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

農業シミュレーションモデルにおける分散協調システムのためのフレームワークに関する研究

田中　慶

目　次

Ⅰ．緒言

1. はじめに

本論文では，食糧不安や気候変動の影響の予測，農作業の意思決定支援，圃場試験の代替などのために，農業分野での有用なシミュレーションモデルやデータベースを組み合わせて利用できるシステムを提案し，その実現のために行った研究について述べる．また，研究成果を利用して農業用の意思決定支援システムを構築することにより行った評価について述べる．

〈I〉では，研究の背景である農業シミュレーションモデル（以降，農業モデルと記す）の歴史，農業モデルのプログラム開発，拡張，保守における問題点について述べる．また，これらの問題を解決するための農業モデル・データベース分散協調システムを提案し，その構成要素を実現するための研究方針を述べる．

〈II〉では，緒言で提案した農業モデル・データベース分散協調システム（AMADIS）の構成と主要な構成要素の概要について述べる．また，本論文での研究対象範囲を示す．

平成 25 年 3 月 11 日受付　平成 25 年 8 月 6 日受理

* 農研機構 中央農業総合研究センター 情報利用研究領域

〈III〉の前半では，農業モデル用フレームワークを構築するために，過去に開発された主要な農業モデルの開発言語やモジュール構造に関する研究について述べる．後半では，前半での研究をもとに，AMADISの構成要素として農業モデルをWebアプリケーションとして実装するために構築した，農業モデル用フレームワークJAMFの機能と特徴について述べる．最後に，JAMFを利用して農業モデルのプログラムを実装したときの効果を示す．

〈IV〉の前半では，AMADISの各要素を連携させるためのメッセージ交換手法に関する研究について述べる．後半では，農業モデルをAjaxアプリケーションやマッシュアップアプリケーションとして実装するための技術について述べる．最後に，実アプリケーションを構築することにより，〈III〉，〈IV〉の成果が開発効率，拡張性，保守性の面で優れていることを示す．また，ネットワーク上に分散している複数の農業モデルを連携させて，農業用の意思決定支援システムを構築できることを示す．

〈V〉では，各章の考察を行い，AMADISがネットワーク上に分散した農業モデルやデータベース等の資源を連携させて有効活用でき，AMADISの主要な構成要素である農業モデルをJAMFによりWebアプリケーションとして実装できることによって，本論文の提案と研究の有用性を示す．また，残された課題や今後の展望について検討し，本論文のまとめを行う．

本論文の題目での「フレームワーク」は，分散協調システムにおいて様々な農業モデルとデータをネットワーク経由でつなぎ，有機的に機能させるための構造のことを意味し，〈III〉での「フレームワーク」は，農業モデルをWebアプリケーションとして構築するためのプログラムの枠組みのことを意味している．

2. 背景

1) 食糧事情

約1万年前の農耕と牧畜が開始される以前，人類は天然の果実や野草の採取と動物の狩猟のみから食糧を得ていたため，地球全体で1000万人の命しか維持できなかった．ところが2005年には人類は65億人にまでに増加している．これは人類の99.9%が，動植物の遺伝や性質の知識，化学肥料，農薬，農業機器，灌漑，保存料，冷蔵技術などの農耕技術とその基礎となっている科学知識によって生き延びていることを意味している．そして多くの農業技術は20世紀に発達した[219]．

しかし，途上国を中心に満足な食糧を得られない人々が未だに多く存在している．さらに近年の，異常気象による洪水や干ばつ，新興国での食糧需要の増加や食生活の欧米化，燃料高騰や温暖化対策のためのバイオエタノール原料としの需要増などの新しい原因により，食糧不安はますます高まっている[50]．

このような状況の中，一朝一夕に食糧生産を増大させることは難しいが，世界中には耕作適地でありながら耕作が行われていないところ，最適な栽培品種の選択の行われていないところが存在する．国内では第3期科学技術基本計画における国家基幹技術の一つとして，データ統合・解析システム（Data Integration and Analysis System; DIAS）[42]において，農業モデルを用いて農作物の栽培適地の探索を行っている．農作物の栽培可能性を農業モデルにより予測できるようになれば，2008年に世界を揺るがせたような食糧高騰による社会不安を未然に回避できるようになり，発展途上国の農業を発展させ，貧困対策としても大いに役立つことが期待できる．

2) 気候変動

地球の平均気温は20世紀の間に0.74℃上昇し，後半になるに従って上昇が加速している[104]．気候変動の影響は，世界各国の都市部や山間部など地域毎に固有な形で現れることが予想される．そのため，各国が連携して様々な地球観測データや予測データを蓄積し，これらのデータを統融合して人類に有益な情報に変換することをめざすGlobal Earth Observation System of Systems（GEOSS）の10年実施計画が進められている[62]．日本ではDIASがGEOSSの推進母体として活動している．GEOSSでは国際的に共通な利用ニーズとして9項目の公共的利益分野が設定され，その1項目が農業分野である．

気候変動（Climate Change）は植物季節（Plant Phenology）に影響を与え，適切な栽培期間が変動して農作業時期が移動したり，最適な栽培品種が変化したりする．水稲栽培では登熟期が高温期と重ならないように栽培する必要があるため，最適

出穂期は関東より西では遅くなり，北では早くなる[82]．また，温暖化により発育速度が高まり，生育期間が短縮すると，減収や品質低下につながる．果樹栽培では冬期の低温が不十分だと自発休眠覚醒（Endodormancy）が不十分となり，開花時期がずれて受粉樹の開花と揃わなくなる．ひどい場合には開花しないこともある[234]．

このような将来の気候変動を想定した研究を行うにあたり，すべての実験を人工気象室や，実験条件の気候に似た別の地域で行うわけにはいかず，農業モデルを利用したシミュレーションによる代替試験の果たす役割は大きい．文部科学省による「気候変動適応戦略イニシアチブ」気候変動適応研究推進プログラム（2010～）では，気候変動に対応した農業モデルの開発が進められている．

3）国内農業問題

国内の農業現場は，農産物の輸入圧力や，生産者の担い手不足といった問題を抱えている．日本は海外の農業国から農産物に対する輸入障壁の解放を求められており，工業製品輸出の競争力を維持するためにも，自由貿易協定（Free Trade Agreement; FTA）や環太平洋戦略的経済連携協定（Trans-Pacific Partnership; TPP）参加への議論が高まっている．このような中，品質や価格で競争力を持つ農産物を生産するための最適農業を実現するために，農作業の意思決定支援へ農業モデルなどの情報技術の利用が進められている．

また，農業者が大幅に減少し，高齢化が進んでいるという問題に対し，新規農業生産者の参入，育成が喫緊の課題となっている．農業生産は気象や土壌などの影響を強く受けるとともに，家族経営が主であり，個々の農家の経験や知識に頼る作業工程が多く存在するため，農作業の標準化が困難である．新規農業者が営農を始めるためには，篤農家の暗黙知を可視化し，他の農業者等に継承する仕組みが必要である．篤農家の暗黙知は一度失われると復活は難しいため，篤農家の高齢化を考慮すると，残されている時間は少ない．暗黙知は意思決定から農業機械の操作まで広範な知識を含んでいるため，それらを可視化するためには農作業内容，環境情報，生体情報の計測，モデル化，データベース化，データマイニングなどを行うためのソフトウェアとハードウェア双方の情報技術が欠かせない．農林水産省による委託プロジェクト「農作業の軽労化に向けた農業自動化・アシストシステムの開発」（2010～）では，そのための研究が進められている．

3. 農業シミュレーションモデル

1950年代にシステム理論（System Theory）を工学分野に応用していたMassachusetts工科大学（MIT）は，インダストリアルダイナミクス（Industrial Dynamics）[53]で，計算機シミュレーションによる企業の経済活動をシミュレーションした．さらに，ローマクラブ（Club of Rome）から委託されて作成された報告書「成長の限界」[152]では，地球全体の人間の行動様式を求める世界モデルを，ワールドダイナミクス（World Dynamics）[54]を用いてシミュレーションを行い，適切な政策が行なわれずに人口と資本の成長が続けば，破局を避けられないという警告を出した．これらの研究を総称してシステムダイナミクス（System Dynamics）と呼ばれている．

システムダイナミクスの成果を受け，農業分野でもモデルによるシミュレーションを行う動きが起こった．1970年代始めに有用な農業シミュレーション用プログラムライブラリ群が登場すると，研究者が農業モデルのプログラムを開発するようになった．1980年代に意思決定支援（Decision Support System; DSS）用として開発された農業モデルの中で成功したものはなかったが，作物生産における生理学的プロセスのモデル化技術は進歩した．その後，以下で紹介する開発グループの活躍により，現在では農業モデルが農業研究のツールとして受け入れられている[221]．

モデル化の対象は作物，病害虫，農業気象と幅広く，相互の影響を考慮できる統合モデルも存在する．モデル化の手法はミクロなモデルを積み上げるシステム的（メカニスティック）モデルや，観測データから関係性を推定する統計モデルなどがあり，モデルの規模は様々である．モデル開発体制は，世界的なネットワークを構成して，世界中の研究者や先進農家が利用できるモデルを開発するグループもあれば，県の農業試験場で県内の普及員や農家を対象としたモデルを開発するグループもある．これらのモデル情報を収集し農業モデルのデータベースとして

まとめる試みも行われた[203,122].

本節では，主に作物モデル開発の歴史について述べた後，作物モデルと関連性の高い病害虫モデルや気象モデルについて述べる.

1）作物モデル

初期の作物モデル（Crop Model）の目的は，植物成長を科学的知見から説明することにあったが，次第に成長予測機能を持った意思決定支援用アプリケーションが開発されるようになった．また，時間とコストのかかる圃場試験の効率化のために，一部を作物モデルによるシミュレーションに置き換えるために利用されている[115].

作物モデルの主な開発グループとして，オランダのWageningenグループ，アメリカのIBSNATグループ，オーストラリアのAPSRUグループの3つが挙げられる．WageningenグループとIBSNATグループは1991年に協力関係を結び，1994年にInternational Consortium for Agricultural Systems Applications（ICASA）[25]が設立された．1993年に10年間のIBSNATプロジェクトが終了したIBSNATグループはICASAに引き継がれた．1995年にはAPSRUグループとも協力関係が結ばれ，その後もいくつかの地域のグループと結ばれた．図1に作物モデル開発グループの関係と，開発された作物モデルの系図を示す.

(1) Wageningenグループ

Wageningenでの作物モデル開発は，de Witと，Department of Theoretical Production Ecology of the Wageningen Agricultural University（TPE-WAU）とDLO-Research Institute for Agrobiology and Soil Fertility（AB-DLO）の彼の同僚によって始められた[39]．その後に続くモデルを含めて'School of de Wit'と呼ばれている[26].

Wageningenグループでは，作物生産システムの成長制限要因を，以下の4つの生産レベルに分類している[201].

生産レベル1：　十分な水と栄養があり（潜在成長），どの生産レベルよりも高い収量を得る．天候（日射量，気温）にのみ影響を受ける.

生産レベル2：　成長期に水ストレスの影響のみを受ける．施肥された半乾燥地域や，軽しょう土での集約栽培で起こる.

生産レベル3：　成長期に窒素ストレスを受け，加えて水ストレスを受けたり，天候不順だったりする．世界中でよく起こる状況である.

生産レベル4：　成長期にリンや他のミネラルの不足により成長が制限される．施肥されずにひどく利用された土地で起こる.

また，作物モデルの開発を以下の3段階に分類している[199].

予備段階（Preliminary Phase）：作物成長の理解や説明を行うために，モデルを定式化し，システムの状態，変数などの概念化を行う.

包括段階（Comprehensive Phase）：状態，変数の間の関係を実験から明らかにし，モデルのプロセスとして組み込んでいき，モデルのプログラミングと検証を行う.

要約段階（Summary Phase）：モデルの評価と感度分析（Sensitivity Analysis）を行った後，複雑になりすぎ，入手しづらい観測データを入力として利用するようになっていたモデルを簡素化し，現場で実利用できるようにする.

de Witは，門司・佐伯によって発展させられていた群落光合成モデル[157]を数値モデル化[38]し，作物成長の動的モデルの基礎とした．そのため，'School of de Wit'のモデルの多くは光合成駆動である．主なモデルとして以下のものがある．1960年代はモデル開発の予備段階であり，光合成モデルであるElementary Crop Growth Simulator（ELCROS）[40]が最初の動的作物成長シミュレータとして開発された.

1970年代は包括段階となり，栄養相における成長と蒸散のモデルであるBasic Crop Growth Simulator（BACROS）[41]がELCROSを発展させて開発された．BACROSからは1日の光合成，呼吸，蒸散をシミュレートするためのSimulation of Daily Photosynthesis and Transpiration（PHOTON）が派生した.

1980年代は要約段階となり，作物パラメータによって多くの作物に対応したSimple and Universal

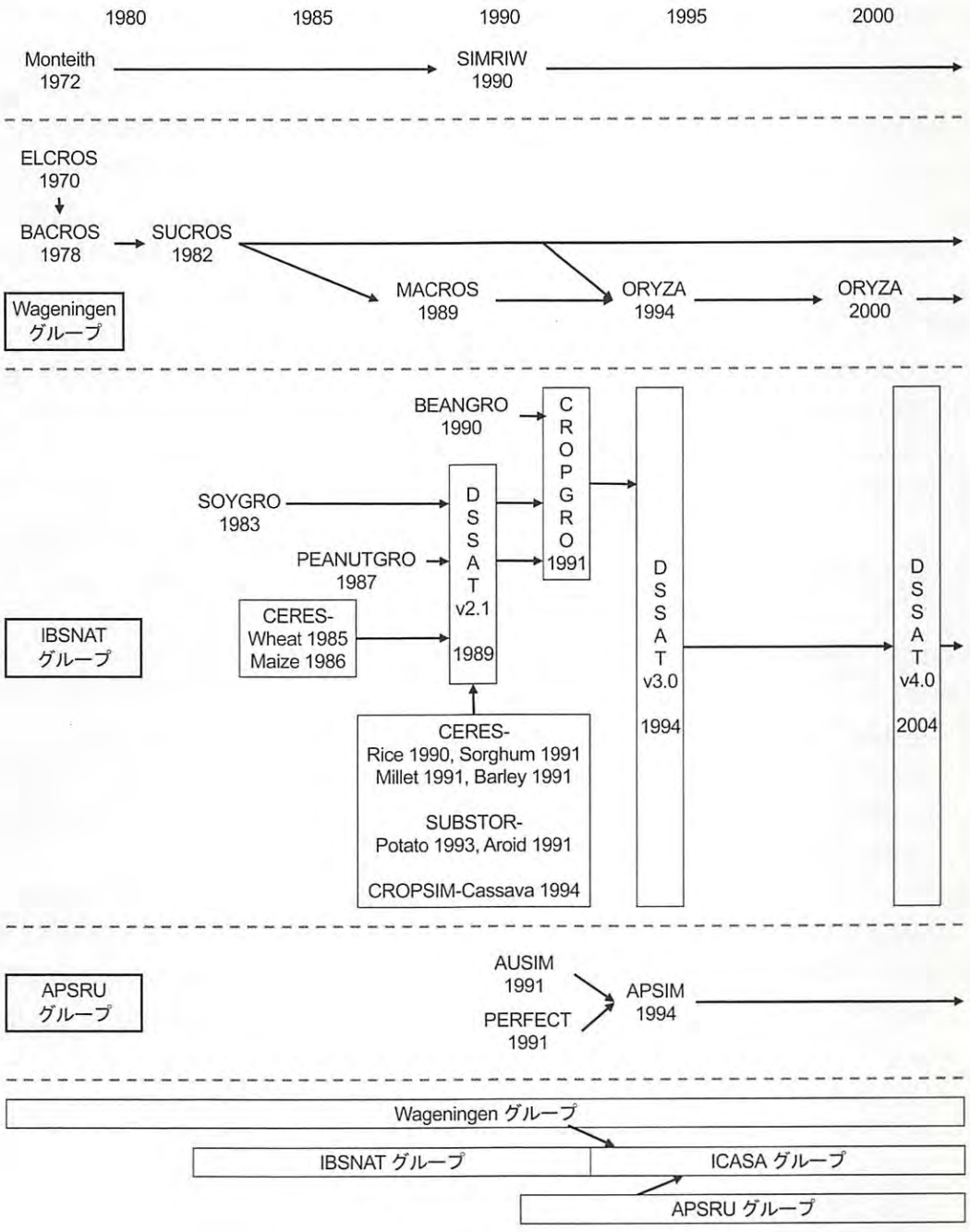

図1　作物モデルの系図と開発グループの関係

Crop growth Simulator (SUCROS)[275]が開発された．SUCROSには生産レベル1のためのSUCROS1と，生産レベル2のためのSUCROS2の2種類がある．SUCROSはSUCROS87を経て，最新版がSUCROS97[281]で，FST〈III.2.2)〉のソースコードをWageningen大学のWebサイトからダウンロードできる[286]．また，SUCROSはモデルとソフトウェア品質の改善のテストにも利用された．

WOFOST[276]はSUCROSから派生した最初の実運用指向モデルで，ユーザフレンドリなインタフェースを持ち，ヨーロッパの生産力調査において成功を収めた．Modules of an Annual Crop Simulator (MACROS)[201]はSimulation and Systems Analysis for Rice Production (SARP)プロジェクトの一部として開発され，東南アジアの研究者にシミュレーションとシステム解析の技術を移転するために利用された．

ORYZAはInternational Rice Research Institute

（IRRI）とWageningen大学により，MACROSとSUCROSをもとにして開発された熱帯低地の水稲生育モデルである．最初に開発されたORYZA1[129]は潜在収量を求めるモデルで，その後，水ストレスありの場合のORYZA_W[300]，窒素ストレスありの場合のORYZA_N[46]，ORYZA1N[4]，窒素ストレスありで低日射量の場合のORYZA_0[263]が開発された．その後，すべての種類のORYZAが統合され，ORYZA2000[27]となった．

ORYZAは1994年にリリースされて以降，ORYZA2000（2001），v2.0（2003），v2.1（2004），v2.13（2009）と改良が重ねられている．ORYZA2000はIRRIのWebページからダウンロードできる[105]．

図2はSUCROSの生産レベル1の場合を，フローダイアグラム[53,137]で示したものである．フローダイアグラム記法は，'School of de Wit'のモデルやモジュールの説明をするために多用されている．作物や各生産レベルの違いによるモデルの変更点の確認や，プログラム作成時のコンポーネント設計に役立てられている．図中のバルブ形は速度変数（Rate Variable），長方形は速度変数を積分した状態変数（State Variable），楕円形は補助変数（Auxiliary Variable），下線は駆動変数（Driving Variable），実線は物質の流れ，点線は情報の流れを意味している．

(2)　IBSNATグループ

International Benchmark Sites Network for Agrotechnology Transfer（IBSNAT）[269,270,267]プロジェクトの目的は，途上国の農業者に技術移転をすることにより，限りある資源を有効利用することにあった．技術移転に当たっては，試行錯誤による非効率的な方法でなく，意思決定支援モデルを利用する方法がとられた．そのために，世界中の研究者ネットワークが形成され，意思決定支援モデルの開発，検証，適用が行われた．IBSNATプロジェクトの主な成果として，最小データセットの定義と，DSSATの開発が挙げられる．

最小データセット〈I.4〉は，モデルの対象や精度により，観測すべき気象，土壌，作物成長，農作業管理に関するデータの種類[99]（表2）と，その観測間隔[172]（表3）を3段階で分類したものである．これにより，試験場などの観測者にとっては観測すべき最低限のデータ種別が明確になり，モデル開発者にとっては開発したモデルに必要なデータが揃わず，モデルを利用できないということを避けられる．また，観測データのデータベース化を考慮した記録書式についても定義された[289]．

Decision Support System for Agrotechnology Transfer（DSSAT）[113,115]は，CROPGRO[23,24]やCrop Environment Resource Synthesis（CERES）[216]のモデルをモジュール化して統合した意思決定支援

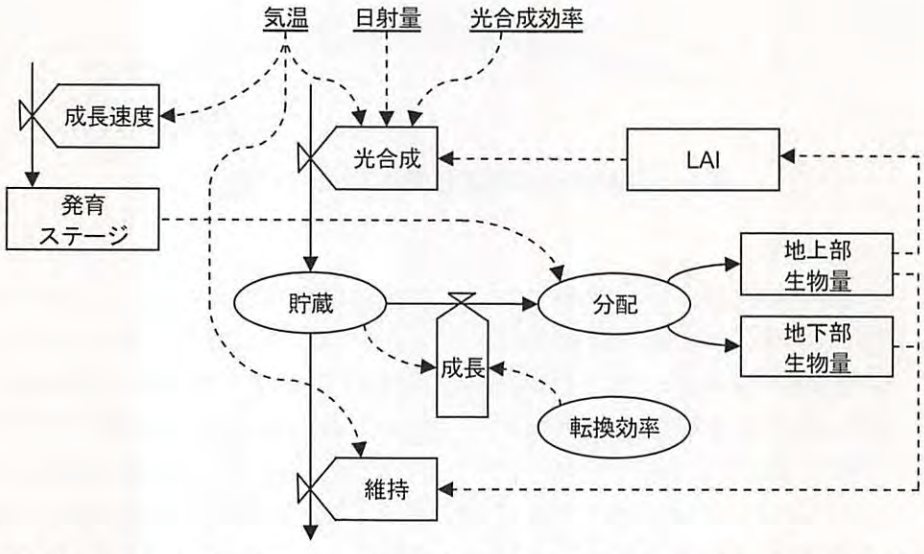

図2　'School of de Wit'の典型的な作物成長モデルのフローダイアグラム[200]

バルブ形は速度変数，長方形は状態変数，楕円形は補助変数，下線は駆動変数，実線は物質の流れ，点線は情報の流れを意味する[53,137]．

モデルである．さらに，作物モデルを活用するための，気象，土壌，遺伝子，病害虫，圃場実験を扱う，データベースや支援ソフトウェアと，4種類のシミュレーションを行うためのアプリケーションを持っている（図3）．

DSSATは1989年にv2.1がリリースされて以降，v2.5, v3.0 (1994), v3.5 (1998), v4.0 (2004), v4.5 (2008)と改良が重ねられている．その間に100ヶ国以上の研究者に配布され，食糧生産問題を検討するために利用されてきた．また，対象作物は11から27に増加している．現在はICASAによってDSSATの研修活動や更新が行われている．DSSATはICASAのWebページから購入できる[103]．

(3) APSRUグループ

Agricultural Production Systems Research Unit（APSRU）はCommonwealth Scientific and Industrial Research Organization（CSIRO）とオーストラリアのQueensland州による農業モデル開発事業により1991年に組織された．

より良い農業戦略を決定するツールのためのソフトウェア要求として，①入力データに対する高感度な作物モデル，②作物，作付け順序，混作，栽培管理，土壌生産性の傾向などの影響をシミュレーションする能力，③よく設計，テストされ，柔軟性と信頼性のあるソースコードを持つ，という3点の優先事項を設けて開発が行われた．

Agricultural Production Systems Simulator (APSIM)[150,116]は，CERES-Maize[112]をモジュール化しつつ，3点のモデル開発優先事項を満たすように開発されたAUSIM[149]と，土壌浸食の影響を評価するPERFECT[140]を統合して開発された．

APSIMは1994年にリリースされて以降，v5.3 (2007)，v6.0 (2008)，v7.0 (2009)と改良が重ねられ，研修活動も行われている．APSIMはWebページからダウンロードし，ライセンス文書を送付することで利用できる[14]．

(4) その他の作物モデル

上記の作物開発グループ[26]に属さない作物モデルの開発も行われている．

CropSyst[227,228]は1990年代初めにWashington州立大学で開発されたプロセスベースのモデルである．CropSystは気候，土壌特性，作物特性，管理が作物システムの生産性と環境へ与える影響を研究するための分析ツールとして開発されており，1日

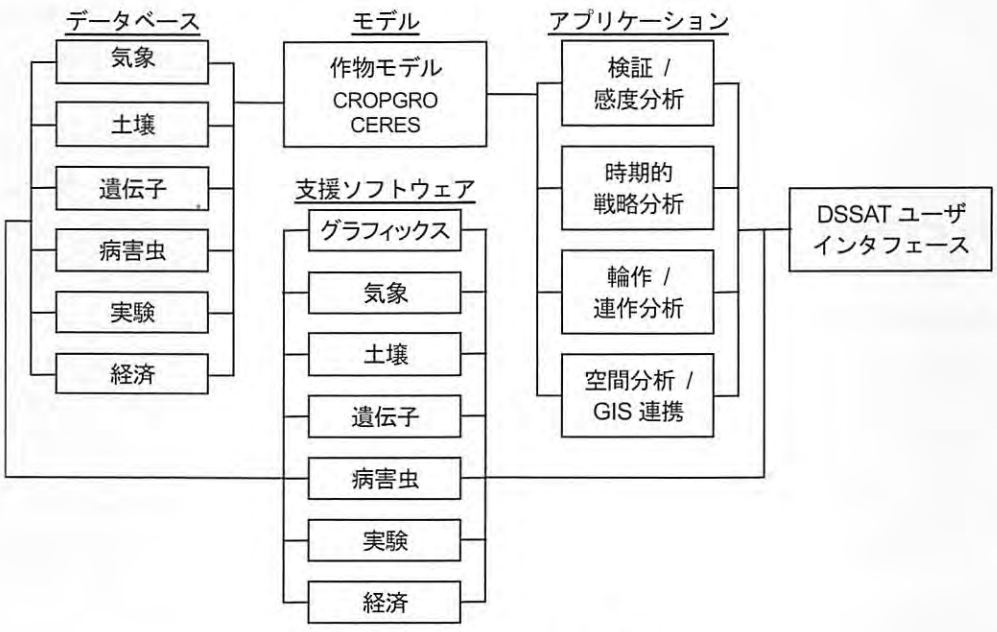

図3　DSSATの構成図[115]

モデルで利用される各種データはデータベースから取得されてモデルに入力されるか，支援ソフトウェアで加工されてから入力される．例えば気象データでは，WGENルーチンでデータの補間を行ったり，HMETルーチンで日別値から時別値を生成したりする．モデルの結果は4種類のアプリケーションにより，モデルの検証，意思決定支援に利用される．これらの操作はDSSATユーザインタフェースを通して行われる．

毎のステップ，複数年，複数作物対応のシミュレーションモデルである．DSSATより早くから輪作シミュレーションに対応し，大規模シミュレーション用ツールとして役立てられている．

CropSystはDSSATと異なり，すべての作物の成長シミュレーションに対して同じアプローチを用いている．この手法のために，作物成長プロセスの記述の単純化が行われた．このことは，作物パラメータ設定と検証を容易にし，ユーザのモデル導入の助けとなっている[33]．

CropSystは1992年にリリースされて以降，v1（1993），v2（1998），v3（2000），v4（2005）と改良が重ねられている．CropSystはWebページからダウンロードできる[290]．

(5)　国内の作物モデル

国内において1960年代以前には，気象データと豊凶考照試験データの間の相関関係を調べ，相関の高い要素を機械的に組み合わせた統計学的な生育・収量予測モデル[148]が作られた．しかし，モデル作成のために観測を行った地域や期間を超えての利用ができなかったため，栽培管理に利用することは難しかった．

1970年代には，気象と作物生産の関係を生理学的プロセスと結びつけるモデルが研究されるようになった．初期のモデル[162,231]は生育期間の全部または一定期間の気象データを入力データとする静的モデルであった．静的モデルは時間をパラメータとしたステップごとの計算を行わないため，生育途中のある時点での生育状況を判断することができないことが欠点であった．

その後，連立微分方程式による動的モデルがシステムとしての作物成長の理解に役立つことが示された[40]．このモデルは発育速度（Developmental Rate; DVR; 式1）を積分した発育指数（Developmental Index; DVI; 式2）で作物の発育ステージを決定しているため，発育速度（DVR）モデルと呼ばれている．

$$DVR = \frac{1\text{-}exp^{\{B(L-Lc)\}}}{G[1+exp^{\{-A(T-Th)\}}]} \qquad 式1$$

$$DVI(t) = \sum_{i=0}^{t} DVR_i \qquad 式2$$

DVRは気温Tと日長Lの関数である．Gは出穂までの最小日数，Thはある日長条件下で発育速度が1/2になる気温，Lcは限界日長，Aは温度係数，Bは日長係数で，品種ごとに用意されるパラメータである．DVI値の0は出芽を，1は出穂を意味している．

また，作物の物質生産を考える際に重要となるのが光合成モデル[38]であるが，国内でde Witに先駆けて，門司・佐伯により植物群落の光合成モデル[157]が研究されていた．この段階で開発された水稲生育モデル[107]やヒマワリ成長モデル[93]では植物体の乾物重の予測のみを行え，収量予測はできなかった．

Monteith[158,159]の考えに基づく，日射—乾物変換効率タイプのモデルの流れを汲み，収量予測も行える国内の代表的モデルが水稲生育予測モデルSimulation Model for Rice-Weather Relations（SIMRIW）[95,97]である．SIMRIWは発育速度の計算において，日長感応性の時期であるかと，発育相が栄養成長相（式3），生殖成長相（式4），成熟相（式5）かによってDVR計算式を使い分けて予測精度向上を図っている[94,164]．

その後，SIMRIWでは明示的に扱われていない窒素動態を取り入れたJava version of Program of *Oryza*-Nitrogen relation for Crop Growth Analysis（JAPONICA）[79,246,80]が開発された．JAPONICAは発育速度の計算において，さらに栄養成長相，生殖成長相，成熟相ごとにDVR計算式のパラメータを使い分けて予測精度向上を図っている．

栄養成長相（Vegetative Phase）：

$$DVR = \begin{cases} f_1(T) & (DVI < DVI_1^*) \\ f_1(T) \cdot g_1(L) & (DVI \geq DVI_1^*, L < Lc) \\ 0 & (DVI \geq DVI^*, L \geq Lc) \end{cases} \qquad 式3$$

生殖成長相（Reproductive Phase）：

$$DVR = \begin{cases} f_2(T) \cdot g_2(L) & (DVI \leq DVI_2^*, L < Lc) \\ 0 & (DVI \leq DVI_2^*, L \geq Lc) \\ f_2(T) & (DVI > DVI_2^*) \end{cases} \qquad 式4$$

成熟相（Maturating Phase）：

$$DVR = \begin{cases} \frac{1}{G_3}\left[1 - exp^{\{-A_3(T-Th_3)\}}\right] & (T > Tc) \\ 0 & (T \leq Tc) \end{cases} \qquad 式5$$

ただし

$$f_j(T) = \frac{1}{G_j\left[1+exp^{\{-A_j(T-Th_j)\}}\right]} \quad (j = 1,2) \qquad 式6$$

$$g_j(L) = 1\text{-}exp^{\{B_j(L-Lc_j)\}} \quad (j = 1,2) \qquad 式7$$

DVR は式1と同様に気温 T と日長 L の関数である．G_j は各発育相の最小日数，Th_j はある日長条件下で発育速度が1/2になる温度，Lc_j は限界日長，A_j は温度係数，B_j は日長係数，DVI_1* は日長に感応し始める発育指数，DVI_2* は日長感応性を失う発育指数で，品種ごとに用意されるパラメータである．SIMRIW の G_j，Th_j，Lc_j，A_j，B_j の各品種パラメータはすべての発育相で同じ値であるが，JAPONICA では異なる値を設定し，精度の向上を図っている．

DVI を求める式は式2と同じである．*DVI* 値の0は出芽（Emergence），1は幼穂分化（Panicle Initiation），2は出穂（Heading），3は成熟（Maturity）の各ステージを意味している[163]．

国内では，上記のような作物モデルが開発され，ユーザが利用できるようになっているものがいくつかあるが，小規模な開発グループによるものである．海外の開発グループのように，普及のためのトレーニングや改良のための栽培試験を，全国的なネットワークを形成して継続的に行っているような例はない[78]．

(6) 水稲生育モデル

水稲はアジアにおける主要作物である[142]ため，多くの水稲生育モデルが開発されてきた．主な水稲生育モデルを表1に示す．多くは水稲のみを対象とするモデルであるが，DSSAT に含まれる CERES-Rice や CropSyst のような複数の作物を対象とするソフトウェアでは，稲－麦－大豆といった輪作シミュレーションも行える．また，潜在成長のみをシミュレートするモデルと，水や窒素ストレスのある場合もシミュレートするモデルがある．国内のような灌漑施設が整い，十分な施肥を行える水田での稲作をシミュレートするには，研究で利用する場合を除いて，入力データやパラメータの少ない前者のモデルが適している．

2) 病害虫モデル

1975年に計算機ベースの最初の Integrated Pest Management（IPM）用の情報発信システム Pest Management Executive（PMEX）[34,293]が，Michigan 州立大学で開発された．もとはリンゴの IPM 用に開発されたが，後に様々な作物に適用された．

病害モデル（Disease Model）の実行には，入力データとして気温，降水量などの気象データと，葉の濡れ時間などの圃場データが必要である．病害は潜伏，感染，拡大の3段階で進行する[272]．病害虫モデルの開発，検証においても，IBSNAT グループによる気象データベース作成のための最小データセット〈I.4〉，気象データモジュールや，研究者ネットワークが役立てられた[264]．

病害虫は作物の成長や収量に影響を与えるので，正確で迅速な意思決定支援を行うためには，作物モデルと病害虫モデルを連携させる必要がある．連携にあたっては，病害虫モデルから作物モデルへ一方向に影響を与えるのでなく，双方向に，しかも，時間ステップごとにお互いの状態変数に影響を与えるようにするのが望ましい[265]．双方向連携により，農薬使用時期や使用回数による，減収と農薬使用コストを比較するようなシミュレーションを行うことができる．

3) 気象モデル

作物モデルの実行には気象データが必須である．規模の小さな作物モデルでは気象データベースから取得して保存したファイルや，圃場で実測した気象データのファイルから読み込んで利用することが多い．ある程度の規模の作物モデルは気象ジェネレータを通して気象データを取得している．気象ジェネレータは気象データを気象データベースから取得するだけなく，欠測値の補間，予測用の未来のデータの生成，気象データを作物モデルが定めた形式で出力するなどの機能を提供する．

また，利用頻度が低かったり，観測機器が高価であったりして，観測されていない気象要素は，気象モデル（Meteorological Model）をサブモデル的に利用して，他の気象要素から推定することがある．

(1) 気象ジェネレータ

現代においても農業ほど天候に依存する人類の活動はない．そのため，農業関連のシミュレーションにおける気象データの重要度は高い．世界気象機関

表1　主な水稲生育モデル

モデルの名称	モデルの特徴	栽培条件		
		最適条件	水不足	窒素不足
RICEMOD [151]	IRRI で開発された灌漑水田と天水田用の簡易モデル	●	●	–
CERES-Rice (DSSAT) [214]	IBSNAT プロジェクトで Michigan 州立大学が DSSAT 用に開発したモデル	●	●	●
WOFOST [276]	世界食料生産可能量の研究のために，CWFS と Wageningen 大学が開発したモデル	●	●	●
MACROS [201]	SARP プロジェクトで開発され，CSMP のモジュールとして構成されたモデル	●	●	●
RICESYS [75]	マダガスカルで Makalioka34 を対象に，人口動態モデルを適用したモデル	●		
Rice Clock [57]	揚子江流域で 4 品種を対象に開発した出穂期を求める簡易モデル	●	–	–
ORYZA [129]	SARP プロジェクトで IRRI と Wageningen 大学が開発したモデル	●	●	●
RICAM [306]	Rice Growth Calendar Simulation Model	●		
VSM [124]	Very Simple Model	●	–	–
RLRice [55]	タイの天水田（Rainfed Lowland）における KDML105 を対象に開発したモデル	●	●	
SIMRIW [95]	日本の各県で収量予測や気候変動の影響を予測するために利用されているモデル	●	–	–
JAPONICA [79]	窒素の影響を考慮したモデル	●	–	●
RICEPSM [302]	Rice Population Simulation Model	●		●
TRYM [297]	Temperature Rice Yield Model	●		
CropSyst [33]	プロセスベースの作物モデル	●	●	●
GEMRICE [98]	Genotype by Environment simulation Model for RICE 遺伝子型－環境相互作用を考慮したモデル	●		
PRYSBI [108]	Process-based Regional-scale Rice Yield Simulator with Bayesian Inference 広域収量予測モデル	●		
RiceGrow [262]	Physiological Development Time（PDT）を用いた Rice Clock の発展モデル	●	●	●

栽培条件が空白の場合は，水稲生育モデルの栽培条件が不明．

（World Meteorological Organization; WMO）による農業気象に関する指針 [299] では，気象要素の計測方法，観測データの保存方法や統計解析手法，各気象が作物生育に与える影響や気候変動による影響について述べられている．

気象ジェネレータ（Weather Generator）は作物成長モデルや病害モデルの実行に必要な気象データを生成するプログラムである．多くの気象ジェネレータでは，降水の有無をマルコフ連鎖（Markov Chain）で，降水量，最高・最低気温，日射量を，各種分布（ガンマ分布，正規分布，ベータ分布）を用いて計算している．計算式中のパラメータを求めるために，過去20～30年間の気象データを利用することが多い．

1980年代には，作物モデルのプログラムで利用できるように，気象ジェネレータのプログラム

がFORTRANで開発された[29,132]．DSSATで利用されるWeatherMan[202,296]は，WGEN[210,211]とSIMMETEO[61]を含む気象ジェネレータである．Weather System[277]はWageningenグループの作物モデル用のFORTRANライブラリとして開発された気象ジェネレータであるが，統計的に気象データを生成する機能はなく，気象データを気象データベースから読み込む機能の他に，欠測値の補間機能と日照時間を日射量に変換する機能がある．ClimGen[166]はCropSyst〈I.3.1）（4）〉で利用される気象ジェネレータである．基本的な部分はWGENと同じであるが，降水量の生成にガンマ分布よりパラメータ生成が容易なワイブル分布を利用し，気温，日射量に加えて，蒸気圧と風速も生成できる．CLIMA[43,45]はC#で開発された気象ジェネレータで，気温，蒸発散，降水量，日射量，風速，葉の濡れを生成できる．コンポーネント指向で開発されているのが特徴で，各要素の生成を行うための複数のモデルコンポーネントが用意されており，その合計は300以上ある．

気象ジェネレータの利用の妨げとなるのが，パラメータを決定するために必要な長期間の日別気象値を用意しなければならないことである．特に途上国への技術移転のために作物モデルや病害虫モデルを利用した指導を行う場合には，気象データ観測網が整備されておらず，問題となることがある．そのため，10年間程度の気象データや，日別値でなく月別値を利用する気象ジェネレータの研究も行われている．

(2)　日照時間－日射量モデル

日射量は植物の成長，地温などへ影響を与える重要な気象要素であるが，アメダス観測地点で観測されておらず，気象官署や試験場などで観測されたデータに頼っていた．作物モデルは温度時間によりステージを決定し，日射量により収量を計算することが多いため，収量計算部分がオプション扱いされていることもある．

日照時間と日射量の関係を推定する研究は古くから行われていた[7,19]．最初のモデルであるÅngström式は日射量と快晴時の日射量の比と，日照時間と可照時間の比の間の線形関数で表されており，その後のモデルの基本となっている．初期のモデルは月別値を対象としていたが，その後，日別値を対象とするモデル[174,125]や時別値を対象とするモデル[146]が研究された．

(3)　水田水温モデル

水稲生育モデルの中には，成長や収量をより正確に計算するために，気温だけでなく水田水温を利用するモデル[79,246,33]がある．また，水温は温暖化ガスであるメタンの水田からの排出にも影響している．そのため，水田水温を気温，日射量，風速，相対湿度から推定するモデル[131]が開発されていた．水田への日射は葉により遮られるため，水田水温モデルは，水稲生育モデルのLeaf Area Index（LAI）と相互作用しながら計算される．

(4)　葉の濡れモデル

葉の濡れ時間（Leaf Wetness Duration）は病害感染プロセスの重要な要素にも関わらず，農業試験場においてさえ，あまり観測されている項目でないため，病害モデルの実行可否を決める限定要因になっていた．そこで，葉の濡れを推定するDEWFOR[292]やSurface Wetness Energy Balance（SWEB）[143]などのモデルが開発された．

(5)　日長時間モデル

日長時間は光周性[58,59]のある作物の生態反応や，昆虫の休眠誘起などのシミュレーションを行う場合には必須のデータである．日長時間を緯度から計算する式は複雑ではないので，農業モデルのプログラム内に計算式を持って，求めている場合[97,246]も多い．

農業モデルで利用されるのは日長時間のみであるが，夜明，日出，日南中，日没，日暮の各時刻を求められる計算式もあり，そのBASICプログラムとともに資料としてまとめられている[118]．

(6)　グリッド

アメダス（Automated Meteorological Data Acquisition System; AMeDAS）[119]の観測地点は約850地点で，設置間隔は約21km（降水量のみなら約1300地点，約17km）あり，国内の地形の複雑さを考慮すると，十分な設置間隔とはいえない．圃場の周辺の気象観測地点を作物モデルで利用する場

合，実際の気象値と観測地点の値との差がモデルの結果に無視できない影響を与える可能性がある．特に，地形の複雑な地域ではその可能性が大きい．

国内では，アメダスデータを1kmメッシュ化するプログラム[223]が作成され，各県の農業試験場などで利用されている．このプログラムでは，実測値の地形因子を除いてから内挿することにより，推定精度の向上が図られている．

また，全球ではThe global dataset of Solar Radiation (SR) and Temperature of near surface Air (TA)（GD-DR&TR，1961～2000年）により，1度グリッド（陸地のみ）の気温，降水量，日射量を利用できるようになっている．

4. 農業モデル用データ

途上国への農業技術移転のための意思決定支援システムとして農業モデルを導入する場合，農業モデルとともに，それが利用するデータの利用可能性の検討が重要である．農業モデルの実行地点で，モデルが必要とするデータを揃えられなければモデルを実行することができない．また，データの利用を見越して不必要なデータの観測を要求すれば，人的，物的資源の無駄遣いになる．農業モデルで必要とされるデータは，IBSNATグループによって最小データセット（Minimum Data Set; MDS）として表2のように定義された[99]．また，農業モデルの分析レベルに依存して表3のように3段階に分類された[172]．

MDSではデータ観測方法についても触れられている．気温はWMOが定めた標準測定法があるが，圃場での測定に適した小規模な装置の利用が許されている．土壌データはいくつかの測定方法[213]があるが難しいため，他のデータからの推定や，表の利用が許されている．作物データの観測は土壌養分，水分，病害虫，耕作方法などの外的要因の影響を除去するのが難しいことを指摘し，測定箇所の選定にあたって圃場の中央と外側，列間，個体数などの注意点を挙げている．

5. 国内の農業分野における情報化

国内でのパソコンや携帯電話などの情報（IT）機器の進化と普及の影響は，21世紀に入ると農家レベルにも及ぶようになった．2002年末には46.4%がパソコンを所有し，29.5%がインターネットを利用していた[181]．同様に，47.5%が携帯電話を所有し，18.2%がインターネットを利用していた．このうちのパソコン所有者の4割，携帯電話所有者の3割が農業経営に利用しているか，利用意向を持っていた．

パソコンの利用内容としては，インターネットを利用しない場合は，簿記・青色申告などの経営管理（61.4%）が最も多かったのに対し，インターネットを利用する場合は，栽培技術等の生産管理情報の収集（61.5%）が最も多かった．携帯電話を利用する場合は，市況・気象等の情報収集（47.9%）が最も多かった．以上のことから，農家による意思決定において，情報機器によるインターネット経由で取得できる情報に，期待を寄せていたことをうかがうことができる．

農業情報の実利用の例としては，①センサを利用した施設栽培の自動化，Global Positioning System（GPS）や衛星画像を利用した圃場管理，畜産における個体管理，市場評価向上のためのトレーサビリティシステムなどの生産管理への利用，②インターネットや携帯電話メールを利用した気象，農業技術，市況などの情報発信への利用，③直売所のPoint of Sale System（POS）を利用した効率的な出荷，販売への利用などがある[183]．この中で農業モデルは②において，生育予測や病害虫発生予察情報の発信のために利用されている．

6. 国内の農業モデルアプリケーションの問題

国内で開発された農業モデルアプリケーションは数多くあるが[122]，開発グループを構成して開発されることは少なく，各研究室や研究者単位で開発されてきた．そのため，プログラム設計やドキュメント整備を十分に行った開発がされず，保守や再利用に支障をきたすことがある．また，開発時には最新のハードウェアやソフトウェア技術の利用を心がけていたとしても，開発終了後の保守が不十分で陳腐化してしまったレガシープログラム（Legacy Program）も多い．レガシー化の原因としては，以下のようなことが挙げられる．

- 使用したプログラム開発言語の実行環境を用意できなかったり，バージョンの違いによるプログラ

ム修正が必要であったりする．
- 気象データなどの入力データを用意できないか，入力データ用の書式への変更が困難である．
- フロッピーディスクなどのレガシー化した記憶媒体に保存しており，読み込み装置を用意できない．
- 十分なドキュメントが用意されておらず，実行方法や，入出力データの書式や内容が分からない．
- 農業モデルのパラメータに地域に関するものが含まれ，パラメータを決定するために利用したデータの観測地以外での予測精度が悪い．
- インストーラが付属せず，ファイルのコピー，設定情報の編集といったインストール作業が煩雑である．
- コンソールアプリケーションでのコマンド入力に慣れていない．
- 農業モデルの情報がインターネットで検索できる状態になっておらず，存在を知ることができない．
- 農業モデルの内容は論文等で公開されているが，モデルのプログラムまでは公開されていない．または，論文の情報のみからプログラムを開発することが難しい．

上記の問題のうち，いくつかはMS-DOS時代に開発されたプログラムをWindows世代のユーザが利用しようとしたときのコンピュータ・リテラシーの問題であり，ネットワークやソフトウェア技術を利用して解決可能なものもある．ハードウェアやデバイスに起因する問題は，時を経るにつれ，それらの入手が困難になるため深刻である．

水稲モデルの場合，水稲モデルが含まれていて，保守作業の行われているORYZAやDSSATを導入することも選択肢の1つである．しかし，現在そ

表2　最小データセット[99]

データの種類	分類	内容
実行用データ	圃場	緯度，経度，高度，年平均気温，土地形状（勾配），水はけ
	気象	日別日射量，最高最低気温，降水量
	土壌	土壌の特徴，有機炭素，有機窒素，pH，リン，カリウムの量
	初期状態	前に栽培した作物，根粒菌の量，土壌中の水分，窒素量
	管理	作物名，移植日（深さ，方法），灌漑（日付，方法，量，深さ），施肥，残渣量，農薬，耕作，収穫
校正用データ	実行用データ＋発芽日，開花日，成熟日，LAI，乾物重，病害虫被害	
評価用データ	ジャックナイフ法：6個のデータセット→5個を校正用，1個を評価用	

表3　最小データセットの対象モデルによる階層[172]

	レベル0	レベル1	レベル2
対象モデル	単純な作物モデル	成長や収量用のプロセスベースモデル	光合成，呼吸，蒸発，形態発生，養分吸収
観測間隔	週別	日別	時別または数分別
気象データ	・実測または気象ジェネレータで生成 ・降水量（圃場で観測） ・日射量，最高最低気温，蒸発能（圃場または近隣の気象観測点）	降水量，日射量，最高最低気温，蒸発能（圃場で観測）	
土壌データ	土壌水分，栄養状態，pH	播種，開花，成熟時の土壌水分と栄養状態 pH，電気伝導率	
作物データ	乾物重，収穫量，播種日，開花日，収穫日	成長ステージごとの部位別乾物重，LAI	
管理データ	耕作，施肥，灌漑，除草剤，殺虫剤（日付と量）	耕作，施肥，灌漑，除草剤，殺虫剤（日付と量）	

れらのモデルを国内で利用している例が少なく[78]，SIMRIWなどの国内で開発されたモデルやその改良版が利用されている．国内の水稲モデルは対応品種やモデル開発時の栽培環境が似ていることや，ORYZAやDSSATと比べ，モデルが単純で実行に必要な観測データの種類が少ないことが，利用されている理由として挙げられる．このことを考慮すると，国内の他のレガシー化している水稲モデルのプログラムの中には，レガシーモデル対策をして有効利用できるものがある程度存在すると思われる．

7. 研究事例

1) モジュール化

プログラムの開発，保守の効率化，有効活用の手法として，プログラムのモジュール化（Modulization），もしくはコンポーネント化（Componentization）がある．このような構造化プログラミング（Structured Programming）の手法は1970年代には存在しており，作物モデル開発グループ〈I.3.1)〉においても，モデルのモジュール化についての研究が行われていた[114]．オブジェクト指向プログラミングが登場すると，その利点の説明をしやすかったこともあり，モジュール化が再び注目を集めることになった．モジュール化はレガシーモデルの有効利用のためのキーとなる技術の1つではあるが，その分かりやすい利点にのみ注目し，失敗した事例も多い．

計算機の性能が向上するにつれ，ソフトウェアに求められる機能は複雑化し，1980年代には将来的なソフトウェア技術者の不足が不安視されていた．そのような背景のなか，1985年に通商産業省（現経済産業省）がΣプロジェクトを立ち上げた．Σプロジェクトの狙いは，全国のΣワークステーションで作られたソフトウェア部品を，通産省内に構築予定だったΣセンターのメインフレームで一元管理，供給し，ソフトウェア開発を単なるソフトウェア部品の組み合わせとすることで，高度なソフトウェア技術者不足に対応することにあった．しかし，計算機がこれ以上進化しないという誤った前提のもとで，ハードウェア指向で進行したために，5年間で成果を出すことができなかった．

その後もソフトウェアの部品化と，それを利用したプログラムの自動生成についての研究と開発は行われた．汎用的なアプリケーション開発を目指したツールでは，ツールが想定したアプリケーションの範囲内での開発はうまくいっても，その範囲を超えるアプリケーション開発をするのが難しかった．そのためには，ツールの解析を行ったうえで，機能を拡張するためのプログラム開発を行うため，一から開発した場合と手間が変わらず，汎用的アプリケーション開発のための部品化による方法は成功しなかった．

ただし，データベースの内容を表示し，ユーザ操作をデータベースに反映させるようなアプリケーションに限定すると，処理の流れが共通であるため，Model-View-Controller（MVC）デザインパターン（図4）の手法がうまく機能し，最小限のプログラミングでアプリケーション開発を行える．画面の表示を行うのがView，データやロジックを扱うのがModel，画面に対するイベントに応じて，処理を行うModelを呼び出し，更新を行うViewに通知するのがControllerである．

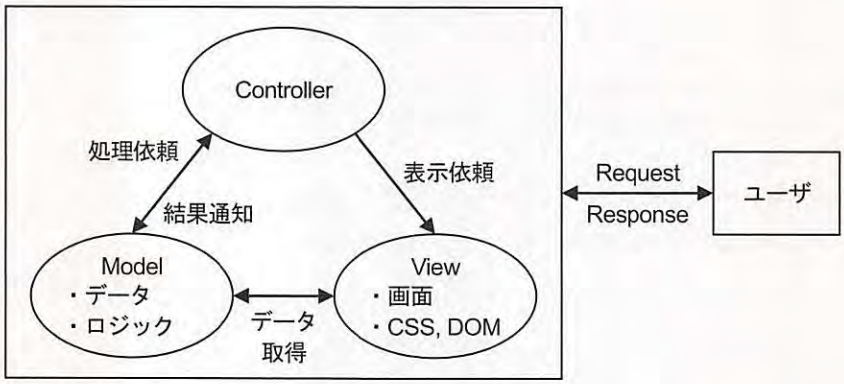

図4　MVCデザインパターン

2) Webアプリケーション

Webアプリケーションとは，Webブラウザ上で動くアプリケーションのことである．初期のWebアプリケーションは，Webサーバから送信されたHyperText Markup Language（HTML）による静的な画面を表示し，ユーザのアクションごとにページ遷移が生じていた．その後，JavaScriptを利用したAjaxの仕組みにより，画面遷移をせずに動的に生成される画面を利用さできるようになった．このような表現力の高いWebアプリケーションはリッチインターネットアプリケーション（Rich Internet Application; RIA）と呼ばれている．

インターネットの普及と，RIAの登場により，アプリケーションの実装方法の主流がスタンドアロンからWebアプリケーションへと移行した．Webアプリケーションの利点は，インストール作業を必要としなかったり，アプリケーション更新の反映が容易であったりすることである．

Webアプリケーション構築においては，MVCデザインパターンを基本としたフレームワークという，アプリケーションの半完成品を利用した開発が行われている．主なフレームワークにはJavaServer Faces（JSF），Apache Struts，Apache Tapestryなどがあり，プログラム開発を最小限に留めることにより，Webアプリケーション構築の効率化に貢献している．

3) 分散協調システム

分散協調システムは，分散（Distributed）して存在する各要素が協調的（Cooperative）に相互作用しながら，全体としての振る舞いをするシステムのことである．

農業における意思決定では，気象データ，作物データ，土壌データなどの様々なデータを，栽培品種選択，病害防除，灌漑管理などの様々な目的で共通して利用するという特徴がある．さらに，農業モデルと各種データは地域性が高く，全国の普及所，農協，大学，試験場などで開発され，分散した状態で運用されている．そのため，農業用の意思決定支援システムを，データを一元管理する集中型のシステムとして構築することは難しい．その反面，分散協調システムでは，各々をネットワークで結び，相互連携させる仕組みの構築の面での難しさはあるが，運用元で保守性を維持しつつ，相互連携により詳細で複雑な農業モデルを動的に構成可能な意思決定支援システムを構築できると考えられる．

初期の分散協調型の農業情報システムの例として，複数の分散したデータベースを協調させて利用する，MetBroker[168,171]や病害虫防除支援システム（PaDB）[126,127,169]がある．MetBrokerは〈II〉で述べる分散協調システムの一要素であるが，それ自身が複数の気象データベースを協調させるためのデータ仲介システムとなっている．PaDBは病害虫図鑑，病害虫防除指針，農薬情報の3種類のデータベースを協調させ，1つのシステム上で，病害虫同定から防除と農薬に関する情報を得ることができる．

8. 提案

海外の大学や研究機関ではDSSAT，APSIM，CropSystのような大規模な作物モデル統合システムが構築されていたのに対し，国内では個々の作物を対象とした小規模なモデルしか開発されていなかった．しかもその多くは，文献として公開されているだけか，プログラムが公開されていたとしてもレガシー化していて，実行するのが困難なことが多い．

しかし，今後は国内の作物生育予測を，海外の作物モデル統合システムに委ねればよいというわけにもいかない．なぜなら，国内の栽培地域や，国内で栽培されている品種用のパラメータが用意されているか，使いこなすための十分な情報が国内で蓄積されているか，修正や改良の要望への対応が迅速に行われるか，表示言語の選択ができるかなどの問題があるからである．

近年の情報技術の進歩は，コンポーネントを利用したプログラム開発，フレームワークを利用したアプリケーション構築，データベースアクセスを容易にする仲介ソフトウェア，インターネットの普及，Web検索技術，充実したWebコンポーネント群など，レガシー化したプログラムの再利用を可能とする多くの技術をもたらした．

そこで，モデル本体，サブモデル，気象データを含む各種データなどをコンポーネント化し，ユーザの要求に応えて柔軟に農業モデルをWebアプリケーションとして構築できるシステムを提案する．このシステムは各コンポーネントが開発元のサーバ

で運用されて，お互いにネットワークで結びつく分散協調型の構成とする．分散協調型とすることで運用面での複雑さが増すという欠点はあるが，各コンポーネントが開発元で運用されることにより，ユーザは常に保守された状態のコンポーネントを利用することができる．また，農業モデルがWebアプリケーションとして実装されることにより，モデルの改良が行われるたびにプログラムをインストールし直したり，実行前に必要なデータをダウンロードしたりしておくような煩雑な作業から解放される．また，中央集中型の構成をとると，ややもするとハードウェア指向に陥ったり，汎用的なソフトウェア部品開発に集中したりして，具体的な利用場面での成果が出せなかったという，Σプロジェクトの失敗という反面教師があることからも，本研究では分散協調型を選択した．

作物モデル統合システムでは，モデルを構成するサブモデルレベルでのコンポーネント化が行われ，モジュール構造やインタフェースが定義されている．それに対し，提案する農業モデル・データベース分散協調システム[85]では，モデル自身や各種データベースまでをコンポーネント化し，さらにネットワーク上に分散した状態で運用することになる．ここで，農業モデル・データベース分散協調システムを英語名から Agricultural Models and Databases with Distributed Cooperative System からAMADIS†（アマディス）と名付ける．

AMADISを実現するためには，①各コンポーネントの所在情報を管理するためのディレクトリベースの構築，②各コンポーネントを結びつけるためのインタフェースの定義，③コンポーネント間でネットワークを介して情報をやりとりするためのプロトコルの選定，④レガシープログラムを再利用する手法の検討，⑤作物モデルやデータベースで利用されるデータの名前の違いを解決するためのオントロジーの整備など多くの課題がある．本研究ではこれらの課題のうち，②～④に関する研究を行う．情報技術の進展は速いので，AMADISを構築するにあたっては，今後登場する優れた技術を随時取り入れることができる仕組みとなるような設計，実装を意識する必要がある．

9. 研究方針

最初に〈II〉で，本章で提案したAMADISの概念を定義し，その構成要素についてまとめる．本論文では主に，農業モデルをWebアプリケーションとして構築するために必要な構成要素についての設計と実装のための研究を行う．残りの構成要素は協力研究の成果を利用する．

次に〈III〉で，AMADISの構成要素として農業モデルをWebアプリケーションとして実装するために，農業モデル用フレームワークを構築する．フレームワークの構築に当り，海外の主要な開発グループにより開発された作物モデル統合システムの構造や実装に関する研究を行い，農業モデルを実装するために必要なコンポーネント群を設計，実装するための参考とする．また，フレームワークを利用して農業モデルを実装し，開発効率などにより評価を行う．

次に〈IV〉で，複数の作物モデルや病害虫モデル，気象モデルなどのサブモデルと連携，相互作用を可能とする仕組みを実装するために，複数のメッセージ交換手法についての研究を行う．メッセージ交換手法を〈III〉で構築した農業モデル用フレームワークに組み込む改良を行うことにより，新たな農業モデル用フレームワークを構築する．同時に，農業モデルをAjaxアプリケーションやマッシュアップアプリケーションとして実装できる機能も組み込む．さらに，〈II〉～〈IV〉で開発したコンポーネント群やフレームワークを利用して，実アプリケーションを構築することにより，本研究で構築したフレームワークが，開発効率，拡張性，保守性の面で優れていることを確認する．

最後に〈V〉で，総合考察として本研究で構築したシステム全体の評価を行い，AMADISのための農業モデル実装基盤技術が確立したことと，提案した分散協調システム型アーキテクチャの正当性を検証する．

---

† Garci Rodríguez de Montalvo による Amadis de Gaula は14世紀頃にスペイン語で書かれた騎士道物語である．Cervantes による小説 Don Quijote de La Mancha の主人公が多くの中世の騎士道物語を読んだ中で，最も強い影響を受けたものとして知られている．

Ⅱ．農業モデル・データベース分散協調システム AMADIS

1. はじめに

農業分野における情報技術研究は，農林水産技術会議予算による一般別枠研究「増殖情報ベースによる生産支援システム開発のための基盤研究」（増殖情報ベース，1997～2000年度）[175,167]や，農林水産省の委託による「農林水産研究情報デジタルコミュニティの構築」の一環としての「データベース・モデル協調システムの開発」（協調システム，2001～2005年度）[179,182]で実施されてきた．

農業分野では，予測できない天候，地域や圃場ごとに異なる土壌や栽培方法，作物や品種の違いなど，多くの条件が複雑に絡み合うため，既存の情報技術の当てはめだけではうまく機能しなかった．2つのプロジェクトでは，問題を解決するために必要な要素技術を，農業，工学，情報科学の研究者が学際的に研究，開発する体制をとり，インターネットを活用して各成果を組み合わせ，実際の現場で利用できる農業ITシステムを構築することを目的としていた．特に，近年の農業関連の課題である「食の安全と安心」，「環境に優しい持続的農業生産」，「競争力のある農業生産」，「生産から消費にいたる情報の流通」，「高齢化・新規就農対策」に力点を置いていた．

これまでの農業情報システムでは，プログラムとデータが常に対になって開発されており，これがシステムを複雑で高コストにする原因であった．プロジェクトでは，農業モデルやデータベースをコンポーネント化して分離することにより，開発（Development），運用（Operation），拡張（Extension），保守（Maintenance）の容易化を図った．さらに，それらをインターネット上で共有し，実行時に適宜組み合わせて必要な機能を実現することによりコストダウンを図った[84,85]．

図5はプロジェクトにおいて提案された農業モデル・データベース分散協調システム（AMADIS）である[170]．各種サーバ，農業モデル，データベース，観測機器などから構成され，ユーザが求める機能を実現するために，ネットワーク上に存在する分散オブジェクトが結びつけられる．AMADISは巨大なシステムであるため，本研究では主に，AMADIS実現のために必要な要素モデル構築手法と要素間連携手法を扱い，本章ではAMADISの各要素の詳細について述べる．

2. 農業モデル・データベース分散協調システムの構成要素

農業モデル・データベース分散協調システム（AMADIS）の構成要素群（図5）のうち，特に農業モデルの実装に関連する要素技術について述べる．AMADISは農業を対象としたシステムであるので，各要素で利用される技術は農業分野で利用されることを想定したカスタマイズがされているものもある．また，ユーザとして研究者，普及指導員だけでなく，農家も対象としているため，ユーザに応じた複数のユーザインタフェースを用意して，切り替えられるようにしている．

1) AMADIS サーバ

AMADISサーバ[123,92]はAMADISの中心にあって，各サーバ要素の連携，調整役として働く．また，入出力インタフェースを通してユーザと情報のやりとりも担う．AMADISサーバとディレクトリベースサーバ〈II.2.3)〉は，木浦らによって開発が行われた．

ユーザの質問に対する回答を出力する場合（図6）を例として，AMADISの働きを以下に示す．①Webブラウザの質問入力欄に記入されたユーザの質問文がAMADISサーバに送信される．②AMADISサーバは質問文を概念検索サーバに送り，自然文による検索を行う．③概念検索サーバによる，回答を得るための農業モデルやデータベースの情報がAMADISサーバへ返される．④AMADISサーバは農業モデルやデータベースの情報をディレクトリベースサーバに送り，所在情報検索を行う．⑤ディレクトリベースサーバにより農業モデルやデータベースのネットワーク上の所在情報がAMADISサーバに返される．⑥AMADISサーバは農業モデルを実行し，データベースを検索するために，所在情報を実行サーバに送り，実行サーバはネットワーク上に分散している農業モデルやデータベースを呼び出す．⑦実行サーバは呼び出した農業モデルを統合して実行するか，データベースの検索を行う．⑧⑨実行サーバによる農業モデルの実行結果や，デー

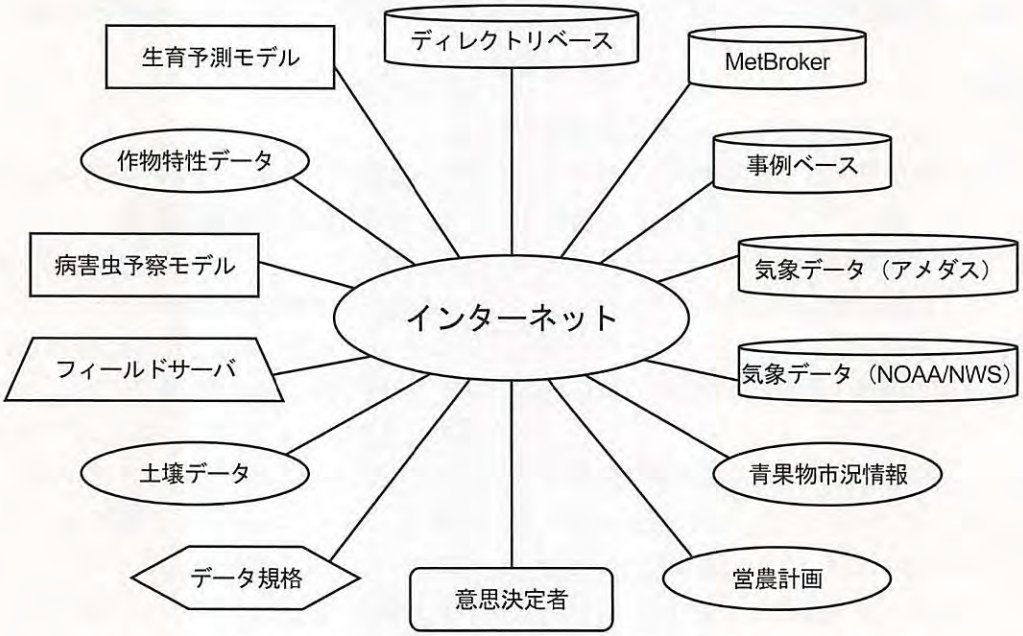

図5　農業モデル・データベース分散協調システム（AMADIS）の概念図

ディレクトリベース，各種データベース，農業モデルなどの要素群がネットワーク上に分散して存在している．AMADISでは，それらの要素が連携することによって，様々な機能を実現する．

タベースの検索結果がAMADISサーバに返され，ユーザに示される．

2）概念検索サーバ

ユーザが膨大な情報の中から知りたいものを得るために，検索システムの役割は重要である．Yahoo! [303]やGoogle [64]などの検索サイトもAMADIS構想当時から存在していたが，検索結果の候補数が多すぎたり，AMADISが対象とする農業関連以外の候補も含まれたりして，検索に慣れないユーザにとっては，そのままでは利用しづらかった．AMADISでは農業事例データの検索用に開発されていた概念検索（Concept Search）を採用した．解決したいことや，疑問に思ったことをそのまま自然文で検索できることが概念検索の特徴である[194]．

概念検索は要素情報の格納，検索に多次元ベクトル空間で表現するベクトル空間法（Vector-Space Model）[221]の一種である潜在意味解析法（Latent Semantic Indexing; LSA）[37]を用いている．格納部では要素情報を形態素解析して各品詞に分けた後，名詞の出現頻度をもとに情報を表現するベクトルを作成し，ディレクトリベースに蓄積する．検索部では問い合わせ文のベクトルと要素情報のベクトルが比較され，類似度の大きい順に検索結果候補がリストとして返される．

さらに，質問の答えに焦点を当てた要約技術[90]を応用し，概念検索で得られた複数の文書から，質問への回答とその背景を知ることができる要約文を生成する，複数文書要約手法[160]が導入された．重要文抽出には，質問に対する文節の関連度と文節間の類似度の両方を考慮して，共通箇所の検出（冗長性削除）と相違点の検出（内容の網羅）を同時に行なう事ができるMaximal Marginal Relevance（MMR）[32]を利用している．

3）ディレクトリベースサーバ

半世紀に渡り，多くの農業モデルのアプリケーションが開発されており，農業モデルのデータベースとして，ソフトウェア・データベース所在情報[122]やCAMASE[203]などが整備されてきた．これらには，モデルの名称，更新日，形態（モデル，ライブラリ，ツール，…），目的，対象，概要，変数や入出力データの詳細，開発環境，動作環境，開発者，連絡先などの情報が記録されている．

AMADISにおけるモデル検索機能としては，〈II.2.2)〉で目的に合う農業モデルを検索するための概念検索サーバについて述べたが，ネットワーク上に分散して存在するそれらの農業モデルを統合して

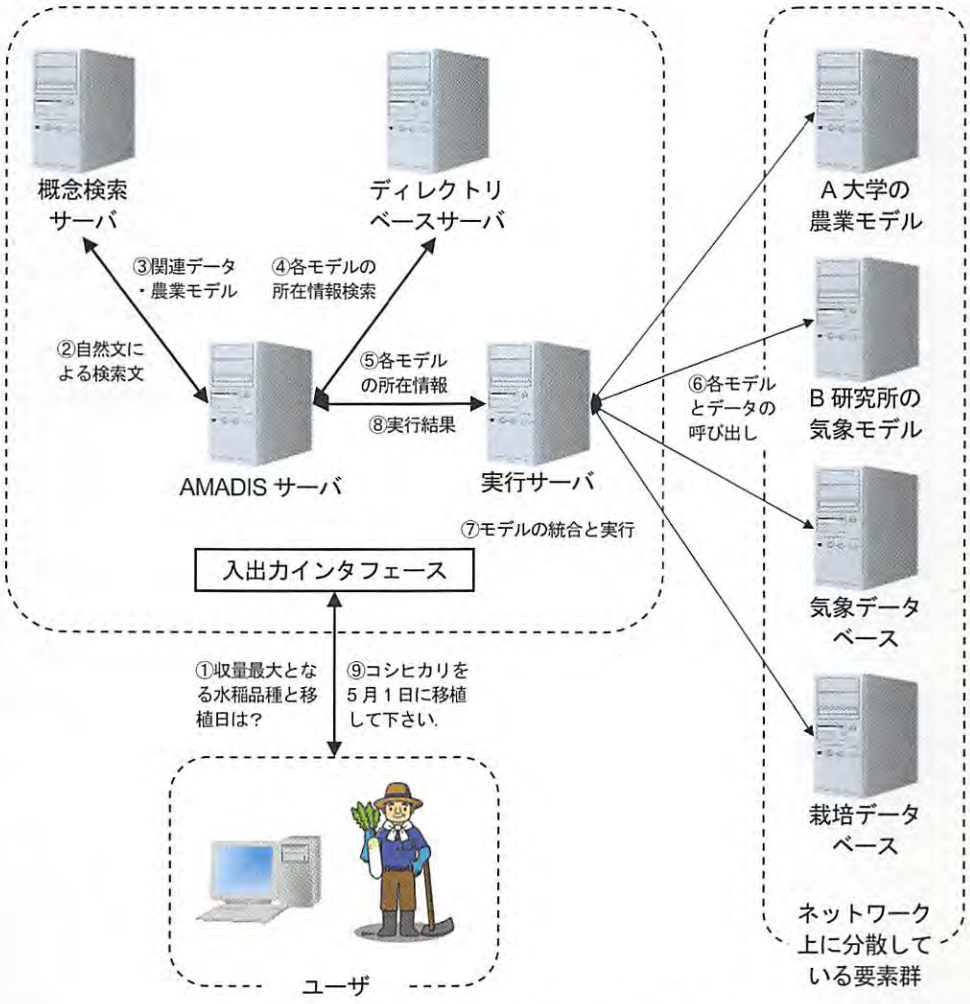

図 6　モデル検索と実行時の AMADIS の構成

ユーザが①「収量最大となる水稲品種と移植日」について質問した場合の分散協調システムの各要素の働きを示す．②③概念検索サーバによって，質問を解決するのに必要な農業モデルが検索される．AMADIS サーバは，④ディレクトリベースサーバに農業モデルや気象データベースの所在情報を問い合わせる．⑤⑥⑦所在情報に基づいて，実行サーバはネットワーク上に分散している農業モデルや気象データベースを呼び出し，それらを統合して実行する．⑧⑨結果がユーザに示される．

利用するためには，ネットワーク上の所在，プログラムの呼び出し方，実行に必要なデータやサブモデル，モデル間連携のインタフェースなどの情報が必要になる．これらの情報を管理，提供するのがディレクトリベース (Directory Base) サーバ[123] である．

4) 実行サーバ

概念検索やディレクトリベースを利用することにより，目的に合う農業モデルを見つけることができたとしても，実行に必要な動作環境を満たしているかの確認，他に必要なプログラムやデータの準備といった煩雑な作業は，ユーザが行わなければならなかった．

実行サーバでは，農業モデルを実行するために必要なデータやサブモデルの所在情報をディレクトリベースから取得して，それらを呼び出し，農業モデルと統合される．農業モデルの実行も実行サーバ内で行われるため，ユーザが農業モデルのプログラムを直接扱う必要はない．実行結果は AMADIS サーバを通して，ユーザに示される．

5) MetBroker

農業モデルにおいて気象データは必須のデータである．それゆえに各機関で多様な気象データベースが構築されている．それらのうち，いくつかはインターネット経由でアクセス可能である．しかし，気

象データベースごとにアクセス方法や出力形式が異なっているため，各農業モデルがデータベースを扱うためのプログラムを持つことは，同じようなプログラムを何度も書くことになり，開発や保守の面で非効率であった．

この問題の解決策として，ミドルウェアまたは仲介ソフトウェア[294,295]がある．仲介ソフトウェアはデータベースとアプリケーションの間にあり，アプリケーションに対して，データベースへの統一したアクセス方法と出力形式を提供することにより，データベースごとのアクセス方法や出力形式の違いを隠蔽する働きをする．

MetBroker（図7）は，ネットワーク上に分散する気象データベースとアプリケーションを結ぶ仲介ソフトウェアである[177,133,135]．データベースアクセスと，データベース出力をMetBrokerのオブジェクトに格納するドライバプログラムを開発することにより，MetBrokerと気象データベースは結ばれる．MetBrokerはLaurensonらによって開発が行われ，現時点では17の気象データベース，8.6万ヶ所の気象観測地点（表4）に対応している．

各農業モデルのプログラムがMetBrokerを通して気象データベースにアクセスするように開発されれば，各気象データベース用のプログラムをいくつも開発することから解放される．また，気象データベースのアクセス方法が変更されたり，MetBrokerから新たな気象データベースが利用できるようになったりしても，MetBroker側で対応がなされるため，アプリケーションのプログラムを修正する必要が生じない．

MetBrokerの特徴的な機能として，スペシャルアクセスとデータ集計がある．スペシャルアクセスは2点の緯度経度によって指定する矩形内に含まれるすべての気象観測地点のデータを取得する機能である．データ集計は気象観測地点が指定された観測間隔で観測を行っていなくても，より短い間隔で観測を行っていれば，自動的に集計する機能である．これらの機能は，ユーザが気象データベースや気象観測地点の位置，地点IDや観測間隔といった詳細を知らなくても，アプリケーションでの気象データ取得に支障をきたさないようにしている．

MetBrokerはJavaのインタフェースのみ提供していたが，SOAP[284]のインタフェースも提供するようになったため，様々なプログラミング言語から利用可能である．MetBrokerのWebページ[176]では，開発に必要なドキュメント[134]の参照，JARファイル（genericbroker.jar）のダウンロードができる．MetBrokerを利用した気象観測地点リストや気象データの取得の容易性を示すために，図8にサンプルプログラムを，図9にその実行結果を示す．

本研究で開発した農業モデル用フレームワークでは，MetBrokerを気象ジェネレータの主要な気象データ提供元とし，気象観測地点選択用のユーザインタフェース，時系列データ用ユーティリティなどを提供している．また，MetBrokerのパッケージには，時系列データ用クラスなどの多くの有用なクラスが含まれているので，農業モデルが扱うデータを格納するために利用している．さらに，農業モデルの実装時に，MetBrokerに不足している機能や，改善点を挙げることにより，MetBrokerの改良に貢献した．

6) フィールドサーバ

フィールドサーバ[56,87]は，Webサーバ，気温，湿度，日射量，土壌水分などのセンサ，デジタルカメラ，無線LAN通信モジュール，LED照明など様々な電子機器を搭載し，フィールド（圃場）に長期間設置して，環境の計測，動植物のモニタリング，農園の監視等を行うための超分散モニタリングデバイスである．平藤らによって開発が行われ，改良が続けられている．

フィールドサーバ開発の当初の目的は，農業モデル実行に必要な気象データを圃場で観測し，農業モデルを実行可能にしたり，実行結果の精度を上げたりすることであった．アメダス気象観測地点の設置間隔は約21kmであるため，圃場と観測地点が離れていると地形の影響を受け，予測精度に影響が生じる程度の気象値の差がある．アメダス観測網の粗さへの対策として，1kmメッシュ化[223]が行われているが，日別値などの短期間では誤差が大きい．また，アメダス観測地点では収量の予測に欠かせない日射量の観測が行われていない．これらの問題はフィールドサーバを圃場に設置し，その観測データをMetBroker経由[86]で利用できるようにすることで解決を図った．

フィールドサーバが観測した気象データは，

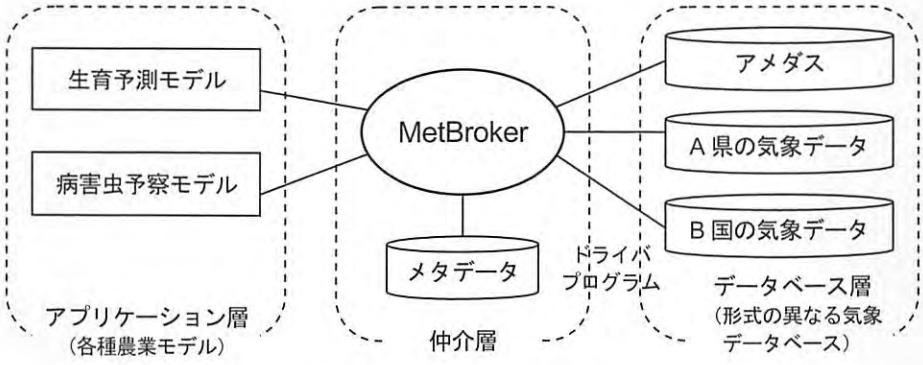

図7　MetBrokerの概念図

各種農業モデルはMetBrokerを介して気象データを取得するため，形式の異なる気象データベースを統一された手法で操作でき，同一な形式で気象データを取得できる．また，対応する気象データベースが増加したり，気象データベースの操作方法が変更されたりしても，農業モデルプログラム側では修正の必要がない．

MetBrokerと各気象データベースは，MetBroker管理者によるドライバプログラムの開発によって結びつけられる．

表4　MetBrokerが扱う気象データベースと気象要素

気象データベース	観測地点数	気温	雨量	風速	日射量	相対湿度	地温	葉の濡れ	日照時間
気象官署	154	●■	●■	●■	●■	●■			●■
アメダス [119]	2065	●■	●■	●■					●■
フィールドサーバ [87]	37	●	●	●	●	●	●	●	
北海道農業研究センター	1	●	●	●	●	●	●		●
北海道芽室町マメダス	8	●	●	●	●	●			●
神奈川県農林水産情報センター	14	●■	●■	●■	●■	●■	●■		●■
千葉県農林総合研究センター	2	●	●	●	●	●	●	●	●
和歌山県雨量現況	137		●						
NOAA/WMO [173,298]	19431	■	■	■					
WRDC (World Radiation Data Center) [301]	1002	■	■	■	■	■			
Global Dataset of DR and TR	64800	■	■		■				
ソウル大学	11	●■	●■	●■	●■	●■	●■	●■	
フィールドサーバ（タイ）	82	●	●		●	●			
フロリダ農業気象ネットワーク	29	●	●	●	●	●	●		
ジョージア農業気象ネットワーク	39	■	■						
オレゴン IPPC (Integrated Plant Protection Center)	152	■	■						
南アフリカ砂糖会	12	■	■	■	■	■			

●：時別値　■：日別値 を観測（一部の観測地点では観測していない気象要素がある）

```
import java.text.*;
import java.util.*;
import net.agmodel.physical.*;
import net.agmodel.weatherData.*;

public class MetBrokerDemo{
  public static void main(String[] args){
    MetBrokerHTTP broker = null;
    String sessionID = null;
    try{
      String host = "www.agmodel.org";
      broker = new MetBrokerHTTP(host, 80);
      Locale locale = Locale.getDefault();
      String language = locale.getLanguage();
      String country = locale.getCountry();
      String encoding = System.getProperty("file.encoding");
      sessionID = broker.getConnection("test", language, country, encoding);
      System.out.println("MetBroker に接続しました. (sessionID = " + sessionID + ")¥n");

      MetSourceDetail[] sources = broker.listMetSourceDetails(sessionID);
      System.out.println("データベースリストを表示します. ");
      for(int i=0; i<sources.length; i++)
        System.out.println(sources[i]);

      String sourceID = "amedas";
      String regionID = "08";
      WeatherStation[] stations = broker.listStations(sessionID, sourceID, regionID);
      System.out.println("¥nアメダスの気象観測地点リストを表示します. ");
      for(int i=0; i<stations.length; i++)
        System.out.println(stations[i]);

      Calendar start = new GregorianCalendar(2010, Calendar.APRIL, 1);
      Calendar end = (Calendar)start.clone();
      end.add(Calendar.MONTH, 1);
      Interval interval = new Interval(start.getTime(), end.getTime());
      MetDuration resolution = MetDuration.DAILY;
      Set<MetElement> elements = new HashSet<MetElement>();
      elements.add(MetElement.AIRTEMPERATURE);
      elements.add(MetElement.RAIN);
      String stationID = "40336";              //つくば
      boolean summarise = true;                //集計機能を利用
      boolean interpolation = false;           //補完機能を利用せず

      StationMetRequest stmr = new StationMetRequest(interval, elements, resolution,
                               sourceID, stationID, summarise, interpolation);
      System.out.println("¥nリクエストの内容" + stmr + "¥n");

      StationDataSet data = broker.supplyMetData(sessionID, stmr);

      DateFormat df = DateFormat.getDateInstance();
      String str = data.dumpDuration(df, ",¥t", "¥n", "日時","欠測");
      System.out.println(str);

      Location2D nw = new Location2D(36.8683, 139.7167);
      Location2D se = new Location2D(35.89, 140.77);
      GeographicalArea area = new GeographicalBox(nw, se);
      SpatialMetRequest spmr = new SpatialMetRequest(interval, elements, resolution,
                                       area, summarise, interpolation);
      System.out.println("リクエストの内容" + spmr + "¥n");

      SpatialMetSet sms = broker.supplyMetData(sessionID, spmr);
```

① MetBrokerと接続

② MetBrokerが扱う気象データベースリストを表示

③アメダス（茨城県）の気象観測地点リストを表示

2010/4/1～1ヶ月間

日別気温・降水量

④地点リクエストの生成

MetBrokerから気象データを取得

⑤気象データを表示（簡易な方法）

⑥エリア・リクエスト（茨城県を含む矩形）の生成

MetBrokerから気象データを取得

図8—1　MetBrokerを利用するサンプルプログラム

```
    JigsawQuantity jqTemp, jqRain;
    NumberFormat nf = new DecimalFormat("0.00");
    String devoid = "x";    //欠測値
    String delimiter = "¥t¥t";        //区切り文字
    for(int i=0; i<sms.getNumberOfStations(); i++){
      data = sms.getStationDataSet(i);
      WeatherStation station = data.getWeatherStation();
      AirTemperature temp = (AirTemperature)data.getSequence(MetElement.AIRTEMPERATURE);
      Rain rain = (Rain)data.getSequence(MetElement.RAIN);
      System.out.println("¥n"+ station.getMetSourceName() +" "+ station.getPlaceName());
      System.out.println("¥t¥t 最低気温(℃)¥t 最高気温(℃)¥t 平均気温(℃)¥t 降水量(mm)");
      if(temp != null || rain != null){
        Interval inte = new Interval(start.getTime(), Duration.ONE_DAY);
        while(interval.encompasses(inte)){
          String date = df.format(inte.getEnd());
          System.out.print(date +"¥t");
          if(temp != null){
            jqTemp = temp.getMinimum(inte);
            System.out.print((jqTemp.getCoverage() > JigsawQuantity.DEVOID ?
                    nf.format(jqTemp.getAmount()) : devoid) + delimiter);
            jqTemp = temp.getMaximum(inte);
            System.out.print((jqTemp.getCoverage() > JigsawQuantity.DEVOID ?
                    nf.format(jqTemp.getAmount()) : devoid) + delimiter);
            jqTemp = temp.getAverage(inte);
            System.out.print((jqTemp.getCoverage() > JigsawQuantity.DEVOID ?
                    nf.format(jqTemp.getAmount()) : devoid) + delimiter);
          }
          if(rain != null){
            jqRain = rain.getTotal(inte);
            System.out.print((jqRain.getCoverage() > JigsawQuantity.DEVOID ?
                    nf.format(jqRain.getAmount()) : devoid));
          }
          System.out.println("");
          inte = new Interval(inte.getEnd(), Duration.ONE_DAY);
        }
      }
    }
    catch(Exception e){e.printStackTrace();}
    finally{
      try{
        if(broker != null){
          broker.disconnect(sessionID);
          System.out.println("MetBroker との接続を切断しました.
                    (sessionID = " + sessionID + ")¥n");
        }
      }
      catch(Exception e){}
    }
  }
}
```

⑦気象データを表示

MetBroker との接続終了

図 8—2　MetBroker を利用するサンプルプログラム

MetBroker に接続後，データベースリストと気象観測地点リストを出力し，リクエストによって MetBroker から気象データを取得して出力するサンプルプログラム．

```
コンパイル方法 javac -classpath genericbroker.jar MetBrokerDemo.java
実行方法       java -classpath .;genericbroker.jar MetBrokerDemo
```

```
MetBroker に接続しました. (sessionID = MetBroker session31107)                    ①

データベースリストを表示します.
日本国内気象官署
アメダス
フィールドサーバ（時別）
NOAA/WMO
・・・                                                                            ②

アメダスの気象観測地点リストを表示します.
・・・
40242 筑波山  lat:36.2233 lon:140.1017 alt:868.0
40243 筑波山  lat:36.2233 lon:140.1017 alt:868.0
・・・
40341 土浦  lat:36.095 lon:140.2067 alt:26.0
40336 つくば  lat:36.0567 lon:140.125 alt:25.0
・・・                                                                            ③

リクエストの内容
net.agmodel.weatherData.StationMetRequest
  2010/04/01 0:00:00 - 2010/05/01 0:00:00
     elements  気温  雨量
     resolution     日別値
     Summarise true       Interpolate false
     source    amedas     station     40336

dumpDuration start 2010/04/01 end 2010/05/01                                     ④

日時,          気温,   ,       ,          雨量
,              最小,   最大,   平均,      合計
,              (℃),   (℃),   (℃),       (mm)
2010/04/02,  1.9,    20.7,   12.458,    0
2010/04/03,  8.3,    19.9,   13.512,    3.5
・・・                                                                            ⑤

リクエストの内容
net.agmodel.weatherData.SpatialMetRequest
     2010/04/01 0:00:00 - 2010/05/01 0:00:00
     elements  気温  雨量
     resolution     日別値
     Summarise true    Interpolate     false
     area NW: lat:36.87 lon:139.72 SE: lat:35.89 lon:140.77                      ⑥

・・・
アメダス 土浦
            最低気温(℃) 最高気温(℃) 平均気温(℃) 降水量(mm)
2010/04/02 3.60         20.40        12.94        0.00
2010/04/03 8.50         19.80        13.52        2.50

アメダス つくば
            最低気温(℃) 最高気温(℃) 平均気温(℃) 降水量(mm)
2010/04/02 1.90         20.70        12.46        0.00
2010/04/03 8.30         19.90        13.51        3.50
・・・                                                                            ⑦
```

図9　図8のサンプルプログラムの実行結果

図中の番号は，図8中の番号に対応している.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<?xml-stylesheet type="text/xsl" href="../fs01_data.xsl" ?>
<One_Day_Data>
<Object>
<Param_Xml>./../../../../mnt/data4/WeNARC/fs01/
                    fs01_profile.unix.xml</Param_Xml>
<Name>fs01</Name>
<IP>192.168.39.100</IP>
<Standard_Time>JST</Standard_Time>
<Today>2010/04/01</Today>

</Object>
<Data>
<Date>2010/04/01</Date>
<Time>00:01:25</Time>
<RA0>226</RA0>
<RA1>45</RA1>
<RA2>1023</RA2>
<RA3>0</RA3>
<RA5>26</RA5>
<RB4>Low</RB4>
<RB5>High</RB5>

<Air-Temp. unit="C">12.3</Air-Temp.>
<Humid. unit="%RH">100</Humid.>
<PPFD unit="umol/m^2/s">0</PPFD>
<I-Temp. unit="C">26</I-Temp.>
<LED>Low</LED>
<cam>High</cam>

</Data>
<Data>
...
</Data>
...
</One_Day_Data>
```

フィールドサーバ情報

センサの出力値

換算値

次の時刻の観測データ

図10　フィールドサーバが観測して記録した気象データ

フィールドサーバは観測した気象データをXML形式でサーバに蓄積する．

One_Day_Data要素は1つのObject要素と複数のData要素を持ち，1日分の観測データが記録される．Object要素にはフィールドサーバの情報が記録され，Data要素には1回分の観測データが，生データと，気象値に変換したデータとして記録される．

XML形式(図10)でサーバに蓄積されている．フィールドサーバのデータはMetBroker経由で取得する他，直接XML形式のデータを取得して表示（図11）[252,254]したり，アプリケーションで利用したりすることも可能である．

本研究で開発した農業モデル用フレームワークにはXML処理のためのユーティリティプログラムが含まれている．XMLデータのパーサにはDocument Object Model（DOM）やSimple API for XML（SAX）などのAPIがある．DOMは木構造を利用して直感的に操作でき，SAXはメモリ消費量が少なく，処理速度が速いという特徴がある．フィールドサーバの数百行程度の情報ファイルの操作にはDOMでも良いが，1日当り1万行を超えるフィールドサーバの1年分のデータ処理では10分程度を要するため，30秒程度で行えるSAXが適している[254]．

7) 農業モデル

農業モデルには，作物の生育速度，施肥の効果，収量予測を行うための作物モデルや，病害虫の発生を予測して防除計画に反映させるための病害虫発生予察モデルがある．また，気象データの実測値を気象データベースから取得できない場合に，それを他のデータから推定する気象モデルなどのサブモデルがある．

モデルの複雑さはパラメータや入力データの数に比例する．複雑なモデルは作物の成長を細かく再現できるが，パラメータを推定するときに利用した実験データの観測誤差を含み，新たなデータで予測を行うときに，かえって予測誤差が大きくなる場合がある[94]．また，パラメータ数が多いと，それを推定するために必要な観測データ数や計算量が多くなる．さらに，必要な入力データの種類が多くなると，入力データを用意できないためにモデルの利用が制

図 11—1　フィールドサーバ・データビューワ (254)

http://pc105.narc.affrc.go.jp/fieldserver/

（上）フィールドサーバ選択用画面
（下）フィールドサーバの情報と，データ取得設定画面

限されることがある．

農業モデルにはソフトウェアの規模に応じて，作物モデル統合モデル〈I.3.1)〉と個々の作物モデルがある．AMADIS のコンポーネントとして扱うのに適したプログラムサイズは個々の作物モデルである．作物モデル統合モデルにはソースコードやインタフェースのドキュメントが公開されているものもあるので，それらのサブモデルレベルで AMADIS のコンポーネントとして利用できる可能性がある．

作物生育予測モデルと病害虫発生予察モデルは，単独で実行されることもあるが，混作，連作，輪作や病害虫による作物生育への影響をシミュレートするために，複数のモデルが相互作用しながら実行されることもある．ただし，モデル間で相互作用させるためには，相互にデータのやりとりを行うためのインタフェースが備えられていなければならない．作物モデル統合システムは，内部に複数のモデルを持ち，データ交換のためのインタフェースも定義されている．個々の農業モデルも AMADIS 用にコンポーネント化されれば，相互作用実行に必要な機能を満たすことができる．

8）パラメータ修正

作物モデルは品種や地域ごとのパラメータを持つ

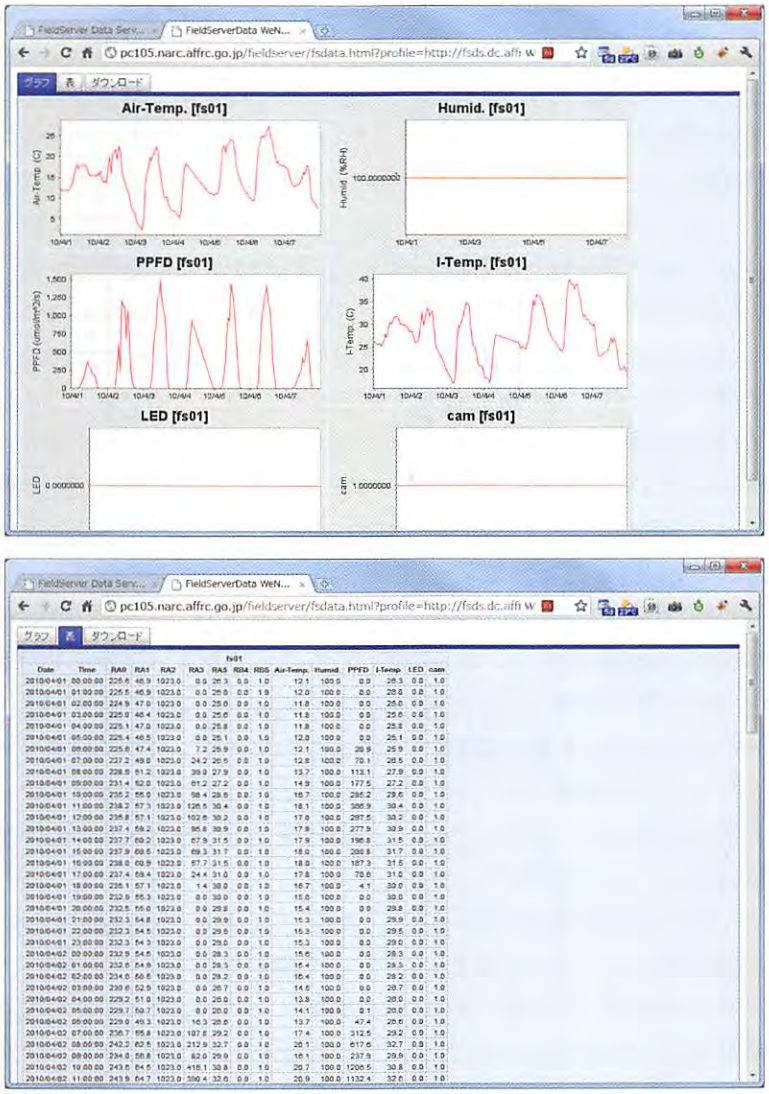

図 11—2　フィールドサーバ・データビューワ

（上）フィールドサーバによる観測データのグラフ表示画面
（下）フィールドサーバによる観測データの表による表示画面

ことにより，ユーザの栽培条件に合うように計算式をカスタマイズして精度を上げている．作物モデルを開発するために利用した品種や地域のパラメータは用意されているが，それ以外の品種，地域において作物モデルによる予測を行う場合，あらかじめパラメータ修正（Parameter Correction）を行わないと，期待通りの精度の結果を得られない．

パラメータを修正する方法にはシンプレックス法（Simplex Method）[77] や，遺伝的アルゴリズム（Genetic Algorithm; GA）[91,121] などがある．また，学習用データとテスト用データとして，気象データ，移植日，開花日，収穫日や収量などの実測データを数年分用意する必要がある．それらのデータが手元に無い場合，気象データは MetBroker から，水稲の栽培情報はイネデータベース [180,308] から近くの試験場のデータを取得できる．イネデータベースは MetBroker から気象データを取得し，栽培データと結合して提供する機能も備えている．

3.　考察

本章では，〈I.8〉で提案した農業モデル・データベース分散協調システム AMADIS の概要と，構成要素の詳細を示した．

増殖情報プロジェクトが始まる前から，多くの農業モデルのプログラムやデータベースが各地の大学や試験場で開発されていた．これらを有効活用し，

さらに各機関で開発された複数の農業モデルやデータベースを組み合わせ，より詳細な意思決定支援を行えるシステムを構築しようとした場合，すべてを集めてパッケージ化する集中型のシステムと，各機関を結ぶネットワークを利用して結びつける分散型のシステムがある．

集中型のシステムを構築する場合，中核となる機関のサーバにプログラムやデータベースを収集し，管理，運用，保守を行うことになる．それを実現するためには，各農業モデルやデータベースの研究者や開発者が開発グループのような体制をとり，永続的に維持するための中核機関のための資金や人材を確保する必要がある．しかし，期限のあるプロジェクト研究では，システム構築まではできたとしても，開発者の手を離れてサーバに集められたプログラムやデータベースを維持していくのは難しい．

分散型システムの場合，その特徴は図5に示されるようにネットワークを介して様々な構成要素が結びついていることである．分散協調システムAMADISの構成要素となる各要素がネットワーク上に分散して協調するためには，各要素間でメッセージ交換を行うために，プロトコルや書式を統一しておく必要がある．増殖情報プロジェクト研究の初期にはメッセージ交換にHORB〈IV.2.2)〉やJava RMI〈IV.2.3)〉が利用され，開発や運用面で問題が生じることがあったが，その後，通信にはHTTPを利用し，書式には処理が容易なXML形式を利用することが一般的になったため，プロトコルや書式の統一が問題になることは少ない．サーバが分散していることにより，かえって各要素による機能がサーバ単位で完結し，カプセル化の要件を自然に満たすことになっている．ただし，それぞれのサーバの運用は各要素の担当者に任されるため，サービスが休止されたり，廃止されたりする恐れがある．

AMADISの構成は図6に示されるように複雑に見えるが，ユーザからはAMADISサーバが見えるのみで，AMADISサーバの先で複数のサーバが稼働していることを意識させない．ネットワークを介して多くのサーバを利用した場合の応答時間は，10数年前ならともかく現在では考慮するほどではなくなっている．実例として，Googleによる様々な検索が多くのサーバを利用して行われているにも関わらず，ミリ秒単位で結果が返ってくることなどが挙げられる．ただし，個々の農業モデルやデータベースへの改善や修正の要望を直接開発者に届けることはできず，AMADIS管理者経由で挙げるために，対応に時間を要する場合がある．

AMADISの個々の要素については，気象データ取得のMetBrokerの担う役割は大きい．データベースアクセスのためのプログラム開発は，熟練者がまとめて行えば効率良く行えるが，個々の農業モデルのプログラム開発者には敷居が高かった．そのために農業モデルが利用する気象データを，ローカルなテキストデータから取得するように開発されてしまうと，後のプログラム公開段階でユーザに不便を強いることになり，普及の妨げになる．MetBroker経由で気象データを取得するように農業モデルを実装しさえすれば，自動的に多くの気象データベースを利用でき，新しい気象データベースへの対応のための作業はMetBrokerの管理者に委ねることができる．

フィールドサーバは，ユーザの圃場近くに気象データ観測地点がない場合に，ユーザ自身によって気象データを低コストで観測することを可能にした．観測データはネットワーク経由で自動的にサーバに蓄積され，MetBroker経由で取得できる．また，取り付けるセンサ，デジタルカメラ，通信方法や電源などを目的，設置場所やコストに応じて選択できる柔軟性がある．

AMADISの構成要素となる農業モデルは，ネットワークを通してAMADISサーバから呼び出されたり，他の農業モデルと結びつけられたりするため，Webアプリケーションである必要がある．農業モデルが必要とする入力データやパラメータ，計算の内容は様々であるが，動的モデルであれば時間ループにより繰り返し計算を行う構造は共通である．この共通部分を農業モデル用フレームワークとして構築し，農業モデルのプログラム開発，保守を効率的に行えるようにするのが本研究の目的の1つである．その具体的な内容は〈III〉で述べていく．

Ⅲ．農業モデル用フレームワーク JAMF

1. はじめに

作物モデル統合システムで利用されている，データ構造，気象データ取得方法，モジュール構造による開発について検討し，農業モデルを AMADIS のコンポーネントとして実装するための手法を提案する．

農業モデル用プログラムが開発されるようになって 40 年の間にソフトウェアの開発環境は大きく変化した．CPU の性能は 100 万倍に，プログラミング言語は手続き型言語の FORTRAN から C を経て C++ や Java などのオブジェクト指向言語へ，アプリケーション構築はメインフレーム用 OS ベースから，MS-DOS ベース，Windows ベースを経て Web ベースへと変遷し，情報の表現方法は文字から画像主体へ，操作方法はキーボードによるコマンド入力を行う CUI 操作からマウスによる GUI 上での操作へ，実行形態はユーザサイドでのスタンドアロン実行からネットワークを利用したサーバサイドへ，データ表現方法は固定長フィールドや Comma-Separated Values（CSV）などのテキスト形式から Extensible Markup Language（XML）[282] 形式へと変化した．

近年，Web アプリケーションが主流となると，様々な Web アプリケーション構築用のフレームワークが登場した．フレームワーク [266] とは，アプリケーションを構築するときに，それに当てはめていけば出来上がっていくような枠組みのことである．フレームワークを利用したアプリケーション構築は，プログラミング経験の浅いモデル開発者にとっては，サンプルプログラムやドキュメントを参考にアプリケーション開発が行えることが期待できる．また，プログラミング開発経験の長い開発者にとっても，開発作業を削減してくれる有用なものである．

農業モデルの実行では，期間や地点の設定，気象データの取得，モデルの計算，結果の表示といった，共通した処理が多い．そのため，コンポーネント用モデルをいくつも実装するためには，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を実装するのが効果的である．また，情報技術の進歩は非常に速く，一度構築した農業モデルアプリケーションも数年でレガシー化することがある．そのため，農業モデルの中心となる計算部分はフレームワークと分離可能な構成とし，フレームワーク構築に利用した技術が古くなったときには，新しいフレームワークに乗り換えることで，最新の技術を利用できるようにする．

本章では，最初に農業モデル用フレームワークを構築するための開発言語について検討する．続いて 3 つの作物モデル開発グループによるモジュール構造による作物モデル実装手法を比較しながら，農業モデル用フレームワークが備えるべき機能を抽出する．次に農業モデル用フレームワークの全体像と，各機能の詳細を示す．最後に農業モデル用フレームワークを利用したサンプルアプリケーションの構築を行い，それを利用したことによる効果を示す．

2. 開発言語

新規にアプリケーションを構築するにあたり，開発言語の選択は重要である．過去のプログラム資産を全く利用しないのであれば，アプリケーション構築に適した，今後もしばらく主流であり続けるプログラミング言語を選択すればよい．過去のプログラムを利用する場合でも，大規模な変更や移植作業を伴わなければ，ラッパープログラム（Wrapper Program）などで対処 [307] でき，開発言語選択の自由度は高い．

開発言語の選択における基準として，以下のことを考慮した．

- 開発，拡張，保守のコストが高くない（可読性が高く，修正，変更が容易）．
- プラットフォーム非依存である．
- Web アプリケーションを構築できる（ネットワークの利用が容易）．
- 開発環境の構築が容易である（フリーの開発ツール）．
- 主流である（更新頻度が高い．ユーザが多く，参考情報を得やすい）．

以上の条件を満たす開発言語選択のために，主要なプログラミング言語と，既存の作物モデルの開発

言語について述べた後，AMADISを実装するための開発言語を決定する．

1940年代に登場した計算機のプログラミングでは機械語やアセンブリ言語が使われていた．1950年代後半にはFORTRAN，LISP，COBOLといった高級言語が登場し，その後のほとんどのプログラミング言語は，これらから派生したものである（図12）．

FORTRAN，ALGOL，PL/Iが登場すると，自然界における複雑な非線形システム解析への利用が提案され，1960年代にシミュレーション言語が登場すると，動的な作物モデル実装に利用された．

1980年代には作物モデル実装用のFORTRANのプログラミング環境やユーティリティが，作物モデル開発グループにより開発され，FORTRANで作物モデルが実装されるようになった．その後，BASICやCのような主要なプログラミング言語が登場したが，FORTRANで書かれたプログラム資産を継承するために，ユーザインタフェース開発などへの利用にとどめられ，モデルの主要部分の開発ではFORTRANが使い続けられた．

1990年代にオブジェクト指向によるプログラミングの有効性が認められると，C++やJavaなどのオブジェクト指向言語での作物モデル実装や，FORTRANプログラムからの移植が行われるようになった．

2000年代はネットワークを利用するアプリケーション構築が普及し，スタンドアロンのアプリケーションから，Webブラウザをユーザインタフェースとして利用するWebアプリケーションが主流となった．Webアプリケーションでは，プレゼンテーション層（ユーザインタフェース），ビジネスロジック層（アプリケーションサーバ），データ層（データベース）の3つの処理区分に分けて開発する3層モデル（Three-Tier Model）が利用されている．プレゼンテーション層の開発には，Rich Internet

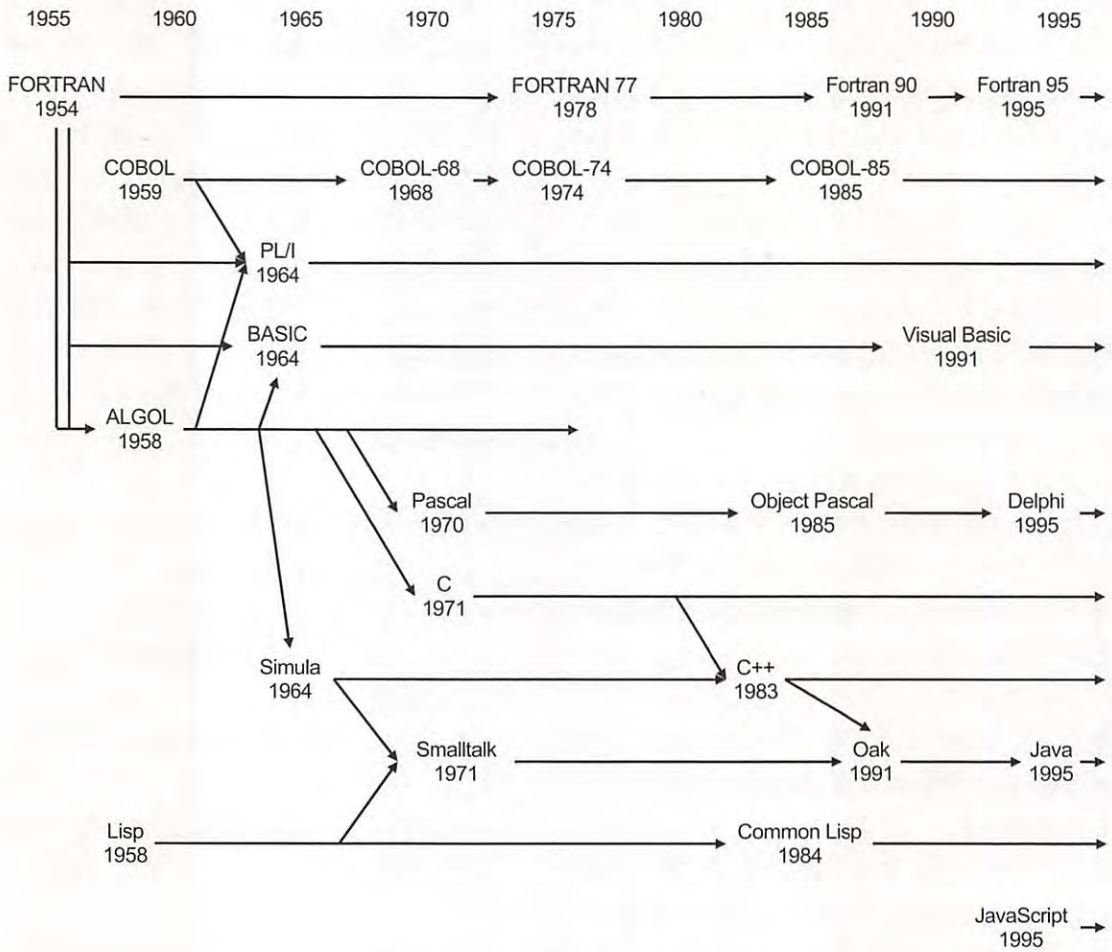

図12　プログラミング言語の系統図

Application（RIA）と呼ばれる，動的なWebページを記述できる．JavaScript，Dynamic HTML（DHTML; JavaScript + CSS），Ajax（Asynchronous JavaScript + XML），JavaServer Pages（JSP），ActionScript（Flash）[2] などが利用されている．ビジネスロジック層ではJavaやC++，データ層ではStructured Query Language（SQL）が利用されている．

1）シミュレーション言語

シミュレーション言語[30]はシミュレーション専用のプログラミング言語のことで，シミュレーションで使用する各種コンポーネントが提供されているため，プログラミングよりも生物プロセスの解析に集中できる．数学や計算機の専門的知識のない者にとって，シミュレーション言語はシンプルで学習しやすく，動的モデル実装が容易であるという利点があった[28]．

シミュレーション言語には，製造ライン，物流，通信ネットワークなどを対象とした待ち行列型モデルの離散型（Discrete-Event）シミュレーション言語と，化学プロセス，生態系の予測，流体解析，熱伝導解析などを対象とした連立微分方程式モデルの連続型（Continuous）シミュレーション言語がある．

作物モデルを実装するには連続型シミュレーション言語が適している．システムダイナミクス〈III.3〉で作成されたモデルの論理演算をするために開発されたDynamic Model（DYNAMO）[212]は，水稲生育モデル[107]で利用されている．また，Continuous System Modeling Program III（CSMP III）[102]はde Witらによる SUCROSなどの初期のモデル[114]やヒマワリ生育モデル[93]で利用されている．

CSMPは連続プロセスのシミュレーションを容易にする問題向き言語である．構文はFORTRANに似ており，FORTRANでサブルーチンを書くこともできる．CSMPではプログラム文が自動的に計算順序通りにソートされるので，実行順に記述する必要がない．そのため，プロセスやシステムについて最も理解しやすく，読みやすい順序での記述が可能である[17,138]．図13はCSMPによる単純な乾物重生産シミュレーションプログラムの例である．

2）FSE / FST

Wageningenグループでは作物モデルプログラム開発のための環境構築が進められた．1990年には，ファイルや画面の入出力，文字列操作，日付時刻処理などのための100以上のサブルーチンを集めた，FORTRANユーティリティライブラリであるTTUTIL[280]が開発された．TTUTILはプラットフォーム非依存であるので，異なるハードウェア間や，OS間での可搬性を保証していた．

1991年には，農業プロセスのシミュレーションのためのプログラミング環境をFORTRANのモジュール構造としてまとめた，FORTRAN Simulation Environment（FSE）が開発された[279]．FSEではモデル特有の科学的プロセスと，そうでない入出力などの部分が分離され，モデル開発者が前者の開発に集中できるようになっている．後者ではTTUTILのサブルーチンを多用しており，前者とはFSEドライバによって結ばれる．ORYZAシリーズはFSEで開発されている．

1990年代にはFORTRANコンパイラは利用しやすく，求めやすくなっていたため，多くの科学コミュニティがCSMPからFORTRANへと移行していた．CSMPの実行環境自体もWageningen大学の計算機センターのいくつかの計算機でのみ維持されている状態であった．そのため，1994年にCSMPのプログラムをモジュール構造のFSE-FORTRANに変換できる[139] FORTRAN Simulation Translator（FST）が開発された[278,208]．FSTはシミュレーション言語でもある．FSTは変換対象のソースコードをFORTRAN 77によるソースコードMODEL.FORに変換し，ライブラリをリンクする（図14）．気象データはデータベースから取得され，プログラム内にハードコーディングされる．

FST，FSEは2001年にWageningen大学によるプロジェクトでWindows版が開発され，現在でも利用されている．FSTWin，FSEWinはWebページからダウンロードできる[288]が，実行するには別途Visual Fortranコンパイラを用意する必要がある．

3）FORTRAN

FORTRAN[106,271]は1954年に開発された最初の高級言語であり，主に数値計算用に利用されてきた．

```
TITLE DRY MATTER PRODUCTION
* total dry matter weight
     TWT = WSH + WRT

* sum of dry matter weight of shoots and roots
     WSH = INTGRL(WSHI, GSH)
     WRT = INTGRL(WRTI, GRT)

* initial conditions
INCON WSHI = 50., WRTI = 50.

* growth rates
     GSH = 0.7 * GTW
     GRT = 0.3 * GTW

* net rate of total dry matter increase
     GTW = (GPHOT - MAINT) * CVF

* maintenance respiration
     MAINT = (GSH + WRT) * 0.015

* gross photosynthetic rate
     GPHOT = GPHST * (1. -EXP(-.7 * LAI))

* leaf area index
     LAI = AMIN1(WSH / 500, 5.)

* parameters
PARAM CVF = .7, GPHST = 400.

* timer variables
* FINTIM: time of finishing the simulation
* DELT: size of time step for integration
* PRDEL: printed output interval
* OUTDEL: plotted output interval
TIMER FINTIM = 100., DELT = 1., PRDEL = 5., OUTDEL = 5.

* perform simulation using rectanglar method after Euler
METHOD RECT

* printed variables
PRINT TWT, WSH, WRT, GTW

* plotted varible
OUTPUT TWT

* end of simulation model
END

* end of simulation program
STOP

* finish computer job
ENDJOB
```

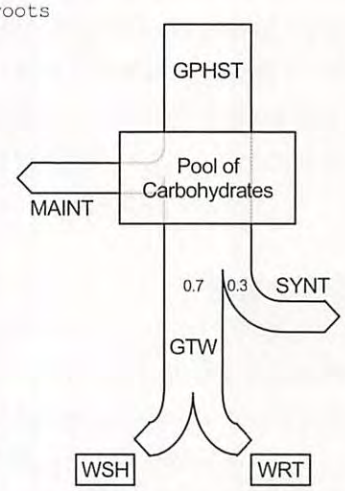

図13 CSMPによる乾物重生産のシミュレーションプログラム[17]

CSMPの構文はFORTRANに似ており，FORTRANでサブルーチンを書くこともできる．CSMPではプログラム文が自動的に計算順序通りにソートされるので，実行順に記述する必要がない．

その後の何度かの仕様改訂を経て（図12，Fortran 90からFortranと表記），現在でも利用されている．地球シミュレータでは並列化Fortranの一種であるHigh Performance Fortran（HPF）が利用されている．

生育モデル実装においてもFORTRANは広く利用され，APSRUグループのAPSIMではFORTRAN 77を，IBSNATグループのDSSATではFortran 90を，国内のSIMRIWではMS-FORTRANを利用して実装された．Wageningenグループでは，FORTRAN 77用のユーティリティTTUTILを開発し，FSEを通してCompaq Visual Fortranを利用している．

その後，多くのプログラミング言語が登場したが，

FORTRANの既存コードを書き換えるコスト以上のメリットがなかったため，FORTRANが使い続けられている[150]．

4) オブジェクト指向プログラミング言語

オブジェクト指向プログラミング（Object-Oriented Programming; OOP）が生まれた背景には，計算機の性能向上に伴い，より巨大で複雑になったアプリケーションの開発や保守コストの上昇が無視できなくなったことがある[22,218,109]．このような状況で最初に提唱されたのがCやPascalなどの構造化プログラミング言語により，関数を部品化することであった．部品の独立性を高めることにより，部品を再利用して新規開発するコード量を減らし，部品の修正が他の部品へ影響することを防ぐことができる．しかし，部品を修正せずにそのまま利用できることは少なく，データの表現には基本データ型を利用するままで，データの抽象化が行われなかった．

次の段階で提唱されたのが，状態としてのデータと機能としてのメソッドをオブジェクトとして扱い，オブジェクトとそれに対して送られるメッセージだけで世界を表現するオブジェクト指向プログラミングであった．オブジェクト指向プログラミングでは，データ抽象化（Data Abstraction），継承（Inheritance），動的結合（Dynamic Binding）といった特徴を持つ．

データ抽象化とは，公開されたインタフェースを通してのみ，オブジェクト内部のデータや属性にアクセスしたり，変更したりできるようにすることである．このようにデータとそれに対する操作をひとまとめにすることをカプセル化（Encapsulation）と呼ぶ．データ抽象化の利点は，カプセル内の変更が外部に及ぶことを防ぐ変更に対する耐久性と，インタフェース以外の具体的な内部構造を隠蔽できる概念の抽象化にある．

継承とは，あるオブジェクトの機能を受け継いで，新しいオブジェクトを作成することである．継承により既存のオブジェクトに機能を加えたり，修正したりでき，プログラムの再利用性を向上させられる．

動的結合とは，プログラムで実行するメソッドの

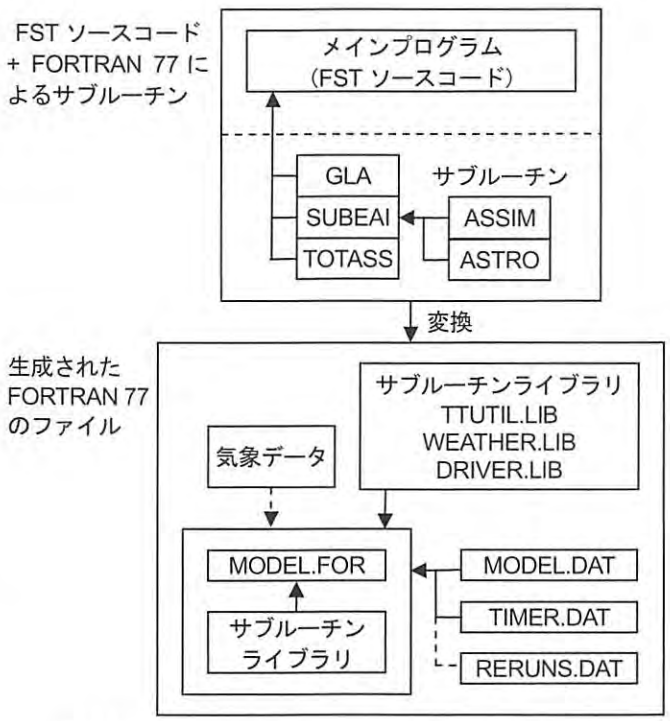

図14　FSTによるFORTRANへの変換の仕組み

FSTソースコードはFORTRAN 77によるソースコードMODEL.FORに変換され，ライブラリがリンクされる．気象データはデータベースから取得され，プログラム内にハードコーディングされる[278]．

MODEL.DATにはモデルの標準パラメータが格納されている．TIMER.DATには時間ループや積算に関する情報が格納されている．RERUNS.DATにはモデルのパラメータを変更して繰り返し実行する場合のパラメータセットが格納されている[139]．

属するクラスを実行時に動的に決めることであり，プログラムに柔軟性を持たせられる．

これらの技法により開発や保守の複雑さを低減し，開発効率と保守性を高める．どんなに成功したソフトウェアもいずれは保守期間に移行し，ソフトウェア保守コストがソフトウェア開発コストを上回ることもあるため，これらの特徴は重要である[21]．

最初のオブジェクト指向プログラミング言語（OOPL）は1960年代に登場したSimple Universal Language（Simula）である（図12）．Simulaはもともと Simulation Languageの略で，シミュレーション言語でもあった．Simulaにはシミュレーションを行うためのオブジェクト，クラス，継承，メソッドなどのオブジェクト指向の概念があった．そのため，それ以前のプログラミング言語では記述することが難しかった，シミュレーション対象（オブジェクト）の状態と振る舞いを表現し，かつ，並列に相互作用させるようなプログラムの開発が可能であった．Simula自身は広く普及することはなかったが，オブジェクト指向の概念はC++など多くの言語へと引き継がれていった．ちなみに，「オブジェクト指向」という言葉は，1970年代にAlan KayがSmalltalkの概念として使い始めたのが最初である．

作物モデルでのオブジェクト指向プログラミング言語を利用した初期の例として，個体や各器官をオブジェクトの階層構造で表現し，相互作用をシミュレートした綿花モデル[226]，CERES-Wheat[215]をオブジェクト指向版としてC++で書き直したCropSim[31]，時間，気象，作物，土壌，作業，害虫といったコンポーネントとそのインタフェースを提供する作物システムモデルCropSyst[273]がTurbo Pascalで実装されたことなどを挙げられる．また，情報工学分野で行われている，オブジェクト指向手法によるモデルとシミュレーションに関する研究[309]も参考になる．

5）Java

Java[235,16,20]は1990年代にSun Microsystems（2010年にOracleにより買収された）のJames Goslingによって開発され，1995年に公開されたクラスベースのオブジェクト指向プログラミング言語である．従来のプログラミング言語の良い部分を引き継ぎ，欠点を克服するよう設計され，CやC++の構文を引き継いでいる．定義したクラス（Class）をもとにインスタンス（Instance）が生成される．プログラムは複数のクラスから構成され，クラスを実体化した複数のオブジェクト（Object）がメッセージを交換しながら実行される．

Javaは静的型付けを採用しており，コンパイル時の型検査によりプログラムの間違いを発見できる．また，CやC++のプログラムでは，ポインタやメモリ管理に起因するプログラムの不具合を起こしやすかったが，Javaではポインタを廃止し，ガベージコレクションによりメモリ管理を行うようになったため，プログラムの頑健性が増し，安全性，開発効率，保守性が向上した．

Javaプログラムはコンパイルされると，機械語ではなくバイトコード（Byte Code）と呼ばれる中間言語に変換される．バイトコードは堅牢な実行環境であるJava仮想マシン（Java Virtual Machine; Java VM）のもとで動作する．各プラットフォーム用のJava VMで同じバイトコードを実行できるので，プラットフォーム非依存という特徴を持ち，"Write Once, Run Anywhere"は初期のJavaの標語でもあった．初期の頃には実行時にバイトコードをインタプリタで機械語に変換していたため，実行速度が遅いという問題があったが，ジャストインタイム（Just In Time; JIT）コンパイル方式で改善された．

Javaには標準で充実したApplication Program Interface（API）が用意されており，マルチスレッド，ネットワーク，XML文書，GUIコンポーネントなどを利用し，多言語対応なプログラムを簡単に開発できる．また，プログラム開発時に，コード中に定められた書式で処理，変数，返り値の内容など記述しておけば，Javadocにより保守に必須の参照しやすいHTML形式のドキュメントを生成できる．

Webアプリケーション構築においては，サーバ側では2000年前後にJavaサーブレット（Servlet）が急速に普及し，多くのサーバプログラムの開発に利用されている．クライアント側では，初期の頃のJavaアプレット（Applet）の起動に時間がかかった問題が解消され，RIA開発に利用されている．デスクトップアプリケーション構築においても，計算機性能の向上とJava VM，Javaコンパイラの改良により，NetBeans[236]やEclipse[48]などの統合開発

環境（Integrated Development Environment; IDE）がJavaで開発されている．

Javaで構築したアプリケーションのサーバへの配備は，Javaアプレットは Java Archive（JAR）ファイル，Javaサーブレットは Web Application Resources（WAR）ファイルという，プログラムをZIP形式でアーカイブしたファイルを，Webサーバやサーブレットコンテナの所定の位置にコピーするだけである．

携帯端末などのリソースの限られた実行環境用には，Javaの小型セット Java Micro Edition（Java ME）があり，多くの携帯電話や Personal Digital Assistant（PDA）に搭載され，携帯端末用アプリケーション開発に利用されている．

以上のような，オブジェクト指向プログラミングによる開発効率と保守性の良さ，ネットワーク，分散オブジェクト，XML文書などを扱うための豊富なAPI群，また，アプリケーションが広く利用されるための必須条件である Webアプリケーションとしての構築，多言語対応といったJavaの優れた多くの特徴により，本研究での提案を実装するための開発言語としてJavaを採用することにした．

6）UML

統一モデリング言語（Unified Modeling Language; UML）[189]は，オブジェクト指向プログラミングによるモデル化の設計，分析を行うためのモデリング（仕様記述）言語である．グラフィカルな記述を用いるため，大規模なプログラムの構造や動きを理解しやすくし，グループ開発におけるコミュニケーションツールとして機能する．

プログラムを図解する方法としては，フローチャートやデータフロー図などがあった．オブジェクト指向言語が注目されるようになると，それらを図解するために，Booch法[22]や Object Modeling Technique（OMT）法[218]などが登場した．これらの多様なモデリング言語の存在による混乱を回避するために，統一仕様としてまとめられたのがUMLである．

UMLはシステムの静的な構造を表現するための構造図（Structure Diagram）と，振る舞いを表現するための振る舞い図（Behaviour Diagram）から成る．特にオブジェクトに注目した図として，構造図の一種でオブジェクト同士の関係を表現するためのオブジェクト図（Object Diagram），振る舞い図の一種でオブジェクト間のメッセージのやりとりを表現するためのコミュニケーション図（Communication Diagram）がある．

UMLの農業モデルへの適用は Florida大学を中心に行われた．特に農業モデルを再利用するためのドキュメント記述用ツールとしての役割に注目している[195,196,197]．

さらに，年々登場する新たな技術に対し，アーキテクチャと設計を分離することで対応しようとする，モデル駆動型アーキテクチャ（Model-Driven Architecture; MDA）[191]が提唱された．複雑な農業システム構築のために，UMLを利用した農業モデル実装を発展させた，MDAを利用したコード生成が試みられた[198]．

7）JavaScript

JavaScript は Netscape Communications（1998年にAOLに買収された）の Brendan Eichによって開発され，1995年に公開された，プロトタイプベースのオブジェクト指向スクリプト言語である．開発当初は LiveScript と呼ばれていたが，同時期に登場したJavaが大きな注目を浴びていたため，JavaScriptという名前に変更された．しばしば混同されるが，JavaScript と Java は別物である．

1996年に Internet Explorer に搭載されると，手軽に動的な Webページを構築できることから急速に普及していったが，各Webブラウザで独自の拡張が行われたため，Webブラウザ間での互換性は低かった．1997年に通信に関する標準を策定する国際団体 Ecma International によって，JavaScriptの仕様が ECMAScript[19]として標準化され，多くの Webブラウザで利用できるようになった．しかし，Webブラウザの実装の違いによる互換性の問題は完全には解消されていない．

Googleマップが公開されると，Ajaxを利用した高機能な Webアプリケーション開発言語として再び注目を集めるようになった．さらに YUI Library[304]，Ext JS[225]などの本格的な GUIライブラリの登場により，デスクトップアプリケーションと同様なユーザインタフェースの構築が可能になった．

3. モジュール構造

気象や土壌水分，窒素の影響のみを利用する作物モデルの精度を向上させようと，既存のプログラムに雑草や病害虫の影響を追加していくだけでは，複雑で保守不可能なプログラムとなるか，もとのプログラムの大幅な書き換えを強いられることになる．そのため，モデルのプログラム開発にあたっては，あらかじめ開発，機能追加や保守の効率性を考慮したプログラム構造を採用することが重要である．

ICASA に参加する3つの作物モデル開発グループ〈I.3.1)〉のすべてに共通していることは，モジュール構造（Module Structure）によるモデル実装を行っていることである[114]．

IBSNAT グループの DSSAT では CERES 由来のモデルの保守が困難になったため，モジュール構造の CROPGRO への移行作業が行われていた．この作業は APSRU グループによる APSIM のモジュール構造に動機付けられていた．さらに，APSIM は Wageningen グループによる FSE/FST のモジュール構造の手法を利用していたというように，各グループはモジュール化について相互に影響を与えていた[115]．

モジュール構造の要件として，以下のことが挙げられる[I.209]．

- モジュールは容易に機能ごとに分離可能である．
- モジュールは最小限の入出力変数を持ち，インタフェースを通してのみ入出力を行う．
- あるモジュールに対する修正の影響が他のモジュールへ及ばず，他のモジュールと独立に検証されている．

また，モジュール構造の要件を満たすことによる利点として，以下のことが挙げられる[114]．

- メインプログラムや他のモジュールとは，インタフェースを通してのみ影響を与えるので，わずかな変更でコンポーネントを追加したり，分離したりできる．
- 同じ目的の異なるモデルの比較が，モジュール単位で容易に行える．
- ソースコードの文書化や共有が容易である．
- 異なるプログラミング言語で記述されたモデルをリンクできる．
- プログラムがモジュール単位でカプセル化されるので，モデルの保守性が高まり，利用年数が延びる．
- 研究者は科学的な機能のモジュール開発に専念し，それ以外のプログラムを開発するプログラマと役割分担ができる．

1) 作物モデルのプログラム構造

(1) Wageningen グループ

Wageningen グループ〈I.3.1) (1)〉は SUCROS や ORYZA などを FSE〈III.2.2)〉で実装している．図15は FSE による作物モデル実行の流れを表している．FSE では開始，成長速度計算，集計，終了の4つのセクションを持つ．作物モデルの実行が開始されて初期化が行われると時間ループに入る．時間ループ内では，成長速度計算と集計が時間 Δt だけ進みながら，終了条件が成立するまで繰り返される．その後，終了処理が行われて作物モデルの実行が終了する．この開始から終了までの流れは FSE ドライバによって管理される．

図16はモデル実行時の FSE ドライバとモジュールの関係を表している．FSE ドライバでの初期化，集計，終了処理に合わせて，作物や土壌モジュールの対応する処理が実行される．集計では FSE ドライバによって，モジュール内部の状態が参照され，次のループでその状態が他のモジュールに反映される．

(2) IBSNAT グループ

IBSNAT グループ〈I.3.1) (2)〉による DSSAT の実装は，世界中の多くの機関に分散している開発者の協力によって行われるため，実装された様々な作物モデルに互換性がなく，保守に支障をきたすようになっていた[114]．この問題点に対応するために，DSSAT の v3.5 から v4.0 への改良において，DSSAT-Cropping System Model（CSM）を用いた作物モデル実装への移行が行われた[115]．DSSAT-CSM のプログラムはモジュール構造になっていて，〈I.3〉で挙げたモジュール構造の利点を享受している．

図17は DSSAT-CSM のコンポーネントとモジュール構造の概要を示している．メインプログ

ラムはランドユニットモジュールを通して，各モジュールを呼び出している．モジュールの呼び出しはプライマリモジュールと，その配下のモジュールの2段階で行われる．各モジュールの入出力や実行は独立していて，インタフェースを通してのみコミュニケーションするため，モジュールの入れ替えが容易である．初期のDSSATの各作物モデルは同じような土壌モデルを内部に持っていたが，DSSAT-CSM導入により，作物モデルと土壌モデルは分離され，独立したモジュールとなった．これにより，モデル構造が単純化され，実行，保守が効率化された．

新しい作物モデルを追加するには2つの方法がある．新しいモジュールを作成するCERESモデル

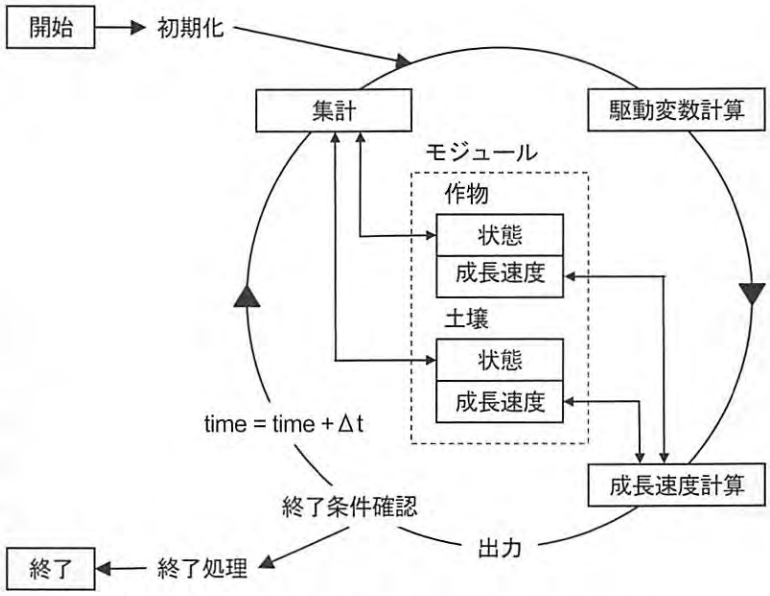

図15　FSEによる作物モデル実行の基本構造 (279)

作物モデルの実行が「開始」されると初期化が行われる．その後，時間Δtだけ進めながら，「成長速度計算」と「集計」が終了条件成立まで繰り返される．終了処理が行われると，作物モデルの実行が「終了」する．

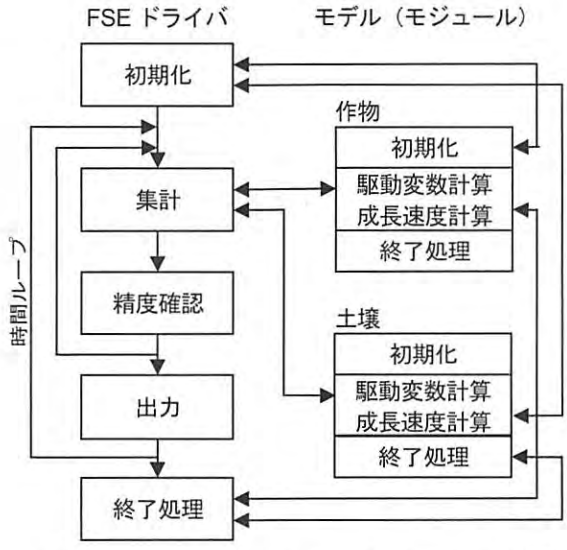

図16　FSEによるシミュレーションループ (139)

FSEドライバでの初期化，集計，終了処理に合わせて，作物や土壌モジュールの対応する処理が実行される．

の方法と，テンプレートの値を変更することによるCROPGROモデルの方法がある（図17右下）．CROPGROのテンプレートは，作物の特性を定義する，生態系（ECO）ファイルと品種（CUL）ファイルから構成され，現在，DSSATの作物モデル実装の主流となっている．

図18はメインプログラムとモジュールの実行時の関係を表している．各モジュールの実行サイクルは，開始時の初期化，時期毎の初期化，計算，集計，要約，出力の6ステップになっている．メインプログラムが実行サイクルと，ステップに応じてどのモジュールを呼び出すかを制御している．メインプログラムの役割は基本的にFSEドライバと同じである．

(3)　APSRUグループ

APSRUグループ〈I.3.1）(3)〉は，作物モデルの実装時に十分なソフトウェア設計を行わなかったことに起因するモデルの複雑さが，ソフトウェアを不安定で柔軟性がないものにした先例に学び，開発当初から十分なソフトウェア設計を行い，モジュール構造による作物モデル実装を行った[114]．APSIMの作物モジュールの約50%のコードが作物ライブラリの汎用ルーチンで構成されていることが示すように，モジュール化は同じコードが複数のモジュールに存在することを防ぎ，開発，保守の効率化に貢献

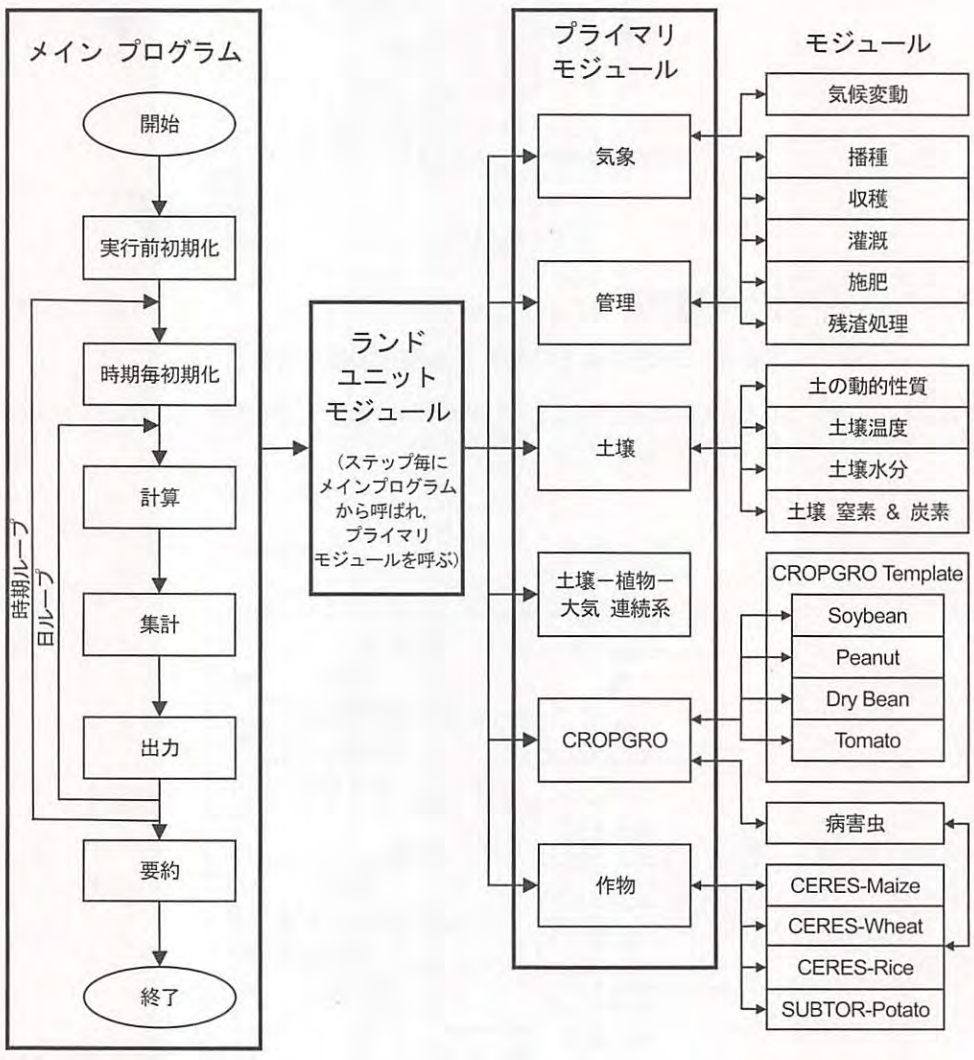

図17　DSSAT-CSMのコンポーネントとモジュール構造の概要[115]

メインプログラムはランドユニットモジュールを通して，プライマリモジュールと，その配下のモジュールの2段階で各モジュールを呼び出す．

する．

APSIM のモジュール構造の中心となる要素は，図 19 に示されるコミュニケーションフレームワーク（APSIM エンジン）である．APSIM エンジンはモジュールインタフェースを通して各モジュールとリンクする．また，リンクの解除や，モジュールの交換も容易である．このモジュール方式は柔軟性があるので，複数の生物モジュールを組み込むことにより，複雑な農業システムを構築できる．

図には示されていないが，APSIM エンジンは，CLOCK モジュールを用いてシミュレーション内の時間を管理し，各モジュール間の同期をとりながら，1 日ごとや数時間ごとの様々な間隔で繰り返し実行を行っている．また，APSIM は FSE ドライバのように，初期化，前処理，処理，後処理のような実行サイクルを持っている．

3 つの作物モデル開発グループによるモジュール化は一定のレベルに達している．どのモジュール構造も，作物モデル実行の中核部分である FSE ドライバ，DSSAT のメインプログラム，APSIM エンジンなどの機能や，各モジュールがインタフェースを通して中核部分と容易に結合，分離できることは類似している．

しかし，同じような仕組みに見えても，パラメータ，初期値，データの読み込みメソッド，モジュール間の交換メッセージ，計算順序，イベント処理などの実装に多くの違いがあった．そのため，他のグループのモジュールを利用するためには，無視できない量のプログラミングが必要であった[114]．

1990 年代後半には，Java の登場と同時に注目され始めたオブジェクト指向プログラミングをモジュール構造での実装に取り入れる研究が開始されている．

4. フレームワーク

フレームワーク（Framework）とは，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[110,111]．フレームワークは半完成品のアプリケーションのようなものである．フレームワークを利用した

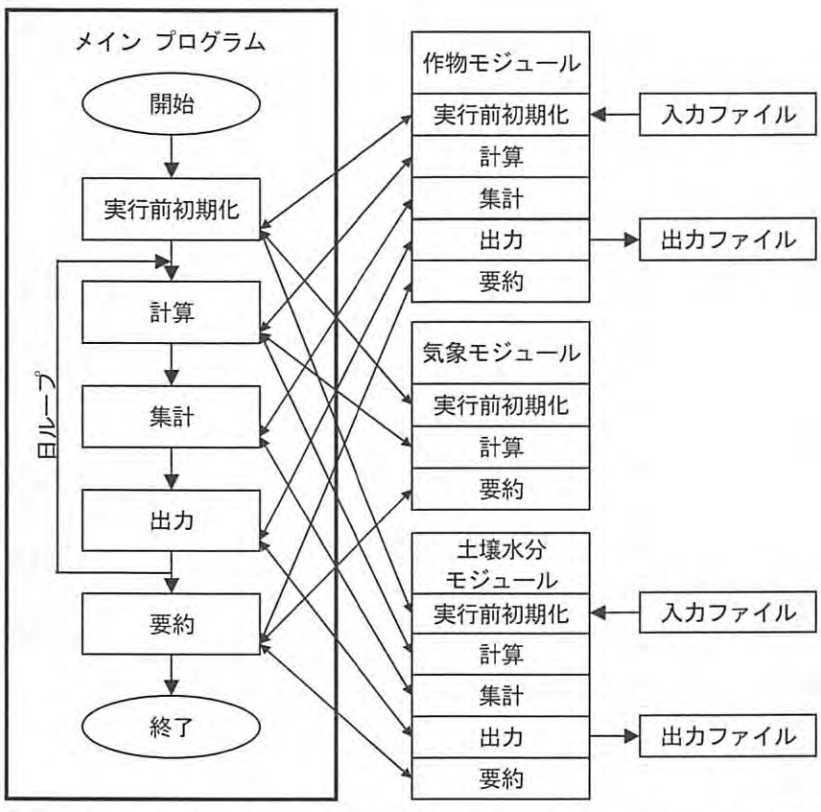

図 18　DSSAT におけるメインプログラムとモジュールの実行時の関係 [204,114]

メインプログラムが実行サイクルと，ステップに応じてどのモジュールを呼び出すかを制御するのは，FSE ドライバと同じである．

アプリケーション構築は，フレームワークのデフォルトの機能と異なる部分を新たに開発することによって行われる．新たに開発されたプログラムは，アプリケーション実行時にフレームワークから呼び出される．この点がプログラムから呼び出される側にある普通のライブラリと異なるところである．開発効率を高めるためのフレームワークであるが，依存しすぎると他のフレームワークへの乗り換えが制限され，かえってプログラムの再利用性が低下することもある[217,44]．

デフォルトの機能は継承して，新しい機能のみを開発するために，フレームワークでは一般的にオブジェクト指向言語が利用される．Javaによる Web アプリケーション構築用フレームワークとして，Apache Struts[11]，JavaServer Faces（JSF）[240]，Apache Tapestry[12]，Google Web Toolkit[73] などがある．

水文モデル用には，いくつかのフレームワークが構築され，実運用されているアプリケーション構築に利用されているものがある．主な水文モデル用フレームワークとして Modular Modeling System（MMS）[136]，Object Modeling System（OMS）[35]，OpenMI[76]，TIME[206]，Jena Adaptable Modelling System（JAMS）[128] などがある．また，複数のフレームワークで利用可能なコンポーネントについても研究されている[15]．

作物モデル用のフレームワークは，作物モデル開発グループによりモジュール構造による作物モデル実装の研究が十分に行われていたので，公開されているものは多くないが，農業生態系シミュレーションモデル用の ModCom[83,287,274] などがある．ModCom は FSE ドライバと同様なモデル実行エンジンの機能や，各種コンポーネントを記述するインタフェースの仕様を提供する．開発言語とオペレーティングシステムに依存しないために，コンポーネント間通信に Component Object Model（COM）を利用している．

5. 農業モデル用フレームワーク

〈I.3〉のモジュール構造は，作物モデル開発グループでのみ利用されており，グループ外でモデル実装に利用されていなかった．モジュール構造の開発時期から対象はスタンドアロンのアプリケーションであり，Web アプリケーションとして，また，ネッ

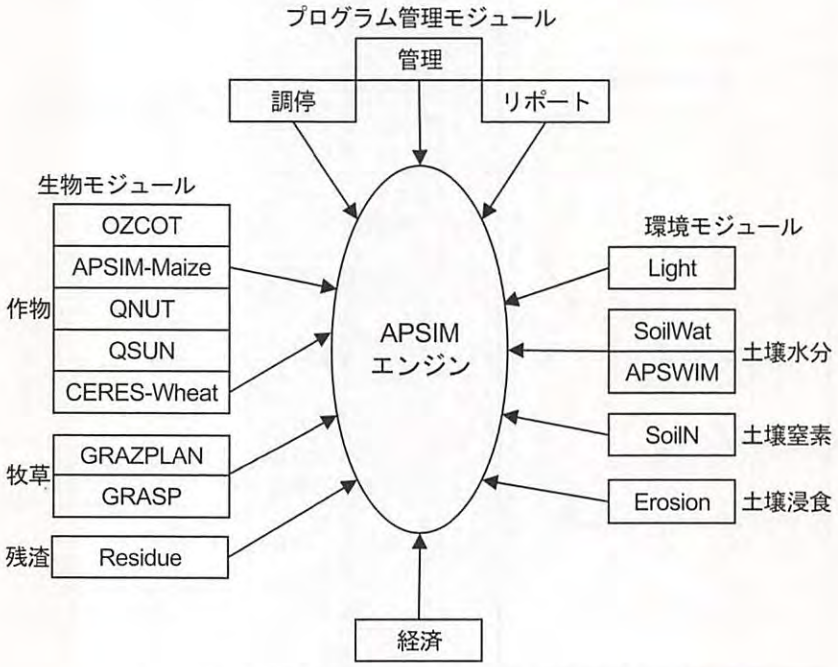

図 19　APSIM の plug-in pull-out システムの構成[150]

APSIM のモジュール構造のキー要素である APSIM エンジンは，モジュールインタフェースを通して各モジュールとリンクする．また，リンクの解除やモジュールの交換も容易である．

トワーク上に分散したコンポーネントとして開発することを想定していなかった．

AMADIS用の農業モデルの実装を助けるためのプログラムパッケージは，単なるプログラムライブラリとしてではなく，GUIコンポーネントも含めて，農業モデルのWebアプリケーションのすべてを構築できるパッケージとなっていることが望ましい．それを満たす，Javaによる農業モデル実装フレームワークを構築する．このフレームワークは英語名のJava Agricultural Model Frameworkから JAMF（ジャムフ）と名付ける．

JAMFでは農業モデルに共通するモデル実行機能（モデルエンジン），データ構造，気象データの取得機能や，GUIコンポーネントによるデータの表示機能が提供されている．農業モデル実装時には，個々の農業モデルに固有な計算部分や，利用するデータやパラメータの設定部分に対応するJAMFのクラスを継承したり，インタフェースを実装したりするだけでよい[253]．以下でJAMFの詳細を，プログラムのコードの一部を示しながら紹介する．

農業モデルの実行の流れは，作物モデル開発グループのモデル構造（図15～図19）を参考に，図20のような，モデル実行エンジンによる実行用データから結果データへの変換処理として表すことができる．また，この変換処理のために，データ構造やGUIコンポーネント群が付随している．この処理やデータなどを機能によりパッケージとして分類したものが，JAMFのパッケージ構成（表5）である．すべてのJAMFパッケージと，パッケージに含まれるクラスやインタフェースのドキュメントは，Javadocにより生成されたWebドキュメント[249]として閲覧できる．JAMFを利用したサンプルモ

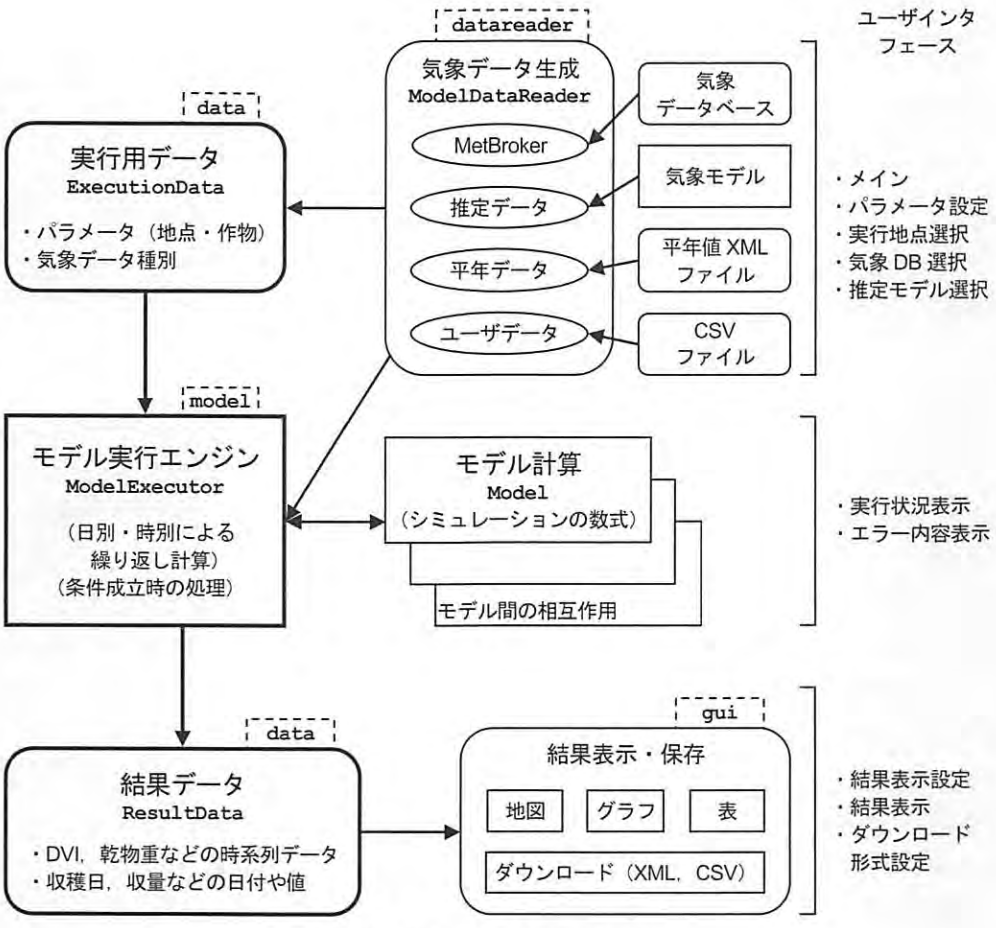

図20　農業モデル実行の流れ

JAMFでの農業モデルの実行の流れと，実行時に画面表示されるユーザインタフェース．
各要素の**Courier**体の文字列は，その機能を開発するために利用されるJAMFパッケージ名（破線内）と，インタフェース名やクラス名（実線内）である．

表5 農業モデルフレームワークJAMFのパッケージ構成（主要部）

分類	パッケージ名	パッケージの内容
気象ジェネレータ〈III.5.1〉	broker	MetBrokerやChizuBrokerを扱うためのクラスを提供
	datareader	MetBrokerを含む，各データリーダのインタフェース，およびクラスを提供
	data.normalvalue	平年気象データに関するクラスを提供
モデルデータ〈III.5.2〉	data	モデルのデータクラスに関するクラスを提供
	data.xml	モデルのデータをXML形式のデータとして扱うためのクラスを提供
ユーザインタフェース〈III.5.4〉	gui	GUIクラスを提供
	gui.chart	グラフを利用するためのGUIクラスを提供
	gui.map	地図上にデータを表示するためのGUIクラスを提供
	gui.metbroker	MetBroker設定用のGUIクラスを提供
モデル実行エンジン〈III.5.3〉	model	モデル実行エンジンのためのクラスを提供
その他	text	各種データのための書式に関するクラスを提供
	util	様々なユーティリティクラスを提供
	xml	様々なXMLドキュメント用ユーティリティクラスを提供

すべてのJAMFパッケージと，パッケージに含まれるクラスのドキュメントはWebドキュメント[249]として参照できる．

デルのプログラム開発のための解説もあるので，アプリケーション開発者はJAMFを利用した農業モデル実装技術を円滑に身につけることができる．

JAMFを利用したJavaアプレット版の農業モデル実装に必要なプログラムファイルは，表6に挙げる約10個のプログラムファイルから構成される．プログラムファイルの数は，モデルが扱うデータ種別，設定画面の数，対応言語の数により増減する．

1) 気象ジェネレータ

農業モデル統合システムで利用される気象データは，気象ジェネレータ〈I.3.3）(1)〉によって提供される．これらの気象ジェネレータの主な役割は，過去の気象データから気象モデル用のパラメータを生成することと，作物モデルを利用した予測用の未来の気象データを生成することである．観測されたり，生成されたりした気象データは，書式の定義されたテキストファイルで扱われるため，気象データベースからデータを取得するためのデータベースアクセス機能は備えていない．

AMADISの構成要素である農業モデルも実行には気象データが必須であるため，JAMFは気象ジェネレータ関連パッケージを提供し，農業モデルが複数の方法で気象データを取得できるようにしている．JAMFの気象ジェネレータの気象データ取得元は，MetBroker（気象データベース），気象データ推定モデル，平年気象データ，ユーザデータ（テキストファイル）の4種類からなる（図21）．また，取得した気象データに欠測値があった場合のための補間機能がある．

MetBroker〈II.2.5)〉は気象データベースアクセスの中核機能となっている．MetBrokerを利用す

表6　個別農業モデル実装に必要な追加的プログラム

分類	クラス名	クラスの内容
データ	SampleConstants	農業モデルで扱うデータを特定するためのキーを定義する.
	SampleData (ExecutionDataImpl)	農業モデルが利用するデータを扱う．時系列データの読み込み元の設定や，その他のデータの初期値設定を行う.
	SampleResultData (ResultDataImpl)	農業モデルの実行結果（値・日付・時系列データ）を扱う.
モデル実行エンジン	Sample (AbstractModel)	設定されたパラメータや取得された気象データを利用してモデルの計算を行う.
	SampleMain (ModelMain)	農業モデルを Java アプレットまたはアプリケーションとして起動する．実行時にパラメータが渡された場合は，その設定も行う.
ユーザインタフェース	SampleUI (AbstractModelUI)	メイン画面に表示する GUI の設定とレイアウトを行う．また，GUI のイベントを処理する.
	SampleTableUI (ModelTableFrame)	モデルの計算後に，モデルの結果データの内容を表やグラフとして表示する.
	SampleMap (MultipleStationResultMap)	モデルの結果データの内容を地図上に表示する.
	SampleResources SampleResources_ja (ModelResources)	モデルの実行環境に応じて，適切な言語を表示するためのリソースファイル．デフォルト用と，その他の対応言語用がある.

各クラスは，クラス名の下の括弧内のパッケージに含まれるクラスや抽象クラスを継承したり，インタフェースを実装したりすることにより，個別モデルごとの機能の違いを実現する.

クラス名は農業モデルの名前が Sample の場合．SIMRIW モデルなら，各クラス名の Sample の部分が Simriw となる.

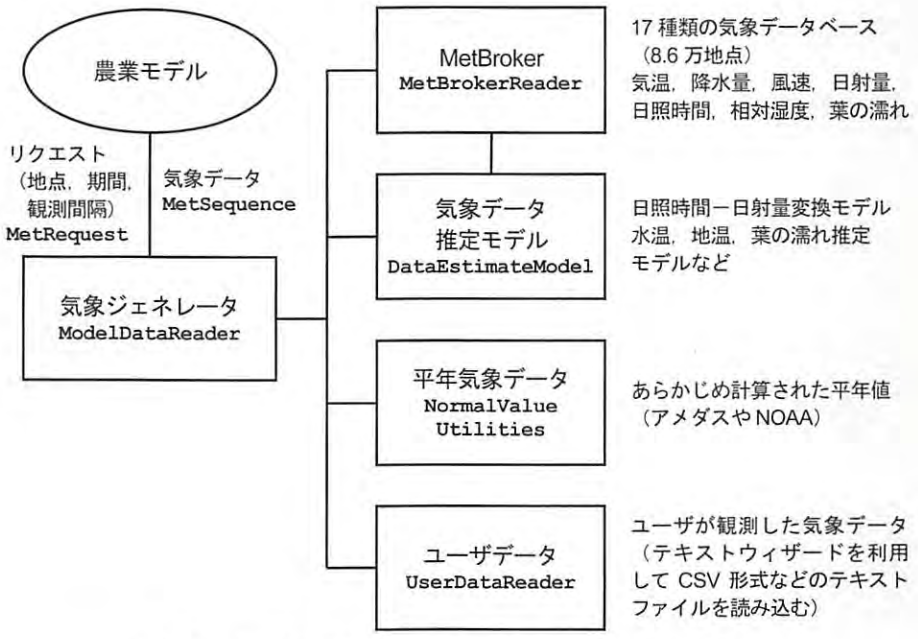

図21　JAMF の気象ジェネレータと気象データ取得元

気象ジェネレータは，農業モデルからのリクエストに応じて，MetBroker，気象データ推定モデル，平年気象データ，ユーザデータの4種類の気象データ取得元から気象データを取得し，リクエストに適うデータオブジェクトとして，農業モデルに返す.

各要素の Courier 体の文字列は，JAMF のインタフェース名やクラス名である.

ることにより，気象ジェネレータはデータベースごとにアクセスプログラムを記述することなしに，MetBrokerが扱う国内外の様々な気象データベースを利用可能になる．また，MetBrokerのデータ取得機能により，ユーザは農業モデルの実行地点を，気象観測地点IDによって指定するか，2点の緯度経度を指定した矩形領域内に存在するすべての気象観測地点として指定するかを選択できる．

気象データ推定モデルは，農業モデルの実行地点で観測されていないためにMetBrokerから取得できない気象データを，その他の観測されている気象データから気象モデルを利用して推定する機能を提供する．JAMFの気象ジェネレータでは，日射量，土壌温度，土壌水分，水田水温，葉の濡れを推定する機能を提供している．例えば，日射量は日照時間から推定され，葉の濡れは降水量から推定される．データ推定に利用されるモデルは複数登録することが可能であり，ユーザが選択できる．日射量を利用する農業モデルは，国内で日射量を観測している約150ヶ所の気象観測地点でしか実行できないが，気象データ推定機能により，日照時間を観測している約1000ヶ所のアメダスの気象観測地点でも実行可能となり，農業モデルの実行可能地点を日本全国の多くの地点に広げることができる．

平年気象データは，農業モデルが未来の予測を行う場合に利用される．現在までの気象データは，気象データベースなどからの実測値を利用し，それ以降の予測用データとして平年値が挿入される．異常気象や温暖化の影響を反映させたるため，平年気温データに定数値を加算する機能がある．JAMFの気象ジェネレータでは，アメダスやNational Oceanic and Atmospheric Administration（NOAA）が提供するNational Weather Service（NWS）[173]などの主要な気象データベースの平年値をXML形式のデータで保持している（表7）．当初は，平年値用クラスのオブジェクトをシリアライズしたものを保存形式としていたが，バイナリファイルで可読性がないことと，平年値用クラスの修正を行うとデシリアライズ操作が煩雑になるため，ファイルサイズが数倍に増加する（表8）という欠点があるが，保存形式をXML形式に変更した．

ユーザデータは，ユーザの手元にある気象データが記録されたテキストファイルを，農業モデルで利用できるように取り込む機能を提供する．主にユーザによって観測された気象データや，MetBrokerが対応していない種類の気象データを扱うために利用される．JAMFが提供するテキストデータウィザードを利用することにより，固定長形式やCSV形式などのテキストデータ形式に柔軟に対応できる．

表7　平年値データの保存形式によるファイルサイズ

保存形式	日別値			時別値	
シリアライズ	気温 その他	9KB 6KB		39KB	
XML	気温 その他	37KB 13KB	1112行 376行	330KB	8794行

日別気温は最高，最低，平均の3要素を含むため，その他の気象要素よりファイルサイズが大きくなる．

表8　データベースごとの平年値データの合計ファイルサイズ

保存形式	アメダス （9600ファイル）	NOAA/NWS （44000ファイル）
シリアライズ	207MB	274MB
XML	1600MB	763MB

NOAA/NWSは日別値のみ扱うのに対し，アメダスはファイルサイズの大きい時別値を扱うため，保存形式をXMLにしたときの合計ファイルサイズの増加率が大きい．

ファイル数は，観測地点数×気象要素数に対応している．

図22はJAMFを利用した気象データ取得元の設定を行うためのソースコードである．JAMFの気象ジェネレータ関係のプログラムは，**datareader**パッケージとして提供される．農業モデルが利用する気象データの設定は，実行データ用の**ExecutionDataImpl**クラスを継承するクラスで行われる．

1. 実行に必要な気象要素が**addSequenceElement()**メソッドによって設定される．ここでは気温と葉の濡れが設定されている．
2. 気象データに関する属性を管理するための**DataSourceAttribute**オブジェクトが取得される．
3. 気温のデータ取得元としてMetBrokerとユーザデータが**addUsableDataSource()**メソッドによって設定される．葉の濡れに対しては，さらに**setDataEstimateModel()**メソッドによって葉の濡れ推定モデルも設定される．
4. **getUserDataReader()**メソッドはユーザデータ用のテキストデータウィザードの機能が設定された**UserDataReader**オブジェクトを返す．

個々の農業モデルで異なる，必要な気象データの設定に関する記述のみで，気象ジェネレータの気象データ取得元設定ためのGUIコンポーネント群（図23）が自動的に生成される．これらのGUIコンポーネント群に限らず，気象データの取得機能，取得したデータを農業モデルの実行用データオブジェクトとしてまとめる機能などは，JAMFが提供するデフォルト機能が自動的に継承される．このように，新しい農業モデルのためにデフォルトと異なる部分のみを新たに開発するだけで済むところが，フレームワークを利用したアプリケーション構築の利点である．

JAMFの気象ジェネレータで，各取得元から気象データを取得する管理を行うのは**ModelDataReader**クラスであり，ユーザの設定に応じて，**DataReader**インタフェースを実装した**MetBrokerReader**クラスや**UserDataReader**

```
import amf.data;
import amf.datareader;

public class SampleModelData extends ExecutionDataImpl{
  public SampleModelData(){
    setResolution(Duration.ONE_HOUR);                    //モデルの実行サイクルの設定（時別）

    addValueElement(LEAF_WETNESS_THRESHOLD);             //値の定義（葉の濡れの閾値）
    setValue_(LEAF_WETNESS_THRESHOLD, 80.0);             //値の設定

    addSequenceElement(AIRTEMPERATURE);                  //時系列データの定義（気温）
    addSequenceElement(LEAFWETNESS);                     //時系列データの定義（葉の濡れ）
    DataSourceAttribute dsAttrAirTemp = getDataSourceAttribute(AIRTEMPERATURE);
    DataSourceAttribute dsAttrLeafWet = getDataSourceAttribute(LEAFWETNESS);
    //時系列データの取得元設定
    dsAttrAirTemp.addUsableDataSource(DataSourceElement.MET_BROKER);      //MetBroker
    dsAttrAirTemp.addUsableDataSource(DataSourceElement.USER_DATA);       //ユーザデータ
    dsAttrLeafWet.addUsableDataSource(DataSourceElement.MET_BROKER);
    dsAttrLeafWet.addUsableDataSource(DataSourceElement.USER_DATA);
    dsAttrLeafWet.addUsableDataSource(DataSourceElement.ESTIMATED_DATA); //推定モデル
    setDataEstimateModel(LEAFWETNESS, new LeafWetnessEstimateModel());
  }

  public UserDataReader getUserDataReader(){
    //ユーザデータの設定　気温と葉の濡れ
    UserDataReaderImpl udReader = new UserDataReaderImpl();
    udReader.addTextDataElement(
              new TextDataElement(AIRTEMPERATURE, AIRTEMPERATURE.toString(), "C"));
    udReader.addTextDataElement(
              new TextDataElement(LEAFWETNESS, LEAFWETNESS.toString(), ""));
    return udReader;
  }
  …
}
```

図22　JAMFを利用した気象データ取得元設定を行うためのソースコード

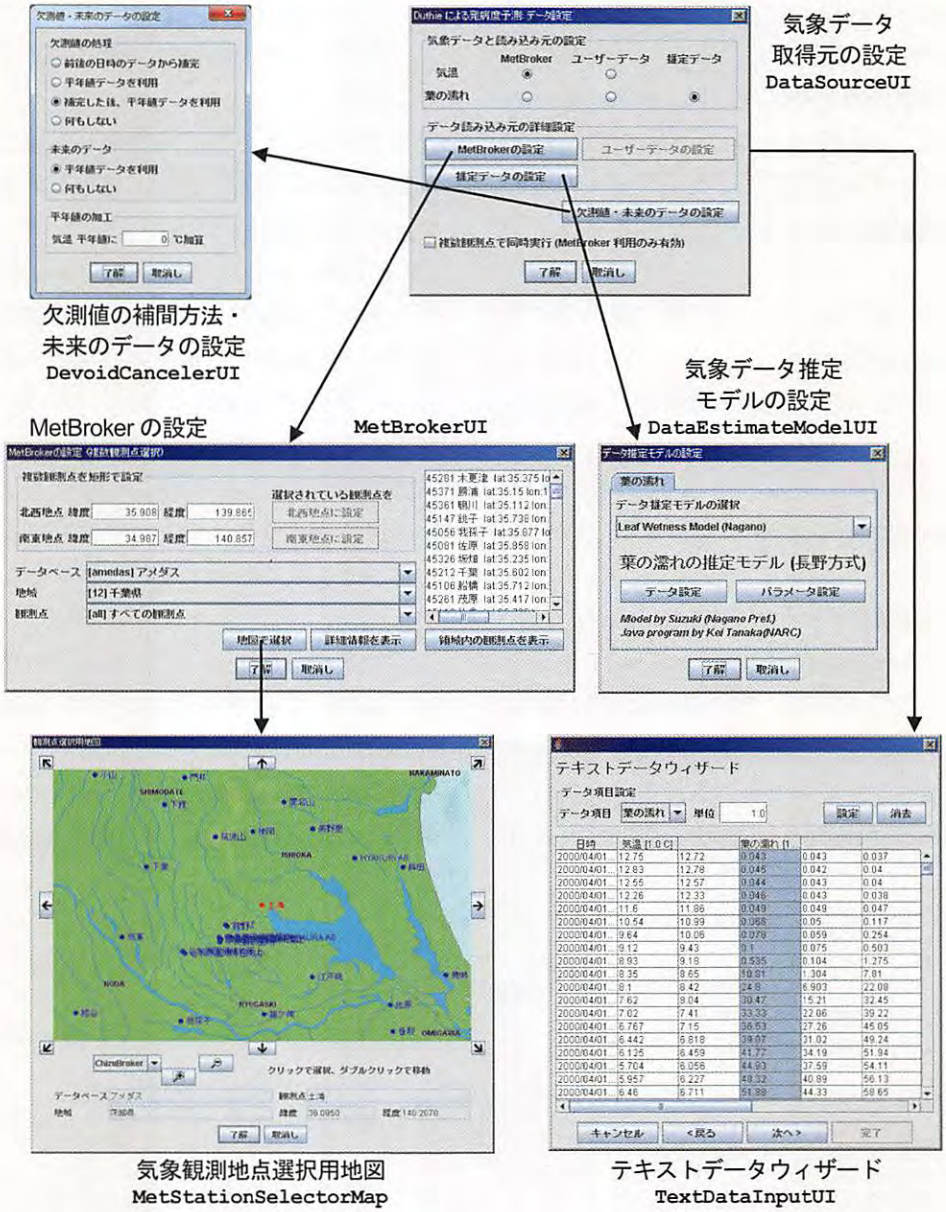

図 23 気象ジェネレータのための GUI コンポーネント群

図 22 の気象データの設定プログラムのみで，GUI コンポーネント群は JAMF により生成される．

個々の農業モデルで異なる，必要な気象データの種類や取得元のみ指定すれば，各モデル共通で利用できる GUI コンポーネント群のプログラムは，JAMF により自動的に継承される．

各要素の **Courier** 体の文字列は，JAMF のインタフェース名やクラス名である．

クラスに実際の気象データ取得を委譲する．

MetBroker へ接続する **getConnection()** メソッド，気象観測地点リストを取得する **listStations()** メソッド，リクエスト用の **StationMetRequest** や **SpatialMetRequest** オブジェクトの生成を行い，気象データを取得する **supplyMetData()** メソッドなど，MetBroker 関連操作を行うメソッドを持つ **KMetBroker** クラスなどは **datareader** パッケージとは別に，**broker** パッケージとしてまとめられている．

JAMF の気象ジェネレータの欠測値の補間や未来の予測用に平年値を挿入する機能は，**DevoidCanceler** クラスの **fillDevoid()** メソッドによって行われ，補間方法を過去と未来に分けて設定できる．ユーザはこれらの設定を図 23 の GUI コンポーネントで行える．平年気象データに

関するクラスは**data.normalvalue**パッケージとしてまとめられている．平年気象データを利用するときに温暖化の影響を反映させたい場合には，**setAirTempAddValue()**メソッドによって気温に一定値を加算できる．

GUIコンポーネント群のうち，気象データ取得元設定用の**DataSourceUI**クラスと気象データ推定モデル設定用の**DataEstimateModelUI**クラスは**gui**パッケージで，MetBroker関連の設定用の**MetBrokerUI**クラスなどは**gui.metbroker**パッケージで，ユーザデータ設定用の**TextDataInputUI**クラスなどは**gui.textdata**パッケージで提供される．

2) データ構造

シミュレーションモデルの実行は，入力データを計算式によって出力データに変換する作業のことである．モデルの実行過程では，データの集合に対し値の設定や取得のような基本的操作を繰り返し実行することが多い．このような基本操作を効率良く実行できるようなデータ構造を利用することにより，データ操作の煩雑さから解放され，モデルの全体的な開発に集中できる．また，工夫されたデータ構造の利用は，モデル実行効率の向上につながる[18]．

MS-DOS時代には，データはソフトウェアが直接読み書きできるASCIIコードを利用し，固定長形式やカンマ区切り形式で記録されていた．実験を行った圃場，年，番号などは，ファイル名8文字＋拡張子3文字の制限のもと，ファイル名により管理されていた[99]．また，ファイルサイズ節約のために，データ項目名を短縮形で表し，可読性を犠牲にしていた．

IBSNATグループではDSSAT用のデータ標準のためのIBSNATファイルを定めた[100]．IBSNATファイルでは計測項目に共通語彙を割り当て，ファイルの構造と書式が定められている．DSSATを用いた実験や調査の文書化に採用され，モデルの比較や改善を支援している．しかし，各計測値の計測方法，単位，計測項目の定義などの曖昧さに問題があった．

その後，IBSNATファイルはICASAのデータ標準[101]として引き継がれ，XML形式での記述の検討が行われている．ICASAのデータ標準(図24)では，データ本体の前のヘッダ部分に，データ観測者，機関，連絡先，測定方法，備考，データ観測地点情報などが埋め込まれている．データ本体は固定長テキストで記録されている．

JAMFの農業モデル用のデータ関連のクラスは，**data**パッケージとしてまとめられている．農業モデル用のデータには，実行のための設定値を格納するための実行用データと，気象データやモデルの計算結果を格納するための結果データがある(図25)．実行用データには**ExecutionData**インタフェースと**ExecutionDataImpl**抽象クラスがある．同様に結果データ用には**ResultData**インタフェースと**ResultDataImpl**抽象クラスがある．両データ用のインタフェースとクラスともに**ModelData**インタフェースと，そのデフォルト実装を提供する**ModelDataImpl**抽象クラスを継承している．

アプリケーション開発者は，農業モデルが利用する値，日付，時系列データなどを，**ExecutionDataImpl**を継承したデータ用クラス(図22)に設定する記述を行う．

農業モデルのデータクラスは，真偽(**boolean**)，数値(**double**)，日付(**Calendar**)，時系列データ(**Sequence**)，文字列(**String**)の5つの型を標準で扱える．データクラスを拡張することにより，これ以外の型を扱うことも可能である．JAMFではこれらのデータを取得するための**isState()**，**getValue()**，**getDate()**，**getSequence()**，**getText()**メソッドを**ModelData**インタフェースで定義している．データを設定するメソッドは**is**または**get**を**set**にする．各データを特定するためのキーを引数としてとる．これらのメソッド以外の方法ではデータにアクセスすることはできない．これは一般的なデータのカプセル化の手法であるが，農業モデルをAMADISの構成要素とし，複数のモデルを連携させ，相互にデータのやりとりを行う場合に重要なことである．

データクラスで扱うデータは，データを設定したり取得したりする前に，データを特定するためのキーを**addValueElement()**や**addSequenceElement()**メソッドを用いて登録しなければならない．登録されていないキーでデータオブジェクトへアクセスを行うと例外が発生す

```
$WEATHER:KSAS
!The following data subset is desirable.
*GENERAL
@ PEOPLE
  Wagger,M.G. Kissel,D.
@ INSTITUTES
  Kansas State Univ.
@ CONTACTS
  Hunt,L.A. thunt@uoguelph.ca
@ NOTES
  Wind and dewpoint included; wind in m/s as average over 24h
  Data from DSSAT 3.0 file
@ DISTRIBUTION
  Use at will but acknowledge source. No secondary distribution
@ VERSION
  ICASA1.0 10-08-2006 (GH,JW,LAH;email)
! And additional data items can be added for comprehensive documentation:
@ METHODS
  Standard Met Station instruments
@ PUBLICATIONS
  None of direct application
@ FLAG FLAG_DETAILS
    0 All data ok
    1 SRAD estimated from sun hours

!And data subsets similar to the following are necessary for all stations.

*WEATHER_STATION:KSAS2004
@ NAME
  Example weather dataset
@ COUNTRY REGION LOCATION
  USA Kansas Ashland
@ LAT        LONG        ELEV  TAV  TAMP  TEMHT  WNDHT  CO2  CO2A
  37.1137  -90.4567  81     8.5  18.9  2.0     2.0    370  1.1
@ YEAR  DOY  SRAD  TMAX  TMIN  RAIN  FLAGW
  2004  1    11.5   1.4   -1.4  3.5    1
  2004  2    11.4   0.0   -2.1  2.5    0

[data series truncated]
=
```

図24 ICASAデータ標準による気象データ形式

データ本体の前のヘッダ部分に，データ観測者，機関，連絡先，測定方法，備考，データ観測地点情報などが記述されている．データ本体は固定長テキストで記録されている．

る．これは，データを表やグラフなどへの表示を自動化するために必要なことであり，プログラミングミスを防ぐためでもある．

結果データをファイルに出力するためのクラスは**broker.xml**や**data.xml**パッケージとしてまとめられている．XML形式ファイル（図26）として出力する場合は，Java Architecture for XML Binding（JAXB）[238]を利用した**Sequence2XML**クラスによりシリアライズ化されて出力される．逆にXMLファイルからデータオブジェクトへのデシリアライズも可能である．MetBrokerから取得した気象データには，メタデータとして気象観測地点情報しか付随しないので，ICASAデータ標準のような観測者や観測方法に関する情報は含まれない．

XML文書の構造は，XMLスキーマ（Schema）[285] metxml.xsdで定義されている．スキーマを見ることにより，XML文書に含まれる要素名，要素に含まれる属性，要素値の型，要素が現れる順番，要素や属性が必須かどうかを知ることができる．逆に，スキーマでユーザが生成したXMLファイルの妥当性を検証できる．また，XML Stylesheet Language Transformations（XSLT）[283]により，XML文書の全部または必要部分を抜き出して，別のXML文書やCSV形式に変換できる．

ResultData2CSVクラスによりCSV形式で，**Sequence2KML**クラスでGoogle Earth用のKeyhole Markup Language（KML）形式ファイルとして出力できる．また，**util**パッケージに含まれる**SequenceC**や**SequenceTM**クラスを利用することにより，時系列データをグラフや表として表示できる．

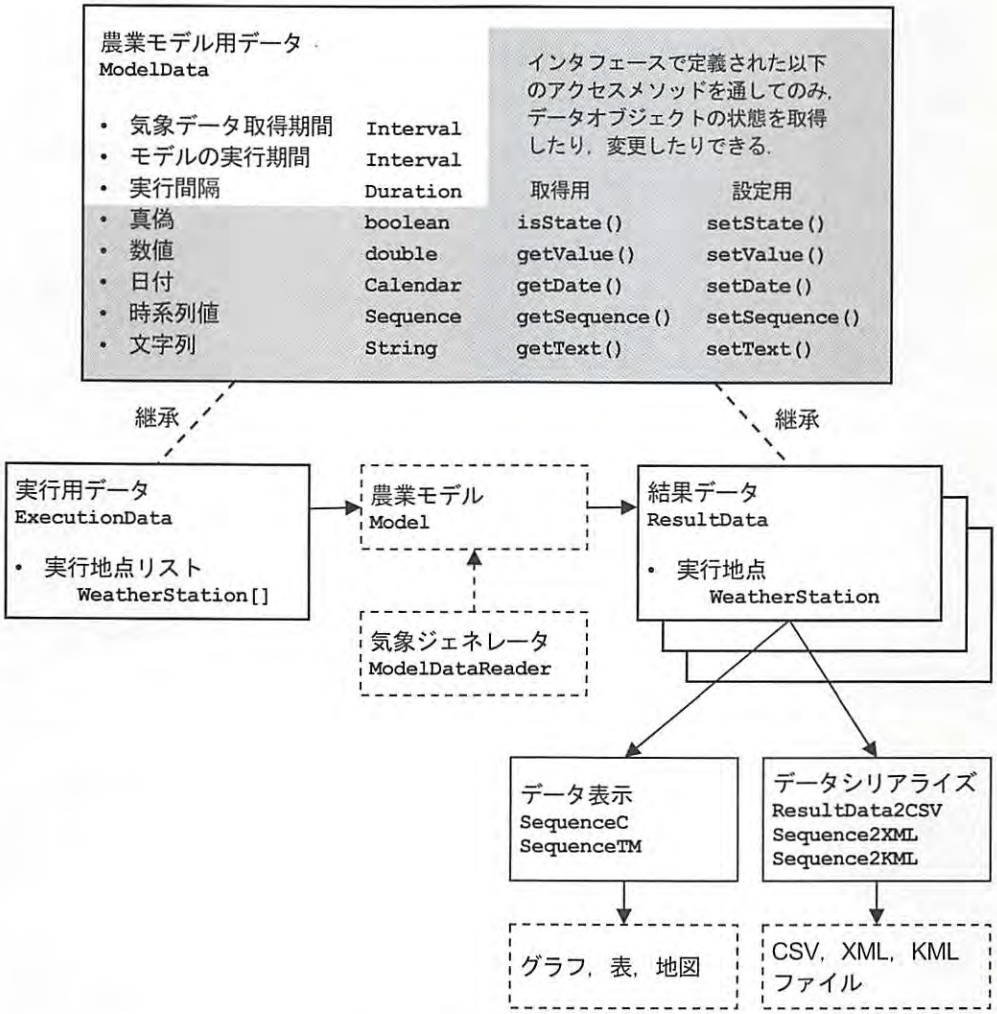

図 25　JAMF のデータ構造と関係

JAMF では農業モデル用データのための **ModelData** インタフェースを提供している．**ModelData** はデータとして，真偽，数値，日付，時系列値，文字列を扱える．**ModelData** 内のデータにアクセスするためには，インタフェースで定義された **isState()**，**getValue()**，**getDate()**，**getSequence()**，**getText()** メソッドを利用しなければならない．データを設定するメソッドは **get** を **set** にする．各データを特定するためのキーを引数としてとる．

3) モデル実行エンジン

JAMF のモデル実行の中心的役割を担っているのがモデル実行エンジン（図 27）である．モデル実行エンジン関連のクラスは **model** パッケージにまとめられている．

アプリケーションの実行ボタンがクリックされるなどして，農業モデルの実行が開始すると，モデル実行エンジン **ModelExecutor** クラスの **execute()** メソッドにより，初期化，気象ジェネレータから気象データ取得，終了条件を満たすまでのループ処理，結果出力の順に実行される．終了条件は農業モデルにより，ある変数が定められた値に達した場合や，ユーザにより設定された実行期間の終了日に達した場合，枯死条件などを満たして異常終了した場合などがある．1 回のループで進むモデル内時間は，作物モデルでは 1 日，病害モデルでは 1 時間であることが多い．

ModelExecutor クラスは DSSAT のメインプログラム（図 18）に相当し，作物モデル，病害虫モデル，気象モデルなどの計算部分である **Model** インタフェースを実装したクラスはモジュールに相当する．ループ処理中は **Model** インタフェースの **run()** メソッドが毎回実行される．DSSAT のループ処理では，モジュール内の計算や集計などの処理を個別に呼び出していたが，JAMF のモデル実行エンジンのループ処理では，簡単化のために各モデ

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<dataset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
SchemaLocation="http://localhost/metbroker/metbroker/schema/metxml.xsd">
<data>
 <source id="amedas">
  <name lang="ja">アメダス</name>
  <region id="08">
   <name lang="ja">茨城県</name>
   <station id="40336">
    <name lang="ja">つくば</name>
    <interval start="2010/1/1" end="2010/2/1"/>
    <duration id="daily">
     <name lang="ja">日別値</name>
    </duration>
    <element id="airtemperature">
     <name lang="ja">気温</name>
     <subelement id="Min" unit="℃">
      <name lang="ja">最低</name>
      <value date="2010/1/1">-2.7</value>
      <value date="2010/1/2">-2.3</value>
[data series truncated]
     </subelement>
     <subelement id="Max" unit="℃">
      <name lang="ja">最高</name>
      <value date="2010/1/1">8.2</value>
      <value date="2010/1/2">11.3</value>
[data series truncated]
     </subelement>
     <subelement id="Ave" unit="℃">
      <name lang="ja">平均</name>
      <value date="2010/1/1">2.6</value>
      <value date="2010/1/2">4.5</value>
[data series truncated]
     </subelement>
    </element>
    <element id="rain">
     <name lang="ja">雨量</name>
     <subelement id="Total" unit="mm">
      <name lang="ja">合計</name>
      <value date="2010/1/1">0.0</value>
      <value date="2010/1/2">0.0</value>
[data series truncated]
     </subelement>
    </element>
[data series truncated]
   </station>
  </region>
 </source>
</data>
</dataset>
```

気象観測地点情報

気象データ情報

最低気温

最高気温

平均気温

降水量

図26 JAMFの気象データ用のXML書式

MetBrokerから取得した気象データには，メタデータとして気象観測地点情報しか付随していないので，ICASAデータ標準のように観測者や観測方法に関する情報は含まれていない．

ルのループ内処理単位で呼び出しを行う．

モデルの計算処理関連で，アプリケーション開発者が作成する必要がある部分は，**Model**インタフェースを実装するクラスで，農業モデルの計算を行う**run()**メソッドのみである．農業モデル実行のための他の部分はJAMFが提供する機能が自動的に継承される．

4) ユーザインタフェース

基本的なJavaアプレット用のGUIコンポーネントは，Javaの開発環境Java Development Kit (JDK)の**java.awt**や**javax.swing**パッケージで提供されている．それらを拡張したり，組み合わせたりした農業モデル用のカスタムGUIコンポーネント用クラスは**gui**パッケージにまとめられている．

ユーザインタフェース開発のためのプログラム

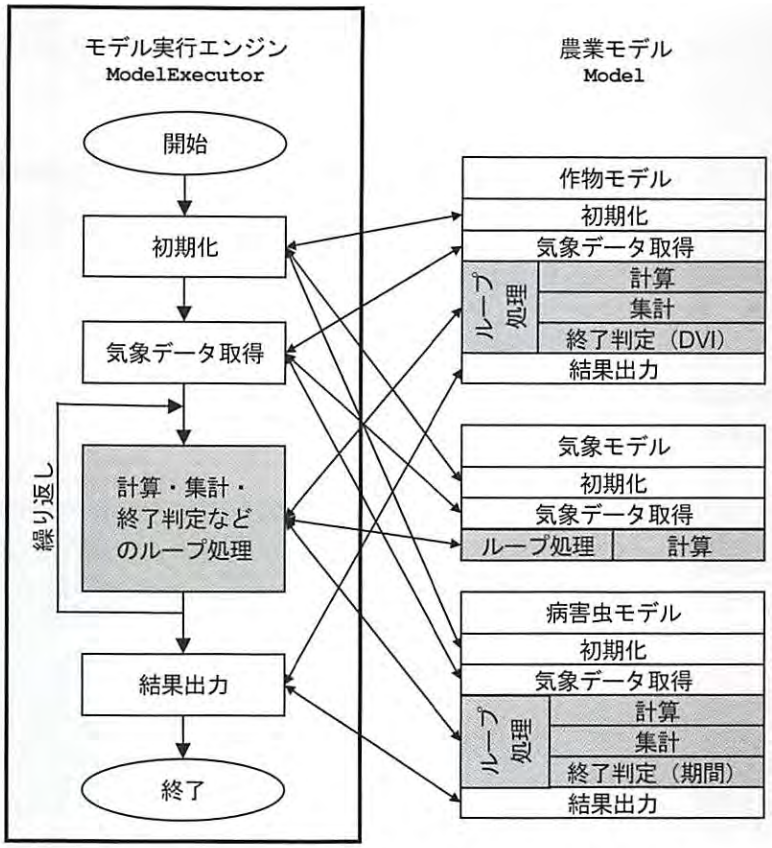

図 27　JAMF のモデル実行エンジンと各モデルの関係

農業モデルの実行が開始すると，モデル実行エンジンにより，初期化，気象データ取得，ループ処理，結果出力の順に実行される．ループ処理中は各農業モデルのループ処理用メソッドが毎回実行される．

は，画面上での表示位置やサイズの設定，マウス操作での選択項目の変更や，キーボード入力に対するイベント処理など，煩雑な作業が多い．JAMF では，NetBeans [236] や Eclipse [48] などの統合開発環境のような GUI による画面作成ツールを提供していないが，多くの農業モデルに対応できるデフォルトのメイン画面や結果表示画面を提供している．JAMF による農業モデルは，主にメイン画面，気象データ取得元設定画面，結果表示画面から構成される（図 23，図 28）．

(1)　メイン画面

農業モデルのメイン画面用 **AbstractModelUI** クラスは，タイトル欄，実行期間設定ボックス，気象データ設定ボタン，実行ボタンなどのデフォルトメイン画面コンポーネントのための抽象クラスであり，各ボタンに対するイベント処理も定義されている．各モデル用に継承したクラスで，パラメータ設定欄などを追加することは可能であるが，設定項目が多い場合にはパラメータ設定用画面を別に設けるようにしている．

(2)　結果表示画面

農業モデルの結果データのうち，開花日，収穫日などの日付や収量などの数値は表で，気象データや DVI などの時系列データはグラフや表で表示している．

表は JDK 標準の **JTable** クラスが提供されている．JAMF では，JAMF が扱う時系列データ用クラスを自動的に表に変換できるように拡張した **KTable** クラスと，複数の表を切り替えて表示したり，データをダウンロードできたりする機能を持つ **KTableFrame** クラスなどを提供している．

グラフは Java のグラフ用ライブラリ JFreeChart [185] を利用し，JAMF 用に拡張されたクラスが **gui.chart** パッケージにまとめられている．

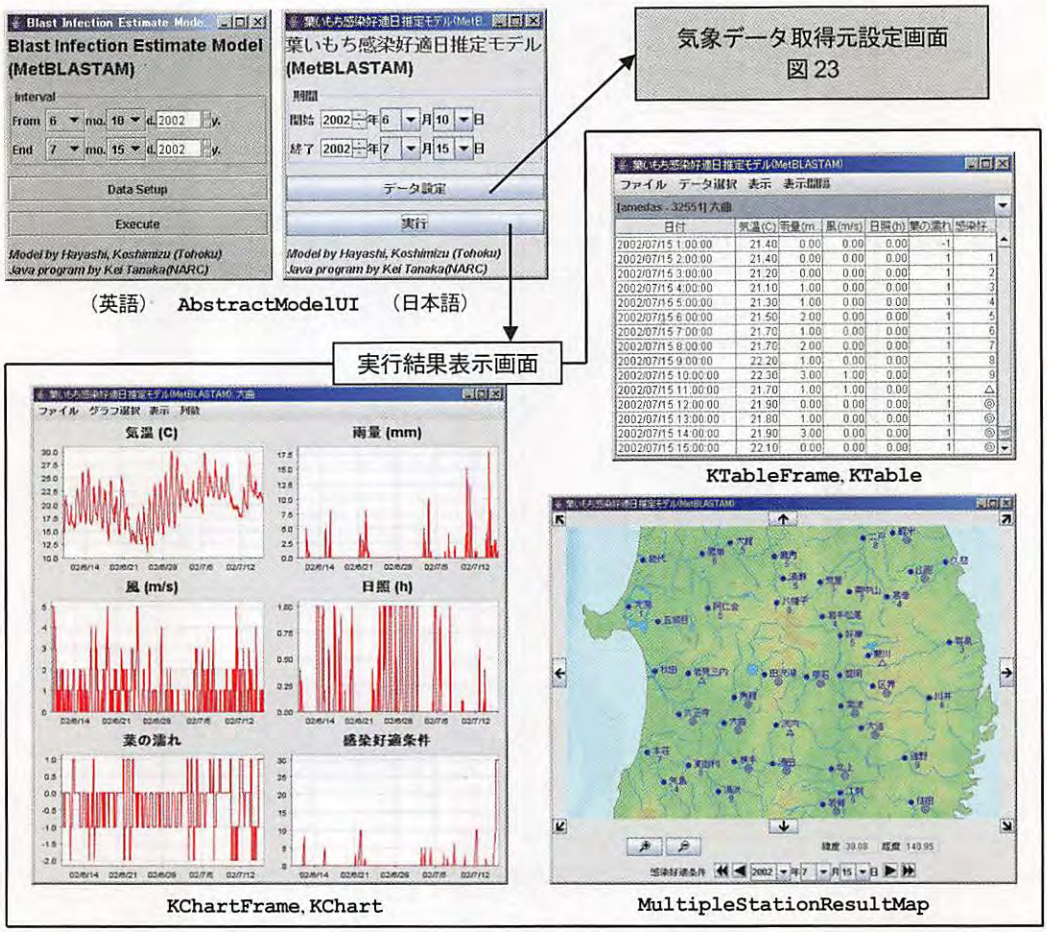

図 28 JAMF を利用して構築した農業モデルのアプリケーション実行画面

メイン画面で実行期間を設定し，気象データ取得元設定画面（図 23）で実行地点の設定を行った後，メイン画面の実行ボタンをクリックすると，モデルの実行結果が表，グラフ，地図で表示される．

各要素の **Courier** 体の文字列は，JAMF のインタフェース名やクラス名である．

JAMF では，複雑な JFreeChart を容易に利用できるように拡張した **JFCTimeSeriesChart** と **JFreeChartAttribute** クラスや，複数のグラフを選択表示するための **KChartFrame** クラスなどを提供している．

(3) 地図インタフェース

農業モデルの実行には入力として気象データや圃場に関するデータを必要とするため，入力データを基に計算した出力データと共に，地点と関連づけられるデータを扱うことが多い．農業モデルが扱う地点と関連づけられた数値や日付といった情報をユーザに直感的に示す方法として地図インタフェースがある．地図インタフェース上に一度に表示できる情報は，表やグラフで表示する場合と比べて制約を受けることがあるが，視認性において優れており，ユーザに短時間で情報を伝えることができる[259]．

また，地図インタフェースは地点情報の入力といった利用法でも操作性において優れている．MetBroker の気象観測地点を選択するために「データベース－（地域）－地点」と階層的に 2 または 3 段階の選択操作をしたり，緯度経度の値を入力したりするよりは，地図上でクリックして地点選択できる方が地点の位置を確認でき，素早く行える．

JAMF の地図インタフェース関連のクラスは **gui.map** パッケージにまとめられている．農業モデルの結果データの数値，日付，時系列データはともに **MultipleStationResultMap** クラスによって地図上に表示できる（図 28）．

JAMF 用の地図は ChizuBroker[178] から取得される．ChizuBroker は，指定された緯度経度を中心とする地図画像を提供する中間アプリケーションであ

る．ChizuBrokerはMetBrokerと同様，インターネット上で公開されている地図提供サービスとユーザが利用するアプリケーションの間にあって，ユーザの要求や地図画像を仲介する．開発者はアプリケーションで利用する地図画像をChizuBroker経由で取得するように開発することにより，様々な地図提供元から，統一的な手続きで地図画像を取得できるようになる．しかし，地図提供者は著作権の関係から地図画像の二次利用を禁止している場合が多いため，ChizuBrokerは対応する地図サービスを増やすことができず，成功しなかった．

(4)　多言語対応

農業モデルのアプリケーションが国内だけでなく，広く海外でも利用されるためには，画面の表示言語が日本語だけでなく，最低限，英語表示に切り替えられることが望ましい（図28）．

Javaにはユーザの実行環境（システムやWebブラウザの言語設定）に応じてアプリケーションの表示言語を自動的に切り替えることができる．多言語対応（Internationalization; i18n）のための**ResourceBundle**クラスが提供されている．JAMFでは，農業モデルでよく利用される言語リソースを用意した**ModelResourceBundle**クラスを**util**パッケージで提供している．

6.　アプリケーション構築

すでにJAMFを利用して20以上の農業モデルのアプリケーションを構築した（表9）．その経験からJAMFの機能と有用性の検証を行う．

FORTRANで書かれていた葉いもち感染好適日推定モデルBLASTAM[81]を，農業モデル用フレームワークJAMFを利用して，MetBLASTAM（図28）としてJavaに移植したときのプログラム行数（コメント行を除く）の比較を行った（表10）．JAMFを利用してMetBLASTAM用に新たに記述したプログラム行数は960行であった．このプログラムが呼び出したJAMF内のプログラム行数は17,361行であった．JAMFを利用することにより，新規のプログラム開発行数を全体の5%で済ますことができた．このことは，JAMFの利用が95%分の開発工程を省いたとも，また，17,361行分のプログラムを再利用できたともいえる．

さらに，農業モデルの機能ごとにプログラム行数を見ていくと，主要部分であるモデル計算部分は，新規に開発されたプログラムのうち66%を占め，開発者が主要部分の開発に専念できることを示している．データ取得に関するプログラムは，フレームワークの中で最も多い44%を占めるが，新規に開発したプログラムでは4%にすぎず，データ取得部分に関してはほとんど自動化できることを示している．画面表示部分の行数の割合はどちらも2番目であるが，フレームワークではGUIコンポーネントのためのプログラムが主であるのに対し，新規に開発されたプログラムでは，モデルの画面レイアウトや，画面を操作したときのイベント処理が主になっている．

フレームワークを利用する利点は開発行数の削減だけではない．気象データ取得のためにMetBrokerを利用するには，MetBrokerを利用する手順を学習する必要がある．また，結果を表示するための表，グラフや地図を利用するためにも，それぞれのGUIコンポーネントの利用方法を学習し，モデルのプログラムが保持するデータとそれらへの入力データの変換方法についても考慮する必要がある．フレームワークの利用により，これらの学習のための時間を節約できる．さらに，フレームワークが機能拡張された場合には，JARファイルの更新だけでその恩恵を受けることができる．

7.　考察

本章では，先ず3つの農業モデル開発グループそれぞれによる農業モデルの構造の研究と，開発言語選択の検討を行った後，農業モデルをWebアプリケーションとして実装するための農業モデル用フレームワークJAMFを構築し，その利用により農業モデルのプログラム開発効率と拡張性が高められたことを示した．

「成長の限界」に触発されて始まった農業モデルのプログラム開発において，1970年代（図1の初期まで）はDYNAMOやCSMPなどのシミュレーション言語が利用された．その後FORTRANでの開発環境が普及すると，1980年代の多くの農業モデルはFORTRANで実装されるようになった．1990年代（図1後半）に実装が始まった農業モデルのうち，それ以前のプログラム資産を引き継ぐ農

表9 JAMFを利用して構築した農業モデルのアプリケーション一覧[248]

農業モデル名	内容
SIMRIW[95,97,260]	水稲生育予測モデル
JAPONICA[79,246,80]	
RiceHeadingMaturity[229]	
RiceDVI[305]	
MetBLASTAM[81,250]	葉いもち感染好適日推定モデル
WheatHeading[145]	小麦の出穂期予測モデル
WheatRipening[130]	小麦の出穂期・成熟期予測モデル
WheatDuthie[47]	小麦赤かび病予察モデル
PearSugiura[233]	ナシの開花日・収穫期・果実体積予測モデル
PearDuthie[47]	ナシの黒星病・黒斑病発生予察モデル
PearMills[155,141]	
InsectDVR[117,291]	昆虫の世代推定モデル
WeedEmergence[247]	雑草発生予察モデル
WeedDamage[247]	雑草害予測モデル
SunRiseSet[118]	日出・日没・夜明・日暮計算
SolarRadiationKuwagata[146]	全天日射量推定モデル
LeafWetnessOhtani[193]	葉の濡れデータ推定モデル
LeafWetnessSuzuki[245]	
SWEB (Surface Wetness Energy Balance)[143]	
PaddyWaterTempKuwagata[131]	水田の水温データ推定モデル
MetBrokerDemo[133,255]	気象データ表示

表10 JAMFを利用して構築されたMetBLASTAMのプログラム行数

	JAMFを利用してMetBLASTAM用に新規に作成されたプログラム行数	MetBLASTAMが呼び出したJAMF内のプログラム行数
モデル計算部分	636行 (66%)	2296行 (13%)
データ取得部分	39行 (4%)	7612行 (44%)
画面表示部分	251行 (26%)	6689行 (39%)
多言語対応部分	34行 (4%)	764行 (4%)
合計	960行 (100%)	17361行 (100%)

MetBLASTAMが呼び出したJAMF内のプログラムとは，JAMFを利用しなければ開発する必要があったプログラムと考えることができる．または，複数の農業モデルを実装する場合に再利用されるプログラムと考えることもできる．

MetBLASTAMのプログラム行数は，MetBLASTAM用に新規に作成された960行＋MetBLASTAMから呼び出されたJAMF内の17361行で，18321行である．JAMFを利用することにより，MetBLASTAMのために新規に作成するプログラムは，全体の5.2%で済む．

プログラム行数はコメント行を除いた行数である．

業モデルではFORTRANが使い続けられた．それ以外の新規に実装された農業モデルは，オブジェクト指向プログラミングによる保守性の高さの利点を享受するために，C++やJavaなどのオブジェクト指向言語で実装されるようになった．AMADISの構成要素開発においてもJavaが広く用いられた．

農業情報分野における情報技術の利用の特徴は，農業機械などの他の分野に比べ，先端技術を試行錯誤しながら積極的に利用してきたことである．農業シミュレーションモデル実装に利用したプログラミング言語は，そのときの主流な言語が選択され，その変化は主要なプログラミング言語の推移（図12）と重なっている．増殖情報ベースプロジェクトの開始時には，登場して間もなかったがオブジェクト指

向言語として注目されつつあったJavaが標準開発言語として選択された．また，分散オブジェクトの実装に，登場間もないHORBやJava RMIを利用した．

本研究ではAMADISのシステム全体としての機能を検証するために，AMADISの構成要素として，プロトタイプとなるいくつかの農業モデルを短期間に実装する必要があった．参考文献に書かれた数式から農業モデルの計算部分を実装したものもあるが，多くはFORTRANやBASICで実装され，レガシー化していたプログラムのJavaへの移植によって実装された．もとのプログラムを利用し，Java Native Interface(JNI)などを利用したラッパープログラムで対応する方法もあったが，メソッドやデータのマッピングを個々の農業モデル用に行わなければならなかったため，短期間に確実に行える，モデル計算部分をJavaに移植する方法を選択した．

農業モデルの計算部分は個々のモデルで異なるが，農業モデルの実行の流れ（図20），モデル実行エンジン（図27），気象ジェネレータ（図21），データ構造(図25)，各種設定や結果表示のためのGUI(図28）に関するプログラムはほぼ共通している．これらのプログラムをまとめ，本研究の目的の1つである農業モデル用フレームワークJAMFを構築した．

農業モデル開発グループによる農業モデルは1990年代の同時期に影響し合いながら実装されたため，プログラム構造に共通点が多い．Wageningenグループの FSEドライバ（図15），IBSNATグループのDSSAT-CSM（図17），APSRUグループのAPSIMエンジン（図19）のいずれも，モデル実行エンジンとリンクした作物モデルや病害虫モデルのモジュールに対し，終了条件を満たすまで時間ループ内で計算，集計を繰り返し実行する．JAMFのモデル実行エンジンもこれらと同様な機能を持つように設計，実装を行った．

JAMFの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．特に，アプリケーション開発者に負担を強いることなく，国内外の多くの気象データベースを利用可能とするMetBrokerを利用していることが特徴である．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，ユーザにより観測されたデータを利用できる．

JAMFの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，設定画面や結果表示画面などのユーザインタフェースの自動的な構築に貢献する．オブジェクト指向言語のインタフェースを利用したカプセル化の特徴を生かし，データのアクセスは公開されたメソッドの利用に限定されている．これにより安全性と，ユーザインタフェース関連プログラムの再利用性を高めている．

JAMFを利用することにより，アプリケーション開発者は主にモデルの計算部分のプログラム開発のみで，農業モデルをWebアプリケーションとして実装可能になった．その効果は〈III.6〉（表10）に示した通りである．プログラム行数の割合で示すと，MetBLASTAMのために新規に開発したプログラム行数は全体の5.2%で，残りはJAMFのライブラリ内のプログラムが呼び出されていた．これは今後新しい農業モデルを実装する場合に，すべてを開発する場合の5%程度のプログラム記述量で済ますことができることを示している．

さらに，JAMFを利用することにより，MetBrokerアクセス，グラフ，表，地図などの複雑なGUIコンポーネント操作や，そこに表示する時系列データ操作といった処理を自動化できた．そのため，アプリケーション開発者がそれらについて学習する時間を省け，農業モデルのプログラム開発が容易になった．フレームワークであるJAMFが機能拡張された場合には，そのプログラムライブラリのJARファイルを更新するだけで，改良された機能を農業モデルアプリケーションに反映させることができる．特定の農業モデル用に機能拡張する必要があれば，個々の農業モデルのプログラムの中でライブラリ内のクラスに対する拡張をすることで対応できる．

JAMFを利用して約20の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た．また，開発したWebアプリケーションを数年にわたりインターネット上で公開し，いくつかの試験場での生育予測や病害虫発生予察に利用されてきたことにより，長期の安定運用を行えることが示された．

Ⅳ．AMADIS要素間連携手法

1．はじめに

農業モデルがAMADISの構成要素として機能し，相互に作用する連携を行うためには，分散オブジェクトとしてネットワークを通し，他の農業モデルとメッセージ交換を行えなければならない．JAMFを利用した農業モデルでは，モデル実行エンジンが中心となり，各農業モデル間の連携を実現している．各農業モデルとJAMFのモデル実行エンジンの間には，クライアントであるモデル実行エンジンが，サーバである農業モデルを呼び出す，クライアント・サーバ（Client-server; C/S）モデルの関係が構成されている．しかし，JAMFによりJavaアプレットとして実装された農業モデルを，セキュリティ対策がなされたネットワーク上に分散させて運用することは難しく，そのための改良が必要である．

ユーザインタフェースにおいては，JavaScriptとCSS（Cascading Style Sheet）を利用して動的なWebページを作成できるDHTMLの登場により，Javaアプレットや Flashを利用しなくても，表現力が高く，操作性の良いWebアプリケーションを構築できるようになった．同時にJavaの役割はJavaアプレットやスタンドアロンアプリケーションを構築することから，Javaサーブレットなどのサーバサイドプログラムを構築することに遷移していった．Javaはもともとネットワークや XML文書を扱うプログラム開発が容易だったために，Webアプリケーション開発言語としての不動の地位を確立していった．

Ajaxの登場は，画面遷移を必要としないWebアプリケーション構築を可能にした．それにより，スタンドアロンアプリケーションと同様な操作を行える，多くのWebアプリケーションが登場した．さらに，複数のWebサービスを組み合わせて，より魅力的なWebサービスを構築するマッシュアップ（Mashup）の手法が提案された．特にGoogleマップと位置情報を持つデータの組み合わせは効果的で，多くの優れたマッシュアップアプリケーションが構築された．

本章では，最初に農業モデル間のメッセージ交換手法について検討する．次に農業モデルにメッセージ交換機能を持たせるために，〈III〉で構築した農業モデル用フレームワークJAMFを改良する．また，農業モデルをJavaサーブレット化するのに合わせて，Ajaxを利用した優れた操作性や地図インタフェースなどを持たせる．最後に改良した農業モデル用フレームワークを利用して実アプリケーションを構築し，その有用性を示す．

2．メッセージ交換

AMADISの各要素はインターネット上に分散して存在しているので，要素間のリクエストや結果はメッセージ交換（Message Exchange）によって行われる．

JAMFの構築に利用したJavaは，登場時からネットワークやマルチスレッドを利用するためのクラスライブラリが含まれていたので，分散処理プログラムの開発に適していた．農業モデルをJavaアプレットとして実装していた当時には，メッセージ交換にはソケット，CORBA，Java RMIなどが検討され，主にJava RMIが利用されていた．Javaサーブレットによる Webアプリケーションとしての実装が中心となると，その後登場したSOAP，REST，GWT RPCなどが検討された．

図29に示したいくつかのメッセージ交換手法のサンプルプログラム示し，AMADISの構成要素間の通信に適した手法を選択する．サンプルプログラムでは，クライアントからサーバに対し，気象データ取得用リクエストがテキストまたは**StationMetRequest**オブジェクトとして送信され，サーバでMetBrokerから気象データを取得した後，サーバからクライアントに対し，テキストまたは**StationDataSet**オブジェクトとして送信される．メッセージとして送受信可能なオブジェクトの条件はシリアライズ化可能（**Serializable**イ

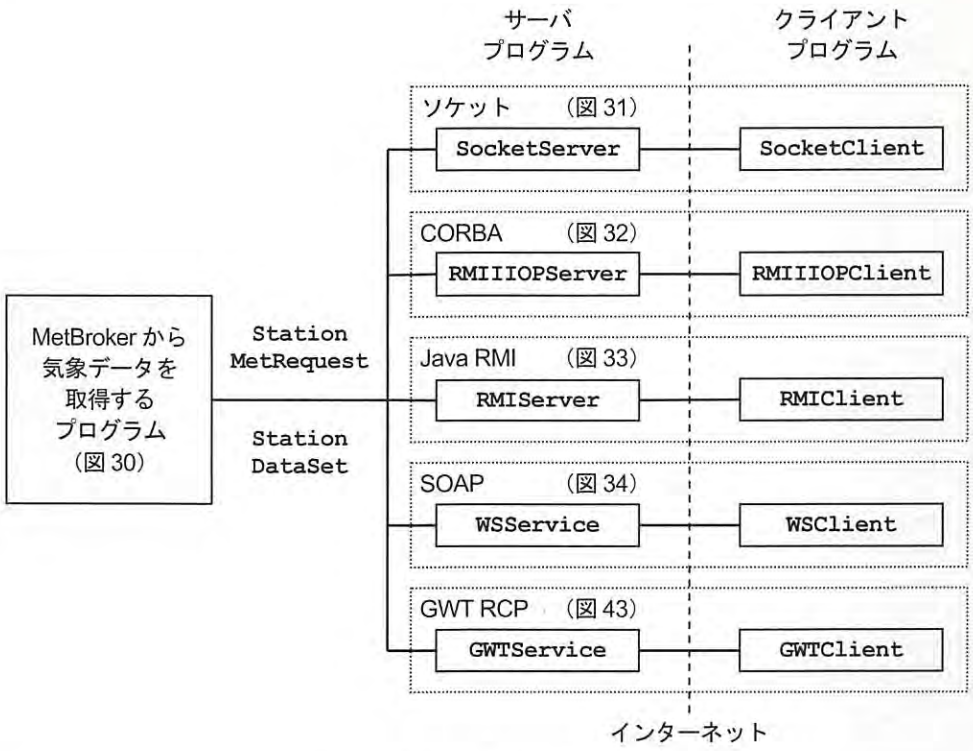

図29　様々なメッセージ交換手法

簡単のために，サーバでは計算などを行わずに，MetBroker から取得した気象データをそのままクライアントに返している．

ンタフェースを実装）であることである．メッセージをオブジェクトのまま送信できるか，テキストに変換して送信しなければならないかは，メッセージ交換手法による．図30はサンプルプログラムのサーバにおいて，MetBroker から気象データを取得するためのクラスである．気象データオブジェクト **StationDataSet** は **dumpDuration()** メソッドにより CSV 形式のテキストに変換できる．

1）ソケット

Java の登場時から存在する **java.net** パッケージに含まれるサーバ用の **ServerSocket** とクライアント用の **Socket** クラスを利用することにより，ソケットを利用したメッセージ交換プログラムを容易に開発できる．

ソケット通信では **int** や **double** などのプリミティブ型と文字列の他に，シリアライズ化可能（Serializable）なオブジェクトも **ObjectOutputStream** と **ObjectInputStream** クラスを利用することにより送受信可能である．他のメッセージ交換手法のようにサーバインタフェースを用いないため，クライアント側で受信したオブジェクトを利用する前に，型検査が必要である．また，サーバで複数のクライアントからの要求を同時に処理できるようにするためには，スレッドを利用する必要がある（図31）．

2）CORBA

Common Object Request Broaker Architecture（CORBA）は，ネットワーク上の分散環境でのオブジェクト間通信をサポートするためのミドルウェア基盤技術である[188,192]．CORBA は標準化機構 Object Management Group（OMG）によって標準化が行われている[190]．環境や言語に依存しないオープンな仕様であり，複数の環境や言語の混在した分散アプリケーションを構築できる．また，アプリケーションから利用可能なサービスが豊富に用意されており，高機能な分散アプリケーション構築が可能である．

CORBA の異なる ORB 間のメッセージ交換のプロトコルとして Internet Inter-ORB Protocol（IIOP）が利用されている．しかし，実際のネットワーク運

```
import java.text.*;
import java.util.*;
import net.agmodel.physical.*;
import net.agmodel.weatherData.*;

public class SimpleMetBroker{
  public static final String REQUEST = "start:2010/4/1,end:2010/5/1,elements:airtemperature,rain,
                                        resolution:daily,sourceid:amedas,stationid:40336";

  //気象データを返します.
  public StationDataSet getData(String request){
    try{
      return getData(createStationMetRequest(request));
    }
    catch(Exception e){
      e.printStackTrace();
      return null;
    }
  }

  //気象データを返します.
  public StationDataSet getData(StationMetRequest smr){
    MetBrokerHTTP broker = null;
    String sessionID = null;
    try{
      //MetBrokerに接続
      broker = new MetBrokerHTTP("www.agmodel.org", 80);
      Locale locale = Locale.getDefault();
      String language = locale.getLanguage();
      String country = locale.getCountry();
      String encoding = System.getProperty("file.encoding");
      sessionID = broker.getConnection("test", language, country, encoding);

      //気象データを取得
      return broker.supplyMetData(sessionID, smr);
    }
    catch(Exception e){
      e.printStackTrace();
      return null;
    }
    finally{
      if(broker != null){
        try{
          broker.disconnect(sessionID);
        }
        catch(Exception e){}
      }
    }
  }

  //文字列からStationMetRequestを生成します.
  public StationMetRequest createStationMetRequest(String request) throws ParseException{
    ...
    return new StationMetRequest(interval, elements, resolution, sourceID, stationID, true, false);
  }

  //デフォルトのStationMetRequestを生成します.
  public StationMetRequest createStationMetRequest(){
    ...
    return new StationMetRequest(interval, elements, resolution, sourceID, stationID, true, false);
  }
}
```

図30 メッセージ交換プログラムで利用される気象データ取得プログラム

コンパイル方法 `javac -classpath genericbroker.jar SimpleMetBroker.java`

```
//ソケットサーバ用クラス  SimpleSocketServer.java  ////////////////////////////////////////

import java.io.*;
import java.net.*;
import net.agmodel.weatherData.StationDataSet;

public class SimpleSocketServer{
  public static final int SOCKET_SERVER_PORT = 10000;              //受付ポート

  public static void main(String[] argv) throws Exception{
    ServerSocket serverSocket = null;
    try{                                                           //受付ポートを指定して,
      serverSocket = new ServerSocket(SOCKET_SERVER_PORT);         //サーバソケットを作成
      while(true){
        Socket socket = serverSocket.accept();                     //クライアントからの接続待機
        SocketServerMain serverMain = new SocketServerMain(socket);//スレッドでクライアント処理
        serverMain.start();
      }
    }
    catch(IOException e){e.printStackTrace();}
    finally{
      try{
        if(serverSocket != null)
          serverSocket.close();                                    //サーバソケットの終了
      }
      catch(IOException e){}
    }
  }
}

//クライアント処理用スレッドクラス
class SocketServerMain extends Thread{
  private Socket socket;
  private SocketAddress address;

  public SocketServerMain(Socket socket){
    this.socket = socket;
    this.address = socket.getRemoteSocketAddress();
  }

  public void run(){
    try{
      //クライアントからMetBrokerへのリクエストオブジェクトをメッセージとして取得
      ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
      StationMetRequest smr = (StationMetRequest)in.readObject();

      //MetBrokerから気象データ取得
      SimpleMetBroker mb = new SimpleMetBroker();
      str = mb.getData(smr);

      //クライアントへ気象データオブジェクトをメッセージとして送信
      ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
      oos.writeObject(sds);
    }
    catch(IOException e){e.printStackTrace();}
    finally{
      try{
        if(socket != null)
          socket.close();
      }
      catch (IOException e){}
    }
  }
}
```

図 31—1　Socket によるメッセージ交換のサーバとクライアントのコード

```
//ソケットクライアント用クラス SimpleSocketClient.java ////////////////////////////////////

import java.io.*;
import java.net.*;
import java.text.*;
import net.agmodel.weatherData.StationDataSet;

public class SimpleSocketClient{
  public static final int SOCKET_SERVER_PORT = 10000;                    //サーバの受付ポート

  public static void main(String args[]){
    String host = (args.length < 1) ? "localhost" : args[0];
    Socket socket = null;
    SocketAddress address = null;
    try{
      socket = new Socket(host, SOCKET_SERVER_PORT);                     //サーバに接続
      address = socket.getRemoteSocketAddress();

      //MetBroker へのリクエストオブジェクトをメッセージとしてサーバへ送信
      ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
      SimpleMetBroker mb = new SimpleMetBroker();
      StationMetRequest smr = mb.createStationMetRequest();
      out.writeObject(smr);

      //サーバから気象データオブジェクトをメッセージとして受信
      ObjectInputStream is = new ObjectInputStream(socket.getInputStream());
      StationDataSet sds = (StationDataSet)is.readObject();
      DateFormat df = new SimpleDateFormat("yyyy/M/d");
      String str = sds.dumpDuration(df, ",", "¥n", "日時", "-");
    }
    catch(Exception e){e.printStackTrace();}
    finally{
      try{
        if(socket != null)
          socket.close();                                                //サーバとの接続終了
      }
      catch (IOException e){}
    }
  }
}
```

図31—2 Socketによるメッセージ交換のサーバとクライアントのコード

```
コンパイル方法 javac -classpath .;genericbroker.jar SimpleSocket*.java
実行方法 （サーバ起動） java -classpath .;genericbroker.jar SimpleSocketServer
       （クライアント実行） java -classpath .;genericbroker.jar SimpleSocketClient
```

用環境ではHTTPしかファイアウォールを通過できないように設定されていることが多いため，個別対応の可能な限定されたユーザ向けのシステム以外では，IIOPを利用してインターネット上で稼働する分散システムを構築することは難しい．

JavaによるCORBAサーバやアプリケーション構築は，図32に示されるようにRemote Method Invocation over Internet Inter-ORB Protocol（RMI-IIOP）を利用することで可能である[237]．Java Remote Method Protocol（JRMP）をトランスポートとして使用してJava言語だけで開発することも，Internet InterORB Protocol（IIOP）を使用して他のCORBA対応言語を併用して開発することもできる．

RMIと異なるところは，リモートインタフェースの実装クラスを作成すること，rmicを-iiopオプションを指定して実行し，**_SimpleRMIIIOPImpl_Tie** と **_SimpleRMIIIOPInterface_Stub** クラスを生成しておくこと，rmiregistryの代わりにネームサービスObject Request Broker Daemon（ORDB）を起動することである．

CORBAの複雑さを解消するために，最初のJavaベースの分散オブジェクト技術としてHORB[89]が，電子技術総合研究所（現 産業技術総合研究所）で開発された．CORBAと比べると分散処理のためのコード変更が少ない．

```
//リモートインタフェース SimpleRMIIIOPInterface.java ////////////////////////////////////////

import java.rmi.*;
import net.agmodel.weatherData.*;

public interface SimpleRMIIIOPInterface extends Remote{
  StationDataSet getData(StationMetRequest smr) throws RemoteException;
}

//リモートオブジェクトの実装クラス SimpleRMIIIOPImpl.java ////////////////////////////////////

import java.rmi.RemoteException;
import javax.rmi.PortableRemoteObject;
import net.agmodel.weatherData.*;

public class SimpleRMIIIOPImpl extends PortableRemoteObject implements SimpleRMIIIOPInterface{
   public SimpleRMIIIOPImpl() throws RemoteException{
    super();                  //invoke rmi linking and remote object initialization
  }

  public StationDataSet getData(StationMetRequest smr) throws RemoteException{
    SimpleMetBroker mb = new SimpleMetBroker();
    return mb.getData(smr);
  }
}

//RMI サーバー用クラス SimpleRMIIIOPServer.java ////////////////////////////////////////////
//リモートオブジェクト実装のインスタンスを作成し，ネームサービスの名前にバインドする

import javax.naming.*;

public class SimpleRMIIIOPServer{
  public static void main(String[] args){
    try{
      //Step 1: Instantiate the SimpleRMIIIOP servant
      SimpleRMIIIOPImpl ref = new SimpleRMIIIOPImpl();

      //Step 2: Publish the reference in the Naming Service using JNDI API
      Context initialNamingContext = new InitialContext();
      initialNamingContext.rebind("SimpleRMIIIOPService", ref);
      System.out.println("SimpleRMIIIOP Server: ready");
    }
    catch(Exception e){
      e.printStackTrace();
    }
  }
}
```

図 32—1　RMI-IIOP によるメッセージ交換のサーバとクライアントのコード

3) Java RMI

Java Remote Method Invocation（Java RMI）は分散オブジェクト同士のメッソドのやりとりを，普通のオブジェクトのメソッドの呼び出しと同様に行えるようにする仕組みで，JDK1.1 から導入された[241]．内部ではソケットを利用しているが，通信に関する高度な知識がなくても，容易にネットワークアプリケーションを構築できるようになっている．

RMI を利用した分散システムは，リモートから呼び出すメソッドを定義した **Remote** インタフェースを継承したリモートインタフェース，それを実装した RMI サーバ用クラス，RMI クライアント用クラスの 3 つのプログラムから構成される（図 33）．RMI 関連のクラスは **java.rmi** パッケージにまとめられている．初期の頃は，RMI スタブコンパイラ rmic によって，あらかじめリモートオブジェクトのスタブ（Stub）とスケルトン（Skelton）クラスを生成しておく必要があったが，J2SE 5.0 から不要となった．

リモートオブジェクトの参照はネームサービス rmiregistry によって管理されている．RMI サーバが自分自身の存在するホストのネームサーバに対してリモートオブジェクトへの参照を登録することにより，クライアントはリモートオブジェクトの参照を得ることができ，リモートメソッドを呼び出すことができる．

```
//クライアントアプリケーション用クラス SimpleRMIIIOPClient.java ////////////////////////////////
//リモートメソッドgetData()を呼び出す

import java.text.*;
import javax.rmi.PortableRemoteObject;
import javax.naming.*;
import net.agmodel.weatherData.*;

public class SimpleRMIIIOPClient{
  public static void  main(String args[]){
    try{
      Context ic = new InitialContext();

      //STEP 1: Get the Object reference from the Name Service using JNDI call.
      Object objref = ic.lookup("SimpleRMIIIOPService");
      System.out.println("Client: Obtained a ref. to SimpleRMIIIOP server.");

      //STEP 2: Narrow the object reference to the concrete type and invoke the method.
      SimpleRMIIIOPInterface hi = (SimpleRMIIIOPInterface)PortableRemoteObject.narrow(objref,
                    SimpleRMIIIOPInterface.class);
      SimpleMetBroker mb = new SimpleMetBroker();
      StationMetRequest smr = mb.createStationMetRequest();
      StationDataSet sds = hi.getData(smr);
      DateFormat df = new SimpleDateFormat("yyyy/M/d");
      System.out.println(sds.dumpDuration(df, ",", "¥n", "日時", "-"));
    }
    catch(Exception e){
      e.printStackTrace( );
      return;
    }
  }
}
```

図32—2 RMI-IIOPによるメッセージ交換のサーバとクライアントのコード

```
コンパイル方法 javac -classpath .;genericbroker.jar SimpleRMIIIOP*.java
          rmic -classpath .;genericbroker.jar -iiop SimpleRMIIIOPImpl
実行方法  (ネームサービス起動)
        start orbd -ORBInitialPort 1050
        (サーバ起動)
        start java -classpath .;genericbroker.jar -Djava.naming.factory.initial=com.sun.jndi.cosnaming.
                    CNCtxFactory -Djava.naming.provider.url=iiop://localhost:1050 SimpleRMIIIOPServer
        (クライアント実行)
        java -classpath .;genericbroker.jar -Djava.naming.factory.initial=com.sun.jndi.cosnaming.
                    CNCtxFactory -Djava.naming.provider.url=iiop://localhost:1050 SimpleRMIIIOPClient
```

4) SOAP

Java API for XML-based RPC（JAX-RPC）は，図35に示されるようにJavaアプリケーションからWeb Services Description Language（WSDL）で記述されたJavaベースのWebサービスを呼び出すことを可能にした[239]．当初はRemote Procedure Call（RPC）に主眼が置かれて開発されていたが，Webサービスでは RPC以外にメッセージ交換でやり取りを行うことも多いため，メッセージ交換を含めたWebサービスを扱えるように拡張された．それが図34に示されるJava API for XML Web Services（JAX-WS）で，JavaでのSOAPの基盤として利用されている[242]．

SOAP[284]は状態を持たない，一方向のメッセージ交換用のプロトコルで，主にWebサービスにおけるメッセージ交換で利用されている．SOAPはv.1.2以前はSimple Object Access Protocolの略であったが，現在では固有名詞となっている．SOAPによるメッセージは，ルーティング，セキュリティ，トランザクションなどのためのメタ情報を格納するSOAP Headerと，主要な情報を格納するSOAP Bodyから構成されるSOAP Envelopeにより表現される．SOAPによる通信では，XML文書のメッセージをHypertext Transfer Protocol（HTTP），Simple Mail Transfer Protocol（SMTP），File Transfer Protocol（FTP）などのプロトコルで交換している．サービスを利用するクライアントとサーバがSOAPの生成，解釈エンジンを持つことにより，

```
//リモートインタフェース SimpleRMI.java ////////////////////////////////////
import java.rmi.*;
import net.agmodel.weatherData.*;

public interface SimpleRMI extends Remote{
  StationDataSet getData(StationMetRequest smr) throws RemoteException;
}

//RMI サーバクラス SimpleRMIServer.java ////////////////////////////////////
import java.rmi.*;
import java.rmi.registry.*;
import java.rmi.server.UnicastRemoteObject;
import net.agmodel.weatherData.*;

public class SimpleRMIServer implements SimpleRMI{
  public StationDataSet getData(StationMetRequest smr){
    SimpleMetBroker mb = new SimpleMetBroker();
    return mb.getData(smr);
  }

  public static void main(String args[]){
    try{
      SimpleRMIServer obj = new SimpleRMIServer();
      SimpleRMI stub = (SimpleRMI)UnicastRemoteObject.exportObject(obj, 0);

      //リモートオブジェクトのスタブをレジストリに登録
      Registry registry = LocateRegistry.getRegistry();
      registry.bind("SimpleRMI", stub);
      System.out.println("SimpleRMIServer ready");
    }
    catch(Exception e){e.printStackTrace();}
  }
}

//RMI クライアントクラス SimpleRMIClient.java ////////////////////////////////
import java.text.*;
import java.util.*;
import java.rmi.registry.*;
import net.agmodel.weatherData.*;

public class SimpleRMIClient{
  public static void main(String[] args){
    String host = (args.length < 1) ? null : args[0];
    try{
      Registry registry = LocateRegistry.getRegistry(host);
      SimpleRMI stub = (SimpleRMI)registry.lookup("SimpleRMI");
      SimpleMetBroker mb = new SimpleMetBroker();
      StationMetRequest smr = mb.createStationMetRequest();
      StationDataSet sds = stub.getData(smr);
      DateFormat df = new SimpleDateFormat("yyyy/M/d");
      System.out.println(sds.dumpDuration(df, ",", "¥n", "日時", "-"));
    }
    catch(Exception e){e.printStackTrace();}
  }
}
```

図 33　Java RMI によるメッセージ交換のサーバとクライアントのコード

コンパイル方法 javac -classpath .;genericbroker.jar SimpleRMI*.java
実行方法　(RMI レジストリ起動)
start rmiregistry -J-classpath -Jgenericbroker.jar
(サーバ起動)
start java -classpath genericbroker.jar -Djava.rmi.server.codebase=file:./ SimpleRMIServer
(クライアント実行)
java -classpath .;genericbroker.jar SimpleRMIClient

```
//Webサービス用クラス SimpleWS.java ////////////////////////////////////////////////////

package dissertation.ws;

import java.text.*;
import javax.jws.*;
import net.agmodel.weatherData.StationDataSet;

@WebService
public class SimpleWS{
  @WebMethod
  public String getData(String request){
    SimpleMetBroker mb = new SimpleMetBroker();
    StationDataSet sds = mb.getData(request);
    DateFormat df = new SimpleDateFormat("yyyy/M/d");
    return sds.dumpDuration(df, ",", "¥n", "日時", "-");
  }
}

//Webサービス用クラス SimpleWSServiceLauncher.java ////////////////////////////////////

package dissertation.ws;

import javax.xml.ws.Endpoint;

public class SimpleWSServiceLauncher{
  public static void main(String[] args){
    Endpoint.publish("http://localhost:8888/simplews", new SimpleWS()); //Web サービスの公開
  }
}

//Webサービスクライアント用クラス SimpleWSClient.java ////////////////////////////////////

package dissertation.ws;

public class SimpleWSClient{
  public SimpleWSClient() {
    SimpleWSService service = new SimpleWSService();          //Web サービスのポートを取得
    SimpleWS port = service.getSimpleWSPort();
    String result = port.getData(SimpleMetBroker.REQUEST);          //Web サービスの実行
    System.out.println(result);
  }

  public static void main(String[] args){
    new SimpleWSClient();
  }
}
```

図34 JAX-WSによるメッセージ交換のサーバとクライアントのコード

```
コンパイル方法 cd dissertation¥ws
        javac -classpath ..¥..;..¥..¥lib¥genericbroker.jar SimpleWS*.java
        wsgen -d ..¥.. -classpath ..¥..;..¥..¥lib¥genericbroker.jar dissertation.ws.SimpleWS
        cd ..¥..
サーバ起動   start java -classpath .;lib¥genericbroker.jar dissertation.ws.SimpleWSServiceLauncher
        wsimport -d . http://localhost:8888/simplews?wsdl
実行方法     java -classpath ..¥.. dissertation.ws.SimpleWSClient
```

異なる環境間でのオブジェクト呼び出しが可能になる．SOAPのJavaによる実装としてApache Axis 2[13]がある．

Webサービスは，異なるアーキテクチャのアプリケーションの相互運用を行えるという利点がある．しかし，情報を特定のプラットフォームに依存しないXML形式で，通信をファイアウォールによって遮られないHTTPで行っているため，Java RMIと比べると動作速度は遅くなる．特に送受信するデータの構造が複雑かつ大量になると，XMLの組み立て，パースといった処理でのパフォーマンス低下が問題となる．

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.6 in JDK 6. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.6 in JDK 6. -->
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ws.dissertation/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://ws.dissertation/" name="SimpleWSService">
<types>
  <xsd:schema>
    <xsd:import namespace="http://ws.dissertation/"
                schemaLocation="http://localhost:8888/simplews?xsd=1"></xsd:import>
  </xsd:schema>
</types>
<message name="getData">
  <part name="parameters" element="tns:getData"></part>
</message>
<message name="getDataResponse">
  <part name="parameters" element="tns:getDataResponse"></part>
</message>
<portType name="SimpleWS">
  <operation name="getData">
    <input message="tns:getData"></input>
    <output message="tns:getDataResponse"></output>
  </operation>
</portType>
<binding name="SimpleWSPortBinding" type="tns:SimpleWS">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"></soap:binding>
  <operation name="getData">
    <soap:operation soapAction=""></soap:operation>
    <input>
      <soap:body use="literal"></soap:body>
    </input>
    <output>
      <soap:body use="literal"></soap:body>
    </output>
  </operation>
</binding>
<service name="SimpleWSService">
  <port name="SimpleWSPort" binding="tns:SimpleWSPortBinding">
    <soap:address location="http://localhost:8888/simplews"></soap:address>
  </port>
</service>
</definitions>
```

図 35　図 34 で生成された WSDL ファイル

5) REST

Representational State Transfer（REST）は HTTP を使って通信を行う手法で，HTTP の **GET** メソッドを使って URL にアクセスすると XML が返ってくるものが REST と呼ばれている[52]．ある URL にアクセスして XML を得るという流れは，Web ブラウザが URL にアクセスして HTML を得るのと同じである．REST は Web ブラウザの Ajax や，クライアントアプリケーションから使われることが多いが，サーバ間のシステム連携でも利用できる．REST の最大の特徴は，Web ブラウザに URL を入力すれば動作確認できることであり，テスト用アプリケーション開発の手間を省略できる．

JavaScript Object Notation（JSON）は REST とほぼ同じだが，レスポンスとして XML ではなく JavaScript のオブジェクト表記法を使ったデータを返す．JavaScript の **eval** 関数で簡単にオブジェクトに変換できる．XML と比べるとデータの転送量が少なくて済み，XML をパースする必要がないので高速に処理できる．JavaScript に特化したデータ形式であるが，単純な処理で書き出しや読み込みを行えるため，他の多くのプログラミング言語で扱える．

6) 農業モデル連携手法

AMADIS の各要素はネットワーク上に分散し，各サーバのプラットフォームは様々であるため，要素間の連携は，情報を特定のプラットフォームに依存しない XML 形式で，通信をファイアウォールによって遮られない HTTP で行う，REST によるメッセージ交換を基本とする（図 36）．ただし，ユーザインタフェースを Google Web Toolkit 〈IV.3.3〉(5)〉

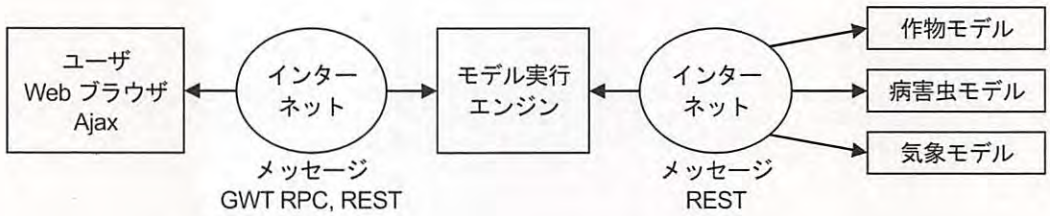

図36 AMADIS 要素間のメッセージ交換

モデル実行エンジンとユーザインタフェース間のメッセージ交換にはRESTまたはGWT RPCを用いる．モデル実行エンジンと各モデル間のメッセージ交換にはRESTを用いる．

で開発する場合には，モデル実行エンジンとユーザインタフェース間の通信にGWT RPCを用いる．

農業モデル連携のためのRESTによるメッセージ交換を実現するために，パラメータを指定したURLによるリクエストに対し，結果をXML形式で返すRESTアプリケーションとしての機能が必要である．これは，Javaアプレットとして実装された農業モデルを，Javaサーブレットとして実装し直すことで実現できる．

3. サーバサイドアプリケーション

Webアプリケーションの操作性を低下させる原因は，通信や画面遷移による待ち時間である．Javaアプレットのようにクライアントサイドで計算を行うと，農業モデルのプログラムや，農業モデルの計算を行うために大量の気象データをサーバからクライアントへ送信する必要があり，農業モデルの実行時間はユーザの通信環境や計算機環境に大きく依存していた．Javaサーブレットようにサーバサイドで計算を行う場合，高性能なサーバで計算して実行時間を短縮し，少量の結果データのみを送信することにより通信時間を減らせる．また，Ajaxは非同期通信と，DHTMLによる動的な表示内容の更新により，画面遷移の回数を減らすか，不要とする．

JAMFを用いてJavaで実装した農業モデルのプログラムを利用しつつ，通信や画面遷移による待ち時間を短縮できるのが，サーブレットとしての農業モデルWebアプリケーションである．また，アプリケーションの出力をXML形式にすれば，RESTアプリケーションとすることができる．

1) Javaサーブレット

Javaサーブレット[244]は，サーバでデータを処理し，動的なWebページを生成するサーバサイドプログラムである．サーブレットの登場以降，サーバ側でJavaプログラムを稼動させるサーバサイドアプリケーション開発言語としてのJavaの利用が急速に進んだ．

サーブレットでもWebページのHTMLは生成できるが，静的な部分も含めてすべてを`print()`メソッドで出力するのは煩雑で，拡張性，保守性が低下する．MVCモデルの点からも，サーブレットではデータの入出力処理のみ行い，画面生成はJSPが担当するように機能を分離するのが適当である．

JSP[243]を利用すると，静的な部分はHTMLで，動的な部分はJavaで記述でき，画面デザインとプログラム開発の棲み分けが容易になる．さらに，プログラム部分をライブラリとして，タグの記述で呼び出せるようにしたJSP Tag Library（JSTL）を利用することにより，Web画面開発効率を高められる．

サーブレットやJSPを実行するのはWebコンテナであり，Apache Tomcat[9]などがある．TomcatはWebサーバとしての機能も持つが，Apache HTTP Server[8]ほど速くなく，頑健でないため，実運用ではApacheと連携させて利用される．ApacheがHTTPリクエストを受け付け，必要に応じてTomcatにリクエストが渡されて処理される．

Tomcatへサーブレットを配備するには，webappsディレクトリにWeb Application Archive（WAR）ファイルを置くだけである．WARファイルには，サーバプログラム，画面表示用のJSPファイル，CSSファイルや画像ファイル，サーブ

レットの動作を規定する配備記述子（Deployment Descriptor）web.xml ファイルなどが含まれ，ZIP 形式でアーカイブされている．

2）地図サービス

圃場の位置と結びつけられる農業モデルは，気象観測地点の選択や結果データの表示において地図インタフェースが必須である〈III.5.4）(3)〉．地図データの取得には商用またはオープンソースの地図サービスを利用することになる．前者は使用料が高額であり，後者としては 1996 年に Minnesota 大学によって開発された MapServer[144] がある．MapServer はオープンソースのソフトウェアであるが，商用の地図サービスに劣らない機能を持っている．Open Geospatial Consortium（OGC）が公開している地図画像提供標準インタフェースに対応していれば，ユーザのデータを自由に地図上に表示できる[186]．しかし，機能の豊富さと自由度の高さがかえって利用対象ユーザを限定してしまっていた．

インターネットの利用が一般的になってくると，Web 上で場所を検索して地図を表示するサービスが登場し，その利便性から同様の地図サービスが多数出現した．初期の地図サービスには表 11 のようなものがある．これらは主に国内の地図や地図ソフトを開発していた会社によるものである．これらのサービスでは，Common Gateway Interface（CGI）のパラメータで緯度，経度，ズームレベルを指定して，目的とする場所を表示する機能に限定されていた[147]．しかも，Web ページから地図画像のみを取り出して再利用することが禁じられていた．

その後，表 12 のような Web 検索サービス大手による地図サービスが相次いで公開された．これらは，サービスをプログラムから利用するための API が公開されていることが共通しており，そこが表 11 の地図サービスと大きく異なるところである．縮尺に応じて表示する地図解像度の選択や，表示に必要な範囲の地図の切り出しなどは，地図サービス側で行われる．Google マップや Yahoo! 地図では UI コンポーネントとして組み込むこともできる．そのため，この API を利用することにより，地図を利用

表 11　初期の主な Web 地図サービス

提供者	サービス名	サービス開始時期	URL
サイバーマップ・ジャパン	マピオン	1997 年 4 月	http://www.mapion.co.jp/
インクリメント P	マップファン	1997 年 7 月	http://www.mapfan.com/
国土地理院	ウォッちず	2000 年 7 月 地形図閲覧システムとして	http://watchizu.gsi.go.jp/
ゼンリンデータコム	いつもガイド	2001 年 4 月 Do-map として	http://www.its-mo.com/

表 12　API が公開されている主な Web 地図サービス

提供者	サービス名	サービス開始時期	URL（下段は API ドキュメント）
Minnesota 大学	MapServer	1996 年	http://mapserver.org/ http://mapserver.org/documentation.html
Google	Google マップ	2005 年 2 月	http://maps.google.co.jp/ http://www.google.com/apis/maps/
	Google Earth	2005 年 6 月	http://earth.google.co.jp/ http://earth.google.com/kml/kml_tut.html
Yahoo!	Yahoo! 地図	2005 年 11 月	http://map.yahoo.co.jp/ http://developer.yahoo.com/maps/
Microsoft	bing 地図	2005 年 12 月	http://bing.com/maps/ https://connect.microsoft.com/bingmapsapps

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<meta http-equiv="content-style-type" content="text/css" />
<meta http-equiv="content-script-type" content="text/javascript" />
<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=..."
                                          type="text/javascript"></script>
<script type="text/javascript">
var map, icon;
function init(){
  map = new GMap2(document.getElementById('map'));
  map.addControl(new GLargeMapControl());
  map.addControl(new GMapTypeControl());
  map.addControl(new GOverviewMapControl());
  map.setCenter(new GLatLng(latitude, longitude), scale);
  createIcon();
  getStationData();
}

function createIcon(){       //目印の準備
  var baseIcon = new GIcon();
  baseIcon.shadow = 'img/shadow.png';
  baseIcon.iconSize = new GSize(12, 20);
  baseIcon.shadowSize = new GSize(22, 20);
  baseIcon.iconAnchor = new GPoint(6, 20);
  baseIcon.infoWindowAnchor = new GPoint(5, 1);
  icon = new GIcon(baseIcon);
  icon.image = 'img/pin.png';
}

function getStationData(){                    //地点データを非同期に取得（Ajaxによる処理）
  var request = GXmlHttp.create();
  request.open('GET', 'station.xml', true);
  request.onreadystatechange = function(){
    if(request.readyState == 4){
      var xmlDoc = request.responseXML;
      var stations = xmlDoc.documentElement.getElementsByTagName('stations');
      for(var i=0; i<stations.length; i++){
        var latitude = parseFloat(stations [i].getAttribute('latitude'));
        var longitude = parseFloat(stations [i].getAttribute('longitude'));
        var point = new GPoint(longitude, latitude);
        var name = stations [i].getAttribute('name');
        var marker = createMarker(point, name);
        map.addOverlay(marker);
      }
    }
  }
  request.send(null);
}

function createMarker(point, name){                    //地点データをもとに目印を生成
  var marker = new GMarker(point, icon);
  GEvent.addListener(marker, 'click', function(){
    marker.openInfoWindowHtml(name);
  });
  return marker;
}
</script>
</head>
<body onload="init()">
<div id="map" style="width: 600px; height: 400px"></div>
</body>
</html>
```

図37　Googleマップ上に地点情報を表示するHTMLとJavaScript

するアプリケーション構築が容易となった．

3) Google

Google は，Google マップや Google Earth などの農業モデル用の地図インタフェースとして利用できる Web サービスを含め，多くの Web サービスを提供している[68]．それらの API は公開されているので，Web アプリケーションから利用したり，気象データや農業モデルの結果データと組み合わせて，マッシュアップアプリケーションを構築したりできる．また，Ajax アプリケーション構築を Java で行える Google Web Toolkit を提供している．

(1) Google マップ

Google マップ[66]は Google が提供する地図サービスである．これは地図，航空写真，地図＋写真などの方式で表示でき，全世界から道路名表示レベルまでの 20 段階程度のズーム表示を行うことができる．

Google マップ以前の地図検索サイトの地図は，地図の周りに配置された移動ボタンや縮尺変更ボタンをクリックすることにより，新しい地図画像をリロードしていた．この方法は，毎回処理が終わるまで操作が中断してしまうため，ユーザにストレスを感じさせることがあった．Google マップが公開されると，Java アプレットや Flash といった Web ブラウザ用のプラグインのインストールを不要としながら，マウスによる直感的な操作で，滑らかに地図が表示されることが注目された．さらに，それが Ajax という既存の技術のみで実現されているということでも注目された．

Google は同時に Google ローカルという，Google が持つ膨大な地点情報を含むデータと地図を結びつけるサービスも提供した．さらに，ユーザが Google マップに目印，線や領域を自由に追加し，他のユーザと Web 上で共有したり，重ねて表示したりできるマイマップや，ルート検索，ストリートビューなどのユーザの要求に応える地図用の新機能を追加し続けている．

Google マップの API[67]は公開されているので，ユーザのデータを地図上に目印として表示したり，目印をクリックしたときに吹き出しを表示させたりできる．吹き出しに表示する内容は HTML 形式で記述するため，他の Web ページへのリンクや画像を入れることもでき，編集も容易である．Google マップを利用して地点情報を地図上に表示する HTML と JavaScript の主要部分を図 37 に示す．

Google マップを利用した地図インタフェースは，Java サーブレット版農業モデル[248]で利用するために開発された．図 38 は MetBroker が扱う気象観測地点選択に利用した実行画面である．Google マップの地図を表示することに関しては，サンプルプ

図 38　Google マップを利用した気象観測地点選択用地図インタフェース

http://pc105.narc.affrc.go.jp/metbroker/

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Document>
  <name>MetBroker 気象観測地点</name>
  <open>1</open>
  <Style id="orangeN">
    <IconStyle>
      <scale>0.5</scale>
      <Icon>
        <href>img/orange.png</href>
      </Icon>
    </IconStyle>
    <LabelStyle>
      <scale>0</scale>
    </LabelStyle>
  </Style>
  ・・・
  <StyleMap id="orange">
    <Pair>
      <key>normal</key>
      <styleUrl>orangeN</styleUrl>
    </Pair>
    ・・・
  </StyleMap>
  ・・・
  <LookAt>
    <longitude>135</longitude>
    <latitude>38</latitude>
    <range>8000000</range>
    <tilt>0</tilt>
    <heading>0</heading>
  </LookAt>
  <Folder>
    <name>アメダス</name>
    <LookAt>
      <longitude>134.0</longitude>
      <latitude>33.5</latitude>
      <range>3031665.0</range>
    </LookAt>
    <Folder>
      <name>北海道</name>
      <LookAt>
        <longitude>142.5929946899414</longitude>
        <latitude>43.471500396728516</latitude>
        <range>664794.6089286384</range>
      </LookAt>
      <Placemark>
        <name>宗谷岬</name>
        <description>宗谷岬 (11001) <a href="http://.../">気象データ</a></description>
        <Snippet></Snippet>
        <LookAt>
          <longitude>141.93499755859375</longitude>
          <latitude>45.52000045776367</latitude>
          <range>10000</range>
        </LookAt>
        <StyleUrl>#orange</StyleUrl>
        <Point>
          <coordinates>141.93499755859375,45.52000045776367,0</coordinates>
        </Point>
      </Placemark>
      ・・・
    </Folder>
    ・・・
  </Folder>
  ・・・
</Document>
</kml>
```

アイコンスタイル情報

初期視点情報

データベース情報

地域情報

地点情報

階層構造

図 39　Google Earth 上に地点を表示させるための KML データ

ログラムを大きく変更する必要はないが，取得したXMLデータを処理し，目印や吹き出しの内容に反映させるためには，JavaScriptによるDocument Object Model（DOM）プログラミングが必要である．サンプルプログラムでは目印の表示のみであったが，パスやポリゴンを描画するAPIも提供されている．年々，安定性と処理速度の向上が図られているが，地図上に大量の目印を表示するときには，処理に長時間かかったり，動作が不安定になったりするため，その対策が必要になる．

(2) Google Earth

Google Earth[69]は衛星写真や航空写真の地図データをストリーミング表示するためのスタンドアロンのソフトウェアである．衛星写真の上に目印やポリゴンを描画したり，吹き出しを表示したりできることはGoogleマップと同様であるが，データに高度や日時を関連づけて表示したり，鳥瞰する視点を設定したり，複数のKMLファイルや，ネットワークを通じて得られる様々な種類のデータをレイヤとして重ねて表示できる．また，Googleマップは JavaScriptのプログラムによりデータを加工し，地図上に表示するデータレイヤを構築するのに対し，Google EarthではKML形式のデータによってデータレイヤを構築する．Googleマップとの連携も図られ，Googleマップのページにある「リンク」で表示されるURLに「&output=kml」を追加すると，Googleマップで表示している内容をGoogle Earthで表示できる．

図39はGoogle Earth上に地点情報を表示するためのKMLデータの一部である．KMLデータはXMLのルート要素である**Document**要素の中の**Folder**要素が階層構造になって構成されている．各**Folder**要素は**name**要素（名前），**LookAt**要素（視点）を持っている．末端の**Folder**要素はさらに**Placemark**要素（目印）を持っている．**Placemark**要素は目印の**name**要素（名前），**coordinates**要素（緯度，経度，高度），**StyleUrl**要素（アイコン形式）や**description**要素（クリックされたときに表示される吹き出しの内容）が記述された各要素を持っている．

図39のような地点情報を繰り返していくことにより，Google Earthを利用したMetBrokerの気象観測地点選択用地図インタフェース（図40）を構築できる．地球上に多数描画された点が気象観測地点であり，クリックすると気象観測地点の情報と，気象データ取得アプリケーションへのリンクが含まれた吹き出しが表示される．MetBrokerが扱う2.3万ヶ所の気象観測地点（GD-DR&TRを除く）を表示するKMLファイルは33万行になり，ファイルサイズは17MByteであった．ファイルサイズが大きい場合やアイコン画像を含める場合は，ZIPファ

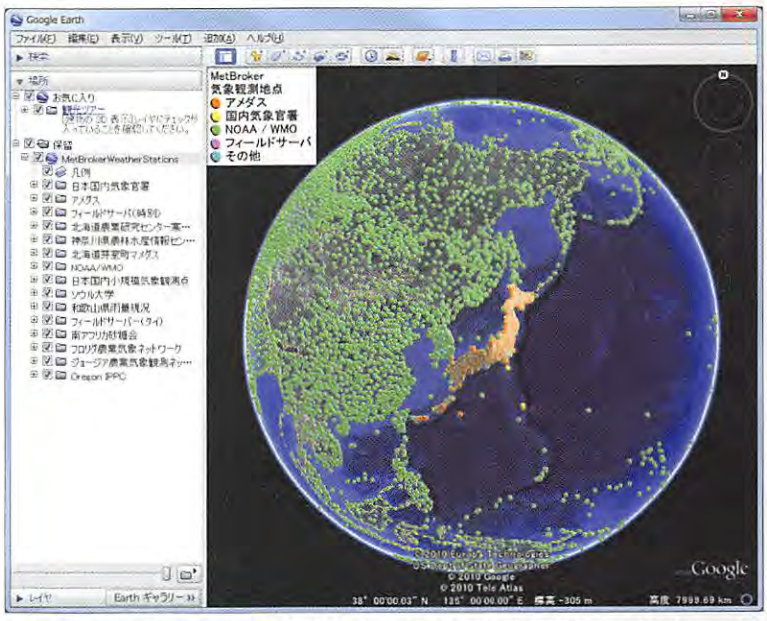

図40　Google Earthを利用した気象観測地点選択インタフェース

http://pc105.narc.affrc.go.jp/metbroker/kml/metbroker_ja.kmz

イルとして圧縮してまとめられたKMZファイルとして扱う．この場合，ファイルサイズは1.2MByteになった．KMLファイルやKMZファイルはWebページのリンク先に指定することにより，クリックしたときにGoogle Earthを起動して表示させることができる．

Keyhole Markup Language（KML）はKeyhole（2004年にGoogleにより買収された）によって仕様が決められたXML形式のデータである．2008年4月に地理情報を記述するOGC標準の1つとして承認された．詳しい仕様がドキュメント[70,187]として公開されているので，ユーザが独自のデータを加工して，地図上に目印を表示したり，目印をクリックしたときに吹き出しを表示したりできる．

Google Earth上に表示できるオブジェクトは目印以外に，パスやポリゴンもある．さらにCADで作成した3Dオブジェクトの表示も可能であり，3Dモデリングツールである Google SketchUp[71]も提供されている．大量の目印を表示させることも問題なく，MetBrokerが扱う2.3万ヶ所の気象観測地点を表示しても，データの読み込みから表示までも数秒であった．

最新のデータやユーザの要求に応じたデータを動的にKMLデータとして生成し，Google Earthで表示するためにネットワークリンク機能がある．図41はネットワークリンク機能を利用したKMLファイルである．**NetworkLink**要素は**Url**要素を持ち，そこに設定されているURLに，KMLデータの読み込み時や，設定された時間ごとにアクセスする．URLで指定されたリンク先のサーバでKMLデータが動的に生成され，Google Earthに返されて表示される．

KMLデータの**Folder**要素内のデータの有効期間を設定する**TimeSpan**要素を利用することにより，Google Earthで時系列データをアニメーション表示できる．図42はユーザが指定した期間の気象データをMetBrokerから取得し，動的にKMLデータを生成するサービスによって，Google Earth上にアニメーション表示している画面である．

(3) Ajax

Ajax[60]とはAsynchronous JavaScript + XMLという既存の技術の組み合わせに対して付けられた名前である．AjaxはGoogleマップ上にユーザのデータを表示する（図37）ときのデータ取得手法としても利用されるため，Googleマップを利用したアプリケーション構築には欠かせない技術である．Ajaxの動的な画面表示は次のように実現されている．

1．JavaScriptのXMLHttpRequestを利用した非同期通信により，XMLデータをサーバから取得する．
2．JavaScriptでCSSやDOMを動的に変更して，データをWebページに反映させる．

(4) マッシュアップ

近年，マッシュアップ（Mashup）と呼ばれる，ネットワーク上の複数のソースから提供されるコンテンツを組み合わせた複合型ソフトウェアが注目されて

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<NetworkLink>
  <name>2007/4/1 0:00-2007/4/2 0:00 気温 (1hour, amedas)</name>
  <LookAt>
    <longitude>134</longitude>
    <latitude>33</latitude>
    <range>3753490</range>
    <tilt>0</tilt>
    <heading>0</heading>
  </LookAt>
  <flyToView>1</flyToView>
  <Url>
    <href>http://pc105.narc.affrc.go.jp/metbroker/kml/weather-sequence-
data.kmz?element=airtemperature&amp;duration=1hour&amp;area=amedas&am
p;interval=2007/4/1 0:00-2007/4/2 0:00&amp;lang=ja</href>
  </Url>
</NetworkLink>
</kml>
```

図41 Google Earth上に動的に生成するデータを表示させるためのKML

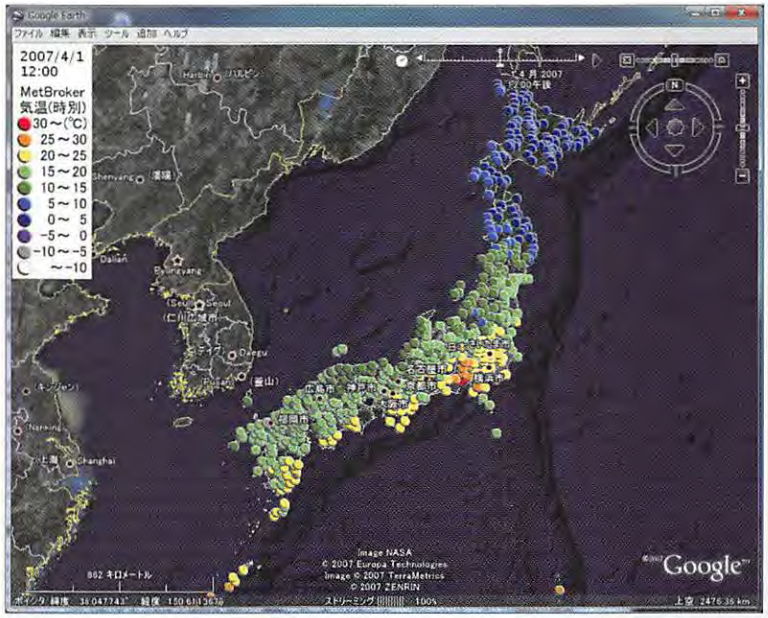

図 42　Google Earth 上に表示したアニメーション表示可能な時別気温値

http://pc105.narc.affrc.go.jp/metbroker/kml/

上部のスライドバーによりアニメーション表示の設定や表示日時の選択を行える．

いる．マッシュアップブームの先駆けとなったのは，Google の地図情報と Craigslist の不動産情報を組み合わせた HausingMaps であった．すでに Web 上に存在していた情報が，地図と組み合わさることによって，数段利用しやすくなり，多くの新たなユーザを獲得した有名な例である[156,207]．

地図と気象データを組み合わせた Weather Bonk や Global Cloud Map などのようなマッシュアップサイトもいくつかある．図 38，図 40，図 42 の Google マップや Google Earth と MetBroker から取得したデータを結びつけたアプリケーションもマッシュアップアプリケーションの一例である．農業モデルで利用されるデータの多くは位置情報を持ち，随時更新されるため，地図インタフェースと組み合わせて，マッシュアップアプリケーションとしての構築に適している．

マッシュアップアプリケーション構築は，無料で入手できる開発環境や実行環境を利用し，Web アプリケーションとデータを結びつけることで行える．そのため，IT 技術者だけでなく誰でもアイデア次第で開発を始めることができる．自分のデータ，もしくは既存の公開データを Google マップ上に表示するには，Google マップの API に合わせて，データを加工するだけである．しかし，データを他サイトから取得して JavaScript で扱うためには，同一生成元ポリシー（Same Origin Policy）のセキュリティ制限を回避する工夫が必要である．

(5)　Google Web Toolkit

最新の Web アプリケーション構築において，大規模な JavaScript のコードや Ajax コンポーネントの作成，再利用，保守が必須となっている．しかしスクリプト言語である JavaScript による大規模な開発の作業効率は良くない．この問題を解決できる Ajax アプリケーション構築フレームワークが Google Web Toolkit（GWT）である[73]．

HTML と JavaScript で開発していた Ajax アプリケーションの Web 画面の作成を，GWT ではすべて Java で開発できる．Java のコードは GWT によりクロスコンパイルされ，主要なブラウザに対応した JavaScript のコードに変換されるため，ブラウザによる違いを考慮したプログラムが不要となる．また，生成された JavaScript は最適化されているため，開発者が直接 JavaScript で記述するより，ファイルの読み込み時間，処理時間の双方でパフォーマンスの向上が期待できる．

よく利用される UI コンポーネントは標準で提供されているが，GWT 用ライブラリを組み込めば，

Googleマップやグラフなどの複雑なUIコンポーネントを利用できるようになる．また，各自で開発したUIコンポーネントをJARファイルにパッケージ化しておけば，別のWebアプリケーションで再利用できる．Javaで開発されたUIコンポーネントがコンパイルされると，HTMLで記述されたWebページが生成されるのではなく，DOMによるWebページを構築するためのJavaScriptコードが生成される．

GWTでは，必要なライブラリファイルのコピー，コンパイル，Javadocによるドキュメントの生成，JARファイルやWARファイルの生成，サーバへの配備までの，Webアプリケーション構築のためのすべての作業をApache Ant[10]を利用して行う．作業内容はXML形式のビルドファイルに記述される．

(6) GWT RPC

GWTではクライアントとサーバ間の通信に，Ajaxアプリケーションで標準的に利用されるJSON，XMLに加え，GWT Remote Procedure Call（RPC）を提供している．GWT RPCを利用すると，シリアライズ化可能なJavaのオブジェクトをそのまま通信で利用でき，メッセージ交換が容易になり効率化できる．GWT RPCはJava RMIと似ていて，リモートサービスと非同期通信用の2つの

```
package net.agmodel.gwt.model.client.rpc;

import com.google.gwt.user.client.rpc.*;

public interface ModelService extends RemoteService{
  Result[] execute(String execID, ExecutionParameter[] exParams, String lang);
}

//モデルを実行するサーブレットの非同期インタフェース  ModelServiceAsync.java  ////////////////////////

package net.agmodel.gwt.model.client.rpc;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface ModelServiceAsync{
  void execute(String execID, ExecutionParameter[] exParams, String lang,
                  AsyncCallback<Result[]> callback);
}

//モデルを実行するサーブレットクラス  ModelServiceImpl.java  ////////////////////////////////////////

package net.agmodel.gwt.model.server;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

public abstract class ModelServiceImpl extends RemoteServiceServlet implements ModelService{
  public Result[] execute(String execID, ExecutionParameter[] exParams, String lang){
    ...
    List<Result> resultList = new ArrayList<Result>();
    try{
      ...
      //exParamの個数回モデルを繰り返し実行する．
      for(int i=0; i<exParams.length; i++){
        ...
        try{
          ExecutionData data = createExecutionData(exParams[i]);
          Result result = executeModel(data, broker, exParams[i]);  //気象データを取得してモデルを実行
          resultList.add(result);
        }
        catch(Exception e){}
        ...
      }
    }
    catch(Exception e){}
    return resultList.toArray(new Result[0]);
  }
}
```

図43 GWT RPCによるサーバとクライアント間通信のコード

ExecutionParameterやResultオブジェクトの配列がクライアントとサーバ間でやりとりされる．

インタフェースと，その実装クラス（図43）を作成し，サーブレットとして登録することにより利用できる．ブラウザからリモートメソッドを呼び出すと，GWT RPCは引数で渡されたオブジェクトをシリアライズし，サーバ上のメソッドを呼び出す．また，メソッドからの返り値のオブジェクトをクライアントのためにデシリアライズする．

4) サーブレット版農業モデル用フレームワーク

農業モデルをRESTアプリケーション化し，近年，主流になったAjaxアプリケーションとするために，JAMFを利用してJavaアプレットとして実装した農業モデルを，Javaサーブレットとして実装し直す必要がある．

農業モデルの実行の流れ（図20）は，基本的にJavaアプレットと変わらない．大きな変更点は，気象データの取得やモデルの計算をサーバで行うようにすることと，ユーザインタフェースの構築をアプレット用のSwingコンポーネントから，HTMLコンポーネントを利用して行うようにすることである．

気象データの取得やモデルの計算はサーバで行うように変更されるが，それらのプログラム自体の変更は必要ない．それらを呼び出していたJavaアプレット用プログラム**AbstractModel**（表6）を，Javaサーブレット用プログラム**AbstractModelServlet**に取り替えるだけである．さらに，Javaのプラットフォーム非依存の特徴〈III.2.5)〉により，サーバでの実行のためにプログラムを修正したり，コンパイルし直したりする必要もない．

JavaアプレットのJAMFとして構築したように，Javaサーブレット用の農業モデル実装フレームワークをJava Agricultural Model Framework for Servlet (JAMF-S) として構築する．JAMF-Sはサーブレットでも利用可能なJAMFのプログラムはそのまま利用したが，サーバでの実行用のモデル実行エンジン，グラフ画像ファイル生成用のサーブレット，UIコンポーネントとそのイベント処理用プログラム，RPC用プログラムなど(表13)が新たに開発された．

サーブレットの開発にはTomcatのServlet API，JSPとJavaScriptを組み合わせて開発する方法と，すべてをGoogle Web Toolkitで開発する方法の2つがある．表14にJAMF-Sを利用して実装された農業モデルが示されている．これらのうちのほとんどは，初期に実装されたServlet API版である．GWT登場後に，MetBrokerDemoとSIMRIWのGWT版が実装された．

Servlet APIによるサーブレットでは，ユーザからのリクエスト**HttpServletRequest**オブジェクトは，サーバの**HttpServlet**クラスを継承したクラスの**doGet()**または**doPost()**メソッドで受け取って処理される．その結果は画面遷移後の新しいWeb画面をJSPで生成して表示される．Web画面に非同期通信を行い，取得したデータを動的にUIコンポーネントに反映させるJavaScriptプログラムを埋め込めば，Ajaxアプリケーションにできる．また，結果を**ServletOutputStream**オブジェクトに書き出すことにより，グラフ画像や，ZIPファイルなどのバイナリファイルとして出力できる．

GWTはAjaxアプリケーション構築用フレームワークであるので，クライアント用のWeb画面も含めて，すべてJavaで開発でき，コンパイルすればAjaxアプリケーションとなる．GWTではサーバプログラムも，クライアント画面もすべてJavaで開発できるため，Javaのオブジェクト指向プログラミングの特徴を生かし，既存のプログラムに対する拡張を容易に行える．デフォルト機能のみを実装した農業モデルの半完成品をGWTのライブラリとしてまとめることにより，デフォルト機能と異なる部分のみの開発で，サーブレット版農業モデルを実装できるようになる．

いずれの方法で開発しても，サーブレットプログラムの働きは，配備記述子web.xmlファイルの設定で制御する．

5) MetXML

MetBrokerは多くの気象データベースのデータに統一的な手法でアクセスすることを可能にし，気象データを利用する農業アプリケーション構築において，気象データ取得のためのプログラム開発コストを大幅に削減した．しかし，アプリケーション開発者にはJavaやSOAPによる開発スキルが必要であったため，MetBrokerを利用しているアプリケー

表13 JAMF-Sのパッケージ構成（主要部）

分類	パッケージ名	パッケージの内容
サーバ	**server**	サーバ実行版のモデル実行エンジン，RPCの実装クラス，グラフ画像やXMLファイルを生成するサーブレットクラスを提供
ユーザインタフェース	**ui**	Webページ用UIコンポーネントを提供
	event	UIコンポーネント用のイベントとイベントハンドラを提供
	resource	多言語対応のためのリソースを提供
RPC	**rpc**	RPCのインタフェースと非同期インタフェースを提供
	data	RPCでの通信に利用するためのシリアライズ可能なデータ用クラスを提供

表14 JAMF-Sを利用して実装した農業モデルのアプリケーション一覧[248]

農業モデル名		内容
SIMRIW[95,97,260]	（Servlet API版）	水稲生育予測モデル
	（GWT版）	
RiceHeadingMaturity[229]		
RiceDVI[305]		
MetBLASTAM[81,250]		葉いもち感染好適日推定モデル
WheatHeading[145]		小麦の出穂期予測モデル
WheatDuthie[47]		小麦赤かび病予察モデル
SolarRadiationKuwagata[146]		全天日射量推定モデル
MetBroker[133,255,258,261]	MetBrokerDemo（Servlet API版）	気象データ表示
	MetBrokerDemo（GWT版）	
	MetBroker-i	気象データ表示（携帯電話用）
	MetBroker Taglib	気象データ表示用JSP用カスタムタグ
FieldServer[56,254]		フィールドサーバの観測データ，撮影画像表示
ImageChangeDetection[120,256]		フィールドサーバで撮影した間欠画像の変化抽出

ション開発者は未だに多くない．一方，XML形式でデータを扱うことはすでに一般的になっており，様々なプログラミング言語を利用してXMLデータを処理することは難しくなくなっている．

そこで，必要な気象データの種類などをURLのパラメータ文字列（図45）で指定してXMLデータ（図46）として取得できるRESTアプリケーションMetXML（図44）を開発した．MetXMLを利用することにより，農業アプリケーション開発者は，URLパラメータ構築のための文字列処理とXMLデータ処理のプログラム開発のみで，気象データを扱えるようになる．MetXMLはURLのパラメータの設定により，気象データだけでなくMetBrokerで利用できるデータベースや気象観測地点などの情報も取得できる．また，XML形式だけでなく，CSV形式でのデータ取得やHTMLによるグラフや表によるデータ出力もできる．

(1) MetBroker ICS

MetBroker ICS（図47）はMetBrokerから取得できる気温，降水量のデータをGoogleカレンダー[65]に表示するためのWebサービスである．Googleカレンダーは定期的に，MetBroker ICSからiCalendar形式のテキストデータを取得して，最新の気象データをGoogleカレンダーの予定欄に表示する．

iCalendar形式のデータは，Internet Engineering Task Forceのカレンダー・スケジュール作業部会で定められた，スケジュール管理用アプリケーションで利用されるテキスト形式の標準書式[36]である．

(2) 携帯電話用アプリケーション

Webブラウザでの利用を想定して開発されたサーブレットも，簡易な表示インタフェースを用意するだけで，携帯電話での利用に適したWebアプ

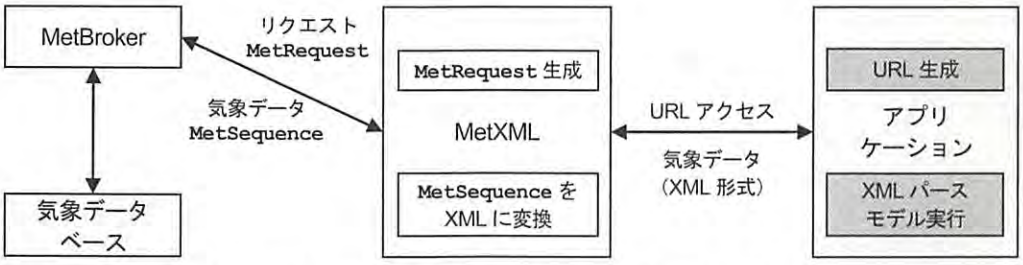

図 44　MetXML と農業モデル用アプリケーションとの関係

MetXML が MetBroker とやりとりを行うため，農業モデル用アプリケーション開発者は灰色部分の気象データ取得用 URL 生成と XML パースのプログラムを開発するのみで気象データを取得できる．

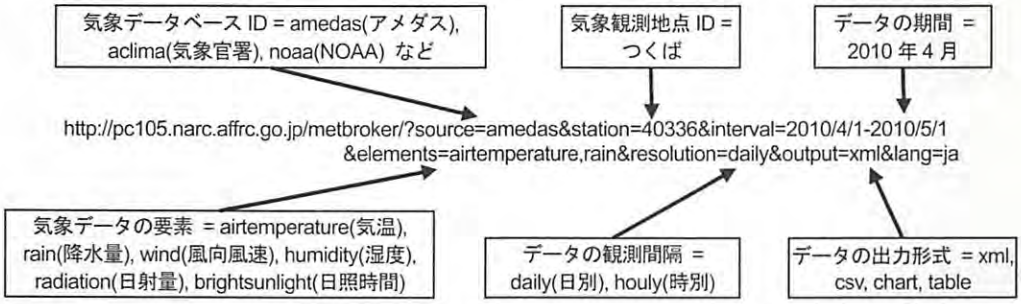

図 45　MetXML から気象データを取得する URL パラメータ

この URL にアクセスすることにより，アメダスの気象観測地点つくばの 2010 年 4 月の気温と降水量の日別値が MetBroker から取得され，XML 形式で返される．

http://pc105.narc.affrc.go.jp/metbroker/

リケーションにするができる（図 48）．

携帯電話用の農業モデルは，画面サイズやキー操作の制約から表示内容や操作性をパソコン用と同等にすることは難しい．しかし，携帯電話は圃場での作業中に利用するための端末として，電源やネットワーク環境の面から適当である．

4. SIMRIW を利用した水稲栽培可能性予測支援ツール

本節では，〈III〉，〈IV〉で構築した農業モデル用フレームワーク JAMF と JAMF-S（以降，JAMF とのみ記す）を利用した実アプリケーションを構築し，JAMF の有効性を示す．

実アプリケーションとして水稲栽培可能性予測支援ツールを構築する．水稲栽培可能性予測支援ツールは，水稲生育予測モデルによって全球を対象とした栽培可能性を提示するツールである．このツールの構成は図 49 に示されるように，ツール全体を構築するために，JAMF を利用したいくつかの農業モデルの実装，REST や GWT RPC によるアプリケーション間のメッセージ交換が必要であるため，JAMF の機能を利用することによる有効性を示すのに適している．

水稲生育予測モデルとして SIMRIW〈IV.4.2〉を選択した．SIMRIW は水稲生育予測モデルとしてよく知られたモデルで，国内での利用事例も多い．必要とする気象データが気温と日射量だけであり，地域パラメータを必要とせず，潜在成長のみを対象とするため，本章で扱うモデルとしての規模や必要データ数が適当であることが選択理由である．

1）背景

GEOSS[62] では，各国が連携して様々な地球観測データや予測データを蓄積し，これらのデータを統融合して人類に有益な情報に変換することをめざしており，国際的に共通な利用ニーズの 1 つが農業分野となっている．

日本での GEOSS の推進母体である DIAS[42] には，

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<dataset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="http://pc105.narc.affrc.go.jp/
                  metbroker/metbroker/schema/metxml.xsd">
  <data>
    <source id="amedas">
      <name lang="ja">アメダス</name>
      <region id="08">
        <name lang="ja">茨城県</name>
        <station id="40336">
          <name lang="ja">つくば</name>
          <interval start="2010/4/1" end="2010/4/30"/>
          <duration id="daily">
            <name lang="ja">日別値</name>
          </duration>
          <element id="airtemperature">
            <name lang="ja">気温</name>
            <subelement id="Min" unit="℃">
              <name lang="ja">最低</name>
              <value date="2010/4/1">1.5</value>
              <value date="2010/4/2">8.3</value>
              <value date="2010/4/3">4.5</value>
              <value date="2010/4/4">1.4</value>
              ...
            </subelement>
            <subelement id="Max" unit="℃">
              <name lang="ja">最高</name>
              <value date="2010/4/1">21.2</value>
              <value date="2010/4/2">20.4</value>
              <value date="2010/4/3">15.2</value>
              <value date="2010/4/4">11.0</value>
              ...
            </subelement>
            <subelement id="Ave" unit="℃">
              <name lang="ja">平均</name>
              <value date="2010/4/1">12.5</value>
              <value date="2010/4/2">13.5</value>
              <value date="2010/4/3">9.1</value>
              <value date="2010/4/4">6.8</value>
              ...
            </subelement>
          </element>
          <element id="rain">
            <name lang="ja">雨量</name>
            <subelement id="Total" unit="mm">
              <name lang="ja">合計</name>
              <value date="2010/4/1">0.0</value>
              <value date="2010/4/2">4.0</value>
              <value date="2010/4/3">0.0</value>
              <value date="2010/4/4">0.0</value>
              ...
            </subelement>
          </element>
        </station>
      </region>
    </source>
  </data>
</dataset>
```

スキーマ情報

気象観測地点情報

気象データ情報

最低気温

最高気温

平均気温

降水量

図46 図45のURLにアクセスすると返されるXML形式の気象データ

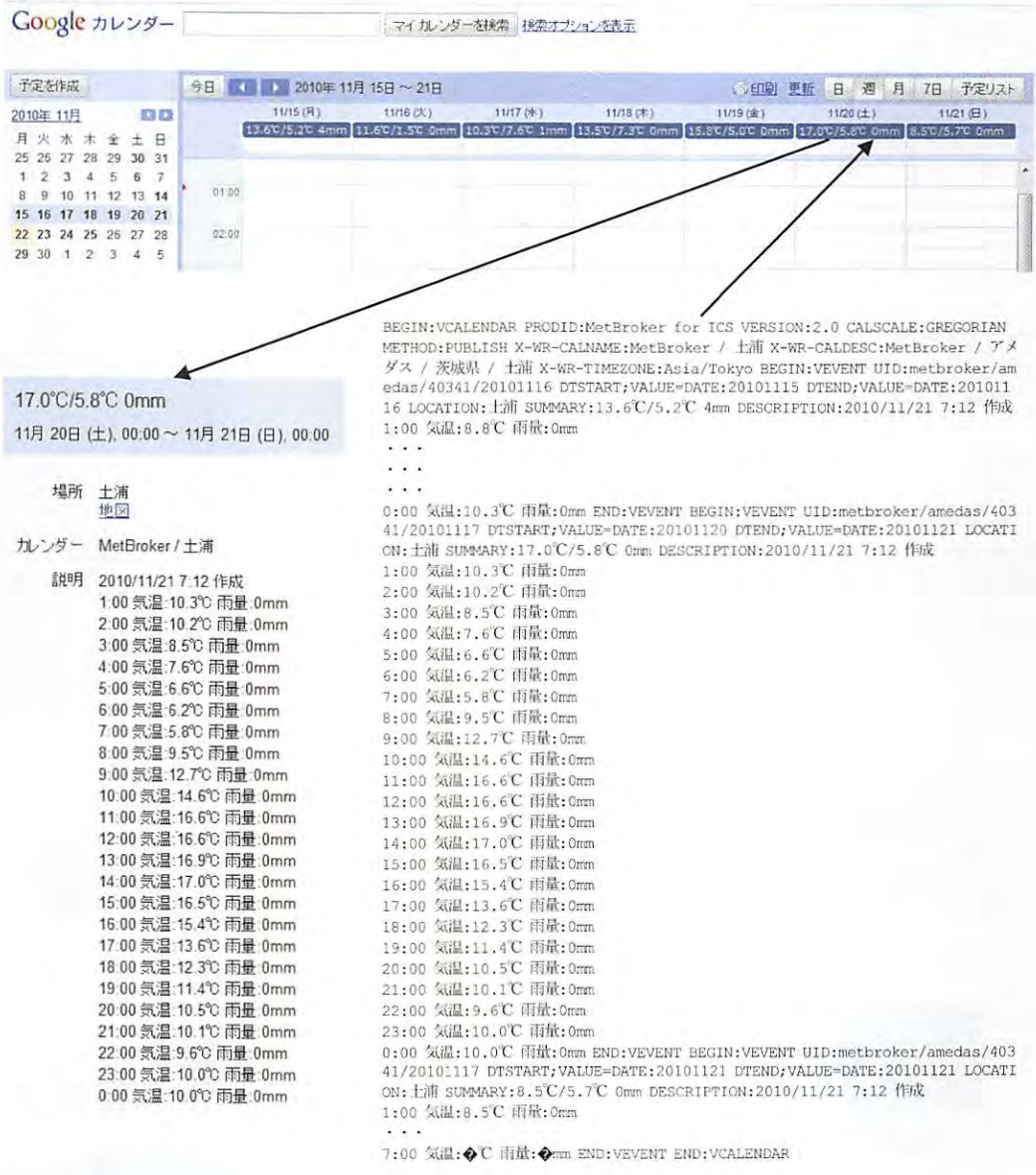

図 47　MetBroker ICS による Google カレンダーへのアメダスデータの表示

Google カレンダーは定期的に，MetBroker ICS から iCalendar 形式のテキストデータ（右下）を取得する．そのデータのうち，日別の最高，最低気温，降水量が Google カレンダーの予定欄（上）に表示され，その部分をクリックすると予定詳細欄に時別値（左下）が表示される．

http://pc105.narc.affrc.go.jp/metbroker/ics/

図 48 携帯電話用気象データ取得サーブレット MetBroker-i

携帯電話用気象データ取得サーブレット MetBroker-i に i-mode HTML Simulator II[184] でアクセスした画面.
Web ブラウザでの利用を想定して開発されたサーブレットも，簡易な表示インタフェースを用意するだけで，携帯電話での利用に適した表示にできる.

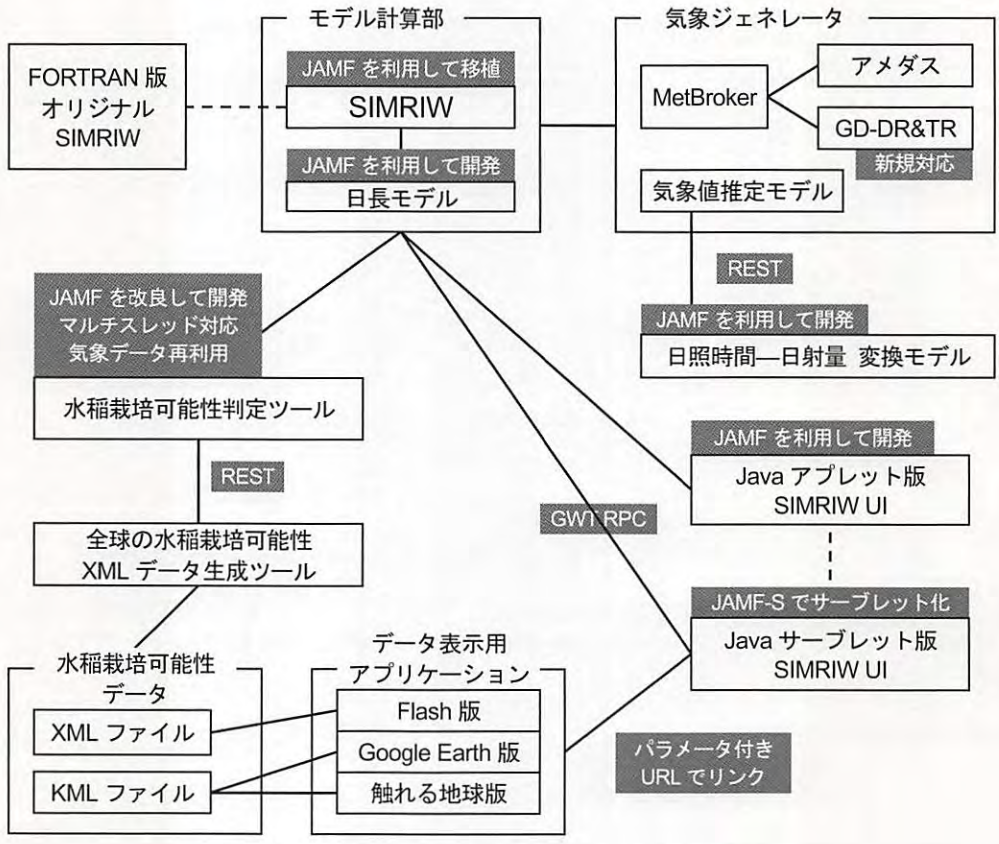

図 49 水稲栽培可能性予測支援ツールの構成

〈III〉，〈IV〉で構築した農業モデル実装フレームワーク JAMF，JAMF-S とメッセージ交換手法を利用している部分を，背景灰色の白抜き文字で示している.

「安全な農作物生産管理技術とトレーサビリティシステムの開発」の課題がある．これは，DIASコアシステムで提供されるデータ群を利用して，農業生産管理支援情報や，地球温暖化による食糧生産への影響等を長期的にも短期的にも，誰でも簡単に知ることができるシステムを構築することが目的である．本節で開発する水稲栽培可能性予測支援ツールはDIASのシステムを構成するアプリケーションの1つである．

2) 水稲生育予測モデル

文献として公開されているものだけでも，多くの水稲生育モデル（表1）がある．水稲栽培可能性予測の対象を全球とした場合，世界中で栽培試験を行って地域パラメータを決定することは不可能なため，地域パラメータを必要としないモデルを採用する必要がある．Simulation Model for Rice-Weather Relations（SIMRIW）[95,97]は，地域パラメータを持たないが，品種パラメータの推定に気象条件の異なる広範な地域の栽培データが利用されていれば予測精度が安定するので[164]，この条件に適っている．また，国内を対象に気候変動の影響を予測するために利用されたこともある[96,165]．

SIMRIWは京都大学の堀江らによって開発された水稲生育モデルである．このモデルは，発育速度（DVR）を積分した発育指数（DVI）で作物の発育ステージを決定する発育速度モデル[40]である．SIMRIWでは発育速度の計算において，日長感応性の時期であるかと，発育相が栄養成長相，生殖成長相，成熟相かによってDVR計算式を使い分けることにより，予測精度の向上が図られている[94,164]．SIMRIWは施肥された灌漑水田での栽培を前提とした潜在成長のみをシミュレートするため，水ストレスや窒素ストレスの影響をシミュレートすることはできない．

SIMRIWは23個の品種パラメータを持っている．これらの値は栽培試験データをもとにシンプレックス法[77]で求められた[94]．出穂期までのDVRを計算するための6つのパラメータは，品種ごとに異なる値が設定されているが，それ以外のパラメータでは，ジャポニカ米とインディカ米で異なる値が設定されるか，すべての品種で同じ値が設定されている．

3) 気象データ

SIMRIWは気象データとして日別の気温と日射量を利用している．実行に必要な気象データの種類が少ないことは，全球で農業モデルを実行可能にするために有利な条件である．しかし，日射量は気温や降水量に比べると入手しづらいことを考慮して，SIMRIWに日照時間－日射量変換モデル[146]を組み込んで利用できるようにした．この変換モデルを利用することにより，日射量は観測していないが，日照時間を計測しているアメダス観測地点[119]でSIMRIWが実行可能になった．

農業モデル用フレームワークJAMF〈III.5〉の気象ジェネレータ〈III.5.1)〉では，MetBroker〈II.2.5)〉を通して各種気象データベースの気象データを取得している．主な気象データベースとして，国内用にアメダス，海外用にNOAA/NWSを利用できる．しかし，NOAA/NWSからは日射量，日照時間ともに取得できないため，海外地点でのSIMRIW実行は，日射量を扱うその他の気象データベースを利用できる一部の国を除いて困難であった．今回，全球を対象とするために，新たにGD-DR&TR〈I.3.3)(6)〉をデータベース化してMetBrokerで扱えるようにした．気温，降水量，日射量をGD-DR&TRから1度グリッド(陸地のみ)で取得できるようになったため，SIMRIWを利用した全球を対象としたシミュレーションが可能となった．MetBrokerが新たにGD-DR&TRを扱えるようになったことに伴って，SIMRIW実行に関するプログラムを変更する必要はなかった．これは，農業モデルが気象データをMetBroker経由で取得するように開発することによる利点の1つである．

4) SIMRIWプログラム

FORTRANで書かれたオリジナルのSIMRIWプログラムと，8品種（ジャポニカ米5品種：イシカリ，ササニシキ，コシヒカリ，日本晴，ミズホ；インディカ米3品種：IR36，IR64，IR58）のパラメータが科学研究費補助金の報告書[97]に付録として記載されている．このプログラムの入出力部を除く，モデル計算部をもとに，農業モデル用フレームワークJAMFを利用して，Webアプリケーション版のSIMRIW[257]をJavaで実装した．

オブジェクト指向プログラミングの特徴を生か

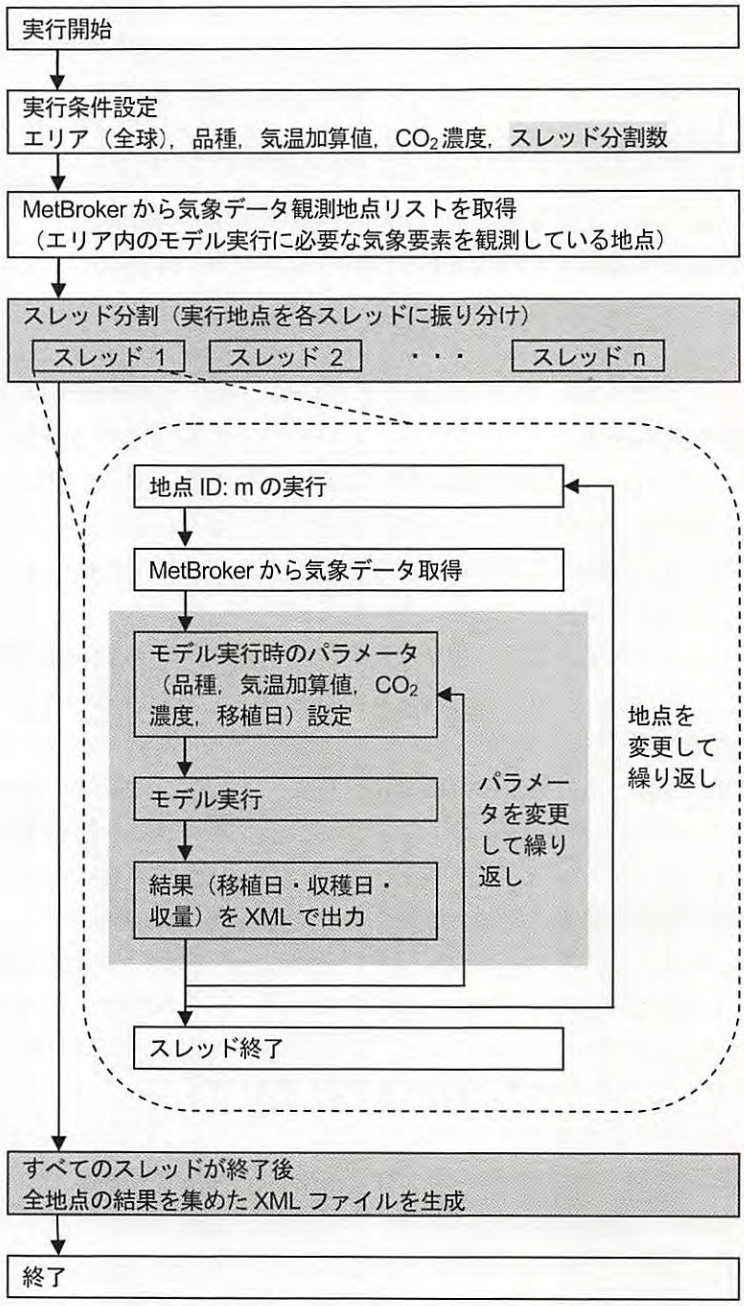

図 50 水稲栽培可能性データ生成の実行サイクル

モデル実行エンジンの改良部分が灰色の背景色で示されている．

し，モデル計算部や状態変数が適切に設計されているので，開発されたプログラムの保守や再利用は容易である．これまでに，Java アプレット版，Java サーブレット版，計算高速化のためのモデル実行エンジン改良などが行われてきたが，モデル計算部分を修正する必要はなかった．また，降水量による閾値や，移植直後の低温による枯死条件の導入では，モデルの終了条件判定部分へのわずかな追加のみで済んだ．

5) モデル実行エンジンの改良

SIMRIW を含めて，既存の Web アプリケーション版農業モデルは，ユーザとの対話形式による実行を想定した実装であった．そのため，「気象データ取得－実行－結果表示」という実行サイクルで，1 回の実行ごとに気象データを取得し，表示した実行結果を保存するかはユーザの操作に委ねられていた．

JAMF を利用して実装された農業モデルを実行するための中核的なプログラムであるモデル実行エン

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cultivation_possibility>
  <cultivar>Koshihikari</cultivar>                                          ┐ 品種
    <year>1990</year>                                                       │
    <air_temp_add_value>0</air_temp_add_value>                              │ 気温加算値
    <co2 unit="ppm">350</co2>                                               ┘ CO2 濃度
    <station region_id="12032" source_id="GD-DR&amp;amp;TR" station_id="19401">┐
      <place_name>latitude=36.5_longitude=140.5</place_name>                │
      <place latitude="36.5" longitude="140.5"/>                            │ 地点情報
      <area ne_latitude="37.0" ne_longitude="141.0"                         │
            sw_latitude="36.0" sw_longitude="140.0"/>                       ┘
    </station>
    <yields>                                                                ┐
      <yield>                                                               │
        <transplanting_date>1990/3/22</transplanting_date>                  │ 移植日を 1/1～
        <heading_date>1990/8/1</heading_date>                               │ 12/31 で実行
        <maturity_date>1990/9/12</maturity_date>                            │ した場合の
        <weight state="14%moist">                                           │ 出穂日，収穫日，
          <actual_brown_rice unit="t/ha">5.61</actual_brown_rice>           │ 収量が並ぶ
        </weight>                                                           │
      </yield>                                                              │
      ・・・                                                                │
    </yields>                                                               ┘
  <possibility>true</possibility>                                           ┐ 栽培可能性
  <max_yield>                                                               ┘
    <transplanting_date>1990/3/30</transplanting_date>                      ┐
    <heading_date>1990/8/2</heading_date>                                   │
    <maturity_date>1990/9/12</maturity_date>                                │
    <weight state="dry">                                                    │
      <crop_including_roots unit="t/ha">18.14</crop_including_roots>        │
      <panicle unit="t/ha">9.48</panicle>                                   │
      <potential_brown_rice unit="t/ha">6.48</potential_brown_rice>         │ 最大収量となっ
      <actual_brown_rice unit="t/ha">4.86</actual_brown_rice>               │ たときの条件
    </weight>                                                               │
    <weight state="14%moist">                                               │
      <potential_brown_rice unit="t/ha">7.54</potential_brown_rice>         │
      <potential_rough_rice unit="t/ha">9.92</potential_rough_rice>         │
      <actual_brown_rice unit="t/ha">5.65</actual_brown_rice>               │
      <actual_rough_rice unit="t/ha">7.44</actual_rough_rice>               │
    </weight>                                                               ┘
    <sequential_data>                                                       ┐
      <element id="airtemperature" name="Air temp.">                        │
        <subelement name="ave." unit="C">                                   │
          <value date="1990/3/30">11.1</value>                              │
          ・・・                                                            │
        </subelement>                                                       │
      </element>                                                            │
      <element id="radiation" name="Radiation">                             │ 最大収量となっ
        <subelement name="Global" unit="MJ/m2">                             │ たときの，気温，
          <value date="1990/3/30">14.61</value>                             │ 日射量，DVI 値の
          ・・・                                                            │ 時系列データ
        </subelement>                                                       │
      </element>                                                            │
      <element id="DVI" name="DVI">                                         │
        <subelement name="max." unit="">                                    │
          <value date="1990/3/30">0.2</value>                               │
          ・・・                                                            │
          <value date="1990/9/11">2.0</value>                               │
        </subelement>                                                       │
      </element>                                                            │
    </sequential_data>                                                      ┘
  </max_yield>
</cultivation_possibility>
```

図 51　地点ごとの栽培可能性データの XML ファイル

データベース GD-DR&TR，地点番号 19401 の栽培可能性データ GD-DR&TR19401.xml

ジン〈III.5.3)〉を改良することにより，同じ実行地点で品種や各設定（移植日 1/1 ～ 12/31，気温加算値，CO_2 濃度）のみを変更して実行する場合には，一度取得した気象データを再利用して繰り返し実行できるようにし，実行時に最も時間を要していた気象データ取得の効率化を図った．同時に，マルチスレッドを利用した高速化を行った．また，実行結果は画面に表示する代わりに XML 形式のファイルとして出力するようにした．水稲栽培可能性データ生成の実行サイクルを図 50 に示す．図中のモデル実行エンジンの改良部分が灰色の背景色で示されている．

6）栽培可能性データの出力形式

全球の栽培可能性データ生成には長時間を要するため，あらかじめ計算した結果を XML ファイルとして保存し，データ表示アプリケーションが呼び出すようにしている．

水稲栽培可能性の計算結果は，地点，品種，各設定につき 1 つの XML ファイル（ファイル名は「データベース ID + 地点 ID .xml」）に出力される（図 51）．XML ファイル中には，収穫に達したすべての場合の移植日，収穫日，収量が記録され，そのうちの最大収量時の気象データや DVI 値の時系列データも記録される．ファイルサイズは，すべての移植日で栽培可能な場合で 100KByte（3200 行），全く栽培不可の場合で 500Byte（14 行），全球すべて

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cultivation_possibility>
  <cultivar>Koshihikari</cultivar>                                          ┐ 品種
  <year>1990</year>                                                         │ 気温加算値
  <air_temp_add_value>0</air_temp_add_value>                                │ CO2 濃度
  <co2 unit="ppm">350</co2>                                                 ┘
  <stations>
    <station region_id="03010" source_id="GD-DR&amp;TR" station_id="50506">  ┐
      <place_name>latitude=-50.5_longitude=-74.5</place_name>               │
      <place latitude="-50.5" longitude="-74.5"/>                           │ 地点情報と
      <area ne_latitude="-50.0" ne_longitude="-74.0"                        │ 栽培可能性
          sw_latitude="-51.0" sw_longitude="-75.0"/>                        │ （false）
      <possibility>false</possibility>                                      │
    </station>                                                              ┘
    ・・・
    <station region_id="12032" source_id="GD-DR&amp;TR" station_id="19401">
      <place_name>latitude=36.5_longitude=140.5</place_name>                ┐
      <place latitude="36.5" longitude="140.5"/>                            │
      <area ne_latitude="37.0" ne_longitude="141.0"                         │
          sw_latitude="36.0" sw_longitude="140.0"/>                         │
      <possibility>true</possibility>                                       │ 地点情報と
      <max_yield>                                                           │ 栽培可能性
        <transplanting_date>1990/3/30</transplanting_date>                  │ （true）
        <heading_date>1990/8/2</heading_date>                               │
        <maturity_date>1990/9/12</maturity_date>                            │ 栽培可能な場
        <weight state="dry">                                                │ 合は，移植日，
          <crop_including_roots unit="t/ha">18.14</crop_including_roots>    │ 出穂日，収穫
          <panicle unit="t/ha">9.48</panicle>                               │ 日，収量
          <potential_brown_rice unit="t/ha">6.48</potential_brown_rice>     │
          <actual_brown_rice unit="t/ha">4.86</actual_brown_rice>           │ （図 51 の最大
        </weight>                                                           │ 収量の情報が
        <weight state="14%moist">                                           │ 抜き出されて
          <potential_brown_rice unit="t/ha">7.54</potential_brown_rice>     │ いる）
          <potential_rough_rice unit="t/ha">9.92</potential_rough_rice>     │
          <actual_brown_rice unit="t/ha">5.65</actual_brown_rice>           │
          <actual_rough_rice unit="t/ha">7.44</actual_rough_rice>           │
        </weight>                                                           │
      </max_yield>                                                          │
    </station>                                                              ┘
    ・・・
  </stations>
</cultivation_possibility>
```

図 52　すべての地点の栽培可能性データを集めた XML ファイル

全球の栽培可能性データ max-yield.xml

図 53　Java サーブレット版 SIMRIW

JAMF のデフォルトのユーザインタフェースで開発した SIMRIW の設定画面と結果表示画面．一般の Web アプリケーションとして実行することもできるが，図 55 や図 57 からパラメータ付き URL で呼び出すこともできる．

の 1.5 万地点の合計で 600MByte 程度（ZIP 圧縮で 46MByte）になる．

さらに，全球すべての 1.5 万地点のファイルから，最大収量データのみを集めた全球データの XML ファイル max-yield.xml が生成される（図 52）．max-yield.xml は約 23 万行で，ファイルサイズは 10MByte（ZIP 圧縮で 600KByte）になる．

7）表示アプリケーション

JAMF のデフォルトのユーザインタフェースは

図 54　水稲栽培可能性予測支援ツールのトップページ画面

図 55　品種を指定した栽培可能性表示画面（Flash 版）

研究者を対象としているため，入力インタフェースには設定項目が多く，出力インタフェースは時系列値のグラフや表による表示が中心で，一般向けではなかった（図 53）．そのため，誰でも利用できることを考慮した新しいユーザインタフェースを開発した．

(1)　Flash 版

Adobe Flash を利用して，Google マップ上に水稲栽培可能性データを表示する Web アプリケーション [232] を，Adobe Flex [3] を利用して開発した．全球の XML ファイル max-yield.xml のサイズは 10MByte 程度と大きく，リアルタイムに解釈することは難しい．そのため，XML データをあらかじめ地図上に表示する画像に変換しておき，Google マップ上にレイヤ表示している．

水稲栽培可能性データは，ツールのトップページ（図 54）を経て，指定した品種の栽培可能性表示画

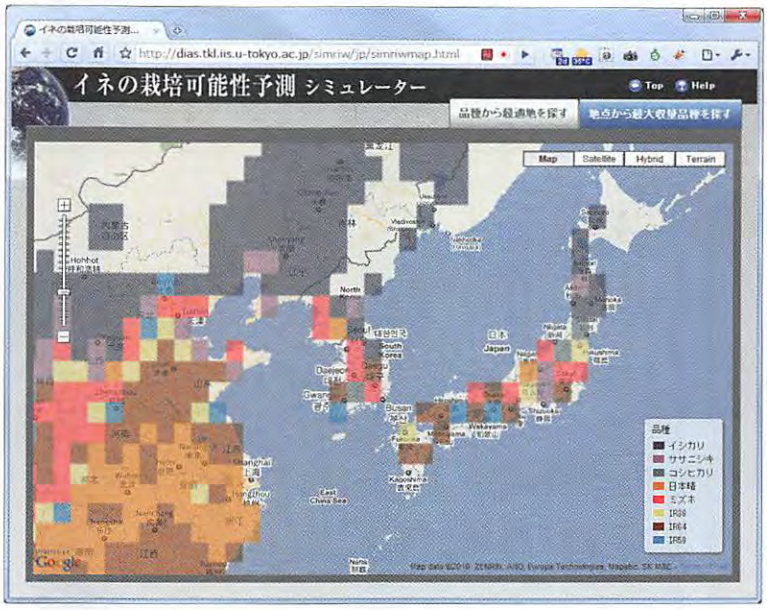

図 56　収量最大となる品種の表示画面（Flash 版）

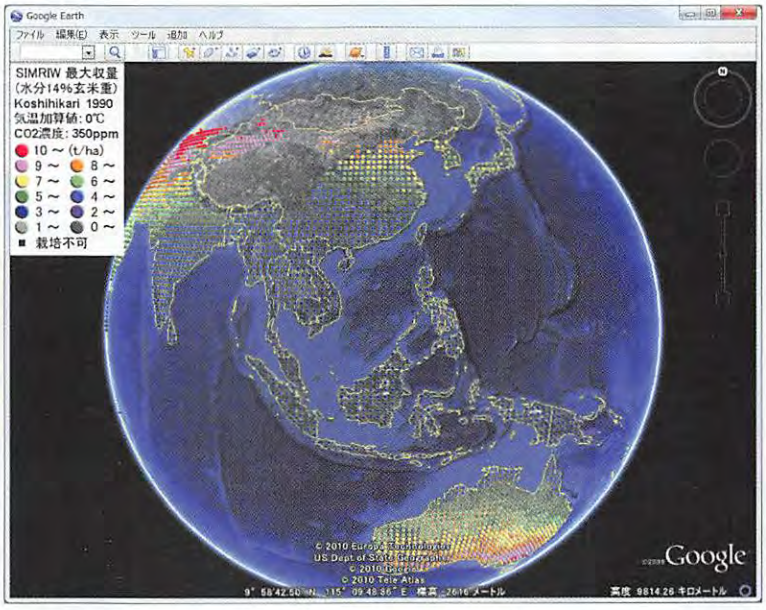

図 57　栽培可能性表示画面（Google Earth 版）

面（図 55）や，収量最大となる品種の表示画面（図 56）のように表示される．図 55 の地図上の地点をクリックすれば，地点情報や成長曲線，移植日別の収量のグラフが地図の下に表示される．

(2)　Google Earth 版

Google Earth 上に水稲栽培可能性データを表示するには，max-yield.xml を KML ファイルに変換することで可能である（図 57）．max-yield.xml から変換された max-yield.kmz（約 300KByte）は，max-yield.xml を変換した KML ファイルとアイコンの画像ファイルを ZIP 形式でまとめたものである．max-yield.xml から max-yield.kmz への変換は，同じ XML 形式どうしの変換なので容易である．

Google Earth 表示用の KML ファイルは Flash 版の表示確認用に作成したものであったが，複数の品種やパラメータのデータを重ね合わせ表示して比較することや，時系列データのアニメーション表示が

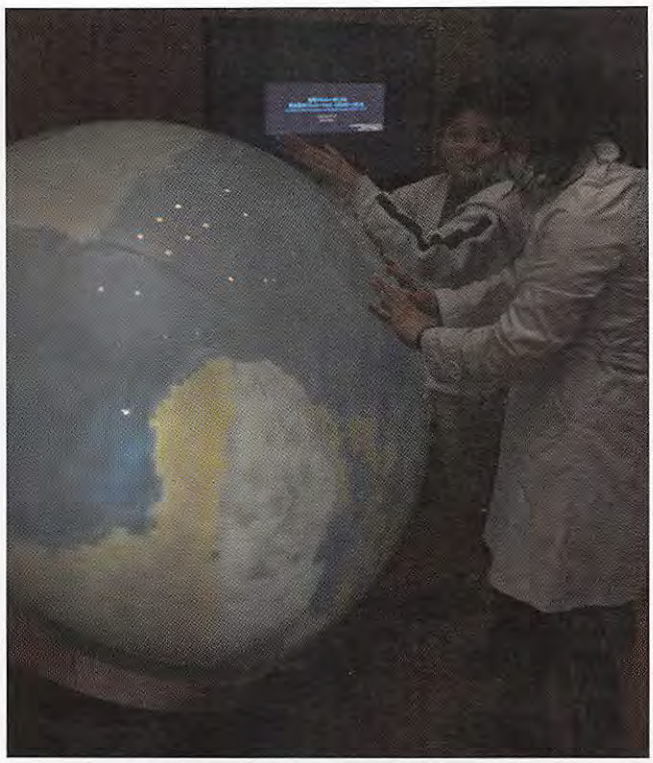

図58 栽培可能性表示画面（触れる地球版）

手で撫でることにより表示地域を回転させているところ．陸地の白色でないところが栽培可能地域を表している．

可能であるため，Flash版と同様に実利用されている．MetBrokerが扱う2.3万地点の気象観測地点用のKMLファイルをGoogle Earthで表示するのに問題がなかった〈IV.3.3）（2）〉ように，全球の1.5万地点の栽培可能性データ用KMLファイルはストレス無く表示される．

（3） 触れる地球版

触れる地球[63]は地球のダイナミズムを体感できるデジタル地球儀である．触れる地球は球体の内部から陸地，海やその上に重ねるデータを投影し，手で触れることにより，見たい地域に移動させたり，拡大させたりできるインタフェースを持つ．触れる地球用の表示データ（図58）は，データ制作会社に依頼してGoogle Earth用のKMLファイルを加工することにより作成される．触れる地球版は本成果をイベントで展示するために利用した．

8） 結果

8品種，気温加算値3通り，CO_2濃度2通りの条件の組み合わせで，移植日を365日間で実行すると，1地点あたり17,520回モデルを実行することになる．これを1.5万地点で実行するのに要した時間は，Core i7-860のPCで7日間であった．取得した気象データの再利用による気象データ取得回数の削減と，スレッド分割による高速化による，JAMFのモデル実行エンジンの改良で2日間強に短縮された．全球での栽培可能性データの生成には長時間を要するため，あらかじめ計算したデータを表示用に利用している．

気象データのあるすべての年（GD-DR&TRは40年間）で，多くのパラメータの組み合わせで実行しようとすると，データ表示前にあらかじめデータを計算しておくとしても，かなりの時間を要する．今回の条件では3ヶ月間かかるため，40年間のうちの5年分の計算を行った．また，モデルの計算式に枯死条件の変更などのために修正を加えた場合には，すべて計算し直さなければならない．ただし，対象エリアを数十地点程度に絞り込めば実行時間が1分以内に収まるので，ユーザによるオンデマンドでの実行が可能である．

このシステムのモデル実行エンジンに投入する作

物モデルは，JAMF を利用して実装したモデルであれば入れ替え可能である．そのため，パラメータや入力データを用意できるなら，より複雑な水稲生育モデルと入れ替えできる柔軟性がある．また，国や地域ごとに適したモデルに切り替えて栽培可能性データを生成するような発展性もある．さらに，他の作物の栽培可能性データを生成するために，水稲以外の作物モデルと入れ替えることもできる．

水稲栽培可能性データは XML 形式で保存されているので，図 51 や図 52 の XML ファイルを解釈するための XML パーサを組み込めば，ユーザの要求を反映した新たな表示用アプリケーションを開発できる．

水稲栽培可能性データの表示用アプリケーション（図 55 ～図 58）は，操作性や視認性などにおいて優れ，洗練されたユーザインタフェースを持つものとして開発された．世界の食糧問題に関心のある研究者以外の一般の方が，気候変動が食糧生産に与える影響などを容易に体感できるようなツールにするためである．2010 年 7 月に開催された DIAS フォーラムの会場に展示された「触れる地球」を体験した一般参加者は，データ統融合技術の進歩に驚きながら世界の食糧問題に強い関心を示していた．その意味では，水稲栽培可能性予測シミュレータは，研究の成果を分かりやすく伝えるアウトリーチ活動のための教育ツールとしての価値も持っている．

今後，SIMRIW の対応品種を増やすとともに，水稲以外の作物モデルにもこのツールを適用していく予定である．また，このツールを利用して栽培可能性を判定するには 1 度グリッドでは粗すぎるため，0.1 度グリッドデータを MetBroker 経由で利用するための作業を進めている．

5. 考察

本章では，複数の作物モデルや病害虫モデル，気象モデルなどのサブモデルと連携，相互作用を行うためのメッセージ交換手法を検討した後，農業モデルを REST アプリケーションとして実装可能な，新たな農業モデル用フレームワーク JAMF-S を構築した．JAMF-S には Google による様々な Web サービスを利用する機能も追加された．JAMF-S を利用すると，Ajax アプリケーションとして，またマッシュアップアプリケーションとして農業モデルを実装できる．

1990 年代後半の増殖情報ベースプロジェクト進行時には，分散協調システム構築用に CORBA や HORB の利用が検討されたが，複雑さのために本格的な利用に至らなかった．その後，プロジェクトの主なアプリケーション構築に Java を利用することになったことと，Java RMI の登場により，分散協調システムのオブジェクト間通信に RMI が利用された．農業モデルと MetBroker の通信にも RMI が利用されていた．しかし，実運用段階でファイアウォール越しに RMI による通信が行えない場合があるという問題が起きた．この問題は，MetBroker を HTTP による通信を行う Java サーブレットとして実装し直すことで解決された．この修正においては MetBroker のインタフェースの変更も行われたため，農業モデル側でも MetBroker との接続処理を中心にプログラムの修正が必要であった．

農業モデルの REST アプリケーション化のために，Java アプレットから Java サーブレットとして実装し直す必要があり，Java サーブレット用の農業モデル実装フレームワーク JAMF-S が構築された．Java サーブレット化による大きな変更点は，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．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，気象ジェネレータのプログラムに対する変更は必要なかった．

ユーザインタフェースとなる Web 画面は，当初，JSP により HTML+JavaScript で生成していた．Ajax 登場以前は JavaScript の役割はコンボボックス用リストの生成や入力値の確認程度で，小規模であったために開発の負担は少なかった．その後，Web アプリケーションの Ajax 化により JavaScript の役割が増してくると，prototype.js [205] などの Ajax アプリケーション構築用の JavaScript ライブラリを利用できるようになったが，JavaScript のコード量は激増していった．また，JavaScript，HTML，CSS のデバッグを行える Firebug [161] の利用により，デバッグ作業が効率化されたが，規模の大きな JavaScript 開発は，コンパイラ言語である Java による開発に比べて負担が大きく，開発や保

守の効率が落ちた．

Google Web Toolkitが登場すると，クライアント画面も含めてすべてをJavaで開発できるようになり，ライブラリ化して再利用することが容易になった．これにより，JavaScript開発の問題は解消した．また，GoogleはAjaxアプリケーションやマッシュアップアプリケーションの起源となるGoogleマップやGoogle EarthなどのAPIを公開したWebサービスを提供しており，地図インタフェース開発において重要な地位を占めている．

RESTアプリケーションはAMADIS要素間連携のために構築されたものであったが，結果として受け取ったXMLデータを加工して再利用するアプリケーションも現れた．病害の発生条件は日々の気象条件で変化し，防除適期も長くない．そのため，病害発生予察モデルを病害防除の適期を判断するために利用するには，ユーザは日常的に病害発生予察モデルを実行して最新の病害発生予測情報を得る必要がある．この手間を改善するために，病害発生予察モデルを定期的に実行し，実行結果のXMLデータをRSSに変換してユーザに配信するサービス[230]や，Googleカレンダー上に表示するサービスなどが構築された．

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．1年分のデータを生成するためには，全球の1度グリッド（陸地のみ）の1.5万地点に対し，8品種，気温加算値3通り，CO_2濃度2通りの条件の組み合わせで，移植日を365日間で実行し，1地点あたり17,520回モデルを実行することになった．JAMFのモデル実行エンジンをそのまま利用すると，1年分のデータ生成に1週間を要した．そのため，取得した気象データの再利用による気象データ取得回数の削減と，スレッド分割によるモデル実行エンジンの改良で計算時間は2日間に短縮できた．この改良にあたっては，オブジェクト指向で構築されたJAMFの特徴を生かし，モデル実行エンジンのプログラムを拡張することにより，最小限のプログラミングで対応できた．このことから，JAMFの利用により，個々の農業モデルをWebアプリケーションとして効率的に構築できるだけでなく，実装するアプリケーションに応じて，モデル実行エンジンなどのJAMFの中核機能に対する拡張による機能追加や変更が柔軟に行えることが示された．

V．総合考察

1．各章の考察

本論文の目的は，農業用の意思決定支援システムを，既存のレガシー化しつつある農業モデルやデータベースを組み合わせて構築可能とする分散協調システムと，その構築手法について研究することである．

〈I〉では農業シミュレーションモデルの歴史と共に，海外の主な農業モデル開発グループにより開発された農業モデルについて，作物モデルを中心に紹介した．また，国内向けの農業用意思決定支援システムを構築するにあたり，集中型システムと分散型システムの比較，検討を行った．既存の農業モデルやデータベースが全国の研究機関で運用されていること，今後の保守や拡張も開発元で行われる方が望ましいこと，情報技術の進化やネットワーク普及の度合から，分散協調型システムとして構築することを選択した．

〈II〉では，農業用の意思決定支援システムのための分散協調型システムAMADISの構成と，各種サーバ，農業モデル，データベースなどの構成要素を紹介した．〈I〉で紹介した農業モデルに対するAMADISの特徴は，農業モデル自身やデータベースまでをコンポーネントとして扱うことができ，それらがネットワークで結びつけられることである．AMADISは2つの農業情報技術のためのプロジェクト研究において構築された巨大なシステムであるため，本論文では主に，農業モデルをAMADIS構成要素として実装するためのフレームワーク構築に関する研究を行った．

〈III〉では，海外のモデル開発グループによる農業モデルの開発言語や構造についての研究を行った後，農業モデル用フレームワークJAMFを構

築した．農業モデルの多くは，数値計算が得意なFORTRANやそれに似たシミュレーション言語で開発されていたが，規模が大きくなり保守が難しくなってくると，過去のプログラム資産より，開発効率，拡張性，保守性を重視するようになり，それらに優れたオブジェクト指向言語での開発に推移しつつある．本研究ではオブジェクト指向言語であることと，Webアプリケーション開発に適していることからプログラム開発言語としてJavaを選択した．また，レガシー化した農業モデルを利用する上で，そのプログラムサイズの小ささや，最新の技術の利用に対する制限を設けないためにも，ラッパーによる方法でなく，Javaに移植する方法を選択した．その後の機能拡張などが円滑に行えたことからも，この選択は正しかった．

農業モデル用フレームワークJAMFを構築するために参考にした，モデル開発グループによる農業モデルの構造は，同時期に開発されたことから多くの類似点があり，実行制御方法やモジュール構造などがJAMFに取り入れられた．

JAMFは農業モデルの計算を制御するためのモデル実行エンジン，実行に必要な気象データを用意する気象ジェネレータ，データ構造，ユーザインタフェース，各種ユーティリティプログラムを提供し，農業モデルをWebアプリケーションとして実装できる．農業モデルの実行は，気象データを取得して，終了条件が成立するまで繰り返し計算を行い，結果を出力するという共通した流れがあり，設定画面や結果表示画面も共通している部分が多い．そのため，JAMFが提供する共通部分のプログラムを利用すると，個々の農業モデルを実装するために新たに開発するプログラム行数は5%程度で済み，その大半は農業モデルの計算部分となる．このことから，JAMFを利用することにより，農業モデルのプログラム開発者はわずかな労力で農業モデルを実装でき，さらに農業モデルの中核部分であるモデル計算部分の実装に集中できるようになった．

〈IV〉では，複数の農業モデルがネットワークを通して連携するための，メッセージ交換手法についての研究を行った後，メッセージ交換機能を持つようにJAMFを改良したJAMF-Sを構築した．メッセージ交換手法として当初はJava RMIを選択したが，セキュリティ対策に起因する問題により，その後登場したRESTへ移行した．RESTでは文字列処理で設定できるURLのパラメータにより，農業モデルの実行条件を指定でき，結果はXML形式で出力されるため，他のWebサービスと組み合わせて，マッシュアップアプリケーションを構築することが容易であった．また，サーバやクライアントの環境に依存しないプラットフォーム非依存性という利点も享受できた．

最後に，JAMFやJAMF-Sの有効性を示すための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が必要であったため，モデル実行エンジンに対して，マルチスレッドや入力データ再利用の仕組みの導入が必要であったが，オブジェクト指向で構築されたJAMFの拡張性の特徴を生かし，最小限の改良で対応できた．また，実運用の面でも問題ないことを示した．

これにより，AMADISのための農業モデル実装基盤技術が確立し，多様な農業モデルをWebアプリケーションとして効率的に開発，拡張，保守できることが示された．また，農業モデル連携による機能拡張を実現できたことから，提案した分散協調システム型アーキテクチャの正当性が検証された．

2. 本システムの有効性

1) 農業モデル用フレームワークJAMF

農業モデルはユーザが限られていることもあり，研究，開発の主体は大学や農業試験場の研究室単位であることが多かった．有名なモデルの中には20～30年前のコードがそのまま使われているものもあった．それらのプログラムの中には，設計，開発後の保守，ドキュメントが不十分であったり，GUI操作に非対応であったりと，プログラムがレガシー化する要因を抱えているものが少なくなかった．また，気象データの形式や取得を固有の方法で行っているためにプログラムの再利用を妨げているものもあった．

本研究で構築した農業モデル用フレームワークJAMFを利用して農業モデルを実装すると，オブジェクト指向プログラミングの特徴である保守性や拡張性の高さ，MetBroker対応の気象ジェネレー

タを利用した気象データの取得，Google などにより提供される Web サービスとのマッシュアップが可能となる．

JAMF を利用してレガシー化した農業モデルを Web アプリケーションとして復活させるためには，農業モデルの計算部分を Java で書き直すことになる．元のプログラムサイズは小さく，データ入出力などのモデルの本質でないプログラムの占める割合が大きいために，計算部分の割合が半分程度であることが多く，書き換えのための負担は大きくない．このことは，葉いもち感染好適日推定モデル MetBLASTAM の構築を例にして示した．農業モデルのプログラムを新規に開発したり，他言語から移植したりする場合に必要なプログラムは全体の 5.2% のみで，残りはフレームワークのプログラムライブラリにより提供されていた．さらに，新規に開発するプログラムの大半が個々の農業モデルに固有な計算部分であることから，プログラム開発者はモデル計算部分の実装に集中できることが示された．すでに MetBLASTAM 以外にも 20 以上の農業モデル（表 9）を Web アプリケーションとして構築し，誰でも利用できる形でインターネット上に公開し，長期間の運用をすることにより安定性も示した．また，JAMF の API とサンプルプログラムを公開しているので，ある程度のプログラミング経験があれば，農業モデルを Web アプリケーションとして構築できるようになっている．

農業モデルが AMADIS の構成要素となるには，農業モデル間で相互に連携するためのネットワークを介したメッセージ交換機能が不可欠であるため，JAMF は REST アプリケーションとして農業モデルを実装できるようになっている．REST では URL のリクエスト文で農業モデルの実行条件を指定し，実行結果は XML 形式で取得できるため，リクエストの生成も実行結果の処理も容易である．また，通信には HTTP を利用するため，ファイアウォールに起因する問題が起こりにくい．さらに，サーバやクライアントの環境に依存しないプラットフォーム非依存性という利点も享受できた．

これらの結果から，JAMF が AMADIS の構成要素である多様な農業モデルを，相互連携機能のある Web アプリケーションとして，効率的に開発，拡張，保守できるという有効性が示された．

2) 農業モデル・データベース分散協調システム AMADIS

本研究で提案した農業用の意思決定支援システム AMADIS は，分散協調型システムである．AMADIS では農業モデルや各種データベースまでをコンポーネント化し，全国の大学や試験場などの開発元で運用したまま，ネットワークで結びつけて協調動作させることが大きな特徴である．分散協調型とした理由は，農業モデルやデータベースが小規模で地域性が高く，開発元による保守や拡張が迅速に行える点で優れているからである．

農業モデルの機能をより詳細にするために，機能追加のためのプログラムを開発するだけでなく，他の農業モデルと連携させることによって実現できる柔軟性が，分散協調型の優れているところである．実際に，JAMF を利用して AMADIS の構成要素として実装した作物モデルに，同様に JAMF を用いて実装した病害モデルや気象モデルなどを連携させたアプリケーションを構築できたことから，分散協調型システム AMADIS のアーキテクチャの有効性が示された．

3. 残された課題

〈III.3〉で触れたように，プログラムをモジュール構造化してもインタフェースが異なると相互利用が難しい[114]．JAMF-S では容易化のために AMADIS の要素間の通信に REST を採用した．REST のリクエストは URL のパラメータで行うため，事前にパラメータの妥当性を検査することができない．レスポンスは XML で返されるため，スキーマ宣言されていれば，スキーマにより XML の内容を事前に理解できる．ただし，同じ意味に対する要素名の違いを吸収するためにはオントロジーの利用が必要である．今後，AMADIS の要素間通信で REST 利用により，このことが問題化するようであれば，複雑化するが SOAP を併用することが考えられる．

農業モデルは対象作物の品種と，栽培地域に関するパラメータを利用することにより，品種や地域による生育の差を表現し，予測精度を向上させている．しかし，数年分の気象データと生育データを集め，シンプレックス法や遺伝的アルゴリズムなどでこれらのパラメータを推定する〈II.2.8)〉ことは，一般

の利用者には難しい．すでに遺伝的アルゴリズムでパラメータを推定するためのプログラム開発は行っていたが，それを AMADIS の要素として提供するまでには至っていない．農業モデルの精度向上のために，農業モデルにパラメータ推定機能を組み込み，ユーザが必要に応じて利用できるようにする予定である．パラメータ推定に利用する観測データをユーザ自身で用意できない場合には，気象データは MetBroker から，水稲の栽培情報はイネデータベースから取得できる．

IT システム（サーバやネットワーク，システムを運用するデータセンター，それらを監視，運用する仕組み）は，構築時だけでなく管理，維持していくにもコストがかかる．プロジェクト研究において AMADIS 関連で研究開発した多くのアプリケーションを維持するためには，OS，アプリケーションの保守，ドメイン管理なども必要である．しかし，プロジェクト終了後には，それらを維持するための予算，人的資源がなくなり問題となっている．

4. 今後の展望

1) 他の Web サービスとの連携

JAMF-S で実装された農業モデルの REST アプリケーションは，Web ブラウザ上でユーザインタフェースにより実行条件を設定するだけでなく，パラメータ付き URL により実行条件を設定できる．そのため，リクエスト用 URL の文字列操作のみで，ネットワーク処理に対応したどのようなプログラミング言語からも農業モデルを呼び出すことができる．

また，出力された XML 形式の結果データはテキスト形式で可読性があり，XML 処理が可能なプログラミング言語で容易に扱うことができる．Google マップなどの Web サービス上に表示したり，他の農業モデルなどの入力データとして利用したりすることにより，マッシュアップアプリケーションも容易に構築できる．また，サーブレットとして出力を表やグラフに表示するだけでなく，表示用プログラムを変更するだけで，携帯電話や Google カレンダーなど多様な出力先への対応ができる．

2) 情報配信

一般情報を広く素早く伝える手段として，今まではテレビやラジオが利用されてきたが，携帯情報端末の普及により電子メール，RDF Site Summary (RSS)，Twitter を利用する例が増えてきている．例えば，災害情報がメール[268]や Twitter[51]で配信されている．

農業分野では，農協から農家への FAX による情報配信などが行われていた．JAMF や JAMF-S を利用した例としては，病害虫発生予察情報などの農業モデルの結果をメール[251]や RSS[230]で配信するシステムが構築された．また，気象データを Google カレンダーに表示〈IV.3.5）(1)〉したり，フィールドサーバの観測データを Twitter で配信[88]したりする試みもなされている．

3) クラウドコンピューティング

計算機処理をネットワーク経由で提供するサービスは，1960 年代には大型計算機の CPU を遠隔利用する形で存在していたが，2006 年に提唱されたクラウドコンピューティング（Cloud Computing）[222]が注目を集めている．世界中に分散したユーザがサーバを意識せずにサービスを受けることが，クラウド以前のサービスと異なるところである．それまでユーザが IT システム（ハードウェア，ソフトウェア，データベースなど）を保有，管理していたのに対し，クラウドコンピューティングではネットワーク経由のサービスという形で IT システムを利用できる．最低限のネットワーク環境があれば利用でき，必要に応じて利用する資源を調節できる柔軟性がある．

クラウドコンピューティングには以下の 3 つの形態がある．① Software as a Service（SaaS）はインターネット経由で電子メールやグループウェアなどのソフトウェアを提供するサービスで，Microsoft Online Services[153]や Google Apps[72]などがある．② Platform as a Service（PaaS）はインターネット経由で，アプリケーションサーバやデータベースなどのアプリケーション実行用プラットフォームを提供するサービスで，Google App Engine（GAE）[74]，Face.com[220]や Microsoft Windows Azure[154]などがある．③ Infrastracture as a Service(IaaS)はユーザ自身が OS を含むシステムを導入可能な，インターネット経由のインフラを提供するサービスで，Amazon S3[5]　や Amazon EC2（Elastic Compute

Cloud)[6]などがある．

Google App Engine for Java（GAEj）では標準のJava技術を使用してWebアプリケーションを構築し，Googleのスケーラブルなインフラ環境で実行できる．GAEjは，信頼性の高い永続データの格納，HTTPによるネットワーク経由のリソースアクセス，メールメッセージ送信，画像データ処理，Googleアカウントによるユーザ認証，スケジュールタスク実行などのスケーラブルなサービスを提供している．GAEでは1日単位で無料割り当て分のリソースが定められており，それを超える使用を行う場合に課金される．

クラウドコンピューティングは〈V.3〉で述べたITシステムの管理，維持の問題を解決できる可能性がある．GAEjのアプリケーション構築は，GWTによるサーブレット開発とほぼ同じで，WARファイルの配備先をTomcatからApp Engineへ変更する程度である．JAMF-SとGWTを利用し，Javaサーブレットとしていくつかの農業モデルを開発しているので，GAEjの利用は比較的スムーズに行えると考えられる．HTTPによるネットワーク経由のリソースアクセスが行えることから，GAEj上の農業モデル間の連携も可能である．

農業の現場で利用される主なIT機器は，屋外で利用することから携帯電話，スマートフォン，PDAなどのモバイル端末である．クラウドコンピューティングの利用は，ネットワークに接続できる環境にあれば，クラウドコンピューティングが提供する高速な計算機資源や，膨大な記憶容量を利用可能となることを意味する．このことは，パソコンと比べてCPUやメモリなどの機能に制限のあるモバイル端末に与える影響の方が大きく，工夫次第で今までになかったアプリケーションが誕生し，農業現場で利用される可能性がある．

このように，Javaがサーバサイドプログラム開発の主流となっていることもあり，JAMFで実装された農業モデルは様々な環境で利用できる．今後，農業モデル実装基盤技術として更なる研究を進めることで多くの活用と発展が期待できる．

謝辞

本研究の実施ならびに本論文のとりまとめにあたり，ご指導ならびにご助言をいただいた，平藤雅之博士（筑波大学，北海道農業研究センター）に深甚なる感謝の意を表します．また，本論文をとりまとめるにあたりご指導をいただいた，二宮正士 博士（東京大学，当時 筑波大学，中央農業総合研究センター），林武司 博士（筑波大学，中央農業総合研究センター），竹澤邦夫 博士（同）に感謝の意を表します．

本研究の遂行にあたり，システム構築についてご協力をいただいた，木浦卓治氏（中央農業総合研究センター），深津時広 博士（同），Dr. Matthew R. Laurenson（元 中央農業総合研究センター），山川敦之氏（同）に感謝の意を表します．また，農業モデルの実装についてご協力をいただいた，渡邊朋也博士（中央農業総合研究センター），菅原孝治 博士（同），南石晃明 博士（九州大学），高橋渉氏（富山県農林水産総合技術センター），長谷川利拡 博士（農業環境技術研究所），桑形恒男 博士（同），溝口勝博士（東京大学），星岳彦 博士（東海大学），鈴木剛伸氏（長野県農業総合試験場）に感謝の意を表します．

なお，本研究の一部は，農林水産省研究プロジェクト「増殖情報ベースによる生産支援システム開発のための基盤研究」，「データベース・モデル協調システムの開発」，文部科学省 科学技術振興調整費重点解決型研究等の推進「地球環境データ統合・情報融合基盤技術の開発」，国家基幹技術「海洋地球観測探査システム」の基幹要素であるデータ統合・解析システムの支援によって行われた．プロジェクト研究において貴重なご助言，ご協力を頂いた関係者諸氏に感謝申し上げます．

また，研究を進めるにあたり，ご支援，ご協力を頂きながら，ここにお名前を記すことが出来なかった多くの方々に心より感謝申しあげます．

引用文献

1. Acock, B. and J.F. Reynolds (1989) The rationale for adopting a modular generic structure for crop simulators. Acta Horticulturae 248:391-396, http://www.actahort.org/books/248/248_49.htm.
2. Adobe (2000) ActionScript Technology Center. http://www.adobe.com/devnet/actionscript.html
3. Adobe (2004) Flex Developer Center. http://www.adobe.com/devnet/flex.html
4. Aggarwal, P.K., M.J. Kropff, K.G. Cassman, and H.F.M. ten Berge (1997) Simulating genotypic strategies for increasing rice yield potential in irrigated tropical environments. Field Crops Research 51(1-2):5-17, doi:10.1016/S0378-4290(96)01044-1.
5. Amazon (2006) Amazon S3. http://aws.amazon.com/s3/
6. Amazon (2006) Amazon EC2. http://aws.amazon.com/ec2/
7. Ångström, A. (1924) Solar and terrestrial radiation. Quartely Journal of the Royal Meteorological Society 50:121-126, doi:10.1002/qj.49705021008.
8. Apache Software Foundation (1995) Apache HTTP Server Project. http://httpd.apache.org/
9. Apache Software Foundation (1999) Apache Tomcat. http://tomcat.apache.org/
10. Apache Software Foundation (2000) Apache Ant. http://ant.apache.org/
11. Apache Software Foundation (2000) Apache Struts. http://struts.apache.org/
12. Apache Software Foundation (2001) Apache Tapestry. http://tapestry.apache.org/
13. Apache Software Foundation (2004) Apache Axis2/Java. http://ws.apache.org/axis2/
14. APSIM (2006) APSIM Wiki. http://www.apsim.info/
15. Argent, R.M. and A.E. Rizzoli (2004) Development of Multi-Framework Model Components. Proceedings of the 2nd Biennial Meeting of iEMSs 1, Osnabrück, 365-370.
16. Arnold, K., J. Gosling, and D. Holmes (2001) プログラミング言語 Java, 第3版, ピアソンエデュケーション, 東京, 597pp., ISBN4-8947-1343-7.
17. Basstanie, L.J.M. and H.H. van Laar (1982) Introduction of CSMP by an elementary simulation program in *Simulation of plant growth and crop production*, F.W.T. Penning de Vries and H.H. van Laar, Eds. Pudoc, Wageningen, the Netherlands, ch.2.2, 50-65, ISBN90-220-0809-6.
18. Bentley, J. (2000) 珠玉のプログラミング, 2nd ed. ピアソンエデュケーション, 東京, 305pp., ISBN4-8947-1236-9.
19. Black, J.N., C.W. Bonython, and J.A. Prescott (1954) Solar radiation and the duration of sunshine. Quartely Journal of the Royal Meteorological Society 80:231-235, doi:10.1002/qj.49708034411.
20. Bloch, J. (2001) Effective Java プログラミング言語ガイド, ピアソン・エデュケーション, 東京, 236pp., ISBN4-8947-1436-1.
21. Bolte, J. (1998) Object-Oriented Programming for Decision Systems in *Agricultural Systems Modeling and Simulation*, R.M. Peart and R.B. Curry, Eds. Marcel Dekker, New York, USA, ch.17, 629-650, ISBN0-8247-0041-4.
22. Booch, G. (1995) Booch 法：オブジェクト指向分析と設計, 第2版, アジソンウェスレイパブリッシャーズジャパン, 東京, 625pp., ISBN4-7952-9654-5.
23. Boote, K.J., J.W. Jones, and G. Hoogenboom (1998) Simulation of Crop Growth: CROPGRO Model in *Agricultural Systems Modeling and Simulation*, R.M. Peart and R.B. Curry, Eds. Marcel Dekker, New York, USA, ch.18, 651-692, ISBN0-8247-0041-4.
24. Boote, K.J., J.W. Jones, G. Hoogenboom, W.D. Batchelor, and C.H. Porter (2000) CROPGRO Plant Growth and Partitioning Module in *DSSAT v4 Data Management and Analysis Tools*, vol.4. University of Hawaii, Gainesville, Honolulu, USA, ch.2, 127, ISBN1-886684-08-1.

25. Bouma, J. and J.W. Jones (2001) An international collaborative network for agricultural systems applications (ICASA). Agricultural Systems 70(2-3):355-368, doi:10.1016/S0308-521X(01)00051-8.
26. Bouman, B.A.M., H. van Keulen, H.H. van Laar, and R. Rabbinge (1996) The 'School of de Wit' Crop Growth Simulation Models: A Pedigree and Historical Overview. Agricultural Science 52(2-3):171-198, doi:10.1016/0308-521X(96)00011-X.
27. Bouman, B.A.M., M.J. Kropff, T.P. Tuong, M.C.S. Wopereis, H.F.M. ten Berge, and H.H. van Laar (2001) ORYZA2000: modeling lowland rice. International Rice Research Institute, Wageningen University and Research Centre, Los Baños, Wageningen, 235pp., ISBN971-22-0171-6, with CD-ROM.
28. Brennan, R.D., C.T. de Wit, W.A. Williams, and E.V. Quattrin (1970) The Utility of a Digital Simulation Language for Ecological Modeling. Oecologia (Berl.) 4(2):113-132, doi:10.1007/BF00377096.
29. Bruhn, J.A., W.E. Fry, and G.W. Fick (1980) Simulation of Daily Weather Data Using Theoretical Probability Distributions. Journal of Applied Meteorology 19(9):1029-1036.
30. J.N. Buxton, Ed. (1968) Simulation Programming Languages. North-Holland Publishing Company, Amsterdam, the Netherlands, 463pp.
31. Canpolat, N. and J.P. Bolte (1993) Object-oriented implementation of the CERES-Wheat model. ASAE Paper 934052.
32. Carbonell, J. and J. Goldstein (1998) The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. Proceedings of 21st Annual international ACM-SIGIR Conference on Research and Development in Information Retrieval, 335-336.
33. Confalonieri, R. and S. Bocchi (2005) Evaluation of CropSyst for simulating the yield of flooded rice in northern Italy. European Journal of Agronomy 23(4):315-326, doi:10.1016/j.eja.2004.12.002.
34. Croft, B.A., J.L. Howes, and S.M. Welch (1976) A Computer-based, Extension Pest Management Delivery System. Environmental Entomology 5(1):20-34.
35. David, O., S.L. Markstrom, K.W. Rojas, L.R. Ahuja, and I.W. Schneider (2002) The Object Modeling System in *Agricultural System Models in Field Research and Technology Transfer*, L.R. Ahuja, L. Ma, and T.A. Howell, Eds. Lewis Publishers, Florida, USA, ch.15, 317-330, ISBN1-56670-563-0.
36. Dawson, F. and D. Stenerson (1998) Internet Calendaring and Scheduling Core Object Specification (iCalendar). RFC 2445.
37. Deerwester, S., S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman (1990) Indexing by Latent Semantic Analysis. Journal of American Society of Information Science 41(6):391-407, doi:10.1002/(SICI)1097-4571(199009)41:63.0.CO;2-9.
38. de Wit, C.T. (1965) Photosynthesis of leaf canopies. Agricultural Research Report 663. Pudoc, Wageningen, the Netherlands, 57pp.
39. de Wit, C.T. (1970) Dynamic Concepts in biology. Proceedings of the International Biological Program / Plant Production Technical Meeting, Trebon, 17-23.
40. de Wit, C.T., R. Brouwer, and F.W.T. Penning de Vries (1970) The simulation of photosynthetic systems. Proceedings of the International Biological Program / Plant Production Technical Meeting, Trebon, 47-70.
41. de Wit, C.T. (1978) Simulation of assimilation, respiration and transpiration of crops. Pudoc, Wageningen, the Netherlands, 148pp., ISBN90-220-0601-8, with FST source code of BACROS and PHOTON.
42. DIAS (2006) DIAS データ統合・解析システム. http://www.editoria.u-tokyo.ac.jp/dias/
43. Donatelli, M., G. Bellocchi, L. Carlini, and M. Colauzzi (2005) CLIMA: a component-based weather generator. Proceedings of MODSIM 2005, Melbourne, 627-633.

44. Donatelli, M. and A.E. Rizzoli (2008) A Design for Framework-Independent Model Components of Biophysical Systems. Proceedings of the 4th Biennial Meeting of iEMSs, 727-734.
45. Donatelli, M., G. Bellocchi, E. Habyarimana, R. Confalonieri, and B. Baruth (2009) CLIMA: a weather generator framework. Proceedings of 18th World IMACS / MODSIM Congress, Cairns, 852-858.
46. Drenth, H., H.F.M. ten Berge, and J.J.M. Riethoven (1994) ORYZA simulation modules for potential and nitrogen limited rice production (SARP Research Proceedings). AB-DLO, TPE-WAU, IRRI, Wageningen, the Netherlands, 223pp., ISBN90-73384-28-1, with FSE source code of ORYZA_N and ORYZA_0.
47. Duthie, J.A. (1997) Models of the Response of Foliar Parasites to the Combined Effects of Temperature and Duration of Wetness. Phytopathology 87:1088-1095.
48. The Eclipse Foundation (2001) Eclipse. http://www.eclipse.org/
49. Ecma International (2009) ECMAScript Language Specification, 5th ed., 241pp. http://www.ecma-international.org/publications/standards/Ecma-262.htm
50. FAO (2010) 世界の食糧不安の現状 2008 年報告. 国際農林業協働協会, 東京, 56pp.
51. 消　防　庁 (2010) FDMA_JAPAN. http://twitter.com/FDMA_JAPAN
52. Fielding, R.T. (2000) Architectural Styles and the Design of Network-based Software Architectures. University of California Irvine, Dissertation.
53. Forrester, J.W. (1961) Industrial Dynamics. The MIT Press, Cambridge, USA, 464pp., ISBN0-262-56001-1.
54. Forrester, J.W. (1971) World Dynamics. Wright-Allen Press, Cambridge, USA, 142pp.
55. Fukai, S. and M. Cooper (1995) Development of drought-resistant cultivars using physiomorphological traits in rice. Field Crops Research 40(2):67-86, doi:10.1016/0378-4290(94)00096-U.
56. 深津時広・平藤雅之 (2003) 圃場モニタリングのためのフィールドサーバの開発. 農業情報研究 12(1):1-12, J-GLOBAL ID:200902251820992584.
57. Gao, L., Z. Jin, Y. Huang, and L. Zhang (1992) Rice Clock model - a computer model to simulate rice development. Agricultural and Forest Meteorology 60(1-2):1-16, doi:10.1016/0168-1923(92)90071-B.
58. Garner, W.W. and H.A. Allard (1920) Effect of the relative length of day and night and other factors of the environment on growth and reproduction in plants. Journal of Agricultural Research 18:553-606.
59. Garner, W.W. and H.A. Allard (1923) Further studies in photoperiodism, the response of the plant to relative length of day and night. Journal of Agricultural Research 23:871-920.
60. Garett, J.J. (2005) Ajax: A New Approach to Web Applications. http://www.adaptivepath.com/ideas/essays/archives/000385.php
61. Geng, S., F.W.T. Penning de Vries, and I. Supit (1986) A Simple Method for Generating Daily Rainfall Data. Agricultural and Forest Meteorology 36(4):363-376, doi:10.1016/0168-1923(86)90014-6.
62. GEO (2009) Group on Earth Observation. http://www.earthobservations.org/
63. GK TECH (2001) 触れる地球. http://www.tangible-earth.com/
64. Google (2001) Google 検索. http://www.google.co.jp/
65. Google (2005) Google Calendar. http://www.google.com/googlecalendar/about.html
66. Google (2005) Google マップ. http://maps.google.co.jp/
67. Google (2005) Google Maps API. http://code.google.com/apis/maps/
68. Google (2005) Google code. http://code.google.com/
69. Google (2006) Google Earth. http://earth.google.co.jp/
70. Google (2006) KML チュートリアル. http://code.google.com/apis/kml/documentation/kml_tut.html

71. Google (2006) Google SketchUp. http://sketchup.google.com/
72. Google (2006) Google Apps. http://www.google.com/apps/
73. Google (2006) Google Web Toolkit. http://code.google.com/webtoolkit/
74. Google (2008) Google App Engine. http://code.google.com/appengine/
75. Graf, B., O. Rakotobe, P. Zahner, V. Delucchi, and A.P. Gutierrez (1990) A Simulation Model for the Dynamics of Rice Growth and Development: Part I - The Carbon Balance. Agricultural Systems 32(4):341-365, doi:10.1016/0308-521X(90)90099-C.
76. Gregersen, J.B., P.J.A. Gijsbers, and S.J.P. Westen (2007) OpenMI: Open modelling interface. Journal of Hydroinformatics 9(3):175-191, doi:10.2166/hydro.2007.023.
77. 芳賀敏郎・橋本茂司 (1980) 非線形回帰，回帰分析と主成分分析．日科技連出版社，東京，204-217, ISBN4-8171-2011-8.
78. 長谷川浩 (1999) 圃場試験における土壌－作物系包括的シミュレーションモデル [14]. 農業および園芸 74(12):64-68.
79. Hasegawa, T. and T. Horie (1997) Modeling the effects of nitrogen nutrition on rice growth and development in *Applications of Systems Approaches at the Field Level*, vol.2, M.J. Kropff, P.S. Teng, P.K. Aggarwal, J. Bouma, B.A.M. Bouman, J.W. Jones, and H.H. van Laar, Eds. Kluwer Academic Publisher, Dordrecht, the Netherlands, 243-257, ISBN0-7923-4286-0.
80. Hasegawa, T., K. Tanaka, W. Takahashi, and R.L. Williams (2001) JAPONICA, a field-level rice growth model, and its use. Proceedings of the NIAES-STA International Workshop 2001 on Crop Monitoring and Prediction at Regional Scales:85-94.
81. 林孝・越水幸男 (1988) 葉いもち発生予察のコンピュータプログラム (BLASTAM) の開発．東北農業試験場研究報告 78:123-138.
82. 林陽生・石郷岡康史・横沢正幸・烏谷均・後藤慎吉 (2001) 温暖化が日本の水稲栽培の潜在的特性に及ぼすインパクト．地球環境 6(2):141-148.
83. Hillyer, C., J. Bolte, F. van Evert, and A. Lamaker (2003) The ModCom modular simulation system. European Journal of Agronomy 18(3-4):333-343, doi:10.1016/S1161-0301(02)00111-9.
84. 平藤雅之 (1999) 増殖情報ナビゲータ. http://agrinfo.narc.affrc.go.jp/
85. Hirafuji, M., K. Tanaka, T. Kiura, and A. Otsuka (2000) Modelbase System: A Distributed Model Database on The Internet. Proceedings of IWS2000 International Workshop on Asia Pacific Advanced Network and its Applications Application Area, Tsukuba, 57-61.
86. 平藤雅之 (2004) フィールドサーバによるユビキタス環境とセンサネットワークの構築．第18回 回路とシステム ワークショップ 講演要旨集，軽井沢，175-180.
87. 平藤雅之 (2005) Field Server. http://model.job.affrc.go.jp/FieldServer/default.htm
88. 平藤雅之 (2010) Sensor Cloud. http://twitter.com/sensorcloud
89. Hirano, S. (1997) HORB: Distributed execution of Java programs. Lecture Notes in Computer Science 1274/1997:29-42, doi:10.1007/3-540-63343-X_36.
90. Hirao, T., Y. Sasaki, and H. Isozaki (2001) An Extrinsic Evaluation for Question-Biased Text Summarization on QA tasks. Proceedings of NAACL 2001 workshop on Automatic Summarization, 61-68.
91. Holland, J.H. (1975) Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. University of Michigan Press, 206pp., ISBN0472084607.
92. 本田茂広・下村道彦・南石晃明・木浦卓治・二宮正士・大谷信博 (1998) 青果物市況情報データベースNAPASを用いたJAVA-RMIとHORBの性能比較．情報処理学会研報 98(15):43-48.
93. 堀江武 (1981) 気象と作物の光合成，蒸散そして生長に関するシステム生態学的研究．京都大学，博士論文．

94. 堀江武・中川博視 (1990) イネの発育過程のモデル化と予測に関する研究 第1報 モデルの基本構造とパラメータ推定法および出穂予測への適用. 日本作物学会紀事 59(4):687-695, J-GLOBAL ID:200902079851988600.

95. Horie, T., H. Nakagawa, H.G.S. Centeno, and M.J. Kropff (1995a) The Rice Crop Simulation Model SIMRIW and Its Testing in *Modeling the Impact of Climate Change on Rice Production in Asia*, R.B. Matthews, M.J. Kropff, D. Bachelet, and H.H. van Laar, Eds. CAB International, Wallingford, UK, ch.5, 51-66, ISBN0-85198-959-4.

96. Horie, T., H. Nakagawa, M. Ohnishi, and J. Nakano (1995b) Rice Production in Japan under Current and Future Climates in *Modeling the Impact of Climate Change on Rice Production in Asia*, R.B. Matthews, M.J. Kropff, D. Bachelet, and H.H. van Laar, Eds. CAB International, Wallingford, UK, ch.8, 143-164, ISBN0-85198-959-4.

97. 堀江武・井上直人・大西政夫・中川博視・松井勤 (1995) 水稲の生育・生産過程の動的予測モデルの開発. 平成6年度科学研究費補助金（一般研究A）研究成果報告書, 研究課題番号 03404007, SIMRIW の FORTRAN ソースコード付.

98. Horie, T., H. Yoshida, S. Kawatsu, K. Katsura, K. Homma, and T. Shiraiwa (2005) Effects of elevated atmospheric CO2 concentration and increased temperature on rice: implications for Asian rice production in *Rice is life: scientific perspectives for the 21st century*, IRRI, Manila, Philippines, 536-539, ISBN971-22-0204-6.

99. Hunt, L.A. and K.J. Boote (1998) Data for model operation, calibration, and evaluation in *Understanding Options for Agricultural Production*, G.Y. Tsuji, G. Hoogenboom, and P.K. Thornton, Eds. Kluwer Academic Publishers, Dordrecht, the Netherlands, 9-39, ISBN0-7923-4833-8.

100. Hunt, L.A., J.W. White, and G. Hoogenboom (2001) Agronomic data: advances in documentation and protocols for exchange and use. Agricultural Systems 70(2-3):477-492, doi:10.1016/S0308-521X(01)00056-7.

101. Hunt, L.A., G. Hoogenboom, J.W. Jones, and J.W. White (2006) ICASA Version 1.0 Data Standards for Agricultural Research and Decision Support. International Consortium for Agricultural System Applications.

102. IBM (1975) Continuous System Modeling Program III (CSMP III), Program Reference Manual., 206pp., SH19-7001-3.

103. ICASA (2004) International Consortium for Agricultural Systems Applications. http://www.icasa.net/

104. IPCC (2007) Climate Change 2007: The Physical Science Basis. Cambridge University Press, Cambridge, United Kingdom, ISBN978-0521-88009-1.

105. IRRI (2004) ORYZA2000. http://www.knowledgebank.irri.org/oryza2000/

106. ISO/IEC JTC1/SC22/WG5 (1982) Welcome to the official home of Fortran Standards. http://www.nag.co.uk/SC22WG5/

107. Iwaki, H. (1977) Computer Simulation of Growth Process of Paddy Rice. JARQ 11(1):6-11.

108. Izumi, T., M. Yokozawa, and M. Nishimori (2009) Parameter estimation and uncertainty analysis of a large-scale crop model for paddy rice: Application of a Bayesian approach. Agricultural and Forest Meteorology 149(2):333-348, doi:10.1016/j.agrformet.2008.08.015.

109. Jacobson, I., P. Jonsson, M. Christerson, and G. Overgaard (1995) オブジェクト指向ソフトウェア工学 OOSE. トッパン, 東京, 491pp., ISBN4-8101-8066-2.

110. Johnson, R.E. and V.F. Russo (1991) Reusing Object-Oriented Designs. University of Illinois technical report UIUCDCS:91-1696.

111. Johnson, R.E. (1997) Components, Frameworks, Patterns. ACM 40(10):39-42, doi:10.1145/258366.258378.

112. C.A. Jones and J.R. Kiniry, Eds. (1986) CERES-Maize: A Simulation Model of Maize Growth and Development. Texas A&M University Press, Texas, USA, 194pp., ISBN0-89096-269-3, with 5 inch floppy disk.

113. Jones, J.W., G.Y. Tsuji, G. Hoogenboom, L.A. Hunt, P.K. Thornton, P.W. Wilkens, D.T. Imamura, W.T. Bowen, and U. Singh (1998) Decision support system for agrotechnology transfer: DSSAT v3 in *Understanding Options for Agricultural Production*, G.Y. Tsuji, G. Hoogenboom, and P.K. Thornton, Eds. Kluwer Academic Publishers, Dordrecht, the Netherlands, 129-156, ISBN0-7923-4833-8.

114. Jones, J.W., B.A. Keating, and C.H. Porter (2001) Approaches to modular model development. Agricultural Systems 70(2-3):421-443, doi:10.1016/S0308-521X(01)00054-3.

115. Jones, J.W., G. Hoogenboom, C.H. Porter, K.J. Boote, W.D. Batchelor, L.A. Hunt, P.W. Wilkens, U. Singh, A.J. Gijsman, and J.T. Ritchie (2003) The DSSAT cropping system model. European Journal of Agronomy 18(3-4):235-265, doi:10.1016/S1161-0301(02)00107-7.

116. Keating, B.A., P.S. Carberry, G.L. Hammer, M.E. Probert, M.J. Robertson, D. Holzworth, N.I. Huth, J.N.G. Hargreaves, H. Meinke, Z. Hochman, *et al.* (2003) An overview of APSIM, a model designed for farming systems simulation. European Journal of Agronomy 18(3-4):267-288, doi:10.1016/S1161-0301(02)00108-9.

117. 桐谷圭治 (1997) 日本産昆虫，ダニ，線虫の発育零点と有効積算温度．農業環境技術研究所資料 21:1-72, ISSN0912-7542.

118. 岸田恭允 (1985) 九州・沖縄地方における日出没、南中、可照時間、薄明．九州農試研究資料 65:55, ISSN0453-0365, BASIC ソースコード付．

119. 気象庁（2002）アメダス．http://www.jma.go.jp/jp/amedas/

120. 喜多泰代 (2007) 二次元濃度ヒストグラムを用いた画像間変化抽出．電子情報通信学会論文誌 J90-D(8):1957-1965.

121. 北野宏 (1993) 遺伝的アルゴリズム．産業図書，東京，328pp., ISBN4782851367.

122. 木浦卓治 (1996) 開発ソフトウェア・データベース所在情報．http://agrinfo.narc.affrc.go.jp/fs/cdrom/shiryou/shiryou4/html/

123. Kiura, T., K. Tanaka, M.R. Laurenson, S. Honda, and M. Shimomura (2000) Distributed Agricultural Model System. Proceedings of AFITA2000 2nd Asian Conference for Information Technology in Agriculture, Suwon, 251-256.

124. Kobayashi, K. (1994) A very simple model of crop growth: derivation and application. International Rice Research Notes 19(3):50-51.

125. 近藤純正・中村亘・山崎剛 (1991) 日射量および下向き大気放射量の推定．天気 38(1):41-48.

126. Kouno, T., M. Ayabe, H. Hitomi, T. Machida, and S. Moriizumi (2000) Development of Relational System between Plant Pathology Database and Pesticide Database. Proceedings of AFITA2000 2nd Asian Conference for Information Technology in Agriculture, Suwon, 140-144.

127. Kouno, T., T. Machida, and R. Tagami (2002) Improvement of Crop Protection Support System "PaDB". -Addition of cultivation calendar and Distributed databases-. Agricultural Information Research 11(1):51-63.

128. Kralisch, S. and P. Krause (2006) JAMS - A Framework for Natural Resource Model Development and Application. Proceedings of the 3rd Biennial Meeting of iEMSs, Burlington.

129. Kropff, M.J., H.H. van Laar, and R.B. Matthews (1994) ORYZA 1, An ecophysiological model for irrigated rice production (SARP Research Proceedings). AB-DLO, TPE-WAU, IRRI, Wageningen, the Netherlands, 110pp.

130. 黒瀬義孝・丸山篤志・大場和彦 (2004) インターネットを使った小麦の出穂期，成熟期予測情報の公開．九州農業研究 66:23.

131. Kuwagata, T., T. Hamasaki, and T. Watanabe (2008) Modeling water temperature in a rice paddy for agro-environmental research. Agricultural and Forest Meteorology 148(11):1754-1766, doi:10.1016/j.agrformet.2008.06.011.

132. Larsen, G.A. and R.B. Pense (1981) Stochastic Simulation of Daily Climate Data. United States Department of Agriculture Statistical Reporting Service Research Division, Washington, D.C., Report No. AGES810831.

133. Laurenson, M.R., T. Kiura, and S. Ninomiya (2000) Accessing online weather databases from Java. Proceedings of IWS2000 International Workshop on Asia Pacific Advanced Network and its Applications Application Area, Tsukuba, 193-198.

134. Laurenson, M.R. (2001) MetBroker Programmers Guide. http://www.agmodel.net/Programmers-Guide.pdf

135. Laurenson, M.R., T. Kiura, and S. Ninomiya (2002) Providing Agricultural Models with Mediated Access to Heterogeneous Weather Databases. Applied Engineering in Agriculture 18(5):617-625.

136. Leavesley, G.H., P.J. Restrepo, L.G. Stannard, L.A. Frankoski, and A.M. Sautins (1996) The Modular Modeling System (MMS) - A Modeling Framework for Multidisciplinary Research and Operational Applications in *GIS and Environmental Modeling: Progress and Research Issues*, Wiley, Ft. Collins, USA, 155-158, ISBN0-470-23677-9.

137. Leffelaar, P.A. and T.J. Ferrari (1989) Some elements of dynamic simulation in *Simulation and systems management in crop protection*, R. Rabbinge, S.A. Ward, and H.H. van Laar, Eds. Pudoc, Wageningen, the Netherlands, ch.2.1, 19-45, ISBN90-220-0899-1.

138. Leffelaar, P.A. (1999) A simulation language: Continuous System Modeling Program III in *On Systems Analysis and Simulation of Ecological Processes with Examples in CSMP, FST and FORTRAN (Current Issues in Production Ecology)*, 2nd ed., P.A. Leffelaar, Ed. Kuluwer Academic Publishers, the Netherlands, Dordrecht, ch.3, 29-40, ISBN0-7923-5526-1.

139. Leffelaar, P.A., C. Rappoldt, and D.W.G. van Kraalingen (1999) Simulation using a general purpose computer language in *On Systems Analysis and Simulation of Ecological Processes with Examples in CSMP, FST and FORTRAN (Current Issues in Production Ecology)*, 2nd ed., P.A. Leffelaar, Ed. Kuluwer Academic Publishers, the Netherlands, Dordrecht, ch.10, 143-168, ISBN0-7923-5526-1.

140. Littleboy, M., D.M. Silburn, D.M. Freebairn, D.R. Woodruff, G.L. Hammer, and J.K. Leslie (1992) Impact of soil erosion on production in cropping systems. I. Development and validation of a simulation model. Australian Journal of Soil Research 30(5):757-774, doi:10.1071/SR9920757.

141. MacHardy, W.E. and D.M. Gardoury (1989) A Revision of Mills's Criteria for Predicting Apple Scab Infection Periods. Phytopathology 79:304-310.

142. Maclean, J.L., D.C. Dawe, B. Hardy, and G. Hettel (2002) Rice Almanac, 3rd ed. IRRI, Los Baños, Philippines, 253pp.

143. Magarey, R.D., J.M. Russo, R.C. Seem, and D.M. Gadoury (2005) Surface wetness duration under controlled environmental conditions. Agricultural and Forest Meteorology 128(1-2):111-122, doi:10.1016/j.agrformet.2004.07.017.

144. University of Minnesota (2008) MapServer. http://mapserver.org/

145. 丸山篤志・大場和彦・黒瀬義孝 (2002) 秋播性程度の異なるコムギ3品種のDVR法による出穂期予測. 九州農業研究 64:15.

146. Masaki, Y., T. Kuwagata, and Y. Ishigooka (2009) Precise estimation of hourly global solar radiation for micrometeorological analysis by using data classification and hourly sunshine. Theoretical and Applied Climatology 100(3-4):283-297, doi:10.1007/s00704-009-0191-0.

147. 増井俊之 (1999) インターフェイスの街角 (16) 地図データベースの活用, UNIX MAGAZINE., http://pitecan.com/articles/UnixMagazine/PDF/if9903.pdf.

148. 松田昭美 (1959) 暖地における水稲栽培と気象要因の統計学的研究 (1). 農業気象 15(1):15-20.

149. McCown, R.L. and J. Williams (1989) AUSIM: a cropping systems model for operational research. Proceedings of the Simulation Society of Australia and International Association for Mathematics and Computers in Simulation 8th Biennial Conference, Canberra, 54–59.

150. McCown, R.L., G.L. Hammer, J.N.G. Hargreaves, D.P. Holzworth, and D.M. Freebairn (1996) APSIM: a Novel Software System for Model Development, Model Testing and Simulation in Agricultural Systems Research. Agricultural Systems 50(3):255-271, doi:10.1016/0308-521X(94)00055-V.

151. McMennamy, J.A. and J.C. O'Toole (1983) RICEMOD: a physiologically based rice growth and yield model, IRRI Research Paper Series 87. IRRI, Los Baños, Philippines, 33pp.

152. Meadows, H.D., L.D. Meadows, J. Randers, and W.W. Behrens III (1972) 成長の限界．ダイヤモンド社，東京，203pp., ISBN4478200017.

153. Microsoft (2008) Microsoft Online Services. http://www.microsoft.com/online/

154. Microsoft (2010) Microsoft Windows Azure. http://www.microsoft.com/windowsazure/

155. Mills, W.D. and A.A. Laplante (1951) Diseases and insects in the orchard. Cornell Extension Bulletin 711:20-28.

156. Mills, E. (2005) Mapping a revolution with 'mashups'. http://news.com.com/2009-1025_3-5944608.html

157. Monsi, M. and T. Saeki (1953) Über den Lichtfaktor in den Pfanzengesellschaften und seine Bedeutung für die Stoffproduktion. Japanese Journal of Botany 14:22-52.

158. Monteith, J.L. (1972) Solar Radiation and Productivity in Tropical Ecosystems. The Journal of Applied Ecology 9(3):747-766.

159. Monteith, J.L. and C.J. Moss (1977) Climate and the Efficiency of Crop Production in Britain. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 281(980):277-294.

160. Mori, T., M. Nozawa, and Y. Asada (2005) Multi-Answer-Focused Multi-Document Summarization Using a Question-Answering Engine. ACM Transactions on Asian Language Information Processing (TALIP) 4(3):305-320, doi:10.1145/1111667.1111672.

161. Mozilla (2005) Firebug. http://getfirebug.com/

162. Munakata, K. (1976) Effects of temperature and light on the reproductive growth and ripening of rice. Proceedings of the Symposium on Climate & Rice, IRRI, 187-207.

163. Nakagawa, H., K. Sudo, and T. Horie (1993) A Prediction System of Rice Development Stage for the Crop Management. Proceedings of First Asian Crop Science Conference, Korea, 243-249.

164. 中川博視・堀江武 (1995) イネの発育過程のモデル化と予測に関する研究 第2報 幼穂の分化・発達過程の気象的予測モデル．日本作物学会紀事 64(1):33-42, J-GLOBAL ID:200902155333059180.

165. Nakagawa, H., T. Horie, and T. Matsui (2003) Effects of climate change on rice production and adaptive technologies in *Rice Science: Innovations and Impact for Livelihood, Proceedings of the International Rice Research Conference*, T.W. Mew, D.S. Brar, S. Peng, D. Dawe, and B. Hardy, Eds. IRRI, Chinese Academy of Engineering, Chinese Academy of Agricultural Sciences, Manila, Philippines, 635-658, ISBN971-22-01848.

166. Nelson, R. (2002) Description of ClimGen, a Weather Generation Program. http:// www.bsyse.wsu.edu/climgen/documentation/description.htm.

167. 二宮正士 (2000) インターネットを活用した農業判断支援構築に向けて．農林水産技術研究ジャーナル 23(4):46-54.

168. 二宮正士 (2001) インターネット上の気象データの効率利用を可能にする支援ソフト MetBroker. 農林水産技術研究ジャーナル 24(3):9-12.

169. 二宮正士 (2001) IT農業に向けた技術開発の考え方．農林水産研究ジャーナル 24(7):16-23.

170. 二宮正士 (2001) 農林水産省を中心とする農業IT研究プロジェクトの現状と今後の課題．農業機械学会誌 63(4):4-11.

171. 二宮正士・LaurensonMatthew・木浦卓治 (2003) 分散協調型農業情報システムと気象データ利用仲介ソフトウェア MetBroker. 農業および園芸 78(1):175-181.

172. Nix, H.A. (1984) Minimum Data Sets for Agrotechnology Transfer. Proceedings of the International Symposium on Minimum Data Sets for Agrotechnology Transfer, ICRISAT, Patancheru, India, 181-188.

173. NOAA (2004) National Weather Service. http://www.weather.gov/
174. Noguchi, Y. (1981) Solar Radiation and Sunshine Duration in East Asia. Archives for Meteorology, Geophysics, and Bioclimatology 29(1-2):111-128. doi:10.1007/BF02278195.
175. 農研センター；現 農研機構 中央農研 (1999) 増殖情報ベース. http://agrinfo.narc.affrc.go.jp/zousoku.htm
176. 農研機構 中央農研 (2000) AgModel. http://www.agmodel.org/
177. 農研機構 中央農研 (2000) MetBroker. http://www.agmodel.org/projects/metbroker.html
178. 農研機構 中央農研 (2000) ChizuBroker. http://www.agmodel.org/projects/chizubroker.html
179. 農研機構 中央農研 (2005) データベース・モデル協調システム. http://www.agmodel.net/DataModel/
180. 農研機構 中央農研 (2005) イネデータベース. http://www.agmodel.org/RiceDB/
181. 農林水産省 (2003) 農林業家のパソコン・インターネットの利用等に関する意向調査結果.
182. 農林水産省 農林水産技術会議事務局 (2007) データベース・モデル協調システムの開発. 研究成果448., 139pp.
183. 農林水産省 (2008) 地域のITを取り入れた農林水産業の取組事例. http://www.maff.go.jp/j/finding/zirei/08_it/
184. NTTドコモ (2006) 作ろうiモードコンテンツ. http://www.nttdocomo.co.jp/service/imode/make/
185. Object Refinery Limited (2005) JFreeChart. http://www.jfree.org/jfreechart/
186. OGC (2002) OpenGIS Web Map Service (WMS) Implementation Specification. OGC01-068r3., http://www.opengeospatial.org/standards/wms.
187. OGC (2008) OGC KML. OGC07-147r2, http://www.opengeospatial.org/standards/kml.
188. OMG (1991) CORBA. http://www.corba.org/
189. OMG (1994) Unified Modeling Language. http://www.uml.org/
190. OMG (1997) Object Management Group. http://www.omg.org/
191. OMG (2002) Model Driven Architecture. http://www.omg.org/mda/
192. OMG (2008) Common Object Request Broker Archtecture (CORBA) Specification., 540pp.
193. 大谷徹・田中慶・菅原幸治・LaurensonMatthew・渡邊朋也・梅本清作 (2001) ナシ黒星病の感染予測モデルによる発生予察の試み. 農業情報学 3:39-41.
194. Otsuka, A. and S. Ninomiya (1998) Web-based, Conceptual Retrieval System of Agricultural Cases. Proceedings of AFITA1998 1st Asian Conference for Information Technology in Agriculture, Wakayama, 203-206.
195. Papajorgji, P.J. and T.M. Shatar (2004) Using the Unified Modeling Language to develop soil water-balance and irrigation-scheduling models. Environmental Modelling & Software 19:451-459, doi:10.1016/S1364-8152(03)00160-9.
196. Papajorgji, P.J., H.W. Beck, and J.L. Braga (2004) An architecture for developing service-oriented and component-based environmental models. Ecological Modelling 179(1):61-76, doi:10.1016/j.ecolmodel.2004.05.013.
197. Papajorgji, P.J. and P.M. Pardalos (2006) Software Engineering Techniques Applied to Agricultural Systems. Springer, New York, USA, 247pp., ISBN0-387-28170-3.
198. Papajorgji, P.J., R. Clark, and E. Jallas (2009) The Model Driven Architecture Approach: A Framework for Dveloping Complex Agricultural Systems in *Advances in Modeling Agricultural Systems*. Springer, New York, USA, 1-18, ISBN0-387-75180-7.
199. Penning de Vries, F.W.T. (1980) Simulation models of growth of crops, particularly under nutrient stress. Proceedings of the 15th Colloquium of the International Potash Institute, Wageningen, 213-226.
200. Penning de Vries, F.W.T. (1982) Systems analysis and models of crop growth in *Simulation of plant growth and crop production*, F.W.T. Penning de Vries and H.H. van Laar, Eds. Pudoc, Wageningen, the Netherlands, ch.1.2, 9-19, ISBN90-220-0809-6.

201. Penning de Vries, F.W.T., D.M. Jansen, H.F.M. ten Berge, and A. Bakema (1989) Simulation of ecophysiological processes of growth in several annual crops. Pudoc, Wageningen, the Netherlands, 271pp., ISBN90-220-0937-8, with CSMP source code of MACROS.

202. Pickering, N.B., J.W. Hansen, J.W. Jones, C.M. Wells, V.K. Chan, and D.C. Godwin (1994) WeatherMan: A Utility for Managing and Generating Daily Weather Data. Agronomy Journal 86:332-337.

203. Plentinger, M.C. and F.W.T. Penning de Vries (1996) CAMASE, Register of Agro-ecosystems Models. AB-DLO, Wageningen, the Netherlands, 420pp.

204. Porter, C.H. and J.W. Jones (1998) Module Structure in *DSSAT v4 Crop Model Documentation*, vol.4, J.W. Jones, G. Hoogenboom, K.J. Boote, and C.H. Porter, Eds. ICASA, University of Hawaii, Hawaii, USA, ch.Appendix C, 7.

205. Prototype Core Team (2006) Prototype JavaScript Framework. http://www.prototypejs.org/

206. Rahman, J.M., S.P. Seaton, and S.M. Cuddy (2004) Making Frameworks More Useable: Using Model Introspection and Metadata to Develop Model Processing Tools. Environmental Modelling and Software 19(3):275-284, doi:10.1016/S1364-8152(03)00153-1.

207. Ramsey, P. (2006) Mashing Up the Enterprise. http://www.gpsworld.com/gis/integration-and-standards/mashing-up-enterprise-7450

208. Rappoldt, C. and D.W.G. Kraalingen (1996) The Fortran Simulation Translator FST version 2.0. Introduction and Reference Manual. AB-DLO, Wageningen, Quantitative Approaches in Systems Analysis No. 5.

209. Reynolds, J.F. and B. Acock (1997) Modularity and genericness in plant and ecosystem models. Ecological Modelling 94(1):7-16, doi:10.1016/S0304-3800(96)01924-2.

210. Richardson, C.W. (1981) Stochastic simulation of daily precipitation, temperature, and solar radiation. Water Resources Research 17(1):182-190, doi:10.1029/WR017i001p00182.

211. Richardson, C.W. and D.A. Wright (1984) WGEN: A Model for Generating Daily Weather Variables. United States Department of Agriculture Agriculture Research Service, ARS-8, with FORTRAN source code of WGEN.

212. Richardson, G.P. and A.L. Pugh III (1981) Introduction to System Dynamics Modeling. Productivity Press, Portland, USA, 413pp., ISBN0-915299-24-0.

213. Ritchie, J.T. (1981) Soil Water Availability. Plant and Soil 58(1-3):327-338, doi:10.1007/BF02180061.

214. Ritchie, J.T., E.C. Alocilja, U. Singh, and G. Uehara (1987) IBSNAT and the CERES-Rice model in *Weather and Rice*. IRRI, Los Baños, Philippines, 271-281, ISBN971-104-178-2.

215. Ritchie, J.T., D.C. Godwin, and S. Otter-Nacke (1988) CERES-wheat: A Simulation Model of Wheat Growth and Development. Texas A&M University Press, Texas, USA.

216. Ritchie, J.T., U. Singh, D.C. Godwin, and W.T. Bowen (1998) Cereal growth, development and yield in *Understanding Options for Agricultural Production*, G.Y. Tsuji, G. Hoogenboom, and P.K. Thornton, Eds. Kluwer Academic Publishers, Dordrecht, the Netherlands, 79-98, ISBN0-7923-4833-8.

217. Rizzoli, A.E., G.H. Leavesley, J.C. Ascough II, R.M. Argent, I.N. Athanasiadis, V. Brilhante, F.H.A. Claeys, O. David, M. Donatelli, P. Gijsbers, *et al.* (2008) Integrated Modelling Frameworks for Environmental Assessment and Decision Support in *Environmental Modelling, Software and Decision Support*, A.J. Jakeman, A.A. Voinov, A.E. Rizzoli, and S.H. Chen, Eds. Elsevier, Amsterdam, the Netherlands, ch.7, 101-118, ISBN0-08-056886-6.

218. Rumbaugh, J.R., M.R. Blaha, W. Lorensen, F. Eddy, and W. Premerlani (1992) オブジェクト指向方法論 OMT. トッパン, 東京, 544pp., ISBN4-8101-8527-0.

219. Sagan, C. (2008) 20世紀 in 百億の星と千億の生命. 新潮社, 東京, ch.18, 357-372, ISBN978-4-10-229405-5.

220. salesforce.com (2008) Force.com. http://www.salesforce.com/platform/

221. 佐藤理史 (1998) 情報検索．長尾真・黒橋禎夫・佐藤理史・池原悟・中野洋 編．言語情報処理．言語の科学 9 巻．岩波書店．東京．第 2 章．51-93. ISBN4-00-010859-X.
222. Schmidt, E. (2006) A Conversation With Google CEO Eric Schmidt. Proceedings of Search Engine Strategies Conference, San Jose.
223. 清野豁 (1993) アメダスデータのメッシュ化について．農業気象 48(4):379-383.
224. Seligman, N.G. (1990) The crop model record: promise or poor show? in *Theoretical Production Ecology: Reflections and Prospects*, R. Rabbinge, J. Goudriaan, H. van Keulen, F.W.T. Penning de Vries, and H.H. van Laar, Eds. Pudoc, Wageningen, the Netherlands, 249-258. ISBN90-220-1004-X.
225. Sencha (2006) Ext JS. http://www.sencha.com/products/js/
226. Sequeira, R.A., P.J.H. Sharpe, N.D. Stone, K.M. El-Zik, and M.E. Makela (1991) Object-oriented simulation; plant growth and discrete organ to organ interactions. Ecological Modelling 58(1-4):55-89. doi:10.1016/0304-3800(91)90030-5.
227. Stöckle, C.O., S.A. Martin, and G.S. Campbell (1994) CropSyst, a Cropping Systems Simulation Model: Water/Nitrogen Budgets and Crop Yield. Agricultural Systems 46(3):335-359. doi:10.1016/0308-521X(94)90006-2.
228. Stöckle, C.O., M. Donatelli, and R. Nelson (2003) CropSyst, a cropping systems simulation model. European Journal of Agronomy 18(3-4):289-307. doi:10.1016/S1161-0301(02)00109-0.
229. 須藤健一・牛尾昭浩・吉田智一・高橋英博・寺元郁博 (2010) 兵庫県立農林水産技術総合センター研究報告（農業）58:31-35.
230. 菅原幸治・田中慶 (2008) 病害発生予測情報の RSS 自動配信システム．農研機構．成果情報．
231. 杉原保幸・羽生寿郎 (1980) 水稲の気候生産力の評価に関する研究 I. 水稲の気候生産力評価の試み．農業気象 36(2):71-79.
232. 杉村昌彦 (2009) イネの栽培可能性予測シミュレーター．http://dias.tkl.iis.u-tokyo.ac.jp/simriw/jp/
233. 杉浦俊彦 (1997) ニホンナシの気象生体反応の解析と生育予測モデルの開発．京都大学学位論文．
234. 杉浦俊彦・黒田治之・杉浦裕義 (2007) 温暖化がわが国の果樹生育に及ぼしている影響の現状．園芸学研究 6(2):257-263.
235. Sun Microsystems (1995) The Source for Java Developers. http://java.sun.com/
236. Sun Microsystems (1999) NetBeans. http://netbeans.org/
237. Sun Microsystems (1999) Java RMI over IIOP. http://java.sun.com/products/rmi-iiop/
238. Sun Microsystems (2003) JAXB Reference Implementation. https://jaxb.dev.java.net/
239. Sun Microsystems (2003) Java API for XML-based RPC JAX-RPC 1.1 Sprcifications., 167pp., https:// jax-rpc.dev.java.net/.
240. Sun Microsystesms (2003) JavaServer Faces. https://javaserverfaces.dev.java.net/
241. Sun Microsystems (2004) Java Remote Method Invocation Specification., 121pp.
242. Sun Microsystems (2005) The Java API for XML Web Services (JAX-WS) 2.0., 143pp., https:// jax-ws.dev.java.net/.
243. Sun Microsystems (2009) JavaServer Pages Specification Version 2.2., 594pp., http://java.sun.com/products/jsp/.
244. Sun Microsystems (2009) Java Servlet Specification Version 3.0., 230pp.
245. 鈴木剛伸・木浦卓治・菅原幸治 (2000) Java を利用したリンゴ黒星病感染予測モデルの作成 – 情報の共有化をめざしたモデルの作成方法．農業情報学 2:100-103.
246. 高橋渉・長谷川利拡・田中慶 (1998) 窒素の影響を考慮した水稲生長モデルの GUI の開発とその検証：JAPONICA の開発と富山県における施肥試験への適用．日本作物学会紀事 67(別号 2):126-127. naid:110001726385.
247. 高柳繁 (1991) 雑草害早期診断法開発のためのメヒシバとダイズ単植群落の成長・発育モデルの策定．雑草研究 36(4):372-379.
248. 田中慶 (1997) Java による作物生育、病害虫・雑草発生予測モデル．http://cse.naro.affrc.go.jp/ketanaka/model/

249. 田 中 慶 (1997) Java Agricultural Model Framework API Documentation. http://cse.naro.affrc.go.jp/ketanaka/model/docs/amf/
250. 田中慶・林孝 (2002) MetBrokerに対応した葉いもち感染好適日推定プログラムJava版MetBLASTAM. 農研機構, 成果情報.
251. 田中慶 (2004) 各種モデルを定期的に実行し結果を配信する自動システム. 農林水産技術会議, 成果情報.
252. 田中慶 (2004) Field Server Data & Image Viewer. http://pc105.narc.affrc.go.jp/fieldserver/
253. 田中慶 (2006) Javaによる作物生育・病害虫発生予測モデル開発のためのフレームワーク. 農業情報研究 15(2):183-194, doi:10.3173/air.15.183.
254. Tanaka, K., T. Fukatsu, and M. Hirafuji (2006) Data and Image Viewer Application for FieldServer. Proceedings of SICE-ICASE International Joint Conference 2006, Busan, 4852-4855.
255. Tanaka, K. (2006) The Utility Web Applications for MetBroker. Proceedings of AFITA2006 5th International Conference of the Asian Federation of Information Technology in Agriculture, Bangalore, 603-609.
256. Tanaka, K., Y. Kita, M. Hirafuji, and S. Ninomiya (2008) An Image Change Detection Application for Field Server. Proceedings of World Conference on Agricultural Information and IT / IAALD AFITA WCCA2008, Atsugi, 49-54.
257. 田 中 慶 (2008a) SIMRIW. http://pc105.narc.affrc.go.jp/simriw/
258. 田中慶 (2008) MetBrokerから取得した気象データをGoogle Earth上に表示するWebアプリ. 農研機構, 成果情報.
259. 田中慶・平藤雅之 (2009) 農業モデルにおけるWeb地図サービスを利用した地図インタフェース. 農業情報研究 18(2):98-109, doi:10.3173/air.18.98.
260. Tanaka, K., T. Kiura, M. Sugimura, S. Ninomiya, and M. Mizoguchi (2010) Tool for Predicting the Possibility of Rice Cultivation using SIMRIW. Proceedings of AFITA 2010 International Conference, Bogor, 199-204.
261. 田中慶 (2010) アメダスなどの気象データをXML形式でアプリに取り込めるMetXML. 農研機構, 成果情報.
262. Tang, L., Y. Zhu, D. Hannaway, Y. Meng, L. Liu, L. Chen, and W. Cao (2009) RiceGrow: A rice growth and productivity model. Wageningen Journal of Life Science 57(1):83-92, doi:10.1016/j.njas.2009.12.003.
263. ten Berge, H.F.M., M.C.S. Wopereis, J.J.M. Riethoven, T.M. Thiyagarajan, and R. Sivasamy (1994) ORYZA_0 model applied to optimize N use in rice in *N Economy of Irrigated Rice: Field and Simulation Studies (SARP Research Proceedings)*, H.F.M. ten Berge, M.C.S. Wopereis, and J.C. Shin, Eds. AB-DLO, TPE-WAU, IRRI, Wageningen, the Netherlands, 235-253.
264. Teng, P.S. and S. Savary (1992) Implementing the Systems Approach in Pest Management. Agricultural Systems 40(1-3):237-264, doi:10.1016/0308-521X(92)90023-H.
265. Teng, P.S., W.D. Bachelor, H.O. Pinnschmidt, and G.G. Wilkerson (1998) Simulation of pest effects on crops using coupled pest-crop models: the potential for decision support in *Understanding Options for Agricultural Production*, G.Y. Tsuji, G. Hoogenboom, and P.K. Thornton, Eds. Kluwer Academic Publishers, Dordrecht, the Netherlands, 221-266, ISBN0-7923-4833-8.
266. 戸松豊和 (1997) フレームワーク, JAVAプログラムデザイン. ソフトバンク パブリッシング, 東京, 第6章, 149-160, ISBN4-7973-0221-6.
267. Tsuji, G.Y., A. du Toit, A. Jintrawet, J.W. Jones, W.T. Bowen, R.M. Ogoshi, and G. Uehara (2002) Benefit of Models in Research and Decision Support: The IBSNAT Experience in *Agricultural System Models in Field Research and Technology Transfer*, L.R. Ahuja, L. Ma, and T.A. Howell, Eds. Lewis Publishers, Florida, USA, ch.5, 71-89, ISBN1-56670-563-0.
268. つくば市 (2004) つくば市 災害通知メールサービス. http://www1.city.tsukuba.ibaraki.jp/MailSys/disaster/

269. Uehara, G. and G.Y. Tsuji (1993) The IBSNAT Project in *Systems Approaches for Agricultural Development*, F.W.T. Penning de Vries, P. Teng, and K. Metselaar, Eds. Kluwer Academic Publishers, Dordrecht, the Netherlands, 505-513, ISBN0-7923-1880-3.

270. Uehara, G. and G.Y. Tsuji (1998) Overview of IBSNAT in *Understanding Options for Agricultural Production*, G.Y. Tsuji, G. Hoogenboom, and P.K. Thornton, Eds. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1-7, ISBN0-7923-4833-8.

271. 浦昭二 編 (1990) FORTRAN77 入門，第改訂版，培風館，東京，424pp., ISBN4-563-01358-7.

272. van der Plank, J.E. (1963) Plant Disease: Epidemics and Control. Academic Press, New York, USA, 349pp.

273. van Evert, F.K. and G.S. Campbell (1994) CropSyst: A Collection of Object-Oriented Simulation Models of Agricultural Systems. Agronomy Journal 86:325-331.

274. van Evert, F. and B. Rutgers (2005) Getting started with the.NET version of MODCOM.

275. van Keulen, H., F.W.T. Penning de Vries, and E.M. Drees (1982) A summary model for crop growth in *Simulation of plant growth and crop production*. Pudoc, Wageningen, the Netherlands, ch.3.1, 87-97, ISBN90-220-0809-6, with FST source code of SUCROS.

276. H. van Keulen and J. Wolf, Eds. (1986) Modelling of agricultural production: weather, soils and crops, 9022008584th ed. Pudoc, Wageningen, the Netherlands, 479pp.

277. van Kraalingen, D.W.G., W. Stol, P.W.J. Uithol, and M.G.M. Verbeek (1991) User manual of AB/TPE Weather System. AB-DLO, Wageningen, AB/TPE internal communication.

278. van Kraalingen, D.W.G., C. Rappoldt, and H.H. van Laar (1994) The Fortran Simulation Translator (FST), a simulation language in *Modelling Potential Crop Growth Processes*, J. Goudriaan and H.H. van Laar, Eds. Kluwer Academic Publishers, Dordrecht, the Netherlands, ch.Appendix 5, 219-230, ISBN0-7923-3220-2.

279. van Kraalingen, D.W.G. (1995) The FSE system for crop simulation, version 2.1. AB-DLO, Wageningen, Simulation Reports AB-TPE.

280. van Kraalingen, D.W.G. and C. Rappoldt (2000) Reference manual of the FORTRAN utility library TTUTIL v.4. AB-DLO, Wageningen, Quantitative Approaches in System Analysis.

281. van Laar, H.H., J. Goudriaan, and H. van Keulen (1997) SUCROS97: Simulation of crop growth for potential and water-limited production situations. Quantitative Approaches in Systems Analysis 14:58, with FST source code of SUCROS97.

282. W3C (1996) Extensible Markup Language (XML). http://www.w3.org/XML/

283. W3C (1999) XSL Transformations (XSLT). http://www.w3.org/TR/xslt

284. W3C (2000) SOAP Specifications. http://www.w3.org/TR/soap/

285. W3C (2006) XML Schema. http://www.w3.org/XML/Schema

286. Wageningen University (1997) SUCROS. http://www.csa.wur.nl/UK/Downloads/SUCROS/

287. Wageningen UR (2003) MODCOM - Framework for Component-based Modeling. http://www.modcom.wur.nl/

288. Wageningen UR (2005) FSTWin/FSEWin/FSEWinRunOnly. http://www.csa.wur.nl/UK/Downloads/FSTWin+FSEWin+FSEWinRunOnly/

289. Wallach, D. (1984) The Organization of Agronomic Experiment Data for Crop Modeling. Proceedings of the International Symposium on Minimum Data Sets for Agrotechnology Transfer, ICRISAT, Patancheru, India, 147-154.

290. Washington State University (2003) CropSyst. http://www.bsyse.wsu.edu/CS_Suite/CropSyst/

291. 渡邊朋也・田中慶 (2002) 長距離移動性イネウンカ類の近年の飛来傾向と気象解析に関する新しい取り組み，植物防疫 56:1-4.

292. Weihong, L. and J. Goudriaan (1991) Leaf wetness in rice crop caused by dew formation: a simulation study in *Simulation and systems analysis for rice production (SARP)*, F.W.T. Penning de Vries, H.H. van Laar, and M.J. Kropff, Eds. Pudoc, Wageningen, the Netherlands, 320-327, ISBN90-220-1059-7.

293. Welch, S.M. (1984) Developments in Computer-Based IPM Extension Delivery Systems. Annual Review of Entomology 29:359-381, doi:10.1146/annurev.en.29.010184.002043.

294. Wiederhold, G. (1992) Mediators in the architecture of future information systems. IEEE Computer 25(3):38-49, doi:10.1109/2.121508.

295. Wiederhold, G. (1999) Mediation to deal with heterogeneous data sources. Proceedings of 2nd International Conference, INTEROP'99, Zurich, 1-16, ISBN3-540-65725-8.

296. Wilkens, P.W. (2004) Weather Data Editing Program (Weatherman) in *DSSAT v4 Data Management and Analysis Tools*, vol.2, P.W. Wilkens, G. Hoogenboom, C.H. Porter, J.W. Jones, and O. Uryasev, Eds. University of Hawaii, Honolulu, USA, ch.4, 56, ISBN1-886684-07-3.

297. Williams, R.L. (2002) TRYM: a simplified process model in use by the NSW rice industry in *Modelling Irrigated Cropping Systems, with Special Attention to Rice–Wheat Sequences and Raised Bed Planting*, E. Humphreys and J. Timsina, Eds. CSIRO Land and Water Technical Report 25/02, 51-56.

298. WMO (2004) World Meteorological Organization. http://www.wmo.int/

299. WMO (2008) Guide to Agricultural Meteorological Practices (GAMP). WMO-No.134.

300. Wopereis, M.C.S., B.A.M. Bouman, T.P. Tuong, H.F.M. ten Berge, and M.J. Kropff (1996) ORYZA_W: Rice growth model for irrigated and rainfed environments (SARP Research Proceedings). AB-DLO, TPE-WAU, IRRI, Wageningen, the Netherlands, 159pp., ISBN90-73384-39-7, with FSE source code of ORYZA_W.

301. WRDC (2002) World Radiation Data Centre. http://wrdc-mgo.nrel.gov/

302. Wu, G.W. and L.T. Wilson (1998) Parameterization, Verification, and Validation of a Physiologically Complex Age-structured Rice Simulation Model. Agricultural Systems 56(4):483-511, doi:10.1016/S0308-521X(97)00070-X.

303. Yahoo! (1997) Yahoo! ディレクトリ検索. http://dir.yahoo.co.jp/

304. Yahoo! Developer Network (2005) YUI Library. http://developer.yahoo.com/yui/

305. 矢島正晴・寺島一男・丸山幸夫 (1999) 我が国で栽培されている水稲主要品種の発育パラメータ. 日本作物學會紀事 68(別号 2):64-65.

306. Yin, X.Y. and C.H. Qi (1994) Studies on the rice growth calendar simulation model(RICAM) and its application. Acta Agronomica Sinica 20(3):339-346.

307. 吉田智一・高橋英博・大原源二 (2003) XML Web Service 化によるレガシーモデルの統合. 農業情報研究 12(1):13-24, J-GLOBAL ID:200902257937045398.

308. Yu, X., A. Yamakawa, T. Kiura, T. Hasegawa, and S. Ninomiya (2007) CROWIS: A System for Sharing and Integrating Crop and Weather Data. 農業情報研究 16(3):124-131, J-GLOBAL ID:200902208580433342.

309. Zeigler, B.P., H. Praehofer, and T.G. Kim (2000) Theory of Modeling and Simulation, 2nd ed. Academic Press, San Diego, USA, 510pp., ISBN0-12-778455-1.

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．

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

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.