KNOWLEDGE-BASED DESIGN AND EVOLUTIONARY DESIGN: ADVERSARIES OR COMPATRIOTS?

FRANCES M.T. BRAZIER, SANDER VAN SPLUNTER, AND NIEK J.E. WIJNGAARDS
Intelligent Interactive Distributed Systems,
Faculty of Sciences, Vrije Universiteit Amsterdam
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: {frances, sander, niek}@cs.vu.nl
http://www.iids.org

1   Introduction

Software agents operate in dynamic environments (Jennings, 2000). Within its environment an agent, however, may also be a dynamic artefact: a dynamic artefact that may autonomously decide to adapt, or to be adapted due to changes in its environment. These adaptations may be performed by the agent itself, i.e., a self-modifying agent, or by an external service, e.g., an agent factory. It may be up to the agent to choose the service provider. In all cases adaptive agents need to be prepared for adaptation when first created (e.g. compositional design is one of the prerequisites).

In evolutionary computation (Saunders and Gero, 2001) agents are often seen as individuals within a population, where individual agents can be adapted or combined to form new agents. An important difference between an agent factory and evolutionary computation is that the agent factory is based on explicit knowledge, and evolutionary computation on implicit knowledge or none at all (completely random).

The goal of this position paper is to discuss both approaches with respect to how evolution and adaptation are dealt with. Potential 'crossovers' between the two approaches are explored.

2   Automated Agent Adaptation

An *agent factory* (Brazier and Wijngaards, 2001; 2002) is an automated agent servicing facility for the re-design of agents. Agent descriptions are specified at two levels: conceptual and operational. A conceptual description of (parts of) an agent is an architectural description: a blueprint of the components, interfaces and interactions between components. An operational description includes code, together with definitions of e.g. interfaces. A mapping (not necessarily structure preserving) between these descriptions defines the relationship between the elements at one level with elements at the other. The re-design process is a configuration process of a conceptual blueprint and an operational blueprint.

Reasoning required to (re-)design a software agent includes strategic knowledge on how and when to focus on specific aspects/characteristics of an agent. Different viewpoints on the blueprints of an agent are considered, for each of which strategic knowledge can be specified (Brazier, Splunter and Wijngaards, 2001).

Disadvantages of this explicit knowledge approach include the high dependency on availability of building blocks, assuming sufficient richness of the annotation, and the knowledge that solutions may be sub-optimal (depending on quality of available explicit knowledge). The main advantage is that the (re-)design process is strongly guided and traceable so that the design can be rationalised.

Reasoning in, e.g., *genetic algorithms* (Koza, 1992), highly depends on the definition of genes, fitness selection, crossover, and mutation. These factors are pre-determined before running the algorithm. A characteristic is that evolutionary algorithms search for optimal solutions, and each individual is a possible solution. Often individuals are not adapted, but new solutions are created from existing ones. In most cases the individuals have no autonomy in the decision on whether they are to be adapted, or in what way. Note that not all knowledge is implicit, explicit knowledge includes the fitness function, the definition of the crossover and mutation, and the representation of a solution in genes. These items are part of the strategic knowledge of evolutionary algorithms. Advantages include the wide applicability fast exploration of a search space, and avoiding local maxima. Disadvantages include: computationally expensive, may need parameter tuning (e.g. agent creation by Spector and Robinson, 2002), and a weak theoretical basis.

3   Explicit knowledge in Evolving & Adaptive Systems

As both agent factories and evolutionary algorithms may be used to evolve agents, it may be worth exploring the options of "cross-over" for their mutual benefit.

3.1   APPLYING AN AGENT FACTORY IN EVOLVING & ADAPTIVE SYSTEMS

In evolutionary algorithms the *manipulations* are uninformed. Ann agent factory enables informed manipulations. If an agent factory performs cross-overs and mutations, it may be easier to include domain knowledge. These informed manipulations may, however, be computationally more expensive. Genes are often kept simple, e.g., Lipson and

Pollack (2000) use in total two simple building blocks of which physical machines are automatically designed. An additional advantage of using an agent factory is in the complexity of the descriptions of the individuals: (heterogeneous) complex artefacts with highly interdependent subparts can be difficult to represent in normal evolutionary algorithms. An agent factory can handle these interdependencies, when they are made explicit, thereby circumventing uninformed manipulations resulting in 'impossible solutions'. This implies that populations that have a heterogeneous genetic structure, can become pools on which evolutionary algorithms can function.

3.2 APPLYING EVOLVING & ADAPTIVE SYSTEMS IN AGENT FACTORIES

The agent factory can benefit from concepts from evolutionary algorithms. The crossover manipulation is interesting from the view of agents. For example, an agent may request an ability perceived in another agent. An agent factory could integrate the essential part of the blueprint of the other agent with the blueprint of the current agent, extending its ability with the requested one. Alternatively, two or more agents may together approach an agent factory and require the creation of a new agent, e.g. a mediator agent to help in co-ordinating work. The requesting agents may need to be adapted to be able to co-operate with the new mediating agent. A fitness function may be of help for an agent factory to evaluate (intermediary) designs and effectiveness of strategic knowledge. An agent factory may use evolutionary algorithms with a fitness function to produce better blueprints of agents. If possible, and depending on theoretic results concerning evolutionary algorithms, an agent factory may even 'learn', and formulate explicit knowledge on the basis of (un)successful modification experiments based on evolutionary algorithms. The interoperability of building blocks plays an important role in the aforementioned possibilities.

4 Discussion

Evolutionary algorithms and knowledge-based approaches are often viewed as contradictory methods in design, requiring a different formulation of the design problem. Both approaches have their strengths and weaknesses. A combination of these two approaches may lead to interesting results. For the research on our agent factory new fields of application are examined, exploring possible improvements, and uncovering previously unnoticed assumptions. Evolutionary computing may also benefit, as described above.

Acknowledgements

References

Brazier, F.M.T., Splunter, S. van, and Wijngaards, N.J.E.: 2001, Strategies for integrating multiple viewpoints and levels of detail, *in* J.S. Gero and K. Hori (eds) *Strategic Knowledge and Concept Formation III, Key Centre of Design Computing and Cognition*, University of Sydney, 103-128.

Brazier, F. M. T. and Wijngaards, N. J. E.: 2001, Automated Servicing of Agents, *AISB Journal*, Special Issue on Agent Technology, 1(1), 5-20.

Brazier, F. M. T. and Wijngaards, N. J. E.: 2002, Automated (Re-)Design of Software Agents, *in Proceedings of the Artificial Intelligence in Design Conference 2002*, to appear.

Jennings, N. R.: 2000, On agent-based software engineering, *Artificial Intelligence*, **117**, 277-296.

Koza J.R.: 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Massachusetts.

Lipson, H., and Pollack, J.B.: 2000, Evolution of Physical Machines, *in* J. S. Gero (ed), *Artificial Intelligence in Design '00*, Kluwer Academic Publishers, Dordrecht, pp. 269-285.

Saunders, R and Gero, J. S.: 2001, Artificial creativity: A synthetic approach to the study of creative behaviour, *in* J. S. Gero and M. L. Maher (eds), *Computational and Cognitive Models of Creative Design V*, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, pp. 113-139.

Spector, L. and Robinson, A.: 2002, Multi-type, Self-adaptive Genetic Programming as an Agent Creation Tool, *in Proceedings of the Workshop on Evolutionary Computation for Multi-Agent Systems*, ECOMAS-2002, International Society for Genetic and Evolutionary Computation.