論　文

# コンピュータプレイヤーのプログラム作成を通して競い合うゲームプラットフォームの開発を支援するフレームワーク

坂本　一憲[†]　　大橋　昭[†]　　鷲崎　弘宜[††]　　深澤　良彰[††]

A Framework for Game Software Which Users Play through Artificial Intelligence Programming

Kazunori SAKAMOTO[†], Akira OHASHI[†], Hironori WASHIZAKI[††], and Yoshiaki FUKAZAWA[††]

**あらまし**　近年，プログラミングコンテスト，特にゲームソフトウェア上で自律的に動作するプレイヤーのプログラムを作成して競い合う形式のコンテストが，教育向けの枠組みとして注目を浴びている [1]．一方，コンテストで利用されるゲームプラットフォームは，一般的なゲームソフトウェアやエンタープライズシステムと異なり，開発手法や技術的な知見が体系化されていない．本論文では，過去の開発経験からゲームプラットフォームの要求を分析した結果をもとに，開発における知見を設計原則としてまとめる．更に，ゲームプラットフォームの品質を保証して，開発コストを低減させるため，設計原則を踏まえたアーキテクチャと汎用的な共通処理を提供するフレームワークを提案する．その上で，提案フレームワークを利用するゲームプラットフォームとコンテストの事例報告をもとに，提案フレームワークを利用するゲームプラットフォームが分析した要求を満たすかどうかという定性的評価と，ソフトウェアメトリクスや参加者のアンケートによる定量的評価の両方から提案フレームワークの有用性を示す．

**キーワード**　フレームワーク，要求分析，設計原則，プログラミングコンテスト，ゲームプラットフォーム

## 1. まえがき

近年，ゲームソフトウェア上で自律的に動作するプレイヤーのプログラム (以降，AI プログラム) を作成して競い合うコンテスト (以降，コンテスト) が活発に開催されている．コンテストは，プログラミングの面白さを伝えて学習機会を創出するため，学習機材として教育界から注目を浴びている [1]．

コンテストでは，AI プログラムを動かす基盤となるゲームソフトウェア (以降，ゲームプラットフォーム) が必要となる．例えば，IBM 社が開発した Robocode [2] は，参加者が作成したロボットを制御する AI プログラム同士を対戦させるゲームプラットフォームである．

コンテストの例として，国際大学対抗プログラミングコンテスト (International Collegiate Programming Contest;ICPC) [3] に併設される JavaChallenge が挙げられる．JavaChallenge は，Java 言語を用いて AI プログラムを作成して競い合うコンテストである．

コンテストでは，参加者の公平を期すため，コンテスト毎に新しいゲームプラットフォームを開発する．しかし，コンテストの普及にも関わらず，ゲームプラットフォームの開発方法論が体系化されていない．そのため，コンテストの成功を目的とするゲームプラットフォームの開発者にとって，開発コストを抑えながら品質を保証することが難しく，それに起因するゲームプラットフォームの欠陥やコンテストの不備から，プログラミングの面白さを伝えられないケースがある．

そこで，本論文では，ソフトウェア工学で利用される手法を取り入れて，過去の開発経験からゲームプラットフォームの要求を分析する．分析した要求を満たすために，設計において採用すべき指針を設計原則としてまとめる．設計原則に基づいたゲームプラットフォームを開発することで，分析した要求を満たせる．更に，本論文では，設計原則に基づくフレーム

[†] 早稲田大学大学院基幹理工学研究科情報理工学専攻
Dept. of Computer Science, Waseda University
[††] 早稲田大学基幹理工学部情報理工学科
Dept. of Computer Science, Waseda University

ワーク，題して Game AI Arena(以降，GAIA) を提案する．GAIA は設計原則に基づいたアーキテクチャを提供することで，ゲームプラットフォームの品質を保証する．また，汎用的な共通処理を提供することで，ゲームプラットフォームの開発コストを低減する．なお，GAIA のソースコードは Google Code にてオープンソースとして公開している [4].

本論文の構成は以下のとおりである．2 節で対象とするゲームプラットフォームとコンテストについて説明する．3 節で GAIA を利用しないゲームプラットフォームの開発経験を説明する．4 節ではゲームプラットフォームの要求を分析して，5 節で設計原則を提案する．6 節にて GAIA を提案して，7 節で GAIA を利用した事例報告をもとに GAIA を評価する．8 節にて関連研究を述べ，9 節にて本論文をまとめる．

## 2. 対象とするゲームプラットフォーム

GAIA が対象とするゲームプラットフォームとコンテストについて説明する．開発者はゲームプラットフォームを開発してコンテストを主催する人，参加者は AI プログラムを作成して競技する人，視聴者は参加者以外で AI プログラムの戦いを観戦する人を示す．

図 1 で GAIA を利用しないゲームプラットフォームの概観，図 2 で GAIA とそれを利用するゲームプラットフォームの概観を示す．従来では，複数のゲームプラットフォーム間に共通の設計や処理があるにも関わらず，それぞれが独自に開発されており，開発コストの低減や品質保証について問題があった．そこで，我々はゲームプラットフォームを開発するためのフレームワークとして GAIA を提案して，上述の問題を解決する．


図 1　GAIA を利用しないゲームプラットフォーム概観
Fig. 1　An overview of game platform without GAIA

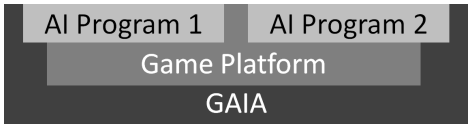



図 2　GAIA を利用するゲームプラットフォーム概観
Fig. 2　An overview of game platform with GAIA

コンテストでは，AI プログラムを組み込めるゲームプラットフォームを参加者に配布する．参加者は与えられた期間内で，ゲームプラットフォームが提供する API を用いて AI プログラムを作成する．AI プログラムの作成期間は 3 時間から 1ヶ月間など様々である．参加者は作成期間内に AI プログラムを開発者に送付する．開発者は受け取った AI プログラム同士をゲームプラットフォーム上で対戦させて参加者を競い合わせる．開発者は Web 上，若しくは，会場に人を集めて，対戦結果のゲーム内容を放映する．

AI プログラムの実行環境は，準備の容易さから一般的なデスクトップコンピュータが利用される．我々が主催した全てのコンテストでは，CPU：Intel Core 2 Duo E8500 3.16GHz，メモリ：4GB，OS：Windows 7 のデスクトップコンピュータ 1 台で，AI プログラムを実行した．GAIA は同等の環境を想定する．

ゲームプラットフォームで実施するゲームとして，リアルタイム制とターン制のゲームが考えられる．ターン制のゲームとは，将棋や囲碁のように手番のプレイヤーのみが行動でき，一方，リアルタイム制のゲームとは，トランプゲームのスピードのようにどのプレイヤーも任意のタイミングで行動できるゲームである．コンテストでは，ゲームのルールを複雑にして参加者の敷居を上げないように，リアルタイム制ではなく，ターン制のゲームであることが多い．そのため，GAIA ではターン制のゲームを対象とする．

## 3. ゲームプラットフォームの開発経験

GAIA を利用しないゲームプラットフォームの開発経験とコンテストの開催経験を説明する．

我々は 2009 と 2010 年度に開催された ICPC の併設コンテストである JavaChallenge(以降，それぞれ JC2009, JC2010) [5] [6] のゲームプラットフォームを開発してコンテストを主催した．各ゲームプラットフォームは Google Code にてオープンソースソフトウェアとして公開している [7] [8].

両者の内容は類似するため，以下で JC2010 の事例を報告する．1 チーム 3 人の構成で 45 チームが参加して，4 時間で Java 言語の AI プログラムを作成した．

ゲームプラットフォームが実施するゲームは，2 プレイヤーで対戦する 2 次元座標上の陣取りゲームである．プレイヤーはキャラクタを移動して，魔方陣を配置させることで所有権を得る．プレイヤーが所有するマスは図 3 のように色付く．所有するマスで大小 2 種

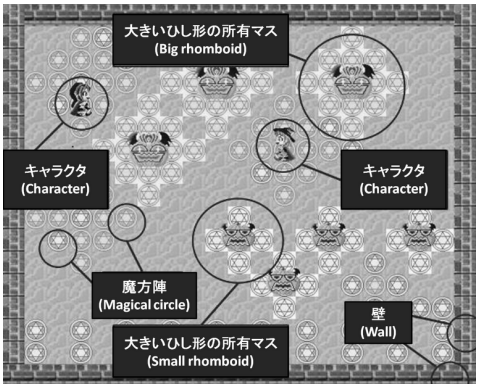



図 3　2010 年度 JavaChallenge のゲーム画面
Fig. 3　A game screen of JC2010

類のひし形を形成すると，該当するマスのグラフィックが変化する．大きいひし形，小さいひし形，それ以外の所有マスそれぞれ 1 つにつき，50 点，20 点，1 点を得る．また，相手が所有するマスを奪えるが，ひし形のマスを奪う際は時間がかかる．一定時間の経過後に，最も得点の高いプレイヤーの勝利となる．

JC2009 と JC2010 は一定の成功を収めたが，納期の尊守と品質保証で大きな問題が発生した．具体的には，JC2009 ではソースコードが複雑になり，見積もり以上に開発工数が必要となり，プロジェクトの途中で開発メンバーを追加することが必要になった．しかし，短絡的に開発メンバーを増やしたため，ゲームプラットフォームの品質低下を招き，結果的にゲームプラットフォームが提供する API に欠陥が残留した．また，参加者がコンテストの試合を視聴する際に，ゲームの進行速度にばらつきが生じて，時折ゲーム画面が一切変化しなくなり，ゲームが止まっているかのように見える問題が生じた．

我々はこうした問題に対処するために，要求分析から開発者が注力すべき点を調査して，設計原則としてまとめた．しかし，JC2009 の開催経験から設計原則が得られていたにも関わらず，JC2010 にて問題が発生した．JC2009 では再利用性を考慮せずに開発を行ったため，JC2009 で開発したゲームプラットフォームの再利用が上手く行かず，JC2010 では設計原則をもとに最初からゲームプラットフォームの設計，及び，実装を行う必要があった．その上，JC2010 ではゲームプラットフォームの規模が拡大しており，開発工数と欠陥の数が増大してしまい，欠陥の修正が間に合わず，予定したコンテストの開始時刻を遅らせるという納期を尊守できない事態となった．

したがって，本論文では GAIA を提案することで，設計原則という知見の提供のみならず，再利用可能なフレームワークを提供して，設計原則に基づいた高品質なゲームプラットフォームを少ない開発工数で開発できるように支援する．

## 4. ゲームプラットフォームの要求分析

JC2009 と JC2010 の開催経験から，ISO9126 で規定される品質特性に基づいて，ゲームプラットフォームの要求を分析する．以降，要求を**太字**で示す．なお，以降の小節には，満たすべき品質特性について,「品質特性：品質副特性」という節名を与える．

### 4.1 移植性：環境適応性

ゲームプラットフォームを用いてコンテストを開催した場合，参加者数はコンテスト成功の是非に大きく影響する．そのため，多くの参加者が参加できるようなゲームプラットフォームが必要である．そこで，ゲームプラットフォームは以下の性質を持つべきである．

- **プラットフォーム非依存**：OS などの特定のプラットフォームに依存しない．例えば，ゲームプラットフォームは Windows と Linux 両方の OS で動作する．
- **多言語対応**：様々なプログラミング言語で作成した AI プログラムを実行する．例えば，ゲームプラットフォームは Java，Ruby，Python 言語で作成したいずれの AI プログラムも実行できるようにする．

### 4.2 機能性：セキュリティ

ゲームプラットフォームは AI プログラムがルールに従って動作するように制御する．AI プログラムがルールを守ることで，ゲームにおける公平性が担保され，ゲームの面白さが生じる．そこで，ゲームプラットフォームは以下のような性質を持つべきである．

- **書き換え不可**：開発者が想定する方法以外によるゲームプラットフォームの内部状態の書き換えを禁止する．例えば，Java 言語においてダウンキャストによる情報隠蔽に反した内部状態の書き換えを禁止する．
- **AI プログラム間の独立性**：他の AI プログラムの処理を妨害する行為を禁止する．例えば，他の AI プログラムが計算している際に不必要な計算を行い，計算時間を奪うことを禁止する．

### 4.3 信頼性：障害許容性

コンテストを円滑に実施するため，ゲームプラットフォームは障害に対処する必要がある．しかし，参加者が作成した AI プログラムを組み込むため，ゲームプ

ラットフォームの設計段階では想定できない障害が発生する可能性がある．そこで，ゲームプラットフォームは以下のような性質を持つべきである．

- **AI プログラムからの独立**：AI プログラムの障害の影響を受けない．例えば，AI プログラムが無限ループに陥ったり例外を発生させても正常に動作する．

### 4.4 効率性：時間効率性

コンテストでは，参加者が AI プログラムを作成して競い合うだけではなく，視聴者がゲームを観戦することも考慮する必要がある．AI プログラムの計算時間が非常に長くなることもあるため，観戦に適さない速度でゲームが進行することがある．そこで，ゲームプラットフォームは以下のような性質を持つべきである．

- **視聴に向いたゲーム速度**：視聴者がゲームを楽しめるように，適切な速度でゲームを進行させる．例えば，視聴者を退屈させないように，計算時間によらずにゲームの進行速度を変更できるようにする．

### 4.5 使用性：理解性と習得性と魅力性

コンテストでは，参加者と視聴者を楽しませるために，AI プログラムを作成するために使用するゲームプラットフォームの API の使いやすさや，ゲームそのものの魅力が重要である．そこで，ゲームプラットフォームは以下のような性質を持つべきである．

- **理解が容易な API**：AI プログラムの計算結果の出力方法が分かりやすい API を提供する．
- **習得が容易な API**：ゲームプラットフォームが公開する情報を取得する方法を覚えやすい API を提供する．
- **魅力的なゲーム**：集客しやすいように，ゲーム画面が美しく魅力的なゲームを提供する．

### 4.6 保守性：解析性と変更性

コンテストでは，参加者が AI プログラムを作成する期間は長くとも 1ヶ月程度，また，期間が短いコンテストでは 3 時間程度であり，欠陥が発見された際や参加者から要望を受けた際に，迅速にソフトウェアを修正する必要がある．そこで，ゲームプラットフォームは以下のような性質を持つべきである．

- **解析が容易なコード**：ゲームプラットフォームのソースコードが解析しやすい．例えば，欠陥が発生した際に，迅速に欠陥の原因を発見できるようにする．
- **変更が容易なコード**：ゲームプラットフォームのソースコードが変更しやすい．例えば，仕様変更が発生した際に，迅速に修正できるようにする．

## 5. ゲームプラットフォームの設計原則

前節で分析した要求を満たすように，既知のデザインパターンや設計における知見を参考に，ゲームプラットフォームの設計原則を提案する．以降，設計原則をかぎ括弧で括って示す．

### 5.1 移植性：プラットフォーム非依存な中間言語

ゲームプラットフォームのソースコードを公開すると，ゲームの状態を保存するフィールド変数などの解析が容易になり，AI プログラムからゲームの状態を書き換えやすくなってしまう．このような不正を防ぐためにはリフレクションの禁止や後述する Immutable パターンの導入も必要であるが，ゲームプラットフォームの解析の敷居を上げるために，ゲームプラットフォームのソースコードは公開しないほうが良い．

しかし，機械語のバイナリ形式は実行できる環境が制限される上，環境毎にバイナリ形式を用意する手間もかかる．そこで，できる限り多くの参加者を集めるために，プラットフォーム非依存な中間言語のバイナリ形式の提供が必要である．例えば，Java のバイナリ形式でゲームプラットフォームを配布することで，Java の実行環境が存在する環境であれば，どこでもゲームプラットフォームを実行できる．

以上から,「プラットフォーム非依存な中間言語」は,**プラットフォーム非依存**を満たす.

### 5.2 移植性：言語処理系の内蔵

参加者の得意な言語は様々であり，多くの参加者を集めるため，AI プログラムを様々な言語で作成できると良い．そこで，ゲームプラットフォームは様々な言語処理系を内蔵する．例えば，ゲームプラットフォームを Java 言語で開発した場合，JRuby という言語処理系をゲームプラットフォームに内蔵することで，Java 言語のみならず Ruby 言語で作成された AI プログラムも実行可能とする．

以上から,「言語処理系の内蔵」は，**多言語対応**を満たす.

### 5.3 機能性：Immutable パターン

開発者が想定する方法以外で，AI プログラムがゲームプラットフォームの内部状態を操作できないように，内部状態の書き換えを禁止する必要がある．そこで，必要に応じてクラスに書き換え操作を定義しない Immutable パターン [9] を導入すべきである．Immutable パターンとは，作成後にその状態を変えることができないイミュータブルなオブジェクトを設計す

るためのデザインパターンである．

以上から,「Immutable パターン」は，内部状態の不正な操作の禁止を目的として，**書き換え不可**を満たすように働きかける 1 因子となる．

### 5.4 機能性：時間独立な AI プログラムの実行

AI プログラムを同じマシンで同時並行に実行すると，ある AI プログラムが大きな CPU 負荷をかけた場合，その他の AI プログラムの処理時間の増加を招き，計算時間を奪うような妨害行為が可能になる．各 AI プログラムを異なるマシンで実行すれば回避できるが，実行環境の要求を難しくして，移植性を低下させてしまう．そこで，二つ以上の AI プログラムが同時に動作しないように，ある AI プログラムが終了してから次の AI プログラムを順番に実行することで，AI プログラムを時間独立に実行して，妨害行為を禁止する．

以上から,「時間独立な AI プログラムの実行」は，**AI プログラム間の独立性**を満たすように働きかける 1 因子となる．

### 5.5 信頼性：AI プログラムの計算時間の制限

参加者の AI プログラムを開発者が修正できないため，AI プログラムの欠陥を取り除けない．そのため，ゲームプラットフォームが以下の AI プログラムの障害に対処する必要がある．AI プログラムが例外を発生させる場合は，プログラミング言語の例外機能を用いて対処する．無限ループなどでゲームプラットフォームへ制御を返さない場合は，スレッドを用いて AI プログラムを実行しておき，スレッドを停止することで対処する．なお，AI プログラムの計算処理を対戦結果が得られる現実的な計算時間内に抑える必要がある．そのため，AI プログラムが計算時間を超えた場合は，スレッドを停止させることで時間制限を設ける．

以上から,「AI プログラムの計算時間の制限」は，**AI プログラム間の独立性**を満たすように働きかける 1 因子となる．

### 5.6 効率性：ゲーム内容の再現

コンテストの参加者と視聴者の両方がコンテストを楽しめるように，視聴に適した速度でゲームを進行させる必要がある．しかし，例えば，1 フレームが 0.01 秒程度のゲームにおいて，1 フレームの AI プログラムの計算時間が 1 秒程度必要になり，人が快適にゲームを視聴するには計算時間が長すぎる．そこで，AI プログラムの計算結果を記憶して，ゲーム内容を再現する機能を導入する．つまり，あらかじめ AI プログラムの計算を行なっておき，閲覧するときにリプレイのように表示させる．これにより，人が快適にゲームを視聴できる速度でゲーム内容を再現できる．ただし，AI プログラムを動かしてゲーム内容をリアルタイムに視聴する場合は，この方法は適用できない．

以上から,「ゲーム内容の再現」は，**視聴に向いたゲーム速度**を満たすように働きかける 1 因子となる．

### 5.7 使用性：入出力の口の限定

AI プログラムは制御の反転 (Inversion of control;IoC) [10] によりゲームプラットフォームから呼び出されて実行される．

ゲームプラットフォームが AI プログラムを呼び出す際，AI プログラムに公開する情報を引数として渡して，AI プログラムの計算結果を戻り値として受け取る．その際，入力と出力の口を増やすと，情報の取得方法や結果の出力方法が複雑になり，API の理解が容易でなくなるため使用性が低下する．そこで，AI プログラムに公開する情報を 1 つのクラスにまとめて引数で渡し，計算結果を 1 つの戻り値で返すことで，API を簡潔にして理解性を向上させる．

また，近年は IDE の進化により，コード補完機能を用いて API を習得できる [11]．コード補完機能を用いると，任意のクラスが持つメソッドやフィールドを簡単に把握できるため，1 つの引数のみで入力の口を限定することで，ゲームプラットフォームから得られる全ての情報を容易に把握できるようにする．

以上から,「入出力の口の限定」は，**理解が容易な API** と**習得が容易な API** を満たすように働きかける 1 因子となる．

### 5.8 使用性：メインループと FPS 制御

ゲームの内容を描画する際，描画間隔がばらつくと画面がちらつき，魅力性が低下する．そのため，一定間隔で描画できるようにメインループを導入する．メインループとは，ゲーム実行中にループし続けており，1 回のループで 1 フレームの処理を行う仕組みである．フレームとはゲーム内の単位時間に該当して，1 フレーム分のゲームの進行，ゲーム内容の描画処理などを行う．なお，1 秒間に行う画面の更新回数，すなわちフレーム数を FPS(Frame Per Second) と呼び，FPS が示す間隔で画面を描画することを FPS 制御と呼ぶ．FPS 制御では，1 フレームの処理を行った際に，処理時間が FPS で示された間隔に満たない場合はその分だけ待機する．一方，超えた場合は 1 フレームの処理を簡略化して処理時間を短縮する．これをフレー

ムスキップと呼び，FPS 制御で導入する．

以上から,「メインループと FPS 制御」は，**魅力的なゲーム**を満たすために必要な 1 因子である．

### 5.9 保守性：シーンによるフレーム処理の構造化

1 フレームの処理はゲームプラットフォームの内部状態によって変化する．例えば，ゲームのタイトル画面を表示する場合と，AI プログラムの対戦の様子を表示する場合では，1 フレーム分のゲームの進行処理と画面の描画処理が大きく異なる．したがって，これらの処理を切り替えられるように構造化する必要がある．

フレーム処理を構造化させる手法としてタスクリストが存在する．タスクリストとは，1 フレーム分の処理を複数のオブジェクト (これをタスクと呼ぶ) で表現して，タスクに優先度を与えて，優先度順に複数のタスクを実行する手法である．タスクリストは優先度付きの Command パターン [12] と捉えられる．例えば，1 フレーム分のゲームの進行，ゲーム画面の描画という 2 つの処理を構造化する場合，1 フレームの処理を 2 つのタスクで表現する．しかし，ゲームの開発経験がないような不慣れな設計者では，不適切にタスクが設計されて，かえって保守性が低下する．このことは，タスクがゲームプラットフォームのドメインにおいて十分に詳細化されておらず，粒度に振れ幅があるためである．

そこで，ゲームプラットフォームのドメインにおいてより十分に詳細化された概念としてシーンを導入する．シーンとは，例えば，タイトルシーンや AI プログラム同士の対戦シーンなど，画面に表示するゲーム内容によってフレーム処理を構造化する．シーンは 1 フレーム分のゲームの進行処理と，ゲーム内容の描画処理を別のメソッドで表現するため，これらを同じタスクとして扱うタスクリストより役割が明確である．

以上から,「シーンによるフレーム処理の構造化」は，**解析が容易なコード**と**変更が容易なコード**を満たすように働きかける 1 因子となる．

### 5.10 保守性：グローバル変数導入の抑制

グローバル変数は保守性における解析性と変更性を低下させる要因である [13]．しかし，ゲームプラットフォームにおいて，内部状態全体を参照できるグローバル変数が必要とされる．ルールを表現するパラメータなどの情報が，ゲームの処理を行う様々な箇所で必要になるためである．そこで，シーンが持つメソッドに対して，内部状態全体を参照できるような値を引数で渡すことで，グローバル変数の導入を抑制する．

以上から,「グローバル変数導入の抑制」は，**解析が容易なコード**と**変更が容易なコード**を満たすように働きかける 1 因子となる．

## 6. GAIA の提案

本節では，前節の設計原則に基づいて GAIA を提案する．図 4 と図 5 で，品質特性と品質副特性，要求，設計原則，GAIA が提供する機能の対応関係を示す．以降，設計原則を満たす GAIA が提供する機能に下線を引いて示す．

図 6 で GAIA のアーキテクチャを示す．GAIA，ゲームプラットフォーム，AI プログラムの三層構造によるレイヤーアーキテクチャになっている．AI プログラムはゲームプラットフォームが提供する API を利用して計算処理を行う．ゲームプラットフォームは GAIA を利用して，AI プログラムに提供する API とゲーム特有の処理を実装する．

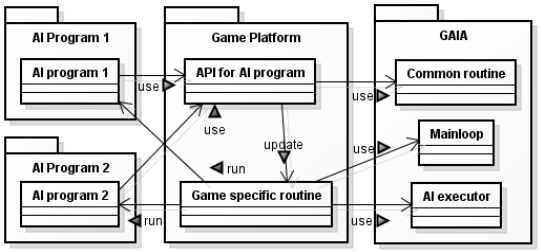



図 6　GAIA のアーキテクチャ
Fig. 6　An architecture of GAIA

GAIA は Java 言語で実装 されており，いくつかの言語処理系ライブラリを内蔵 する．また，GAIA は大きく分けて，ゲーム共通処理部，AI プログラム実行部，メインループ部 の 3 つに分かれる．ゲーム共通処理部 は，二次元座標を用いたゲームがよく利用する汎用的な処理や入力機器の処理，描画処理などを提供する．AI プログラム実行部 は，計算時間に制約を加えたり，ゲーム内容の再現機能を加えたりして，AI プログラムを実行する仕組みを提供する．メインループ部 は，シーン概念を導入したメインループとそれに付随する FPS 制御とグローバル変数の導入を抑制する仕組みを提供する．以降で，設計原則の観点からそれぞれの詳細を説明する．

### 6.1 Java 言語による実装

GAIA は Java6.0 を用いて実装した．そのため，GAIA を利用するゲームプラットフォームはマルチプ

| 品質特性 | 移植性 | | 機能性 | | 信頼性 |
|---|---|---|---|---|---|
| 品質副特性 | 環境適応性 | | セキュリティ | | 障害許容性 |
| 要求 | **プラットフォーム非依存** | **多言語対応** | **書き換え不可** | **AIプログラム間の独立性** | **AIプログラムからの独立** |
| 設計原則 | 「プラットフォーム非依存な中間言語」 | 「言語処理系の内蔵」 | 「Immutableパターン」 | 「時間独立なAIプログラムの実行」 | 「AIプログラムの計算時間の制限」 |
| フレームワーク | Java言語による実装 | 言語処理系ライブラリの内蔵 | ゲーム共通処理部 | AIプログラム実行部 | |

図 4　移植性と機能性と信頼性における GAIA が提供する機能の対応関係
Fig. 4　A relation about for portability, functionality and reliability

| 品質特性 | 効率性 | 使用性 | | | 保守性 | |
|---|---|---|---|---|---|---|
| 品質副特性 | 時間効率性 | 理解性 | 習得性 | 魅力性 | 解析性 | 変更性 |
| 要求 | **視聴に向いたゲーム速度** | **理解が容易なAPI** | **習得が容易なAPI** | **魅力的なゲーム** | **解析が容易なコード** | **変更が容易なコード** |
| 設計原則 | 「ゲーム内容の再現」 | 「入出力の口の限定」 | | 「メインループとFPS制御」 | 「シーンによるフレーム処理の構造化」「グローバル変数導入の抑制」 | |
| フレームワーク | AIプログラム実行部 | | | メインループ部 | | |

図 5　効率性と使用性と保守性における GAIA が提供する機能の対応関係
Fig. 5　A relation about efficiency, usability and maintainability

ラットフォームになり，Windows, Macintosh, Linux, FreeBSD など様々な OS，また，様々な PC アーキテクチャ上で動作する．このことは,「プラットフォーム非依存な中間言語」を実現する．

### 6.2　言語処理系ライブラリの内蔵

GAIA は Jython, JRuby, Rhino を内蔵しており，それぞれ Python, Ruby, JavaScript 言語のコードを実行できる．また，JavaVM 上で動作するプログラミング言語が存在しており，Scala や Clojure 言語は特別な準備をせずに，Java が動作する環境で実行できる．したがって，GAIA を用いて開発するゲームプラットフォームでは，上述のプログラミング言語を全て実行でき，参加者が作成する AI プログラムを作成する言語として利用可能である．このことは,「言語処理系の内蔵」を実現する．

### 6.3　ゲーム共通処理部

ゲーム共通処理部 が提供する処理のうち，二次元座標に関する処理が存在する．

例えば，`Point2` クラスは，二次元座標内の位置を表現する責務を負う．`Point2` クラスは,「Immutable パターン」を利用しており，全てのフィールドは読み取り専用で，どのようなメソッドを呼び出してもフィールドの値は変化しない．例えば，`add` メソッドは，新たに `Point2` 型のオブジェクトを生成して，レシーバーオブジェクトのフィールドの値を変化させない．なお，`Point2` クラスはゲームプラットフォームと AI プログラムの両方からの利用を想定している．

このように，汎用的なゲームの共通処理として，Immutable オブジェクトを提供することで,「Immutable パターン」を実現する．

### 6.4　AI プログラム実行部

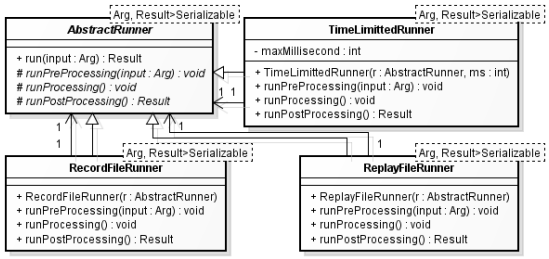



図 7　AI プログラム実行部のクラス図
Fig. 7　A class diagram of the execution component

AI プログラム実行部 はゲーム内容の再現機能とスレッドを用いて AI プログラムの計算時間を制限する機能を備えて，AI プログラムを実行する．図 7 で AI プログラム実行部 のクラス図を示す．

`AbstractRunner` 抽象クラスは AI プログラムを実行する責務を負う．`run` メソッドは，1 つの引数を受け取って AI プログラムの計算結果を初期化する `runPreProcessing` メソッドと AI プログラムの計算処理を行う `runProcessing` メソッド，1 つの戻り値で AI プログラムの計算結果を返す `runPostProcessing`

メソッドを順番に実行する．これらのクラスは Template Method パターン [12] により設計されており，3 つのメソッドをオーバーライドして，AI プログラムの実行処理を実装する．また，これらのクラスは Decorator パターン [12] により設計されており，`AbstractRunner` 型のオブジェクトに委譲して，ゲーム内容の再現機能と計算時間の制限機能を付加する．なお，`run` メソッドは，1 つの引数を受け取って 1 つの戻り値を返すため，自然と入出力の口はそれぞれ 1 つに限定される．

`RecordFileRunner` クラスは AI プログラムを記録しながら実行する責務を負う．コンストラクタで受け取った `AbstractRunner` 型のオブジェクトに委譲して，AI プログラムを実行する．`runPostProcessing` メソッドは AI プログラムの計算結果を記憶する．

`ReplayFileRunner` クラスは AI プログラムの計算結果を再生する責務を負う．`runPreProcessing` メソッドと `runProcessing` メソッドは何も処理せず，`runPostProcessing` メソッドは記憶した AI プログラムの計算結果を返す．

`TimeLimittedRunner` クラスは AI プログラムに計算時間の制限を加えて実行する責務を負う．`RecordFileRunner` クラスと同様に，委譲により AI プログラムを実行する．`runProcessing` メソッドはスレッドを用いて計算時間の制限を加えながら AI プログラムを実行して，AI プログラムの計算が完了する，若しくは，計算時間が制限に達するまで動作し続ける．つまり，`runProcessing` メソッドは同期的に 1 つの AI プログラムを実行するため，各 AI プログラムは時間的に独立して順番に実行される．

このように，AI プログラムの実行に関する汎用的な処理を提供して,「入出力の口の限定」と「ゲーム内容の再現」,「AI プログラムの計算時間の制限」,「時間独立な AI プログラムの実行」を実現する．

### 6.5 メインループ部

メインループ部 はシーンによって構造化された処理をフレーム単位で行う．図 8 で メインループ部 のクラス図を示す．

`Scene` 抽象クラスはシーンを表現して，ゲームを進行させる責務を負う．シーンが切り替わる際に新しいシーンの `initialize` メソッドが呼び出され，古いシーンの `release` メソッドが呼び出される．開発者はこのメソッドを実装して初期化と解放処理を実装する．`run` メソッドは 1 フレーム分だけゲームの進行処理をする．`run` メソッドは次のシーンの `Scene` 型のオブジェクトを戻り値で返す．開発者はシーンを切り替えない場合に自分自身を返すように実装する．`draw` メソッドはシーンの内容を画面に表示する．

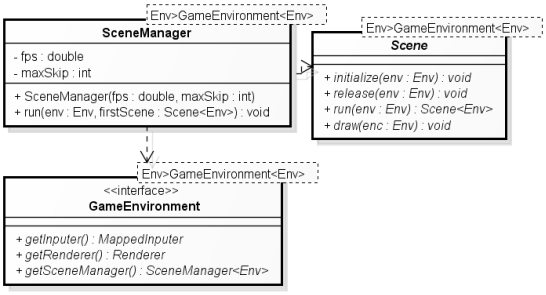



図 8　メインループ部のクラス図
Fig. 8　A class diagram of the mainloop component

`GameEnvironment` インタフェースはグローバル変数の導入を抑制するために，ゲームプラットフォーム全体の内部状態を表現する責務を負う．描画処理を行う `Renderer` 型のオブジェクト，メインループを表現する `SceneManager` 型のオブジェクトを持つ．また，開発者によるデバッグのために，入力機器を処理する `MappedInputer` 型のオブジェクトも持つ．

`GameEnvironment` 型のオブジェクトは，`SceneManager` クラスが様々な処理を行う際に引数として伝搬される．なお，グローバル変数に必要なフィールドはゲームプラットフォームによって異なるため，開発者は `GameEnvironment` インタフェースを実装する際に，任意のフィールドを追加できる．`GameEnvironment` インタフェースを実装した具象型のフィールドへ容易にアクセスするため，メインループ部 では，ジェネリクスを利用して具象型の情報を持つ．

`SceneManager` クラスはメインループの処理を行い，シーンによる 1 フレーム分のゲームの進行処理，シーンの画面表示処理を司る責務を負う．`run` メソッドはメインループの処理を開始する．その際，`fps` と `maxSkip` フィールドによって示された値にしたがって FPS 制御とフレームスキップを行う．

このように，メインループに関する汎用的な処理を提供して,「メインループと FPS 制御」と「シーンによるゲーム処理の構造化」,「グローバル変数利用の抑制」を実現する．

### 6.6 GAIA の利用方法

GAIA を利用してゲームプラットフォームを開発す

る際に，開発者が実装すべき内容を以下に列挙する．

- GameManager インタフェースを実装するクラスを作成して，ゲームの進行中に共有すべき情報をフィールド変数として定義する．
- Scene インタフェースを実装するクラスを作成して，シーンの表示処理とシーンの表示中に行うゲームの進行処理をメソッドとして定義する．特に，シーンの切り替えが必要な際は，Scene インタフェースの run メソッドの戻り値で，切り替え先のシーンに該当するクラスのオブジェクトを返すように実装する．
- AbstractRunner 抽象クラスを継承するクラスを作成して，AI プログラムを実行して，得られた計算結果を返すように実装する．また，必要に応じてユーザーが入力機器を介してゲームを操作するクラスも作成する．
- ゲーム進行の処理内で，AbstractRunner 型のオブジェクトを介して，AI プログラム及びユーザーからゲームへの入力を反映する処理を実装する．AbstractRunner 型のオブジェクトは，AI プログラム実行部 が提供するクラスや，AbstractRunner 抽象クラスを継承するクラスから生成する．必要に応じて Decorator パターンを利用して録画機能などを付加する．
- AI プログラム同士の対戦結果の録画，録画した内容の再生など，ゲームプラットフォームの実行目的に応じて，ゲーム開始時に，生成するオブジェクトのクラスが適切に選ばれるように実装する．

上述した開発者が実装すべき内容の詳細を理解する助けとなるよう，我々が想定する形で GAIA を利用して開発した際の，ゲームプラットフォームの大まかな処理の流れを以下に示す．

（1） **ゲームの開始：**GameManager インタフェースを実装するクラス，最初のシーンに該当する Scene インタフェースを実装するクラス，SceneManager クラス，AbstractRunner 抽象クラスを継承するクラスなど，必要なクラスのオブジェクトをそれぞれ初期化して，SceneManager クラスの run メソッドを呼び出す．

（2） **ゲームの進行：**SceneManager クラスの run メソッドから，Scene インタフェースを実装するクラスの run メソッドと draw メソッドを呼び出す．run メソッドでは，AbstractRunner 型のオブジェクトを介して，AI プログラムやユーザーからゲームプラットフォームへの入力を受け取って反映する．

（3） **ゲームの終了：**最後のシーンに該当する Scene インタフェースを実装するクラスの run メソッドの戻り値で null を返して，メインループを抜ける．

開発者は我々が想定する構成に従ってクラスを実装して，更に，GAIA が提供する機能を積極的に利用することで，第 7 節で述べるような品質保証や開発コスト低減の効果を得て，ゲームプラットフォームを開発できると考えられる．一方で，GAIA への理解不足などの理由から，上述したような実装を行わずに GAIA を利用すると，そのような効果が得られないどころか，かえって品質を低下させる可能性がある．ゲームプラットフォームの品質低下は，ゲームそのものの魅力低下に繋がり，参加者のコンテストへの参加意欲の喪失，AI プログラムの質の低下を引き起こす．

GAIA では，ゲームプラットフォームの実装に必要なインタフェースを提供して，GAIA が提供する機能を利用する上で，これらのインタフェースの実装を必要不可欠とする．また，開発者が実装すべきインタフェースのメソッドは，引数に GAIA が提供するクラスのオブジェクトを持ち，GAIA が提供するクラスの利用する機会を生み出す．このことは，GAIA を利用する中で自然と GAIA への理解を深め，我々が想定するような構成に従ってクラスを実装して，GAIA が提供する機能を積極的に利用するように開発者を促す．

## 7. GAIA の評価

本節では，設計原則に基づいた GAIA を利用して開発したゲームプラットフォームが 4 節で説明した各要求を適切に満たしているかどうか議論する．

### 7.1 GAIA を利用した事例報告

2010 年度に開催された楽天テクノロジーカンファレンスの 1 セッションである早稲田楽天プログラミングコンテスト (以降，WR2010) [14] のゲームプラットフォームを開発して運用した．1 チーム 3 人までの構成で 19 チームが参加した．参加者は 17 日間の期間で Java，Ruby，Python，JavaScript，Scala 言語のいずれかで AI プログラムを作成して競い合う．なお，コンテスト当日の会場の様子を図 9 にて示す．

ゲームプラットフォームは，図 10 のように 4 プレイヤーで対戦するカーソルと兵士の 2 つのキャラクタを動かす 2 次元座標上のパズルゲームである．カーソルは任意の場所に移動して，2 × 2 マス内の色を時計回り若しくは反時計回りに回転できる．一方，兵士はプレイヤー固有の色と同じ色のマスを 1 歩ずつ移動できる．兵士が相手プレイヤーの大小いずれかの城門に


図 9　早稲田楽天プログラミングコンテストの会場の様子
Fig. 9　A picture of WR2010

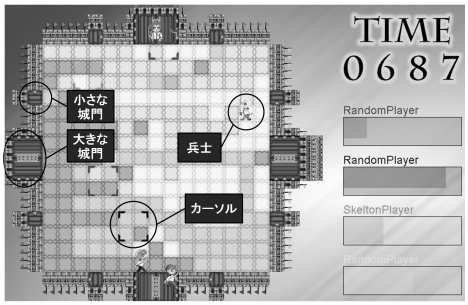



図 10　早稲田楽天プログラミングコンテストのゲーム画面
Fig. 10　A game screen of WR2010

表 1　主催したコンテストのまとめ
Table 1　A summary of contests

| | チーム数 | 作成期間 | 開発規模 | 開発工数 | GAIA 利用 |
|---|---|---|---|---|---|
| JC2009 | 35 | 1 時間半 | 2870 行 | 9 人月 | 無 |
| JC2010 | 45 | 4 時間 | 3187 行 | 5 人月 | 無 |
| WR2010 | 19 | 19 日 | 3751 行 | 10 人月 | 有 |

移動すると，該当プレイヤーの点数を奪える．つまり，カーソルでマスの色を変化させ，兵士が移動可能な経路を作成して，兵士を相手プレイヤーの城門に移動させて戦うゲームである．大きな門と小さな門に移動させた場合はそれぞれ 20%と 10%の点数を奪える．一定時間の経過後に，最も得点の高いプレイヤーの勝利となる．ゲームプラットフォームは Google Code にてオープンソースとして公開しており [15]，ゲーム学会第 8 回ゲーム作品コンペにて優秀賞を受賞した．オープンソースであるため，GAIA の利用例としても有用である．

表 1 で，我々が主催したコンテストを簡単にまとめる．各項目は，参加者のチーム数，AI プログラムの作成期間，ゲームプラットフォームの総コード行数，ゲームプラットフォームの開発工数，GAIA 利用の有無を示す．JC2010 と WR2010 の開発メンバーは等しく，WR2010 では GAIA と平行してゲームプラットフォームを同じ開発メンバーで開発した．ただし，GAIA とゲームプラットフォームの開発工数の比率は開発メンバーによって様々である．WR2010 の開発規模と開発工数はゲームプラットフォームと GAIA の合計である．

### 7.2　移　植　性

ゲームプラットフォームを Java 言語を用いて実装して，更に，言語処理系ライブラリを内蔵することで，**プラットフォーム非依存**と**多言語対応**を満たす．実際に WR2010 では，Java 以外に Ruby と Scala 言語で AI プログラムを作成したチームが存在した．

### 7.3　機　能　性

GAIA が ゲーム共通処理部 にて，イミュータブルなクラスを提供することで，**書き換え不可**を満たすように働きかける．実際に GAIA が提供する `Point2` クラスを利用する前の WR2010 において，`Point2` クラスのフィールドが書き換え可能な形で公開されており，AI プログラムから不正にゲーム内情報を書き換えられる欠陥があった．しかし，GAIA を導入することで，少なくとも汎用的なクラスにおいては，上述のような欠陥を防ぐことが可能になる．

GAIA が AI プログラム実行部 にて，AI プログラムの実行機能を提供することで，**AI プログラム間の独立性**を満たすように働きかける．また，Decorator パターンを利用して，スレッド利用の有無の切り替えを容易にした．

### 7.4　信　頼　性

GAIA が AI プログラム実行部 にて，AI プログラムを時間的に独立して実行する機能を提供することで，**AI プログラムからの独立**を満たすように働きかける．実際に WR2010 では AI プログラムの欠陥が，ゲームプラットフォームに障害に繋がらなかった．

### 7.5　効　率　性

GAIA が AI プログラム実行部 にて，AI プログラムの計算結果の記録機能と再生機能を提供することで，**視聴に向いたゲーム速度**を満たすように働きかける．実際に FPS 制御によって FPS を調整することで，1 ゲームに必要な再生時間を自由に変更可能であり，WR2010 では，視聴者にとって快適な速度での放映に成功した．

### 7.6　使　用　性

GAIA が AI プログラム実行部 にて，AI プログラムが入出力の口を限定することで，**理解が容易な API** と**習得が容易な API** を満たすように働きかける．ま

表 2　参加者のアンケート結果
Table 2　A questionnaire result of entrants

| アンケート項目 | API の使用性 | コンテストの満足度 |
|---|---|---|
| 大変良い | 10% | 40% |
| 良い | 20% | 40% |
| 普通 | 30% | 20% |
| 悪い | 40% | 0% |
| 大変悪い | 0% | 0% |

表 3　ソフトウェアメトリクスの測定結果
Table 3　A measurement result of software metrics

| 測定対象＼メトリクス | LOC | サイクロマチック数 | RFC | LCOM |
|---|---|---|---|---|
| JC2009(利用前) | 2870 | 671 | 16 | 1.8 |
| JC2009(利用後) | 1981 | 380 | 15 | 1.6 |
| JC2010(利用前) | 3187 | 652 | 17 | 1.8 |
| JC2010(利用後) | 2432 | 462 | 16 | 1.6 |
| WR2010(参考) | 2873 | 648 | 15 | 1.4 |
| GAIA(参考) | 878 | 197 | 9 | 1.2 |

た，GAIA が メインループ部 にて，FPS 制御による描画機能を提供することで，**魅力的なゲーム**を満たすように働きかける．

表 2 で WR2010 を開催した際に，参加者のうち 10 チームから回収したアンケート結果を回答者の割合によって示す．

API の使用性において，4 割が使いにくいという評価とともにコメントを残しているが,「ドキュメントと API の挙動に相違がある点が分かりにくい」，若しくは,「API の仕様がコンテスト期間中に変更した点が分かりにくい」という内容であった．一方で，悪い以外を回答した評価のコメントでは,「API が使いやすい」，若しくは,「必要最低限な API のみならず付加的な API も提供した点が良い」という内容であった．したがって，API ドキュメントの不備と API の仕様変更に対する批判はあったものの，API そのものは評価されており，この点において API の使用性は十分に良かったと考えられる．ただし，設計原則や GAIA がドキュメントや仕様変更に対しても対処すべきであることが伺える．コンテストの満足度は 8 割が満足以上となっており，大変高い評価を受けている．更に，楽天テクノロジーカンファレンス 2010 内の 16 セッションのうち，視聴者からの良かったというアンケートの得票数が 5 位であった．したがって，コンテストの参加者と視聴者の両方において魅力的なゲームであると考えられる．

### 7.7 保　守　性

GAIA が メインループ部 にて，シーンによって構造化されたクラスを提供することで，**解析が容易なコード**と**変更が容易なコード**を満たすように働きかける．

GAIA によって保守性が向上することを確認するため，GAIA を利用しなかった JC2009 と JC2010 のゲームプラットフォームに対して，GAIA を利用するようそれぞれプログラマ 1 名が 3 時間程度かけて修正した．表 3 で利用前と利用後の保守性に関するソフトウェアメトリクスの測定結果を示す．

LOC はプログラムの規模を示し，コメントを除いたコード行数である．表の値は LOC の総数で，GAIA によって 800 から 900 行程度低下して，プログラムの規模の縮退化に成功している．サイクロマチック数は複雑度を示し，プログラムにおける基本パスの数である [16]．表の値はサイクロマチック数の総数で，GAIA によって 200 から 300 程度低下して，プログラムの簡潔化に成功している．RFC は複雑度を示し，あるクラスが関係するメソッドの総数である [17]．表の値は RFC のクラス単位の平均値で，GAIA によって 1 低下して，プログラムの簡潔化に成功している．LCOM は凝集度を示し，クラス内のメソッド同士が関係しあう度合い (低い方が良い) である [17]．表の値は LCOM のクラス単位の平均値で，GAIA によって 0.2 低下して，クラスについて適切な粒度によるモジュール化に成功している．したがって，GAIA によって保守性が向上すると考えられる．

なお，プログラムの規模の縮退化とプログラムの簡潔化は，プログラムの開発に必要なコードの記述量を減らし，更に，コードを記述しやすくすると考えられる．したがって，開発工数による直接的な比較はされていないものの，LOC とサイクロマチック数については 3 割程度の減少が見られるため，GAIA はゲームプラットフォームの開発工数も低減すると考えられる．

## 8. 関 連 研 究

O'Kelly ら [18] は，問題解決型学習 (Problem-based Learning;PBL) として Robocode を利用した教育手法を提案した．問題解決型学習が現実世界の問題に対処することを強調している点を踏まえ，Robocode を利用した競い合いが問題解決型学習の良い問題であるということを提案した．更に，問題解決型学習の問題を分類し，いくつかの評価基準を用いて，Robocode が適している理由を説明した．GAIA では，Robocode などのゲームプラットフォームの開発を支援するため，彼らが提案する問題解決型学習による教育手法を相補できる点で関連がある．

Sergio ら [19] は，多人数同時参加型オンライン RPG という形態の高品質なゲームエンジンを開発するためのアーキテクチャを提案した．提案アーキテクチャはアーキテクチャパターンをいくつか組み合わせて構成される．更に，彼らはアーキテクチャに対して 6 つの目標を設定して，各目標が提案アーキテクチャによってどの程度達成されるかということを議論した．GAIA が対象とするゲームプラットフォームと形態が異なるものの，同じくゲームソフトウェアを対象としたフレームワークであるという点で関連がある．

## 9. むすびに

本論文では，品質特性の観点からゲームプラットフォームの要求を分析して，必要な知見を設計原則としてまとめ，GAIA を提案した．GAIA は設計原則に基づいてアーキテクチャと汎用的な共通処理を提供することで，ゲームプラットフォームの品質を保証して，更に，開発コストを低減する．また，事例報告とソフトウェアメトリクスやアンケートによる評価を通して GAIA の有用性を示した．

本論文では，ソフトウェア工学的な立場による議論を主題としたため，ゲームデザインに踏み込んだ議論を行わなかった．しかし，より面白く教育上有用なコンテストを開催するためには，魅力的なゲームルールを持つゲームプラットフォームを開発しなければならない．そうした，ゲームの面白さに着目することで，今後更に設計原則と GAIA を発展させる．

**謝辞** ゲームプラットフォームの開発，及び，コンテストの開催に協力頂いた志水 理哉氏, 高橋 周平氏, 村上 真一氏，中村 悠人氏, 庄山 昭彦氏, 内山 諭氏, 城間 祐輝氏, 野本 悠太郎氏に感謝する．

## 文　献

[1] N.V. Shilov and K. Yi, “Engaging students with theory through acm collegiate programming contest,” Commun. ACM, vol.45, pp.98–101, Sept. 2002. http://doi.acm.org/10.1145/567498.567506

[2] IBM, “Robocode home”. http://robocode.sourceforge.net/

[3] ACM, “The acm-icpc international collegiate programming contest web site sponsored by ibm”. http://cm.baylor.edu/welcome.icpc

[4] 坂本一憲，大橋　昭，志水理哉，高橋周平，村上真一，中村悠人，庄山昭彦，内山　諭，城間祐輝，野本悠太郎, “Gameaiarena”. http://code.google.com/p/gameaiarena/

[5] ACM, “Acm/icpc asia regional contest 2009, tokyo, japan”. http://www.waseda.jp/assoc-icpc2009/jp/

[6] 坂本一憲, 大橋　昭, 志水理哉, 高橋周平, 村上真一, “Java チャレンジ 2010 summer”. http://www.washi.cs.waseda-.ac.jp/javachallenge2010summer/

[7] 坂本一憲, 中村悠人, 庄山昭彦, 内山　諭, 城間祐輝, 野本悠太郎, “Javachallenge2009”. http://code.google.com/p/-javachallenge2009/

[8] 坂本一憲，大橋　昭，志水理哉，高橋周平，村上真一, “Javachallenge2010”. http://code.google.com/p/java-challenge2010/

[9] M. Grand, Patterns in Java, Volume 1, A Catalog of Reusable Design Patterns Illustrated with UML, John Wiley and Sons, 1998.

[10] R.E. Johnson and B. Foote, “Designing Reusable Classes,” Journal of Object-Oriented Programming, vol.1, no.2, pp.22–35, 1988.

[11] K. Cwalina and B. Abrams, Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries, 2nd edition, Addison-Wesley Professional, Nov. 2008.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software, Addison-Wesley Professional, 1995.

[13] W. Wulf and M. Shaw, “Global variable considered harmful,” SIGPLAN Not., vol.8, pp.28–34, Feb. 1973.

[14] 坂本一憲, 大橋　昭, 志水理哉, 高橋周平, 村上真一, “プログラミングコンテスト”. http://www.washi.cs.waseda.ac.jp/-rakuten-waseda-pc2010/ （ゲーム学会第 8 回ゲーム作品コンペ 優秀賞）

[15] 坂本一憲，大橋　昭，志水理哉，高橋周平，村上真一, “Quinte”. http://code.google.com/p/quinte/

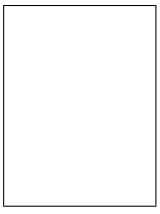
[16] T.J. McCabe, “A complexity measure,” Proceedings of the 2nd international conference on Software engineering, pp.407–, ICSE ’76, IEEE Computer Society Press, Los Alamitos, CA, USA, 1976.

[17] S.R. Chidamber and C.F. Kemerer, “A metrics suite for object oriented design,” IEEE Trans. Softw. Eng., vol.20, pp.476–493, June 1994.

[18] J. O’Kelly and J.P. Gibson, “Robocode & problem-based learning: a non-prescriptive approach to teaching programming,” SIGCSE Bull., vol.38, pp.217–221, June 2006.

[19] S. Caltagirone, M. Keys, B. Schlief, and M.J. Willshire, “Architecture for a massively multiplayer online role playing game engine,” J. Comput. Small Coll., vol.18, pp.105–116, Dec. 2002.

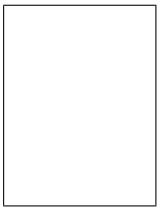
（平成 xx 年 xx 月 xx 日受付）



**坂本一憲**

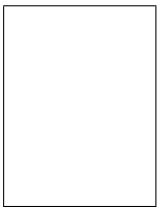
2009 年早稲田大学コンピュータ・ネットワーク工学科卒．2010 年同大学大学院基幹理工学研究科修士課程修了．2010 年より同大学大学院基幹理工学研究科博士課程に所属．2011 年より同大学助手．現在に

至る．情報処理学会，日本ソフトウェア科学会，ACM 各学生会員．



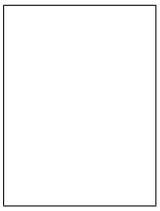
**大橋昭**

2010 年早稲田大学コンピュータ・ネットワーク工学科卒．2010 年より同大学大学院基幹理工学研究科修士課程に所属．現在に至る．



**鷲崎弘宜**

2003 年早稲田大学大学院博士後期課程修了、博士（情報科学）．同大学助手，国立情報学研究所助手を経て 08 年より同大学理工学術院准教授，同研究所客員准教授．再利用と品質保証を中心にソフトウェア工学の研究や教育に従事．情報処理学会論文誌編集委員会基盤グループ主査，同学会代表会員，情報規格調査会 SC7/WG20 小委員会主査，日科技連 SQiP 研究会運営委員長．日本ソフトウェア科学会，電子情報通信学会，IEEE，ACM 各会員．



**深澤良彰**

1976 年早稲田大学理工学部電気工学科卒．1983 年同大学大学院博士課程修了．同年相模工業大学工学部情報工学科専任講師．1987 年早稲田大学理工学部助教授．1992 年同教授. 工学博士．主として，ソフトウェア再利用技術を中心としたソフトウェア工学の研究に従事．情報処理学会，電子情報通信学会，日本ソフトウェア科学会，IEEE，ACM 各会員．

**Abstract** Programming contest, which has the style that people participate in the contest through artificial intelligence programming for given game software, is remarkable. However, the game software is different from a general game software and an enterprises system. In this paper, we systematize design principles for the game software and propose a framework which helps to develop the game software. The framework provides a software architecture referring the design principles and general common code. The framework guarantees the quality of the game software and reduces development cost. Moreover, we reports three cases which use the framework, user evaluation by questionnaire and evaluation in lines of code to explain effectiveness of proposal framework.

**Key words** Framework, Requirements Analysis, Design Principle, Programming Contest, Game Platform