Reflections on the Simulation of Complex Systems for Science

Fiona A. C. Polack,
Paul S. Andrews, Teodor Ghetiu,
Mark Read and Susan Stepney
Department of Computer Science
University of York
York, UK YO10 5DD
Email: Fiona.Polack@cs.york.ac.uk

Jon Timmis
Department of Computer Science
and Department of Electronics
University of York
York, UK YO10 5DD
Email: jtimmis@cs.york.ac.uk

Adam T. Sampson
School of Computing
University of Kent
Canterbury, UK CT2 7NF
Email: ats@offog.org

***Abstract*—In studying complex systems, agent-based simulations offer the possibility of directly modelling components in an environment. However, the scientific value of agent-based simulations has been limited by inadequate scientific rigour. The paper focuses on agent-based simulations that are used in biological and bio-medical research. Starting from a review of best practice in simulation engineering, the paper identifies some of the key activities in developing complex systems simulations that support scientific research, and how these contribute to the essential development of mutual trust among developers and scientists. Examples from the authors' own experience illustrate how a range of studies have manifested these key activities, and identifies some successes and problems encountered.**

Keywords: Complex system modelling; scientific simulation; simulation validation

I. Introduction

A computer simulation is an executable model that abstracts from the real system that it models in order to focus attention on particular features and dynamic aspects. This paper is primarily concerned with agent-based simulations that are constructed to support exploration of complex biological systems.

Agent-based simulations comprise many interacting agents; these can be used to model selected biological and environmental components of systems. The scope for individual variation and the flexibility that the agent-based paradigm offers is potentially attractive to the scientific study of complex systems. In section II, we consider the reasons for lack of exploitation of the technology. We note that scientists use methods that they trust, even when the methods are less-than-completely appropriate or accurate for the defined purpose. This leads us to consider how the approach to collaboration, and to the engineering of simulations, can support use of agent-based simulation as a scientific technology.

In our experience of modelling and simulating biological and bio-medical systems, a trusting relationship between scientists and simulation developers is the essential basis for scientifically-useful and acceptable simulation. In section VI, we outline five of our own collaborative projects, showing how close collaboration and careful development practices have contributed to useful scientific simulations. The practices that we have used are not very different from other successful collaborations between computer modellers and biological research (for example the work of Calder et al. [1] or of Harel and Cohen's group [2]). In all such simulations, validation is crucial and thorough. Our ongoing work differs from others in the emphasis placed on arguing the validity – in terms of fitness for purpose – of our simulations. This aspect, which uses critical-systems engineering techniques, is alluded to throughout, but is not the main focus of this paper: a fuller coverage can be found in [3]–[5].

All these successful biological simulations are characterised by active, close collaboration. The simulation must be developed and presented in such a way that the developers and scientists understand its strengths, limitations and applicability. This requirement dominates all the conventional engineering rationale – how to design, how to implement, what to record, what validation means.

II. Observations on simulation and scientific research

It is widely asserted that much of the state-of-the-art in agent-based simulation is inadequate for scientific use, because the engineering basis of simulation is unclear (at least to those other than the developers) [6]–[9]. Wheeler et al. [8] identify the need to address deep questions of comparability: to keep a record of experience; to understand how good solutions are designed to rationalise choice of parameters and to calibrate agents; and, above all, to validate complex system simulations. Bray [10] observes that computer modellers have tended to stray from strict interpretation of scientific evidence, introducing arbitrary factors, for instance to make the simulation appear more significant by boosting its sensitivity.

Further insight into the requirements for scientifically-useful simulation comes from consideration of how scientists do research. Although it can be shown that the popular view of "scientific method", as a process of either logical induction or logical deduction from objective observation, is at worst a myth and at best an optimistic oversimplification [11]–[13], the practice of science requires careful observation and recording, and rational design of experiments. Scientific research could be characterised by a culture of mutual trust: even if scientists

disagree, or challenge the results of other groups, there are common standards of working, enforced through accepted ways of presenting research, experiments and results – many biological publications, for example, have a *methods and materials* section that contributes to the readers' understanding, and thus trust, of the theory or results presented.

Trust extends to methods and techniques. In undertaking research, scientists, individually and as a community, reject techniques that they do not trust. There is some anecdotal observation that off-the-shelf packages (including simulators) that are in general use in teaching or using science are accepted for research use – presumably because their widespread use engenders trust. It is apparent that scientists are sometimes led astray by their trust, even though it is well-founded and community-based. Ecological communities, for example, tend to be modelled as stable communities that respond to external events [14], but these are, in reality, systems in dynamic equilibrium, with behaviours due to time and internal interactions. It is sometimes the case that scientific analysis loses sight of the limitations of models. Heuristic solutions, equational and statistical methods in non-linear systems analysis, have known weaknesses in certain situations [15], that may be overlooked. Furthermore, in mathematical approaches such as ODE modelling, it is sometimes the case that scientists overlook the limitations of standard mathematical methods (such as Euler's method) – ODE integration methods may be applied outside their reliable envelope [16].

In relation to agent-based simulations, the starting position is that, currently, *most scientists do not trust agent-based computer simulations*. Lack of trust may arise either because the presentation of the simulations does not give sufficient insight to engender trust, or because the simulations are not used by others in the community. There is a need to address two connected problems: the threat of novelty, and the question of how to engage scientists appropriately in the development and use of agent-based simulations. Before considering how simulations can be constructed in scientifically-credible ways, we briefly review the ways in which simulations (of traditional as well as complex systems) are engineered.

A. Engineering and Simulation

There are many existing development methods for multi-agent systems (reviews can be found in [17], [18]). Many methods are tailored versions of general software engineering development methods: for example, ADELFE [19] is a customisation of the Rational Unified Process. Most methods are tailored to specific forms of agent-based simulation (ADELFE is for adaptive multi-agent systems with co-operative agents [19]). There are a few general simulation approaches, such as Prometheus [20]. The engineering problems of existing methods can be summarised as follows (based on [5]):

- Methods typically provide a range of complementary views (the agents, their data and interactions, actions and object message-passing), but do not provide strong support for consistency between views.

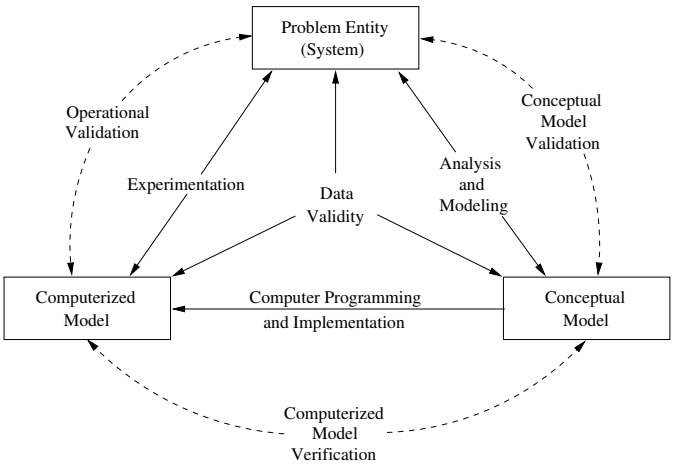

Fig. 1. Sargent's model of the simulation development process [23]

- The views provided by methods relate to a particular programming paradigm, rather than to critical features of complex systems – the components, their environment, and desired or undesirable emergent behaviours.
- Whilst methods provide seamless routes to implementation if the developer is using the appropriate family of languages, the languages supported are not always the most appropriate for simulation.

Engineering is an exercise in quality control as well as development. *Validation* checks that the right system is built – it is about meeting requirements and quality (whereas *verification* checks that the system is built right – it is about correctness of construction). Validity implies both adequate abstraction, and adequate development processes.

In high-integrity systems engineering, the validation of simulation has been a focus of interest since the late 1970s. Recent interest in complex systems validation [4], [21], [22] takes inspiration from Sargent's development lifecycle [23], shown in Figure 1. The *problem entity* is the phenomenon to be modelled. From understanding the problem entity, a *conceptual model* is developed in a suitable representation – Sargent reviews diagrammatic models [23], and also considers mathematical or logical representations [24]. The *computerised model* implements the conceptual model as a simulation. The *experimentation* link between the problem entity and the computerised model allows iterative (even agile) trial-and-error simulation, with the models and results compared to the problem entity at each step. Sargent explicitly incorporates verification (in the software engineering of the computerised model) and validation – of all models against the problem entity, and of the data used to test or populate the conceptual and computerised models. The lifecycle has much in common with conventional software engineering lifecycles – it presents a high-level summary of the necessary attributes of a development, rather than a comprehensive guide to achieving a high-quality engineered product.

Elsewhere [25], we consider how complex systems differ from conventional systems, and use this understanding to propose an extension to Sargent's process [4], [26]. Essentially, the components of a complex system can be designed and verified by a conventional engineering process, but the complex effects of the components only become apparent when they are considered in their environmental context. A small change in the environment can dramatically change the nature, and even the occurrence, of high-level behaviours. In describing the problem entity and developing the conceptual model, *the extraction of relevant environmental characteristics is as important as the modelling of the components*. In [5], we show that the quality of software engineering, and particularly the feasibility of verification, is enhanced by continuity and consistency in engineering of environment, components and their interactions. Sargent [24] reminds us that a *model should be developed for a specific purpose... and its validity determined with respect to that purpose.* The level of assurance also depends on the purpose of the simulation, and should be set independently of the development of the simulation – good software engineering practice.

Traditional techniques for validation of simulations [24], [27] do not transfer easily to complex systems simulation. For instance, one of Sargent's proposals entails detailed analysis of the simulated and real systems in terms of event validity and traces: if we knew the workings of a complex system well enough to understand event validity and traces, we would not need a computer simulation to help us understand it. Other tests, such as face validity [28], are even less sufficient for evaluating complex systems simulations (though these traditional techniques may be appropriate for parts of complex systems: event validity and traces can be appropriately applied in the design of component behaviour).

The most useful of Sargent's suggestions [24] is the *analysis of assumptions*. This approach is typical of critical systems techniques, where methods based on flaw hypothesis and systematic challenge are used to extract and document assumptions and side-effects. Techniques using argumentation can be used to structure assumptions and the engineering rationale. These approaches can also be applied to validation in complex system simulation [3], [5].

III. Domain and domain model for a simulation

The domain and domain model are the starting points in developing a simulation to assist in scientific study. These define not only the scientific context and scope of a simulation exercise, but also who is involved. Without these considerations, a simulation is unlikely to be understood and accepted for research use.

A. Scientists and developers

It is necessary to establish who the simulation is constructed for, and who it is constructed by – in software engineering terms, we need to establish the *stakeholders*. Here, the *scientist* stakeholder comprises a group of scientists investigating a particular scientific subject: the scientific subject forms the *domain*. The *developer* stakeholder is the team that is to engineer a suitable simulation. The labelling of the stakeholders is for convenience in exploring and explaining the process of doing science with simulations, which is fundamentally a collaborative enterprise.

The team of scientists and developers could comprise one or more people, but we assume here that the scientist is not the developer. It is not unusual for scientists to produce computer simulations, and the result is likely to be satisfactory for small-scale demonstration, but, in general, scalability and good-quality engineering requires specialist knowledge. Without engineering expertise, it is unlikely that a simulation can be scaled to the millions of components that are needed to mimic many natural systems. It is also unlikely that the simulation can be easily modified or maintained to support ongoing scientific experimentation and the investigation of new or variant scientific hypotheses. Finally, in the same way that people who are not mathematical experts accidentally abuse mathematical techniques, credible scientific use of simulation results requires deep understanding of the construction and limitations of the simulation.

Section VI summarises a number of simulation studies in which the key to success was a collaboration that engendered mutual trust. It is hard to write about collaboration and the development of trust without separating out the parties involved – trust is a state of mutual understanding between *independent* parties. Furthermore, this paper is presented from the point of view of the developers, and is targeted at an audience of developers, so it is inevitable that the scientific role is presented as, in some sense, separate. However, in relation to collaboration of stakeholders, *the simulation is only useful to a scientist if the scientist trusts the simulation*; trust is a mutual state, arrived at by sharing.

The criticality of the collaboration sounds a note of caution: not all scientific researchers and computer engineering proficients are able or willing to engage in this level of collaboration. Commitment and openness are necessary. Both sides need to be able to ask naive questions and challenge the other side's traditions, and both sides need to be prepared to spend time recording and checking their sources. In sections VI-A and VI-B, we summarise studies in which computer scientists and immunologists collaborated to produced simple models that allowed insight into aspects of systems that could not be studied directly in laboratory experiments. Here, the collaboration was initially established by a meeting of the whole team, but subsequently involved a close working relationship between one scientist and one developer, with clear links to the rest of the team. A similarly-close relationship exists in the EAE study, section VI-C. In the plant studies (section VI-D and VI-E), collaboration is established more formally through co-supervised doctoral students.

It is important to note that, although the scientist is, and remains, the domain expert, the development of the collaboration should involve, and probably requires, a mutual transfer of expertise. It is certainly the case, in the examples considered, that the key developer in each team has gained considerable

knowledge of a small part of the domain. The scientists have also learnt about the process and potential of agent-based simulation; they become better able to identify their own (and the developers') assumptions, to present the environment of their research, and to respond to apparently-naive questions with appropriate information.

It is tempting to see the close working relationship between scientists and developers as representing *agile* development. Scientific-simulation engineering shares many features of agility – customer involvement, changing requirements, motivated individuals, sustainable development (see agilemanifesto.org). However, simulation engineering differs from agility in the crucial respect that scientists and developers construct sketches and diagrams to establish and record their mutual understanding of the domain model. These models are also crucial in validation.

B. Establishing domain, domain model and purpose

The domain is a definition of the scientific scope, from the perspective of the scientist. From the domain, the scientist and developer determine the purpose of the simulation, to guide the developer towards an appropriate form of simulation.

The *developer is seeking to simulate part of the scientists' model of reality*; the developer is not modelling the domain itself. If the *domain* represents all the scientists' knowledge about the relevant subject area, then the *domain model* represents the mutual understanding of a subset of that knowledge, the developers' starting point for simulation. In one sense, the domain model can be thought of as a development contract: the developer relies on the scientist to provide appropriate information about the domain, and guarantees a suitable simulation; the scientist relies on the developer to use the domain information appropriately, and guarantees to work with the developer to ensure that this is the case.

In some cases, simulation development is prompted by a clearly-defined scientific question that cannot be answered by laboratory experiments. The studies summarised in sections VI-A and VI-B both address the dynamics of internal cell interactions that are not directly observable. Here, the domain model and the purpose of simulation is easy to establish – though there is still a lot of mutual transfer of expertise.

In other cases, the fact of needing to agree on a domain model and purpose become part of the scientific research. This is well illustrated by the EAE autoimmune study summarised in section VI-C. Here the development of the simulation has been, itself, a part of the research, to understand the internal processes of the natural system. A lot of modelling took place before it was clear which specific hypotheses could be addressed by agent-based simulation. The scientist-developer relationship here is similar to the relationship in a research laboratory that allows scientists working together to build on and use each others' results without each researcher having complete knowledge of everyone's work. This has led to laboratory-style practices such as calibration and sensitivity analysis being applied to the simulation, to help demonstrate the validity of the computer model.

C. The value of the act of finding out

In most of our studies, the scientist would refer the developer to standard texts, research papers and graduate students, for information about a domain. It is part of the trust relationship that the developers are able to query all this material (and the scientists' models) – to ask for explanations in the context of the domain model, and to check the meanings that are inherent to the science but not obvious to the outsider.

Scientists may model a domain in a specific notation, or describe it informally (in text or "sketch" diagrams). If a specific notation is used, then the developers need to probe to ensure that they have sufficient understanding of the semantics of the notation, as well as the concepts that are expressed in the models. Where the domain is described in sketches, however, the developers need to understand the informal role of the sketch; they must not assume hidden semantics, but should probe to find the "formal" details behind the sketch.

These "finding out" activities need to be recorded. Some examples of hidden assumptions that were elicited by the developers' probing are given in [4], [5]. The developers also kept notes of the sources that were used as background or in answer to questions. Scientists may have to revisit background material, or even their own results, to be able to answer the developers' naive questions – and, in some cases, had to initiate new research to check a previously-assumed fact. Eliciting and recording information in a collaboration is an important activity, not least because it opens the eyes of the stakeholders to the problems inherent in each other's fields. To a developer, it is illuminating that a scientist may not be able to give dimensions, flow rates, etc to within an order of magnitude; to a scientist, the inherent scientific assumptions made in interpreting observations are illuminated by the developer's need for concrete dimensions and facts from which to construct the simulation.

There is currently little technical support for "finding out activities", other than the traditional lab-book. However, as noted above, the elicitation of assumptions is part of engineering practice in safety- and security-critical systems. In these fields, as in collaborative scientific simulation, the establishing of trust through quality underpins the activities undertaken. The critical systems discipline has techniques, and some tools, that could be adapted to support collaborative scientific simulation.

IV. Modelling for the simulation

Developers typically interpret the domain models into software engineering design models, because they are need to allow the dialogue between stakeholders to continue in a meaningful way.

In general, the scientist is interested in understanding what the simulation does and does not capture; the comprehensibility of the developer's design is of more relevance (to the scientist) than the implementation language. A developer with a software engineering background would be comfortable working at a high level of abstraction and is accustomed to moving between abstraction levels (since this is what happens in specifying, designing and implementing software). In the

researching EAE autoimmunity (section VI-C), Read et al. [29] use adapted UML models, because a range of mutually-understood views is needed to explore and express the complex structures and interactions of EAE, whilst Andrews et al. (section VI-A) find simple state-transition diagrams sufficient to capture a mutually-agreed model of the abstract concepts of a lymphocyte migration [4], [5]. Once design models are created, the simulation exercise could be seen as a conventional software engineering exercise, though with a strong emphasis on validation.

A key feature in the design of the simulation is arriving at *an appropriate level of abstraction*. Most simulations consider one component level and one emergent level. For example, in the lymphocyte migration study, section VI-A, lymphocytes are the components, and capture rates are at the emergent level [4], [5]. Here, considerable discussion was needed to arrive at a mutually-acceptable model; the result abstracts from the bio-chemistry but otherwise maps well to the domain understanding of the scientists. It is essential that the whole team understands what has been omitted through abstraction, in order to understand what the simulation is telling. In general, a complex-system simulation could be made up of many layers: we could have simulated bio-chemical interactions and other concepts at other levels of abstraction. The decision to abstract from these should not be made by either stakeholder alone, since neither is fully aware of the other's field. Thus, for example, a scientist may specialise in lymphocytes, which circulate in blood for part of their lifecycle. If the scientist unilaterally decides to abstract from the blood-chemistry, the developer may not be prompted to ask pertinent questions about lymphocyte dynamics, and important issues such as flow-characteristics or vesicle form may be overlooked. If the developer decides unilaterally to abstract from the blood-chemistry, then the scientist will not be able to interpret the results correctly.

The agreement on an abstraction also focuses attention on the need to simulate both the system that the scientist wishes to study and the environment in which that system exists. Abstraction applies to both, and, if the simulation ultimately does not match the expectations of the scientists, the mismatch could be in either the system (the agents of interest in an agent-based simulation) or the environment. One way that the stakeholders can engage with the issues here is to construct an argument, akin to a safety assurance argument, that addresses the question of adequacy of each aspect of the design. The argument presents the rationale and evidence for adequacy, and is evaluated by the stakeholders. If both stakeholders are happy with the argument and evidence, then development proceeds. If the simulation does not behave as expected, then the adequacy argument is revisited to identify possible flaws in the component or environment designs.

V. Software engineering, implementation and beyond

Software engineering is the field of the developer; it is the expertise in quality software development that is their contribution to the collaboration. Verification and validation are critical engineering activities. Most simulation, at least in support of science, needs a relatively lightweight development approach, but with an emphasis on quality. There is unlikely to be a large development team with related management practices; agile-style implementation is likely to be useful. However, the trust motive means that a good standard of communication, and good recording of assumptions and decisions is as important in the implementation as it is in the modelling stages.

Rather than re-invent software engineering, here we draw attention to some existing practices that have real or potential value in engineering simulations of complex systems.

A. Mapping models to code

Engineering an implementation (as well as later activities relating to maintaining and adapting the implementation) is often simplified if there is a clear semantic mapping between design concepts and implementation concepts. This reduces the possibility of implementing something other than what was agreed with the scientists. If we want to maintain lightweight development processes, we need to record how concepts in the design are represented in the simulation; a more heavyweight approach would rework design models produced in determining abstraction levels into design models appropriate to implementation, with appropriate demonstrations of acceptability.

In simpler simulations, it is relatively easy to select modelling approaches that can be easily understood and map cleanly to code. In the lymphocyte migration study (section VI-A), we show how a scientist's sketch relates to a design diagram, and the abstract design diagram maps to the code-design [4], [5]. The simulation has a simple environment and a limited range of component types, and this level of comparison was appropriate to the purpose of simulation.

In more complex studies, such as the EAE autoimmunity study (section VI-C), the need to communicate through diagrams was the key determinant in the choice of UML-like modelling. This was one (of several) factors in deciding to implement in an object-oriented language. However, the need to record and communicate understanding also led to modifications of the UML semantics, and additional diagramming to capture parallel structures, explore possible hypotheses etc. [30].

B. Recording decisions and assumptions

Although simulation engineering tends to be agile and lightweight, the recording of design decisions and assumptions is essential. Anything that has implications for understanding how the domain model has been rendered by the simulation must be discussed by the stakeholders. Maintaining the dialogue becomes harder as the developer moves to implementation, as the scientist is unlikely to want to know the detail of the coding. This is where the trust built up in design is critical – the scientist must trust the developer not to introduce flights of fancy, and to discuss any essential simplifications (or complications) that become necessary to implement the design. Not least of these is the need to be clear about the

scale of the simulation: if data is handed to the scientist on the basis of a few tens of component agents, but reality has tens of thousands, the scientist needs to know and to understand that emergent behaviours of the natural system may not be reproducible. Deviational analysis (e.g. what-if, HAZOP) from critical systems engineering can be used to challenge assumptions, dimensions, rates, etc. Argumentation techniques can be used to record the rationale behind the simulation – an argument can be constructed that the simulation is a mutually-acceptable executable model of the domain model. Recording rationale as an argument helps to develop trust, but is a relatively tedious activity. However, when the scientists' domain knowledge changes (either from their own work, or from insights developed through the simulation exercise), the rationale for simulation changes. The changes and their consequences can be considered by comparison with the record of the previous rationale.

C. Iterative, incremental development

In our studies, a recurring theme is the limitations of scientific knowledge. Modelling and simulation activities help to clarify scientific understanding, and an incremental development means that there is always a partial simulation (working and verified) on which to develop and test ideas about the domain model.

Part of the trust relationship is that the scientist needs to see where the development is difficult or incomplete, just as the developer needs to be aware of which aspects of the science are uncertain.

In relation to quality issues, agile approaches that promote incremental development mean that interim tests of the simulation are encouraged – checks that particular abstractions or simplifications are acceptable in practice as well as in design; investigations of computational feasibility of agreed solutions, etc.

D. Maintainability

Best-practice in software engineering leads naturally to maintainability, through principles such as modularity, in-code documentation standards, version control, and keeping records up to date. There are two principle motivations for maintainability. The first is the maintenance of trust: just as the scientist contracted to answer naive questions from the developer, the developer contracts to answer probing questions from the scientist about what is going on in the simulation. The second motive is that collaborative scientific simulation is not a closed process: having a simulator constructed for an initial purpose inevitably prompts new in silico experiments that can be addressed by extending or modifying the initial implementation. This is one of the big pay-backs of agent-based simulation for science.

VI. Examples of engineered simulations

The CoSMoS project is working on principled approaches to modelling and simulating complex systems. A number of simulations have been constructed, either as part of the project or in collaboration with the project, to assist scientific investigations. These are summarised from an engineering perspective, as exemplars (in part at least) of the approaches considered here.

A. Lymphocyte migration [4], [5]

The CoSMoS lymphocyte study investigated dynamic process related to migration of lymphocytes (types of white blood cells that are key agents in immune responses) from blood circulation to lymph nodes. During immune responses, blood vessels dilate, but how dilation affects the rate of lymphocyte migration is not known. Simulation was used because it was infeasible to monitor dynamic processes inside living animals using traditional laboratory techniques.

1) Domain and domain modelling: The domain is research on lymphocyte migration at the York Centre for Immunology and Infection, with collaborators Mark Coles and Lisa Scott. The domain model was derived from research articles, data on population, rate and timing from experimental work, the scientists' interpretations of biological models at various levels of abstraction, and insight into the (lack of) precision of published and observed data. Much of the information was summarised in sketches of the biological processes, at various levels of abstraction. A key step in the development was agreeing an appropriate level of abstraction for the domain model, based on the domain information.

2) Modelling for the simulation: From the biological sketches of the domain model, a simple state chart was used to model the stages in the transition of a lymphocyte from the blood circulation to the lymph node. The state transition was driven by rates from laboratory experiments and published research.

3) Implementation: The implementation is in occam-π, a powerful process-oriented programming language that supports visualisation and distribution. The design model was close to the simulation model, with clear mappings from lymphocyte states to occam-π processes. The implementation re-used patterns from existing (well-engineered and verified) occam-π simulations (see [25], [31], [32]).

4) Beyond implementation: The simulation provided numerical data output, which was the primary requirement of the scientists. In addition, the occam-π patterns provided a visualisation, which proved of interest to the scientists, because it revealed subtleties of behaviour that might be relevant to their interpretation of observational data.

The visual simulator raised issues about the applicability of the rates provided by the scientists. The rates are derived from one-dimensional observations – for example estimates of the proportion of lymphocytes in each state. In the visual simulation, space is explicit: a lymphocyte's chance of changing state is partly determined by its adjacency to the blood vessel wall. The connotations of this were discussed between the scientists and developers, and affect the treatment of the results.

5) Engineering issues: A range of issues relating to the environment arose. For instance, laboratory-based immunology takes the circulatory system as a given, but the simulation

needs a model of the environment. Much discussion was involved in determining appropriate dimensions for blood vessels and lymphocytes, concerns over accuracy, and whether blood-flow characteristics might influence migration and transition rates. In many, cases, the simulation required arbitrary decisions, which were all recorded as possible limiting factors in the applicability of results.

B. Granuloma formation in Leishmaniasis [33]

A granuloma is a structure made up of various types of immune cell, that forms around certain parasite-infected cells. This study, undertaken as a masters project under CoSMoS supervision, addressed the question of why some granulomas form faster than others.

1) Domain and Domain Modelling: As in the lymphocyte migration study, the domain is well-defined; it is the research of John Moore and Paul Kaye from the York Centre for Immunology and Infection. The domain model draws on the expertise of immunologists and relevant scientific papers. Data was available for various rates of cell movement, size and interaction, though there was no direct data relating to the whole granuloma formation.

The simulation purpose was to test two hypotheses of granuloma formation: (a) a positive feedback system, or (b) so-called more-efficient cell signalling.

2) Modelling for the simulation: Modelling uses state charts and activity diagrams to represent the agents identified in granuloma formation and their interactions. Sequence diagrams were used to abstract models of the two hypotheses.

3) Implementation: The simulation was implemented in Java, and employed the MASON simulation framework [34], [35]. Mason provides a useful graphical interface that made it easy to visualise the formation of the granuloma structures.

4) Beyond Implementation: In this domain, the data rates are not robust or complete enough to support data generation. However, the visualisation gave useful qualitative insights, that were taken forward into targeted laboratory experiments.

C. Autoimmune modelling [29], [30]

EAE is an autoimmune disease, similar to multiple sclerosis, that attacks the central nervous system in mice [36]. Some mice exhibit spontaneous recovery following induction of autoimmunity, which motivates this detailed study. In the disease, many types of immune cell interact across a range of body compartments. Conflicting feedback mechanisms are known to influence disease progression. Simulation offers a route through which hypotheses about the nature of the disease can be investigated in the context of established knowledge.

1) Domain and domain modelling: The domain expert is immunologist Vipin Kumar, and his laboratory at the Torrey Pines Institute for Molecular Studies, San Diego, USA. Much of the domain model was derived from text books and immunological theory, all of which was discussed with, and approved for incorporation by, Dr Kumar.

2) Modelling for the simulation: UML-style diagrams were used to express various aspects of system behaviour: activity diagrams conveyed how emergent behaviours (such as autoimmunity and recovery) were hypothesised to manifest from the interactions of cell types at an abstract level; class diagrams depicted static relationships, each diagram focusing on a single emergent behaviour, to limit diagram complexity; state charts captured low-level dynamics of the various cell types involved. The state charts formed the specification for agent behaviours.

3) Implementation: Java and MASON [34], [35] provide much of the simulation infrastructure. The powerful visualisation framework is useful in observing the simulated progression of the disease. The implementation is portable, and there is a clear mapping from the design diagrams to facilitate mutual understanding of the simulation – an important consideration when collaboration takes place across oceans.

4) Beyond implementation: The act of creating the simulation has raised many scientific questions. Modelling the disease as an agent-based simulation requires a comprehensive low-level specification of each cell type's dynamics. Laboratory techniques such as observing system-wide effects of unnatural perturbations provide data at the level of cell populations, but it took significant research to extrapolate to details of cell-type dynamics. Simulation has allowed investigation of a system from the bottom up, integrating existing knowledge in an effort to recreate observed emergent phenomena. Where data is not available, hypotheses can be formed and tested within the context of what is already known. The simulation is never "complete"; it is a dynamic tool that is continually modified to explore different aspects of the domain model, and to remain consistent with the developing scientific understanding.

5) Engineering issues: The simulation behaviour is composed at two levels: the behavioural dynamics encoded as cell states and actions; and parameter values, such as the duration of a cell state, that fine-tune the behavioural dynamics. A milestone in the simulation was the identification of a "baseline", a representation of EAE that Dr Kumar accepts to be a faithful representation of the disease *in vivo*. Significant validation, including calibration and systematic sensitivity analysis has been applied to the milestone simulation.

The challenge of such a simulation exercise should not be underestimated. There is no complete scientific model of the disease in terms of the cell population dynamics. The domain model has many imprecise elements, essentially hypotheses about the relative values of parameters and populations. The construction is necessarily iterative, with tuning at each iteration. If (when) the simulation's behaviour does not match the observational evidence, the fault may lie in any or all of: parameter values; inappropriate abstraction of behavioural dynamics from observed *in vivo* behaviours; incorrect or incomplete facts from the domain. The stakeholders engage in careful deliberation of strategies that might reconcile the simulation and domain understanding, and arrive at a mutually agreed next iteration. Decisions must be recorded: the simulation is always an abstract representation of the real-world

system, and its results must be appreciated in that context.

D. Trait-based plant ecology [3]

The CoSMoS plant ecology study is based on research at the SIMBIOS Centre, Dundee; the domain expert, James Bown, co-supervises a doctoral study to replicate and extend an original model. The original research used "traits" to introduce individual variation into traditional population-based modelling of plant ecology. The SIMBIOS team had in-house simulations, but these do not scale to support further research.

Unlike our other studies, the domain model here exploits the original simulation. The study uses critical-systems argumentation techniques to evaluate the acceptability of the new simulation relative to the existing one and to the underlying science.

1) Domain, domain modelling, and modelling for the simulation: The domain is the SIMBIOS research on trait-based modelling of plant communities. The new simulation study started by reverse-engineering a domain model from the original simulation, represented by the C/C++ code, and related literature (e.g. [37]). The stakeholders collaborated to separate the scientific underlying model from the implementation-specific design decisions, but it has not always been possible to separate out the science from the design of the original simulation. The main modelling task must translate class diagrams derived from the original C/C++ code into a model of the occam-π code structures for concurrent processes and synchronisation [3].

2) Implementation: The new simulation was implemented in occam-π, which supports distribution and explicit, light-weight concurrency. As in the lymphocyte migration study, occam-π implementation patterns facilitated aspects of the simulation, such as visualisation and spatial layout [31].

3) Beyond implementation: Converting from a sequential simulation to a process-oriented concurrent simulation raised some interesting general issues. Process-oriented modelling supported a naturalistic model of plants and their interaction with the resource environment – plants were represented as individual processes with separate threads of control, and interacted directly with environmental processes. However, the model of concurrency sometimes requires an unnatural dialogue: a seed falling from a plant (a new process) must ask a location (an environmental process) if it is attached to a plant process; if the location has a plant, then the seed decides to die (its process terminates).

In the SIMBIOS research, there is an inherent link between the domain (the science) and infrastructural features: for example, ecological research often assumes a two-dimensional grid model as the basis for observation and modelling [38]. This raises issues, since we separate the domain (the scientific scope) and the domain model (the view of the science that is the starting point for the simulation exercise). In principle, the implementation should be free to adopt whatever simulation infrastructure is considered appropriate, but here, the science dictates part of that infrastructure.

4) Engineering issues: Working from a simulation in one language to a new simulation in another does not remove the need for the developer to acquire domain knowledge; the level of understanding needed to create the new simulation still involved the collaborative dialogues seen in other studies. Indeed, an additional dialogue is needed to uncover the motivation for features of the original design, and to determine if these are necessary features of the new design.

Because this study seeks to extend the original research, the simulation development included considerable investment in identification of assumptions and abstractions that were made in the original simulation, and in recording of design assumption in the new simulation. As in other studies, this improves the ability to validate the scientific basis of the model, and to show where the simulation diverges from reality. The study devised a strategy for maintaining "adequate equivalence" between simulations, agreed between stakeholders [3], presented as an agreed argument (in the Goal Structuring Notation [39]). The argument construction technique has proved useful in developing trust and understanding, and in supporting the extension to the scope and scale of the new simulation.

E. Auxin transport canalisation [40], [41]

Auxin is a plant growth hormone. In a plant stem, auxin transport canalisation is a process that occurs between auxin production sites and auxin sinks, and results in vascular tissue development. This is a complex self-organising process, regulated in the cells of the plant. The question of interest here is the role of PIN proteins within cells in regulating the direction of auxin transport (the canalisation process).

1) Domain and domain modelling: This doctoral study applies the modelling and simulation ideas being developed in the CoSMoS project to development of a research simulation. The study is part of auxin research in the Biology Department at York, and is co-supervised by the domain expert, Ottoline Leyser. The focus of the simulation is the process of auxin canalisation in a multi-cellular slice of plant stem. The input of the domain expert is supplemented in the domain model by research papers. Discussion focused on gaps in existing knowledge and hypotheses of PIN placement within the cell. In modelling, PIN placement must use biologically plausible local information, yet result in a global polarisation in the direction of the emergent canal.

2) Modelling for the simulation: There were two phases of modelling in this study. Initially, a UML-style class diagram captured the important entities in the canalisation process, including the emergent canal. State charts capture the auxin transport process and express the different hypotheses for the PIN placement process.

Subsequently, a "software simulation model" was derived. This adds detail relating to simulation (for example, how space is modelled, and user interface issues) and removes features that are not directly implemented; for example, the auxin transport canal no longer appears as a class, as it is a required emergent property.

3) Implementation: The implementation is in Java. The software follows the simulation model, with features to support the agent-based style of specification, and to support visualisation. One important feature is the use of 2D cells, with parameters modified to correspond to the 3D reality.

4) Beyond implementation and Engineering issues: Key issues in this study was the number of system parameters (concentrations, rates, etc), and, again, the fact that, in many cases, the biological values of these parameters are not well known. The research applied meta-heuristic (evolutionary) search over parameter values to find those that gave the best simulation of canalisation. The identified values were then checked for biological plausibility. This illustrates how the simulation process becomes part of the scientific research. For example, in simulation, the canalisation needed PIN proteins to stay in place on the cell membrane for a longer time had been suggested, before disassociating and moving to a new site; subsequent experimental data has shown that this is biologically correct.

The simulation is slow, with tens of cells and thousands of agents taking several hours to run. This impacts most on the meta-heuristic search techniques.

VII. Discussion

This paper set out to review state of the art engineering techniques that can assist credible scientific simulation, but much of the discussion is about development of trust. It is pertinent that the critical systems techniques that we have investigated to support modelling, analysis and validation of complex systems simulations are also related to the development of trust. For instance, safety critical systems engineering must produce certifiable products; certification is dependent on convincing expert reviewers that the system is engineered and constructed to be safe. To do this, assumptions must be stated, mutual understanding established, and, ultimately, certifiers must trust the argument and evidence presented by the developers.

Collaborative simulation development is a form of interdisciplinary research, and, like other such areas, there is an investment to be made, which must be balanced against the potential pay-back for both stakeholders. It is unlikely that a good collaboration would develop if the scientist could see a cost-effective conventional scientific way to address the questions posed by their domain. In many of our studies, collaboration has started cautiously, as a side-line from the main scientific research. On the developers' side, significant effort has gone into developing adequate domain knowledge and scientifically-robust recording and reporting. Techniques of analysis and argumentation from critical systems engineering have proved beneficial here, and help to attain a scientific level of precision in the presentation of engineering products.

There is an obvious question about the wider feasibility of the resources required to support collaborative simulation in scientific research. This paper reviews scientific simulation collaborations that range from small, focused studies that address specific questions, to significant research contributions. It is feasible to undertake the small-scale focused studies as one-off exercises (note that one of those above is a 6-month masters project). Larger studies involve doctoral or post-doctoral researchers on both sides. In these cases, once the initial commitment is made, the studies develop their own momentum – initial modelling and simulation iterations overcome the steepest learning curves on both sides, and the marginal returns on extending simulation to related hypotheses rise steeply as the collaboration develops.

Finally, to make explicit a point that is implicit throughout the previous discussions, collaborative scientific simulations do not produce simple answers. Complex systems are layered amalgamations of interacting components and systems; they are hard to understand and hard to model. Systems-based approaches to science are now widely practiced, and well-engineered, scientifically-acceptable simulations have a significant potential role, so long as scientists and developers understand the limitations inherent in these tools.

We have not reviewed the considerable research literature on trust – mostly from the security domain; it is possible that such a review would help to understand how and why the activities that we describe produce the necessary trusting collaboration.

As noted above, the CoSMoS project is working on principled approaches to simulation and modelling of complex systems. We are constructing guidelines for development, focusing on the key stages of modelling and validation: the studies reviewed here are helping to derive these guidelines, and new studies are using the guidelines.

To support practical collaborative scientific simulation, tool support for modelling, validation, and argumentation also needs to be considered. Support is needed for identification and analysis of assumptions and design decisions. State of the art modelling and model management can be used to construct integrated tool suites, but these need to respect the overarching concerns of collaboration: communication among people from different disciplines.

VIII. Conclusion

This paper presents a range of issues that inhibit and promote the use of agent-based simulations in scientific study of complex systems. We summarise some work on simulation engineering from conventional and agent-based simulation, and critical systems engineering work that seeks to contain, rather than exploit, the emergent effects of systems of systems. We highlight key engineering issues such as validation, recording of design decisions, and identification of assumptions.

From engineering concerns, the paper moves to an outline of key stages in the collaborative development of scientific simulations, discussing appropriate techniques, as well as the ways in which mutual trust is maintained. Five studies are summarised, to show how the issues identified relate to actual scientific simulations.

Acknowledgment

This work is part of the CoSMoS project, EPSRC grants EP/E053505/1 and EP/E049419/1, www.cosmos-research.org.

The authors would like to acknowledge the key role of all our scientific collaborators, many of whom are identified in the text of the paper.

REFERENCES

[1] M. Calder, S. Gilmore, and J. Hillston, "Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA," *Transactions on Computational Systems Biology VII*, vol. 4230, pp. 1–23, 2006.

[2] A. Sadot, J. Fisher, D. Barak, Y. Admanit, M. J. Stern, E. J. A. Hubbard, and D. Harel, "Towards verified biological models," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2007.

[3] T. Ghetiu, R. D. Alexander, P. S. Andrews, F. A. C. Polack, and J. Bown, "Equivalence arguments for complex systems simulations - a case-study," in *Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2009, pp. 101–140.

[4] P. S. Andrews, F. Polack, A. T. Sampson, J. Timmis, L. Scott, and M. Coles, "Simulating biology: towards understanding what the simulation shows," in *Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2008, pp. 93–123.

[5] F. A. C. Polack, P. S. Andrews, and A. T. Sampson, "The engineering of concurrent simulations of complex systems," in *Congress on Evolutionary Computation*. IEEE Press, 2009, pp. 217–224.

[6] G. F. Miller, "Artificial life as theoretical biology: How to do real science with computer simulation," School of Cognitive and Computing Sciences, University of Sussex, Tech. Rep. Cognitive Science Research Paper 378, 1995.

[7] E. D. Paolo, J. Noble, and S. Bullock, "Simulation models as opaque thought experiments," in *Articial Life VII*. MIT Press, 2000, pp. 497–506.

[8] M. Wheeler, S. Bullock, E. D. Paolo, J. Noble, M. Bedau, P. Husbands, S. Kirby, and A. Seth, "The view from elsewhere: Perspectives on alife modelling," *Artificial Life*, vol. 8, no. 1, pp. 87–100, 2002.

[9] J. Bryden and J. Noble, "Computational modelling, explicit mathematical treatments, and scientific explanation," in *Artificial Life X*. MIT Press, 2006, pp. 520–526.

[10] D. Bray, "Bacterial chemotaxis and the question of gain," *PNAS*, vol. 99, no. 1, pp. 7–9, 2002.

[11] H. Poincare, *Science and Method*. Dover Publications, 1958, transl. F. Maitland, from the French, 1908.

[12] W. I. B. Beveridge, *The art of scientific investigation*. Heinemann Educational, 1957.

[13] P. B. Medawar, *Induction and Intuition in Scientific Thought*. Methuen, 1969.

[14] J. Uchmanski and V. Grimm, "Individual-based modelling in ecology: what makes the difference?" *Trends in Ecology & Evolution*, vol. 11, no. 10, pp. 437–441, 1996. [Online]. Available: http://dx.doi.org/10.1016/0169-5347(96)20091-6

[15] P. Deuflhard, "From molecular dynamics to conformation dynamics in drug design," in *Trends in Nonlinear Analysis*, M. Kirkilionis, S. Krmker, R. Rannacher, and F. Toni, Eds. Springer, 2003, pp. 269–288.

[16] A. Hone, "On non-standard numerical integration methods for biological oscillators," in *Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2009, pp. 45–66.

[17] J. Sudeikat, L. Braubach, A. Pokahr, and W. Lamersdorf, "Evaluation of agent-oriented software methodologies – examination of the gap between modeling and platform," in *AOSE 2004*, ser. LNCS, vol. 3382. Sringer, 2004, pp. 126–141.

[18] C. A. Iglesias, M. Garijo, and J. C. González, "A survey of agent-oriented methodologies," in *International Workshop on Intelligent Agents, Agent Theories, Architectures, and Languages*, ser. LNAI, vol. 1555. Springer, 2000, pp. 317–330.

[19] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard, "ADELFE: A methodology for adaptive multi-agent systems engineering," in *Engineering Societies in the Agents World*, ser. LNCS, vol. 2577. Springer, 2003, pp. 70–81.

[20] L. Padgham and M. Winikoff, "Prometheus: A methodology for developing intelligent agents," in *AOSE III*, ser. LNCS, vol. 2585. Springer, 2003, pp. 174–185.

[21] R. Alexander, "Using simulation for systems of systems hazard analysis," Ph.D. dissertation, Department of Computer Science, University of York, 2007.

[22] O. Paunovski, G. Eleftherakis, and T. Cowling, "Framework for empirical exploration of emergence using multi-agent simulation," in *Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2008, pp. 1–31.

[23] R. G. Sargent, "The use of graphical models in model validation," in *18th Winter Simulation Conference*. ACM, 1986, pp. 237–241.

[24] ——, "Verification and validation of simulation models," in *37th Winter Simulation Conference*. ACM, 2005, pp. 130–143.

[25] F. Polack, S. Stepney, H. Turner, P. Welch, and F. Barnes, "An architecture for modelling emergence in CA-like systems," in *European Conference: Advances in Artificial Life*, ser. LNAI, vol. 3630. Springer, 2005, pp. 433–442.

[26] F. A. C. Polack, T. Hoverd, A. T. Sampson, S. Stepney, and J. Timmis, "Complex systems models: Engineering simulations," in *ALife XI*. MIT press, 2008.

[27] B. P. Zeigler, "A theory-based conceptual terminology for M&S VV&A," Arizona Center for Integrative Modeling and Simulation, Tech. Rep. 99S-SIW-064, 1999, www.acims.arizona.edu/PUBLICATIONS/publications.shtml.

[28] H. Stanislaw, "Tests of computer simulation validity: what do they measure?" *Simulation and Gaming*, vol. 17, no. 2, pp. 173–191, 1986.

[29] M. Read, P. S. Andrews, J. Timmis, and V. Kumar, "Using UML to model EAE and its regulatory network," in *International Conference on Artificial Immune Systems*, ser. LNCS, vol. 5666. Springer, 2009.

[30] ——, "A domain model of Experimental Autoimmune Encephalomyelitis," in *Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2009, pp. 9–44.

[31] P. S. Andrews, A. T. Sampson, J. M. Bjorndalen, S. Stepney, J. Timmis, D. N. Warren, and P. H. Welch, "Investigating patterns for the process-oriented modelling and simulation of space in complex systems," in *Artificial Life XI*. MIT Press, 2008.

[32] A. T. Sampson, J. M. Bjørndalen, and P. S. Andrews, "Birds on the Wall: Distributing a process-oriented simulation," in *Congress on Evolutionary Computation*. IEEE Press, 2009, pp. 225–231.

[33] A. J. Flügge, J. Timmis, P. Andrews, J. Moore, and P. Kaye, "Modelling and simulation of granuloma formation in visceral Leishmaniasis," in *Congress on Evolutionary Computation*. IEEE Press, 2009, pp. 3052–3059.

[34] G. C. Balan, C. Cioffi-Revilla, S. Luke, L. Panait, and S. Paus, "MASON: A Java multi-agent simulation library," in *Agent 2003 Conference*. Argonne National Laboratory, 2003. [Online]. Available: http://agent2002.anl.gov/Agent2003.pdf

[35] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan, "MASON: A new multi-agent simulation toolkit," in *Swarmfest Workshop*, 2004. [Online]. Available: http://cs.gmu.edu/ eclab/projects/mason/publications/SwarmFest04.pdf

[36] V. Kumar, J. Maglione, J. Thatte, B. Pederson, E. Sercarz, , and E. S. Ward, "Induction of a type 1 regulatory CD4 T cell response following Vβ8.2 DNA vaccination results in immune deviation and protection from experimental autoimmune encephalomyelitis," *International Immunology*, vol. 13, no. 6, pp. 835–841, 2001.

[37] J. Bown, E.Pachepsky, A. Eberst, U. Bausenwein, P. Millard, G. Squire, and J. Crawford, "Consequences of intraspecific variation for the structure and function of ecological communities: Part 1. model development and predicted patterns of diversity," *Ecological Modelling*, vol. 207, no. 2-4, pp. 264–276, October 2007.

[38] U. Berger, C. Piou, K. Schiffers, and V. Grimm, "Competition among plants: Concepts, individual-based modelling approaches, and a proposal for a future research strategy," *Perspectives in Plant Ecology, Evolution and Systematics*, vol. In Press, Corrected Proof, 2008.

[39] T. P. Kelly, "Arguing safety – a systematic approach to managing safety cases," Ph.D. dissertation, Department of Computer Science, University of York, 1999, yCST 99/05.

[40] P. Garnett, S. Stepney, and O. Leyser, "Integrative hybrid modelling of plant shoot branching," in *Workshop on Functional Structural Plant Models*, 2007, pp. 9.1–9.5. [Online]. Available: http://www-users.cs.york.ac.uk/˜susan/bib/ss/nonstd/fpsm07.pdf

[41] ——, "Towards an executable model of auxin transport canalisation," in *Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2008, pp. 63–91.