

第33回CASTeXセミナー

日時:平成23年11月13日(日)15:00~17:00 E

会場:工学院大学新宿キャンパス 27階 A2710教室

# 「CASの教育,プログラミングへのソフトウェア工学最新技法の援用」

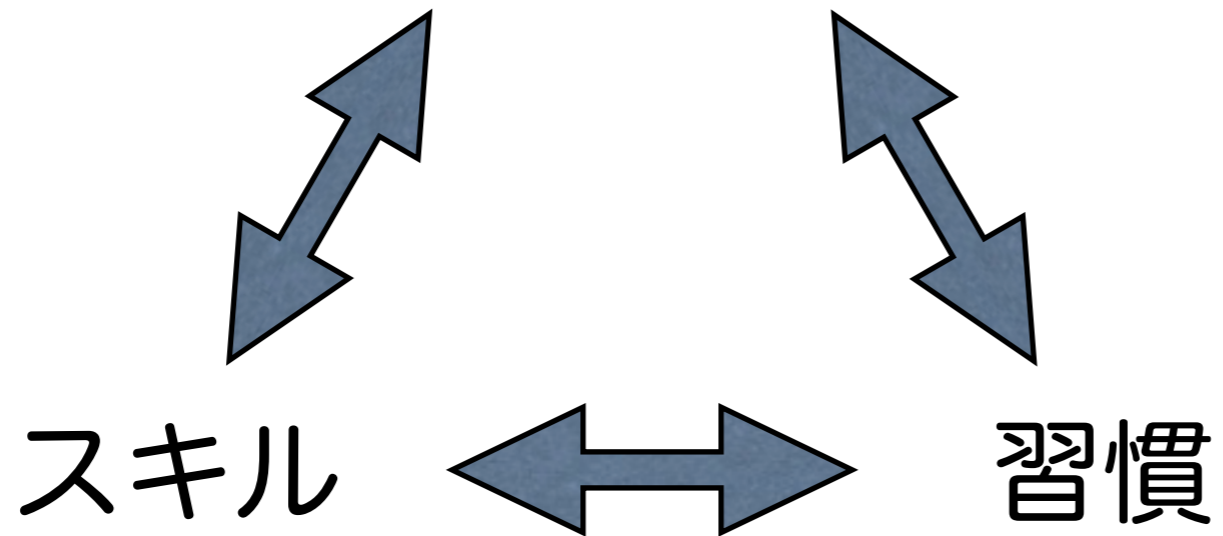
関西学院大学 西谷滋人

# 教育：知識の定着

知識の形：

パターン, wikipedia

動機



リファクタリング

フロー, ペアプロ

# 名称

# 写真や図

# 上位参照

# 問題

## 115 生き生きとした中庭\*\*

COURTYARDS WHICH LIVE



・・・正の屋外空間(106)と段階的な屋外空間(114)により明らかになった屋外空間の全体計画のなかで、30から40フィート(9~12m)以下の最小の屋外空間——中庭——には、特別の注意を払わねばならない。というのは、うっかりすると死んだ空間になる場合が多いからである。

\*\*\*

現代建築に組み込まれた中庭は、大抵の場合、死んだ空間である。私的に使用するオープンスペースとして意図されているにもかか

わず、結局、砂利と抽象彫刻に埋まった無用の長物に終わっている。



死んだ中庭

中庭が失敗する顕著な原因が3つ考えられる。

1. 屋内と屋外とのあいだにあいまいな領域がない。屋内から屋外へ導く壁、引き戸、ドアなどが唐突すぎると、両者の中間でためらい、一瞬のひらめきで屋外に出ようという機会が失われてしまう。人にはあいまいな中間領域が必要である——つまり、住居内の日常生活でたびたび通り過ぎながら、ごく自然に屋外にさまよい出られるようなポーチとかベランダである。

2. 中庭に出るドアが少ない。ドアが1つしかない、住居内に中庭をはさむ2つの活動が成立しないから、中庭を通過して日常的にそこを活気づけるということがない。これを克服するには、中庭の両端に少なくとも2つのドアを設けねばならない。そうすれば中庭が異なる活動の合流点になり、近道になり、受け皿になり、双方の活動に交流が生まれるのである。

3. 中庭が閉鎖的すぎる。居心地のよい中庭には、つねに何かより広く遠い空間を見通せるような「抜け穴」があるように思える。中庭を完全に部屋で囲まずに、何か先の空間が少しでも垣間見れるようにすべきである。

つぎに、世界各地の生き生きとした中庭の例を、大小とり混ぜて示すことにする。



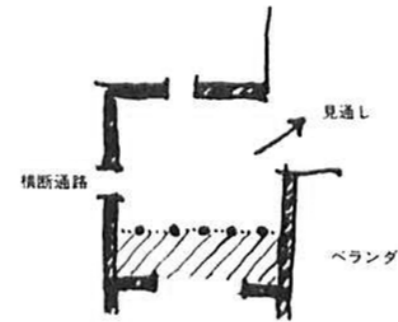
生き生きとした中庭

どれも周囲の建物の活動に部分的に開かれているが、なお私的な空間である。そこを通りぬける人や走りぬける子供たちを、垣間見たり感知できるが、中庭の雰囲気をぶち壊すほどではない。もう一度、どの中庭も別の空間と強いつながりをもつことに注目されたい。これらの写真がすべてを物語るわけではないが、中庭から通路ぞいに、または建物を貫通してより大きな空間を見通せることが分かる。さらに最も注目すべきは、どの中庭でも、天気と気分しだいでさまざまな場所を占

められるということである。そこには、庇のかかっている場所、日の当たる場所、木もれ日の落ちる場所、地面で横になれる場所、居眠りのできる場所などが備わっている。中庭の外周部と隅部はあいまいで豊かな表情をもち、建物の壁面は開かれており、中庭と建物内部が直接つながっている部分もある。

したがって、

どんな中庭でも、つねにより大きな屋外空間を見通せるように配置すること。周囲の建物に少なくとも2、3か所ドアを設け、これらのドアを結ぶ自然な通路が中庭を横切るようにすること。さらに中庭の一边のドアの脇に、屋内と中庭の両方に連続する屋根つきのベランダかポーチを設けること。



\*\*\*

アーケード(119)、外廊(166)、一間バルコニー(167)にしたがってポーチをつくり、日当たりを確保すること——日のあたる場所(161)。段階的な屋外空間(114)と禅窓(134)にしたがって、外側への見通しを用意すること。戸外室(163)や庭田(173)のような造りにして、さらに中庭を囲うこと。中庭の周囲の軒高は、同じ高さにすること。切妻屋根が面する場合は、寄棟にして軒先の高さを

揃えること——屋根の割り付け(29)中央の焦点(126)・・・

# 解答

# 下位参照

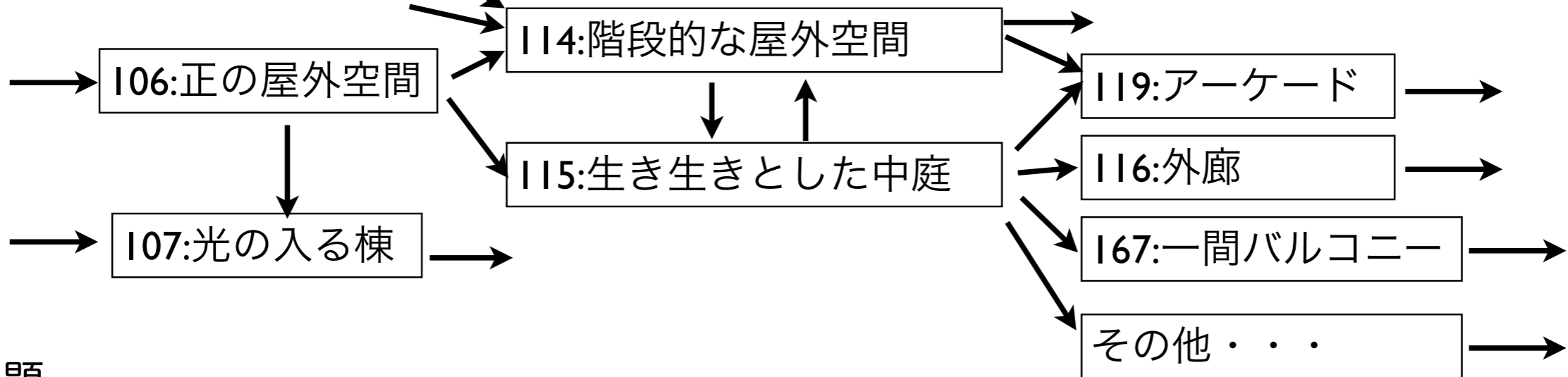
# パタン・ランゲージの「No.115:生き生きとした中庭」を例にした体裁

大 ← 粒度 → 小

町やコミュニティー

建物

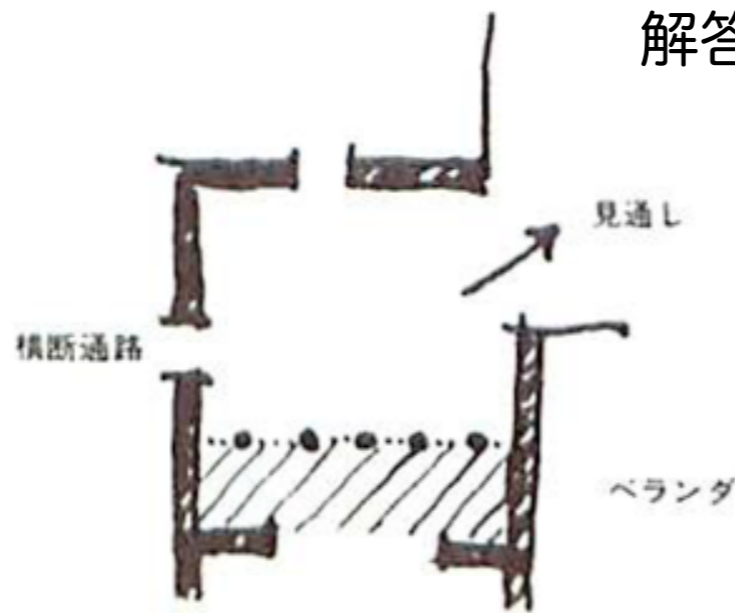
施工



問題

現代建築に組み込まれた中庭は、大抵の場合、死んだ空間である。私的に使用するオープンスペースとして意図されているにもかかわらず、結局、砂利と抽象彫刻に埋まった無用の長物に終わっている。

解答



- ・どんな中庭でも、つねにより大きな屋外空間を見通せるように配置すること。
- ・周囲の建物に少なくとも2, 3カ所ドアを設け、これらのドアを結ぶ自然な通路が中庭を横切るようにすること。
- ・さらに中庭の一边のドアの脇に、屋内と中庭の両方に連続する屋根付きのベランダかポーチを設けること。

パタン・ランゲージの「No.115:生き生きとした中庭」を中心にして、粒度,上下 参照の模式図,問題,覚えやすいイラスト,箇条書きの解答.

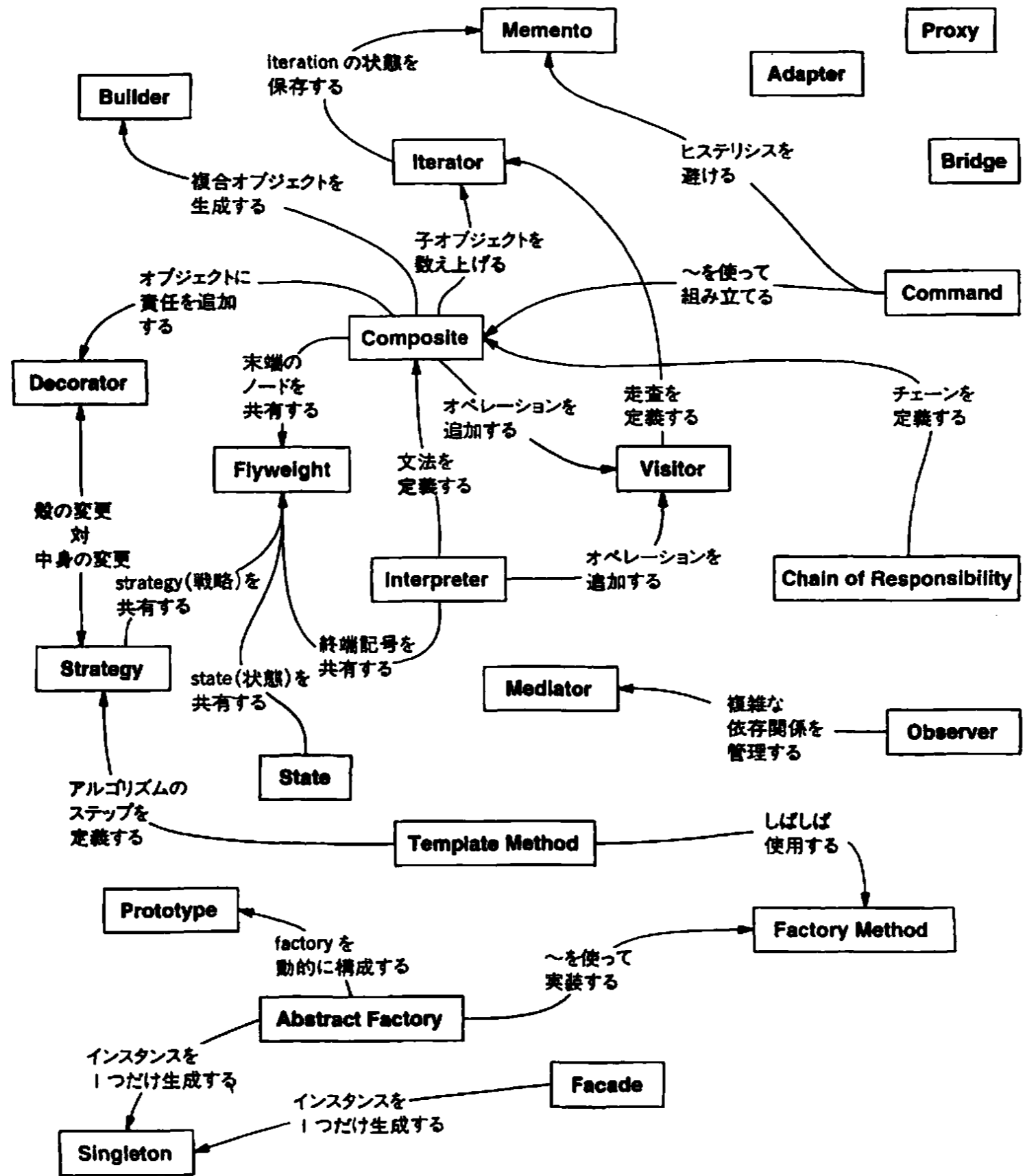


図 3: 23 のデザイン・パターンの相互関係 [4].

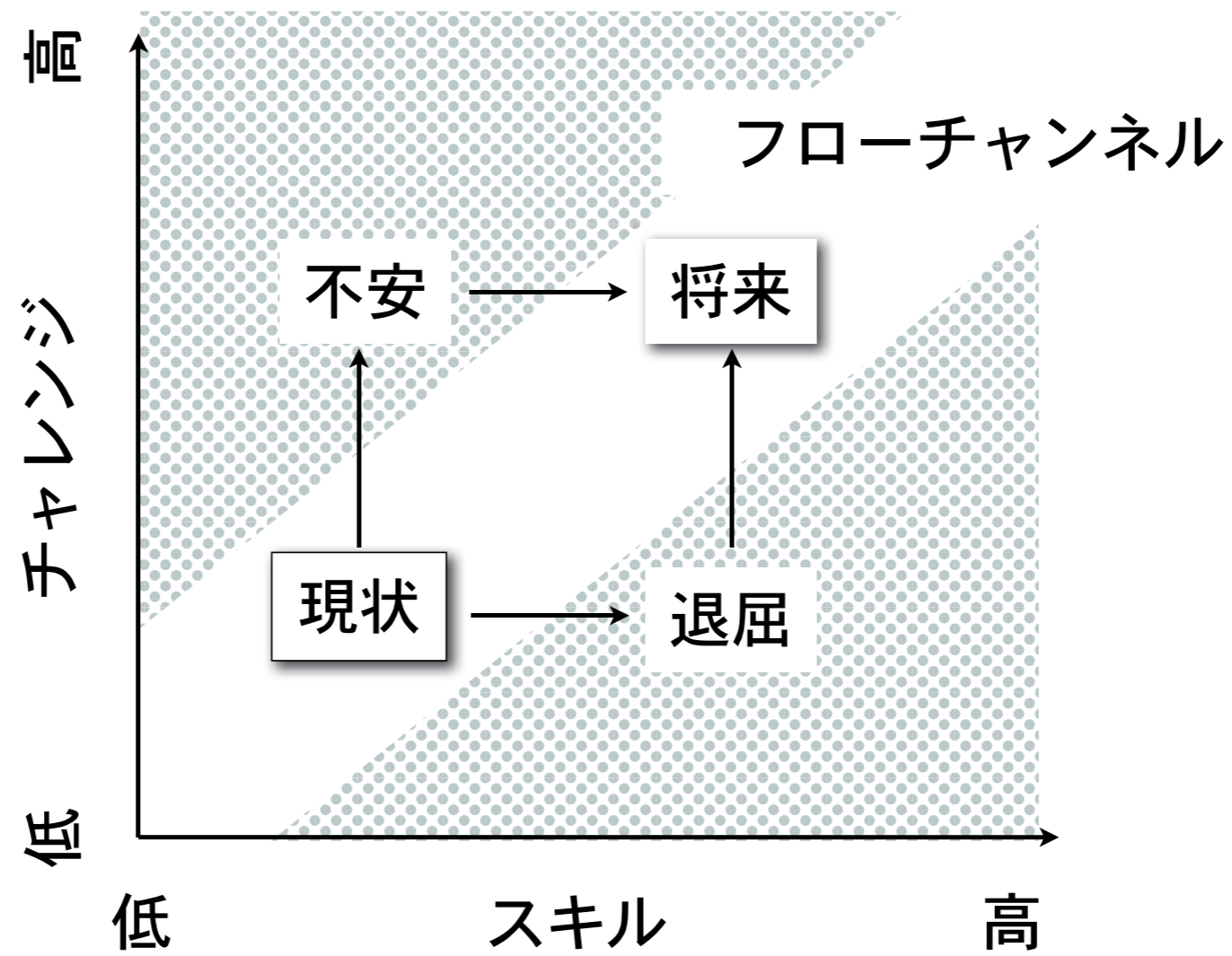
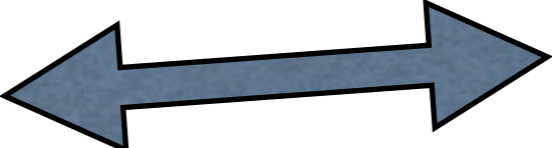


図 6: フロー状態のスキルとチャレンジの関係を示す模式図.

# アジャイルソフトウェア開発宣言

## (2001,Utah)

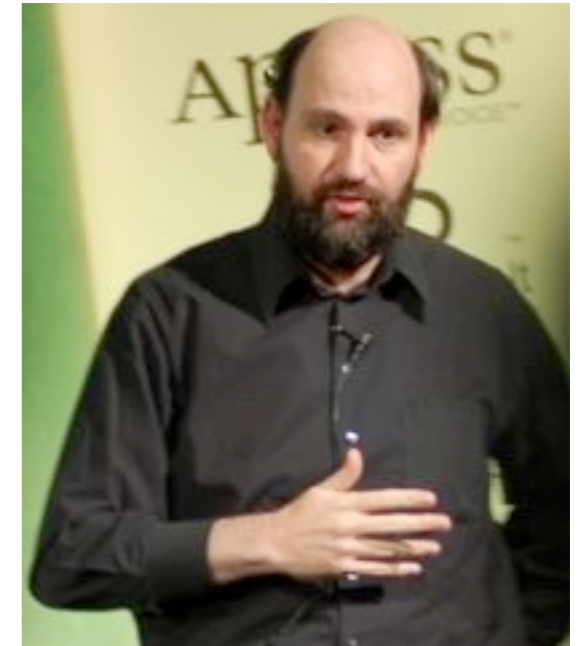
- アジャイルソフトウェア開発の価値
  - プロセスやツールより人と人同士の相互作用を
  - 包括的なドキュメントより動作するソフトウェアを
  - 契約上の交渉よりも顧客との協調を
  - 計画に従うことよりも変化に対応することを
- 重視する。

- ・アジャイル開発  ウォーターフォール
- ・ペアプロ（学生実験）
  - ・リファクタリング（カイゼン）
  - ・オブジェクト指向
    - ・制御構造（構造化）
    - ・カプセル化（モジュール）
    - ・ポリモーフィズム
    - ・インヘリタンス
    - ・動的束縛
  - ・パターン（チャート式）



# リファクタリング

- 動作を変えない
- 中身を再構成
- ぷーんと臭ったら換えてあげるの
- Martin Fowler,
  - 『リファクタリング: プログラムの体質改善テクニック』 (ピアソン, 2000).
  - Refactoring: Improving the Design of Existing Code (June 1999)
  - リファクタリング: Rubyエディション(2010)
- Kevin Rutherford,
  - リファクタリングRuby—実践ワークブック(ピアソン, 2010/11)



```
sparky.rb
/Users/bob/Desktop/Ruby/Refactoring/rwb/before/sparky.rb   モード: Ruby   最終更新日時: 2011/1/6 17:49
UTF-8   LF (UNIX)   ウィンドウ幅   Ruby   +   ◀▶

1  NUMBER_OF_TOSSES = 1000 ↓
2  BORDER_WIDTH = 50 ↓
3  ↓
4  def toss ↓
5    2 * (rand(2)*2 - 1) ↓
6  end ↓
7  ↓
8  def values(n) ↓
9    a = [0] ↓
10   n.times { a << (toss + a[-1]) } ↓
11   a ↓
12 end ↓
13 ↓
14 def spark(centre_x, centre_y, value) ↓
15   "<rect x=\"#{centre_x-2}\" y=\"#{centre_y-2}\" ↓
16     width=\"4\" height=\"4\" ↓
17     fill=\"red\" stroke=\"none\" stroke-width=\"0\" /> ↓
18   <text x=\"#{centre_x+6}\" y=\"#{centre_y+4}\" ↓
19     font-family=\"Verdana\" font-size=\"9\" ↓
20     fill=\"red\" >#{value}</text>" ↓
21 end ↓
22 ↓
23 $tosses = values(NUMBER_OF_TOSSES) ↓
24 points = [] ↓
25 $tosses.each_index { |i| points << "#{i},#{200-$tosses[i]}" } ↓
26 ↓
27 data = "<svg xmlns=\"http://www.w3.org/2000/svg\" ↓
28     xmlns:xlink=\"http://www.w3.org/1999/xlink\" > ↓
29   <!-- x-axis --> ↓
30   <line x1=\"0\" y1=\"200\" x2=\"#{NUMBER_OF_TOSSES}\" y2=\"200\" ↓
31     stroke=\"#999\" stroke-width=\"1\" /> ↓
32   <polyline fill=\"none\" stroke=\"#333\" stroke-width=\"1\" ↓
33     points = \"#{points.join(' ')}\" /> ↓
34   #{spark(NUMBER_OF_TOSSES-1, 200-$tosses[-1], $tosses[-1])} ↓
35 </svg>" ↓
36 ↓
37 puts "Content-Type: image/svg+xml ↓
38 Content-Length: #{data.length} ↓
39 ↓
40 #{data}" ↓
41
```

```
1.9.0.rb
/Users/bob/Desktop/Ruby/Refactoring/rwb/lecture/1.9.0.rb   モード: Ruby   最終更新日時: 2011/1/6 17:49
UTF-8   LF (UNIX)   ウィンドウ幅   Ruby   +   ◀▶

1  require 'sparkline' ↓
2  require 'pp' ↓
3  ↓
4  def zero_or_one ↓
5    rand(2) ↓
6  end ↓
7  ↓
8  def one_or_minus_one ↓
9    (zero_or_one * 2) - 1 ↓
10 end ↓
11 ↓
12 def next_value(y_values) ↓
13   t = y_values[-1] + one_or_minus_one ↓
14 end ↓
15 ↓
16 def make_y_values ↓
17   result = [0] ↓
18   1000.times { ↓
19     result << next_value(result) ↓
20   } ↓
21   return result ↓
22 end ↓
23 ↓
24 puts Sparkline.new(make_y_values).to_svg ↓
25 ↓
26 ↓
27 ↓
28
```

段落 1/28