CoSMoS process, models, and metamodels

Paul S. Andrews[1], Susan Stepney[1], Tim Hoverd[1],
Fiona A. C. Polack[1], Adam T. Sampson[2], and Jon Timmis[1]

[1] YCCSA, University of York, UK, YO10 5DD
[2] Institute of Arts, Media and Computer Games, University of Abertay Dundee, UK, DD1 1HG

Abstract. We summarise the existing CoSMoS approach to modelling and simulating complex systems, then introduce how the various CoSMoS models are related via their metamodel, and demonstrate the generality of the process by discussing its application to engineering bio-inspired systems.

1 Introduction

The CoSMoS project is developing tools and techniques that enable the construction and exploration of simulations for the purpose of scientific research. This is a necessarily interdisciplinary endeavour between scientists who study a particular domain (the *domain experts*), and software engineers who construct simulations to facilitate the study of that domain (the *developers*). Together, the domain experts and developers are involved in open-ended scientific research: the simulations are used as a tool to support theory exploration, hypothesis generation, and design of real-world experimentation [10, 11].

Our work is driven by a series of simulation case-studies from a broad range of disciplines (immunology, ecology, and sociology), with a focus on simulating *complex systems*. Such systems are amenable to computer simulations, displaying high-level system behaviours that emerge as a consequence of many simple behaviours at a lower level, where the high-level behaviour cannot be readily deduced as a simple combination of the low-level behaviours. A classic example of a complex system behaviour is bird flocking: the individual behaviour of birds flying together can result in the emergence of a flock at the population level. Computer simulation allows us to examine such systems from a *holistic* point of view.

To run computer simulations we need to engineer a simulation platform. This requires us to explicitly represent some knowledge of the system being studied in a form that can be implemented on a computer. This representation, the *source code*, is either designed manually by the developers or automatically generated from a higher-level description. In many existing approaches the source code is the only explicit description of the aspects of the target domain that are being simulated. Source code contains numerous implicit assumptions[3] concern-

[3] An assumption is any kind of abstraction, simplification, axiom, idealisation or approximation.

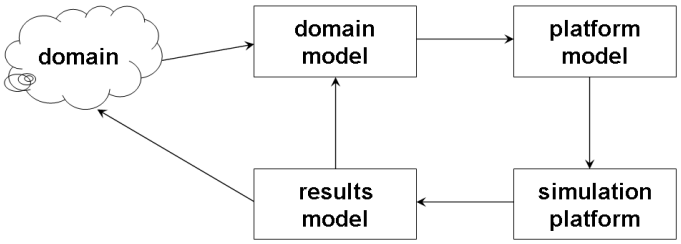

Fig. 1. Relationship between models, domain and simulation platform, where arrows represent flows of information. These are all framed by the research context.

ing both the scientific aspects of the work, and the engineering design of the simulation platform.

To mitigate inappropriate assumptions in the design of simulation platforms, and to have greater confidence that simulation results can actually tell us something that relates to the real system being studied, we use a series of related models to drive and describe the development of the simulation platform and simulation results generated from its use. Systematic development assists interaction between domain experts and developers, and improves our confidence in, and interpretation of, the results of simulations [12].

Here we summarise the existing CoSMoS modelling process[4], and introduce the CoSMoS metamodel. In §2 we describe the existing CoSMoS modelling approach in terms of its various models. In §3 we introduce a metamodel and explain how it relates these CoSMoS models. In §4 we demonstrate the generality of the CoSMoS process by discussing its application to engineering bio-inspired systems.

2 The CoSMoS process

Our approach uses the following concepts [1]: *domain*, *domain model*, *platform model*, *simulation platform*, and *results model*. The domain represents the real-world system (or part of a system) that is being investigated. This is related to the models and simulation platform as shown in figure 1, where the arrows denote a flow of information from one concept to the next. Like all scientific work, the five development concepts are shaped by the *research context*.

Research Context: identifies the overall scientific context and scope of the simulation-based research being conducted. The scientific context can be elucidated by recording high-level motivations or goals, research questions, hypotheses, general definitions, and success criteria (how will you know the simulation has been successful). It is important to identify when, why and how these change

[4] Project documentation of simulation, modelling and process descriptions [1, 11, 12], of validation and argumentation [5, 6, 10], of various biological system simulation case studies [2–4, 13], and of the CoSMoS workshop proceedings [15, 17, 18], is available from the CoSMoS project website www.cosmos-research.org

throughout the course of developing and using the simulation. The scope of the research determines how the simulation results can be interpreted and applied. Importantly, it captures any requirements for validation and evaluation of simulation outputs. Consideration should be made of the intended criticality and impact of the simulation-based research, and if these are judged to be high, then an exploration of how the work can be validated and evaluated should be carried out. In summary, the role of the research context is to collate and track any contextual underpinnings of the simulation-based research, including the scientific background, and the technical and human limitations (resources) of the work.

Domain Model: explicitly captures understanding of the domain, identifying and describing the structures, behaviours and interactions present in the domain at a level of detail and abstraction suitable for addressing any identified research questions to be posed of the simulation platform. It is a model based on the science as presented by the domain experts, and its design should be free from simulation platform implementation bias; it separates the model of the science from the implementation details of the simulation platform. Importantly, the domain model is developed with the domain experts, and is used as a tool to exchange and discuss domain understanding between developers and domain experts. The domain model forms the agreed scientific basis for the eventual simulation platform.

Platform Model: an engineering derivation from the domain model, and a step towards simulation platform construction. The model is shaped by engineering design decisions, detailing the implementation of the structures, behaviours and interactions identified in the domain model in a way that naturally translates to simulation platform technologies. This might dictate that some concepts in the domain model are abstracted or simplified, to allow efficient implementation. Some high-level emergent behaviours identified in the domain model are removed from the platform model, if the purpose of the research is to investigate the emergence of these behaviours from other model components. In general, given a hypothesis under consideration, components in the domain model that are *outcomes* of hypothesised mechanisms should not appear in the platform model: the answer should not be explicitly coded into the simulation platform, but must appear in some model. The platform model also adds instrumentation and interfaces to allow observation (visualisation), user interaction, and recording of the eventual results of using the simulation platform.

Simulation Platform: encodes the platform model in software and hardware platforms with which simulations can be performed. The simulation platform defines a set of parameters (variables) that allow the encoded model to be manipulated. The parameters are derived from the domain model and interpreted through the platform model, thus making the simulation platform accessible to domain experts with knowledge of the domain model.

Results Model captures understanding of the simulation platform based on the output of simulation runs, and provides the basis for interpretation of what the simulation results show. Its relationship to the simulation platform is analogous to the relationship between the domain and domain model. The results

model is constructed by experimentation and observation of simulations, and might record observations, screen-shots, dynamic sequences, raw output data, result statistics, as well as qualitative or subjective observations. The contents of the results model are compared to the domain model to establish whether the simulation platform provides a suitable representation of the real-world domain being investigated. The results model might also provide details to develop new experimentation, either on the simulation platform or in the real domain.

The domain, platform, and results models are generated and updated throughout simulation-based research: establishing the scientific basis, developing the simulation platform, and using the simulation platform to explore the domain. The models are used as devices to capture, communicate and reason about different aspects of the construction and use of the simulation platform, and how this relates to the domain. This includes annotating the appropriate model with scientific or engineering assumptions. By using a principled approach to simulation, the research is ultimately open to review and challenge, and provides a basis for scientific reproducibility.

3 Models and Metamodels

The CoSMoS process emphasises the need for separate domain, platform, and results models. In this section we describe how these three models are related, and how they differ, in terms of metamodelling. The discussion here assumes that the modelling approach is agent-based, and that the research context is investigating emergent properties. Other modelling approaches and other research contexts are also possible: the specifics change, but the overall approach remains the same.

A model such as used in CoSMoS provides the abstract language of the relevant concepts; the platform model captures the concepts to be implemented in the (simulation platform) code. A metamodel provides the analogous language for writing a *model*: it defines the kinds of things that can occur in the model (it is the model of the model) [8, ch.8].

A metamodel can provide a rigorous link between different yet related models. For example, in agent-based modelling of systems with emergent properties, the metamodel includes concepts such as Agent, Rule, and Emergent. For a different style of model, such as an ODE model, the metamodel includes different concepts, such as Concentration and RateOfChange. An agent-based model of ant pheromone trails would include classes such as: Ant, an instance of Agent; and Trail, an instance of Emergent. An agent-based model of bird flocking would include classes such as: Bird, an instance of Agent; and Flock, an instance of Emergent.

The metamodel for the CoSMoS domain and results models must be essentially the same: the results model is constructed in the same language as the domain model, to enable direct comparison of the two, and so that the results are presented in a domain-relevant language. The platform metamodel differs

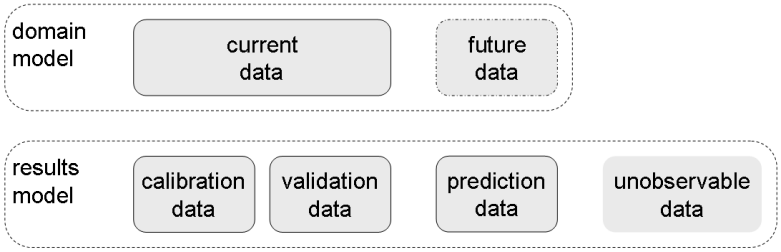

Fig. 2. The relationship between data in the domain and results models.

in that it does not include the hypothesised emergent properties, but it does include interface and instrumentation concepts.

The components of an agent-based domain and results metamodel could include:

- Agent: the types of the entities in the agent-based model (eg, birds)
- Environment: the environment within which they act (eg, obstacles, gravity, wind)
- Rule: the agents' behavioural rules, including how they interact with each other (eg, flocking rules), and with their environment (eg, obstacle avoidance rules)
- Emergent: emergent properties exhibited by the dynamics of the model (eg, flocking)
- Data: scientific data used to quantify the model (eg, bird positions and velocities)
- Measure: a quantitative measure calculated from the data (eg, an entropy-based flocking statistic)

The domain and results models share a common metamodel. This does not mean that the two models are identical; it means that they are cast in the same language. The domain model has instances of metamodel concepts that capture specific domain concepts; the results model has instances of simulation analogues of those domain concepts. So where the domain model has Bird, the results model has Boid, the simulation analogue of Bird; both are instances of the metamodel concept Agent. The results model has Data instances, which stand in the same relation to its Agent instances as they do in the domain model (so if the domain model has bird positions and velocities, the results model has the corresponding boid positions and velocities). This means that the Measure instances can be essentially identical, allowing a direct comparison of the models *in domain terms*, see figure 2. The results model needs data so that it can be calibrated suitably; it needs further data so that it can be validated against the domain model (this is analogous to training and test data in machine learning [7]). At this stage it can be used to analyse data from novel scenarios, to make predictions; the domain can be augmented with new experimental data to test those predictions. Data in the

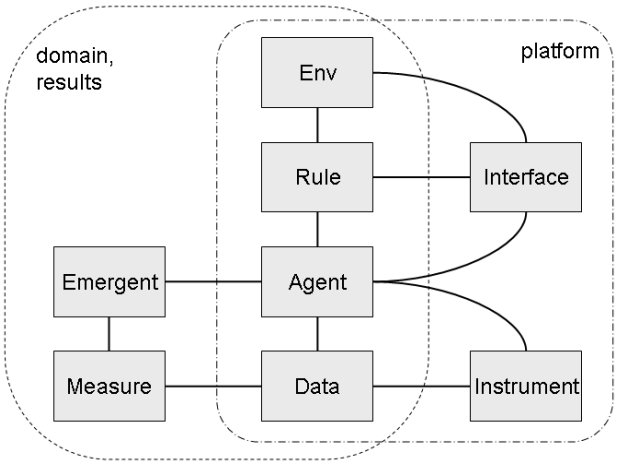

Fig. 3. A metamodel for agent-based simulation of emergent properties. The dashed boxes indicate the components of the metamodel that are used to describe the domain and results models, and the platform model.

results model that is not observable (even indirectly, through surrogates, or by investigating predictions) in the domain model is of little use. The necessity for suitable data in the results model implies requirements on the platform model.

The platform metamodel is different from that of the domain and results metamodel. In particular, it has no Emergent concept; it is important that there is no way to program the desired answer into the simulation: it must emerge. (If the research context is not concerned with emergent properties, but some other kind of property, it is equally important to ensure that this does not get programmed in to the simulation.) The platform model adds Interface concepts, to allow user interaction with the simulation, and Instrumentation concepts, to allow extraction of the Data. The Data instances are common to the platform model and results model: that are extracted from the simulation, and analysed in the results model.

Although it is important that the domain and platform metamodels are different, it is also important that they share many concepts, allowing a common language. We define a single metamodel sufficient for all the individual models, and indicate which parts of it are specific to particular models, and which parts are common across the models. A schematic view of this metamodel is shown in figure 3.

A schematic view of part of a possible model is shown in figure 4. Not all the concepts in the metamodel need be instantiated as *classes* in the model. For example, Rule might be instantiated as rules of interaction, as behaviours of boids; Env might be instantiated as a system, of several classes and methods.

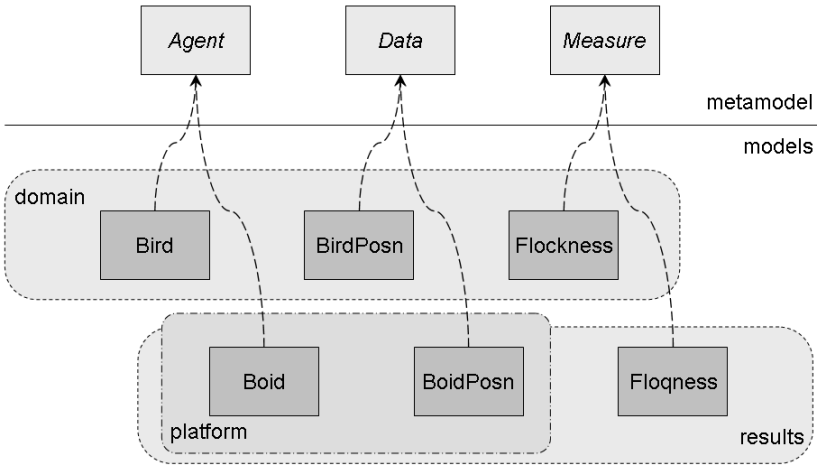

Fig. 4. The domain, platform, and results models for agent-based simulation of emergent flocking properties. The dashed arrows represent instance relationships between model concepts and their metamodel concepts. The dashed boxes show the components of the models: the domain has different concepts from those shared by the platform and results models.

4 CoSMoS process applications

The CoSMoS process described in §2 is a minimal process, suitable for building scientific simulations of real world domains. The underlying concepts of different models, and of a common metamodel, are generic, however, and can be used in a wider range of applications. Here we illustrate this by discussing their use for designing a class of bio-inspired algorithms (where the real-world domain is not being investigated for its own reasons, but is being used for inspiration), and for engineering a bio-inspired domain (where the engineered domain is not pre-existing, but is being built as part of the process).

4.1 The conceptual framework for bio-inspired algorithms

Some of us have previously described a conceptual framework [16] for the principled development of bio-inspired algorithms. This conceptual framework (figure 5) describes a process of modelling the biology, abstracting out principles, and instantiating those principles as computational concepts (rather than building a direct, and naive, analogue of the biology). Probes (observations and experiments) are used to provide a partial view of the complex biological system. From this we build and validate simplified abstract representations of the biology. From these biological models we build analytical computational frameworks. These frameworks provide principles for designing and analysing bio-inspired algorithms applicable to non-biological problems, possibly tailored to a range of

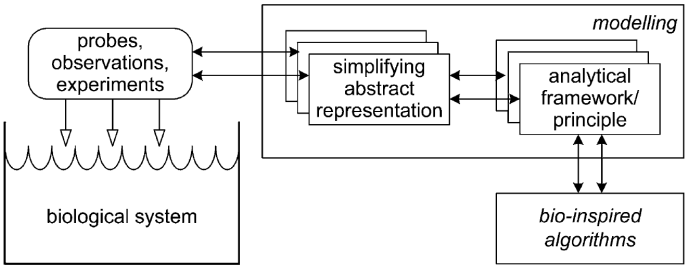

Fig. 5. A conceptual framework for bio-inspired algorithm design [16, fig.1].

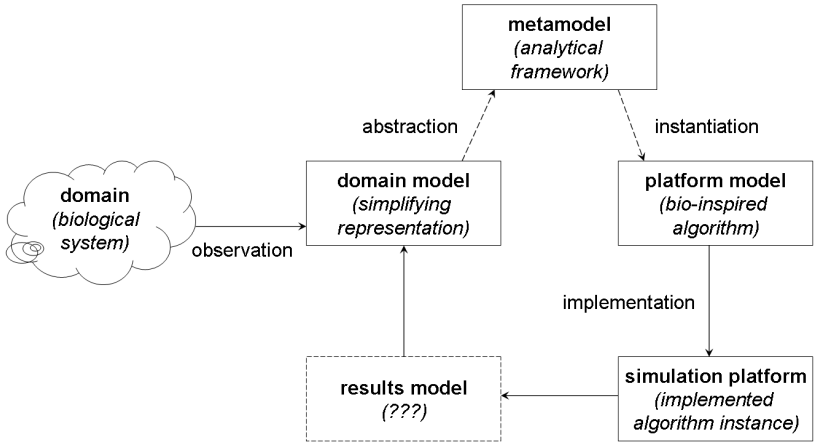

Fig. 6. The conceptual framework in terms of CoSMoS process models and metamodels. The framework components fit into the CoSMoS process. There is nothing in the framework explicitly corresponding to the CoSMoS results model.

problem domains, and contain as much or as little biological realism as appropriate.

The conceptual framework is consistent with the CoSMoS modelling approach, with the identification of the components as shown in figure 6. The domain is the biological system from which we wish to draw our inspiration. The domain model is a suitable model of this biological system, that will form the basis for the bio-inspired algorithm design. We then abstract this model into a metamodel, capturing the relevant concepts and relationships of the biological model. At this meta-level, we can abstract away contingent details of the biology that are of no relevance to an algorithm. We then develop a separate instantiation of this same metamodel in the computational domain: the platform model. Thus the bio-inspired model is related to the biological model *via the metalevel abstraction*, rather than by a direct naive mapping.

There is nothing in the conceptual framework [16] that explicitly corresponds to the CoSMoS results model. The purpose of the results model in simulation experiments is to compare the simulation with the real-world domain being investigated. Here the real-world domain (biological system) is not being investigated; it is being used as inspiration for the algorithm development. The domain model and platform model might be very far removed from each other. However, the reason that the biological system is being used as inspiration is that it has (usually emergent) properties that are wanted in the algorithm, for example, robustness. The results model could therefore be used to capture the algorithm's properties, and to validate that the algorithm as implemented has indeed captured these desired bio-inspired properties (as defined at some suitable level of abstraction, in the metamodel). Additionally, an explicit focus on these properties helps guide the initial domain modelling and abstraction process.

The description of the conceptual framework [16] does mention validation, but it is only a point-to-point process between consecutive models, and is not explicit about the purpose or means of the validation. Here we have an explicit place and approach to include validation of algorithm properties within the process.

4.2 Two domains

Bio-inspired algorithms are often developed for a particular purpose, for example, to engineer a particular bio-inspired system. The CoSMoS process for bio-inspired algorithm development described in §4.1 can be combined with the base simulation process of §2, as shown in figure 7. For example, this could describe the process of engineering a bio-inspired swarm robotics system.

The lower five boxes in figure 7, those joined by solid lines, represent a small modification of the base CoSMoS process. The main difference is that the domain is engineered, and so does not exist initially; the domain model is the engineering specification: the domain is engineered to respect the model (not the other way round, as when modelling a natural domain). The 'design validation' demonstrates whether the simulation is an adequate simulation of the engineered domain.

The upper boxes represent the biological inspiration, as seen in §4.1. The 'property validation' demonstrates whether the engineered domain (via its capture in simulation) exhibits the desired biological properties.

This approach is not specific to bio-inspired engineering. It can cover any case where there is some 'inspiration domain' being exploited to help engineer a 'solution domain'. What is important is to understand what information is coming from what domain, and how the domains are linked, not directly, but through abstract models and metamodels.

We are currently applying this approach to developing a swarm robot system to investigate Simon's Loose Horizontal Coupling hypothesis [9, 14].

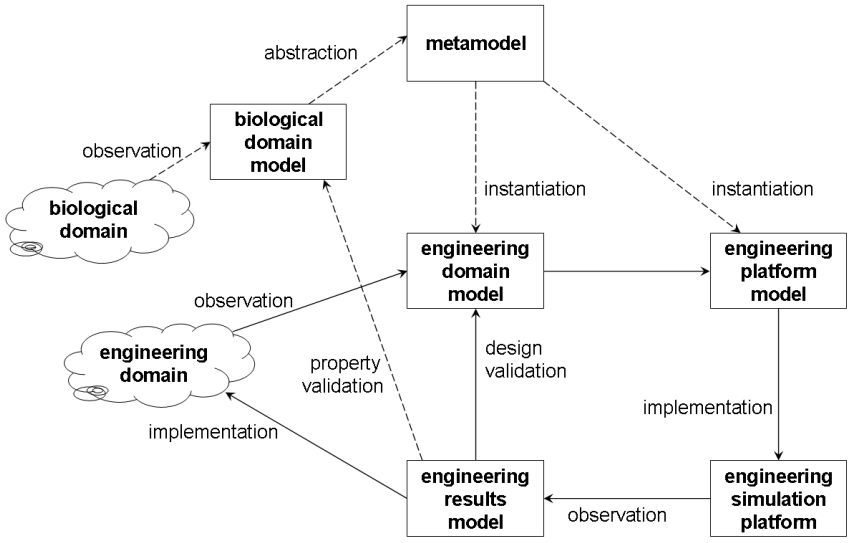

Fig. 7. A bio-inspired engineered domain.

5 Summary and Conclusions

The CoSMoS process emphasises the building of a range of models, satisfying different purposes. In particular, the hypothesised properties (for example, emergent properties) are captured in the domain model, where they can be analysed, but are not present in the platform model, ensuring that they are not explicitly implemented. The results model allows the platform simulation outputs to be analysed and compared to the domain model.

Here we have introduced the use of an overarching metamodel in the CoSMoS process, to ensure that the various models are expressed in a common language, and to define the relationship between them.

We have shown an example of how the CoSMoS model and meta-model approach can be used to analyse pre-existing development frameworks, and to identify missing components that have a valuable function. We have also sketched how the process can be applied to simulations of bio-inspired engineered domains, in addition to simulation of pre-existing natural domains.

Acknowledgements

This work is part of the CoSMoS project, funded by EPSRC grants EP/E053505/1 and EP/E049419/1, and a Microsoft Research Europe PhD studentship. We thank the anonymous referees for helpful suggestions.

References

1. Paul S. Andrews, Fiona A. C. Polack, Adam T. Sampson, Susan Stepney, and Jon Timmis. The CoSMoS process, version 0.1: A process for the modelling and simulation of complex systems. Technical Report YCS-2010-453, Department of Computer Science, University of York, March 2010.
2. Anton Jakob Flügge, Jon Timmis, Paul Andrews, John Moore, and Paul Kaye. Modelling and simulation of granuloma formation in visceral leishmaniasis. In *CEC 2009*, pages 3052–3059. IEEE Press, 2009.
3. Philip Garnett, Susan Stepney, Francesca Day, and Ottoline Leyser. Using the CoSMoS process to enhance an executable model of auxin transport canalisation. In Stepney et al. [17], pages 9–32.
4. Philip Garnett, Susan Stepney, and Ottoline Leyser. Towards an executable model of auxin transport canalisation. In Stepney et al. [15], pages 63–91.
5. Teodor Ghetiu, Robert D. Alexander, Paul S. Andrews, Fiona A. C. Polack, and James Bown. Equivalence arguments for complex systems simulations - a case-study. In Stepney et al. [18], pages 101–140.
6. Teodor Ghetiu, Fiona A. C. Polack, and James L. Bown. Argument-driven validation of computer simulations – a necessity rather than an option. In *VALID 2010:*, pages 1–4. IEEE Press, 2010.
7. Paolo Giudici. *Applied Data Mining: Statistical Methods for Business and Industry*. Wiley, 2003.
8. Anneke Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: the Model Driven Architecture: practice and promise*. Addison-Wesley, 2003.
9. Jennifer Owen, Susan Stepney, Jon Timmis, and Alan Winfield. Exploiting loose horizontal coupling in evolutionary swarm robotics. In *ANTS 2010, Brussels, Belgium, September 2010*, volume 6234 of *LNCS*, pages 432–439. Springer, 2010.
10. Fiona A. C. Polack. Arguing validation of simulations in science. In Stepney et al. [17], pages 51–74.
11. Fiona A. C. Polack, Paul S. Andrews, Teodor Ghetiu, Mark Read, Susan Stepney, Jon Timmis, and Adam T. Sampson. Reflections on the simulation of complex systems for science. In *ICECCS 2010*, pages 276–285. IEEE Press, 2010.
12. Fiona A. C. Polack, Paul S. Andrews, and Adam T. Sampson. The engineering of concurrent simulations of complex systems. In *CEC 2009*, pages 217–224. IEEE Press, 2009.
13. Mark Read, Paul S. Andrews, Jon Timmis, and Vipin Kumar. A domain model of experimental autoimmune encephalomyelitis. In Stepney et al. [18], pages 9–44.
14. Herbert A. Simon. The organization of complex systems. In Howard H. Pattee, editor, *Hierarchy Theory*, pages 1–27. George Braziller, 1973.
15. Susan Stepney, Fiona Polack, and Peter Welch, editors. *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2008.
16. Susan Stepney, Robert E. Smith, Jon Timmis, Andy M. Tyrrell, Mark J. Neal, and Andrew N. W. Hone. Conceptual frameworks for artificial immune systems. *International Journal of Unconventional Computing*, 1(3):315–338, July 2005.
17. Susan Stepney, Peter H. Welch, Paul S. Andrews, and Adam T. Sampson, editors. *Proceedings of the 2010 Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2010.
18. Susan Stepney, Peter H. Welch, Paul S. Andrews, and Jon Timmis, editors. *Proceedings of the 2009 Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2009.