Towards Valid Models and Simulations of Complex Systems.
A Plant Ecology Case-Study

Author: Teodor Ghetiu

Supervisors:
Dr. Fiona Polack
Dr. James Bown (University of Abertay)

November 6, 2009
## Contents

1 Introduction
   1.1 CoSMoS and Plant ecologies ............................................. 1
   1.2 Complex Systems modelling and simulating ................................. 1
   1.3 This dissertation ............................................................ 2

2 Complexity and complex systems ............................................. 3
   2.1 Complexity ................................................................. 3
   2.2 Types of complexity ......................................................... 3
      2.2.1 Algorithmic complexity ............................................. 4
      2.2.2 Deterministic complexity ........................................... 5
      2.2.3 Aggregate complexity ............................................... 5
   2.3 Summary .......................................................... 8

3 Engineering complex systems and CoSMoS ................................. 9
   3.1 Engineering emergence ...................................................... 9
      3.1.1 Engineering and science ........................................... 9
      3.1.2 Difficulties .......................................................... 10
      3.1.3 AOSE Methods ......................................................... 10
      3.1.4 The scientific method .............................................. 13
      3.1.5 Assurance-driven engineering ...................................... 14
      3.1.6 Equation-free macroscopic analysis ................................ 15
   3.2 Validity ............................................................ 16
      3.2.1 Introductory vocabulary ............................................ 16
      3.2.2 Validity problems ................................................... 17
      3.2.3 Validity Tests ......................................................... 18
      3.2.4 Development strategies ............................................ 19
      3.2.5 Critique .............................................................. 20
   3.3 CoSMoS contributions ....................................................... 20
   3.4 Summary .......................................................... 22

4 Plant ecology and complex systems ........................................ 23
   4.1 Ecology and the “Holy Grail” ............................................ 23
   4.2 Introductory plant ecology concepts ..................................... 23
      4.2.1 The individual level ............................................... 24
      4.2.2 The population and community levels .................................. 24
   4.3 Models and simulations ..................................................... 25
      4.3.1 EBM and IBM ......................................................... 26
      4.3.2 Stochasticity .......................................................... 26
      4.3.3 Spatiality ............................................................. 26
      4.3.4 Scale ................................................................. 27
      4.3.5 Protocols ............................................................. 27
   4.4 Summary .......................................................... 28
List of Figures

2.1 Fromm’s taxonomy for emergence (from [59]) ................................. 7
3.1 Game of Life gliders and glider gun .............................................. 10
3.2 First three workflows of ADELFE (from [19]) ................................. 11
3.3 Gaia’s models (from [146]) ......................................................... 12
3.4 The Prometheus method (from [105]) ........................................... 13
3.5 Relation between theory, engineering and adaptation (from [52]) ........ 14
3.6 Elements in the development of a system of systems (SoS) (from [134]) 15
3.7 Sargent’s process for developing computer simulations [124] ............... 17
3.8 The initial CoSMoS process (from [62]) ........................................ 21
4.1 Plant physiological processes and resource cycling (from [148]) ........... 24
4.2 Species-area curve (from Wikipedia) ............................................ 25
4.3 The ODD protocol (from [69]) ..................................................... 28
5.1 Research plan ............................................................................. 35
Abstract

Complexity and complex systems have been intriguing research objectives since the discovery of the well known Aristotelian paradox: the whole is more than the sum of its parts. While defining complexity remains elusive and "emergent", studying complex systems may be the only way towards further conceptual insights.

However, current day techniques of modelling and simulating are not subject to principled methods of development, while their scientific scope is often insufficiently motivated. In addition, the way computer models are presented is often distilled into verbal descriptions that do not facilitate complete reimplementation.

One way of addressing these limitations is to use rigorous software engineering methods, coupled with techniques that allow the construction of structured arguments of validity. This is not an easy task, as complex systems are through their definition decentralised and emergent. Bringing together the knowledge necessary to develop such principled methods is a first step towards their refinement.

Here we review first the literature relevant to the domain of complex systems, focusing on techniques for engineering emergent properties and for validating computer simulations. We then introduce plant ecology, the domain of our case-study, together with the main types of modelling techniques that have been employed so far. Two protocols that may potentially contribute to the wider field of computer modelling and simulating, are also described.

Preliminary results are available in the form of a published article on developing equivalence arguments for complex systems and the development of a large-scale, process-oriented simulation. While arguing about equivalence is a first step towards formulating arguments of validity, making use of the developed simulation will enable us to address scales that are closer to reality.

In conclusion, this document reviews literature from complexity science, computer and software engineering, in an effort to structure the way towards valid computer models and simulations. We are in the initial phase of our research, but addressing hard matter subjects such as arguing validity are within reach.
Chapter 1

Introduction

The work introduced in this report is part of the CoSMoS project\(^1\), a trans-disciplinary initiative aiming to contribute methodologically and substantially to the improvement of modelling and simulating complex systems. Its first and main expected deliverable is a process that addresses all stages of “in silico” research, from the formulation of abstract models for the real-world complex system under study, to the development of simulations which are accompanied by arguments of validity against the original system. In addition, making use of a massively parallel and distributed computing environment will facilitate the creation of simulations that match more closely the scale of real systems, an achievement which is of transdisciplinary relevance.

In order to accomplish these objectives, a case-study driven approach has been adopted. Research is being undertaken in parallel, in fields ranging from swarm robotics and artificial immune systems, to plant ecology and electronics. Different domains have different requirements and pose different challenges, so through this range of case-studies, the CoSMoS deliverables are intended to be developed and refined to a level of high representivity, generality.

This paper presents work undertaken as part of the plant ecology case-study.

1.1 CoSMoS and Plant ecologies

The defining role of the plant ecology case-study is to model and simulate a plant community, according to and feeding back into the CoSMoS process. A previously built C++ simulation [25] is the starting point and at the same time, the benchmark for the new implementation. This objective requires software engineering operations such as reverse-engineering and re-engineering, while software verification will add up to the CoSMoS way of developing simulations of complex systems. In addition, argumentation techniques from the domain of Safety Critical Systems (SCS) will be employed in order to substantiate the validity of the simulation. Following this primary goal, the case-study has the potential of strengthening the bond between Computer Science and ecology by turning the above into valuable ecological contributions. This might be called the distant or the emergent objective. Ecology can certainly benefit from thoroughly developed, argumented and up-scaled simulations. The “science of forecasting” envisaged by ecologists [81] is also a science of understanding complex systems.

1.2 Complex Systems modelling and simulating

Modelling and simulating are two activities that are beginning to receive attention in an increasing number of scientific disciplines. Complementing field experiments, providing means of generating predictions or just enabling the study of artificial realities [144], the two form a indispensable couple. Models are abstractions of objects or entities while simulations add dynamics and context. This is why, when dealing with complex systems, the way in which these models and simulations are developed is critical.

\(^1\)The CoSMoS project is funded through the EPSRC grant EP/E053505/1 and a Microsoft Research Europe PhD studentship, shared by the University of York and Kent University. More information can be found at http://www.cosmos-research.org
Complex systems are composed of many simple, interacting entities, that as a whole display non-simple behaviours. A cell is a complex system formed of many subunits, molecular machines and an environment of proteins and other chemical compounds. It is not only the high number of elements or their variety that makes modelling of such systems a complicated process, but also that richness of interactions. A modelling paradigm must be chosen (or developed), assumptions must be made, data that is considered not relevant filtered out, etc. Each of these steps introduce uncertainties [81]. Developing the actual simulation will add to the set of constraints that need to be taken into account.

In order to have a real scientific scope, the models and simulations that result from the above processes must be also accompanied by data attesting their validity. Usually, a statistical approach, if any, is undertaken to demonstrate that simulation results are within a certain acceptable range. However, it is important to discern between what is sound scientifically and data fitting. This is why, when developing models and simulations of complex systems, making use of proper methods, constructing arguments for their validity and making them available for external scrutiny are essential.

1.3 This dissertation

The following part of this dissertation reviews the background literature, then continues with a research proposal and the preliminary results of our efforts. Chapter 2 introduces the concepts of complexity and complex systems. Starting from the etymological roots of the word ‘complex’ and its wide use across disciplines, we concentrate our discussion on the essential aspects deriving from complexity theory. The main complexity types are detailed, together with the various notions associated with them. A wide variety of meanings and metrics are found in the literature and this is seen as an aspect preventing the development of a unified view, but contributing to the formation of an emerging one. We focus especially on aggregate complex systems and the pervasive phenomenon of emergence, as they are key points of our work. Being able to engineer systems with particular emerging properties or the reverse, to identify the properties that lead to specific emergent properties, will shed more light into the nature of complexity per se and provide means for further developments.

Having considered the essence of complexity research, we present in Chapter 3 relevant work on modelling and simulating complex systems. Firstly, we focus on the topic of engineering emergent systems, presenting the difficulties that are involved in the process, together with already established methods. This is followed by a review of validating techniques applied to computer modelling and simulating. The chapter ends with an overview of work that has been published under the CoSMoS project, which aims to develop a methodology for constructing models and simulations of complex systems, that are also supported by arguments of validity.

Chapter 4 is dedicated to plant ecology. We start by introducing the domain of ecology and its focus on the ‘distribution and abundance’ of species. This is continued by an overview of plant ecology, starting from physiological notions and building up to the community level patterns that are described in literature. Following this part is a section dedicated to modelling and simulating techniques applied in plant ecology.

Chapter 5 describes our research proposal. We discuss here also the plant ecology case study. Finally, Chapter 6 contains preliminary results while the appendix lists the paper “Equivalence Arguments for Complex Systems Simulations – A Case-Study” awaiting publication.
Chapter 2

Complexity and complex systems

Complexity is “a word-problem and not a word-solution” [4]; at least this is how it used to be considered in many academic and non-academic circles. Steering views towards such an opinion has been the confusion caused by the uses of the word “complex”, which has been improperly associated with notions such as size, difficulty, order [51]. Even the difference between what is complex and what is complicated has not always been taken into account. What is important though is that complexity is a pervasive feature of the real-world. Efforts concentrated on understanding the real-world have the potential of being fed back into the general research on complexity and change this connotation.

2.1 Complexity

The word complex comes from the 14th century Latin expression ‘complexus’ meaning ‘embracing or comprehending several elements’ that are ‘plated together, interwoven’ [4]. Literature provides a wide range of uses for the term complex, ranging from composite, compound to complicated, involved or intricate. Understood as a plural of quantity and quality, different sciences have conveyed it different specific meanings: mathematics makes use of complex numbers or complex fractions, linguistics studies complex sentences, semiotics considers complex terms, music defines complex sounds, etc. The list of domains and meanings of the word ‘complex’ does not end here, but already the above enumeration should point to the fact that complexity, as a concept, has a pervasive nature.

In most of the cases, the terms ‘complex’ and ‘complexity’ were involved in expressions looking to indicate the opposite of simplicity. This is where the confusion between ‘complex’, as defined above, and ‘complicated’ becomes clearer. Complicated systems consist of a large number of elements, but these are all “knowable, definable and capable of being catalogued as are all of the relationships” between them; in addition, “cause and effect can be separated" effectively [131]. Examples are many engineered systems such as aircrafts, vehicles, etc, which are predictable and usable because of these properties. Complex systems are those where the “whole is more than the sum of its parts”, as expressed by Aristotle in 4th century BC [8]. They consist of entities that interact in non-linear ways, so overall, the system is “irreducible” to just the set of its parts. Cause and effect are intertwined, thus cannot be separated [83].

In the second part of the 20th century, the science of complexity and complexity theory were developed. These have been concerned directly with the development of complexity semantics, metrics and the study of complex systems [141]. When aiming to quantify complexity, the same general meaning can be related to different metrics. On the other hand, when undertaking complex systems research, the behaviours and the interactions that bring them about are primary objects of study. The following section will provide an insight into this diversity of meanings, metrics and approaches.

2.2 Types of complexity

One result of complexity research is the classification of research problems into three classes of complexity: organised simplicity, disorganised complexity and organised complexity [99]. Organised simplicity
characterizes systems composed of a smaller number of significant components, interacting in a deterministic way. Studying such systems benefits from readily available mathematical methods, that can be applied to the system as a whole or to its constituent parts separately. Classical physics with its focus on objectivity, causal explanations and certainty stands as a source for this approach [4].

Disorganised complexity refers to systems that display chaotic, irregular behaviours, being thus the opposite of simplicity. They generally have a higher number of significant components, displaying a specific degree of randomness (e.g. molecular vibrations, the weather and climate, population growth in ecology). Consequently, statistics and the theory of probability have played a key role in their analysis, approach which challenged the paradigm of rational mechanics.

Organised complexity is situated between the two poles of organised simplicity and disorganised complexity and encompasses “all problems which involve dealing simultaneously with a sizeable number of factors which are interrelated into an organic whole” [143]. Examples of systems showing this type of complexity can be found in all types of social systems, from ant colonies and the problem of food foraging, to human societies and their specific functions. Contemporary problems from many sciences have been considered too complicated for being addressed with models of rational mechanics, while disorganised complexity methods do not fit the large number of non-random system components [57]. As a consequence, the set of quantitative methods for “untangling” organised complexity is still incomplete [147].

A more detailed classification of complexity types is provided by Manson [95]. In his review, Manson describes also three types of complexity: *algorithmic, deterministic* and *aggregate*. These are reviewed in the following three subsections.

### 2.2.1 Algorithmic complexity

Algorithmic complexity is based on mathematical complexity theory and information theory and considers complexity as the difficulty of solving a given problem or describing a system’s characteristics. In the first case, the effort necessary for problem solving is measured by the *computational complexity*. According to their difficulty, problems can be classified in complexity classes such as *P*, *NP* or *NP complete*. In the latter case, the focus is placed on the compressibility of bit strings rather than other factors. If a high rate of compression can be achieved without loss, the bit string is considered simpler than one which cannot be compressed similarly. To measure the complexity of a system, this must first be described as a bit string and then a complexity metric applied to it.

A wide series of complexity metrics of informational nature have been developed. The following are extracted from [64, 51]:

- **algorithmic information content (AIC):** is defined as “the length of the shortest program that will cause a standard universal computer to print out the string of bits and then halt” [64] – this metric is also known as the Kolmogorov complexity. According to this metric, random bit strings have maximal AIC, which is why it is considered unsuited as a definition of natural complexity.

- **effective complexity:** peers down to the importance of regularities and is defined as the length of the description of a string’s regularities. This way, a random string will have minimal complexity and so will a completely regular one.

- **mutual AIC:** in order to find more regularities, the mutual AIC was proposed. Dividing a string in parts “in a particular way” [64], the mutual AIC is the difference between the sum of the AIC of the segments and the AIC of the whole string.

- **logical depth:** is defined as the time necessary for a standard universal computer to process the string, print it and then halt. In the case of fractals, which can be described through compact mathematical formulas, logical depth is more relevant than AIC.

- **crypticity:** is the reverse of logical depth, meaning that it is the time required by a standard universal computer to go from a bit string to a shorter program that can generate it.

- **potential complexity:** represents the average effective complexity of an entity, from the present up to a specified future. This metric had been proposed for evolving systems, where the complexity at any present stage may be much smaller than that of the future.
The literature of algorithmic complexity metrics contains more examples, but for reasons of conciseness we will limit our enumeration here. As a final remark on this complexity class, according to Gell-Mann, these candidate quantities are often uncomputable and:

“all such quantities are to some extent context-dependent or even subjective. They depend on the coarse graining (level of detail) of the description of the entity, on the previous knowledge and understanding of the world that is assumed, on the language employed, on the coding method used for conversion from that language into a string of bits, and on the particular ideal computer chosen as a standard.” [64]

2.2.2 Deterministic complexity

Deterministic complexity builds on chaos and catastrophe theory and considers that systems described by two or more variables can be largely stable but prone to sudden discontinuities. Having received a deterministic mathematical definition, these systems show chaotic behaviours: under certain conditions, small fluctuations of a variable may have large, non-linear effects on the others [95].

To exemplify this situation, we shall look at the well known logistic equation [140], while more details on deterministic complexity can be found in [96, 95]. The standard logistic equation for population growth can be defined as:

\[ X_{t+1} = \alpha X_t (1 - X_t) \] (2.1)

Here, the population size at time \( t \) is defined by the variable \( X_t \in (0, 1) \) – the real numbers in the \((0, 1)\) interval map to integer values in \((1, \infty)\). The growth factor \( \alpha \in (0, 4) \) controls the rate population size increase for each timestep. If \( \alpha \) belongs to the open interval \((1, 3)\), then, regardless of the initial population size \( X_0 \), the system will always evolve towards a constant value of \((1 - 1/\alpha)\) – this is called an attractor. The other two attractors of the systems are 0 and \( \infty \), reached when \( \alpha < 1 \) or \( \alpha > 4 \) respectively. However, if \( \alpha \in (3, 4) \), the system oscillates between attractors – the population size peaks and drops periodically.

Associated to deterministic complexity is also the notion of feedback. In the above example, the future population size \( X_{t+1} \) is dependent on the present one \( X_t \). Two types of feedback, negative and positive, may occur. Negative feedback keeps in a stable state the values of a set of variables, as influenced by the changes of other variables. Attractors such as \((1 - 1/\alpha)\) are a consequence of negative feedback. Positive feedback has the opposite, self-reinforcing effect: the set of influenced variables moves towards a point of no return. This is the case for the other two system attractors, 0 and \( \infty \).

2.2.3 Aggregate complexity

Aggregate complexity is concerned with systems formed of many simple elements, that display complex behaviours. Taking a step forward from algorithmic and deterministic complexity, which focused more on artificial metrics and simplified abstractions, this complexity class provides a mechanistical way of studying complex systems and the phenomena associated with them. The objects of focus map more naturally to real-world systems. They do not need to be translated into bit strings or described by a limited set of variables. Also, the domain of concern is much wider. Aggregate complexity studies key concepts that relate to a system’s definition, such as internal structure, component relationships, environment, emergence, learning or evolution. Each of these concepts is equally important and necessary to be addressed in order to obtain a full understanding of the object of study.

Internal structure

In a complex system, the “interactions of many simple agents give rise to complex patterns at the global level” [48]. In doing so, systems are structured according to a certain hierarchy. Work focusing on this subject led to the development of the hierarchy theory [128], according to which a “partial ordering” can be observed within systems [129].

Two types of hierarchy can be identified: nested and non-nested. Nested hierarchy applies to systems consisting of holons – components, that are similarly composed of lower-level sub-components. This ordering continues until the level of elementary particles. For example, the human body is a holonic system displaying nested hierarchy: it is composed of different systems such as the circulatory and
nervous systems, which, in their turn, are composed of a numerous number of cells; the ordering continues until the level of sub-atomic particles. Non-nested hierarchy applies to systems where containment is not observed. An example, in this case, would be any trophic level hierarchy: the different organisms are topologically organised according to feeding relationships, rather than containment.

Hierarchical systems possess both a vertical structure, consisting of layers, and a horizontal one defined by holons. There are important differences between the layers of a system: higher levels display larger and slower entities, while lower level entities are smaller and faster [147]. Within one layer, there are differences with respect to the degree of interactions among holons.

Relevant to a hierarchical structure is also the concept of decomposability. A complex system is not completely decomposable as “complete decomposability only occurs when coupling between components becomes zero, which seems a trivial case because, by definition, a system is composed of interacting parts” [147]. To account for this situation, the term nearly decomposable has been introduced [128]. Systems in which components are dependent in the long term, but independent in the short term, are examples of near decomposability.

Interactions

The interactions among components are equally, if not more relevant in describing a system [95]. Three classes of interactions can be identified: interactions within sub-systems, among sub-systems, and between the system and its environment. There are often differences of orders of magnitude between these classes of relations. For example, intermolecular forces are generally weaker than molecular ones, and these than nuclear forces. Being able to capture the number and variety of inter-component relationships is an aspect through which aggregate complexity extends beyond the possibilities of the algorithmic and deterministic classes. Costanza et al [42] note that, due to these two factors, inter-component relationships are described by higher order, nonlinear processes rather than just feedback.

Environment

The environment is also central to understanding a complex system. As a definition, anything external to a system is considered to be part of the environment. For example, in the real world, an organism’s environment is formed by all biotic and abiotic factors existent in its surroundings [24]. Because such systems are usually open and far-from-equilibrium [134], there is a bidirectional flow or information, matter or energy, between them and their environment [95] and this in turn, drives the systems’ functioning and evolution.

One must also note that the distinction between the environment and the system is not always that sharp. The behaviour of a complex system is dependent on the way it integrates within its environment. In other words, behaviours that are specific of complex systems are also contextual.

Emergence

Emergence is a fundamental process observed throughout our complex systems rich world [75] and can be described as the means through which non-simple behaviours or properties arise in systems composed of simple, interacting components. System properties are emergent if they are not a property of any individual element, while emergence represents the appearance of such properties and structures on a higher level of organization [12]. While the list of definitions may continue, or in some cases, be replaced by descriptions [75], it is important to record the generality and pervasiveness of the process.

In nature, the study of emergence is usually associated with desirable properties that result from decentralised behaviours. For example, ant colonies successfully forage for food without having a central coordinator, but guided only by the pheromones left on the ground by food carrying ants [67]. Flocking behaviour and the use of pheromones stand behind other natural ensembles such as piles of termites, flocks of birds, swarms of bees, schools of fish and packs of wolves [59]. Other properties can be identified in other natural systems, e.g. robustness, adaptability, fault tolerance. Considering that these examples describe non-hierarchical systems, the opposite of centralised systems which still represent the norm in science and technology [115], the importance of emergence is further emphasized. This goes in line with efforts for scientific and technological novelty through bio-inspiration [94].
Similar to the term ‘complexity’, which bears different meanings for its different facets, emergence can also undergo different types of classifications. For example Chalmers [36] distinguishes between strong and weak emergence. Strong emergence cannot be even theoretically inferred from the laws of the low-level domain. Contrary to the accepted understanding that the behavior of the whole is determined by the parts, strong emergence characterizes the fact that the properties of the whole may also determine the behavior of the parts. Weak emergence is an “unexpected” property that is difficult to be inferred, due to high computational requirements. One way of addressing it is through “data mining” like techniques, considers Bar-Yam [14]. Given the availability of information regarding the positions, momenta of all the particles and their interactions, macroscopic properties of systems can be extrapolated from microscopic ones. As an example, statistical physics may be considered a domain focusing on weak emergence: “the successes of describing equilibrium systems, including phase diagrams and thermodynamic transitions from statistical averages over microscopic representations, demonstrate the validity of the approach of this field” [14]. However, weak emergence is insufficient to describe all collective behaviours.

Bedau [15] adds nominal emergence to this list, and associates it the classical meaning of the appearance of a system property, which is not displayed by any single component. Seager [126], as cited by [59], sees a different classification: benign emergence, which can be described in a specific explanatory scheme (e.g. the temperature and pressure of a gas can be described in mathematical terms), and radical emergence, which is semantically similar to the previous strong emergence.

Taxonomies for emergence are abundant. For example, Wolfram [145] proposes a taxonomy for cellular automata (CA) based on four classes, homogeneous (Class I), regular (Class II), chaotic (Class III) and complex (Class IV). Eppstein focuses on the types of feedback observed, negative, positive or combined. Bar-Yam [14] sets up a taxonomy for “ensembles” and “particles”, instead of agents or groups. He considers weak emergence (Type 1) and two types of strong emergence, one relating to the properties of the system (Type 2) and one deriving from the relationships between these properties and those of the system’s environment (Type 3). If Type 2 emergence relates to Chalmers’ [36] strong emergence, Type 3 emergence states that the higher level pattern or property appears through the coupling of the system to its environment, e.g. a key’s ability in opening a door depends on the relationship between its properties and those of the door’s lock. Fromm [59] builds on this previous work and proposes a four layered classification, which reflects both causality and feedback driven effects (fig. 2.1). In his opinion, “a question about a process of emergence is always a question of causation and causality – the search of a hidden cause for an apparent effect”. His taxonomy is slightly more detailed and bears the structure presented in figure 2.1.

![Figure 2.1: Fromm’s taxonomy for emergence (from [59])](image)

As observed in the above taxonomies, the concepts of feedback and causality are strongly related to

---

1http://www.ics.uci.edu/~eppstein/ca/wolfram.html
that of emergence. If Fromm’s [59] simple emergence is seen as just a feedforward effect, weak emergence is defined by a simple negative or positive feedback. Type III emergence relates to a learning, adaptation and multiple feedback and appears in very complex systems or complex adaptive systems (CAS). Type IV emergence is classed as a multi-level emergence, characterised by a combinatorial explosion of emergent states within a system. In terms of causality, two types are widely encountered: upward and downward causation [34]. Upward causation refers to the fact that the lower-level entities define the properties and behaviour of the overall system, while downward causation expresses the fact that the system influences the behaviour of the lower-level entities. Opinions vary in terms of accepting downward causation, ranging from total rejection [2] to partial recognition [15] and acceptance as a fact [35].

Complex systems

We have presented descriptions of complex systems, as they are understood through aggregate complexity. Providing a unifying definition is not an objective here, especially considering the variety of such systems. Apart from complex systems, researchers focus on CAS, systems of systems, etc. All of these obey the “simple interacting components leading to non-simple system behaviours rule”. Here we list three more definitions that have become relevant throughout literature.

Simon [128], in his early work on hierarchy theory, notes: “nearly decomposable systems are systems where interactions between components are weak, but not negligible (e.g. rare gases, organisations)”. Here the focus is placed on component interactions. Later on he expands by stating that a hierarchical system is:

"one made up of a large number of parts that interact in a non-simple way. In such systems, the whole is more than the sum of its parts [...] in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole.” [128]

While Simon’s definition allows room for ambiguities, Cilliers [37] describes complex systems in terms of ten properties: these have a large number of elements, engaged in rich, non-linear, local and recurrent interactions. The elements have history and react based on local knowledge. Overall, systems are usually open and far-from-equilibrium.

Finally, a more conceptual definition comes from the Santa Fe Institute: they consider complex systems in terms of concepts such as heterogeneity, feedback, organisation, emergence. This is an example of a compact definition, where the notion of “complexity” emerges out of the ensemble formed by the aforementioned concepts.

2.3 Summary

In this chapter we have focused on the notion of complexity and on relevant concepts that are associated with it. We started our discussion by describing the problematic nature of complexity and the term ‘complex’. While being the opposite of ‘simple’, ‘complex’ is not identical to ‘complicated’ and it refers to more than a quantity: it takes into account behaviours, architectures and relationships.

Observed phenomena can be classified into different types of complexity. Ranging from simplicity, to organised and disorganised complexity [99], or divided into algorithmic, deterministic and aggregate complexity [95], these taxonomies are comprehensive, but they also show the lack of a unifying view on complexity.

Current understanding of complex systems, as framed on aggregate complexity, is that they are ensembles of interacting, simple elements that, as a whole, display complex behaviours. Similar to the notion of ‘complexity’, providing a unique definition for what is a complex system seems to not be a feasible objective. Apart from the common aspects mentioned before, various authors see different properties as being essential for defining a system as being complex. Concepts like architecture, interactions, environment, emergence, learning or evolution need to be taken into account.
Chapter 3

Engineering complex systems and CoSMoS

In this chapter we present the current set of contributions made by the CoSMoS project, together with adjacent work on validation and argumentation techniques. We focus initially on the engineering of emergent systems, activity which greatly depends on the use of sound models (and simulations) for its success. The problem of computer models and simulations validity is then addressed, followed by a short presentation of argumentation techniques from the domain of Safety Critical Systems. The last part of the chapter details the work published by the CoSMoS group.

3.1 Engineering emergence

As detailed in Section 2.2.3, emergence is a hallmark of complex systems, in which system-level properties appear from low-level component interactions. In nature, these properties often take the form of beneficial features that, in the case of many engineered systems, represent a requirement. However, classical engineering techniques are still limited by a prevalent centralised mindset [115], which means that natural, distributed solutions do not abound. Engineering decentralised systems that are able to display prescribed emergent properties is thus an essential research direction.

3.1.1 Engineering and science

According to the Oxford English Dictionary, *engineering* is “the branch of science and technology concerned with the design, building, and use of engines, machines, and structures” [132]. The same source regards the Latin originating term *science* as “the intellectual and practical activity encompassing the systematic study of the structure and behaviour of the physical and natural world through observation and experiment”. From these two definitions it can easily be observed that engineering is not only a branch of science, but it shares a common keyword with it: *structure*. Science studies structures while engineering is concerned with their design, building and use.

An even stronger connection can be identified through the study of emergence. The central question of emergence is that of obtaining higher levels of organisation from simple local rules. At the same time, this is also the core problem of science: “to find simple rules for complex patterns, to uncover simple laws behind complex behaviours and to discover simple theories for complex phenomena” [60].

Understanding the mechanisms and limitations of emergence is important, as it would simplify science and engineering. In order to accomplish this, underlying principles must be primary research objectives [93]. This will allow “local-to-global” or “micro-to-macro” problems [149] to receive their long expected solution. Science and engineering are having then complementary roles in this effort:

“In science we want to understand complex systems, in engineering we want to construct them; in the former we want to explain complexity through a simple rule, equation or formula, in the latter we want to hide complexity behind a simple interface.” [60]
3.1.2 Difficulties

Academia has addressed the subject of emergence from three major perspectives: philosophy, biology and computer science [134]. How the mind emerges from the collective set of neurons, how life spawns from inanimate matter and how analogous properties emerge from non-biological substrates (computers) have been the three main drives of research. In all three of these wide scientific domains, a number of difficulties are encountered.

Firstly, emergent properties are not analytically tractable from the attributes of system components [11]. In this sense, system wide characteristics do not result from superposition (adding together the system components) [88].

Secondly, there is a discontinuity between the system level and the component level. Not only do they have different length and time scales, but they are also described by different languages, using different concepts [134]. For example, the well known Game of Life [61] sees a cellular automaton defined in terms of cells, states and update rules, display patterns such as gliders and glider guns (fig 3.1). The degree of distinction between the two languages may characterise the strength of the emergence, ranging from weak to strong.

![Figure 3.1: Game of Life gliders and glider gun](image)

Thirdly, emergence is associated with unpredictability, as caused by the autonomous nature of system elements. If engineering associates specific purposes, functions and roles, complex systems are formed of components who do not make such a clear distinction [60]: in a multi-agent system (MAS), agents may play multiple roles or change their behaviour according to their environment, whereas a CA cell is not purposeful, etc. Autonomy and self-organisation, two concepts closely related to emergence, are then seen as being opposed to engineering.

Problematic is also the fact that emergent properties are paradoxical: “they are changeless and changing, constant and fluctuating, persistent and shifting, inevitable and unpredictable. Moreover an emergent property is a part of the system and at the same time it is not a part of the system, it depends on a system because it appears in it and is yet independent from it to a certain degree” [59]. In spite of these aspects, emergent properties are patterns that persist throughout the changing of components [75].

The difficulties detailed here add up to the important aspects of emergence which are still debatable: the existence of downward causation and strong emergence [14], the scientific status of their study [2] or even the very definition of emergence [75, 20].

A number of methods for dealing with the difficulties of working with complex, emergent systems, have been proposed. A considerable part of them come from the domain of Agent-Oriented Software Engineering (AOSE), as this agent-based paradigm naturally maps to the aggregate understanding of reality. Other suggestions focus on more traditional Software Engineering (SE) approaches or even the classical scientific method. They shall be detailed in the following part of this section.

3.1.3 AOSE Methods

AOSE methods for developing multi-agent systems are focused on refining aspects such as internal and social architecture, communication, pro-activity, distribution and autonomy [7]. There is a high variety among methods though, in terms of which of these aspects are captured, and through what means. The remainder of this section will detail a number of well known methods, but the literature is much wider (e.g. see methods such as MESSAGE [32], AAII [85], DESIRE [26]). A more extensive review of AOSE methods can be found in [135, 79].
ADELFE

ADELFE [19] is a method that is claimed to be supporting emergent behaviour, as suggested by its name (Atelier de Développement de Logiciels à Fonctionnalité Emergente). Based on the Rational Unified Process (RUP) [80], this method builds on top of the Adaptive MAS theory (AMAS) [66] which considers the design of adaptive systems composed of non-adaptive, cooperative agents. The reasoning behind this theory is: “for any functionally adequate system, there is at least a cooperative interior medium system which fulfills an equivalent function in the same environment” [66]. Hence the interest in cooperative (interior medium) systems.

Following from the above, ADELFE is not a general modelling method, but one profiled for cooperative, AMAS applications. In doing so, it covers all stages of software engineering, implementing the requirements workflow, the analysis workflow, the design workflow, etc. Adaptation is essential, so requirements include a model of the environment. This consists of definitions for the actors, the context and the environment itself. Analysis allows the identification of agents, which in this case are restricted to being cooperative. The design phase sees the construction of agent models, which make use of designer defined non cooperative situations (NCS). Agents detect and address these NCS in a predefined way. Figure 3.2 depicts the first three workflows of ADELFE.

![Figure 3.2: First three workflows of ADELFE (from [19])](image)

GAIA

The GAIA methodology [146] is an example of an approach with general applicability, that is addressing both the micro and macro aspects of a system. Its authors target real-world, large-scale applications the display characteristics such as:

- agents are heterogeneous (not bound to any implementation requirements) and represent coarse-grained computational systems
- the organisation structure is fixed (relationships do not change in time)
- agents and their services are not adaptive
- the total number of agent types is small (less than 100)

GAIA is a top-down process that moves from “abstract to increasingly concrete concepts” [146]. In doing so, successive models of the systems are developed (see figure 3.3).

The analysis starts with the identification of roles and the interactions between them. Each role is defined by responsibilities, permissions, activities and interaction protocols. Being platform independent, GAIA provides support for the design of applications up to an intermediary level, which can then be addressed through traditional design techniques. Three models are developed in this stage: the agent model defines the types and instances of agents that will make up the system; the services model describes the services associated with the specified roles; the acquaintance model lists the inter-agent communication protocols.
Tropos

*Tropos* [27] is developed on top of the notion of agency and its related mentalistic concepts (e.g. beliefs, goals, actions). The method puts accent on the early phase of requirements analysis, directing attention towards the description of the environment where the software will operate and on the human-agent interactions.

In more detail, Tropos supports five phases of software development: early requirements, late requirements, architectural design, detailed design and implementation. Early requirements lead to the definition of agents, with their goals and their inter-relationships. These are described in a diagramatic way – through actor and rational diagrams. Late requirements provide the overall system description. The architectural design phase sees the specification of the system’s agents being finalised. Following it is the detailed design phase, where the agent internals are defined – events, plans, beliefs, communication protocols. The implementation is carried out through the JACK programming platform [77].

Prometheus

Prometheus [105] is another method for designing, documenting and building agent systems. Aiming to cover all phases of software development, Prometheus not only details each of them, but also provides guidance for constructing design artifacts. In doing so, a set of hierarchical structuring mechanisms through which design can be performed at multiple levels of abstraction are proposed. Figure 3.4 shows the overall strategy behind Prometheus.

Three phases define the development process: the system specification phase, the architectural design phase and the detailed design phase. Starting from the high-level view, the first phase leads to the formulation of system inputs (percepts), outputs (actions), functions and any relevant shared data sources. Based on the outputs, the second phase is focused on defining the agent types that will form the system and describing their interactions. Finally, the internal architecture of each agent type is designed in order to enable the accomplishment of goals.

Critique

Each of the methods proposed in the literature has strengths and limitations. They get distinguished by the scope of their use of through the capturing of aspects that is not common to others.

To start with, ADELFE is a niche oriented method that applies to systems respecting the AMAS theory. Other methods such as GAIA, Tropos aim to have general applicability. This aim can sometimes be illusory. Andrews et al [7] notice two limiting aspects: firstly, the methods are oriented on
Figure 3.4: The Prometheus method (from [105])

social systems, displaying thus a tendance for capturing properties such as autonomy, learning abilities; secondly, they “do not capture the time, space and component-quantity aspects of complex systems, or the layered abstraction aspect”. In other words, generality is marketed without the provision of detailed considerations on aspects critical to complex systems.

Another aspect to be assessed is their scope. ADELFE covers the whole life cycle of software engineering, together with Prometheus and Tropos, while GAIA does not address the implementation phase. However, aspects of software testing are not included in the above descriptions, nor the more pressing needs of software validation [108]. Also, not all of them provide the same level of guidance and resolution of details. If Prometheus is a result of both academic and industrial expertise [105], and has been applied to the design of various complex systems [3], not the same can be said about other methods which have been demonstrated on toy systems. ADELFE benefits from using the UML and agent-based UML (AUML) notations [103], whereas Tropos employs actor and rational diagrams.

A differentiating aspect, very important to complex systems, is the role of the environment. While ADELFE defines an explicit model of the environment and Tropos provides a weaker description based on agents and their interactions, methods such as GAIA or MaSE [46] have no particular model of the environment. If systems are open and far-from-equilibrium [134], how can emergent properties be controlled without having explicitly considered the environment?

Finally, a positive aspect is that most of the methods are iterative. GAIA, ADELFE and Prometheus are only a few such examples. Iterative development helps avoiding the need for faking the rationality of the design process [106].

3.1.4 The scientific method

Edmonds and Bryson [52] propose a method that combines engineering and adaptation, aiming to put “more emphasis on natural scientific approaches than has been usual in multi-agent systems” (fig. 3.5). In their opinion, adaptation and experimentation must be better reflected in the design and engineering stages of development. Starting by distinguishing between the “engineering” approach and the “adaptive” approach to obtaining a certain construct, they consider formal design methods as insufficient for generating programs of various specifications, in particular for self-organising software systems. Engineering and adaptation (fine tuning) must work together and this is exemplified by applying evolutionary techniques to MAS. For their method to work, Edmonds and Bryson require testable hypotheses to be
made about the behaviour of the system, and propose that these accompany the final product. As such, “this is simply an application of the ‘classic’ scientific experimental method” [52].

Figure 3.5: Relation between theory, engineering and adaptation (from [52])

Fromm [60] also reviews examples of “an intelligent design based on the classic scientific method”. Having identified emergence as the link between scientific and engineering efforts, he chooses the scientific method as the principal way to advance, as “there is no reason why the scientific method cannot be applied to artificial worlds”. Fromm continues by detailing five customizations of this approach: the graduate student method, the interactive man-machine method [31], the iterative two-way approach [40], synthetic microanalysis [10] and goal-directed simulations [100]. While these approaches were marked by various successes (e.g. Codd’s self-reproducing CA [31] obtained through the interactive man-machine method), they also have serious deficiencies. Fromm delivers the bad news: “the approach is not a formal approach or detailed recipe, it does not offer explicit guidance or a complete construction manual, and relies on human experience and creativity [...] the performance of the approach is certainly difficult to measure” [60].

3.1.5 Assurance-driven engineering

Stepney et al focus on assurance-driven engineering [134]. Following the lines of evidence-based software engineering [86, 142], attention is directed towards the dependability of engineered systems, property which is itself emergent [47]. Accepting the products of engineering, both for commercial and academic purposes, requires confidence which is gained through dependability, and this must be demonstrated through “clear, defensible arguments” [134].

Previous work from Polack and Stepney [109] showed that applying classic refinement techniques to the engineering of emergent systems is not possible. Such systems are defined by at least two levels of description, a low level one for to the individual components and a high level one for the aggregate ensemble formed by them – in refinement terms, a concrete and an abstract level. Because the concepts describing one level may not be used in describing the other, “there is no single refinement relation” that can bridge the two.

As finding a common language definition is potentially a restrictive task, the authors propose linking the two levels through intermediary refinements. To exemplify this strategy, Conway’s Game of Life is
used as a case study. Here, the components are the CA cells, while the system is described by emergent properties such as gliders or glider guns. Refining a (low level) cell to a cell on a representation and a (high level) glider to a glider on the same representation allows the two specifications to be linked by the physics of state and motion [109].

Emerging from this work is a three-fold architecture (fig. 3.6), composed of a high-level system model, a local component model and an integrated model of the implementation [134]. The three models are linked by an integration environment. Descriptions of the way this composite method may be applied are provided for GoL gliders [109], CA and mobile processes simulations of platelet systems [138] and for an imaginary nanite platelet system.

![Figure 3.6: Elements in the development of a system of systems (SoS) (from [134])](image)

### 3.1.6 Equation-free macroscopic analysis

A technique which may be used for researching the mechanisms behind emergent properties, and hence integrated into the toolset required for engineering emergent systems, is equation-free macroscopic analysis (EFMA) [63, 45]. This technique enables the analysis of a system’s emergent properties and the identification of the essential factors determining them. Introduced as a “framework for computer-aided multiscale analysis, which enables models at a fine (microscopic/stochastic) level of description to perform modeling tasks at a coarse (macroscopic, systems) level” [63], the method distinguishes itself through the use of a so called microscopic simulator or coarse-time stepper. Through appropriately initialized calls to this simulator, over short times and small spatial domains, macroscopic modeling tasks spanning over long time and large space scales may be accomplished.

Mathematical models capturing the complexity of certain systems cannot easily be built[78], this is why simulations are required. In their turn, these may not always provide “well-founded guarantees about the system-wide behaviour” [120]. EFMA bridges these mathematical and computational approaches, objective recognised as essential for “in silico” research [68]. According to EFMA, the macroscopic description of a system is replaced by the coarse time-stepper, which executes one time step of the unavailable macroscopic model through a three-step sequence:

1. lift: initialising the full microscopic system starting from the macroscopic variables that describe it;
2. run: performing a simulation with the microscopic system over a specified time interval;
3. restrict (measure): computing the macroscopic state based on the system’s full microscopic state.
The technique has two limitations: it needs the identification of all macroscopic variables that define a system’s behaviour, and the definition of an operator for initialising the microscopic system according to the macroscopic variables. The problem may be very difficult for engineered systems, as shown also in the work by Pulack and Stepney [109]: in a complex system, the microscopic and macroscopic levels are expressed in two different languages, using different concepts, so emergent properties are not refinable. To address the first limitation, authors propose a bottom-up approach of aggregating microscale properties into higher level ones. The initialising operator is specific to the system under study and dependent on its complexity.

Authors exemplify this technique through a data clustering case-study [63] – an agent based simulation running a termite pile formation algorithm [116]. Starting from a set of microscale properties (e.g. the positions of all agents and data items, the orientation of the agents), a first aggregation is performed: cluster centres are determined and the system is reinitialised, this time with data items randomly dispersed around the cluster centres, rather than according to their absolute position. Replicate simulations show that the system behaviour is not affected, so aggregation is applied again. The process continues until aggregation is not anymore necessary. In the data clustering case-study, the macroscopic variables that were in direct relation with the microscopic properties were the average distance between clusters and the average cluster size (and its standard deviation). Aspects such as cluster shape or location are found to not affect results qualitatively.

The range of engineering methods does not stop here. Bar-Yam’s [13] evolutionary inspired method is just one example that extends this domain. What remains important is their ability to provide valid solutions for the construction of valid emergent systems.

3.2 Validity

An aspect of central importance to engineering is the ability of any method to steer development towards valid solutions [108]. Computer simulations, a product of such methods, are often suffering in terms of credibility. Causes are multiple, being proportional to the number of stages involved in their reification process. Accordingly, validity is a “task-related concept” [133] that must be considered in close relation to the purpose of the simulation.

3.2.1 Introductory vocabulary

Validity has been considered for decades “the most elusive of all the unresolved problems associated with computer simulation techniques” [101]. The situation was similar twenty years ago [133] and remains equally challenging today [108]. Being a comparative property, validity is considered between a simuland and a simulator. In the case of computer simulations, the simuland can be any real or artificial system, whereas the simulator is the computer program built for simulating it.

Similar to the word ‘complexity’, there are a number of definitions for the meaning of validity. Vandierendonck [139] for example, considers it the “degree of homomorphism between one system and a second system that it purportedly represents”. Homomorphism is chosen rather than isomorphism because abstractions need to be taken into account – the modeled system is less complex than the real one. Schlesinger et al consider validity as a:

“substantiation that a computerized model within its domain of applicability, possesses a satisfactory range of accuracy consistent with the intended application of the model” [125]

In this case, three more concepts need to be defined:

- **domain of applicability**: “prescribed conditions for which the computerized model has been tested, compared against reality to the extent possible, and judged suitable for use (by model validation)” [125]

- **range of accuracy**: “demonstrated agreement between the computerized model and reality within a stipulated domain of applicability” [125]

- **reality**: “an entity, situation, or system which has been selected for analysis” [125]
The development of a computer simulation requires at least three stages, according to Stanislaw [133]: (1) the development of a theory, which is used for (2) the formulation of a model, that is the basis for (3) the construction of the computer program – the simulation. The theory is composed of a set of explanatory statements. Provided they do not need to represent the “truth”, they must point to causal relationships – they would be mere descriptions otherwise. Statements must also be falsifiable, so that the theory may be later on tested (e.g. “it will snow tomorrow” is falsifiable, while “it will or will not snow tomorrow” is not). The model of the simuland can be a physical system or a symbolic representation. Deriving from the theory, the model is essential in making the link between observations and understanding. For some authors, model building is the true goal of science [136]. Finally, a computer program is built through specific programming techniques so that it accurately reflects the model.

According to the three stages of simulation development [133], three types of validity must be considered: theory validity, model validity and program validity. Other authors also include experimentation as a phase in the development of a computer simulation. Sargent [123], for example, proposes a development process (fig 3.7) composed of three different phases: (1) an analysis and modelling phase, followed by a (2) programming and implementing phase and concluding with an (3) experimentation phase. This process seems a refinement of a previously introduced one by the SCS Technical Committee on Model Credibility [125]. What Sargent adds are the explicit validation and verification links between the System and the Conceptual and Computerized Model, emphasizing at the same time the iterative nature of the process through the bidirectional nature of these links. Compared to Stanislaw’s [133] strategy, this process is more comprehensive.

![Figure 3.7: Sargent’s process for developing computer simulations [124]](image)

### 3.2.2 Validity problems

Irrespective of the development strategy adopted (e.g. waterfall, spiral), specific assumptions, abstractions and distortions are introduced with each transition between phases. Errors deriving from theory building differ from those induced by model building, which similarly differ from the programming ones. All of these aspects may affect validity.

Constructing the theory is a step characterised by loss of information. This is actually the goal of theory per se[9]. The amount and type of information that is lost may affect the quality of the model and thus of the entire simulation. This loss must be in accordance with the needs of the researcher and this is why it is said that validity is task-related.

Moving from theory towards the model, formalisation and parametrisation are two main operations that may affect validity [133]. This is firstly due to the rules of the system in which the model is
formulated (e.g. different types of models will be built through Euclidian geometry and through non-
Euclidian geometry). Secondly, the model is a refinement of the theory, so it requires more accurate,
quantifiable definitions – ambiguities must be resolved. Statements that were correct in theory might be
incorrect in the model. Validity depends then on the degree of equivalence between reality and parameter
values and their relationships.

The simulator needs also to be more detailed than the model. The same discussion on equivalence
applies here. In addition, new constraints related to the implementation platform of the simulator may
be introduced. For example, the model may refer to parallel processes, whereas the simulation may
be sequential or run on a sequential processor; time may be modelled as continuous, but simulated as
discrete; long chains of calculations may be affected by rounding errors, etc. For each of these problematic
situations there are solutions, but they need to be acknowledged as potential sources of invalidity.

Another aspect that is influencing a model’s validity is the purpose for which it was built. Sargent
[123] emphasizes the fact that “a model should be developed for a specific purpose (or application) and
its validity determined with respect to that purpose”. If a model is intended to address a variety of
questions, then its validity must be determined with respect to each of them. Thus, the wider a model’s
scope is, the more chances are that it is invalid for addressing a specific research aim.

The same author identifies data validity as another relevant aspect. Similar to the relation between
a complex system and its environment, where the two are not completely separable, valid data allows
the system to express its intrinsic validity. In his well-known process (fig. 3.7), Sargent presents this
type of validity as related to all development stages. As a definition, data validity ensures that “the data
necessary for model building, model evaluation and testing, and conducting the model experiments to
solve the problem are adequate and correct” [124].

Finally, computer simulations have often been contrasted with mathematical treatments, especially
due to the paradigmatic differences between them [30]. Systems composed of equations are explicit
in terms of the underlying mechanisms which lead to their results and provide fully analytic scientific
explanations. While still being subject to theory and model validity, program validity is not a source of
current here – models are evaluated rather than simulated [107]. At the other extreme, “opaque computer
simulations”, lacking a proper documentation, may provide only a “performance of explanation of some
phenomenon” [30]. This is why demonstrating validity is essential.

3.2.3 Validity Tests

All stages of computer simulation development must be valid in order for the final product to be con-
sidered similarly valid. However, distinction has not been made, in many cases, between the various
types of validity, this leading to confusing statements or proofs of validity. In addition, regarding this
subject as dichotomous [102] added to the misunderstanding of the concept. As Schlesinger et al [125]
state, validity requires a “satisfactory range of accuracy”, Stanislaw [133] adding that this is a question
of the degree of homomorphism between two systems. For Sargent [124], this is an economic problem –
able validity might be verified given sufficient time and resources, but this is often too costly.

Validity is therefore a matter of confidence, obtained through tests and evaluations. If a model is
insufficiently accurate on a test run over any set of experimental conditions, then the model is not valid.
However, sufficient accuracy over numerous experimental conditions will not guarantee validity over the
whole domain of applicability.

Simulators can capture the function and the form of a system (by emulating its underlying processes).
Sloman [130] names simulations shallow and deep according to their ability to capture the function and
the form respectively. Overall, the roles of a simulation can be to [133]:

• emulate the input/output behaviour of the simuland. Expert systems [49] are examples of such
  simulations. Not being based on theories, theory validity may not be taken into account here.

• test and demonstrate the sufficiency of a theory [58] or to increase the visibility of a model [114].
  Here, simulations are employed for identifying assumptions that underlie a theory or to reveal a
  model’s inconsistencies. Consequently, program validity is the object of focus, while model validity
  is relevant if the model differs from theory.

• provide extrapolations, e.g. forecasting the future, evaluating unnatural parameter values [151],
  assessing systems on different scales (micro-macro view) [43]

18
In relation to these roles of a simulation, various tests that quantify validity have been proposed. We present here a selection of well known examples, while reviews such as [127, 87, 124] detail further analysis techniques:

- **face validity** implies human assessment of a simulation’s results. Due to its subjectivity, it was criticised as meaningless [54]. While useful for debugging, validating based on such a method would imply that the simulator’s output could be predicted apriori, hence simulating would be not necessary.

- **historical or event validity** [73] refers to the comparison between real events and simulation predicted ones, based on historical data. Through this test theory validity is addressable only if two sets of data are available: one for the simulator’s development and one for the validity assessment. Provided this happens, the test reflects the combined effects of theory, model and program validity.

- the **Turing test** [102] sees the system and the model’s outputs being compared by a set of judges. Without knowing which result corresponds to which source, the judges give a measure of similarity, based on which validity is defined. However, the test includes a degree of uncertainty due to the human factor – judges must be validated too, while a processing of results prior to their assessment may add further biases. Consequently, the test is considered weak, not serious [117].

- **internal validity** measures a simulator’s variation given a number of starts with the identical parameter values [73]. Low variation in results would point towards simulator validity. However, this is “nonsensical” [133] as stochasticity is not a measure of validity.

- **hypothesis validity** is a measure introduced by Hermann [73], who states that the pattern of correlations between variables in the simuland should be similarly reflected in the simulator. All three types of validity are addressed through this test.

- **covariance matrix comparison**: program validity can be measured by comparing the covariance matrix of the model’s variables to that of the simulator’s, provided the model is completely describable in mathematical terms. The comparison can be undertaken through covariance structure analysis programs such as LISREL [82]. Constants and redundant variables may be thus identified. A more detailed discussion of covariance structure analysis can be found in [17].

Other methods are promising but similarly challenging. For example, Sargent [123] mentions “comparison with other models” as a technique through which the results of a computer simulation are contrasted to those of a valid analytical or computer model. However, this leads to a recursive question referring to the facts on which the validity of the latter is supported [7]. Especially in the case of complex systems, there is a need for less ambiguous tests. To be mentioned are also tests such as extreme conditions tests or traces, that are detailed in [124].

The aspect of simulator verification is considered a separate topic and will not be treated here.

### 3.2.4 Development strategies

In science there is usually a bigger concern for theory rather than the engineering aspects of tools that enable their investigation [133]. Similar is the consideration given to scientific computer simulations. Consequently, a series of steps may be taken in order to obtain validity.

Sargent [124] specifies a series of eight such steps, spanning from the pre-development phase up to the post-experimental one. He considers that firstly the validation approach and a minimal set of validation techniques should be selected. The amount of accuracy required from the simulator’s outputs is also a factor that is later on associated to the assumptions and theories underlying it. Having tested these assumptions and theories, development may proceed. Validation tests should be undertaken with each new version. Finally, the simulator’s validity should be documented and a schedule for its periodic review set up.

Other authors provide more a detailed guidance. Stanislaw [133] for example, recognizes that simulators should be “fully debugged, and all assumptions inherent in developing the program must be exposed and evaluated”. Restrictions imposed by the implementation infrastructure may be subtle [1], but they still must be accounted for. Even in the case when they seem trivial, they should be exposed. If
necessary, sensitivity analysis may be conducted for identifying the sections of the program that must be “specified with care” [133]. In terms of applying validity tests, mean values for the dependent variables should be at the very least compared (e.g. through a t-test). Because variances should be compared as well it is then advisable to compare the entire covariance matrices.

Factor analysis is another method proposed by Cohen and Cyert [39]. In this case, the factor loadings of the two systems are contrasted. For Conway [41], validity is dependent on the system’s stimuli, thus caution is required when extrapolating results. If the mechanisms underlying the modeled reality are truly captured, extrapolation should not be problematic. Otherwise, if the simulation is shallow, extrapolating is problematic. This goes in line with Sargent’s [123] observations on the applicability of a model and the necessity to develop it purposefully.

Other proposals include the construction of multiple models where by exposing them more variations, their number is gradually reduced to a small set of viable ones. Science advances by disproof [133]. Tuggle and Barron [137] propose a four-phased method for quantifying the performance of simulators based on the use of three metrics: the Theory Content (TC) – the ‘size’ of the model, Explanatory Power (EP) – index of robustness, and Explanatory Yield (EY) – index of parsimony. Having firstly defined a universal set \( U \) of phenomena of interest, models are developed and the three mentioned quantities calculated. Tuggle and Barron’s method allows them to answer questions such as “how long should a given model be retained?”, “when should an existing model be made more complex?”, “what variables are most useful to add to a model”, “when should the model development process cease?” and finally, “should the resulting model be accepted or rejected?” [137].

### 3.2.5 Critique

The above mentioned tests and development strategies provide a wide range of means through which validity, with its relative meaning, may be enforced. It is clear that this process is multi-phased and that validity should be reflected in each development stage. From the above opinions, general or absolute validity seem two unfeasible notions, so similar to the term ‘complex’, greater specificity needs to be added when using them.

With respect to the proposed validating tests, many of them have already been criticised as being too subjective or inappropriate. The human factor in assessments such as face validation or the Turing test is considered one of their main drawbacks. However, the use of ‘domain experts’ is often cited. Sargent [124] invokes them for the validation of (conceptual) models, Alexander [3] too and the list does not end here. In doing so, the level of confidence that is expected from the experts or what is the perimeter of insight that is required from them are not clearly defined. Also, if validity is assessed in a quantitative and transparent way, should not attention be similarly directed towards the selection and the use of the domain expert?

Assumptions, abstractions and distortions are often mentioned as sources of uncertainty that influence validity. All of them are necessary to a certain extent, as any model must be simpler than its simuland. However, the line between these factors is not always clear. Motivating the introduction of each assumption or distortion and showing how they are overall transformed in simulator abstractions, would also describe the validity of the development process. Sargent is one of the authors who emphasise the necessity to document simulations. In his view, this should include “specifics on the tests, evaluations made, data, results” [124]. As such, the reasoning behind the structure of the model would not be captured.

Another observation is that most tests preclude the existence of a ‘reality’ that may be evaluated and compared with the developed models and simulations. However, this does not apply to domains such as Artificial Life (ALife), where the ‘reality’ may be a totally virtual construct.

Overall, the literature surrounding computer simulation validity reflects the need for a more consistent application of techniques, provides tools but lacks in comprehensive case-studies that use them throughout.

### 3.3 CoSMoS contributions

The CoSMoS project was initiated in October 2007 and is planned to end in March 2012. Bringing together scientists from the Universities of York and Kent, this project aims to “build capacity in generic
modelling tools and simulation techniques for complex systems, to support the modelling, analysis and prediction of complex systems, and to help design and validate complex systems” ¹. Its core research objective is to develop a modelling and analysis process for the study of complex systems, while making use of a developing massively parallel and distributed simulation environment in order to support CoSMoS’s case-study driven approach. The York group is mainly responsible with the first objective, whilst Kent with the second.

Research prior to CoSMoS relates to the engineering of emergent, complex systems, part of which has already been presented in Section 3.1.5. In parallel with these efforts, attention has been directed towards the use of process-orientated programming (POP) as a paradigm that constitutes an “effective approach for complex systems simulation” [122]. As many complex systems are composed of concurrently interacting components, through the process-oriented approach these can be directly modelled as processes, resulting simulation showing a high degree of parallelism. CoSMoS simulations are developed mainly through occam-π, a programming language implementing the CSP [74] and π-calculus [97] paradigms.

In [110], Polack et al discuss about downward causation and a strategy of rule migration, through which this may be implemented. Considering the layered architecture of complex systems, this method implies the transfer of low-level rules to the higher-level layer. Similar to a subsumption architecture [29], the high-level layer controls the execution of low-level behaviours. The method is applied theoretically to Game of Life gliders and practically to an artificial blood platelet case-study, authors observing that the approach allows the higher layer to be exploited for more natural control laws, while allowing simulations to be easier extended with new features. Supporting these claims was the comparison made between a CA and an occam-π simulation [119] developed especially for the blood platelets case-study.

The initial CoSMoS approach to modelling and simulating was introduced in [62] (fig 3.8). The approach consisted of a five-phased iterative process of developing a computer simulation, spanning from the incipient step of selecting the subject of study, to the analysis of results. Each phase concluded with the provision of a deliverable: the Domain Model enclosed relevant information about the Domain, obtained from certified sources such as experts, literature or further experiments and observations; this in turn was used for developing the Software Model that led to the creation of the Simulator; having run the necessary simulations, an Analysis Model would finally provide a structured view over the ‘reality’ captured by the modelling and simulating process, in contrast with that expressed by the Domain. The paper describes how the CoSMoS approach applies to developing a simulation for studying the formation of auxin channels in plant stems, leaving the construction of the analysis model as further work.

Figure 3.8: The initial CoSMoS process (from [62])

Continuing the trend of biological case-studies, the work of Andrews et al [7] introduces research aiming to apply software engineering validation and verification techniques to the development of complex systems models and simulations. Authors criticise traditional mathematical and agent-based modelling

¹ Retrieved from www.cosmos-research.org
techniques, which either lack in realism or are specialised in dealing with particular types of systems, and then discuss high-integrity and critical systems methods for developing “scientifically-valid simulations”. Emphasizing the need for transparency in terms of assumptions made during the development phases, the paper provides practical examples of conflicting situations that may be prevented when these assumptions are made explicit. More explicitly, the authors find a subtle, but potentially invalidating discordance between one of the research hypotheses (the lymph node’s lymphocyte volume is dependent on the dilation of the HEV) and an implementation aspect (homogeneous HEV environment). Finally, an extension to Sargent’s process [123] is proposed in order the accommodate need for identifying environmental factors that should be captured by the conceptual model.

Interest in engineering simulations is also expressed in works such as [108] and [111]. In [108], authors provide a brief review of two state-of-the-art techniques for undertaking computer modelling, Reactive Animation [53] and PEPA [33], and note what desirable aspects of complex systems are expressible through them. Proposing a list of “requirements for complex emergent systems design”, authors observe that the above methods do not address issues such as context representation (space, time or environmental features), emergence, dimensionality and scale. In addition, modifiability and understandability are presented as two more desirable features of engineered models of complex systems, the paper providing guidelines towards meeting the proposed requirements. Polack et al [111] aggregate the work provided by [108] and [7], into a coherent claim for a CoSMoS approach to modelling and simulating.

In parallel, the simulating infrastructure has been enhanced so that it allows execution across computer grids [122]. This process of distributing it necessitated architectural modifications which led to new artifacts being introduced. Design patterns for developing simulations with different types of topologies (e.g. continuous space and network space) were extracted [6] from the analysis of different case-studies [22].

3.4 Summary

This chapter has focused on the challenging domain of engineering complex systems. We started the discussion by emphasizing the close relation between engineering and science, and the way emergence represents in fact the link between them. This topic was continued by a brief presentation of difficulties raised by engineering complex systems and various methods provided by the literature. Here we looked both at classical techniques, extending the scientific method, to more advanced ones coming from the field of agent-oriented software engineering.

An essential requirement of any development method is to provide valid solutions, according to the specifications and the purposes of the engineered systems. We addressed then the concept of validity, detailing the problems that it entails, together with literature examples of tests and strategies associated with it.

Finally, we looked at relevant work published under the CoSMoS project. We note that consistent contributions have been made to the domain of engineering simulations, both from a methodological point and a technological point of view. While process-orientation pervades the case-studies employed by CoSMoS, and various design patterns are starting to be identified, the CoSMoS approach to scientific simulations requires further work for its crystallisation.
Chapter 4

Plant ecology and complex systems

4.1 Ecology and the “Holy Grail”

Ecology can be understood as “the scientific study of the distribution and abundance of organisms and the interactions that determine distribution and abundance” [16]. Deriving from Haeckel’s 1870 definition [70], the above description includes both the subject matter of ecology – the study of ‘distribution and abundance of organisms’, and the objects of it research – the ‘interactions’ between biotic (living) and abiotic (environmental) factors.

Ecological research addresses issues on multiple scales, mainly from individual organisms to populations and communities. Larger aggregations, such as ecosystems and even the biosphere are also accounted for. This is why, as a science, ecology has specialised into a set of nine sub-domains: ecophysiology, behavioral, population, community, ecosystem, systems, landscape, evolutionary and political ecology.

The long term interest, also known as the “holy grail” of ecology [89], is that of uncovering the relations between environmental conditions, species characteristics and community composition. In other words, identifying the rules through which the “distribution and abundance” of organisms are determined by the environment and by the biological characteristics of organisms. Such knowledge is essential. The study of both the history and future of ecosystems relies on it.

Last, but not least, engaging the “distribution and abundance” research quest allows ecologists to address the most pressing environmental problems of the present. One such example is the loss of biodiversity. During the last decades, this decrease has had strong effects on human communities: biota is not only a source of food, fuel and fiber, but it also acts in processing nutrients, sequestering harmful chemicals or mediating climatic processes [91]. Managerial decisions rely on predictions of potential effects.

From this point onwards, we shall focus on plant (population) ecology as it is the relevant sub-domain for our work. Ecological concepts shall be first introduced and the state of modelling and simulating in plant ecology detailed. The chapter will conclude with a summary and a set of considerations.

4.2 Introductory plant ecology concepts

Ecology was previously said to focus on individual organisms, populations and communities. Individuals are studied in terms of their inter-plant interactions, mediated by the environment. Populations are considered as being formed by plants of a single specie, while communities are composed by two or more populations [16].

Before discussing the state of computer modelling and simulating in ecology, a set of core ecological concepts will be introduced. This will be done by taking into consideration the three levels of organisation earlier described.
4.2.1 The individual level

Plant processes are closely connected to the flow of resources throughout the environment [55]. Carbon is captured from the air, or obtained from below-ground mycelia, and released into the ground. At the same time, water and nutrients are \textit{uptaken} (extracted) from soil and dispatched to the various parts of the plant. In fact, the whole plant can be regarded as a community of units competing for resources [150]. In figure 4.1 we present Wu et al’s [148] more detailed view on the interaction between plants and the environmental flow of resources.

![Figure 4.1: Plant physiological processes and resource cycling (from [148])](image)

Plants possess a level of plasticity that is not common to animals. Resources can be used for growth, reproduction, or stored for later use. When resource levels are low, plants are able to adapt: surpluses previously stored are redirected towards more important areas; other parts are discarded altogether (e.g. leaves); fecundity can also be decreased. Consequently, an abstract representation of plants may see them distributing resources towards three main parts [25]: a \textit{structural store}, a \textit{structural compartment} and a \textit{surplus compartment}. The structural store reflects the fixed structure of a plant (e.g. the shoot). Resources that are \textit{allocated} for \textit{reproduction} can be considered as being stored in the structural compartment. Finally, the surplus compartment relates to resources that exceed plant uptake necessities and can be remobilised for reproduction, when required.

A process fundamental to ecology is reproduction. Plants reproduce either sexually, through seeds, or asexually. There are many ways through which seeds get \textit{dispersed}, the principal ones being gravity, wind, animals, force or water. Dispersal can happen over shorter or longer distances. This latter type is more difficult to study, as experimental data is harder to collect; at the same time, it is critical – rare events can influence the rate of population spread [38].

4.2.2 The population and community levels

When referring to plant-to-plant interactions, \textit{competition} is one of the fundamental processes to be considered. Situated in an environment of limited size and having access to limited amounts of light, water and nutrients, plants are viewed as individuals engaged in a continuous competition for resources. This eventually drives their survival and evolutionary fitness [72]. More recently, studies of plant \textit{facilitation} have emerged. Facilitation is a process complementary to competition, plants having positive influences on one another e.g. “nurse” plants sheltering plants situated under their canopy, from intensive sunlight or other stress factors. A thorough review of the developing field of plant facilitation can be found in [28].

Different species impact differently on the environment, and through it, on the demography of the other species sharing it. Thus, the number of species populating an environment and the abundance of each of them are relevant aspects in the study of communities. In ecology, these two aspects are known...
as the species richness and relative species-abundance and form the definition of biodiversity. There are many ways through which biodiversity influences community functions. For example, plant productivity – the total amount of resources obtained by plants from their environment, is higher in more diverse communities [24]. These communities can also resist better to environmental stress.

The study of the distribution and abundance of organisms was initially motivated by the search for universal rules and patterns that would apply to all species [112]. One example of such a pattern, a landmark in ecological research, is the species-area curve (fig. 4.2). Made popular by Preston’s 1962 work [113], this log-normal curve shows that species-richness grows logarithmically with the environment’s size. Preston used the power law for formalising this relation \( S = kA^z \), where \( S \) represents the number of species, \( A \) the area size, while \( k \) and \( z \) are constants. Even if the relation is not absolute (the slope \( z \) depends on environmental factors [44]), the pattern is a strong indicator of equilibrium within a community.

![Species-area Relationship on Arithmetic Axes](image1.png)

![Species-area Relationship on Log-log Axes](image2.png)

Figure 4.2: Species-area curve (from Wikipedia)

In addition to the species-area curve, ecological studies also refer to the relative species-abundance curve. Communities are often composed of a small number of common, highly abundant species and many rare species of low abundance and. The relationship has been shown to also be log-normal [112].

Finally, it is worth mentioning that the quest for a general theory of biodiversity is still ongoing [104].

### 4.3 Models and simulations

Coupled with field work, modelling has played an essential role in ecological research. If field experiments have provided data, models allow ecologists to extract knowledge, make predictions and inform decision
makers. Similar to other sciences, two main types of modelling paradigms have been used in ecology: equation-based modelling (EBM) and individual-based modelling (IBM).

Before describing the two types of modelling technique, it is useful to reflect on the motivations behind the development of a particular modelling technique in general. Grimm [68] discusses about two such types of motivations: pragmatic and paradigmatic. A pragmatic motivation sees the development of a modelling technique in order to account for aspects that had not been accessible to others e.g. individual variability is not considered in EBM. Paradigmatic motivations concentrate on the serious problems of existing modelling techniques, that are liable of providing misleading results. Theories resulting from the use of state variable models bear deficiencies which motivate IBM in a paradigmatic way.

In the following part of this section we are going to discuss issues of stochasticity, spatiality and scale, referring to these two modelling techniques, followed by a presentation of two ecological protocols that may be of relevance to the wider literature of computer modelling and simulating.

4.3.1 EBM and IBM

Early efforts were aimed at developing a conceptual framework for the understanding of ecological processes and properties that would be applicable to all populations under study [68]. Looking at high-level community properties such as resistance, persistence, resilience, extinction, ecologists developed mathematical models that made the link between them and aggregate population features. Consequently, EBM are said to be top-down, the benefit of such an approach being that models offer an integrative view over population processes. This view can be directly used in the development of new theories.

Research using deterministic systems of low dimensionality had to discard essential aspects of reality such as stochasticity and heterogeneity [92]. The important aspects of spatiality and locality remained also unrepresented. This lead to a shift of focus towards IBM and simulations “that treat individuals as unique and discrete entities which have at least one property in addition to age, that changes during the life cycle e.g. weight, rank in a social hierarchy, etc” [78].

IBMs are bottom-up models that, starting from a mechanistic definition of individuals, may be used for studying community level processes and properties. From this perspective, IBM and EBM are two complimentary paradigms, addressing similar research questions from completely opposite angles.

One fact that needs to be recorded is that, although IBMs are agent-based models (ABMs), they do not refer to the same systems as MAS do. Bousquet and Le Page [23] state that, while IBMs are oriented on the study of heterogeneity and usually involve simple types of individuals, MAS are socio-oriented, focussing more on decision-making processes and social interactions.

4.3.2 Stochasticity

EBMs are deterministic in their nature. The logistic equation presented in Section 2.2.2 constitutes such an example. One way through which this limitation is addressed is through the use of Monte Carlo simulations [21]. The general principle behind this approach is to perform multiple evaluations of the same deterministic model, using randomly generated inputs. Aggregating results provides the solution to the researched problems. Critically though, the Monte Carlo method relies on the use of good random number generators and on sufficient simulations in order to minimise sampling effects.

IBMs may easily implement stochasticity, due to their engineered architecture. However, the freedom of implementation comes at a price. Not only are results influenced by the type of used stochasticity, but also by the way it is applied. If the former aspect relates to the unrealistic probability distributions that may be used for sampling random numbers, the latter issue refers to the way random number generators are used. Mize [98] notes, for example, that “in many simulation models, a single sequence of random numbers is inadequate”. The danger is to develop “ill-behaved models having a much larger variability than is possible with multiple sequences”.

4.3.3 Spatiality

EBMs do not naturally account for the spatial dimension. Abstracted away due to both methodological and strategic reasons (i.e. capturing space through equations is a difficult task, or even unwanted), lack of spatiality was one of EBM’s major criticisms. As such, Durrett and Levin [50] show that “being discrete (and spatial)” leads to results that are qualitatively different to non-spatial models. This difference is
noticeable when individuals are in competitive relations [5], whereas facilitation points towards similar results.

A step forward in bringing mathematical models closer towards spatiality is the introduction of spatial logistic equations. Thus, IBMs can potentially be approximated deterministically through the description of first and second spatial moment dynamics. Applying this technique to single species population growth, Law et al [90] show that the spatial logistic equation overcomes the unrealistic behaviour of its non-spatial homologue, being able to capture events such as extinctions and different patterns of population growth. However, the method uses advanced mathematics that make analysis more difficult.

4.3.4 Scale

Another relevant aspect differentiating simulation models is the scale that they are addressing. Throughout this dissertation, the topic of scale has already been addressed (see sections 3.1.2, 3.1.5, 3.3). Here we will note that in ecological research, addressing the scale issue is equally important.

Levin et al [92] mention addressing complex systems and scaling up from small spatial regions to large ones as the two main modelling challenges. According to them, not only scaling up is problematic, but also scaling down - mathematical models had resolutions of hundreds of kilometers, whereas natural or managed systems could have been much smaller. Consequently:

“the challenge, then, is to develop mechanistic models that begin from what is understood (or hypothesized) about the interactions of the individual units, and to use computation an analysis to explain emergent behaviour in terms of the statistical mechanics of ensembles of such units.”

Jeltsch et al [81] emphasize the fact that the application of plant modelling is limited due to the compromise between the necessary resolution and the need for generality. Grimm [68] also discusses about scaling-up and scaling-down models. In his view, scaling-up means to decrease a model’s resolution, abstracting away details, while scaling-down means increasing the model’s resolution. Accordingly, scaling-up is characteristic of top-down, EBM approaches, while scaling-down relates to bottom-up IBMs.

4.3.5 Protocols

Here we will briefly describe two protocols coming from the domain of ecology, that are contributing to modelling and simulating in general.

Reconciling EBM and IBM

Fahse et al [56] propose a protocol for reconciling EBM and IBM. The two techniques seem incompatible due to their orientation towards different scales: while EBM focuses on population dynamics, IBM addresses physiology or behaviour. In other words, there is an incompatibility between the macroscopic system level addressed by EBMs and the microscopic one simulated by IBMs. This separation means that slower macroscopic processes can be treated separately from faster microscopic ones, hence a simulation oriented version of the adiabatic technique [71] may be applied.

To reconcile the different targets of the two modelling paradigms, the authors propose a protocol through which basic macroscale parameters (e.g. equilibrium population size or return time in the case of population dynamics) may be extracted from an IBM, thus allowing the two approaches to address the same scales. This is exemplified through a case-study on nomadic birds.

The protocol consists of four steps, customised in the following list for the study of the overall population growth rate $f$:

1. identify all microscale processes $p_1, p_2, \ldots, p_n$ that affect the macroscale variable $f = f(N, p_1, p_2, \ldots, p_n)$, $N(t)$ being the population size at time $t$

2. assess if they can be grouped under a key process $P$ (the interface), such as $f = f(N, P[p_1, p_2, \ldots, p_n])$
3. if $P$ varies much faster than $N(t)$, apply the separation of timescales – deactivate demography, calculate the potential rates of death and birth for $P$’s equilibrium value $P^*(N)$, thus $f = f(N, P^*(N)) = f(N)$

4. use the function $f(N)$ for calculating dependent variables

The results obtained through the protocol come very close to the ones obtained through a full mathematical model. However, two aspects must be noted: (1) the approach depends on the existence of an interface $P$ and (2) on the clear separation of timescales. Also, the method is still tailored to ecological problems and, to our knowledge, is yet to surface the general literature of computer modelling and simulating.

### Overview, Design concepts and Details (ODD)

Grimm et al [69] propose a standard protocol for describing individual-based and agent-based models. They start by identifying two main problems with the description of IBMs: (1) the lack of a standard protocol for describing them and (2) the unstructured way in which they are currently presented. More specifically, equations, rules and schedules are not properly organised, descriptions being more verbal rather than formal. Consequently, they propose the **Overview, Design concepts and Details (ODD)** protocol, which consists of a general template for describing IBMs involving a clear separation between its mathematical and verbal constituents.

![Figure 4.3: The ODD protocol (from [69])](image)

As depicted in figure 4.3, the protocol is formed of seven sequential elements that are grouped in the Overview, Design concepts and Details blocks. The Overview introduces the purpose and overall structure of the model. Thus, the “model’s focus, resolution and complexity” can be easily identified. In the Design concepts element, the way general complex adaptive systems concepts (e.g. emergence, interactions, stochasticity) are implemented in the model is described. Finally, Details contains all information required to re-implement the model and run its characteristic simulations. In case of space constraints, it should contain links towards online appendices or other publications.

The authors provide comprehensive descriptions of all these elements, together with “hints” and a case-study to exemplify their use. While still being a compromise between generality of scope and specificity of modelling purpose, the protocol is deemed as facilitating the writing of model descriptions, which became more comprehensive and easier to understand. In addition, it has been showed to be applicable to bottom-up simulations in general, e.g. grid-based models. Overall, the protocol was refined through 19 models that completely implemented it.

### 4.4 Summary

This chapter has introduced the domain of plant ecology, focusing on topics related to modelling and simulating. Starting with a description of the wider domain of ecology, plant ecology reflects the search for links between scales, commonly named as the “holy grail”. Many modelling techniques have been
employed here, but current interests are biased towards IBM. As such, the need for a tighter integration with EBMs [68, 56], for improving the way modelling is conducted and applied [81], in addition to standardising the way simulations are presented [69] are central issues.

IBMs are subject to a multitude of spatial representations. Models such as grid-based (CA), tessellations, distance models (e.g. fixed-radius-neighbourhood – FRN, zone-of-influence – ZOI, or ecological field – EF) in addition to more recent approaches such as 3D and particle-in-cell (PIC) models, define the palette of representations that may be readily used. All these types of spatial representations have advantages but also constraints. For example, FRNs are easily parametrisable, but neglect the configuration (the absolute positions of individuals) as they rely on neighbourhood densities; ZOI models take into account the configuration, allowing also the study of plant resource sharing and of gradients and heterogeneities within the spatial environment, but they “cannot provide a final proof whether their processes are really the ones that drive” [18] the observed patterns.

A good review of these models is provided by Berger et al [18]. They identify the fact that essential biological processes such as modification, adaptation, and below-ground competition, are poorly captured by these techniques, hence models must be more mechanistic. This will induce higher knowledge and data requirements for their parametrisation, as they will be more complex. To address this issue, authors advocate for “a research strategy that integrates field studies, experiments, and modelling” and emphasize the necessity for improvement of model parametrisation, validation and selection.
Chapter 5

Proposal

This dissertation has started from the concept of *complexity* and the challenge of unravelling it. While this goal is not attainable, we turn towards the study of complex systems – the embodiment of ‘complexity’. Here too there are many issues to be addressed and the scientific method can offer but a partial guidance. In addition, the means through which research is undertaken are just as important as the subject of research itself.

Modelling and simulating are two research techniques that have been integrated in the toolset of many sciences. Here, through the use of a plant ecology case-study, we intend to contribute to the study of *validity* in the context of complex systems models and simulations, and to the development of the CoSMoS approach to *in silico* research.

5.1 Plant ecology case-study

The plant ecology case-study is aimed to support the refinement of the CoSMoS approach, while contributing to the list of large-scale simulations that may be endorsed by groups and sciences requiring such research tools. These are the two main project requirements for this case-study. Its starting point is the work of Bown et al [25], which proposes an individual-based model and simulation for the study of plant ecology issues. We will use this work and re-engineer the simulation in a new, process-oriented way, that complies with large-scale simulation requirements.

The CoSMoS project investigates a principled way to building computer models and simulations. Similar to Sargent’s view on the “Simulation World” [124], CoSMoS addresses a conceptual level that preceeds any practical undertakings. Research starts from a Domain of interest and proceeds with the definition of a Domain Model that crystalises the knowledge about the Domain; this is then transformed into a Software Model which is used for developing a Simulation; the simulation can be then used for experimentation.

In our case, we are applying the CoSMoS approach to an already existant simulation. Consequently, we need to create the models that led to it: Bown *et al*’s [25] work must be firstly reverse-engineered and the Domain to Software Model [62] obtained. Development will be continued through a re-engineering process which will deliver the new, occam-π simulation. This simulation will then be transformed into a distributed applications, so that large-scale experimentation will be possible across computer grids. In addition, the availability of two computer simulations employing relevantly different computing approaches will enable comparative studies to be conducted.

In parallel with the above, focus will be placed on the concept of *validity* in the context of complex simulations. Benefiting from the collaboration of the domain expert who authored the original simulation model, the plant ecology case-study is intended to provide examples of structured arguments of validity that apply to complex systems. To support this process, investigations will be made into using the Goal Structuring Notation (GSN) [84] for representing such arguments.

Stepping forward from scope of Bown *et al*’s [25] research, more advanced ecological scenarios will be addressed. This will imply an increase in the model and simulation complexity, this being directly relevant to the study and maintenance of validity. Along the way of extending the occam-π simulation, design patterns will be extracted and added to the CoSMoS collection of reusable concepts and structures.
5.2 Research objectives

The rationale for starting this case-study is represented by the need to re-engineer Bown et al’s [25] simulation. This is the driving force that will enable further studies on validity and argumentation. The first objective is then to:

A. re-engineering an existant, small-scale C++ simulation into a large-scale one

The thread of research will build on this re-engineering process. Overall, we aim to make a threefolded contribution: firstly, to computer science, through the work on re-engineering and argumentation; secondly, to science in general, through the work on refining the model and simulation; finally, to ecologists, through the large-scale simulation and its results. Consequently, other main research interests are to:

B. introduce argumentation techniques to the domain of complex systems models and simulations
   
   • provide a structured argument that our re-engineered product is appropriate

C. develop an understanding of validity in the context of complex systems models and simulations
   
   • link simulation results with argument solutions (e.g. assumptions, abstractions)

As a property, validity has been said not to take simple true/false values (it is a task-related concept), however this is the principal way in which it has been applied. Due to the difficulties encountered in predicting complex systems behaviour, and influenced by the lack of widely acknowledged and trusted validity tests, validity is even harder to address – it might not even be the term we are looking for. Hence, an understanding of the concept of validity, in the context of complex systems models and simulations, is required.

We start from the assertion that it is possible to demonstrate that a simulation is an adequate model of a complex system, at least to the satisfaction of the people involved in its development (here, the plant ecologists and the simulation author). The research recreates and extends a simulation that addresses specific scientific questions in relation to a trait-based plant model [25]. Here, the simulation validity has not been addressed directly, but was implied through the qualitative results that it obtained and the trusted sources of information that had been used for its construction. We want to take a step forward on this path: at each development stage, we will consider how validation (and verification) can be applied, considering both scientific and software engineering issues.

Scientific research is based on a series a transformations from a domain problem to an analytic or synthetic solution. Each aspect of this process is relevant in terms of the outcome validity. Relevant here is not only the input-output relations of a simulation model, but also the process through which it was obtained, transiting from theory to model and then simulation.

Through the use of argumentation techniques, so called argument maps may be formed. So far we have focussed on capturing argument structures in GSN. These top-down structures consist of a main claim which is supported by sub-claims and the information solving them. However, the research needs to assess how best to capture and verify the arguments that support the development: does GSN contain the right argument structures for this context? It is already apparent that explicitly linking results with argument solutions (e.g. assumptions, abstractions) is a non-trivial task, but one that has the potential of providing new insights over the scientific method and the modelling process.

Developing a large-scale simulation for performing research in ecology will be relevant to both the argumentation and validation process. The research findings will be generalised and reflected back to the requirements of the CoSMoS project, and the wider context of simulation validation.

5.3 Research plan

In order to address the above objectives, a two-year plan presented in figure 5.1 is proposed. Note that the execution order for each of the specified tasks is not sequential, nor their duration is fixed.

The proposed research plan includes implicit additions to the literature review presented here. For example, there is a need to continue reviewing conventional approaches to simulation and validation,
and consider argument modelling more fully. In addition, we will also need to review strengths and weaknesses of simulation approaches (and in particular, of occam-π simulation). The plan presented in figure 5.1 refers to these activities of documentation (in terms of justifying what has been done), and the preparation of publications (articles – in terms of related work), as well as various explorations of alternatives.

5.3.1 Equivalence argument

As will be detailed in the following chapter, work has already been done on developing an argument of equivalence between Bown et al.’s [25] original simulation and the newly developed one, this effort being synthesized in the [65] article. However, this argument is not completely solved. Experimentation will be carried out to support the argument’s claims – more specifically, the experiments detailed in [25] will be replicated. We shall define the confidence level of we require and then assess if results fall into the calculated confidence interval.

Following experimentation, we shall proceed with an argument expansion phase. Justifications will be added where necessary (e.g. for the chosen strategy: a different one could have been proposed – optimising the original simulation so that it would be comparable to the occam-π one, thus our current proposal needs to be justified). It is clear that justifications could be considered top claims themselves, hence it would be an interesting exercise to develop this argument’s elements until all of them would have solutions.

As stated before, a different strategy might have been used. This would imply that more equivalence arguments can be constructed. One of our interests related to the use of argument maps is to identify ways of evaluating them and deciding upon criteria for sorting them according to their strength. Having two or more arguments for the same top claim would also give us opportunity to make further observations with respect to the structure of arguments, and possibly extract argumentation patterns.

We shall finish this phase through the completion of a technical report that will condense all our work related to equivalence arguments.

5.3.2 Validity argument

Prior to developing an argument of validity for the new computer simulation, more effort shall be invested in understanding validity and assessing if it is the right concept to use. We will require further documentation on the literature of validity tests, critical systems and statistics.

We would like to develop an argument having ‘validity’ as its top claim. The equivalence argument previously presented is a combination of quantitative and qualitative evidence, while the top claim is bound to be subjectively accepted. It would be possible to develop an argument supported in a purely quantitative manner, for example in order to address certain subgoals – this would not guarantee that the top claim would be objectively proven (counter arguments can always be raised), but it would provide a solid foundation for further extensions.

Claiming validity of a simulation requires more then verifying the correctness of its implementation. Model and theory validity need also to be evaluated. This implies that the CoSMoS artefacts (Domain to Software Model) have to be reviewed: clear descriptions for each type of model are required, in addition to making explicit the way they relate. The Domain Model, for example, is constructed based on knowledge obtained from relevant sources of information (according to the Domain). This knowledge is processed in order to obtain the model, various operations being involved: assuming, simplifying, distorting, adding, removing. The operations and data involved in this process of obtaining the simulation are highly relevant to investigating validity.

In order to develop an initial validity argument, we need to clarify the scope of the simulation, and this requires interactions with its author (Dr Bown). The model [25] is generic, but at the same time introduces certain distortions (e.g. clonal reproduction through seeds – which require fertilisation, in nature) which don’t qualify it as directly relevant to various plant systems. We need then to scale down this scope, possibly by starting from the opposite claim (the model is not valid) and extending the scope until the first support for invalidity is found.

We shall finalise this phase with the writing of a research article, introducing structured arguments of validity to the literature of complex systems simulations.
5.3.3 Analysis

Our long term goal, in this context, is to provide an understanding of the impact modelling decisions will have in terms of ‘validity’. At this stage, we should have the initial argument of validity constructed (as comprehensive as it may be) and a clear registry of the operations, models and data involved in the research. We would like then to explicitly link the interpretation of results with the validity argument. This may seem self-referential: we consider a convincing validity argument as being based on experimental results, but at the same time we are aiming to link the results’ interpretations with the argument’s structure. However, the difference stands in the term “interpretation” (see next section).

Simulations should be constructed according to a clearly scoped purpose [124]. For our research, this purpose will be referenced by the validity argument – we would be then arguing that the simulation is validly addressing it; we would not be arguing general simulation validity.

For our validity we will need to perform sensitivity analysis (SA). SA addresses the relations between the variation of a simulation’s outputs and the different sources of variation in its inputs. Complex systems however are more than complicated (defined by many parameters), they are potentially on the edge of chaos [95], hence some emergent behaviours may be exhibited only under certain (internal or external) conditions. Consequently, we need at least to perform a global sensitivity analysis and for this we will need to further document – relying only on brute force methods is likely not to be sufficient. Also, analysis has to be focused not only on the input variation, but also on the modelling operations described before. This possible opens up an even wider research space e.g. we could test the influence of the spatial representations – there are many alternatives to grid-based space [18].

If different spatial representations may lead to different emerging effects, even in simpler scenarios [50], the other assumptions and abstractions involved in the development of an agent-based model are relevant factors. However, not every assumption or abstraction may have the same impact over the simulation results, and here is critical to identify the aspects that support validity or lead to invalidation.

The final objective for this phase is to extract from these results, general (modelling, argumentation, validation) patterns. We may contrast this work with other case-studies employed in the CoSMoS project. In order to facilitate this process, we have described our simulation according to the ODD protocol [69] – we are also looking at extending it with an explicit and separate ‘validity’ section.

How do we properly associate simulation results with validity arguments remains to be researched, especially when interpretations relating to wider or different domains are sought.

5.3.4 Argumentation

Argumentation has been mentioned as critical to our approach. We intend to develop alternative validity arguments and draw conclusions on the appropriatedness of various structures. It is important to discriminate between a strong argument and a weaker one. However, as currently there is no metric for assessing arguments, and the GSN notation lacks such a numerical dimension, different qualitative approaches must be undertaken. We must at least be able to order arguments, from strongest to weakest – the proper criteria for ordering will be investigated.

In addition, extracting argumentation patterns is also an important objective. This will facilitate the use of structured arguments by the complexity science community.

Interpretations are, in essence, claims. Validity, for example, can be considered an interpretation of the relation between the purpose of a simulation and its realisation (including simulation results). Simulation results usually entail more than one interpretation and more claims can be based on them. These other claims would need to also argue (as a subgoal) the validity of the simulation for their particular context. There are clearly connections between all the arguments that can be constructed around a simulation, and we consider important investigating this aspect.

Results and conclusions will be synthesized in a published article.

5.3.5 Simulation update

Two important operations will be executed after this stage. The first one will be distributing the occam-π simulation. This will be reflected on to the argumentation structure and in the link between it and the results interpretations. We will be studying these changes and drawing more argumentation patterns.
The distributed simulation will enable scientists to perform simulations at scales usually addressed by mathematical models, consequently allowing a direct comparison between ABM and EBM.

5.3.6 Domain update

The second change will be to extend the biological domain. This could be carried forward either through using a more elaborate model of space e.g. 3D environment, or through introducing more biological details into the simulation e.g. gene flow. The motivation behind this update is two-fold: firstly, it will require from us to reassess our arguments and give us the opportunity to observe new (argumentation or modelling) patterns and secondly, it will bring us closer to performing original biological research. Addressing unanswered questions would be the pinnacle of our validation effort.

We will need further documentation for the new scenario and also to reflect these changes on the CoSMoS artefacts (models) we are using. Simulation changes will require tasks similar to the ones described in the previous section.

5.3.7 Experimentation

While addressing these scenarios, attention will be directed towards rare events, discontinuities that are characteristic of complex or chaotic systems. Our arguments must properly reflect these critical aspects. Experimentation will not only span over “normal” parameter ranges, but will test boundary conditions too. The link between results interpretation and the argumentation structure should motivate the identification or absence of rare events.

The outcomes of the proposed research process can be defined in terms of the qualitative (and partially quantitative) understanding of the relation between modelling and simulation outcomes, in the context of complex systems. We will have provided solutions for the use and emergence of ‘validity’, with respect to complex systems simulations. Finally, argumentation techniques and patterns will be made available for the complexity science community and the large-scale simulation will be adapted to biological research.
<table>
<thead>
<tr>
<th>ID</th>
<th>Notes</th>
<th>Duration</th>
<th>Task Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>40 days</td>
<td>Equivalence argument</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>15 days</td>
<td>Equivalence experiments</td>
</tr>
<tr>
<td>3</td>
<td>5.3.2</td>
<td>10 days</td>
<td>Argument expansion</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>15 days</td>
<td>Technical report</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>50 days</td>
<td>Validity argument</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>10 days</td>
<td>Further documentation</td>
</tr>
<tr>
<td>7</td>
<td>5.3.3</td>
<td>10 days</td>
<td>CoSMoS artefacts</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>10 days</td>
<td>Initial validity argument</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>20 days</td>
<td>Article</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>65 days</td>
<td>Analysis</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>5 days</td>
<td>Further documentation</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>30 days</td>
<td>Sensitivity analysis</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>15 days</td>
<td>Modelling alternatives</td>
</tr>
<tr>
<td>14</td>
<td>5.3.4</td>
<td>15 days</td>
<td>Pattern extraction</td>
</tr>
<tr>
<td>15</td>
<td>5.3.5</td>
<td>55 days</td>
<td>Argumentation</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>10 days</td>
<td>Alternative structures</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>5 days</td>
<td>Ordering arguments</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>10 days</td>
<td>Pattern extraction</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>15 days</td>
<td>Article</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>15 days</td>
<td>Further documentation</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>288 days</td>
<td>Simulation update</td>
</tr>
<tr>
<td>22</td>
<td>5.3.6</td>
<td>40 days</td>
<td>Distribute simulation</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>25 days</td>
<td>Update arguments</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>20 days</td>
<td>Pattern extraction</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>25 days</td>
<td>Article</td>
</tr>
<tr>
<td>26</td>
<td>5.3.7</td>
<td>65 days</td>
<td>Domain update</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>16 days</td>
<td>Ecological documentation</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>5 days</td>
<td>CoSMoS artefacts update</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>5 days</td>
<td>Simulation update</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>10 days</td>
<td>Arguments update</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>10 days</td>
<td>Verifications, reiterations</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>10 days</td>
<td>Pattern extraction</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>10 days</td>
<td>Article</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>65 days</td>
<td>Experimentation</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>10 days</td>
<td>Further documentation</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>20 days</td>
<td>Predictive simulations</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>10 days</td>
<td>Domain expert interactions</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>15 days</td>
<td>Analysis and pattern extraction</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td>10 days</td>
<td>Article</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>15 days</td>
<td>Pattern analysis and conclusions</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>100 days</td>
<td>Thesis write-up</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td>30 days</td>
<td>Thesis</td>
</tr>
</tbody>
</table>
Chapter 6

Preliminary results

This chapter presents the work that has been carried out up to this stage of the PhD. There are three main areas to present: firstly, the simulator developed for supporting work on equivalence and up-scaling will be detailed in section 6.1; secondly, the framing within the CoSMoS process will be addressed in section 6.2; finally, arguing about aspects of complex system research is summarised in section 6.3.

6.1 Occplants

One concrete deliverable of the plant-ecology case study is the construction of an explicitly parallel, process-oriented simulator, through the use of the occam-\(\pi\) programming language. Starting from the relatively simple setup described in [25], this simulator is computationally and conceptually fit for being used in large-scale simulations, of higher resolution and better performance than classical models. Preliminary simulations on a regular desktop computer have allowed the evaluation of environments containing \(10^6\) location processes and an order of magnitude less plants, so there is a great potential for reaching real-world scales.

During the development we have noted many differences between the process-oriented framework and the object-oriented one. This may be described in terms of control flow (decentralised in occam-\(\pi\) vs centralised in C++), algorithm framing (e.g. individual behaviours vs location centric) or architecture flexibility (self-organising process networks vs more rigid OO architectures). Thus, there are considerable advantages to using POP.

However, there are various disadvantages too. Channel communication, as favourably presented as it is [121], is still unnatural. In the real world, deadlocks do not appear, as it can happen through (synchronised) channel communication. If algorithms have the potential of becoming simpler due to decentralisation, the constraint of using only synchronised channel communication implies an increase in the communication infrastructure complexity.

In terms of our implementation, we need to take greater care of the way stochasticity is implemented. Parallel processes are grouped in occam-\(\pi\) under PAR constructs. If execution would be truly parallel (provided a sufficient number of processing units would be available), then theoretically we would not have reasons to consider pre-shuffling the processes or applying a double-pass approach. However, in occam-\(\pi\), these are organised into batches [118], in the sequential order in which they were defined. Even if batches would not be executed sequentially, their inner processes would. Accordingly, the simulation needs to take greater care when using PAR constructs. There is also the question of using a single random number generator (which acts as a bottle-neck on the performance of the simulation), or to make this functionality available to each process. These issues will be addressed in the following part of the PhD.

A more extensive discussion of architectures for simulating complex systems is available in [76].

6.2 The CoSMoS process

The CoSMoS process is intended to provide a walk-through for researchers that undertake modelling and simulating activities, in order to provide convincing and structured arguments of their work’s validity.
The plant ecology case-study has so far instantiated the Domain, Domain Model and partially the Software Model artifacts, from which the following conclusions can be drawn:

- apart from building tables of assumptions, we need to clearly separate abstractions too. If the former relate to the biological aspects of the simulation, the later to the computational ones. We need explicit links between the two types of simplifications, as this is a possible cause of invalidation.

- assumptions vary in the amount of information they filter out. It might not be possible to quantify the impact of each of them, but obtaining such insights would be valuable both for the study of emergence in general and for ecological questions in particular. For example, one of the assumptions introduced in Bown et al’s model [25] is that seed reproduction is clonal, and demographic research follows the spread in time, of an initial set of genotypes. However, in reality seed reproduction is sexual, so genotypes are not constant in time. For the objective’s of the study, the assumption may hold, but an argument of validity has not yet been built for it.

- hypotheses drive research and model building. As they do not constitute our understanding of the Domain, they are not part of the Domain Model. In the initial CoSMoS “lifecycle” [62], they are not explicitly represented.

6.3 Argument of equivalence

Work on the development of structured arguments over aspects of complex systems models and simulations is presented in [65], paper listed in appendix A. We mention here that in the initial phase of our research, we have developed equivalence arguments between our Occplants simulation and the original C++ one, showing how to cope with situations where the scale addressable by one simulation is much smaller than the other.
Appendix A

Equivalence Arguments for Complex Systems Simulations — A Case-Study

Equivalence Arguments for Complex Systems Simulations – A Case-Study

Teodor Ghetiu¹, Robert D. Alexander¹, Paul S. Andrews¹, Fiona A. C. Polack¹, and James Bown²

¹ Dept of Computer Science, University of York, York, YO10 5DD, UK
tedorg.rda,psa,fiona@cs.york.ac.uk
² University of Abertay, Dundee, UK j.bown@abertay.ac.uk

Abstract. Complex systems are often simulated to provide a basis for research or analysis. However, complex systems simulation often fails to properly demonstrate that the constructed simulation is an adequate tool to support investigation of the system under study. To address this issue we adopt and adapt argumentation techniques traditionally used for safety critical systems (SCS). Here we present part of an on-going case-study in which these techniques are used to demonstrate that two different implementations of a complex system simulation are adequately equivalent. This is a first step in producing further simulations of the system under study, which will be shown to be valid models on which to explore particular ecological phenomena.

1 Introduction

This paper presents part of a case study that is using a principled approach to computer simulation of a complex system. The work is part of the CoSMoS project¹, which is developing a general framework for the simulation of complex systems using agent-based approaches. One of our long-term goals is to argue the validity of complex systems simulations against domain models that capture an explicit expression of scientific understanding. More generally, we want to present properly-evidenced arguments that one model is an adequate representation of another model, or a particular perspective on reality. Such arguments form the basis for discussion between the simulators and the domain experts, and capture the rationale for the simulation, in terms of both the domain understanding of the science, and the engineering of the simulation – see [1, 2] for further discussion.

We believe the ability to argue properties of a complex system simulation (such as equivalence or validity) is an important element of CoSMoS and any other similar approach.

In this case study, a first step is to re-engineer a simulation of intra-specific plant variation [3]. The existing object-oriented simulation, in C++, needs to be scaled-up to support the scientific research. Elsewhere, we discuss the use of occam-π for efficient, process-oriented parallel agent-based simulation [4].

¹ http://www.cosmos-research.org
number of recent complex systems simulations [5–8] have successfully used this programming language. Here, we will ultimately exploit parallelisation to distribute the occam-π simulations over a cluster of machines [9]. This will allow us to simulate larger numbers and variations of plant, and a greater range of environmental influences than is possible in a purely sequential implementation.

The re-engineering is supported by construction of an argument that the occam-π simulation is adequately equivalent to the original C++ simulation. We call this an equivalence argument. The description of this argument forms the main subject of this paper. The original C++ simulation was used in ecological research that has some credibility within its research community, and through the equivalence argument we can support a claim that our re-implementation should share that credibility. Thus, our definition of adequate equivalence must make a case that the simulations capture equivalent aspects of the scientific domain, rather than simply presenting evidence of the technical equivalence of the two programs. Our argument is acceptable if the scientists – here represented by the person who has overseen the C++ simulation effort, James Bown (referred to subsequently as the scientist) – accept the argument that we have captured equivalent aspects of the scientific domain.

The paper continues with a brief introduction to argumentation techniques and their relation to simulation validity, section 2. Section 3 then describes the plant ecology case-study. Section 4 considers what adequately equivalent means and shows how we can build an explicit, structured argument of adequate equivalence for the two simulations. Section 5 gives examples of the required evidence from the simulations, to support the argument presented in section 4. The paper concludes with a discussion, section 6, conclusions and proposals for future work.

2 Argumentation and Complex Systems Validity

Elsewhere, we summarise current scepticism about the ability of computer simulations to adequately support scientific research (see [10], and cited work, [11–15]). In [1, 2], we report on an immunological case study undertaken in conjunction with immunologists, in which we found that a systematic collection and exposure of assumptions, made by the immunologists in relation to the scientific domain and by us as modellers and simulation-engineers, helped the immunologists to understand the value and limitations of our simulations. This understanding meant that the immunologists could use even basic agent-based models to test their understanding and guide their laboratory experiments; the documented assumptions gave rise to new avenues of scientific research. Here, we follow a suggestion in [2], and turn to conventional techniques from critical systems engineering, to start the process of systematising the use of arguments to capture and analyse evidence and assumptions.

In critical systems engineering, arguments are used to demonstrate a case to regulators that a system has certain properties, most commonly properties related to safety. In critical systems, it is impossible to absolutely demonstrate properties such as safety; instead evidence is collected based on criteria such as
Equivalence Arguments for Complex Systems Simulations – A Case-Study

use of accepted development practices, software, system and sub-system testing, mechanical analysis, past experience or cumulative usage outcomes, and field trials. The evidence is used to support an argument that the risk associated with the system is As Low As Reasonably Practicable (ALARP), within the operational environment for which the system is designed. A general approach to constructing and documenting safety cases can be found in Kelly [16], whose other published research includes a range of studies and applications of critical systems argumentation. For an example of safety case creation for a – hypothetical – complex system, see [17].

2.1 Argumentation in Safety Critical Systems

Early safety-critical systems were unregulated, and were potentially grossly unsafe [18]. Consequent deaths and damage costs from, for instance, industrial and vehicle accidents, led in time to regulation, part of which is usually certification. Potentially-dangerous systems are allowed if there is sufficient evidence that they would be safe to operate. For a long time, evidence was based on process – “I have followed good engineering practice, so my system is safe”. This approach is unsatisfactory in many ways, not least of which is its limiting of engineers to use only approved processes, thus inhibiting innovation.

A significant improvement in safety management came with product-based certification. Independent regulators are appointed, who set the safety criteria that a system must meet, in terms of specific evidence requirements. Developers collect evidence, and tie it together by means of a structured argument known as a safety case. It is still possible to cite an approved process as evidence, but this evidence is relegated to an appropriately-subordinate role. A safety case is accepted or rejected based on independent review of its arguments and evidence. Acceptability is not an absolute, and can change over time, in the light of experience or new evidence. This presents an important parallel to scientific investigation, particularly in biological domains, where the understanding of complex natural systems is a developing area, with much debate and many competing theories.

2.2 Summarising the Structure of an Argument: GSN

Safety cases were conventionally presented as free text, which is easy to create and immediately readable, but hard to systematically review. As Kelly [16] notes, not all safety engineers are gifted writers, and free text safety cases are often ambiguous. Construction and review of cases is improved if the structure of the argument and evidence can be summarised, for example using the Goal Structuring Notation (GSN) [19, 16]. Existing examples and patterns for GSN are predominantly concerned with safety cases.

GSN is a graphical way to express argument structures. A GSN diagram shows a hierarchy from the top-level claim – a typical safety case might seek to establish that The system is safe – down through sub-claims that support that claim (e.g. The hazard ‘loss of temperature control’ will not occur) and
eventually to the evidence supporting those claims (e.g. *Software test results for component X show no faults*). Anybody using GSN is guided by the rules of the notation, which helps to avoid gross errors of logic.

It is important to understand that GSN as a notation is of limited value—it is the argumentation culture and the safety-case literature that gives it its power in the safety field. Similarly, it would be a culture of argued validation that would be most important in addressing the criticisms (noted above) of complex systems simulation for scientific research.

### 2.3 Adapting Argumentation for Scientific Simulation Validity

When a computer simulation is used in a scientific study, the user of a simulation needs to demonstrate the extent to which the computer simulation matches reality (and other models). Traditionally, these arguments have been, at best, informal discussions in papers and reports. This causes many problems. Evidence or detail is omitted, making it difficult to assess the validity of simulation results. In an attempt at clarity, many arguments are reduced to vacuous or partial claims. There is a need to improve the quality and presentation of validation arguments; GSN is an obvious candidate for constructing argument structures and recording the evidence that supports (or could support) the argument.

Whilst there is a range of work on the validity in simulation, for example [20, 21], we are not aware of any existing work on structured arguments of computer simulation validity.

In safety analysis, the safety properties and the case for safety are normally created and rehearsed by the developers before the argument is constructed and represented in GSN. There are few specific argument construction methods, and experience shows that, whilst a top-down approach is impractical because it requires oversight of the body of evidence before the top-down structure can be identified, a bottom-up approach risks losing sight of the point of the argument. The argumentation that we require for simulations is somewhat different to safety case argumentation, in that we are constructing arguments in parallel to simulation development, and can use the top-down construction of the argument to guide development. Similarly, we do not have a regulator dictating what is and is not acceptable evidence, but instead we have a scientific collaborator who must be able to understand and review our argument. In this paper, the goal is to demonstrate that two simulations are adequately equivalent. Our argument proceeds by analysing and recording what we will accept as evidence of adequately equivalent. We then establish this evidence by systematic analysis, recording the result as a GSN argument structure. First, we briefly introduce the intra-specific plant variation domain and the existing C++ simulation.

### 3 The Example: Intra-specific Plant Variation Simulations

The work presented in this paper is the first phase of a case study to provide computer simulations to support extensions to the ecological research of Bown...
et al [3], based on their novel model of plant physiology and interactions, based on physiological traits.

In [3], computer simulation is used to demonstrate that defining plants in terms of a suitable set of traits yields results that are acceptable to the ecological community, for example, the model produces species-area and species-abundance distributions that have typical characteristic statistical signatures (curves) [22]. However, the existing simulations are limited in the number and complexity of components that can be modelled, even if the implementation and platform were fully optimised, because of the difficulty of distributing a C++ program.

3.1 Ecological Modelling and Plant Trait Models

Begon et al define ecology as the “scientific study of the distribution and abundance of organisms and the interactions that determine distribution and abundance” [23]. The “holy grail” of ecology [24] is to find general rules that relate environmental conditions, species characteristics and community composition.

To complement field experiments, ecologists attempt to capture observational patterns and behaviours in models. At one extreme, equation-based models (EBMs) focus on characteristics of the plant population as a whole, while at the other extreme individual-based models (IBMs) that allow for some of the individual variations within and between species. IBM is the more appropriate technique for study of intra-specific variation, and has the advantage that IBM individuals can map directly to and from real plants, so biological understanding can be mechanistically reflected in computer models. However, a computer model cannot hope to express all the characteristics of a real plant. A popular ecological technique is to summarise the characteristics of a plant in terms of numerical traits, with much ecological research to establish the most appropriate traits and value-ranges. Traits typically characterise visible, phenotypical properties such as shoot height, as well as ongoing biological processes such as water uptake capacity. A good model has rich informational content built using traits whose validity is supported by the direct mapping to biological data.

Ecological research has shown that trait trade-off is important in explaining the distribution and abundance of ecological communities [25]. Computer-based IBMs that model plants in terms of traits play a key role in this research. However, the models do not always map well to research goals, and it has been shown that the identification and representation of traits has a significant influence on the simulation results [26, 27].

3.2 The Computer Simulation of Bown et al

The intra-specific plant variation models of Bown et al [3] uses an IBM based on a resource-centric physiological scheme [28]. The model allows the study of the relationship between trait trade-off and the distribution and abundance of species.
Firstly, Bown et al [3] establish twelve traits (table 1) that adequately describe plant physiology. The plant species is described by a set of twelve distributions, one for each of these traits. The distributions determine the probability of each trait value across the set of plants, with individual trait values assigned to achieve the species distribution. This approach gives appropriate intra-specific variation.

<table>
<thead>
<tr>
<th>Phenotype</th>
<th>Structural compartment</th>
<th>Structural store</th>
<th>Surplus store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource uptake area</td>
<td>Age</td>
<td>Development stage</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Genotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trait 1</td>
</tr>
<tr>
<td>Trait 7</td>
</tr>
</tbody>
</table>

Fig. 1. Bown et al’s model of an individual plant [3]

In the model of Bown et al [3], a plant individual is modelled as a phenotype and a genotype, figure 1. The phenotype consists of appropriate representations of the resource storage and usage of a plant: the structural compartment represents resources corresponding to the plant’s fixed structure; the structural store holds resources that are used for reproduction; and the surplus store represents any excess of resource-uptake over the level essential to maintain the plant. In addition, the phenotype records age and development stage. In the genotype, a value is assigned to each of the twelve species traits, using a random sampling of the trait distribution to introduce intra-specific variation. Trait value distributions were obtained from field observations of the Rumex acetosa plant species [29].

Four biological processes drive the generic life-cycle of a plant: resource uptake, resource allocation, reproduction and development, as shown in figure 2. In the model, each plant takes up resources from the environment and allocate it to the three resource components of the phenotype. As resource is accumulated, the plant develops, which is denoted by incrementing the Development stage in the phenotype. Four of the trait values are related to a plant’s development stage: spatial distribution of uptake, development dependent reproduction, and the two uptake traits.

There is an initial population of plants. When a plant reproduces the distribution of seeds is controlled by the seed dispersal pattern trait. A seed is only
Table 1. Bown et al’s twelve plant traits [3]

<table>
<thead>
<tr>
<th>Trait</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Essential uptake</td>
<td>Amount of resources that a plant needs for normal development without reproduction</td>
</tr>
<tr>
<td>Requested uptake</td>
<td>Amount of resources that a plant will request to support development and reproduction.</td>
</tr>
<tr>
<td>Spatial distribution of uptake</td>
<td>Uptake capacity of a plant with respect to the distance</td>
</tr>
<tr>
<td>Compartment partition</td>
<td>Resource allocation ratio of structural compartment to structural store</td>
</tr>
<tr>
<td>Structural store release proportion</td>
<td>Proportion of structural store that can be released</td>
</tr>
<tr>
<td>Surplus store release proportion</td>
<td>Proportion of surplus store resource that can be released</td>
</tr>
<tr>
<td>Time dependent reproduction</td>
<td>Time needed before initiating reproduction</td>
</tr>
<tr>
<td>Development dependent reproduction</td>
<td>Resource level needed to initiate reproduction</td>
</tr>
<tr>
<td>Storage/fecundity relation</td>
<td>Ratio of the resource available for reproduction to the resources necessary for creating a seed</td>
</tr>
<tr>
<td>Seed dispersal pattern</td>
<td>Radius of the area of local seed dispersal</td>
</tr>
<tr>
<td>Survival threshold</td>
<td>Minimal resource level for plant survival</td>
</tr>
<tr>
<td>Survival assessment period</td>
<td>Number of consecutive timesteps over which the resources level can be below the survival threshold before the plant dies</td>
</tr>
</tbody>
</table>

viable if it lands at a valid location that does not contain a plant. In [3], reproduction is clonal, so a seed has the same trait values as its (single) parent plant. The Reproduction process may be triggered according to the trait value for time dependent reproduction or for development dependent reproduction.

The environment is represented by a single type of resource, which is distributed evenly across its surface. The resource level has an upper limit defined by a saturation level. The flow or resource to plants is constrained by release and replenishment rates, which specify the maximum quantity of resource that can be released or added to the environment at any time. In the computer simulation, the environment is modelled on a two-dimensional grid. Bown et al [3] note that a cell represents an area of approximately 100cm², which, in the model, can be occupied by at most one plant. The number of plants that take resource from a cell is determined by the location of each plant and its root area, as represented by the spatial distribution of uptake trait. Grid cells contain a resource substrate, which is parametrised by the saturation level and the release and replenishment rates.

A timestep in the simulation represents one day in the real-world. Accordingly, the values that are used for parametrisation of the model reflect the resource flow through a plant during one day [3].
In order to compare the trait-model intra-specific results to inter-species distribution results, Bown et al [3] introduce 75 individual plants, which are treated as representing 75 different species. Because the model uses clonal reproduction, these 75 species either persist and increase in numbers, or die out. A simulation run lasted for 50,000 timesteps, which corresponded to around 1250 generations of plants. The simulation was run over different environment sizes (grids of 10×10 up to 50×50 cells) to collect statistics on the relationship of the size of the environment to the number of species co-existing (the species-area curve), and to the abundance of each species (the species-abundance curve).

3.3 The C++ and occam-π Simulations

Bown et al [3] use a mechanistic model of plants through which community level processes can be studied. We have re-implemented the simulation in occam-π, but in order to use our simulation to scale up the original experiments, we need to show that the new simulation is still based on the same underlying biological model.

The C++ simulation code is sequential, running on a single thread of execution. The model uses two passes per timestep to reduce sequential bias. For example, for resource uptake, all plant demands are made in the first pass, then, in a second pass, each plant receives a normalised percentage of the quantity it requested – where the total demand on a grid cell is more then the cell can release then each demand is reduced accordingly. The limitation of running on a single thread constrains the size of the environment and population that can
be used in this simulator, which cannot handle the real-world scale of several hectares containing millions of plants.

A traditional re-engineering approach would create an abstract model of the data and processing implemented in the C++ simulation, and then re-develop this model in occam-\(\pi\). This would, in theory at least, allow formal refinement relations to be established between each implementation and the abstract model, and a formal proof of equivalence. In practice, whilst model-driven engineering provides semi-formal transformation approaches to move between object-oriented models at different levels of abstraction, the potential for formal refinement between abstract models and object-oriented code is limited. Furthermore, having extracted an abstract model from the object-oriented code, there is no established way to refine this model into the process-oriented occam-\(\pi\) language – occam-\(\pi\) is formally underpinned, but by CSP [30], an event-driven formal language.

If a formal approach were to be found, it could establish a measure of equivalence between the implementation codes of the two simulations, but would not allow the re-engineered version to take full advantage of the strengths of occam-\(\pi\). Most significantly, here, the re-engineered occam-\(\pi\) simulation can represent plants and locations as individual occam-\(\pi\) processes, each having its own thread of control. The occam-\(\pi\) processes communicate through channels through which data can be passed. This gives a closer mapping between the implementation and the biological reality than was evident in the C++ simulation.

4 A Structured Argument for Adequate Equivalence

This section works through the argument of adequate equivalence constructed for the C++ and occam-\(\pi\) implementations. For simplicity, we will refer to the C++ implementation as \(C\), and the occam-\(\pi\) implementation as \(O\). The argument is presented in GSN, using the standard notations, given in figure 3. The meaning of these symbols in our work is elucidated in the description of the argument that follows.

Note that the equivalence argument does not attempt to address the rationale or engineering of the C++ simulation – this is an established system that we cannot change. We do not compare the performance of the two implementations, as the motive for the re-engineering is not any immediate performance gain, but the distribution potential of the occam-\(\pi\) simulation across computer grids [6], with the efficient management of processes and events [31].

4.1 The Top Goal

A GSN argument starts with a top goal. In figure 5, this is shown as the rectangle labelled OCEquiv – O simulation is adequately equivalent to C simulation. This is the claim that we want to make, and the whole argument below is devoted to making that claim. In the diagram, lines with solid arrowheads connect each goal to lower-level components that together meet the goal.
A goal exists in a context. In figure 5, DefAdEq labels a context node, here promising that a definition of adequately equivalent is given elsewhere – in fact, the definition is given and explained in this section of the paper.

It is hard to definitively define equivalence. Structures in different languages may be syntactically different but semantically equivalent, or vice versa; we may have behavioural bi-similarity from different structures, or, since we are modelling complex systems, we may observe different results from similar initial conditions even within the same implementation. Despite this we need a definition of what we mean by adequate equivalence in order to argue convincingly about it.

We therefore propose that:

*the two simulations are adequately equivalent if they produce the same results over the whole range of concern.*

In common with most analyses of complex systems, same results can be defined by statistical analysis – we run each simulation many times, and collect the results. This gives a distribution for each result. We then use an accepted statistical test (usually a non-parametric test that medians and inter-quartile ranges represent the same distribution at some confidence level) to determine whether the results can be considered equivalent.

The range of concern is defined by ranges for parameters over which the equivalence should hold. In the plant simulations, this relates to the range of environment sizes and initial plant numbers. Note that, because we cannot execute the C++ simulation on very large populations, we can only consider equivalence within the range of this simulation. Instead, we present direct comparisons of results within the range of the C++, and theoretical arguments for the rest of the range. The comparison of the results gives us high confidence within part
of the range, while the theoretical arguments give us some confidence, but at a lower level, beyond that. This is represented figuratively in figure 4.

Fig. 4. Range of concern for arguing adequate equivalence: we wish to be convinced over the whole range of both simulations (the Top Goal), but we can only produce results evidence for part of the range; in the rest of the range we rely on other forms of evidence.

Note that the crucial factor in determining whether the definition of adequately equivalent is sufficient is a discussion with the scientists. Thus, in our case study, we consult the scientist directly; since he considers that our definition is sufficient, we can proceed. It is, of course, possible that this initial acceptance may be reversed when the evidence is complete and the whole argument presented – perhaps the scientist can demonstrate that our non-parametric tests of statistical equivalence are inappropriate, or our theoretical arguments are flawed, or perhaps we find that there are bugs in one of the simulations that affect the comparability of the results in other ways. The dialogue to establish the definition and associated argument is essential in the establishment of trust and understanding between simulators and scientists [1, 2].

4.2 Decomposing the Top Goal

Having agreed a top goal and the definition of the key terms that it uses, we need to provide an argument that the goal is satisfied. In figure 5, the top goal OCE-quiv is met by following the ArgSciImplRes strategy. A strategy in an argument explains the connection between a goal and its sub-goals. Here, ArgSciImplRes states that we argue over three distinct areas – the underlying science, the details of how the simulations are implemented, and the actual results that they produce. The relationship here is complementary – each child goal gives us some confidence that the parent goal holds, and together, they give us adequate confidence that the goal is met.
Note that the three-goal sub-argument in figure 5 is not an alternative definition of what it means for two simulations to be adequately equivalent. Rather, it is an approach to substantiating such a claim. We are using the three-legged argument to support a claim that the results will be the same across the whole range of concern.

![Diagram of the argument structure](image)

**Fig. 5.** Top level of the argument that the C++ (C) and occam-π (O) simulations are adequately equivalent.

The text in the GSN goal boxes is necessarily terse, and refers to concepts that need to be defined, as in the above discussion of *adequately equivalent*. It is hard to provide compelling contexts and definitive definitions. This is seen as a benefit, not a cost, of making structured arguments – you get to see where your definitions are vague or unsatisfying. (It is also much easier to see when another person’s arguments are weak.)

In GSN, an upward triangle beneath a context box means that it has yet to be instantiated – it is a placeholder for concrete content that is not yet available. In figure 5, the CDesc and ODesc context boxes could be instantiated by a reference to the code of the simulations, to common abstractions such as figure 2, above, or to summary text such as the descriptions in section 3. The argument is not complete until this instantiation is performed.
Again, the point of GSN arguments is not to demonstrate with absolute certainty that the top goal holds, but to demonstrate why the author of the argument believes that it is holds. The reviewer can disagree with the assumptions, strategy, and eventual evidence, and can challenge the author to find a better argument. Here, for instance, our scientist may dispute the strategy of arguing over three distinct areas, or may dispute the totality of these complementary areas, and challenge the author to make better justifications for its argument.

The three lowest-level goals shown in figure 5 are expanded in figures 6 to 8. Each of these argument fragments terminates in a circular solution node. Solutions refer to the evidence that supports a claim. In very simple arguments, evidence might directly support the top goal, but in practice, such intermediate sub-goals and strategies are needed to create a compelling argument. The following sections consider each of the three sub-goals in turn.

4.3 The Science Goal

Fig. 6. Elaboration of the sub-goal to show that the simulations represent the same science, from figure 5

In figure 6, the goal, \textit{ORepScience}, is shown as being solved by two further goals. \textit{OAllBioAssumptions} presents an argument that the occam-\textpi version is based on the same assumptions about the actual biology as the C++ version. \textit{OSameAbs} argues that the occam-\textpi simulation abstracts from the details of the biology in the same way as the C++ version. Again, we expand the goals by
providing context. From these goals, we directly reach the evidence required, with solutions pointing to tabular comparisons.

We could expand the argument, and the GSN, further to argue over each compared assumption or abstraction, providing a specific argument that each pair is adequately equivalent. This might be necessary if scientist found the comparison tables unacceptable without further evidence.

4.4 The Implementation Goal

![Diagram of OReplmpAbs and its sub-goals]

**Fig. 7.** Elaboration of the goal to show that the simulations represent the same implementation abstractions, from figure 5

The second child goal in figure 5, OReplmpAbs, is expanded in figure 7, with new sub-goal relating to the adequate equivalence of the code structure (OCodeStructures) and parameters (OParameters) in the two simulations. The reasoning here is that the simulation implementations are adequately equivalent if they run equivalent algorithms on equivalent data structures and use the same parameter settings.
OParameters is solved directly by a table comparing parameters in the two simulations, whilst OCodeStructures is further decomposed into a claim about algorithms and a claim about data structures. Each of these is, again, solved by a table that compares key elements of the two implementations.

Again, despite the appearance of precision provided by the GSN notation, much of the argument here is still implicit and left to the reader to infer. For example, it is implicit that equivalence of code structures and equivalence of parameters is sufficient to argue equivalence of implementation. Similarly, the argument that two algorithms in the table are equivalent is not made explicit. A software expert could verify or refute our claims, by whatever means they chose, but the non-expert must take our assertions on trust or ask for a further level of argument.

Also note that although the top-level goal talks about equivalence in terms of a black box that produces results, the argument here is white-box – we talk about how the simulation works internally. We are using white-box methods to support a claim expressed in black-box terms. This is similar to software testing, where it is common to combine white-box and black-box methods.

4.5 The Results Goal

The third child goal in figure 5, OSameResults, is decomposed, in figure 8, into claims relating to the testing and experimentation on the two simulations. OBoundaryCases claims that the two simulations provide the same results for boundary and extreme cases within the valid range. This is based on a common testing strategy, to establish that unusual situations are properly managed. We have not developed this goal yet, as is shown by the diamond beneath the goal box. To develop it, we need to consider what cases to test, in terms of the parameter and value settings that characterise each case – for instance, we may test both simulations on the case where all plants are the same, in order to check that clonal reproduction is implemented similarly; we might then check the behaviours that result with very small and very large initial numbers of plants (starting with the same plant populations), then look at the effects of extreme environments. Unless equivalence were obvious – in a very poor environment, we might be able to see that all plants died as soon as the minimum time (trait survival assessment period) had elapsed – in all cases, we would be using statistical analysis to determine acceptable equivalence of the results, as described above.

OExperiments states that, when the simulations are set up to replicate the same experiments (e.g. same environment and plant population, same trait and resource distributions), the results are the same – again using statistical analysis to determine equivalence.

As OExperiments is critical to our argument, we expand the goal further to argue under the strategy of result similarity from \( n \) experiments (ArgCExp) – we could add a context here, that \( n \) represents the specific experiments conducted on the C++ simulation, as reported in the literature. Below ArgCExp, experiments are enumerated – here using the GSN version of ellipsis for brevity. We are showing that each entry in some list has been considered, and evidence produced.
The whole argument fragment in figure 8 is in the context of RangeOfConcern. This returns to the point made in defining adequately equivalent for the top goal, that there is a range over which we can produce equivalent results, and that, in this case, we can only claim that the two simulations are equivalent when performing the type and scale of experiments for which the C++ simulation was originally designed.

In figure 8, CExperimentsGood is a justification node – shown by a J next to the node. When expanded, it identifies a justification of why we can assume that OCExperiments supports OSameResults.

5 Solution Data

The previous section summarises the argument of adequate equivalence which we are making, and which we present to the scientist for review and external
Table 2. Expanding CAssumptions – some of the assumptions made in the C++ model [3], and mirrored in the occam-\(\pi\) simulation

<table>
<thead>
<tr>
<th><strong>Environment Assumptions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The soil properties do not change radically in time.</td>
</tr>
<tr>
<td>2. The environment can be seen as a plain. Various three-dimensional landscapes will not affect the outcome.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Plant Assumptions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>3. The uptake area of a plant can be considered conic.</td>
</tr>
<tr>
<td>4. The tap root is generally more developed than the fine roots.</td>
</tr>
<tr>
<td>5. The ratio between resource allocation towards growth and towards reproduction varies slowly in time.</td>
</tr>
<tr>
<td>6. Germination takes place in no longer than one day.</td>
</tr>
<tr>
<td>7. Plants develop unhindered, if having necessary resources.</td>
</tr>
<tr>
<td>8. Plants release their resources back into their environment, when they die.</td>
</tr>
<tr>
<td>9. Plants may die of starvation or due to unpredictable events.</td>
</tr>
<tr>
<td>10. Seed dispersal happens over a short period (a matter of days).</td>
</tr>
<tr>
<td>11. Each seed requires a similar amount of resource.</td>
</tr>
<tr>
<td>12. Seeds that fall in populated areas, most often do not germinate.</td>
</tr>
</tbody>
</table>

scrutiny. We now consider some of the evidence, or solutions, that support the argument.

Most of our argument of adequate equivalence points to tabular comparisons. We briefly cover two of the biological aspects, but then focus on structural comparison from the implementation argument structure, which raises most of the interesting issues of equivalence. The generation of evidence for the argument of adequately equivalent science is, in general, more interesting, and the establishment of this argument will be essential when we extend the simulation to support further experiments on the intra-specific plant variation. However, for the argument of simulation equivalence, the science has already been captured by J. Bown in constructing the original C++ simulation (and reviewed by the scientists with whom he was working). We have essentially one source, Bown et al [3], and, throughout, we refer to an interpretation of it that is directly expressed in the C++ implementation.

5.1 Biological Assumptions

Biological assumptions were not explicitly identified in the body of work represented by Bown et al [3]. However, we have had to identify some assumptions in order to complete the re-engineering, and can use these to strengthen the argument of equivalence. Table 2 lists some of the assumptions that form the context CAssumptions in figure 6. These have been confirmed by the scientist, giving us confidence that the occam-\(\pi\) simulation captures the assumptions on which the C++ simulation was based.
Table 3. Expanding CAbstractions – some of the abstractions made in the C++ model [3], and mirrored in the occam-π simulation

<table>
<thead>
<tr>
<th>Environment Abstractions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Resource release and replenishment rates are constant.</td>
</tr>
<tr>
<td>2 The environment is 2D and each grid cell can hold only one plant.</td>
</tr>
<tr>
<td>3 The maximal level of resource is homogeneous across the environment.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plant Abstractions</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Requested uptake is homogeneous with respect to the distance from the plant.</td>
</tr>
<tr>
<td>5 The uptake area has a regular shape and is not affected by neighbouring competitors roots.</td>
</tr>
<tr>
<td>6 The ratio between resource allocation towards growth and towards reproduction, does not vary in time.</td>
</tr>
<tr>
<td>7 Germination is instantaneous (takes only one time step).</td>
</tr>
<tr>
<td>8 When they die, plants release all of their resources into the environment.</td>
</tr>
<tr>
<td>9 Plants die of random events and starvation.</td>
</tr>
<tr>
<td>10 Reproduction is instantaneous (takes only one time step).</td>
</tr>
<tr>
<td>11 Each seed requires an identical amount of resource.</td>
</tr>
<tr>
<td>11 Seeds die if cells are occupied, otherwise they become plants.</td>
</tr>
<tr>
<td>12 Reproduction is clonal.</td>
</tr>
</tbody>
</table>

5.2 Biological Abstractions

Between biological facts and assumptions and the construction of computer simulations, we make various abstractions to map the real world into the computational one. The abstractions are influenced by the platform on which the computer simulation is built, as well as subjective factors. To expand the CAbstractions context in figure 6, we collect the abstractions made by Bown et al [3], some of which are listed in table 3). We then checked that the occam-π simulation respects each of these abstractions.

5.3 Algorithm Mapping

To compare the algorithms of the two simulations, a sub-goal of OCodeStructures in figure 7, we present pseudo-code summaries and check subjectively for similarity. Figure 9 gives a pseudo-code overview of the two simulations, whilst figure 10 focuses on the algorithm for resource uptake. Note that the pseudo-code for the occam-π implementation is written to facilitate comparison with the C++, rather than in a way that native occam-π programmers would use.

The sequential C++ implementation has a centralised architecture. This requires loop-iteration over, for example, all instances of location and all plant individuals. Because occam-π is a parallel language, all the occam-π processes (plants, locations) could execute in parallel, shown in figure 9 as each individual and each location.

In the C++ model, a double-pass approach is used to reduce positional biases – resource uptake and usage are separated into two phases, otherwise subsequent
behaviours such as seed dispersal would take place in the order in which plants are iterated. In the occam-π simulation, *synchronisation* means that the plant processes will be blocked until all have finished sending their resource requests, when all processes will be released to proceed to resource uptake.

In reviewing the complete comparison of the high-level algorithm, we found that, in terms of semantics and results, the two implementations can be considered equivalent. The resource flow is identical; only the architecture through which it is carried out differs.

The second pseudo-code comparison, figure 10, refers to the process of resource uptake. In the C++ implementation, resource uptake is location-centric – the neighbourhood of each location is scanned for plants and the demand of each plant is calculated and stored. A normalisation process is necessary to divide the resource fairly among the plants. In the occam-π implementation, however, the process relates more closely to the biology, as each plant interacts directly with its location. The computational abstraction is, in this case, that of plants and locations interacting through a *client-server protocol* [32].

In this case, the algorithm comparison shows that, although the input and output of the algorithms is equivalent, the detail is different. To make a strong equivalence argument, we would need also to look at evidence of resource uptake behaviours through experimentation and testing.

---

<table>
<thead>
<tr>
<th><strong>C++ simulation</strong></th>
<th><strong>occam-π simulation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>instantiate locations</td>
<td>instantiate locations and servers</td>
</tr>
<tr>
<td>instantiate individuals</td>
<td>instantiate individuals</td>
</tr>
<tr>
<td>for each timestep</td>
<td>while simulation_running</td>
</tr>
<tr>
<td>/* resource uptake */</td>
<td>/* resource uptake */</td>
</tr>
<tr>
<td>for each location</td>
<td>each individual</td>
</tr>
<tr>
<td>assess resource demand</td>
<td>place resource demand</td>
</tr>
<tr>
<td>release resources</td>
<td>SYNCHRONISE</td>
</tr>
<tr>
<td>replenish substrate</td>
<td>each location</td>
</tr>
<tr>
<td>/* resource usage */</td>
<td>process resource demands</td>
</tr>
<tr>
<td>for each individual</td>
<td>replenish substrate</td>
</tr>
<tr>
<td>allocate uptake</td>
<td>/* resource usage */</td>
</tr>
<tr>
<td>assess death</td>
<td>each individual</td>
</tr>
<tr>
<td>if not dead</td>
<td>allocate uptake</td>
</tr>
<tr>
<td>assess development</td>
<td>assess death</td>
</tr>
<tr>
<td>assess reproduction</td>
<td>if not dead</td>
</tr>
</tbody>
</table>

Fig. 9. Comparing the C++ and occam-π simulations
Resource uptake (C++)
Sequential algorithm
Main control loop:
for each location
select uptake_area of location
create empty demand_list
for each loc in uptake_area
if loc occupied
select occupying plant
calculate plant’s demand on location
add demand to demand_list
normalise demand_list
for each demand in demand_list
select demanding plant
add resources to plant’s uptake
replenish location’s substrate

Resource uptake (occam-π)
Parallel algorithm
Plants:
for each location in uptake_area
send resource request
SYNCHRONISE
for each location in uptake_area
if resources released
uptake resource
Locations:
create empty demand_list
while running
if resource request received
store request
if all requests received
normalise demand vector
for each demand in demand_list
select demanding plant
send resources to plant
reset demand_list
replenish substrate

Fig. 10. Comparing resource uptake algorithms for the C++ and occam-π simulations

5.4 Data Mapping
The second sub-goal of OCodeStructures in figure 7 concerns the argument of adequate equivalence of data representations. We can explore this similarity starting from a class diagram of the C++ implementation, figure 11, and a similar diagram of the occam-π processes and channels, figure 12.

UML provides an object-oriented modelling notation which is well-adapted to expressing the class structure of C++, but the notation of figure 12 is just an ad hoc representation of occam-π processes and channels. However, informally, we can compare data types between the two diagrams. As in earlier argument fragments, we present the evidence at this level for review; we only need to elaborate the comparison if the scientist is not prepared to accept it.

The C++ implementation of Bown et al [3] uses the class Location to represent cells of the environment. Each location contains a resource substrate, of class Substrate, and a plant individual, of class Plant. Because plants do not move, a Location instance is represented as being composed of one Plant instance and one Substrate instance. The Location attribute, occupied, takes the value 1 if there is a plant growing at a location, and 0 when a location is empty – in the C++ an unoccupied location is associated to an “empty” plant instance,
rather than to no plant instance. The diagram does not show the relationship between a location and the plants that are taking up resource from it, as the C++ implementation calculates this from the plant traits and location at run-time.

In the occam-π implementation, a similar form is used for the location substrate, but occam-π supports more flexible data structuring for locations and plants. These are dynamic processes (the occam-π PROC structure), which interact through channels (the occam-π CHAN structure). The relation between plants and locations is implemented through explicit channel communication.
The channel ends held by each plant process can be connected to the corresponding channel ends in any location process.

Comparing the two data structure implementations, we can observe differences in terms of attributes and their data types, the nature of plants, locations and their relationship. By reference to the biological model that these represent, we could declare ourselves adequately confident that these implementations represent implementations of the same abstract model. However, there are some subtleties that may present problems, such as the subtle quantitative effects of internal data formats: the C++ implementation uses the type double while the occam-π one uses REAL32. The two differ in terms of precision, double being represented on 64 bits, while REAL32 on 32 bits. Again, we need to check the effect of this difference through appropriate experimentation and testing – at this stage, we do not believe that the difference in precision qualitatively affects the simulations results, but we may need more evidence to convince the scientist.

6 Discussion

In this paper, we present a summary of an argument of adequate equivalence between an existing C++ simulation and a re-engineered version in occam-π. If we can assume that the original simulation is valid, then establishing the equivalence of the occam-π version would imply its validity in the same context and for the same purposes as the original simulation (see [21]). We used GSN to visualise the argument structure: those faced with evaluating our argument can immediately see the basis of the belief that the two models are adequately equivalent, and can challenge areas that they do not consider to be sufficiently supported by evidence.

This work is part of the CoSMoS project\(^2\), which is developing a general framework for the simulation of complex systems. Part of this framework concerns the routine collection of assumptions – about the domain, the design, and the implementation. In the CoSMoS context, just the exposure of assumptions has led to scientific acceptance of some of our experimental simulations [1]. The work presented in this paper is a first step towards producing guidance and techniques for systematising argumentation relating to simulation development. However, turning assumptions into evidence for arguments that a simulation is a valid imitation of the real world, for a given scientific purpose, is a non-trivial activity, which is the subject of ongoing research.

Computer scientists who have spent a career in the deterministic world of the digital computer are often sceptical about the value of arguments of validity, safety etc. However, in simulating complex systems for scientific study, we are not seeking to model or implement traditional deterministic computer systems. A simulation that reduces the interacting complex systems of the real world to a deterministic system is unlikely to be adequate for the areas of scientific research that we seek to support.

\(^2\) http://www.cosmos-research.org
Our work, here and in the CoSMoS project, also signals a departure from the common form of computer applications, in that our simulations are designed to support specific domain models – a particular expert’s view of a particular scientific context. The aim of the simulation is to support those areas of scientific experimentation that rely on that specific scientific context. If the scientist wishes to extend or adjust the context, then the simulation models must be extended or adjusted, and the adequate equivalence re-established. Later revisions can be facilitated through the careful recording of the argument of equivalence or validity for each simulation; if a preceding simulation was already acceptable, scientifically, and a new simulation corresponds to that simulation for part of its range, then we need concentrate only on what has changed.

This brings us to the use of occam-\pi in the re-engineered simulation presented here. To support the need to extend or adjust simulations in line with scientists’ requirements, we need flexible implementations. In CoSMoS, we have used a range of implementation languages, and, although occam-\pi does not have the mature support of languages like C++ and Java, we have found that applications written in occam-\pi are easy to adapt and re-use. The CoSMoS project is assembling concrete evidence of this assertion, as well as seeking to improve the maturity of the occam-\pi programming environment.

7 Future Work

In relation to the specific example presented here, we need to complete the argument of adequate equivalence, and expose it all to the critical review of our scientist and his colleagues. Our next step is then to use the occam-\pi implementation to scale up the original experiments, which will improve the quality of the scientific evidence we can provide. We will then produce a series of modified simulations to support other experiments on intra-species and inter-species plant ecology, in collaboration with Bown’s group.

In relation to the CoSMoS project, the work is contributing to the body of evidence on use and suitability of occam-\pi for developing flexible, validated simulations to support scientific work. The argumentation processes will form part of the CoSMoS framework for complex systems modelling and simulation – we continue to review SCS work for guidance in analysis, evidence collection and management, argument construction and validation. We plan to provide specific guidance on producing GSN type arguments in the context of complex systems simulation. In addition, we are applying the activities outlined here in a range of other case studies including various scientific studies of the immune system and work on swarm robotics.

8 Acknowledgements

The work of Teodor Ghetiu, Paul Andrews and Fiona Polack is supported by the CoSMoS project, EPSRC grants EP/E053505/1 and EP/E049419/1.
References

Bibliography

SIGSIM Simulation Digest 7, 3 (1976), 3.


[26] Seager, W. Emergence and supervenience; emergence and efficacy.


