# Open-Endedness: Definitions and Shortcuts

Susan Stepney, Guillaume Beslon

# Open-Endedness: Definitions and Shortcuts

Susan Stepney[1] and Guillaume Beslon[2]

[1] Department of Computer Science, and York Centre for Complex Systems Analysis, University of York, YO10 5DD, UK
[2] Université de Lyon, INSA-Lyon, INRIA Beagle, CNRS LIRIS UMR5205, Villeurbanne, France

## 1 Introduction

The open-endedness of a system is often defined as *the continual production of novelty*. Here we report on recent work by Banzhaf et al. [1] that pins down this concept more rigorously. We define several types of novelty that a system may exhibit, and classify these as *variation*, *innovation*, and *emergence*. We also define an architecture suitable for building simulations of open-ended novelty-generating systems.

## 2 A meta-model for open-ended systems

Our definition of open-endedness (OE) is relative to the model and meta-model of the system under investigation. The concepts of "model" and "meta-model" are used for building both scientific and engineering models of systems: *Models* provide an abstract language for the relevant concepts. A scientific (descriptive) model is used when observing and experimenting on a (natural or artificial) system; An engineering (prescriptive) model (often a computational model) is used to design and implement an artificial system. In the case of a computational model, the model defines the concepts to be implemented in code. *Meta-models* provide the analogous abstract language to define models, comprising the concepts that can be used to build the model. For example, in an object-oriented system, the meta-model would contain the concepts of 'class', 'object', 'method', 'association', and so on.

There are many possible models and meta-models for capturing and analysing the behaviour of systems. These models and meta-models may be *implicit* or *explicit*. Here we assume that we will implement and analyse agent-based (or entity-based) systems. A suitable meta-model for such an analysis, which allows us to capture intuitions about major transitions, contains the following concepts (figure 1):

- **Entity**: an identifiable integrated whole within the model: a "thing" with structure (organisation) and behaviour (activity, processing). An entity may be an **atomic entity**, with no (modelled) internal structure, or a **system entity**, composed of internal components.

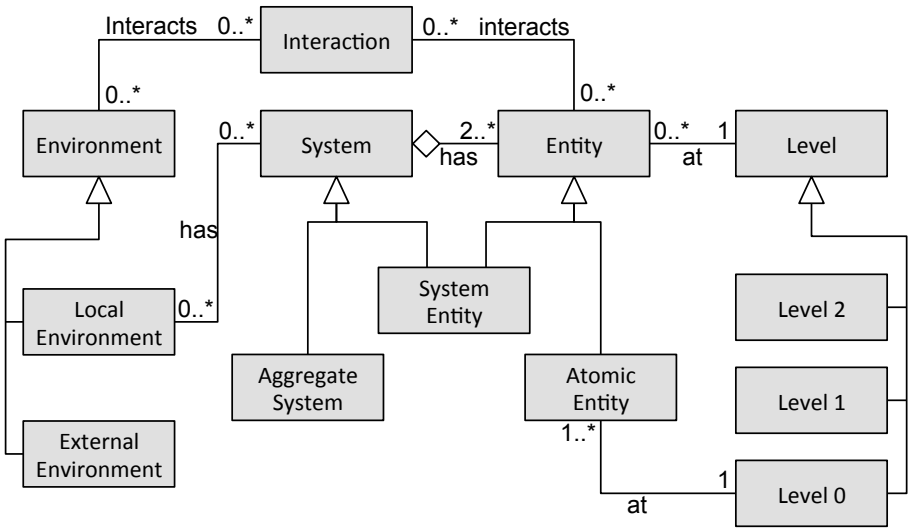


**Fig. 1.** Our meta-model, written in UML, defining the concepts to be used in any model conforming to it. Boxes are classes from which objects in the model can be instantiated. Links indicate associations between classes (associations being named and valued), arrows indicate inheritance ("kind of"), diamonds indicate aggregation. An Environment can be a Local Environment or External Environment. A System can be an Aggregate System or a System Entity. An Entity can be a System Entity or an Atomic Entity. A System has a Local Environment and two or more Entities. An Entity exists at both a system Level and an information structure (model) Level. An Atomic Entity exists at Level 0. Many Environments and Entities can be involved in each Interaction.

- **Environment**: part of the domain of interest not modelled as explicit entities, for example: space, fields, flows.
- **System**: a local environment plus a collection of interacting entities, forming some identifiable whole. A system may be an **aggregate system**, comprising a collection of entities in a local environment but not considered to form an entity in its own right; or a **system entity**, modelled as an entity at a higher level than its component entities.
- **Interaction**: entities interact with each other and with their environment, potentially forming systems
- **Level**: level-0 entities are atomic entities; level-$N > 0$ entities are system entities that contain lower-level entities, including at least one level-$(N-1)$ entity.

This meta-model can be used to provide the concepts used in building specific entity-based models of system.

## 3 Definitions

We define types of novelty and OE *with respect to the system's current model and meta-model*. *Novelty* in an observed system is classified as:

0. *Variation: novelty within the model.* Variation changes an instance of the model, such as a change to the values of a variable that exists in the model, without changing the model itself. Variation explores a pre-defined (modelled) state space, producing new values of existing variables.
1. *Innovation: novelty that changes the model.* Innovation changes the model: for example, adds a new type or relationship (that conforms to the meta-model), or eliminating an existing one. Innovation changes the combinatorics and the size/structure of the state space, thereby growing/shrinking the possibilities of variation.
2. *Emergence: novelty that changes the meta-model.* Emergence changes the meta-model: a change that adds a new meta-type or relationship, or possibly eliminates an existing one. A change to our meta-model that adds a new level is a *major transition*. Another change to our meta-model might include the addition of the concept of *process* as a first-class thing.

An *open-ended event* is an event that results in innovation or emergence. An *open-ended system* is a system with the ability to continually produce open-ended events.

## 4 Implementation shortcuts

To study a system such as the ones captured by our meta-model, we need to be able to simulate multi-level systems. We may need to introduce new components in our simulation to simplify or accelerate the dynamics of the higher levels. We call these new elements **shortcuts**, since they directly implement some properties of the higher level that, in an ideal simulation, would emerge from the generative level-0 entities. Shortcuts are hard-coded design optimisations manually introduced into the simulation. See figure 2.

Shortcuts provide optimisations by explicitly constraining structures and behaviours at their level, rather than requiring these constraints to emerge from the system's behaviour. Hence, specific shortcuts will enable or constrain certain classes of open-endedness.

When designing the simulation of an open-ended system, one crucial design step is then the identification and implementation of relevant shortcuts. These will be research-dependent: they will depend on the specific question one wants to answer with the simulation. Examples of shortcuts used (usually implicitly) in simulations are:

- **Individuality**: entities at level $N > 0$ are hard-coded rather than emergent.
- **Replication**: the explicit implementation of operators that replicate the entities at level $N$ external to the entities.

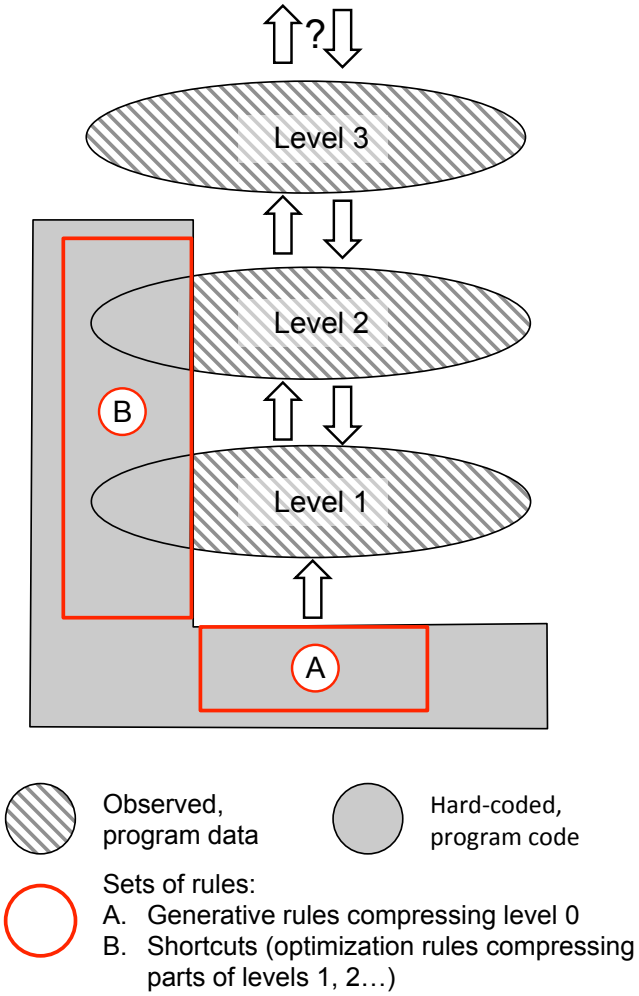


**Fig. 2.** Simulation with shortcuts. Levels 0 is abstracted by the set of generative rules encoded in "A". Levels 1 and 2 are predefined by the simulation (parts of them are hard-coded) but they include emergent parts that enable variation, differences or innovation. Level 3 is fully emergent, not at all included in "B" (but may be anticipated by the modelers when designing "A" and "B" since they want to observe 3). If level 3 is a system made of entities of lower levels, then its emergence is a transition. If the level 3 entities are replicators, then emergence of level 3 is a major transition.

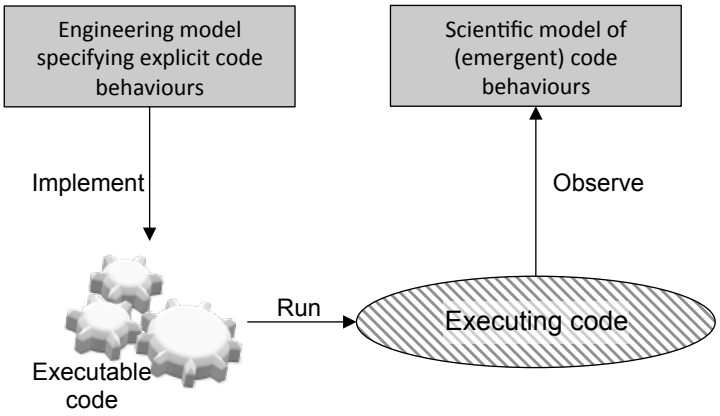


**Fig. 3.** Two kinds of computational model. An engineering model is used to specify the software to be implemented. A scientific model is derived from observations of the execution of the software. These models can have different structures, even about the same software.

– **Fitness**: replacement of the differential reproductive success emerging from the difference between entities by an explicit computation of reproductive success according to some target task.

## 5 Open-ended simulations

We have defined OE in terms of innovation and emergence: changes to models and meta-models. In classical software simulations, a model is designed (say, in UML) conforming to some meta-model (e.g. of object orientation, or agent-based systems), and then implemented. The running code conforms to the model. That model does not change as the system runs. Therefore, can such simulations ever exhibit innovation or emergence? Can they be open-ended?

The short answer is that they exhibit innovation and emergence. The implementation model is an *engineering model*. However, the running system can be analysed with a *scientific model*, and this model *can* change. (See figure 3.) Consider Conway's Game of Life. The engineering model consists of dead/alive cells, which interact with their neighbours. This model is (typically) fixed. However, the model used to analyse the running system can include concepts such as blocks, and gliders, and other higher-level components. These higher levels are *emergent*, according to our definition.

The longer answer, however, is that they probably cannot exhibit OE, i.e. the continual production of open-ended events. The reason is that the code, conforming to the fixed model, cannot directly exploit the (scientifically modelled) innovations and emergents. It may well be the case that changes to the scientific (observational) model need to be fed back into changes to the engineering model. Steps along the way include:

1. The emergent novelty is recognised outside the simulation, by analysing the scientific model.
2. The emergent novelty is anticipated and recognised, using pre-coded recognisers present in the code (hence in the engineering model), available to report it once it appears.
3. The type of emergent novelty is anticipated and captured, using pre-coded rules available to recognise it, and make it a component of the simulation once it appears.
4. The emergent novelty is somehow emergently recognised by the simulator, and new code is generated by the simulator to capture the recognition.
5. The emergent novelty is somehow emergently captured by the simulator: once recognised, new shortcut is generated by the simulator to capture the specific emergence.

How to realise the later steps remains an open research question. Until then, simulations can use hard-coded shortcuts to overcome some of these issues.

## 6 Conclusion

We have defined open-endedness in terms of novelty, and identified three classes of novelty in terms of models and meta-models. We have sketched a meta-model for entity-based simulations, and a shortcut architecture for efficient implementation. We have outlined research requirements for fully open-ended simulations that can incorporate model changes and meta-model changes automatically.

### Acknowledgments

We acknowledge funding from the European Commission (FP7-ICT-2013.9.6 FET Proactive: Evolving Living Technologies) EvoEvo project (ICT-610427). We are grateful to Wolfgang Banzhaf, René Doursat, and the rest of the team behind [1], for their contributions to the ideas outlined in this abstract.

## References

1. Wolfgang Banzhaf, Bert Baumgaertner, Guillaume Beslon, René Doursat, James A. Foster, Barry McMullin, Vinicius Veloso de Melo, Thomas Miconi, Lee Spector, Susan Stepney, and Roger White. Defining and simulating open-ended novelty: Requirements, guidelines, and challenges. *Theory in Biosciences*, 2016. doi: 10.1007/s12064-016-0229-7, 54pp.