A component to simulate agricultural management

Marcello Donatelli, CRA-ISCI, Bologna, Italy (m.donatelli@isci.it)
Frits K. van Evert, PRI, Wageningen, The Netherlands
Andrea Di Guardo, Informatica Ambientale S.r.l., Milano, Italy
Myriam Adam, INRA, Toulouse, France
Kamal Kansou, INRA, Montpellier, France

Abstract: Quantifying the impact of agricultural management on production and system externalities is the goal of many agricultural modeling studies. Here we consider only those drivers of farmers' decision making that are based on the state of the agricultural system. Agricultural management must be simulated in such a way to mimic as closely as possible farmers' behaviour. Limiting the drivers of the decision making process to the biophysical system implies that each action must be triggered at run time via a set of rules which can be based on the state of the system, on constraints of resources availability, and on the physical characteristics of the system. Furthermore, the implementation of the management simulation must account for a broad range of actions within each of the typologies of management. Simulation of complex systems is increasingly being implemented using a modular, component based approach. Implementing the simulation of management in a component based system poses challenges in defining a framework which must be reusable and able to account for a variety of agricultural management technologies applied to different enterprises. Furthermore, the implementation of management must allow using different approaches to model its impact on different model components. This paper presents a conceptual framework and a reusable software component to implement agricultural management in simulation systems. The framework is based on "rules" and "impacts", and it is extensible for both. A generic proof of concept is presented, and an application in a ModCom project is also described.

***Keywords**:* Agricultural management, component-oriented programming, modelling, expert knowledge

1. INTRODUCTION

Many models developed to simulate agricultural production activities target the analysis/evaluation of agricultural management impacts on production and system externalities (Brisson et al., 2003; Keating et al., 2003; Jones et al., 2003; Stockle et al., 2003). All these models use a proprietary ontology to define management events and they embed in their systems part of the information needed to model the relevant impact. Moreover, the implementation of the management handling is hard-coded; changes in the models (for instance to account for a new management type) requires coding it in the modelling systems.

To evaluate alternate management options accounting for weather variability, typically a multiple year sample of weather is used; hence, the unattended implementation of management events in a simulation must account for system variables values at run time. Limiting the drivers of the decision making process to the biophysical system implies that each action must be triggered via a set of rules which are based on the state of the system, constraints of resources availability, and physical characteristics of the system. Furthermore, the implementation of the management simulation must account for a broad range of actions within each of the typologies of management (e.g. tillage operations within tillage).

In the last decade there has been an increasing demand for modularity and interchangeability in biophysical model development (e.g. Jones et al., 2001; David et al., 2002), aiming at improving the efficiency of use of resources, at fostering higher quality of modelling units via specialization, and at allowing comparison of alternate approaches to process simulation. The concept of developing modular systems for biophysical simulation has lead to the development of several modelling frameworks (e.g. Simile, ModCom, IMA, TIME, OpenMI, SME, OMS, as listed in Argent and Rizzoli, 2004,), which allow the construction of whole-system models by linking independent component models.

Implementing the simulation of management in a component based system requires: 1) a formalization which allows accounting for expert knowledge to configure production technologies, and 2) a framework capable of accounting for a variety of agro-technologies applied to different agricultural activities. Such a component must allow the implementation of management using different approaches to model its impact via different model components, and it should be extensible to account for new model requirements.

This paper presents an approach and a software component to implement management in a component based simulation system.

2. THE RULE-IMPACT APPROACH

An agricultural production activity comprises one or more production enterprises (e.g. crops in a rotation, an orchard). Each production enterprise is managed using a production technique, which is a set of planned actions. An action will be executed whenever all of a set of conditions (a "rule") are

met; whenever that happens, a management event is fired. The management event is quantified via a set of parameters (an "impact" – not to be confused with rule parameters) to model the effect of its implementation in the system via a model.

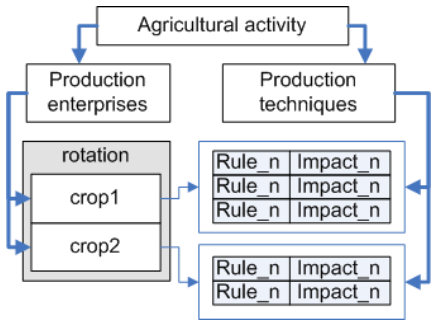

Fig. 1 Example configuration of management for a two years rotation: each crop has a set of rules to be tested each year of the simulation when the crop is planted (i.e. the sequence of crops is repeated over time)

This framework allows scheduling automatic management for any agricultural production. When new expert knowledge becomes available to trigger events, it can be formalized as a new "rule".
If new modelling knowledge can be implemented in a model component, the needed parameters can be made available via a new "impact".

2.1 Rules

Rules are a formal way to model farmers' behaviour. A rule based model is characterized by 3 main sections:

- Inputs: state of the system, and time (e.g. soil plant available water, and currentDay)
- Parameters (e.g. soil plant available water threshold to trigger irrigation)
- Model which returns a true/false output

Rule parameter values are tested against time and/or the state of the system. The state of the system is known via the dynamic variables made available by the component models of the modeled system.
It may be desirable to model certain types of management even if the relevant impact on production is not simulated (e.g. weeds, pests, diseases impact on production), so as to allow quantification of the use of inputs such as pesticides and possibly to allow modeling the fate of applied pesticides. Finally, given that almost all rule-and-impacts are specific for time in the rotation, all rules require as input "rotationYear", which in the example of Fig. 1 is 1 for crop1, and 2 for crop2. Table 1 shows a sample of the rules currently implemented.

Table. 1 Sample of the rules available in the component AgroManagement

Rule	Rule inputs	Rule parameters	Description
DateWindow	currentDay, rotationYear	beginDay, endDay, dayInterval	Triggers events within the dates whenever cumulated days = dayInterval (reset after event)
DateEventGDD	currentDay, rotationYear, phenologicalDate*, airTemperature Average	accumulatedGDD, baseTemperature	Triggers an event when a given number of accumulated GDDs computed using baseTemperature is reached since a phonological event occurred
TemperatureSum	rotationYear, airTemperature Average	consecutiveDays, temperatureThreshold	Triggers an event whenever a number of consecutive days has an average air temperature above a given threshold
IrrigationPAW	currentDay, rotationYear, soilLayers*	beginDay, endDay, plantAvailableWater Threshold, referenceDepth, maxNumberOfEvents	Triggers an event whenever the average value of plant available water over a soil depth is below the threshold, within the dates, for a maximum number of events
HarvestGrapes	currentDay, berriesSugarContent	beginDay, endDay, sugarContent	Triggers an event after beginDay, and when the sugar content of berries is above a sugar content threshold; if level is not reached before endDay triggers the event at endDay
ClippingGrasses	currentDay, aboveground Biomass, leafAreaIndex	beginDay, endDay, biomassThreshold, leafAreaIndex Trheshold	Triggers events within dates whenever either the biomass (or the leaf area index, different approach) threshold are reached

* complex type (e.g. soilLayers is an array of items soilLayer, which has attributes soilWaterContentVolumetric, SoilWaterContentAtFC, soilWaterContentAtPWP, layerThickness)

2.2 Impacts

Impacts stands for: "sets of parameters to implement the impact of a management event in a model component". Such sets are different changing management event, and can be different within management event if the modelling approach to implement the impact is based on alternate approaches. Impact can contain actual values of parameters, and/or they can contain an alphanumeric field/enumerator which can simplify the building of the agro-management configuration file. Such enumerators are then interpreted by model components which then can associate to the enumerator all values corresponding to the specific enumerator. As an example of the former, an impact for tillage may include two parameters:

- `tillageDepth`
- `soilMixingCoefficent`

An example of using an alphanumeric field / enumerators again for tillage can be:

- `tillageDepth`
- `implementType`

Where the implementType is an enumerator from a list such as:

```
PLOW_MOLDBOARD_0_2_M,
MULCH_TREADER,
.....
```

The items above are extracted from the list of implements of the model Wepp (Alberts et al., 1995). In the relevant database, a list of 8 parameters is associated to each of the items, and it allows a model component to implement the impact according to the Wepp approach. In particular, tillageDepth is one of the parameters encapsulated in the enumerator, but providing it separately makes possible to override the value, allowing for more flexibility. Using enumerators requires that model components access a dedicated data base to retrieve the parameter values associated to the enumerator value.

3. AGROMANAGEMENT COMPONENT

We have created a generic AgroManagement component that implements the rule-impact mechanism described in this paper. The functionality of rules is defined in the `IRules` interface and the functionality of impacts is defined in the `IManagement` interface. Thus, authors of rules and impacts are not limited to deriving their classes from specific base classes. The component has two main methods that 1) load the agro-management configuration, and 2) check rules at run time and fire events whenever a rule evaluates to "true". The modelling framework in which the AgroManagement component is registered provides at each time step time and states. An example of implementation is described in section 4.

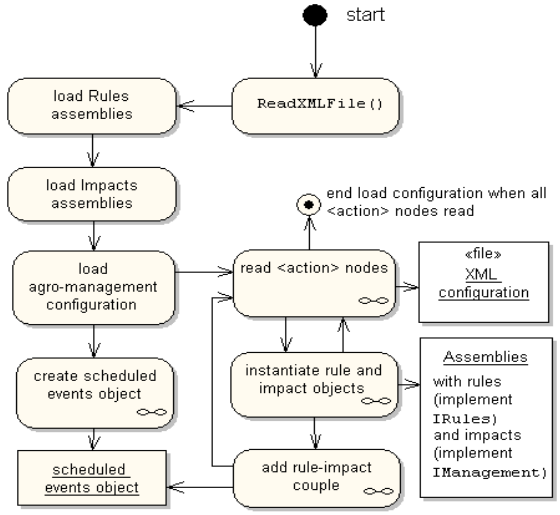

Fig. 2 Activity diagram of the initialization method (`ReadXMLFile`)

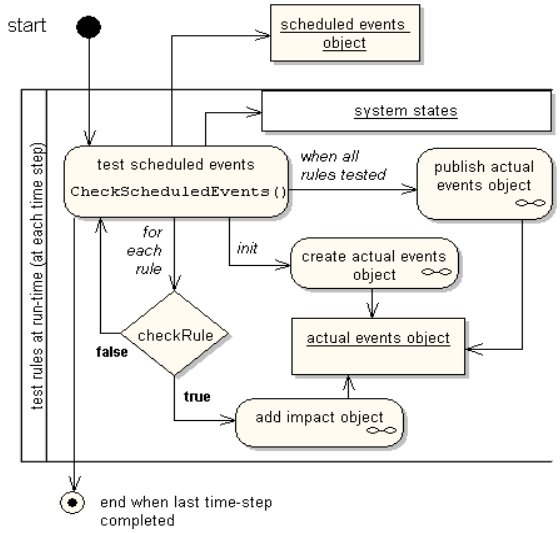

Fig. 3 Activity diagram of the run-time method (`CheckScheduledEvents`).

3.1 The schema and an example XML input

The schema (XSD) of the agro-management configuration input is shown in Fig. 4; an excerpt of an XML sample input file is shown below

```
<AgroManagement xmlns="http://pluto.esagr.org/APESwiki"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <action name="NITROGEN_FERTILIZATION">
  - <rule assembly="CRA.AgroManagement.rules" name="RuleDate">
      <parameter name="relativeDate" value="80" />
      <parameter name="rotationYear" value="1" />
    </rule>
  - <impact assembly="CRA.AgroManagement.impacts" name="NitrogenOne">
      <parameter name="totalNitrateAmount" value="50" />
      <parameter name="totalAmmoniaAmount" value="100" />
      <parameter name="surfaceBroadcast" value="False" />
    </impact>
  </action>
- <action name="IRRIGATION">
  - <rule assembly="CRA.AgroManagement.rules" name="RuleIrrigatePAW">
      <parameter name="relativeDateBegin" value="180" />
      <parameter name="relativeDateEnd" value="220" />
      <parameter name="rotationYear" value="1" />
      <parameter name="maxNumberIrrigations" value="3" />
      <parameter name="thresholdPAW" value="0.5" />
      <parameter name="referenceDepth" value="0.7" />
    </rule>
  - <impact assembly="CRA.AgroManagement.impacts" name="IrrigationSimple">
      <parameter name="irrigationVolume" value="45" />
      <parameter name="irrigationType" value="SPRINKLER" />
    </impact>
  </action>
</AgroManagement>
```

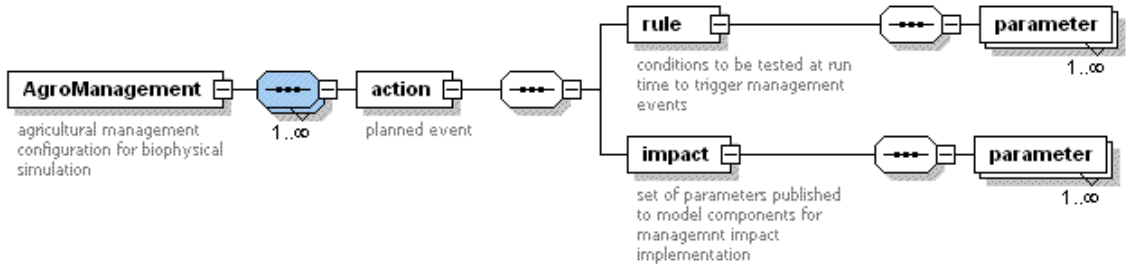

Fig. 4 Diagram representation of the XML schema of agro-management configuration for a simulation according to the rule-impact approach: for each planned event, a set of conditions is tested; if conditions are met, the associated set of management parameters is made available to model components.

3.2 The IRules interface

The `IRules` interface inherits from `IStrategy`, (i.e. an interface from the component CRA.core.preconditions.dll). The interface is the same implemented by model classes in a set of components (e.g. Donatelli et al., 2006), and it allows discovering rules, their inputs and parameters via reflection in applications like the Model Component Explorer - MCE (CRA-ISCI, 2005).
The interface consists of:

```
public interface IRules
{
bool CheckRule(AStates st, IManagement m);
void TestPreConditions(AStates st,
       IManagement m, string callID);
void LoadXml(XmlNode node);
void SaveXml(ref XmlTextWriter writer);
}
```

The second method allows for testing pre-conditions (Meyer, 1997) on the inputs and parameters of the rule; the test is done via the component CRA.core.preconditions.dll.
The preconditions component allows for different types of outputs and it allows for adding custom developed output drivers. The `LoadXML` method is used when the agro-management configuration is loaded as initialization at run-time.Finally, the `SaveXML` method allows writing the rule relevant information on an XML structured as in Fig. 4.
The abstract type `AStates` contains the attributes which get run-time values from the simulation system. It contains all the information needed by the rules currently implemented, but it can be extended as discussed in section 3.4

3.3 The IManagement interfaces

IManagement is the parent interface of a set of specialized to management type interfaces, as in the class diagram of Fig. 5.

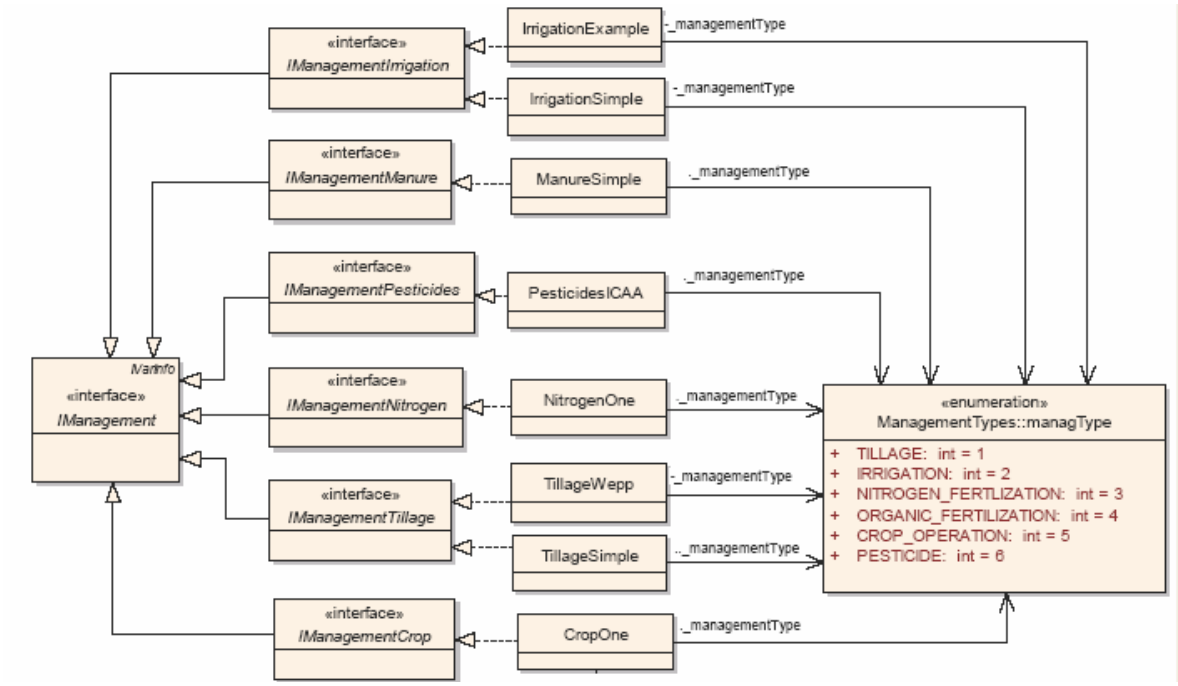

Fig. 5 Class diagram of the impact types and interfaces (not all classes shown). The diagram shows two alternate impact classes for irrigation and tillage.

3.3 Extensibility of rules and impacts

Rules can be extended with other classes implemented in a separate assembly. The new rules must implement the `IRules` interface. If an extension of the `AStates` type is needed, it can be done by inheriting from the class `States`, which is the default implementation of `AStates`.
New impacts can be defined within management type by implementing the relevant interface (e.g. `IManagementIrrigation`), or a new interface can be defined inheriting from `IManagement`. The proper recast will then be done into the components using the impact defined as shown in section 4.

3.4 Developments

The goal of on going work is to rationalize existing rules and to develop rules for different production activities. A specific interest is given to rules for pesticides spray, targeting at incorporating the empirical models frequently used by extension services to guide pesticides use.
The software component AgroManagement is available at http://www.isci.it/tools, inclusive of sample applications, HTML-style help and code documentation (NDoc).
.

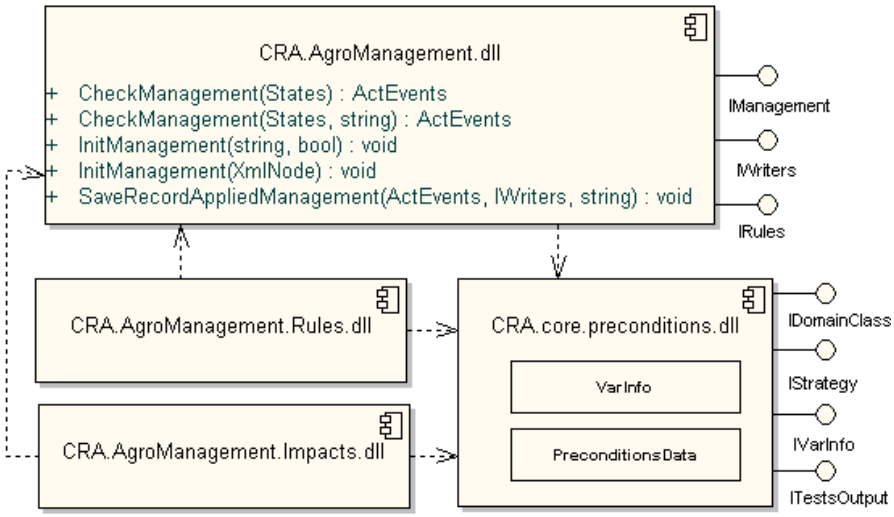

Fig. 6 Component diagram of CRA.AgroManagement. Rules and impacts can be extended by making available other assemblies with classes implementing the `IRules` and `IManagement` interfaces. Management specific interfaces (e.g. `IManagementTillage`) are implemented in the impacts assembly.

4. A ModCom APPLICATION

ModCom is a modelling framework that was first described by Hillyer et al. (2003). Recent developments, including a C# implementation, are available online (Anonymous, 2006).
A ModCom simulation models consists of a number of independent component models. These component models are recognized because they implement the `ISimObj` interface. A component model becomes part of a composite model when it is registered with a "simulation environment" object, which implements functionality defined in the `ISimEnv` interface. Component models communicate with each other via input- and output ports. Component models indicate to the `ISimEnv` instance with which they are registered what kinds of services they require. Objects that want to receive periodic notification of the progress of simulated time, implement the `IUpdateable` interface through which they communicate the interval at which they wish to receive notification. Objects that implement a state/rate model implement the `IOdeProvider` interface through which an external integrator can provide numerical integration services to these objects.
The AgroManagement component needs to examine its rules every simulated day; this indicates the need of a component that implements the `IUpdateable` interface. Thus, we created a wrapper class (Gamma, 1994) that implements `ISimObj` and `IUpdateable` and contains an instance of the AgroManagement class. This wrapper class is notified every simulated day by the framework and then calls on the wrapped instance of the AgroManagement class to evaluate the rules.
When a management rule fires, the wrapper class uses the ModCom event mechanism to notify target objects of the action to be taken. The target object itself implements the logic that realizes the action. Specifically, whenever evaluation of the rules indicates that an action must be initiated, the wrapper creates an object that implements the `ITimeEvent` interface, sets parameters as appropriate, and registers the event with the `ISimEnv` class. The event may be set to be handled at the current simulation time or at some future time.
When the simulation time becomes equal to the time at which the event is set to happen, the event is sent by the framework to the target object by invoking the target object's `HandleEvent()` method with the event object as a parameter. The target object can cast the event object to the type(s) of event(s) it can handle, obtain the

parameters it needs to implement the action, and perform the action. The AgroManagement component is configured by providing it with an XML fragment that corresponds to the schema in Fig. 4. The wrapper obtains this fragment by declaring an input port of type string and passing the value of this input to the AgroManagement component during initialization of the simulation. The ModCom utility `mrun.exe` is a console application that takes as its only input an XML file which contains a description of the simulation to be run as well as input data for each component model. This utility reads the configuration for the AgroManagement component from the xml file and assigns it to the appropriate input of the wrapper with the statement:

```
wrapper.Inputs["management"].Data.AsString="<AgroManagement>...</AgroManagement>";
```

CONCLUSIONS

The rule-impact approach allows specifying any biophysical driver of the decisional process to apply management, specifying any agro-technical input, and using any impact model to implement the impact of the action. The conceptual framework defined allows formalizing in a transparent and extensible way all the concepts relevant to agro-management, providing instances for a domain specific ontology. The approach allows formalizing and making use of expert knowledge in simulation tools. The independent implementation via the AgroManagement component allows de-coupling agro-management from biophysical models, thus adhering to the component-oriented design paradigm of simulation models. A major feature of the component AgroManagement is the ease to extend it; rules and impact models can be added without the need of recompiling the component.

5. ACKNOWLEDGEMENTS

This publication has been partially funded under the SEAMLESS integrated project, EU 6th Framework Programme for Research, Technological Development and Demonstration, Priority 1.1.6.3. Global Change and Ecosystems (European Commission, DG Research, contract no. 010036-2).

6. REFERENCES

Alberts, E.E., Nearing, M.A., Weltz, M.A. Risse, L.M., Pierson, F.B., Zhang, X.C., Laflen J.M., Simanton J.R., 1995. WEPP Model User guide. Chapter 7

Anomymous, 2006. ModCom [Online]. http://www.modcom.wur.nl (verified on February 6, 2006).

Argent, R.M. and A.E. Rizzoli, Development of multi-framework model components. In: Pahl-Wostl C., Schmidt S., Rizzoli A.E., Jakeman A.J. (Eds.), Trans. of the 2nd biennial meeting of the International Environmental Modelling and Software Society, Osnabrück, Germany, vol. 1, p. 365-370, 2004.

Brisson, N., C. Gary, E. Justes, R. Roche, B. Mary, D. Ripoche, D. Zimmer, J. Sierra, P. Bertuzzi, P. Burger. 2003. An overview of the crop model STICS. European Journal of Agronomy, Vol. 18, No. 3-4, pp 309-332.

CRA-ISCI, 2005 The Model Component Explorer [online] http://www.isci.it/tools , page XP Utils (verified on February 6, 2006)

David, O., S.L. Markstrom, K.W. Rojas, L.R. Ahuja and W. Schneider, The object modelling system. In: Ahuja L.R., Ma L., Howell T.A., (Eds.), Agricultural system models in field research and technology transfer. Lewis Publishers, Boca Raton, FL, USA, p. 317-344, 2002.

Donatelli M., L. Carlini, G. Bellocchi. 2006. A software component for estimating solar radiation Environmental Modelling and Software. Vol. 21, No. 3, pp 411-416

Gamma, E., R. Helm, R. Johnson, J. Vlissides. 1994. Design Patterns: elements of reusable object-oriented software. Addison-Wesley, Boston,MA.

Hillyer, C., J. Bolte, F. van Evert, and A. Lamaker. 2003. The ModCom modular simulation system. European Journal of Agronomy Vol. 18, No. 1, pp. 333-343.

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 2003. An overview of APSIM, a model designed for farming systems simulation. European Journal of Agronomy, Vol. 18, No. 3-4, pp 267-288

Jones, J.W., B.A. Keating and C.H. Porter, Approaches to modular model development. Agricultural Systems 70, 421-443, 2001.

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, J. T. Ritchie. 2003. The DSSAT cropping system model. European Journal of Agronomy, Vol. 18, No. 3-4, pp 235-265

Meyer, B. 1997 Object-oriented software construction, 2nd edition. Prentice Hall, Upper Saddle River, NJ, USA,.

Stöckle, C. O., M. Donatelli, R. Nelson. 2003. CropSyst, a cropping systems simulation model. European Journal of Agronomy, Vol. 18, No. 3-4, pp 289-307.