Case Studies of Autonomy

Henry Hexmoor
University of North Dakota
Grand Forks, North Dakota, USA
hexmoor@cs.und.edu

Abstract

Autonomy is a quality for devices and creatures that perform tasks with relative independence from their designers and sources of authority. We discuss the idea and present two case studies to illustrate that the parameters for Autonomy are highly domain dependent.

1 Introduction

Autonomy is a quality that is desirable in systems that need to be long-lived and with little or no human maintenance. There are many examples such as the softball-sized flying robot designed to operate autonomously onboard manned and unmanned spacecraft in micro gravity, pressurized environments, (Gawdiak, et al, 1999). The recent debate on autonomy attempts to create a single metric that quantifies this quality, e.g., (Barber, Goal, and Martin, 2000). We believe these metrics are premature and may confuse the very concept. In this paper we present taxonomy of autonomy and discuss two different implemented agents.

At any given time, an agent forms a sense of Autonomy with respect to tasks, emotions, and mental states only if it can affect those things. An agent that has no ability to "move a rock" either by itself or getting someone or something else to do it has no autonomy with respect to moving the rock. Therefore, an ordinary rock that cannot do anything has no autonomy whatsoever. The agent forms its autonomy partly by factors that are completely internal to it and partly by perception of externally determined autonomy. We'll consider these self- and other-liberties, respectively introspective and interaction sources of autonomy.

Autonomy needs to be accompanied with two concepts: Context and Target. Context of autonomy is all the factors that will affect the autonomy and Target is the thing about which we have autonomy. We believe autonomy is best understood when we define a context and a target and the agent has abilities to affect the target. Otherwise, we can only make abstract or philosophical statements about autonomy and any general sense of metrics needs to be refined in order to be useful. To be more concrete in this paper, we chose to discuss case studies.

Let's first consider Autonomy that is an agent-internal quality. This is the introspective view of autonomy, (Castelfranchi, 1995, 1997; Hexmoor, et. al., 1999a). The agent determines a measure of liberty to pursue and to indulge in its own or delegated interests. Autonomy is the level of freedom it allows itself to form and to pursue its own mental states such as beliefs, goals, plans and its own emotions. For example, an agent that is self-confident usually allows itself to act more autonomously with respect to most tasks. An agent's perception of any number of emotions such as obligation, pressure, coercion will affect its autonomy with respect to its mental states. The internal autonomy of an agent is related to externally observable behavior that will appear to display a level of autonomy.

Let's now turn to autonomy that is determined external to the agent and may dynamically change by other agents. This is the interaction view of autonomy. An agent is permitted to have a measure of liberty to pursue and to indulge in its own or delegated interests. This is often called Social autonomy, (Sickman, et al, 1994). This notion of autonomy captures a sense of independence and power over the target of autonomy. Society at large, as well as a team of agents, creates deontic systems (i.e., systems of obligation) for each agent, involving role, authority, delegation, convention, obligation, and responsibility. These are some of the elements that make up a context for determining autonomy. In an organization with assigned roles and tasks, each agent has an externally developed level of autonomy. A boss who has to decide tasks for its agents may use may factors such as reliability, capability, and preference of its underlings to decide a level of autonomy with respect to a delegated task. We presented a preliminary conceptualization of this in (Hexmoor, Lafary, and Trosen, 1999b).

We believe autonomy can be implemented in three ways. The first is by off-line encoding or by design. Let's consider an example where the context is

empty and the target is all agent abilities. We can prescribe individuals to have either an externally imposed full autonomy about all tasks or to have no autonomy over any task and to follow a prescribed doctrine about all tasks. Autonomy levels of agents in a multi-agent system benefit problems differently. There has been some experiments that illustrate the tradeoff between autonomy and problem complexity, (Barber, Goal, and Martin, 2000; Hexmoor, Lafary, and Trosen 1999b). The second implementation method is to perform a fixed cost analysis and to change autonomy online. Our case studies in this paper illustrate simple cases of this online change. The third implementation method is by online learning the utility of given levels of autonomy.

In the remainder of this paper we will discuss two agents in completely different domains with different targets and contexts. Both of these agents show a changing level of autonomy.

2 Case Study I: Air Traffic Control Agent

We have written a program that assists a tower air traffic control operator. Just as the operator would, the agent monitors the nearby tower sky, processes requests for landing, and instructs the pilots with maneuvers that maintain safe separation distances and collision aversion. The human operator is informed of all agent activities. If there is a possible collision threat, a pop-up window gives details, shows the agent's choice of action, and the countdown time before the agent issues that command. The human operator may interrupt the agent and issue his own command (Hexmoor and Heng 1999a,b).

The agent's autonomy differs with respect of each of the two tasks of collision aversion and landing. These two tasks are two targets for autonomy.

The agent receives landing requests and based on configuration of planes in the processes of landing and perceived criticality of pilot's need to land, it generates landing priorities which ranges form 1-4 with 4 being the most urgent landing condition. In addition to priority, the agent must consider the traffic and before allowing a plane to land, the traffic ahead of it must be sufficiently clear. The agent uses the priorities and landing traffic for determining it autonomy. If the priority is 3 or 4, the agent assumes full (set to 1.0) autonomy. If the priority is 1 or 2, the agent pops up a window for the human operator. It then allows the user to override its decisions for 2 seconds. We use a counter (**t** in the equation below) that counts from 0 up to 2 seconds in 0.1 increments. This is summarized in the following equations.

If 3.0 <= LandingPriority <= 4.0:
Autonomy = 1.0

If LandingPriority < 3.0:
Autonomy = (LandingPriority/4.0) +
((($|$LandingPriority – 4.0$|$) * **t**) / **2**)

The agent attends to relative distances among planes. If two planes come near one another and there is prediction of collision, the agent will alert and generate evasive maneuvers. Depending of configuration of distances and predictions, the agent generates a priority number form 1-4. 4 signify the most dangerous condition of collision and the time is very critical and the human operator may not have time to react. In this case, the agent autonomy is set at its highest and it handles the collision. With lower numbers, the time to possible collision is larger and the human operator may want to intervene. So a window with a timer is presented to the human operator with typical times 5-20 seconds. Let **T** be the time to collision when a possible collision is first detected and let **t** be a counter that counts up to **T**. When **t** = **T**, Autonomy = 1.0. In general:

Autonomy = (CollisionPriority / 4.0) +
((($|$CollisionPriority – 4.0$|$) * **t**) / **T**)

3 Case Study II: Sheep-dog agent and the Shepard

We have written a program that simulates a sheep-dog following orders of a Shepard in Penning sheep. The dynamics of this scenario are commonplace in most organization of human or machines. Although it may appear as an esoteric application, the interactions between a Shepard and a sheep-dog transfer to other domains. You can imagine that a lead satellite may be delegating tasks to other satellites or to its interior components.

The sheep-dog may take high level or low level orders from the Shepard. We will focus on the task of Penning where the sheep-dog walks behind the sheep so the sheep is guided into a gated area. This consists of moving into a flanking position where the sheep is not afraid of the dog but the dog is in a dominant position. This dog's behavior is *Flanking*. Situated at flanking position, the dog moves slowly toward the sheep and in a circular pattern around the sheep but maintaining a distance called *balance*. While

maintaining the balance distance, the dog positions itself between the sheep and the gate and moves toward the sheep. This is called *Driving* the sheep. Penning is a series of Flanking and Driving. For a novice dog, the Shepard may help with instructions that point out the flanking position. In certain situations, the Shepard may help drive the dog.

We designed our system focusing on the dog's Penning behavior. We find that the dog's Autonomy with respect to the Penning task is dependent on the following parameters: dog's capability, dog's sense of achievement, dog's sense of discovery, level of communication by the Shepard, and the amount of help Shepard gives. The dog's capability is in turn a function of (a) tiredness, and (b) perceptual acuity. As the dog moves around it gets tired. The dog's freshness (1- tiredness) is started at 1.0 at the start and is then gradually lowered until a threshold reaches. At that time the dog rests and regains its freshness back to 1.0. A novice dog may not be able to perceive the correct flanking position. If the dog has perfect perception, perceptual acuity is set to 1.0. Otherwise, it is a real value between 0 and 1.0. If the dog is close to the gate, it will experience high achievement and conversely being far from the gate gives the dog a low achievement. This is implemented as a quadratic function that produces a number near 1.0 close to the gate and lower real values farther away from the gate. If the dog receives help from the Shepard in the form of Shepard showing the dog flanking position or moving along to help with Driving, the dog's autonomy is lowered. This is implemented by setting any instance of Help to 0.2 when Shepard is driving the sheep and to 0.1 when it only assists with the correct flank position. Otherwise Help is set to 0. If the dog is near the sheep and the Shepard is far away, the dog experiences a sense of discovery, which positively contributes to its Autonomy. This is implemented by setting Discovery to 0.2 when the Dog alone is handling the sheep and the Shepard is far away. Discovery is set to 0.1 when the dog alone is handling the sheep but the Shepard is nearby and silent. Otherwise, Discovery is set to 0. In this system, our dog's overall autonomy is computed by the following equation. Autonomy is adjusted continuously with every change in the components of this equation.

Autonomy = (Achievement * Capability) – Help + Discovery

For simplicity, the Shepard's perception of dog autonomy is identical to dog's. However, agents in a team may not correctly perceive autonomy of other agents.

Unlike this example, in many multiagent teams, flexible autonomy of individuals is very useful. Imagine a soccer team where each player has an assigned role. Roles limit autonomy with respect to the team goal. The dynamics of the game may make it necessary for a player to play a role that is not assigned. So the player has to increase its autonomy with respect to team goal to go beyond its role.

5 Conclusion

In this paper we presented two case studies of autonomy to illustrate that parameters as well as targets of autonomy are highly domain dependent. What makes sense in one domain may not have meaning in another.

In general, autonomy has sources that are internal to the agent as well as sources that are imposed form being in a community of other agents.

Clearly, we cannot prescribe an optimal level of autonomy without considering nature of the problem.

References

Barber, K.S,, and Martin, C.E., 2000. The Motivation for Dynamic Adaptive Autonomy in Multi-Agent System, JETAI special issue on Autonomy Control Software, Kluwer Pub.

Castelfranchi. C. Modeling Social Action for AI Agents. (Invited paper) 1997. In International Joint Conference of Artificial Intelligence - IJCAI'97, Nagoya, August 23-29, pp.1567-76)

Castelfranchi, C., Falcone, R., 1997. Delegation Conflicts, in M. Boman & W. Van de Velde (eds.) Multi-Agent Rationality - MAAMAW'97, Lecture Notes in Artificial Intelligence, 1237. Springer-Verlag pg.234-254.

Castelfranchi, C., 1995. Guaranties for Autonomy in Cognitive Agent Architecture, in N. Jennings and M. Wooldridge (eds.) Agent Theories, Architectures, and Languages, Heidelberg, Spinger/Verlag.

Gawdiak, Y., Bradshaw, J., Williams, B., Thomas, H., 2000. R2D2 in a Softshell: The Portable Satellite Assistant. Submitted to Agents2000, Barcelona. Spain.

Hexmoor, H., Lafary, M., Trosen, M., 1999a. Adusting Autonomy by Introspection, AAAI Spring Sympoisum: Agents with Adjustible Autonomy, Stanford, CA.

Hexmoor, H., Lafary, M., Trosen, M., 1999b. Towards Empirical Evaluation of Agent Architecture Qualities, ATAL-99, Orlando, Florida.

Hexmoor, H., Heng, T., 1999a. ATC Tower Simulator: TACUND, Proceedings of IASTED on Applied Modeling and Simulation, Cairns, Australia.

Hexmoor, H., Heng, T. 1999b. Air Traffic Control and Alert Agent, Agents2000, Barcelona. Spain.

Sichman, J.S., Conte, R., Castelfranchi, C., Demazeau, Y. 1994. A Social Reasoning Mechanism Based On Dependence Networks. In Proceedings of the European Conference on Artificial Intelligence - ECAI'94, Amsterdam August 8-12, 188-192.).