19th International Congress on Modelling and Simulation, Perth, Australia, 12–16 December 2011
http://mssanz.org.au/modsim2011

Dynamic resource allocation in a farm management simulation

R. Martin-Clouaire and **J.-P. Rellier**

INRA, UR875 - Biométrie et Intelligence Artificielle, BP52627, 31326 Castanet-Tolosan, France
rmc@toulouse.inra.fr; rellier@toulouse.inra.fr

Abstract: The managerial skills involved in planning and coordinating the activities required in a production process are critical in any business, including agriculture. Poor design may lead to poor performance, e.g., bad timing of activities and unbalanced utilisation of resources. Consequently, production processes need to be carefully analysed in order to evaluate their potential performance, identify vulnerability or flaws, and contribute to the design of improved systems. This paper presents a software platform that makes it possible to model and simulate agricultural production processes, including the management processes responsible for the dynamic scheduling of activities.

The successful completion of any agricultural production process requires solving successive resource-constrained scheduling problems, which are defined as the assignment of a limited set of resources to a collection of relevant activities, with the intent of satisfying the farmer's constraints and preferences. The most important resources include labour, equipments (e.g., tractors) and energy.

The dynamic nature of the scheduling process is made necessary by the changing environment in which the management problem takes place (e.g., weather, epidemic hazard, resource failures, etc.). Due to the unpredictability of changes, scheduling must be reactive in order to produce activities that are coherent with intentions and current circumstances. However, production management is not purely reactive. Farmers rely on activity plans that include a type of proactive anticipatory behaviour. Proper articulation between anticipation and reaction highly conditions production performance. The simulation framework presented here aims at enabling the study of production management by virtual experimentation.

In this framework, dynamic scheduling complies with organisational constraints specified in the form of a flexible plan that makes explicit the temporal and procedural dependencies between the activities devised toward the overall objective. The preservation of the flexibility of this anticipatory specification is obtained via the ability to incorporate alternative execution paths, optional activities and the underspecification of both the objects to be processed by the activities and the resources to be used in these activities. Moreover, the plan may be adjusted in response to particular events. The simulation of the application of such a plan requires a continuous interpretation process that aims at generating and executing the flow of activities that obeys the constraints and preferences coming with the plan. This involves the iteration of four steps:

1. Monitoring the intended activities that are eligible for execution according to the plan;
2. Allocating resources to the largest possible subsets of these activities;
3. Choosing the subset of allocated activities to execute; and
4. Executing the activities concurrently.

This paper only addresses Steps 2 and 3, and outlines the modelling framework developed to simulate the application of management strategies that are explicitly and formally represented. The first originality of this work lies in the characterisation of the different types of resources (e.g., discrete state or capacitated, atomic or aggregated) and the various kinds of constraints that restrict their use (e.g., state or capacity-related availability, simultaneous use of a resource in different activities and with other resources). The second one concerns the algorithm of dynamic resource allocation. It efficiently processes the various constraints applying to the resources and activities, and assigns resources to the set of all currently pertinent activities, if possible or to a subset of them otherwise.

Keywords: *Plan, resource allocation, simulation, farm management*

1. INTRODUCTION

The world we live in is changing at a very rapid pace. If agricultural production is to remain viable, it must adopt appropriate practices and adapt to new constraints and criteria concerning climate change, environmental quality, working conditions, energy prices and market demand. Agricultural production processes generally involve a number of coherently organised activities that are undertaken by a farmer to achieve his goals. Production management deals with how farmers combine land, water, machinery, structures, commercial inputs, labour and management skills to produce crop and livestock commodities, as well as other services. Good management principles are critical to maintain or increase profitability and to improve the sustainability of the production system. An inadequate design of the production processes may lead to poor performance, e.g., bad timing of activities, inefficient utilisation of resources or frequent failure to achieve goals. Consequently, agricultural production systems require enhanced analysis instruments that make it possible to represent their behaviour in different scenarios, and to identify possible flaws, vulnerability and performance before they are actually built. Such capabilities are extremely useful to understand and improve production systems and to adapt them to new conditions.

Simulation is widely used as a tool for analysing complex systems (Yilmaz and Ören 2009), but most studies focus on the abstract steady-state situation rather than realistic cases and the resolution of operational decision-making problems. In agriculture, simulation has essentially been used to study biophysical performance in response to climatic conditions and management rules that implement unconstrained management logic (McCown *et al.* 1996). Modelling and analysing resource management in production processes is either restricted to a single resource (e.g., irrigation water) or not addressed at all in spite of its high practical importance (Dillon 1980). Indeed, since resources are limited in supply, their efficient allocation to competing activities can have a major impact on production system performance.

This paper presents the DIESE modelling and simulation platform (Martin-Clouaire and Rellier 2009), developed to support the investigation of management processes in agricultural production systems. Section 2 introduces the main concepts used to represent the declarative knowledge about the farmer's production logic in the form of a flexible activity plan and their resource requirements (see Martin-Clouaire and Rellier (2011) for full details). Section 3 is devoted, first, to the representation of resources and the constraints involving their use and, second, to the algorithm responsible for allocating resources to the activities when their execution becomes relevant. Pointers to applications of DIESE to particular agricultural production systems are provided in the last section.

2. PLAN-BASED PRODUCTION LOGIC

In order to scientifically study a production process, it is necessary to build a sufficiently precise and rigorous model of the activities involved in this process and the constraints that constitute the farmer's production logic, in particular, the coordination dependencies between them. The commitment to understand how things work from a farmer's point of view is fundamental to our approach. To be effective, the production process must be described through constructs and language that are intelligible and conceptually close to those actually used in an agricultural setting. The basic unit of analysis in our approach is 'work activity', which is a common high-level concept in production and business process management (van der Aalst and van Hee 2002). This 'work activity' is an intentional commitment motivated by the production management requirements necessary to achieve an overall objective. It is contextual in the sense that it is time-related (i.e., refers to a date, a period, or notions of duration), and actual circumstances may condition its relevance and greatly affect the way the intended objective is achieved. Activities usually involve the use of resources (e.g., equipment, labour). Whenever a combination of activities must be undertaken with a view to achieving a preconceived result, a plan is needed to express how those composite activities should be coordinated. A work plan is the result of reflection on prior experiences and the anticipation of particular goals and probable occurrences of important events. Because of this, plans are not rigid in the sense of a definite and precise specification of the execution steps. Plans guide the flow of work to be done in a responsive manner and can be adapted to different circumstances.

In its simplest form, a 'work activity', which we will refer to here as a primitive activity, denotes something to be done to a particular biophysical object or location (e.g., a herd batch or building) by an executor (e.g., a worker, a robot, or a combination of both of them). Every activity has a status e.g., *sleeping*, *waiting*, *open*, *closed* and *cancelled*. A primitive activity is characterised by local opening and closing conditions, defined by time windows and/or predicates (Boolean functions) that refer to the biophysical states or indicators. An indicator is a piece of knowledge or information contextually invoked, assembled or structured to substantiate a decision-making step, e.g., estimate of the amount of remaining forage on a field to determine whether or not to withdraw the herd from it. The opening and closing conditions are used to determine which activities are eligible (according to the manager's intention) for execution at any given moment. They play a key role in defining the timing flexibility.

The "something-to-be-done" component of a primitive activity is an intentional transformation referred to as an operation (e.g., fertilisation). The progressive changes to the biophysical system as the operation is carried out constitute a functional attribute of the operation. These changes take place over a period of time by means of a process that increases the degree of achievement of the operation until it is completed. An operation is said to be instantaneous if its degree of achievement goes from 0 to 1 in a single atomic time unit (the smallest time interval defined by the modeller, e.g., typically, in hours or days). Otherwise, the operation is durative, which implies that its execution might be interrupted. An operation may require resources such as a sickle bar mower and a tractor in the case of cutting. In addition, the execution of an operation is constrained by feasibility conditions related to the biophysical state. Objects on which an operation is carried out can be individual objects, e.g., a field or a set of fields, or objects with numerical descriptors, e.g., an area. Speed is defined as a quantity, e.g., the number of items or the area that can be processed in a unit of time. The duration of the operation is the ratio of the total quantity to the speed. In order for the effect to be achieved, the operation must satisfy certain enabling conditions that refer to the current state of the biophysical system. For example, the field to be processed should not be too muddy, muddy being expressed as a function of some state variables. The ability to reap the benefits of organisational and timing flexibility depends on execution competence determined by the resources involved (both operation resources and the executor). Careful representation of the resources and their availability might therefore be essential to arrive at a proper understanding of the situation under study. Because their availability is discontinuous over time (e.g., the working hours of employees are limited within a day), the execution of any activity may be interrupted and resumed as soon as possible or not, depending on whether the decision-maker decides to enforce the earliest continuation or not of this activity.

Activities can be further constrained by using programming constructs that enable specification of temporal ordering, iteration, aggregation and optional execution. In doing this, we build non-primitive (i.e., aggregated) activities with names such as *before*, *iterate* and *optional*. Other constructs are used to specify the choice of one activity from among several using an '*or*'-based statement, grouping of activities with the '*and*'-based statement and concurrency of execution between some of them using constructs such as *co-start*, *equal*, *include* and *overlap*. Non-primitive activities also have status and can, therefore, also be given opening and closing conditions. In addition, they might involve temporal properties such as a delay between two activities in a *before* activity or a delay in an *iteration* activity between the closing of an iterated activity and the opening of the next one. The semantics of each class of non-primitive activity is defined by a set of procedural attributes that convey the principles governing its change in status. Every non-primitive activity has a relational property that makes a link to the set of the other activities directly involved in it or constrained by it. All the activities are connected; the only one that does not have a higher-level activity is the plan itself, which has the status *waiting* at the beginning of a simulation.

The model designer typically begins by capturing the nominal management behaviour, that is, the behaviour under idealistic circumstances, in other words, how the activities and resources should be coordinated when everything is going well. This can be modelled as a collection of primitive activities involving different actors and tools, with some that occur in parallel and others sequentially, some with earliest and latest times to be completed and some that risk conflicts and contentions that will have to be resolved at some point. Once the nominal behaviour has been established, the modeller should next think about what might go wrong and thus introduce flexibility into the plan by rendering optional the activities that are not absolutely necessary but that threaten the soundness of the plan, by enabling one or several branching points in the plan, or by enabling changes in the plan in response to events. Indeed, since resources are in short supply and they might be unavailable when they are needed, deadlines might not be met or the context may have evolved in an unexpected manner, and an external event might even make the plan obsolete. Therefore, in each of these cases, other management possibilities may be required beyond the nominal path.

The use of flexibility provides an essential means to cope with uncertainty. In addition to timing flexibility and state-dependent opening and closing of its activities, a plan is made flexible by the use of patterns that enable optional execution, choice between candidate activities and adjustable iteration of activities. Implementing such a plan involves context-dependent decisions to determine: (1) whether an optional activity should be executed; (2) whether an iteration of activity should continue; and (3) which alternative activity should be chosen. Finally another source of flexibility supported by our management plan representation is obtained through the abstract specification of the entities to be processed by an operation. The entities are defined in intension, that is, by rules that enable on-the-fly generation of the set of them that satisfy the rules, e.g., the 'set of fields' to be ploughed that are specified by a time frame (number of years since they were last ploughed). Any operation to be performed on a set of entities can be done in three ways, depending on the resources allotted. If, for instance, the execution of the operation on any entity requires one unit of equipment and one unit of labour, the work may be treated as a single activity that consists in successively applying the operation to all the entities of the set using the same resources or, alternatively, as a set of similar activities applying each to a single entity with possibly different resources. In the latter case the activities may be sequentially executed using the required resources that are available at execution time, or, alternatively, a concurrent execution may take place in which as many activities are scheduled as are made possible by the availability of resources (i.e., in this case, pairs of an equipment unit and a labour unit). In such a case, this fragment of the plan is specified at design time using a placeholder activity to be expanded at runtime into a sequence of primitive activities (using a *meet* type of activity) or a conjunction of these activities (using an *and* type of activity), depending on the manager's intention. In addition to the flexibility provided by the late binding, the ability to group activities to be performed on a set of entities is a quite useful and common organisational practice in production management.

Notwithstanding the flexibility of activities and the flexibility of resource assignment (see next section), it may be necessary to adapt the plan when particular circumstances occur. Indeed, a nominal plan conveys the rough course of intended steps to go through under normal circumstances. The specification of when and what changes should be made to a nominal plan is referred to as a conditional adjustment. The trigger for a conditional adjustment is either a calendar condition that becomes true when a specific date is reached, or a state-related condition that becomes true when the current circumstances match this condition. The adjustment can be any change to the nominal plan such as the deletion or insertion of activities. It can also affect the resources used in some activities. A conditional adjustment can also actually specify a change to be made to conditional adjustments themselves. By this means, the management can be reactive and thus cope with unexpected fluctuations of the external environment (e.g., drought) and various contingencies.

The change in status of each activity (primitive or non-primitive) of the plan is governed by a specific state transition that defines its semantics for each type of activity (Martin-Clouaire and Rellier 2011). The advance of time and the evolution of the production system (the biophysical system, in particular) may render the opening and closing conditions of activities true. The status of the activities is updated by a process that reacts to events scheduled to occur at any examination time specified by the manager (typically, at discontinuity points induced by a new day or a new week) or upon the termination of an operation. For each activity the updating process essentially checks that the opening or closing conditions can be satisfied and that the constraints linking this activity to others would be satisfied if the change proceeded. When applied to the plan (initially set to *waiting*), this method causes a recursive examination of all the activities that are *waiting* or *open*. Any activity whose change in status is validated is updated, and the change is immediately propagated to the connected activities. The particular activities that are placeholders are expanded in function of the expansion of the set of entities to be operated.

3. RESOURCES AND ALLOCATION

3.1. Resource types and usage constraints

The ability to reap the benefits of organisational and timing flexibility depends on execution competence determined by the resources involved (both operation resources and the executor). Careful representation of the resources and their availability might therefore be essential to obtain a proper understanding of the situation under study and the possibilities of improvement. Basically, a resource is an entity that supports or enables the execution of activities. Typically, the activity executors, the machinery involved and the various inputs (seeds, fertiliser, water, fuel) are resources. Resources are generally in finite supply and have a significant influence on when and how activities may be executed. The availability of a resource is restricted by availability constraints that specify the conditions under which their use or consumption is allowed. The constraints are temporal (statically or dynamically established time windows of availability), capacity-related (the amount available) or state-related. The temporal constraints related to the availability of executors might be flexible: the length of a working day can vary slightly and be opportunistically changed from one day to the other if needed, but the cumulated working time over a year must comply with strict conditions. Any resource can be possibly constrained with respect to the maximum number of operations simultaneously supported and the maximum number of resources of other types that can be simultaneously used.

There are many types of resources that must be dealt with (Smith and Becker 1997). A resource can be either consumable (usable only once) or reusable after it has been released. It can be a discrete-state resource such as equipment (whose availability is a function of possible values taken by a discrete variable) or a capacity resource such as diesel fuel (whose availability is characterised by a vector of numerical values that express a multi-dimensional capacity). We distinguish between single resources and aggregate resources, which are collections of resources.

In a primitive activity, the resource role is played by the operated object, the operation resources and the executor. An operated object is a discrete-state resource that is a part of the biophysical system (an entity or a set of entities of the biophysical system). It is characterised by its ability to be transformed by several operations simultaneously. It may allow several resources to be simultaneously involved in transformations, and several executors to carry out certain transformations simultaneously.

An operation resource is also characterised by its ability to be used simultaneously for several objects acted upon in the biophysical system, to be involved simultaneously in several operations, and to be used simultaneously by several executors.

An executor is a discrete-state resource that is either an individual resource (e.g., a worker) or a labour team (a set of individual workers). Another feature of an executor is it's (his) work power that has an effect on the speed of the operation and on the requirement of operation resources if the latter are declared proportional to the power. The power of a team is, by default, the sum of the powers of the individual workers it comprises.

As an illustration, consider a grass cutting activity with the resource specifications shown in Table 1. The operated object specification refers to a set of spatial entities that are dynamically generated by expanding the entity set specification defining this set. Considering it as a resource is useful in the event that it is decided to disallow two simultaneous operations on any of these entities. The specification of resources associated with the operation component states that two machines are required: a mower and a tractor. The executor is a person to be selected, either from among the farmer's sons or his employees. If instances are available in each of these classes, two alternative allocations have to be considered. In this example, at the time of allocation, the procedure would provide two alternatives {(f1, m1, t2, s2), (f1, m1, t2, e)} if f1 is the only field that satisfies the request, m1 and t2 are the mower and tractor that are available, and s2 and e are the second son and the employee, respectively, that are free at that time. It might provide a set of only one collection of assignments if either one son or one employee is available. It might obviously provide no solution at all in some cases. All instances of resource of a class mentioned in the specifications (e.g., MOWER, SON) are considered equivalent unless otherwise specified, which would require considering the alternative allocations formed by each instance. In the resource specifications shown in Table 1, only one resource is required in each of the last two cases and a set in the case of the operated objects. The representation language supports the expression of richer specifications involving the requirement of a fixed number of resources of the same type or the use of the maximum number of them or even all of them.

Table **1.** Resource requirements in a grass cutting activity
(*): Upper case letters refer to class names and lower case letters refer to existing instances of the class concerned.

What is specified	*Specification*	*Instances of entities or resources*
Operated objects	"all non-grazing fields greater than 0.5 ha"	FIELD: {f1, f2, f3}
Operation resources	"one mower and one tractor"	MOWER: {m1, m2} TRACTOR: {t2}
Executors	"one person from among farmer's sons or his employees"	SON: {s1, s2, s3} EMPLOYEE: {e}

The use of resources is restricted by various constraints that make resource allocation a tricky combinatorial task. In addition to availability constraints, the ontology makes it possible to specify co-usage restrictions that concern the simultaneous use of a resource in different operations and combined with other resources. These co-usage restrictions consist of a set (conjunction) of inconsistency conditions defined as cardinality limitations (i.e., number of elements allowed). The restriction referred to as *activity-inconsistency-condition* applies to an activity, whereas the one referred to as *resource-sharing-violation-condition* applies to a resource. Finally, a third type of usage restriction referred to as *activities-resources-inconsistent-commitment* is available and comprises a set of *activity-inconsistency-condition* and a set of *resource-sharing-violation-condition*. See Table 2 for an example of each. The *activity-inconsistency-condition* entity specifies that it is forbidden for any of the farmer's sons to use a tractor to cut a field. Checking this constraint amounts to making cardinality verifications relative to the number of sons and the number of tractors involved in the activity (their number should not be greater than 0 for both of them simultaneously). The *resource-sharing-violation-condition* restriction specifies that there cannot be more than one executor at any location. Again, checking the constraint is a matter of cardinality verification, but this time, all allocations made so far for the current activity list are considered. The *activities-resources-inconsistent-commitment* constraint means that there is an incompatibility between a grazing activity by any dairy herd and the concomitant use of any pesticide at any location.

Table **2.** Examples of restrictions on resource usage

Type of constraint entity that is specified	***Example***
a*ctivity-inconsistency-condition*	{CUTTING; ((SON > 0)(TRACTOR > 0))}
resource-sharing-violation-condition	{LOCATION; ((EXECUTOR > 1))}
activities-resources-inconsistent-commitment	{ {GRAZING; ((DAIRY-HERD > 0)) } { LOCATION; ((PESTICIDE > 0)) }}

3.2. Resource allocation algorithm

This subsection outlines the main ideas implemented in the process that is responsible for the allocation of resources. Assume that we have a set of three primitive activities {*a b c*} at the current time that are open and have operations that satisfy their feasibility conditions. The activities are ordered (the set is actually a list) to put in the front the activities that have been declared to be preferably resumed in the event that their execution is interrupted. An activity that involves the specification of a maximum number of units of a specific type of resource should be put at the end of the list to force its allocation to be done last. The ordering may also be done in relation to allocation priorities defined by the modeller for each activity.

The algorithm operates on a lattice of subsets of primitive activities (see Figure 1) where a link between two nodes represents the relation "is superset of". The nodes are gone through in breadth-first order, starting from the top. In a given node, the algorithm attempts to allocate each activity in turn. It visits each resource requirement successively, obtains an instance that satisfies the requirement, and then checks the three types of co-usage constraints. It moves to another instance in the event of failure.

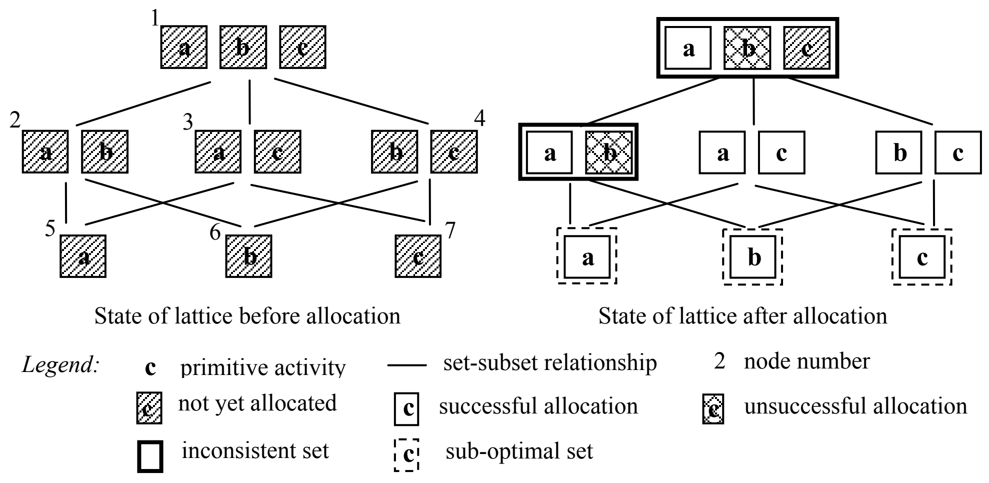

Figure 1. Lattice of activities

The result of a tentative allocation of resources to an activity (whether it is successful or not) is propagated in the lattice so that the search space can be significantly reduced. More specifically, the algorithm duplicates and transfers this allocation to descending nodes that have the same starting activities up to the current one (e.g., node 1 and node 2 have the same starting activities up to *b*). In the example of Figure 1, the resources allocated to activity *a* in node 1 can be propagated to the same activity in nodes 2, 5 and 3. If a failure is encountered in an attempt to allocate an activity in a node, the node is declared inconsistent and all descending nodes with the same starting activities (up to the current one) are also declared inconsistent. For instance, since activity *b* in node 1 cannot be allocated once activity *a* has been allocated, node 1 is declared inconsistent (no need to visit c), which causes node 2 to be declared inconsistent as well. If a node becomes fully allocated, the descending allocated nodes are marked "sub-optimal" in the sense that they are included in larger set of executable activities. Once node 3 is fully allocated in Figure 1, nodes 5 and 7 are marked sub-optimal. Note that, due to the co-usage restrictions, the set of resources allocated to an activity (e.g., *c*) in a node (e.g., node 3) at a given level may be different from the set of resources allocated to the same activity in another node (e.g., node 4) at the same level.

At the end, the algorithm returns all the nodes that are fully allocated and not sub-optimal, that is, nodes 3 and 4 in the example. The algorithm is complete in the sense that all solutions are produced. The choice of the one to be executed results from the computation of a scoring method that combines various preferences of the decision-maker in relation to, for example, the average priority of the operations in the primitive activities involved, the urgency of the activities expressed with respect to their latest opening or closing dates, the expected duration of the execution or the cost of resource consumption.

4. CONCLUDING REMARKS

We have outlined the software platform, DIESE, which makes it possible to model and simulate agricultural production processes, including the management processes responsible for the dynamic scheduling of activities. Its originality concerns, first, the knowledge representation capabilities that rely on an ontology of the production management domain (activities, resources and dependencies between them) and, second, the resource allocation algorithm that implements a dedicated constraint satisfaction approach. The framework is well-adapted to the study of centralised decision-making with a focus on the timing of activities in a highly changing environment such as the one characteristic of agricultural production systems that have to cope with uncontrollable driving factors such as weather. Our simulation framework enhances traditional farm analysis instruments by providing a holistic and process-oriented view of farm operations. The simulation of production processes helps to understand, analyse and design such processes. With the use of simulation, the (re)designed processes, which can hardly be reduced to ad hoc changes of the system design parameter values, can be evaluated and compared. Simulation provides quantitative estimates of the impact that a process design is likely to have on process performance, and a quantitatively supported choice for the best design can be made. DIESE was first used in the study of livestock systems (Chardon *et al*. 2007; Martin *et al*. 2011) and intercropping in vineyards (Ripoche *et al*. 2011). Two applications that heavily rely on the resource management capabilities presented in this paper have been developed in cropping (Rellier *et al.* 2011) and vine growing (Paré 2011) systems.

REFERENCES

Chardon X., C. Rigolot, C. Baratte, A. Le Gall, S. Espagnol, R. Martin-Clouaire, J.-P. Rellier, C. Raison, J.-C. Poupa and P. Faverdin (2007) MELODIE: a whole-farm model to study the dynamics of nutrients in integrated dairy and pig farms. Proc. of MODSIM07, Christchurch, NZ, 1638-1645.

Dillon, J.L. (1980) The definition of farm management. *J. of Agricultural Economics*, 31(2), 257-258.

Martin, G., R. Martin-Clouaire, J.-P. Rellier, and M. Duru, (2011) A simulation framework for the design of grassland-based beef-cattle farms. *Environmental Modeling & Software*, 26(4), 371-385.

Martin-Clouaire, R., and J.-P.Rellier (2009) Modelling and simulating work practices in agriculture. *Int. J. of Metadata, Semantics and Ontologies*, 4(1-2), 42-53.

Martin-Clouaire, R., and J-P. Rellier (2011) Fondements ontologiques des systèmes pilotés. Internal report UBIA-INRA (First version: 2001). http://carlit.toulouse.inra.fr/diese/docs/ri_ontologie.pdf

McCown, R., G. Hammer, J. Hargreaves, D. Holzworth, and D. Freebairn (1996) APSIM: a novel software system for model development, model testing and simulation in agricultural systems research. *Agricultural Systems*, 50(3), 255-271.

Paré, N. (2011) Pollution de l'eau par les pesticides en milieu viticole languedocien. Construction d'un modèle couplé pression-impact pour l'expérimentation virtuelle de pratiques culturales à l'échelle de petits bassins versants. Ph. D. thesis, Supagro, Montpellier, F.

Ripoche, A., J.-P. Rellier, R. Martin-Clouaire, N. Paré, A. Biarnès, and C. Gary (2011) Modelling adaptive management of intercropping in vineyards to satisfy agronomic and environmental performances under Mediterranean climate, *Environmental Modelling & Software* (to appear).

Rellier, J.-P., R. Martin-Clouaire, N. Cialdella, M.-H. Jeuffroy, and J.-M. Meynard (2011) Modélisation de l'organisation du travail en systèmes de grande culture : méthode et application à l'évaluation ex ante d'innovations variétales de pois. In Béguin, P., B. Dedieu, and E. Sabourin (eds.) « Le travail en agriculture : son organisation et ses valeurs face à l'innovation », L'Harmattan, Paris, 205-22.

Smith, S., and M. Becker (1997) An ontology for constructing scheduling systems. In Proceedings AAAI-97 Spring Symposium on Ontological Engineering, Palo Alto, CA, 120–129.

van der Aalst, W., and K. van Hee (2002) *Workflow Management: Models, Methods and Systems*. MIT press, Cambridge, MA.

Yilmaz, L., and T.I. Ören (2009) *Agent-directed simulation and systems engineering*. Wiley, Berlin.