Considerations for Flexible Autonomy within BDI Intelligent Agent Architectures

Marcus J. Huber

Intelligent Reasoning Systems
4976 Lassen Drive
Oceanside, California, 92056
marcush@home.com

Abstract

This paper discusses issues related to autonomy within BDI-based intelligent agent architectures and applications. One advantage that agent-based applications built upon formal theories such as BDI (Belief-Desire-Intention) theories have over standard application implementation paradigms is their explicit implementation of some of the mentalistic states, such as goals, that are typically associated with autonomy. To capture the intuition that autonomy represents independence from external influences such as other agents, we define autonomy as the level of separation between these external influences and an agent's internal structures, considering not only goals, but also intentions, beliefs, and capabilities. Taking an existing BDI architecture (JAM) as an example, we measure the existing autonomy level for each of these attributes and then introduce and discuss ideas related to architectural modifications and programming practices that support flexible and perhaps dynamic modification of an agent's autonomy.

Introduction

To be one's own master is to be the slave of self.
Natalie Clifford Barney (1876-1972)

Autonomy is an intuitive but subtle characteristic of agency. Our intuition with respect to autonomy is that it is an attribute that needs to be defined as a relationship between an agent's internal structures and external influences. Autonomy is therefore highly dependent upon the agent's internal design (i.e., its architecture). Belief-Desire-Intention (BDI) theoretic (Bratman 1987) architectures such as PRS (Georgeff and Lansky 1987) and IRMA (Bratman, Israel and Pollack 1988) have highly formalized internal structures that facilitate definition of autonomy with respect to these constructs. In this paper, we definite what we mean by both the terms agent and autonomy and then discuss the factors relevant to autonomy within BDI agent architectures in particular. We show how an agent architecture can be concretely characterized with respect to its level of autonomy and that it should be possible to design an agent architecture that facilitates programming agents that exhibit different levels of autonomy in different situations.

The remainder of the paper has the following structure. We first define autonomy and agency and discuss our proposed measure of a BDI agent's autonomy. We describe how our definition of autonomy results in four numeric autonomy-level measures that express how autonomous a BDI agent is from external influences. We then formalize the JAM BDI intelligent agent architecture in order to provide a concrete example of our autonomy measure and discuss aspects of the architecture that impact upon the architecture's autonomy level. Following this, we then characterize JAM according to the level of autonomy for each of its major architectural components in the next section. Finally, we discuss possible modifications to the architecture and possible programming practices that provide flexible, dynamically adjustable agent autonomy.

Agency and Autonomy

Agency

A common requirement of agency by most theoretically-founded intelligent agent architectures is that they must be goal-directed. That is, an agent must exhibit purposeful behavior and have an explicit representation of that which they are working toward (i.e., a *goal*). BDI-theoretic agents introduce the additional constructs of *beliefs* and *intentions* to that of *desires* (the name of goals within BDI frameworks).

Beliefs are an explicit representation of the declarative knowledge of the agent. Beliefs are the agent's view of the current state of the world and, as such, have an indirect but significant impact upon the agent's behavior.

Intentions are a representation of a commitment on the part of the agent to a particular means (i.e., a plan) of performing a task or achieving a goal. An agent's intentions have a very direct impact upon an agents behavior.

A pragmatic aspect not considered within formal definitions of BDI systems is an agent's *capabilities*, a representation of an agent's functionality, which must always be specified within a concrete agent framework. Capabilities models may consist of low-level functionality (e.g., primitive functions) and/or high-level functionality (e.g., plans). Concrete implementations of BDI agents such as UMPRS (Huber et al. 1993, Lee et al. 1994) and JAM (Huber 1998) include such models. Below, when discussing measures of autonomy, we will distinguish autonomy for each of beliefs, desires, intentions, and capabilities.

Autonomy

There is a wide range of definitions of the term *autonomy*. We will start by looking at popular definitions and then look at more research-based technical definitions. According to the American Heritage Dictionary of the English Language,

> *autonomous* (adjective): 1. Not controlled by others or by outside forces; independent: 2. Independent in mind or judgment; self-directed.

Synonyms of autonomy include: uninfluenced, unforced, uncompelled, uncommitted, unaffiliated, isolationist, unsociable, self-sufficient, self-supporting, self-contained, self-motivated, inner-directed, ungoverned, and masterless. An antonym that provides additional insight is,

> *heteronomous* (adjective): 1. Subject to external or foreign laws or domination; not autonomous.

Both definitions refer to some form of external influence, implying that autonomy is a relation and not a factor of an individual in and of itself. Most of the synonyms also imply this feature of autonomy, although some, such as *uncompelled* and *unforced,* may be self-referential (i.e., “slaves to self” from the quote above). For this reason, we believe that autonomy must be defined relative to influences outside of an agent and cannot therefore be defined simply in terms of a single agent’s attributes.

The essential aspect of external forces within autonomy is missed by some researchers, however. Covrigaru and Lindsay (Covrigaru and Lindsay 1991) claim that an entity’s level of autonomy can be determined by looking only at the characteristics of an individual. According to Covrigaru and Lindsay, an entity is “more likely” to be autonomous the more of the following features it exhibits: goal-directed behavior, movement, self-initiated activity, adaptability and flexibility, robustness, and self-sufficiency. In their definition, an entity is autonomous if it has multiple top-level goals, some of which are homeostatic (i.e., need to be maintained over time rather than simply achieved and dropped), and it has freedom to choose between the goals. Covrigaru and Lindsay’s definition is somewhat appealing as it does capture a number features that seem desirable for an autonomous agent to exhibit. Two of their key features, self-initiation and self-sufficiency, do suggest that they share some intuition regarding autonomy as isolation from other agents. However, they explicitly state that it does not matter whether goals came from internal processing or received from external sources which is completely counter to our intuition.

Luck and d’Inverno define an autonomous agent to be anything that has *motivations* (Luck and d’Inverno 1995), where a motivation is defined to be any desire or preference that can lead to the generation and adoption of goals. In their formalization therefore, they consider goals to be a derivative of motivations, which are themselves non-derivational. In essence, they also view goal-directed behavior as being an essential feature of agent autonomy. Luck and d’Inverno’s definition also does not concern itself with where the agent’s motivations originate; they imply internally, but this is not required.

In both the definition by Covrigaru and Lindsay and the definition by Luck and d’Inverno, an agent’s motivations or goals could be completely dominated by an external force but would still be considered autonomous! We find this completely antithetical to the idea of autonomy.

Castelfranchi (Castelfranchi 1995) defines autonomy as the amount of separation between external influences and an agent’s goals. Castelfranchi recommends a "double filter" upon goal autonomy. First, he requires that an agent perform some form of reasoning about externally derived information before internalizing it. Second, he requires external influences must be filtered through beliefs before an agent’s goals can be modified. His definition of belief and goal autonomy is quite compatible with our intuition but we believe it is too narrow for application to BDI-based agents.

Huhns and Singh define a set of autonomy measures that are generally compatible with our own but capture autonomy at a different level of detail. In their recent summary of the field of intelligent agents (Huhns and Singh 1998), they distinguish between Absolute autonomy, Social autonomy, Interface autonomy, Execution autonomy, and Design autonomy. Absolute autonomy describes an extreme agent that completely ignores other agents and does whatever it wants to do. As we will see, this would be equivalent to an autonomy level of infinity in all four of the autonomy measures introduced in this paper. Absolute autonomy is an extreme of Huhns' and Singh's Social autonomy, which captures the notion of level of isolation from other agents and correlates well to what would be a summary measure of the four autonomy measures we introduce later. Design autonomy relates to how much freedom an agent’s designer has in terms of designing the agent and therefore is not relevant here. Interface autonomy is also a design-related feature and is also not relevant to this discussion. Their Execution autonomy measure captures the notion of how much freedom an agent has in making decisions while executing its plans. We do not directly address this autonomy measure but instead let it fall out as a factor of our different set of autonomy measures.

Because we are interested in defining autonomy with respect to BDI-based agent architectures, we refine Huhns and Singh’s Social autonomy measure by extending and revising Castelfranchi’s idea of levels of isolation from other agents. To match well with BDI architectures, we define autonomy measures with respect to each of an agent's beliefs, goals, intentions, and capabilities. In this work, in order to provide an explicit measure of autonomy with respect to an agent's beliefs, we remove Castelfranchi's requirement that the agent must filter everything through its beliefs.

Defining autonomy with respect to each of these constructs seems reasonable to us. For instance, it seems useful to be able to distinguish between agents whose beliefs are easily modified by external influences (e.g., an agent that "believes everything that it reads" (via perception) or "believes everything that it hears" (via communication with other agents)) from those that are less "gullible" because they perform more reasoning to verify that the information is accurate. Similarly, it seems useful to distinguish between agents that blindly accept plans from other agents from those that inspect the plans to make sure they would not make the agent perform restricted actions (e.g., reformat the computer's hard drive).

Within this paper, we define belief, goal, intention, and capability autonomy to be the number of representations and reasoning layers intervening between external influences and the agent's internal representation of beliefs, goals, intentions, and plan library, respectively. We interpret each layer of reasoning as representing an opportunity for the agent to examine the incoming information and to accept or reject internalizing the information. We interpret each representation layer as a distinctly different semantic form of information that originates from external influences and is transformed as it progresses to the agent's internal structures, becoming less and less "contaminated" by outside influences at each transformation.

With respect to representations, we count intervening layers even if they have the same syntactic form as prior or subsequent layers. For example, an agent that uses a vision system may have a long sequence of representations from the original input image to some final "belief" and all of which are of the "image" representation. Even though the representation form stays the same, the image contents may change from the input image to one with enhanced edges to one with edges joined to form boundaries to one with regions and so on. Hence, each level is semantically distinct and therefore countable in our autonomy level calculation.

Our definition of goal autonomy is most similar to the definition of goal autonomy of Castlefranchi (Castlefranchi 1995) and the *social autonomy* of Huhns and Singh (Huhns and Singh 1998). We know of no prior research that considers intention autonomy explicitly. Our definition of capability autonomy is most similar to the definition of *execution autonomy* of Castlefranchi and also of Huhns and Singh.

In order for an agent to alter its level of autonomy, the agent would need to add or remove levels of reasoning and/or representation transformations between its internals and external influences. Such an idea is depicted in Figure 1, where more or less autonomy could be realized by changing the separation between the external forces on the left and the agent's internals on the right. We discuss ideas on how to do this in more detail later in the section titled "Toward Flexible Autonomy".

First however, we want to provide a concrete example of our definition of autonomy, so in the next section we formalize an instance of an implemented BDI agent architecture called JAM. This provides the basis for the subsequent section where we calculate the autonomy levels in the current JAM implementation.

Formalizing JAM

In this section, we formalize the JAM BDI-based agent architecture (Huber 1999). The JAM architecture is based strongly upon PRS (Georgeff and Lansky 1986, Georgeff

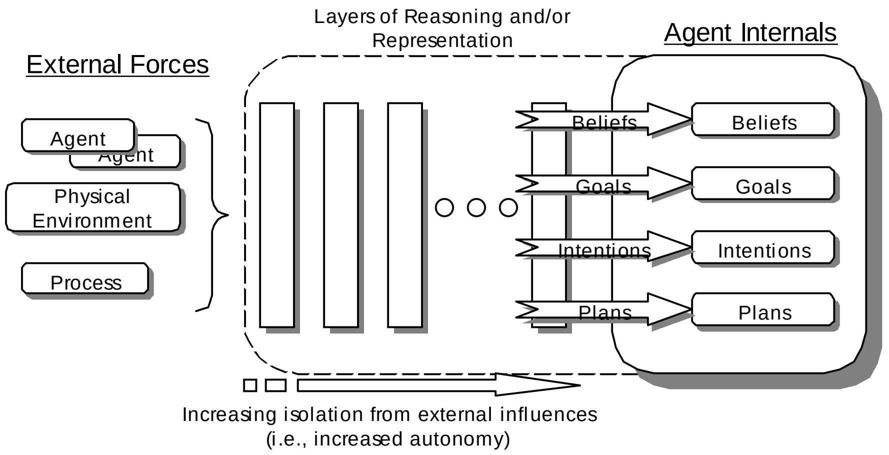

Figure 1. An agent's level of autonomy may be characterized by how much isolation, in terms of representational and reasoning layers, the internal constructs of an agent have from external influences.

and Lansky 1987), but has also been influenced by the Structured Circuit Semantics (SCS) research of Lee and Durfee (Lee 1996) and the Act plan formalism of Wilkins and Myers (Wilkins and Myers 1995). In our formalization, we start with low-level JAM representations and move steadily through intermediate representations and functions toward the agent's final high-level representations of beliefs, intentions and so on, and the functions that operate on these representations.

In the following definitions, "(" and ")" group terms, "[" and "]" enclose optional terms, "*" indicates zero or more repetitions of a term, "+" indicates one or more repetitions of a term, "·" indicates concatenation of terms, "∨" indicates disjunction, "::=" indicates the definition of a representation, ":" indicates the definition of a function, and "→" indicates a mapping from a function's domain to a function's range. We present our definition first and follow the definition with a description.

const	$::= \{\chi \mid \chi = number \vee \chi = string \vee \chi = Java\ object\}$
var	$::= \{\ v \mid value(v) = const\}$
varList	$::= (var)^+$

Constants may be numbers or strings, while variables are dynamic placeholders for constants. The function *value-of(var)* returns the constant currently held by variable *var*. A variable list is simply a list of one or more variables.

arg	$::= const \vee var$
argList	$::= (arg)^*$

Each argument of a function may be either a constant or a variable and a function's argument list may consist of zero or more arguments.

label	$::= string$
proposition	$::= label \cdot argList$
belief β	$::= proposition$
worldModel Ψ	$::= (\beta)^*$
$\Psi_{initial}$	$::= file_{\Psi}$

Each individual belief within a JAM agent is a simple proposition formed by a string label and an argument list. A JAM agent's World Model is formed by zero or more beliefs and is initially specified in a text file that is parsed by the agent at invocation.

condition φ	$::= relation \cdot argList,\ value(eval(\varphi)) = True \vee False$

A condition is a function that when evaluated returns a Boolean value and is represented by a relational function and an argument list.

binding	$::= \{\ (v, \delta) \mid v \in varList, \delta \in const, value(v) = \delta\ \}$

A variable binding provides a lookup table that maps variables to the values held by the variables. Variable bindings are scoped locally to goals and plans.

goal	$\gamma ::= (\text{ACHIEVE} \vee \text{PERFORM} \vee \text{MAINTAIN}) \cdot proposition \cdot argList \cdot [:utility\ num]$
goalList	$\Gamma ::= \{\ \gamma \mid \gamma \in IntentionStructure,\ intention\text{-}of(\gamma) = null\ \}$
$\Gamma_{initial}$	$::= file_{\Gamma}$

Goals are represented by a goal type, a proposition, an argument list, and an optional utility value. A JAM agent does not actually have an explicit goal list distinct from its Intention Structure. We provide the notation to distinguish the set of goals for which an agent has not yet instantiated plans from the entire set of all goals on the Intention Structure, some of which do have intentions. The JAM agent's Goal List is initially specified in a text file that is parsed by the agent at invocation.

precondition	$::= (\varphi)^*$
runtimecondition	$::= (\varphi)^*$
body	$::= procedure$
effects	$::= procedure$
failure	$::= procedure$
attributes	$::= (attribute, value)^*$
plan ρ	$::= (\ \gamma \vee \beta\ ) \cdot precondition \cdot runtimecondition \cdot body \cdot effects \cdot failure \cdot attributes \cdot argList$
planLibrary P	$::= (\rho)^*$
$P_{initial}$	$::= file_P$

Plans are a complex composition of components. A plan can be either goal or belief invoked. Preconditions and runtime conditions filter where the plan is applicable. The procedural components of a plan are arbitrarily complex "programs" comprised of a wide range of constructs (e.g., iteration and conditional branching) and primitive actions. The body component, which is the primary procedural component, may include subgoaling actions. The effects component, executed when the body procedure completes successfully, and the failure component, executed when the plan fails during execution, may not contain subgoaling actions. The JAM agent's plan library is initially specified in a text file that is parsed by the agent at invocation.

intention ι	$::= \gamma \cdot \rho \cdot \beta \cdot binding$
intentionStack ς	$::= \gamma \vee (\gamma \cdot \iota)^+,\ \gamma = goal\text{-}of(\iota)$
intentionStructure I	$::= (\varsigma)^+$
applicablePlanList α	$::= (intention)^*$

Intentions are plans that are instantiated with their variables bound to values. An intention stack is either a "barren" goal that has not yet had an intention selected for it, or it is a sequence of goals paired with intentions. In the latter case, the first goal in the sequence is a top-level goal, the second goal is a subgoal for that top-level goal, and so on. A JAM agent's Intention Structure then, is a collection of intention stacks, which represents all of the competing tasks the agent is considering at any point in time.

generateAPL:	$\Psi \times \Gamma \times P \rightarrow \alpha$
selectAPLElement:	$\alpha \rightarrow \iota$, ι = *highest-utility*(α)
intend:	$I \times \iota \rightarrow I$
sortIntentions:	$\Psi \times I \rightarrow I$

The JAM interpreter reasons over its unintended goals, available plans, and set of beliefs to create a list of possible intentions, of which it selects the highest-utility intention and adds it to the agent's Intention Structure. Every cycle through the interpreter, the JAM interpreter sorts the intentions according to their utilities.

communication	κ
percepts	ζ

Communication- and perception-based information is unconstrained by the JAM agent architecture and we provide symbology for these terms for reference purposes later. Relevant information incorporated via communication and perception are at the representational level of beliefs, goals, plans, and possibly individual intentions.

executePlan	$P: \kappa \times \zeta \times \Psi \times \Gamma \rightarrow \Psi \times \Gamma \times P$
executeMetalevelPlan	$M: \kappa \times \zeta \times \Psi \times \Gamma \times \alpha \rightarrow \Psi \times \Gamma \times I \times P$

Plans permit an agent to reason about communicated information, perceived information, and its own beliefs and goals and then arbitrarily modify its beliefs and goals. Plans actions can also possibly result in the parsing and incorporation of plans into the agent's Plan Library. Metalevel plans are distinguished from "ground-level" plans by their access and manipulation of an APL and the selection and intention of one of the intentions within the APL to the agent's Intention Structure.

executeObserver	$\Theta: \kappa \times \zeta \times \Psi \times \alpha \rightarrow \Psi \times \Gamma \times I \times P$

The JAM architecture executes an optional procedure called the Observer every cycle through the interpreter. The Observer procedure can be an ingress point for perceptual and communicative information and can access the agent's World Model and APLs on the agent's World Model. The Observer can perform operations on an agent's World Model and create top-level goals. Like metalevel plans, the Observer has access to APLs placed on the World Model and can intend elements onto the Intention Structure, although this is not an intended use of the Observer and is not advised in real applications.

We informally depict the high-level portions of the formalisms above in Figure 2, below. Figure 2 should make it easier to visualize and count the JAM architecture's representation and functional levels.

JAM Autonomy

In this section, we calculate the level of autonomy λ of the JAM agent architecture with respect to intentions (λ_I), capabilities (λ_P), beliefs (λ_Ψ), and goals (λ_Γ), in that order.

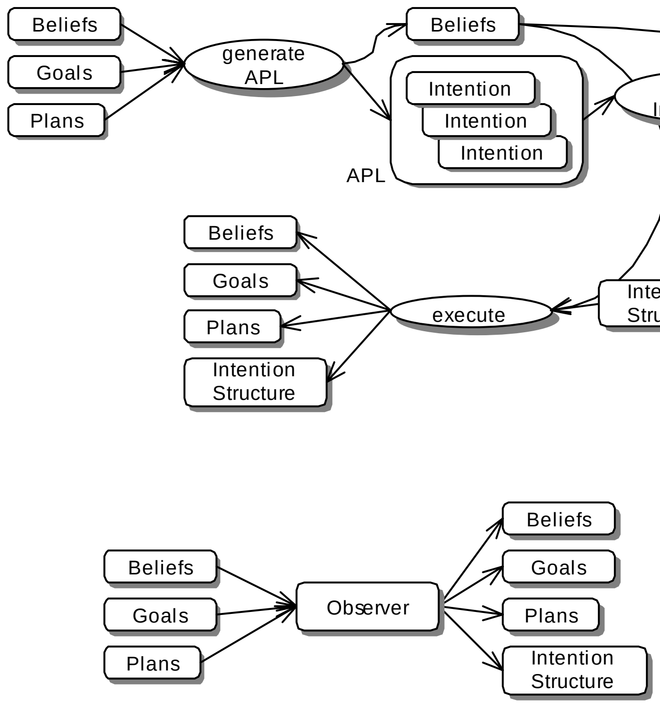

Figure 2. Processes and representations in the JAM architecture. Ovals represent functions and round-cornered rectangles represent representations.

Later, we discuss possible modifications to the agent architecture and possible application-development paradigms that provide improved flexibility of control over an agent's autonomy level.

To calculate each λ, we count the procedural and representational transformations between external influences and the agent's internals for all possible sources for the attribute in question and take the minimum of these values. Using the minimum function results in a conservative measure, as it provides a value that represents the *shortest influence path* into an agent's internals. With this formulation, a value of zero would indicate that external influences have a direct path into the agent's internal structures and have complete control over the agent. A value of ∞ (equivalent to Huhns and Singh's Absolute autonomy) would indicate that external influences have no possible way of influencing the agent's internal structures and would probably indicate a completely sociopathic agent. It is easy to see that our formulation results in autonomy levels that take on positive, odd values. An internal representation (world model, intention structure, etc.) is the final state derived from a final procedure and there can be zero or more [procedure $\rightarrow$ representation] pairs preceding this final representation.

Intention Autonomy

The Intention Structure can be modified, 1) during default interpreter reasoning, 2) during execution of metalevel plans, or 3) during Observer procedure execution. From the preceding section, we see that default interpreter reasoning results in at least five representational and functional steps [$\Psi \times \Gamma \times P \rightarrow generateAPL() \rightarrow \alpha \rightarrow selectIntention() \rightarrow \iota \rightarrow intend() \rightarrow I$] between initial inputs and the point where the Intention Structure is modified ($\Rightarrow \lambda_I = 5$). Methods 2 and 3 rely upon an APL as input and results in the following representational and functional steps [$\Psi \times \Gamma \times P \rightarrow generateAPL() \rightarrow \alpha \rightarrow executeX \rightarrow I$] ($\Rightarrow \lambda_I = 3$, where *executeX* stands for *executeMetalevelPlan* or *executeObserver*). JAM therefore currently has an inherent architectural autonomy level of $\lambda_I = min(5, 3, 3) = 3$ with respect to its Intention Structure. As mentioned earlier, metalevel plans are the intended place for manipulation of the Intention Structure. However, since the Observer can potentially be used for this purpose too, it represents the shortest influence path.

Capabilities Autonomy

A JAM agent's plans come from either 1) a file specified at agent invocation (we consider these to be internally generated and therefore not subject to evaluation), 2) during execution of plans, 3) during execution of metalevel plans, or 4) during Observer procedure execution. According to our representation in the preceding section, plans, metalevel plans, and the Observer each result in one level of functional separation (see the definition of P, M, and Θ, respectively) between external influences and the agent's plan library. JAM therefore currently has an inherent architectural autonomy level of $\lambda_P = min(3, 3, 1) = 1$ with respect to its Plan Library. It might be more accurate to consider the autonomy level of methods 2 and 3 based upon the full reasoning and representation path for plan execution ($\Psi \times \Gamma \times P \rightarrow generateAPL() \rightarrow \alpha \rightarrow selectIntention() \rightarrow \iota \rightarrow intend() \rightarrow I \rightarrow sortIntentions() \rightarrow I \rightarrow execute() \rightarrow P$) which yields an λ_P of at least 9 with respect to these two options. However, even if this less conservative interpretation is taken, the result is still $\lambda_P = min(9, 9, 1) = 1$ due to the Observer constituting the shortest influence path.

Note that even an autonomy level of one means that an agent will not blindly incorporate plans from external sources (which is Castelfranchi's idea for belief autonomy), but that it has at least *some* insulation between the external influences and the agent's internals. In the case of plans, the low autonomy value of one is a little misleading. For, even if a plan from an antagonistic source *is* added to an agent's plan library, there is no way that that particular plan can be guaranteed to be executed. Within JAM, the plan has to be applicable to a specific pre-existing goal or conclusion, which may never arise during normal execution. However, once such a "virus" plan *is* executed, it essentially has complete control over the agent and could therefore force the agent to do anything, including becoming a "slave" to another agent if the plan was designed to accomplish this.

Belief Autonomy

A JAM agent's World Model elements arise from either 1) a file specified at agent invocation (we consider these to be internally generated), 2) during execution of a plan, 3) during execution of a metalevel plan, or 4) during Observer procedure execution. JAM's autonomy level analysis is identical to that for its plan autonomy and yields an inherent architectural autonomy level of $\lambda_\Psi = min(3, 3, 1) = 1$ when using the most conservative values (the less conservative interpretation yields $\lambda_\Psi = min(9, 9, 1) = 1$ as in the case of λ_P).

Goal Autonomy

The JAM architecture constrains goal autonomy in a manner very similar to its belief and plan autonomy. In the previous section, we specified that the goals on the JAM agent's Goal List arise from either 1) a file specified at agent invocation (we consider these to be internally generated), 2) during execution of plans, 3) during execution of a metalevel plan, or 4) during Observer procedure execution. JAM therefore currently has an inherent architectural autonomy level of $\lambda_\Gamma = min(3, 3, 1) = 1$ when using the most conservative values (the less conservative interpretation yields $\lambda_\Gamma = min(9, 9, 1) = 1$ as in the case of λ_P).

Toward Flexible Autonomy

In this section, we discuss how we can modify the JAM agent architecture to best support flexible autonomy and how agent programmers can take advantage of this flexibility. The goal is to provide flexible autonomy independently with respect to beliefs, goals, intentions, and capabilities. The JAM agent architecture currently has an inherent minimum autonomy level of 1 for each of λ_P, λ_Ψ and λ_Γ and a minimum autonomy level of 3 for λ_I. Maximum flexibility will be realized if the JAM architecture can be modified so that all autonomy level measures can be varied between 1 and ∞. We will discuss architectural modifications aimed at increasing λ_P, λ_Ψ and λ_Γ and both increasing and decreasing (from 3 to 1) λ_I.

Flexible Architectural Autonomy

Architectural modifications that result in flexibly increasable λ_P, λ_Ψ and λ_Γ implies providing the agent some mechanism by which to introduce and remove these layers dynamically, perhaps conditionally upon the current situation or environment. It also means either adding reasoning and representation layers between these internal structures and external influences or removing or lengthening relevant shortest influence paths.

One obvious architectural modification to provide increased autonomy with respect to beliefs, plans, and goals is to remove or disable the Observer and therefore removing the shortest influence path for these structures. This results in an immediate increase of λ_P, λ_Ψ and λ_Γ to 3 (or 9, depending upon whether a conservative or liberal interpretation is made). Adding a primitive action to JAM to perform this functionality would be a very simple task but would have to be used with great care. The Observer procedure typically contains crucial processing steps that the agent cannot effectively run without for any extended length of time.

Lengthening an influence path means introducing domain-independent and domain-independent reasoning and representation layers. For example, to increase belief autonomy, the World Model might perform additional reasoning to ensure global belief consistency, or might reason about the credibility (we would have to add this attribute as an architectural feature) of the source, before adding a world model entry. As another example, we can increase goal autonomy by adding a reasoning layer in front of the agent's goal list to reason about goal coherency. To increase plan autonomy, we might introduce plan inspection to make sure that a plan contains no "dangerous" actions before adding the plan to the Plan Library. At some point, however, it is likely that we are likely to run out of value-added domain-independent layers of reasoning and adding further such layers simply to increase the autonomy measures becomes gratuitous without any real benefit.

Intention autonomy is different from our other measures it that it is not currently Observer-dominated, as the metalevel reasoning path provides the same level of separation as the Observer path. The most direct method of reducing λ_I to 1 is to implement a means of directly incorporating externally-generated intentions through the Observer (resulting in one level of procedural separation between an external source and the agent's Intention Structure). Generating an intention for another agent to directly incorporate requires a great deal of very accurate, and very specific information about the other agent. At the minimum, an external agent would need a specification of a top-level goal, a viable plan template for that goal, and a set of variable bindings and world model references that satisfy the plan's context. So, pragmatically, even if an Observer-based means of directly incorporating intentions were implemented, it would still be very difficult for an external force to take advantage of.

In order to support flexible autonomy, the JAM agent architecture would need to support dynamic modification of the number of representation and procedural layers shown in Figure 1. It is unclear to us at this point where it would make sense for the architecture to change autonomy levels. So beyond making architectural changes to support flexible autonomy, it seems most reasonable for an agent programmer to encode domain-specific criterion that takes advantage of this capability. We discuss programmer-determined agent autonomy in the following subsection.

Flexible Programmer-determined Autonomy

An agent programmer can realize flexible agent autonomy in two ways. The first is by taking advantage of the architectural support and primitive functions for adding and disabling architectural layers discussed above. The agent programmer can write plans that reason about the current situation and intelligently invoke the primitive functions appropriately. The second is by carefully designing, structuring, and implementing the agent's plans in its Plan Library, particularly with respect to metalevel plans, the agent designer can interpose layers of metalevel reasoning about beliefs, goals, plans, and intentions before concrete-level plans act upon these structures. The first means is relevant, but a discussion of all of the possible domain-specific conditions and uses is outside the scope of this paper. We discuss the second means in more detail below.

In JAM, plans can be triggered either by goals or by beliefs. Metalevel plans become applicable whenever a plan is being considered and the metalevel plan can select between alternative plans or reject applicable plans as it sees fit. If the plan space of a JAM agent is partitioned according to metalevel level (i.e., concrete plans, metalevel plans, meta-metalevel plans, and so on), the separation between external influences and the agent's internal structures can be conceptually and practically increased one level for each level of metalevel reasoning. In this case, each metalevel plan can reason about the appropriateness of how best to pursue goals, respond to changes in its world model, and remove potentially dangerous goals, beliefs, etc. Arbitrary levels of autonomy can then be realized through engineering and appropriate implementation of plan and metalevel plan contexts and preconditions.

Summary

This paper introduces a concrete definition of a measure of autonomy within BDI systems. We define autonomy as the number of distinct representation and procedural layers between external influences and the agent's internal structure independently with respect to beliefs, goals, intentions, and capabilities. We then characterized an example of an implemented BDI-theoretic architecture's autonomy level with respect to each of these internal structures. Finally, we discussed how we might modify the JAM agent architecture to support a wider range of autonomy levels and listed some architectural modifications and programmer practices that facilitate flexible, context-sensitive agent autonomy.

Acknowledgements

We would like to thank Dan Damouth for several interesting discussions related to autonomy and also for helpful comments on earlier drafts of this paper. We would also like to thank SSC San Diego for funding that supported this work.

References

Bratman, M. 1987. *Intentions, Plans, and Practical Reason.* Cambridge, Mass.: Harvard University Press.

Bratman, M., Israel, D., and Pollack, M. 1988. Plans and Resource-bounded Practical Reasoning. *Computational Intelligence* 4:349-355.

Castelfranchi, C. Guarantees for Autonomy in Cognitive Agent Architecture. 1995. In *Intelligent Agents – Theories, Architectures, and Languages*, 56-70, Michael Wooldridge and Nicholas Jennings editors, Springer-Verlag.

Georgeff M., and Lansky, A. L. 1987. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 677-682, Seattle, Washington.

Georgeff, M. P., and Lansky, A. L. 1986. Procedural Knowledge. *IEEE Special Issue on Knowledge Representation*, 74(10):1383-1398.

Huhns, M., and Singh, M. 1998. Agents and Multiagent Systems: Themes, Approaches, and Challenges. *Readings in Agents*, 1-23. Michael Huhns and Munindar Singh eds., San Francisco, California: Morgan Kaufmann.

Huber, M. J., Lee, J., Kenny, P., and Durfee, E. H. 1993. UM-PRS Programmer and User Guide. The University of Michigan, Ann Arbor, MI 48109. [Also available at http://members.home.net/marcush/IRS]

Huber, M. J. 1999. JAM: A BDI-theoretic Mobile Agent Architecture. *Proceedings of the Third International Conference on Autonomous Agents*. Forthcoming.

Konolige, K., and Pollack, M. 1993. A Representationalist Theory of Intention. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.* Chambery, France.

Lee, J. *Structured Circuit Semantics.* 1996. Ph.D.diss., Department of Computer Science, The University of Michigan.

Lee, J., Huber, M. J., Durfee, E., H., and Kenny, P. G. 1994. UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications. In *Conference on Intelligent Robotics in Field, Factory, Service, and Space,* 842-849, Houston, Texas.

Michael Luck and Mark d'Inverno. 1995. A Formal Framework for Agency and Autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems*, 254-268. MIT Press.

David E. Wilkins and Karen L. Myers. 1995. A Common Knowledge Representation for Plan Generation and Reactive Execution. In *Journal of Logic and Computation*, 5(6):731-761.