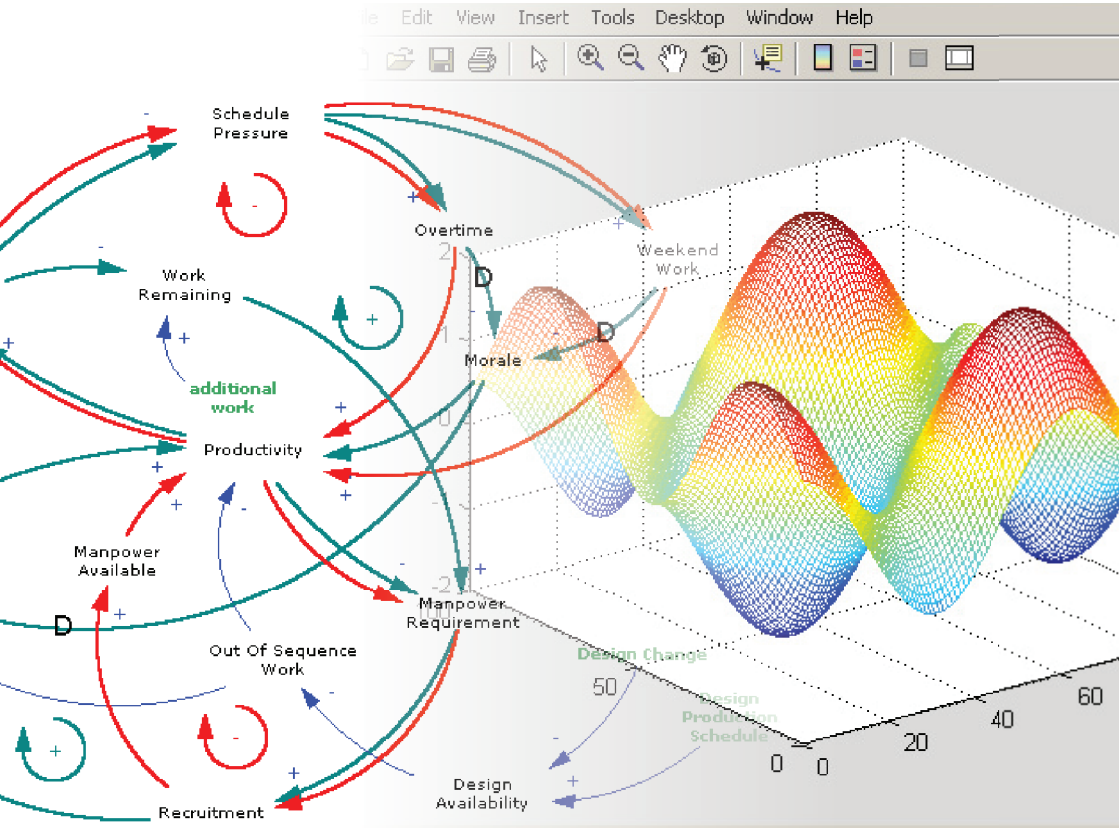
Boletim de Pesquisa e Desenvolvimento 36
ISSN 1677-9266
December, 2014

A comparison between three different approaches to omplement a system dynamic model: an assessment by a multidisciplinary team



Embrapa

ISSN 1677-9266
December, 2014

Brazilian Corporation of Agricultural Research
Embrapa Agriculture Informatics
Ministry of Agriculture, Livestock and Food Supply

Boletim de Pesquisa e Desenvolvimento 36

A comparison between three different approaches to implement a system dynamic model: an assessment by a multidisciplinary team

Mateus Castelani Freua
Luís Gustavo Barioni
Raphael Gustavo d'Almeida Vilamiu
Fernando Rodrigues Teixeira Dias

Embrapa Agriculture Informatics
Campinas, SP
2014

Embrapa Agriculture Informatics
Avenida André Tosello, 209 - Barão Geraldo
C. Postal 6041 - 13083-886 - Campinas, SP
Telefone: (19) 3211-5700
www.embrapa.br/informatica-agropecuaria
sac: www.embrapa.br/fale-conosco/sac/

Publication Committee

President: *Silvia Maria Fonseca Silveira Massruhá*

Secretary: *Carla Cristiane Osawa*

Members: *Adhemar Zerlotini Neto, Stanley Robson de Medeiros Oliveira, Thiago Teixeira Santos, Maria Goretti Gurgel Praxedes, Adriana Farah Gonzalez, Neide Makiko Furukawa, Carla Cristiane Osawa*

Substitute members: *Felipe Rodrigues da Silva, José Ruy Porto de Carvalho, Eduardo Delgado Assad, Fábio César da Silva*

Editorial supervisor: *Stanley Robson de Medeiros Oliveira, Neide Makiko Furukawa*

Text reviewer: *Adriana Farah Gonzalez*

Bibliographic standardization: *Maria Goretti Gurgel Praxedes*

Electronic editing/cover page: *Neide Makiko Furukawa*

Photos: *www.ventanasystems.co.uk ; httpwww.aquaphoenix.comlecturematlab8page3.html*

1st edition
on-line 2014

International Data of Publication Cataloging (CIP)

A comparison between three different approaches to implement a system dynamic model : an assessment by a multidisciplinary team / Freua, Mateus Castelani... [et al.]. - Campinas : Embrapa Agriculture Informatics, 2014.

19 p. il.: 14.8 cm x 21.0 cm. - (Boletim de pesquisa e desenvolvimento / Embrapa Agriculture Informatics, ISSN 1677-9266; 36).

1. Dynamic model. 2. Mathematic model. I. Freua, Mateus Castelani. II. Barioni, Luís Gustavo. III. Vilamiu, Raphael Gustavo d'Almeida. IV. Dias, Fernando Rodrigues Teixieira. V. Embrapa Agriculture Informatics. VI. Title. VII. Serie.

CDD 21 ed 003.3

Contents

A comparison between three different approaches to implement a system dynamic model: an assessment by a multidisciplinary team

Mateus Castelani Freua[1]
Luís Gustavo Barioni[2]
Raphael Gustavo d'Almeida Vilamiu[3]
Fernando Rodrigues Teixeira Dias[4]

Abstract

In the last few decades, application of system dynamics models (SDM) has disseminated through the agricultural sciences. Modeling groups are now much more multidisciplinary once the models currently developed are applied to larger frameworks. The literature has been inconclusive with regard to empirical evidence of the trade-offs between different paradigms to implement SDMs. In order to gain insight on the advantages and disadvantages between the paradigms we simulated a working group environment with seven researchers coming from various educational backgrounds where they had to implement a process-based SDM

[1] *BS in Animal Science, MS candidate, College of Animal Science and Food Engineering, University of São Paulo, Pirassununga, SP, Brazil*

[2] *BSc in Agricultural Science, Ph.D. in Animal Science and Pasture Production, researcher at the Computational Mathematics Laboratory, Embrapa Agriculture Informatics, Campinas, SP*

[3] *BSc in Physics, PhD in Applied Mathematics, external collaborator, professor at Cefet-RJ on campus Angra dos Reis, RJ*

[4] *Electronics engineer, Master of Business Administration, researcher at Embrapa Pantanal, Corumbá, MS, Brazil*

and fill in a questionnaire scoring characteristics of the paradigms and implementation process. The participants were divided into three groups according to their expertise: NAA, formed by procedural programming experts that performed the exercise in MATLAB®; OOA, formed by objected-oriented programming experts that performed the exercise in a C++ simulation framework; and GDA, the domain expert that performed the exercise in Vensim®. The approaches were ranked NAA > OOA > GDA for mathematical expressiveness, GDA > OOA > NAA for visual expressiveness, OOA > NAA = GDA for scalability and code reuse, and NAA = OOA > GDA for software integration. Based on the questionnaire and group discussions, a descriptive framework of what should be considered to identify the most appropriate implementation strategy for a SDM was developed. We suggest that the choice of what strategy to use should be driven by a combination of variables related to the model characteristics, group member's expertise and the properties intrinsic to each programming paradigm. Further research is needed to extend the analysis of how the decision on the paradigms should be related to the SDM characteristics, the time available, and the modeling group members' expertise.

Keywords: Agricultural models, dynamic models, objected-oriented programming, MATLAB®, Vensim®

Uma comparação entre três diferentes abordagens para a implementação de um modelo de sistema dinâmico: um estudo por uma equipe multidisciplinar

Resumo

Nas últimas décadas, a aplicação de modelos de sistema dinâmico (MSD) disseminou-se através das ciências agrárias. Grupos de pesquisa em modelagem matemática são, agora, muito mais multidisciplinares, uma vez que os modelos atualmente desenvolvidos aplicam-se a estruturas mais abrangentes. A literatura não está definida no que diz respeito à evidência empírica dos trade-offs *entre os diferentes paradigmas para implementar MSDs. Com o objetivo de identificar vantagens e desvantagens entre os diferentes paradigmas, grupos de trabalho foram simulados com sete pesquisadores oriundos de diversas áreas do conhecimento. Cada grupo implementou um MSD de processos biológicos e cada pesquisador preencheu um questionário pontuando características dos paradigmas e do processo de implementação. Os participantes foram divididos em três grupos de acordo com seus conhecimentos: NAA, formado por especialistas em programação procedural, que realizaram o exercício em MATLAB®; OOA, formado por especialistas em programação orientada a objetos, que realizaram o exercício em um* framework *de simulação em C++; e GDA, o especialista de domínio, que realizou o exercício em Vensim®. As abordagens foram classificadas NAA > OOA > GDA para expressividade matemática, GDA > OOA > NAA para expressividade visual, OOA > NAA = GDA para escalabilidade e reutilização de código e NAA = OOA > GDA para integração de software. Com base nos*

questionários e grupos de discussão, foi desenvolvido um quadro descritivo do que poderia ser considerado para identificar a estratégia de implementação mais bem apropriada a um MSD. Sugere-se que a escolha de qual paradigma adotar deve estar fundamentada por uma combinação de variáveis relacionadas com as características do modelo, a área de conhecimento dos membros do grupo e as propriedades intrínsecas de cada paradigma de programação. Mais pesquisas são necessárias para aprofundar a análise de como a decisão sobre qual paradigma adotar relaciona-se com as características do MSD, o tempo disponível para conclusão do projeto e o conhecimento dos membros do grupo.

Palavras-chave: Modelos agrícolas, modelos dinâmicos, programação orientada a objetos, MATLAB®, Vensim®

1 Introduction

System dynamics models (SDM) have become fundamental in agricultural sciences. Many efforts have been done to develop SDMs that are suitable for both decision support systems (DSS) and as a sound scientific base for inference of contemporaneous issues such as mitigation and adaptation practices in a changing climate. SDMs are usually mechanistic and describe at a certain degree hierarchies of different entities and processes in a system (HILLYER et al., 2003). Research projects may include modeling groups with diverse educational background, and they are often composed by domain experts, quantitative methods experts (some with stronger mathematics and statistics background) and computer scientists. Faced with the programming task, researchers usually have three main paradigms to implement a model: a) numerical analysis software using procedural programming (NAA); b) objected-oriented simulation (OOA); c) graphically oriented model implementation (GDA). The implementation paradigm influences the way programmers analyze the problem and design the computing solution. Thus, the way models will be designed, implemented, calibrated, communicated and transferred may be largely driven by the modeling tools at hand.

Agricultural and environmental modeling have been done more by domain experts rather than programmers or mathematicians. Thus, relaxing the straight relation between computer modeling and programming would bring benefits to modelers, particularly when programming is not part of their background. The broad application of SDMs in several disciplines, whose practitioners are not usually required to have knowledge on programming and software engineering, has driven the development of some user-friendly tools for non-programmers (e.g. Vensim®, Stella®, ExtendSim®, Simile®). Graphically driven modeling tools have also been part of traditionally procedural programming tools as in the case of MATLAB®, Simulink® or SAS® Simulation Studio.

The wide range of SMDs with agricultural applications has made the processes of adaptation, coupling and substitution of previous existing models a rule rather than an exception. Thus, systems models are developed iteratively and are increasing in size in hope of handling more complex problems. In this regard, the efficiency of designing, writing,

correcting, maintaining and scaling the models has become paramount for successful implementation, similarly to what was reported in the software crisis (RAMAMOORTHY et al., 1984). It is widely believed that objected oriented programming increases software maintainability, improves software quality, and simplifies program design and understandability over more structural methods. Object oriented simulation is often promoted based on the same principles (JOINES; ROBERTS, 1998). However, the literature has been inconclusive regarding studies with empirical evidence of advantages of object oriented programming over other approaches (EIERMAN; DISHAW, 2007; LIM et al., 2005). Therefore, when it comes to implement biological models by multidisciplinary teams, the trade-offs between the paradigms is still a subject for research, once they are related to gains in efficiency and quality of the computing solutions.

In this paper we report an experiment where both SDM design and implementation paradigms were evaluated by multidisciplinary team in order to gain insight on the advantages and disadvantages of procedural, objected-oriented and graphically-oriented programming. We conclude by suggesting a decision diagram to help choosing the most appropriate model implementation strategy considering model properties, paradigm characteristics and the group members' expertise.

2 Research methodology

2.1 Participants and procedure

Seven researchers with various educational backgrounds were recruited from the Computational Mathematics Laboratory of the Embrapa Agriculture Informatics. Three of them were familiar with objected-oriented programming and design, other three were experts in procedural programming (one of them was also expert in MATLAB®); and the other was a biologist expert on the model's domain. The participants were divided into three programming groups according to their expertise: a) NAA was formed by MATLAB® and procedural programming experts and performed the exercise in

MATLAB®; b) OOA was formed by objected oriented programming experts and performed the exercise in the C++ simulation framework described by Mancini et al. (2013); c) GDA was the domain expert using Vensim®. Before the implementation task, each group had time to reasoning out the domain problem space and the design solution. It is important to stress that although it is possible to perform objected-oriented programming in MATLAB®, we chose to use procedural programming because it is the most common paradigm used within this class of software and to make a clear distinction from the OOA approach using C++.

The three groups were assigned the same dynamic model as described by Dijkstra et al. (1992). This model was chosen because it had already been included in a broader biophysical modeling project with the participation of the computational laboratory. The model is comprised of seventeen state variables (ordinary differential equations) representing different substrates and biochemical processes of rumen fermentation.

Each group only received the model's documentation with a detailed description of the equations. After the groups had completed their respective tasks, the three implementations were shared for a collective appreciation. Participants were asked to fill in a questionnaire to score according to their evaluation the characteristics of all three paradigms and model implementation process. In order to ensure that each model was equally implemented for consistent comparisons of the three different approaches, model specification was carried out by the same participant (the domain expert) for all groups. Since our goal was to assess the implementation process, model's results and computational performance were not compared.

2.2 Analysis

The following variables were scored in order to capture the programmer's perceptions for each implementation strategy:

• **characteristics of the paradigms**: "Design complexity" refers to the effort required to produce a well designed implementation; "Design support" refers to diagram or other artifacts that are readily available for a given paradigm;

"Implementation effort" is related to time per person required to implement the model; "Intuitiveness" is time and support required to translate model specification (from the domain expert) into an implementation strategy; "Multidisciplinary communication" refers to how easily the model concepts in the implementation can be understood by non-experts; "Mathematical expressiveness" refers to how close the implementation is to the symbolic math formulation; "Visual expressiveness" refers to the capacity of code or diagrams to visually convey the model itself.

• **characteristics of the implementation process and results**: "Code reuse" refers to the likelihood that the code can be used again with slight or no modification for the implementation of a new model; "Code length/ visualization" refers to the number of lines of code (including equation editing in GDA) and how easy is to visualize the code (i.e. without the need to navigate through multiple windows and files); "Ease of transcription" refers to the effort needed to implement the model from the specification in a paper; "Software integration" refers to the ease to communicate with other software or to incorporate the model in a simulation application; "Code maintenance" refers to the ease of modification of the code to correct faults or to improve the model; "Scalability" refers to the ability to extend the model to cope with new requirements; "Mathematical analysis support" refers to the capacity of the tool used for that paradigm to support sophisticated numerical analysis.

3 Results

3.1 Example of model component implementation

Figures 1, 2, and 3 present code fragments of the PdPs (degradable protein to soluble protein) transaction written by the NAA, OOA and GDA groups, respectively.


```
% calculating the Michaelis-Menten constant
M(Pd,Ps) = Mref(Pd,Ps)*T(Pd)/Tref(Pd);

% calculating velocity for PdPs transaction
v(Pd,Ps) = vmax(Pd,Ps)*(Q(Ma) + Q(Mc))

% calculating the PdPs transaction rate
U(Pd,Ps) = v(Pd,Ps)/(1 +M(Pd,Ps)/C(Pd));
```

Figure 1. Code fragment of the PdPs transaction written in MATLAB®.

```
/*Function calculating main uptake (it is passed to the constructor of the
superclass SRT (simple rumen transaction) in order to allow calculating the
uptake of the main substrate. MM1S(VMax, M, CS) is the standard Michaelis-
Menten function VMax: maximum transaction rate, M: Michaelis-Menten
parameter, CS: substrate concentration*/
double Cls_PdPs::Aux_UMain() {
   return MM1S(par_Vref->Value()*(inp_Ma->Value()+inp_Mc->Value()),par_M-
>Value(), inp_CPd->Value());
}

// Parameter M is calculated before starting the simulation
void Cls_PdPs::Initialize() {
   par_M->Value(par_Mr->Value()*inp_TPd->Value()/par_TrPd->Value());
}

// Constructor
Cls_PdPs::Cls_PdPs(string name_arg, Cls_Model * aOwner_arg) :
Cls_SRT_Transaction(name_arg, aOwner_arg, (functionType_ptr) & Cls_
PdPs::Aux_UMain) {
   inp_CPd->Name("PdPs_CPd");
   par_M = AddVariable("M", -INFINITY);
   par_Mr = AddVariable("Mr", 0.264);
   par_TrPd = AddVariable("TrPd", 0.66);
   Par_Vref(0.0576);
   Par_YMain(1.0);
}
```

Figure 2. C++ code fragment of the PdPs transaction written for the simulation framework.

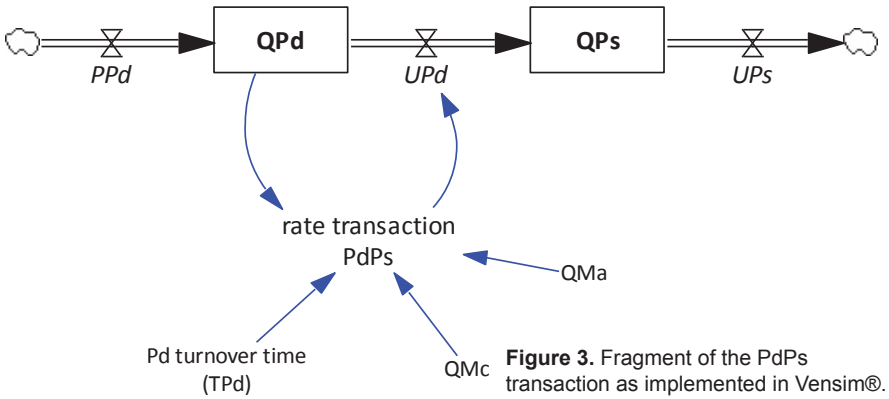


Figure 3. Fragment of the PdPs transaction as implemented in Vensim®.

3.2 Questionnaire

The average response for each variable analyzed by the three groups in the questionnaire is shown in Table 1 and 2. We wanted to capture the programmers' perception on the exercise performed in their group and how the same exercise would perform in a different paradigm.

Table 1. Average score for characteristics of the paradigms*.

	NAA	OOA	GDA
Design complexity (small models)	4	2	4
Design complexity (large models)	4	5	3
Design support (small models)	4	4	4
Design support (large models)	3	4	2
Implementation effort (small model)	4	2	4
Implementation effort (large model)	4	5	3
Intuitiveness	3	4	4
Multidisciplinary communication	3	3	4
Mathematical expressiveness	5	3	2
Visual expressiveness	2	3	5

*score 1 to 5 for the 3 paradigms (5 is better).

Table 2. Average score for characteristics of the model implementation process and results*.

	NAA	OOA	GDA
Code reuse	3	5	3
Code length/visualization	4	3	3
Ease of transcription	5	3	4
Software integration	4	4	2
Code maintenance	4	4	3
Scalability	3	5	3
Mathematical analysis support	5	2	1

*score 1 to 5 for the 3 paradigms (5 is better).

4 Discussion and conclusions

By looking at how programmers and domain experts evaluate their outcome in comparison to other paradigms, we expected to draw some conclusions about what should be considered when a modeler or a modeling group needs to decide on what is the most appropriate implementation strategy for a SDM.

The main advantage of the NAA approach is related to the highly developed mathematical tools available to implement and simulate complicated models without requiring a deep background in numerical methods for the solution and simulation of SDMs. In this context, a group with no specialist in numerical computing can solve fairly complicated SDMs in a reasonable amount of time. When the numerical computing specialist is present, really complicated SDMs with detailed scenarios can be easily simulated.

A fundamental issue about the NAA approach is what software to use. The financial resources needed to acquire the licenses for MATLAB® may be impeditive for NAA, and although free software solutions exist (e.g. Octave®, Maxima®, Scilab®, Sage®), their functionality are not as good as in commercial solutions. Another fundamental drawback is the impossibility to generate executable files or, as is the case with MATLAB®, the size

for the executable generated. Very simple applications can generate hundreds of megabytes of executable files. The choice for NAA can also be precluded by a layman user of an implemented SDM.

When looking at small models, NAA and GDA paradigms seem to be the best strategy for implementation. Although the design support was ranked equal for the three paradigms analyzed, the design complexity associated with OOA, perhaps because of the need to define logic for classes and objects, makes the implementation effort for OOA not appropriate for small models. OOA seems to be more adequate for large models, where it was found to be the best strategy in terms of design complexity, design support and implementation effort according to the programmers' perceptions.

For the GDA, better visual expressiveness comes at the expense of mathematical expressiveness. This is perhaps a feature that can drive decisions if there is no domain expert in the group. NAA and OOA groups would have found the model difficult to implement with a graphically-oriented paradigm only by having access to the model's equations. In Vensim®, it is necessary to define stocks, flows and feedbacks between variables, which require a deeper knowledge of the problem domain. This is also the reason why GDA would require from the NAA and OOA programmers more multidisciplinary communication. On the other hand, NAA's better mathematical expressiveness comes at the expense of intuitiveness, as OOA and GDA were found to be more intuitive than NAA. This result is probably related to the fact that both OOA and GDA design required a better comprehension of the concepts behind the problem domain in order to define stocks and flows or to design classes and objects.

As the groups had access only to a detailed description of the equations to be implemented, the greater mathematical expressiveness of the NAA paradigm favored the transcription of the equations into code. Both OOA and GDA required a deeper model comprehension to design the implementation, which made the exercise more demanding in comparison to the NAA paradigm. This observation seems to be in line with Rosson and Gold (1989) suggestion that in the earlier phases of the OOA, understanding the problem is more important than the expertise of the paradigm.

Based on the questionnaire and group discussions, a descriptive framework of what should be considered when deciding on what implementation strategy to follow was developed. We suggest that the choice of what strategy to use should be driven by a combination of variables related to the model characteristics, group member's expertise and the properties intrinsic to each programming paradigm. From the diagram (Figure 4), we can recognize two very distinct situations regarding the mathematical complexity of the SDM. In the case where the model is mathematically or numerically complex, GDA is excluded from the choices, given the lack of sophisticated mathematical tools in this approach. The OOA approach, on the other hand, would be a good choice only if there is an objected-oriented specialist in the group, the amount of time available is large and the characteristics of the solution are better attained with the OOA approach. Furthermore, a domain expert would be necessary in the case of a structurally complex SDM without an elegant and complete description.

The second characteristic of the SDM to take into account when the mathematical complexity is not high is its structural complexity. If this complexity is low, the decision of what approach to follow is broad and

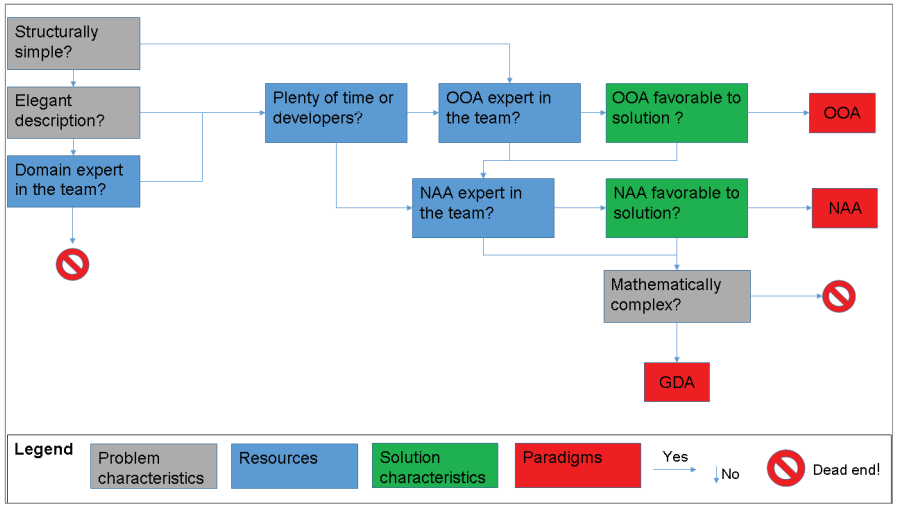


Figure 4. Descriptive framework to choose what is the most appropriate model implementation strategy considering model properties, paradigm.

should be related to the experts available and the characteristics of the solution being sought. However, if the structural complexity is high, another characteristic of the SDM have to be considered: the “elegance” of the model description. Elegance in this context expresses how much the model is objectively described and how much of its underlying process are captured in the description. If the model description is elegant, the OOA approach is attractive only in the case in which there is plenty of time available for developing the solution. On the other hand, if the SDM description lacks elegance, the group will need to incorporate a domain expert to clarify the underlying model structure in order to develop an OOA structure, regardless of time and resource constraints.

We have also analyzed the impact of the programmers’ expertise on their perceptions by calculating an average for each variable analyzed in the questionnaire, weighted by their level of knowledge reported for each paradigm. No significant differences were found, suggesting that the expertise in the paradigm did not have a significant impact on programmers’ perception of the implementation strategy. This may be due to the discussions that the three groups were allowed to engage after the exercise. When the groups compared their implementation products, a common perception of how each paradigm would perform on the exercise was built. It is also important to note that, as is the case for NAA, perceptions of the OOA approach may depend on what software is available. Here, we used a C++ simulation framework, which was probably more difficult for a layman to follow than if we had used another computer language. Further research with different participants, a better balance of participants and with different software is needed to extend the analysis of how paradigms relate to the model and group’s different expertise. Such studies would assist researchers in deciding what would be the most appropriate implementation strategy for each SDM.

5 References

DIJKSTRA, J.; NEAL, H. D.; BEEVER, D. E.; FRANCE, J. Simulation of nutrient digestion, absorption and outflow in the rumen: model description. **The Journal of Nutrition**, v. 122, n. 11, p. 2239-2256, Nov. 1992.

EIERMAN, M. A.; DISHAW, M. T. The process of software maintenance: a comparison of object-oriented and third-generation development languages. **Journal of Software Maintenance and Evolution**: Research and Practice, v. 19, n. 1, p. 33-47, Jan./Feb. 2007. DOI:10.1002/smr.343.

HILLYER, C.; BOLTE, J.; EVERT, F. van; LAMAKER, A. The ModCom modular simulation system. **European Journal of Agronomy**, v. 18, n. 3-4, p. 333–343, Jan. 2003. DOI:10.1016/S1161-0301(02)00111-9.

JOINES, J. A.; ROBERTS, S. D. Fundamentals of object-oriented simulation, In: CONFERENCE ON WINTER SIMULATION, 30., 1998, Washington, D.C. **Proceedings**... New York: Association for Computing Machinery; Piscataway: IEEE 1998. p. 141-150. DOI: 10.1109/WSC.1998.744909.

LIM, J. S.; JEONG, S. R.; SCHACH, S. R. An empirical investigation of the impact of the object-oriented paradigm on the maintainability of real-world mission-critical software. **Journal of Systems and Software**, v. 77, n. 2, p. 131-138, 2005.

MANCINI, A. L.; BARIONI, L. G.; LIMA, H. N.; SANTOS, J. W.; SILVA, R. D. R.; SANTOS, E. H.; DIAS, F. R. T. A compact and flexible C++ framework to support modular development of hierarchical dynamic systems simulators. In: INTERNATIONAL CONGRESS ON MODELLING AND SIMULATION, 20., 2013. Adelaide. **Proceedings**... San Diego: Society for Computer Simulation International, 2013. Disponível em: <http://dl.acm.org/ft_gateway.cfm?id=2665041&ftid=1496326&dwn=1&CFID=463709126&CFTOKEN=84911368>. Acesso em: 16 dez. 2014.

RAMAMOORTHY, C. V.; PRAKASH, A.; TSAI, W. T.; USUDA, Y. Software engineering: problems and perspectives. **Computer**, v.17, n. 10, p. 191-209, Oct.1984. DOI: 10.1109/MC.1984.1658970.

ROSSON, M. B.; GOLD, E. Problem-solution mapping in object-oriented design. **ACM Sigplan Notices**, v. 24, n. 10, p. 7-10, Oct. 1989. Special issue: Proceedings of the Conference on object-oriented programming systems, languages and applications, New York, 1989.

Agriculture Informatics

Ministry of
Agriculture, Livestock
and Food Supply

CGPE 11735