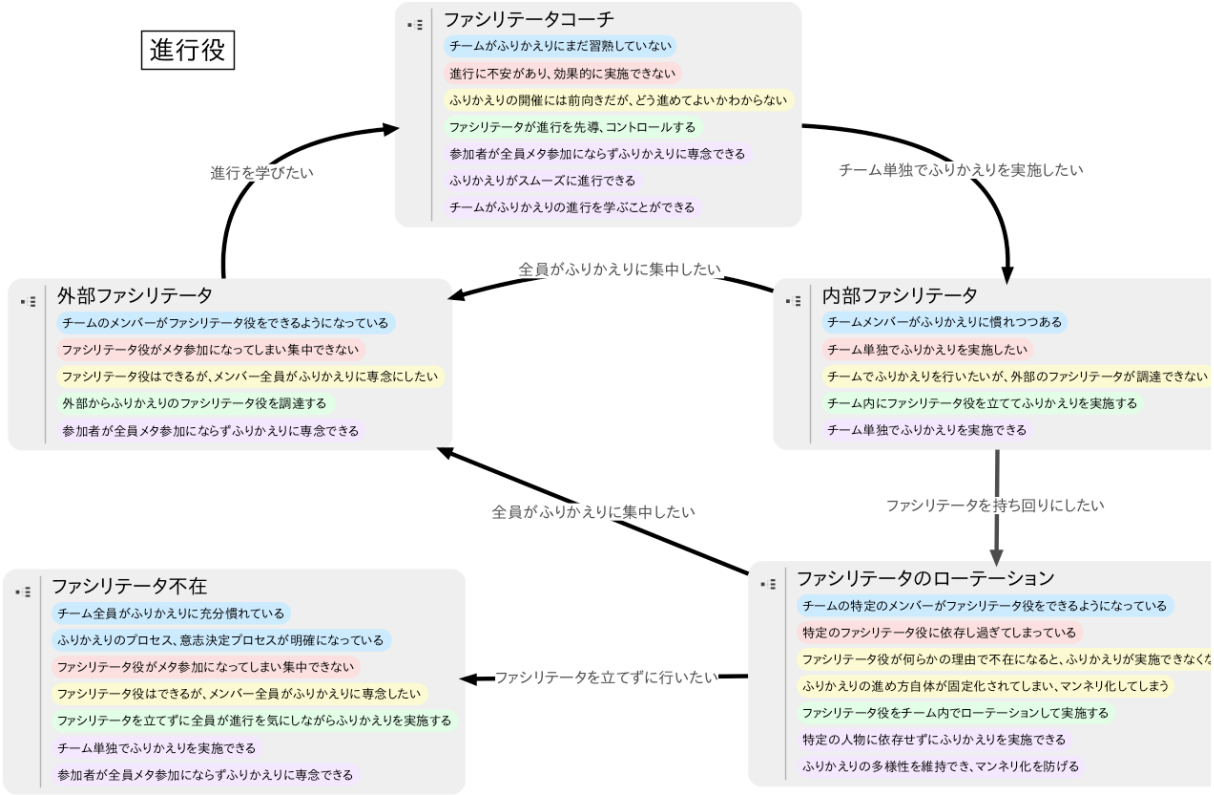
概要

本パターンランゲージは、ふりかえりに関してのパターンランゲージのうち、進行役(ファシリテータ)についてフォーカスしたパターンを紹介している。「ふりかえり」とは、日本国内のソフトウェア開発の現場において、日々の改善/カイゼンを実施するアクティビティとして、2005年頃より実施されてきている。特に国内では「KPT(Keep Problem Tryの頭文字)」と呼ばれるふりかえりの手法が主流となっている。

筆者の「ふりかえり」の経験上、大多数はKPTを使っているが、本パターンランゲージに関して言うと手法に関してはニュートラルな立場をとっている。そのためKPT以外の手法においても充分に適用できると想定している。[1]

[1] 国内でのふりかえりに関しては、「プロジェクトファシリテーション 実践編 ふりかえりガイド」(http://www.objectclub.jp/download/files/pf/RetrospectiveMeetingGuide.pdf)や、「アジャイルレトロスペクティブズ」(http://www.amazon.co.jp/dp/4274066983) が主要な手引きとなっている。

パターンマップ



進行役全体について

ふりかえりの進行に関しては**ファシリテータ**の存在が重要である。このファシリテータをいかに育てるかが、ふりかえりの成功のポイントとなる。本パターン群を用いることで、進行役について生じうる各問題を解決することができるだろう。

まずふりかえりを実施していないチームや組織においては、まず進行やポイントを実施しながら学ぶための**ファシリテータコーチ**を用いるのが最善の策だろう。しかしいつまでもコーチに頼っているわけにはいかない。そこで**内部ファシリテータ**をたてて、自分達の中でふりかえりを回すことができるようにする必要がある。しかしいつまでも特定の人物がファシリテータ役を専任で実施していると、他の構成員が彼に依存してしまったり、彼が組織やチームから抜けて不在になってしまうと、自分達でふりかえりを回していくのが困難になってしまう。そこで**ファシリテータのローテーション**によって、特定のファシリテータ役に依存しすぎないように、ファシリテータ役の平準化を実現する必要がでてくる。またファシリテータ役の構成員は、ふりかえりに対してメタ参加になるため、自分の

意見を発言する機会、あるいは進行などを気にせずに集中する機会が、他の構成に比べて減るため不公平感が生じてしまう。参加構成員がファシリテータのローテーションによってファシリテータ役に慣れ、またふりかえりの進め方が全員の周知する状態に進むと、全員がファシリテータ役つまり**ファシリテータ不在**によって、ふりかえり自体が更に自然に、かつ全員が参加者でありながらも、ファシリテータ役としての進行管理や場づくりを意識しながら参加できるようになってくる。構成員が充分に慣れた状態が生まれると、そこから硬直化が発生してしまう危険性がある。またファシリテータ役のメタ参加による発言機会の欠落が発生するため、あえて**外部ファシリテータ**によって、参加者のメタ参加を防ぎ、ふりかえりに対して集中できる。

ファシリテータコーチ

状況

チームはふりかえりの実施に不慣れである、あるいは未経験の状況の場合、どういう手順でふりかえりを実施するかがぼんやりとしている。

問題

進行に不安があると、ふりかえりが効果的に実施できない可能性が高い。

フォース

チームはふりかえりの開催自体には前向きではあるが、実際の経験が浅いため進行に対してのスキルセットが足らない。そのため、チーム全体に混乱を巻き起してしまう。

解決策

チームのふりかえりの初期段階には、ファシリテータ兼コーチを招聘して、ふりかえりセッションを進行してもらう。実際に進行してもらいながら、チームは進め方や、細かなポイントを体験しながら、身につけていく。

この際、ファシリテータはコーチ役にもなり、後に参加者自身だけで進行、実施できるように、各ポイントを説明したり、参加者の質問に答えたりもする。

結果

メンバー全員に、進行について混乱することなく、ふりかえりを体験してもらうことができる。ふりかえりの進行という行為を気にすることなく、ふりかえり自体の成功体験を積むことができる。このふりかえりの成功体験が以降の継続的な実施に繋がってくる。

また、進行を体験することで、コーチ役が不在でも自分で進行することができるようになってくる。

関連

本パターンは**内部ファシリテータ**の実施の前に実施しておくのが望ましい。こうすることで、自律的にふりかえりを実現できる土壌を育成できる。**外部ファシリテータ**を招聘するタイミングで、再度ファシリテータ兼コーチを雇ってもよい。

内部ファシリテータ

状況

チームの全員、あるいは一部メンバーは、ふりかえりを体験して、進行についても流れやポイントを掴めてきている状況である。

問題

それまでは外部のファシリテータに頼っていても、組織として自分達の改善を行いたい場合には、いつまでも外部リソースに頼っているわけにもいかない。

フォース

チームとして単独でふりかえりを実施してみたいが、外部のファシリテータを調達することができない。これはファシリテータ兼コーチとしてふりかえりを進行できる人材が不足している場合もあるし、そういった人材を招聘する予算がない場合もある。

解決策

ふりかえりを体験したことがある人物が、ふりかえりのファシリテータを買ってでる。この際にファシリテータとして自信があまりない場合は、自分から、参加者にその旨を告げ、協力を依頼するのが望ましい。

結果

チームで完結してふりかえりを実施することができる。この形は本来的にチームとしては最も望ましい状態である。チームを超えて組織という単位で見た場合においても、横展開を行う上でファシリテータができる人材が増えることはプラスの影響を与える。

関連

本パターンを適用後に、チーム内の構成員に展開したい場合は**ファシリテータのローテーション**を検討する。またいくら内部でファシリテータを調達できるとしても、ファシリテータ役のメンバーも含めてふりかえりに集中したい場合には**外部ファシリテータ**を考慮する必要がある。

ファシリテータのローテーション

状況

ふりかえりの開催にはメンバーが慣れてきている。ファシリテータ役をこなせる人もメンバーの中に現れてきた。

問題

ファシリテータ役の人が固定されてしまっている。そのためその人が不在だと、ふりかえり自体が立ち行かない。

フォース

ふりかえりには慣れてきたが、ファシリテータ役ができる特定の個人に依存してしまっている。このような状況では、ふりかえり自体の進め方が固定されてしまい、マンネリ化が促進する可能性が高くなる。

解決策

ファシリテータ役を各メンバーにローテーションとして回して実施してもらう。

結果

特定の人物に依存せずに、ふりかえりを実施することができるようになる。また進行の固定化を防ぎ、多様性を促進させる。この結果、常に新しいことにチャレンジできる土壌を作ることになる。

注意点

ローテーション初期の頃は、ファシリテーションが充分な状態であるとは言い難い。そのため各メンバーにファシリテータのフォローを依頼しておくことが必要となる。重要なのは全員がファシリテータを経験しておくことなので目先の効率性だけに囚われてしまってはいけない。

関連

チーム内で手本にすることができる**内部ファシリテータ**を演じることができる人物を最低一人は育成しておく必要がある。ファシリテータ役の人も含めてふりかえりに集中したい場合は**外部ファシリテータ**を検討する必要があるだろう。またローテーションが充分

に行われて、全構成員のファシリテートスキルが高い場合には**ファシリテータ不在**を検討してみるのもよいだろう。

外部ファシリテータ

状況

チームでふりかえりを実施することに前向きな状況になっており、チームのメンバーの中には、ふりかえりのファシリテータ役ができる人材がいる。

問題

チーム内のメンバーが、ファシリテータ役になってしまうと、ふりかえり自体に集中できないのが難点である。 これはファシリテータ役が、進行や場の状態を逐次把握しながら進行をするという理由に起因するものである。この状態をメタ参加と呼ぶ。

フォース

チームメンバーがファシリテータ役を引き受けて実施することはできるが、ファシリテータ役自身がふりかえりに集中できないというデメリットがある。

ファシリテータ役を買ってできる人物とは、大体においてそのチームの中でも優秀な人物の場合が多い。それ故に、その本人には、進行ではなく、ふりかえり自体、つまり課題の抽出、対策の発想に集中してもらいたい。チームとして自律しているが、それが故にスポイルしてしまう。

解決策

外部からファシリテータ役の人物を探してきて、ふりかえりの進行を依頼する。専門のファシリテータを準備するのが最適だが、別チームのファシリテータ役のメンバーに依頼するのでもよい。

結果

チームメンバー全員が、ふりかえりに対して参加者という立場に集中して参加することができる。このためメタ参加には陥らずに済むことができる。

注意点

ふりかえりの進行が各チームによって異なる場合には、ファシリテータによってその内容が異なる可能性がある。そのため事前にファシリテータ役と打合せを行って、ふりかえりの進行について合意しておく必要がある。

関連

単なる外部ファシリテータとしてだけでなく、進行も含めて学びたい場合には**ファシリテータコーチ**を検討するのがよいだろう。全員がふりかえりに集中したいという点からしても**内部ファシリテータ**が可能なスキルのメンバーがいる場合、または全員がファシリテータを演じることができる**ファシリテータのローテーション**時にこそ、本パターンは活きてくる。

ファシリテータ不在

状況

チームの中でふりかえりが習慣となり、根づきつつある。またチーム内にファシリテータ役ができるメンバーも増えている。

ふりかえりの進め方がメンバー間で合意がとれており、別段説明などをしなくても自然と進行ができる状態になっている。

問題

チーム内のメンバーが、ファシリテータ役になってしまうと、メタ参加になり、ふりかえり自体に集中できない。

フォース

ファシリテータ役を買ってできる人物とは、大体においてそのチームの中でも優秀な人物の場合が多い。それ故に、その本人には、進行ではなく、ふりかえり自体、つまり課題の抽出、対策の発想に集中してもらいたい。

解決策

特定のファシリテータ役を設けないでふりかえりを実施する。ただし参加者全員が進行を気にかけながら進めていく。

結果

特定の人物に負荷をかけることなく、全体としての進行を円滑に進めることができる。

注意点

特定の人物に依存しないため、物事の意志決定プロセスは、明確にしておく必要はある。たとえば複数の対策が挙がった場合にどうやって実施する対策を選ぶか、などである。意志決定プロセスを明確にしておかないと、決定に時間がかかり混乱を巻き起す可能性が高い。

関連

チームのふりかえりの成熟度を高めるために、まずは**ファシリテータコーチ**でチームにふりかえりを浸透させておく必要がある。また**ファシリテータのローテーション**を用いて、全メンバーにファシリテータの経験を詰ませておくことが不可欠である。問題やフォースが**外部ファシリテータ**に近いが対策が異なる。

情報システムのためのパタンランゲージに向けて

児玉公信
(株)情報システム総研

要約

ソフトウエア工学では，建築学で生まれたパタンランゲージというアイデアを，ノウハウの記述という側面を重視して，デザインパターンやソフトウエア・アーキテクチャ・パターンといった大きな成果を産んだ．しかし，パタンランゲージのもう一つの側面である“時を越えた道”と呼ぶ継続組織学習という側面を忘れてはならない．むしろ CIO（情報担当役員）は，この側面を重視して，企業情報システム全体の長期基本計画を緩く記述するものとして活用すべきである．個々の情報システムの企画から要求記述，設計，実装，利用，廃棄までのシステムライフサイクルでは，構想の記述（原要求）や設計の制約（デザインコード）としてこれらを活用する．本稿では，このための情報システムの構想から施工に渡って用いられるパタンランゲージを記述することを提案し，その一部を例示する．

1. パタンランゲージの再発見

パタンランゲージの考え方がソフトウエア工学に与えた影響は大きい．その偉大な成果である「デザインパターン」[1]やそれに続く多くのパターンカタログ，そしてそれらを生み出す原動力となった PLoP の活動[2]に敬意を表する．これらが，ソフトウエアの製造における品質向上に果たした役割は大きい．

しかし，これまでの情報技術におけるパタンランゲージの活用は，ノウハウの記述形式という側面だけに限られていた．これがパターンカタログはあるが，パタンランゲージがないという状況をもたらしていたと考えられる．そこで Alexander が考えていた元々のパタンランゲージとは何であったのかを，著者なりに整理したうえで，あらためて情報技術分野におけるパタンランゲージのあり方を考えてみる．

1.1 活動としてのパタンランゲージ

Alexander の著作[3-7]で述べられた建築プロジェクトでは，プロジェクトのルールが制定され，パタンを追加・修正し，パタンの使用に関して一定の拘束力を持つ運営組織が設けられる．本来のパタンランゲージとは，文書の形式のことではなく，“時を越えた建築の道”を行くための「活動」ととらえるべきではないだろうか．

(1) ルール

建築プロジェクト開始の前に，パタン活動を支える原則的なルールが制定される．ルールとは，プロジェクトの遂行において，さまざまな活動の原則を示したものである．

「オレゴン大学の実験」[3]では，①有機的秩序，②参加，③漸進的成長，④“パタン”，⑤診断，⑥調整となっている．「パタンランゲージによる住宅の建設」[6]では，①アーキテクトビルダ，②ビルダーズヤード，③共有地の共同設計，④個々の住宅のレイアウト，⑤一歩一歩の施工，⑥コストコントロール，⑦プロセスの人間的なリズム，となっている．

(2) 計画評議会

「オレゴン大学の実験」では，パタンランゲージの制定，改訂を諮る組織が設けられた．これは，施主（Owner, 大学），利用者（Customer, 学生），アーキテクトの三者からなる．施工者は含まれていない．ただし，これ以外の Alexander のプロジェクトで，計画評議会に相当する組織を設けたかどうかの明確な記述はない．

(3) 計画ランゲージ

「パタンランゲージ」[4]によると，その使い方は，253 あるパタンの中から，建築しようとするテーマに沿って，語り始めのパタンを探し，前後のパタンを参考にして，パタンをいくつか取り出して，必要に応じて変更したものを，そのプロジェクトで使う「計画ランゲージ（Project Language）」とする．これは一つの詩であり，それを語る順序が設計の順序に対応するとされる．

どのように語るかについて，「パタンランゲージ」には「街路を見おろすテラス」で始まる具体例が載っている．もう一つ，彼が手がけた日本の盈進学園東野高校の計画ランゲージの例[8]がある（図 1）．ここで，下線部分がパタンであり，これに対応するパタン集が別に作られているはずである．この記述を読むと，あたかも建物の中を歩いて行く（walk-through）ように書かれ，これから出来上がる学校のイメージがありありと浮かぶ．それは，「静かである」とか「憧憬を抱く」といった感情語が多く用いられていることからもわかる．これこそが，「詩のように述べる」と言われる所以であろう．

2-3 玄関道
- 正門より内境界に向けて「玄関道」がある。玄関道の両側には壁か樹木が並び非常に静かである。

2-5 中央広場
- 第2の門の内側には、中央広場がある。この広場は大講堂とともに構成され、講堂の正面は庭に向かっている。

2-6 大切な中心
- 中央広場の先に、第3の門を通り抜けると高校と大学の最も大切な中心がある。ここは幾重もの層を通り抜けて到達できる場であり、そこには静かさがある。

4-7 田の字センターの静かな地域
- 田の字センターの他半分は、より神秘的である。大学回廊に向かっている部分で、たぶん学生会館の裏手となる。この部分は、入り口からあるいは柱の間やアーケードを通して一瞥することができる。しかし、そこは静かで高校生には行くことができない。その部分を一瞥するたびに、高校生たちは、そこで受ける教育に対する憧憬を抱くのである。

図 1 計画ランゲージの例（一部）

計画ランゲージにおいて，パタン名は，施主，利用者，アーキテクトの三者間で通用する技術語彙として用いられる．たとえば，「玄関道」といえば，三者が直ちに理解する“あれ”である．誤解があれば修正すればよい．

(4) 現場でのイメージアップ

このほかに，Alexander は施主とともに建設現場に行って棒を立てて間取りを決めたり，紙でドアのイメージを書いて貼ったりして，現場で原要求を具体化する手段をとっている．これは，原寸設計の有効な方法と言える．

1.2 パタンランゲージのその他の側面

このほかに，パタンランゲージの活動に関して，特筆すべき特性がいくつかある．

(1) アンチ・マスタプラン

建築物は 100 年，200 年の時を越えて存在する．それを現時点で基本計画（マスタプラン）を立てて実行することは意味がない．前述のように，現代的都市計画においては，現状の都市基盤（インフラ）を活かしながら，住民の合意を得て，漸進的に街区を再開発していくしかない．

トップダウンの構想は必要だが，建設は小規模に，現場に根ざした活動としてボトムアップで行われる．これが，まったくの無計画に行われるのではなく，一つのパタンランゲージに基づいて行われれば，都市は全体として有機的秩序を保つことになる．それは，パタンランゲージが持つデザインコードという性格による．

(2) デザインコード

パタンランゲージは設計者の自由度を制約するルール，すなわち「デザインコード」の側面を持つ．これによって許される設計が限定されても，アーキテクトが設計過程で用いる探索空間が狭められることで，設計効率は上がるだろう．異なるアーキテクトによる設計であっても，住み心地や景観上の秩序は守られるだろう．

(3) 学習する都市

パタンランゲージが建築プロジェクトからのフィードバックを受けて，追加修正されることで，その都市建設における良いパタンが蓄積され，長い時間の中で引き継がれていく．これは，都市自身が学習していく過程と見なすことができる．それが「無名の質」の本質なのであろう．

1.3 パタンランゲージの利点と欠点

長い時間をかけながら，有機的秩序を保って作られていく都市にあって，住民が生き生きと生きることこそ，Alexander が見た無名の質であろう．反面，非現代的な都市が作られることになるかも知れない．これを欠点と見るかどうかは意見が分かれる．

しかし，たとえば「4 階建ての制限」[4]というパタンが，高層建築の非人間性を指摘していたとしても，高層建築が良いと計画評議会が判断すれば，そのようなパタンを作ればよいだけの話である．Alexander の「パタンランゲージ（A Pattern Language）」は一つの例であり，これとパタンランゲージの活動とは別物である．後者については，継続的な合意形成の手法として，その有効性を試みる価値はある．

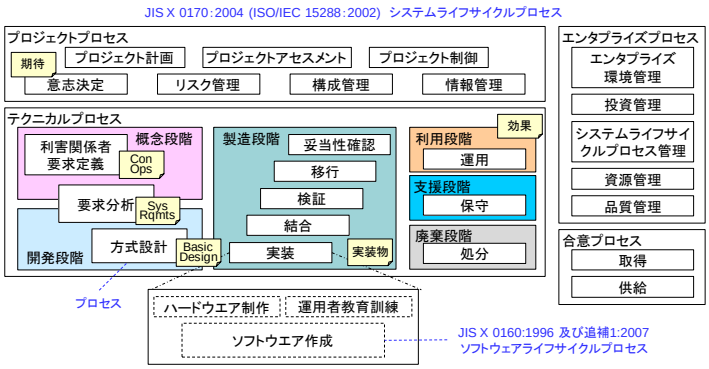


図 2 システムライフサイクルサイクル（ISO 15288:2002）

2. 情報システムパタンランゲージの構想

国，社会，企業，組織は，それぞれが人間活動システム[9] であり，それらは相互に作用しあって上位システムを構成することがある．情報システムは，それらの人間活動システム内の活動を支援するしくみである．そのため，情報システムは，その人間活動システムがそれまでに受け継いできた伝統や文化を色濃く反映し，これからも長期間にわたってその人間活動システムを支え続けることを期待される．

2.1 情報システムとソフトウエアの違い

まず，情報システムとソフトウエアの違いを明らかにしておく．

(1) ライフサイクルプロセスの違い

図 2 に，ISO 15288:2002(JIS X0170-2004）のシステムライフサイクルプロセスの定義を示した．これによると，システムライフサイクルプロセスは，ソフトウエアライフサイクルプロセスを包含し，さらにシステムの構想，利用，保守，廃棄，投資計画，調達計画までを含む広範なプロセスであることがわかる．ここで，利害関係者要求定義，要求分析，方式設計などのプロセスが明確化された点は特徴的である．それぞれの成果物を，ここでは，Concept of Operations[10]（原要求），システム要求仕様書[11]，基本設計書[12]としておく．ただしここでは，児玉[13]が定義した“情報システムサイクル”におけるシステム評価と新たな問題の形成，すなわち次のサイクルの開始に対応するプロセスの定義が漏れている．

なお，ソフトウエアライフサイクルプロセスは，ISO 12207:2008 で規定されており，ソフトウエアの外部調達から作成，運用，保守までを含むが，あくまでソフトウエア要求に基づく活動である．

(2) 設計対象の違い

また，情報システムとソフトウエアの違いを，“学校を設計する”ことと“校舎を設計する”ことに模して考えることもできるだろう．こ

こでは，学校を情報システムに，校舎をソフトウエアに対比させている．設計者の姿勢は，この二つで大きく異なるだろう．

学校は，校舎という“もの”ではなく，学生が 3～4 年の間，多感な時期を過ごす学びの“場”であり，先生たちが働く仕事の“場”，いわば“こと”である．もちろん，校舎あっての学校だが，学校の設計がなければ，校舎の設計は意味がない．“もの”づくりの価値観は QCD（Quality, Cost, Delivery）の確保にあるが，“こと”づくりの価値観は EEE（Efficacy, Efficiency, Effectiveness）の実現にあると言える．

上述の盈進学園の計画ランゲージの例は，Alexander が学校を設計しようとしていたことを端的に示している．実際，学校を述べるのには詩が適しているとも言える．

(3) 変化を促す要因の違い

情報システムは，多くの顧客および利用者によって，長期間にわたって利用され，拡張され，変化してきたものである．そして，これからも変化し続けるはずである．情報システムの変化は社会環境の変化や社会的要請に基づくことが多い．

一方，ソフトウエアの変遷は技術的要因によると考えられる．技術的要因は，小さな幅の変化の連続であるが，ときとして大きく変化することがある．

こうした変化を，前もって予想し，その対策を立てておくのは意味がない．企業情報システムを作り，育てていく活動は，まさに「時を越えた道」であり，マスタプランよりも，設計知識の継承と発展，緩いデザインコードによる有機的秩序を求めるべきであろう．ここにパタンランゲージの活動を適用する価値がある．

2.2 情報システム版パタンランゲージ

以上の議論を基に，システムライフサイクルプロセスにおいて，パタンランゲージをどのように使うべきかの案を図 3 のプロセス図[14]に示す．

(1) パタンの作成者と利用者

パタンランゲージを作り，維持するのは計画評議会に相当する“行政組織”である．これが図 3 の上半分であり，繰り返しのプロセスになっている．個別の業務システムの構築プロジェクトでは，システムライフサイクルプロセスの段階に応じて，図 3 の下半分に示すように，既存のパタンランゲージを語彙やデザインコードとして参照する．特に，原要求は計画ランゲージとして記述され，施主とアーキテクトの綿密な対話によって作られる．

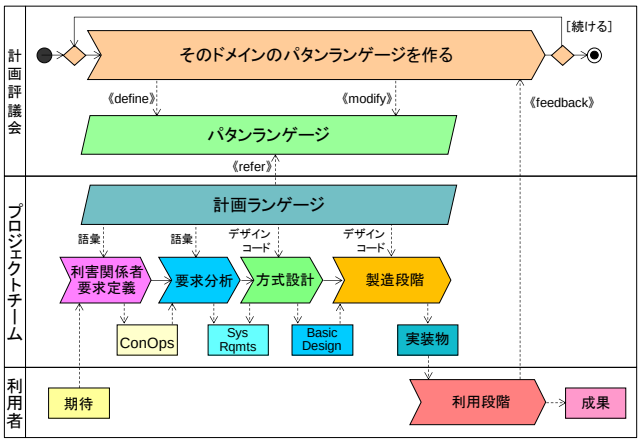


図 3 パタンランゲージとシステムライフサイクルプロセス

これは，行政組織が企業情報システムのマスタプランをパタンランゲージの形式で述べることに相当する．そこでは，行政組織を構成する施主の積極的関与が要請される．

(2) システム評価とパタンの評価

情報システムが利用段階になると，利用者はシステムを実際に運用した上で，利害関係者要求定義の良否，要求分析の良否，方式設計の良否，実装の良否などを総合的に評価し，施主にフィードバックすると同時に，計画評議会にもフィードバックする．計画評議会はシステムの評価を受けて，必要ならばパタンランゲージに修正を加え，それをパタンの利用者に広く通知する．これによって，特に，業務領域の原則，業界の掟など，明示されにくい知識も蓄積されていくことが期待される．これが，継続的に企業システム全体の質の改善に寄与する．

(3) 要求管理

原要求を語るに当たって，パタンを語彙として用いることで，原要求とシステム要求仕様書の対応づけが明確になる．それ以降の段階でも同様に基本設計書と実装のトレーサビリティが確保される．変更要求も同様に扱える．

(4) 現場でのイメージアップ

情報システムの構築において，現場でのイメージアップに対応するのはプロトタイピングであろう．原要求に対応するプロトタイピングは，技術的要素よりも，設計すべき“こと”がどうあるべきかを，想定するために用いられる．ここでは，アジャイルプロセスが現場でのイメージアップのために有効であり，施主の積極的な参加を結果的に促す．

3. 情報システムパタンランゲージの編纂

情報システムパタンランゲージの活動を行うに当たって，Alexander が自ら行ったように，その領域の参照モデルとなる一例としての「パタンランゲージ」があった方がよい．情報システムパタンランゲージは次のような手順で作っていく．

①個別開発ケースの要求記述を集めて原要求を設計順に述べるように語り直す．そのときに使われた有用な技術語彙を取り出してパタンフォーマットに書き起こす．
②収集されたパタンの間で，重複やギャップがあれば，パタン記述を修正・追加する．
③最後に，全体の構成が，社会および企業全体に関わるパタンから，小さな業務のプラクティスに至るまで通して書かれるように，再度，重複やギャップを調整する．

現在，この「パタンランゲージ」の編纂中である．候補名だけを

挙げて，内容を記述していないパタンも多い．実際に記述できるかどうかも，やりながら考えるしかないだろう．

現在の情報システムパタンランゲージの構想と，今のところのパタン記述の一部を付録に示す．この情報システムパタンランゲージは，Alexander の「パタンランゲージ」に倣って，大きなシステムに対応するパタンから並べて，「社会」，「組織」，「ビジネス機能」，「アスペクト」と題する章から構成される．最初の二章では，社会や組織のレベルの人間活動システムをそれぞれ支援する情報システムに関わる問題提起とその解決策を与える．三つ目の「ビジネス機能」では，企業活動を支援する情報システムのよいプラクティスを与える．最後の「アスペクト」は，未分類のパタンの置き場である．いずれ，適切な章に配置される．以上の章立ては，今後の編纂作業によって変わる可能性がある．

4. おわりに

企業情報システムの構築は，1980 年代の前半までは，自前で作るのが普通であった．1980 年代の後半からは，情報システムが経営戦略の一翼を担うようになって，巨大化，複雑化し，プロフェッショナルが代わって作る時代になった．これと並行して，多くの企業が情報子会社を作ることになった．このとき，発注者と受注者の関係が生まれたが，やがて本来発注者が行うべき情報システムの計画，期待と原要求の表明までもが，受注者の仕事にされてしまった．発注者が手放してしまったそれらの責任はやがて失われ，1990 年代後半からは，自称「ベストプラクティス」からなる「企業情報システム」がパッケージ化されて販売される状態になった．しかし，施主の主体性が失われたパッケージの導入は大した効果を生み出さない．

Zachman[15][16]のフレームワークを契機とする Enterprise Architecture（EA）のアイデアの意味は，本来，情報システムの計画，実施，運用の主権を施主に取り戻す活動にあったろう．施主や行政組織が，長期にわたって主体的に情報システムの実現に関わるとき，その活動のよりどころとなるのはパタンランゲージのようなゆるいガイドラインと，それが硬直化してしまわないように絶えず変化を受け入れる合議のプロセスである．

本稿は，このような活動，すなわち「時を越える情報システムの道」の実現に向けて，多くのアーキテクトの知恵を集結して，情報システムのパタンランゲージを編纂して行こうという初期的な提案である．

参考文献

[1] Gamma, E., et al, “Design Patterns,” Reading, MA, Addison-Wesley, 1995. 本位田真一ほか監訳，「デザインパターン（改訂版）」，ソフトバンクパブリッシング，1999

[2] Hillside Group, URL http://hillside.net/

[3] Alexander, C., “The Oregon Experiment,” Oxford Univ. Press, 1975. 宮本雅明訳，「オレゴン大学の実験」，鹿島出版，1977.

[4] Alexander, C. et al, “A Pattern Language,” Oxford Univ. Press, 1977. 平田翰那訳，「パタン・ランゲージ」，鹿島出版，1984.

[5] Alexander, C., “The Timeless Way of Building,” Oxford Univ. Press, 1979. 平田翰那訳，「時を越えた建築の道」，鹿島出版，1993

[6] Alexander, C. et al, The Production of House, Oxford Univ. Press, 1985. 中埜博監訳，「パタンランゲージによる住宅の建設」，鹿島出版，1991.

[7] Alexander, C., “A New Theory of Urban Design,” Oxford Univ. Press, 1987. 難波和彦訳，「まちづくりの新しい理論」，鹿島出版，1989.

[8] 羽生田栄一，personal communication, 2007

[9] Checkland, P., “Systems Thinking, Systems Practice,” John Wiley & Sons, 1981. 高原康彦ほか監訳，「新しいシステムアプローチ」，オーム社，1985

[10] IEEE 1632, “Guide for Information Technology –System Definition -- Concept of Operation Document,” IEEE, 1998

[11] Robertson, S. and Robertson, J., “ Mastering the Requirements Process, 2nd Ed. ,” Addison- Wesley, 1999

[12] IEEE 830, “Recommended Practice for Software Requirements Specifications --Description,” IEEE, 1998

[13] 児玉公信，「情報システムサイクルと原要求の記述」，日本情報経営学会誌，Vol. 28(2), 77-87, 2007

[14] Eriksson and Penker，鞍田友美ほか（監訳），「UML によるビジネスモデリング」，ソフトバンク，14-141，2002.

[15] Zachman, J. A., “A Framework for information systems architecture,” IBM SYSTEMS JOURNAL, Vol. 26(3), 1987. Reprinted Vol. 38(2,3), 1999

[16] Sowa, J. F. and Zachman, J. A., “Extending and formalizing the framework for information systems architecture,” IBM SYSTEMS JOURNAL, Vol. 31(3), 1992.

付録. 情報システムパタンランゲージ

A.1 パタンランゲージの構成

(1) 社会

社会と企業
1. 企業の役割

企業システムの輪郭
2. 都市計画
3. コンプライアンス
4. 企業間連携

環境と防災, 安全
5. 事業継続計画
6. フェールセーフ

自治的コミュニティ
企業間ネットワーク
部門システム政策

(2) 組織

機能階層
7. 4 階層の制限
8. 機能階層（IEC 62264-1:2003）

仕事コミュニティの形成
9. Customer-Performer（Medina-Mora et al, 1992）
10. 公共の空き地

煙突の通信路
小さなサービス
作業集団

(3) ビジネス機能

未来予知
11. 予定と実績（Kodama, 2010）
12. 資源の予定残（Kodama, 2010）
13. 曖昧な資源（Kodama, 2010）
14. 座席予約

時間とホライゾン
群体と個体
15. 観測と計測（Fowler,1997）

在庫の原則
16. 勘定（Fowler, 1997）
17. 価値, 位置, 状態の移動
18. 消費と占有

容器の原則
19. 入れ子構造

事実の記録
20. 仕訳と traceability

パーティと権限

(4) アスペクト

21. PDCA サイクル
22. 知識と実行
23. 報告
24. 必然と蓋然
25. 明日出会うもの（Tomono, 1999）
26. 能力（availability）
27. 能力を変化させるもの

A.2 パタン記述（一部）

11. 予定と実績

<写真：工場で製品が流れている>

概要

ビジネスシステムは，ほとんどすべて，活動の予定と実績，あるいは目標と結果を扱っている．

✤ ✤ ✤

背景

ビジネス活動の予定がどのような結果となったかを追跡することは，未来予知の精度を高め，失敗のリスクを下げる．しかし，現実世界では予定を無視してことが運ぶこともある．たとえば，1 件の注文に対して複数回に分けて納入する（分割納入）ことも，逆に複数の注文に対して一度にまとめて納入する（一括納入）することもある．また，注文が履行されないで終わることも，注文なしで納入が行われることもある．また，予定の製品とは異なる製品が納入されることもある．

ビジネス活動の結果を Fowler は勘定パタンの「取引－移動－保有」の三つ組で表すことを提案している．では，予定のビジネス活動をどのように表すべきか．

問題

実際は，予定と実績間の対応関係は自明ではないので，できるだけもっともらしい対応づけが必要である．

解決策

したがって，実績だけを記録するのではなく，予定を記録し，それをベースにして実績を把握すること．予定と実績の対応関係を記録する場合は，「取引」の下位型として「予定」と「実績」を設けること．この間の多重度は“0..1”対“0..1”ではなく，図 11-1 に示すように，“*”対“*”としておくこと．

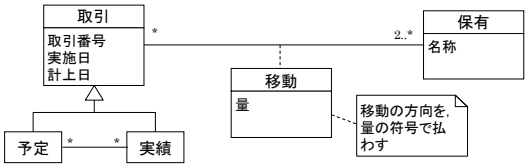


図 11-1　予定と実績の対応（一括実施）

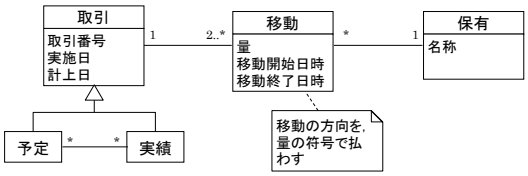


図 11-2　予定と実績の対応（分割実施）

一般に，同じ「取引」のインスタンスにおいて同じ「資源保有」のインスタンスとリンクすることはないので，「移動」クラスを関連クラスとする（図 11-1）が，製造取引などでは，同じインスタンスどうしでも「移動」が何回かに分けて実施されることがあるので，通常のクラスにしておく（図 11-2）．その際は，「移動」に移動開始日時，移動終了日時などのタイミングや順序を表す属性を追加する．

「予定」の「取引」の計上日は，予定が発生した時点で，実施日はその実施予定日である．「実績」の「取引」の計上日はこの事実を記録した日，実施日は実績発生日である．

「資源保有」の容量（および「資産保有」の状態）の導出処理において採用する「移動」のインスタンスは，原則として，過去については実績取引にリンクするもの，未来については予定取引にリンクするものとする．ただし，対応する「実績」の取引がない過去の「予定」取引の扱い，および現在実行中の「移動」については，個別のシステム要求で明示すること．

フォース

予定の取引では，「資源保有」を曖昧な“もの”として扱う場合が多い．この場合は「**曖昧な資源**」を参照のこと．

12. 資源の予定残

<写真：旅行計画を立てている>

概要

企業活動では，近未来を予測し，それに合わせて実行を制御する．このため，企業情報システムは，近未来を表すデータも扱わなければならない．「**予定と実績**」は近未来を表すように活用できる．

✤ ✤ ✤

背景

ビジネスの形態によって未来の情報の重要度は異なる．それは，予定が成就されない可能性，つまりリスクと，その影響の大きさに左右される．予定した未来がそのまま成就しないからこそ，近未来の在庫や状態を表示して，意思決定に役立てたい．

企業が持つ資源や資産の未来の状態または容量を常に把握し，それらを変化させるデータに敏感であることが，ビジネスの要諦である．これまでの情報システムでは，未来を規定するデータを持ちながら，それと意識しないためにデータが活用できていない．たとえば，注文は未来の出荷であり，予定の出荷日にその商品の在庫量を減じるものである．また，請求は未来の入金であり，予定の入金日に現金の残高を増加させることを意味する．

下図は，ある在庫単位の変化を累積で表した流動表と呼ばれるグラフである．この図で示すように，在庫は累積の受払の差である．未来の在庫は，過去の受払の累積値に予定の受払を反映したものとなる．過去の累積値の導出方法は Fowler の「勘定」パタンにおいて，「保有（Holding）」として述べられたとおりである．

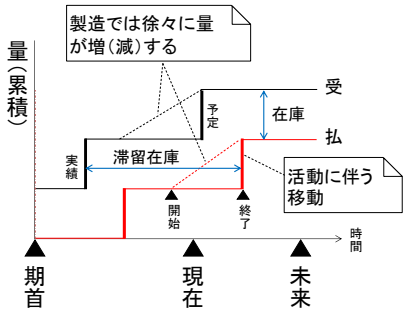


図 12-1 流動表のグラフ

問題

過去のデータだけを基に意思決定はできない．これまでの情報システムの多くは，過去のデータを集計して表示したものである．

解決策

したがって，把握すべき資源や資産について，時間による変化を把握する単位「保有」の概念を拡張して，「資源保有（Resource Holding）」および「資産保有（Asset Holding）」の二つを設定し，それぞれに「容量（availability）」，「状態（status）」なる型を設定すること．それらの属性「量」または「状態名」は，必要の都度，予定の受払を含めて導出すること（図 12-2）．

状態および容量の初期値，可能な下限値および上限値を規定する属性を持ってもよい．これらは当該の型のロールとして規定すること．

「容量」は，在庫量や設備能力のように，時間で変化する量である．在庫の引当や設備能力の引当は，これらの能力の残量（availability）が要求量を満たしている未来の時点を見つけて行われる．なお，引当は未来における残量の減少を意味する．

「状態」は，容量の特殊形であり，「引当済」，「ready」など，量の代わりに状態名（Categorical Scale）を使って表現される．

指定された時点における「資源保有」の容量および「資産保有」の状態の導出処理は，初期値に対して，その時点までの変動量（増減）の実績と予定を累積することによる．それらが必要となる時点で導出処理を行うこと．

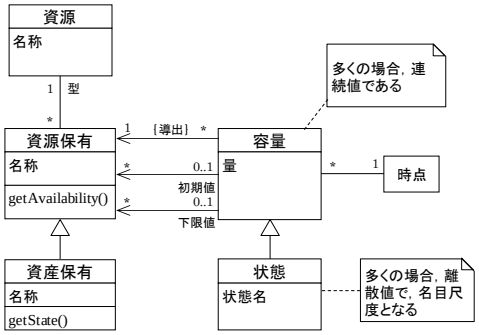


図 12-2 容量と状態

フォース

資源および資産を勘定として，容量の移動取引を「**勘定**」パタンで表現できることが前提である．さらに勘定パタンを拡張して，位置の移動を記録する場合は，資産に「/位置」の導出属性を設定することもできる．

容量の導出処理の負荷が大きい場合には，実装時の配慮が必要である．

同種の品目を均質であると見なす場合は，量を引き当てるが，同種の品目であっても別ものと見なす，すなわち個体を識別する場合は，特別な考慮が必要となる．「**個体と群体**」を参照のこと．

13. 曖昧な資源

<写真:鉄材を切っているところ>

概要

予定とは，確定していない曖昧なものである．

✤ ✤ ✤

背景

いくら予定を厳密に立てても，実績が予定と異なることはよくある．特に，製造業における“もの”の移動や変化は，厳密には予測ができず，予定と異なる製品ができることもある．だからといって，予定を立てなければ，必要な資源や資産を引き当てられず，スケジュールが立てられない．

むしろ予定の活動は，扱う対象物を曖昧にしておき，ある種の確率過程として確定するものとして記録するほうが効果的である．

問題

資源または資産を予定で扱うとき，実績並みの精度を強制すると，多くの予定情報が適時に記録されなくなる．

解決策

したがって，製品や部品，資源の現物の情報を曖昧に扱うこと．これらを，単純に識別番号で識別してはならない．“もの”は可変の属性の組として表現し，属性の組には，“未定”や“不定”，値の幅，発生確率などの表現を許すこと．

また，「もの」をそれ以上分けられないものとして扱わないこと．固体の状態では切断して使うこと，液体の状態ではその一部だけを使うことが当然のこととしてある．

属性の組は，「**観測**」パタンを用いて表現する．すなわち，対象と「属性種」間に関連クラスとして「属性値」を設ける．

図 13-1 において「資源」は型であり，知識レベルにある．これに関連する属性の組は型の仕様を与える．たとえば，机:資源に対して，幅:属性種，奥行:属性種，高さ:属性種，色:属性種，オプション:属性種をリンクし，それぞれ，120cm:属性値，60cm:属性値，80cm:属性値，白:属性値，{袖机，電源コンセント}:属性値を持たせる．このようにすることで，仕様の異なるさまざまな机が表現できる．ここで属性値に“未定”や“不定”などの特殊状況と，“60..65cm”などの幅を持った値の設定を許すことで，資源を曖昧に表現できる．

「資源保有」は型である「資源」の実体を表す．実体の扱いは，同種のもの全体を指す総称から，ロット単位，もの一つひとつを識別する個称というように，階層的になる．図 13-1 において「資源保有」を Composite にしない理由は，個体がさらに分割されることがあるためである．

「資源保有」の扱いが総称レベルなら，その仕様は「資源」の属性値の組で十分だが，個称レベルになればなるほど，個別仕様を記述する必要が出てくる．この場合も「**観測**」パタンの構造を取る．この場合は属性値に当たるクラスは，実現されている仕様の観測値である．観測値は時間によって変化するものもあるので，観測した「時点」を付けて導出属性として扱う．こうしてみると，容量や状態も，個別仕様の「観測値」として汎化できることに気がつくだろう．

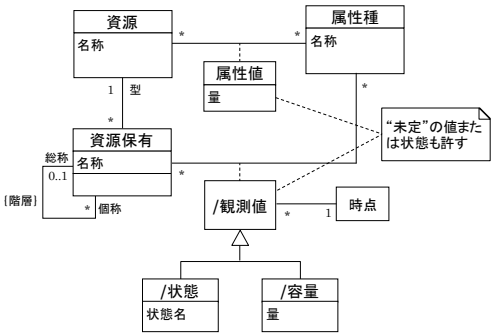


図 13-1 曖昧な資源

フォース

観測値が変化したことを「**勘定**」パタンの「取引」と「移動」表す．「資源保有」は分割されることがあり，個体（object）は個体（individual）ではない．「資源保有」が分割される場合は「**個体と群体**」を参照のこと．

ビジネスプロセスパターンの宣言的記述

Declarative Description of Business Process Patterns

竹村　司

Tsukasa Takemura

株式会社日本システムディベロップメント

NSD Co.,Ltd.

t-takemura@nsd.co.jp

概 要

ビジネス環境が激しく変化する中，企業がビジネスプロセスを迅速に構築かつ変更する能力そのものが競争力を左右するようになってきている．ところが，ビジネスプロセスは無形の実体の複雑な振る舞いであるため，ビジネスプロセスに関する知識を記述することが難しく，企業においてビジネスプロセスに関する知識を共有したり継承したりすることが困難となっている．このような「無形の実体の複雑な振る舞いを記述する」という活動はソフトウェアの開発と共通するものであるが，ソフトウェア開発においては様々なパターンを利用することによりソフトウェア開発に関する知識を共有を図ってきた．これに対してビジネスプロセスは Business Process Modeling Notation (BPMN) といったグラフィカルでかつ手続き的な手段で記述されることが多く，パターンとして知識を共有したり継承したりすることが困難であった．本論文では，ビジネスプロセスを宣言的に記述することによりパターン化でき，これによりビジネスプロセスに関する知識を共有したり継承したりすることが可能になることを示す．

1　はじめに

企業は，収益や顧客満足度といった指標値が基準値を満たすというビジネスゴールを達成するために，業務活動を行っている．その中でも，繰り返し行われる定型的な業務活動をビジネスプロセスと呼ぶ．ビジネスプロセスも企業における業務活動の一部であるため，ビジネスゴールもしくはその一部を達成することを目的として実行されている．

それにもかかわらず，市場の要求や組織構造といったビジネス環境の変化によって，既存のビジネスプロセスではビジネスゴールを達成できなくなったり，ビジネスプロセスそのものが使えなくなったりすることがある．このような場合には，企業は既存のビジネスプロセスを変更したり，新たなビジネスプロセスを構築したりしなければなければならない．特に，近年はビジネス環境の変化が激しく，ビジネスプロセスの陳腐化が早くなっているため，迅速なビジネスプロセスの変更や構築が必要となっている．

しかし，ビジネスプロセスを変更したり構築したりするためには，ビジネスプロセスに関する知識を企業内の関係者間で共有している必要がある．ところが，ビジネスプロセスは無形の実体の複雑な振る舞いであるため，ビジネスプロセスに関する知識を記述することが難しく，企業においてビジネスプロセスに関する知識を共有したり継承したりすることが困難となっている．

他方，このような「無形の実体の複雑な振る舞いを記述する」という活動は，ソフトウェアの開発と共通するものであるにもかかわらず，ソフトウェア工学の世界では，様々な方法論により，ソフトウェ

ア開発に関する知識を共有してきた．ソフトウェア開発におけるパターンの利用も，ソフトウェア開発に関する知識を共有するための手段の一つである．

このことから，ビジネスプロセスの記述においても，ビジネスプロセスに関する知識をパターンとして表現することによって企業内でのビジネスプロセスに関する知識の共有を促進することが期待できる．

ところが，これまでビジネスプロセスは Business Process Modeling Notation (BPMN)[7] [13] [15] といったグラフィカルでかつ手続き的な手段で記述されることが多く，ビジネスプロセスに関する知識を共有したり継承したりすることが困難であった．これは BPMN のような言語でビジネスプロセスを記述した場合に，ビジネスプロセスとビジネスゴールの関係を記述することができず,「なぜこのビジネスゴールを達成するためにこのビジネスプロセスを使用するのか」とか「ビジネスゴールが変更された場合にビジネスプロセスのどこを変更すべきか」といった知識を表現することができなかったためである．

本論文では，ビジネスプロセスを形式的に記述することで上述のようなビジネスプロセスに関する知識を表現できること，形式的記述により知識をパターン化できること，パターン化によって知識を再利用できることを示す．

2 例題

本論文で提唱するビジネスプロセスの宣言的記述の意義を説明するために，ここでは製造業における生産管理を例題として用いる．

ここでは，社外から購入した部材を加工・組み立てして製品を製造し，顧客に出荷する企業における部材購入と製品製造・出荷を計画するビジネスプロセスについて考察する．

この計画系ビジネスプロセスを以下のように想定する．

- 生産の８週間前に予定注文数を予測する．
- 予定注文数に基づいて，長期生産計画を作成する．
- 顧客は希望納期の３週間前に確定注文を出す．
- 顧客からの確定注文数と製品在庫数に基づいて，生産指示を作成する．
- 製品の製造に必要な部材の種類は２種類とする．
- 部材のリードタイム（手配量の決定から入手までの期間）を７週間と２週間とする．
- 部材の在庫と長期生産計画に基づいて，リードタイムが長い部材を手配する．
- 部材の在庫と生産指示に基づいて，リードタイムの短い部材を手配する．
- 生産指示，部材の手配を週に一度行う．

これらの需要と供給の値と決定時期の関係を図 2.1 に示す．

BPMN のような記法を用いて生産管理のビジネスプロセスを記述しようとする場合，製品の在庫が少なくなってから 生産を開始したり，部材の在庫が少なくなってから部材を発注するというように，実行するタスクの順序を記述することになる．しかし，例題のビジネスプロセスにおいては，顧客からの注文数が確定してから生産指示や部材の手配をすることはできない．また，生産済みの実在庫と未生産の生産計画を考慮して確定注文に対する引き当てを行う必要がある．このため，例題のビジネスプロセスを BPMN のような記法を用いて記述することは困難である．

また，このビジネスプロセスでは以下のようなドメイン知識が利用されており，このような知識を記述する必要がある．

- 将来の特定の日に提供可能な製品量は，現在庫量にその日までに生産が予定されている製品量と新たに加える新規の生産計画量を加えたものから，その日までに注文に引き当てられている製品量を差し引いた量である．
- 新規の計画する生産計画量は，その日に利用可能な部材量以内である．

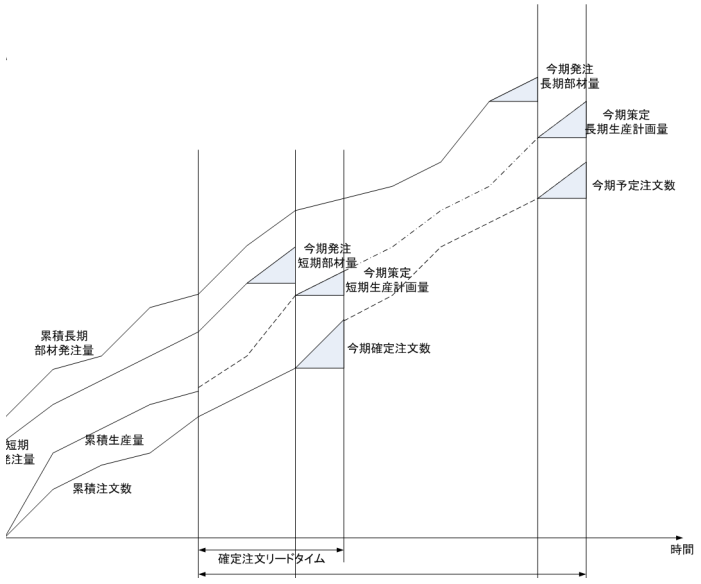


図 2.1: 見込み生産における需要と供給の値と決定時期の関係

- 将来の特定の日に利用可能な部材量は，部材の現在庫量に特定の日までに納入予定の部材量と新規の部材発注量を加えたものから，すでに製品計画に割り当てられている部材量を差し引いた量である．
- 新規の部材発注は，希望納入日が，発注日にリードタイムを加えた日付以降であり，発注量は発注可能量以下である．

本論文では，このようなビジネスプロセスを記述する手法として，ビジネスプロセスの宣言的な記述を提唱する．

3 既存のビジネスプロセス記述

企業が既存のビジネスプロセスを変更したり，新たなビジネスプロセスを構築しようとする場合,「なぜこのビジネスゴールを達成するためにこのビジネスプロセスを使用するのか」とか「ビジネスゴールが変更された場合にビジネスプロセスのどこを変更すべきか」といった知識を表現し，共有していなければならない．

もしビジネスプロセスの記述がこのような知識を表現していなければ，そのビジネスプロセスがどのようにビジネスゴールを達成するのかの根拠が曖昧となり，新たなビジネスゴールを達成するためのビジネスプロセスの変更や構築を行うことが困難となってしまう．

これまで，ビジネスプロセスは一連のタスクやアクティビティから構成されるという考えから，ビジネスプロセスの記述には BPMN に代表されるグラフィカルでかつ手続き的な手段が用いられてきた．

しかし，このような既存のビジネスプロセスの記述方法では問題に対する解決方法を示すことはできるが,「ビジネスプロセスがどのように問題を解決しているのか」や,「ビジネスプロセスの各タスクはどのように振る舞うべきか」といった知識を表現していない．このため，既存のビジネスプロセスにどの

ように変更すれば新たなビジネスゴールを達成できるのかが明確になっていない.

4 ビジネスプロセスの形式的記述

本論文では，例題のようなビジネスプロセスを記述する手法として，宣言的にビジネスプロセスを記述する手法を提案し，さらに宣言的にビジネスプロセスを記述することによってビジネスプロセスの記述をパターン化できることを示す.

4.1 ビジネス指標

企業においてビジネスの状況を判断するために用いる指標をビジネス指標と呼ぶ．企業のビジネス状況を正確に把握するためには，ビジネス指標は客観的に計測可能な値もしくは客観的に計測可能な値から導出可能な値でなければならない．客観的に計測可能な値の例としては，企業における製品の売上高などが挙げられる．また，導出可能な値の例としては，売上高と経費から導出可能な指標である利益などが挙げられる.

本論文ではビジネス指標を次のように定義する.

定義 4.1 (ビジネス指標) 企業のビジネス状況を正確に把握するための，客観的に計測可能な値もしくは客観的に計測可能な値から導出可能な値をビジネス指標と呼ぶ.

ビジネスプロセスを実行するうえで，達成すべき目標を定義する場合，判断基準となる指標とその基準値を定める必要がある．本論文では，これらの基準値をビジネス基準値と呼ぶ.

定義 4.2 (ビジネス基準値) 企業のビジネス状況を正確に把握するためにビジネス指標値と比較可能な目標となる値をビジネス基準値と呼ぶ.

4.2 ビジネス評価式

ビジネスプロセスは，企業においてビジネスゴールを達成することを目的として，繰り返し実行される定型的な業務活動である．このため，ビジネスゴールは達成されたか否かが客観的に判断できるものでなければならない.

そこで本論文では，ビジネスゴールの達成・未達成を判断するために，真偽値を返す評価関数を導入し，この評価関数をビジネス評価式と呼ぶこととする．この評価関数の値は時間の経過に伴って変化するため，ビジネス評価式は時間を変数とする関数であり，指定された時点でのビジネスの状況が望ましい状況である場合に真，望ましくない状況である場合に偽を返すものとする.

定義 4.3 (ビジネス評価式) 指定された時点でのビジネスの状況を示す，時間を定義域，真偽値を値域に持つ写像をビジネス評価式と呼ぶ．評価式の値は，指定された時間にビジネスの状況が望ましい状況である場合に真，望ましくない状況である場合に偽にとなる．ビジネス評価式 E を次のように定義する.

$E : t \in \mathbb{T} \mapsto E(t) \in \mathbb{B}$

ただし，$\mathbb{T} = \{t_0, t_1, t_2, ..., t_n\}$，$\mathbb{B} = \{true, false\}$ とする.

4.3 ビジネスゴール

企業は何らかの目的をもって企業活動を行っている．企業理念といった抽象的もしくは主観的な目的も存在するが，企業の状況を正しく判断するためには客観的に達成状況を判断できる目標を定める必要がある．本論文では，このような達成・未達成を客観的に判断することができる目標をビジネスゴールと呼ぶ．ビジネスゴールは，企業が企業活動を行うことによって達成しようとする目標の一つである．この目標は，特定の期間に達成したか否かを判定できるものでなければならない．このためビジネスゴールは，期間を定義域，真偽値を値域に持つ写像と考えることができる.

定義 4.4 (ビジネスゴール) 指定された期間に目標を達成したか否かを示す，期間を定義域，真偽値を値域に持つ写像をビジネスゴールと呼ぶ.

ビジネスゴール G を次のように定義する.

$G : (t_b, t_e) \in \mathbb{T} \times \mathbb{T} \mapsto G(t_b, t_e) \in \mathbb{B}$

ただし，$\mathbb{T} = \{t_0, t_1, t_2, ..., t_n\}$，$\mathbb{B} = \{true, false\}$ とする.

4.4 ビジネスゴールのカテゴリー

ビジネスゴールの値は，指定された期間中に目標を達成した場合に真，達成していない場合に偽となる.

本論文では，ビジネスゴールを以下のカテゴリーに分類する.

定義 4.5 (達成型ビジネスゴール) 指定された期間中にビジネス評価式が一度でも真になれば真となるビジネスゴールを達成型ビジネスゴールと呼ぶ．ビジネス評価式 E に対する達成型ビジネスゴール G_A を次のように定義する.

$G_A(t_b, t_e) \stackrel{\text{def}}{=} (\exists t \in \{t \mid t_b \leq t, t \leq t_e\})(E(t) = true) \Rightarrow true$

定義 4.6 (維持型ビジネスゴール) 指定された期間中にビジネス評価式が常に真である場合に真となるビジネスゴールを維持型ビジネスゴールと呼ぶ．ビジネス評価式 E に対する維持型ビジネスゴール G_M を次のように定義する.

$G_M(t_b, t_e) \stackrel{\text{def}}{=} (\forall t \in \{t \mid t_b \leq t, t \leq t_e\})(E(t) = true) \Rightarrow true$

4.5 ビジネスプロセス

企業において，ビジネスゴールを達成するために繰り返し実行される定型的な業務をビジネスプロセスと呼ぶ．本研究では，ビジネスプロセスが達成すべきビジネスゴールのカテゴリーに基づいて，ビジネスプロセスを達成型と維持型の２つのカテゴリーに分類する.

定義 4.7 (達成型ビジネスプロセス) 達成型ビジネスゴールを持つビジネスプロセスを達成型ビジネスプロセスと呼ぶ.

定義 4.8 (維持型ビジネスプロセス) 維持型ビジネスゴールを持つビジネスプロセスを維持型ビジネスプロセスと呼ぶ.

4.5.1 宣言的ビジネスプロセス記述

本論文では，ビジネスプロセスの形式的な記述を宣言的ビジネスプロセス記述と呼び，宣言的ビジネスプロセス記述を次のように定義する.

定義 4.9 (宣言的ビジネスプロセス記述) 宣言的ビジネスプロセス記述は，クラス，クラスの属性，クラスの操作，クラス間の関連，これらの要素に対する制約から構成される．これを次の九つ組 $\mathcal{M}$ で表す.

$\mathcal{M} = (\mathbb{E}, \mathbb{A}, \mathbb{O}, \mathbb{S}, \mathbb{C}, \alpha, \pi, \sigma, \gamma)$

ただし，$\mathbb{E}$ をクラスの有限集合，$\mathbb{A}$ を属性の有限集合，$\mathbb{O}$ を操作の有限集合，$\mathbb{S}$ を関連の有限集合，$\mathbb{C}$ を制約の有限集合とする.

また，α は写像 $\alpha : a \in \mathbb{A} \mapsto e \in \mathbb{E}$ であり，この写像はそれぞれの属性のスコープを指定する．π は写像 $\pi : o \in \mathbb{O} \mapsto e \in \mathbb{E}$ であり，この写像はそれぞれの操作のスコープを指定する．σ は写像 $\sigma : s \in \mathbb{S} \mapsto r \in \mathbb{E} \times \mathbb{E}$ であり，この写像は関連で接続される２つのクラスを指定する．γ は写像 $\gamma : c \in \mathbb{C} \mapsto x \in \mathbb{E} \cup \mathbb{A} \cup \mathbb{O} \cup \mathbb{S}$ であり，この写像はそれぞれの制約のコンテクストを指定する.

本論文では，宣言的ビジネスプロセス記述のうちクラス，クラスの属性，クラスの操作，クラス間の関連を UML[8] のクラス図を用いて表し，要素に対する制約を OCL[6]，[14] を用いて表す.

4.5.2 宣言的ビジネスプロセス記述の合成

本論文では，既存の複数のビジネスプロセスを組み合わせることで，新たなビジネスプロセスを定義できることを示す．このために，宣言的ビジネスプロセス記述の合成を次のように定義する.

定義 4.10 (宣言的ビジネスプロセス記述の合成) ２つの宣言的ビジネスプロセス記述の合成は，２つの宣言的ビジネスプロセス記述から１つの宣言的ビジネスプロセス記述への写像であり，次の写像 C で表される．

$C : (\mathcal{M}_i, \mathcal{M}_j) \in \mathbb{M} \times \mathbb{M} \mapsto C(\mathcal{M}_i, \mathcal{M}_j) \in \mathbb{M}$

ただし，$\mathbb{M}$ を宣言的ビジネスプロセス記述の有限集合とする．

ここで，元の２つの宣言的ビジネスプロセス記述を $\mathcal{M}_i = (\mathbb{E}_i, \mathbb{A}_i, \mathbb{O}_i, \mathbb{S}_i, \mathbb{C}_i, \alpha_i, \pi_i, \sigma_i, \gamma_i)$ と $\mathcal{M}_j = (\mathbb{E}_j, \mathbb{A}_j, \mathbb{O}_j, \mathbb{S}_j, \mathbb{C}_j, \alpha_j, \pi_j, \sigma_j, \gamma_j)$ とするとき，宣言的ビジネスプロセス記述の合成を表す写像 $C(\mathcal{M}_i, \mathcal{M}_j)$ を次のように定義する．

$C(\mathcal{M}_i, \mathcal{M}_j) = (\mathbb{E}_c, \mathbb{A}_c, \mathbb{O}_c, \mathbb{S}_c, \mathbb{C}_c, \alpha_c, \pi_c, \sigma_c, \gamma_c)$

ただし，$\mathbb{E}_c = \mathbb{E}_i \cup \mathbb{E}_j$，$\mathbb{A}_c = \mathbb{A}_i \cup \mathbb{A}_j$，$\mathbb{O}_c = \mathbb{O}_i \cup \mathbb{O}_j$，$\mathbb{S}_c = \mathbb{S}_i \cup \mathbb{S}_j$，$\mathbb{C}_c = \mathbb{C}_i \cup \mathbb{C}_j$，$\alpha_c = \alpha_i \cup \alpha_j$，$\pi_c = \pi_i \cup \pi_j$，$\sigma_c = \sigma_i \cup \sigma_j$，$\gamma_c = \gamma_i \cup \gamma_j$ である．

4.5.3 宣言的ビジネスプロセス記述のパターン化

本論文では，クラス，クラスの属性，クラスの操作，クラス間の関連，これらの要素に対する制約の一部をパラメーターとすることによって宣言的ビジネスプロセス記述を汎化したものを宣言的ビジネスプロセスパターンと呼ぶことにする．

宣言的ビジネスプロセスパターンに対しても上述の合成が適用できるとともに，記述内のパターンを具体的な要素で置き換えることによって，宣言的ビジネスプロセス記述を得ることができる．

5 ビジネスプロセスパターン

ここでは，ドメインモデルとそれに対する制約を宣言的に記述することによって，ビジネスプロセスをパターンとして表現できることを示す．

5.1 維持型ビジネスプロセス

維持型ビジネスプロセスはビジネス評価式の値が真であり続けることを目的とするビジネスプロセスである．換言すれば，維持型ビジネスプロセスは「ビジネス評価式の値が真である」という状態を維持することを目的とするビジネスプロセスである．

このため,「ビジネス評価式」と「ある状態を維持すること」を形式的に記述することができれば，このカテゴリーのビジネスプロセスを形式的に記述することができると考えられる．

時刻 T 以降「ある状態を維持すること」というのは，時刻 T に「ある状態にあり」かつ，時刻 T+1 以降も「ある状態を維持すること」である．このことから,「ある状態を維持すること」という目的を持つ維持型ビジネスプロセスを，Prolog 風に記述すると図 5.1 のようになる．

```
維持型ビジネスゴールを達成している :-
                    ある状態を維持している.

ある状態を維持している :-
                    時刻 (T) である状態にある,
                    時刻 (T+1) 以降もある状態を維持している.

時刻 (T+1) 以降もある状態を維持している :-
                    時刻 (T+1) である状態にある,
                    時刻 (T+2) 以降もある状態を維持している.
```

図 5.1: 宣言的に記述した維持型ビジネスプロセス

このように,「ある状態を維持すること」を宣言的に記述できる．このため,「ある状態」に達しているかどうかを判断するビジネス評価式を宣言的に記述することができれば，維持型ビジネスゴールが達成されるとはどういうことかを宣言的に記述することができる．つまり，維持型ビジネスプロセスは，このような宣言的記述とビジネス評価式を組み合わせることによって宣言的に記述することができる．

例えば，需給バランスを維持するビジネスプロセスにとって，ビジネス評価式は「需要と供給のバランスが取れている状態」で真となり,「需要と供給のバランスが取れていない状態」で偽となる論理式で表現できる．この論理式を維持型ビジネスプロセスのパターンの「ある状態」に当てはめることによって，需給バランスを維持するビジネスプロセスを記述することができる．

5.2 維持型ビジネスプロセスパターンの形式的記述

前節で述べたように,「ビジネス評価式」と「ある状態を維持すること」を形式的に記述することができれば，維持型ビジネスプロセスを形式的に記述することができる．統一モデリング言語（UML)[8] とオブジェクト制約言語（OCL）[6] を用いて記述した維持型ビジネスプロセスのパターンを図 5.2 に示す.

5.2.1 知識の名前

維持型ビジネスプロセス

5.2.2 コンテクスト

以下のようなビジネスプロセスで用いられる.

- 製造業において顧客からの注文に応じて生産計画を策定するビジネスプロセス
- 製造業において生産量に応じて部材を発注するビジネスプロセス
- 人材管理部門において今後の需要に応じて人材育成計画を策定するビジネスプロセス
- 金融業において預金額と貸付額のバランスをとるビジネスプロセス

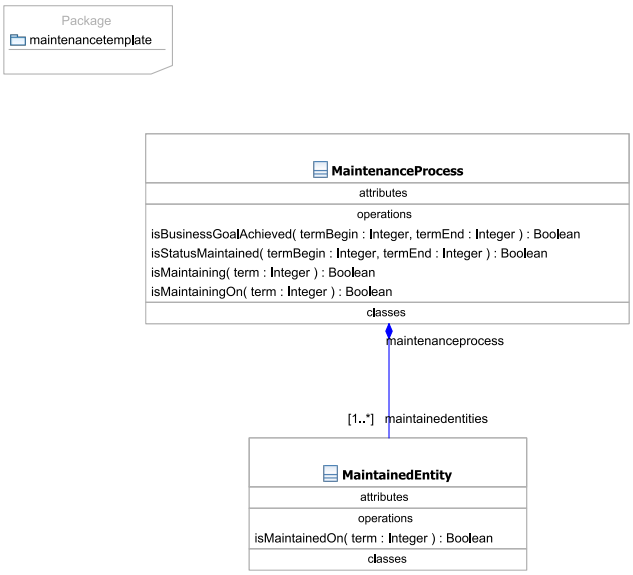
5.2.3 問題の存在

どのようにして望ましい状態を維持するのか？

5.2.4 解決方法

「ある状態を維持すること」を目的とする維持型ビジネスプロセスの UML クラス図とこれに対する制約を以下に示す.
このクラス図では，維持型ビジネスプロセスを MaintenanceProcess というクラスで表現し，維持する対象を `MaintainedEntity` というクラスで表現している．このクラス図では，複数の対象を維持する事を想定して，`MaintenanceProcess` と `MaintainedEntity` の関連を 1 対多の関連としている.
この記述の `isMaintaining( term:Integer )` という関数が「ある状態を維持すること」を表現している．この関数は `isMaintainingOn( term ) and isMaintaining( term + 1 )` と定義されているので，`isMaintaining( term:Integer )` が真となるのは，その時点で望ましい状態が成立しており（`MaintenanceProcess::isMaintainingOn( term )` が真）かつ以降も望ましい状態を維持している（`MaintenanceProcess::isMaintaining( term+1 )` が真）場合であることを再帰的に表現している.
さらに `MaintenanceProcess::isMaintainingOn( term:Integer )` が真となるのは，その時点で維持すべきすべての対象 `MaintainedEntity` である状態が維持されている（`MaintainedEntity::isMaintainedOn( term )` が真）場合であることを表現している.
つまり，この維持型ビジネスプロセスのビジネスゴールは `MaintenanceProcess::isMaintainingOn( term:Integer )` を真に保つことであり，この式がビジネス評価式となる．このビジネス評価式をパラメーターとすることによりビジネスプロセスのパターンを得ることができ，具体的なビジネス評価式を指定することで様々なビジネスプロセスの記述を得ることができる.
このパターンを利用する場合は，`MaintenanceProcess::isMaintainingOn( term:Integer )` の実体を定義することになる.

図 5.2: 宣言的記述による維持型ビジネスプロセスのパターン




```
package maintenancetemplate

-- ビジネスゴールを達成しているのは，ある状態を維持して場合である．
context MaintenanceProcess
def MaintenanceBusinessGoal:
isBusinessGoalAchieved( termBegin:Integer, termEnd:Integer ):Boolean =
isStatusMaintained( termBegin, termEnd )

-- ある状態を維持しているのは，開始時点である状態にあり，
-- それ以降もある状態を維持している場合である．
context MaintenanceProcess
def StatusIsMaitained:
isStatusMaintained( termBegin:Integer, termEnd:Integer ):Boolean =
isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )

-- ある時点以降プロセスが維持し続けているのは，　その時点でプロセスが維持しており，
-- かつ，以降もプロセスが維持しつつけている場合である．
context MaintenanceProcess
def ProcessIsMaintaining:
isMaintaining( termBegin:Integer, termEnd:Integer ):Boolean =
if termBegin > termEnd then
isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )
else
isMaintainingOn( termBegin )
endif

-- ある時点でプロセスが維持しているのは，
-- すべてのエンティティが維持されている場合である．
context MaintenanceProcess
def ProcessIsMaintainingOn: isMaintainingOn( term:Integer ):Boolean =
maintainedentities->forAll( me | me.isMaintainedOn( term ))

endpackage
```

図 5.2: （続き）

6 ビジネスプロセスパターンの利用

6.1 ビジネス評価式の指定

宣言的ビジネスプロセスのパターンはそのパラメーターであるビジネス評価式を定義することによって利用することができる．ビジネス評価式として，量に関するビジネス評価式を指定する例と時間に関するビジネス評価式を指定する例を示す.

6.1.1 量に関するビジネス評価式

ここでは，達成型ビジネスプロセスの宣言的記述と量に関するビジネス評価式を含む宣言的記述を合成することによって，量に関するビジネスゴールを達成するためのビジネスプロセスの宣言的記述を得られることを示す.

バランスを保ち続けることをビジネスゴールとするビジネスプロセスは，何らかのバランスを保った状態が維持すべき「望ましい状態」である維持型ビジネスプロセスととらえることができる．また，需要と供給のバランスを維持するビジネスプロセスは，バランスを維持するビジネスプロセスの一種である.

そこで図 5.2 のビジネス評価式である `MaintainedEntity::isMaintainedOn(term:Integer)` を段階的に詳細化していくことにより需要と供給のバランスを維持するビジネスプロセスの宣言的記述を得ることができる.

この結果得られたビジネスプロセスのクラス図を図 A.1 に示す．また，このクラス図に対する制約を OCL で記述したものを図 A.2 に示す.

6.1.2 時間に関するビジネス評価式

ここでは，供給計画を策定してから供給を行うまでに時間差（リードタイム）が存在する状況で，需要と供給のバランスを維持するビジネスプロセスについて考察する.

達成型ビジネスプロセスのパターンと時間に関するビジネス評価式を含む宣言的記述を合成することによって，時間に関するビジネスゴールを達成するためのビジネスプロセスを宣言的に記述できることを示す.

リードタイムを考慮するビジネスプロセスは，ある時点のリードタイム前になんらかの状態に達していることが維持すべき「望ましい状態」である維持型ビジネスプロセスと捉えることができる．例えば，継続的に計画を立案するビジネスプロセスはリードタイムを考慮して計画を立案する必要があるので，リードタイムを考慮するビジネスプロセスの一種である.

そこで図 5.2 のビジネス評価式である `MaintainedEntity::isMaintainedOn(term:Integer)` を段階的に詳細化していくことにより継続的に有効な計画を立案するビジネスプロセスの宣言的記述を得ることができる.

この結果得られたビジネスプロセスのクラス図を図 A.3 に示す．また，このクラス図に対する制約を OCL で記述したものを図 A.4 に示す.

6.2 宣言的ビジネスプロセス記述の合成

ここでは，同じビジネスプロセスパターンから導出したビジネスプロセスの宣言的記述を合成することで，複数のビジネス評価式を真にすることをビジネスゴールとするビジネスプロセスの宣言的記述を導出できることを示す.

6.2.1 需要と供給のバランスを維持する計画を立案し実行するビジネスプロセスの宣言的記述

需要と供給のバランスを維持するための供給計画を立案するビジネスプロセスは，実行可能な計画を継続的に立案するビジネスプロセスにおいて立案する計画が需要と供給のバランスを保つものである．それゆえ，このビジネスプロセスは，需要と供給のバランスを維持するビジネスプロセスのビジネスゴー

ルと実行可能な計画を継続的に立案するビジネスプロセスのビジネスゴールの両方を達成することを目的とするビジネスプロセスである．

このビジネスプロセスにおいては計画する対象が需要であると考えられるので，需要と供給のバランスを維持するビジネスプロセスにおける需要と，実行可能な計画を継続的に立案するビジネスプロセスにおける計画は同一の対象である．このため，この２つのビジネスプロセスの合成において，Supply と Plan を合成して PlannedSupply とする．この合成において Supply::suppliedOn は Plan::executedOn から導かれるので，この関係を制約として追加する．

このビジネスプロセスは，立案した計画によって需要と供給のバランスを保つものである．このため，MaintainedEntity::isPreparedOn(term:Integer) に MaintainedEntity::isBalanced(term:Integer) を加える．

これらの結果として得られるビジネスプロセスのクラス図を図 A.5 に，制約を図 A.6 に示す．

6.3 例

第 2 章で説明した例題に対して，ここまで説明したビジネスプロセスの宣言的記述を適用できることを示す．ただし，簡単のために部材の種類は一種類とする．

このビジネスプロセスは，顧客からの確定注文と生産指示，生産指示と部材発注，のそれぞれのバランスを維持するように生産指示，部材発注を行うビジネスプロセスである．

そこで，上述の「需要と供給のバランスを維持する計画を立てるビジネスプロセス」（図 A.5 および図 A.6）を用いて，このビジネスプロセスを形式的に記述する．

上述の２つの組み合わせにとって，実際のビジネスプロセスと「需要と供給のバランスを維持する計画を立てて実行するビジネスプロセス」の間には，表 6.1 の関係があると考えることができる．そこで，この関係を用いて「需要と供給のバランスを維持する計画を立てて実行するビジネスプロセス」に対して置換を行い，生産指示ビジネスプロセスと部材発注ビジネスプロセスの宣言的記述を得る．

表 6.1: パターン内の要素の対応

パターン中の要素 ＼ 計画対象	生産指示	部材手配
MaintenanceProcess	製品生産計画プロセス	製品生産計画プロセス
MaintainedEntity	製品	部材
Demand	確定注文	生産
PlannedSupply	生産指示	部材発注

例題のプロセスでは，生産指示と部材発注の両方を行う必要がある．そこで，生産指示ビジネスプロセスと部材発注ビジネスプロセスの宣言的記述を合成することで，例題のビジネスプロセスに対する宣言的記述をえる．

この合成の過程を以下に示す．

1. 生産指示のビジネスゴールは次の示す定義を含んでいる．

```
context ProductionPlanningProcess
def ProcessIsMaintainingOn:
   isMaintainingOn( term:Integer ):Boolean =
   product.isMaintainedOn( term )
```

2. 部材発注のビジネスゴールは次の示す定義を含んでいる．

```
context ProductionPlanningProcess
def ProcessIsMaintainingOn:
   isMaintainingOn( term:Integer ):Boolean =
   material.isMaintainedOn( term )
```

3. これらを合成すると，isMaintainingOn は両方に存在するが内容が異なるものである．このため，両方の内容を満たすように連言を用いて合成する．

```
context ProductionPlanningProcess
def ProcessIsMaintainingOn:
   isMaintainingOn( term:Integer ):Boolean =
   product.isMaintainedOn( term )
   and
   material.isMaintainedOn( term )
```

4. ManufacturingOrder は生産指示ビジネスプロセスにとっては供給であるので，executedOn, suppliedOn と quantity という属性を持つ．また，部材発注ビジネスプロセスにとっ

ては需要であるであるので，expectedOn と quantity という属性を持つ．これらを統合して，ManufacturingOrder には executedOn, suppliedOn, expectedOn, quantity とう属性を定義する．

5. ManufacturingOrder では suppliedOn と expectedOn は等しいので以下の制約を追加する．

```
context Production::suppliedOn
derive: self.expectedOn
```

この合成の結果得られるビジネスプロセスの宣言的記述のクラス図を図 A.7，制約を図 A.8 に示す．

7 関連研究

Lamsweerde らが提唱したシステム要求獲得のための手法の一つである KAOS では，ゴールを「システムに期待される特性（Property）」と位置付け，ゴールを達成，中止，維持，回避，最適化の 5 種類のパターンに分類し，それぞれのパターンを線形時相論理を用いて定義している [2][3]．これらの分類のうち，達成と中止，維持と回避は，それぞれ述語が肯定であるか否定であるか異なるだけで，本質的に同じものだと考えられる．KAOS における達成型と中止型が本論文の達成型に相当し，維持型と回避型が本論文の維持型に相当することになる．Lamsweerde らが線形時相論理を用いてゴールを表現したのに対して，本論文では再帰的に定義した制約を用いてビジネスゴールを表現できることを示した．

Castro らはステークホルダーの情報をゴールとして用い，i*を拡張してテキスト形式の形式的な言語を定義した [1]．この研究でも，ゴールのモードとして Achieve 型と Maintain 型を扱っている．Koen らは [9] で Achievement goal と Maintenance goal を分類し，Achievement goal に対してセマンティクスを与えている．Dastani らも [4] で Perform Goal, Achieve Goal，Maintain Goal に対して形式的なセマンティクスを与えている．

Endl らはビジネスルールを段階的に詳細化することによってビジネスプロセスを定義する方法について述べている [5]．これに対して本研究では，ビジネスプロセスのパターンに宣言的に記述したビジネス評価式を追加することによって，そのビジネス評価式を真にすることをビジネスプロセスを宣言的に記述できることを示した．

Kristoffersen らは Timed LTL で記述されたビジネスルールを Disjunctive Normalized Equation System に展開し，証明する方法について述べている [11]．これは直感的には，ビジネスルールを現在の項と将来の項の選言に展開して，証明することになる．この方法は，本論文で述べた「時刻 T 以降にある状態を維持する」ことを形式的に記述するために,「時刻 T にある状態にあること」と,「時刻 T+1 以降にある状態を維持していること」の連言を用いたのと類似のアプローチである．

Kueng らは，ゴールに基づいてビジネスプロセスモデルを作成し，評価する手法について述べている [12]．彼らが扱っているのは非機能ゴールであり，非機能ゴールが与えられたときにアクティビティで表現されたビジネスプロセスをどのように変換すべきかについて述べている．

Jacobs らは情報システムを開発する際の意思決定をゴールに基づいて行うことの必要性を述べ，そのためにゴールモデルとビジネスモデルを統合することを提唱している [10]．ここで扱っているビジネスモデルはアクティビティを中心としたもので，この点では従来からのビジネスプロセスモデリング手法を強化したものといえる．

これらの研究に対して，本研究ではビジネスプロセスをカテゴリーに分類した上でパターン化し，宣言的に記述したビジネス評価式を組み合わせることでビジネスプロセスを宣言的に記述できることを示した．本研究のアプローチは，他の研究とは異なり，UML や OCL といった一般的な技術を用いてビジネスプロセスを記述することにより，企業における関係者間でビジネスプロセスに関する知識の共有と継承を図ろうとするものである．

8 結論

本論文では，パターンとビジネス評価式の組み合わせでビジネスプロセスを宣言的に記述できることを示した．ソフトウェア開発におけるパターンの利用と同様に，ビジネスプロセスを記述するためにパターンを利用することにより関係者間でのビジネスプロセスに関する知識の共有と継承が可能となる．

今までは困難であった関係者間でのビジネスプロセスに関する知識の共有・継承が可能になることにより，市場の要求や組織構造といったビジネス環境の変化による既存のビジネスプロセスの変更や新たなビジネスプロセスの構築を迅速に行えるようになると期待できる．

また，本手法では，ビジネスプロセスパターンとビジネス評価式を組み合わせてビジネスプロセスを宣言的に表現しているため，パターンの再利用が容易であり，ビジネス評価式を変更することにより新たなビジネスプロセスの構築を迅速に行えるようになると期待できる．さらに，複数のビジネスプロセス記述を合成することも可能であるので，複雑なビジネスゴールを持つビジネスプロセスの構築も可能になる．

本論文では，ビジネスプロセスを達成型ビジネスプロセスと維持型ビジネスプロセスの２つのカテゴリーに分類した上で，維持型ビジネスプロセスについてパターンとビジネス評価式として宣言的に記述できることを示した．今後の課題として，達成型ビジネスプロセスについても宣言的に記述できることを示す必要がある．また，ビジネスプロセスを実行可能なものとして実装するためには，宣言的なビジネスプロセスの記述から手続き的な記述を導き出す必要がある．この手法についても今後の課題である．

参考文献

[1] Jaelson Castro, Manuel Kolp, and John Mylopoulos C. Towards requirements-driven information systems engineering: The tropos project. *Information Systems*, Vol. 27, pp. 365–389, 2002.

[2] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. In *Science of Computer Programming*, pp. 3–50, 1993.

[3] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde. Grail/kaos: an environment for goal-driven requirements engineering. In *ICSE '97: Proceedings of the 19th international conference on Software engineering*, pp. 612–613, New York, NY, USA, 1997. ACM.

[4] Mehdi Dastani, M. Birna van Riemsdijk, and John-Jules Ch. Meyer. Goal types in agent programming. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 1285–1287, New York, NY, USA, 2006. ACM.

[5] Rainer Endl, Gerhard Knolmayer, and Marcel Pfahrer. Modeling processes and workflows by business rules, 1998.

[6] Object Management Group. Business Process Modeling Notation Specification, October 2003.

[7] Object Management Group. UML 2.0 OCL Specification, February 2006.

[8] Object Management Group. OMG Unified Modeling Language (OMG UML), Superstructure Version 2.2, Februrary 2009.

[9] Koen V. Hindriks and M. Birna Van Riemsdijk. Satisfying maintenance goals. In Proc. of DALT '07. Springer, 2007.

[10] Stephan Jacobs and Roland Holten. Goal driven business modelling: supporting decision

making within information systems development. In *COCS '95: Proceedings of conference on Organizational computing systems*, pp. 96–105, New York, NY, USA, 1995. ACM Press.

[11] Kare J. Kristoffersen, Christian Pedersen, and Henrik R. Andersen. Checking temporal business rules. In *In: Proceedings of the First International REA Workshop*, p. 2004, 2004.

[12] Peter Kueng, Peter Kueng, Peter Kawalek, and Peter Kawalek. Goal-based business process models: Creation and evaluation. *Business Process Management Journal*, Vol. 3, pp. 17–38, 1997.

[13] Martin Owen and Jog Raj. BPMN and business process management - introduction to the new business process modeling standard, 2003.

[14] ヨシュ・ヴァルメル, アーネク・クレッペ. UML/MDA のためのオブジェクト制約言語 OCL 第 2 版. エスアイビーアクセス, Dec 2004. 翻訳　竹村　司.

[15] 加藤正人 06. BPMN によるビジネスプロセスモデリング入門. SRC, Jun 2006.

A　付録

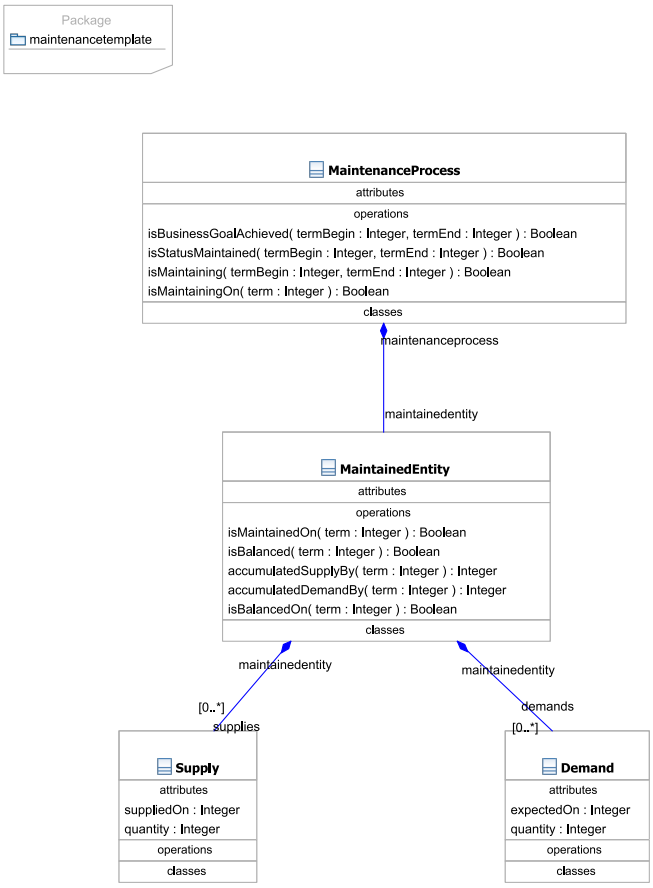


図 A.1: 需要と供給のバランスを維持するビジネスプロセスのクラス図

```
package maintenancetemplate
-- ビジネスゴールを達成しているのは，ある状態を維持して場合である．
context MaintenanceProcess
def MaintenanceBusinessGoal:
isBusinessGoalAchieved( termBegin:Integer, termEnd:Integer ):Boolean =
isStatusMaintained( termBegin, termEnd )

-- ある状態を維持しているのは，開始時点である状態にあり，
-- それ以降もある状態を維持している場合である．
context MaintenanceProcess
def StatusIsMaitained:
isStatusMaintained( termBegin:Integer, termEnd:Integer ):Boolean =
isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )

-- ある時点以降プロセスが維持し続けているのは，
-- その時点でプロセスが維持しており，
-- かつ，以降もプロセスが維持しつつけている場合である．
context MaintenanceProcess
def ProcessIsMaintaining:
isMaintaining( termBegin:Integer, termEnd:Integer ):Boolean =
if termBegin > termEnd then
isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )
else
isMaintainingOn( termBegin )
endif

-- ある時点でプロセスが維持しているのは，
-- すべてのエンティティが維持されている場合である．
context MaintenanceProcess
def ProcessIsMaintainingOn: isMaintainingOn( term:Integer ):Boolean =
maintainedentities->forAll( me | me.isMaintainedOn( term ))

-- ある時点でエンティティが維持されているというのは，
-- その時点でエンティティのバランスが維持されている場合である．
context MaintainedEntity
def EntityIsBalancedOn: isMaintainedOn( term:Integer ):Boolean =
isBalanced( term )

-- ある時点で エンティティのバランスが維持されているというのは，
-- その時点でエンティティのバランスが取れており，
-- それ以前までのバランスが維持されていら場合である．．
context MaintainedEntity
def EntityIsBalanced: isBalanced( term:Integer ):Boolean =
isBalancedOn( term )
and
if( term > 0 ) then isBalanced( term - 1 )
else true
endif

-- ある時点で エンティティのバランスが取れているというのは，
-- 需要の総計よりも供給の総計が多い場合である．
context MaintainedEntity
def EntityIsBalancedOn: isBalancedOn( term:Integer ):Boolean =
accumulatedDemandBy( term ) <= accumulatedSupplyBy( term )

-- ある時点の需要の総計は，需要の数量の合計である．
context MaintainedEntity
def AccumulatedDemand: accumulatedDemandBy( term:Integer ):Integer =
demands->select( d | d.expectedOn <= term
and not d.quantity.oclIsUndefined()).quantity->sum()

-- ある時点の供給の総計は，供給の数量の合計である．
context MaintainedEntity
def AccumulatedSupply: accumulatedSupplyBy( term:Integer ):Integer =
supplies->select( s | s.suppliedOn <= term
and not s.quantity.oclIsUndefined()).quantity->sum()

endpackage
```

図 A.2: 需要と供給のバランスを維持するビジネスプロセスに対する制約

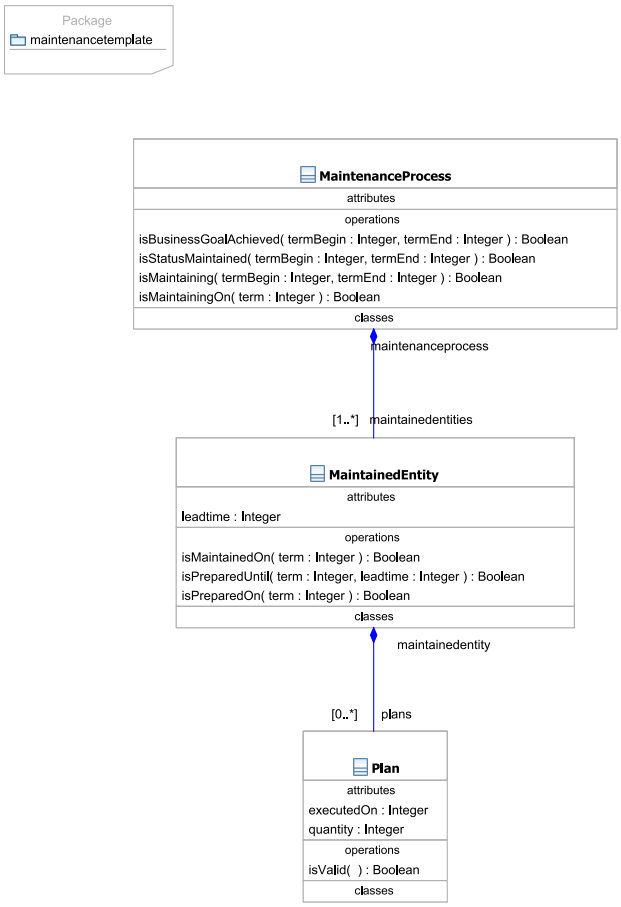


図 A.3: 継続的に有効な計画を立案するビジネスプロセスのクラス図

```
package maintenancetemplate
-- ビジネスゴールを達成しているのは，ある状態を維持して場合である．
context MaintenanceProcess
def MaintenanceBusinessGoal:
isBusinessGoalAchieved( termBegin:Integer, termEnd:Integer ):Boolean =
isStatusMaintained( termBegin, termEnd )

-- ある状態を維持しているのは，開始時点である状態にあり，
-- それ以降もある状態を維持している場合である．
context MaintenanceProcess
def StatusIsMaitained:
isStatusMaintained( termBegin:Integer, termEnd:Integer ):Boolean =
isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )

-- ある時点以降プロセスが維持し続けているのは，
-- その時点でプロセスが維持しており，
-- かつ，以降もプロセスが維持しつつけている場合である．
context MaintenanceProcess
def ProcessIsMaintaining:
isMaintaining( termBegin:Integer, termEnd:Integer ):Boolean =
if termBegin > termEnd then
isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )
else
isMaintainingOn( termBegin )
endif

-- ある時点でプロセスが維持しているのは，
-- すべてのエンティティが維持されている場合である．
context MaintenanceProcess
def ProcessIsMaintainingOn: isMaintainingOn( term : Integer ) : Boolean =
maintainedentities->forAll( me | me.isMaintainedOn( term ))

-- ある時点で維持されているというのは，リードタイム後まで準備が完了している場合である．
context MaintainedEntity
def EntityIsMaintainedOn: isMaintainedOn( term:Integer ):Boolean =
isPreparedUntil( term, leadtime )

-- ある時点からリードタイム後まで準備できているというのは，ある時点で準備ができており，
-- かつ，次の時期からリードタイムー１後まで準備ができている場合である．
context MaintainedEntity
def EntityIsPreparedUntil: isPreparedUntil( term:Integer, lt:Integer ):Boolean =
isPreparedOn( term )
and
if leadtime > 0 then isPreparedUntil( term + 1, lt - 1 )
else true
endif

-- ある時点で準備ができているというのは，
-- その時点に実行可能な有効な計画が存在する場合である．
context MaintainedEntity
def EntityIsPreparedOn: isPreparedOn( term:Integer ):Boolean =
plans->one( executedOn = term and isValid() )

-- 計画が有効なのは，計画の実行期間が指定された期間からリードタイムの間にあり，
-- 数量が未定義でない場合である．
context Plan
def PlanIsValid: isValid():Boolean =
not quantity.oclIsUndefined()

endpackage
```

図 A.4: 継続的に有効な計画を立案するビジネスプロセスに対する制約

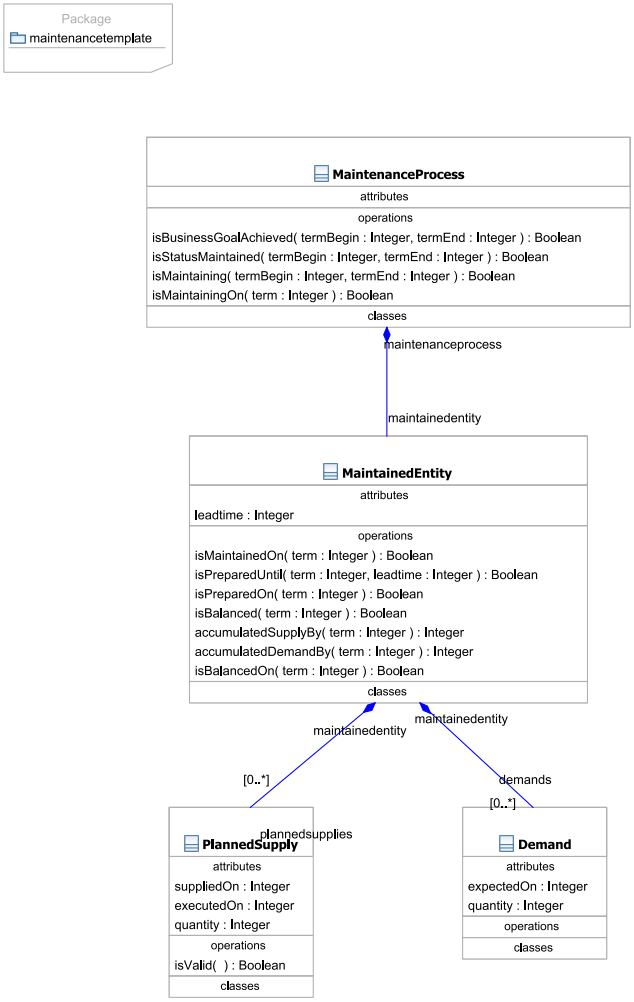


図 A.5: 需要と供給のバランスを維持する計画を立てるビジネスプロセスのクラス図

```
package maintenancetemplate
-- ビジネスゴールを達成しているのは，ある状態を維持して場合である．
context MaintenanceProcess
def MaintenanceBusinessGoal:
   isBusinessGoalAchieved( termBegin:Integer, termEnd:Integer ):Boolean =
   isStatusMaintained( termBegin, termEnd )

-- ある状態を維持しているのは，開始時点である状態にあり，
-- それ以降もある状態を維持している場合である．
context MaintenanceProcess
def StatusIsMaitained:
   isStatusMaintained( termBegin:Integer, termEnd:Integer ):Boolean =
   isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )

-- ある時点以降プロセスが維持し続けているのは，
-- その時点でプロセスが維持しており，
-- かつ，以降もプロセスが維持しつつけている場合である．
context MaintenanceProcess
def ProcessIsMaintaining:
   isMaintaining( termBegin:Integer, termEnd:Integer ):Boolean =
   if termBegin < termEnd then
      isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )
   else
      isMaintainingOn( termBegin )
   endif

-- ある時点でプロセスが維持しているのは，
-- エンティティが維持されている場合である．
context MaintenanceProcess
def ProcessIsMaintainingOn: isMaintainingOn( term:Integer ):Boolean =
   maintainedentity.isMaintainedOn( term )

-- ある時点で維持されているというのは，リードタイム後まで準備が完了している場合である．
context MaintainedEntity
def EntityIsMaintainedOn: isMaintainedOn( term:Integer):Boolean =
   isPreparedUntil( term, leadtime )

-- ある時点からリードタイム後まで準備できているというのは，ある時点で準備ができており，
-- かつ，次の時期からリードタイムー１後まで準備ができている場合である．
context MaintainedEntity
def EntityIsPreparedUntil: isPreparedUntil( term:Integer, lt:Integer ):Boolean =
   isPreparedOn( term )
   and
   if lt > 0 then isPreparedUntil( term + 1, lt - 1 )
   else true
   endif

-- ある時点で準備ができているというのは，
-- その時点でエンティティのバランスが維持されて
-- その時点に実行可能な有効な計画が存在する場合である．
context MaintainedEntity
def EntityIsPreparedOn: isPreparedOn( term:Integer ):Boolean =
   isBalanced( term )
   and
   plannedsupplies->one( executedOn = term and isValid() )

-- ある時点でエンティティのバランスが維持されているというのは，
-- その時点でエンティティのバランスが取れており，
-- それ以前までのバランスが維持されていら場合である．
context MaintainedEntity
def EntityIsBalanced: isBalanced( term:Integer ):Boolean =
   isBalancedOn( term )
   and
   if( term > 1 ) then isBalanced( term - 1 )
   else true
   endif
```

図 A.6: 需要と供給のバランスを維持する計画を立案するビジネスプロセスに対する制約

```
-- ある時点でエンティティのバランスが取れているというのは,
-- 需要の総計よりも供給の総計が多い場合である.
context MaintainedEntity
def EntityIsBalancedOn: isBalancedOn( term:Integer ):Boolean =
   accumulatedDemandBy( term ) <= accumulatedSupplyBy( term )

-- ある時点の需要の総計は, 需要の数量の合計である.
context MaintainedEntity
def AccumulatedDemand: accumulatedDemandBy( term:Integer ):Integer =
   demands->select( d | d.expectedOn <= term
   and not d.quantity.oclIsUndefined()).quantity->sum()

-- ある時点の供給の総計は, 供給の数量の合計である.
context MaintainedEntity
def AccumulatedSupply: accumulatedSupplyBy( term:Integer ):Integer =
   plannedsupplies->select( s | s.suppliedOn <= term
   and not s.quantity.oclIsUndefined()).quantity->sum()

-- 計画が有効なのは, 計画の実行期間が指定された期間からリードタイムの間にあり,
-- 数量が未定義でない場合である.
context PlannedSupply
def PlanIsValid: isValid():Boolean =
   not quantity.oclIsUndefined()

-- 対応する計画と供給は, 期間が等しい.
context PlannedSupply::suppliedOn
derive: self.executedOn

endpackage
```

図 A.6: （続き）

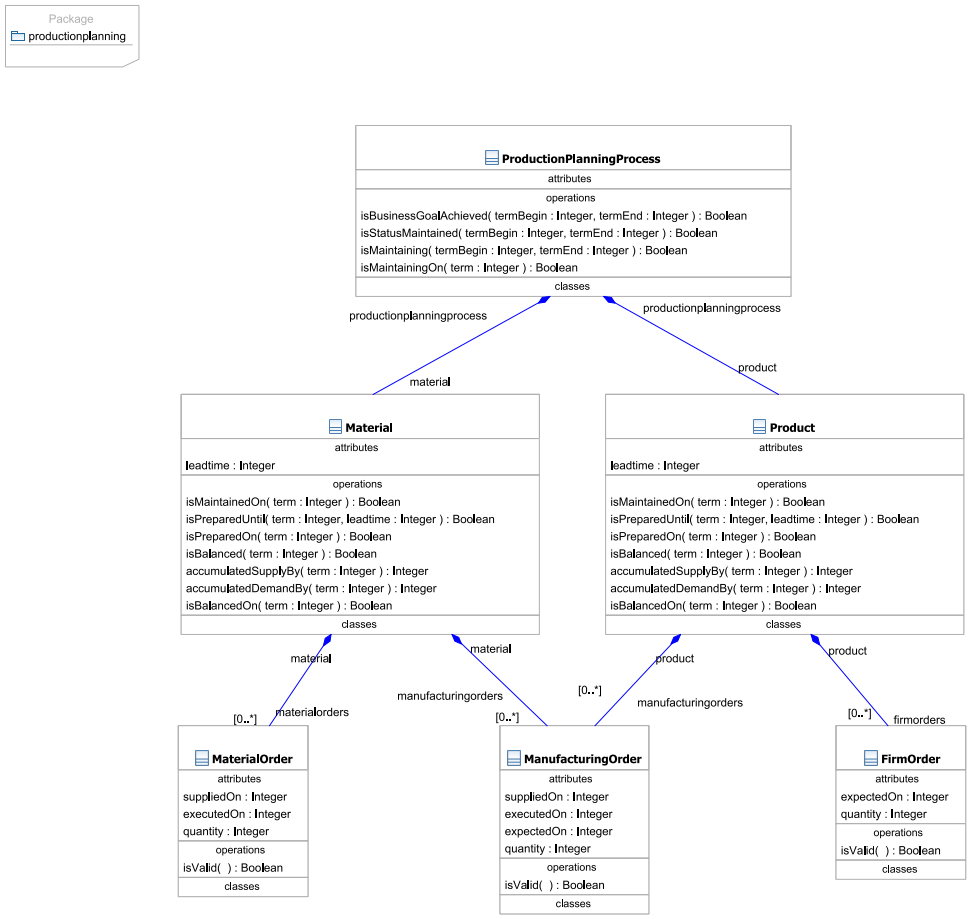


図 A.7: 生産管理ビジネスプロセスのクラス図

```
package productionplanning

-- ビジネスゴールを達成しているのは，ある状態を維持して場合である．
context ProductionPlanningProcess
def MaintenanceBusinessGoal:
   isBusinessGoalAchieved( termBegin:Integer, termEnd:Integer ):Boolean =
   isStatusMaintained( termBegin, termEnd )

-- ある状態を維持しているのは，開始時点である状態にあり，
-- それ以降もある状態を維持している場合である．
context ProductionPlanningProcess
def StatusIsMaitained:
   isStatusMaintained( termBegin:Integer, termEnd:Integer ):Boolean =
   isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )

-- ある時点以降プロセスが維持し続けているのは，
-- その時点でプロセスが維持しており，
-- かつ，以降もプロセスが維持しつつけている場合である．
context ProductionPlanningProcess
def ProcessIsMaintaining:
   isMaintaining( termBegin:Integer, termEnd:Integer ):Boolean =
   if termBegin < termEnd then
      isMaintainingOn( termBegin ) and isMaintaining( termBegin + 1, termEnd )
   else
      isMaintainingOn( termBegin )
   endif

-- ある時点でプロセスが維持しているのは，
-- エンティティが維持されている場合である．
context ProductionPlanningProcess
def ProcessIsMaintainingOn: isMaintainingOn( term:Integer ):Boolean =
   product.isMaintainedOn( term )
   and
   material.isMaintainedOn( term )

-- ある時点で維持されているというのは，リードタイム後まで準備が完了していて
-- その時点でエンティティのバランスが維持されている場合である．
context Product
def EntityIsMaintainedOn: isMaintainedOn( term : Integer) : Boolean =
   isPreparedUntil( term, leadtime )

--  ある時点からリードタイム後まで準備できているというのは，ある時点で準備ができており，
-- かつ，次の時期からリードタイムー１後まで準備ができている場合である．
context Product
def EntityIsPreparedUntil: isPreparedUntil( term : Integer, lt : Integer ) : Boolean =
   isPreparedOn( term )
   and
   if lt > 0 then isPreparedUntil( term + 1, lt - 1 )
   else true
   endif

-- ある時点で準備ができているというのは，
-- その時点でエンティティのバランスが維持されている場合である．
context Product
def EntityIsPreparedOn: isPreparedOn( term : Integer ) : Boolean =
   isBalanced( term )

-- ある時点でエンティティのバランスが維持されているというのは，
-- その時点でエンティティのバランスが取れており，
-- それ以前までのバランスが維持されている場合である．
context Product
def EntityIsBalanced: isBalanced( term : Integer ) : Boolean =
   isBalancedOn( term )
   and
   if( term > 1 ) then isBalanced( term - 1 )
   else true
   endif
```

図 A.8: 生産管理ビジネスプロセスに対する制約

```
-- ある時点でエンティティのバランスが取れているというのは,
-- その時点の需要と供給が決定しており
-- 需要の総計よりも供給の総計が多い場合である.
context Product
def EntityIsBalancedOn: isBalancedOn( term : Integer ) : Boolean =
   manufacturingorders->one( executedOn = term and isValid() )
   and
   firmorders->one( expectedOn = term and isValid() )
   and
   accumulatedDemandBy( term ) <= accumulatedSupplyBy( term )

-- ある時点の需要の総計は, 需要の数量の合計である.
context Product
def AccumulatedDemand: accumulatedDemandBy( term : Integer ) : Integer =
   firmorders->select( d | d.expectedOn <= term
   and not d.quantity.oclIsUndefined()).quantity->sum()

-- ある時点の供給の総計は, 供給の数量の合計である.
context Product
def AccumulatedSupply: accumulatedSupplyBy( term : Integer ) : Integer =
   manufacturingorders->select( s | s.suppliedOn <= term
   and not s.quantity.oclIsUndefined()).quantity->sum()

-- ある時点で維持されているというのは, リードタイム後まで準備が完了していて
-- その時点でエンティティのバランスが維持されている場合である.
context Material
def EntityIsMaintainedOn: isMaintainedOn( term : Integer) : Boolean =
   isPreparedUntil( term, leadtime )

--  ある時点からリードタイム後まで準備できているというのは, ある時点で準備ができており,
-- かつ, 次の時期からリードタイムー1後まで準備ができている場合である.
context Material
def EntityIsPreparedUntil: isPreparedUntil( term : Integer, lt : Integer ) : Boolean =
   isPreparedOn( term )
   and
   if lt > 0 then isPreparedUntil( term + 1, lt - 1 )
   else true
   endif

-- ある時点で準備ができているというのは,
-- その時点でエンティティのバランスが維持されている場合である.
context Material
def EntityIsPreparedOn: isPreparedOn( term : Integer ) : Boolean =
   isBalanced( term )

-- ある時点でエンティティのバランスが維持されているというのは,
-- その時点でエンティティのバランスが取れており,
-- それ以前までのバランスが維持されている場合である.
context Material
def EntityIsBalanced: isBalanced( term : Integer ) : Boolean =
   isBalancedOn( term )
   and
   if( term > 0 ) then isBalanced( term - 1 )
   else true
   endif

-- ある時点でエンティティのバランスが取れているというのは,
-- その時点の需要と供給が決定しており
-- 需要の総計よりも供給の総計が多い場合である.
context Material
def EntityIsBalancedOn: isBalancedOn( term : Integer ) : Boolean =
   materialorders->one( executedOn = term and isValid() )
   and
   manufacturingorders->one( expectedOn = term and isValid() )
   and
   accumulatedDemandBy( term ) <= accumulatedSupplyBy( term )
```

図 A.8: （続き）

```
-- ある時点の需要の総計は，需要の数量の合計である．
context Material
def AccumulatedDemand: accumulatedDemandBy( term : Integer ) : Integer =
   manufacturingorders->select( d | d.expectedOn <= term
   and not d.quantity.oclIsUndefined()).quantity->sum()

-- ある時点の供給の総計は，供給の数量の合計である．
context Material
def AccumulatedSupply: accumulatedSupplyBy( term : Integer ) : Integer =
   materialorders->select( s | s.suppliedOn <= term
   and not s.quantity.oclIsUndefined()).quantity->sum()

-- 計画が有効なのは，計画の実行期間が指定された期間からリードタイムの間にあり，
-- 数量が未定義でない場合である．
context ManufacturingOrder
def PlanIsValid: isValid() : Boolean =
   not quantity.oclIsUndefined()

-- 計画が有効なのは，計画の実行期間が指定された期間からリードタイムの間にあり，
-- 数量が未定義でない場合である．
context MaterialOrder
def PlanIsValid: isValid() : Boolean =
   not quantity.oclIsUndefined()

-- 製品の供給期間と計画実行期間は等しい．
--context Production::suppliedOn
--derive: self.manufacturingorder.executedOn
context ManufacturingOrder
inv: suppliedOn = self.executedOn

--context Production::suppliedOn
--derive: self.expectedOn
context ManufacturingOrder
inv: suppliedOn = self.expectedOn

-- 需要が有効なのは，数量が未定義でない場合である．
context FirmOrder
def OrderIsValid: isValid() : Boolean =
   not quantity.oclIsUndefined()

-- 計画が有効なのは，計画の実行期間が指定された期間からリードタイムの間にあり，
-- 数量が未定義でない場合である．
context MaterialOrder
def PlanIsValid: isValid() : Boolean =
   not quantity.oclIsUndefined()

-- 製品の供給期間と計画実行期間は等しい．
context MaterialOrder::suppliedOn
derive: self.executedOn

-- 需要が有効なのは，数量が未定義でない場合である．
context ManufacturingOrder
def OrderIsValid: isValid() : Boolean =
   not quantity.oclIsUndefined()

endpackage
```

図 A.8: （続き）

テストカバレッジに基づくテストコードの再構成パターン

坂本 一憲 * **和田 卓人** † **鷲崎 弘宜** ‡ **深澤 良彰** §
Kazunori Sakamoto Takuto Wada Hironori Washizaki Yoshiaki Fukazawa

1. Name

テストカバレッジに基づくテストコードの再構成パターン

2. Context

近年，テストコードを記述するための開発手法が進歩しており，製品コードに対してテストコードが十分に記述されるようになったが，その一方でテストコードの重複が起こりやすくなっている．重複したテストコードの量が増大することによって，以下で説明するテストの効率性の低下とテストコードの保守性の低下の 2 問題が引き起こされる．

- **テストの効率性の低下**
  テストコードの量があまりに増大しすぎると，テストの実行効率性が低下する．この問題は，テストコードの量の増大に比例して，単純にテストの実施項目数が増大することで，テストの実施時間の増大が引き起こされるためである．テストの効率性の低下は，手動によるテストでも，テストフレームワークによる自動テストにおいても問題となり，単体テストや結合テストの実施が困難になってしまう．
  例えば，開発者が単体テストの実施を渋るようになったり，Hudson のような継続的インテグレーションツールを利用して，テストを自動的に実施する場合でも，バージョン管理システムへのコミット，コンパイル，テストの実施のサイクルに遅れが生じ，正常に実施できなくなる場合がある．特に，テスト駆動開発 [1] やリファクタリングを行う際は，コーディングとともに頻繁にテストを実施する必要があるため，テストの実施時間の増大がコーディング作業の能率低下に直結してしまう．また，テストコードの増大によってテストの実施が困難になると，ソフトウェアの品質向上のためにテストコードを追加することが，より一層テストの実施を困難にさせてしまう．そのため，かえって試験性の低下を招いてしまい，ソフトウェア全体の品質問題を引き起こす場合がある．
- **テストコードの保守性の低下**
  テストコードの量があまりに増大しすぎると，テストコードの保守性が低下する．特に，テストコードもソースコードの一部であるため，ソフトウェア全体の保守性の低下につながる．この問題は，テストコードの量の増大によって，人間がソースコードを読んで理解する作業に要する時間が増加したり，ソースコードを変更した際に影響が波及する範囲を推定することが困難になるなど，品質副特性における解析性や変更性の低下，また前節による試験性の低下から，品質特性における保守性の低下が引き起こされるためである．特に，テストコードの重複は DRY 原則に反するため，保守性の低下が顕著である．テストコードの保守性の低下は，テストコードの管理や修正を困難にする．
  例えば，テストが仕様を十分に検証しているかどうかのチェックや，新しいテストコードを追加したり修正することが困難になるため，テストの品質低下を招いてしまい，ソフトウェア全体の品質問題を引き起こす．

3. Problem

ソフトウェア開発においてソフトウェアテスティング，例えば単体テストや結合テストなどにおける重複したテストコードの量が増大しすぎることがある．特に，テスト駆動開発といったアジャイルな手法を用いることでその傾向が強くなる．

テスト駆動開発とは，失敗するテストコードの記述，テストをパスするような最低限の実装，再構成というサイクルを小さく早く繰り返すことによって，ソフトウェアを実装する開発手法である．テスト駆動開発におけるテストコードは，極めて粒度の小さい簡単なものから記述するため，一般的なテストコードよりもテストコードの量が増大しがちである．また，テスト駆動開発におけるテストコードは，ソフトウェアの品質保証のためではなく，ソフトウェアの実装のために記述されるため，別途，品質保証のためのテストコードが必要となる．なお，テスト駆動開発によって記述されたテストコードを，品質保証のためのテストコードに再利用することは可能だが，品質保証のためのテストコードと重複しやすい．このようにして記述されたテストコードの量は一般のプロジェクトと比較して多くなる傾向があり，規模の小さいプロジェクトであっても，一回の単体テストの実施に 5 分以上かかる場合がある．

また，近年のテストフレームワークでは，単体テストのみならず結合テストも実施可能であり，両方のフェーズのテストコードが入り混じる傾向がある．結合テストでは単

* 早稲田大学 大学院 基幹理工学研究科
kazuu@ruri.waseda.jp
† タワーズ・クエスト 株式会社
takuto.wada@towersquest.jp
‡ 早稲田大学 大学院 基幹理工学研究科
washizaki@waseda.jp
§ 早稲田大学 大学院 基幹理工学研究科
fukazawa@waseda.jp

体テストが対象とするコードを複数組み合わせたテストを実施するため，結合テストのテストコードは単体テストのテストコードを包含するケースが多い．そのため，テストコードの重複が発生しやすい．

4. Forces

テストの実行効率の改善，テストコードの保守性の改善，最小限の改善コスト，機械的な判断の 4 つのフォースが挙げられる．

- **テストの実行効率の改善**
  テストの実行効率を改善し，なるべく短い時間でテストを実施する必要がある．
- **テストコードの保守性の改善**
  テストコードを保守するために，可読性や変更容易性の高いテストコードを記述する必要がある．
- **最小限の改善コスト**
  上記二つの改善を行うために，なるべく金銭や時間などのコストを抑える必要がある．
- **機械的な判断**
  上記二つの改善を行うために，極力機械的な判断を元に安全に改善する必要がある．

例えば，テストの実行効率を改善するために，テストフレームワークの改良や分散技術の利用，物理的な PC パフォーマンスの改善などが考えられる．また，テストコードの保守性を改善するために，人手によるテストコードの再構成などが考えられる．この場合，どれも金銭や時間などのコストがかかる上，それぞれが両方の問題を解決することはできない．他に，テストの実施範囲を狭める方法も考えられる．この場合，テストの実行効率のみを改善することができるが，ソースコードのリファクタリングなどの修正を行った際に，どこに変更の影響が生じているか人手で判断してテストの実施範囲を狭めるため，開発者の意思が介入することによってテストが正常に実施されない可能性がある．

したがって，これらの問題を解決するために，不要なテストコードの削除，特に，テストカバレッジに基づいて削除しても安全であると機械的に判断されたテストコードを削除することによって，上述の 4 つのフォースを満たす解決策を利用する．

5. Solution

テストコード網羅している製品コードの部分をテストカバレッジによって算出する．算出結果からテストコードの包含関係を検出する．包含関係にあるテストコードの中から，開発者が重複したテストコードであると判断したものを削除する．

テストカバレッジにはいくつかのレベルが存在する．例えば，すべてのステートメントにおいて，少なくとも一回以上実行されたステートメントの割合を示すステートメントカバレッジ (C0)，すべての条件分岐において，各分岐先が少なくとも一回以上実行された条件分岐の割合を示すデシジョンカバレッジ (C1)，すべての条件分岐の条件式を構成する論理項において，すべての項が少なくとも一回以上はそれぞれ真と偽の両方の値に評価された論理項の割合を示すコンディションカバレッジ (C2), すべての実行パスにおいて，少なくとも一回以上実行されたパスの割合を示すパスカバレッジ (C∞) がある．

テストカバレッジに基づいて判定されたテストコードの包含関係について定義する．テストコード A，B を実行した際に，任意のカバレッジによって製品コードが網羅されたと判定された要素の集合をそれぞれ S_A, S_B と定義する．例えば，ステートメントカバレッジでは網羅されたステートメントの集合が，デシジョンカバレッジでは網羅された条件分岐の集合が，コンディションカバレッジでは網羅された条件式内の論理項の集合が，パスカバレッジでは網羅された実行パスの集合が S_A, S_B に該当する．このとき，カバレッジに基づいて判定されたテストコードの包含関係は，S_A と S_B の包含関係と等価であると定義する．すなわち，S_B が S_A の部分集合である時，S_A は S_B を包含すると呼び，その際，同様にテストコード A は B を包含すると定義する．この包含関係はテストコード A と B といった二要素間の関係のみならず，テストコード A，B と C といったように複数要素間の関係について定義できる．S_A と S_B の和集合が S_C を包含している場合，テストコード A と B の組み合わせがテストコード C を包含していると定義する．同様にして 4 つ以上のテストコードの包含関係も定義する．

図 1 では，テストコードの包含関係のコンセプトを示す．右側の棒がテストカバレッジにおける製品コード中の測定単位の要素，数字は要素に番号付けしたものである．例えば，ステートメントカバレッジに基づいて判断する場合，1 つの棒が 1 つのステートメントに，数字が行番号に該当する．テストコード A によって網羅された部分はテストコード B によって網羅された部分を全て含んでおり，テストコード A はテストコード B を包含している様子を示す．

テストコードの重複はリファクタリングを行う際の匂い [3] として考えられており，重複したテストコードの削除が推奨されている．そこで，先ほど述べた定義に基づいて包含関係にあるテストコードを機械的に検出して，その上で開発者が重複していると判断したテストコードについて削除する．これによって，開発者が重複しているか判断すべきテストコードを機械的に絞り込むことができる．なお，包含関係にあるテストコードを機械的に検出するためには，各テストコードが網羅する要素の集合をそれぞれ算出して，その集合が包含関係にあるか調べればよい．この処理は，

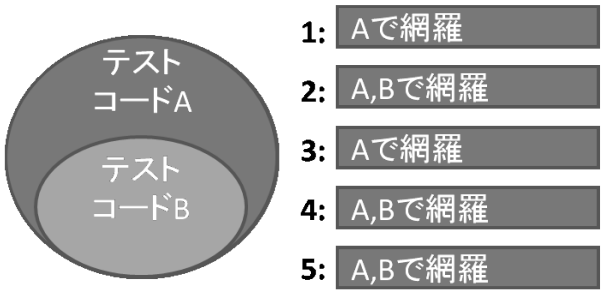


図 1　テストコードの包含

既存のツールやフレームワークで，包含関係を調べるテストコードの単位，例えば各テストケースについて，テストカバレッジと共に網羅する要素の集合を算出することで実現できる．例えば，Open Code Coverage Framework [2] を利用する場合，各テストケースが網羅する要素の集合を出力できるので，その集合について包含関係を調べることで，テストコードの包含関係を検出できる．

以下で，解法の手順を説明する．

1. **網羅する要素集合の算出**
   各テストケースなど，任意のテストコード単位で，網羅する製品コードの要素の集合を算出する．
2. **包含関係にあるテストコードの検出**
   得られた各集合がそれ以外の集合と包含関係にあるかどうか集合論に基づいて計算することで，包含関係にあるテストコードを検出する．
3. **重複テストコードの発見**
   得られた包含関係にあるテストコードの中から，開発者が重複していて不要なテストコードを探す．これは，ブラックボックステストやホワイトボックステストを問わず，片方のテストコードだけで十分テストできると判断した場合，重複していると考える．
4. **重複テストコードの削除**
   重複していると判断されたテストコードを削除する．
5. **テストコード再構成の確認**
   テストを実施して，削除を行う前とテストの結果に変化がないことを確認する．

このように，テストコードの包含関係をテストカバレッジによって判断する仕組みを用いることで，重複したテストコードの候補を機械的に検出して，それに基づいて開発者が不要なテストコードを発見して削除を行える．この解決方法によって，開発者が判断する部分を最小限に抑え，低コストで安全な削除の実施が可能となる．

6. Examples

Java 言語で実装された LRU アルゴリズムを用いたキーと値をマッピングするサンプルプログラムを考える．保持する要素数を LRU アルゴリズムによって 2 個に限定したマップになっており，3 個目の要素追加するタイミングで，使われてから最も長い時間が経過した要素を削除する．

リスト 1 がプログラム本体のソースコードで，リスト 2 がテストコードにあたる．両方のコードはテスト駆動開発の手法にのっとって実装されており，テストには少し冗長なコードも含まれている．例えば，**キー** `abc` **で値** `0` **を設定して_キー** `abc` **で値** `0` **を取得する**テストコードは，設定した値を取得できることを確かめる最低限のテストコードになっているが，**キーと値のペアを** `3` **種類設定すると一つ目のキーのみ値** `null` **を取得する**テストコードも存在する．これは，最低限のテストコードを記述して，そのテストをパスするような最低限の実装を行い，それからさらに厳密なテストコードを記述して，さらにテストをパスするような最低限の実装を行うというテスト駆動開発のサイクルにそってソースコードとテストコードが実装されたためである．この二つのテストコードは明らかに重複関係にあり，後者のテストコードがあれば前者は不要であると考えられる．他にも，**インスタンスを生成する**という極めて簡潔なテストコードも存在する．これは，ソースコードの実装を開始するにあたって，テストフレームワークが正常に動作しているかどうかの確認も込めて記述されている．しかし，明らかにそれ以外のテストコードに包含されているため，このテストコードも不要であると考えられる．

次に，テストカバレッジに基づくテストコードの重複判定の例を示す．**キー** `abc` **で値** `0` **を設定して_キー** `abc` **で値** `0` **を取得する**テストコードに対して，ステートメントカバレッジを測定すると 20,32,33 行目が未実行と出力される．**キーと値のペアを** `3` **種類設定すると一つ目のキーのみ値** `null` **を取得する**テストコードに対して，ステートメントカバレッジを測定すると 20 行目のみが未実行と出力される．したがって，ステートメントカバレッジという測定基準において，後者は前者を包含している．この場合，テストコードを実際に見たうえで，後者のみで十分テストできると判断できる．テストコードの増大によって保守性と効率性の低下が問題になっている場合は，重複している前者のテストコードを削除することで，改善を図ることができる．

このように，テストカバレッジに基づいて機械的に重複関係にあたるテストコードを発見することができ，見つかったテストコードを削除するか否かを開発者が判断をした上で削除を行える．

List 1　LRU を利用したマップの Java ソースコード

```
1 import java.util.HashMap;
2 import java.util.LinkedList;
```

```
3  import java.util.Map;
4  import java.util.Queue;
5
6  public class LruCache<TKey, TValue> {
7
8    private Map<TKey, TValue> _map;
9    private Queue<TKey> _orderedKeys;
10
11   public LruCache() {
12     _map = new HashMap<TKey, TValue>();
13     _orderedKeys = new LinkedList<TKey>();
14   }
15   public TValue get(TKey key) {
16     return _map.get(key);
17   }
18   public void put(TKey key, TValue value) {
19     if (_map.containsKey(key)) {
20       _orderedKeys.remove(key);
21     }
22     else {
23       updateCache();
24     }
25
26     _map.put(key, value);
27     _orderedKeys.add(key);
28   }
29   private void updateCache() {
30     // サイズ2のLRUリスト
31     if (_orderedKeys.size() == 2) {
32       TKey key = _orderedKeys.poll();
33       _map.remove(key);
34     }
35   }
36 }
```

List 2　LRU を利用したマップの Java テストコード

```
1  import static org.junit.Assert.*;
2  import org.junit.Test;
3
4  public class LruCacheTest {
5    @Test
6    public void インスタンスを生成する() {
7      // テストが正常に実施できることを確認
8      assertNotNull(new LruCache<String,Integer>());
9    }
10   @Test
11   public void 空の状態から値nullを取得する() {
12     LruCache<String, Integer> cache =
13       new LruCache<String, Integer>();
14     assertNull(cache.get("a"));
15   }
16   @Test
17   public void キーabcで値0を設定する() {
18     LruCache<String, Integer> cache =
19       new LruCache<String, Integer>();
20     // 例外が発生しないことを確認
21     cache.put("abc", 0);
22   }
23   @Test
24   public void キーabcで値0を設定して_
25       キーabcで値0を取得する() {
26     LruCache<String, Integer> cache =
27       new LruCache<String, Integer>();
28     cache.put("abc", 0);
29     assertEquals(0, (int)cache.get("abc"));
30   }
31   @Test
32   public void キーxyzで値1を設定して_
33       キーabcで値nullを取得する() {
34     LruCache<String, Integer> cache =
35       new LruCache<String, Integer>();
36     cache.put("xyz", 1);
37     assertEquals(null, cache.get("abc"));
38   }
39   @Test
40   public void キーabcで値0_キーxyzで値1を設定して_
41       キーabcで値0_キーxyzで値1を取得する() {
42     LruCache<String, Integer> cache =
43       new LruCache<String, Integer>();
44     cache.put("abc", 0);
45     cache.put("xyz", 1);
46     assertEquals(0, (int)cache.get("abc"));
47     assertEquals(1, (int)cache.get("xyz"));
48   }
49   @Test
50   public void キーと値のペアを3種類設定すると
51       一つ目のキーのみ値nullを取得する() {
52     LruCache<String, Integer> cache =
53       new LruCache<String, Integer>();
54     cache.put("abc", 0);
55     cache.put("xyz", 1);
56     cache.put("def", 2);
57     assertNull(cache.get("abc"));
58     assertEquals(1, (int)cache.get("xyz"));
59     assertEquals(2, (int)cache.get("def"));
60   }
61 }
```

7. Consequences

テストカバレッジに基づいて包含関係にあるテストコードを検出することで，機械的に削除候補である重複テストコードの絞り込みを行うことができる．開発者が包含関係にあるテストコードの中から重複テストコードを発見することで，人手による判断を極力減らして，効率よく安全にテストコードを削除できる．このようにして，重複したテストコードを削除することで，テストコードの増大によるテストの効率性の低下と，ソフトウェアの保守性の低下を改善することができる．さらに，テストコードにおける DRY 原則を遵守できる．

以上から，提案するパターンによって，最小限のコストで重複テストコードの削除を達成する．

8. Known Uses

.NET ライブラリの Paraiba [4] は，本パターンによる重複テストコードの削除により，テスト実施時間の大幅削減に成功している．

Paraiba は GitHub にて管理されているオープンソースプロジェクトで，リビジョン番号が 6c5a74cc832a865969b6c30883b000390753d8de のコードに対して本パターンが適用されて，リビジョン番号が 125da5dacd0d16b4546b5c63052a34869c5d846d のコードが得られている．

Paraiba では MergeSort のテストコードとして，長さ 0 から 11 までの添え字と値が一致しているようなリストに対して，全ての順列を求めてソーティングを行うテストケース 12 個と，特定の値を保持するリストに対するテストケース 14 個が存在していた．ステートメントカバレッジとデシジョンカバレッジに基づいて重複するテストコードを判断して削除することで，本パターンの適用後のテストケースは 14 個にまで削減され，テストの実施時間は 2 分程度かかっていたものが，1 秒程度まで削減されている．

この事例から本パターンの有用性が示されている．

9. Related Patterns

Template Method Pattern は，ある親クラスに対して子クラスの細部の処理は異なっていても，大まかな処理の流れが同じ場合に適用されるパターンである．この時，どの子クラスも処理の大まかな流れは同じであるため，それぞれの子クラスが全ての処理を記述するのではなく，親クラ

スで処理の大まかな流れを共通処理として括り出し，それ以外の部分を抽象メソッドとして定義しておく．子クラスでは抽象メソッドをオーバーライドすることによって，それぞれ異なる細部の処理を記述する．したがって，Template Method Pattern を適用することで，それぞれの子クラスでの重複処理が削除され，DRY の原則が遵守される．この点で，本パターンは Template Method Pattern と関連性が強い．

参考文献

[1] Kent Beck, ”Test-Driven Development: By Example”, The Addison-Wesley Signature Series, 2003.

[2] Kazunori Sakamoto, Open Code Coverage Framework, http://sourceforge.jp/projects/codecoverage/.

[3] A. van Deursen, L. Moonen, A. van den Bergh, and G. Kok, “ Refactoring Test Code ”, Extreme Programming Perspectives, M. Marchesi, ed., pp.92-95, 2001.

[4] Kazunori Sakamoto, Paraiba, http://github.com/KAZUu/Paraiba.