"It's Not Just Goals All the Way Down" – "It's Activities All the Way Down"

Maarten Sierhuis

Research Institute for Advanced Computer Science
NASA Ames Research Center
M/S 269-1
Moffett Field,
CA 94035-1000, USA
Maarten.Sierhuis-1@nasa.gov

Abstract. The rational agent community uses Michael Bratman's *planning theory of intention* as its theoretical foundation for the development of its agent-oriented BDI languages. We present an alternative framework based on situated action and activity theory, combining both BDI and activity-based modeling, to provide a more general agent framework. We describe an activity-based, as opposed to a goal-based, BDI language and agent architecture, and provide an example that shows the flexibility of this language compared to a goal-based language.

1 Introduction

This chapter is written based on a talk with the same title given as an invited talk at the seventh annual international "Engineering Societies in the Agents World" (ESAW 2006) conference. The title is meant as a tongue-in-cheek provocation towards the BDI agent community being proponents of agents as goal-driven planners. A previous presentation about the Brahms multiagent simulation and development environment, at the 2006 Dagstuhl seminar on Foundations and Practice of Programming Multi-Agent Systems, had created debates with agent language researchers about the primary use of goal-driven behavior in agent languages [1]. In this talk, we intended to put forth an alternative to agents being solely goal-driven planners. We not only put forward an alternative view on human behavior (a view that relies on activities instead of goal-driven action), but we also presented our Brahms multiagent language and simulation environment that is BDI-like without the notion of goals as the driving force of agent action [2][3]. In this chapter, we try to more thoroughly explain this activity-based BDI approach and present not only the Brahms language and the workings of the architecture, but present the theoretical basis for this alternative view of human behavior. The theory of activity-based modeling is rooted in *situated action*, *cognition in practice, situated cognition* and *activity theory*.

It has to be noted that this paper cannot do justice to the large number of research articles that have been written on these concepts. We merely can scratch the surface, and we do not claim that we explain these concepts completely. We quote heavily from writings of other, more prominent, researchers to explain these concepts, and we

G. O'Hare et al. (Eds.): ESAW 2006, LNAI 4457, pp. 1–24, 2007.

advice the reader to study them in order to get a deeper understanding of the theories and research behind them. What we hope this paper contributes to is a theory, practice and tool of how to analyze, design and implement large agent systems. Our hope is also that we succeed in providing a well enough argument for activity-based rational agents as a more general alternative to goal-driven ones.

The rational agent community, over the last ten or so years, has settled on using Michael Bratman's *planning theory of intention* as its theoretical foundation for the development of its agent-oriented BDI languages [4][5]. Fundamentally, we believe that it is very good for a programming paradigm and its subsequent languages to be based on a theory. In general, programming is about modeling the world as we know it, and a specific program embeds a particular theory—the programmer's theory of the world being modeled. When a programming language itself is based on a solid theory it can help the programmer. However, this is both a good and a bad thing. The good thing is that we can have a tool that is sound—based on a theory—and thus the programs that are being developed with it will, by definition, adhere to the theory. This helps the programmer not to have to invent a new theory before implementing a model based on it. The theory is already there. For example, all computer programmers use Turing's theory of universal computing machines as their basis for developing programs as a sequence of instructions.[1] The bad thing is that believers of the theory tend to apply the theory in all sorts of ways that might, or more importantly, might not be within the theory's relevance. A second problem is that the theory might be flawed, or at least, can have alternative theories that put in question a fundamental part of the theory.

In this chapter we discuss some of the fundamental underpinnings of the theory on which the BDI programming languages of today are based. We point out that a fundamental assumption of Bratman's theory has at least one alternative theory that is based on the *inverse* assumption. This is interesting in the sense that both theories cannot be correct. After we discuss some of Bratman's theory and its fundamental assumption, we will move on to describe the alternative theory and its assumption. Then, we will merge the two theories into a theory that combines both and thus will provide a more general theory on which future BDI languages could be based. We then move on to describe an activity-based BDI language and agent architecture based on this new more general theory, and provide an example that shows the flexibility of this language compared to a goal-based BDI language.

2 Theoretical Stances

2.1 Bratman's Planning Theory

Bratman's philosophical treatise on human behavior is based on commonsense or folk psychology—a not falsifiable theory based on propositional attitude ascriptions, such as "belief", "desire", "pain", "love," "fear". The fact that the theory on which Bratman's treatise is based is not falsifiable makes it not a theory, but a theory about

[1] Alan Turing (1936), "On Computable Numbers, With an Application to the Entscheidungs-problem", Proceedings of the London Mathematical Society, Series 2, Volume 42 (1936). Eprint. Reprinted in The Undecidable pp.115-154.

the existence of a theory, also known as a theory-theory. Herein lays the division in camps of believers: Those who believe in the theory and those who do not, and then those who believe in some of it. It is difficult to decipher in which camp the rational agent community lays. To me there are only two possibilities. First, it is completely possible that people in the rational agent community believe in Bratman's *Planning Theory* [4]. Falling in this camp means that one agrees with the assumption that people *are* planning agents. This has a major implication, to which we will return in the next section. The other possibility is that people in this camp do not necessary believe that Bratman's theory says something in particular about people, but it says something about any kind of animal or any non-living object. People in this camp care less about the fact that Bratman's theory only talks about people and not about systems in general, but they nevertheless believe that using planning to deliberatively decide on future actions is a very attractive presupposition, and even more, it is a very useful concept for computer programming. For people in this camp every behavior, whether human or not, can be reduced to *a problem that needs to be solved*, because in that case Bratman's Planning Theory can be used to predict and execute actions to solve it. Thus, in short, Bratman's theory can easily be used to solve any behavioral "problem" with an algorithm developed by researchers of artificial intelligence and cognitive science.

Let us briefly turn to Bratman's theory, and let us start by stating that we do not have the space or the time to fully describe Bratman's theory here. We only focus on those parts that are relevant for our ultimate points. Since this paper is part of a book about engineering software agents, we assume for simplicity sake that the reader is familiar with the concepts *belief*, *desire* and *intention*, and we therefore do not explain these concepts. What does Bratman's theory say?

> **People are planning agents => they settle in advance on complex plans for the future and these plans guide their conduct.**

This is Bratman's fundamental assumption. This is neither fully explained, nor is it proven by his theory. It is simply a supposition. The only evidence for this supposition being true is given by Bratman in the form of a reference to Simon's notion that people are *bounded rational agents* [6]. Most artificial intelligence researchers, economists and cognitive scientists agree with Simon's claim. Never mind that Simon's notion came about as a reaction on *economic models* always assuming that people are hyperrational[2]. Simon postulated that people are boundedly rational and the only way of behaving is by *satisficing*[3].

> **People, as planning agents, have two capacities: 1) the capacity to act purposeful, 2) the capacity to form and execute plans.**

The following quote exemplifies the extent to which Bratman justifies his claim that people must be planning agents; "So we need ways for deliberation and rational

[2] In this context, hyperrational means that people are always rational, to a fault, and would never do anything to violate their preferences.

[3] "Satisficing is a behaviour which attempts to achieve at least some minimum level of a particular variable, but which does not necessarily maximize its value." from Wikipedia, 12/24/2006.

reflection to influence action beyond the present; and we need sources of support for coordination, both intrapersonal and interpersonal. Our capacities as planners help us meet these needs" [4, p. 3]. In here lies Bratman's basic "proof" that people are planning agents. His reasoning goes something like this; Planning is an algorithmic way to deal with deciding on what to do in the future, i.e. future action. Planning can be seen as a sort of coordination. People need to decide on future action; otherwise they could never do anything. Deliberation takes time and it is not in and of it-self useful for taking action. We need a way to influence what we do in the future. People need to coordinate their actions, as well as what they do together. Therfore, people need a planning algorithm to do this. Quod erat demonstrandum.

Goal-Based Modeling

Those who believe in Bratman's Planning Theory almost always also believe in people being general problem-solvers, and that deciding what to do next is a problem to be solved for which people use a goal-based or goal-driven reasoning method. The reason that this way of thinking is almost always synonymous with believing in the BDI agent architecture is that it fits very well with Bratman's Planning Theory. To be goal-based means that one stores information about the world and about situations that are desirable, which we refer to as the goals to be reached. Our model of current situation or current state and of goals or future state, allows us then to choose among the multiple future states we can think of and select which one should be next. This next state is then called our (sub)goal. This maps, in principle, very well onto the concepts of beliefs, desires and intentions. In a more general way, not necessarily only related to how people operate, we might change the names of these concepts into stored current state, desired future state or goal, and intent to act. It is the concept of *desire* that makes the BDI architecture inherently a goal-driven architecture and BDI agents problem-solving agents.

This model of agent behavior is compelling to people adhering to Bratman's Planning Theory, because it also fits very well with some of the most prominent theories of cognition, namely Newell's Unified Theory of Cognition and Anderson's ACT-R model of cognition [7][8]. However, there are theories and implemented architectures out there that are alternatives to goal-driven planning. One of the most prominent alternatives is the behavior-based robot architecture that shows that robots can achieve intelligent behavior that is not based on the existence of an explicit plan and of goal-driven behavior. Indeed, Brooks argues convincingly, as we are trying to do in this chapter, that "planning is just a way of avoiding figuring out what to do next." [9, Chapter 6, p. 103]. Brooks' basic argument goes something like this: The idea of planning and plan execution is based on an *intuition*. There is no evidence that it *has to be* the only way to develop intelligent behavior. Putting sensors and effectors together using a program can also create intelligent behavior.

Thus a process-way of acting is just as possible as a state way of reasoning. In the Artificial Intelligence and Robotics community Brooks is the most well known researcher that argues against Bratman's Planning Theory. However, there have also been social- and cognitive scientists that have argued against the Planning Theory, not from an engineering or computational angle, but from a social and behavioral angle. This is the subject of the next section, and together with Brooks' argument will form the basis for our thesis that an activity-based approach is a more general approach,

incorporating both a goal-based and a behavior-based approach for BDI agent architectures.

2.2 The Alternative View

Let us turn to an alternative view of human behavior, namely a view of human purposeful action based on ethnomethodology[4]. Suchman, in her book "Plans and Situated Actions," takes a strong stance against the traditional view of planned purposeful action. It is this alternative view that in our opinion best expresses the problems with Bratman's Planning Theory as a theory about human purposeful action [8].

Situated Action

Suchman's view takes human purposeful action as its starting point for understanding human behavior, but it is not a theory of the brain's functioning. Suchman's view is not presented as a theory of human behavior, as it does not predict human behavior. However, Suchman investigates an alternative view to the view deeply rooted in Western cognitive science, that human action is only and essentially determined by plans. Suchman describes human activity from a socially-, culturally- and environmentally situated point of view, rather than from a cognitive point of view. This different angle on human activity opens up an entirely new take on purposeful action, namely that learning how to act is dependent on the culture in which one grows up, and thus there is variation based on the situation. Whether our actions are ad hoc or planned depends on the nature of the activity and the situation in which it occurs.

Although we might contrast goal-directed activities from creative or expressive activities, or contrast novice with expert behavior, Suchman argues convincingly that every purposeful action is a *situated action*, however planned or unanticipated. Situated action is defined as "actions taken in the context of particular, concrete circumstances." [p. viii] Suchman posits an important consequence of this definition of purposeful action that is best retold with a quote:

> "… our actions, while systematic, are never planned in the strong sense that cognitive science would have it. Rather, plans are best viewed as weak resources for what is primarily *ad hoc* activity. It is only when we are pressed to account for the rationality of our actions, given the biases of European culture, that we invoke guidance of a plan." [p. ix]

Suchman goes on to say that plans stated in advance of our action are *resources* that are necessarily vague, and they filter out the precise and necessary detail of situated actions. Plans are not only resources used in advance of situated action, they are also used as a reconstruction of our actions in an explanation of what happened. However, we filter out those particulars that define our situated actions in favor of explaining aspects of our actions that can be seen in accordance with the plan. Thus,

[4] Ethnomethodology (literally, 'the study of people's (folk) methods') is a sociological discipline which focuses on the ways in which people make sense of their world, display this understanding to others, and produce the mutually shared social order in which they live. [Wikipedia, 12/27/06].

according to Suchman, in strong contrast with the view of Bratman, plans are necessarily abstract representations of our actions and as such cannot serve as an account of our actions in a particular situation. In other words, our actions are not planned in advance, but are always *ad hoc*. Let us take a brief look at how cognitive science and AI are changing to a science of *situated cognition*.

Situated Cognition

Behavioral robotics is part of a new stream in AI, evolving from text-based or symbolic architectures of expert systems, descriptive cognitive models, and indeed BDI-like planning, to reactive, self-organizing situated robots and connectionist models. Many scientists have started to write about alternatives to the "wet computer" as a symbolic planning machine, such as Suchman [10], Brooks [9], Agre [11], Fodor [12], Edelman [13], Winograd and Flores [14]. We surely are leaving out others. In this section we will briefly present the work of Bill Clancey, because he is both the father of Brahms (the activity-based BDI language and multiagent architecture we describe in this paper) and a well-known father of expert tutoring systems (a purely symbolic approach), who has "gone the other way." Clancey is one of the proponents of situated cognition, and having written two books about it, one of the experts in the field [15][16].

Situated cognition goes one-step further than Suchman's situated action, namely it posits an alternative human memory architecture to the symbolic memory architecture of Newell and Anderson [7][8]. Clancey, in [16], develops an architecture of human memory at the neural level that is quite different from an architecture of a long-term memory storing productions and a short-term memory for matching and copying text (or symbols) into a "buffer." Clancey's arguments against the "memory as stored structures" hypothesis of Newell and Anderson has far reaching consequences for the Planning Theory of Bratman, namely that BDI-type planning cannot be the process for human thinking. It are exactly the concepts on which planning is based, like *search*, *retrieve*, and, *match*, that are associated with thinking as a "process of manipulating copies of structures, retrieved from memory" and "learning [as] a process of selectively writing associations back into long-term memory [p. 3]."

What is important for *situatedness* is the notion that thinking is a situated experience over time. In Newell and Anderson's model, time plays an important role in the matching of antecedents (of productions in long-term memory) and retrieval of the matched production from long-term memory and copying into short-term memory. In other words, the time it takes to think has little to do with being in the activity of doing something in the world, but rather it has to do with being in the activity of matching, retrieving and copying of symbols. In Clancey's model, however, the overall time it takes for neuronal processes to subsequently activate in the brain *is* the essence of situated action and (subconscious) conceptualization occurs as an activity. The essential part of the Newell and Anderson models of cognition that is disputed is the sequential relation of:

$$sensation \Rightarrow perception\ memory \Rightarrow processing/thought\ and\ planning \Rightarrow action.$$

Instead, these are processes *coupled* in activation relations called *conceptual coordination*.

All else being equal, we adhere to the view that actions are situated as the starting point for developing an activity modeling approach. Not in the least, because the situated view of human activity constrains us less, by not having to commit to theories about brain function that model problem-solving tasks at the millisecond range. It frees us from having to model what is "in between our ears," and opens up our focus on modeling an agent's activities as larger time-taking situated actions within *the environment*. It finally enables us to model human behavior based on individual and social interaction with other individuals and artifacts in the world.

Activity-Based Modeling: Combining BDI with Situated Action

In the previous section, we laid the groundwork for activity-based modeling. Suchman's treatise on situated action is important, because it proofs that modeling intention to act may not only be goal-based. In fact, Suchman argues convincingly that an intention leading every-day life action is always situated in an environment, and is only goal-driven when there is a real-world problem to be solved. The overarching conclusion of this alternative view is that acting in the world is not only bounded by rationality, i.e. reasoning—or rather problem-solving—in the abstract, but is mainly situated in an environment that is historically, culturally and socially learned by its changes over time. It is thus important to not leave out the environment. In the remainder of this chapter, we define a modeling framework that combines the concepts of belief, desire and intention with that of situated action, to form an activity-based modeling approach that can be used not only to model the work practices of people, but also to model rational agent behavior in any multiagent system. By now, it should be no surprise that in this modeling framework an agent's behavior is not goal-based, but rather activity-based. This means that an agent's *desires* are *not* the possible subgoals the agent derives from its *intention* to solve a particular problem. Rather, as we will see, an agent's desires are reflected in the agent's *motives* for performing activities, and these motives are dependent on the environment and the situation, i.e. the state of the world at that moment. In the next section we describe what is meant with the word *situated*.

Modeling the Environment

The social anthropologist Jean Lave studied how people apply cognition in practice [17]. In the Adult Match Project she studied how people do math. She found that the same individuals do arithmetic completely different depending on the situation they are in. The results from her studies are significant, because solving mathematical problems has for a long time been the way scientists have studied general problem-solving in school and in the laboratory. Lave and her colleagues studied adult arithmetic practices in every-day settings, such as cooking and grocery shopping. Their work shows that solving mathematical problems is not so much constraint by how much of a math expert one is because of their math education, but because of the setting or situation in which the arithmetic activity takes place. Lave argues convincingly that, in order to understand problem solving we need to model the problem-solving process *in situ*.

Russian psychologist Lev Semyonovich Vygotsky plays an important role in developmental psychology. Vygotsky's deeply experimental work on child learning and behavior developed the notion that cognition develops at a young age through

tools and the shared social and cultural understanding of tools in the form of *internalized signs* [18]. For example, a ten-month-old child is able to pull a string to obtain a cookie. However, it is not till the child is about 18 months that it is able to remove a ring from a post by lifting it, rather then pulling on it. Vygotsky writes; "Consequently, the child's system of activity is determined at each specific stage both by the child's degree of organic development and by his or her degree of mastery in the use of tools." [p. 21] The internalization of mimicry of tool use by young infants into a shared meaning of signs is, according to Vygotsky, intertwined with the development of speech. At the same time, speech plays an important role in the organization of a child's activities. Vygotsky writes; "Our analysis accords symbolic activity a specific organization function that penetrates the process of tool use and produces fundamentally new forms of behavior." [p. 24] While we will discuss the concept of activity later on, here it is important to see the essential importance of physical artifacts used as tools *within* an activity. Similar to the role of an artifact as a tool, the role of artifacts as the products of an activity plays an equally important role in what Vygotsky calls *practical intelligence*. Figure 1. shows the use of an artifact in the activity—its role—that transforms the artifact into a *tool* or a *product* of the activity, used or created by the subject. Outside the activity the artifact is just an object in the world. To the observer the object is necessary for the activity to be performed.

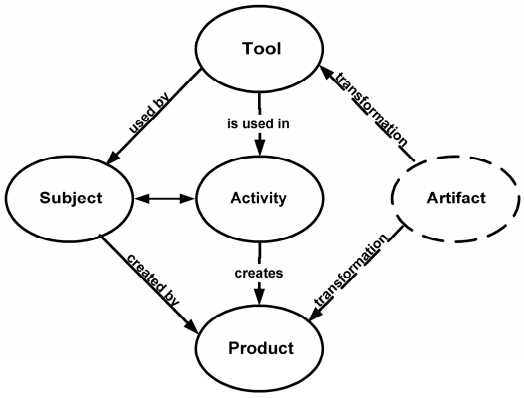

Fig. 1. Mediated relationship of artifacts in activities [3, p. 62]

Situatedness in Brahms is modeled by representing places and locations as objects in a conceptual hierarchical geography model and by representing artifacts as objects in a hierarchical object model, separate from modeling the agents and their possible activity behavior [3].

3 Modeling Places and Locations

In Brahms' geography model relevant locations are modeled conceptually as area objects in a *part-of* hierarchy. Locations are *areas*, which are *instances* of classes of areas called *areadefintions*. Areas can be part of other areas. Using these two simple

concepts we are able to model any environment as a conceptual hierarchical structure of area objects. For example, we can model the geography for UC Berkeley students studying at South Hall and going for lunch on Telegraph or Bancroft Avenue in Berkeley as follows:

```
area AtmGeography instanceof World
// Berkeley
area Berkeley instanceof City partof AtmModelGeography
// inside Berkeley
area UCB instanceof UniversityCampus partof Berkeley
area SouthHall instanceof UniversityHall partof UCB
area SpraulHall instanceof UniversityHall partof UCB
area BofA_Telegraph_Av_2347 instanceof BankBranch
     partof Berkeley
area Wells_Fargo_Bancroft_Av_2460 instanceof BankBranch
     partof Berkeley
area Blondies_Pizza_Telegraph_Av_2347 instanceof
     Restaurant partof Berkeley
area Tako_Sushi_Telegraph_Av_2379 instanceof Restaurant
     part of Berkeley
```

Being situated means that Brahms agents are located within the geography model. Area objects are special kinds of objects that can inhabit agents. Agents can have an initial location. For example, agent Joe can initially be located in area SouthHall as follows:

```
agent Joe {
  intial_location: SouthHall;
}
```

People do not stay in one location, but move around. Similarly, Brahms agents can move around within a geography model. Agent (and object) movement is performed with a *move* activity. How activities are performed is described in the next section. Although a geography model and agent and object location and movement within such a model is essential for modeling situatedness, the state of the world needs to be modeled as well. Current BDI architectures do not model the state of the world and therefore do not allow for representing a factual state of the world independent of the agent's beliefs about that state. In Brahms the state of the world is modeled as *facts*[5]. For example, the location of an agent is a fact, because, regardless of the beliefs of agents, the agents have a location within the geography model.

```
Fact: (Joe.location = SouthHall)
```

To model facts about areas we give the areas attributes and relations. For example, we can model the temperature in South Hall and the fact that South Hall is part of the UC Berkeley campus. To do this, we first define these as an attribute and relation in the area definition class. This way all university hall areas we model will inherit them.

[5] Here we deal with solipsism, since the model is an interpretation of the world by the modeler. Facts, as such, are thus not real facts, but only the modeler's representation of world states independent of the beliefs of agents.

```
areadefinition UniversityHall {
  attributes:
    public int temperature;
  relations:
    public UniversityCampus partOf;
}
```

Having defined the attributes and relations, we can have additional facts about the world.

```
Facts:
  (SouthHall partOf UCB);
  (SouthHall.temperature = 72);
```

4 Modeling Objects and Artifacts

As discussed above, artifacts in the world are important aspects for modeling situated action. Similar to the geography model, we model artifacts as object instances of a class hierarchy. Actually, areas are special kind of objects and areadefinitions are special classes. However, artifacts are more like agents than they are like areas. Artifacts are located and some can display situated behavior. Some types of objects, for instance a book, can also store information. Then there are data objects, which can represent located information conceptually. We use a class hierarchy to model these artifacts and data objects in the world. For example, to model a book Joe the student is reading we can define the following object model:

```
class Book {
  attributes:
    public String title;
    public Writer author;
  relations:
    public Student belongsTo;
    public BookChapter hasChapter
}

object BookPlansAndSituatedAction instanceof Book {
  initial_location: SouthHall;
  initial_facts:
    (current.title = "Plans and Situated Action");
    (current.author = LucySuchman);
    (current.belongsTo = Joe);
}
```

We can model the content of the book as *conceptual* chapter data objects. For example:

```
conceptual_class BookChapter {
  attributes:
    public String title;
    public String text;
}
```

We model the information held by the object as "beliefs" of the object[6]. Thus, we can model Chapter 4 in Suchman's book as a conceptual BookChapter object the Book object "stores" information about:

```
conceptual_object Chapter4 instanceof BookChapter { }

object BookPlansAndSituatedAction instanceof Book {
  initial_beliefs:
    (current hasChapter Chapter1);
    (current hasChapter Chapter2);
    (current hasChapter Chapter3);
    (current hasChapter Chapter4);
    (current hasChapter Chapter5);
    (current hasChapter Chapter6);
    (current hasChapter Chapter7);
    (current hasChapter Chapter8);
    ...
    (Chapter4.title = "Situated Action");
    (Chapter4.text = "… I have introduced the
                     term situated action …");
}
```

The key point in modeling information is to locate the BookChapter content as information held within the Book object. To do this, we model the content of the book as the *"beliefs of the book."*

Now that we discussed how situatedness is modeled in Brahms, we turn to modeling activities. To understand what activities are, we must first understand that human action is inherently social, which means it is "outside the brain" involving the environment. The key thing is that action is meant in the broad sense of an activity, and not in the narrow sense of altering the state of the world. In the next section we discuss the concept of an activity and how to model it.

5 Modeling Activities

Like Bratman's notion of humans as planners based on commonsense psychology, the notion of human behavior in terms of activities is based on the Marxist *activity theory*, developed initially between 1920s and 1930s by Russian psychologists Vygotsky and Leont'ev [18][19]. Activity theory is also a meta-theory, as is commonsense psychology, and can better be seen as a framework with which human behavior can be analyzed as a system of activities. Activity theory has become more established in last twenty years in its further development by educational and social scientists [20][21], as well as human-computer interaction scientists [22].

The unit of behavioral analysis in Activity theory is, not surprisingly, an *activity*. Activity theory defines a relationship between an activity and the concept of *motive*. Motives are socially learned and shared with people from the same culture, organization, or more generally a *community of practice* [23]. For instance, all activities of flight controllers and directors at NASA Johnson Space Center's Mission

[6] We call information contained in an object also beliefs to minimize language keywords.

Control for the International Space Station are driven by the *shared motives* of keeping the space station healthy and making sure the space station crew is safe. Motives and goals, at first glance, seem to be similar, but motives, unlike goals, are situational and environmental states that we want to maintain over relatively long periods of time, such as keeping the astronauts safe and healthy onboard the space station. Motives are not internal states that drive our minute-by-minute action and deliberation, but enable a shared external understanding of our actions in the world (e.g. flight controllers are not, every minute of their time, working on tasks that are driven by the goal of keeping astronauts alive). Motives keep us in "equilibrium" with the system's environment and allow us to categorize actions as socially situated activities (e.g. everything flight controllers do is in line with their motives and have as an overall result that the ISS is safe and the astronauts are healthy).

Describing human activities as socially situated does not mean people doing things together, as in "socializing at a party" or "the social chat before a meeting." Describing human activities as social means that the tools and materials we use, and how we conceive of what we are doing, are socially constructed, based on our culture. Although an individual may be alone, as when reading a book, there is always some larger social activity in which he or she is engaged. For instance, the individual is watching television in his hotel, as relaxation, while on a business trip. Engaging in the activity of "being on a business trip," there is an even larger social activity that is being engaged in, namely "working for the company," and so on. The point is that we are always engaged in multiple social activities, which is to say that our activities, as human beings, are always shaped, constrained, and given meaning by our motives and our ongoing interactions within a business, family, and community. An activity is therefore not just something we do, but a manner of conceiving our action and our interaction with the social environment we are in. Viewing activities as a *form of engagement* emphasizes that the conception of activity constitutes a means of coordinating action, a means of deciding what to do next, what goal to pursue; In other words, a manner of being engaged with other people and artifacts in the environment. The social perspective adds the emphasis of time, rhythm, place, shared understanding and a well-defined beginning and end.

This can be contrasted with the concept of a task, which in AI is defined as being composed of only those subtasks and actions that are relevant to accomplish the goal of the task. A task is by definition goal-driven, and the conception of a task *leaves out* the inherent social, cultural and emotional aspects of why, when, where and how the task is performed within the socially and culturally bounded environment. Clancey provides a nice example that portrays the difference between an activity and a task:

> "All human activity is purposeful. But not every goal is a problem to be solved (cf. Newell & Simon 1972), and not every action is motivated by a task (cf. Kantowitz & Sorkin 1983). For example, listening to music while driving home is part of the practice of driving for many people, but it is not a subgoal [subtask] for reaching the destination. Listening to music is part of the activity of driving, with an emotional motive." [24, p. 1]

In this view of human behavior, activities are socially constructed engagements situated in the real world, taking time, effort and application of knowledge. Activities have a well-defined beginning and end, but do not have goals in the sense of

problem-solving planning models. Viewing behavior as activities of individuals allows us to understand why a person is working on a particular task at a particular time, why certain tools are being used or not, and why others are participating or not. This contextual perspective helps us explain the quality of a task-oriented performance. In this sense, as is shown in Figure 2, activities are orthogonal to tasks and goals.

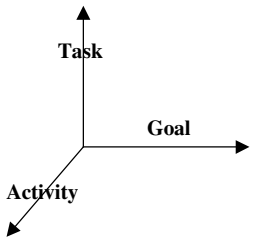

Fig. 2. Dimensions of behavior [3, p. 55]

While modeling an activity we might want to use a goal-plan approach to represent a specific task to solve a particular problem, but this can also be done within the activity model itself. This is to say that a goal-driven planning approach is subsumed by an activity-based approach, and a goal-oriented task can be modeled with a goal-driven activity. An activity-based approach is more general and allows the modeling of all kinds of activities, including activities that are not necessarily work-oriented and goal-driven, but are based on social and cultural norms, such as the kinds of activities described by Clancey [24, p. 4]:

- Intellectual: These include activities that represent any form of conceptual inquiry or manipulation of things or ideas. This includes work-oriented problem solving, but also activities that are less directed, such as exploration or browsing; artistic and documentation activities, such as taking photographs, writing, etc.
- Interactional: These include activities in which we interact directly with other people, such as in a fact-to-face conversation, or using a communication artifact, such as a telephone or fax machine.
- Physical/body maintenance: These include activities where we "take care" of ourselves physically, such as eating, sleeping, etc.

5.1 Activities Are Like Scripts

Engström integrates the notion of activity with the notion of practice. How do we know what to do, what actions we should take, when and how? Habitual scripts drive our practice, our learned actions in a context: "At an intermediate level, the continuity of actions is accounted for by the existence of standardized or habitual scripts that dictate the expected normal order of actions." [25, p. 964].

In cognitive science and AI, the notion of scripts was developed by Shank and Abelson for the purpose of language processing and understanding of knowledge structures. Although they did not refer to activity theory, they did refer to something

that sounds very much like the notion of behavior as situated activities: "People know how to act appropriately because they have knowledge about the world they live in." [26, p. 36]. Unlike Engström, Shank and Abelson focus not on how and why people get to engage in a social activity, although they use stories of social activities (such as eating in a restaurant) as examples, but they focus on the knowledge that people need to bring to bear to understand and process a situation. They argue that known scripts, which can be based on general knowledge (i.e. knowledge we have because we are people living in the world) and specific knowledge (i.e. knowledge we get, based on being in the same situation over and over), are what people use to understand a story:

> "A script is a structure that defines appropriate sequences of events in a particular context. A script is made up of slots and requirements about what can fill those slots. The structure is an interconnected whole, and what is in one slot affects what can be in another. Scripts handle stylized everyday situations. They are not subject to much change, nor do they provide the apparatus for handling totally novel situations. Thus, a script is a predetermined, stereotyped sequence of actions that defines a well-known situation … There are scripts for eating in a restaurant, riding a bus, watching and playing a football game, participating in a birthday party, and so on." [p. 41]

Modeling activities as scripts enables us to model people's behavior as situated activities and is how we model activities in the Brahms language. This is the subject of the next section.

6 Activities in Brahms

In Brahms, an activity system is modeled by individually performed activities by agents. An agent while executing activities can be located in a geography model (see geography discussion above), representing the agent executing located behaviors. Activities are like scripts having "predetermined, stereotyped sequence of actions that defines a well-known situation." An activity abstracts a behavioral episode into a sequence of subactivities, or, if not further decomposed, it abstracts an episode into a single activity taking time, using resources—objects in the object model—or creating products—other objects in the object model—or both. In Brahms we call these *composite-* and *primitive activities* subsequently.

6.1 Primitive Activities

Activities define episodes of actions that an agent *can* perform. They are not the episodes themselves, but provide the "slot definitions" for possible instantiations of them. Following is the grammar definitions, in BNF form, for a *primitive activity* in the Brahms language:

```
primitive activity activity-name( { param-decl
[ , param-decl ]* } )
{
{ display : literal-string ; }
{ priority : [ unsigned | param-name ] ; }
```

```
{ random : [ truth-value | param-name ] ; }
{ min duration : [ unsigned | param-name ] ; }
{ max duration : [ unsigned | param-name ] ; }
{ resources : [ param-name | object-name ] [ ,
              [param-name | object-name ]*; }
}
```

A primitive activity can have parameters that at runtime are instantiated with parameter values, depending on the type in the parameter declaration. Parameters can be assigned to the slots in the activity definition, enabling the activity to be situated in a particular instantiation. The *display* slot provides a way to give the activity a more appropriate name when displayed. For example, the display string can include blank spaces. The *priority* slot is a positive integer that gives the activity a priority. Activity priorities are used by the agent's engine in case more than one activity can be chosen in the next event cycle. The *max_duration* slot gives the (maximum) duration of the primitive activity. Primitive activities only take an amount of time representing the time the agent performs the activity. In case the modeler wants a random time to be assigned, the modeler has to provide a *min_duration* as well, and set the *random* slot to *true*. For example, the activity bellow simulates the life of an agent in one simple primitive activity that takes a random lifetime:

```
primitive activity Being_Alive( int pri,
  int max_life_time, Body body_obj )
{
  display : "Alive and kicking" ;
  priority : pri ;
  random : true ;
  min duration : 0 ;
  max duration : max_life_time ;
  resources : body_obj ;
}
```

The above Brahms activity code is only the definition of the activity. It will not make the agent execute the activity. For this, the activity needs to be placed in a situation-action rule called a *workframe*. A workframe can have a set of preconditions that are matched against the belief-set of an agent. When there exist a set of beliefs that match the precondition, the variables in the precondition are bound to those matching the beliefs, and an instance of the workframe is created. The body of the workframe-instance is then executed. For example:

```
workframe wf_Being_Alive {
  repeat: true ;
  when (knowval(current.alive = true)) {
  do {
   Being_Alive( 100, 3600 ) ;
   conclude((current.alive = false), bc: 100, fc: 100);
  }
}
```

In this workframe, the precondition *knownval(current.active =true)* is matched against the belief-set of the agent. The *current* keyword means the agent that is

executing the workframe. The *knownval* predicate returns true if the agent has a belief that matches exactly the precondition. Other possible precondition predicates are *known, unknown* and *not*, which subsequently would return true, if a) the agent has *any* belief about the agent-attribute pair *current.alive* irrespective of the value, b) the agent has *no* belief about this agent-attribute pair, and c) the agent has a belief about the agent-attribute pair, but its value is not *true* (i.e. in this specific case the value is either *false* or *unknown*).

If the preconditions are all true, a *workframe instantiation* for each set of variable bindings is created. In this above example, there are no variables and thus there will only be one workframe instantiation created. A workframe instantiation is an instance-copy of the workframe with every precondition variable bound, available to be executed (see section bellow about composite activities). If a workframe instantiation is selected as the current to be executed, the *body* of the workframe (i.e. the *do-part*) is executed. In the above workframe, the *Being_Alive* activity is subsequently executed. It should be noted that the agent's engine matches the preconditions and starts executing the first activity in the workframe all at the same time (i.e. the same simulation clock tick).

It is at the start of the execution of the activity that the activity duration time is calculated. For the *Being_Alive* activity a random duration, between the value of the min- and max-duration slot, is selected. This then becomes the duration of the of the workframe instantiation for *wf_Alive*. After the activity ends, the *conclude* statement is executed in the same clock tick. In other words, conclude statements do *not* take any time, only activities. The *conclude* statement creates a new belief for the agent, and/or a new fact in the world. In the above example there is a belief created for the agent in hundred percent of the time (*bc:100*). There is also a fact created in hundred percent of the time (*fc:100*). If the belief- or fact certainty factor were set to zero percent, no belief or fact would be created. Using these certainty factors, the modeler thus has control over the creation of facts in the world and beliefs of an agent.

6.2 Composite Activities

How can we represent complex activities that are decomposed into lower-level activities without using a goal-driven planning approach? This is an essential question in finding a way to model every-day situated activities that are not goal-driven. One of the important capabilities of people is that we can easily resume an activity that was previously interrupted by another activity. For example, while in the activity of reading e-mail your cell phone rings. Without hesitation you suspend the "reading work-related e-mail" activity and start your "talking to your child on the phone activity," switching not only your perceptual-motor context, from reading e-mail on your computer to talking on a cell phone, but also switching your situated social context from being an employee reading work-related e-mail to the role of a father. One question is how you get to change this context? It is obvious that this is not triggered by a sub-goal of the reading e-mail activity. It is detecting (i.e. hearing) your cell phone ringing that makes you interrupt your e-mail reading activity, and the social norms of today makes the answering your cell phone activity most likely be of higher priority.

The organization principle we use to enable this type of interrupt and resume behavior is Brooks' subsumption architecture [9]. Brooks developed his subsumption architecture for situated robots, with the premise that this architecture enables the development of robots without a central declarative world model and without a planning system that acts upon that model. In Brahms agents do have a declarative world model, namely the belief-set of the agent represent the agent's view of the world. However, similarly to Brooks' subsumption architecture, Brahms agents do not use a goal-directed planning system, but rather an activation system that is based upon a "computational substrate that is organized into a series of incremental layers, each, in a general case, connecting perception to action." [p.39]. In our case the substrate is a hierarchical network of situation-action rules with timing elements in the form of primitive activities with duration. This hierarchical workframe-activity network enables flexible context switching between independent activities at all levels in the hierarchy. Similar to Brooks' augmented finite state machine (AFSM) language, each individual Brahms agent engine executes activities as manageable units selectively activated and deactivated (i.e. interrupted or impassed). Each Brahms agent engine works similar to Brooks' AFSMs, namely "[(activity behaviors] are not directly specified, but rather as rule sets of real-time rules which compile into AFSMs in a one-to-one manner." [p. 40].

Activity Subsumption

It is obvious that we want to be able to decompose primitive activities into more complex activities in order to model more complex behavior. For example, we want to be able to decompose the *Being_Alive* activity into more specific activities that the agent performs while alive. In Brahms this is done with a *composite activity*. Following is the grammar definitions, in BNF form, for a *composite activity* in the Brahms language:

```
composite-activity activity-name (
{ param-decl [ , param-decl ]* } )
{
{ display : literal-string ; }
{ priority : [ unsigned | param-name ] ; }
{ end_condition : [ detectable | nowork ] ; }
{ detectable-decl }
{ activities }
{ workframes }
{ thoughtframes }
}
```

The "slots" of a composite activity are different from that of a primitive activity, except for the display- and priority slots. A composite activity has sections to define the decomposed behavior of the activity: *detectable-decl*, *activities*, *workframes*, *thoughtframes*. First, the *end_condition* slot declares how a composite activity can end. There are two possibilities; 1) the activity ends when there is nothing more to do. This is called end when there is no work (i.e. no workframe or thoughtframe in the activity fires); 2) the activity ends, because the agent has detected some condition in the world that makes it end the activity. In this case, when the activity should be ended is defined in the *detectable-decl* slot, by declaring so-called *detectlables*. How

detectables work is explained bellow in the section about reactive behavior. Here we first explain the other sections of a composite activity. The other three sections define a composite activity in terms of more specialized activities in the activities section, and workframes calling these activities in the workframes section. The thoughtframes section contains thoughtframes. Thoughtframes are simple forward-chaining production rules. Actually, thoughtframes are like workframes except they cannot call activities but only conclude beliefs, based on matching belief preconditions. Also, thoughtframes do not take any simulated time, unlike workframes that always perform activities that take time. Thus, composite activities allow the definition of detailed scripts for well-known situations (a.k.a. work practice).

Next, we provide an example of how composite activities are used. We will use the example from before about an agent who is in the activity of being alive. We expand the example to have the agent go from being alive to being in a coma. Instead of defining the *Being_Alive* activity as a primitive activity that simply takes an amount of time, let us define this activity in more detail as a composite activity of two subactivities called *PAC_1* and *PAC_2*. Both of these subactivities are primitive activities being called in two separate workframes *wf_PAC_1* and *wf_PAC_2*:

```
composite_activity Being_Alive( ) {
   priority: 0;
     detectables:
       detectable det_Impasse {
         detect((current.headTrauma = true))
           then impasse;
       }

     activities:
       primitive_activity PAC_1(int pri) {
         display: "PAC 1";
         priority: pri;
         max_duration: 900;
       }

       primitive_activity PAC_2(int pri, int dur) {
         display: "PAC 2";
         priority: pri;
         max_duration: dur;
       }

     workframes:
       workframe wf_PAC_1 {
         repeat: true;
         when (knownval(current.execute_PAC_1 = true))
         do {
           PAC_1(1);
           conclude((current.headTrauma = true), fc:50);
         } //end do
       } //end wf_PAC_1

       workframe wf_PAC_2 {
```

```
        repeat: true;
        do {
          PAC_2(0, 1800);
          conclude((current.execute_PAC_1 = true), bc:25);
        PAC_2(0, 600);
        } //end do
      } //end wf_PAC_2
} //end composite activity Being_Alive
```

The left side of Figure 3 shows the workframe-activity subsumption hierarchy for the *Being_Alive* activity. The *wf_Being_Alive* workframe from before now calls the composite activity *Being_Alive*, instead of the previous primitive activity. From the above source code you can see that, at first, the workframe *wf_PAC_2* will fire, because that workframe does not have any preconditions and can thus fire immediately. This workframe will fire forever due to the *repeat: true* statement. Thus, if nothing else changes, the agent will first execute primitive activity *PAC_2* for 1800 clock ticks, and then again for 600 clock ticks, after which the *wf_PAC_2* fires again, and again. However, the *conclude* statement, in between the two *PAC_2* activity calls concludes its specified belief 25% of the time, due to the belief-certainty of 25. This means that approximately one out of four executions of the workframe *wf_PAC_2* the agent gets the belief to execute *PAC_1*.

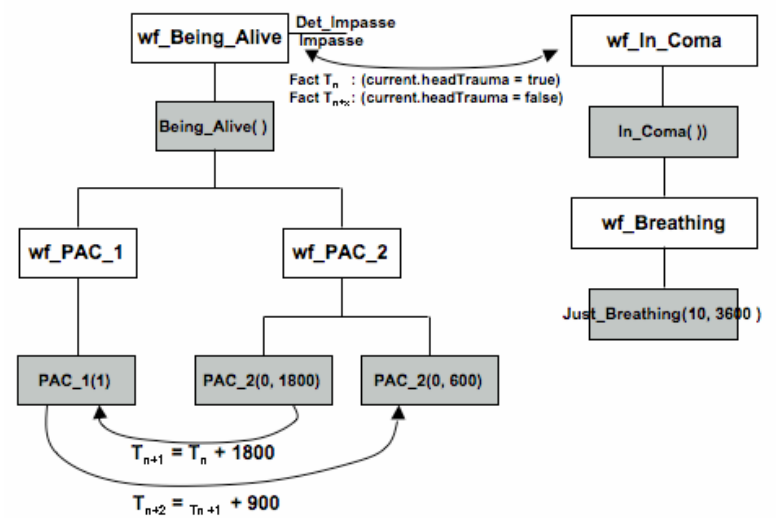

Fig. 3. Coma Model Workframe-Activity Subsumption Hierarchy

When this happens the belief immediately matches with the precondition of the *wf_PAC_1* workframe, which then becomes available to fire. Now the agent's engine needs to determine which workframe to execute in the next clock tick. This is done using the priorities of the workframes. If no priority is specified directly on the workframe, the workframe will get the priority of the highest priority activity within its body. In this example workframe *wf_PAC_2* has a priority of zero, because both

PAC_2 activity calls get a priority of zero as a parameter value. Workframe *wf_PAC_1* on the other hand will get the priority one, due to *PAC_1* having a priority parameter value of one. Thus, *wf_PAC_1* has the highest priority and the engine will pick this workframe as the next workframe to execute, with as the result that the agent will start performing *PAC_1* for its defined duration of 900 clock ticks. Workframe *wf_PAC_2* will be *interrupted* at the point of the beginning of the second *PAC_2* activity call, right after the execution of the conclude statement. Therefore, when the *wf_PAC_1* workframe finishes its execution of the *PAC_1* activity (i.e. 900 clock ticks later) the agent will switch immediately back to the execution of the interrupted *wf_PAC_2* workframe, and will continue with executing the second *PAC_2* activity in the workframe.

This example shows the selective activation and deactivation of subsumed activities via a perception-action architecture based on an activity-priority scheme. Agents can easily switch their activity context, independent of the level in the activity network hierarchy. When a workframe with activities within it becomes available, no matter where in the workframe-activity network, the workframe with the highest priority becomes the agent's current context—this is called the agent's *current work*—and thus the activities within this current workframe will execute.

Thus far the example shows how an agent can easily switch activities based on new belief creation, i.e. either through performance of activities in workframes or through pure reasoning in thoughtframes. Next, we will show how an agent can also react to changes in the environment by detecting *facts* in the world. This approach enables flexible reactive behavior by the agent due to changes in "the outside" environment.

6.3 Reactive Behavior

The example of the composite *Being_Alive* activity shows how agents can switch activity contexts based on "internally" created beliefs and activity-workframe priorities. However, we want agents be able to react to fact chances in the environment outside of the agent. This is done through the definition of detectables in activities and workframes. The above source code shows the *det_Impasse* activity detectable declared in the *Being_Alive* activity. What this means is that while the agent is "in the *Being_Alive* activity" this detectable is active and the agent will detect any *fact* changes, made by any agent or object, to the *headTrauma* attibute of the agent.

The process of firing a detectable goes as follows: When a new fact is detected 1) the fact becomes a belief for the agent that can then trigger a switch in activity context as shown in the example before, 2) the agent matches the detectable condition of all active detectables that refer to the fact and checks if the fact matches the condition, 3) in case the detectable condition matches the fact the detectable action statement is executed. There are four types of detectable actions possible, *continue*, *abort*, *complete*, and *impasse*.

In the Coma model source code for the *Being_Alive* composite activity, the *det_Impasse* detectable has an *impasse* action. An *impasse* action means that the activity will be impassed (i.e. interrupted) until the impasse condition is resolved. The impasse condition is the detect condition of the detectable. Thus, in the

det_Impasse detectable, the activity is impassed when the fact *(current.headTrauma = true)* is created. The *wf_Being_Alive* will be impassed until the fact is changed.

Figure 3, on the right hand side, shows an additional workframe for the agent called *wf_In_Coma*. This workframe has a precondition that matches on the belief *(current.headTrauma = true)* created by the detection of the fact, and will activate the *In_Coma* activity. If the *In_Coma* activity, or something else, at some point in the future creates the fact *(current.headTrauma = false)*, the impasse is resolved and the *Being_Alive* activity will be available again for the agent to continue executing it, depending its priority compared to other available activities.

The above example shows how a Brahms agent's behavior is modeled as decomposed script-like activities that are executed using a perception-action subsumption architecture, enabling both rational and reactive behavior. There is one more important organizational element in the Brahms language that provides an important agent organizational modeling capability. This is briefly discussed in the next section.

7 Modeling Agent Organization

Societies consist of many different types of behaviors. As a design principle we want to be able to organize these behaviors in categories that are logically and culturally understandable, and moreover useful for the design of complex agent communities. In Brahms there is the notion of a *group* as the concept allowing the creation of agent pastiches. Not only did we develop groups based on the notion of organization in categories or classes, groups are based on the important idea of *communities of practice*:

> "Being alive as human beings means that we are constantly engaged in the pursuit of enterprises of all kinds, from ensuring our physical survival to seeking the most lofty pleasures. As we define these enterprises and engage in their pursuit together, we interact with each other and with the world and we tune our relations with each other and with the world accordingly. In other words we learn. Over time, this collective learning results in practices that reflect both the pursuit of our enterprises and the attendant social relations. These practices are thus the property of a kind of community created over time by the pursuit of a shared enterprise. It makes sense, therefore, to call these kinds of communities *communities of practice*." [23, p. 45]

Groups are thus meant to be the representation of the *practices* of communities of agents. People belong to many communities at once, blending practices from many different groups into one, so called, *work practice* [3]. We are students, parents, workers, children, engineers of a particular kind, etc. But we also are social creatures, belonging to a community of like-minded individuals playing sports, having hobbies, going to the same coffee shop every day, playing roles in an organization, etc. It is therefore that people belong to many communities of practice at once. Agents in the Brahms language can thus belong to many different groups, enabling the design of complex organization models of agents.

Groups in Brahms, just like agents, can contain *attributes*, *relations*, *initial beliefs and facts*, *activities*, *workframes* and *thoughtframes*. An agent can be *a member of* a group, but a group itself can also be a *member of* another group, enabling the design of a complex hierarchical structure of agent group-membership. An agent or group that is a member of another group will *inherit* all of the contents of that group. For example, agent *Joe*, from our first example, will inherit its behavior as a student from the *Student* group. But, we can create another group, let's call it *HumanBeing*, in which we put the *BeingAlive* and *BeingInComa* activities. We can now have agent *Joe* inherit both the behavior from the *Student* group and from the *HumanBeing* group;

```
group HumanBeing {…}
group Student {…}
agent Joe memberof HumanBeing, Student {…}
```

Groups and multiple group inheritance allows us to model common behavior as communities of practice, from which all group members will inherit its behavior. Using this simple group membership relation we can design any type of organization we want. Groups can represent a functional organization, such as groups representing particular roles that agents play, performing certain functions (i.e. activities) in an organization. However, groups can also represent social organizations, such the relationships and social knowledge people share about a particular community. For example, everyone who comes to drink coffee at the same coffee shop everyday knows the name of the shop's barista.

8 Conclusions

In this chapter, we discussed some of the issues and limitations of BDI agent architectures based on Bratman's Planning Theory that explicitly states that humans are goal-driven planning agents. We posited an alternative view of human behavior based on a combined notion of situated action, cognition in practice, situated cognition and activity theory. Based on this alternative view, we developed an activity-based theory of behavior that allows for the description of complex behavior that is not only based on goals. We then described the Brahms multiagent language and execution architecture that implements this activity-based theory into a BDI agent language. Brahms allows for designing and implementing complex agent societies not based on goal-based planning agents.

The Brahms multiagent language, for each agent, "groups multiple processes (each of which turns out to be usually implemented as a single [composite activity]) into behaviors. There can be message passing, suppression and inhibition between processes within a [composite activity], and there can be message passing, suppression and inhibition between [composite activities]. [Activities] act as abstraction barriers, and one [activity] cannot reach inside another." [9, p.41].

Compared to the goal-driven paradigm, the Brahms activity-based paradigm is a more flexible execution paradigm. In a goal-driven execution engine only sub-goal contexts within the task being executed can be called as the next action, unless the current task is finished and the current goal is or is not reached and thus "popped off"

the goal stack. This limits an agent's ability to flexibly react to new belief "perceptions" unrelated to the current goal it is trying to reach.

With the simple examples in this paper, we hope we have convincingly shown that Brahms agents are both rational and reactive, and use composite architecture with situation-action rules to implement a perception-action approach similar to Brooks' behavioral architecture, all *without* the use of goals and goal-driven planning. However, it should not be forgotten that a forward-driven approach might just as well implement goal-directed behavior as a backward-driven goal-based approach. It is thus that in Brahms we can implement goal-driven activities without any problem, and it can be said that Brahms enables modeling of agent behaviors much more flexibly than a goal-based planning architecture. In other words, the activity approach is more general than the goal-based approach, which justifies the title of this paper and our claim that goals develop *within* an activity, but they are not the *driving force* of behavior, and are only useful in activities where problem solving is necessary. In other words: "All human behavior is activity-based, but not every activity is a problem to be solved."

References

1. Bordini, R.H., Dastani, M., Meyer, J.C. (eds.): Foundations and Practice of Programming Multi-Agent Systems. Dagstuhl Seminar Proceedings 06261, 25.06. - 30.06, Schloss Dagsthuhl (2006)
2. Clancey, W.J., et al.: Brahms: Simulating practice for work systems design. International Journal on Human-Computer Studies 49, 831–865 (1998)
3. Sierhuis, M.: Modeling and Simulating Work Practice; Brahms: A multiagent modeling and simulation language for work system analysis and design. In: Social Science Informatics (SWI), University of Amsterdam. SIKS Dissertation Series No. 2001-10, p. 350. Amsterdam, The Netherlands (2001)
4. Bratman, M.E.: Intention, Plans, and Practical Reason. The David Hume Series of Philosophy and Cognitive Science Reissues. CLSI Publications, Stanford, CA (1999)
5. Bordini, R.H., et al. (eds.): Multi-Agent Programming: Languages, Platforms and Applications, Springer Science+Business Media, Inc. (2005)
6. Simon, H.: A behavioral model of rational choice. Quarterly Journal of Economics 69, 99–118 (1955)
7. Newell, A.: Unified theories of cognition. Harvard University Press, Cambridge, MA (1990)
8. Anderson, J.R.: Rules of the mind. Lawrence Erlbaum Associates, Hillsdale, NJ (1993)
9. Brooks, R.A.: Cambrian intelligence: the early history of the new AI. MIT Press, Cambridge, MA (1999)
10. Suchman, L.A.: Plans and Situated Action: The Problem of Human Machine Communication. Cambridge University Press, Cambridge, MA (1987)
11. Agre, P.E.: Computation and Human Experience. In: Pea, R., Brown, J.S. (eds.) Learning in doing: Social, cognitive, and computational perspectives. Cambridge University Press, Cambridge, MA (1997)
12. Fodor, J.A.: The Modularity of Mind. The MIT Press, Cambridge, MA (1987)
13. Edelman, G.M.: Bright Air, Brilliant Fire: On the Matter of the Mind. BasicBooks, New York, NY (1992)

14. Winograd, T., Flores, F.: Understanding Computers and Cognition. Addison-Wesley Publsihing Corporation, Menlo Park, CA (1986)
15. Clancey, W.J.: Situated Cognition: On Human Knowledge and Computer Representations. Cambridge University Press, Cambridge (1997)
16. Clancey, W.J.: Conceptual Coordination: How the mind Orders Experiences in Time. Lawrence Erlbaum Associates, Mahwah, New Jersey (1999)
17. Lave, J.: Cognition in Practice. Cambridge University Press, Cambridge, UK (1988)
18. Vygotsky, L.S.: In: Cole, M., et al. (ed.) Mind in Society: The Development of Higher Psychological Processes. Harvard University Press, Cambridge, MA (1978)
19. Leont'ev, A.N.: Activity, Consciousness and Personality. Prentice-Hall, Englewood Cliffs, NJ (1978)
20. Engström, Y., Miettinen, R., Panamaki (eds.): Perspectives on Activity Theory. In: Pea, R., Brown, J.S. (eds.) Learning in doing: Social, Cognitive, and Computational Perspectives. Cambridge University Press, Cambridge, MA (1999)
21. Chaiklin, S., Lave, J. (eds.): Understanding practice: Perspectives on activity and context. In: Pea, R., Brown, J.S. (eds.) Learning in doing: Social, cognitive, and computational perspectives. Cambridge University Press, Cambridge, MA (1996)
22. Nardi, B.A.: Context and Consciousness: Activity Theory and Human-Computer Interaction. The MIT Press, Cambridge, MA (1996)
23. Wenger, E.: Communities of Practice; Learning, meaning, and identity (Draft). In: Pea, R., Brown, J.S. (eds.) Learning in doing: Social, cognitive and computational perspectives. Cambridge University Press, Cambridge, MA (1998)
24. Clancey, W.J.: Simulating Activities: Relating Motives, Deliberation, and Attentive Coordination. Cognitive Systems Research 3(3), 471–499 (2002)
25. Engeström, Y.: Activity theory as a framework for analyzing and redesigning work. Ergonomics 43(7), 960–974 (2000)
26. Schank, R., Abelson, R.: Scripts, Plans, Goals, and Understanding. Lawrence Erlbaum Associates, Inc. Hillsdale, NJ (1977)