

# カジュアルゲームにおけるゲーム要素の分解・整理を ArchiMate で表現する提案

北野不凡, 山本修一郎  
名古屋国際工科専門職大学

愛知県名古屋市中村区名駅 4-27-1

## Decomposition and organization of game elements in casual games with ArchiMate

Kitano Masaru, Shuichiro Yamamoto

IPUT in Nagoya  
4-27-1, Meieki, Nakamura-ku, Nagoya Aichi Japan

### 概要

前稿ゲームデザインにおける MDA フレームワークの ArchiMate 表現法の提案において、メカニクスデザインを ArchiMate を用いて表現する提案をおこなったが、今回それをさらに推し進め、ゲームにおける既存のゲームメカニクス要素を分解・整理し、その整理された要素を用いてメカニクスデザインを再構築する手法を提案する。

- (1) ゲーム全体を ArchiMate で階層的に表現する明確なルールを設ける。
- (2) カジュアルゲームを例に採り、ゲームをメカニクス（ルールや振る舞いなどプログラムやデータで表現する部分）とフレーバー（世界観、UI など視覚的デザイン部分）に分離する。本稿ではメカニクスのみを論ずる。
- (3) ゲームメカニクスに設置するメカニクス要素の目的・機能・効果を明らかにし、ArchiMate のパーツとして再利用可能にする。
- (4) パーツ化したコンポーネントを組み合わせる方式を採用することで、メカニクスデザインの設計と理解の容易化が実現できる。

### Abstract

In the previous article, we proposed a method of expressing mechanics design using ArchiMate. In this article, we will further advance this proposal by classifying and organizing existing game mechanics elements in a game, and propose a method of restructuring the mechanics design using these organized elements.

- (1) Establish clear rules to represent the entire game hierarchically in ArchiMate.
- (2) Taking a casual game as an example, separate the game into mechanics (rules, behavior, and other aspects expressed by programs and data) and flavor (worldview, UI, and other visual design aspects). In this paper, only the mechanics will be discussed.
- (3) Clarify the purpose, function, and effect of the game elements to be placed in the mechanics, and create ArchiMate components.
- (4) By combining the component parts, the mechanics design can be easily designed and understood.

## 1. はじめに

ゲーム中にはさまざまなルール（メカニクス）が内蔵されている。それらはプログラム上の手続きによって実現されているが、共通化できる物も多い。それらを、本稿ではメカニクス要素と呼び、共通化できるメカニクス要素の目的と機能、およびその効果を分類する。

現在は設計者の経験から、採用する要素を選択してゲームの設計が行われている。

この提案によって使用に適したメカニクス要素が明瞭になり、経験の浅い設計者も要素の選択、利用目的を的確に判断にすることができる。

またメカニクスデザインの表現方法に一般的規則はなく、設計者個人の人々の経験などによって定められるため第三者が設計内容を理解するには時間を要する。当研究によりメカニクスデザインの表現方法を一般化することで、このような弊害を軽減できると考える。

本稿の構成は以下の通りである。2節で関連研究を説明する。3節で ArchiMate によるゲームアーキテクチャ表現を提案する。4節で提案手法の適用例を示す。5節で考察を述べ、6節でまとめる。

## 2. 関連研究

以下では関連研究について説明する。

### 2.1 ゲームメカニクス

ゲームメカニクスはゲームにおける、ルール、ゲーム内経済、オブジェクトの振る舞いを示す用語である。ゲームメカニクスは後述する MDA フレームワークの枠組み内で、Robin Hunicke の研究のように、Aesthetics との関係で研究が進んでいる。本稿では様々なゲームが内包する共有可能な要素を整理、分類する。またメカニクスに則った特定の遊び方を Ludeme と表現する場合もあるが、日本では一般的な用語ではない。ゲーム個別の遊び方や共有可能なメカニクスの部品を総称して本稿ではメカニクス要素と表現している。

### 2.2 MDA フレームワーク

MDA (Mechanics, Dynamics, Aesthetics) フレームワークは Game Developers Conference, San Jose 2001-2004 の Game Design and Tuning Workshop の中で Robin Hunicke, Marc LeBlanc, Robert Zubek の 3 人によって発表された。

ユーザーから開発者までの広い視野でゲームを評価、調整する目的で作られたもので、直感的で理解しやすい。MDA の例を表 1 に示す。

表 1 MDA の例

Mechanics	Dynamics	Aesthetics
通常の闘争	正当な報酬	楽しい
知識の取得	正当な報酬	楽しい
知識の生成	戦術的勝利	楽しい・喜び
物語の開始と進行	積極的な進行 (関与)	楽しい

一方、Mechanics, Dynamics, Aesthetics の 3 つのレイヤーを表現する方法が定められておらず、意図したゲームを自然言語を用いて表現する場合、正確に記述することと、正確に理解することはやや困難である。

本稿では MDA フレームワークの曖昧さを図式化言語によって排除することを試みる。

### 2.3 ArchiMate

ArchiMate は、The Open Group が標準化する EA モデリング言語である [2]。ArchiMate は 2002 年から 2004 年までオランダの産官学連携プロジェクトで開発された。その後 The Open Group で EA モデリング言語として標準化されることになり、2009 年に ArchiMate 1.0 が公開された。最新版は 2019 年末に公開された ArchiMate 3.1 である [3]。The Open Group は ArchiMate の認定試験を実施しており、世界中で 2019 年までに約 1 万人を認定している。

TOGAF (The Open Group Architecture Framework) [4] の Architecture Development Method

(ADM) の全工程の成果物を ArchiMate で記述できる。

最も豊富な機能を持つ EA フレームワークが

TOGAF である。また、Groenewegen らが、ArchiMate モデルをレビューするためのゲームを提案している [5]。

### 2.4 オブジェクト指向プログラミング

従来のデザインパターンをゲームプログラミングに適用する事を研究した書籍がある。

Game Programming Patterns Robert Nystrom 著等だが、本稿はいわゆるデザインパターンとは直接の関係はない。本稿でいう「ゲームデザイン」とはゲーム全体の設計行為を表しており、デザインパターンの「デザイン」とは意味が異なる。

## 3. ArchiMate によるゲームアーキテクチャ表現

ゲームはプログラムやデータによる振る舞いを表現する部分とキャラクタやストーリーを表現する部分に大別される。

本研究ではこれをメカニクスとフレーバーと呼ぶ。本研究では特にメカニクスに焦点を当てこれを ArchiMate で表現しメカニクス設計の一般化を試みる。

### 3.1 ゲームにおけるレイヤー

前述したようにゲームアーキテクチャはメカニクスとフレーバーに大別される。本稿ではそれぞれをメカニクス層、フレーバー層と呼ぶ。

メカニクス層には以下の層を内包する。

コンセプト層

ArchiMate のビジネス層を用いて記述。何をやるゲームかを記述する。

メカニクス層

ArchiMate のビジネス層を用いて記述する。ゲームメカニクスを構成する詳細な要素や、定型

のメカニクスを本稿ではメカニクス要素と呼ぶ。メカニクス要素を追加する事でゲームメカニクス全体を記述する。

コンポーネント層

ArchiMate のアプリケーション層を用いて記述する。ゲームメカニクスを実現するコンポーネントの構成及び関係を記述する。

プレーヤー層

ゲームにおけるビジュアル部分やサウンド、ストーリー、UIなどを記述するが本稿では言及しない。

### 3.2 研究仮説

RQ1:ArchiMate でメカニクス要素を定義できる

RQ2:定義したメカニクス要素の組み合わせで、ゲームメカニクスを表現できる。

### 3.3 その他のゲームメカニクス

代表的なジャンルから基盤となるゲームメカニクスを定義した。この定義に従ってメカニクス要素を配置していく。(これらは互いに内包が可能) 今回の具体例は「エスケープ」のゲームメカニクスを用いて ArchiMate による表現を行う。

表2 ゲームメカニクス

名称	内容
アボイド	避けるのみ
キャッチ	リワードをキャッチ
タッチ	目標をタッチ (クリック)
エスケープ&キャッチ	避けつつ、リワードをキャッチ
エスケープ&アタック	避けつつ、敵を攻撃
エスケープ&キャッチ&アタック	避けつつ、敵を攻撃し、リワードをキャッチ
ナラティブ	ストーリー
ナラティブ&セレクト	ストーリー+選択肢
ナラティブ&セレクト&バトル	ストーリー+選択肢+バトル
オープンワールド	ワールド内の行動が自由であるもの。
ダンジョン	ダンジョン脱出
ダンジョン RPG	ダンジョン脱出+バトル
ターン制バトル	敵味方交代するバトル
リアルタイムバトル	敵味方同時のバトル
固定ガンシューティング	プレイヤー固定のガンシューティング
レール・ガンシューティング	設定された軌道を移動するガンシューティング
ターン制シミュレーション	ターン制で複数の自キャラを移動・バトル

リアルタイムシミュレーション	リアルタイムに複数の自キャラを移動・バトル
----------------	-----------------------

### 3.4 隕石避けゲームのコンセプト層

図1は隕石避けゲームのコンセプト層である。ここではゲーム内容を簡単に記述し、目的となるプレイヤーのエステティクスに結びつける。

### 3.5 ゲームを表示するカメラ要素

ゲームをコンピュータ画面に表示する場合のカメラを定義する要素である。

表3 カメラの定義

属性 カメラ

表3 カメラ属性の値

値	種類	内容
SCAM	固定カメラ	ゲーム画面の特定位置に固定する
FCAM	追従カメラ	指定要素を追従する
DCAM	移動カメラ	カメラがゲーム画面中を移動する

このとき全てのカメラは以下のような属性と値を持つ。

属性 視点

TPV: トップビュー上からの視点

SDV: サイドビュー横からの視点

QTV: クォータービュー斜め上からの俯瞰視点

TDV: 第三者視点

FTV: 一人称視点

属性 表現方法

2D: 2D 表現を行う

3DI: アイソメトリックな 3D 表現を行う

3DP: パースペクティブな 3D 表現を行う

カメラ要素による種類や属性を設定することで多くのゲームジャンルのカバーが可能となる。

### 3.6 ゲーム内のメカニクス要素

ゲーム画面内に表示されるメカニクス要素を定義する。

表4 描画メカニクス要素

名称	内容
プレイヤー	プレイヤー操作キャラ
パートナー	味方となるパートナーキャラ
NPC	中立的なキャラ
敵バリア	バトルの対象
バリア	ダメージを与えないバリア (壁など)
ダメージバリア	ダメージを与えるバリア (敵弾など)
動くバリア	ちいさな喜び
スイッチ	何かを切り換える
リワードS	小さな喜び
リワードM	中程度の喜び
リワードL	大きな喜び

リワードF	最大の喜び
固定情報 (文字)	固定文字列
可変情報 (文字)	可変文字列
固定情報 (グラフィック)	固定グラフィック
可変情報 (グラフィック)	可変グラフィック

これらの表示される要素は以下のような属性と値を持つ。

属性 移動方法

SMV:プレイヤー操作による移動

DMV:データによる移動

CMV:計算式による移動 (含ゲーム AI)

NMV:位置固定

属性 重力

属性 摩擦

属性 弾性

属性 衝突判定範囲

### 3.7 ゲーム内経済要素

ゲーム内経済要素は得点や HP、制限時間などゲーム内で変動する要素として用いる。

実体は変数となる。

表 5 ゲーム内経済要素

名称	内容
加算要素	スコア、生存タイムなど
減算要素	残り時間、武器数など
制限付き加算要素	所持品数など上限がある
制限付き減算要素	所持品数など下限がある
固定要素	変化しない要素
フラグ要素	フラグとして用いる

## 4. 具体例

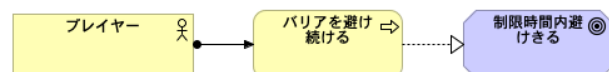
ゲームプログラミングを学ぶ最初に例として用いられる「隕石よけゲーム」を例に取る。

これは制限時間内に HP の有る限り飛来する隕石を避け続けられればステージクリアというゲームメカニクスを持っている。(ステージをクリアする毎に難易度が上がる)

### 4.1 コンセプト層

このゲームの概略をコンセプト層で表現した。

図 1



### 4.2 基盤状態

この状態ではただバリアを避けるだけで終了条件もない。つまり、まだゲームとしての条件を満たしていない。

勝敗の条件が確定しており、プレイ時その勝敗が不確定であることがゲームであることの条件として必要である。

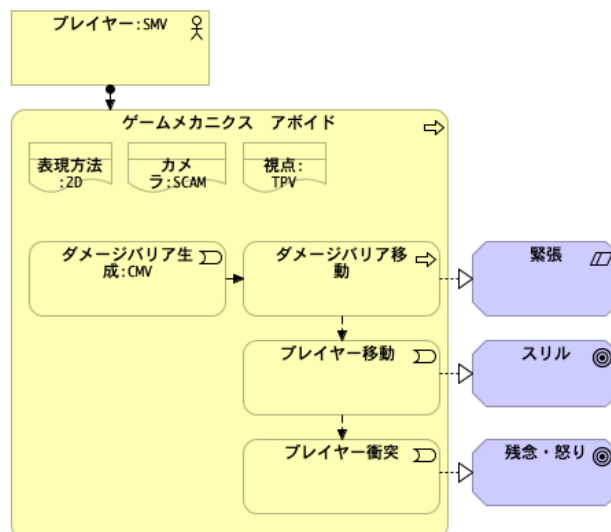


図 2 メカニクス層の初期段階

視点、カメラ、表現方法を設定することで画面の見え方が設定できる。

さらに登場するオブジェクトである隕石とプレイヤーに移動方法を設定している。

また、それらが産み出すエスティクスを表現している。

### 4.3 メカニクス要素の追加

4.2 の隕石が落ちてきてプレイヤーが避けるだけの状態に減算要素 (制限時間) とステージクリアのイベントを追加した。これによって進んだステージ数を競うゲームメカニクスが表現された。この場合、難易度はステージ毎に上昇する事とする。

このようにゲーム開始からゲームクリアまでの挙動を表現することができる。

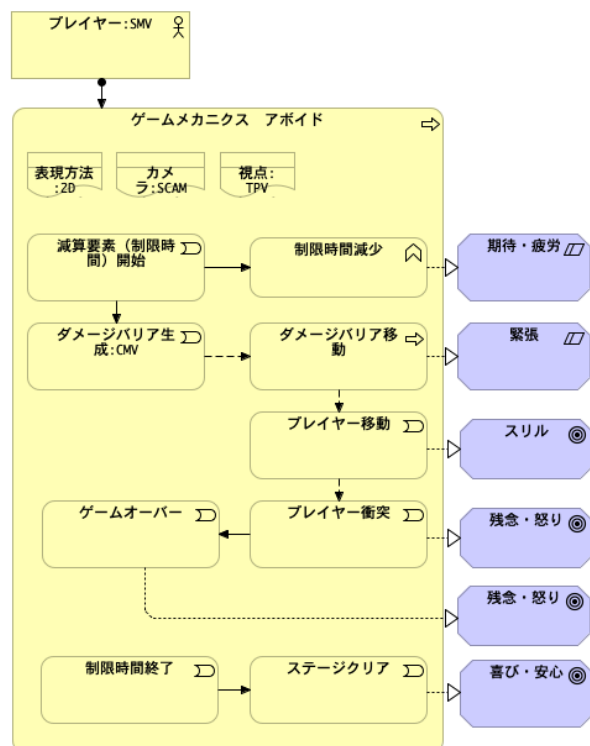


図 3 メカニクス要素を加えた状態

#### 4.4 メカニクス要素組替えによるゲーム性の変化

図4は図3からメカニクス要素を組み替えた状態である。

加算要素に衝突回数と、生存時間を加えている。衝突回数によってゲームオーバーとなり、そのゲームオーバーまでの生存時間を競うゲームに変化している。

基本的な「避ける」という基盤となるゲームメカニクスを変化させずにゲーム性の変更が可能である。このように ArchiMate を用いメカニクス要素を組み替えることで変化するゲーム性を表現することができている。

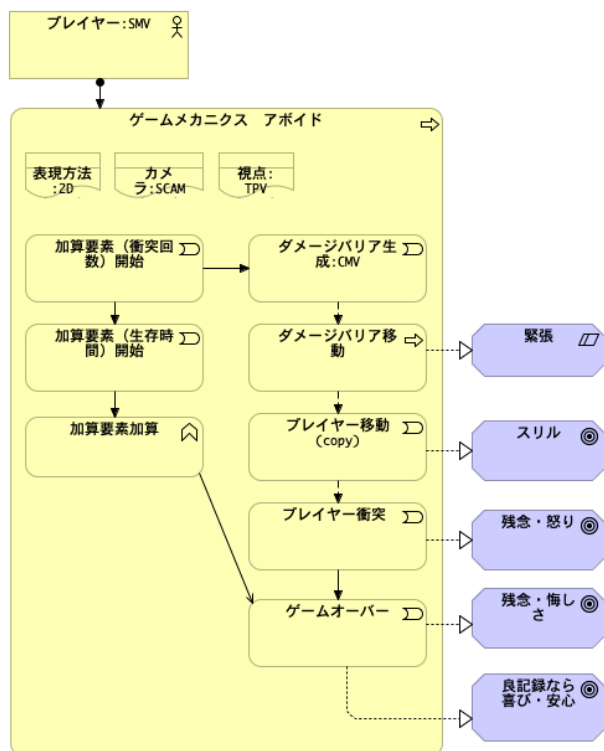


図4 変化したゲーム性

#### 4.4 コンポーネント層

図5にコンポーネント層を示す。本稿ではメカニクス層での表現を主眼とするため、コンポーネント層について詳細は言及しない。実装に当たってのモジュールとその機能が概観できるものとなっている。

### 5. 考察

#### 5.1 仮説の検証

本稿では、ArchiMate でゲームメカニクスを表現する手法を提案し、ゲーム事例に適用することにより、記述言語によってゲームメカニクス全体を表現出来ることを確認した。

【RQ1の検証】具体例からゲームに必要なメカニクス要素を記述できていることから RQ1の要件を満たしていると考えられる。

【RQ2の検証】具体例からメカニクス要素をくみ換えることでゲームメカニクスの変更に対応できることで RQ2の要件を満たしたと考える。

#### 5.2 留意点

表2のゲームメカニクス要素はカジュアルゲームから代表的なものを抜き出したが、全体に未整理である。ジャンルとゲームメカニクスが混同された状態にあり、複合的なゲームメカニクスを持ったゲームのゲームメカニクスをいかに整理・表現するかが今後の課題である。また、パズルゲームなどのメカニクスについては全く含まれていない。パズルのメカニクスは多種多様であり、分解・整理には多くの時間が必要と考えられる。

#### 5.3 限界

##### (1) 表現可能なゲームメカニクスの限界

ゲーム中におけるバトル要素などが、現在未定義であり、表現できないゲームメカニクスがある。今後この要素定義を充実させていく。

##### (2) 実験の限界

本稿では ArchiMate でゲームメカニクスを表現する方法について考察した。

今後 ArchiMate によるゲームメカニクス表現の有効性を確認するために、下記のようなパターンの実験が必要と考える。

- ・同様のゲームメカニクスを複数の被験者で表現し、同様の図になるかの確認。

表現方法のルールが適切であればメカニクス層の図やオブジェクトは同様になるはずである。

- ・元のゲームを知らない閲覧者の理解度確認。ゲーム内容を知らない被験者が ArchiMate による図からゲームメカニクスを読み取れるかの確認実験。

### 6. まとめ

ArchiMate3.0の仕様を用いてメカニクス要素を定義し、それに則ってゲームメカニクスを表現する提案を行った。

RQ1, RQ2 を満たす事例があることは立証した。

しかし、すべてのゲームについては、RQ1, RQ2 が立証できるかどうかについては、今後の課題となる。本稿をさらに追求することで以下のようなメリットがあると考えられる。

- (1) ゲームの企画、設計時の効率向上が期待できる。特にメカニクス要素を最初から用意していることでゲーム企画の経験の浅い者でも既存のメカニクス要素を利用できる。

- (2) ゲームの企画書、仕様書の第三者への理解容易性が高められる。

- (3) 作図することによって内容の誤りや欠如の発見が容易になる。

以上のような利点があることから、さらに深くこの研究を進め、メカニクス要素の拡張、プレーヤー層の整理とメカニクス層との連結などを研究したい。

また、これらの図から Unity のビジュアルプログラミングを用いて実装を容易にする事も今後の課題として考えられる。



## 参考文献

- [1] Robin Hunicke, Marc LeBlanc, Robert Zubek, MDA: A Formal Approach to Game Design and Game Research, 2004
- [2] 山本修一郎, 現代エンタープライズ・アーキテクチャ概論 - ArchiMate入門, デザインエッグ社, 2016
- [3] The Open Group, TOGAF v9.2, C182, 2018
- [4] The Open Group, ArchiMate® 3.1. Specification. C197.2019
- [5] Game Programming Patterns Robert Nystrom 著 武舎広幸監訳

阿部和也+上西昌弘訳 インプレス社 2015

[6] 大野功二, 2Dゲームをおもしろくする技術 SB Creative 2017

[7] Stephen Trinh 2020, The 4Elements of GameMechanics

[8] Chuah Kee Man 2021, *Game Elements, Components, Mechanics and Dynamics: What are they?*

[9] ゲームデザインにおける MDA フレームワークの ArchiMate 表現法の提案

[https://www.jstage.jst.go.jp/article/jsaisigtwo/2022/KS-N-030/2022\\_08/\\_article/-char/ja/#author-information-wrap](https://www.jstage.jst.go.jp/article/jsaisigtwo/2022/KS-N-030/2022_08/_article/-char/ja/#author-information-wrap)

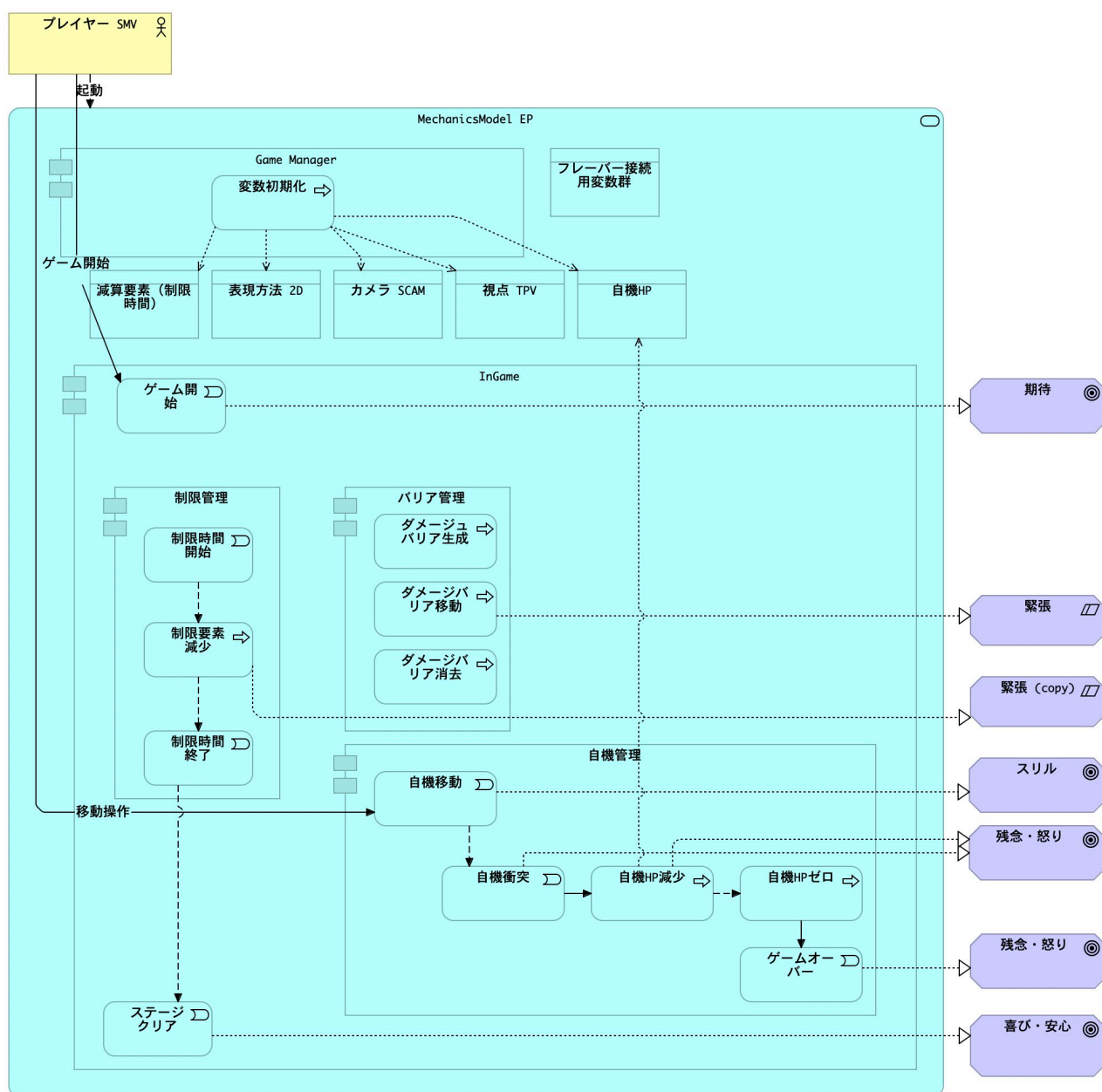


図 5 具体例図 3 の隕石ゲームのコンポーネント層