Towards adaptive self-aware software

Peter Andras (1), Bruce Charlton (2)
(1) School of Computing Science
(2) School of Biology / Psychology
University of Newcastle
Newcastle upon Tyne, NE1 7RU, UK

Abstract

Self-aware adaptive software is a saint Grail of computer science. Recent advances are very promising, and software systems have several features required for being self-aware and adaptive (e.g., introspection and self-modification). However no truly self-aware adaptive software system exists currently. We analyse biological and social systems as abstract communication systems, building an analogy with software systems (i.e., systems of communicating objects). We highlight three critical features of natural systems: the asymmetry of true/false, the growth by correctness checks, and error-induced adaptation based on self-monitoring. Based on the established analogy we propose to build software systems with non-terminating proofs of correctness, to achieve self-aware adaptive software.

Since the beginning of computer science researchers aimed to build systems adaptive self-aware software systems that are comparable with animals in terms of self-awareness and adaptability [1, 2]. Recent progress in this direction has been significant. In particular, object oriented thinking led to systems with features required for adaptive self-awareness. Such features include the ability for introspection and self-modification (i.e., reflection [3]), the ability for dynamic integration, re-use of components and use of integration patterns (i.e., component-based programming [4] and use of design patterns [5]), and the ability for improved self-monitoring and action selection based on self-monitoring (i.e., aspect oriented programming [6]).

However, self-aware adaptive software systems comparable to natural systems are not yet available. In our view to make such systems possible we need further paradigmatic changes in software theory and design. We suggest that the needed change is to aim to build programs and software systems with non-terminating proofs of correctness.

Biological systems, like cells or animals, are composed of a very large number of constituents. These components interact with each other in well-defined ways forming patterns of interactions, which define the biological system [7]. In case of social systems [8] humans interact with other humans forming patterns of human communications obeying to a variety of regularities. We can see these systems as a self-aware, adaptive, self-reproducing and expanding abstract communication system made of patterns of communications obeying a set of rules, which define the system [9]. This leads to an analogy with object oriented software systems, made of a multitude of objects, which constitute the system by producing communication patterns between themselves, according to the rules of the software system [9].

Analysing natural systems in terms of abstract communication systems reveals critical features of them, which are closely linked with their adaptive self-aware nature. These features are: (1) the asymmetry of

true/false or the Popper principle, i.e., communications can be proven to be false (incorrect) by not leading to further continuation of the system, while the opposite never can be proven; (2) growth by correctness checks, i.e., the system growth by generating new communications that check the validity of earlier communications; (3) self-monitoring and error-induced adaptation, i.e., the system monitors itself by correctness checking communications and in case of errors (large scale halting of continuation of communications) the blueprint of the system is modified such that errors are avoided in the future. In our view these features are needed in truly self-aware adaptive software systems, and by analogy with natural systems, such features may be achievable if software systems aimed to have non-terminating proofs.

Acknowledgement: The authors thank Matthew Pocock for very useful discussions.

References

1.. Landauer C, Bellman, KL: Self-modelling systems. In: LNCS 2614 2001; pp 238-256.

2. Ungar, D, Smith, RB: SELF: The power of simplicity. LISP and Symbolic Computation 1991; 4: 187-205.

3. Bracha, G, Ungar D: Mirrors: Design principles for meta-level facilities of object-oriented programming languages. In: Proceedings of OOPSLA'2004; pp 331-344.

4. Jarzabek, S, Knauber, P: Synergy between component-based and generative approaches. ACM SIGSOFT Software Engineering Notes 1999; 24: 429-445.

5. Gamma, E, Helm, R, Johnson, R, and Vlissides, E: Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA, Addison-Wesley, 1995.

6. Katara, M, Katz, S: Architectural views of aspects. In: Proceedings of AOSD 2003; pp 1-10.

7. Miller, JG: Living Systems. McGraw-Hill, 1978.

8. Luhmann, N: Social Systems. Palo Alto, Stanford University Press, 1996

9. Andras, P, Charlton, BG: Self-aware software – Will it become a reality ? In: Babaoglu, O et al (eds) SELF-STAR, LNCS 3460 2005; pp 229-259.