中央農研研究報告 20: 1 - 115(2013)
Bull.NARO.Agric.Res.Cent.

A Study on a Framework for Distributed Cooperative System in an Agricultural Simulation Model

Kei Tanaka*

Summary

People have dealt with ever-faster population growth by increasing food production based on the development of agricultural technology. However, food shortages in developing countries have been left unsolved due to abnormal climates, westernized eating habits in emerging countries, and the use of cereal as a raw material for bio-ethanol. Also, in Japan, we are facing changes to the cultivating season and the best breed of farm product caused by climate change, and the problem of the knowledge of aging farmers not being passed on. In these circumstances, information technology centering on agricultural simulation models (hereinafter called "agricultural model") to help farmers make decisions is playing a more and more important role.

Affected by the results of system dynamics, the development of a plant growth model to dynamically describe plant growth began in the 1970s. And then, an agricultural model such as the growth prediction model, and disease and pest damage forecasting model, became widely used as an alternative way to help farmers make decisions and as an alternative to field trials. In the 1990s, large agricultural models were developed by each development group of Wageningen, IBSNAT, and APSRU, and the paddy-rice growth prediction model SIMRIW was developed in Japan.

Agricultural data includes meteorological data that is required by the agricultural model, and cultivation data that is used to develop agricultural models and assume parameters. Among data that has been obtained at weather stations and experiment stations for many years, some may be recorded only for printed material, while other data may be put into a database and shared over a network.

The domestic agricultural model and database are characterized by their high regionality with small size and dispersal to universities and research institutes. And also many programs of the agricultural model have become legacy and many databases are operated in a specific manner. Thus, there is a need to build a decision making system connected to these various agricultural models and databases, considering future operation and maintenance. To this end, it was thought that a distributed cooperative system, which manages agricultural models and databases at the development site and connects with networks, is suitable.

In this dissertation, we suggested a distributed cooperative system to help farmers make decisions, and named it the Agricultural Model and Databases with Distributed Cooperative System (AMADIS). To solve problems in connecting components such as agricultural models and databases, AMADIS needs to have a search function allowing people to find a suitable agricultural model and data, a management function to manage the location information of each component on the network, communication protocols to connect multiple components, and an executive function to execute agricultural models. In this dissertation, we conducted research mainly on a method to develop a program in the agricultural model as a component and the cooperative method between components, from the functions required for

*NARO Agricultural Research Center Agroinformatics Division

AMADIS.

Analyzing the legacy program of a domestically developed agricultural model showed that the calculation part of the program—the core of the agricultural model—other than data reading and result display, accounts for approximately half of the program. The program size of the domestic agricultural model is not large and some Web applications of the agriculture model as components need to be developed quickly. Therefore, translation into Java was adopted rather than developing wrapper programs for the legacy model. Java is an object-oriented language, and developing programs in a way that exploits its characteristics not only makes development more efficient but also makes future expansion and maintenance easier. It also has standard APIs that are useful for building Web applications, such as networks, distributed objects, thread, XML, and multi-language support. After that, Java became a main language for building server-side applications and it is apparent that the choice of development language was right.

When agricultural programs were developed, we studied the method of developing models and peripheral technology using three agricultural model development groups. These three groups developed while influencing each other in the 1990s, so their developed structures were similar. All of them have a module structure and center on a model execution engine to manage model calculations. They all use modules such as growth models, disease models, and soil models as components. Calculation and accumulation are repeated on the execution engine until the termination criteria are met by time loop after initialization. The meteorological data required for executing the agricultural model are provided by a program called a "weather generator."

To develop the program of the agricultural model in this dissertation, a framework for an agricultural model was first built and programs of the agricultural model were secondary developed using a framework. The framework is a program library which is organized to be reused for building the specific target application. The framework is also a semi-finished application so developing the program of the agricultural model requires only deficient parts of the default functions provided by framework. We named this framework Java Agricultural Model Framework (JAMF).

JAMF consists of many program packages, such as the model execution engine, model data, the weather generator, user interfaces, and utility programs. The model execution engine is simpler than the ones used by the three development groups but has the same functionalities. The model data class effectively processes Boolean value, numeric value, date, and time-series data that are handled by the agricultural model, and helps to automatically build user interfaces. The weather generator acquires meteorological data used by the agricultural model from MetBroker, average year data, estimation data, and user data, and converts them for use in the model. MetBroker is middleware between various weather databases and the agricultural model, and provides unified database access methods and data forms. The weather generator allows the data at weather stations all over the world to be used via MetBroker, and normal year value, estimation data from the meteorological model instead of non-observation data, and data observed by users can be used to make predictions, as required.

To develop the programs of the agricultural model using JAMF, approximately ten programs related to agricultural model calculations, data, and interfaces need to be developed. Those are additional programs to the functions provided by JAMF. The number of lines of source code of MetBLASTAM translated into Java from FORTRAN took only 5.2% (about 1,000 lines, the ratio of source code without comment lines) of all functions of the agricultural model. In more detail, 66% was for calculations that varied by model and 4% was for data acquired, part of

摘要

人類は農業技術の進歩による食糧増産によって加速度的な人口増加に対応してきたが，異常気象，新興国の食生活の欧米化，バイオエタノール原料としての消費などにより，途上国の食糧不足は未だに解消されていない．さらに，国内では気候変動による栽培時期や最適品種の変化，高齢化した篤農家の知識が失われつつあるという問題にも直面している．このような状況において，意思決定支援のための農業シミュレーションモデル（以降，農業モデル）を中心とした情報技術の果たす役割が大きくなっている．

システムダイナミクスの成果に刺激を受け，植物成長を動的に説明する植物生育モデルが開発され始めたのは1970年代であった．その後，生育予測モデルや病害虫発生予察モデルなどの農業モデルが意思決定支援や圃場試験の代替として広く利用されるようになった．1990年代には大規模な農業モデルがWageningen，IBSNAT，APSRUの各開発グループにより開発され，国内では水稲生育予測モデルSIMRIWが開発された．

農業関連のデータには，農業モデルの実行に必須である気象データ，農業モデルの開発やパラメータ推定に利用される栽培データなどがある．古くから各地の観測所や試験場で観測されてきたデータの中には，印刷物としてしか記録されていないものがある一方，データベース化されてネットワーク経由で利用できるものもある．

国内の農業モデルやデータベースの特徴は，小規模で地域性が高く，大学や研究機関などに分散していることである．また，農業モデルのプログラムにはレガシー化してしまっているものも多く，データベースはそれぞれの操作方法が異なっている．これらの多様な農業モデルやデータベースを結びつけて意思決定支援システムを構築する場合，その後の運用や保守を考慮すると，農業モデルやデータベースを開発元で管理し，ネットワークで結びつける分散協調型が適していると考えられた．

本論文では，意思決定支援を行うための分散協調システムを提案し，AMADIS（Agricultural Models and Databases with Distributed Cooperative System）と名付けた．農業モデルやデータベースなどの構成要素を結びつけて問題解決を行うために，AMADISには，適した農業モデルやデータを見つけるための検索機能，要素のネットワーク上での所在情報を管理する機能，複数の要素を結びつけるための通信プロトコル，農業モデルを実行する機能などが必要である．本論文ではAMADISに必要なこれらの機能のうち，主に構成要素としての農業モデルのプログラム開発手法，要素間の連携手法についての研究を行った．

国内で開発された農業モデルのレガシープログラムを分析したところ，データの入力や結果表示以外の，農業モデルの中核である計算部分のプログラムの割合は半分程度であった．国内の農業モデルのプログラムサイズが大きくないことと，構成要素としての農業モデルWebアプリケーションを短期間にいくつか開発する必要があったことから，レガシーモデル用のラッパープログラムを開発して対応する方法ではなく，Javaに移植する方法をとることにした．Javaはオブジェクト指向言語であり，その特徴を生かしてプログラム開発を行えば，開発効率の向上だけでなく，その後の拡張や保守も容易になる．また，ネットワーク，分散オブジェクト，スレッド，XML，多言語対応などのWebアプリケーション構築に有用な標準APIを持っている．その後，サーバサイドアプリケーション構築の主流言語となり，開発言語選択は正しさが示された．

農業モデルのプログラムを開発するにあたり，3つの農業モデル開発グループによるモデル開発手法と周辺技術の研究を行った．3グループとも1990年代にお互いに影響し合いながら開発を行ったため，似たような仕組みになっている．いずれもモジュール構造をとり，モデルの計算を管理するモデル実行エンジンを中心とし，そこに生育モデル，病害モデル，土壌モデルなどのモジュールを部品として組み込むようになっている．実行エンジンでは初期化後，時間ループにより終了条件を満たすまで計算と集計が繰り返される．農業モデルの実行に必要な気象データは気象ジェネレータと呼ばれるプログラムにより提供される．

本論文での農業モデルのプログラム開発では，先ず農業モデル用フレームワークを構築し，それを利

用して農業モデルのプログラムを開発することにした．フレームワークは特定の目的のアプリケーション構築のために再利用できるようにまとめられたプログラムライブラリである．フレームワークは半完成品のアプリケーションであるので，農業モデルのプログラム開発は，フレームワークで提供されるデフォルト機能に足りない部分のみで済む．このフレームワークをJAMF（Java Agricultural Model Framework）と名付けた．

JAMFはモデル実行エンジン，モデルデータ，気象ジェネレータ，ユーザインタフェース，ユーティリティプログラムなどのプログラムパッケージから構成される．モデル実行エンジンは3つの開発グループのものより簡易であるが，同様な機能を持っている．モデルデータクラスは農業モデルが扱う真偽値，数値，日付，時系列値を効率的に処理し，ユーザインタフェースの自動的な構築に貢献する．気象ジェネレータは農業モデルが利用する気象データを，データ取得元であるMetBroker，平年データ，推定データ，ユーザデータから取得し，実行用に加工する．MetBrokerは様々な気象データベースと農業モデルの間にあって，統一的なデータベースアクセス手法と，データ形式を提供するミドルウェアである．気象ジェネレータにより，MetBroker経由で世界中の気象観測地点のデータを利用でき，必要に応じて未来予測のために平年値や，未観測データの代わりに気象モデルによる推定データや，ユーザが観測したデータを利用できる．

JAMFを利用して農業モデルのプログラムを開発するには，農業モデルの計算，データ，インタフェースに関する約10個のプログラムを開発することになる．それらはJAMFで提供される機能に対する追加部分のプログラムである．FORTRANからJavaに移植したMetBLASTAMのソースコードの行数を数えたところ，MetBLASTAM用に開発したコードは，農業モデルのすべての機能のうちの5.2%（約1000行，コメント行を除いたソースコードの割合）にすぎなかった．その内訳は，モデルごとに異なる計算部分が66%，各モデル共通のデータ取得部分が4%であった．JAMFを利用して約20の作物生育予測モデルや病害虫発生予察モデルなどを実装することにより，多様な農業モデルWebアプリケーション構築に利用できることを示した．ドキュメントの整備された農業モデルで，複雑なユーザインタフェースを必要としなければ，1～2日間でAMADISの構成要素となるWebアプリケーションを構築できる．

AMADISの構成要素となるためには，要素間でネットワークを介したメッセージ交換機能が必要である．当初はRMIを利用していたが，ファイアウォールに起因する不具合のため，HTTP通信でXML形式のデータをやりとりする，RESTを利用することにした．RESTでは，WebアプリケーションのURLパラメータとしてリクエストを送信でき，結果はXML形式で返されるので，直接Webブラウザで見ることができる．リクエストの構築は文字列処理で済み，結果の処理はAjaxブームにより一般的になったXMLプログラミングで済む．さらに，サーバやクライアントの環境を特定しないプラットフォーム非依存性をもたらした．

JAMFによりJavaアプレットとして実装された農業モデルを，RESTアプリケーションとしてJavaサーブレットに変換するためのフレームワークJAMF-S（JAMF for Servlet）を構築した．JAMFとJAMF-Sで異なるところは，気象データの取得やモデルの計算を行うのがクライアントからサーバに移ったことと，ユーザインタフェースの構築をSwingコンポーネントからHTMLコンポーネントを利用するようになったことである．これらの変更のために，サーブレット用のモデル実行エンジン，インタフェース用のJSP，グラフ画像生成用サーブレットなどを新たに開発した．モデル実行エンジンから呼び出される，農業モデルの計算やデータ，気象ジェネレータに対する変更は必要なかった．

Javaサーブレット化に合わせ，地図インタフェースにGoogleマップを利用し，Ajaxアプリケーション化した．インタフェース部分のJavaScriptによる開発はJavaに比べて煩雑であったが，Google Web Toolkitの登場により，すべての開発が一貫してJavaのみで行えるようになり，開発，保守が効率化された．

JAMFの有効性を示すために実アプリケーションとして，「SIMRIWを利用した水稲栽培可能性予測支援ツール」を構築した．これは全球の気象データを利用して様々な条件で生育予測を行い，地点ごとに水稲の栽培が可能かを判定するツールである．大

量の繰り返し計算を必要としたために，モデル実行エンジンに対して，マルチスレッド化や入力データの再利用のための改良が必要であったが，オブジェクト指向で構築されたJAMFの特徴を生かし，最小限のプログラミングで対応できた．

以上のことより，多様な農業モデルをAMADISの構成要素であるWebアプリケーションとして実装し，開発効率，保守性，拡張性に優れたプログラムを開発できる基盤的技術がJAMFとして確立されたことが示された．また，複数の農業モデルを柔軟に連携させて，より詳細な機能を持つ農業モデルを構築できるという，分散協調型として提案した農業用の意思決定支援システムAMADISの正当性が検証された．

which was common to every model. Implementing approximately twenty plant growth prediction models and disease and pest damage forecasting models using JAMF shows that it can be used to build various agricultural model Web applications. If it is an agricultural model with a maintained document and does not require complex user interfaces, Web applications that can be components of AMADIS can be built within a day or two.

To become a component of AMADIS, message exchange functionality between components via a network is required. Although we used RMI at first, we decided to use REST to exchange data in XML form using HTTP because of a defect caused by a firewall. For REST, requests can be sent as a URL parameter of a Web application and the result can be received in XML form, so users can see it on a Web browser. Request building needs only strings process, and the result process needs only XML programming that has become more popular thanks to the boom in the use of Ajax. And it also provides platform-independence which means not specifying a server/client environment.

The framework JAMF-S (JAMF for Servlet) for converting the agricultural model implemented by JAMF as a Java applet to a Java servlet as a REST application has been built. JAMF-S differs from JAMF in terms of moving the process of obtaining meteorological data and model calculation from the client to the server, and using HTML components from a Swing component to build user interfaces. Those changes meant that the model execution engine for servlets, JSP for interfaces, and servlets for graphic image generation needed to be newly developed. Agricultural model calculation and data, and weather generator which are called from the model execution engine, did not need to be changed.

When we changed them to Java Servlet, we redeveloped them as Ajax application that uses Google Maps for the map interface. Although, the development with JavaScript for the interface part was more complex than Java, by emerging the Google Web Toolkit, the whole development could be carried out consistently only with Java, and this made the development and maintenance efficient.

We built a simulator for cultivation possiblity of rice using SIMRIW as an actual application to show the effectiveness of JAMF. It is a tool that predicts plant growth with various criteria using global meteorological data and examines whether rice cultivation is possible or not at each site. Although it required a massive repetitive calculation and the model execution engine should have been improved for multi-threading and reuses of reading data, we were able to accommodate the modification with the minimum amount of programming by exploiting the characteristics of JAMF built with an object-oriented system.

For all of the above reasons, we showed that a basic technique that can develop an excellent program in the development efficiency, maintainability, and the extendibility to implement an agricultural model as Web application that was the component of AMADIS has been build as JAMF. Moreover, the validity of decision support system AMADIS for the agriculture proposed as a distributed cooperative system, which can construct agricultural model with more detailed function by making multiple agricultural models cooperate flexibly, was verified.