筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

進化的計算の実験可視化システムの開発
-要求管理の実践と
スケジューリング実行機能の開発-

岡谷 亮

修士（工学）

（コンピュータサイエンス専攻）

指導教員　高橋 伸

2014年3月

概要

本報告書では，機械学習分野の計算手法の一つである進化的計算の実験作業を効率化する Web アプリケーションの開発プロジェクトについて，プロジェクトの概要と筆者の担当や取り組みについて記述した．本プロジェクトは，進化的計算の研究者である筑波大学教員を顧客とし, 筆者を含めた筑波大学大学院博士前期課程コンピュータサイエンス専攻 2 年の学生 4 人でチームを組み，約 6 か月の期間で要件定義，設計，製造，テスト工程を含んだシステム開発を行った．

本プロジェクトの目的は，顧客が進化的計算の実験を行う時の作業負担を軽減することである．顧客は進化的計算の最適なパラメータを求めるために計算実験を行い，その計算実験で計算プログラムの実行，実行プロセスの監視，実行結果の分析，パラメータ値の変更等の作業工程を反復的に行っている．しかし，この実験は顧客にとって非常に手間がかかる作業であり，その原因には計算時間が長いこと，実行プロセスの監視が困難であること，パラメータ設定や結果分析の手間が大きいこと等の問題が挙げられる．本プロジェクトではこの作業負担を軽減するために，実験の実行管理と実験監視，実験結果の可視化機能により進化的計算実験を総合的に支援する Web アプリケーション (GAM) を開発した．顧客を対象にした GAM の評価アンケートの結果では，GAM は主に「実験の実行管理機能」や「実行プロセス状態の通知」の観点で高い評価を得た．

筆者の本プロジェクトへの貢献としては，主に筆者の開発担当部分であるスケジューリング実行機能の開発と，筆者の取り組みである要求管理の実践について記述した．スケジューリング実行機能の開発では，顧客がブラウザ上で実行操作を行った実験を．監視プロセスの非同期実行により実際にサーバマシン上で実行する機能を開発した．要求管理の実践では，顧客の要求の種類を分類してプロダクトバックログに記述することで要求を管理した．また，筆者の担当作業である関連技術の調査とプロジェクト指標の計測について，scientific workflow の実行管理システムである Kepler との比較や，チケット記録情報とファンクションポイントを計測を行った．

目 次

図 目 次

第1章　プロジェクトの背景

この章では，本プロジェクトの背景として，プロジェクトの顧客について説明し，顧客の問題解決の要望の対象である進化的計算の実験について説明する．

1.1　本プロジェクトの顧客

本プロジェクトの顧客は筑波大学に所属する Claus Aranha 教員であり，進化的計算 (Evolutionary Computation) などの研究を行っている．進化的計算は最適化問題の解決に有用な方法であり，特に解空間構造が不明で，全探索が不可能なほど広大な解空間を持つ問題に有効な手法である [1]．顧客が過去に携わった進化的計算の研究例には，ポートフォリオ最適化 [2] や，多次元データの二次元座標への変換最適化 [3] についての研究がある．

ポートフォリオ最適化の研究は，金融工学分野のポートフォリオ (金融資産の配分) について，収益を最大化しリスクを最小化するアルゴリズムについての研究である．多次元データの二次元座標への変換最適化の研究は，多次元データを 2 次元のプロットに射影して，視覚的に識別しやすくするようにする研究で，例えば手書き数字の識別や，血液中の成分の情報から血中細胞の種類を識別することに応用することができる．これらの研究に，進化的計算手法が用いられている．

1.2　進化的計算手法

進化的計算手法とは，生物の遺伝子の複製や自然淘汰のメカニズムをモデルとした，工学的に幅広く応用されている計算手法のことである [4]．進化的計算の代表的な例として，遺伝的アルゴリズム (Genetic Algorithms, GA) や進化的プログラミング (Evolutionary Programming, EP) がある．

本プロジェクトで特に注意すべき進化的計算の特徴として，遺伝的アルゴリズムの解探索性能はパラメータ (個体数・突然変異確率・交叉確率など) に依存するという特徴がある．また，進化的計算のパラメータの最適値は最適化問題の対象によって異なり，このパラメータ設計の決定は経験的にしか決められない．このパラメータの最適化のために，GA のパラメータを最適化するためにさらに GA を用いる手法 [5] などが提案されているが，確立された方法は存在していない [6].

1.3　進化的計算の計算実験

進化的計算の研究者は，進化的計算の最適なパラメータについて調査するために，図 1.1 に示すような計算実験を行う．

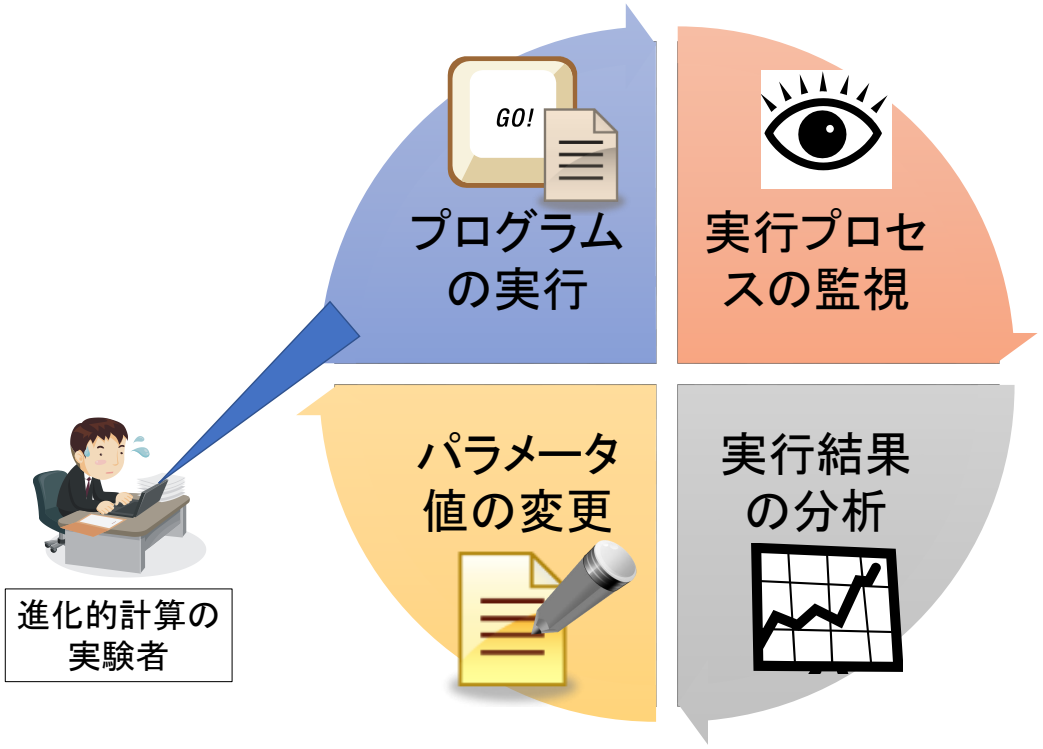


図 1.1: 進化的計算の実験にて最適なパラメータについて調査するプロセス

プログラムの実行
: 計算実験ではます，ある設定したパラメータについて，進化的アルゴリズムのプログラムを実行する．顧客の場合は，パラメータをプログラムとは別のテキストファイルに記述してパラメータファイルを作成し，進化的計算を java 言語や python 言語を用いて実装して，リモートサーバ上で実装したプログラムをパラメータファイルを引数にして実行することが多い．

実行プロセスの監視
: 実験者は実行されたプロセスを監視し，実行プロセスが正常に終了したことを確認した後に次のプロセスに移る．途中でエラー終了等が発生した場合は，プログラムやパラメータ値を修正してもう一度実行をやり直す，

実行結果の分析
: 実行プロセスから出力された結果データを分析する．出力されるデータには，例えば進

化的計算の個体の適応度や多様性，距離などがあり，そのデータを統計処理または可視化することでパラメータ値が進化的計算に与える影響を調べる．

パラメータ値の変更
: 分析結果を元にして，次に実行すべきパラメータを決定する．そして，パラメータを書き換えて再度プログラムの実行プロセスに移行する．顧客は，パラメータの変更をテキストファイルの内容をテキストエディタ等で書き換えることで行っている．

これらのプロセスを繰り返すことで，進化的計算の研究者は最適なパラメータを探索している．

1.4 計算実験の問題と顧客の要望

1.3 で述べた計算実験は，実験者にとって作業負担が高い．例えば，この計算実験には以下のような問題がある．

計算時間が長い
: 進化的計算では一般に長い計算時間がかかる上に，必ずしも一意の計算結果が得られないため，同じパラメータの計算も複数回実行して統計的な結果を得る必要がある．そのため，計算実験に要する時間が非常に長く，顧客の経験では，計算実験に合計で数十時間以上かかる場合もある．

実行プロセスの監視が困難
: 計算実験の効率化のためには，実行プロセスが終了したら，実験者はすぐに計算結果の分析や他の計算の実行に移るべきである．また，実行プロセスがエラー終了する可能性もあり，その場合はエラー内容を修正して計算を再度実行するべきである．しかし，実験者が常にコンソール画面を確認していなければ，計算プロセスの終了を即座に知ることができない．

パラメータ設定や結果分析の手間が大きい
: 繰り返しプログラムを実行し，同じパラメータの計算結果も複数出力される．そのため，パラメータの変更作業の回数が多く，分析しなければならない出力結果のデータも非常に多い．

これらの問題について，顧客は以下のような要望を持っている．

実験の実行管理
: 計算実験の管理を楽に行えるようにしたい．また，パラメータの設定や変更も楽に行えるようにしたい．

実験の監視
: メール通知等で，実験の終了等をすぐに認識できるようにしたい．また，実験の進捗状況も知ることができるとよい．

実験の可視化

実験結果を可視化することで，結果の分析を楽に行えるようにしたい．

1.5 本報告書の構成

本報告書の第 1 章では，プロジェクトの背景について述べた．以降は本プロジェクトについて，第 2 章でプロジェクトの概要を，第 3 章でプロジェクト体制とチーム内の役割分担を述べる．また，第 4 章で本プロジェクトの関連技術調査について述べる．

そして，筆者の役割として，第 5 章で要求管理の実践について，第 6 章でスケジューリング実行機能の開発について，第 7 章でプロジェクト指標の計測について述べる．最後に，第 8 章でプロジェクトの評価と今後の展望について述べ，第 9 章で本報告書のまとめを述べる．

第2章　プロジェクト概要

この章では，本プロジェクトの概要について述べる．

2.1　目的とアプローチ

本プロジェクトの目的は以下である．

- 顧客が進化的計算の計算実験を行う際の負担を軽減すること

この目的を達成するためのアプローチとして，本プロジェクトでは進化的計算の実験の実行管理，実験の監視，実験結果の可視化の機能を持ち，進化的計算実験を総合的に支援するWebアプリケーションを開発する．

進化的計算実験のサンプルとして，顧客の研究であるポートフォリオ最適化の研究と多次元データの二次元座標への変換最適化の研究 (1.1 参照) についての実験ファイルを顧客に頂き．それを要件定義や設計，システムテストの参考に用いる．開発するWebアプリケーションの命名はGAM(Genetic Algorithm Manager) とする．

2.2　対象ユーザ

GAMのユーザは，本プロジェクトでは基本的に顧客のみを想定する．ただし，顧客は本プロジェクトが成功した場合は，将来的にGAMを公開したいという要望もある．そのため，ユーザの範囲を拡張してもGAMの機能が損なわれないようにする．

例えば，Webブラウザ上で複数ユーザが操作してもエラーが発生しないように設計を行い，英語圏ユーザへの将来的な対応のために画面上の言語表記は英語とする．

2.3　GAMの機能

顧客の要望 (1.4 参照) を満たすためのGAMの機能を表2.1に記述する．

実験の実行管理

GAMは実験・パラメータ管理機能とスケジューリング実行機能を持つ．実験・パラメータ管理機能は，顧客が計算実験とそのパラメータの作成・編集・削除や，実行する計算

表 2.1: 顧客の要望を満たす GAM の機能

目的	解決策
実験の実行管理	実験・パラメータ管理機能 スケジューリング実行機能
実験の監視	メール通知機能 実行プロセス管理機能
実験結果の可視化	グラフ表示機能 ライブラリ提供

実験とパラメータを選択することを可能にする．スケジューリング実行機能は，ブラウザ上で行われた実行操作について，計算実験を実際にサーバ上で実行する．

実験の監視

GAM はメール通知機能と実行プロセス管理機能を持つ．メール通知機能は実行された計算プロセスが終了した場合等に，登録されたメールアドレスにメール通知する．実行プロセス管理機能は，顧客がブラウザ上で現在実行中の計算プロセスの状態 (実行状態，実行時間など) を確認することを可能にする．

実験結果の可視化

GAM はグラフ表示機能を持ち，それを実現するためのライブラリを顧客に提供する．グラフ表示機能は，顧客が結果データを分析する時の参考になるように，完了した計算実験の結果データをグラフ表示する．また，この機能は計算結果のデータが GAM の読み込み規則と同じでなければ実現できないので，GAM とプログラム間のインタフェースとして，プログラムの結果を特定の規則で出力するライブラリを作成する．

2.4 GAM の概要

GAM のシステム概要図を図 2.1 に示す．GAM はクライアント・サーバ型のシステムであり，Web アプリケーションとして動作する．ユーザは GAM を以下のような手順で用いることができる．

1. 実験パッケージの作成

   まず，計算実験のために必要なファイルセットを作成する．GAM は，このセットのことを実験パッケージと呼んでいる．実験パッケージの実体は，実行ファイル (プログラム)，パラメータファイル，計算に必要な外部データセットなど 1 つのディレクトリに入れたものである．ユーザはこの実験パッケージを作成し，圧縮ファイルしてサーバ上にアップロードする．これにより，実験パッケージを GAM 上で管理できるようになる．

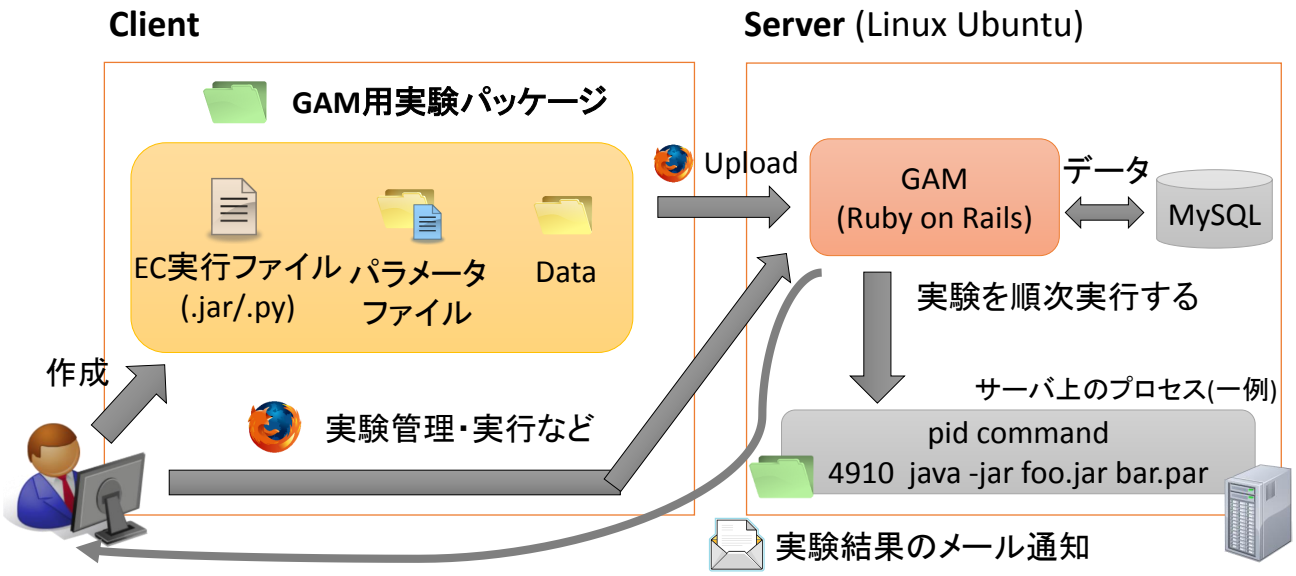


図 2.1: GAM のシステム構成と機能の概要

2. 実験の実行
   ユーザは GAM 上で実行したい実験パッケージとパラメータファイルを選択し，実行操作を行う．すると，選択した計算実験が開始し，実行プロセスがサーバ上で起動する．GAM は実行された計算プロセスを監視していて，このプロセスの状態が変化すると GAM から通知メールが送信される．

3. 実行結果の確認
   終了した計算実験の出力はサーバ上に保存され，ユーザはこの出力結果を GAM 上で閲覧することができる．実行したプログラムを GAM のライブラリを用いて記述した場合は，出力結果グラフを GAM 上で確認することができる．

2.5 システム構成

システム構成を表 2.2 に示す．サーバは Ubuntu Server 12.04LTS とし，クライアントの Web ブラウザは Firefox とする．Web Server には Apache を，DB には MySQL を用いる．Web アプリケーションの実装言語はフレームワークに Ruby on Rails を用いるために Ruby にする．Web 画面の CSS フレームワークに Bootstrap 3 を用いる．

表 2.2: システム構成

サーバ	Ubuntu Server 12.04LTS
クライアント	Firefox
Web Server	Apache 2.2
DB	MySQL 5 系
実装言語	Ruby 1.9.3p484
フレームワーク	Ruby on Rails 3.2.13
CSS フレームワーク	Bootstrap 3

2.6 セキュリティ対策

GAM のセキュリティ対策としては，意図しないユーザの利用を防ぐために，HTTP 認証を用いたアクセス制限を設ける．HTTP 認証を用いることで，ユーザが GAM にアクセスしたときにブラウザ上でユーザ名とパスワードの入力を求められるようにし，それらの情報を知らないユーザが GAM を利用することを防ぐことができる．

2.7 プロジェクトの目標

顧客の要求は，EC 実験を行う時の手間を軽減するために，実験の管理・監視・可視化機能を持った実験支援システムが欲しいということである．本プロジェクトでは，この実験の管理・監視・可視化機能の達成評価を測定可能な指標として，以下の 3 つの目標を定めた．

パラメータ作成の支援
: 複数パラメータを一括作成する機能を提供し，実験実行時のパラメータ設定の手間を軽減する

実行プロセスの監視
: 実行プロセスの状態をすぐに知ることができるようにする

実験結果の分析支援
: 結果分析の手間を軽減するために，実験結果を可視化する機能を提供する

2.8 スケジュール

本プロジェクトでは反復型開発プロセスを採用する．反復型開発の 1 反復の期間を 2 週間とし，2013 年 12 月上旬にシステムを納品することを目標として複数回の反復を繰り返す．

実績としては，2013 年 6 月 11 日から 2013 年 12 月 20 日までに，中間報告書の執筆のための期間と夏休み別にして，合計 10 回の反復を行った．最初の反復は 0 回目の反復として，プロジェクトの発足準備や開発環境の整備等を行った．各イテレーションの実績を表 2.3 に示す．

表 2.3: 各反復の主要な作業内容

反復	作業内容
0	プロジェクト方針の策定 開発環境の整備
1	実験管理機能のプロトタイプ 関連技術の調査
2	実験パッケージ管理機能 パラメータ管理機能
3	実験の実行機能 実行プロセス管理機能の設計
4	実行プロセス管理機能 グラフ表示機能の設計
5	メール通知機能 GAM ライブラリ作成
6	実行キャンセル機能 グラフ表示機能
7	イベントログ機能 複数パラメータ一括作成機能
8	マニュアル作成 総合テスト
9	バグ対応 特定課題研究報告書の準備

5 月下旬にプロジェクト開始し，6 月下旬までに開発方針の決定や要求分析，開発環境の整備などのプロジェクトの準備を行う．7 月以降に反復開発を開始し，以降は 2 週間の反復ごとに要求分析・設計・実装・テストを繰り返し行う．12 月上旬に総合テストと受け入れテストを完了し，納品とする．

2.9 開発規模

開発規模を計測する指標として，LOC(Lines of Code) を計測した．LOC とは，ソースコードのうちコメントと空行を除いた行の数であり，開発システムの規模を計測する指標として一般的に用いられる．ＧＡＭの最終的な計測値を表 2.4 に示す．表中の Application は GAM の実装コードであり，Test は単体テストコードである．この値の計測は，納品時の GAM を Ruby on Rails の stats コマンドを用いることで行い，自動計測されない Application の Processes クラスと Mailer クラスは手動で計測した値を表記している．

表中のクラスの総 LOC は 4494 となった．なお，この表には View Class(HTML) と javascript

表 2.4: Rstats

Application/Test	Class	LOC
Application	Controllers	990
	Helpers	131
	Models	686
	Libraries	86
	Processes	125
	Mailer	59
	小計	2077
Test	Controller specs	1243
	View specs	40
	Helper specs	24
	Model specs	726
	Process specs	302
	Mailer specs	82
	小計	2417
	合計	4494

の LOC は含まれていないので，GAM のサーバ側の処理についての LOC だと捉えるべきだと考える．View Class の LOC が含まれていないのは，HTML の LOC は開発規模の指標として適切ではないためで，javascript の LOC が含まれていないのは，実験結果のグラフ表示のために javascript を自動生成する外部ライブラリ (SVGware) を用いているため，開発規模としての LOC の計測が難しいためである．

第3章　プロジェクト体制と役割分担

この章では，プロジェクト体制と，開発メンバーの役割分担について記述する．

3.1　プロジェクトメンバ

本プロジェクトは，4 人の開発メンバと 1 名の顧客により構成されている．開発メンバの構成とそれぞれの主な役割を表 3.1 に表記する．筆者は主にスケジューリング実行機能の開発とチケット記録の管理を担当している．

表 3.1: プロジェクトの開発メンバと主な役割

氏名	役割
中西　陽平 (リーダ)	システム環境の構築と管理 継続的インテグレーション
岡谷　亮 (筆者)	スケジューリング実行機能の開発 チケット記録の管理
栗　克	パラメータ管理・作成機能の開発 グラフ描画機能の開発
人見圭一郎	実験パッケージ管理機能の開発 テスト計画の策定と実行

3.2　開発環境

本プロジェクトの開発環境を図 3.1 に示す．

開発用個人 PC

各開発メンバが実装を行う PC．これらの PC の OS は Windowns8 であるが，開発は仮想ソフトウェアである VMware を用いて仮想 OS の Ubuntu 上で行った．この理由は，サーバ環境が Ubuntu であることと，開発フレームワークである Ruby on Rails は Ubuntu OS が開発しやすいためである．開発コードの管理のために git client をインストールしている．

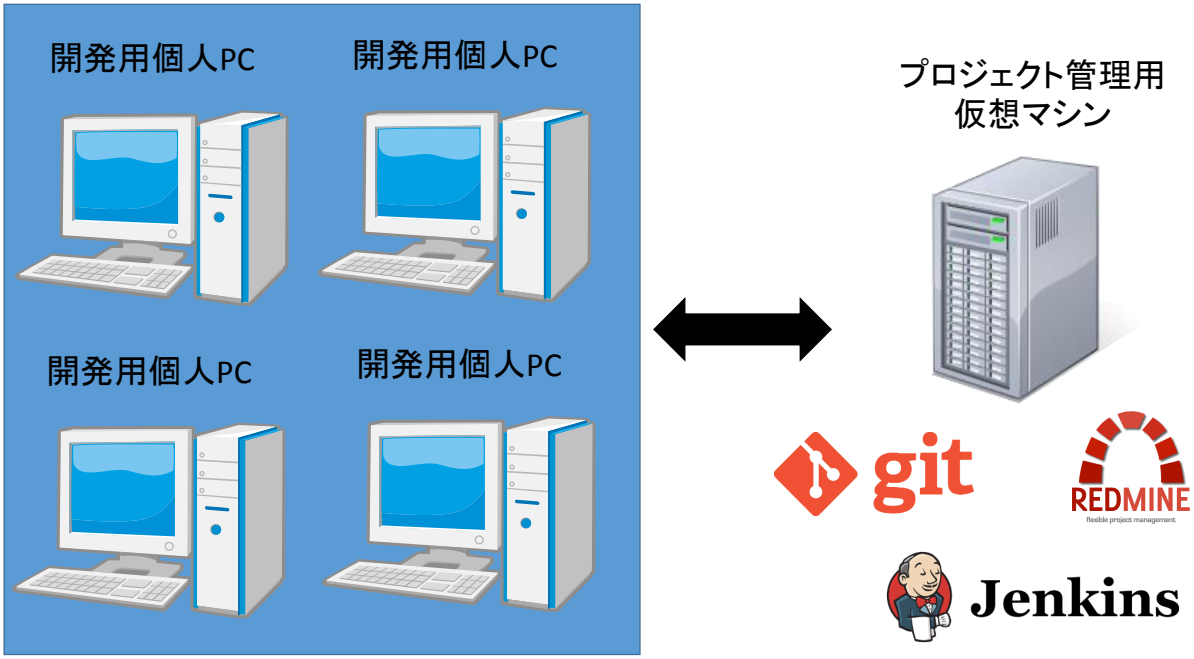


図 3.1: 本プロジェクトの開発環境

プロジェクト管理用仮想マシン

仮想マシンには，以下の 3 つのソフトウェアをインストールして用いる．なお，仮想マシンの管理は中西が行う．

git server

開発用個人 PC で生成されたコードを管理するリポジトリサーバ．git client でプッシュされたコードをここで管理・共有する．

Redmine

プロジェクト支援の Web アプリケーション．開発メンバはこれに Web ブラウザからアクセスして，プロジェクトに関する情報の管理や共有を行う．具体的には，議事録の共有，git server のリポジトリ情報の閲覧，チケット駆動開発の運用，要件やスケジュールの管理などのために用いる．

Jenkins

継続的インテグレーションのためのツール．開発メンバが作成したテストコードは，Jenkins により回帰テストとして継続的に実行される．

3.3 スクラム開発の導入

本プロジェクトの反復型開発は，アジャイルソフトウェア開発手法の一種であるスクラム開発 [7] をモデルにして行う．スクラム開発の導入については，主に中西が担当した．本プロジェクトに採用したスクラム開発の工夫を以下に述べる．

インセプションデッキ
: PMBOK のプロジェクト憲章にあたる．本プロジェクトでは，これを用いてプロジェクトの始めに顧客とチーム間でプロジェクトのビジョンの確認を行った．

ストーリーポイント
: ストーリー (要求) に対して，開発メンバが工数の尺度として割り当てた値．スケジュールを決定する時に開発規模の参考として用いた．

バックログ
: 要求や作業タスクなどを並べたもので，要求管理表や要件定義書にあたる．本プロジェクトでは Redmine のバックログプラグインを用いて行った．

デイリースクラム
: 仕事の前に，各メンバが 15 分程度で簡単な報告会を行う．本プロジェクトでは，平日 14:00-17:00 をコアタイムとして，14:00 から 15 分程度で各自の進捗を口頭で報告するようにした．効果として，他人のタスクの進捗が分かり，問題の対処やタスクのサポートが行いやすくなることが挙げられる．

振り返り
: 各スプリントの終了時にスプリントの問題点などの振り返りを行う．本プロジェクトでは，各イテレーションの終了日に KPT 法を用いた振り返りを行った．

各反復の始めにそのイテレーション内の開発計画を立てて，イテレーションの終了時に達成度の評価と KPT 法を用いた振り返りを行う．プロジェクト管理には，チケット駆動システム開発の支援ツールである Redmine を利用する．

3.4 開発メンバの役割分担

本プロジェクトでの開発メンバの役割分担について述べる．

3.4.1 ドキュメント作成と管理

ドキュメント資料の整理と，顧客ミーティング用資料の作成は主に筆者が担当した．その他の主要なドキュメントの作成担当者について，表 3.2 に記述する．筆者はインセプションデッキ，関連技術調査資料，画面設計書，ユーザマニュアルなどの作成の一部を担当した．

表 3.2: 主なドキュメントとその担当者一覧

ドキュメント名	主な担当者
インセプションデッキ	中西・岡谷
関連技術調査資料	岡谷・栗
要件定義資料	各自開発担当部分
画面設計書	人見・岡谷
ER 図	中西
テスト計画書	人見
ユーザマニュアル	岡谷・栗・人見
導入マニュアル	中西

3.4.2 画面設計

画面遷移の設計は実験管理から実行管理までは主に人見が，実行プロセス管理からは主に筆者が担当した．個別の画面について，画面設計の主な担当者を表 3.3 に記述する．

表 3.3: 画面設計担当者一覧

種別	画面名	設計担当者
実験管理	実験選択画面	人見
	実験パッケージ作成画面	人見
	実験パッケージ詳細情報画面	人見
	実験パッケージ編集画面	人見
	実験削除確認画面	中西
パラメータ管理	パラメータ選択画面	人見
	パラメータ編集画面	人見
	パラメータ削除確認画面	中西
	複数パラメータ一括作成画面	栗
実行管理	通知設定画面	岡谷
	実験実行完了画面	中西
実行プロセス管理	実行実験一覧画面	岡谷
	実行プロセス一覧画面	岡谷
	イベントログ画面	岡谷
グラフ表示	グラフ表示画面	栗

3.5 筆者の役割

筆者の主要な担当は，関連技術の調査，スケジューリング実行機能の開発，要求管理の実践，チケット記録の管理である．これらの内容については第 4 章以降で述べる．各反復で筆者が取り組んでいた作業内容を表 3.4 に示す．

表 3.4: 筆者の各反復の主な担当

反復	内容
準備	インセプションデッキ作成
0	要求分析・開発環境整備
1	関連技術の調査
2	実験実行機能関連の設計
3	実行プロセス管理関連の設計
4	実験実行機能の製造
5	実行プロセス管理機能の製造 メール通知機能の製造
6	実験実行機能関連のバグ修正
7	実行キャンセル機能の製造
8	ユーザマニュアル作成
9	バグ修正

第4章　関連技術の調査

開発を始めるにあたり，GAM に関連する既存の技術やツールの調査を行った．

4.1　関連技術調査の目的

本プロジェクトにおける関連技術の調査は，他の関連技術を利用して顧客の要望を達成できるかということを目的として行った．したがって，GAM の研究としての位置づけを明確にすることよりも，この分野の研究成果として公開されているツールの拡張開発で顧客の課題を容易に解決できるかということを中心に調査した．

4.2　Scientific workflow

本プロジェクトで調査した関連技術の中で， Scientific workflow[8] の分野が最も関連性が高いと考えられる．Scientific workflow の研究分野の目的は，Ludasche ら [9] によると，研究者がデータの共有や計算手法に煩わされず，研究の本来の目的に集中できるようにすることだとしている．つまり，データや情報を分析して新しい科学的知見を得る種類の研究では，研究を行うために研究コミュニティ内での科学計算手法や分析データを共有する必要性がしばしば高まるが，このことは研究の本来の目的ではなく研究を行うための手段であり，科学者は研究の目的以外のことに多くの労力を費やさないことが望ましいとしている，

Scientific workflow そのものは，具体的には科学技術分野で発生する計算やデータ変更のプロセスの流れのこと指している，本プロジェクトの目的で効率化の対象にしている進化的計算実験も scientific workflow の一種だと考えることができるので，Scientific workflow の分野は本プロジェクトと特に関連が高いと考えられる．

この Scientific workflow の研究分野では，ツールを用いて管理や分析，シミュレーション，可視化等を行うことで，Scientific workflow を効率化する試みが行われていて，代表的な成果物として Ludasche らが開発した Kepler[9] がある．

4.3　Kepler

Ludasche らが開発した Kepler は，研究者が研究データを分析・共有することを支援する目的のソフトウェアであり，Kepler を用いることで workflow の作成や実行，共有を行うことができる．Kepler の利用イメージとして，図 4.1 に捕食者と被食者の増減関係の微分法的式モデ

ル化である Lotka-Volterra の方程式を Kepler 上に記述した例を示す．なお，この図は Kepler のソフトウェア中の Demos からの引用である．

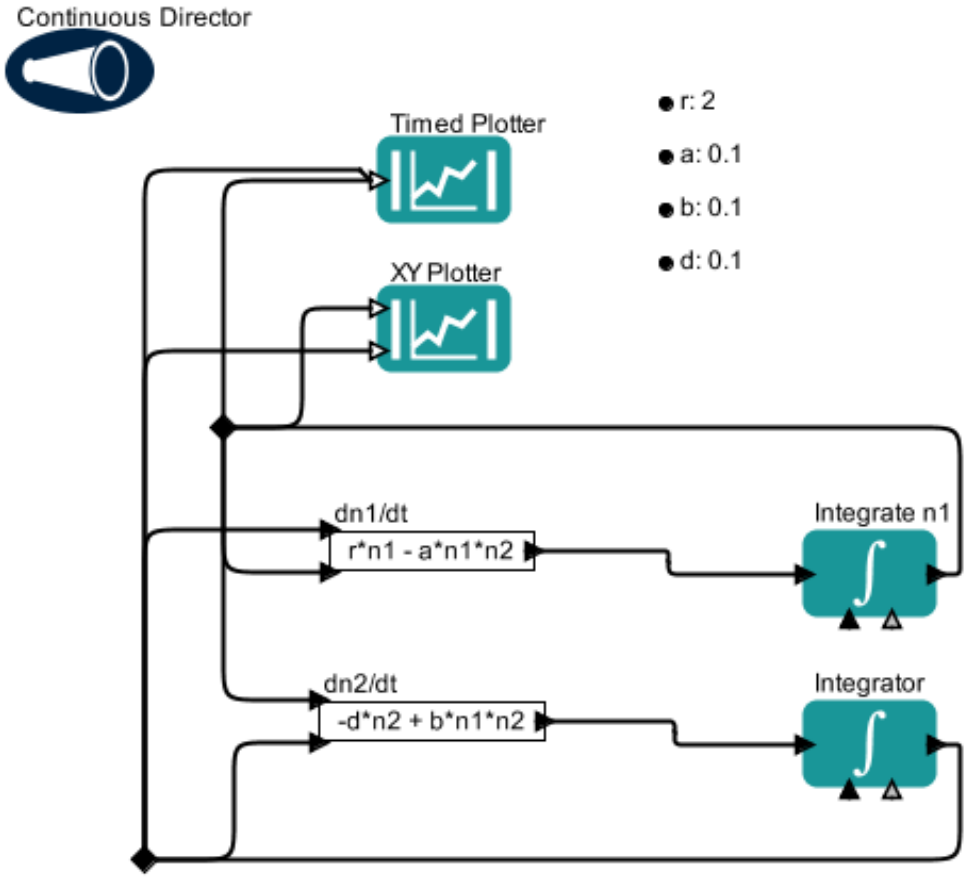


図 4.1: Kepler の Lotka-Volterra の方程式の記述例 (Kepler の Demos から引用)

このモデルを Kepler 上で実行すると，Kepler 上の別ウィンドウで X 値と Y 値のプロット図と，時間ごとの捕食者と被捕食者の増減の推移プロット図が出力される．このように workflow を作成することで，Kepler で計算の実行管理を行うことができる．

4.3.1 Kepler の特徴と GAM との比較

Kepler はクライアント PC にインストールして用いるデスクトップ型アプリケーションであり，Java 言語で実装されていて，オープンソースである．ただし，実行方法，監視方法および分析方法については，Kepler では workflow の定義以外の方法で実現することはあまり想定されていないため，workflow にそれぞれの方法を定義する必要がある．これらの点について，GAM との比較を表 4.1 に示す．

表 4.1: Kepler と新規開発の主要項目の比較

項目	Kepler	GAM
構成	デスクトップアプリケーション	Web アプリケーション
実装言語	Java	Ruby on Rails
対象分野	多分野 (bioinformatics, cheminformatics など)	進化的計算
実行方法	workflow を作成して実行する	実験とパラメータを選択して実行する
監視方法	workflow に監視用アクタ (部品) を加える	メール通知設定を行う，実行状態を確認する
分析方法	workflow を作成して実行結果を確認する	グラフ等の GAM 提供のデータを確認する

4.3.2 Kepler の拡張開発についての考察

Kepler を用いて進化的計算の実験を行う方法を比較して，GAM を用いると以下の利点があると考えられる．

Web アプリケーションの利点

Web アプリケーションをデスクトップアプリケーションと比較したときの利点として，同じ実験を複数のクライアントから管理しやすいこと，クライアントの利用開始時にインストールが不要であること，実行環境がクライアント側の利用環境に依存しないことなどが挙げられる．進化的計算実験は実験に長い時間がかかるため，複数のクライアントから同じ実験を管理しやすいことは大きな利点になると考えられる．

Workflow 変更作業の負担

Kepler では計算の管理は workflow に大きく依存していて，計算の実行管理や監視を実現するためにはこのための workflow を共有または新規作成しなければならない．特に進化的計算の実験では，実験の管理機能として，実験とパラメータを管理する機能，複数のパラメータを一括作成する機能や一度に実行する機能，メール通知機能など，管理のために必要な複数の機能が想定される．これらの管理方法の変更を進化的計算の実験者が workflow の変更操作により行うことは，実験者の作業負担が大きいと考えられる．

一方で，Kepler を用いることの利点としては，以下のことが考えられる．

充実した分析機能を用いた柔軟な分析

Kepler は workflow に組み込むことができる分析手法も豊富であり，workflow を拡張することで結果データの分析を柔軟に行うことができる．GAM では，Kepler のような結果の分析手法の作成・編集機能をクライアントに提供することは想定されていず，サーバ上に保存されたデータについて，要件として定義された分析手法の結果を実験者に提供する以上のことを行うことは困難だと考えられる．

総合すると，進化的計算実験の実行管理と監視については GAM の方が，分析については Kepler の方が使いやすいことが想定される．本プロジェクトでは，Kepler をそのまま本プロ

ジェクトの解決案として用いると，実行や監視の面で実験者にとって運用の負担が大きい問題の影響が大きく，実験結果の分析については，実験結果を GAM からダウンロードすることで実験者はある程度自由に分析ができると判断した．

また，GAM が Web アプリケーションであることは，前に述べた利用環境の依存性の利点だけでなく，本プロジェクトでは Ruby on Rails に熟達したメンバが存在するため，Web アプリケーション開発であれば開発工数の増大リスクが少なくなることが考えられた．Kepler を拡張開発した場合の開発工数についても，実行プロセス監視等の Kepler が持つ機能の拡張や，パラメータの変更設定等の新しい機能の追加が必要であり，GAM と比較して開発工数が低下しないと考えられた．以上の点を考慮し，本プロジェクトは GAM の新規開発の方向で開発を進めた．

第5章　要求管理の実践

この章では，本プロジェクトの要求管理について述べる．

5.1　要求管理

Wiegers[10][11] によれば，要求は開発者が何を実装するべきかの仕様，システムがどのように動作するべきかの記述，システムの属性や品質の記述，システム開発方法への制約など，様々な種類の情報が含まれている．要求の誤りは顧客の期待とシステムの実装との間の違いを生み出し，その修正のためにプロジェクトに大きな手戻りを引き起こすことがある．

Wiegers は，要求アナリストは要求を引き出し，分析，仕様作成，妥当性確認のアクティビティを繰り返すことで要求の記述を洗練させ，要求管理により合意済みの要求に対する変更を管理すべきだとしている．本プロジェクトでは，筆者は主に要求の分析や使用作成の部分を担当して要求の作成や整理を行い，要求管理を要求管理表やプロダクトバックログを用いて行った．

5.2　要求の分類

顧客ミーティングで引き出した様々な要求のレベルを整理して要求仕様として記述することで，要求仕様の品質が高まり，要求仕様の本当の意味とシステムの実装との間の違いが生じにくくすることができると考えられる．例えば,「進化的計算実験の実行時に，進化的計算のパラメータを選択できること」という要求は，顧客にとっては重要であるが，システム開発者はこの要求を直接実装することはできない要求である．逆に,「パラメータファイルをチェック入力して Next ボタンを押すと，次の画面にチェック入力情報が保持されること」という要求は，システム開発者はこの要求に従って直接設計や実装を行うことができるが，顧客にとって直接的な恩恵がない要求である．

Wiegers は要求の分類方法として，顧客の要求を 8 種類に分けているが，特にその大きなくくりとして「ビジネス要求」と「ユーザ要求」,「機能要求」の大きく分けて 3 つのレベルに分類している．

ビジネス要求
: 「ビジネス要求」はシステムのビジョンとしての要求である．例えば，GAM システムのビジネス要求は「実験者が進化的計算の実験を行う時の負担が減ること」と記述することができる．

ユーザ要求

「ユーザ要求」はユーザがシステムを用いて何ができるかについての要求である．ユースケースやユーザストーリーなどを用いてユーザ要求を表現することができる．例えば，「Web ブラウザ上でリモートサーバ上の実験を実行できる」という要求が考えられる．

機能要求

「機能要求」はシステムの振る舞いとしてできなければならないことの要求である．例えば上記のユーザ要求は，「ユーザが実行ボタンを押すと DB に実行プロセスの情報が保存される」と「管理プロセスが DB に保存されている計算プロセスを実行する」などの複数の機能要求のレベルに分けて記述することができる．

5.3 要求分類の統一

本プロジェクトでは，できるだけ要求一覧をユーザ要求のレベルで管理し，個々のタスクを機能要求のレベルで管理するようにした．これにより，要求一覧の優先度を顧客と相談して決める際や，イテレーション計画でメンバのタスクを分配する際に，要求内容の重複や漏れができるだけ生じないようにした．要求一覧の作成の段階では機能要求のレベルまで分けて考えることはせず，各実装担当者の裁量で機能を実装した．本プロジェクトの主要なユーザ要求を表 5.1 に示す．

表 5.1: GAM の主要なユーザ要求一覧

種別	ユーザ要求名
実験管理	実験の作成
	実験ファイルのアップロード
	実験情報の編集
	実験の削除
パラメータ管理	パラメータの作成
	パラメータの編集
	パラメータのアップロード
	複数パラメータ一括作成
	パラメータのコメント表示
実験実行	実験の実行
	メール通知
	通知設定のデフォルト登録
	実行キャンセル
	イベントログの閲覧
可視化	実行結果のグラフ表示
	実験の進捗表示

5.4 要求管理ツール

本プロジェクトは要求管理を行うために，プロジェクトの初期は要求管理表を用いて要求を仕様記述化することを試みた．しかし，本プロジェクトのスクラム開発手法の導入に伴い，要求管理表を用いるよりも，プロダクトバックログを用いる方法のほうが要求管理を行いやすいと考えたため，後にプロダクトバッグログを用いる方法に代替した．

5.4.1 要求管理表を用いる方法

本論文での要求管理表は，Excel 等の表計算ソフトウェアに記述した要求仕様のことを言う．要求管理表を用いた要求管理では，まず，googledocs の excel を用いて要求を分類して記述した要求管理表を用いて，それを要求仕様として扱う方法で行った．要求の優先度の設定は，要求管理表では機能の優先度を高・中・低の段階に分けて記述することで行った．

5.4.2 プロダクトバックログを用いる方法

プロダクトバックログは，Redmine のバックログプラグインを用いてストーリーをチケットと対応付けて並べたものである．ストーリーとは，顧客の要求仕様を簡潔に記したものを言う．プロダクトバックログでは上から優先度の高い順番で並べることで行う．プロダクトバックログは実装するストーリーの優先度が高い順番に上から並んでいるので，

図 5.1 は，本プロジェクトの完了時点のプロダクトバックログである．すなわち，本プロジェクトで要求として挙がったが，実現されなかった要求の一覧である．

5.4.3 要求管理表と比較したプロダクトバックログの利点

要求管理表を用いる方法とプロダクトバックログを用いる方法を比較して，プロダクトバックログを用いる方法は以下の 3 つの利点があると感じた．そのため，本プロジェクトでは要求管理ツールにプロダクトバックログを採用する．

チケットとの連携
: プロダクトバックログのストーリーは Redmine のチケットと関連付けられているため，タスクをストーリーの子チケットとして発行することでタスク管理が容易になる点である．

並び替え時の操作性
: プロダクトバックログは実装する優先度が高いストーリーから順番に，上からストーリーが並んでいる．ストーリーや要求の優先度を変更する際には，要求管理表ではテーブル内の優先度の列の内容を書き換えるか，行の切り取り＆貼り付け操作で順番を並び替える必要があるが，プロダクトバックログはブラウザ上の操作でストーリーをドラッグ＆ドロップで並び替えをすることで，楽な操作で優先度を変更することができる．

プロダクトバックログ			完了したスプリントをクローズ	10
419	GAM	Python製ライブラリを用いて実験の結果を出力する機能	新規	
326	GAM	一連のパラメータの概要をmetaファイルにより表示する機能	新規	
330	GAM	メール通知の時間間隔を設定できる機能	新規	
734	GAM	最大実行時間の閾値を設定できる機能	新規	
86	GAM	マシンリソースを表示する機能	新規	
279	GAM	パラメータファイルの項目を画面上で追加、削除する機能	新規	5.0
275	GAM	データファイルの追加アップロード機能	新規	
336	GAM	リアルタイムな適応度グラフ表示機能	新規	
271	GAM	実験パッケージのダウンロード機能	新規	3.0
272	GAM	パラメータファイルのダウンロード機能	新規	2.0
87	GAM	スナップショット機能	新規	
84	GAM	個体の木構造グラフを表示する機能	新規	
329	GAM	EventLog画面のフィルタリング機能	新規	
273	GAM	大容量ファイルをアップロードできる機能	新規	
328	GAM	GAMシステムのDebianパッケージ化	新規	
83	GAM	実験結果をANOVAする機能	新規	
85	GAM	実験結果の統計分析機能	新規	
197	GAM	実験パッケージのディレクトリ構成を表示する機能	新規	
338	GAM	実験を並列実行する機能	新規	
327	GAM	サーバを直接操作できる設計	新規	
525	GAM	実験パッケージを削除する機能の修正	新規	

図 5.1: プロダクトバックログ

プロジェクト管理ツールとの統合

要求管理表を用いる方法では，要求管理表は googledocs で作成されている．一方で，プロダクトバックログは Redmine 上で閲覧や編集ができる．本プロジェクトではチケット駆動開発を採用していて，開発メンバーは Redmine のかんばん (図 7.1) に頻繁にアクセスするため，プロダクトバックログの方が開発メンバにとって閲覧しやすい．

5.5 要求の変更管理

本プロジェクトでは，プロダクトバックログ上で要求の変更管理を行う．具体的には，非定期に行う顧客との要求の見直しプロセスで，開発チームはプロダクトバックログを顧客に見せて，現在のプロジェクトの進捗と今後の開発予定を顧客と確認する．そして，優先的に実装してほしいストーリーや，本プロジェクトで達成できるストーリーについて顧客と話し合い，ブラウザ上の操作でプロダクトバックログ内のストーリーをその場で並び替えたり追加したりすることにより調整する．この方法で要求の変更管理を行うことで，実装するストーリーについて反復的に顧客との合意形成を行った．

第6章　スケジューリング実行機能の開発

この章では，筆者が主に開発を担当したスケジューリング実行機能について述べる．

6.1　筆者の開発担当範囲

筆者の主な開発担当はスケジューリング実行機能であるが，それに関連する実行プロセス管理機能とメール通知機能の一部も担当している．筆者はこれらの機能の要件定義や，スケジューリング実行機能の開発，メール送信機能の監視プロセスの処理への組み込みの部分，実行プロセス管理機能の進捗表示以外の部分の開発も担当する．

スケジューリング実行機能
: ブラウザ上で実行操作が行われた実験パッケージを実際にサーバ上で実行する機能．6.2 で詳細に説明する．

実行プロセス管理機能
: 実行中や実行待ち状態の進化的計算実験やその計算プロセスの情報を GAM 上に表示する機能と，それらの実験の実行キャンセルを行うことができる機能．6.3 で詳細に説明する．

メール通知機能
: 進化的計算の実行プロセスの状態を，GAM に登録されたメールアドレスにメール通知する機能．送信されるメールの種類には，例えば，実験の開始や終了，エラー終了がある．6.4 で詳細に説明する．

6.2　スケジューリング実行機能

本節では，スケジューリング実行機能の要件と実現方法について述べる．

6.2.1　スケジューリング実行機能の要件

1.3 で述べたように，進化的計算の実験では，パラメータ (例えば突然変異率など) の適切な設定値を調査するためにプログラムを複数回実行して結果を比較する．そのため，GAM にアップロードされた実験パッケージは，通常は異なるパラメータが設定された複数の実行プロセスを持つ．スケジューリング実行機能の要件を以下に述べる．

- GAM 上の画面操作により予約された実験について，その実験が含んでいる計算プロセスを 1 つずつサーバ上で順次実行する
- 実験および計算プロセスの処理方法は FIFO(First In First Out) とする
- 計算プロセスの実行形式は jar 形式および py 形式とし，それぞれの実行ファイルは必ずパラメータファイルのみを引数にとる
  具体的には，以下の実行形式に対応する．
  - java -jar foo.jar bar.par
  - python foo.py bar.par

  (foo, bar は任意のファイル名)

実行プロセスを順次実行するのは，並列実行の場合は実行管理の設計や実装が困難なためである．なお，計算プロセスの並列実行や順番の並び替えは本プロジェクトでは要件外とする．

6.2.2 スケジューリング実行の実現方法

スケジューリング実行機能の実現方法を図 6.1 に示す．この図について，以下に説明する．

1. 監視プロセスの自動起動
   サーバにインストールされた GAM が初めて起動された時点で，GAM はある新たなプロセスを起動しサーバ上に daemon として常駐させる．このプロセス (以下監視プロセスとする) は，DB の情報を監視して実行すべき進化的計算実験の計算プロセスが存在するかを調べる役割や，実行された進化的計算の計算プロセスの実行状態を監視するなどの役割を持つ．

   監視プロセスの実体は，Ruby on Rails 用のプロセス並列実行ライブラリである delayed_job(https://github.com/collectiveidea/delayed_job) のプロセスであり，DB の delayed_job 用のテーブルに 5 秒に 1 度アクセスすることにより，DB に実行すべき計算プロセスが存在するかを調べている．

2. Web ブラウザ上での進化的計算実験の実行操作
   ユーザが Web ブラウザから GAM を用いて進化的計算実験の実行操作を行う．すると，GAM は実行操作が行われた実験の計算プロセスの実行情報 (実行ファイル ID，実行ファイル名など) をサーバ上の DB に登録する．

3. 進化的計算実験の計算プロセスの実行
   監視プロセスが実行すべき計算プロセスの存在を検知すると，監視プロセスは DB の情報の更新 (実行状態や，実行開始時刻など) を行った後に，その計算プロセスの実行を開始する．その後，監視プロセスは計算プロセスの状態 (実行中，終了，エラー終了) を判

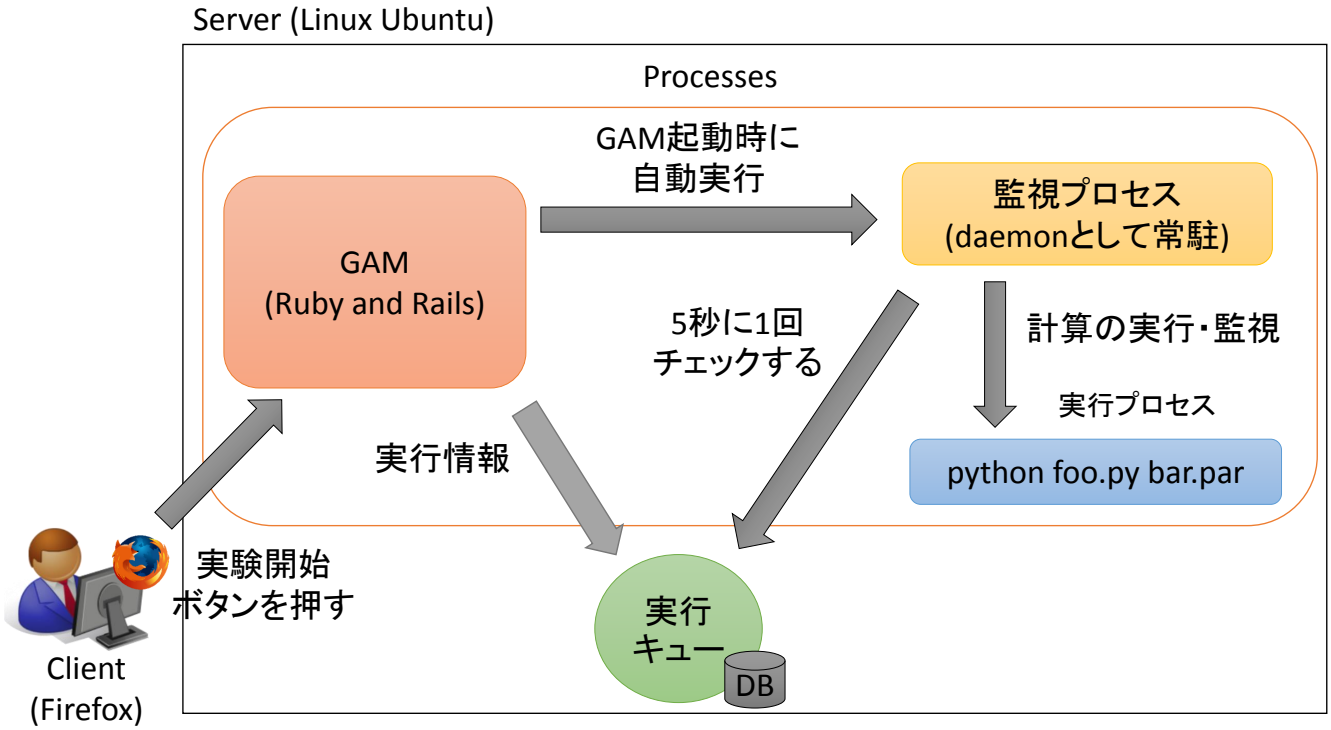


図 6.1: GAM のスケジューリング実行の実現方法

断し，その状態に応じて DB の情報 (実行状態や，実行完了時刻など) の更新を行う．この際に，実行時の通知設定に応じて，DB に登録されているメールアドレスに実行開始や実行完了，実行エラーの通知メールを送信する。

4. 計算プロセス実行完了後の振る舞い
   計算プロセスの実行終了処理が完了すると，監視プロセスは再び DB にアクセスして，次に実行すべき進化的計算実験の計算プロセスが登録されているかどうかを調べる．計算プロセスが存在していればその計算プロセスの実行処理を同様に開始し，存在していなければまた 5 秒に 1 度 DB にアクセスして実行すべき計算プロセスが存在するかを調べる状態に戻る．

6.2.3 非同期処理方法の比較検討

本機能を実現するためには，開発システムはユーザによる Web ブラウザ上での操作処理と非同期に，進化的計算実験の実行と監視を行うことができる必要がある．GAM はこの非同期処理の実現のために delayed_job ライブラリを用いているが，Ruby on Rails でこの非同期処理を実現する方法はそれ以外にも存在する．

本プロジェクトでは，非同期に進化的計算実験の実行・管理処理を行う手段として以下の3つの方法を検討した．

1. Fork 関数
   Ruby の組み込み関数である fork 関数を用いて，実行プロセスを複製する方法である．

2. delayed_job
   GAM が採用した方法である．この方法の特徴は，実行情報が DB に保存される点である．

3. Rescue
   delayed_job と同じく，Ruby on Rails 用の外部プロセス非同期実行ライブラリである．この方法の特徴は，実行情報がメモリに保存される点である．

当初の予定では，Fork 関数を用いる方法で実行プロセス監視の非同期処理を実現する予定であった．しかし，調査と実装の結果，Fork 関数を用いる方法は，Fork 関数が生成したプロセスの終了時に GAM の MySQL への接続が自動的に切断されるため，GAM には適さないことが判明した．また，Rescue と delayed_job との比較では，GAM はサーバのメモリをあまり占有しないほうが良いという観点から，delayed_job の方が適していると判断した．

6.3 実行プロセス管理機能

本節では，実行プロセス管理機能の概要とキャンセル機能について述べる．

6.3.1 実行プロセス管理機能の概要

実行プロセス管理機能は，実行待ちおよび実行中の実験や計算プロセスを確認できる機能である．実行プロセス管理機能の説明として，実行プロセス管理画面を図 6.2 に示す．
実行プロセス管理機能に関連する機能として，GAM は以下の機能を持つ．

- 実行状態表示
  実行待ちおよび実行中の，実験や計算プロセスの状態を確認できる．計算プロセスの状態には，実行中，キャンセル，実行完了，エラー終了の 4 つの状態がある (表 6.1)．これにより，間違えて実験を実行した場合などに取り消すことができる．

- 計算プロセスの進捗表示
  計算プロセスの実行時間 (running time) と，世代 (generation) と反復回数 (repetition) を表示する．これにより，ユーザは実行中の計算プロセスの進捗が確認できる．

- 標準出力・標準エラー出力・出力ファイル表示
  計算プロセスの標準出力と標準エラー出力を画面上に表示する．また，実験の実行ファ

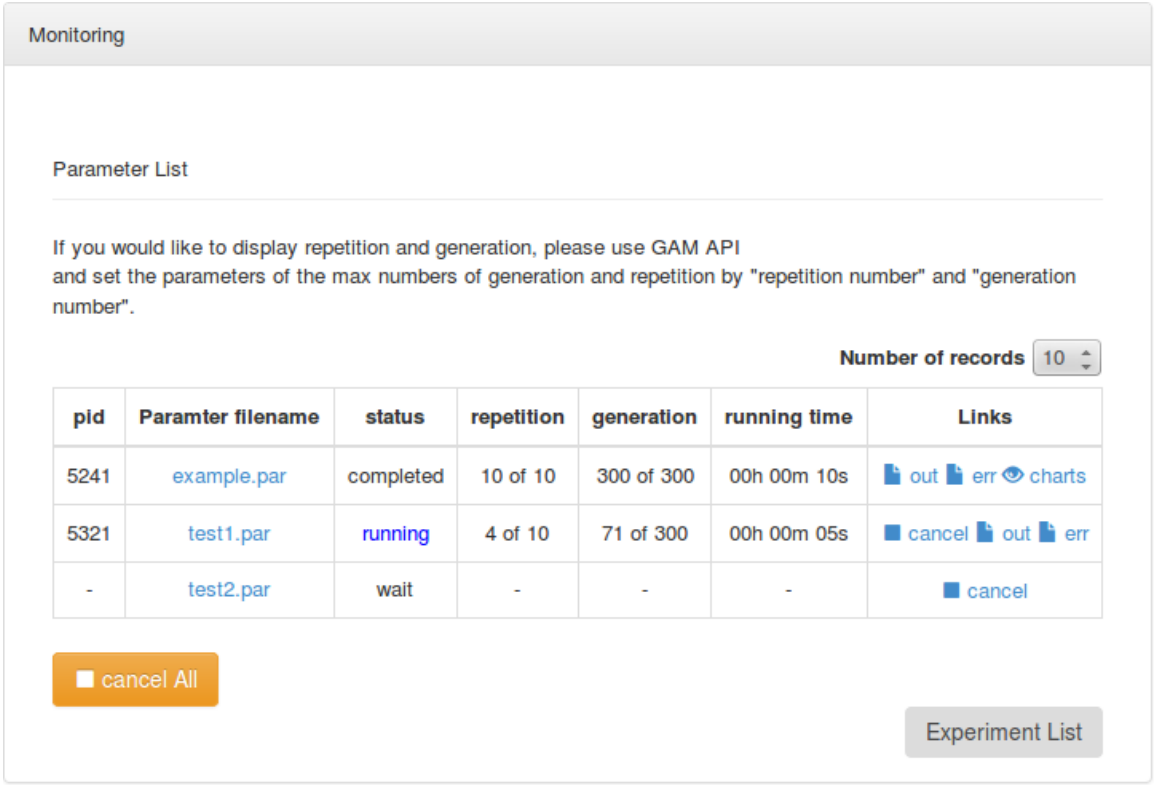


図 6.2: 実行プロセス管理画面

イルが GAM API を用いている場合のみ，ファイル出力結果を確認することができる．これにより，ユーザは実験の実行結果を確認することができ，これを自分で分析することができる．

- 実行キャンセル
  実行待ちおよび実行中の計算プロセスをキャンセルすることができる．また，特定の実験の全計算プロセスをキャンセルすることもできる．

6.3.2 実行キャンセルの実現方法

実行プロセスのキャンセルの実現方法について述べる．まず，対象の計算プロセスの実行状態を canceled に変更する．その後，キャンセル対象の計算プロセスの実行状態によって異なる処理を行う．

- キャンセル対象が実行待ち (wait) の場合
  DB から対象の実行情報を削除し，管理プロセスが DB にアクセスした時にその情報が読み込まれないようにする．

表 6.1: 実行プロセス状態の一覧表

Status	意味
wait	実行待ち
running	実行中
canceled	キャンセル
completed	実行完了
error	エラー終了

- キャンセル対象が実行中 (running) の場合
  実行中のプロセスを Kill して強制終了する．その後，監視プロセスはキャンセル時用の通知メールの送信処理を行い，実行状態の変更を行わない．

以上が特定の計算プロセスについてキャンセルする方法である．特定の実験の全計算プロセスをキャンセルする方法は，その実験に属する計算プロセスについて，実行順番が最後尾のプロセスから以上のキャンセル処理を順番に実行することにより実現している．最後尾のプロセスからキャンセル処理を行うのは，実行中のプロセスをキャンセルした時に，その次のプロセスを実行しようとする非同期の監視プロセスの処理との競合を防ぐためである．

本機能では，Web アプリケーションによるキャンセル処理とは非同期に実行されている監視プロセスと計算プロセスから，処理の途中で割り込みが発生しても問題が発生しないように，DB のデータ更新のタイミングに注意して実装している．

6.4 メール送信要件の定義

実行プロセス状態の定義 (表 6.1) と同様に，メール送信要件を定義した (表 6.2)．GAM は実験パッケージや実行プロセスについて，表中の 9 種類のタイミングでメール通知を行う．

実際にメール通知を行うかどうかは，ユーザはそれぞれのタイミングについて計算実験の実行時に画面上で送信/非送信の設定をすることができる．このメール送信要件の定義に基づいて中西が Ruby on Rails 用のメール送信ライブラリである ActionMailer (http://api.rubyonrails.org/classes/ActionMailer/Base.html) を用いてメール送信処理を実装し，筆者は中西が作成したメール送信処理を監視プロセスの処理に組み込んだ．

6.5 開発上の問題と対処

この節では，筆者が担当した開発で発生した問題のうち，特に筆者が言及する意味があると判断した問題として，非同期実行処理のテストと実行プロセス状態の定義について述べる．

表 6.2: メール送信タイミングの一覧表

種別	Event	Substatus	送信条件
Exeriment	Started	-	実験が開始する
	Completed	-	実験が終了し，全プロセスが正常終了している
	Canceled	-	実験の全プロセスがキャンセルされた
	Error	-	実験が終了し，エラー状態のプロセスが存在する
Process	Started	-	プロセスが開始する
	Completed	-	プロセスが終了し，終了ステータスが 0 である
	Error	Abnormal	プロセスが終了し，終了ステータスが 0 以外である
		Signal	プロセスが signal により終了する
		Unknown	プロセスの終了を GAM が検知できない

6.5.1 非同期実行処理のテストの問題

スケジューリング実行機能やプロセス管理機能では，Web アプリケーションの処理，監視プロセス，計算プロセスがそれぞれ非同期で実行される．ここでの問題は，DB のトランザクションを考慮しなければならないことと，テストが困難であることである．

特にテストが困難であることについては，例えば,「計算プロセスがシグナルにより終了された場合に，シグナルにより終了されたという通知メールが送られること」のテストについて考える．実行プロセスは監視プロセスにより並列実行され，delayed_job が DB にアクセスしたタイミングで実行プロセスが実行されるため，プロセスが実行されるタイミングを制御することが困難である．シグナルは実行プロセスが存在していなければ送ることができないので，プロセスが実行されるまで最低 5 秒待たなければならない．

メール通知機能やキャンセル機能にこのような問題が続発したので，回帰テストに非常に長い時間がかかるようになることを懸念し，非同期実行に関係する処理の一部は手動で操作を再現して動作を確認した．

6.5.2 実行プロセス状態の定義について

DB のデータモデルを作成する際に，始めは error を複数の種類に分けていた．error の種類を実行ステータスに持たせるように設計を変更した．後に PID(Process ID) を表示してほしいという要求や，実行ステータスを保存していたほうが便利なためである，

また，後で追加された要件で，計算プロセスの実行時間が一定時間 (120 時間) 以上続いている場合は，計算プロセスを強制終了させる機能が発生した．GAM ではこの場合も error 状態として扱ったが，要件定義の段階でこのことを考慮に入れて，実行状態に expired を計画しておけば，ユーザやシステムが error と expired を区別することができ，より良い設計ができたと考える．

第7章　プロジェクト指標の計測と評価

この章では，筆者の役割としてチケット記録によるタスク管理とファンクションポイントの計測について述べる．

7.1　チケット駆動開発

本プロジェクトでは，チケット駆動開発 [12] を取り入れている．チケット駆動開発では，プロジェクトに関するタスクを行う時に必ずチケットを発行し，タスクを終える時にそのチケットを閉じる．これにより，誰がどのタスクを行っているのかが分かるようになり，メンバ間のコミュニケーションが楽になる．さらに，チケットの記録を分析することで，プロジェクトの傾向や問題を把握しやすくなる効果が得られるという．

チケット駆動開発を本プロジェクトで運用するために，プロジェクト管理ソフトウェアである Redmine(http://www.redmine.org/) を用いる．また，Redmine のプラグインである Backlogs(http://www.redminebacklogs.net/) の機能により，イテレーションごとのチケットをタスクボード (かんばん) に可視化する (図 7.1)．各メンバはかんばん上で各自のタスクをチケットとして発行し，タスクの状態に応じてチケットの状態を変更することで，その時点でのチケットの進捗を共有できる．

7.2　チケットの記録方法

チケットの記録は，開発メンバーが Redmine 上でチケットを発行する時とそのチケットを終了する時に，Redmine にチケット情報を入力することで行われる．チケットの発行操作方法には 2 つあり，1 つ目の方法は Redmine 上で新規チケットタブをクリックして発行する方法であり，もう 1 つの方法はかんばん (図 7.1) 上にタスクを追加することで発行する方法である．

本プロジェクトでチケットに記録する情報を表 7.1 に示す．チケットが属する工程分類は，かんばん上で発行するチケットについてはトラッカーをタスクと入力する制約があるため，チケットのトラッカーの項目またはカテゴリの項目に入力した．

7.2.1　チケットの工程別分類

本プロジェクトでは，全ての各チケットについて，どの工程の作業に当たるのかを分類した．この本プロジェクトで分類したタスクの工程の分類を表 7.2 に示す．なお，チケット記録

図 7.1: イテレーション 9 のかんばんの一部

時に工程分類情報を入力するかは任意なので，この情報が入力されていないチケットについては，筆者がチケットを集計する時に筆者の判断で分類した．

7.3　チケット記録の集計と比較分析

本プロジェクトでは，チケット記録を集計して，その結果からいくつかの分析を行った．本プロジェクトの完了時の総チケット数は 500，総作業時間は約 1360 時間となった．チケット記録を評価するために，ミーティング時間とテスト時間を評価を対象とした．

表 7.1: チケット記録情報

記録情報	必須か
チケット名	必須
トラッカーまたはカテゴリ (工程分類)	任意
説明	任意
担当者	必須 (新規作成時を除く)
予定工数	任意
作業時間	必須

表 7.2: チケットの工程分類一覧

工程	内容
調査・分析	設計方針を決めるための調査
設計	画面設計やデータベース設計など
実装	コーディング
テスト	単体・結合・総合テスト
バグ	バグチケットの対応
ミーティング	チーム・顧客ミーティング
報告書・論文	報告書執筆・発表スライド作成
環境設定	開発ツールのインストールや設定

7.3.1 ミーティング時間

本プロジェクトに占めるミーティング時間の割合について，10 月 15 日時点での工程別作業時間の割合を図 7.2 に示す．

分析の目的

著者が去年経験した PBL では，ミーティングで決定事項がなかなか決定せず，ミーティング時間内が大幅に圧迫して各個人の作業時間が取れなくなる問題が発生していた．この調査の目的は，本プロジェクトではミーティングの時間をチケットに記録することで，ミーティング時間の超過問題を定量的に把握できるようにすることであった．

分析結果

本プロジェクトではミーティング時間が全体の時間に占める割合は約 31%であり，異常な数値ではないと考えられる．また，実際にミーティングにかかる時間がプロジェクトの進行を妨げているといった議論や認識は本プロジェクトでは特に発生しなかった．著者が昨年経験したミーティング超過の問題は，ミーティングの方法やメンバのミーティングへの態度についての固有の問題であったと考えられる．

7.3.2 テスト時間の比較

プロジェクトのテスト工程に占める作業時間の割合について，本プロジェクトの 10 月 15 日時点での開発工程の作業時間の割合の円グラフと，IPA 刊行のプロジェクト白書 2012-2013 年版 [13] のプロジェクト統計の開発工程の工期の割合の円グラフとの比較を図 7.3 に示す．

比較の目的

本プロジェクトで，テストのために時間をかけることで新しい機能の開発に割く時間が減っていると感じたことから，プロジェクトの統計を比較して，テスト工程にかける時間の妥当性について調査した．

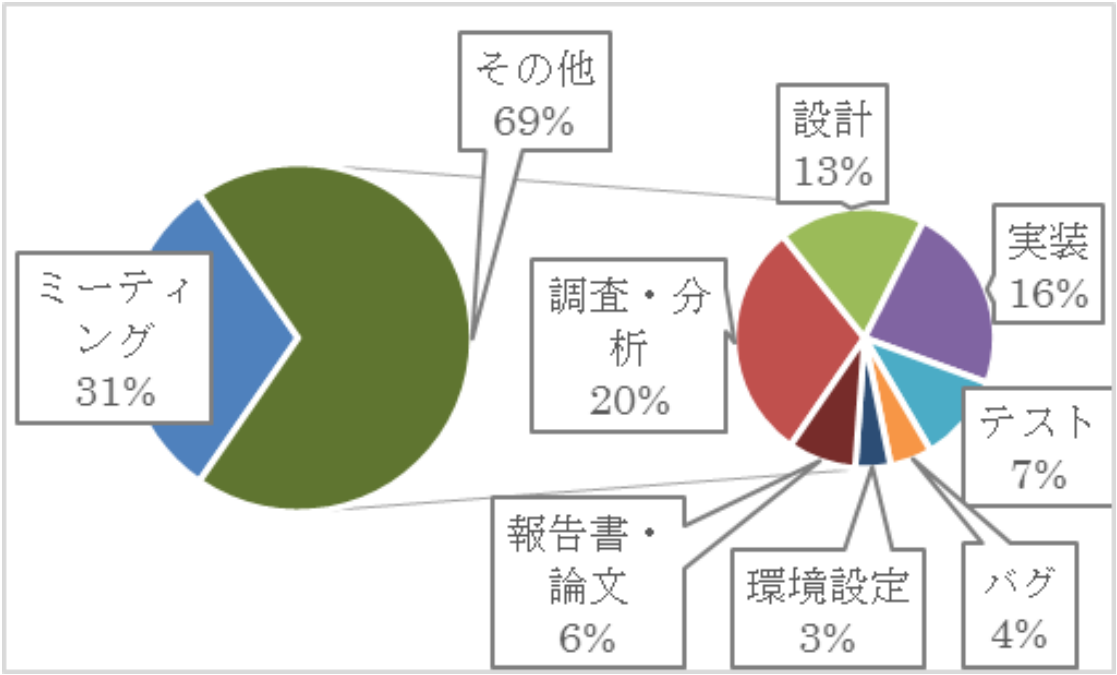


図 7.2: 全作業時間に占めるミーティング時間の割合 (10 月 15 日時点)

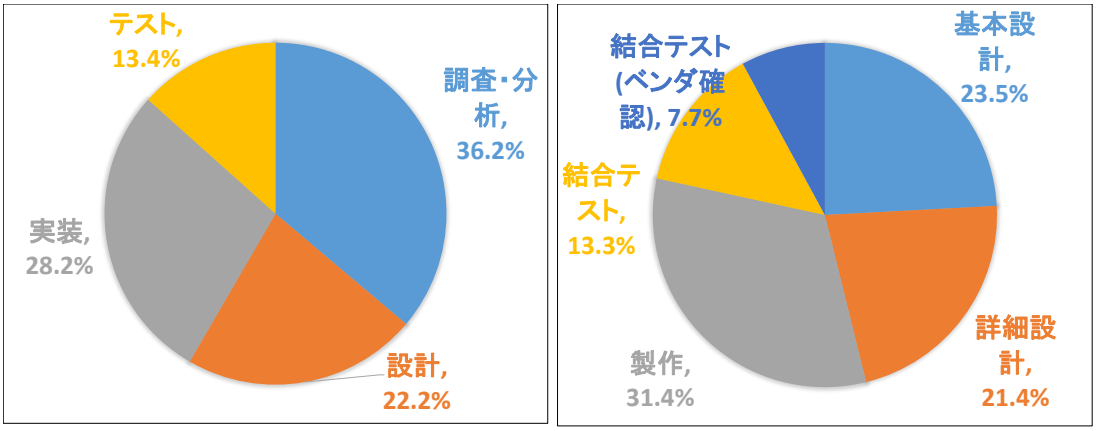
分析結果

プロジェクトの統計データでは結合テストに要しているテスト時間の割合は 21.0%で，本プロジェクトでは単体・結合テストを含めて 13.5%であった．したがって，今後プロジェクトの終盤にテスト工程が増加することを見越しても，テスト工程に過剰な時間を費やしているとは言えないことが分かった．

注意すべき点として，本プロジェクトのデータはプロジェクト途中時点に表 7.2 の分類で作業時間単位で計測したデータであるのに対して，プロジェクト白書では開発工程を基本設計，詳細設計，製造，結合テスト，結合テスト (ベンダ確認) の開発 5 工程で分類していて，プロジェクト終了後に工期単位で計測したデータである．したがって，単純に数値だけの比較で読み取れるのは，あくまでも工程にかける時間の比率の傾向である．

7.4　チケット記録の評価による効果

これらのチケット記録の統計データとの比較は，比較する項目や単位にずれがあり，必ずしも意味がある比較だとは言えず，プロジェクトに劇的な効率の改善をもたらすことはなかったが，全プロジェクトの中の本プロジェクトの傾向が把握できたことでプロジェクトの安心感を得ることができた旨のコメントをプロジェクトメンバから得た．チケット記録を比較するプロジェクト統計での計測方法に合わせて記録していれば，より適切にプロジェクトの統



左図:
本プロジェクトの工程別
作業時間の比率(10月15日時点)

右図:
IPAプロジェクト白書2012-2013
工程別の実績月数の比率の中央値
(新規開発,400FP未満, N=30)

図 7.3: 開発 5 工程の比較 (10 月 15 日時点)

計データの中で自分が携わっているプロジェクトの立ち位置を把握できたと考える.

7.5 ファンクションポイント法

ファンクションポイント (Function Point，FP) 法は，1979 年に Allan が提唱したソフトウェア規模の計測手法で，これを用いることでソフトウェアの規模を定量化することができる [14]. ファンクションポイントの計測手法には IFPUG 法，MKII 法，NESMA 法など複数の手法が提案されているが，現在最も普及しているのは IFPUG(International Function Point Users Group) 法 (http://www.ifpug.org/) である.

本プロジェクトにおける開発規模の見積もりは，主にストーリーポイントの手法を用いてストーリーごとの見積もりを立てることにより行った．しかし，ストーリーポイント法はチームの力量に合った見積もりを簡単に出せる利点がある一方で，指標の尺度が現実の測定値として正しく機能していることの配慮が必要である [15]．また，ストーリーポイントはプロジェクト特有の指標であって，客観的にソフトウェアの規模を評価できる指標ではない．ファンクションポイント法を用いることで，プロジェクトの規模の客観的な値を示すことができる.

7.6 ファンクションポイント法の実践

IFPUG 法では，ソフトウェアの規模をトランザクションファンクション (Transaction Function) とデータファンクション (Data Function) の 2 つの視点で計測し，その 2 つの計測値を足し合

わせたものをソフトウェアのファンクションポイントとする．IFPUG 法では，そのファンクションポイントに調整要員を掛け合わせることで調整済みファンクションポイントを算出するが，未調整ファンクションポイントが用いられることが多い．未調整ファンクションポイントは，調整済みファンクションポイント以上に最終的な規模と強く関係しているという調査もある [16]．筆者は IFPUG 法に従って未調整ファンクションポイントを計測した．

ファンクションポイントを 7 月末時点と 10 月末時点，プロジェクト完了後の 3 度修正を加えながら計測したが，以下に示すのはプロジェクト完了後に計測したファンクションポイントの実績値である．

7.6.1 データファンクションの計測

データのまとまり (ファイル) についてファンクションポイントを割り当てる．本プロジェクトでは，主に DB のテーブルをファイルとしている．データファンクションの計測結果を表 7.3 に示す．

計測方法

データのまとまり (ファイル) は内部論理ファイル (Internal Logical File, ILF) と外部インタフェースファイル (External Interface File. EIF) に種類分けし，それぞれのファイルについてデータ項目 (Data Element Type, DET) とサブクラスの種類 (Record Element Type, RET) を数える．そして，ファイルの種類と DET 数，RET 数から複雑度を決定し，その複雑度からファンクションポイントを決定する．これらの決定には IFPUG 法で定められている対応表を用いる．

ファイルの種類と計測項目

ファイルの種類について，ILF は開発システムのアプリケーション内のファイルであり，ELF は開発システムのアプリケーションでは参照のみを行う他のアプリケーションのファイルである．ファイルの項目について，今回は DET はカラムとして，RET は構造や種類が同じで内容が異なるもの (例: Java の実行ファイルと Python の実行ファイル) とした．

7.6.2 トランザクションファンクションの計測

ユーザにとって意味がある機能の最小単位についてファンクションポイントを割り当てる．本プロジェクトでは，基本的には 1 つのユーザ機能をファンクションとしている．トランザクションファンクションの計測結果を表 7.4 に示す．

計測方法

トランザクションを外部出力 (External Output, EO)，外部入力 (External Input, EI)，外部照会 (External Inquiry, EQ) の 3 種類に分ける．そして，それぞれのトランザクションに

表 7.3: データファンクション

ファイル名	種類	DET(カラム数)	RET(サブクラス種数)	複雑度	FP
実験パッケージ	ILF	9	1	low	7
一括パラメータ	ILF	4	1	low	7
パラメータファイル	ILF	5	1	low	7
実験の実行リスト	ILF	9	1	low	7
実行プロセスリスト	ILF	10	1	low	7
メール通知イベント	ILF	9	1	low	7
デフォルト設定	ILF	5	1	low	7
イベントログ	ILF	7	1	low	7
stdout/err	ELF	1	2	low	5
出力ファイル	ELF	6	1	low	5
実行ファイル	ELF	1	2	low	5
ライブラリファイル	ILF	7	1	low	7
				合計	78

ついてデータ項目 (Data Element Type, DET) と参照ファイル (File Type Reference, FTR) の数を数える．トランザクションの種類と DET 数，FTR 数から複雑度を決定し，その複雑度からファンクションポイントを決定する．これらの決定には，データファンクションの計測と同様に IFPUG 法で定められている対応表を用いる．

トランザクションの種類と計測項目

トランザクションの種類について，3 種それぞれについて説明する．EI はデータベースへの入力・更新などのファンクションで，副次的な EQ が発生してよいが，EO が発生してはいけない．EO はデータベース等の情報に計算等の処理を施して出力するファンクションで，副次的な EQ や EI が発生してよい．EQ は単純な検索処理とその出力のみのファンクションで，EI や EO が発生してはいけない．トランザクションの計測項目について，DET は入出力項目とそのトリガー (ボタンなど) であり，FTR（File Type Reference）はトランザクションで参照するファイルである．

7.7 ファンクションポイントの評価

本プロジェクトのファンクションポイントの計測値は，データファンクションが 78FP，トランザクショナルファンクションが 80FP で，合計 158FP となった．Kermerer らの調査では，熟練したファンクションポイントの計測者は計測誤差を 10%以内に抑えられるが，それが未熟な技術者であると 20-25%程度のばらつきがあるとしている [17]．筆者はファンクションポイ

表 7.4: トランザクションファンクション

トランザクション	種類	DET 数	FTR 数	複雑度	FP
ファイルアップロード	EI	3	4	average	4
圧縮ファイル自動解凍	EQ	1	3	low	3
実験パッケージ選択・削除	EI	2	3	low	3
実験パッケージ作成・変更	EI	1	3	low	3
実験パッケージ情報表示	EQ	1	3	low	3
パラメータファイル選択・削除	EI	2	3	low	3
パラメータファイル作成・編集	EO	2	5	average	5
複数パラメータファイルの一括作成	EO	3	5	average	5
実行設定の選択	EI	2	3	low	3
スケジューリング実行	EI	6	4	average	4
メール通知	EO	5	7	high	7
実行設定のデフォルト登録	EI	1	3	low	3
実験一覧表示	EQ	1	3	low	3
実行プロセス表示	EQ	1	3	low	3
実行結果表示	EQ	4	3	low	3
実行プロセスキャンセル	EI	2	3	low	3
全実行プロセスキャンセル	EI	2	2	low	3
世代ごとのグラフ出力	EO	1	4	low	4
反復ごとのグラフ出力	EO	2	5	low	4
GAM ライブラリのファイル出力	EO	2	3	average	5
イベントログ表示	EQ	2	3	low	3
テーブルのソート・ページング	EQ	1	4	low	3
				合計	80

ントの計測に熟練していないため，GAM の実際のファンクションポイントの値は 118-198FP の範囲内にあるといえる．

ファンクションポイントの生産性について考えると，本プロジェクトの開発に約 1400 時間費やしていると考えると，開発チームの FP 生産性はおよそ 0.113FP/人時である．ソフトウェア開発白書 2012-2013[5] のデータでは，400FP 未満の新規開発プロジェクトの FP 生産性の中央値は 0.100FP/人時であり，月当たりの要員数が 5 人未満の新規開発プロジェクトの FP 生産性の中央値は 0.161FP/人時 (N=113) である．したがって，単純にこの数値を比較すると，筆者の開発チームはメンバ数が 4 人のプロジェクトとしては開発生産性は低いが．全体として見た場合ではやや高い結果と言える．

第8章　プロジェクトの評価と今後の展望

8.1　プロジェクトの評価

顧客に GAM を納品した後に，本プロジェクトについての評価アンケートを作成し、顧客からの回答を得た。この評価アンケートの内容と結果を表 8.1 に示す．なお，目標の中で実験の実行管理については，点数の評価が難しいと考え，評価対象をパラメータ作成の支援とした．

表 8.1: プロジェクトの目的・目標達成評価アンケートの内容と結果

種別	質問	点数 (5 点満点)
目標	GAM は「パラメータ作成の支援」を達成していますか？	4
	GAM は「実行プロセスの状態の通知」を達成していますか？	5
	GAM は「実験結果の分析の支援」を達成していますか？	2
目的	GAM は「EC の実験を行う時の手間を軽減する」を達成していますか？	4

目標の評価のうち，パラメータ作成の支援と実行プロセスの状態の通知については，それぞれ 5 点満点中 4 点と 5 点と高い評価が得られたが，分析の支援についての評価は 2 点という結果になった．分析支援についての質問の自由記述欄には「(GAM は) 実験結果を単にグラフ表示しているだけで、分析まではまだ行っていません。」とのコメントがあった．

この結果から，本プロジェクトでは前者 2 つの目標については，GAM の機能として顧客にある程度十分に提供できたと考えられる．分析の支援についても，GAM が進化的計算実験の結果分析機能を持つことで，評価値を向上することができると考えらえる．

全体的に言えば，本プロジェクトの目的達成評価は 5 点満点中 4 点であるので，本プロジェクトの達成評価としては概ね良い結果が得られた．また，目的を達成するための要件定義や機能提供に，顧客と特に大きなすれ違いは発生しなかったと考えられる．

8.2　システムの評価

GAM システムの達成評価アンケートの顧客評価を表 8.2 に示す．この評価アンケートの回答は，点数ではなく日本語の文章の 5 段階評価で得た．なお，この評価アンケートは第 9 回目の反復で顧客が GAM を数日間の期間テストを行った時点での評価であり，顧客が実際の進化的計算実験に運用したときに，また異なる評価が得られる可能性はある．

表 8.2: GAM のシステム達成評価アンケートの内容と結果

実行管理機能	
実験プログラムの管理機能について、使いやすいですか？	とても使いやすい
実験の情報管理にかかる手間が改善されましたか？	とても手間が減った
パラメータファイルの作成機能は使いやすいですか？	どちらともいえない
パラメータファイルの作成にかかる手間は軽減されましたか？	少し手間が軽減された
パラメータファイルを一括作成できる機能は使いやすいですか？	すこし使いやすい
まとめてパラメータファイルを作成する手間は軽減されましたか？	手間が軽減された
実験を実行できる機能は使いやすいですか？	とても使いやすい
コンソールから実行する場合と比べて、手間が軽減されましたか？	少し手間が軽減された
通知機能	
実験の実行状態をメールで通知する機能は使いやすいですか？	とても使いやすい
メール通知により、従来と比べてどの程度改善されたと思いますか？	とても良くなった
実験の実行状態を Web から確認できる機能は使いやすいですか？	とても使いやすい
実行状態の確認により、従来と比べてどの程度改善されたと思いますか？	良くなった
結果の可視化機能	
実験結果のグラフ化機能は使いやすいですか？	どちらともいえない
実験結果のグラフ化機能で表示されるグラフは見やすいですか？	どちらともいえない
実験結果のグラフ化機能で、どの程度手間が軽減されましたか？	少し手間が軽減された
マニュアル	
運用マニュアルから目的の情報を簡単に見つけることができましたか？	どちらともいえない

システム評価アンケートの結果を見ると，通知機能の評価と実行管理機能の評価は高いが，実験結果の可視化機能についての評価はあまり高くないという，プロジェクト評価アンケートの結果と同様の傾向が見られる．これらの結果から，実験結果の可視化・分析機能の充実が今後の課題だと言える．

8.3 今後の展望

本プロジェクトで開発した GAM システムは，いくつかの不十分な点がある．これらについて，GAM システムの今後の展望について述べる．

8.3.1 一般公開と対象ユーザの拡大

顧客は，将来的には全世界の進化的計算の研究者が GAM を利用できることが望ましいと考えている．GAM の対象ユーザは進化的計算実験を頻繁に行う研究者であるため，日本語を

理解できるユーザのみを対象とするのは望ましくないと考えられる．

GAM はその点を考慮し，Web ブラウザ上に表示されるページは全て英語で表記されている．しかし，ユーザマニュアルやインストールマニュアルは日本語版のみしか作成していない．したがって，ユーザマニュアルやインストールマニュアルの英語版を作成して，英語圏のユーザも利用できるようにするのが望ましい．

また，GAM はパッケージ化を行っていないため，GAM をサーバにインストール時の作業が膨大である．このため，GAM のインストールを簡単にサーバ上にインストールできるようにパッケージ化を行い，サーバ管理者の負担を軽減すべきである．

8.3.2 実験データの共有支援

顧客は将来的には GAM を大学の研究室での使用を要望として持っている．GAM を大学の研究室で用いる場合は，複数ユーザの使用が想定する必要がある．誰がどの操作を行ったのかを把握できるようにするためのユーザ管理機能や，特定の進化的計算実験を複数人で共有できる機能が要件に含まれることが考えられる．

ユーザ管理機能を実現するために，ユーザのログイン機能や，ユーザ別の実行権限を設定できる機能が必要だと考えられる．進化的計算実験の共有のために，実験パッケージや実験結果のダウンロード機能があれば研究者や学生間の共有が促進すると考えられる．

8.3.3 実験結果の分析支援

顧客は実験結果の分析機能について要望を持っていたが，本プロジェクトでは工数の大きさからこの機能は実現できず，結果のグラフ表示機能の実現にとどまった．しかし，分析機能が備わっていれば，結果の自動分析機能を提供することで，実験結果の分析の手間の問題が解決すると思われる．実装すべき具体的な分析手法については，顧客からは ANOVA(分散分析) が挙がっている．

ただし，この方向性で GAM を発展させるには，現状では実験結果の分析機能について，具体的にどのような分析方法を行えば分析が楽になるのか明確になっていない問題があるように思われる．例えば，ANOVA 以外に実装すべき分析方法として，例えば散布図の生成や線形回帰分析などの統計的分析手法が考えられるが，これらの手法が本当に進化的計算実験の結果分析に役立つかは明確になっていない．

将来のこのプロジェクトの引継ぎなどでこの分析機能の開発を担当するチームは，必ずしも進化的計算についての知識が十分であるとは限らないので，進化的計算の効果的な分析手法の調査のために，分析手法の要件定義の段階で多くの時間が費やされるリスクが想定される．その場合でも，どのような手法が進化的計算にパラメータが与える影響の分析に役に立つのかを，よく検討すべきである．

第9章　まとめ

本プロジェクトは進化的計算の実験における手間を削減することを目的として，進化的計算実験支援システムである Web アプリケーション (GAM) を開発した．GAM は主に実験やパラメータの管理，実験のスケジューリング実行，実行プロセスの管理，実行結果のグラフ表示などの機能を持つ．GAM は「パラメータ作成の支援」や「実行プロセス状態の通知」の目標を達成でき，顧客から全体として概ね良い評価が得られた．

筆者は本プロジェクトで，主に要求管理の実践とスケジューリング実行機能の開発を担当した．要求管理の実践では，要求管理を行いやすいように，顧客の要望をできるだけユーザ要求のレベルに整理してプロダクトバックログに記載した．スケジューリング実行機能の開発では，進化的計算実験を順次実行する機能や実行プロセスを通知する機能，実行プロセスの状態を管理する機能などの開発を行い，実行管理機能や通知機能の一部として提供した．また，プロジェクトの指標値を導出するために，最終的に総チケット数が 500 となったチケット記録を管理し，GAM の最終的なファンクションポイントを 158FP と計測した．

GAM の今後の課題としては，一般公開と対象ユーザの拡大，実験データの共有支援，実験データの分析支援が考えられる．GAM はこれらの課題を解決することで，進化的計算の実験者とってより便利なツールになるだろうと考える．

謝辞

本プロジェクトを進めるにあたり，多くの助言とご指導を頂きました課題担当教員の Claus Aranha 先生，指導教員の高橋伸先生，田中二郎先生をはじめとした IPLAB の先生方，ならびに他教員のみなさまに深く感謝致します．特に，本プロジェクトの顧客であり，プロジェクトに快く協力をして頂いた Claus Aranha 先生には，度重なるミーティングにお付き合い下さり本当にありがとうございました．心より感謝申し上げます．

本プロジェクトのチームメンバである中西陽平君，人見圭一郎君，栗克君には，広くお世話になりました．皆と同じチームで一つのプロジェクトを完了できたことを誇りに思います．

最後に，様々な面でご支援いただきました家族，友人，背後から支えていただきました山戸先生，コンピュータサイエンス専攻事務の方々，大学生活でお世話になった全ての方々に心より感謝申し上げます．

参考文献

[1] システム制御情報学会. 遺伝アルゴリズムと最適化. 朝倉書店, 1998

[2] Claus Aranha, Carlos R.B. Azevedo, Hitoshi Iba. Money in trees: How memes, trees, and isolation can optimize financial portfolios *Information Sciences*, Volume 182, Issue 1, pp. 184-198, 2012.

[3] Rodrigo T. Peresa, Claus Aranha, Carlos E. Pedreira. Optimized Bi-Dimensional Data Projection For Clustering Visualization *Information Sciences*, Volume 232, pp. 104-115, 2013.

[4] 伊庭斉志. 進化論的計算の方法. 東京大学出版, 1999.

[5] J.J. Grefenstette. Optimization of control parameters for genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Vol.16, No.1, pp. 122-128, 1986.

[6] 坂和正敏, 田中雅博. 遺伝的アルゴリズム, 朝倉書店, 1995.

[7] Jonathan Rasmusson. アジャイルサムライ-達人開発者への道- オーム社, 2011.

[8] Adam Barker, Jano van Hemert. Scientific Workflow: A Survey and Research Directions Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science Volume 4967, pp.746-753, 2008.

[9] Ludasche Bertram, et al. Scientific Workflow Management and the KEPLER System, Concurrency and Computation: Practice and Experience 18(10), 2006, pp.1039-1065.

[10] Karl E. Wiegers. 要求開発と要求管理-顧客の声を引き出すには- , 日経 BP ソフトプレス, 2006.

[11] Karl E. Wiegers. ソフトウェア要求-顧客が望むシステムとは- , 日経 BP ソフトプレス, 2003.

[12] 小川 明彦, 阪井 誠. チケット駆動開発, 翔泳社, 2012.

[13] IPA ソフトウェア・エンジニアリング・センター. ソフトウェア開発データ白書 2012-2013, IPA, 2012.

[14] IPA ソフトウェア・エンジニアリング・センター. ソフトウェア開発見積りガイドブック ～IT ユーザとベンダにおける定量的見積りの実現～, オーム社, 2006.

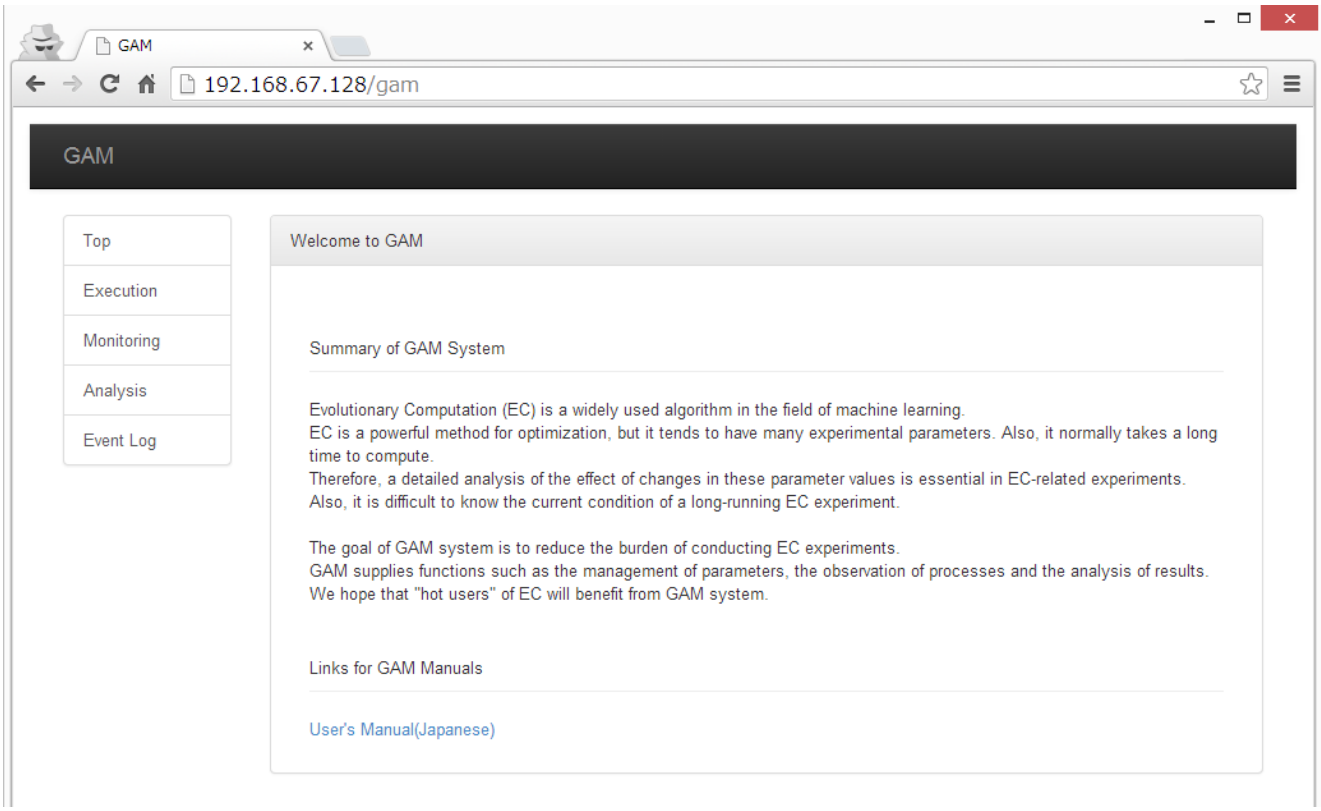
[15] Steve McConnel. ソフトウェア見積り-人月の暗黙知を解き明かす, 日経 BP ソフトプレス, 2006.

[16] John E. Gaffney, Jr., Richard Werling, Estimating Software Size from Counts of Externals, A Generalization of Function Points. Analytical Methods in Software Engineering Economics, 1993, pp 193-203.

[17] Kemerer Chris, Benjamin Porter. Improving the Reliability of Function Point Measurement: An Empirical Study. IEEE Transactions on Software Engineering Accuracy, Vol. 18, No 11, November 1992, Page 1011.

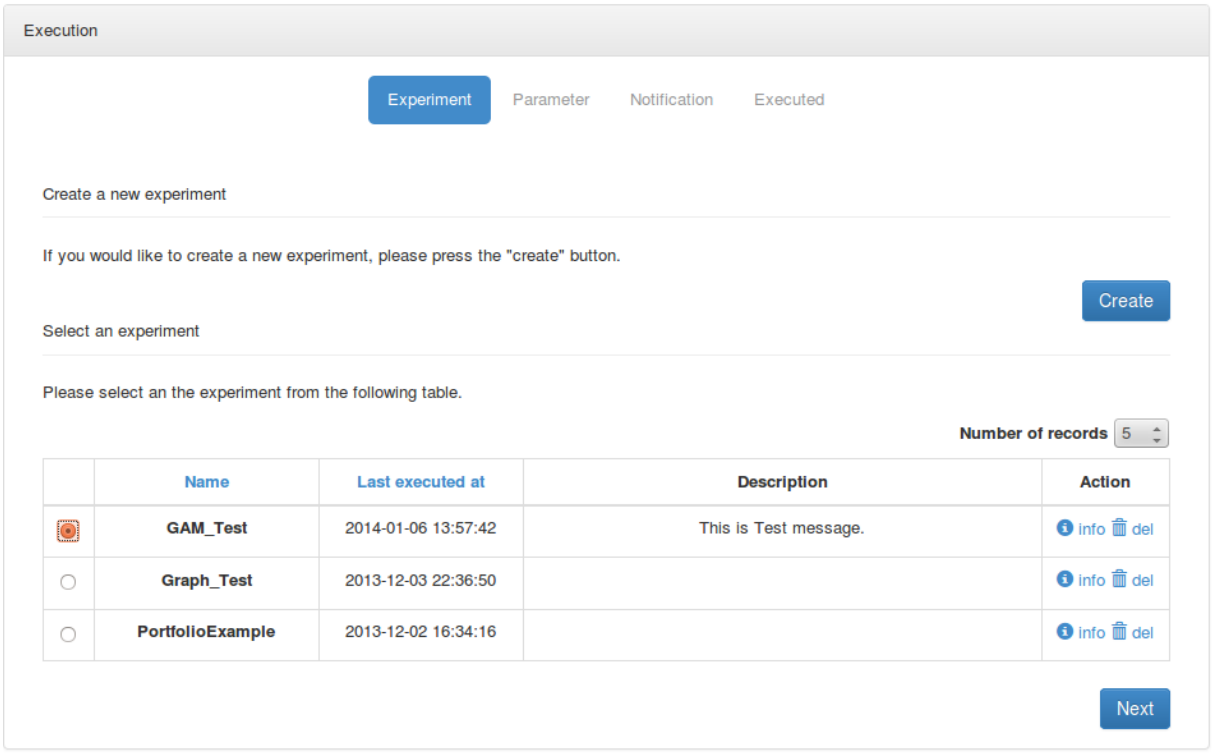
付録 A 画面のスクリーンショット

0. TOP 画面

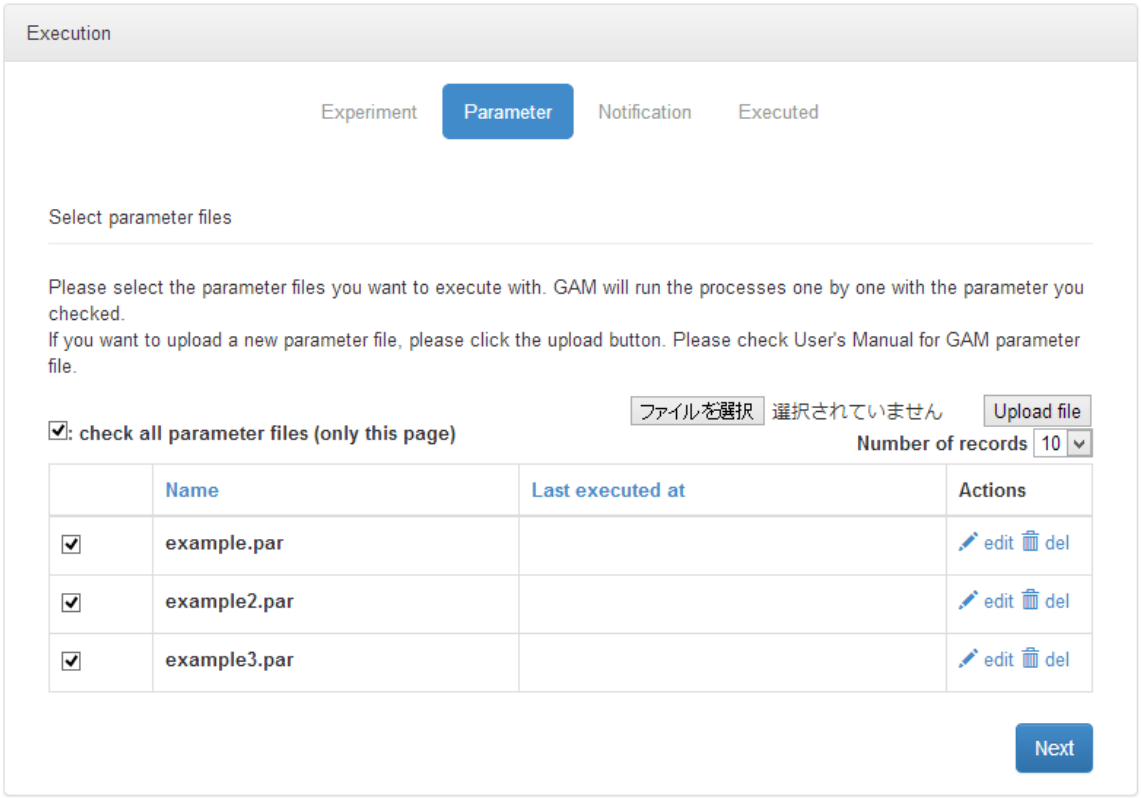


1. Execution

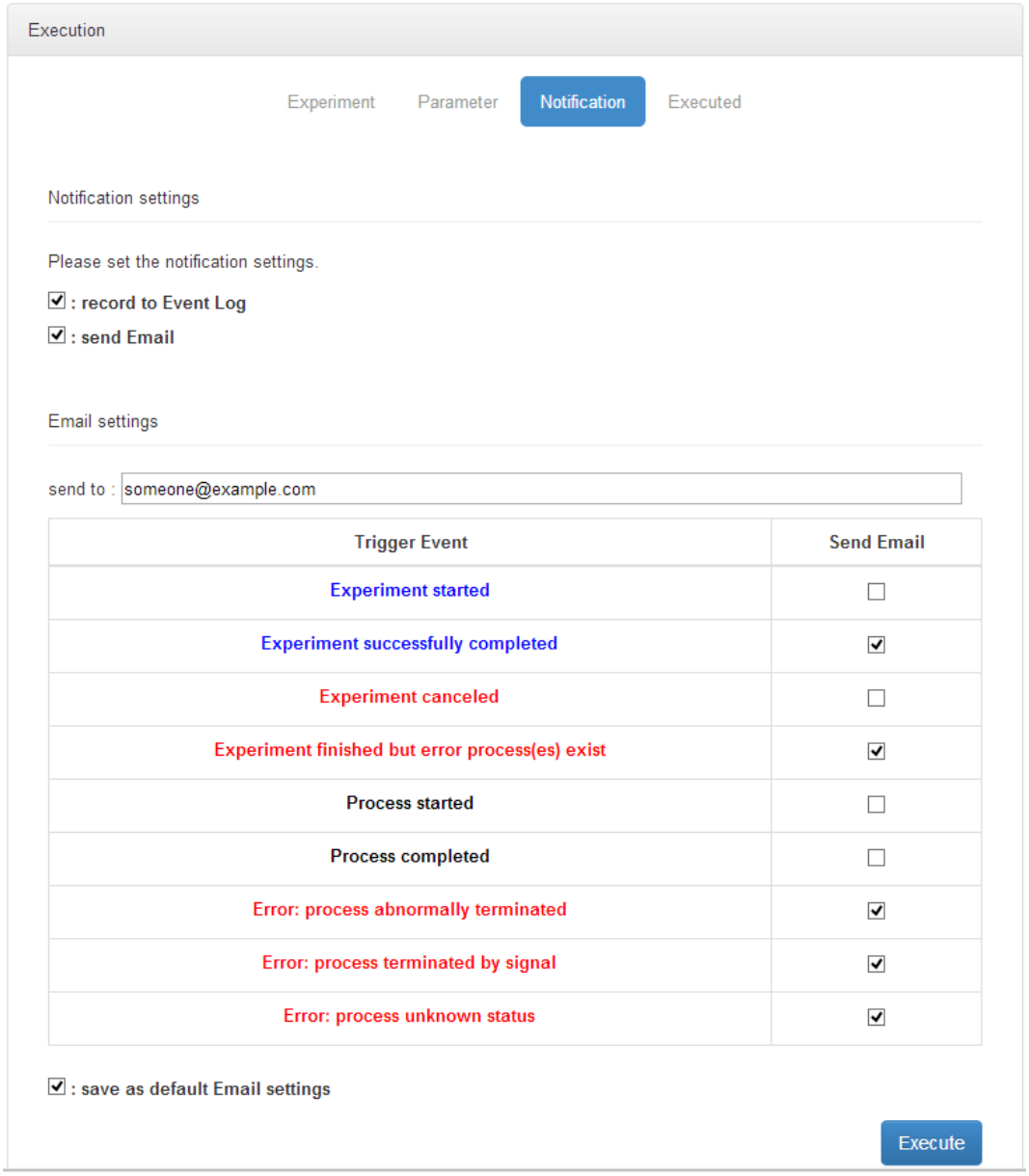
1-1. 実験選択画面



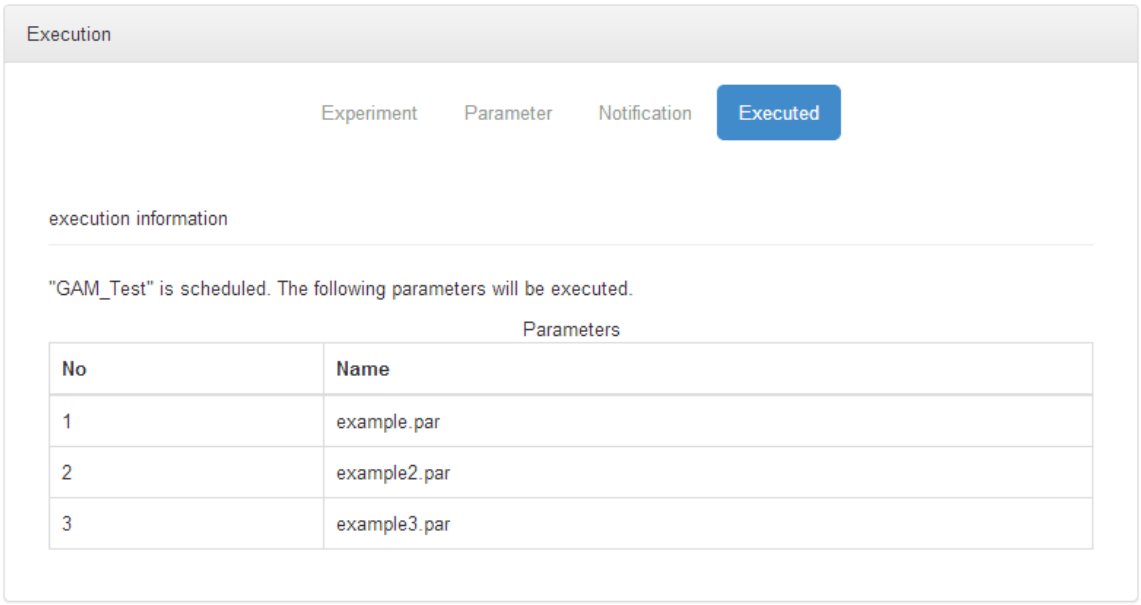
1-2. パラメータ選択画面



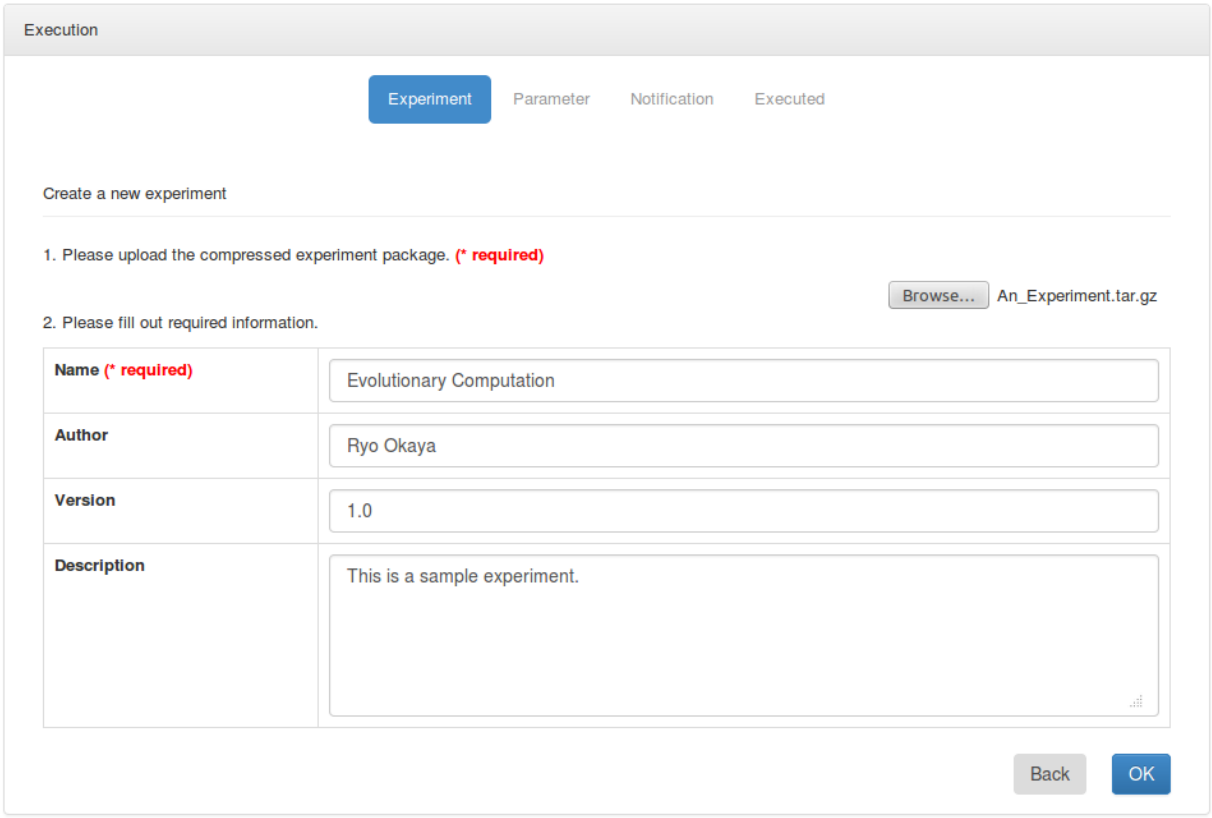
1-3. 通知設定画面


Execution
Experiment
Parameter
Notification
Executed
Notification settings
Please set the notification settings.
: record to Event Log
: send Email
Email settings
send to : someone@example.com
Trigger Event
Send Email
Experiment started
Experiment successfully completed
Experiment canceled
Experiment finished but error process(es) exist
Process started
Process completed
Error: process abnormally terminated
Error: process terminated by signal
Error: process unknown status
: save as default Email settings
Execute

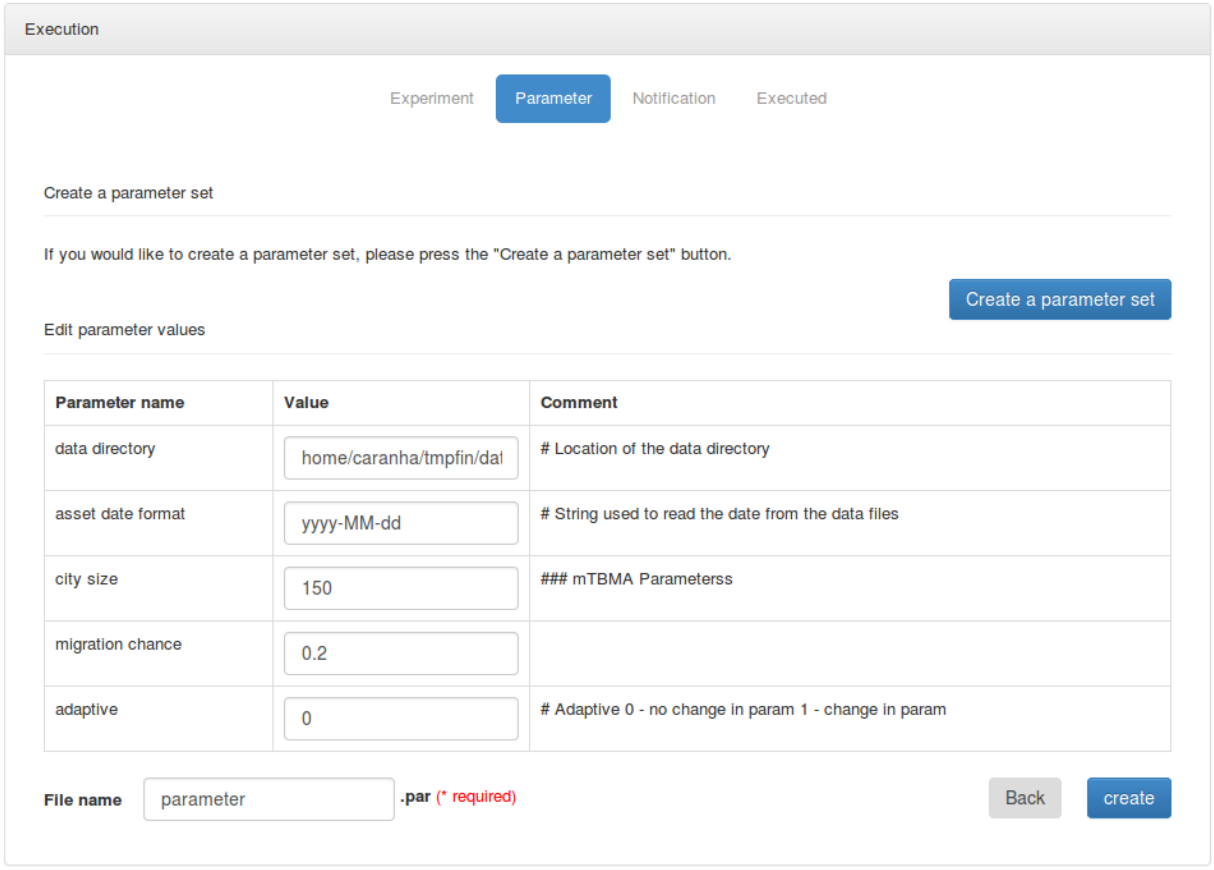
1-4. 実行完了画面


Execution
Experiment
Parameter
Notification
Executed
execution information
"GAM_Test" is scheduled. The following parameters will be executed.
Parameters
No
Name
1
example.par
2
example2.par
3
example3.par

1-1-1. 実験作成画面

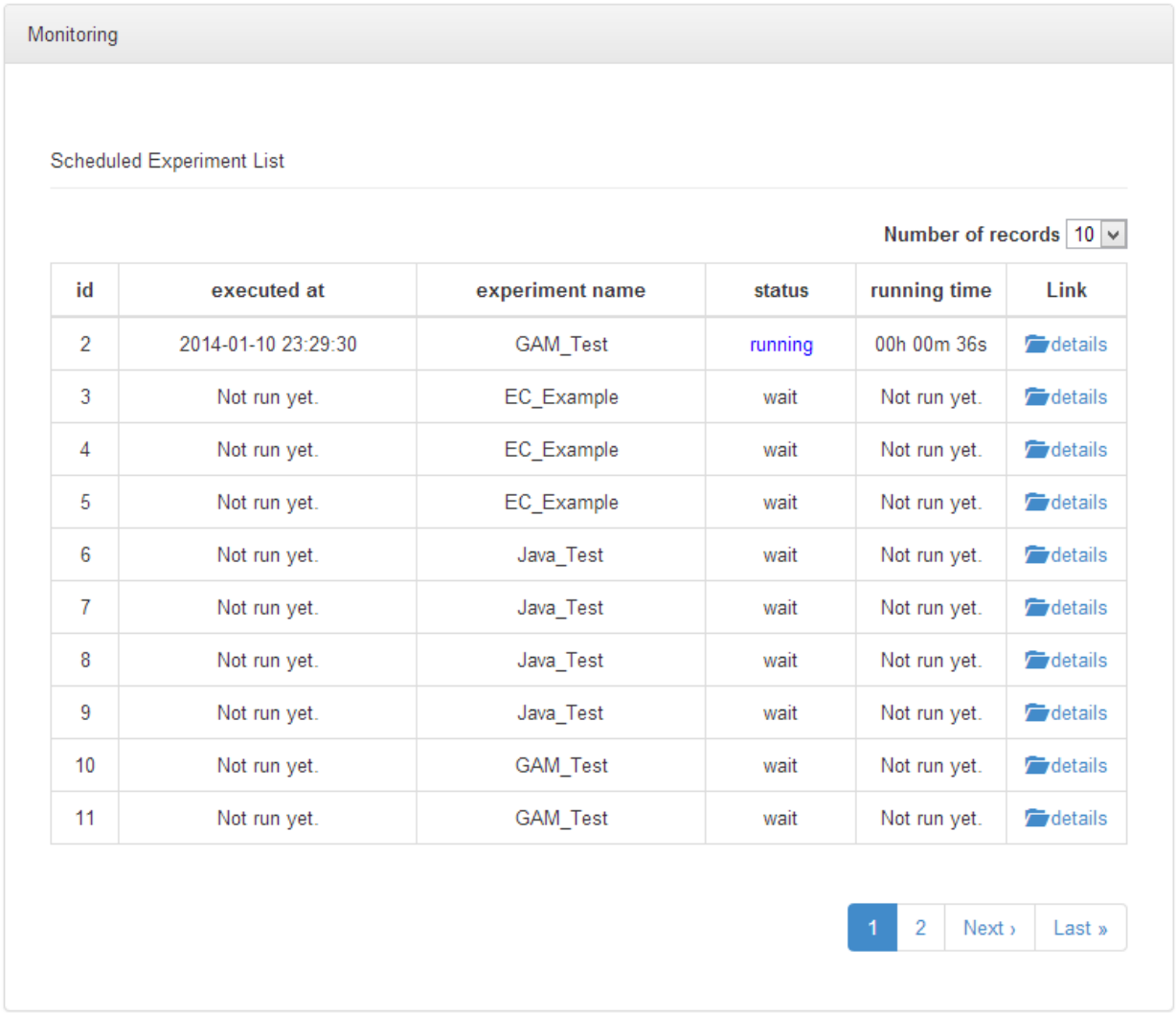

Execution
Experiment
Parameter
Notification
Executed
Create a new experiment
1. Please upload the compressed experiment package. (* required)
Browse...
An_Experiment.tar.gz
2. Please fill out required information.
Name (* required)
Evolutionary Computation
Author
Ryo Okaya
Version
1.0
Description
This is a sample experiment.
Back
OK

1-2-1. パラメータ編集画面

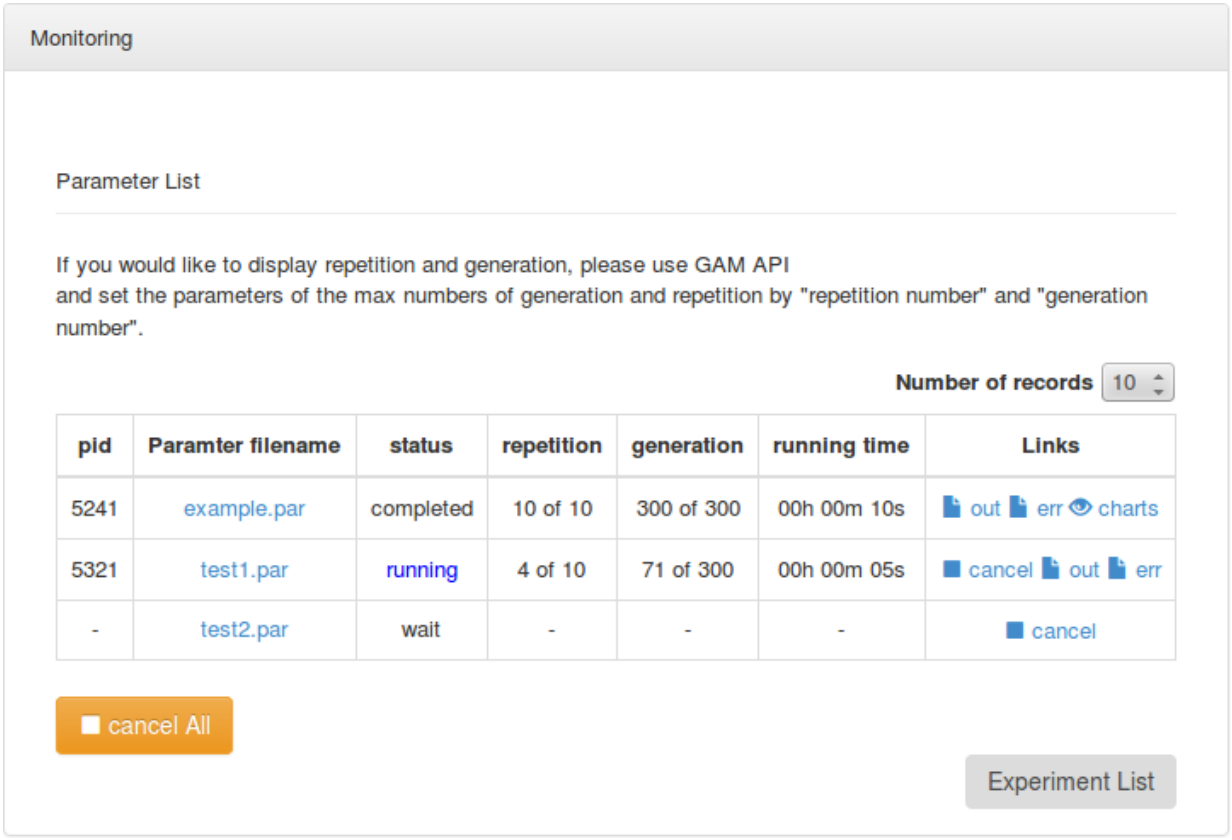


2. Monitoring

2-1. 実行実験一覧画面

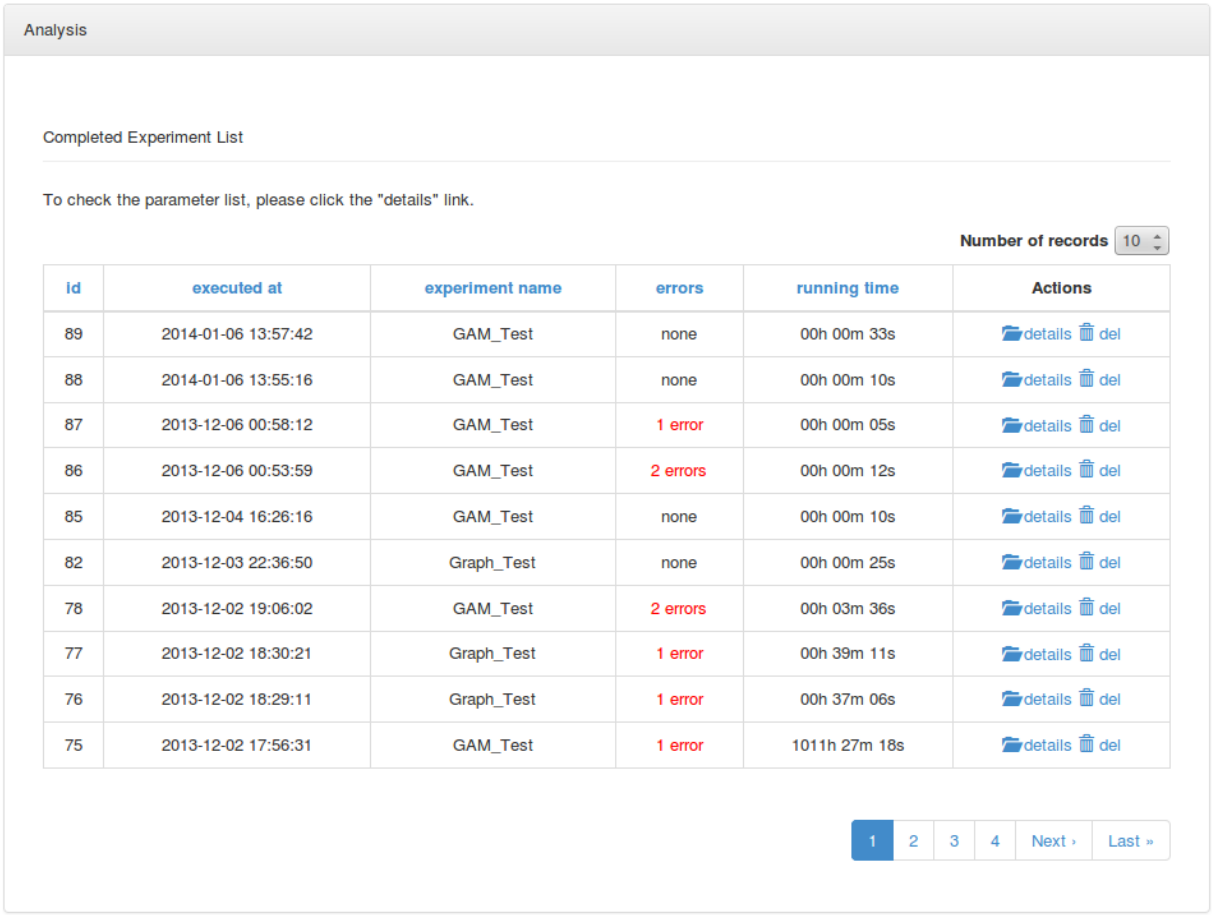


2-2. 実行プロセス一覧画面

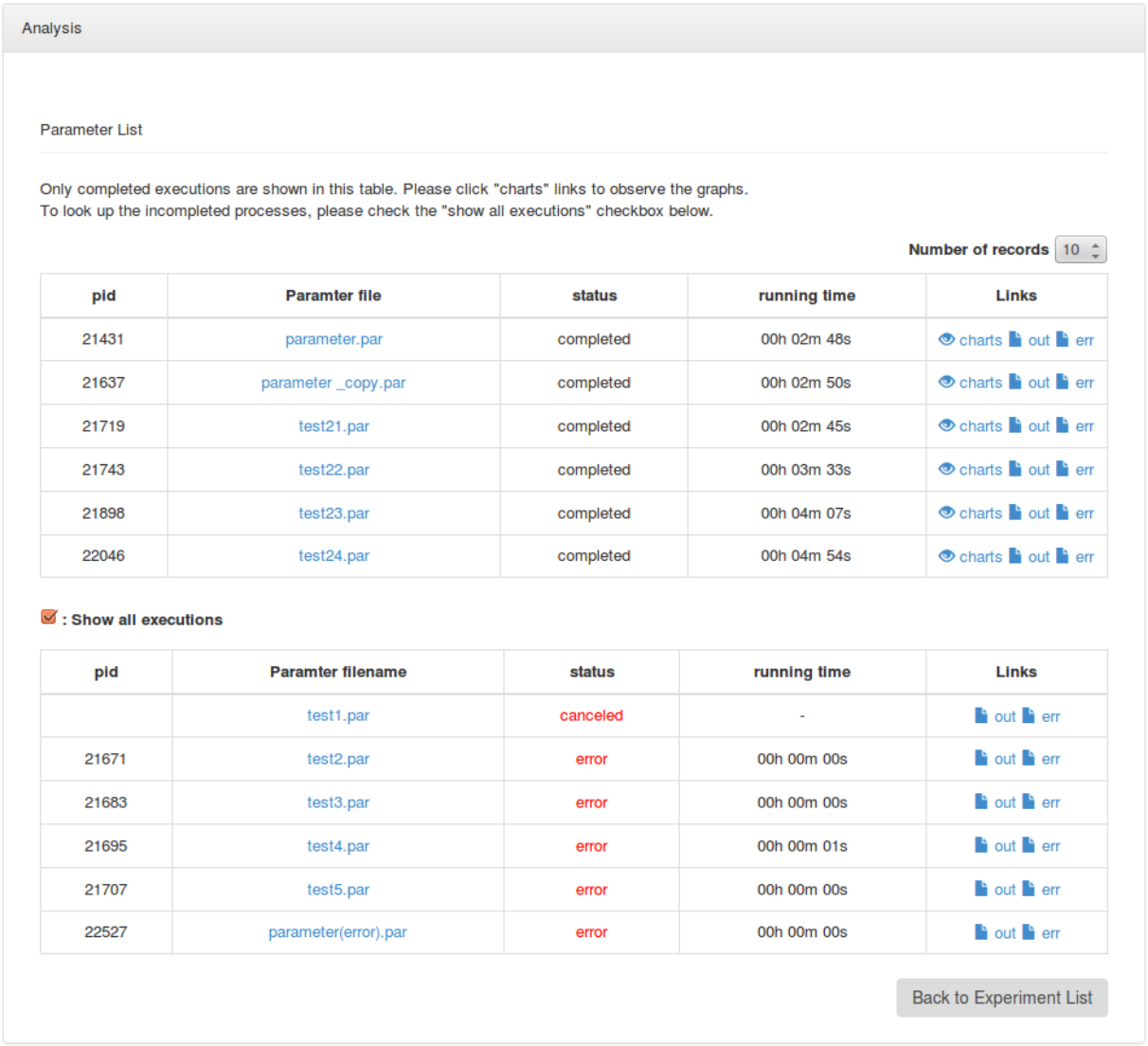


3. Analysis

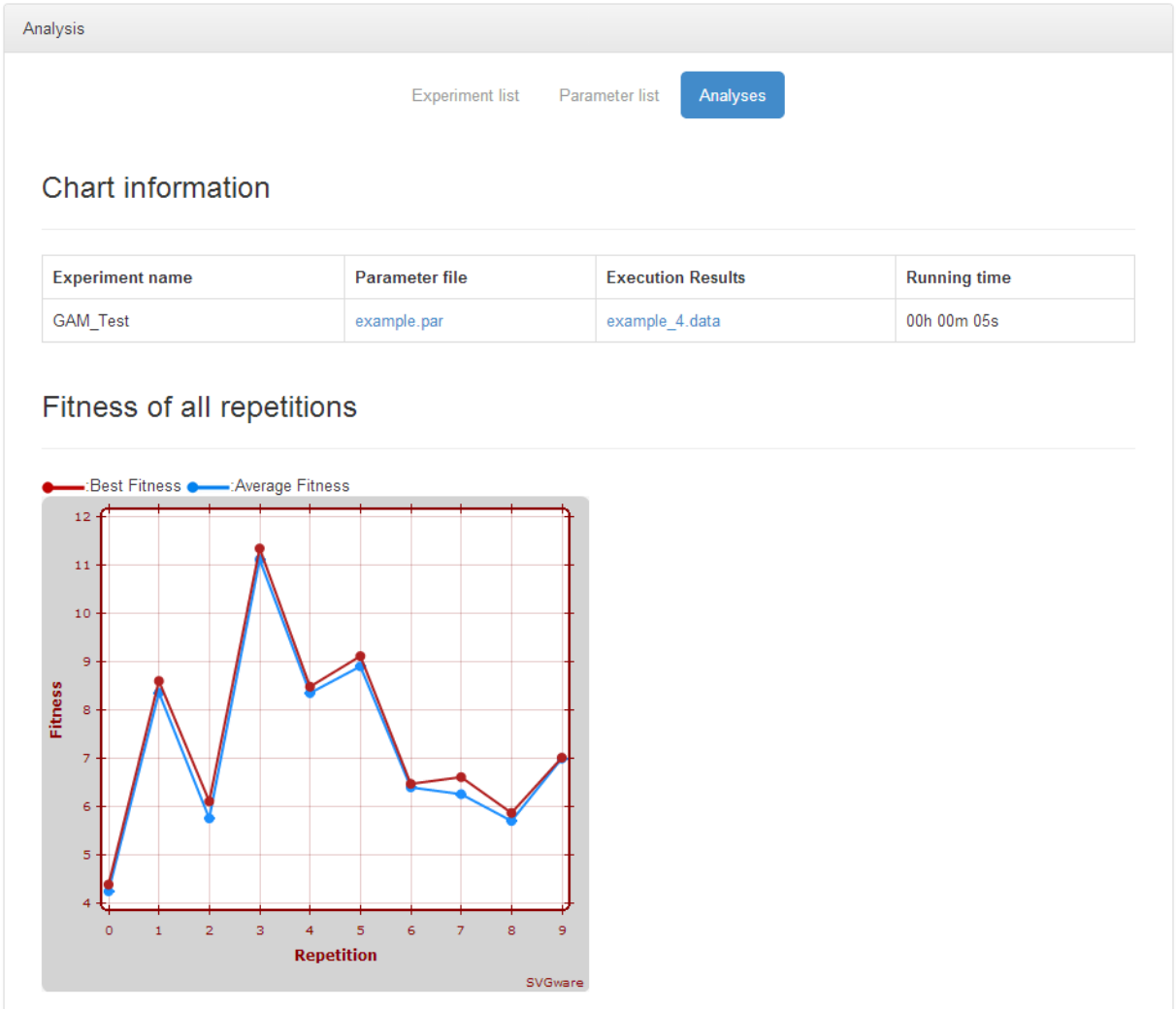
3-1. 完了実験一覧画面



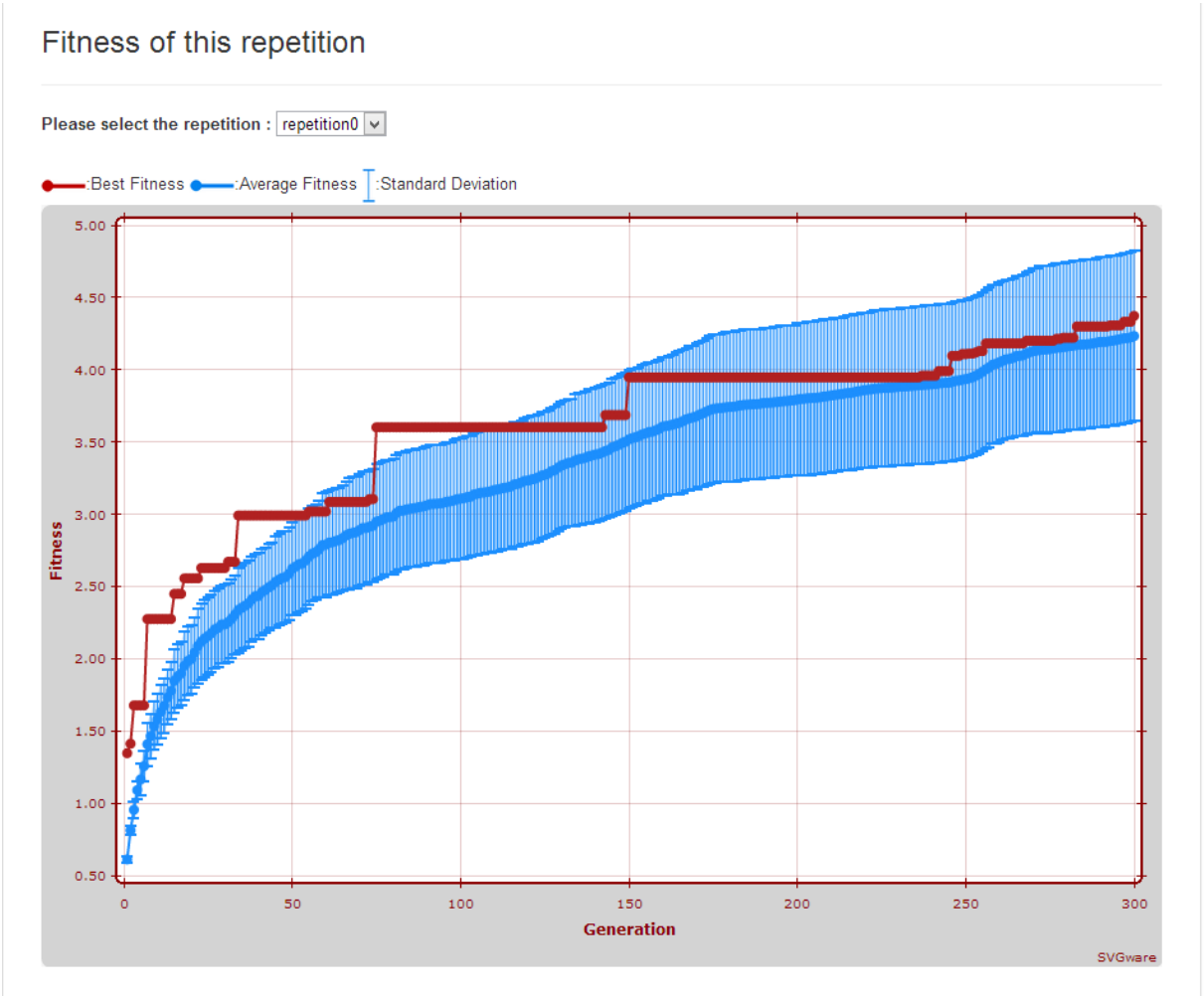
3-2. 完了プロセス一覧画面



3-3. グラフ表示画面 (各反復の適応度)

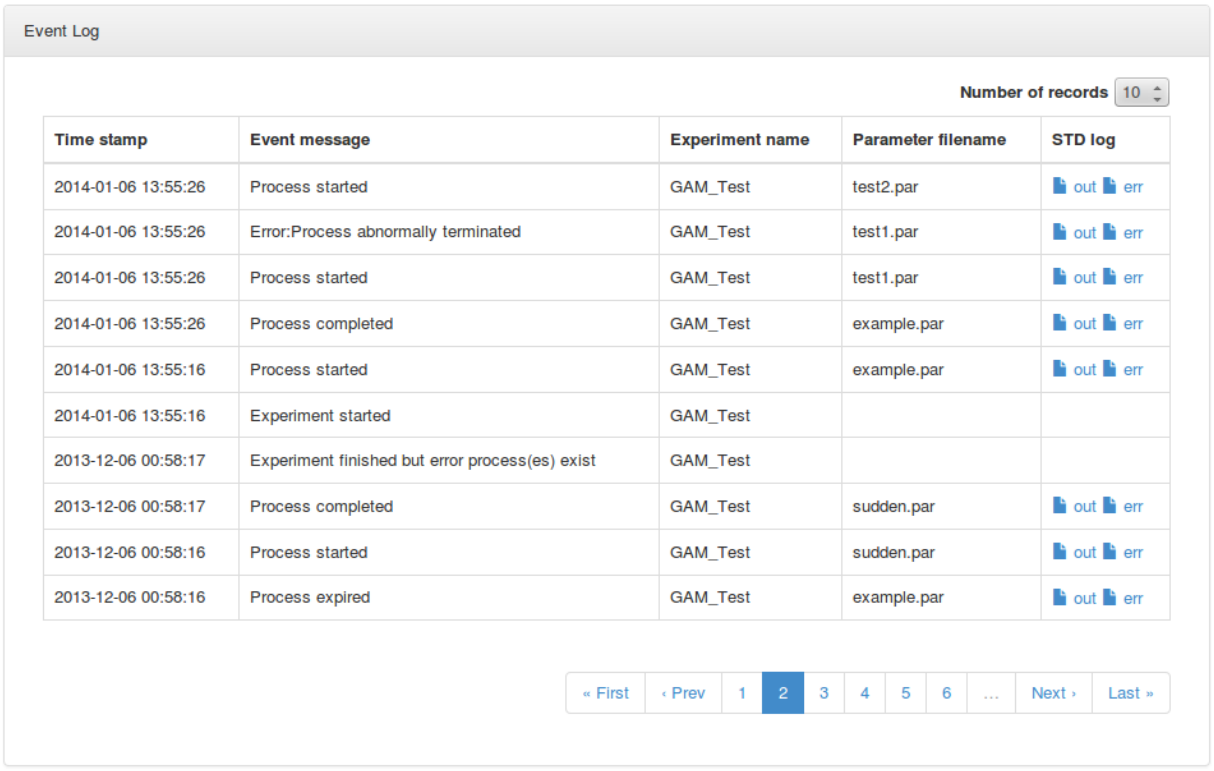


3-3. グラフ表示画面 (各世代の適応度)



4. Event Log

イベントログ画面



付録B ユーザマニュアル

1. 事前準備

GAM で実験を行うには、実行ファイルとパラメータファイルを作成し、それらを適切なディレクトリ構成に配置する必要があります。また、GAM 用の出力ライブラリを用いることで、計算結果を自動でグラフ表示できるようになります。このページでは、これらの作成規則について以下の順に説明します。

1.1 実行ファイルの作成

GAM で実験を行うには、実行ファイルの第一引数にパラメータファイルのパスを指定する必要があります。実行ファイルを作成する際は、進化的計算のパラメータとして用いる値を、第一引数に指定するパラメータファイルの内容から参照して用いるようにしてください。

GAM が実行ファイルを起動する方法は、実行ファイルのファイル形式に応じて下表のコマンドを実行することで行います。下表のコマンドで起動できるように、実行ファイルを作成してください。

実行ファイルの実行コマンド

ファイル形式	実行コマンドの例
Java 形式	java -jar foo.jar bar.par
Python 形式	py foo.py bar.par

* foo,bar は任意のファイル名

1.2 パラメータファイルの作成

実行ファイルに用いるパラメータファイルは、PAR 形式で記述する必要があります。PAR 形式のファイルを作成するには、以下の 3 つの記述ルールに従ってファイル内にテキストを記述し、拡張子を”.par”として保存してください。

ルール 1 : パラメータ項目とパラメータ値

パラメータ項目とパラメータ値は、”パラメータ項目 = パラメータ値 ”の形式で記述します。

例: example1.par

```
Algorism = GA
Generation = 200
```

注意

- パラメータ値の記述は、文字列と数値の両方に対応しています
- イコール記号の前後の半角スペースの有無は任意です

ルール 2 : コメント

行頭に#を付けることで、GAM で パラメータファイルを編集する際に無視されるコメントを記述できます。パラメータ項目の行の一つ上の行にコメントを記述することで、画面上にコメント文を表示することができます。

例: example2.par

```
#Evolutionary parameters Algorism = GA
Generation = 200
```

ルール 3 : 世代数と反復数の最大値¿ *任意

パラメータ項目名を”repetition number”、”generation number”とすることで、GAM で実験の進捗を確認する際に世代数と反復数の最大値を表示することができます。

例: example3.par

注意

- repetition number と generation number の大文字、小文字は区別されません
- repetition number と generation number のそれぞれの単語間のスペースの有無は任意です

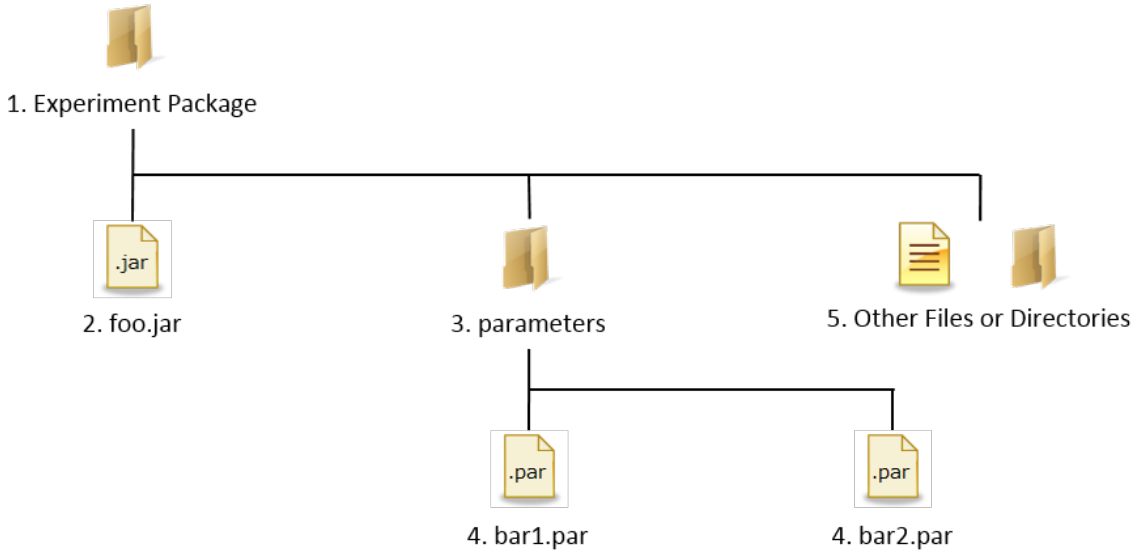
```
#Repetition Number
repetition number = 10

#Generation Number
generation number = 200
```

1.3 ディレクトリ構成

作成した実行ファイルとパラメータファイルを下図のディレクトリ構成とファイル規則に従って配置し、下表の対応圧縮形式の中の圧縮形式で保存してください。GAM では、これにより作成した実験に関係するファイルを含んだ圧縮ファイルを実験パッケージと呼んでいます。実験パッケージを GAM にアップロードすることで、GAM で実験が行えるようになります。

実験パッケージのディレクトリ構成



ファイル規則

番号	項目名	ファイルの種類	命名規則	必須ファイル
1	実験パッケージ	ディレクトリ	任意	-
2	実行ファイル	.jar または.py	任意	○
3	パラメータディレクトリ	ディレクトリ	parameters	-
4	パラメータファイル	.par	任意	-
5	その他のファイル	任意	任意	-

注意

- 実行ファイルが、実験パッケージに 1 つも存在しない場合、2 つ以上存在する場合はエラーになります
- 実験に用いる外部データは、5. その他ファイルとして配置してください
- その他ファイルのパスを実行ファイルやパラメータファイル内で指定する場合は、実験フォルダの階層から相対パスで記述してください

対応圧縮形式

圧縮形式	拡張子
Tar	.tar
Gzip	.tar.gz .tgz
Bzip2	.tar.bz2 .tbz2
ZIP	.zip
RAR	.rar

1.4 GAM 用の出力ライブラリ

GAM は、進化的計算の結果出力用の Java クラスライブラリとして GAM Writer を提供しています。実行ファイルの出力を GAM Writer を用いて行うことで、実験の出力結果を GAM 上でグラフ表示することができます。

GAM Writer を Java プロジェクトにインポートする (Eclipse の場合)

1. Java プロジェクト内に新しいフォルダを作り、そこに Gam Writer の.jar ファイルをコピーします。
2. Java プロジェクトの”properties”メニューを開き、“ Java Build Path ”-¿“ Libraries ” -¿ “ add JARs… ”をクリックします。
3. コピーした.jar ファイルを選択して、OK をクリックします。これでインポートの操作が完了です。

GAM Writer を用いてプログラムを記述する

1. RWriter クラスを import します。
2. RWriter クラスをインスタンス化して、各世代の結果を RWriter の Setter メソッドを用いてインスタンスに渡します。その後、RWriter の”write_line()”メソッドを呼び出して、GAM 用の結果ファイルを出力します。GAM Writer のメソッドについては、GAM Writer リファレンスマニュアル (JavaDocs) を参照してください。

記述例

```
public void RunAndWriteout()
{
    RWriter r_writer = new RWriter();
    for(int rep = 0; rep < max_repetition; rep ++)
    {
        Population pop = new Population();
        r_writer.set_repetition_no(rep);
        for(int gen = 0; gen < max_generation; gen ++ )
        {
            pop.runGeneration();
            pop.updateStatus();

            r_writer.set_generation_no(gen);
            r_writer.set_best_fitness(pop.best_fitness[gen]);
            r_writer.set_avg_fitness(pop.avg_fitness[gen]);
            r_writer.set_fitness_stdev(pop.fitness_stdev[gen]);
            r_writer.set_diversity(pop.diversity[gen]);
            r_writer.write_line();
        }
    }
}
```

2. 使用方法

2.1 実験の実行方法

1. 実験を選択する
   左のメニューバーの Executions ボタンを押し、実験選択画面を表示します。実験選択画面のテーブルから実行する実験を選択し、Next ボタンを押します。
   実験選択画面では、以下のことを行うことができます。

   - 実験を作成する
   - 実験を編集する
   - 実験を削除する

2. パラメータファイルを選択する
   パラメータ選択画面のテーブルから実験に用いるパラメータファイルをチェックし、Next ボタンを押します。複数チェックの場合は、それぞれのパラメータを引数として 1 つずつ実行ファイルが実行されます。
   パラメータ選択画面では、以下のことを行うことができます。

   - パラメータファイルをアップロードする
   - 複数のパラメータファイルを一括作成する
   - パラメータファイルを編集する
   - パラメータファイルを削除する

3. 通知設定を行う
   通知設定画面で通知設定を行い、Execute ボタンを押すと、実験の実行が完了します。
   通知設定画面では、以下の設定を行うことができます。

   - イベントログを記録する
   - 通知メールを送信する

   注意

   - Execute ボタンを押してから実際に実験がサーバ上で実行されるまで、最大で 5 秒ほど時間がかかる場合があります。これは、サーバ上で予約されている実験が存在するかのチェックを 5 秒間隔で行っているためです。

2.2 実験後の実験について

左メニューの Monitoring ボタンを押すと、実行中の実験情報を確認することができます。

- 実行予定の実験一覧
- 実行中のプロセスの状態
- 実行中の実行プロセスの世代と反復回数 (要 GAM Writer)

左メニューの Analysis ボタンを押すと、実行後の実験情報を確認することができます。

- 実験終了後の実験一覧
- 実験の標準出力と標準エラー出力
- 適応度と多様性のグラフ (要 GAM Writer)

3. 機能一覧

3.1 実験について

実験を作成する

1. 左メニューの Execution ボタンを押します。
2. 実験選択画面の右上にある Create ボタンを押します。
3. 実験作成画面の右上にある Browse ボタンをクリックして、実験パッケージをアップロードします。
4. 入力フォームに情報を入力して、OK ボタンを押します。

実験を編集する

1. 左メニューの Execution ボタンを押します。
2. 実験選択画面の Action の列の info アイコンをクリックします。
3. 実験情報表示画面の右下にある Edit ボタンを押します。
4. 入力フォームに変更情報を入力して、OK ボタンを押します。

実験を削除する

1. 左メニューの Execution ボタンを押します。
2. 実験選択画面の Action の列の del アイコンをクリックします。
3. 実験削除確認画面で実験の削除に関連して消去される情報を確認して、OK ボタンを押します。

実験の実行を予約する

1. 実験の実行方法を参考に、実験を実行します。
2. その時点で未完了の実験が存在する場合は、それらの実験の終了後に実行が開始します。

3.2 パラメータについて

パラメータファイルをアップロードする

事前に実験パッケージ内に配置する方法

1. パラメータファイルを参考に、パラメータファイルを作成します。
2. 実験パッケージの parameters ディレクトリ内にパラメータファイルを配置します。
3. 実験を作成するを参考に、パラメータファイルを含めた実験パッケージを用いて、実験を作成します。

パラメータファイルを個別にアップロードする方法

1. パラメータファイルを参考に、パラメータファイルを作成します。
2. 左のメニューから Execution ボタンを押します。
3. 実験選択画面でアップロードする対象の実験を選択し、OK ボタンを押します。
4. パラメータ選択画面の右上にある参照ボタンをクリックして、パラメータファイルをアップロードします。

パラメータファイルを編集する

1. 左のメニューから Execution ボタンを押します。
2. 実験選択画面で編集するパラメータファイルが含まれる実験を選択し、OK ボタンを押します。
3. パラメータ選択画面のテーブルの Actions 列にある edit アイコンをクリックします。

注意

- パラメータファイルの項目の追加・削除を行いたい場合や、コメント文を変更したい場合は、パラメータファイルを再作成してください。

パラメータファイルを削除する

1. 左メニューの Execution ボタンを押します。
2. 実験選択画面で、削除するパラメータが含まれる実験を選択して、OK ボタンを押します。

3. パラメータ選択画面で、テーブルの Action 列の del アイコンをクリックします。
4. 削除確認画面でパラメータの削除に関連して削除される情報を確認し、Delete ボタンを押します。

複数のパラメータファイルを一括作成する

1. パラメータファイルを編集する場合と同様に操作して、パラメータファイル編集画面を表示します。
2. パラメータ編集画面で”create a parameter set”ボタンをクリックします。
3. 変更を加えたいパラメータの範囲と刻み値を指定します。
4. パラメータセット名を入力し、”create”ボタンをクリックします。
5. 生成されるパラメータファイルの個数を確認し、OK ボタンをクリックします。

複数パラメーター一括作成の例

注意

- 上図の例では、2 つのパラメータにそれぞれ 5 通りと 6 通りのパラメータが生成されるので、合計 30 個のパラメータファイルを含んだパラメータセットが作成されます。
- 刻み値が小数である場合、生成されるパラメータ値も小数形式 (例: 1.0) で記述されます。パラメータセットを編集したい場合は、作成したパラメータセットを削除して再作成してください

3.3 結果について

実験の一覧を表示する

- 左メニューの Execution ボタンを押すと、実験パッケージの一覧が表示されます。
- 左メニューの Monitoring ボタンを押すと、実行中および予約中の実験の一覧が表示されます。
- 左メニューの Analysis ボタンを押すと、完了した実験の一覧が表示されます。

実験の進捗を表示する

1. 左メニューの Monitoring ボタンを押します。
2. 進捗を確認したい実験の details アイコンをクリックします。

注意

- 最大世代数と最大反復数を表示するには、パラメータファイルに”Generation Number”と”Repetition Number”が含まれている必要があります。
- 現在の世代数と反復数を表示するには、実行プログラムが GAM Writer を用いて記述されている必要があります。

実験の出力を表示する

実験の一覧から見る方法

1. 左メニューの Monitoring または Analysis ボタンを押します。
2. 実験一覧から details をクリックします。
3. 実行プロセス一覧の stdout または stderr をクリックすると、実行プログラムの標準出力と標準エラー出力を確認できます。

イベントログから見る方法

1. 左メニューの Event Log ボタンを押します。
2. イベントログの実験の stdout または stderr をクリックすると、実行プログラムの標準出力と標準エラー出力を確認できます。

実験結果のグラフを表示する

1. 左メニューの Analysis ボタンを押します。
2. 実験一覧から details をクリックします。
3. 実行プロセス一覧から charts をクリックします。
4. 以下のグラフを確認することができます。

- 反復ごとの適応度の最大適と平均値
- 世代ごとの適応度の最大値と平均値、標準偏差
- 世代ごとの多様性の値

3.4 通知について

通知メールを送信する

1. 左のメニューから Execution ボタンを押します。
2. 実行する実験を選択し、Next ボタンを押します。
3. 用いるパラメータファイルをチェックし、Next ボタンを押します。
4. 表示される通知設定画面で、その実験のメール送信設定を行うことができます。

通知設定について

- Send Email をチェックすることで、通知メールを送信できます
- メールアドレスは、カンマで繋げて記述することで、複数の宛先を指定することができます。
  例) someone@example.com,another@example.com

イベントログを表示する

左メニューの Event Log をクリックすると、イベントログの一覧が表示されます。
イベントログのメッセージと記録条件の一覧

メッセージ	記録条件
Experiment started	実験が開始する
Experiment successfully completed	実験が終了し、全プロセスが正常終了している
Experiment canceled	実験/プロセスの全プロセスがキャンセル状態である
Experiment finished but error process(es) exist	実験が終了し、エラー状態のプロセスが存在する
Process started	プロセスが開始する
Process canceled	プロセスがキャンセルされる
Process expired	プロセスが最大実行時間 (120 時間) を超える
Error: process abnormally terminated	プロセスが終了し、終了ステータスが 0 以外である
Error: process terminated by signal	プロセスが signal により終了する
Error: process unknown status	プロセスの終了を GAM が検知できない

付録C インストールマニュアル

対象環境

GAM システムは以下の環境でのインストールを対象としています。
OS: Ubuntu Server 12.04LTS
Memory: 1GB 以上
Disk: 1GB 以上

なお、GAM システム上で実行する実験プログラムによって必要な性能は左右されますので、必要に応じて性能を調整してください。

インストール手順

1:事前準備（システム管理者アカウントによる作業）

1.1:依存パッケージのインストール

GAM システムでは以下のパッケージを利用しています。GAM システムのインストールの前にインストールを済ませておいてください。
コマンド例
$ sudo apt-get install git gcc make build-essential automake zlib1g-dev libssl-dev libreadline6-dev libyaml-dev libxml2-dev libxslt-dev libmysqld-dev libcurl4-openssl-dev apache2 apache2-prefork-dev libapr1-dev libaprutil1-dev mysql-server postfix

1.2:JRE のインストール

GAM システムはユーザがアップロードした Java 形式または Python 形式の実行ファイルを実行します。Java 形式のファイルを実行するために、JRE をインストールしてください。
コマンド例
$ sudo apt-get install openjdk-7-jre-headless

1.3:mysql に gamadmin ユーザーを作成

GAM システムではデータの保存に MySQL を利用しています。利用するデータベース名は [gam] となっています。gam データベースに対してすべての実行権限を持つ gamadmin ユーザーを作成し、パスワードに Xe7AnfZ3 を設定してください。
コマンド例
$ mysql -u root (-p) mysql¿GRANT ALL PRIVILEGES ON gam.* TO 'gamadmin'@'localhost' IDENTIFIED BY 'Xe7AnfZ3'; mysql¿exit

1.4:gam-system ユーザーの作成

GAM システムを配置するにあたり、gam-system ユーザーを作成してください。
コマンド例
$ sudo adduser gam-system

2:インストール作業（gam-system ユーザでの作業）

システム管理者アカウントでログインしている場合は、gam-system にログインし直してください。
コマンド例
$ su gam-system

2.1:rbenv および ruby-build のインストール

Ruby のバージョン管理システムである rbenv を利用します。これにより、既存の Ruby のバージョンに依存することなく GAM システムのインストールを行うことができるようになります。

2.1.1:ダウンロード

以下の通りコマンドを実行し、rbenv および ruby-build をダウンロードします。
コマンド例
$ git clone https://github.com/sstephenson/rbenv.git /.rbenv
$ git clone https://github.com/sstephenson/ruby-build.git /.rbenv/plugins/ruby-build

2.1.2:rbenv の設定

/.bashrc ファイルの末尾に以下の内容を追加してください。
追記内容
export PATH=”$HOME/.rbenv/bin:$PATH”
eval ”$(rbenv init -)”

2.1.3:rbenv の有効化

/.bashrc の再読み込みを行います。
コマンド例
$ source /.bashrc

2.1.4: ruby のインストール

GAM システムは Ruby on Rails で動作するため、専用の Ruby をインストールします。以下の通りにコマンドを実行してください。
コマンド例
$ rbenv install 1.9.3-p484
$ rbenv global 1.9.3-p484
$ rbenv rehash

2.2:bundler のインストール

GAM システムで利用する外部ライブラリを管理するためのツールとして、bundler をインストールします。以下のようにコマンドを実行してください。
コマンド例
$ gem install bundler
$ rbenv rehash

2.3:GAM システムのインストール

GAM システムをダウンロードし、利用可能にするための準備を行います。なお、ここでは /www/gam にインストールすることを前提としています。他のパスへインストールする場合は以後適宜読み替えてください。
コマンド例
$ git clone http://vm12.sit.cs.tsukuba.ac.jp/git/gam /www/gam
$ cd /www/gam
$ bundle install –without development test

$ rake db:create RAILS_ENV=production
$ rake db:migrate RAILS_ENV=production

2.4:passenger のインストール

Apache2　Web Server 上で GAM システムを動作させるために、Phusion Passenger モジュールをインストールします。以下のようにコマンドを実行してください。
コマンド例:
$ gem install passenger
$ rbenv rehash
$ passenger-install-apache2-module
※インストール中に 2 度 Enter を押す確認が出てきます。
1 度目はインストールをするかどうかの確認です。Enter を押すことでインストールが実行されます。
2 度目は Apache のモジュールに関する設定情報が出てきます。Enter を押す前に設定情報をメモしてください。Enter を押すことでインストールが完了します。

2 度目の Enter を押すかどうかの確認メッセージ

以下のような内容が表示されますので、LoadModule,PassengerRoot,PassengerDefaultRuby の行をメモしておいてください。
出力例
The Apache 2 module was successfully installed.

Please edit your Apache configuration file, and add these lines:

LoadModule passenger_module /home/gam-system/.rbenv/versions/1.9.3-p484/lib/ruby/gems/1.9.1/gems/passenger-4.0.26/buildout/apache2/mod_passenger.so
PassengerRoot /home/gam-system/.rbenv/versions/1.9.3-p484/lib/ruby/gems/1.9.1/gems/passenger-4.0.26
PassengerDefaultRuby /home/gam-system/.rbenv/versions/1.9.3-p484/bin/ruby

After you restart Apache, you are ready to deploy any number of Ruby on Rails
applications on Apache, without any further Ruby on Rails-specific
configuration!
.———————————————————

Press ENTER to continue.

3:Apache の設定（システム管理者アカウントによる作業）

GAM システムを Apache2 Web Server 上で動作させるための設定を行います。gam-system でログインしている場合は、システム管理者アカウントにログインし直してください。

3.1:Passenger モジュールの設定

2.4 でメモした内容を以下のようにそれぞれ設定ファイルに書き出します。

3.1.1/etc/apache2/mods-available/passenger.load

2.4 でメモした LoadModule の行のみを記入した/etc/apache2/mods-available/passenger.load というファイルを作成します
例
LoadModule passenger_module /home/gam-system/.rbenv/versions/1.9.3-p484/lib/ruby/gems/1.9.1/gems/passenger-4.0.26/buildout/apache2/mod_passenger.so

3.1.2/etc/apache2/mods-available/passenger.conf

2.4 でメモした内容を以下の通り編集し、/etc/apache2/mods-available/passenger.conf というファイルを作成します
1 行目に¡IfModule mod_passenger.c¿
2 行目に PassengerRoot の行
3 行目に PassengerDefaultRuby の行
4 行目に¡/IfModule¿
を書き出します。
例
¡IfModule mod_passenger.c¿
PassengerRoot /home/gam-system/.rbenv/versions/1.9.3-p484/lib/ruby/gems/1.9.1/gems/passenger-4.0.26
PassengerDefaultRuby /home/gam-system/.rbenv/versions/1.9.3-p484/bin/ruby
¡/IfModule¿

3.1.3 モジュールの有効化

以下のコマンドを実行し Passenger モジュールを有効化します
コマンド例
$ sudo a2enmod passenger

3.2:Apache のサイト設定用ファイルの作成

Apache2 Web Server 上でサイトを構築するための設定情報を書き出します。すでに Apache 上で別の Web サイトが動作しているかどうかで設定が変わりますので、3.2.1-a もしくは 3.2.1-b のみを実行してください。

3.2.1-a：Apache に既存 Web アプリケーションが存在しない場合

/etc/apache2/sites-available/gam-system に以下の内容を書き出します。
/etc/apache2/sites-available/gam-system の内容
¡VirtualHost *:80¿
ServerName （サーバの FQDN）
DocumentRoot /home/gam-system/www/gam/public
RailsEnv production
¡Directory /home/gam-system/www/gam/public¿
AllowOverride all
Options -MultiViews
¡/Directory¿
¡/VirtualHost¿

3.2.1-b:Apache で既存 Web サイトを運用している場合

既存 Web サイトを運用するにあたり作成したセッティングファイルを、以下のように編集してください。
¡VirtualHost *:80¿
ServerName ...(既存の設定)...
DocumentRoot ...(既存の設定)...

Alias /gam /home/gam-system/www/gam/public
¡Directory /home/gam-system/www/gam/public¿
RailsEnv production

PassengerBaseURI /gam
PassengerAppRoot /home/gam-system/www/gam
AllowOverride all
Options -MultiViews
¡/Directory¿
¡/VirtualHost¿

3.2.2:サイトの有効化

以下のコマンドを実行し、サイトを有効化してください。
コマンド例
$ sudo a2ensite gam-system
$ sudo service apache2 reload

4:動作確認

Web クライアントからシステムにアクセスし、正常に表示されればインストール完了となります。

アンインストール方法

1:gam-system ユーザーの削除

gam-system ユーザーとそのホームディレクトリをすべて削除します。

2

:mysql からデータの削除 MySQL 上に作成されている gamadmin ユーザーを削除し、gam データベースを drop します。
$ mysql -u root (-p)
mysql¿REVOKE ALL PRIVILEGES ON gam.* FROM ’gamadmin’@’localhost’;
mysql¿DROP DATABASE gam

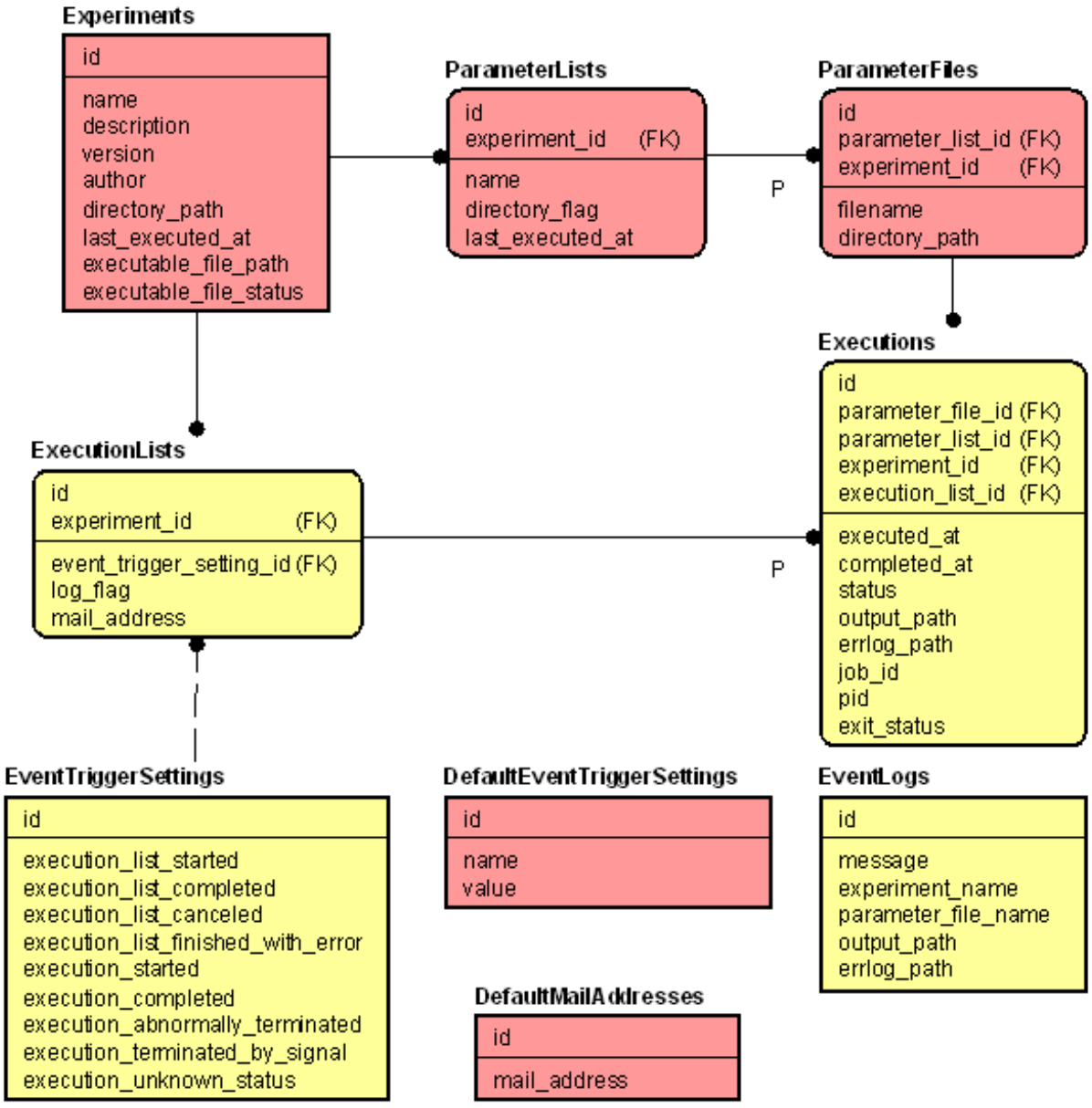
3:Apache2 Web Server の設定を削除

以下の手順で Apache2 Web Server から GAM システムの設定情報を削除します。
実行例

```
$ sudo a2dismod passenger
$ sudo rm /etc/apache2/mods-available/passenger.load
$ sudo rm /etc/apache2/mods-available/passenger.conf
$ sudo a2dissite gam-system
$ sudo rm /etc/apache2/sites-available/gam-system
```

以上

付録D ER図


Experiments
id
name
description
version
author
directory_path
last_executed_at
executable_file_path
executable_file_status
ParameterLists
id
experiment_id (FK)
name
directory_flag
last_executed_at
ParameterFiles
id
parameter_list_id (FK)
experiment_id (FK)
filename
directory_path
P
Executions
id
parameter_file_id (FK)
parameter_list_id (FK)
experiment_id (FK)
execution_list_id (FK)
executed_at
completed_at
status
output_path
errlog_path
job_id
pid
exit_status
P
ExecutionLists
id
experiment_id (FK)
event_trigger_setting_id (FK)
log_flag
mail_address
EventTriggerSettings
id
execution_list_started
execution_list_completed
execution_list_canceled
execution_list_finished_with_error
execution_started
execution_completed
execution_abnormally_terminated
execution_terminated_by_signal
execution_unknown_status
DefaultEventTriggerSettings
id
name
value
EventLogs
id
message
experiment_name
parameter_file_name
output_path
errlog_path
DefaultMailAddresses
id
mail_address