EvoEvo Deliverable 4.5

Reflective applications

Due date:	M36 (October 2016)
Person in charge:	Susan Stepney
Partner in charge:	University of York (UoY)
Workpackage:	WP4 (A computational EvoEvo framework)
Deliverable description:	Reflective application: A report containing a description of a CoSMoS approach "Simulation Experiment": a description and evaluation of a prototype reflective application built using the Computational reflective run-time platform (D4.4) as the evoevo component.

Revisions:

Revision no.	Revision description	Date	Person in charge
0.1	Initial release for comment	24/10/16	Simon Hickinbotham (UoY)
1.0	Release version	28/10/16	Susan Stepney (UoY)
1.1	Code repository locations added	9/11/16	Susan Stepney (UoY)

Table of Contents

1. Introduction

This document describes the prototype applications developed using concepts from the reflective EvoEvo approach described in D4.4.

2. Application 1: EvoEvo in automata-based R-P models

Rather than design a wholly new system, we focussed here on using EvoEvo principles to increase the stability of our existing Stringmol system (see D4.4 for an overview of stringmol). Much of the work in WP3 is concerned with *Replicator-parasite* (RP) systems, in which evolving agents are able to survive parasitism by virtue of their spatial distribution. This results in higher levels of selection, where `waves' of replicators are driven across the arena by parasites.

By implementing a stringmol replicase system in a spatial arena, we were able to avoid extinction by this parasite mechanism (Hickinbotham & Hogeweg 2016). In addition, we have observed novel behaviours emerging. The new replicative behaviours are described in Deliverable 3.4.8.3, where the interactions are studied from a *biological* perspective. The relevant *computational* components of this system are noted here. These are:

- **Longer interaction times.** A feature of the evolved replication is that program fragments interact over longer timescales, due to double execution of the copying loop. This is a strategy for negating parasites, because they must wait for the molecule to scan itself before getting copied, which slows their replication disproportionately to replicators.
- **Re-purposing of code fragments**. These systems are seeded with a hand-designed copying program which binds to other instances of itself and copies them. The programs have bind sites and copying regions. What emerges towards the end of experiments is a *repurposing* of these regions. The bind site is no longer used to recognise ‘self’ - rather it is tuned to recognise a ‘partner’ replicating molecule, which makes the landscape more rugged in terms of parasitism. We have also observed instances of the use of the copying region as a binding site. This means parasites cannot bind *unless they have code for copying* - again reducing the ability of parasites to destroy the system.
- **Higher level organisation**. The addition of the spatial component to the stringmol model allows new levels of organisation to emerge, since the products of local interactions are only available locally.

3. Application 2: Live coded music

Reflective EvoEvo is most suited to situations where the application environment is constantly changing, and in which the system can interact in a dynamic way with that environment. One of the attractions of working in *musical* systems is that time must be considered implicitly, and the dynamics of the performance must be accommodated. In addition, it is desirable that the application involves the direct use of code fragments. For these reasons, we decided to study evolutionary experiments in *live coding* (McLean 2014) of computer-generated music (Hickinbotham & Stepney 2016a).

Live coding is the use of domain-specific languages (DSLs) to improvise new musical pieces in a live concert setting. The code can be edit on the fly whilst the music is playing because the

interpreter is robust enough to detect bad syntax before attempting to generate audio from the intended patterns.

Audiences at live coding events are very open-minded, and are used to 'glitches' appearing as the performance continues. As long as unsympathetic patterns are removed, than that is accepted as part of the experience. These are the ideal conditions for experiments with music evolution.

We have developed a new reflective evolutionary system that *augments* this live coding process by generating and maintaining a *population* of coded musical patterns that are interactively expressed as audio during the performance. The population of evolved patterns changes gradually with the piece, and provides a reference point and storage for a changing bank of aesthetically pleasing patterns. The system allows the actions of the human coders to be fed back to the population as adjustments to the fitness of patterns, resulting in a novel musical interactive genetic algorithm (MIGA) (Rodriguez & Vico 2013).

Our system is sufficiently flexible to allow the generation of pieces using evolved patterns alone, or any mix of evolved patterns and manually configured patterns. It is accessed through an extension to the live coding approach as supported through the Extramuros browser-based platform (Ogborn et al 2015). Extramuros is a system that allows browser-based collaboration of a group of performers on live-coded pieces of music and graphics across a network. The availability of this system makes it easier to use automation to suggest patterns for use and further manipulation by the coder. An evolutionary process is an ideal candidate for the automatic generation of new coding patterns within this context.

We selected Tidal (McLean 2014) as our live coding language because it is relatively well documented and straightforward to install in an Extramuros framework. The live coding approach allows more immediate interaction between human and automaton. Live coding languages are more like conventional computer code than other musical notation systems but with more emphasis on brevity and ease of editing than is common in programming languages, sometimes at the expense of clarity.

Live coding requires a text-based grammar to encode the musical sounds, designed to be constantly manipulated rather than simply written once and interpreted many times. This grammar is amenable to a genetic programming approach to evolving systems. We have developed this application as a means of applying ideas from EvoEvo to the generation of mutations in these Tidal music fragment patterns (Hickinbotham & Stepney 2016a). The mutations are based on the Tidal musical grammar: incoming code fragments are converted to parse trees, which allows mutations to be generated that conform to the grammar.

Two example videos of our system being used to evolve live coded music are available at www.youtube.com/watch?v=UXAMjvPn_VY and www.youtube.com/watch?v=iCTOtfxpElU. In each of these, it takes some time to evolve interesting patterns, So, in the second video, to start listening at a time when the music has had sufficient chance to evolve, start at youtu.be/iCTOtfxpElU?t=705. In each of these examples, the 'pullmut' function (where the user requests a mutation of the currently selected patter) is the main source of innovation in the patterns, with the occasional hand-coding or editing of evolved patterns by the user. These performances compare favourably with hand-coded Extramuros-based performances such as those at www.youtube.com/watch?v=zLR02FQDqOM, especially when it is recognised that the users of the evolved system have little or no previous experience of live coding. We currently have no means of quantitatively comparing live coding performances, however.

In that delivered implementation, the rates and types of mutation that could be generated are hard-coded. This is the first stage of development of a system that uses a reflective architecture to generate mutations. Future work should incorporate ideas of mutable mutation rates from EvoMachina (D4.3)

4. Application 3: Co-evolved music and dance

Our system is also amenable to using a range of different methods of interaction to drive the evolutionary process. A collaboration of York and Lyon INRIA EvoEvo partners resulted in an architecture called *commensal computing* (Abernot et al 2016), in which a motion-based control system is *co-evolved* with the generated musical patterns. This complements the dance application developed in WP5.

This approach maps onto a model of creativity called “Creative Systems Framework", derived by Wiggins from the work of Margaret Boden (Wiggins & Forth to appear); this also has analogues to our bio-reflective architecture (Hickinbotham & Stepney 2016a).

5. Code locations

The stringmol code repository, including spatial stringmol, is at github.com/franticspider/stringmol

The tidalGE code repository is at github.com/franticspider/tidalGE

Two example evolutions are given at www.youtube.com/watch?v=UXAMjvPn_VY and www.youtube.com/watch?v=iCTOtfxpElU

6. Conclusion

The core of this workpackage concerns selecting or designing an appropriate *language* for both supporting *bio-reflection* and for implementing *applications*. We have examined two approaches: using a single language (stringmol) to evolve R-P systems, and using a Domain Specific Language (Tidal) as the link between bio-reflection and the application environment. This approach has allowed us to gain understanding of how bio-reflection runs evoevo ‘on itelf’, and how the outputs of bio-reflection can be used in a time-varying application.

The results of this approach are that we have a new level of understanding of how a bio-reflective system can be implemented: by adding a spatial component to the stringmol architecture we have a more open-ended system in which to continue our research into bio-reflection. However, it is difficult enough to design a *reflective* language that simply maintains itself even without generating suitable outputs to drive an application. It would be wasteful to have to redesign the bio-reflective language for each application. A more efficient approach would be to develop standard ways to interface the reflective language with the application language. The biological analogy is with the expression of proteins: the DNA/RNA component captures the *reflection*, but the protein component forms the active *machine* that carries out the task. Our work in live coding illustrates the merits of this approach, with which new ways of interacting with evolving system can be produced. Thus a future implementation of the bio-reflective architecture in the EvoMachina framework (D4.3) could be fruitful.

7. References

References marked with (*) were produced wholly or in part by the EvoEvo project.

(*) Jonas Abernot, Guillaume Beslon, Simon Hickinbotham, Sergio Peignier, Christophe Rigotti. A Commensal Architecture for Evolving Living Instruments. In *Conference on Computer Simulation of Musical Creativity, Huddersfield, UK*, June 2016.

(*) Simon Hickinbotham, Paulien Hogeweg. Evolution towards extinction in replicase models: inevitable unless.... In *2nd EvoEvo workshop, Amsterdam*, 2016.

(*) Simon Hickinbotham, Susan Stepney. Augmenting live coding with evolved patterns. In *EvoMusArt, Porto, Portugal, March 2016*, volume 9596 of LNCS, pages 31-46. Springer, 2016a.

(*) Simon Hickinbotham, Susan Stepney. Bio-reflective architectures for evolutionary innovation. In *ALife 2016, Cancun, Mexico, July 2016*, pages 192-199. MIT Press, 2016b.

Jose David Fernández Rodriguez, Francisco J Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513-582, 2013.

Alex McLean. Making programming languages to dance to: live coding with Tidal. In: Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design. pages 63–70. ACM, 2014

D. Ogborn, E. Tsabary, I. Jarvis, A. Cardenas, A. McLean: Extramuros: making music in a browser-based, language-neutral collaborative live coding environment. In: A. McLean, T. Magnusson, K. Ng, S. Knotts, J. Armitage (eds.) *Proceedings of the First International Conference on Live Coding*. p. 300. ICSRiM, University of Leeds, 2015

Geraint A Wiggins, Jamie Forth. Computational creativity and live algorithms. In Alex McLean, Roger Dean (eds) *Oxford Handbook on Algorithmic Music*, in preparation.