ソフトウェア工学 121－20
(1998. 11. 6)

デザインパターンへの適合性確認手法について

畑口剛之, 池田健次郎, 岸知二
NEC ソフトウェアデザイン研究所 システムデザイン技術部
108-8557 東京都港区芝浦 2 丁目 11-5 NEC 五十嵐ビル
03-5476-1089
thataguc@ccs.mt.nec.co.jp

あらまし
デザインパターンを効果的に用いてソフトウェアの設計を行なうことは, 生産性やソフトウェアの品質を高めるために有効な手段である. そうして得られた設計のレビューを行なう際には, デザインパターンがその意図どおり適切に使われているかどうか確認をする必要がある. しかし, その確認はデザインパターンが組合わさって適用されている場合や, デザインパターンと設計の抽象度に差がある場合などに必ずしも容易ではない. 本稿では, デザインパターンに対する適合性の概念を導入するとともに, 適合性確認をツール支援するための手法について考察する.

キーワード　デザインパターン, 設計, 適合性, コンフォーマンス, オブジェクト指向設計

On Conformance Checking to Design Pattern

Takeshi HATAGUCHI, Kenjiroh IKEDA and Tomoji KISHI,
NEC Software Design Laboratories System Design Technology Laboratory
2-11-5 Shibaura Minato Ku Tokyo Japan 108-8557
+81-3-5476-1089
thataguc@ccs.mt.nec.co.jp

Abstract
Designing software using the design patterns is one of the promising ways to improve the productivity and the quality of software.
However, it is not straightforward to check whether the patterns are used properly in the design, especially because dynamic aspect of the software tends to be complicated.
In this paper, we introduce the notion of "conformance to design pattern", and discuss on the tool support for the conformance checking.

key words　design pattern, design, conformance, Object Oriented Design

1 はじめに

デザインパターンは，ソフトウェアの設計において生じる課題を解決するために使われるパターンであり，設計課題や，それを解決するためのヒントなどと共に提供される．それはしばしばカタログの形でまとめられており，特にGammaらによるデザインパターン [1] は有名である．また，パターンを用いたソフトウェアの開発手法についての研究も行なわれている [3][4][5].

デザインパターンを用いた設計のレビューを行う際には，デザインパターンがその意図どおり適切に使われているかどうか確認をする必要があるが，一般にその確認は容易ではない．特に協調関係の確認はデザインパターンのメッセージのインタラクションが様々なバリエーションをもって実行トレースに現れることがあること，複数のインタラクションが組合わさって現れることなどの理由で極めて繁雑である．

我々はデザインパターンを有効に活用する方法を検討しているが，本稿では上記の確認を容易にすることを目的とし，デザインパターンに対する適合性という概念を導入するとともに，適合性の確認の手法について検討する．特に，この適合性確認をすべて人手で行うのは困難なので，ツール支援の可能性を考える．

しかしながら，デザインパターンは設計者が読んで理解するためのドキュメントとして提供されているため，その記述に基づいてツール処理することは困難である．また，適切な問題に対してデザインパターンが適用されているかどうかなどの意味的な解釈を必要とする確認は，本質的にツール支援が困難である．

そこで本稿では，適合性確認作業のうち，ツール支援が可能であり，かつ有効と考えられる作業を洗い出し，そこに対するツール支援の方式について考察する．

2章ではデザインパターンへの適合性の確認とはなにかを説明する．3章では適合性の確認を行なう上での問題点を分析し，具体的にどのような状況で適合性の確認が困難であるか例をあげる．4章ではこの適合性の確認のうち，本稿で取り扱う問題を明らかにすると共にその解法の方針を述べる．5章では適合性の確認を行なう一手法について，どのような方針で処理するかの概略を述べる．6章では具体的な例題をあげて，5章で述べた処理がどのように適用されるか述べる．7章では本ツール支援の利用場面と他研究との関係について述べる．

"On Conformance Checking to Design Pattern", Takeshi HATAGUCHI, Kenjiroh IKEDA and Tomoji KISHI, NEC Corporation

2 デザインパターンへの適合性の確認

本章ではデザインパターンへの適合性について説明する．

デザインパターンの記述の内容は，"目的"，"構造"，"協調関係" に集約されて表現されると考えられる [1].

目的に関しては，そのデザインパターンの意図や設計課題が文章を用いて表現される．

構造に関しては，そのデザインパターンに参加するクラスやクラス間の静的な関係がクラスダイアグラムを用いて表現される．

協調関係に関しては，構造中で示されたクラスに対応するオブジェクトが，どのような協調動作を行うかが示される．こうした協調動作を表現するためにインタラクションダイアグラムが用いられる．

一方，適合性を確認する対象の設計は，一般的にソースコードや，モデル (クラスダイアグラム，ステートダイアグラム，インタラクションダイアグラムなどからなる) などの形式で表現されている．

本稿では，設計のデザインパターンへの適合性は次の3つの適合性から構成されるものとする．

目的への適合性
: デザインパターンが適切な問題に対して適用されていること

構造への適合性
: 対象の設計の該当部分の構造が，デザインパターンの示す構造と対応していること

協調関係への適合性
: 対象の設計の該当部分の協調関係が，デザインパターンの示す協調関係と対応していること

本稿では，デザインパターンを用いた設計のレビューを行なう状況を想定し，上記の適合性のうち特に協調関係の適合性の確認をツール支援することを検討する．構造への適合性は，人手で行なうものとし，構造上の対応は，設計者が指定するものとする．また，目的への適合性の確認を自動的に行なうのは本質的に困難であると考えられるので本稿では扱わない．

3 問題の分析

本章では適合性の確認を行なう上での問題点を分析し，具体的にどのような状況で適合性の確認が困難であるか例を示す．

3.1 デザインパターンの記述性

デザインパターンは人が読むことを前提として記述されている．例えばインタラクションダイアグラムは，協調関係の一例を示しているに過ぎないが，文章として記述されたパターンの説明を読むことにより，例示されたインタラクション以外の様々なバリエーションが含まれることを理解することができる．しかしながら，ツール支援をする際には，こうした潜在的なバリエーションをより明示的に表現しなければならない．この問題をデザインパターンの記述性の問題と呼ぶことにする．

例えば以下のような状況において，デザインパターンの記述性の問題がおこる．

状況 1-1

パターンでメッセージの組の繰り返しを意図したい状況

まさしく例示どおりの回数のインタラクションしか許されないのか，他のバリエーションが許容されるのか等

状況 1-2

オブジェクトの数が決まっていない状況

まさしく例示どおりの数のオブジェクトしか存在し得ないのか，他のバリエーションが許容されるのか等

状況 1-3

クラスの数が決まっていない状況

まさしく例示どおりの数のクラスしか存在し得ないのか，他のバリエーションが許容されるのか等

3.2 抽象度の違い

一般にデザインパターンのクラスやメッセージは設計のクラスやメッセージと必ずしも 1 : 1 に対応しない．このような状況は，デザインパターンと設計の抽象度が違うことがあるために起こる．この問題をデザインパターンと設計の抽象度の違いの問題と呼ぶことにする．

デザインパターンのクラスやメッセージが，設計のクラスやメッセージと 1 : 1 に対応しない状況としては，以下のようなものが考えられる．

状況 2-1

デザインパターンのクラスと設計のクラスが

$n : m$ $(n, m = 1, 2, ...,\ \neg(n = 1 \wedge m = 1))$

に対応している状況

状況 2-2

デザインパターンのクラスと設計のメッセージが

$n : m$ $(n, m = 1, 2, ...,\ \neg(n = 1 \wedge m = 1))$

に対応している状況

例えば状況 2-1 において，$n = 1$ の場合は，デザインパターンのひとつのクラスの役割が，設計では複数のクラスに分割されて表現されており，$m = 1$ の場合は，それらが統合されて設計に現れている状況である．

4 問題設定，方針

前章までに適合性の確認がどのように役立つものか，自動的に確認を行なおうとするとどのような問題点があるかをまとめた．本稿ではこれらの問題のうち，いくつかの問題に対する改善方法を検討する．

4.1 問題設定

本稿では，協調関係の適合性の確認を行なう際に対象となる設計をなんらかの方式で実行させ，その実行トレースを用いて適合性の確認を行なうものとする．ソースコードであればコンパイルして実行したり，モデルであればシミュレータを用いて実行したりするなどの方式が考えられる．

本稿で提案する方式の入力は以下のようなものであるとする．

- デザインパターンの情報
  - クラス，メッセージの種類
  - メッセージのインタラクション
  - 記述性をあげるための情報
- 適合性確認の対象の設計の情報
  - クラス，メッセージの種類
  - 実行トレース
- 構造への適合性の指定情報

なお，記述性をあげるための情報は 3.1 節で述べたような状況に対して考慮を行なうためのものである．ただし，本稿では “状況 1-1” のみを対象にすることにする．

また，構造への適合性の指定情報は，3.2節で述べた抽象度の違いを考慮していなければならない．ただし，本稿では “状況 2-1”，“状況 2-2” の状況の中で，$n = 1$ の状況だけを対照することにする．他の状況に対する確認手法については、今後検討するものとする。

本稿で提案する方式の出力は以下のようなものであるとする.

- 適合性確認の対象の設計の実行トレースの中でデザインパターンで定義しているメッセージのインタラクションと一致する部分

4.2 適合性確認手法の方針

上記入力に対して以下で示す方針で処理を行ない, 上記出力を得る.

- デザインパターンの情報, 適合性確認の対象の設計の情報を入力として受けつける
- 構造への適合性の指定情報を人が指定する
- 上記情報を元に, 設計の実行トレースのメッセージをパターンのメッセージのインタラクションのメッセージに対応づける
- 対応づけることができた設計の実行トレースのメッセージを出力する

この出力はデザインパターンに対してインタラクションのパターンが適合する部分を示すものである. もちろんパターンとしてマッチしているからといって, 本当に適切な状況でパターンが使われているかどうかといった意味的な判断は人手で行わなければならない. 実際に適合性を確認する際には, この出力結果がそうした判断の参考として用いられることを想定している.

5 適合性の確認手法

前章で本稿で扱う適合性の確認の範囲と方針を述べた. 本章では適合性の確認を行なう一手法について, どのように確認をするのか、その処理の概要を述べる。

そのために, まず前章で定義した入力情報を具体化し, 次に具体的な確認方法について述べ, 最後に得られる出力について述べる.

5.1 デザインパターンが持つ情報

前章で示したデザインパターンが持つ情報の中で本章で説明する方式で必要なものをまとめる.

デザインパターンではメッセージの送信とメッセージの受信のタイムラグは考えないモデルで表現されているが, 本稿では適合性確認の対象の設計をなんらかの形で実行させ, その実行トレースとデザインパターンのインタラクションダイアグラムの適合を確認するという立場をとっているため, メッセージの送信とメッセージの受信のタイムラグは無視できない. 本稿では, メッセージ単位ではなく以下のように定義できるイベントという概念を用いてデザインパターンが持つ情報を表現する.

送信イベント
　あるメッセージの送信に対応するもの.

　情報
　　< 送信元オブジェクト, 受信先オブジェクト, メッセージの種類 >

受信イベント
　あるメッセージの受信に対応するもの.

　情報
　　< 対応する送信イベント >

イベント
　送信イベント, もしくは受信イベント

イベントの順序列
　順序を定義したイベントの列

　情報
　　E1 < E2 < E3 < ...
　　(イベント **E1**, イベント **E2**, イベント **E3** の順で発生した)

ただし, デザインパターンの協調関係は, 協調関係を構成するすべてのイベントの全順序は定義せず, 半順序関係しか定義しない. イベントの半順序関係は, 複数のイベントの順序列で表現することができる (ただし, 一意に表現できるわけではない).

例えばイベントの半順序関係は図 1のような形で定義される, **E1**, **E2**, **E3** がイベントであり, **BEGIN** はどのイベントよりも早い時間に発生したとする仮のイベントであり, **END** は同様にどのイベントより遅い時間に発生したとする仮のイベントである.

図 1の例はイベントの順序列で表現すると, 例えば以下のように表現される.

BEGIN < E1 < E3 < END
E1 < E2 < END

図 1では, **E1** と **E2** の間の順番は定義されているが, **E2** と **E3** の間の順番が定義されていない.

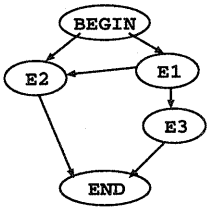

図 1: イベントの半順序関係

3.1 節で述べた状況の中で, “状況 1-1: パターンでメッセージの組の繰り返しを意図したい状況” に対処できるように, どのメッセージの組を繰り返すかの情報を加えることにする.

以上により, デザインパターンが持つ情報の中で本章で説明する方式で必要なものは以下のようになる.

構造の情報

クラス, メッセージの種類

協調関係の情報

送信イベント, 受信イベント, イベントの半順序, メッセージの組の繰り返し

5.2 適合性確認の対象の設計が持つ情報

前章で示した適合性確認の対象の設計が持つ情報の中で本章で説明する方式で必要なものをまとめる.

協調関係は, デザインパターンの協調関係の情報と同様に実行トレースをイベントを用いて表現したものにする. 適合性確認の対象の設計の場合, イベントには図 2に表されるような全順序関係が定義される.

図 2では, E1, E2, E3, E4 の任意の組合せのイベントについて, 順番は定義されている.

図 2の例はイベントの順序列で表現すると, 以下のように表現される.

$$\mathrm{BEGIN} < \mathrm{E1} < \mathrm{E2} < \mathrm{E3} < \mathrm{E4} < \mathrm{END}$$

以上により, 適合性確認の対象の設計が持つ情報の中で本章で説明する方式で必要なものは以下のようになる.

構造の情報

クラス, メッセージの種類

協調関係の情報

実行トレースから得られるイベントの全順序

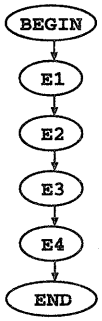

図 2: イベントの全順序関係

5.3 構造への適合性の指定情報

2節で述べたように, 本稿で構造への適合性は, 人が指定することにする.

構造への適合性の指定情報を次のように定義する.

- デザインパターンと設計のクラス対応関係
- デザインパターンと設計のメッセージの対応関係

5.4 確認方法

以上の問題設定に対して, 適合性の確認を行なうために以下の言葉を定義する.

構造の対応

デザインパターンと設計のクラスの対応とメッセージの対応.

構造の対応に矛盾しない

設計のイベントをデザインパターンのイベントに対応づける際に, その対応づけ方が構造の対応で定義されているクラスの対応と, メッセージの対応と矛盾しないこと.

特に, デザインパターンと設計のクラスが $1:n(n=2,3,...)$ に対応している場合には, デザインパターンのクラスは, 設計の n 個のクラスどれに対応づけられてもいい.

ただし, メッセージが $1:n(n=2,3,...)$ に対応している場合には, デザインパターンのメッセージの送信イベントは, 設計の n 個のメッセージの先頭のメッセージの送信イベントに対応し, デザインパターンのメッセージの受信イベントは, 設計の n 個のメッセー

ジの最後のメッセージの受信イベントに対応するとする.

適合性の確認は以下の方法で行う.

1. 設計のイベントを先頭から, 次の条件を満たすデザインパターンのイベントに対応づける
   - 構造の対応に矛盾しない
   - 対応づけられるデザインパターンのイベントより, 前の順序のすべてのイベントがすでに設計のイベントに対応づけられている
   - 設計のあるメッセージの送信イベントと受信イベントはデザインパターンの同一メッセージの送信イベントと受信イベントに対応づけられる

ただし, メッセージが $1:n(n=2,3,...)$ に対応している場合には, 設計の n 個のメッセージの先頭のメッセージの送信イベントと設計の n 個のメッセージの最後のメッセージの受信イベントが, それぞれデザインパターンのイベントに対応づけられることができるのと同時に, n 個のメッセージに対応するイベントがそれらに定義されている半順序関係を満たしていなければならない.

また, メッセージの組の繰り返しが含まれている場合は, 可能な限り多くの回数の繰り返しがあったと仮定して, 設計のイベントをデザインパターンのイベントに対応づけ, 割り当てに失敗したならバックトラックが生じるようにする.

構造の対応と矛盾しないすべてのイベントの割当の組合せを求めるには, バックトラック法を用いればよい.

5.5 結果の得られ方

結果はデザインパターンのイベントに対応づけられた設計のイベントの列で与えられる. これは, 設計のすべてのイベントの部分列となる.

設計のイベントの全順序が,

BEGIN < E1 < E2 < E3 < E4 < END

のように表され, 5.4節の方法で割当を行なった結果, 例えば E1, E2, E4 がデザインパターンのイベントに割り当てられたとすると E1, E2, E4 の列が結果として得られるイベントの列となり, 出力は

E1 < E2 < E4

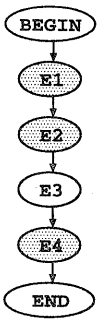

図 3: 適合性の確認の結果の例

という形で得られるとする.

図 2のイベントの中で, 結果として得られるイベントを網を書けて図示したものが, 図 3である.

バックトラック方を用いればすべての可能なイベントの割り当てに対応するイベントの列の集合が得られる.

6 例題

例題として, Observer パターンを用いて作られた設計の適合性確認について示す.

6.1 デザインパターンが持つ情報

オブザーバパターンの協調関係は, 図 4で定義され, その協調関係をイベントの半順序で表現すると, 図 5となる.

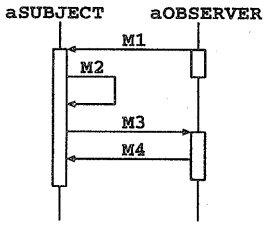

図 4: オブザーバパターンの協調関係 (インタラクションチャート)

5.1節に基づいて, デザインパターンを次のように表現した.

構造の情報

クラス: SUBJECT, OBSERVER

メッセージ: M1, M2, M3, M4

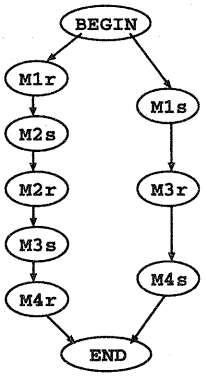

図 5: オブザーバパターンの協調関係 (イベントの半順序)

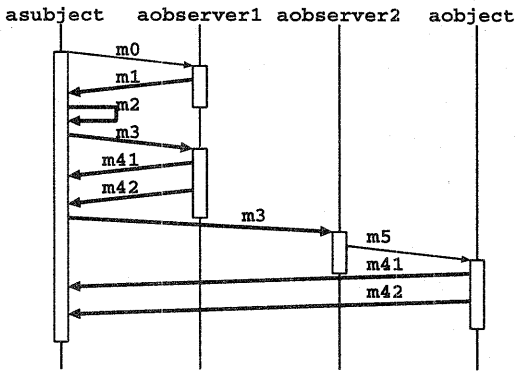

図 6: 比較する設計の協調関係 (インタラクションチャート)

協調関係の定義

送信イベント

M1s: <aOBSERVER, aSUBJECT, M1>,

M2s: <aSUBJECT, aSUBJECT, M2>,

M3s: <aSUBJECT, aOBSERVER, M3>,

M4s: <aOBSERVER, aSUBJECT, M4>

受信イベント

M1r: <M1s>,

M2r: <M2s>,

M3r: <M3s>,

M4r: <M4s>

イベントの半順序 (図 4)

M1r < M2s < M2r < M3s < M4r

M1s < M3r < M4s

メッセージの組の繰り返し

繰り返すメッセージ: M3, M4

一方, 図 6は比較する設計の協調関係を表現しているインタラクションチャートで, その協調関係をイベントの全順序で表現したものの一部が図 7である.

5.1節に基づいて, これを次のように表現した.

構造の情報

クラス: subject, observer, object

メッセージ: m0, m1, m2, m3, m4,...

協調関係の情報

送信イベント

m0s: <aobserver, asubject, m0>,

m1s: <aobserver, asubject, m1>,

m2s: <asubject, asubject, m2>,

.....

受信イベント

m0r: <m0s>,

m1r: <m1s>,

m2r: <m2s>,

.....

イベントの全順序 (図 6)

m0s < m0r < m1s < ... m42r

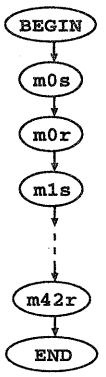

図 7: 比較する設計の協調関係 (イベントの全順序)

図4, 図5において, aSUBJECT は, デザインパターンで定義されているクラス SUBJECT のインスタンス, M1 はデザインパターンで定義されたメッセージ, M1r はメッセージ M1 の受信イベントを表すとする. 図6, 図7においても同様とする.

構造の対応は次のように取れているとする.

デザインパターン	設計
SUBJECT	subject
OBSERVER	observer, object
M1	m1
M2	m2
M3	m3
M4	m41, m42

また, デザインパターンの M3, M4 がメッセージの繰り返しを意図しているとする.

本稿で述べたようにイベントどうしを対応づけていくと, 図6で太線のメッセージに関するイベントがすべてデザインパターンのイベントに対応づけられることになり, 設計のその部分がデザインパターンと適合しているということができる.

7 議論

すでに述べたとおり, 本稿の提唱する方式のツールは協調関係への適合性の確認の支援を行なうものである. 本方式では作成したソフトウェアやモデルをなんらかの形で実行させ, その実行トレースに対して適合性を確認するのに用いる. つまり, テスト実行した限りにおいて適合しているかどうかは確認できるが, およそすべての実行トレースを確認できるわけではないので, 設計が本当に適合しているかどうかを確認することはできないことになる. これは, デザインパターンがインタラクションダイアグラムによる制約という形で与えられるため, ステートダイアグラム等を直接確認することが困難だからである.

Rapide [2] は, 特にシステムアーキテクチャのプロトタイピングのために設計されたオブジェクト指向言語である. Rapide にも自動的に適合性の確認を取るための枠組が用意されている. それは,

- 抽象度を合わせるために構造のマッピングを行なう
- 協調関係が参照するモデルの制約条件を満たすかチェックする

ことを特徴としている.

本稿の方法とは, 本稿の方法がメッセージを送信イベントと受信イベントに分けているのに対して, Rapide ではそれを分けていないところが大きく異なる. 送信から受信にかかる時間が無視できない場合や, 送信したメッセージがブロックされる可能性がある場合には, 本稿の方式が有利であると考える.

8 終りに

本稿では, デザインパターンへの適合性確認を支援する方式について議論した.

どのような場合にも適合性の確認を自動的に行うのは原理的に不可能であるが, 適用範囲を広げるための考察を今後行いたい.

例えば, 3.1節であげた以下の状況への対処方法を今後まとめる予定である.

状況 1-2

オブジェクトの数が決まっていない状況

状況 1-3

クラスの数が決まっていない状況

ところで 3.2節であげた状況は, $n = 1$ もしくは $m = 1$ の状況の対処しか検討しなかったが, これは検討しなかった状況においては適合性の確認を行なう意味がないと判断したからである.

今後, 現実問題に適用して実用性を検証したい.

参考文献

[1] Erich Gamma, 他: デザインパターン. ソフトバンク, 1995.

[2] David C. Luckham, 他: Specification and Analysis of System Architecture Using Rapide, IEEE transactions on software engineering, Vol. 21, No. 4, April 1995.

[3] 岸知二: アーキテクチャパターンに基づく設計手法の考察, SIGSE, Vol97, No.74, (1997).

[4] 野田夏子, 岸知二: パターンを用いたアーキテクチャ設計, 情報処理学会, OO'97, (1997).

[5] 山本理枝子, 他: 金融アプリケーション開発におけるパターン適用, 情報処理学会, OO'97, (1997).