Motivating *Interactive Self-Organisation*

Sebastian von Mammen

Julius-Maximilians University, Würzburg
sebastian.von.mammen@uni-wuerzburg.de

Abstract

Understanding, engineering and controlling self-organising systems, i.e. of large numbers of interwoven autonomous agents, pose numerous scientific and practical challenges. A considerable body of works aims at theoretical approaches as well as empirically identified solutions to address them—with a focus on interaction among the agents and resulting emergent, global effects. However, only few works systematically consider the means of interaction of these systems with the expert user. In this paper, we address this perspective by introducing the notion of *interactive self-organisation*. To this end, we step through the development life cycle of a self-organising system, we emphasise the need for accessible software solutions for modelling and simulation, we present a concrete application scenario, and highlight the great challenges on this path.

Swarms and Interactive Self-Organisation

Herds of social animals, schools of fish or insect colonies host large populations of individuals[1]. Their strength lies in accomplishments that no single individual could achieve by itself. At the same time, this ability of the whole, sometimes referred to as *swarm intelligence*, does not necessarily require complex agent behaviours equipped with deep intelligence(s). Rather, simple behaviours of reactive agents suffice to yield helpful, system-wide emergent phenomena including mass transport (Marras et al., 2015), foraging (Czaczkes et al., 2015), defence (Parmentier et al., 2015) and adaptive nest construction (Fouquet et al., 2014). As a consequence, swarms have become a metaphor for self-organising systems, i.e. systems without central control but autonomous agents acting locally, and thus, contributing to emergent effects. Looking a bit closer, this metaphor brings along several perspectives. For once, it brings together seemingly opposing perspectives such as the consideration of homogeneous and diverse subpopulations of agents as well as the connection of local interaction and global effect. As a result, it also hints at the notion of hierarchies of abstraction. And at the same time, the term swarm triggers a strong visual association—a spatial, observable, and therefore, accessible outcome of the dynamics of complex systems. Hence, swarms are not only a metaphor for self-organising systems but they render it obvious as to why interactivity is one of their seminal aspects. When considering interactivity in the context of self-organising systems, the following definition may help to cover the various stages of a self-organising system's life cycle and opportunities to interface with different humans involved in the process.

Definition. *Interactive self-organisation describes the effort to making large, self-organising technical systems transparent, malleable and controllable by human designers, decision makers and users (von Mammen, 2016a).*

Development Life Cycle

Typically, the first step into developing a concrete self-organising system is the design of a domain model. Next, this domain model is translated to a platform model, which, in turn, is implemented for a specific simulation environment to yield the results sought after (Andrews et al., 2010; Polack, 2010). In the first steps, interactivity is rarely provided as the processes of modelling and implementation are still largely manually driven. However, there are examples of interactive exploration of the model spaces as part of the simulation (Ritter et al., 2011). This may lead to insights about the model specifics or further an iterative process of model refinements. We want to stress that each step in the development life cycle offers an opportunity for interactivity, and that in self-organising systems in particular, challenges arise when introducing means of interactions.

Performance is one of these challenges. It costs additional computational power to provide interactivity, especially when aiming at the short latency intervals that are expected in interactive real-time systems (Gregory, 2009). And the computational load of computing large numbers of interacting agents poses a challenge in general (Parry and Bithell, 2012), and enriching the inherently large state space of the model by granting freedom to navigate and manipulate the simulation makes it worse. It further costs a lot

[1]The rationale presented in this paper was first published in the author's habilitation, von Mammen (2016a).

of effort to conceptualise and implement user interfaces that can support each of the phases of the development life cycle (*discovery, development, exploration*, see Figure 1) for model definition, specification, testing, refinement and analysis. need to be made accessible to the developer/user as he has to devise, implement, test and refine models. User interfaces work are well-received, if they do not require the user to climb steep learning curves or to invest great cognitive efforts (Foley et al., 1984; Preim and Dachselt, 2015). Designing an according language of interactions that link the self-organising system's details' selection, its control, its navigation to the users' input queries and feedback signals takes time.

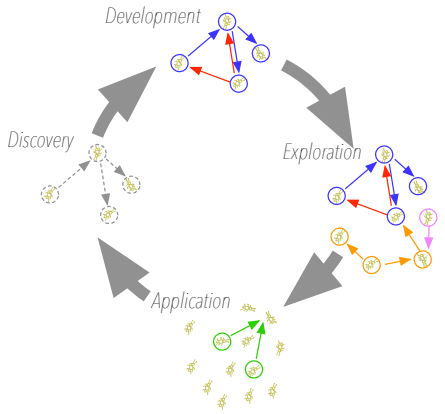

Figure 1: The three development phases *discovery, development, exploration* of the development life cycle of systems of interactive self-organisation, as well as the *application* phase (figure taken from von Mammen (2016a)).

Figure 1 shows the aforementioned phases of the development life cycle of an interactive self-organising system as well as the application stage. The system itself is illustrated by several ants that represent agents and relationships between them are hinted at by pairwise connecting arrows. In the discovery phase, the modeller of an artificial self-organising system or the empiricist observing a natural self-organising system retrace the system's and agents' states and their interrelations (grey-dashed arrows in the discovery phase, Fig. 1). In the development phase, these observations are cast into a model that features all the identified properties, relationships and behaviours (blue arrows in the figure). Considering real-world processes, this translation pre-selects, simplifies and discretises. On the other hand, relationships might be part of the computational model that could not be directly observed in natural systems (red arrows in the figure). In the exploration phase, aspects new to the modeller/developer may be revealed that were not explicitly modelled but emerge during the simulation. This is indicated by the extension of the illustration by means of orange and pink arrows. Finally, when transferring a self-organising system to a real-world application, its user(s) can influence it based on the insights gained from the preceding modelling & simulation efforts. Example domains are sensor and computing networks (Weiser, 1993; Satyanarayanan, 2001), small-scale robots for medical procedures (Sitti et al., 2015), or scouting quadcopter squadrons (Salmon and Meissner, 2015).

Individualised, Agent-based and Interactive

In general, interactive self-organisation aims at methods that support $n : m$-relationships, of n users and m controlled objects or agents, whereas $m >> n$. That is few users, possibly even only one, need to supervise and interact with a potentially very large number of agents. As briefly hinted at above, it poses a veritable challenge to create according, accessible, interactive modes of modelling, simulation, and analysis. The analysis may also bring about new insights about a crafted model of self-organising systems that suggests the application of optimisation either to identify parameter sets that promise the best viable solutions of engineered systems or that best retrace natural phenomena. In this section, we pursue this insight a step further, by introducing the idea of individualised simulation, revisiting the need for agent-based modelling, and highlighting the coming about of interactive simulation.

Individualising Simulation

Arbitrary development phases of (software) engineering tasks can benefit from simulation (Banks et al., 1998), as it can validate a system's design early on, its targeted (emergent) functionality (Jakobi et al., 1995), as well as perturbations at runtime (Tomforde, 2012). In computer science, simulation has been an important driver from day one. However, private users have mostly only consumed simulation results (e.g. weather broadcast and traffic predictions) and been exposed to modelling and interactive simulation in the context of computer games, frequently simulating flying airplanes and driving cars (Williams, 2006; Backlund et al., 2008). Yet, making simulation technology accessible to a broader public bears great transformative potential, especially also in the context of the increasing level of digitization of our built and technological environment (Harper, 2003). Considering recent trends such as 3D printing that propelled the wide-spread growth of the maker scene (Lipson and Kurman, 2013) as well as the digitization of medical records and the prospect of individualised medical treatments (Topol, 2014) unfold a wide space for simulation-driven transformation. With ease of access to simulation technology and clear, high impact benefits, a wide-spread uptake becomes more feasible. In order to get there, supporting the development phases outlined above (Fig. 1), we consider the following aspects crucial for individualising simulation:

1. Provision and (semi-)automated matching of template situations to fit the user's challenges (von Mammen et al., 2019).

2. A "natural" interface for parametric adjustments, introducing additional programming code where necessary, navigating a simulation, as well as harnessing optimisation methods to improve one's basis for decision making.

Consider, for example, that the user wants to gain insight in his/her energy footprint (Nguyen and Aiello, 2013) and to minimise it. There are several standard factors such as heating, nutrition, clothing, and traveling that impact this footprint, can be easily iterated, adjusted and extended. Natural means in this context that the user interface works as expected without the need to think about it—by building on top of established user interaction modalities and gestures, such as swiping on touch devices (Foley et al., 1984; Preim and Dachselt, 2015). Since adding programming code requires the user to invest more cognitive efforts, following an agent-based modelling paradigm can help, especially in combination with visual programming interfaces, which we discuss in more detail in the next paragraph.

Modelling, Bottom-Up, Agent-Based

Software solutions such as Mathematica (Wolfram, 2003), Matlab (Redfern and Campbell, 2012), GNU Octave (Eaton, 2006), Sage (The Sage Development Team, 2015), Magma (Computational Algebra Group, 2015) or Maple (Redfern, 2012) allow the user to define, solve, analyse and optimse mathematical models. This assumes, however, a deep understanding and mastery of mathematics and a considerable amount of time. For specific application scenarios, specialised user interfaces, data import and export pipelines and re-usable models can be plugged into the respective software kernels, for instance plugins for EEG analysis (Dimigen and Reinacher, 2012) or the display of volumetric data (Mikulka, 2014). Further adjustments in the code bases would again require expert knowledge in modelling and programming (Li et al., 2013).

Mathematical models also assume that the modeller is well-informed about the involved parameters that describe a system and how they are intertwined. Bottom-up modelling approaches (Tesfatsion, 2006; von Mammen and Jacob, 2009; Epstein and Axtell, 1996) do not make this assumption. Rather, they describe how in physical contexts individual elements (Brenner and Carstensen, 2004) or particles (Müller et al., 2003; Hosseini and Feng, 2009), or more generically, how numerous agents' states and interactions can be described (Wooldridge, 2008). Among the many definitions of the latter, the following one by Denzinger and Kordt (2000); Denzinger and Winder (2005) underlines its generality and still enforces a clear structure of data flow in an agent-based model: Consider an agent Ag be expressed by a quadruple (Sit, Dat, Act, f), whereas Sit is the set of possible situations, Dat the set of possible (internal) data states, Act the set of possible actions and $f = Sit \times Dat \rightarrow Act$, the agent's decision function that informs it what to do based on which situation and information. This definition also suggests that due to the generic characterisation of agent-based models, individual agents can become rather complex model units. Hence, there are different classifications that try to narrow down the deployed agents' abilities considering for example reactive, reflective, or knowledge-based agents. Beyond this classification of basic model agents, the notion of super agents that recursively subsume other agents has repeatedly been considered to flexibly build up (and dissolve) complex agent models (Parry and Bithell, 2012). This perspective highlights the inherent modularity and flexibility of agent-based models. In combination with simple transfer of the target system's components' properties and descriptions to model agents, agent-based models have been rather popular in diverse scientific communities including economics, social sciences, and the life sciences.

Analogous to the mathematics frameworks mentioned above, the agent-based modelling approach is also supported by a number of dedicated software systems, for a current overview see Kravari and Bassiliades (2015); Abar et al. (2017). As agent-based models often feature spatial relations, basically all of them provide a visualisation environment for showing the model and its evolution during the simulation runtime. Just to name a few examples: NetLogo is a popular environment which makes modelling accessible by means of a simple scripting language and a 2D lattice visualisation (Tisue and Wilensky, 2004). RePast further benefits the modeller, for instance, by flowchart and state chart visualisations and ports to different programming languages for compatibility and performance reasons (North et al., 2013). Relationships between the agents stand in the focus of the Swarm simulation environment, which also allows for the aforementioned hierarchical organisation of agents (Minar et al., 1996).

Interactive Simulation & Self-Organisation

The history of interactive simulation dates at least back to the 1960s—it can, for instance, be retraced by studying the works by Jones (1967); Bell and O'keefe (1987); Rothrock and Narayanan (2011). Compared to traditional modelling & simulation approaches, interactive simulations allow the user to model during runtime, to introduce changes and see immediate effects. One might be tempted to consider this a small technical improvement but it introduces fundamental changes to simulation and how it can be used. For instance, consider simulation data to support a lively discourse among decision makers, simulation of emergency situations for training (Bucher et al., 2019), or simulation results to support clinical personell performing invasive operations (von Mammen et al., 2015). By providing the data

at interactive speed and in understandable formats, interactive simulations allow us to synchronise computation results with the human's sensorimotor and cognitive systems, to enrich the user's perspective, to provide feedback and to also retrieve further data from the user to feed into the simulation model. The consequences are multi-facetted: Systems can be better understood, unaccounted aspects learned, inconsistencies determined, communication improved. In short, interactive simulations put the user in charge of the computation and, thereby, achieve a far greater appeal and qualify for far more cases of using simulation data than merely presenting accomplished facts to the user.

When considering self-organising systems at the centre of one's modelling and simulation efforts, interactivity becomes all the more crucial to cope with phase transitions in technical systems and to generally master the complex dynamics that may arise in real-world contexts.

Wetlab Application Scenario

The following description of an ideal application scenario of interactive self-organisation is directly taken from von Mammen (2016a): In an ideal, unbound interactive self-organisation scenario, the user could quickly prototype a comprehensive simulation model, fleshing out spatial details and behaviours of hundreds of thousands of involved agents. Next, he would be given the opportunity to witness the emergence of system behaviours such as cyclic process patterns, branching points, or the convergence of the system state into global attractors (Nicolis and Rouvas-Nicolis, 2007). He would also be given the opportunity to automatically repeat and evaluate the simulation within predefined parameter ranges, to extract novel insights by learning hypotheses that maximise the information gain (Schmidt and Lipson, 2009), and to consider any modelling efforts as only a small, sub-model part of a grander, multi-scale system (Magnenat-Thalmann et al., 2014; Nickerson et al., 2014). The user's efforts would be supported by meaningful, rich visualisation techniques (Vaquero et al., 2014; Cebulla et al., 2014) and multi-modal, natural user interaction techniques (Turk, 2014)—in this way, he would be empowered to interact, hone and explore the system model in any desirable ways, posing as little cognitive and motor-sensory challenges as possible (Foley et al., 1984). Novel augmented reality technology including head-mounted devices, eye and finger tracking sensors would bridge the gap between simulated predictions and real-world systems, providing invaluable data for learning, decision making and guiding the user's actions (Dunleavy and Dede, 2014; Azuma et al., 1997).

In the context of biological developmental processes (Slack, 2009; Gilbert, 2013), for instance, the ideal simulation workbench would allow a modelling entry at the intercellular level, offering the means to model layers of mesenchymal and epithelial tissues, empower individual cells with the capabilities to adhere to each other, to divide, to migrate, to produce and emit morphogens etc. (Salazar-Ciudad et al., 2003). The model would relate these foundational operations to time, to biochemical or biophysical signals such as the diffusion of homeobox gene concentration (Duboule, 1995) or mechanical forces (Théry and Bornens, 2006). Based on such intercellular interactions, morphological processes would emerge, shaping anatomy and physiological infrastructure of developing organisms (Xu et al., 2015). Fast forwarding in time, the obvious effects of morphology-affecting developmental processes would wane, a metabolic equilibrium would establish itself. The model could be extended to provide more facts at different levels of scale (Eissing et al., 2011), for instance by detailing the production pathways of signalling molecules or by introducing materials that define the cell's structural properties (Dror et al., 2012). Similarly, empirically identified emergent properties such as the cell's surface tension, or its adhesion coefficient, could be superimposed, the parameters of the lower modelling levels be automatically adjusted top-down, resulting in a consistent, self-adapting middle-out model (Noble, 2006). At any point in time, disruptions of the developmental processes could be explored, the formation of anomalies could be traced and countered with minimally invasive treatments, without loosing sight of side-effects at all conceivable scales of the organism's definition.

The tandem of in-vitro and in-silico experiments would ensure the validation of each component of the model and the simulation, respectively, resulting in a profoundly accurate model and providing clear perimeters of the experiments' outcomes and the simulations' predictive powers. A sophisticated, accessible and flexible augmented reality interface could mediate between in-vitro and in-silico models, allowing developmental biologists to setup and experiment relying on standard assay procedures. The scientist's activities would be supported and guided by the augmentation of in-vitro experiments and the projection of in-silico simulations, imparting all the benefits of computing technologies, including virtually limitless resources, the possibility to go back and forward in time and to venture into new exploratory directions. Depending on the application domain, the wide-spread adoption of swarm-based modelling and simulation could also lead to far-reaching model improvements that could accelerate overcoming the gap between in-vitro and in-vivo predictions.

Challenges

To realise the ambitious perspectives touched upon in the application scenario outlined in the previous section, the interplay of three major research directions needs to be promoted. The computational core, i.e. the model representation needs to be standardised, novel perspectives need to be found to create natural user interfaces to access, select, manipulate, supervise or even directly control self-organising

systems. And finally, efforts must be made to further scale up the number of simulated agents to create models of self-organising processes that have great relevance for applications and still run at realtime speeds.

Standardising Representations

The principle of re-using well-established, well-researched building blocks underlies many engineering tasks. This principle might also solve one of the issues of standardisation in self-organising systems. Currently, arbitrary algorithmic designs of individual agents, documented in scientific writings, possibly accompanied by the source code, are the default way to modelling self-organising systems, see for instance Klein (2008); Klopfer et al. (2009). Although this freedom is warmly welcome by an expert modeller, it comes with two drawbacks. The first is the need for programming expertise, the second the difficult comparability of a model and its results. Both drawbacks may, in turn, result in limited dissemination and uptake. Said primitives, i.e. building blocks to combine and further configure to arrive at the desired agent behaviours could mitigate the standardisation problem. However, they do not provide an answer to the question yet, how these building blocks may be combined—for instance as conditional rules (Mota et al., 2013), condition-action pairs (Davison and Denzinger, 2012), subject-predicate-object triples (Whalley, 2006), or numeric decision functions (Spector et al., 2005). Finally, numeric data representation and the order and method of integration may als heavily impact complex system simulations (Derényi and Vicsek, 1994). While a general lack of standardisation yields great problems (Müller et al., 2014), following formal protocols quickly becomes unwieldy, even in conceptually rather simple models (Winikoff et al., 2018). We, therefore, consider algorithmic routines of the formal description of agent-based, self-organising systems and their analyses absolutely necessary.

Innovating User Interfaces

In self-organising systems, the agents' states and interaction topologies can change over time. These dynamic properties need to be considered when crafting user interfaces. In particular, this raises questions about the definition or selection of specific subpopulations, creation or tracing of specific relationships, also considering chains or cycles of interdependencies, as well as emerging patterns in the agents' states. An according interaction infrastructure featuring appropriate, helpful, efficient visualisations and required, natural and equally efficient interaction routines needs to be developed to stimulate, recognise, trace, or interfere with phase transitions alongside of emergent effects that may not be directly captured in individual parameter spaces but need to be identified algorithmically (Müller-Schloer et al., 2011). Some of these challenges have already been recognised and several explorations in this direction have been undertaken. It suggests itself to think of applications of swarm robotics when considering real-world self-organising systems. Therefore, according human-swarm interfaces have been explored in the robotics context, see for instance (McLurkin et al., 2006; Naghsh et al., 2008; Pollini et al., 2009; Kolling et al., 2012).

Next to obvious challenges such as the selection and management of large numbers of agents, problems in user interfaces for self-organising systems consider other perspectives as well such as the translation between lower and higher levels of abstraction (Sycara et al., 2015), the inference of the agents' individual behaviours from high-level goals (Tarantola, 2005; Aster et al., 2011), or balancing the agents' degrees of autonomy von Mammen (2016b).

In general, informative (all that needs to be shown), efficient (quickly discernible), attractive (e.g. by means of useful alignments, symmetries, and consistent design decisions throughout) make for successful user interfaces (Steele and Iliinsky, 2010). In order to capture the structure and dynamics of self-organising systems, graph visualisations are apt—representing agents as nodes, possibly hierarchically embedding further nodes as in super agents and representing vertical relationships as edges (Beck et al., 2014). While visualisations are important to convey the desired information and to provide feedback about any user input, effective interfaces have to offer a simple, consistent "language" to communicate the user's goals to the self-organising system or the modelling and simulation environment across different contexts. With a growing degree of input specificity, the interaction sequences need to carry greater information gain by necessity. This is especially true, when specifying the agents' properties and behaviours. Even if adhering to the aforementioned behavioural representations, such as situation-action pairs, behavioural definitions provide great freedom to the modeller. Here, visual programming approaches can help to keep the learning curve low for non-programmers, to ensure that standard primitives (as described in the last section) are used and that well-phrased agent descriptions are encoded.

In fact, two distinct views have usually been used—one for definition of individuals and another one for observation at the system level. Various additional views may have been offered for simulation navigation and analysis. To our knowledge, we were the first to merge these views into one context, introspecting individuals and establishing inter-agent relations in one global context (von Mammen et al., 2016). This approach also has to blend visualisation (2D/3D) and input (symbolic/textual) modes. And if need be, it has to tightly integrate them with visual programming facilities. It promotes a close link between visual objects and programming logic that had been discussed several times before, e.g. by Burnett et al. (1995) and Citrin et al. (1995), and which had also been considered in visual modelling environments for agent-based systems by linking inspection views to simulated agents and using iconic references for formu-

lating behaviours (Repenning, 1993; Mota et al., 2013). The convergence of modelling and simulation spaces is becoming increasingly important as digital models grow closer to inform real-world situations, e.g. in (Wahby et al., 2015).

Reaching Application-Relevant Scales

Self-organising systems quickly tap into complex regimes due to the large numbers of agents and their incessant potential of interaction. In addition to the agents' state-changes, the neighbourhood topologies within the agent populations can continuously change. This class of dynamic systems with dynamic structures, or D^2S, has been identified as most challenging within the domain of complex system representations (Spicher et al., 2004, 2011). Hence, search for optimal system configurations does not only have to consider the overall system's state but also the path to get there (von Mammen and Jacob, 2008)—topology-altering effects in certain states may open up new state spaces. The ensuing self-referential fitness landscape (Müller-Schloer et al., 2011) might emerge from an interplay of a complex state space paired with a complex topological continuum. Yet, the actual complexities of D^2S are often less costly due to quasi-steady states of subpopulations of agents' states and topologies. Nevertheless, the computational complexity of $\mathcal{O}(n^2)$ of a simple boids model, in which flocking agents coordinate their flight in accordance with their neighbourhood (Reynolds, 1987) can only be brought down to $\mathcal{O}(nlogn)$ by applying acceleration algorithms deploying hierarchical spatial data structures to reduce the impact of the topological variance (Husselmann and Hawick, 2012).

Without consideration of biological behaviours, one can draw an analogy to the problem of detecting collisions among rigid bodies or of calculating the celestial trajectories of n bodies gravitating toward each other (Wang et al., 1990). Again, the costs of $\mathcal{O}(n^2)$ have to be taken into account for accurately calculating mutual influences, whereas pruning less significant influences based on spatial data structures can yield efficient approximations of gravitational influences (Trenti and Hut, 2008) or conservative results in case of collision detection (Lin and Gottschalk, 1998). In all three examples—boids, collision detection and n-body problem—the spatial topological arrangement, despite being dynamic, yields an opportunity for optimisation. We assume that other patterns in the variable dimensions of D^2S that, for instance, might consider the iteration numbers of cycles of biological cells, adhesive forces among them, or their proteomic configurations, might be exploited for optimisation purposes in similar ways.

Computational swarms can be defined as great numbers of agents with great degrees of freedom. Clearly, these attributions result in potentially equally great computational costs. At the same time, interesting simulation results which, for instance, lead to quasi-static or attractor states, have the potential to be used for model optimisation (von Mammen and Steghöfer, 2014). Such model optimisations can restrain the originally granted degrees of freedom—making the model more rigid but only in ways that do not affect its expressiveness and thereby reducing its computational costs. If the boundary conditions change, the original, high-cost model can be re-activated and the model patterns be refined. An according automated approach of model compression and relaxation could play an important role in ensuring efficient model representations and simulation.

Summary

Based on the swarm metaphor, we motivated and defined the term *interactive self-organisation*. It captures the notion of a process to tackle the development and work with self-organising systems in an interactive manner. Accordingly, we stepped through the development life cycle of self-organising systems, highlighting the conceptual intricacies held by self-organising system models and their link to application scenarios. Next, we stressed the need for and the value of accessible simulation. Here, we pointed out that like individualised medical treatment, the use of modelling and simulation of self-organising systems should be individualised, and thus, be made accessible to non-programmers. This is especially urgent due to the increasing degree of digitization of our everyday environments. Based on a short outlook on an application use case of interactive self-organisation in a scientific wetlab, we explained the high priority challenges in research and development of standardisation of representations, innovation of user interfaces, and computational scalability.

References

Abar, S., Theodoropoulos, G. K., Lemarinier, P., and O'Hare, G. M. (2017). Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, 24:13–33.

Andrews, P. S., Polack, F. A., Sampson, A. T., Stepney, S., and Timmis, J. (2010). The cosmos process version 0.1: A process for the modelling and simulation of complex systems. *Department of Computer Science, University of York, Tech. Rep. YCS-2010-453*.

Aster, R. C., Borchers, B., and Thurber, C. H. (2011). *Parameter estimation and inverse problems*. Academic Press.

Azuma, R. T. et al. (1997). A survey of augmented reality. *Presence*, 6(4):355–385.

Backlund, P., Engström, H., Johannesson, M., and Lebram, M. (2008). Games for traffic education: An experimental study of a game-based driving simulator. *Simulation & Gaming*.

Banks, J. et al. (1998). *Handbook of simulation*. Wiley Online Library.

Beck, F., Burch, M., Diehl, S., and Weiskopf, D. (2014). The state of the art in visualizing dynamic graphs. In *Proceedings of the Eurographics Conference on Visualization*, Swansea, UK. IEEE.

Bell, P. C. and O'keefe, R. M. (1987). Visual interactive simulation—history, recent developments, and major issues. *Simulation*, 49(3):109–116.

Brenner, S. C. and Carstensen, C. (2004). Finite element methods. *Encyclopedia of computational mechanics*.

Bucher, K., Blome, T., Rudolph, S., and von Mammen, S. (2019). Vreanimate ii: training first aid and reanimation in virtual reality. *Journal of Computers in Education*, 6(1):53–78.

Burnett, M. M., Goldberg, A., and Lewis, T. G. (1995). *Visual object-oriented programming: concepts and environments*. Manning Publications Co.

Cebulla, J., Kim, E., Rhie, K., Zhang, J., and Pathak, A. P. (2014). Multiscale and multi-modality visualization of angiogenesis in a human breast cancer model. *Angiogenesis*, 17(3):695–709.

Citrin, W., Doherty, M., and Zorn, B. (1995). The design of a completely visual object-oriented programming language. *Visual Object-Oriented Programming: Concepts and Environments. Prentice-Hall, New York*.

Computational Algebra Group (2015). *The Magma Handbook, V2.21*. Computational Algebra Group, University of Sidney.

Czaczkes, T. J., Grüter, C., and Ratnieks, F. L. (2015). Trail pheromones: An integrative view of their role in social insect colony organization. *Annual review of entomology*, 60:581–599.

Davison, T. and Denzinger, J. (2012). The huddle: Combining ai techniques to coordinate a player's game characters. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 203–210. IEEE.

Denzinger, J. and Kordt, M. (2000). Evolutionary on-line learning of cooperative behavior with situation-action-pairs. In *Proceedings of the 4th International Conference on Multi-Agent Systems (ICMAS 2000)*, pages 103–110, Boston, MA, USA.

Denzinger, J. and Winder, C. (2005). Combining coaching and learning to create cooperative character behavior. In *Proceeding of the Symposium on Computational Intelligence and Games*, Essex University, Colchester, Essex. IEEE.

Derényi, I. and Vicsek, T. (1994). Cooperative transport of Brownian particles. *J. Phys. I (France) Phys Rev Lett*, 75:374.

Dimigen, O. and Reinacher, U. (2012). Active vision plugin: An open-source matlab tool for saccade-and fixation-related eeg analysis. In *Perception*, volume 41, pages 246–246, London, UK. Pion Ltd.

Dror, R. O., Dirks, R. M., Grossman, J., Xu, H., and Shaw, D. E. (2012). Biomolecular simulation: a computational microscope for molecular biology. *Annual review of biophysics*, 41:429–452.

Duboule, D. (1995). *Guidebook to the homeobox genes*. Oxford Univ. Press, New York, NY (United States).

Dunleavy, M. and Dede, C. (2014). Augmented reality teaching and learning. In *Handbook of research on educational communications and technology*, pages 735–745. Springer.

Eaton, J. W. (2006). *GNU Octave: A high-level interactive language for numerical computations, 3rd ed.* e Free Software Foundation, Inc., Boston, USA.

Eissing, T., Kuepfer, L., Becker, C., Block, M., Coboeken, K., Gaub, T., Goerlitz, L., Jaeger, J., Loosen, R., Ludewig, B., Meyer, M., Niederalt, C., Sevestre, M., Siegmund, H.-U., Solodenko, J., Thelen, K., Telle, U., Weiss, W., Wendl, T., Willmann, S., and Lippert, J. (2011). A computational systems biology software platform for multiscale modeling and simulation: integrating whole-body physiology, disease biology, and molecular reaction networks. *Frontiers in Physiology*, 2(1–10).

Epstein, J. M. and Axtell, R. (1996). *Growing artificial societies: social science from the bottom up*. Brookings Institution Press.

Foley, J. D., Wallace, V. L., and Chan, P. (1984). The human factors of computer graphics interaction techniques. *Computer Graphics and Applications, IEEE*, 4(11):13–48.

Fouquet, D., Costa-Leonardo, A., Fournier, R., Blanco, S., and Jost, C. (2014). Coordination of construction behavior in the termite procornitermes araujoi: structure is a stronger stimulus than volatile marking. *Insectes sociaux*, 61(3):253–264.

Gilbert, S. F. (2013). *Developmental Biology*. Sinauer Associates, Inc, 10th edition.

Gregory, J. (2009). *Game engine architecture*. CRC Press.

Harper, R. (2003). *Inside the smart home*. Springer Science & Business Media.

Hosseini, S. M. and Feng, J. J. (2009). A particle-based model for the transport of erythrocytes in capillaries. *Chemical Engineering Science*, 64(22):4488–4497.

Husselmann, A. and Hawick, K. (2012). Spatial data structures, sorting and gpu parallelism for situated-agent simulation and visualisation. Technical report, Tech. Rep. CSTN-156, Computer Science, Massey University.

Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in artificial life*, pages 704–720. Springer.

Jones, M. M. (1967). On-line simulation. In *Proceedings of the 1967 22Nd National Conference*, ACM '67, pages 591–599, New York, NY, USA. ACM.

Klein, J. (2008). breve: a 3d simulation environment for multi-agent simulations and artificial life. `http://www.spiderland.org/`.

Klopfer, E., Scheintaub, H., Huang, W., and Wendel, D. (2009). Starlogo tng: Making agent-based modeling accessible and appealing to novices. *Artificial Life Models in Software*, pages 151–182.

Kolling, A., Nunnally, S., and Lewis, M. (2012). Towards human control of robot swarms. In *Proceedings of the seventh annual ACM/IEEE international conference on human-robot interaction*, pages 89–96. ACM.

Kravari, K. and Bassiliades, N. (2015). A survey of agent platforms. *Journal of Artificial Societies and Social Simulation*, 18(1):11.

Li, B., Sun, X., Leung, H., and Zhang, S. (2013). A survey of code-based change impact analysis techniques. *Software Testing, Verification and Reliability*, 23(8):613–646.

Lin, M. and Gottschalk, S. (1998). Collision detection between geometric models: A survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, volume 1, pages 602–608.

Lipson, H. and Kurman, M. (2013). *Fabricated: The new world of 3D printing*. John Wiley & Sons.

Magnenat-Thalmann, N., Ratib, O., and Choi, H. F. (2014). *3D Multiscale Physiological Human*. Springer.

Marras, S., Killen, S. S., Lindström, J., McKenzie, D. J., Steffensen, J. F., and Domenici, P. (2015). Fish swimming in schools save energy regardless of their spatial position. *Behavioral Ecology and Sociobiology*, 69(2):219–226.

McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., and Schmidt, B. (2006). Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 72–75.

Mikulka, J. (2014). Matlab extension for 3dslicer: A robust mr image processing tool. In *Progress in Electromagnetics Research Symposium, Proceedings*, pages 1875–1860, Guangzhou, China.

Minar, N., Burkhart, R., Langton, C., and Askenazi, M. (1996). The swarm simulation system: A toolkit for building multi-agent simulations. Technical report, Santa Fe Institute, Santa Fe, New Mexico, USA.

Mota, M. P., Monteiro, I. T., Ferreira, J. J., Slaviero, C., and de Souza, C. S. (2013). On signifying the complexity of inter-agent relations in agentsheets games and simulations. In *Proceedings of the 31st ACM international conference on Design of communication*, pages 133–142. ACM.

Müller, B., Balbi, S., Buchmann, C. M., De Sousa, L., Dressler, G., Groeneveld, J., Klassert, C. J., Le, Q. B., Millington, J. D., Nolzen, H., et al. (2014). Standardised and transparent model descriptions for agent-based models: Current status and prospects. *Environmental Modelling & Software*, 55:156–163.

Müller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Eurographics Association.

Müller-Schloer, C., Schmeck, H., and Ungerer, T., editors (2011). *Organic Computing - A Paradigm Shift for Complex Systems*. Autonomic Systems. Birkhäuser Verlag.

Naghsh, A. M., Gancet, J., Tanoto, A., and Roast, C. (2008). Analysis and design of human-robot swarm interaction in firefighting. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 255–260. IEEE.

Nguyen, T. A. and Aiello, M. (2013). Energy intelligent buildings based on user activity: A survey. *Energy and buildings*, 56:244–257.

Nickerson, D. P., Ladd, D., Hussan, J. R., Safaei, S., Suresh, V., Hunter, P. J., and Bradley, C. P. (2014). Using cellml with opencmiss to simulate multi-scale physiology. *Frontiers in bioengineering and biotechnology*, 2.

Nicolis, G. and Rouvas-Nicolis, C. (2007). Complex systems. *Scholarpedia*, 2(11):1473.

Noble, D. (2006). *The music of life*. Oxford University Press.

North, M. J., Collier, N. T., Ozik, J., Tatara, E. R., Macal, C. M., Bragen, M., and Sydelko, P. (2013). Complex adaptive systems modeling with repast simphony. *Complex adaptive systems modeling*, 1(1):1–26.

Parmentier, T., Dekoninck, W., and Wenseleers, T. (2015). Context-dependent specialization in colony defence in the red wood ant formica rufa. *Animal Behaviour*, 103:161–167.

Parry, H. R. and Bithell, M. (2012). Large scale agent-based modelling: A review and guidelines for model scaling. In *Agent-based models of geographical systems*, pages 271–308. Springer.

Polack, F. A. C. (2010). Proposals for validation of simulations in science. In *Proceedings of the 2010 Workshop on Complex Systems Modelling and Simulation*, pages pp. 51–74, Odense, Denmark. Luniver Press.

Pollini, L., Niccolini, M., Rosellini, M., and Innocenti, M. (2009). Human-swarm interface for abstraction based control. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Chicago, IL, USA*, pages 10–13.

Preim, B. and Dachselt, R. (2015). *Interaktive Systeme: Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces*. Springer-Verlag.

Redfern, D. (2012). *The maple handbook: maple V release 4*. Springer Science & Business Media.

Redfern, D. and Campbell, C. (2012). *The MATLAB® 5 Handbook*. Springer Science & Business Media.

Repenning, A. (1993). Agentsheets: a tool for building domain-oriented visual programming environments. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 142–143, New York, NY, USA. ACM.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Ritter, F. E., Schoelles, M. J., Quigley, K. S., and Klein, L. C. (2011). *Determining the Number of Simulation Runs: Treating Simulations as Theories by Not Sampling Their Behavior*, pages 97–116. Springer London, London.

Rothrock, L. and Narayanan, S. (2011). *Human-in-the-loop Simulations: Methods and Practice*. Springer.

Salazar-Ciudad, I., Jernvall, J., and Newman, S. (2003). Mechanisms of pattern formation in development and evolution. *Development*, 130(10):2027–2037.

Salmon, P. C. and Meissner, P. L. (2015). Mobile bot swarms: They're closer than you might think! *Consumer Electronics Magazine, IEEE*, 4(1):58–65.

Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17.

Schmidt, M. and Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85.

Sitti, M., Ceylan, H., Hu, W., Giltinan, J., Turan, M., Yim, S., and Diller, E. (2015). Biomedical applications of untethered mobile milli/microrobots. *Proceedings of the IEEE*, 103(2):205–224.

Slack, J. M. (2009). *Essential developmental biology*. John Wiley & Sons.

Spector, L., Klein, J., Perry, C., and Feinstein, M. (2005). Emergence of collective behavior in evolving populations of flying agents. *Genetic Programming and Evolvable Machines*, 6(1):111–125.

Spicher, A., Michel, O., and Giavitto, J.-L. (2004). A topological framework for the specification and the simulation of discrete dynamical systems. *Cellular Automata*, pages 238–247.

Spicher, A., Michel, O., and Giavitto, J.-L. (2011). Interaction-based simulations for integrative spatial systems biology. In Dubitzky, W., Southgate, J., and Fuß, H., editors, *Understanding the Dynamics of Biological Systems*, pages 195–231. Springer New York.

Steele, J. and Iliinsky, N. (2010). *Beautiful visualization*. O'Reilly Media, Inc.

Sycara, K., Lebiere, C., Pei, Y., Morrison, D., Tang, Y., and Lewis, M. (2015). Abstraction of analytical models from cognitive models of human control of robotic swarms. In *Proceedings of International Conference on Cognitive Modeling (ICCM)*, pages 13–19, Groningen, the Netherlands. University of Groningen.

Tarantola, A. (2005). *Inverse problem theory and methods for model parameter estimation*. siam.

Tesfatsion, L. (2006). *Handbook of Computational Economics*, volume 2, chapter Agent-Based Computational Economics: A Constructive Approach to Economic Theory, pages 831–880. Elsevier.

The Sage Development Team (2015). *Sage Tutorial, Release 6.9*. The Sage Development Team.

Théry, M. and Bornens, M. (2006). Cell shape and cell division. *Current opinion in cell biology*, 18(6):648–657.

Tisue, S. and Wilensky, U. (2004). Netlogo: Design and implementation of a multi-agent modeling environment. In *Agent 2004: Conference on Social Dynamics*, Chicago, IL.

Tomforde, S. (2012). *Runtime adaptation of technical systems: An architectural framework for self-configuration and self-improvement at runtime*. Südwestdeutscher Verlag für Hochschulschriften. ISBN: 978-3838131337.

Topol, E. J. (2014). Individualized medicine from prewomb to tomb. *Cell*, 157(1):241–253.

Trenti, M. and Hut, P. (2008). N-body simulations (gravitational). 3(5):3930.

Turk, M. (2014). Multimodal interaction: A review. *Pattern Recognition Letters*, 36:189–195.

Vaquero, R. M. M., Rzepecki, J., Friese, K.-I., and Wolter, F.-E. (2014). Visualization and user interaction methods for multiscale biomedical data. In *3D Multiscale Physiological Human*, pages 107–133. Springer.

von Mammen, S. (2016a). Interactive self-organisation. Habilitation Thesis, University of Augsburg.

von Mammen, S. (2016b). Self-organisation in games, games on self-organisation. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2016 8th International Conference on*, pages 1–8. IEEE.

von Mammen, S. and Jacob, C. (2008). The spatiality of swarms — quantitative analysis of dynamic interaction networks. In *Proceedings of Artificial Life XI*, pages 662–669. MIT Press.

von Mammen, S. and Jacob, C. (2009). The Evolution of Swarm Grammars: Growing Trees, Crafting Art and Bottom-Up Design. *IEEE Computational Intelligence Magazine*.

von Mammen, S., Müller, A., Latoschik, M. E., Botsch, M., Brukamp, K., Schröder, C., and Wacker, M. (2019). Via vr: A technology platform for virtual adventures for healthcare and well-being. In *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, pages 1–2. IEEE.

von Mammen, S., Schellmoser, S., Jacob, C., and Hähner, J. (2016). *The Digital Patient: Advancing Medical Research, Education, and Practice*, chapter 11. Modelling & Understanding the Human Body with Swarmscript, pages 149–170. Wiley.

von Mammen, S. and Steghöfer, J.-P. (2014). *The Computer after Me: Awareness and Self-Awareness in Autonomic Systems*, chapter Bring it on, Complexity! Present and future of self-organising middle-out abstraction. World Scientific Publishing.

von Mammen, S., Weber, M., Opel, H., and Davison, T. (2015). Interactive multi-physics simulation for endodontic treatment. In *Modeling and Simulation in Medicine Symposium at SpringSim 2015*, pages 36–41. Curran Associates, Inc.

Wahby, M., Divband Soorati, M., von Mammen, S., and Hamann, H. (2015). Evolution of controllers for robot-plant biohybdrids: A simple case study using a model of plant growth and motion. In *Proceedings of 25. Workshop Computational Intelligence*, pages 67–86, Dortmund. KIT Scientific Publishing.

Wang, L.-S., Krishnaprasad, P. S., and Maddocks, J. (1990). Hamiltonian dynamics of a rigid body in a central gravitational field. *Celestial Mechanics and Dynamical Astronomy*, 50(4):349–386.

Weiser, M. (1993). Ubiquitous computing. *Computer*, 26(10):71–72.

Whalley, P. (2006). Representing parallelism in a control language designed for young children. In *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on*, pages 173 –176.

Williams, B. (2006). *Microsoft Flight Simulator as a training aid: a guide for pilots, instructors, and virtual aviators*. Aviation Supplies & Academics.

Winikoff, M., Yadav, N., and Padgham, L. (2018). A new hierarchical agent protocol notation. *Autonomous Agents and Multi-Agent Systems*, 32(1):59–133.

Wolfram, S. (2003). *The Mathematica Book, Fifth Edition*. Wolfram Media Inc.

Wooldridge, M. (2008). *An introduction to multiagent systems*. Wiley. com.

Xu, Q., Jamniczky, H., Hu, D., Green, R. M., Marcucio, R. S., Hallgrimsson, B., and Mio, W. (2015). Correlations between the morphology of sonic hedgehog expression domains and embryonic craniofacial shape. *Evolutionary Biology*, pages 1–8.