Organic Programming for Multi-Agents

Hideyuki Nakashima, Itsuki Noda, Ichiro Ohsawa
Electrotechnical Laboratories
1-1-4 Umezono, Tsukuba 305 Japan
nakashim,noda,ohsawa@etl.go.jp

Abstract

We are developing a new software methodology for building large, complicated systems out of simple units. In this paper, we describe application of an organic programming to multi-agent systems. One of the advantages resides in that we can program the system in a subsumptive manner.

Organic Programming

We are developing a new software methodology for building large complicated systems out of simple units(Nakashima 1991). The emphasis is on the architecture that is used to combine the units, rather than on the intelligence of individual units.

By "organic", we refer to the following characteristics of living organisms such as plants and animals: (1) Although all the cells have the same program (genetic information coded in genes), they behave differently according to the environment (including surrounding cells); and (2) Genes (viewed as programs) do not have the full information. The environment supplies the rest. Two key concepts of our approach are situatedness and reflection. Situatedness corresponds to the former property, and reflection to the latter.

An organic program(Nakashima, Noda, & Ohsawa 1995) consists of (1) processes: execution of a program in a certain environment, (2) cells: storage of fragments of programs, and (3) contexts for processes. The context is formed by a collection of cells. Each cell contains information on certain aspects of the environment, and the overall context is determined by the interaction of those fragments. Cells provide the following two functions to programs: (1) name to content mapping, and (2) keeping background conditions.

The process determines the structure of the context, and the context determines the meaning of the program, and thus the behavior of the process. In this sense, the process is reflective.

Subsumptive Programming

We selected soccer as our test bench because the game has many essential properties of multi-agent systems.

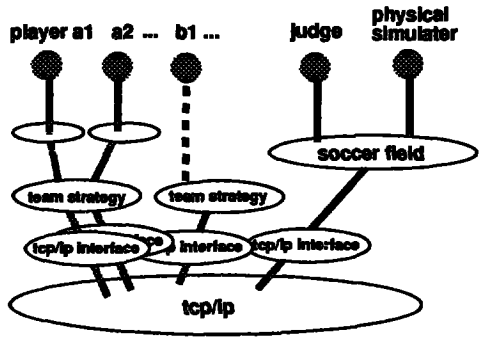

Figure 1: Distributed set up for a soccer game

Our implementation (figure 1) is designed for distributed environment where machines are connected via networks.

One advantage of this architecture is that we can program an agent in a way similar to subsumption architecture(Brooks 1988): (1) We can program a system by functional layers; (2) Programs in higher functional layers can override programs in lower layers.

The top-level process of an agent calls programs by their names. Those programs may exist in any layers. If more than one program of the same name exists, the program in an upper layer gets precedence over those in lower layers. If we use logic programming, there is a further possibility to backtrack to lower levels when a higher level program fails to perform its function.

References

Brooks, R. A. 1988. Intelligence without representation. Technical report, MIT.

Nakashima, H.; Noda, I.; and Ohsawa, I. 1995. Organic programming language gaea for multi-agents. TR-95-11, ETL.

Nakashima, H. 1991. New models for software architecture project. *New Generation Computing* 9(3,4):475–477.