Agent-Oriented Programming

Experimental Version of Enhanced AOP

Mostafa Raad Kamal

Athabasca University

Canada

Abstract:

AI has become an increasingly helpful and powerful tool for rational people. This is the time for modern society to begin adopting Agent-Oriented Programming because of its novel usage of raw AI concepts to return findings that can be scaled with human-expert-level expectations. Object-oriented programming was introduced 60 years ago to abstract away the internal complexity of software by giving the programmer the power of orientation through modularizing it. OOP and AOP albeit work almost in a similar fashion, the main difference between them is that OOP is constructed on modules that solve a problem by communicating with one another. Each module has specific routines and commands encoded to translate incoming messages and inform other modules to reach an acceptable result. In contrast, AOPs operate with one another maintaining their beliefs, capabilities choices and possibly other similar notions. This is a remarkable advancement in the field of software development as it is a significant entry into a different dimension of development which can simulate human-level intelligence. It allows developers to incorporate the logical theory of AI through AOP, this facility is well sought after in this AI era.

Introduction:

The research requires readers to have an idea of the basic functionality of an agent and how it interacts with an environment which can be found in the methodology in this paper. It is expected that the readers also have some knowledge of how model-based system works. In addition to that, the paper aims to give a general idea to the technical people of the improved version of AOP that will be achieved through engineering the Commitment module into the current popular BDI system. The modified information of this paper is strongly based on this novel paper by Yoav Shoham (1992) which is "*Agent-Oriented Programming"* and draw connection between insights from various other sources. I tried to structure this paper in such a way that it provides detailed information about the basic structure of an agent's architecture, following a remodel of the current AOP model, and then ended with a touch on the surface of the overview of the multi-agent system. In this detailed exploration, inside of it the BDI system will be experimented with a separate module Commitment and examining its effect in the Experimentation in Real AOP model section which should be the main focus of this paper. You are mistaken if you think that such a level of competence (human-like

intelligence) is within the reach of the state of the art in AI. This new enhanced version is designed to come to understand how the internal function is attached and operating together, to have it seeing connecting all of these from inside will help technical people to decide whether the subject is worth further research.

Methods:

a)Methodology in Agent-Based Programming:

All agents incorporate four modalities in the mental compartment instead of three: 1. Beliefs which is the knowledge that the agent believes about their environment, itself and other agents; 2. Desire is a state that the agent wants to achieve and 3. The Intention which a collection of a sequence of steps to the achievement of a desire 4. Commitments- To make it happen by processing the intention. The BDI which is Belief-Desire-Intention (BDI) system has these four mental attitudes respectively representing the information, motivational and deliberative steps to continue and lastly having the intention in real action. This model has several interconnected functionalities that are required to make a decision:

1. There is a function which is a belief revision function that receives input information from the environment (e.g. sensors) and it is responsible for updating the belief base.

2. The previous function generates more options that can become current desires based on the belief base and the intention base. We have a filter that is responsible for updating the intention base and taking into account its previous state, the current belief base, and the desired base.

3. Finally, an intention is chosen and then commitment gets executed.

To have all these interconnected functions expressed abstractly, it is all controlled by a basic loop. “It reads the current messages and updates its mental state (Such as BDI). Execute the commitments for the current time, possibly resulting in further belief change. Actions to which agents are committed include communicative ones such as informing and requesting.” (Shoham, 1992,p.68)

b)Agent System Architecture:

I)BSM Model:

Since organizing many behavioural states, therefore fitting our concept into the BSM model is the right thing to do. It is taken from researcher Peter Novák and Wojciech Jamroga on their research *“Code Patterns for Agent-Oriented Programming”*. In the BSM framework, an agent program regarding mental state transformers is encoded**.** Let us consider an agent acting in a physical environment. It consists of a KR module. The notion of a KR module is an abstraction of an agent's partial knowledge base. KR is the Knowledge representation module, typically denoted by M. Each of them is an agent's knowledge base. Formally, a KR module Mi= (Si, Li, Qi, Ui) is characterised by a knowledge representation language Li, a set of states Si ⊆ 2^Li, a set of query operators Qi: a query operator $\models \in Qi$ is a mapping $\models$: Si×Li →{ T,⊥}, a set of update operators Ui: Similarly an update operator $\oplus \in Ui$ is a mapping $\oplus$: Si ×Li →Si. Knowledge representation language Li seems to play a vital role in these mathematical expressions.

II) Mental State Transformation:

“The set of Mental state transformers enables the transition from one state to another such as from desire to intention”. (Novák and Jamroga, p.110,2009) It is the switch that allows this transition. Mental state transformation is likely to happen based on this definition:

A KR module Mi= (Si, Li, Qi, Ui),

1. Skip is a primitive mst
2. if x∈Ui and y∈Li, then xy is a primitive mst,
3. if m is a query, and n is a mst, then m −→ n is a conditional mst
4. if n and n* are mst's, then n | n*(Choice) and n ∘ n *(Sequenced transition) are mst's

III)Capabilities:

This architecture consists of three bases Goal Base=g and Knowledge Base=b and an Interface to an Environment= E in which the agent acts. We assume the basic interfaces for the belief and goal bases: Query operators ⊨B, ⊨G and update operators ⨁B, ⨁G, -B, -G for a KR language formula assertion/retraction respectively. We also assume the following code annotation function A: A(⨁iϕ) ≡ Oϕ it expresses that Agent (i) believes (O) that the formula that has been added is true because of positiveness ⨁ and resulted in the given formula I think it means the operation is successful, A(iϕ) ≡ O¬ϕ An agent is just given the formula will return the negation of it, neutrality will make the Agent taking the current operation negatively and A(⊨iϕ) ≡ ϕ for i ∈{B, G}, where ϕ ∈Li, this querying will return the formula that is given to the Agent (i) with some conditions attached to it. To find and return an item from an environment that has an unfriendly agent. The pseudocode is like this:

"[FIND]A(FIND) ⇒ [FIND∗] holds(item42)

[RUN AWAY]A(RUN AWAY) ⇒ [RUN AWAY∗] safe" (Novák and Jamroga, p.110,2009)

This is where Capability comes into play as it triggers the agent to either run(macro: RUN AWAY) or get the item(macro: FIND) by detecting an unfriendly agent within its' vicinity. It believes that holding an item is positive because that's what the knowledge base has. However, I combined two concepts from two different papers which is a commitment along with all BDI modules. This latter part will be further discussed and experimented in the experimental section in line with the traditional research method.

The overview Agent-Oriented Program framework:

According to the research paper AOP written by Yovam Shoman which revised one published in 1992, An AOP system consists of three main components:

1. An interpreted programming language for programming agents using basic commands like REQUEST and INFORM. The semantics of the programming language will depend in part on the mental state semantics.

2. A restricted formal language with clear syntax and semantics for describing state and logic respectively.

3. An "agentifier" that transforms neutral devices into programmable agents

Experimentation of REAL AOP Model :

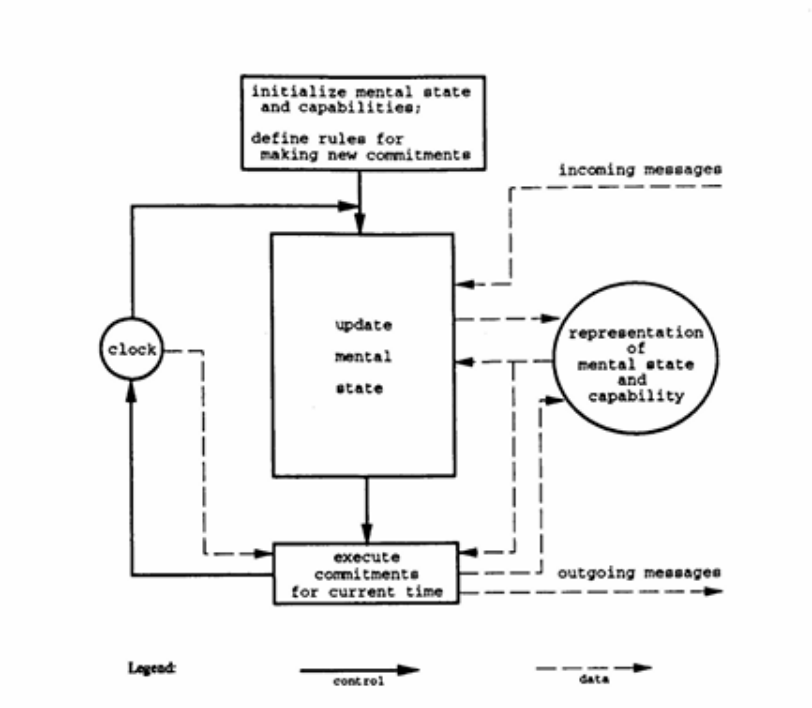

Fig 1: AOP Model

This model includes commitment and update mental state compartment.Along with the commitment module, we have the Knowledge Base, and Goal Base, these all are operational in a given Environment=E. Since the labelled diagram (Fig:1) already includes the Commitment module, we have to initialize the remaining items and see how it goes from there:

Knowledge Base={B, D, I};

Note : Fig 1: AOP model from * *Agent-Oriented Programming* * (p. 69), by Yovam Shoman. Robotics Laboratory Computer Science Department, Stanford University Stanford, CA 94305, USA 1992 by Publisher.

Goal Base =G;

Everything is happening inside the Environment= E.

The incoming message from the environment is that it has an unfriendly agent that directly enters the Mental State Compartment.

The syntax of informing is

(INFORM t a fact)

where t is a time point, a is an agent name and fact is a fact statement. In this case, we have an agent within our vicinity

(INFORM a (1 (Unfriendly Agent=b)))

if $x \in Ui$ and $y \in Li$, then xy is a primitive mst,

Time t is removed for simplicity or let's say for sequenced transition. This way I can get rid of the web of interconnected method among functions.

BB dislikes(a,b)

Knowledge Base. Belief=1

Representation of Mental State and Capability

MST

Knowledge Base. Desire=1

Representation of Mental State and Capability

MST

Knowledge Base. Intention=1

Representation of Mental State and Capability

Final MST

Commitment

Representation of Mental State and Capability

Requesting Knowledge Base of B

(REQUEST b (0 (=b))) (It will be blocked for sure)

We are using the second formula from MST in Method.

1. If y is a language representation(formula) and x acts as an update operator.

2. Querying the Agent to shift to the final mental state that's what we are getting from Cartesian product of xy coming from previous state. [Matrix transformation]

One such realistic Semi-AI example is in a Mig29 (it is missing the mental state representation), starting off your flight from the cockpit, you have this GRID on your screen with many measurements (Belief)along with all other automated services that you can use. The moving GREEN DIAMOND SHAPED BOX (Desire) turns up on the screen when the enemy is within your zone and then it starts to make the BEEPING(Intention) sound indicating it's locking on before the missile finally comes out. And the commitment it CARRIES OUT that it will eventually take it down by following the jet fighter.

There are additional mental terminologies which operate in a similar manner, such as obligation, choice or decision, and action. Then we have weaker ones, but they are sufficient to justify the terminology: Internal Consistency, Good Faith, Introspection and Persistence of Mental State.

My Research and Future Remarks:

My research process is a bit different than Yoav Shoham's; I see that he didn't bring out the configuration of how the transition is happening or the functioning of the internal gut of his model specifically for the Mental State Compartment from a Programming viewpoint. I found his paper very informative in that he developed the semantics and syntax of agents from the ground up with proper denotation. Of course, there are more details to delve deeper on AOP than I could have handled for this paper. He confessed that he lacks clarity on expanding this interesting statement "An "agentifier" that transforms neutral devices into programmable agents". Perhaps this is the most novel feature that a fully functional agent could have had as it would enable the ability to turn any machine(low-level) into an intensional(high level) machine. Here is a debrief of that under my research :

Agentifier has the 'agentification' ability that is the entire process of reconstructing the belief of another agent just like I tried to remake the belief of an unfriendly agent which is coding a device into your own terms and conditions. In the future, when we have robots required to work along with human beings for rapid economic growth, robots will be able to change other robots or inferior/neutral devices through wireless or without wireless protocol or just through communicating (advanced ones). What would be the core difference between human beings teaching a lesson to another human being? There isn't any. Lastly, another thing that came to my mind while writing this paper is the switchability between Machine-Operated and Soul-Operated. To consider long-term goals, the soul might change into a machine-operated for immortality but not the other way around, this could be the extinction of humans in the Machine-Operated World.

Rectifying pitfalls of Agent-Oriented Programming

It has been included for good practice of AOP.

Political Pitfalls

You oversell agents: “But agents are not a magical problem-solving paradigm: tasks that are beyond the scope of automation using non-agent techniques will not necessarily be made tractable simply by adopting an agent-based approach” (Wooldridge and Jennings, p.386). That explains the limitation of an agent.

Getting religious or dogmatic about agents: “Such groups exhibit the single technique syndrome- if the only tool you possess is a hammer, then everything looks like a nail” (Wooldridge and Jennings, p.386). Broadening our thinking ability to accept other techniques otherwise, you would never know of better techniques that do exist.

Management Pitfalls:

You don’t know why you want agents and what they are good for. You want to build generic solutions to one-off problems. You confuse prototypes with systems because there are too many complexities to consider when making a prototype as they operate with human-level capabilities.

Conceptual Pitfalls:

Agents are not a silver bullet: One dangerous fallacy is that AOP has reached a believable state, as it is still largely untested.

You can’t forget what you are in it for and depart from the main purpose, which could be that you forget that you are developing distributed software by confusing buzzwords with concepts.

Analysis And Design Pitfalls :

You don’t exploit related technology: “While the exact set and degree of those technologies which are related vary between applications, many agent projects could benefit from exploiting available technology from the following fields: distributed computing platforms (such as CORBA [19]) to handle low-level inter-operation of heterogeneous distributed components: database systems to handle large information processing requirements; and expert systems to handle reasoning and problem-solving tasks” (Wooldridge and Jennings,

p.388). Having a solid understanding of how to use related software will speed up the development process by getting around reinventing the wheel

Your design doesn't exploit concurrency: Implementing a concurrent system will not only allow developers to have more threads of control but also to receive many other perspectives from many different levels. Thus, it will create a more interactive world for Agents and effectively do many tasks as needed.

Micro (Agents) Level Pitfalls:

You decide you want your agent architecture: Ensuring you are using a robust architecture that is both trustable and high utility score is the one you want to work with.

You think your architecture is generic: Its architect model can be used to resolve other than your issues, It is a bad temptation, need to avoid it to escape generalization.

Your agents use too much AI or too little AI because: Let's say you have don't proper knowledge in building AI equipment.

Macro(Society)level pitfalls :

"A common misconception is that agent-based systems can be developed simply by throwing together several agents in a melting pot; that the system requires no real structuring and all the agents are peers." (Wooldridge and Jennings, p.389).

You spend all your time implementing infrastructure that you don't have enough time to develop the actual solution .

You confuse simulated with real parallelism: what you think is possible in simulation is not what you really can do in an uncontrolled environment. Build agents according to the constraints, be open to communication and be able to change their beliefs if needed.

Implementation Pitfalls :

The Tabula rasa: Having an Agent layer Application Program Interface for an agent is a great way to operate with legacy software.

You ignore de facto standard: At the end of the day, you want your Agent to interoperate with other agency's agents to achieve a common goal which is possible by practicing an internationally accepted standard that can resolve this issue.

Conclusion

Agent-oriented programming (AOP) marks a significant advance in software development, away from traditional object-oriented programming models and towards systems that are more dynamic and adaptive and mimic human-like intelligence. In this paper, the Belief-Desire-Intention (BDI) architecture is augmented with a commitment module to allow a better, more varied decision-making procedure for agents than just beliefs, desires, and intentions. However, with such a commitment-inducing addition agents are making more secure decisions, such as following rules, which they do not have any knowledge about. By incorporating such an additional module, agents may engage in their environment's more complicated interactions. They can respond effectively to uncertainties and interact with other agents to negotiate agreements, managing just the ramifications of the commitments.

Reference/Citations

1. Shoham Y, Robotics Laboratory, Computer Science Department, Stanford University. *Agent-Oriented Programming*. PII: 0004-3702(93)90034-9 (sciencedirectassets.com).

2. Wooldridge M, Jennings NR, Department of Electronic Engineering, Queen Mary & Westfield College, University of London. *Pitfalls of Agent-Oriented Development*. https://dl.acm.org/doi/pdf/10.1145/280765.280867.

3. Shoham Y, Robotics Laboratory, Computer Science Department, Stanford University. *Agent-Oriented Programming: An Overview of the Framework and Summary of Recent Research*. https://ntrs.nasa.gov/api/citations/19930022947/downloads/19930022947.pdf.

4.Novák P, Jamroga W, Department of Informatics, Clausthal University of Technology. *Code Patterns for Agent-Oriented Programming*.; 2009:105-112. untitled (liv.ac.uk)

5. Jennings NR, Wooldridge M, Department of Electronic Engineering, Queen Mary & Westfield College, University of London. *Agent-Oriented Software Engineering*. https://www.cs.upc.edu/~jvazquez/teaching/masd/docs/jennings00agentoriented.pdf.