第2章　開発記録とバージョン管理

2.1　気象庁における開発管理の取り組み [1]

気象庁の数値予報モデル開発において、開発管理についての全庁的な取り組みは 2011 年度に気象庁技術開発推進本部 [2] モデル技術開発部会（モデル部会）に「開発管理グループ」が設置されたことに始まる。これは、欧州中期予報センター (ECMWF) や英国気象局 (UKMO) への派遣経験者からの報告で、これらのセンターにおける開発管理の取り組みが報告され (原・高谷 2013)、その重要性が庁内に認識されたことがきっかけであった。

開発管理グループは数値予報課、気候情報課、気象研究所などの庁内のモデル開発にたずさわる職員から構成され、庁内の各モデル開発の現状とあるべき姿や、諸外国の状況についてのレビューを行った上で、バージョン管理やプロジェクト管理システムを用いた情報共有が必要であることを提言した。

それ以前にも、開発コミュニティ（あるモデルやシステムを開発するための開発者の集まり）によってはバージョン管理システムやプロジェクト管理システムをそれぞれのコミュニティが管理するサーバにインストールして活用してきたが、そのようなサーバ機材や構築技術、維持体制がすべてのコミュニティにあるわけではないことが課題となっていた。開発管理グループによる提言では、庁内のモデル開発にかかわるすべてのプロジェクト管理システム、バージョン管理システムの運用を全庁的に統一することを求め、その提言に基づき、翌年には数値予報モデル開発管理情報共有装置（通称：開発管理サーバ）が導入され、庁内のどの開発コミュニティでも開発管理に関係するツールが利用できるようになった [3]。

2012 年度以降は、開発管理グループに代わって開発管理調整グループがモデル部会に設置され、開発管理サーバの運用ルールの具体化などが行われた。その運用ルールでは以下のようなことが定められた。

- プロジェクト管理システム、バージョン管理システムを活用する。
- プロジェクト管理システムとしては Redmine を用いる [4]。
- バージョン管理システムとしては Subversion または Git を用いることとし、どちらを使うかについては開発コミュニティの裁量とする [5]。
- バージョン管理システムにおける開発本流には業務用または共用最新版のソースコードを登録することとし、開発用のソースコードをバージョン管理システムのブランチ（本流からの枝分かれ）に登録することを推奨する。
- 開発本流の内容に変更を加える際には、必ずプロジェクト管理システムにその説明を記録する。

これらは最低限のルールであり、各開発コミュニティではこれらのルールに沿った上で、それぞれが利用しやすい開発ルールを定めることとしている。

さらに、この開発ルールをより具体的にした上で「統一環境における数値予報モデルの開発管理の指針」が 2014 年 3 月に気象庁技術開発推進本部によって制定された。現在では多くの開発コミュニティがこの指針に基づいて開発管理サーバーを利用し、その開発過程の記録をしている。

これまでに述べた動きは、気象庁技術開発推進本部モデル部会が主導して行ったモデルの開発管理に関するものであったが、開発管理の必要性はモデル開発にとどまるものではなく、開発管理のためのツールはモデル開発の基盤整備、アプリケーション開発、数値予報ルーチンシステム管理でも活用されている。

本章では、プロジェクト管理システム、バージョン管理システムの一般的な事項について簡単に説明したのち、開発管理サーバの仕様と運用、そして、数値予報課内でのプロジェクト管理システム、バージョン管理システムの活用について、各開発コミュニティから報告する。すでに述べたように、最低限のルールに従えば、これらのシステムの利用方法は各開発コミュニティーが自由に定めることができるため、その利用方法は多様である。その多様な利用方法は、これから開発管理にかかわるツールを使い始めようとする開発コミュニティにも大いに参考になると期待する。

なお、本稿では開発管理に関するツールの利用についての数値予報課内での取り組みを取り上げたが、これらのシステムは数値予報課以外にも、気候情報課、海洋気象情報室、環境気象管理官、気象研究所、気象衛星センターなどで利用されている。

参考文献

原旅人, 高谷祐平, 2013: 海外数値予報センターの開発管理の例. 数値予報課報告・別冊第 59 号, 気象庁予報部, 195–199.

[1] 原 旅人、永戸 久喜

[2] 気象庁長官を本部長とし、モデル技術開発部会、豪雨監視予測技術部会、静止衛星データ利活用部会が設置されている。この 3 つの技術開発間における情報共有と相互連携により一体的な推進を図るとともに、これらの技術開発に共通する外部機関との連携・協働の促進等に戦略的な方針と計画の作成を通じた総合的な技術開発体制を構築することを目的にしている。

[3] 開発管理サーバーの管理は数値予報課で行っている。

[4] 当時は開発コミュニティによっては Trac と呼ばれる Redmine と類似したツールを使っていたが、管理コストの低減、利用方法の共有の促進の観点から、機能がより豊富で柔軟に運用しやすい Redmine に統一することとした。

[5] プロジェクト管理システム同様、一つのシステムに統一したいところであったが Subversion と Git で設計思想が大きく違う中で、すでに使用を開始しているものを一方に変更させるのは負担が大きい、ということから、どちらかを選択すればよいようにした。

2.2 プロジェクト管理システム [1]

2.2.1 プロジェクト管理システムとは

ソフトウェア開発の現場では、その開発をプロジェクトと呼ぶことが多い。一般に「プロジェクト」とはある目標を達成するための（定常業務とは異なる）期限がある特別な計画を指すが、ソフトウェアの開発はある明確な目的と期限を設けて行うことが多いので、まさにプロジェクトと呼ぶにふさわしいのであろう。

ソフトウェア開発などでは多くの人が様々なタスクを分担して行い、その集合としてプロジェクトを構成する。ソフトウェアは多くの部品で構成され、それぞれに要件定義（仕様を定めること）、詳細設計（定めた仕様に基づき詳細な挙動を決めること）、実装（ソースコードのプログラミング）、テストなどのステージがあり、それぞれのステージの専門家が担当するのが一般的である。それらの部品が一つでも欠ければそのソフトウェアは完成しないため、それぞれ部品を漏れなく工程管理することが必要となる。

Trac[2] や Redmine[3] に代表されるウェブベースのプロジェクト管理システムは、これらの多くのタスクを登録してリスト化した上で、それぞれについてその過程や関連する議論を記録するとともに、そのタスクの現在の進捗状況（ステータス）を把握しやすくするためのシステムである。プロジェクト管理システムと呼ばれるものは以前から商用のものが多くあったが、Trac や Redmine は無償で利用することができて、ユーザはウェブブラウザ以外の特別なソフトウェアを必要としないことから、急速に利用が広まっている。元来はソフトウェア開発のプロジェクトを管理するためのシステムとして開発されたが、作業過程の記録、現在の進捗状況の把握は、期限が定められたプロジェクトとしてだけではなく、継続的に開発を進めるソフトウェア（モデル開発はこれに該当）、ソフトウェア開発以外のプロジェクト、さらにはプロジェクト以外の日常業務でも必要であるため、さまざまな場面で活用されている。

2.2.2 プロジェクト管理システムの気象庁での利用

モデル開発においてもプロジェクト管理システムにおけるタスクの登録、作業過程の記録、進捗状況の把握は、現在および将来の開発者に自らが行った開発についての説明責任を果たす上で必須のことであり、プロジェクト管理システムを大いに活用することができる [4]。

数値予報課でのプロジェクト管理システムの本格的な利用の始まりは、2008 年ごろの asuca (気象庁予報部 2014) の開発への Trac の導入であった。その後、いくつかの開発コミュニティで Trac、さらには Redmine の利用も始まった。

2012 年以前はさまざまなサーバに分散して Trac や Redmine が運用されていたが、すでに述べたように、開発管理サーバを整備して、数値予報課だけに限らず、庁内の数値予報モデル開発に関するプロジェクト管理システムの運用を集約した。開発管理サーバでは Redmine を標準的なソフトウェアと定め [5]、Trac を利用していたプロジェクトは Redmine に移行した [6]。

2.2.3 プロジェクト管理システム Redmine の機能

以下では、プロジェクト管理システムの一つである Redmine について、本章の後半で紹介される活用例の理解に必要最低限の事項を説明する。詳細については、Redmine の公式ウェブページを参照されたい。

(1) チケット管理

Redmine では、タスク登録のための「チケット」を作成する。そのチケットの管理がプロジェクト管理システムの最も重要な機能の一つである。プロジェクト管理システムにチケットとしてタスクを登録し、それに基づいて進めるソフトウェア開発の手法はチケット駆動開発と呼ばれている。なお、その機能を強調して、プロジェクト管理システムのことを課題管理システムと呼ぶこともある。

それぞれのチケットは以下のような属性を持つ。

- チケット番号（チケットを作成した際に自動的に付加される。この番号がチケットを特定するキーとなる。）
- トラッカー（チケットの分類。初期設定では「バグ」「機能」「サポート」が登録されている。管理者が設定を変更することもできる。）
- チケットのタイトルとその概要
- 担当者
- ステータス（初期設定では「新規」「進行中」「解決」「フィードバック」「終了」「却下」のステータスが登録されている。管理者が設定を変更することもできる。）
- 優先度
- 期日

属性の種類は管理者によって追加・変更することができる。初期設定におけるチケットの新規作成画面の例を図 2.2.1 に示す。

---

[1] 原 旅人

[2] `https://trac.edgewall.org`

[3] `http://www.redmine.org`

[4] 「統一環境における数値予報モデルの開発管理の指針」（第 2.1 節参照）では、役割を明示するために「プロジェクト管理システム」を「情報共有ツール」と読み替えている。

[5] Trac では一つのプロジェクト管理システムに一つのプロジェクトしか登録できなかったが、Redmine では複数のプロジェクトを登録することが可能である。また、プロジェクトの間に親子関係を持たせることもできる。後発の Redmine のほうが機能が充実していた点がいくつかあったので、Redmine を選択し、統一することにした。

[6] Trac から Redmine への移行ツールは、開発管理サーバの管理を担当している数値予報課基盤整備グループによって開発された。

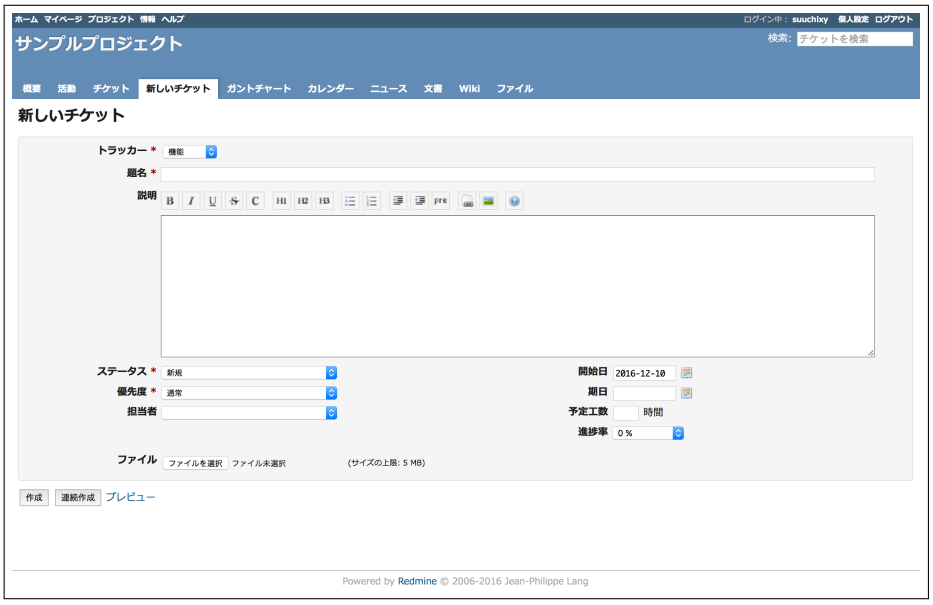

図 2.2.1　Redmine における新規チケット作成画面の例。

図 2.2.2　Redmine におけるチケット表示画面の例。プロジェクト名やユーザ名は架空のものである。

図 2.2.3　Redmine におけるチケット一覧表示画面の例。プロジェクト名やユーザ名は架空のものである。

登録された各チケットは、図 2.2.2 のように表示され、チケットのタイトル、内容、ステータス、担当者などが上部に、チケットに対するコメントの履歴がその下に表示される。

また、このチケットを一覧表示することで、各タスクの現在の進捗状況が一目瞭然となる。その例を図 2.2.3 に示す。

チケットの担当者やステータスは進捗に応じて変更する。また、それぞれのチケットには作業の記録、コメントの記入、ファイルの添付などができて、これらの記述が、現在および未来の開発者への説明責任を果たすことに対応する。

(2) ワークフロー

ソフトウェアの開発においては、ソースコードの作成、そのレビュー、バージョン管理システムへのコミットなど、開発コミュニティによって手順が定められていることが多い。その手順を Redmine 上に実装したのがワークフローである。ワークフローはチケットの属性の一つである「ステータス」の遷移を規定するもので、トラッカーごとに異なるワークフローを定義することができる。トラッカーが「機能」の場合の開発者のワークフローは図 2.2.4 のようになっている（「却下」は「機能」のトラッカーでは利用されていない）。たとえば、現在のステータスが「進行中」の場合に、遷移できるのは「解決」「フィードバック」「終了」であり、「新規」には遷移できない。このようなワークフローは、Redmine ごとにその管理者が設定できる。

これらの機能を活用することで、開発者にその開発コミュニティの開発フローに沿ったステータスの変更を行わせることができる。その活用については、この章で紹介する各開発コミュニティでの利用例を参照されたい。

(3) バージョン管理システムとの連携

ソフトウェア開発のプロジェクト管理はソースコードのバージョン管理と密接に関係している。そのため、プロジェクト管理システムはバージョン管理システムとの連携機能を持っている。例えば、そのプロジェクトに関連するとして登録したリポジトリの変更をブラウザで閲覧でき、そのチケットに関連する変更を見やすく表示することができる。

(4) 情報共有のための機能

Redmine にはいくつかの情報共有のためのツールが実装されている。例えば、各プロジェクトのトップページを含め、Wiki を使うことができる。Wiki はウェブブラウザを利用してウェブサーバ上の文書を書き換えることができるシステムである。Redmine 内部のページを含むウェブページへのハイパーリンクを記述することもでき、プロジェクトに関するお知らせや様々なリンク集を Wiki に作成して公開することができる。そ

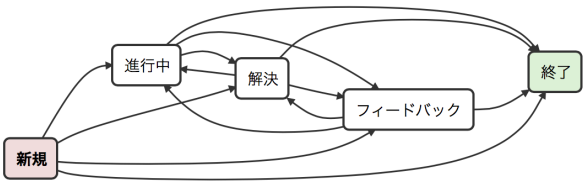

図 2.2.4 Redmine の初期設定におけるワークフロー。トラッカーが「機能」の場合。

の他に、掲示板機能としてフォーラムと呼ばれる機能もある。

また、チケットが新規に作成されたり、編集された場合に、プロジェクトの関係者にメールを送信する機能もあり、Redmine を能動的にチェックしなくても、変更情報などを受け取ることができる。

さらには、Redmine には RSS 機能が搭載されており、RSS リーダーを利用してチケットやリポジトリの更新情報をまとめて閲覧することもできる。

2.2.4 プロジェクト管理システム利用にあたっての留意点

プロジェクト管理システムを利用することのメリットを享受するためには、その開発コミュニティの開発ルールを明確化して、開発に参加している関係者全員がそのルールにしたがって、タスクをチケットとして登録して、そのチケットに開発過程を記録し、ステータスを進捗に応じて正しく変更することなどが必要である。本章で紹介する数値予報課内での活用例は、これから Redmine を使い始める開発コミュニティに参考になるであろう。

Redmine の利用にあたって、チケットに登録する際のタスクの規模、すなわち、タスクをどこまで細かく分割してチケットにすべきかについて悩むユーザが庁内には多いようである。それぞれの開発コミュニティでルールを設けることが必要であるが、タスクの分割の仕方は主観的な面もあり一律に判断するのが難しい。そのような場合は、チケットを作成することを躊躇せずに、まずはチケットを作成してその規模について試行錯誤をしてみることをお勧めしたい。その試行錯誤の中で、開発コミュニティの中での使い方が定まっていくことが多い。また、チケットの統合、分割は頻繁に行われることであり、作業を進める上で統合または分割したほうがやりやすいと判断される場合は、そのときにその統合または分割を行えばよい。ぜひ、利用しながら自分の開発コミュニティにとって便利で有効な使い方を模索していただきたい。

参考文献

気象庁予報部, 2014: 次世代非静力学モデル asuca. 数値予報課報告・別冊第 60 号, 気象庁予報部.

2.3 バージョン管理システム [1]

2.3.1 バージョン管理システムの機能

バージョン管理システム (VCS: Version Control System) は、ソースコードの変更履歴を記録するとともに、並行したいくつかの開発を効率良く行うための支援機能を提供する。

有名な VCS はいくつかあるが、共通していることは以下のことである。

- リポジトリとよばれるデータベースを用意し、リポジトリにファイルの変更履歴などを格納する。
- ユーザはリポジトリからそのコピー（作業コピーと呼ばれる）をローカルのディスクやネットワークを通じて取り出し、その作業コピーのファイルに対して変更を加える。そして、その変更をリポジトリに登録する。多くの VCS ではその登録作業をコミットと呼んでいる。その変更内容の登録の際には、変更者、時刻もあわせて記録される。
- ユーザは任意の時点におけるファイル群をリポジトリから取り出すことができる。
- リポジトリには開発本流があり、ユーザはその本流の枝分かれであるブランチを作成して、同様にブランチへの変更内容をコミットしてリポジトリに登録できる。また、ブランチからさらにブランチを作成することもできる。ブランチへの変更は、マージと呼ばれる操作で枝分かれ元に反映させることができる。これらの流れの模式図を図 2.3.1 に示す。もし、マージの際にすでにコミットされた他の開発者からの変更と衝突する場合はそれを検知してユーザに知らせる。

以下では、数値予報課で以前 VCS として広く使われてきた CVS (Concurrent Versions System) [2]、そして、現在、開発管理サーバで利用可能となっている Subversion[3] および Git[4] について簡単に解説する。なお、開発管理サーバの利用にあたって、Subversion と Git のどちらを使うかは開発コミュニティが選択することができることとしているが、開発過程を共有するためのツールという観点からの運用ルールの留意点について触れる。それぞれの詳細については、各システムの公式ウェブページを参照されたい。

2.3.2 CVS

2000 年代まで VCS として広く使われていたのが CVS である。数値予報課でも 2000 年ごろから利用が始まり、全球モデル、メソモデル、NuSDaS ライブラリなどの変更履歴が CVS で管理されていた [5]。CVS は

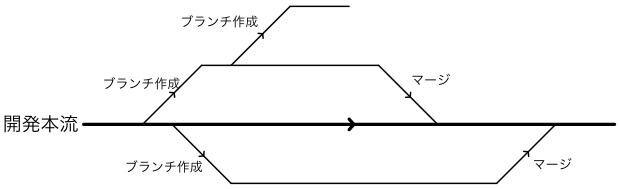

図 2.3.1 バージョン管理システムにおける開発本流とブランチ、マージの模式図。左から右へと時間が流れる。

バージョン管理システムが求められる基本的な機能を有していたものの、ファイル名やディレクトリ名の変更履歴が追跡できないこと [6]、バイナリファイルの扱いが得意ではなかったこと、ファイル単位でバージョン番号が与えられていたので、複数のファイルの変更が同時にコミットされてもそれが分かりづらかったり、ファイルによってコミット時刻が微妙に異なったりするために [7]、指定されたある時刻における状態を確実に取り出せている保証がなかった [8] ことなどの欠点があった。これらの欠点を解決しようと開発されたのが、現在では広く利用されている Subversion である。

2.3.3 Subversion

Subversion は CVS の操作性を継承しつつ、CVS ではファイルごとにつけられていたバージョン番号による履歴の管理ではなく、ファイル群にリビジョン番号を付加して管理することで [9]、リビジョン番号を指定すれば、確実にそのときのファイルの状態を取り出せるようになった。また、ファイル名やディレクトリ名の変更の際にも変更したことを履歴に残せるようになり、変更履歴の連続性を確保できるようになった。また、CVS ではリポジトリと作業コピーの間の差分を調べる際にもリポジトリとのネットワーク通信が発生していたが、Subversion ではリポジトリの内容がローカルにもコピーされているため、リポジトリと作業コピーの間の差分の取得にネットワーク通信が必要なくなり、高速に動作するようになった [10]。

CVS のリポジトリから Subversion のリポジトリへの移行は `cvs2svn`[11] と呼ばれる移行ツールを利用することで行うことができる。数値予報課では、2008 年ごろから一部のモデル開発で CVS から Subversion への

---

[1] 原 旅人
[2] `http://www.nongnu.org/cvs/`
[3] `http://subversion.apache.org`
[4] `https://git-scm.com`
[5] CVS が利用される以前のほとんどの変更履歴は、現在の開発者から見られるようになっていない。
[6] ファイル名などを変更する場合は、一旦削除した上で、新規のものとして登録することになるので、そこで履歴の連続性を失っていた。
[7] 複数のファイルを同時にコミットしても、ごく短時間の間にファイルが一つずつコミットされる扱いになる。
[8] アトミック性を満たさない、と言われる。
[9] Subversion では、ファイルを一つでも変更するとリポジトリのリビジョン番号が 1 つ増える。
[10] ただし、最新のリポジトリの内容との差分を取得するためには、`svn update` によってリポジトリの内容とローカルに保存されている情報を同期させておく必要がある。
[11] `http://cvs2svn.tigris.org`

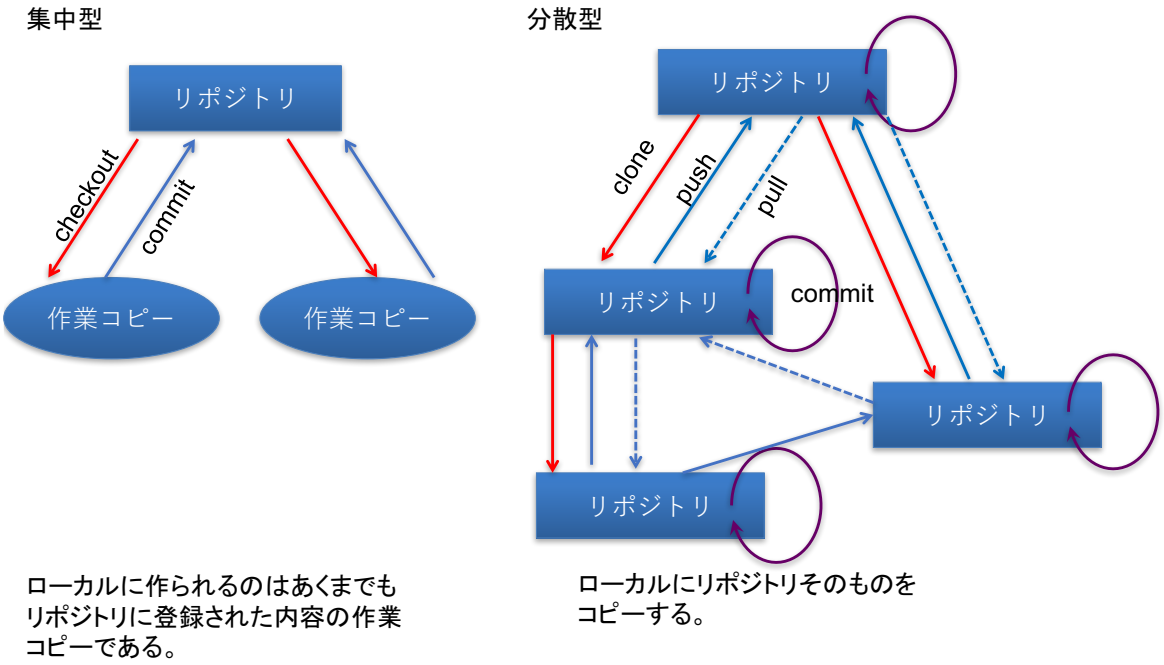

図 2.3.2　集中型 VCS と分散型 VCS のリポジトリとその関連操作の模式図。

移行が行われたり、新しくリポジトリを作成する場合には Subversion が利用されることがあった。現在、数値予報課で CVS のリポジトリが利用されているのは、既に開発がほとんど行われていない古いソフトウェアばかりであり、数値予報課内で利用されているほとんどのリポジトリは Subversion か、後述の Git になっている。

Subverion では開発本流のことを trunk と呼んでいる。リポジトリの最上位のディレクトリに `trunk`, `branches`, `tags`[12] というディレクトリを作成することを推奨し、開発本流は `trunk` 以下に格納することが多い。このことがあくまでも「推奨」であることからもわかるように、ソフトウェア上で trunk を特別扱いする機能はなく、リポジトリの運用ルールのなかで決められるものである。

2.3.4　Git

これまで紹介してきた CVS や Subversion ではリポジトリをあるサーバ上に設置し、ユーザはそのリポジトリに接続して、作業コピーの取得や修正内容のコミットを行う。すなわち、リポジトリは 1 つであり、そのリポジトリは中央リポジトリと呼ばれることがある。このような性質を持つ VCS を集中型 VCS と呼んでいる。

一方、分散型 VCS と呼ばれるものもあり、その代表的なものが Git である。その他にも、Mercurial[13], bazaar[14] などが知られている。集中型では中央リポジトリに格納されているファイルのコピーをユーザは取得するが、分散型ではリポジトリそのもののコピーをローカルやネットワークを通じて取得する。集中型では、ユーザの変更をコミットすることで中央リポジトリに登録するが、分散型ではユーザが取得するのはリポジトリであるので、ユーザは自らの変更内容のコミットを自分が取得したリポジトリに行う。

分散型 VCS におけるコミット操作はローカルの自分のリポジトリに対するものであるので、自分の変更を他のリポジトリに反映させたり、他人の変更を自分のリポジトリに反映させるには、それぞれプッシュ、プルと呼ばれる操作を行う。

分散型においては、コピー元とコピー先のリポジトリはソフトウェア上は対等であり、特に区別するものはない。しかし、VCS の主要な機能の一つは、多くの開発者が行った変更の開発本流への取り込みであり、その取り込みを行うために運用ルールとして中央リポジトリを一つ定めていることが多く、各ユーザが自らのリポジトリに行った変更をその中央リポジトリに反映させる操作を行う。

集中型と分散型のバージョン管理システムのリポジトリとその関連操作の模式図を図 2.3.2 に示す。

集中型 VCS である Subversion と比較したとき、分散型 VCS である Git は、リポジトリをコピーしてしまえば、自らのリポジトリの更新そのものにはネットワーク接続は必要なくオフラインで利用できること（集中

[12] CVS にはリポジトリのある時刻の状態にタグというものを付加できて、そのタグを指定することである時点でのファイル群を取り出すことができた。Subversion では CVS のようにタグを付加する機能はないものの、ある時点の `trunk` を `tags` にブランチを作成するのと同じ要領でコピーすることで、タグの付加と同等の機能を実現している。

[13] `https://www.mercurial-scm.org`

[14] `http://bazaar.canonical.com/en/`

型では変更内容のコミットには中央リポジトリへの接続が必要となる)、中央リポジトリに自らの変更をプッシュをするまでは中央リポジトリに影響をあたえることなく利用できることが大きな特徴であると言えよう。また、リポジトリを簡単にコピーできるため、中央リポジトリとは関係なく、開発本流と少しだけ機能を変えた派生版を作りやすいことも特徴の一つであろう。一方で、ブランチでの開発も含めて共有したい場合には、これらの特徴がデメリットになり得るので注意する必要がある。

2.3.5 バージョン管理システムの運用ルールの留意点

すでに述べたように、開発管理サーバではSubversionとGitを標準的なVCSと位置づけ、利用ができるようになっている。開発が完了した開発本流の変更履歴を管理していくという点では、どちらを使っても大きな差はない。一方、開発途上のソースコードの扱いという面では違いが生じうる。

集中型であるSubversionでは、開発途上のソースコードの変更履歴を残すためには[15]中央リポジトリにコミットする必要がある。そのために、中央リポジトリにブランチを作って、開発途上のソースコードの変更内容をそのブランチにコミットすることが多い。その結果として、その変更履歴を他の開発者が見ることができて、他の開発者の開発内容や進捗を把握することができる。

一方、分散型であるGitでは、変更履歴を残したりバックアップのためにリポジトリにコミットすることは集中型と同じであるが、そのリポジトリは自らのリポジトリであり、他の開発者からも見ることができるリポジトリにプッシュしなければ、自らの変更内容を他の開発者からは見えないようにすることもできてしまう。

一般的なオープンソースのソフトウェアの開発においては、開発途上のソースコードに興味を持たれる場面は多くないかもしれない。しかし、モデル開発においては、最後の完成形だけでなく、開発途中のソースコードにも重要な情報が含まれている場合もある。VCSとしてSubversionを使う場合はブランチに開発途上のソースコードをコミットするタイミング、Gitを用いる場合には、他の開発者からも見えるリポジトリにプッシュするタイミングを開発コミュニティで決めておく必要があるだろう。

[15] 変更履歴の記録とともに、バックアップにもなる。

2.4 数値予報モデル開発管理情報共有装置（開発管理サーバ）[1]

2.4.1 はじめに

気象庁では第 2.2 節で述べた開発管理システム及び第 2.3 節で述べたバージョン管理システムを実際に提供するための環境として、数値予報モデル開発管理情報共有装置（以下、開発管理サーバ）を整備し、気象庁内の数値予報モデルの開発・研究や維持管理に携わる職員（以下、数値予報モデル開発者）が実施する開発について、情報の管理を行っている。

本節では開発管理サーバの設置目的とシステム構成を概説した後、実際どのように運用を行っているか、その方法について解説する。

2.4.2 装置の設置目的

気象庁内で数値予報システムを利用する分野が拡大するのに伴い、気象庁本庁の複数の課室や気象研究所の研究室が数値予報システムの開発に参加するようになっている。しかし、開発項目が拡大するにつれ、他の課室・研究室で行われている開発の内容や進捗状況の情報共有といった連携が十分にとれないという弊害も発生しやすくなる。このような状態に陥ると、開発内容の重複が生じるなど非効率的な開発が発生することに繋がりかねない。また、既知の技術的な手法で解決可能な問題にも関わらず、情報不足によって未解決のまま開発が停滞するなど、業務全体の遅延をもたらす可能性がある。これらの懸念を解決するためにも統一した環境のもとで開発管理を行い、数値予報モデル開発者同士が互いに開発内容を把握するとともに成果を相互利用できることが望ましい。

こうした基本認識を前提に、気象庁技術開発推進本部に設置されたモデル技術開発部会開発管理調整グループでは、2011 年度から統一的な開発環境の整備について継続的に議論し、2014 年 3 月 25 日に「統一環境における数値予報モデルの開発管理の指針」（以下、指針）をとりまとめた。この中では、

- モデル技術開発部会が取り組む、すべての開発内容を開発管理の対象とすること
- 管理の対象とするコンテンツは、現業モデルの開発や維持管理に必要なソースコードや定数[2]などのデータ、開発中のソースコード等の成果とすること
- モデル本体だけではなく、必要により周辺ソフトウェア（検証、可視化など）の管理も実施することを推奨すること
- 関連グループ長[3]は、開発計画をプロジェクト・サブプロジェクト[4]上に公開し、数値予報モデル開発者はこれを遵守して開発を行うこと

などが定められるとともに、

- プロジェクト管理システムとして第 2.2 節で述べた Redmine を利用すること
- バージョン管理システムとして第 2.3 節で述べた Subversion 又は Git を使用すること

が決定された。この指針を基に開発管理環境を提供するシステムとして、開発管理サーバが設置された。

表 2.4.1 開発管理サーバの機器構成

サーバ	主・副 2 台
	CPU: 3 GHz (6 core) × 2
	メモリ: 48 GB (8 GB × 6)
	HDD: 900 GB 2.5 型 SAS × 4
	RAID: RAID5 + spare
大容量ストレージ装置	2 台 + 待機 1 台（非接続）
	HDD: 3 TB SATA × 6
	RAID: RAID6

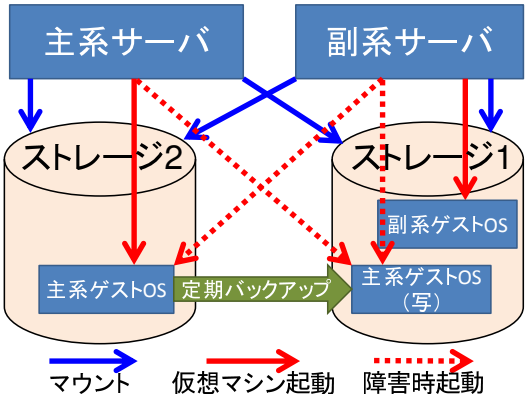

図 2.4.1 開発管理サーバの構成図。両サーバは 2 台の大容量ストレージ装置をマウントしており、それぞれに主系・副系のゲスト OS を記録したイメージファイルを格納している。

2.4.3 システム構成

開発管理サーバは表 2.4.1 のような、主・副のサーバ 2 台と、3 台の大容量ストレージ装置によって構成されている。大容量ストレージ装置のうち 1 台は予備機のためサーバと接続していない。サーバは主・副いずれもホスト OS として CentOS Linux を採用し、KVM (Kernel-based Virtual Machine) による仮想環境を用いたゲスト OS 上に開発管理環境を構築している。ゲスト OS のイメージファイルは大容量ストレージ装置に保存しており、サーバ本体のディスクにはホスト関係のファイルのみが保存されている（図 2.4.1）。現在は開発管理サーバが開発業務の中核を担っていることもあり、これらのデータが保管されているイメージファ

[1] 雁津 克彦
[2] 実行プログラムが利用するデータのうち、標高や気候値のように内容が滅多に変わらないデータ。
[3] 関連グループとは、気象庁技術開発推進本部モデル技術開発部会に設置された各グループを指す。
[4] プロジェクト・サブプロジェクトは第 2.4.4 項 (2) で取り上げる。

イルを毎日 1 回バックアップするとともに、ゲスト OS に保存されているデータのうち、開発管理に直接関係するファイルは個別にバックアップを実施している。

利用者は主系のゲスト OS へアクセスすることで、開発管理環境を利用することができる。このとき、アクセス可能なプロトコルを HTTP(S) に限定して許可している。SSH や RSH といったターミナルによるリモートアクセス、FTP や CIFS などのファイル転送・共有プロトコルは一切受け付けていない。通常、数値予報システム開発では SSH 等を利用して直接サーバにログインを行う。開発作業における各種業務を実施するにあたって、サーバ間のファイル転送、コンパイル作業、画像作成、統計的な検証などでサーバ上のコマンド利用が実務上不可欠なためである。これに対して開発管理サーバの利用では、このようなサーバ上でのコマンド利用を行うことなくほぼ全ての機能を提供可能である。プロジェクト管理システムである Redmine は、Ruby on Rails で構築されたウェブアプリケーションであり [5]、ユーザは通常のウェブブラウザを用いることでほぼ全ての機能を利用することが可能である。また、バージョン管理システムの Subversion と Git についても、利用形態はコマンドライン主体であるが、通信プロトコルとして HTTP(S) を選択可能であり (Sussman et al. 2011; Chacon and Straub 2014)、ユーザの端末でコマンドを実行し、HTTP(S) 経由で開発管理サーバのバージョン管理システム（第 2.3.1 項を参照）を利用することが可能である。こうしたことから、開発管理サーバではユーザが直接サーバにログインすることなく HTTP(S) によるアクセスのみで十分機能するようになっている。

管理面では、ターミナルによるリモートアクセスをサポートする必要がないため、サーバに必要なソフトウェアを限定することができる。また、サーバへの通信手段も限定されることから障害調査時に原因切り分けが容易になる効果も生まれ、結果として管理コストの低減にも繋がっている。

開発管理サーバは気象庁外部から直接アクセスできないようになっており、併せて適宜セキュリティパッチを適用するなどセキュリティ対策を行っている。また、Redmine についてはプラグインにより機能を強化することも可能であることから、必要があればプラグイン導入の作業を実施している [6]。ただし、気象庁全体の情報共有という目的を逸脱しない範囲の利用に留めており、不要なプラグインの導入は実施しない方針としている。

[5] `http://www.redmine.org/projects/redmine/wiki`

[6] Redmine は Ruby で実装されているため、プラグイン管理も Ruby を用いる。Ruby は数値予報ルーチン関連のツールでも利用されている言語であり、管理者育成でも比較的少ない人的教育コストで管理可能という面を持ち合わせている。

表 2.4.2　2017 年 1 月時点で運用中の Redmine 一覧。末尾の「お試し Redmine」は Redmine に不慣れなユーザ・管理者が機能を自由に試用し、慣熟できる環境として公開している。

全球モデル
NHM
asuca
物理過程ライブラリ
海洋気象情報
数値予報事例 DB
共通基盤
化学輸送
同化・観測・QC 関連
結合系
海洋
ガイダンスグループ
お試し Redmine

2.4.4　開発管理サーバの管理体制

(1)　各コミュニティによる利用

開発管理サーバの特徴として、一つのサーバ上に複数の Redmine を運用している点が挙げられる。指針が定める開発管理サーバの利用範囲は、気象庁本庁と気象研究所を含めた複数の課室を想定していることもあり、管理する開発対象が非常に多岐にわたっている。こうした背景からサーバ上で複数の Redmine を運用し、各 Redmine の管理方法の細目は利用するコミュニティに委ねる方式を採用している。これによって、各コミュニティの既存の開発プロセスを踏襲しつつ、第 2.2.3 項で述べた Redmine を活用した開発が随時導入できるよう配慮している。

現在、開発管理サーバでは表 2.4.2 に挙げる Redmine を運用しており、モデル技術開発部会が取り組むほとんどの開発課題は、関連する Redmine 上で開発管理が実施されている。

(2)　各 Redmine の管理

Redmine は内部に複数のプロジェクトを設置することができ、さらにプロジェクト内部にサブプロジェクトを設置することが可能である。指針では開発管理サーバ上の Redmine に設置するプロジェクトとサブプロジェクトについて、数値予報モデル開発の中程度の単位をプロジェクトに、数値予報モデル開発の小さな単位をサブプロジェクトに割り当てることとされている [7]。非常に概念的な指定方法と思うかもしれないが、一方でプロジェクトやサブプロジェクトの単位を各コミュニティが実情に沿った方法で選択可能であることを意味し、柔軟な運用を可能にしている。開発管理サーバ自体の管理業務は数値予報課数値予報班基盤整備グループが担

[7] 数値予報モデル開発の大きな単位は Redmine に割り当てる。

表 2.4.3 Redmine、プロジェクト及びサブプロジェクトの設置、変更及び廃止に必要な手続

対象	必要な手続
Redmine	関連グループ長と開発管理調整グループの承認が必要
プロジェクト	関連グループ長と開発管理調整グループの承認が必要
サブプロジェクト	プロジェクト管理者の承認が必要

当しているが、このように運用面の多くをコミュニティに委ねていることもあり、各 Redmine、プロジェクト及びサブプロジェクトの管理についても、管理者権限を各コミュニティに委譲している。すなわち、開発管理サーバ自体の管理者の他に、各 Redmine に Redmine 管理者を、プロジェクトにプロジェクト管理者を、サブプロジェクトにサブプロジェクト管理者を設け、運用管理を分散して行っている。

開発管理サーバで管理する課題の内容は、業務上定められた各種の開発計画に沿ったものを想定している。このことは新規プロジェクトや新規サブプロジェクトの設置において、計画に沿った開発項目が必ず背景に存在していることを意味している。一方で Redmine の性質上、新規プロジェクトやサブプロジェクトを数分程度の簡単な作業で作成できる。しかし、プロジェクトやサブプロジェクトを乱立させることは、前述の想定を鑑みると決して好ましいことではない。そこで、Redmine、プロジェクト及びサブプロジェクトの設置、変更、及び廃止に際しては、それぞれ表 2.4.3 に示す手続が必要となっている。これによって、業務上必要な課題に必要な資源を投入し、開発に取り組むことを推奨している。

2.4.5 利用方法

(1) 利用開始時の作業

Redmine 上のチケット及びリポジトリは、開発管理サーバが設置されているスーパーコンピュータシステムの支線 LAN に HTTP(S) で接続できる端末から誰でも閲覧することができる。しかし、Redmine 上にチケットを作成したり、リポジトリにコミットするためには Redmine にユーザ登録を行う必要がある [8]。

ユーザ登録は登録を希望する Redmine の画面上で行うことができる。ユーザ登録を行うと Redmine 管理者にメールが送られ、Redmine 管理者が承認することで Redmine 上の各種機能を本格的に利用することができる。注意点として開発管理サーバでは複数の Redmine を運用しているため、異なる Redmine を利用する際にはその都度ユーザ登録が必要となる。例として全球モデル Redmine にユーザ登録済みであったとしても、海洋 Redmine を利用する際には海洋 Redmine にもユーザ登録を行い、海洋 Redmine の Redmine 管理者から利用の承認を受ける必要がある。

Redmine 管理者がユーザを承認する際は必要に応じてユーザにプロジェクト上の役割である「ロール」を設定する。ロールによってチケットが作成できる・できないなどの権限が変わる。どのようなロールが存在するかは、Redmine によって変更することができるため一概には言えないが、概ね、管理者・開発者・報告者の三つは用意されていることが多い。通常、同じ Redmine 内であってもプロジェクトによってユーザに与えられるロールは異なりうる。例えば、あるプロジェクト A では主体的に開発を行うので「開発者」ロールを、別のプロジェクト B では開発成果の利用がメインなので「報告者」ロールを与える、といった方法でユーザとプロジェクトとの関係が割り付けられる。各ロールで何ができるかは Redmine ごとに細かく設定できるため、詳細はシステム上で確認する必要がある。Redmine にログイン可能ユーザであれば Redmine 画面の「情報」⇒「権限レポート」で容易に参照できる。

(2) チケットの登録

指針ではコンテンツを登録する前に、まず課題をチケットに登録することを推奨している。このルールを数値予報モデル開発に導入すると、開発で取り組む予定の全ての課題をチケットとして登録することになる。メリットとして作業の可視化が実現されるとともに作業内容をプロジェクト内で共有できるため、作業の漏れがあればチケットを追加するといった作業抜け防止が期待できる。また、チケット上での各種議論を踏まえた上で開発成果をコミットするという方法をとることで、第三者のレビューによる業務信頼性の向上が期待できる。

指針では、チケットに登録する課題として基本的に開発計画に沿ったものが想定されている。ただし、現業システムの維持管理に必要な開発、軽微な不具合などの外的要因による修正、数値予報モデル開発者個人のアイデアによる成果などは、開発計画に明記されていなくても補完的に登録することができるようになっており、柔軟な開発にも対応している。むしろ、実際の開発ではトライアル・アンド・エラーで様々なアイデアを試すことや予期せぬ不具合への対応が非常に重要になってくることから、こうした柔軟性の確保は不可欠である。

チケットの登録では、プロジェクト内の数値予報モデル開発者が共通して理解できる程度の平易な内容で記載することが推奨されている。これはチケットが登録者だけではなく、相互レビューなどの議論によって第三者から閲覧されることを想定しているためである。また、チケット上の議論は後に過去の開発経緯を調べ

[8] チケット、リポジトリ、コミットなどの用語は第 2.2 節、第 2.3 節を参照のこと。

表 2.4.4　チケットとリポジトリの紐付け方法。以下のコメントを所定の記載箇所に記入することで紐付けできる。

コメント	例	記載箇所	効果
r[リビジョン番号]	r200	チケット	チケット本文から特定リビジョンへのリンク
#[チケット番号]	#1432	コミットログ	リポジトリブラウザからチケットへのリンク
refs #[チケット番号] issueID #[チケット番号] references #[チケット番号]	refs #1432 issueID #1432 references #1432	コミットログ	リポジトリブラウザからチケットへのリンクと、チケットからリポジトリブラウザへの相互リンク（どの書法も同じ効果）

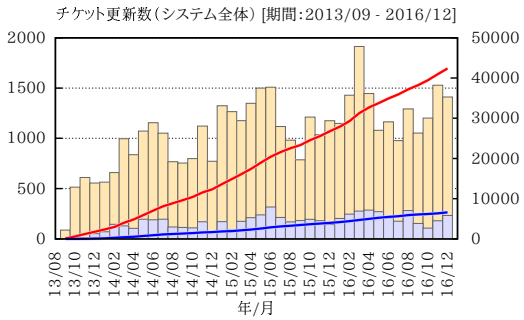

図 2.4.2　月ごとのチケット更新数（棒：左軸）と累計（折れ線：右軸）。橙・赤は全体数を、水色・青は全体数のうち気象研究所の該当数を表す。ここで、2017 年 1 月時点で気象研究所のメールアドレスを登録しているユーザを気象研究所ユーザとし、気象研究所ユーザの更新した件数が気象研究所の更新件数としてカウントしている。

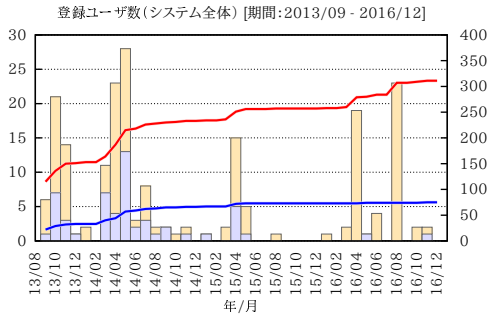

図 2.4.3　月ごとの新規ユーザ登録数（棒：左軸）と累計数（折れ線：右軸）。橙・赤は全体数を、水色・青は全体数のうち気象研究所の該当数を表す。ユーザ判定方法は図 2.4.2 と同じ。

る際に重要な情報源として活用できることから、検索性の面でもプロジェクト内の関係者が理解できる内容で登録することが望ましい。

以下に、開発管理サーバでのチケットを利用した典型的な開発プロセスを挙げる。もちろん、実際の開発手順は各コミュニティに委ねられているものであり、これは一例に過ぎない。

1. 課題発生（不具合修正、機能追加など）。
2. チケット作成。担当者割り当て。
3. 進捗を随時記録。
4. 進捗に応じて担当者を変更。
5. 第三者によるレビュー。
6. 課題解決。チケットを閉じる。
7. 一連の作業が記録に残ることで後から検索。

(3)　リポジトリとの連携

Redmine はチケット駆動開発に大変適している。チケット駆動開発ではチケットなしでコミットしてはいけないという「No Ticket, No Commit!」ルールがある (小川・阪井 2010)。これを導入することで、課題と成果の紐付けが可能になり開発情報が整理される。さらに、コミット前にチケットが作成されることで、作業内容が多くの関係者の目に触れることになり、作業に先駆けて問題点やアドバイスが受けられるなど開発の効率化が期待できる。また、レビューを経てからリポジトリを更新することを追加で義務付けることで開発成果の信頼性向上にも繋げることができる。

Redmine はチケットとリポジトリを連携する機能を提供しており、チケット駆動開発を強力にサポートする。一例として表 2.4.4 のようなコメントを入れることで、チケットとリポジトリを相互に参照することができる。最終的な開発成果は Subversion であればトランクへの、Git であればマスターへのコミットで開発が終了するが、開発管理サーバ独自の運用として、Subversion のトランクへコミットができるユーザをデフォルトではプロジェクト管理者に限定している。これによって、不用意にトランクが改変されることを防ぐとともに、開発計画と異なる改変がないかプロジェクト管理者が確認できるようになっている。

2.4.6　開発管理サーバの利用推進と利用実績

開発管理調整グループでは指針の取りまとめと前後して、開発管理サーバの積極的な普及のため、2013 年度から 2015 年度にかけての 3 年間、気象庁本庁と気象研究所で数値予報モデル開発者向けに利用者説明会を毎年開催し、開発管理サーバの利用目的、利便性、活用方法などについて精力的に紹介を行ってきた。こうした背景もあり、図 2.4.2、図 2.4.3 のように、2013 年度の利用開始以来チケット更新数、ユーザ登録者数が増加し、着実に気象庁内での利用が普及しているところである。特に利用開始からの 2 年間で急速に普及が進んでいることが見て取れる。なお、近年はユーザ登録者数の伸びも減少し開発への導入が定着している状況が伺える。こうした現状を踏まえ、2016 年度は数値予報モデル開発者全体への利用者説明会に代わり、新

規数値予報モデル開発者を対象とした研修で案内を行うよう変更するとともに、より効果的な活用方法についての検討会を気象庁本庁と気象研究所で開催する方針に変更している。こうした取組を通じて、開発管理サーバの運営方法の改善も含めた、効率的な開発方法の模索を継続している。

2.4.7 開発管理サーバの課題

開発管理サーバではユーザビリティを向上させるため、ユーザ登録メールやコミットメールの発出処理、自動バックアップ、管理者一覧情報の作成など、Redmine以外のツールも多数組み合わせて開発管理環境を提供している。また、Redmine自体の利便性を高めるため、サードパーティ製又は気象庁独自開発のプラグイン導入も行っている。こうした各種ツールは、ユーザの要望や障害対応などを踏まえて随時開発を行っているが、これらのツールの管理に開発管理サーバは一切用いられていない。

その最たる理由は障害対応である。開発管理サーバに関する開発情報は、開発管理サーバに障害が発生した際に利用できる必要があり、障害時でも参照できる場所に情報が存在しなければならない。そのため、開発管理サーバ自体の開発に関する情報は開発管理サーバの外で独自に管理を行う必要がある。これらの情報の多くは開発管理サーバ導入前の環境、すなわちRedmineによる開発プロセスが導入される前に整備された環境を利用して情報が集約されている。このため、現在では当然のように利用されているチケット駆動開発やレビューを行うための枠組みが存在しない。

開発管理サーバの導入から5年が経過し更新が予定されていることを鑑みると、今後開発管理サーバの移植に関する作業が増加することが予想される。こうした作業についても指針で定めたプロセスで管理ができるよう、既存の別サーバや開発管理サーバの副系を活用するなど検討が必要と考えている。

参考文献

Chacon, S. and B. Straub, 2014: *Pro Git, (CHAPTER 4: Git on the Server, The Protocols, The HTTP Protocols)*. 2nd ed., Apress, `https://git-scm.com/book/en/v2`, 127–129 pp.

小川明彦, 阪井誠, 2010: Redmineによるタスクマネジメント実践技法. 翔泳社, 190 pp.

Sussman, B.C., B.W. Fitzpatrick, and C.M. Pilato, 2011: *Version Control with Subversion: For Subversion 1.7: (Compiled from r5239), (1. Fundamental Concepts, Version Control the Subversion Way, Addressing the Repository)*. `http://svnbook.red-bean.com/en/1.7/svn-book.html`, 8–9 pp.

2.5 活用例 (1)–全球モデル [1]

本節では全球モデル (GSM) の開発における開発の管理手法について紹介する。まず開発管理の手法を発展させる必要性が高まった経緯と背景について解説した後、対応策として導入した開発の管理手法と検証環境について実例を交えながら紹介する。

2.5.1 背景

GSM は数値予報の中の基盤となるモデルであり、気象庁では 1988 年の運用開始以来継続的に開発・改良を実施している。2007 年 11 月には、その水平分解能を約 55 km から約 20 km に大幅に引き上げ、同時に運用を終了した領域数値予報モデル (RSM) と台風数値予報モデル (TYM) が担っていた役割を引き継いでいる (北川 2006)。この GSM0711[2] は開発の大きな節目であり、RSM, TYM の役割を統合した高分解能 GSM の運用を開始したことは数値予報システムとして大きな発展であった。

一方で、その後の GSM0808 (岩村 2008) において、並行して進めていた関連項目の開発が一段落して以降、GSM1212 (下河邉・古河 2012) まで、数年の間精度改善につながるモデル更新が停滞する期間が続いた（表 2.5.1）。停滞の原因を明確に特定することは不可能ではあるが、当時の開発で問題になっていたことの一つとして、単独の開発項目（あるスキームの更新など）を反映させた試験において、予測精度が大きく悪化する要素が見られ、その問題となる部分を克服できない例が続いたことがある。

例えば、GSM1212 以前の開発で問題になっていたが、その後解決に成功した例として陸面過程の開発が挙げられる。当時、新しい陸面過程を GSM の大気部分と組み合わせると、冬期に北半球高緯度の地表面付近で大きな低温バイアスが生じることが問題となっていた (平井・堀田 2009)。GSM には陸上で雲が少ないという問題があり、地表面へ入射する下向きの長波放射が過少であったため、本来なら旧陸面過程でも地表面では加熱が足りず低温バイアスとなるはずであった。しかし、旧陸面過程や旧境界層過程には、積雪の不適切な取り扱いが原因で地中から過剰な熱輸送を表現する、境界層での強安定時の拡散係数の下限値に非現実的な値を用いて大気から過剰に熱を輸送する、陸域の地表面の熱容量を大きく設定し温度変化を抑えるなど、長波放射過少を補償する不適切な取り扱いや非物理的な対策が多く存在していた。新陸面過程でより物理的に妥当な取り扱いをしても、長波放射不足の問題が顕在化して過剰な夜間の冷却、大きな低温バイアスが生じてしまう状況であり、地表面の放射収支に対してより正しく応答する新陸面過程が、従来の非物理的な調整を多く含む旧過程に対し予測精度で上回れない状況であった [3]。

表 2.5.1 GSM の変更履歴（GSM0711 以降）

	主な変更内容
GSM0711	水平分解能約 20 km、鉛直層 60 層、モデルトップ 0.1 hPa へ仕様変更 (北川 2007)
GSM0801	積雲過程の改良 (気象庁予報部 2007)
GSM0808	力学過程の改良、適合ガウス格子の採用 (岩村 2008)
GSM1011	入出力システムの刷新による高速化
GSM1108	出力専用ジョブ統合による高速化
GSM1212	層積雲スキームの改良 (下河邉・古河 2012)
GSM1304	放射過程（エーロゾル気候値、水蒸気吸収係数）の改良
GSM1403	物理過程改良（放射・境界層・重力波・積雲・陸面)、鉛直層 100 層とモデルトップ 0.01 hPa へ仕様変更、及び入出力システムなどの高速化 (米原 2014)
GSM1603	物理過程改良（積雲・雲・陸面・放射・海面）及び力学過程の高速化 (米原 2016)

このように、根本的な原因が解決されない状態で別の手段により打ち消されて「隠された」問題点は、compensating errors と呼ばれる (堀田・原 2012) [4]。GSM に compensating errors が多く内在しているであろうことは当時も推測されていたが、その要因は複雑に絡み合っており、単純に一つの部分だけを修正すれば全体的な精度改善が得られる状況ではなかった。各過程の担当者は、それぞれの過程の枠内での調査し改良を試みるものの、モデル全体を通した開発・調査は十分には進展しなかった。これには、計算機資源の制約により、複数の過程の変更を組み合わせた実験を十分に行えなかったことも関係している。項目の組み合わせにより必要な実験数は容易に増加してしまうが、水平格子間隔 20 km の高分解能 GSM を実行するには、当時の計算機では大きな資源を必要としたためである。実験結果が無い状態で不確実な推測ベースの議論を行うだけでは、開発者間の問題点に対する共通認識が十分に生まれなかったのである。

転機となったのが、2012 年 6 月 5 日の第 9 世代スーパーコンピュータシステムの更新 (西尾 2011) である。利用できる計算機の能力が飛躍的に向上したことによ

[1] 米原 仁

[2] 2007 年 11 月に運用を開始した GSM のバージョン。本節では、GSM の各バージョンを、改良を導入した西暦の下二桁と月を「GSM」の後ろに付けて呼ぶ。

[3] この問題の解決については本節の第 2.5.3 項で紹介する。

[4] 第 1.2.4 項にも解説がある。

り、それまでは計算機資源の制約のため実行できなかった改良項目の組み合わせ実験が可能になった。計算機の更新は、基礎開発を行い易くなったことも含めて、GSM 開発に非常に大きな発展をもたらしている。一方で、組み合わせ実験の計画と実施、結果の評価検証と分析、担当者間の議論を通じた相互のフィードバック、次の実験の設定といった開発サイクルを効率良く進めていくために、組織的・系統的な進捗の管理、評価検証手法の高度化、各種実験の実施をサポートするツールなど、開発管理手法を発展させる必要性が生じたのである。

このような背景があるため、GSM の開発プロセスでは、GSM を構成する各部分はお互いに強く関連しあっていることを意識し、全体を一つの「パッケージ」と考えて効率的に開発を行なうことに重点が置かれている。力学過程や陸面過程、積雲対流過程といった、GSM を構成する個別部品の開発をそれぞれ独立に行うのではなく、GSM 全体の問題意識を開発者間で密に共有し、組み合わせ実験と議論を繰り返している。パッケージとしての開発を成功させるためには、担当者がそれぞれの高い専門性を持つだけでなく、モデル全般に関する広い知識と経験に基づいて相互に活発な議論を行うことが重要であり、開発者間のコミュニケーションを促進することも開発管理手法の重要な役割の一つである。

第 1.2.4 項の図 1.2.1 に数値予報課における一般的な開発フローが紹介されているが、GSM の開発プロセスでは、その中でも特に性能評価試験を組み合わせ実験の対象として重視している。性能評価試験はモデルによる予測実験だけでなくデータ同化実験（全球解析）を含み、現業運用の形態にかなり近い試験である。全球解析では第一推定値や品質管理に GSM の予測値を用いるため、予測モデルだけを変更した場合でも、データ同化に利用される観測数をはじめとして、観測から引き出す情報量が変わり、解析値の精度自体が大きく変わり得る。同一の解析値を初期値に使用してモデルを比較する場合に比べて、データ同化実験を含めた試験では予測精度の差異がかなり強く現れるため [5]、変更のインパクトが性能評価試験を行って初めて分かることも多い。

また、評価検証は開発フローの一部であり、効率的な開発のためには検証システムの効率化・高度化も重要である。そのため、評価検証の点でも様々な取り組みを行っており、GSM 開発の標準検証環境である DPSIVS (Deterministic Prediction System Integrated Verification System) はその成果の一つである。評価検証についての取り組みも本節の後半で紹介する。

[5] これまでの開発の経験から、統計的なスコアの改善率が数倍程度異なることがあることが知られている。

2.5.2 開発の進め方

ここからは、GSM 本体の開発管理手法について具体的に説明していく。この手法は GSM1403 (米原 2014) の開発時から利用されており、GSM1603 (米原 2016) 及び開発中の次期 GSM の 3 世代の開発で用いられている。Redmine や各種ツール群は、第 1.2.4 項の図 1.2.1 でいうところの基礎開発の段階での活用や、第 2.2 節で解説されている一般的な観点でも利用されている。ただし、ソフトウェア開発として一般的な内容であるので本節では触れず、ソースコードに関するポリシーの紹介に留める。

(1) 開発の基本的な進め方

開発課題の進捗管理方法は、次の GSM へ導入を目指す短期課題とそれ以外の中長期課題で大きく異なる。中長期課題の計画と調整は、気象庁技術開発推進本部・モデル技術開発部会などで議論されている。短期より長い期間におよぶ開発計画は、基本的にはスーパーコンピュータシステムの更新に合わせ、次の世代の計算機で何ができるかを中心に組み立てられることが多い。ただし、複数年にまたがる開発項目である中長期課題の進捗管理は、基本的にそれぞれの開発者に任せられていることが多い。年度当初に各担当が開発会議で昨年度の計画からの変更点を示し、その後の進捗については定例ミーティングなどで報告するのが基本であり、Redmine やリポジトリも各開発者が使い易い方法で利用している。

一方、短期課題は Redmine 上でのチケット作成、共有リポジトリの利用が必須となる。図 2.5.1 に GSM の短期課題についての開発作業の流れを示す。GSM のバージョン更新には、1 年から 2 年程度の期間で区切りを付けている。はじめに、これまでの開発から得られた問題点などの知見を基に議論を行い、優先する開発項目を洗い出す。開発項目を次のバージョンの候補にする提案があった場合、その導入を目指すべきか関係者で議論して優先度を決める。このとき、科学的な正しさ、調査の進捗度合い、現在判明している問題点との関係などを議論するが、ミーティングにおいて議論が論争的であった場合は、その後もチケット上でフォローアップを繰り返す作業を行う。

各項目の担当者は、まずは基礎試験及び、必要な場合は単独で性能評価試験を実施して結果をまとめ、ミーティングで報告することが推奨されている。ミーティングは開催する機会が毎週決まった曜日に設けられており、希望者がいる場合に開催している。単独の試験結果に問題がまったくなければ良いが、通常そのようなことはあまりなく、想定していた部分での改善以外の、打ち消されて隠されていた誤差が顕在化する。そこで、開発者が相互にその問題点を共有し、組み合わせて考えるべき課題の選定や、問題解決に向けて必要な調査・開発項目の新規追加が議論される。そして再

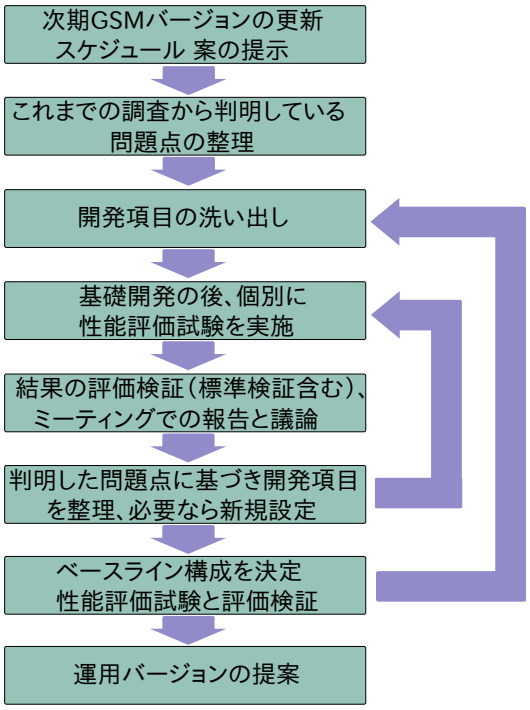

図 2.5.1　GSM の短期課題についての開発作業の流れ。全体が GSM の一つのバージョンの開発に相当する。

び、それぞれにおける基礎開発と性能評価試験を実施するステージに戻る。

開発と調査が進み、議論を経て運用バージョンとしての精度をある程度確保できる見込みが見えてきたら、その構成について性能評価試験を実施したのち、検証の結果を分析して、問題がなければ次のベースライン実験とする。ベースライン実験は、今後の開発のベースにすることが決まった構成による実験を意味し、更新後はその構成に改良を加える形で開発を進める。ベースライン実験への採用基準は変更内容に依存する部分も大きいが、

- 科学的に正しいアプローチか
- 変更の狙いとその影響が明確か
- 必要な精度評価がなされているか
- 予測結果に精度・特性上問題がないか
- 計算安定性や実行時間に問題がないか

といった点について説明し、開発者間のコンセンサスを得ることが基本になる。その時の議論に基づき、最終的には取りまとめの担当者が判断をしている。

ベースライン実験は開発者の共有アカウントで実施され、少なくとも標準的な検証が一通り実行される。その結果について開発者で議論し、問題点の整理と共有を行う。

この一連の手続に基づくベースライン実験の更新は複数回繰り返され、現業運用に問題がないと開発者間で認識を共有できる構成を作成した後、数値予報課内の承認プロセスへ進む。

このように、個々のコンポーネントの開発、取捨選択と組み合わせ、パフォーマンス調整という一本道の進行ではなく、評価検証によりそれまで隠れていた誤差を積極的に明らかにして問題点を開発者で共有し、柔軟に開発項目を入れ替えつつ全体パフォーマンスの調整を行うのが、近年の GSM 開発の特徴である。

(2)　Redmine プロジェクトの利用

解析システム、EPS を含む GSM 関連の開発において、プロジェクト管理・情報共有のためのソフトウェアとしての Redmine の利用は、当時の議論を受けて 2011 年 7 月から開始している (室井ほか 2013)。それ以前の開発情報の共有には、数値予報課が管理するサーバに Wiki を設置して利用するか、電子メールを用いて周知することが多かった。その後、試用期間を経て 2013 年頃から Redmine プロジェクト「全球モデル」として本格的な利用が始まっている。現在この Redmine プロジェクトは、GSM の開発だけでなく解析や全球アンサンブル予報システム、各種ポスト処理、検証システムの開発でも利用されており、2016 年度に利用実績のあるユーザ数は庁内他課室や気象研究所も含めて 30 名程度存在する。

プロジェクトは細かく分けず、GSM に関連する開発全体を 1 つのプロジェクト「GSM」としている。その他のプロジェクトには、数値予報の結果を用いたプロダクト作成処理の管理や、他課室を含めた気象庁全体での開発計画調整などが並ぶ。

一方で、サブプロジェクトは GSM 本体以外の大きな開発対象ごとに分けられている。項目を書きだすと以下のようなものが並ぶ。

- 全球解析
- 全球決定論的検証
- EPS
- アンサンブル検証

プロジェクトとサブプロジェクトはソフトウェアの機能的には同等であり（第 2.2 節）、両者の区分けに利用上の大きな違いはないが、GSM 本体の開発はサブプロジェクト全ての項目の基盤であると考えてプロジェクトに割り当て、その下に各種サブプロジェクトを配置している。

図 2.5.2 に全球モデル Redmine のトップページを示す。左側はよく利用されるページへのリンク集、右側はニュースの表示に利用している。

(3)　チケットの利用

開発における具体的な項目はチケットを用いて管理しているが、チケット利用のポイントはチケットを分類するカテゴリの使い方にある。例えば、モデルであれば GSM のバージョンごとにカテゴリを作成し、各開発項目がどのバージョンで完結予定かで分類している。カテゴリ名は、2017 年に完了予定の GSM 開発について「GSM17XX」と年単位で分けたり、全球 EPS の新規導入であれば「全球 EPS 導入」と開発の節目に

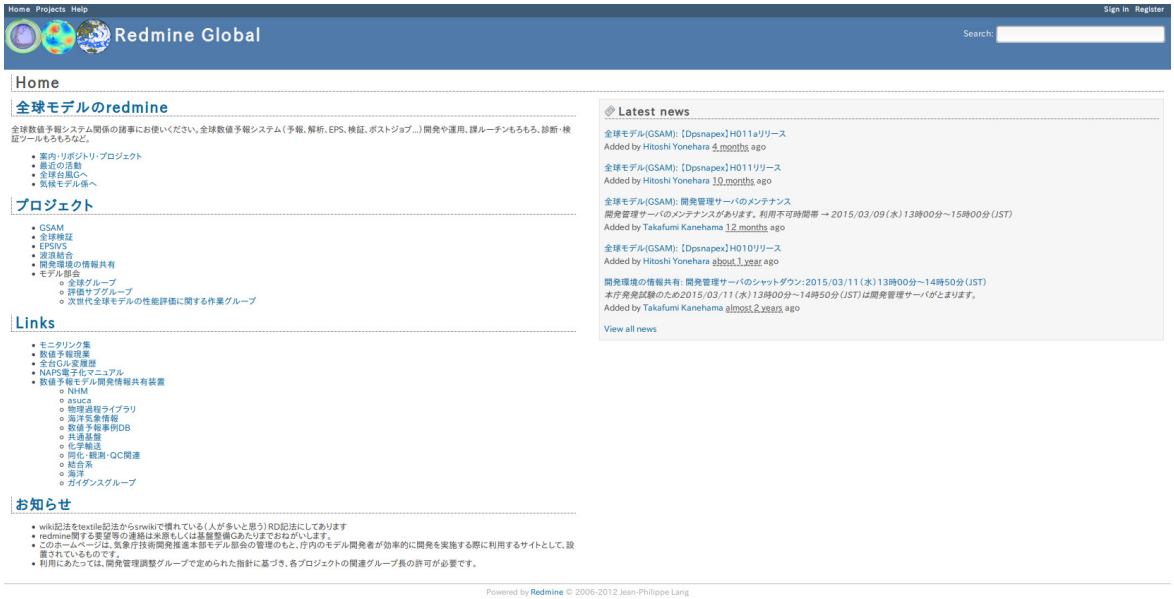

図 2.5.2　全球モデル Redmine のトップページ。

したりしている。具体的なスケジュールが未定のものは開発一般として分類しておき、次期版へ取り込みたい場合にはそのカテゴリへ変更する。ある時点で次期版での候補となっている変更内容・関連項目はカテゴリ別に一覧でき、開発者が相互に内容を共有しやすくなっている。

開発項目についてのチケットの粒度や書き方については特にルールは設けていない。大きいものでは「地表面摩擦の再考」など総合テーマが親チケットとなり、作業別に子チケットを多く持つものや、小さいものでは軽微なバグ修正など一作業に対応するものなどがあり様々である。ただし、実際には「GSM17XX における陸面モデルの改良」などのように、あるカテゴリに対応した大きめの開発項目ごとに作られるものが多い。チケットに関しては、カテゴリ設定が正確で、組み合わせ実験を行う時までに、組み合わせの単位となる変更内容が関係者に十分理解できるよう記載してあれば問題はない。

ステータスは「進行中」か「終了」、「却下」といったシンプルな分類を利用しており、ワークフローは用いていない。これは、バグ修正や機能追加といった状況が明瞭な項目以外の開発項目に関するチケットが多く、チケットの粒度は内容によって様々であるため、ワークフローを共通化しない方が良いからである。トラッカーはバグ修正、改良項目、話題 (TIPS) といった分類で大まかに利用しており、ほとんどの項目が改良項目に分類されている。日単位で管理する必要の無い項目がほとんどであり、緊急に対応すべきことも少ないため、開始日や終了日、優先度の項目を利用する必要性は生じていない。

個別の開発項目とは異なり、共有されるベースライン実験についてのチケット利用には一定のルールを設けている。夏冬の性能評価試験 1 セットに対して 1 チケットが割り当てられ、以下の項目が標準的な記載内容になる。

- ベースライン実験からの変更点
- ベースライン実験への変更内容の組み込み作業
- 内容のクロスチェック（レビュー）
- 実験のセットアップと実施
- 実験内容のレビュー
- 進捗の確認ツールのセットアップ
- 標準検証ツールの実施と結果のまとめ

変更内容が大規模で組み込み作業が多い場合は、その部分だけ別チケットにすることもある。ベースライン実験はカテゴリ名にバージョン番号を付けた名前で呼ばれる。例えば GSM1403 の開発では最初のベースライン実験が GSM1403v0 で、最終的には GSM1403v7 が現業運用版になっている。初回ベースラインには、現業運用版にその時の中心的課題の変更を加えたものが設定されるが、GSM1403v0 は鉛直層数増強（60 層から 100 層、モデルのトップを 0.1 hPa から 0.01 hPa）を中心として、放射過程や上部境界設定など関連する変更が適用されたものであった。

実際の利用例としては、GSM1403 カテゴリで 39 項目、GSM1603 カテゴリで 66 項目のチケットが立てられていた。

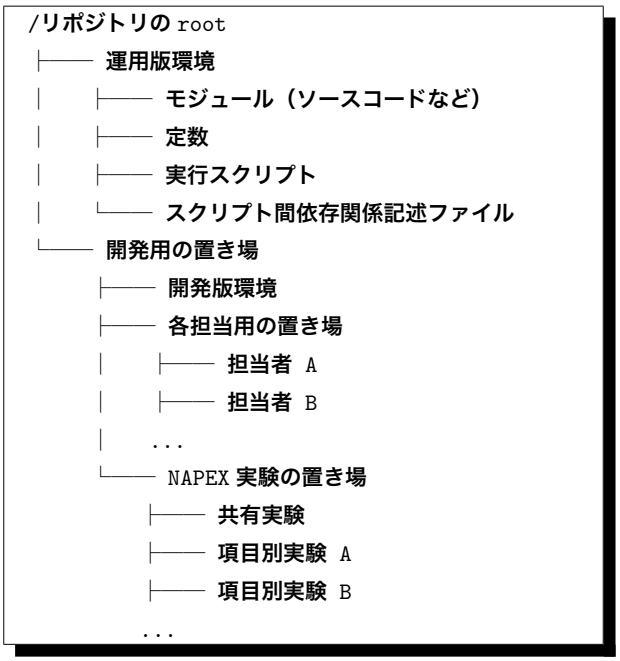

図 2.5.3 全球モデルリポジトリのディレクトリ構造。

(4) Wiki の利用

開発情報をまとめるため、カテゴリごとにポータルを Wiki で作成している。内容としては短期開発の大目標やスケジュールの記載、各種報告資料へのリンクの他に、組み合わせ試験の進捗や各課題の状況などがチケットへのリンクとしてまとめられている。現在は進捗管理の担当者が節目ごとに記載を追加しているが、記載漏れを防ぐ意味では作業の定型化をすすめて規則的に行うことも考えられる。

主な利用目的としては、定期的に開発作業に漏れがないかを確認することや、後で開発全体を振り返りやすくすることである。特に振り返りのための入り口を作成しておく価値は高い。最終的に採用された成果だけでなく、不採用となった項目や予測精度の悪化原因が十分に理解できなかった項目なども非常に重要な知見であり、振り返りが新しい示唆をもたらすことは多い。

(5) 実験内容の管理とリポジトリの利用

性能評価試験の実施には NAPEX（第 3.2 節）を利用しており、そのデータベースには全体の構成が記録されている。GSM の開発ではそれに加えて利便性向上のため、実験内容を構成するファイル群を数値予報モデル開発管理情報共有装置（以下、開発管理サーバ）の Subversion リポジトリ上で、GSM のモデルソースコード、周辺ツール類と同時に管理している。

GSM のリポジトリ構成としては、ソースコードだけでなく実験に必要なファイル全体を一つのブランチとして管理している点に特徴がある。そのディレクトリ構造は大まかには図 2.5.3 のようになっている。

ここで、「運用版環境」以下にある「モジュール」、「定数」、「実行スクリプト」、「スクリプト間依存関係記述ファイル[6]」のファイルセットが実験設定全体を構成する 1 単位である。これらのファイルは第 3.2.3 項 (2) で解説されている、数値予報ルーチン管理のツールに準ずる形式であり、NAPEX 実験にも対応している。この運用版環境へのコミットには開発コミュニティの承認が必要であり、原則として現業システムへ変更申請を行い、プログラム班のチェックを通過したものがコミットされる。

「開発用の置き場」の下には、「開発版環境」とそのブランチ置き場及び、「NAPEX 実験の置き場」が存在している。開発版環境とは、運用版環境に対してモニタ出力の追加や低水平分解能モデル、理想試験環境、サポートツールなどの開発用の変更を加えたものであり、最新の開発成果はまずそこへコミットされる。開発版環境へのコミットは、結果に影響のない修正についてはチケットなどへ報告したのち随時行われるが、結果に影響のあるものについてはミーティングなど関係者の議論と承認、コードレビューを得てコミットが行われる。それぞれの開発者は開発版環境を別ブランチとしてコピーして利用する。

基礎開発を行い変更内容が固まれば、次は性能評価試験へ進む。まずはベースライン実験を各担当者の実験の置き場にリポジトリ上でコピーしたのち、そこに開発項目についてのブランチから変更内容をマージすれば良い。こういった使い方を可能にするため、実験全体を構成するファイルセットを 1 単位としてリポジトリで管理している。図 2.5.4 にリポジトリの各ブランチの分岐と合流の流れを示す。この流れに沿って開発を行う限り、1 つのリポジトリ上に現業運用版、開発版、各開発者が試したもの、性能評価試験が行われたもの全てが記録されることになる。チケットの記載などと突き合わせて、過去の実験を再現したり内容を確認したりすることも容易である。また、この方式には NAPEX 実験間のマージにおいて Subversion の機能を利用できること、Redmine 上のリポジトリブラウザなど、リポジトリ可視化ツールを通じて組み合わせ実験の履歴や差分などを確認できることなどの利点もある。

(6) 開発を補助するツール群

効率的に開発を進めるために、共通して行う作業については補助ツールを整備して共有している。これらのうち、低水平分解能 GSM の実行環境とシングルカラムモデル[7] を含む各種の理想試験環境は、開発版環境に含めている。ツールの共有により開発効率を高めるだけでなく、実験時に単純なミスが発生するのを防止している。

低水平分解能 GSM としては幾つかの設定が用意さ

[6] 全球解析と全球モデルの実行を含むため、同時並列で実行される大量のジョブを含む。
[7] 鉛直 1 次元のモデルであり、何らかの外部境界条件の下でパラメタリゼーション手法の試験を行うもの。

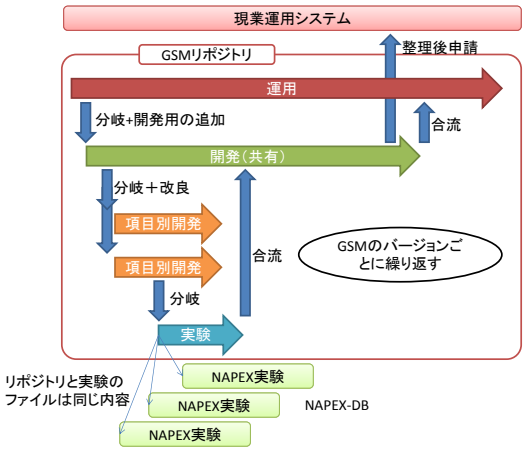

図 2.5.4　全球モデルリポジトリのブランチの分岐と合流の流れ。NAPEX-DB は NAPEX の実験情報を登録しているデータベースを指す。

れている。その中には、デバッグ用の TL31（PC 上で実行可能）、1 年積分実験や AMIP（Atmospheric Model Intercomparison Project）型実験の実施に用いる TL159、1 か月積分試験に用いる TL319、EPS のコントロールランに相当する TL479 が含まれる。鉛直層数は現業運用版と同じ 100 層で全て固定している。これら低水平分解能 GSM を含めて、GSM の試験実行ツール（TestHarness）が用意されており、スクリプト中の実験名と実験タイプなど数行を編集するだけで利用できる。TestHarness はジョブ間の依存関係や利用する実行モジュール、定数ファイル、初期値などを自動で判別する機能を持ち、環境をスーパーコンピュータ上に配置したのちジョブを登録・実行する。このツールを利用することにより、手元のファイルさえきちんと更新しておけば、実験を実行する計算機上へのコピー忘れや実験環境の構築ミスなどは防止できる。単発試験に必要な初期値（低水平分解能含む）は開発者で共通の事例を調べるため、性能評価試験の対象期間中で用意されているが、必要があれば現業運用データから望んだ日時の初期値データを作成するツールも用意されている。開発者は自分の課題の開発ステージや利用可能な計算機資源を考慮しつつ実験タイプを選んで基礎的な試験を進める。

結果を簡単に確認するための可視化には、画像作成スクリプトやウェブブラウザを利用した画像ビューアが用意されている。スクリプトを実行すれば海面更正気圧と降水量や上中下層雲量、対流圏の代表的な面における気温、湿度、高度場、風などの標準的な分布図が描画されると同時にそのビューアが設置され、手間をかける必要はない。より詳細に確認する場合は、TAG（第 4.6 節）を利用したリアルタイムモニタである DynaMo による描画が可能である。図 2.5.5 に DynaMo による描画例を示す。このモニタはセットアップ用のシェルスクリプトにデータの置き場所などを記載して実行するだけで利用でき、GSM から出力される要素を網羅的に水平面表示するだけでなく、実験間の差分や解析値に対する誤差、任意の点間の断面図も瞬時に表示可能である。また、ウェブブラウザ上からの操作で表示領域の変更や拡大・縮小、図の並び替えも行なうことができる。このツールは開発者が実行した予測実験だけでなく、現業運用 GSM の結果や性能評価試験の結果確認にも利用されている。ただし、DynaMo は動的なモニタであるためデータが保存されている間に限られる。この他、GSM の出力は気象庁独自形式である NuSDaS 形式（第 4.2 節）であるため、データを NetCDF など描画ソフトから利用可能な形式へ変換するツールも整備されており、必要に応じて GrADS などの各種描画ソフト（第 4.3 節）も利用している。

更には、リポジトリ上の単位ファイルセットから、NAPEX への実験登録を自動で行うツール (dpsnapex) が整備されており、円滑な実験セットアップを補助している。このツールは手元のファイル群から NAPEX へ登録すべき差分を自動で判別して設定ファイルを作成・登録する。利便性が向上するだけでなく、登録漏れが起こる可能性を排除している。NAPEX では実験名やデータ保存場所、保存するデータの種類など利用者が自由に設定できるが、dpsnapex では設定を簡素にすることによって利用状況が複雑になるのを防いでいる。NAPEX をそのまま使うより簡便なので、開発者は自然と dpsnapex を使い、結果として設定に秩序が生まれる。また、このツールを使うと GSM のリポジトリ上のどのパス、どのリビジョンの実験が登録されたかが、NAPEX のデータベースのコメント欄に自動で記載され、NAPEX 側と GSM のリポジトリの対応を明確にしている。

2.5.3　開発の進め方の実例

開発の進捗の流れの例として GSM1603 の開発履歴を紹介する。GSM1603 は 2014 年 4 月に開発が開始された。2014 年 4 月の全球・台風グループ開発会議で次期 GSM の開発予定案が提示されるとともに、GSM1403 の検証結果に基づいて、熱帯の対流圏中・下層を中心とした低温バイアスや台風の発生・発達の不足など、解決すべき課題が共有された。同時に、当時取り組んでいた開発項目の一覧が更新され、短期課題とする候補が選定された。このとき、新しい陸面過程の導入、新海氷スキームや開水面 (open water surface) と海氷の両方が混在する状態を扱う混在格子の導入、放射における雲量オーバーラップ手法の改良、雲粒の光学的特性の取り扱い精緻化、雲過程・積雲過程の大規模改良など多くの項目をリストアップしている。これら項目の多くは、GSM1403 での導入を目指していたものや、同時並行で開発が進められていたものであり、この時点で各項目のチケットは既に作成されていた。GSM1603 開発のカテゴリが新設され、それぞれのチケットはそ

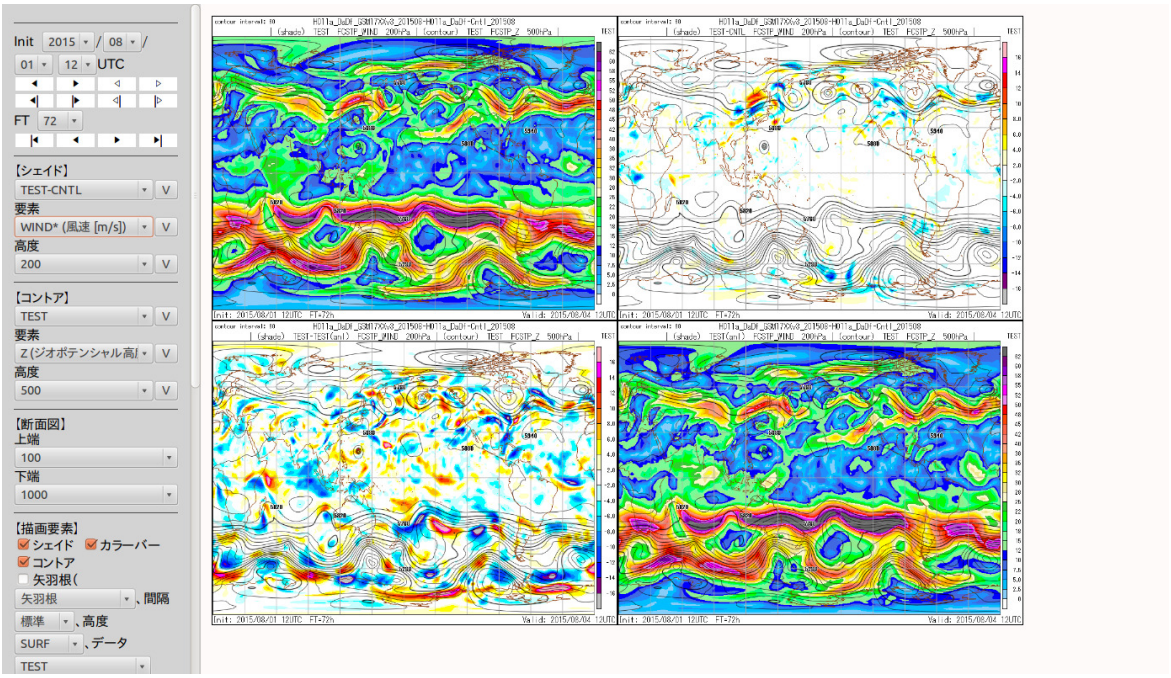

図 2.5.5　DynaMo の描画例。性能評価試験の結果について、等値線で 500 hPa 高度場、色の塗り分けで 200 hPa 風速を表示している。左上が予測値で、右上が対照実験との差分、左下が誤差（解析値との差分）、右下が解析値の図である。

のカテゴリに所属するように変更された。

2014 年 4 月以降、個別項目の基礎開発が進められるとともに、性能評価試験とその評価検証が行われた。性能評価試験の実施には dpsnapex を用いており、開発管理サーバ上で変更内容を共有する時には Subversion のマージ機能を活用している。評価検証では DPSIVS（第 2.5.6 項）を必ず実行し、ミーティングにおいて共通の物差しで議論するための資料にしている。この作業を繰り返してベースライン実験を更新していったのであるが、GSM1603 は最初のベースライン (v0) の構築に 1 年以上の期間を要し、2015 年 6 月に構成が決まっている。その v0 から始まり、v7 まで実験が行われ、最終的には v5 を現業運用版として採用している。この時に行った一連の議論で用いた資料はチケット上で共有しているため、詳細を振り返ることが可能である。

v0 は、GSM1403 に放射、雲、積雲、海面の諸過程の改良と、新しい陸面、海氷過程の導入を含む大規模な変更を施したものであった。v0 を構築するまでには大量の実験と議論が行われているが、地表面付近の低温バイアスに関する視点を中心にその流れを紹介する。

新しい陸面過程導入時の問題については前述したが、境界層過程の強安定時の熱の過剰輸送が GSM1403 で解決されるなど一部進展はあったものの、低温バイアスは依然として大きな問題であった。陸面過程自体もフラックス交換スキームや植生パラメータなど様々な改良と調整が試みられていたが、最終的な問題の解決には、海面・海氷過程と、積雪・土壌での長波射出率、雲氷落下スキームの 3 つの改良が大きく寄与している。

GSM1403 の海面・海氷過程、特に海氷のスキームには大きな問題があり、地表面付近の低温バイアスの原因となっていた。旧陸面過程は冬期に北半球高緯度で高温バイアスを持ち、両者でバイアスの方向が逆であったため表面化していなかったが、新陸面過程は低温バイアス傾向であり、相乗的に低温バイアスを拡大する状況になっていた。GSM1603 の海面・海氷過程では、開水・海氷混在格子、接地境界層における輸送係数の計算手法の改良、氷 4 層と表面を取り扱う新しい海氷スキームなどが導入されて、海面・海氷過程由来の低温バイアスが大幅に減少することが確認された。そのため、海面・海氷過程と陸面過程は一体の開発として扱うこととし、以降の実験は必ず組み合わせて行うことになった。

GSM1403 では地表面は黒体であり、長波の有効射出率は 1 として取り扱っていた。しかし、この近似は精度が良いものではなく、特に陸上では長波放射による冷却が過剰になる原因の一つになっていた。この点を変更するには、放射、陸面、海面、海氷の各過程をまたぐ改変が必要であり、これまで積極的な開発項目としては挙げられてこなかったが、低温バイアス対策のため各過程の担当が共同して改良を行っている。

雲氷落下スキームの改良は、そもそもは熱帯上層雲量が過少な問題と、スキーム自体が持つ時間積分間隔の依存性低減を目指したものである。この変更単独では熱帯対流圏上層の高温ドリフトなどの問題があり、積雲過程とセットで開発と調整が進められていた。一方で、この改良の結果、雲氷はよりゆっくり落下するようになるため、高緯度側でも雲量が増加し、地表面での長波放射収支が改善する。陸面過程の開発において

下向きの長波放射の過少は大きな問題とされていたため、積極的に組み合わせて試験を進めることになった。

上記を含めて多くの実験と評価検証、議論を行った結果、v0 の構成について開発者の認識が共有されたため、実験環境の構築を開始した。実験ブランチに各自の開発ブランチから変更点がマージされると同時にソースコードの変更点についてクロスチェックも実施しており、それらの構築作業に 2 週間程度、その後の実験実行にさらに 2 週間程度を要している。評価検証の結果、概ね想定されたインパクトが見えていることが確認されたが、一方で熱帯陸上の降水過剰、地表面への短波入射過剰に伴うユーラシア中央の地表面付近での高温バイアス、氷床での予測誤差の増加など、様々な問題が局所的ではあるが見つかったため、今後の課題に設定された。

次のベースライン実験となった v1 は、2015 年 8 月にその構成が決まっている。このバージョンでは課題への対応として、陸面過程のフラックス交換スキームの再調整、氷床アルベドの再設定などに加えて、陸上地表面への短波入射が過剰な原因の一つとなっていた、雲オーバーラッピングのパラメータについての再調整が取り込まれた。

2015 年 11 月に実施された v2 では、雲放射における雲粒サイズ診断と雲粒光学特性式を改良し、短波放射に対して水雲が光学的に厚くなる変更が取り込まれている。また、運用版とすることを念頭に、積雲過程の改良により増加した実行時間を運用上の制限の中に収めるため、ルジャンドル変換の高速化などが取り込まれている。変更内容に合わせて、正確な実行時間計測などの作業もチケットに記録されている。

2015 年 12 月から 2016 年 1 月の間に実施された v3 から v5 のベースライン実験では、台風の発達を抑制する幾つかの対策が取り込まれた。v0 の段階では、積雲・雲過程の改良により台風の発達がより強く表現されるようになっていることが確認されていたが、その後実験を進めるうち、過発達した台風周辺での計算安定性に大きな問題が存在することが判明し、その対策が緊急の課題となった。この時の作業と議論はかなり厳しいスケジュールで行われたのであるが、情報共有や実験の管理にはリポジトリとチケットが有効に利用されている。緊急時ほど進捗管理の正確さが求められるため、システマティックな開発管理の有効性は増す。また、このとき採用には至らなかった v7 も、その後問題点の修正や構成の整理が行われて次のバージョンのベースになっており、記録された内容は有効に利用されている。

以上 GSM1603 での実例を紹介したが、開発の履歴は Wiki やチケットの記載から辿れるようになっており、過去どのような点に問題意識が持たれ、どのような解決策が模索されたのかが確認できるようになっている。GSM1603 ではこの他にも積雲・雲・放射にまつわるものなど幾つかの compensating errors を巡って議論が行われている。一連の開発では、性能評価試験を実施した構成だけでも 50 以上の試験が存在しており、低水平分解能での試験や予報実験のみ行ったものも含めるとその数倍になる。これらの試験群は、思い付くものを総当たりで流れ作業的に試したわけではなく、モデル内の各プロセスの問題点を一つ一つ洗い出し、各開発項目を組み合わせた時に何が起こるかを想定しながら試験を進めたものであり、複雑な作業工程を伴っている。このスケジュールでの一連の開発プロセスは、開発管理手法の発展なくしては実現できなかったであろう。

2.5.4 ソースコードの管理

GSM のソースコードの管理方針やルールについて、その現状と狙いを解説する。

(1) リポジトリの利用

GSM のソースコードに対しては、2002 年から CVS (Concurrent Versions System) を使ってバージョン管理が行われていた。当時は、モデルの大きなバージョン更新ごとに履歴を引き継ぎながらリポジトリを乗り換えていた。2010 年 1 月から管理ソフトウェアを Subversion へ変更し、その後開発管理サーバ上へ引き継がれている。定数作成や格子変換を行う GSM の周辺モジュールに関しても、当初は実行モジュールごとに管理方法が異なっていたが、共有可能な Fortran モジュールは共有することとし、一括して GSM と共通のリポジトリで統一的に管理する方針に変更している (宮本 2009)。

GSM ではトランクという概念は明瞭には利用していない。開発ラインを束ねる意味では運用版 (ope) が存在し、共有される開発の先端は次期バージョンのブランチ (dev) が相当する。一連の開発終了時に dev がそのまま ope になるわけではなく、不必要な部分の絞り込みと現業運用モデルとしてのブラッシュアップが行われて差異が生じる。開発で膨らんだ冗長な部分の整理も兼ねて、次の dev は ope から分岐させて再構築している。

(2) システム間での住み分け

現状、最新の GSM をそのまま利用するのは、全球数値予報システムと全球アンサンブル予報システム (GEPS) の二つである。それ以外にも GSM を利用するものは多く存在するが、過去バージョンの GSM にそれぞれが必要とする改変を加えて利用しているものが多く、最新の GSM への追随までの期間は長い。このため、システム間の住み分けについては直接関連する 2 システムのみを考慮している。両システムでソースコードの開発時系列は同じであり、リリース時期が異なるのみである。通常は、まず全球数値予報システムへ新 GSM が導入された後、GEPS へ導入されるが、タイミングによっては GEPS が先行することもあり得

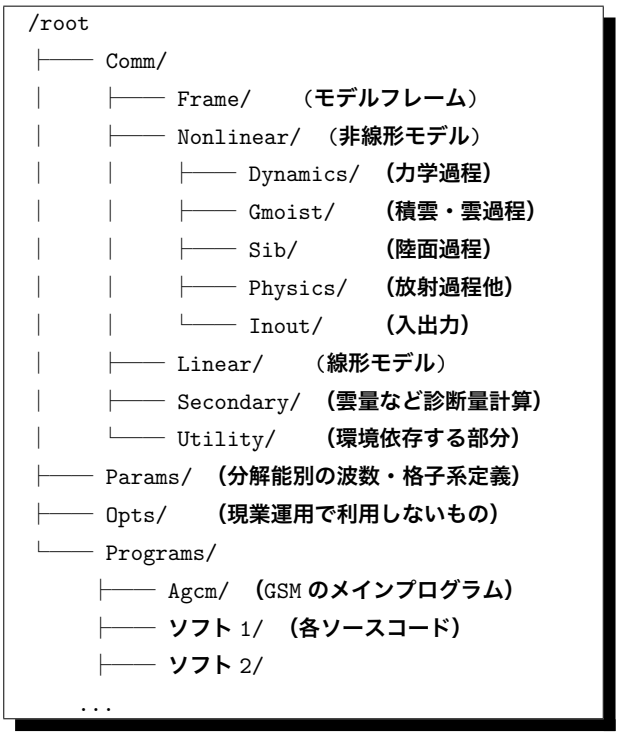

図 2.5.6 GSM のソースコードを格納するディレクトリ構造。

る。システム間の違いは、ネームリスト及び定数ファイルで制御するが、積分時間やプロダクト、モデルアンサンブル摂動に関わるものだけであり、時間発展演算には水平分解能以外に本質的な違いはない。

(3) ソースコードのディレクトリ構造

GSM のソースコードを格納するディレクトリ構造について解説する。基本的な方針としては、GSM の本体だけでなく関連ソフトウェア全体で分かりやすい配置になることを目指している。参考であるが、GSM1603 のソースコードは関連ソフトウェアを除いた本体部分のみでおよそ 10 万行、3.6 MB であり、このうち 3 割はプロダクト作成・出力部分である。図 2.5.6 に、現業運用版の構造の概略を示す。

Comm 以下に GSM のほとんどの部分（メインプログラム以外）が収められているが、これら以下の部分は変分法解析、格子変換、オフライン陸面モデルなどの別アプリケーションで共有することを想定しているためである。関連ソフトウェアのソースコードは、Programs の下のそれぞれの名前のディレクトリ以下に配置されている。

モデルフレームとは座標系、物理定数、通信用関数、球面調和変換用関数、時間情報等を一元管理する Fortran モジュール群のことであり (宮本 2009)、他のものとは別にまとめて配置されている。宮本 (2009) では、GSM0711 までの GSM では各過程の開発者が類似の変数を独自に利用しており混乱を招いていたが、それを教訓にモデルフレームの概念を明確化した上で整理を行ったことが報告されている。

Nonlinear の下には力学過程、物理過程、入出力など、大多数のソースコードが含まれる。力学過程は時間積分制御とコア部分、物理過程は放射・境界層・重力波のグループと、雲・積雲対流のグループ、陸面の部分に分けて配置されている。区分けからも想像できるが、ソースコードのサイズや行数について、陸面過程は物理過程全体の 6 割程度を占める。Inout にはモニタシステム及び、各プロダクト作成モジュールが配置されている。

開発版は現業運用版に理想試験用ソースコードなど様々な機能を追加したものになっている。ただし、現業運用版に含まれるものはほとんど変更せず、サブルーチンの呼び出し元が追加される形にする。これは開発版からの現業運用版を作る作業の時に、ソースコードの齟齬がなるべく発生しないようにするためである。

(4) GSM の実装に関するルール

GSM の実装について、採用されているルールを解説する。これらルールは、現業運用されるソフトウェアとしての品質を保つため、実装内容の維持管理を行いやすくすること、運用上の問題発生を抑制することを目的に定められている。関連するソフトウェアについても事情は同じだが、コード規模が小さいこともあり、GSM 本体ほど厳格ではない。

オプションは必要最小限とし、現業運用や開発で利用しないものは削除するのが原則である。開発で利用する場合でも、利用者が幅広く存在するものでなければならず、基礎試験で必須のものにほぼ限られている。仮にオプションを増やす場合は、必ず他のものを減らせないかを考慮する。また、特にプリプロセッサでの分岐についてはその利用方法を限定しており、I/O 周りの一斉切り替えと計算機・実行環境ごとの特殊な手当てのみに用いている。これは、プリプロセッサはその仕組み上、影響範囲が広がりやすい上に予想外の動作を引き起こしやすく、またコンパイラによるチェックが機能しにくいことが理由である。この方針はコードの可読性向上や想定外のトラブル防止には効果が高く、現業運用モデルとしては重要な方針である。一方で、開発時には不便な面があるため、開発版のブランチでは適用していない。ただし、運用開始時には整理する必要があるため、開発途中でもこまめに整理するのが通常である。

時間積分で引き継がれる予報変数は全体で共通管理とし、各モジュールが独自に持つことを許可していない。これは、時間積分やリスタートに関係する部分は分かりにくいバグの混入や各過程間の取り扱いの矛盾などトラブルが起こりやすいことから、各過程には任せないことにしているためである。また、必要な計算機資源を見積もるときに、入出力に関する部分の透明性を確保することが必要なことも理由である。ただし、この点については今後の GSM の高度化に伴って議論

が必要かもしれない。地表面過程などが多くのサブモデルを抱えた場合、独自に扱う方が簡便であるという意見はある。

ファイル入出力を行う部分は担当者の責任分解点を明確にするため、作成するプロダクトごとに一つの Fortran モジュールを用いることにしている。これは、かつて統一的な入出力機構を用いていた時代に、あるプロダクトのための変更が別のプロダクトへ副作用を起こしてしまい混乱が生じた反省に基づいている。ほとんど同じ処理であっても、ソースコードの共用は原則行わず、変更する可能性のある診断処理などは共用しない。ただし、内挿サブルーチンや一般的な気象要素変換など変更の見込まれない一部のものは共用している。入出力周りは現業運用や開発時に計算機利用上のトラブルが起こりやすいため、予測値の出力だけでなく、モデルのログ出力も全体で統一して共通に管理している。

(5) プログラミング作法

プログラミング作法（コーディング規約）については特定のものを採用していない。もちろん、無秩序ということではなく、

- Fortran90 の規格に従うこと
- 他人が読み、メンテナンスすることを念頭にソースコードを書くこと
- 現行ソースコードの慣習を尊重すること（全面改変は妨げない）
- モデルフレームで管理されている変数に類似した変数名は利用しないこと

といった基本的な事項は指定されており、またバグ混入を減らすための努力は当然求められる。抽象的な表現にとどまらず具体的に決めるべきという話もあるだろうが、コミュニティはオープンではなく限られた範囲に止まるので、不適切なソースコードが提案されたときにはクロスチェックの段階で指摘されるため問題にはならない。結局のところ、保守性、可読性や判読性をどうすれば確保できるかは文脈依存であり、汎用ルールで書き尽くせるものではないし、本質を理解していない作業者が細かいルールに従ったとしても、作業速度が落ちるだけで現業モデルとして品質の良いコードになるわけでもない。

この方針も、既存のコーディング規約にこだわりすぎて開発の効率が悪化した事例の反省に基づいている。現状ではコーディング規約を採用する予定はないが、採用する場合にはコーディング規約の改訂手順を明確化した上で、定期的なブラッシュアップの実施、確認コストを減らすための自動検査ツールの導入、柔軟な例外規定を定めて水掛け論発生を防止する工夫などが必要と考えている。

(6) ソースコードの整理とレビュー

あるバージョンの GSM の構成が決定された段階で、ソースコードが十分に整理されていることがもちろん望ましい。しかし、開発の節目で行われるクロスチェックやコードレビュー時にある程度整理されるとはいえ、開発の締め切り間際において時間的制約により対応を延期する項目も多い。例えば、必要なコメントの不足、変数の使い回しなど不必要な複雑さ、内容を想像しにくいファイル・サブルーチン名など計算結果や速度に影響のない部分は後回しにされやすい。そのため、新しいバージョンを立ち上げた時に、まずソースコード整理を行うようにしている。その他にも、他の計算機環境でのデバッグ結果がフィードバックされたタイミングなど、積極的に整理を行っている。近年では固定形式など FORTRAN77 までの制約に基づいて記述された部分についても整理が進み、次期 GSM ではほとんどの部分が Fortran90/95 以降では推奨されない文法を排除したものになる予定である。

(7) ドキュメント

GSM 及びその周辺プログラムに関するドキュメントは、開発管理サーバ上のリポジトリで管理している。リポジトリの構成には特殊なものは用いておらず、標準的な trunk, branches, tags 構成である。使用するファイルフォーマットは特に定めていないが、差分管理が可能である点や数式記述の利便性から LaTeX が用いられることが多い。

ドキュメントを作成するタイミングについて特に明確なルールは無いが、クロスチェックやコードレビューを受ける段階である程度のメモは作成しておく必要があるため、その内容が反映される GSM バージョンのソースコード凍結までには最低限のものは作成されている。GSM 全体のドキュメントも簡単なものが整備されているが、その更新はやや滞りがちであり、開発の進捗とドキュメント作成をどのように無理なく紐付けるかは今後の課題である。

2.5.5 開発の進め方のまとめと課題

ここでまでは GSM の開発管理について、その背景、進捗管理、プロジェクト管理ソフトウェアやリポジトリの利用、ソースコードの管理方針について紹介した。現在の手法は、GSM 全体を「パッケージ」として効率的に開発することに重点が置かれている点に特色がある。これらの開発プロセス、開発基盤環境はコミュニティ全体の開発効率の向上に貢献しており、今後もブラッシュアップしながら利用を継続することが重要と考えている。

今後の課題であるが、大きな枠のものとしては GSM の中長期的な開発課題と短期的な開発をどうすり合わせていくかという点がある。本節で解説したように、現在の開発の進め方は 1, 2 年単位の短期的な開発に特化しており、中長期的な開発課題については考慮され

ていない。今後この点について考え方を整理していく必要があるだろう。

開発項目の選定についても課題がある。研究コミュニティにおける自然科学の発展や他の現業センターの動向などを背景に、モデルの既存の問題点とユーザからの要望を整理し、対処するためのアプローチとその進捗を整理した上で行うのが理想的である。現状ではその部分が十分システマティックではないため、議論の進め方や共通認識の形成手続きなど工夫の余地があるように思われる。ただし、私見ではあるが、GSM は気象庁の基盤モデルとして多くの役割を担っているため、ユーザからの要望に優先度をつけるための議論には多くの困難があると思われる。GSM は短期予測から中期・延長予測まで様々な時間スケールを予測対象としている。数日先までの台風予測精度の向上など、日本の防災に直結する項目の優先度が高いことは言うまでもないが、その他にも様々な要望に対してどのように優先度をつけて開発（評価）を行なうかは非常に難しい問題であろう。複数の指標を何らかの整理に基づいた重みで組み合わせて、統一的な数値指数 (Index) を作成する方法なども考えられるが、整理をつけることは非常に困難であるし、数値指標の短期的な向上だけを過剰に重視した短絡的な開発に陥ってしまう危険性もあるため、慎重な検討が求められるであろう。

現在取り組んでいる課題として、実験の評価検証の標準化と拡充がある。開発フローの中で、性能評価試験へ至る前段階では、共有の実験・評価が決まっていない。もちろん、その部分で行うべきものは開発項目に強く依存するので、一律のものが定められるわけではないが、幾つか共有の実験設定、特に気象予測実験だけでなく気候予測実験が用意されているべきという議論がある。その一つとして、現在低水平分解能モデル (TL159) での一年積分を評価する実験・検証環境の作成や、AMIP 型実験・検証環境の整備が進められている。

また、気象予測実験についても直近の夏冬だけを対象にするのではなく、再解析を初期値とした実験を行い、GSM の改良内容のインパクトが、エルニーニョ・南方振動やマッデン・ジュリアン振動などスケールの長い現象の状況に依存しないかの確認を行なうべきといった議論や、事例依存性の強い台風進路予測などの現象については低水平分解能でもっと多くの実験を行うべきといった議論がある。最終的には利用できる計算機資源と開発へのフィードバックのバランス次第であるが、検討を続けていく予定である。

2.5.6 検証と検証システムのあり方

ここからは、GSM 開発における検証について、そのあり方と現在の標準検証システム DPSIVS を、特に開発管理の観点から紹介していく。

(1) DPSIVS

DPSIVS は、性能評価試験を主な対象とする、全球決定論予測についての標準検証環境である。近年の GSM 開発においては、複数の改良項目を組み合わせて、compensating errors を解きほぐしながら改良を進めることが鍵となっている。その開発プロセスでは、変更項目を色々と組み合わせた実験を実施するが、実験ごとに結果の評価検証と分析、担当者による議論を通じたフィードバック、次の実験の設定といった手順を踏む。これを繰り返すサイクルにおいて評価検証と分析の作業量は大きなウエイトを占め、検証システムが効率的であることは開発全体を効率的に進めていく上で大切なポイントである。DPSIVS は性能評価試験を主な対象としたツールであるため検証においては、変更された実験 (TEST) を、対照実験 (CNTL) と比較することが基本になっている。

(2) DPSIVS 開発の経緯

DPSIVS は、全球モデルについての評価検証への問題意識を基に 2013 年 3 月から開発を開始している。

GSM 検証ツールの歴史を大まかに振り返ると、初めは開発者が各自・各グループで独立に整備・利用しており、開発者ごとに利用するツールが異なる状況であった。その状況から、山下 (2013) で紹介されている全球実験標準評価モニタ、通称 SPV (Super Verif) が 2004 年に当時の検証ツールを取りまとめる形で整備され、必要最低限の項目について指標・図表の統一がなされた。SPV 作成時の問題意識は、最低限検証すべき項目の標準化や、検証モジュールを統合することによる信頼性の向上、ユーザの利便性向上などが挙げられる。SPV が整備された恩恵は非常に大きかったが、その後幾つかの問題が意識されるようになった。問題の一つが、SPV の検証内容は必要最小限 [8] であり、総合的な評価を目的としたものではなかったが、当初の想定を超えて開発における評価指標として過度に重視された点であった。本来、評価検証は様々な面から行われるのが望ましいが、過去には試験の検証を利用しやすい SPV で取り扱える評価指標のみで行うなどの例も散見された。そこで、問題意識を持った開発者により、補完的な追加検証のツールが複数作成されたが、それらの統合には至らず、開発者ごとに利用するツールが異なる状況であった。

一方で、気象庁内では、2010 年頃より全球モデル開発のための評価検証について議論が深まりつつあった。2010 年 1 月以降、全球モデルの誤差特性について議論する全球評価会合が数値予報課と気候情報課共同で定期的に開催され、モデル開発のための評価検証を充実させていくことの重要性が確認された。その流れは数

[8] どのような内容かは山下 (2013) に解説がある。予測スコアとしては、海面更正気圧や 500 hPa 高度、850 hPa 気温などの要素のアノマリー相関係数差や平方根平均二乗誤差の改善率が主な対象となっている。

値予報課報告・別冊の第 58 号、第 59 号にも繋がっており、特に第 59 号では「モデルの総合的な検証を通した物理過程の開発」が主要テーマの一つに掲げられ、気象庁内の評価手法についてのレビューや新しい検証手法の紹介が行われている (原 2013)。

このような状況のもと、新たな評価検証手法を取り込んだ統合的な検証環境の整備を目指して DPSIVS の開発は始まった。拡張性を始めから考慮して設計し、統一的な検証環境を提供することにより、開発者が利用するツールを共通化することも目標の一つであった。現在でも、数値予報課および気候情報課の開発者が連携しつつ、試験結果を検証する標準検証環境として開発・整備が進められている。

2.5.7 DPSIVS の考え方

ここでは、GSM の開発を効率的に進めるために、DPSIVS がどのような考え方を採用しているのかを紹介する。開発効率を高めるには、手間を減らすことと評価結果から得られる知見を増やすことの両面が求められる。

(1) 目的

GSM の開発は科学的方法に基づいているが、科学的方法には検証プロセスが不可欠である。検証結果は実験に適用された開発成果の正当性を示す資料であり、問題点の把握を通じて次の開発方針を決める材料にもなる。開発者は検証プロセスを通じて予測結果へのインパクトを適正に把握し、更なる改良のための調査も行う。同時に、検証結果は数値予報の利用者へ GSM の変更内容や改善の効果を説明するための材料にもなる。GSM は現業用途の実用モデルであり、開発者は利用者目線での評価検証を最終的には行い、改善内容について説明し理解を得る責任がある。

一方で、評価検証に対するニーズは開発する側と利用する側の両者で必ずしも同じにはならない。この意味で、検証には大きく 2 つのカテゴリが存在すると言えるが、DPSIVS では開発者の側に重きを置いている。その理由の一つとして、性能評価試験を実施する段階はまだ開発途中であり、開発する側のニーズがより重い点がある。無論、検証ツール単体としてみるとより多くの利用目的に適うことが良いのは確かである。ただし、DPSIVS は開発者自らが開発・保守を行っているので、ツールの目的を広げると管理コストが増える点には注意が必要である。現業運用版 GSM を変更する回数や、業務化試験（第 1.2.4 項）を実施する回数は性能評価試験より遥かに少なく、求められる図表もその時々の事情によって異なるため統一的なものは難しい。そのため DPSIVS では開発のためのツールであることに集中し、業務化試験については動作するもののサポートの対象外としている。DPSIVS の検証内容にはモデル開発者の今後の開発へのフィードバックが重視されるべきで、その結果により開発者の理解が深まり、より多くの気付きが引き出されるものが望まれる。

(2) 管理方針

DPSIVS 全体の構造としては、独立した検証ツール群の集合体であることを基本としている。これは、それぞれ独立に存在していた検証ツールをまとめる所から始まったという開発経緯も理由であるが、DPSIVS 自体の改良と維持管理コストを複数の担当者に分散させることが狙いである。また、管理コストを維持可能な範囲に収めるためには、需要が低下したツールをフェードアウトさせることが必要であるが、その基準としては維持管理の担当者として手を挙げる開発者がいるかいないかを用いている。GSM の開発は検証と一体不可分であるので、有用なツールは開発者の手により維持されている。

各ツールは完全に独立に実行できるよう、実行時の依存性は排除しており、ファイルフォーマット変換や格子系変換などの一部共有モジュールを除いて、一つの変更がツールの枠を超えて影響することはない。図 2.5.7 に DPSIVS により作成される結果閲覧用ウェブページのトップ画面を示す。ここにリストされている項目が、それぞれの検証ツールに対応している。DPSIVS ではそれぞれの検証ツールをパッケージと呼んでおり、それらパッケージの集合体に閲覧機能やパッケージ全体を実行するためのツールを加えたものが DPSIVS 全体を構成している。

ソースコードと定数ファイルの管理には数値予報ルーチン用のツールを利用している（第 3.2.3 項）。ルーチンのディレクトリ規則を踏まえると、パッケージごとに親ディレクトリが分かれ、ソースコードや定数の利用状況が分かりやすい。NuSDaS から NetCDF への変換ツールや描画における標準色設定など、共有されるものは Comm という名前のディレクトリ以下に配置している。一方で、JCL（第 3.2.3 項）は利用せず、シェルスクリプトをそのまま利用している。これは、検証処理ではシェルスクリプト内での分岐や繰り返しといった制御が多く使われており、JCL のステップ記述の利用があまり馴染まないためである。仮に利用したとしても、一つのジョブが一つのシェルスクリプトを実行するステップのみを持つ場合が多くなり非効率である。パッケージ内には実行時の依存関係をもつ複数のジョブが含まれるが、その記述にはパッケージの開発者がテストを行いやすいように考慮した結果、スーパーコンピュータシステムにインストールされている LoadLeveler（第 2.10 節）のジョブ記述ファイルを用いている。DPSIVS からジョブを実行する時には、依存関係を解釈し独自制御機構で並列数を制御しつつ順番に実行している。DPSIVS の開発と保守には、開発管理サーバ上のリポジトリおよびチケットを利用している。リポジトリの構成には特殊なものは用いておらず、標準的な trunk，branches，tags 構成である。各

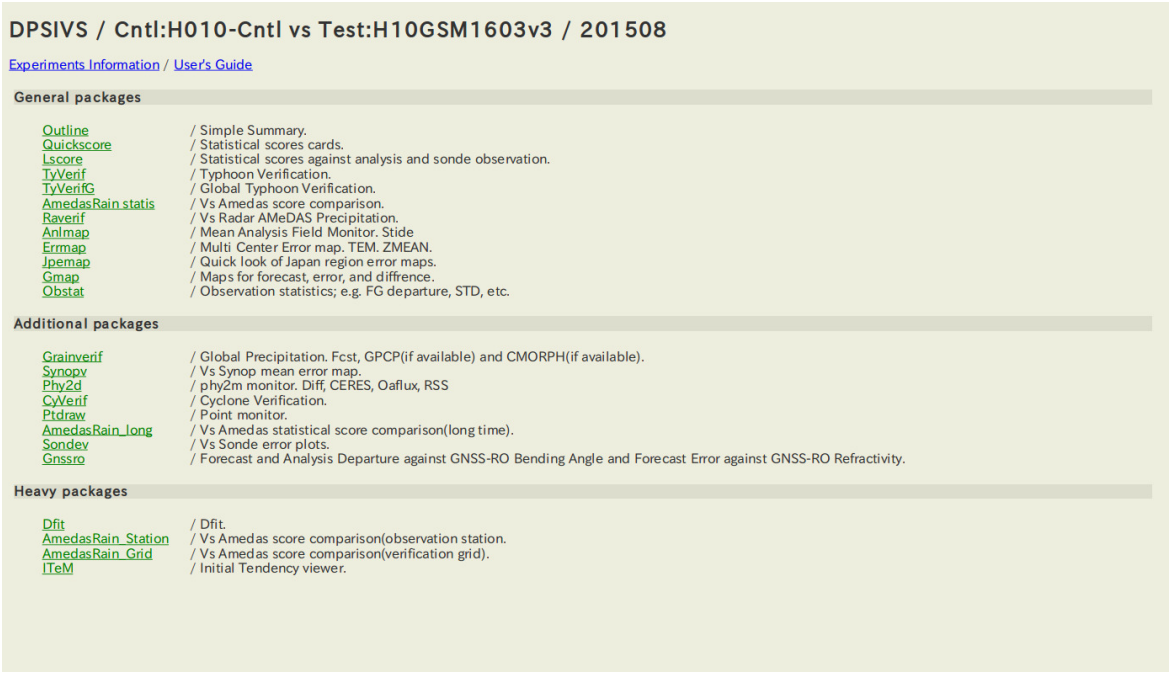

図 2.5.7　DPSIVS の実行により作成されるウェブページのトップ画面。検証パッケージごとに分類されている。

開発者は変更内容をチケットに記載した後、ブランチで作業を行い、管理者がトランクへマージする。ユーザードキュメントや開発者ドキュメントには Wiki を用いており、全球決定論的検証サブプロジェクト上に配置されている。パッケージの内容は各担当者が基本的には自由に決めるものであるが、移植性なども考慮して、作業ディレクトリや環境変数 PATH の設定、利用する外部ツールには制限を設けている。また、ツール内の絶対パスの利用は原則禁止であり、リファレンスデータのパスなども環境変数を通じて利用する。この方式により、例えばパッケージごとに利用する Ruby のバージョンが異なり、移植性に問題が生じる事態などのリスクを減らしている。それらの設定が記載されたテキストファイルが実行時の最初に作成されるため、各パッケージのシェルスクリプトには冒頭でそのテキストファイルを取り込むことが義務づけられている。

DPSIVS の実行はスーパーコンピュータの計算ノード以外のサーバで行うことにしている。他の検証ツールは必ずしもそうではなく、SPV のスコア計算や業務化試験の検証ツールでは計算ノードを利用するが、計算ノードを並列の大型計算が優先的に利用できるように DPSIVS では利用を避けている。また、性能評価試験の解析・予測の全てが終了してから一括で検証を実行することを基本にしている。これは、GSM の 11 日予測の検証において解析値や品質管理済み観測データを利用するためには、それらのデータを作成する解析ジョブが 11 日先まで終了している必要があり、予報ジョブの実行に合わせて検証を行おうとすると、解析ジョブと予報ジョブの実行タイミングに依存性を持ち込んでしまい、NAPEX（第 3.2 節）の利用上好ましくないためである。

(3)　総合的・網羅的な検証の必要性

開発者へのフィードバックの点では、まずは変更した項目に対するインパクトを幅広い観点で確認できることが重要である。自身が変更した内容に対して期待するインパクトが見られるかを確認することは当然だが、想定していない副作用がどの程度現れているのかも十分に把握しておく必要がある。

モデル内の各プロセス間の相互作用は複雑であり、基礎的な調査を十分行った上で科学的に正しい変更を加えたとしても、必ずしも予測結果の精度が向上するわけではない。大気モデルに限らないが、近似を含む方程式系の一部分を改良・精緻化しても、解の精度が向上するとは限らない。また、理想実験で良好な結果を得られていたとしても、理想実験が現実にあらわれる全ての状況を網羅することは不可能であるし、そもそも開発者が何か誤りをおかしている可能性もある。このため、試験結果を検証して副作用を確認することは開発における必須事項である。検証ツールの実行結果を一通り調べることにより、予測精度上の顕著な問題点はなるべく洗い出されていることが望ましい。

また、台風進路予測誤差など、現業運用に向けた重要指標に関する項目については最低限確認しておかないと、最終的な業務化試験で見過ごせない問題点が発覚するケースが増加し、開発における大きな手戻りが問題となり得る。これらの要請に応えるために、DPSIVS では総合的・網羅的な検証を行う幾つかの中核パッケージを用意している。中核パッケージ群は、SPV の検証内容や台風検証、降水検証などこれまで伝統的に行わ

れていた検証内容を含むだけでなく、多くの新規検証項目を取り込んでいる。一方で、網羅的であることと検証の処理時間はトレードオフの関係にあり[9]、バランスをどう取るかが問われる。実行にあまりにも時間がかかってしまうと結局利用されなくなってしまい、網羅的な検証が行われなくなる。この問題に答えはないが、自分たちで利用しながら妥協点を探っているところである。DPSIVS の実行は開発に利用している業務用サーバで行い、全体の実行にはおよそ半日程度の時間がかかる。ただし、実行が終了したパッケージから順にトップページへのリンクが張られ、数分で終わるものから順番に見ていけばあまり待ち時間を意識することなく利用できる。

(4) 効率的な利用

幅広い観点で様々な検証を行うには、大量のデータ・スクリプトのハンドリング技術が要求される。そのため、検証を少ない作業量で効率的に行うためには、検証ジョブを統合する環境整備が大切になってくる。特に、2 つの実験間を比較する場合には、組み合わせ実験の回数に応じて検証ツールの実行回数も増加しやすいため、せっかく検証ツール群が整備されていても、利用するために設定する項目が多く、時間と手間がかかってしまえば、大量の実験を評価することは難しくなる。

また、検証結果を評価していく時に、結果が網羅的でありつつも要点を絞って表示されていることや、ビューアなどでの閲覧のしやすさも開発効率を高めるには必要な要素である。開発者が確認に利用できる時間は有限であり、せっかく多くの検証図が作成されていても、見て分析・理解して知見を得なければ意味はないため、閲覧などの時間と手間は節約されるべきである。

DPSIVS では実行のためのエントリースクリプトを用意している。スクリプトは Ruby で記述されており、各開発者は標準のパスからスクリプトをコピー、必要な箇所を書き換えて実行する。最低限書き換えが必要な箇所は図 2.5.8 のような、TEST と CNTL 実験それぞれの実験番号・枝番[10] と図に表示する名前、検証対象期間である。

実験データの保存場所や、予測実験と初期値に用いた解析実験の関係などはエントリースクリプト側で NAPEX のデータベースから必要な情報を取得しており、利用者はそれを意識せずに必要最小限の設定だけで利用できる。

実際に実行する検証項目についても選択行をコメントアウトすることで取捨選択できるが、通常は全てを

```
# TEST 実験番号・枝番・略称
EXP_NO=9845 ; EXP_SUBNO=2
ABBR_TEST="GSM17XXv3"

# CNTL 実験番号・枝番・略称
EXP_NO_CNTL=9554 ; EXP_SUBNO_CNTL=2
ABBR_CNTL="H011a-Cntl"

# 検証対象期間（YYYYMMDD）
VERIF_SDATE=20150801
VERIF_EDATE=20150831
```

図 2.5.8 DPSIVS を実行するためのエントリースクリプトの例。書き換えが必要な部分のみを抜き出している。

実行して問題はなく、むしろ意図しない変更の副作用を把握するために網羅的な検証を行うことが推奨されている。DPSIVS には簡素ではあるが独自のジョブスケジューラが実装されており、順次 LoadLeveler を始めとする複数のジョブ管理ツールへのバッチジョブ投入を行う。ツールの管理コストを抑えるためには、第 3.2.3 項 (4) で紹介する ROSE を用いることが望ましいかもしれない。しかし、検証ジョブは様々なサーバで実行する可能性があり、ROSE でサポートされない環境も考慮して独自実装としている。今後のサポート状況によっては ROSE 利用へ切り替えることも念頭に置いている。

エントリースクリプトを実行すると、作業領域のパス、結果が保存されるパス、ウェブモニタへの URL が示される。ウェブモニタには、検証が終わった検証パッケージへのリンクが順次追加されていく。

ブラウザから利用するビューアには、共通で利用できるもの（図 2.5.9）を用意している。このモニタツールは HTML と JavaScript で実装されており、画像が入ったディレクトリの横に HTML ファイルを配置するだけで機能し、ファイル名の規則を解析して日付や要素名のリストを自動作成するビューア（リストの展開機能付）を提供する。検証パッケージの開発者はファイル名に気を遣うだけでよく、自分で HTML や JavaScript のファイルを作成または更新する必要はない。利用者もシンプルではあるが必要十分な機能を備えたビューアを様々な用途に対し一貫して利用できる。

(5) 知見の蓄積と拡張性

検証の実施者には、結果を評価する上で多くのことが期待される。例えば、次の開発へより良いフィードバックをもたらすには、モデルの変更点により生まれた予測結果の変化について、モデルの中で何が起きているのかを把握し、その差異を正しく理解することが求められる。リファレンスとする観測データや解析値に対しても、その誤差特性を理解し、検証結果につい

[9] 実行時間を短くするには検証ツールの高速化も重要である。ただし、高度な高速化を行うには開発コストが必要であり、こちらもトレードオフの関係にある。

[10] 記載するのは予測実験のもののみであり、対応する初期値を作成したサイクル解析実験の情報は NAPEX のデータベースから取得する。

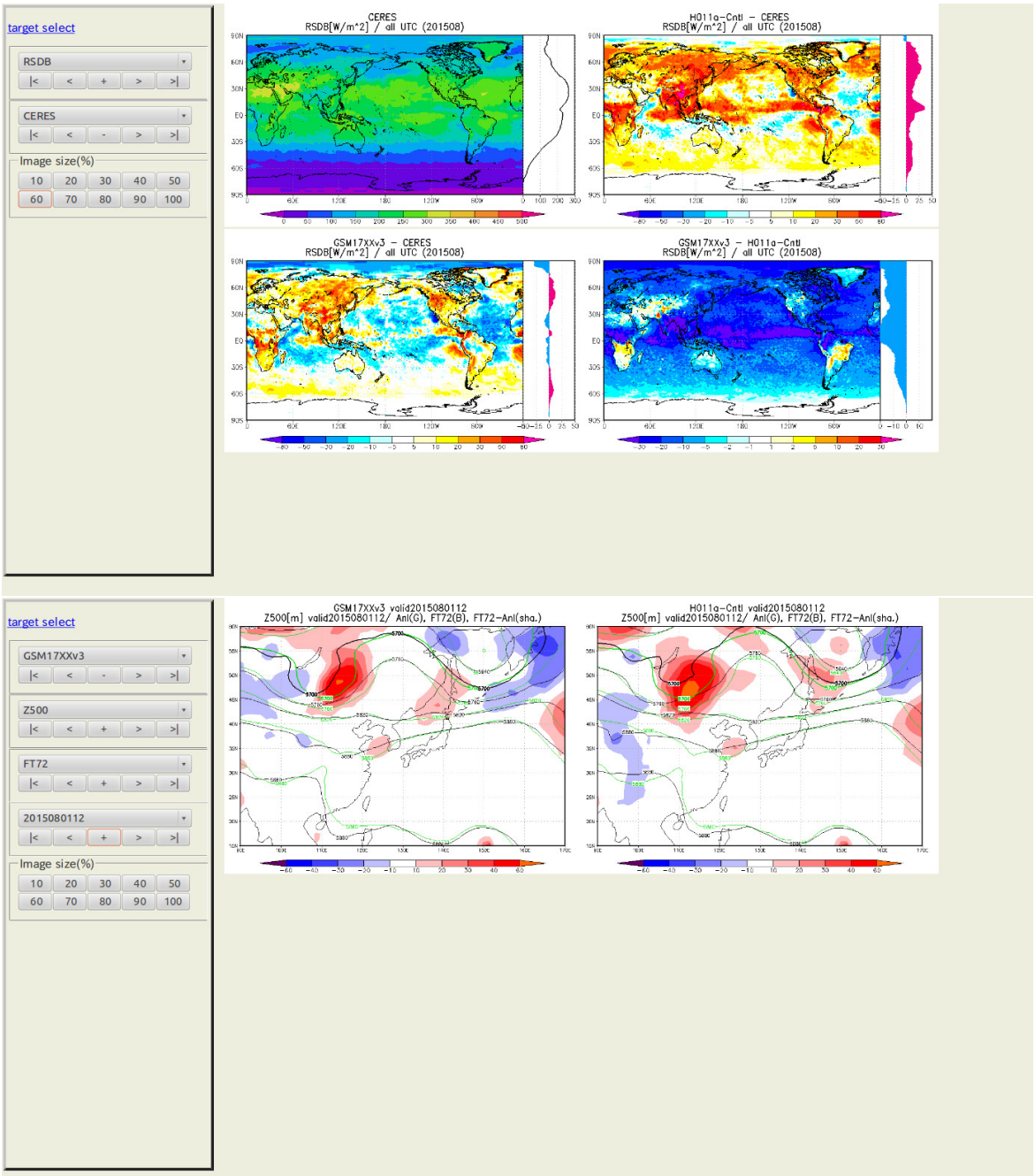

図 2.5.9　DPSIVS に装備されているビューアの表示例（上図と下図の 2 例）。自動リスト化とリストされた項目ごとの同時表示機能、画像表示の拡大縮小、キーボードショートカット機能を持つ。上図は phy2d パッケージの対 CERES 検証（後述）で、RSDB は地表面での下向き短波放射の月平均値を意味し、参照値、TEST と CNTL それぞれの誤差、TEST と CNTL の差分を表示している。下図は jpemap パッケージの 500 hPa 高度場の誤差表示で、黒線で予測値、緑線で解析値、塗りつぶしで誤差を表示している。（パッケージについては第 2.5.8 項を参照）

てどの程度の確からしさで議論が可能であるかに注意しなければならない。また、検証結果の統計性についても、統計学の知見を誤用せずに議論を進める必要がある。

検証ツールの立場から開発を助けるためにどのようなことが出来るであろうか。もちろん、開発者に代わって考えることなど出来ないし、結果を先回りして必要な図を作成することも不可能である。しかし、誰かが切り開いた道をフォローすることは可能であり、検証システム自体を評価検証の知見を蓄積する場と定義することはできる。評価検証における知見は、議論や開発における気付きにより蓄積されていくものである。実際の開発でも、モデルの予測に問題点が見つかり、その原因となるプロセスを議論した結果、新しく注目すべき量や指標などが提案されることはよくある。そのような知見に基づく新しい検証方法が検証システムに

追加されることによって、検証システム自体も成長していく。

そのため、新しい検証が将来追加されることを前提にして、柔軟性を持ちつつ管理コストが膨らみにくいソフトウェアとしての構造が求められる。DPSIVS がパッケージの集合体であることは、拡張性の面でも理に適っている。実際、最初のバージョンには現在の半分程度しかパッケージが含まれていなかったが、その後の開発において新たに必要となった検証ツールを新パッケージとして順次取り込んでいる。例えば、非地形性重力波スキームの新規導入に伴い中層大気を検証する目的で、GNSS 掩蔽観測を用いた検証が追加される、台風だけでなく全球的な熱帯低気圧へ着目することの重要性が認識された結果、熱帯低気圧検証のパッケージが追加されるなど、開発上の必要性から新しいパッケージが複数作成されている。DPSIVS は評価検証の知見を蓄積する場として今も成長している。

(6) 標準とカスタマイズ

評価検証を行うことは幅広い知識と深い思考を求められる作業である。そのため、利用者の思考を妨げる要因はなるべく排除すべきであり、混乱の原因が少ないことは大切である。図の描画に関する不整合や、同じ名前の指標を計算しているが微妙に手法が異なるといった手法の非一貫性、検証内容についての背景説明がない、論文などの解説資料へのアクセスが担保されないなど、利用者に無駄な注意力や知的負荷を要求する点はなるべく減らしたい。DPSIVS では、検証内容の説明には Redmine 上の Wiki を利用して情報共有を行っている。また、開発者一人の理解には限界があり、実験結果をより深く評価するには、検証結果をまとめ、ミーティングなどで議論することが大切になってくる。議論の立て方や資料のまとめ方などが重要ではあるが、同時に、検証結果の図表の視認性の高さ、理解しやすさは効率的な議論に欠かせない。多種大量の図が出てくる中で、図の説明を逐次していては時間が足りないので、色の使い方や目盛の間隔などが統一的であると良い。例えば、TEST と CNTL を示す色や、差分の色が図ごとに整合していない資料を考えると効率が悪いのは想像がつくだろう。検証図にある程度広い要望を満たせる統一的な基準（標準）が存在することの恩恵は大きい。そのため、DPSIVS では表示する要素やカラーレンジについてはなるべく可変でないものを推奨し、色の使い方に対する標準的なガイドラインを簡単ではあるが定めており、GrADS（第 4.4 節）上で利用できる標準カラー設定スクリプトなども提供している。また、パッケージ間で実験名の呼び名が異なったりしないように、環境変数を通じて一貫した値を提供している。ただし、最終的にはパッケージごとに事情があるため、判断は担当者に任せている。

一方で、変更内容によってどの程度のカラーレンジでの議論が必要なのかは異なる。ある要素の差が全体的な精度への影響としては無視できる範囲であったとしても、変更内容から想定される狙ったインパクトである場合には重要な注目点になる。このため、必要に応じて利用者が図を簡単に調整できることが望ましい。その設定がパッケージの奥深くに存在すると利用者では改変できなくなるため、描画する要素やレンジを記載したテキストファイルは Const の Param ディレクトリ以下に配置して改変を行いやすくすることを推奨している。設定変更が必要となった利用者は、自分で DPSIVS 環境をリポジトリから取得して書き換えて利用することが可能である。設定ファイルの内容はパッケージごとに異なるが、例えば図 2.5.10 のように単純なものが多い。

```
#element level int(shad) int(zmean) mask
PSEA 1013   1  0.5   0
T    1000   1  0.5   1
・・・(中略)・・・
Z     500   2   10   0
・・・(中略)・・・
PSI   850   1  0.5   0
PSI   200   1  0.5   0
```

図 2.5.10 描画する要素やレンジを記載したテキストファイルの例。これは平面で平方根平均二乗誤差を比較するための設定ファイルで、描画要素と誤差のレンジタイプ、気圧面が地面の下に潜る場合の特別処理の有無を毎行記載するだけのものである。

(7) 統計検証

DPSIVS が統計検証手法について採用している方針を解説する。モデルの精度向上を確認するためには、数値実験を実施しその結果を検討するのであるが、数値実験には多くの計算機資源が必要であり、精度向上を確信するのに十分な事例数の実験を通常は行えない。そのため、ある程度小さい事例数（通常 30 程度）の試験結果を検討することになる。つまり、地球のレプリカが取り得る状態分布に基づいて実験を行った場合（全数調査、母集団）の結果を、限られた事例数での結果（標本）から推定する必要がある。小さい標本について誤差の平均値などを比較する場合、推計統計学に基づかないと容易に結論を間違ってしまい、議論は往々にして多くの主観を含む非科学的なものになる。統計検証を十分注意深く行うことは実験の結果の分析において必須であるため、検証ツールでも十分に考慮されている必要がある。

統計検証には様々な手法が確立されているが、予測誤差については母集団の確率分布が何に従っているのかは明確ではないと思われる。また、利用者にとっては推定手法が明確であり、その特性を理解した上で誤解のないように使えることは大切である。そこで DPSIVS

では、成立していない可能性のある前提条件を利用者が意識せずに用いてしまうことを避け、また背景が整理されないまま手法が乱立することを防ぐため、必要な仮定が少ないシンプルな手法を全体で共通して利用することにしている。具体的にはブートストラップ法 (Efron and Stein 1981) を採用し、それを行う共有モジュール、サブルーチンを提供している。ただし、データ同化の領域では正規分布がある程度成り立っていると考えられるため、関連する検証には特に制限を加えていない。また、エラーバーの意味を明瞭にするために、95%信頼区間を採用することを義務付けている [11]。もし、複数のエラーバーを表示したい場合は、68%, 95%, 99%を利用することを推奨し統一化を図っている。

しかしそもそも、統計検証の解釈に関しては本質的に難しい問題がいくつか存在している。例えば、我々が本当に知りたいのは未来の現業運用において精度が向上するかであって、過去の期間から抽出した標本は条件が偏っていないといえるのか不明なことが挙げられる。特に、性能評価試験は特定の年の夏と冬を対象に実験を行なっているが、例えば熱帯海上のエルニーニョ・ラニーニャの状況によって、積雲対流過程変更のインパクトは異なって見えたりしないのか、といったことである。性能評価試験の構成では、無作為抽出の前提は疑わしく、偏った標本によりどのような評価の違いが生じ得るかに関する知見は現状では十分にあるとは言えない。

過去の期間を対象とした統計的な推定が示す有意性を、未来の運用での改善の可能性と安易に結びつけることは危険である。未来における再現性について我々は物理法則の普遍性を信じるのが原則であり、観測的事実に基づいて改良内容を検討し、変更が GSM 内にどのような変化をもたらすのか、モデル内のプロセスを十分に調査した結果がまず先にあるべきである。統計検証は、その結果を用いて変更内容の正当性を補強するための材料にするのが基本であり、統計検証自体が正当性の根元になるわけではない点には注意が必要であることは、強調しすぎることはないであろう。モデル改良の成果はどのような実験によって確かめられるべきなのかについて、今後とも調査・議論を通じて理解を深めていきたい。

2.5.8 DPSIVS の各パッケージ

DPSIVS に含まれる各パッケージの概要を簡単に紹介する。

(1) Quickscore

標準的なスコアを手早く確認することが目的である (図 2.5.11 がサンプル)。短い時間で結果を得るために、検証する対象要素を 6 種類 (海面更正気圧 [12]、500 hPa 高度、850 hPa 気温・風速、250 hPa 風速、700 hPa 相対湿度) に絞るとともに、アルゴリズムも冗長な計算・ファイル I/O を行わないよう工夫している。検証領域も北半球 (20°N 以北)、南半球 (20°S 以南)、熱帯 (20°S から 20°N) と日本周辺域 (20°N から 50°N、110°E から 150°E)、北西太平洋域 (0° から 60°N、100°E から 180°E) の 5 種類に絞っている。検証内容は、平方根平均二乗誤差 (RMSE) 、アノマリー相関係数 (ACC) 、平均誤差 (ME) の要素について、CNTL と TEST および両者の差分と、それぞれの区間推定である。差分について信頼度を変えて有意であるかどうかを確認するためのカテゴリ表示図も作成している。参照値は解析値とラジオゾンデ観測値である。

(2) Anlmap

平均解析場を比較するものである。解析場の変化が変更内容に対する応答として自然なものかを把握する。また、対解析値の検証は解析値自体の変化に影響を受けるため、それを丁寧に把握しておく必要がある。CNTL と TEST の比較においては予測精度の差が小さい場合もあり、その場合は特に解析値自体の変化に注意が必要である。また、各現業数値予報センターの解析値を用いた検証結果比較から、極域や熱帯域においては誤差の大きさと解析値のばらつきは同程度になることがあり、また中緯度においても 3 日目程度先までは誤差に対して解析値のばらつきが無視できないことが知られている (Langland et al. 2008; Jung and Matsueda 2016; Kanehama 2014)。このパッケージでは、TIGGE (The Interactive Grand Global Ensemble, Swinbank et al. 2016) データを利用して、他の現業数値予報センターの平均解析場に対して CNTL と TEST それぞれの差分も表示している。それぞれの解析値が分布する範囲から GSM の解析値が大きく外れるようだと特に注意が必要である。また、このパッケージには解析値にみられる太陽周期半日大気潮汐 (S_2) を比較する検証も含まれている。これは GSM の解析値には大気潮汐の位相ずれがあることが知られているためである (堀田・檜垣 2010)。

(3) Obstat / Dfit

データ同化における、第一推定値及び解析値と観測値との整合性評価を行うツールである (Lentze 2016; 計盛 2015)。衛星観測を含む全球解析で利用された観測データについて、観測値と解析値の差の標準偏差、観測値と第一推定値の差の標準偏差、品質管理を通過したデータ数の変化を観測種別、衛星搭載センサーのチャンネル別、領域別に網羅的に検証する。データ同化で利用される観測情報は膨大であり、かつ一定の品質管理も行われているためリファレンスとして非常に有用である。GSM を変更した時に、第一推定値と観測値との整合性が悪化していないかは必須の確認項目となる。

[11] 閾値に 95%を用いるのは慣習に従ったためである。
[12] ただし対ラジオゾンデ検証では代わりに 700 hPa 気温。

(4) TyVerif

台風検証のパッケージで、参照値には気象庁ベストトラックを用いている。進路予測誤差、中心気圧予測誤差についての統計的な検証と、個別の追跡結果比較図、中心気圧散布図を作成する。また、転向ステージごとに、位置誤差を進行方向について平行・直交する成分に分解する検証 (梅津・森安 2013) も行う。台風の誤差については、台風番号ごとに共通した特性を示すことが多いため、個々の台風ごとに統計検証を行ったものや追跡比較図をまとめたものも作成している。

(5) TyVerifG

台風の検証をサポートすることを目的とする熱帯低気圧検証のパッケージである。検証する項目は基本的に台風検証 (TyVerif) と同じであるが、参照値に B-deck[13] を利用して、北西太平洋領域を含む全球的な検証を可能にしている。台風を検証した結果は台風事例による依存性が高く、性能評価試験の範囲では統計的に十分な評価が難しいため、全球に適用範囲を広げて熱帯低気圧全体としてサンプル数を増やしている。また、台風ボーガスにより解析値が修正されない海域の方が、GSM 変更による熱帯低気圧への影響を明瞭に見ることが可能な場合がある。

(6) CyVerif

熱帯・温帯低気圧を含む全ての低気圧を対象とする全球低気圧検証パッケージである。低気圧予測全体の精度変化を把握する。独自の擾乱検出アルゴリズムで低気圧を追跡し (太田 2016)、解析値における追跡結果を参照値に予測値を検証する。位置誤差、中心気圧誤差、低気圧の存在そのものを予測できているかの捕捉率などの検証を含む。

(7) Lscore

解析値とラジオゾンデを参照値とする統計的スコアを表示するパッケージであり、線 (Line) で表示できる統計スコア全般を受け持つのが名前の由来である。実行時間は問わず、検証対象とする要素、面、領域を潤沢に取っているのが特徴である。RMSE, ACC, ME の鉛直プロファイル表示や、予報・解析活動度、絶対誤差、Taylor 図 (梅津ほか 2013)、ラジオゾンデによる検証と地点を揃えた対解析値検証などが含まれている。データ格納にはリレーショナルデータベースを利用しており、DPSIVS で作成される図表だけでなく、各自のアイデアに基づいて柔軟にデータを加工し利用できる。

(8) Errmap

解析値を参照値とする誤差統計について、面的な検証全体を受け持つパッケージである。RMSE と ME を気圧面、帯状平均での高度断面で比較できる。参照値とする解析値には、GSM のものだけでなく他センターの解析値も利用している。Lscore の結果と見比べて、予測誤差の変化を立体的に様々な要素について総合的に把握することが評価の基本となる。また、TEM (Transformed Eulerian Mean) に基づく残差循環の質量流線関数や Eliassen-Palm フラックスの子午面誤差検証も含まれている。

(9) Gmap

代表的な面・要素について、月平均場及び幾つかの初期日の予測結果を比較する。予測値そのものを描画するだけでなく、TEST と CNTL の差分や解析値との差分も描画している。予測結果がどのように変わっているのかについて、大まかな特徴を把握するのに利用する。Errmap と異なり単純な演算しか行わない代わりに、描画対象とする面・要素は多くなっている。

(10) Jpemap

スナップショットの予測誤差比較モニタである。日本周辺域を対象領域として、海面更正気圧、500 hPa 高度、925 hPa 気温、850 hPa 気温、500 hPa 気温、250 hPa 風速の対解析誤差を検証対象期間の全対象時刻、FT=24, 48, 72 について表示する。GSM の主要なターゲットの一つである日本域について、天気図の基本要素に関する予測誤差を確認するためのパッケージである。毎日の誤差マップと統計検証の結果を突き合わせ、どのような現象で予測誤差が変化しているのかを把握する。中緯度帯で誤差が大きく成長する条件は限られているため、モデル更新前後でも誤差のパターンが大きく異なる例は稀であるが、ジェットの速度、トラフの深まりや位相変化、擾乱の進行、高気圧の張り出しなどに着目しながら比較していく。統計スコアを解釈するときに、類似の気象現象で共通した誤差の変化が見られるのか、そうではなく事例ごとに異なるのかは重要な情報である。

(11) Synopv

参照値として SYNOP の 2 m 気温、2 m 比湿、海面更正気圧および前 24 時間積算降水量を用いる地上物理量検証を行う。SYNOP 観測の空間代表性に疑問があり、また計算負荷を抑えるために、全球の全地点に対して検証を行うのではなく 1.25° × 1.25° 格子ごとのメッシュ単位で検証を実施している。気温、比湿、海面更正気圧については 6 時間ごとに検証を行い、日変化を捉えることができるようにしている。実際、GSM の 2 m 気温には夜間と日中で逆符号のバイアスがある地点が多く、日平均で比較すると相殺されて誤差傾向が見えにくくなる。降水量については、後述する Raverif や AmedasRain と同様の検証を行っており、Grainverif の平均降水量評価と相補的に利用する。

[13] National Hurricane Center と Joint Typhoon Warning Center による熱帯低気圧解析。`http://ftp.nhc.noaa.gov/atcf/README`

(12) AmedasRain

アメダスを参照値とする降水検証を行うもので、バイアススコア (BI) 、スレットスコア (TS) 、エクイタブルスレットスコア (ETS) について統計検証と信頼区間推定を行うパッケージの一つである。検証対象とする降水量閾値、積算時間には 20 種類程度の組み合わせが設定されており、どのような降水でスコアが変わっているかを把握できる。また、アメダスの地点別に検証し、結果をマップで表示したものも含まれる。

(13) Raverif

解析雨量を参照値とする降水検証パッケージである。BI, TS, ETS だけでなく、誤検出率や適中率、降水量の RMSE, ME も計算する。また、面的な統計検証結果のマップ表示、日々の降水分布予測の比較図、検証対象期間中の毎日の各種スコアの時系列図、降水頻度分布図などを含む。解析雨量は海上域なども含み、AmedasRain とはお互いに相補的な関係にあるが、検証結果が整合的であることが多い。

(14) Sondev

ラジオゾンデを参照値とする、全球的な地点ごとの統計検証と散布図作成パッケージである。地点ごとに、気温、比湿、高度、東西・南北風の RMSE と ME が気圧面でマップ表示され、Errmap と同様に Lscore の結果と見比べながら利用する。散布図の要素は 850 hPa の相当温位、700 hPa の相対湿度など、解析値とラジオゾンデ観測値の差異が大きくなりやすい下層の量を対象にしている。

(15) Grainverif

期間平均した全球降水量の面的比較を行う。参照値は GPCP[14] (Global Precipitation Climatology Project) によるデータセットと CMORPH[15] (CPC MORPHing technique) プロダクトを用いており、CNTL と TEST で降水分布がどのように変化し、熱帯の加熱強制にどのような問題が考えられるかを評価する。予測時間が進むにつれて予測値の降水分布は参照値から離れていくが、最終的な落ち着き先だけではなく、離れていく速度がどう変化するかも確認する。

(16) Gnssro

GNSS 掩蔽観測を参照値とする統計検証であり、屈折角と屈折率に対して検証を行う。GNSS 掩蔽観測は高高度でも利用でき、観測のバイアスが小さい、鉛直分解能が高い、雲の存在に影響を受けにくいといった特性を持つ非常に有用な観測である。このパッケージでは対流圏上部から成層圏下部付近を対象に、RMSE, ME や利用データ数の変化などを検証している。検証内容は水平面、鉛直断面、時間高度断面など多岐にわたる。

(17) Phy2d

各種 2 次元モニタの統合検証パッケージである。CERES[16] (Clouds and the Earth's Radiant Energy System) データを参照値とする期間平均した全球放射バイアスの検証、OAflux[17] (Objectively analyzed air-sea fluxes) データを参照値とする期間平均した全球海表面フラックスバイアスの検証、RSS[18] (Remote Sensing Systems) によるデータを参照値とする期間平均した可降水量や液水量の検証と各種モデル出力の差分比較を含んでいる。GSM の放射バイアスの確認には CERES を参照値とする面的な比較がよく利用される。また、CNTL と TEST を比較して、気温や高度、風など基本的な要素に顕著な違いが見られなくても、モニタに出力される各種フラックスやパラメタリゼーション内の診断量では明瞭な違いが現れていることがあるため、基本的には全要素の確認を行う。

(18) Ptdraw

地点モニタの比較ツールである。GSM ではラジオゾンデ観測地点や特別観測地点を中心にその地点の予測値を地点モニタとして一時間ごとに出力している。このモニタは GSM 内のほとんど全ての予報変数、診断量が出力されており、モデルの挙動を詳細に把握することが可能である。このパッケージはそれら出力の値および差分を鉛直プロファイルと時系列で表示するものである。ただし、画像枚数が膨大になるため、地点及び初期時刻を間引いて描画している。統計検証の結果から気になった地点について、どのような変化が起きているのかを詳細に確認するのに利用する。

(19) ITeM

初期傾向診断法 (木南 2013) と FT=6 までの各過程の時間変化率を、様々な領域、断面上での平均値で比較するパッケージである。解析サイクルにおける第一推定値の修正量に着目し、その修正量の持つバイアスをモデルの中の各過程の時間変化率に分解して評価することにより、系統誤差の発生場所を診断する。また、このパッケージは鉛直方向にはモデル座標面を用いており、パラメタリゼーションの実装上の問題から発生する、特定の面で発生する時間変化率についての問題などの調査にも利用される。

---

[14] 全球降水気候値プロジェクト。http://precip.gsfc.nasa.gov/

[15] NOAA Climate Prediction Center による全球降水量解析。http://www.cpc.ncep.noaa.gov/products/janowiak/cmorph_description.html

[16] 全球放射収支計によるプロダクト。http://ceres.larc.nasa.gov/

[17] 全球の大気・海洋フラックスについての研究開発プロジェクトによる解析プロダクト。http://oaflux.whoi.edu/

[18] リモートセンシングシステム社。http://www.remss.com/

(20) Outline

基本的なパッケージの結果のうち代表的なものを一覧できるようにまとめ、クイックルックを提供する（図 2.5.11）。まとめる対象になるパッケージは、Quickscore, Anlmap, Obstat / Dfit, TyVerif, Lscore, Errmap, AmedasRain, Raverif, Grainverif である。

2.5.9 DPSIVS のまとめと今後の展望

GSM 開発における標準検証環境である DPSIVS について、特に開発管理の考え方を中心に簡単に紹介した。本検証システムは、近年の GSM 開発において中心的課題となっていた「compensating errors の解消」のため、改良項目の組み合せ実験の評価検証を効率的に実施し、開発者へのフィードバックを増やすことを大きな目的としている。また、検証システム自体の開発・保守を GSM 開発者自身が行い、問題点への理解や評価の知見が深まるにつれて、DPSIVS も成長していくところに特徴がある。本節では、それぞれのパッケージについて簡単に紹介したが、その詳細な内容と具体的な利用例の説明については別の機会に譲りたい。

最後に、今後の GSM 開発における検証環境の課題として、新しい検証手法の利用および、性能評価試験以外の標準実験の必要性に触れておく。

基本的な話として、参照値として利用できる観測データや観測プロダクトをさらに収集して、検証可能な要素を地道に増やす努力、及び新しく提案された検証手法を試しながら取り込んでいくことが重要なのは言うまでもない。同時に、現業数値予報センターである強みを最大限に活かし、データ同化で利用されている観測結果の評価検証でのさらなる有効活用も模索すべきであろう。

また、今後のパラメタリゼーション改良を考えた時に、雲についての 3 次元的な構造や日変化の検証は重要である。他の現業数値予報センターにおいても、衛星観測シミュレーションを利用した雲検証や、降水の日変化特性の検証などが日常的に利用されるようになってきている。今後の開発のためには、GSM においても早期にこれらの検証が実施できるよう環境整備を進める必要があるだろう。

性能評価試験以外にも、低解像度モデルを用いたより気候予測実験に近い構成での試験や、再解析を初期値として幅広い年を対象にした試験など計算コストを大幅に増大させることなく GSM の問題点をより鮮明に引き出せる試験構成の模索も必要である。特に、台風進路予測の検証結果は強い事例依存性を持つことが知られており、事例数を多く稼ぐために低解像度モデルを有効に利用できる可能性はある。ただし、それらの実験を有効に活用するためにも、GSM の分解能依存性についての調査と、Minor treatment（第 1.2.3 項）や数値計算上の問題から来る不必要な感度を持たないモデル開発が必要になってくるだろう。現在、これら低解像度モデルを利用した評価については、気候予測検証の経験をもつ気候情報課のアンサンブル予報システム (EPS) 開発担当者と連携しながら様々な取り組みを行っているところである。実際に 1 年積分実験や AMIP (Atmospheric Model Intercomparison Project) 型実験を行う環境の整備が進められており、それら新しい標準実験に対する検証システムにおいても、DPSIVS の思想とフレームワークは共有され開発が進んでいる。

EPS を対象とした検証システムとしては、EPSIVS (Ensemble Prediction System Integrated Verification System) が開発・利用されている。EPSIVS は 2009 年当時の、現業システムの予測結果検証と一体型であった EPS 検証システムから EPS 開発専用の検証環境として分岐し、その後多くの改良を取り込みながら発展し、全球 EPS (経田 2016) の開発において主要な検証環境として利用されている。EPS 開発が対象であるので、検証内容に確率的検証項目を多く含んでいる点は DPSIVS とは異なるが、利用者視点よりも開発者へのフィードバックを重視することや、開発での必要性や議論を取り込みやすいように拡張性を持った設計にしている点など背景としている思想は共通している。検証システムの管理コスト低減のためにも、お互いの良さは残しつつ、共有できるところは統一していきたい。

2.5.10 最後に

今回、代表して紹介させて頂く機会を得たが、GSM の開発管理手法や検証システムは、多くの開発者の協働により整備されているものである。また、利用しているサーバやソフトウェアの開発・管理については、気象庁内各課室にお世話になっている。ご尽力いただいた全ての方に感謝し、引き続き GSM の精度向上に努めていきたい。

【追記】ここで紹介した開発基盤ツールの開発者を付記しておく。DynaMo は齊藤 慧（数値予報課）、Super Verif は田内 利治（同）、DPSIVS の基盤部分は金浜 貴史（気候情報課）、関口 亮平（同）の寄与が大きい。

参考文献

Efron, B. and C. Stein, 1981: The Jackknife Estimate of Variance. *Annals of Statistics*, 586–596.

原旅人, 2013: 概論. 数値予報課報告・別冊第 59 号, 気象庁予報部, 1–5.

平井雅之, 堀田大介, 2009: 陸面過程. 数値予報課報告・別冊第 55 号, 気象庁予報部, 99–108.

堀田大介, 檜垣将和, 2010: 現業数値予報モデルにおける太陽周期半日潮汐の表現のセンター間比較. 大会講演予講集 98, 日本気象学会, 53.

堀田大介, 原旅人, 2012: 物理過程開発のボトムアップ・アプローチとトップダウン・アプローチ. 数値予報課報告・別冊第 58 号, 気象庁予報部, 120–122.

岩村公太, 2008: 高解像度全球モデルの改良. 平成 20

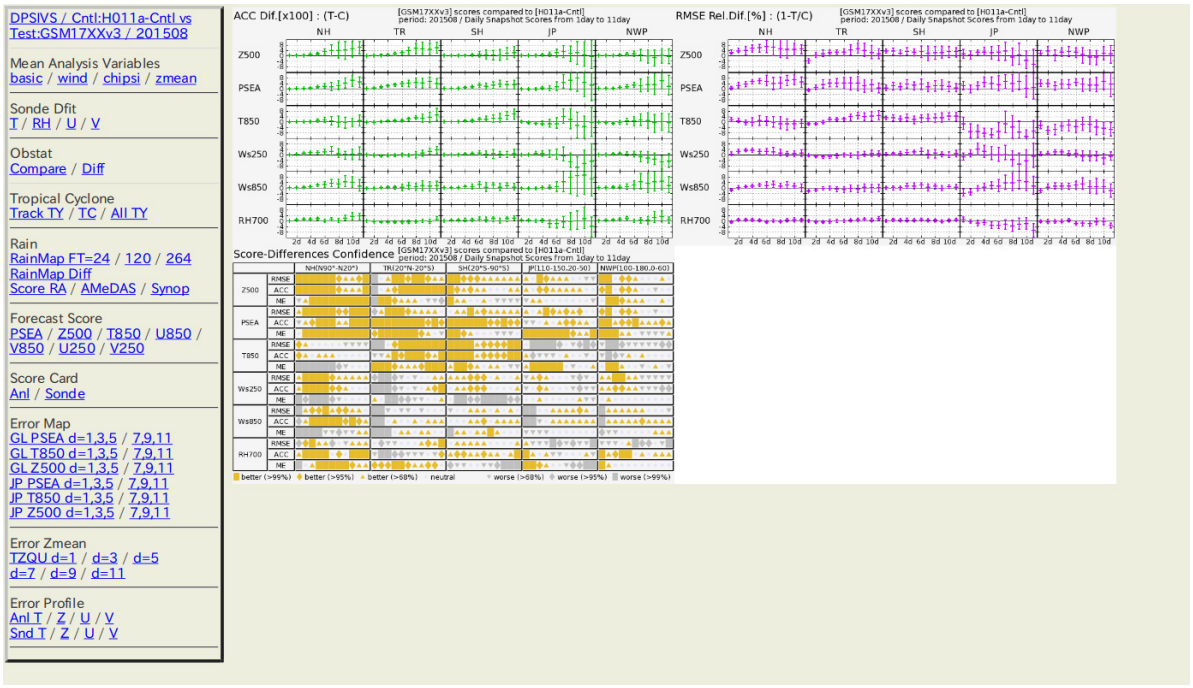

図 2.5.11　Outline の表示例。図は Quickscore パッケージの結果。

年度数値予報研修テキスト, 気象庁予報部, 1–6.

Jung, T. and M. Matsueda, 2016: Verification of global numerical weather forecasting systems in polar regions using TIGGE data. *Quart. J. Roy. Meteor. Soc.*, 574–582.

Kanehama, T., 2014: Verification against multiple analyses. *WGNE-29*, `https://www.wmo.int/pages/prog/arep/wwrp/new/documents/multicenter_verif_cmuroi.pdf`.

計盛正博, 2015: 衛星観測輝度温度データを使った同化サイクルにおける影響評価. 数値予報課報告・別冊第 61 号, 気象庁予報部, 82–85.

木南哲平, 2013: 初期傾向診断法. 数値予報課報告・別冊第 59 号, 気象庁予報部, 70–75.

気象庁予報部, 2007: 全球数値予報モデル (GSM) の積雲対流スキームの改良について. 配信資料に関する技術情報 (気象編) 第 275 号.

北川裕人, 2006: モデルの概要. 平成 18 年度数値予報研修テキスト, 気象庁予報部, 7–10.

北川裕人, 2007: 変更の概要. 平成 19 年度数値予報研修テキスト, 気象庁予報部, 1–4.

経田正幸, 2016: 全球アンサンブル予報システムの開発. 数値予報課報告・別冊第 62 号, 気象庁予報部, 52–57.

Langland, R. H., R. N. Maue, and C. H. Bishop, 2008: Uncertainty in atmospheric temperature analyses. *Tellus*, **60**, 598–603.

Lentze, Georg, 2016: Newsletter No.149 - Autumn 2016. 30–33.

宮本健吾, 2009: 適合ガウス格子版全球モデル. 数値予報課報告・別冊第 55 号, 気象庁予報部, 27–49.

室井ちあし, 藤田司, 大野木和敏, 2013: 具体的な取り組みの実例. 数値予報課報告・別冊第 59 号, 気象庁予報部, 202–203.

西尾利一, 2011: 計算機 (スーパーコンピュータシステム). 平成 23 年度数値予報研修テキスト, 気象庁予報部, 68–70.

太田洋一郎, 2016: 低気圧予測の精度. 数値予報課報告・別冊第 62 号, 気象庁予報部, 43–46.

下河邉明, 古河貴裕, 2012: 層積雲スキームの改良. 平成 24 年度数値予報研修テキスト, 気象庁予報部, 92–96.

Swinbank, R., M. Kyouda, P. Buchanan, L. Froude, T. Hamill, T. Hewson, J. Keller, M. Matsueda, J. Methven, F. Pappenberger, M. Scheuerer, H. Titley, L. Wilson, and M. Yamaguchi, 2016: The TIGGE Project and Its Achievements. *Bull. Amer. Meteor. Soc.*, 49–67.

梅津浩典, 森安聡嗣, 2013: WGNE 熱帯低気圧検証. 数値予報課報告・別冊第 59 号, 気象庁予報部, 98–111.

梅津浩典, 室井ちあし, 原旅人, 2013: 検証指標. 数値予報課報告・別冊第 59 号, 気象庁予報部, 6–15.

山下浩史, 2013: 現業化を判断する指標. 数値予報課報告・別冊第 59 号, 気象庁予報部, 28–32.

米原仁, 2014: 変更の概要. 平成 26 年度数値予報研修テキスト, 気象庁予報部, 1–3.

米原仁, 2016: 全球数値予報システムの物理過程改良の概要. 平成 28 年度数値予報研修テキスト, 気象庁予報部, 1–3.

2.6 活用例 (2)–メソモデル [1]

2.6.1 はじめに

2017 年 1 月現在、メソモデル（MSM, 水平格子間隔 5 km）の予測モデルとして気象庁非静力学モデル（JMA-NHM）が利用されている (Saito et al. 2006)。一方で、次世代の非静力学モデルとして数値予報課で開発してきた asuca(気象庁予報部 2014) の現業利用が、2015 年 1 月には局地モデル（LFM, 水平格子間隔 2 km）で始まり (原ほか 2015)、2017 年 2 月には MSM でも始まる予定である。

現在の MSM の開発は、力学コアとしての asuca と、asuca に物理過程を提供する物理過程ライブラリ (原 2012) を 2 本の柱として開発が行われており、本節ではそれらの開発管理とそのための関連システムの活用について述べる。また、開発の上で必須となる評価実験の管理についても紹介する。

なお、asuca や物理過程ライブラリの開発管理については、石田・藤田 (2014) でも触れられている部分があるが、本稿では開発プロセスに即して、より詳細に記述する。

2.6.2 物理過程ライブラリと asuca の開発における開発管理

物理過程ライブラリと asuca の開発においては、設計思想やプログラム構造を明確にした上で、新しい計算科学の知見の導入、計算安定性の確保、最近および将来の計算機の趨勢を踏まえた効率的なプログラム構造の導入を目指している (石田・藤田 2014)。その中で、加えようとする修正の必要性や科学的・技術的妥当性の検証を重視しており、そのプロセスにおいてプロジェクト管理システムへの開発過程の記録、バージョン管理システムのブランチの活用、そしてレビューが重要な役割を果たしている。

以下では、これらに焦点を当てながら、物理過程ライブラリと asuca の開発管理について述べる。

(1) 物理過程ライブラリの開発における開発管理

物理過程ライブラリの開発は 2010 年 8 月ごろから始まった。開発が始まるとともに、ソースコードに修正を加える際のプロセスとして、英国気象局における手法 (原・高谷 2013) を大いに参考にしながら、以下のプロセスを導入した。

チケットとブランチの作成、作業過程の記録

ソースコードに修正を加えようとする開発者（修正者）は、まず、プロジェクト管理システム（当初は Trac、後に Redmine に移行）にその修正について記録するチケットを作成する。合わせて、バージョン管理システム (Subversion) のリポジトリにその修正をコミットするためのブランチを作成する。修正者は、作成したチケットに自らの作業記録を残しながら、修正内容をブランチにコミットしていく。

これらの作業過程の記録は、後述のレビューや、後にその修正を振り返る必要が生じたときに、修正者の考えや実際の作業を追跡できる資料となっている。また、作業途中の内容も含めてブランチにコミットしながら作業を行うことで、その作業の途中経過を他の開発者が把握することが可能になり、その内容についての議論やテストなどが行われるようになった。

レビューに向けたドキュメントの整備、テストの実行

修正が終わったら、レビューへ向けて、修正の妥当性を示すドキュメントや作業記録を整備し、テストの実行と検証を行う。レビューは他の開発者（レビューア）に依頼して実施される修正内容のチェックで、

- 修正意図が、必要性や科学的・技術的根拠に基づく適切なものであること
- その意図に沿って実装が行われていること
- その実装がライブラリの設計思想やコーディングルールに合致していること
- 必要なテストが適切に行われていること

などの観点から行われる。

これは、レビューアだけではなく他の現在の開発者、未来の開発者への説明責任を果たすためのプロセスである。このようなプロセスがあることで、自らが行おうとするソースコードの修正が他の開発者に説明できるものであるかどうかを、各開発者が意識しながらモデル開発を行うようになった。

レビューの実施

レビューに必要な資料が準備できたら、レビューアにレビューの実施を依頼する。レビューアは先に示した観点などからブランチにコミットされた修正内容をチェックし、問題点が見つかったり、説明や記録が不十分で妥当性の判断がつかない場合には、修正者に差し戻して対処を依頼する。

このプロセスを通じて、修正者が気がつかなかった誤りが見つかったり、より望ましい方法がレビューアによって提案されたりするとともに、修正の必要性や科学的・技術的根拠の検討が修正者と別の開発者であるレビューアによっても行われ、より客観的に修正の妥当性を判断できるようになった。また、設計思想やコーディングルールへの合致をチェックすることで、無秩序な拡張によってソースコードが必要以上に複雑になることを防ぎ、継続的な開発の維持にも寄与している。さらには、新規に参入した開発者に対して設計思想やコーディングルールについての理解を促す機会にもなり、技術継承にも役立っている。

修正内容のブランチから開発本流へのマージ

レビューアから修正内容についての承認が得られたら、リポジトリの管理者がブランチにコミットされている修正内容を開発本流（トランク）にマージする。ト

[1] 原 旅人、荒波 恒平、松林 健吾、石田 純一

ランクへのコミット権限は管理者にのみ付与されており、これまでに述べた手続きを経ずにトランクに修正を加えることができないようになっている。

この仕組みによって、レビューアから承認が得られた修正のみがトランクに反映できるという手続きが明確化された。

次で述べるように、この開発管理プロセスは asuca にも導入され、asuca と物理過程ライブラリの開発管理プロセスは、ほぼ共通化されている。

(2) asuca の開発における開発管理

asuca の開発は 2007 年ごろからごく少人数で始まったが、モデルの大枠が定まってきた 2008 年には開発管理の支援のために Trac や Subversion が導入された。導入当初は、ソースコードに修正を加えようとするときには Trac にチケットを作成すること、このチケットと Subversion のリポジトリへのコミットを必ず対応させることがルールとして定められた一方、当初は Subversion のブランチ機能は活用されておらず、開発者が個別にトランクにコミットしていた。2010 年 1 月ごろにブランチの活用が始まり、修正はブランチに作成した上でトランクにマージするというスタイルが採用された。ただ、この時点では現在の開発プロセスの中で行っているレビューは行っていなかった。これは、開発が本格化して間もない段階で、ソースコードの新規追加や大幅な書き換えが頻繁に行われる中で、コストの方が得られる効果よりも大きいと考えられたこと、ごく少人数での開発であるため設計思想やコーディングルールについての認識が共有されており、レビューの大きな役割の一つであるそれらの観点からのチェックの必要性が低かったことなどが背景にある。しかし、良い開発のためにレビューが必須であることが asuca の開発者の中で共通認識となった今から振り返れば、設計思想やコーディングルールについての共通認識が得られやすい少人数の開発であっても、誤りの発見はもちろんのこと、レビューによる修正内容の相互理解の促進などのメリットは大きく、たとえ少人数でもレビューは行った方が良かったと思われる。

その後、物理過程の実装が始まるなど新規に参入する開発者が増えるにつれて、設計思想やコーディングルールを理解した上で修正が行われているかをチェックすることが従来よりも必要となったこと、また、asuca が物理過程として利用している物理過程ライブラリと開発管理プロセスを共通にしておくことは双方の開発者にメリットがあるとの判断から、既存の開発管理プロセスを拡張して、2010 年 12 月に物理過程ライブラリとほぼ同じ開発管理プロセスが asuca にも導入された。

次項では、現在メソモデルの開発における開発管理とその関連システムの活用について、asuca や物理過程ライブラリの開発に即して、より具体的に説明する。

2.6.3 asuca と物理過程ライブラリの開発（ソースコードの改変）における開発管理関連システムの利用

図 2.6.1 に、asuca や物理過程ライブラリの開発チーム内におけるソースコードの改変を伴う開発ワークフローの典型的な例を示す。

(1) 課題の設定

どのような開発も、課題の設定が起点となる。課題は

- 現在のモデルがもつ問題点を解決する可能性のある手法を考案した、または関連する文献を見つけた
- ソースコードやドキュメントを読んでいて誤りに気がついた
- モデルが異常終了した
- ある値をモニタしてみたら物理的に不自然な値や振動が計算されていることに気がついた
- モデルの評価、検証によるフィードバック
- 新しい機能を追加する必要が出てきた

など、様々なきっかけにより設定される。

(2) チケットの作成

課題が設定されたら、Redmine 上にチケットを作成する。チケットにはその内容を端的にあらわす「タイトル」を付け、課題の内容について「説明」を記述し、「担当者」を設定する。調査の進展具合に応じて、タイトルや説明と内容にずれが生じる場合もあるが、その場合はタイトルや説明を変更したり、関連チケットを作るなどして対応する。担当者は、業務分担や現在行っている仕事との優先順位、緊急度等を勘案して、開発チーム内のメンバーから割り当てられる（チケットを作成した本人とは限らない）。課題の優先度が低い場合には、亡失を防ぐために、当面担当者を割り当てずにチケットのみを作成しておく場合もある。

(3) ブランチの作成

トランクを改変する場合、機能拡張のために新たなソースコードを追加する場合や、ある程度規模の大き

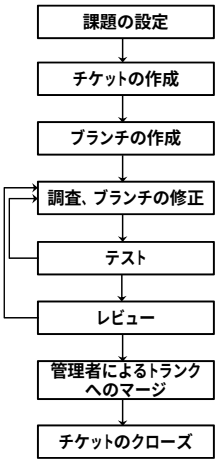

図 2.6.1 メソモデル開発でのソースコードの改変を伴うワークフローの例

な調査用のプログラムを作成する場合は、バージョン管理を行うために、開発ブランチを作成する。作成した開発ブランチの場所、および（可能であれば）その変更履歴をチケットに記録する。次に述べる調査の結果によっては、仕切り直しのために新しいブランチを作成することもある。

(4) 調査、ブランチの修正

次に実際の調査を行う。どのような目的で調査を行ったか、結果はどうだったか、結果を受けて次にどのような調査を行ったか、など、チケットを見た人が理解できるように記録する。現状、記録のやり方については特に制限を設けておらず、チケット上にテキストで記録したり、LaTeX 等でドキュメントを作成したり、スライドでまとめた資料を作成したり、画像ファイルを貼り付けたりするなど、調査の内容や進展状況に応じてやりやすい方法をとっている。調査に用いたプログラム等があれば、それも記録する。調査用の個別のプログラムについては、ブランチを調査用に作って適当なディレクトリを作成し、そこにコミットしていくことで、記録として分かりやすくする方法もしばしば用いられる。すでにあるソースコードの一部を変更する場合も、調査の過程では、作業内容を失わないために、ブランチについても随時更新していくことが多い。ただし、試行錯誤の結果、変更内容が複雑／膨大になった場合には、新しくブランチを作成し、必要な部分だけをブランチにコミットする形で整理しなおすこともある。

調査の内容は、試行錯誤の段階でうまくいかなかったものも含めて記録するようにしている。特に、期待される結果と違う結果が出てきた場合には、すぐに解決できない場合でも、今後の課題として認識しておくことが重要である。そうしておくことで、別の調査を行っている際に、原因を同じくする問題が顕在化してその解決法が考案され、それをきっかけに一連の課題が解決するケースもある。

(5) テスト

プログラムの改変を行い、それをトランクにマージする前には、十分なテストを行う。開発の目的（例えばあるスキームの持つ気温の不自然な時間変化率の改善など）に応じて、それが実現されているかどうかテストを行う。たとえば物理過程の変更では、1 次元モデルでの実験による評価、ある事例についての 3 次元モデルでの実験による評価、ある程度まとまった期間の実験による評価等を行い、力学過程の改良であれば、理想実験を通じた実験 (石田ほか 2014) 等を行って、結果を記録する。

また、asuca については、トランクへの変更がなかった場合も含めて [2]、テストハーネスと呼ばれる仕組みを毎日自動で実行している。これは

- 理想実験
  - 石田ほか (2014) で述べられている、2 次元定常山岳波、周期境界条件における重力波、暖気塊のテスト、重力流、St-MIP
  - 静止大気実験
  - 2 次元スカラー移流
- 3 次元モデル [3] による狭領域実データ実験
  - 局地解析
  - 接線形モデルチェック
  - 随伴モデルチェック
  - 局地予報
  - メソ予報

といった様々なケースについての実験 [4] が、最新のトランクを用いて行われる。入力データは毎日同じものを用いるため、入出力システムの変更等、予報結果を本質的に変える変更でない場合には、結果はビットレベルで一致する。自動実行されたテストハーネスの結果は毎日メールで報告され、結果が前日とビットレベルで合わない場合にはそのようなメッセージが付加され、開発者が覚知することができる。このため、トランクへのマージを行う改変については、テストの段階で開発者が自らテストハーネスを実行し、その結果についてもチケットに記録する。また、変更の前後で結果が一致しない場合には、その理由や差の妥当性（分布図で見た場合の評価等）も含めて、記録を行う。

(6) レビュー

レビューアは、第 2.6.2 項で述べた観点で変更の確認を行い、コメント等をチケットに記録する。その際、実行性能の観点から非効率なものになっていないか、コーディングスタイルが他と調和がとれているかなども含め、様々な観点から検討を行う。実際のレビュープロセスでは、多くの場合、そのまま通過することは少なく、レビューアから何らかのコメントがなされることが多くなっている。これにより、ドキュメントやソースコードの誤りの修正や、より分かりやすくするための変更などが行われ、これらの品質の向上に大いに寄与している。

なお、レビューのプロセスは Redmine 上の公開プロセスとして行われており、レビューアに指名されていない開発者も、チケットを見て変更内容をフォローし、疑問があれば積極的に議論に参加することが推奨されている。最終的に説明が不十分な場合、必要な実験が

[2] 毎日実行することで、ソースコードを変更していないにもかかわらず、100 回に 1 回程度の割合で結果が再現しない問題を検知し、OpenMP によるスレッド並列化に関するバグが特定された例もある。数値予報ルーチンとしては、再実行性の観点から、何度実行しても結果が一致することが重要である。

[3] 局地予報、メソ予報については、水平方向の格子数以外は現業のメソモデル、局地モデルと同じ仕様で実行される。

[4] これ以外にも、有用なテストが作成できれば、随時テストハーネスに組み込んでいる。

行われていない場合などはレビューアから開発者へ差し戻しが行われることもある。

レビューアに理解してもらえるように丁寧に説明することで、そのチケットが開発の記録として非常に価値の高いものとなり、現在だけでなく、将来の開発者への説明責任を果たすことになる。

(7) 管理者によるトランクへのマージ

レビューアによる承認が得られれば、管理者によるトランクへのマージが行われる。開発者は自由にトランクを改変することはできない。トランクへのマージを管理者に限定することで、開発の流れの一つとして、レビューアによる承認を受けることが必須となり、ソースコードの品質維持の担保となる。トランクへのマージの際、ビットレベルで結果が異なる場合には、その旨のメモが Wiki に追記され、履歴を辿りやすいように工夫している。

なお、複数の開発ブランチの差分が、競合（コンフリクト）を起こす場合もある。軽微なコンフリクトは、マージを行う管理者により修正されるが、複雑な場合には、新しいトランクから再度ブランチを作成してもらうよう、管理者から開発者へ依頼する場合もある。

(8) チケットのクローズ

前述の通り、テストハーネスは毎日自動実行されており、その結果はメールで報知される。まれに、自動実行の環境と開発者の環境の違い等により、意図せず結果が一致しない場合があるため、念の為、トランクへのマージが行われたら翌日のテストハーネスの結果を確認し、結果が意図通りなら、チケットをクローズして開発を終了する。なお、後に改変に関連する不具合が発生した場合、特に不具合の修正が小さい場合はチケットを再開して対応することもある。修正が大きい場合は、関連チケットとして、新しくチケットを作成することが多い。

なお、これらのフローに対応して、Redmine のステータスには「新規」「割り当て済み」、「進行中」、「レビュー」「コミット済み」、「終了」を定義しており、これらを活用して、各チケットの状態をわかりやすくしている。

以上、asuca や物理過程ライブラリにおける開発のワークフローの典型例を紹介した。開発管理ツールを活用することで、開発のワークフローが定型化され、レビューとそれに必要な説明を丁寧に心がけることで、ミスを減らすことにもつながるような仕組みとなっていることがお分かりいただけるだろう。

実際の asuca の開発においては、チケット上の議論だけでなく、2 週間に 1 度程度開発者が集まって、ミーティングを行い、主な開発の進捗について議論、共有を行っている。特に、改変の内容について複数の方向性が考えられ、開発者間で意見がわかれるような場合には、ミーティングの形式を取るほうが効率的に議論を行うことができる。こうしたミーティングにおける進捗の共有や改変内容の議論にもチケットをそのまま活用することにより、準備の省力化をはかることができている。

2.6.4 モデルの評価実験を行う際の開発管理と開発管理関連システムの利用

(1) 評価実験のプロセスとその管理

これまで述べてきた開発管理、および関連システムの活用は、モデルのソースコードを修正する際の作業の流れに関するものであった。一方、モデルに対して変更を加える際には、その変更によってもたらされるモデル予測の変化を確認するための評価実験が行われる。以下では、その評価実験のプロセスやその管理について解説する。

まず、asuca の開発においては、実験の管理や記録を一元化するために、ある程度まとまった期間や事例に対して評価実験を行う場合には、開発者各自がそれぞれ実行するのではなく、その実験の必要性、設定を開発者間で議論した上で、共用アカウントを用いて実験を行っている。共用アカウントによる実験に統一することで、実験の必要性を開発者個人の判断ではなく開発者間で議論する機会が必須となる。実験の必要性の議論を通じて、開発状況の進捗の共有や開発者間で現在進行中の開発内容についての理解を深めることに寄与するとともに、必要な実験だけを計画的に行うことで、計算機資源の有効活用にも役立っている [5]。

実験を行うことを決めたら、NAPEX を活用して実験環境を構築し、実験環境を構築した人とは別の人による実験環境の確認（レビュー）を行う。実験設定に誤りがあると、計算機資源が無駄になるのはもとより、その後の開発にも大きな影響を及ぼしかねないためである。

レビューが終わったら実験の実行を開始し、実験が終了したら検証プログラムを実行して統計検証を行う。また、事例調査も併せて行う。

asuca の開発においては、これらの一連のプロセスの進捗状況の管理や記録に Redmine を用いている。特に、2016 年度に進められた MSM 向け asuca の開発では、実験の数が最終的に大小合わせて 200 以上にのぼったが [6]、Redmine を実験の管理に活用することで、そ

[5] 加えて、共用アカウント上の共通の実験環境を引き継いで差分を加えていく事で、実験の間で意図せず仕様が異なってしまうことを防止することができる。

[6] 2016 年度に行った MSM 向け asuca の開発ではメソ解析システムに変更がないため、解析・予報サイクル実験をする必要がなく、必要な計算機資源が比較的小さかった。また、実験期間として連続する期間を設定する必要がなかったため、様々な現象（台風、梅雨前線、高気圧、冬型、南岸低気圧など）を含む事例を夏冬からそれぞれ 20 事例程度を選び出して、それらに対してインパクト実験（モデルの変更に対して個別事例の予測結果や統計的な性質の変化を確認する実験）

のような多数の進捗管理、検証結果の記録、実験結果の参照などを統一的に、かつ効率よく行うことができた。

(2) 評価実験の管理における Redmine の活用

以下では、これまでに述べた評価実験の管理における Redmine の活用法について、実際の開発に即して具体的に紹介する。

開発者間の議論により実験を行うことを決めたら、その実験に対応するチケットを作成し、実験の目的、変更内容などを記す。理論的背景や変更意図については、asuca や物理過程ライブラリで作成済みのチケットへ関連付けを行うことで参照する。実験を行う担当者が決まったら、担当者は NAPEX を用いて実験環境の構築を行い、実験の実行環境などをチケットにまとめる。レビューアによってその環境が妥当であるかの確認を受けた後、担当者は実験の実行を開始する。実験が終了したら、統計検証やモデルの予測結果への影響がどのようなプロセスによってもたらされたものなのか確認・分析し、チケットに記載する。

この一連の流れに対応して、「新規作成」「割り当て済み」「進行中」「実験環境レビュー」「実験環境承認済み」「実験中」「実験終了」「統計検証済み」「終了」というステータスを定義しており、実験の進捗とともにステータスを変更して、各実験の進捗を把握しやすいようにしている。

チケットに記録した内容は、メールや RSS で開発者間に共有されるため、情報共有及び議論のきっかけとしても有用である。実験環境は NAPEX 上でリポジトリ管理されているため、誰でも容易に再現することができる。また、これら実験のチケットは Redmine 上でまとめて表示することができ、各実験が現在どのようなステータスにあるか、これまでにどのような実験を行ったかを一覧で確認することができる。もちろん、行った変更の中には、物理過程間の相互作用などにより思わぬ結果が得られたものや、不採用になった変更も多いが、そのような情報も含めてチケットに残すことも非常に重要である。なぜなら、改善につながった変更はドキュメントや報告書という形で残りやすい一方、採用に至らなかった変更はドキュメント化されにくいが、そこで得られた知見はモデルの解釈やその後の開発の方向性を決める上で重要であるためである。

実際に MSM としての asuca・物理過程ライブラリの開発においては、多くの変更及びそれを評価するための実験を行った。これらの実験の中には採用に至らなかった変更が数多くあるものの、チケットに記録した内容はそのまま簡易なドキュメントとして残るため、開発者の記録用としてだけでなく、他の開発者への解説資料としても役立っている。

2.6.5 まとめと今後の課題

以上、メソモデルの開発において行われている開発管理についてその概要を紹介した。数値予報モデルという特殊なソフトウェアを対象とするものの、ソフトウェア開発管理としては、現在ではごく一般的となった手法を取り入れており、継続的なモデル開発を行っていく上で不可欠となる、ソースコードの品質の維持、向上に大いに役立っている。また、実験を行う際にも Redmine を利用することで、改善につながらなかった変更のように、ドキュメントに残りにくい重要な知見を記録する仕組みが出来上がっている。今後は、asuca-Var(幾田 2014) の開発が本格化し、解析サイクルを含めた実験の重要度が増すことになり、新たな工夫が必要となるが、これまでの経験を活かして、効率的に開発を進めていきたい。

参考文献

原旅人, 2012: 物理過程ライブラリの開発. 数値予報課報告・別冊第 58 号, 気象庁予報部, 205–208.

原旅人, 高谷祐平, 2013: 海外数値予報センターの開発管理の例. 数値予報課報告・別冊第 59 号, 気象庁予報部, 195–199.

原旅人, 幾田泰酵, 伊藤亨洋, 松林健吾, 2015: asuca が導入された局地数値予報システム. 平成 27 年度数値予報研修テキスト, 気象庁予報部, 1–23.

幾田泰酵, 2014: asuca 変分法データ同化システム. 数値予報課報告・別冊第 60 号, 気象庁予報部, 91–97.

石田純一, 藤田匡, 2014: asuca の開発理念. 数値予報課報告・別冊第 60 号, 気象庁予報部, 19–28.

石田純一, 河野耕平, 松林健吾, 2014: 理想実験を通じたドライモデルとしての評価. 数値予報課報告・別冊第 60 号, 気象庁予報部, 62–87.

気象庁予報部, 2014: 次世代非静力学モデル asuca. 数値予報課報告・別冊第 60 号, 気象庁予報部, 151pp.

Saito, K., T. Fujita, Y. Yamada, J. Ishida, Y. Kumagai, K. Aranami, S. Ohmori, R. Nagasawa, S. Kumagai, C. Muroi, T. Kato, H. Eito, and Y. Yamazaki, 2006: The Operational JMA Nonhydrostatic Mesoscale Model. *Mon. Wea. Rev.*, **134**, 1266–1298.

を実施した。インパクト実験の限られた事例数であっても、性能評価試験（夏冬それぞれ 2 週間、計 224 事例）、業務化試験（夏冬それぞれ 1ヶ月、計 448 事例）と同じ期間による実験に近い形でモデルの特性が評価できることがわかっており、このような手法によって一つの実験に必要な計算機資源を小さくすることが可能となり、多くのインパクト実験を実行することができた。

2.7 活用例 (3)–観測データ処理開発 [1]

2.7.1 はじめに

数値予報における観測データ処理では、先ず (a) 国内外から観測データを収集し、(b) 数値予報システムで利用しやすい形式に変換する。次に、(c) 誤差の大きなデータを除去したり系統的誤差を補正したりして観測データの品質を管理する。そして、(d) データ同化システムを用いて観測データを同化して数値予報の初期値を作成する。前述の通り数値予報課では、開発管理にプロジェクト管理システム Redmine を、バージョン管理システムでは主に Subversion を利用している。(a) 及び (b) は数値予報課プログラム班が担当し、(c) 及び (d) は同課数値予報班観測データ処理グループが担当している。観測データ処理については、それぞれで開発管理方針を定めており、かつ、相互に密に連携を取ることで補っている。ここでは、(c) 及び (d) に係る開発管理について述べる。以下、「観測データ処理」は特段の断りが無い限り (c) 及び (d) のことを意味する。

2.7.2 Subversion と Redmine の活用

数値予報で利用されている観測データは、地上観測や高層観測などに代表される従来型観測データから、静止気象衛星や地球観測衛星などによる衛星観測データなど多岐に渡る。また、データ同化システムも全球解析・メソ解析・局地解析・毎時大気解析など複数存在する。このような状況の中、観測データ処理に係る開発者は、品質管理用の実行プログラム及びデータ同化システムの実行プログラムに組み込まれている観測演算子を管理する必要がある。前者については、全ての観測データの品質管理を一つの実行プログラムで行うのは条件分岐が多くなり開発の効率が悪いため、ある程度の観測種別で分類して管理している。後者については、データ同化システム毎に管理している。

ここでは、観測データのうち、衛星観測輝度温度データのデータ処理に関する開発管理について説明する。観測データ処理に関する実行プログラムはひとつの Subversion リポジトリにまとめられており、品質管理に関する実行プログラムとデータ同化システム [2] の実行プログラムのソースコード、定数等が登録されている。前者は、観測データ処理グループで開発を管理している。一方、後者については、データ同化システム本体は、数値予報班の他のグループが開発を担当し、実行プログラムのソースコード、定数等を別のリポジトリで管理しているため、お互いに齟齬が生じないように注意する必要がある。観測データ処理では、現業運用されているデータ同化システム本体のソースコード、定数等に変更があった場合、それを自分たちで管理しているリポジトリのトランクに取り込むことで他のグループの開発に追随し、統一性を確保している。

Redmine には観測データ処理に関するプロジェクトの下に、衛星観測データに関する開発を管理する「衛星 QC」のサブプロジェクトを設けている。主に活用している機能は開発項目の進捗を管理するチケット機能であるが、情報共有のために Wiki 機能も活用している。チケットと実行プログラムをリンクさせるために、前述の Subversion リポジトリが Redmine に登録されている。

2.7.3 実例 – SSMIS (183 GHz) の利用開始

数値予報課では、米国の軍事気象衛星 DMSP に搭載されたマイクロ波放射計 (SSMIS) の 183 GHz 帯の輝度温度データを全球解析で新規に利用するための開発に取り組んでいる。この開発を、Subversion と Redmine の利用の具体例として示す。

この開発では、第 1.2.4 項に記載されている開発プロセスに則り、基礎開発及び性能評価試験を実施し、解析や予報の精度に与える影響について調査をした。本原稿の執筆段階では業務化試験を実施しており、これを経て現業化する予定である。Redmine サブプロジェクト「衛星 QC」に、基礎開発及び性能評価試験の評価に関する進捗を記録するためのチケット「SSMIS 輝度温度データ (183GHz) の利用」や「SSMIS F-17 UPP データの利用に向けた開発」を作成した。また、実験システムを構築する際には、別のチケット「SSMIS UPP 利用に向けた Napex 実験環境の確認」を作成し、動作確認の結果を記録すると共に、他の開発者と相互に確認を行うことで、ミスの混入を防止した。業務化試験は、他の輝度温度データ関係の開発課題と組み合わせて実施したため、新しいチケット「【業務化試験】2016T3：実験環境の構築」や「【業務化試験】SSMIS-UPP 追加」を作成した。Subversion リポジトリに業務化試験用の雛形ソースコードを用意し、各自が雛形ソースコードに加えた変更点及びマージ版の動作確認方法等を Redmine 上で共有・確認するなど、実験環境の構築にあたっては、各開発課題の仕様が適切にソースコードに反映されるように留意しながら作業を行った。

グループ内で利用方法等について意見交換を行った場合には、その記録をチケットに保存し共有している。また、Redmine は、これまでの開発で作成されたツール類の共有にも活用されている。今回の開発でも、183 GHz 帯以外の SSMIS の輝度温度データの利用に向けた開発 (江河・計盛 2009) の際に作成された観測データの可視化のためのツール類等を活用している。

2.7.4 最後に

観測データの品質管理では、それぞれの観測データに合わせて複雑な処理が行われているため、観測データの種別に応じて実行プログラムを分けて開発している。今後も、開発管理のためのシステムやツールを活用した開発体制の構築を進め、解析予報精度の改善に取り組みつつ、より堅固な数値予報システムとすることを目指す。

参考文献

江河拓夢, 計盛正博, 2009: マイクロ波放射計 SSMIS の利用. 平成 21 年度数値予報研修テキスト, 気象庁予報部, 54–56.

[1] 本田 有機、計盛 正博、村上 康隆

[2] 全球解析及びメソ解析のみ登録されている。局地解析は、開発体制により現時点では別の Subversion で管理している。

2.8 活用例 (4)–共通基盤 [1]

2.8.1 概要

開発管理サーバでは、モデル開発に関連する基盤技術の開発とユーザ支援を目的に共通基盤 Redmine を運用している。共通基盤 Redmine では共通基盤ライブラリ、気象庁自主開発のジョブスケジューラ (ROSE) などの自主開発ミドルウェア、第 3 章で解説する NAPEX や第 4.2.2 項で解説する NuSDaS といった数値予報システム開発全体の基盤となる各種ツールの開発管理を行っている。共通基盤 Redmine の利用者としては主に数値予報課プログラム班と数値予報課数値予報班基盤整備グループのメンバーが登録されているが、本節では基盤整備グループが担当している開発管理の内容を中心に説明する。

基盤整備グループでは、NAPEX 及び共通基盤ライブラリの担当者が日常的な開発業務で開発管理サーバを利用しており、担当者が開発の要所でそれぞれ相互にレビューを行うことを開発プロセスの基本として採用している。本節では基盤整備グループでの開発管理の事例として第 3.3 節で述べる NAPEX モデル [2] の管理について概説するとともに、共通基盤 Redmine は開発管理サーバ全体の開発基盤としての役割も担っていることから、全体の管理に関する話についても併せて解説する。

2.8.2 NAPEX モデルの管理事例

基盤整備グループでは実験用の数値予報システムである NAPEX モデルの維持・管理を行っており、この管理作業に開発管理サーバを利用している。NAPEX モデルの管理作業では JDF（第 3.2.3 項 (2)）などを除き、最新数値予報ルーチンとほぼ同等の実験用の数値予報システムを構築することが重要となる。NAPEX モデルの構築においては数値予報システムの入力として利用される観測データや初期値・境界値などのデータである引継ぎデータの登録、実験本体の設定、雛形実験やコントロールデータの作成といった複数の作業が必要である（第 3.3 節）。

基盤整備グループではこれら複数の作業の管理に子チケットによる管理を導入している。Redmine ではチケット同士に関連付けができ、この機能によってチケット間に親子関係を導入することができる。NAPEX モデルの管理では、作業全体の進捗管理を親チケットで行い、個別の作業の管理は親チケットに結びつけられる子チケットを用いて管理するという方法を用いている。図 2.8.1 に、子チケットを利用して NAPEX モデル更新の進捗管理を行った例を示す。この例では全球 NAPEX モデルの更新作業が行われた。図の下部に並んだ各項目が、分割された各作業に関する子チケットである。子チケットの進捗状況は親チケットから簡単に参照することができる。さらに、親チケットの進捗状況（図の上段）は子チケットの進捗に連動しており、子チケットの進捗に応じて自動的に全体の進捗が一目で把握できるようになっている。

子チケットを用いた進捗管理では、子チケットを五月雨式に作成しない点に留意すべきである。進捗が進むたびに子チケットを追加すると、全体の進捗状況が把握できなくなる。親チケットを作成した時点で、必要となる子チケットを予め作成しておくことが望ましい。これには親チケットを作成した時点で必要となる作業を明らかにしておく必要があり、子チケットとして分割するべき作業の洗い出しが必要となる。この作業によって必要な作業の見通しを作業者自身の手で自然と把握することができるようになっている。NAPEX モデルの管理では、子チケットの内容は相互に関連しているものの独立性があり、しかも定型的な作業が多いため事前に見通しが立てやすく、子チケットによる管理に適している。

分割した子チケットに沿って作業を進めていくわけだが、チケットで行う作業の中には基盤整備グループ以外の開発者へ影響が大きい作業も存在する。たとえば、NAPEX モデル本体の更新作業や引継ぎデータの登録作業は、登録結果が他のユーザに利用される前提で作業を行う。こうしたものを不適切な設定で公開してしまうと、利用した多くのユーザの実験結果が誤ったものとなり、実験のために消費された時間と労力が無駄になるといった状況になりかねない。そこで重要な作業のチケットでは相互レビューを必ず行い、設定に問題ないかどうかをレビューアが確認することにしている。指針決定後に実施された NAPEX モデル更新では、全ての更新作業で相互レビューが行われた。相互レビューを行ったことで、登録期間の誤りや定数ファ

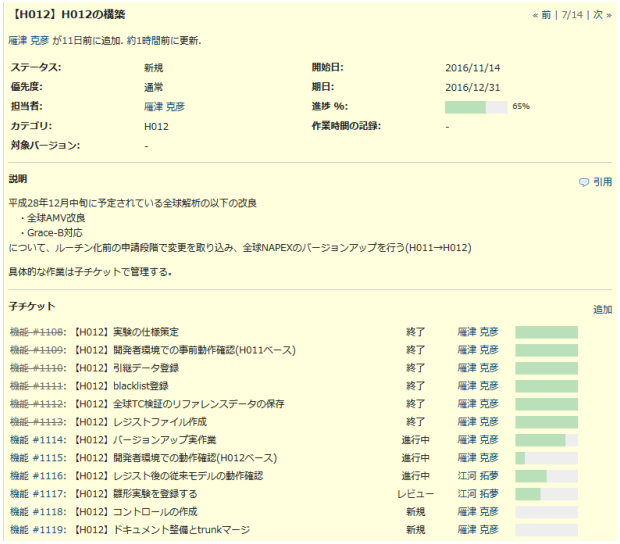

図 2.8.1 子チケットによる進捗管理

[1] 雁津 克彦
[2] NAPEX 環境で動作させる数値予報システムのこと。

イル・ソースコードの登録漏れといったミスが事前に回避された実績があり、相互レビューの導入は信頼性向上に大きな貢献をしていると考えている。

2.8.3 共通プラットフォームとしての共通基盤 Redmine

共通基盤 Redmine は数値予報システム開発の全体の基盤を担っているため、開発管理サーバ全体の情報や共通ツールなどの情報を広く提供する共通プラットフォームとしての役割も果たしている。

開発管理サーバの運用開始当初から継続しているプロジェクトとして「開発管理サーバ（ユーザサポート）」プロジェクトがある。このプロジェクトではフォーラムに「質問掲示板」を設けており、開発管理サーバの利用方法に関する疑義や提案事項がある場合に、管理者への質問や議論を行うことができるようになっている。フォーラムにはこの他にも「メンテナンス情報」のページを用意しており、開発管理サーバのメンテナンスに関する事前告知や新規プラグインの導入に関する情報を掲載し、開発管理サーバを利用する上でのユーザサポートの場を提供している。

2015 年には開発管理調整グループ会合で開発情報の蓄積を目的としたプロジェクトの新設が議論され、新たに「開発情報」プロジェクトが開設された。このプロジェクトでは、ユーザが作成した各種ユーティリティツールの共有を行うことができるようになっており、個別の数値予報システム向けに限らない汎用的なツールを作成した場合に、気軽にツールを公開できる環境を提供している。また、現在利用している第 9 世代スーパーコンピュータシステム (西尾 2011) での開発に関する情報の提供が行われており、一例として OS のイメージ更新 [3] などで開発環境に変更が生じた場合には、変更内容の情報を参照することができるようになっている。

加えて 2016 年には、2018 年に予定されている第 10 世代スーパーコンピュータシステムへの移行を見据えて、新システムへの移植を支援する「NAPS10」プロジェクトが導入されることとなった。

このように、共通基盤 Redmine は共通基盤に関する開発管理にとどまらず、開発管理サーバを利用する全ユーザに対しての共通した情報提供を行う役割も担っている。

参考文献

西尾利一, 2011: 計算機（スーパーコンピュータシステム）. 平成 23 年度数値予報研修テキスト, 気象庁予報部, 68–70.

[3] OS アップデートに相当する作業のこと。スーパーコンピュータシステムでは複数あるノードで更新が必要となるため、イメージファイルを各ノードに適用する形で更新作業を行う。

2.9 活用例 (5)–アプリケーション開発 [1]

2.9.1 はじめに

本節では数値予報の応用処理としてガイダンスやFAX 図など、アプリケーション (松下 2012) の開発を行っている数値予報課アプリケーション班（以下、AP 班）での開発管理について説明する。AP 班の担当するガイダンスや FAX 図を作成するジョブは、数値予報モデルに比べると一つ一つの規模は小さいものの数は多く、数十個にも及ぶ。そのため複数のジョブを担当者一人で管理・開発する体制を執ってきたため、以下のような問題点があった。

- ドキュメントが個人の Wiki ページや Word 文書などで作成されており、引き継ぎ時に新規担当者が自環境に移設する手間が発生するほか、担当者以外がドキュメントを確認・把握することが困難。
- 各ガイダンス間でソースコードの記述形式や開発管理体制などが統一されておらず、新規担当者への引き継ぎコストが大きい。
- ルーチン変更までの手順がルール化されておらず、担当者の裁量に依るため、バグの混入やミスを誘発しやすい。

本節で説明する統一的な開発管理や、ルーチン変更におけるガイドラインの導入により、この状況は改善されてきている。

2.9.2 Redmine と Subversion による開発管理

AP 班では 2010 年から Redmine と Subversion を利用し、前項の問題解決に努めてきた。まずドキュメントを Redmine の Wiki に集約することで、閲覧性を向上し、新規担当者への引き継ぎも円滑に行えるようになった。また各開発項目はルーチン変更に限らず、Redmine のチケット機能を利用して管理することとした。そしてチケット番号をドキュメントや Subversion のリポジトリと関連付けることで、変更点やその意図を把握するのに大いに役立てている。

2.9.3 ガイダンスにおける共通モジュール

第 2.9.1 項で述べた通り、AP 班では開発業務が個人単位で行われていたため、ソースコードの記述形式がほとんど統一されていなかった。こういった状況は、引き継ぎコストを大きくするだけでなく、開発効率の低下やバグの混入、ミスを誘発する。そこで 2012 年から各ガイダンスで共通して利用される機能（地点テーブルの読み込みや GPV の内挿、係数更新の手続など）を共通モジュールとして整備し、各ガイダンスに導入した。この共通モジュールを利用してガイダンスを開発することで、新規にソースコードを書く必要が無くなり、動作実績のあるソースコードを利用できるなど、開発効率・動作の信頼性が大幅に高まった。

2.9.4 ルーチン変更におけるガイドライン

AP 班のジョブは、ユーザが直接利用するという特性上、軽微なルーチン変更（アメダス移設や空港の観測時間変更によるテーブル変更や配信要素の変更など）が多く、かつ従来はその手順が個人依存で、ケアレスミスを誘発しやすい状況であった。そこでルーチン変更時のミスやバグの混入防止のため、2015 年に Redmine と Subversion を利用したルーチン変更ミス防止ガイドラインを設けた。ガイドラインの概要は以下のとおりである。

- Redmine を利用した開発管理を必須とし、開発の目的や経緯、スケジュールなどを記入する。
- チケットのウォッチャーに副担当を入れ、進捗を随時記載することで、情報の共有を行う。
- ソースコード類は Subversion 管理を必須とし、数値予報ルーチンからの変更点について Subversion の差分機能を使って示す。
- ルーチン変更に合わせ、作成したプロダクトの品質を確認するためのモニタとドキュメントの作成・更新を行う。
- ある一定期間以上の実行試験を行い、挙動やログに問題点がないかを確認する。
- ルーチン変更申請作業者は変更に先立ち、副担当に変更点について説明、また上記項目が適切に行われているか確認を受ける。

このミス防止ガイドラインの利用により、担当者に依存していたルーチン変更までの手順が統一・明文化され、副担当による確認作業がより確実かつ効率的に行えるようになった。また、このガイドラインとそれに付随するチェックリストにより、見逃されがちなミスが実際に検出されるなど、より確実な変更作業が行えるようになった。Redmine と Subversion はこれらの確認作業を円滑に進め、その記録の保存に必須のツールとなっている。

2.9.5 AP 班における開発管理の今後について

AP 班で開発管理に Redmine と Subversion を利用するようになって 6 年以上経ったが、ルーチン変更におけるガイドラインの作成や円滑な情報共有等のように、利用していく中でよりよい使い方が見出されることがある。また、描画や検証についてもツールの共通化を進め、他の開発者でも利用しやすいよう汎用性を高めることで、開発コスト削減となったケースもある。今後も開発効率の向上やミス防止を目指して開発管理ツールの使い方を模索していきたい。

参考文献

松下泰広, 2012: アプリケーション. 平成 24 年度数値予報研修テキスト, 気象庁予報部, 42–53.

[1] 後藤 尚親

2.10 活用例 (6)–システム管理 [1]

2.10.1 概要

数値予報課プログラム班（以下、P 班）における開発管理及び、数値予報ルーチンの管理への応用について述べる。

P 班では、数値予報システムの前処理･後処理として、観測データのデコード (佐藤 2012; 西尾 1995)・FAX 図等のプロダクト作成を担当しているほか、NAPEX でも利用されている ROSE (Routine Operation and Scheduling Environment) の開発等を行っている。ここでは、P 班の開発管理の代表例として第 2.10.2 項で ROSE プロジェクトを取り上げる。

また、P 班では数値予報ルーチンの管理を行っている [2]。数値予報課の各班・グループ、他課室の開発成果は、P 班にて「ルーチン変更」（後述）を実施することにより、数値予報ルーチンの改善として反映される。「ルーチン変更」を実施するにあたり、その作業内容の記録と共有を Redmine を活用して実施しているので第 2.10.3 項で紹介する。

2.10.2 ROSE の開発管理

(1) ROSE について

ROSE とは、P 班が主体となって開発しているジョブフロー制御ソフトウェア（ジョブスケジューラ）である。2008 年に開発が始まり、2009 年 6 月には予報支援資料作成 BCP (Business Continuity Plan) 対応装置 [3]（以下、BCP サーバ）での運用を開始した。BCP サーバでは、現在でも毎日 5 回、米国環境予測センター (NCEP: National Centers for Environmental Prediction) からのデータの取得とガイダンス・FAX 図を作成するジョブグループ [4]（以下、JG）が実行されている。

第 9 世代スーパーコンピュータシステム [5] においては、NAPEX に代表される開発業務において利用され、2018 年 6 月運用開始予定の第 10 世代スーパーコンピュータシステムでは、数値予報ルーチン及び各課ルーチンの実行にも ROSE を導入する計画となっている。なお現在は、スーパーコンピュータシステム導入ベンダー作成のジョブスケジューラ「数値予報ルーチン業務運用支援ソフトウェア (JNOS) 」にて運用している。

一般にジョブスケジューラとは、複数のジョブの起動や終了を制御したり、ジョブの実行・終了状態の監視や報告などを行うソフトウェアである。ROSE も同様であり、主な機能を挙げると次の通りである。

- JG を構成するジョブを依存関係に従って実行する
- 一連の JG を依存関係に従って実行する
- 指定した時刻に JG・ジョブの実行を開始する
- 実行開始時刻を指定しない JG を連続して実行する（開発用途向け）
- JG・ジョブの強制的な実行や実行中止 (a)
- JG・ジョブに与える環境変数の変更 (b)
- (a), (b) の操作を GUI で視覚的に行う
- ジョブ異常終了時にユーザへ通知する

ただし、ROSE 単体ではジョブを実行することはできず、バッチジョブのキューイングシステムとして LoadLeveler[6] 等を利用している。

また ROSE は、以下の構成要素から成る。

- ROSE DB[7]：全ての情報を管理
- ROSE Server：データベースの制御
- ROSE Agent：ジョブの投入制御
- ROSE GUI (CGI) ：監視・操作

Server, Agent, GUI (CGI) は、データベースを介して相互に情報をやりとりし、連携して動作する。

(2) 開発体制

ROSE は、Server, Agent, GUI (CGI) それぞれ独立して開発作業が進められるようになっているため、各構成要素に対して担当者が割り当てられている。ROSE の開発は、開発当初から P 班が行っていたが、第 10 世代スーパーコンピュータシステムでルーチン利用することも踏まえ、情報通信課システム運用室においても、ROSE の改善要望の集約や GUI (CGI) のソースコードの改修を行っている。

(3) 開発の進め方

ROSE の機能の改善・追加要望やバグ報告がユーザから上がり、それらの開発課題に対応していくことが基本的な開発の流れである。

開発課題は、まず Redmine のチケットに登録する。この時、Server, Agent, GUI (CGI) のどの課題なのか明確にするため、担当者を設定するとともに、チケットのカテゴリも設定する。

担当者はそれぞれ独立して開発作業を進めるが、開発方針で調整が必要な場合等は、適宜担当者間で調整する。決定事項はチケットに記述し、確実に記録が残るようにする。なお、独立した開発作業には、当然な

---

[1] 河野 正和

[2] 数値予報ルーチンの運用は、情報通信課システム運用室現業にて実施される。

[3] スーパーコンピュータシステムの大規模障害等で数値予報ルーチンプロダクトを提供できない場合に備えて、予報作業で必要となる最低限のプロダクトを作成するバックアップシステムを検証する装置のこと。

[4] ある一連の処理を行う複数のジョブの集合のこと。

[5] 第 7 世代以前は「数値解析予報システム (NAPS: Numerical Analysis and Prediction System)」と呼ばれていたが、第 8 世代からは、静止気象衛星資料処理局電子計算機システム (DPC: Data Processing Center System) と統合一体化し、「スーパーコンピュータシステム」と呼んでいる。

[6] `http://www.ibm.com/support/knowledgecenter/SSFJTW/loadl_welcome.html`

[7] DBMS (DataBase Management System) として PostgreSQL を利用している。

がら単体試験[8]も含まれている。

そして、それぞれの開発成果は、年1, 2回程度のアップデートのタイミングにあわせて結合試験[9]・総合試験[10]を行い、問題がないことを確認した上で、ROSEの更新を実施する。

上述の開発作業において、ソースコード等はSubversionによりバージョン管理を行う。各担当者の開発はブランチで行い、単体試験が終わったものはトランクにマージする。そして、結合試験・総合試験が終了してROSEを更新するタイミングでタグを作成し、ユーザにはタグから作成したROSEの環境を利用してもらうことにしている。

P班では、ROSE開発初期から、P班独自のサーバに整備したRedmineとSubversionを利用して開発を行ってきた経緯があるが、「(2) 開発体制」でも述べた通り、今後の体制強化も視野に入れて、Redmineプロジェクトと Subversionリポジトリを P班独自のサーバから開発管理サーバ上に移し、開発作業の連携や情報共有が行い易くなるようにした。

2.10.3 数値予報ルーチンの管理への応用

(1) 背景

数値予報ルーチンのジョブ及びJGは、以下から構成されている。

- JCLから生成されるスクリプト
- PBFから生成される実行プログラム
- 定数ファイル
- ジョブが生成する可変データ[11]
- JDFで定義され、ジョブスケジューラに登録されるJG・ジョブの依存関係

JCL, PBF, JDFについては、第3.2.3項(2)を参照されたい。

スーパーコンピュータシステムで数値予報ルーチンを運用するにあたって最も重要なことは、毎日決まった時刻にプロダクトを提供できるように、安定してジョブが実行されることである。数値予報ルーチンの安定運用とは、ハードウェア障害などの外的要因を除いた場合、現在動作しているスクリプトや実行プログラム等を全く変えることなくジョブを実行し続けることである。しかし実際には、開発成果を取り込み、数値予報の精度向上を図る必要があるため、スクリプトや実行プログラム等を変更する作業を実施することになる。この作業を、「ルーチン変更」と呼んでいる。ルーチン変更には人が介在するためにどうしてもミスが発生する可能性があり、ジョブが異常終了するなど、安定運用を乱す結果となる場合がある。このため、P班ではルーチン変更に起因する障害を無くすための取り組みを継続してきた。

(2) データベースとSubversionの利用

第3.2.3項(2)で述べている通り、第7世代数値解析予報システムでは、自由度が高くなる利点を得た半面、ミスを誘発しやすい環境でもあった。さらに、数値予報ルーチンもより複雑になり、管理するジョブも増加する一方であった。このため、次のようなミスが多かった。

JGの実行に必要な引継ぎデータがない

各JGは、JG内のジョブの実行に必要な引継ぎデータを最初のジョブで用意する。このジョブを「START」と呼んでいる。STARTのシェルスクリプトは、当該ルーチン変更を担当するP班員が作成していたが、データセットの記述漏れ等により引継ぎデータがなく、ジョブが実行できないことがあった。

定数ファイルがない

ルーチン変更申請を忘れたために定数ファイルがなく、ジョブが実行できないことがあった。

可変データが作成されていない

ジョブの依存関係を間違えたために、入力とするデータセットが後続のジョブで作成されていて、ジョブが実行できないことがあった。

ジョブの実行に必要なノードが足りない

スーパーコンピュータのノード数は有限なので、同時に実行できるジョブ数は制限される。しかし、JGのスケジューリングが悪いと、必要なノード数が不足してジョブの実行ができず、その結果、数値予報ルーチンの遅延を招くおそれがあった。

ルーチン変更時の引継ぎデータの準備不足

ルーチン変更の内容によっては、ルーチン変更時に引継ぎデータを準備しておく必要がある。しかし、ルーチン変更作業に手間取ってJG開始までに準備が終わらない、またはそもそも用意しないといけないことを忘れていたなどの理由で必要な引継ぎデータが不足し、ジョブの実行ができないことがあった。

このようなミスを減らすため、ルーチン変更作業の運用面からは、ルーチン変更1件につき、作業者1名、確認者1名を割り当てて複数名での作業とするなどの対応をしてきた。しかし、人による申請内容のチェックには限界があるため、システマティックに解決する手段が求められた。このため、第7世代数値解析予報システム運用中にJCLを開発し、ジョブの入出力が分かり易くなるよう改善した。さらに第8世代スーパーコンピュータシステム運用開始に向けてPBFとJDFを開発し、JCL等の内容を全てデータベースに格納し

[8] 各構成要素内の1つのプログラム、1つの機能のテスト。
[9] 各構成要素を任意に結合させたテスト。
[10] ROSEとして要求を満たしているか総合的に確認するテスト。
[11] JG実行毎に作成されるプロダクト等のこと。JG内や他のJGのジョブの入力データとしても利用される。

て連携させた。これにより、JG やジョブ間の矛盾等はなくなり、START ジョブも自動生成が可能となった。

また、JCL, PBF, JDF やソースコード等のファイルは Subversion に登録し、ルーチン変更情報と結びつけてバージョン管理することで、万が一問題が発生した場合でも、簡単に以前のバージョンに戻すことが可能になった。

このデータベースと Subversion を用いて管理するシステムを、数値予報ルーチン管理システム (RENS: Routine Environment for Numerical weather prediction System) と呼んでいる。JCL や JDF などの情報を有機的に連携して管理する RENS を開発したことは、これまでの取り組みの中でも大きな成果であった。

(3) Redmine 導入のきっかけ

数値予報ルーチンを RENS で管理するようになって、ルーチン変更に起因するミスは大幅に減少した。しかし、Redmine 導入以前は、ルーチン変更に係る作業内容は、作業者と確認者の間でメールにより共有するのみであった。このため、そのルーチン変更に関わらない他の P 班員は、どのような試験が実施されたかを知る機会は基本的に無く、ルーチン変更内容に対して十分な試験が行われたか分からない状態であった。

そのような中で第 9 世代スーパーコンピュータシステムへの更新作業が始まり、数値予報課内のルーチン移行の進捗管理のために、ROSE の開発で利用してきた Redmine を導入し、チケットのステータスで進捗を把握するようにした。また進捗を管理すると共に、移行作業においてどのようなことを実施したか、どのような問題が起きているかをチケットに記載することで、移行に係る情報共有が容易になり、それらを記録としてきちんと書き残すことができた。

この Redmine の特徴をルーチン変更作業にも取り入れれば、ルーチン変更の作業内容の共有と、記録として残していくことが容易にかつ同時にできると考えて、第 9 世代スーパーコンピュータシステムのルーチン変更作業に Redmine を導入することになった。

(4) Redmine の利用方法と導入の効果

ルーチン変更作業に Redmine を導入するにあたり、以下のように利用することにした。

- ルーチン変更 1 件に対してチケット 1 件を自動発行する
- チケットのデフォルトのウォッチャーは、作業者と確認者の 2 名とする
- 作業者は、試験内容についてチケットの注記に詳細に記述する
- 確認者は、確認内容についてチケットの注記に記述する
- 試験終了、確認終了などのフェイズに併せてチケットのステータスを更新する
- ステータスを更新することにより発信されるメールを契機に、作業者・確認者は次の作業を始める

担当者間でメールで行っていたことが、ウェブベースに変わっただけと思われるかもしれないが、両者には大きな違いがある。それは、作業内容を記録としてきちんと残していくことが可能になったとともに、他の P 班員が能動的に試験内容を知る機会が生まれたことである。これにより、試験内容がさらに多くの人の目に触れることになるだけではなく、それぞれのルーチン変更に対して、担当者の技量の差などから生まれるミスが軽減され、知見を共有することで数値予報ルーチン全体としての品質の向上につながったのである。

また、ルーチン変更に起因する障害の際には、従来はルーチン変更担当者でないと実施内容が不明なために対応が難しい面もあったが、Redmine を導入してからは、他の P 班員に情報が共有されることにより、担当者以外による対応が容易になった。

2.10.4 おわりに

P 班の開発管理についての取り組みと、開発以外の分野への応用例を紹介した。Redmine などのツールを上手く利用することで業務の品質は向上し、良い結果をもたらしたことは明白である。しかし、それぞれのコミュニティに合ったやり方でツールを導入しなければ、開発効率を落とすだけではなく、業務の品質も落とすことになるので、その点には注意が必要と考える。

参考文献

西尾利一, 1995: 観測データとデコード・ソート. 平成 7 年度数値予報研修テキスト, 気象庁予報部, 14–16.

佐藤芳昭, 2012: デコード. 平成 24 年度数値予報研修テキスト, 気象庁予報部, 14–15.