Using CoSMoS to Reverse Engineer a Domain Model for Aevol

Paul S. Andrews[1,2] and Susan Stepney[1,2]

[1] York Centre for Complex Systems Analysis, University of York, UK
[2] Department of Computer Science, University of York, UK

Abstract. The CoSMoS approach and pattern language has typically been used to guide the entire process of development and use of scientific simulators. The resulting CoSMoS components (domain, domain model, platform model, simulation platform and results models) provide an explicit framework for presenting and reasoning about both the engineering and scientific aspects of the simulator and its results. The flexibility of CoSMoS enables us to use the same patterns to reverse engineer CoSMoS model components from pre-existing simulators and associated research literature. We demonstrate this here by applying the CoSMoS patterns to the Aevol (*a*rtificial *evol*ution) simulator in order to extract an explicit Aevol domain model.

1 Introduction

The CoSMoS approach [1, 2] provides guidance on how to engineer, and subsequently use, computer simulators for scientific investigations. At the heart of CoSMoS is a collection of **core components**, shown in figure 1, whose construction let us reason about the process of engineering and using a simulator. The process of identifying and constructing these components is governed by a set of CoSMoS **patterns** [3], which each describe a generalised core solution to a task that one might encounter when working with scientific simulations. Appendix A provides a brief summary of the main CoSMoS patterns referred to in this paper.

The CoSMoS components and patterns have been successfully used to assist the engineering of simulators for science [4, 5]. These simulators have followed a typical application of the CoSMoS approach by applying the CoSMoS Simulation Project pattern that guides the identification and construction of the core components through the application of three phase patterns, Discovery Phase, Development Phase and Exploration Phase. The flexibility of the CoSMoS products and patterns

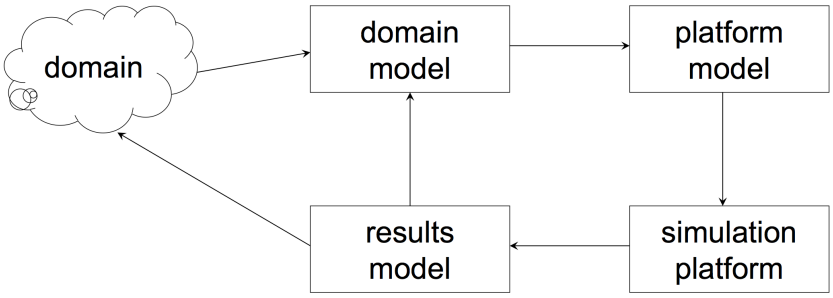

Fig. 1. The CoSMoS core components. Each is captured and referred to by its associated pattern, Domain, Domain Model, Platform Model, Simulation Platform, Results Model.

approach, however, does not dictate that we must always follow the complete CoSMoS Simulation Project pattern. Instead we can apply a subset of the CoSMoS patterns to construct core components in a way that best suits our particular circumstances. Here we illustrate such an example by showing how the CoSMoS patterns can be used to reverse-engineer a Domain Model from pre-existing simulator code (i.e. a simulator that hasn't been created within the CoSMoS approach).

Specifically, we are motivated to reverse engineer a Domain Model of Aevol [6], an *in silico* experimental *a*rtificial *evol*ution platform [7] in which populations of digital bacteria are subject to Darwinian-style evolution. Although there are numerous publication that detail aspects of Aevol [7–9], as well as the simulator's source code [6], a single consistent Domain Model (in the CoSMoS sense) does not exist. As a Domain Model provides an explicit representation of the model of the science that underpins a simulator, an Aevol domain model is desirable to help us reason about Aevol simulation results and to help us produce any future extension of the Aevol platform.

We first outline in section 2 how we can apply the CoSMoS approach to develop an Aevol domain model, identifying our starting point and method for reverse engineering the model. We then describe the Aevol domain in section 3, and establish the Aevol platform model in section 4. From this point we outline the Aevol domain model in section 5, and finally discuss the reverse engineering exercise in section 6. Appendix A provides a brief summary of the main CoSMoS patterns referred to in the paper, and appendix B contains the Aevol domain model and Aevol platform model elements.

2 Application of CoSMoS to Aevol

We have as our starting point for reverse engineering an Aevol domain model the Aevol code (freely available to download at [6]) and a corpus of research literature such as [7–9]. This literature contains descriptions of the biology that has inspired Aevol, descriptions and 'cartoon' depictions (see figure 1 in [8]) of how Aevol has been implemented, and Aevol simulation results. Our first task is to examine how the Aevol code and literature relate to the CoSMoS core components identified in figure 1. This enables us to then identify how we can approach engineering the Aevol domain model, and which CoSMoS patterns will help us do this.

The CoSMoS core components shown in figure 1 are described in detail in [10], and summarised here:

Domain: a particular view or perspective of the real world system under study.

Domain Model: a model that explicitly captures aspects of the domain, identifying and describing the structures, behaviours and interactions. It is a model based on the science and is free from any simulation implementation details.

Platform Model: a model derived from the domain model that details how the concepts captured in the domain model will implemented by the simulation platform.

Simulation Platform: encodes the platform model into a software and hardware system upon which simulation experiments are run, which in turn generate the results.

Results Model: a model describing the behaviour of the simulation platform based on the output of simulation experiments, providing the basis for interpretation of what the simulation results show.

These components are framed within a Research Context that identifies the scientific context of a simulation-based research project, establishing its scope, purpose and success criteria.

The closest match between the Aevol resources we possess and a CoSMoS component is the Aevol source code and the Simulation Platform. In this case the Aevol simulation platform is represented by the Aevol C++ code, relevant programming libraries, and the operating system and computer hardware on which it all runs. Any operating system and computer hardware that has the necessary C++ compiler will be able to run as an Aevol simulation platform, so for the purposes of this paper we just consider the Aevol simulation platform as being the Aevol code.

Elements of the other four components are contained within the Aevol literature, although there is no explicit mapping to any specific core

component. As our task is to construct a Domain Model, we ignore trying to establish the Aevol results model for this exercise and instead focus on identifying the Aevol domain model from the details contained within the literature and our Aevol simulation platform. First we use the literature to infer a description of the Research Context and Domain of Aevol within section 3.

Due to the relationship between the Domain Model, Platform Model and Simulation Platform, the Aevol simulation platform will contain many of the concepts, structures and behaviours present in the Aevol domain model and the Aevol platform model. So, instead of trying to infer the Aevol domain model solely from the Aevol domain and literature we work backwards from the Aevol simulation platform to create the Aevol platform model (section 4). From there we create the Aevol domain model (section 5) by cross-referencing the structures and behaviours in the Aevol platform model with Aevol domain concepts.

3 Aevol Domain

Examining the Aevol research literature, we can use the Domain Identification pattern to outline Aevol's domain. In short, this pattern involves: providing an overview description of the Domain; using the Cartoon pattern to present an informal sketch that identifies system components in domain specific terms; identifying the relevant sources for domain knowledge; applying the Document Assumptions pattern; and defining the scientific scope and boundary of the domain.

In normal circumstances we would identify a Domain Researcher as our source for domain knowledge, however for this exercise we already have published material on Aevol and we wish to demonstrate how a Domain can be extracted from such a body of work. So, the chosen sources for domain knowledge on Aevol are the three publications [7–9], which capture a representative cross-section of the entire body of published work on Aevol (see [6] for a full list of Aevol publications). The domain description now follows.

Aevol is designed to provide insight into the real world dynamic of Darwinian evolution. The Aevol domain, therefore, falls within the areas of evolutionary theory and digital genetics [11] in which populations of artificial organisms evolve within a computer simulation. Specifically, Aevol is focused on the evolutionary dynamics of the size and organisation of bacterial genomes, enabling the user to run digital experiments in an *in silico* laboratory to test evolutionary scenarios [8].

The cartoon in figure 2, adapted from [8], summarises the biological processes that have been modelled by Aevol. Here we see individuals

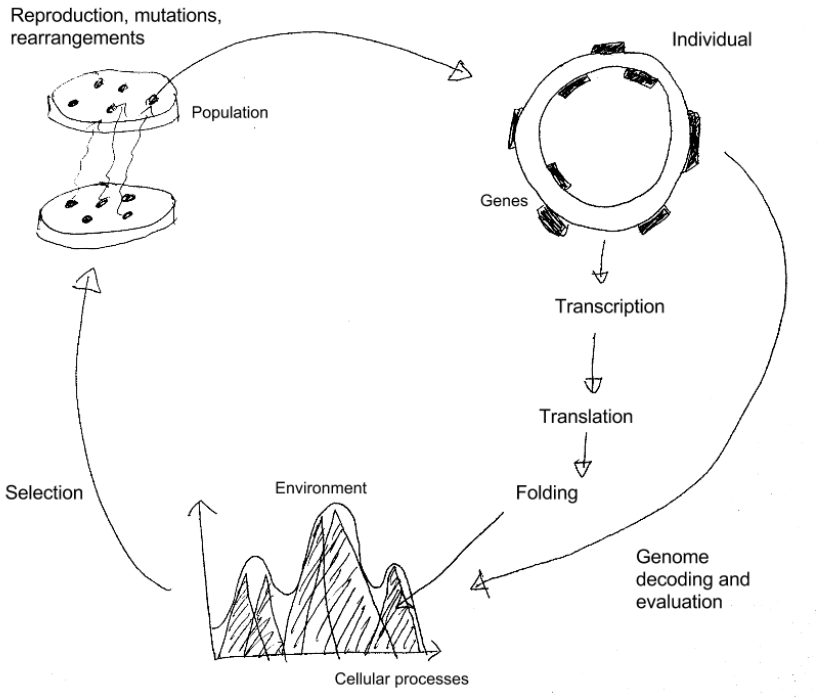

Fig. 2. Aevol domain cartoon, adapted from [8]

have double-stranded circular genomes, which are constructed from a sequence of nucleotides. The nucleotides on the genome are grouped into coding sequences (genes) and non-coding sequences. Genes are decoded via an explicit process of transcription (via messenger RNA), translation and folding, which results in a signature of the cellular processes for that individual. Selection is performed as a function of this signature compared to an environment, with individuals with a closer match to the environment being favoured for selection. Selected individuals are asexually reproduced to form the next generation of the population. Variation is introduced to the population by exposing reproduced individuals to genetic mutations and rearrangements such as point mutations, insertions, deletions, duplications, inversions and translocations. These mutations can create new genes and modify or destroy existing genes.

In addition to the domain description above, we can extract from the background Aevol literature a number of important assumptions (Document Assumptions) and establish the scientific scope and boundary

of the domain. Whilst we don't capture all assumptions, the following should be considered when interpreting outputs from the Aevol simulator:

- Space is not modelled, bacteria exist in a well mixed environment. (Modified versions of Aevol exist which consider a grid-like environment, but this is not considered here).
- Individuals are composed only of their genome and mechanisms to transcribe, translate and fold proteins; there is no cell metabolism.
- The non-linear mapping between genotype and phenotype (cellular processes) is sufficient for the selection process..
- Genetic transfer only occurs vertically during reproduction; there is no horizontal transfer (plasmids) between individuals. (Modified versions of Aevol exist with plasmids, but this is not considered here).
- The environment is static and does not change. (Modified versions of Aevol exist with varying environments, but this is not considered here).
- Complete populations of individuals are replaced per generation.

Elements of the Research Context for Aevol can be identified from the literature as:

- Investigating processes of indirect selection and evolvability
- Insights into circular bacterial genome structures (chromosome length and composition)
- Dynamics of structural changes to genomes only
- Spatial environment effects not taken into account
- Only vertical transfer of genetic material (no plasmids or horizontal transfer)

4 Aevol Platform Model

Although the Platform Modelling pattern (see appendix A) assumes that to develop the Platform Model from the Domain Model, we can still apply it (with caveats) to help us develop the Aevol platform model in the absence of the Aevol domain model. First we apply the Modelling Approach pattern to identify a language to describe the Aevol platform model. Given object-oriented design of the Aevol code, we have chosen the UML's Class Diagrams and Activity Diagrams to express the implemented structures and behaviours respectively. These diagrams are also a natural language in which to express a domain model (see examples [4, 5]), so we keep continuity of modelling language between the Aevol platform model and Aevol domain model.

As previously mentioned, the Platform Modelling pattern states that the platform model should be developed from the Domain Model. This processes considers engineering factors and should result in a Platform Model in which some Domain Model components have been removed (such as emergent behaviours), components not in the Domain Model have been added (such as visualisations), and components differ between the two models (such as number of agents). Here, however, we can extract the platform model components and behaviours directly from the Aevol code (the Simulation Platform), and we consider later how the Platform Model and as yet undefined Domain Model differ.

Extracting the platform model from the Aevol code is essentially a mechanical task with a one-to-one mapping between class structures, behaviours and parameters in the code and Platform Model, as the Simulation Platform is simply a concrete implementation of the Platform Model in a given programming language. The Aevol code is around 20000 lines of native C++ and, other than the standard C++ and system libraries, uses the SIMD-oriented Fast Mersenne Twister (SFMT) library [12] for generating pseudo-random numbers.

The main classes present in the Aevol code, and the relationships between these classes, are shown in figure 5. The methods of the Aevol classes implement the behavioural aspects of the Aevol platform model. Being guided by the Aevol domain description established above, these behaviours have been summarised in three activity diagrams in figures 6, 7, 8 which show the selection process, mutation process, and fitness evaluation process respectively. It is noted that the activities in these diagrams reflect only the gross-level structure of the Aevol code and that more in-depth descriptions of the code could be produced. However, this is infeasible for the many thousands of lines of Aevol code, and the activity diagrams presented provide the necessary level of detail to help us reach our ultimate goal of presenting the Aevol domain model.

Having documented the main structural and behavioural concepts present in the Platform Model diagrams, we can identify which elements of the Platform Model have been added for instrumentation purposes (to run simulations and record data). The following classes shown in figure 5 are instrumentation classes: *ExperimentManager*, *ExperimentSetup*, and *OutputManager*. These instrumentation concepts may, or may not, be present in the Domain Model depending on whether it explicitly captures the domain experiment within. We will revisit this when we consider the Aevol domain model in the next section.

We can identify cases of abstractions that have obviously been made for implementation reasons, and these should be reflected in the Domain Model. First, the DNA nucleotide sequence captured by the *DNA* class

is a binary string, therefore there are just two nucleotides, 0 and 1. This has effects throughout the genotype-to-phenotype mapping, namely in the translation and folding processes encoded within the *GeneticUnit*, *RNA* and *Protein* classes. Second, the *Environment* class is implemented as a *FuzzySet*, which represents a continuous function as a discrete set of points.

These last two platform model abstractions highlight two of the main assumptions (applying Document Assumptions) about the platform model, namely:

- A binary string nucleotide representation of the genome is sufficient to reflect the structural organisation on the genome during the evolutionary process
- The genotype-to-phenotype mapping implementation provides an acceptable non-linear mapping between the two to mimic that of real bacteria.

Lastly, the Platform Model and Domain Model differ from what has been removed from the domain model when constructing the domain model, such as emergent behaviours. We identify these in the next section when comparing the Aevol platform model and Aevol domain to infer the Aevol domain model.

5 Aevol Domain Model

Given our identified Aevol domain and Aevol platform model, we can use the Domain Modelling pattern to construct our Aevol domain model. First we apply the Modelling Approach pattern. As mentioned above, UML's class and activity diagrams are natural tools for expressing modelling structures and behaviours and they have been used to capture the Aevol platform model, so we use the same modelling approach here. Next, we apply a four stage process to identifying the Aevol domain model:

1. Extract the domain structures and behaviours from the Aevol platform model diagrams removing those concepts that were added for additional instrumentation and modifying those concepts that were deemed to be implementation abstractions.
2. Identify the behaviours that are not in the Aevol platform model, but are necessary to explain the Aevol domain.
3. Modify the Aevol platform model diagrams as according to 1. and 2. to make the Aevol domain model.
4. Construct the Aevol domain experiment model (not shown here) based on relevant instrumentation present in Aevol platform model.

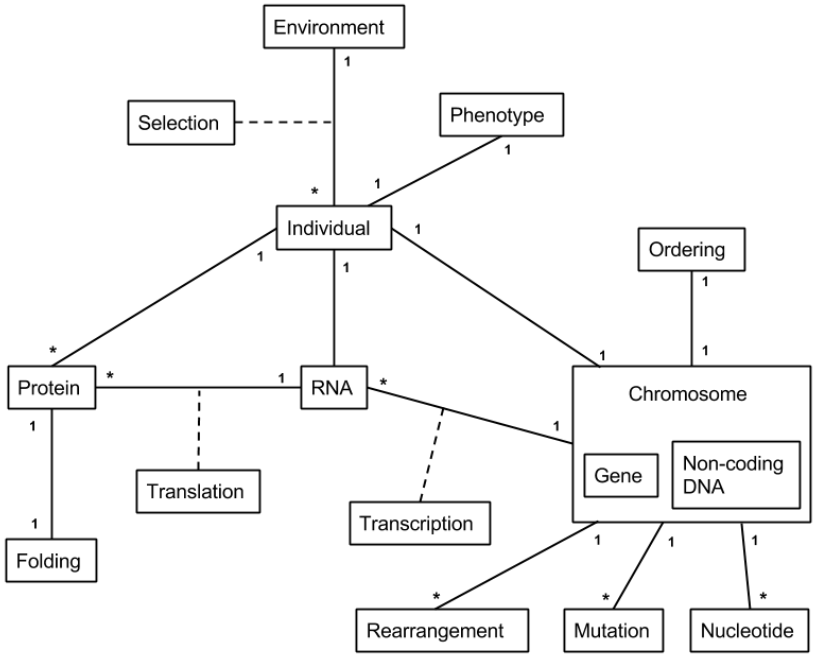

Fig. 3. Aevol domain model class concepts.

The resulting Aevol domain model is summarised in figures 3 and 4. In addition, the Aevol platform model activity diagrams (figures 6, 7 and 8) are also suitable components of the Aevol domain model as they are able to convey the selection, mutation, and fitness evaluation processes present Aevol domain. We may in future wish to create more detailed activity diagrams separately for the Aevol domain model and Aevol platform model, but the current more generic activity diagrams suffice for this exercise.

The Aevol domain model classes in figure 3 differs from the Aevol platform model classes (figure 5) in the following ways:

- We have removed the experiment specific instrumentation classes such as *ExperimentManager* and moved these concepts to the Aevol domain experiment model (not shown here).
- The relevant evolutionary processes (*Folding*, *Translation*, *Transcription*, *Mutation*) are represented explicitly as classes that act upon the structures (*Protein*, *RNA*, *Chromosome*).

- The *Chromosome* is composed of both coding (*Gene*) and *Non-coding DNA*
- The concept of *Ordering* is applied to the *Chromosome* to capture the concept of the **emergent** process that acts up the bacterial genome as a whole. This concept is expanded with the aid of figure 4.

The Cartoon in figure 4 captures the Research Context Diagram element of the Aevol domain model and complements the concepts provided in class diagram, specifically the emergent ordering of the bacterial genome. This provides an overview of the behaviours being modelled, highlighting the components hypothesised to play a significant role in the real-world phenomena and the system-level behaviours that result from their interactions. This links the domain language with expected observables in the simulation. Here we are highlighting the concepts of **indirect selection** and **evolvability** that occur within the research literature with how we expect these concepts to be revealed in the Aevol simulation platform and subsequent Results Model for Aevol: namely the **dynamical changes** in the population chromosome length and make-up of coding and non-coding DNA on that genome over an evolutionary run. We also highlight the components in the Aevol domain that we believe are responsible for the these emergent behaviours. This essentially captures the rates at which mutational and rearrangement processes occur, and how these are selected for according to the individuals phenotype.

Finally, we apply Document Assumptions, recording the main Aevol domain model assumptions as:

- We have suitably identified the source of emergent behaviours we expect to arise in the Aevol simulation platform
- The mutation, rearrangement and selection dynamics are sufficient change the structure on the chromosome
- We can measure ordering on the bacterial chromosomes

6 Conclusions

We have shown how we can use the CoSMoS pattern language to take a pre-existing scientific simulation platform and extract an explicit Domain Model via a process of constructing the associated Domain from the domain literature, and the Platform Model from the simulator code. To do this we have used only those CoSMoS patterns that were required. Importantly, this is made possible as the patterns do not impose a strict ordering on *when* things need to be done, only *what* needs to be done.

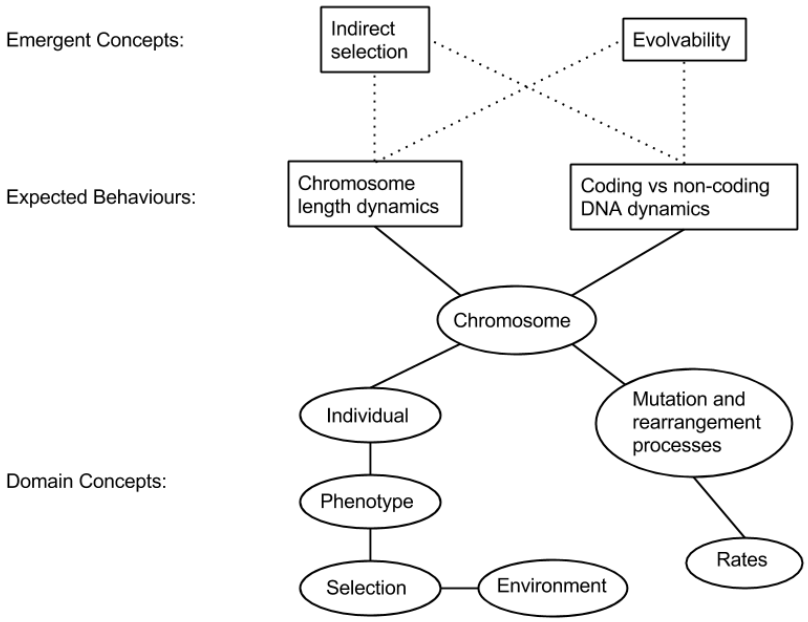

Fig. 4. Aevol research context diagram.

Consequently we have been able to apply the patterns in a different ordering to a typical application of the CoSMoS Simulation Project.

For simplicity we have not followed the argumentation patterns that are associated the other patterns we have used, but there is no reason why these cannot be used for a similar exercise. These patterns help us record and justify assumptions and design decisions. Also, we have not yet employed the Domain Researcher (the developers of Aevol) to check that we have ultimately captured the correct Domain and Domain Model. This is left as a necessary task for the future, and may result in the incremental re-application of the patterns to refine our CoSMoS products.

We have subsequently used the Aevol domain and platform model to create a refactored version of the simulator in the Python programming language. Whilst this does not have the performance of the original C++ version, it is providing to be a very useful prototyping tool to explore extensions of the Aevol simulator. With the explicit Domain Model and Platform Model, these extensions can be done in a principled and transparent manner. Future work (as part of the EvoEvo project [13]) will

apply the same domain model reverse engineering process to a family of simulators based on the "Pearls on a string" (PoaS) formalism that have been used to explore evolutionary dynamics [14, 15]. We can then use the Aevol domain model and the domain model of the PoaS simulators to develop a common metamodel [10]. This metamodel will provide the basis to develop a framework for novel evolutionary algorithms.

Acknowledgments

This work was funded by the EU FP7 project EvoEvo [13], grant number 610427. We would like to thank our colleagues on the project for helpful discussions.

References

[1] P. S. Andrews, F. A. C. Polack, A. T. Sampson, S. Stepney, and J. Timmis, "The CoSMoS process, version 0.1: A process for the modelling and simulation of complex systems," Tech. Rep. YCS-2010-453, Department of Computer Science, University of York, Mar. 2010.

[2] S. Stepney and et al., *Engineering Simulations as Scientific Instruments – A Pattern Language*. Springer, to appear.

[3] S. Stepney, "A pattern language for scientific simulations," in *2012 CoSMoS workshop* (S. Stepney, P. S. Andrews, and M. Read, eds.), pp. 77–103, Luniver Press, 2012.

[4] M. N. Read, P. S. Andrews, J. Timmis, R. A. Williams, R. B. Greaves, H. Sheng, M. C. Coles, and V. Kumar, "Determining disease intervention strategies using spatially resolved simulations," *PloS one*, vol. 8, no. 11, 2013.

[5] K. J. Alden, J. Timmis, P. S. Andrews, H. Veiga-Fernandes, and M. C. Coles, "Pairing experimentation and computational modelling to understand the role of tissue inducer cells in the development of lymphoid organs.," *Frontiers in Immunology*, vol. 3, no. 172, 2012.

[6] `http://www.aevol.fr/`.

[7] D. Parsons, *Indirect Selection in Darwinian Evolution: Mechanisms and Implications*. PhD thesis, L'Institut National des Sciences Appliquée de Lyon, 2011.

[8] B. Batut, D. Parsons, S. Fischer, G. Beslon, and C. Knibbe, "*In silico* experimental evolution: a tool to test evolutionary scenarios," *BMC Bioinformatics*, vol. 14, no. Suppl 15, p. S11, 2013.

[9] C. Knibbe, A. Coulon, O. Mazet, J.-M. Fayard, and G. Beslon, "A long-term evolutionary pressure on the amount of noncoding DNA," *Mol. Biol. Evol.*, vol. 24, no. 10, pp. 2344–2353, 2007.

[10] P. S. Andrews, S. Stepney, T. Hoverd, F. A. C. Polack, A. T. Sampson, and J. Timmis, "CoSMoS process, models, and metamodels," in *2011 CoSMoS workshop* (S. Stepney, P. Welch, P. S. Andrews, and C. G. Ritson, eds.), pp. 1–13, Luniver Press, 2011.
[11] C. Adami, "Digital genetics: unravelling the genetic basis of evolution," *Nature Reviews Genetics*, vol. 7, pp. 109–118, 2006.
[12] `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html`.
[13] `http://evoevo.liris.cnrs.fr/`.
[14] A. Crombach and P. Hogeweg, "Chromosome rearrangements and the evolution of genome structuring and adaptability," *Molecular biology and evolution*, vol. 24, no. 5, pp. 1130–1139, 2007.
[15] A. Crombach and P. Hogeweg, "Evolution of evolvability in gene regulatory networks," *PLoS Computational Biology*, vol. 4, no. 7, 2008.

A CoSMoS Patterns

The patterns briefly summarised here are in alphabetical order and will appear in full in the forthcoming publication [2].

▶ Cartoon

Produces an informal sketch that identifies system components in domain specific terms. It provides a step towards more formal modelling.

▶ CoSMoS Simulation Project

Develop a fit-for-purpose simulation of a scientific or engineering system. Application of this pattern results in the construction of the products shown in figure 1 from the application of the Discovery Phase, Development Phase and Exploration Phase patterns.

▶ Data Dictionary

Defines the modelling data that will be used to parameterise the Simulation Platform, and relevant real-world data that will form the basis for comparison to data produced from simulation experiments.

▶ Development Phase

Based on the outputs from Discovery Phase, produces the simulation platform upon which simulation experiments can run. Will require the completion of the Platform Model and Simulation Platform patterns.

▶ Discovery Phase

Establishes what simulation platform needs to be built. Other patterns required to complete this pattern include the Research Context, Domain and Domain Model patterns detailed below.

▶ Document Assumptions

Records assumptions appropriately to ensure they are explicit and justified. For each assumption record its nature, criticality, reason for existence and a justification such that its consequences are understood.

▶ Domain

A particular view or perspective of the real world system under study.

▶ Domain Experiment Model

A model that explicitly captures the experimental system that is applied to the domain components identified in the Domain Model. This model identifies the domain variables and how we manipulate and record them.

▶ Domain Identification

Identifies the Domain (see figure 1), establishing the perspective on the real world system of study used as the basis for simulation. This pattern makes use of the Cartoon and Document Assumptions patterns.

▶ Domain Model

A model that explicitly captures aspects of the domain, identifying and describing the structures, behaviours and interactions. It is a model based on the science and is free from any simulation implementation details.

▶ Domain Modelling

Generates the Domain Model (see figure 1) and Domain Experiment Model, which forms the scientific description of the identified Domain. This pattern makes use of the Cartoon, Modelling Approach, Document Assumptions and Data Dictionary patterns.

▶ **Domain Researcher**

Identifies the domain researcher who is the point of contact for domain knowledge, providing the interpretation of the domain literature.

▶ **Exploration Phase**

Uses the outputs from Development Phase to run simulation experiments to investigate the questions identified during Discovery Phase.

▶ **Modelling Approach**

Identifies an appropriate approach and notation for producing a model. Takes account of suitability and understandability with respect to the modelling to be performed.

▶ **Platform Implementation**

Generates the Simulation Platform (see figure 1), which incorporates a software and hardware platform capable or running simulations of the implemented Platform Model.

▶ **Platform Model**

A model derived from the domain model that details how the concepts captured in the domain model will implemented by the simulation platform.

▶ **Platform Modelling**

Generates the Platform Model (see figure 1) detailing how the Domain Model and Data Dictionary concepts will be implemented and analysed in the Simulation Platform product. This pattern makes use of the Modelling Approach and Document Assumptions.

▶ **Research Context**

Identifies the scientific context of a simulation-based research project, establishing its scope, purpose and success criteria.

▶ Research Context Diagram

A Cartoon that provides an overview of the behaviours being modelled, highlighting the components hypothesised to play a significant role in the real-world phenomena and the system-level behaviours that result from their interactions.

▶ Simulation Platform

Encodes the platform model into a software and hardware system upon which simulation experiments are run, which in turn generate the results.

B Platform Model Diagrams

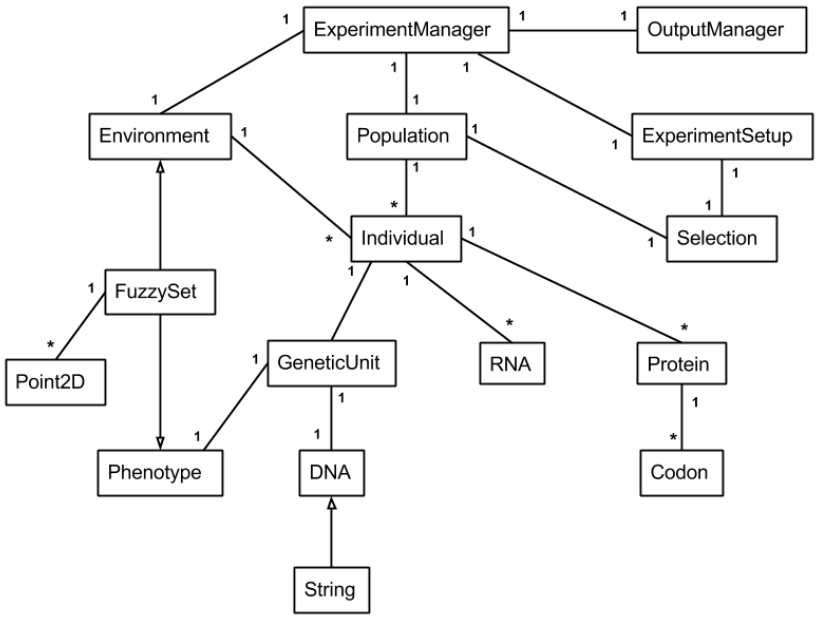

Fig. 5. Aevol platform model classes.

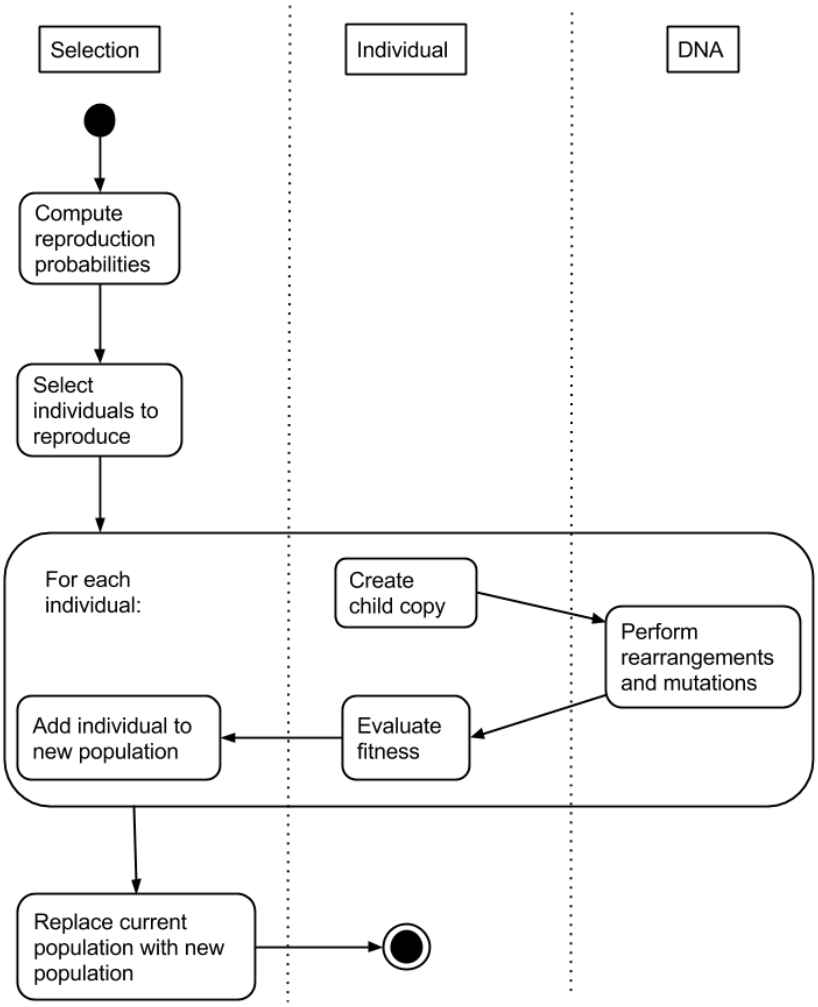

Fig. 6. Aevol platform model selection activity.

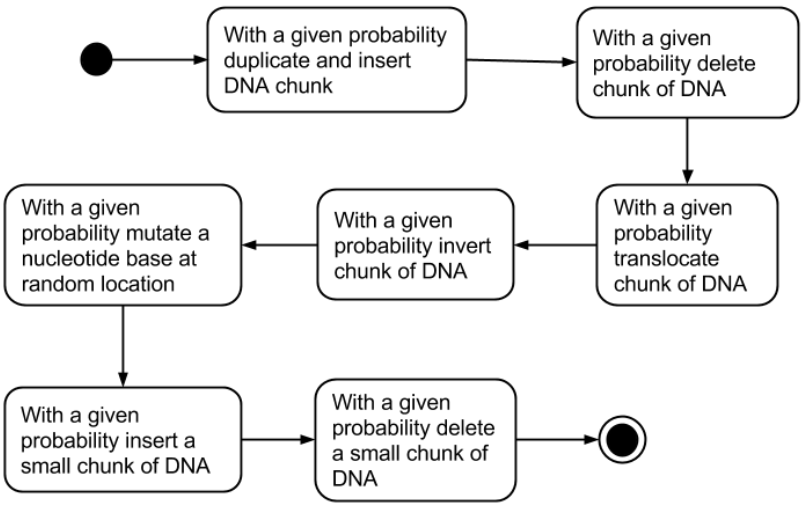

Fig. 7. Aevol platform model mutation activity.

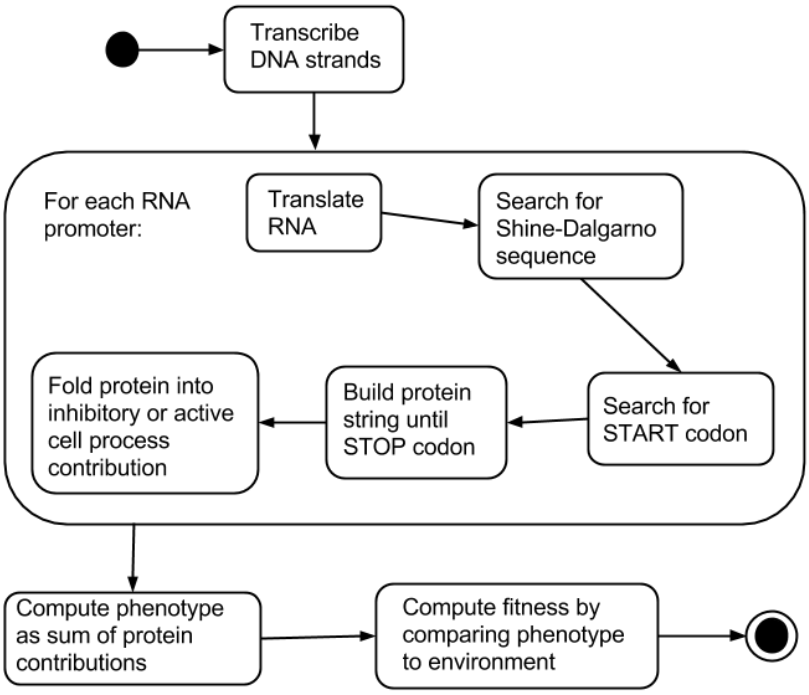

Fig. 8. Aevol platform model evaluation activity.