Agent-based Systems

Stefan Härtel

University of Leipzig

Seminar Complex Systems

Leipzig, March 31, 2022

Contents

1 Introduction

The concept of agents aims to provide a way to model problem domains and solve them. Agents are systems themselves and can be components of larger systems. Most resources about agent-based systems can be found in the information technology and software development disciplines, that's why this article is mostly centered around material gathered from those fields. Ageent-based systems and models have many application in technology, biology, economics and physics. in simulations the agents' behaviour can be investigated.

2 Background

This article was written as a part of the research seminar "Sustainability, Environment, Management" which is part of the course "Modelling Sustainable Systems and Semantic Web" at the Universität Leipzig [Grä22]. The seminar's overall goal to explore the concept of *systems*, how they can be described, developed and managed.

3 Motivation

A system is characterized by its components and their interaction between each other. Interaction means exchanging energy, material and information [Grä21]. When talking about software systems this *interaction* is regarded as the software's most important aspect [WC01]. Building a software system means managing and engineering the interaction between its parts. This complex task is a research topic in computer science and lead to the idea of modelling the interaction between software components as *agents*.

In this article the concept of agents is being explored for understanding, modelling and exploring systems.

4 What is an agent?

An agent is an autonomous system which can interact with other agents to reach its design objective. [WC01]. This plays well together with the concept of a system's self-similarity in which its components can be seen as systems again. This is also true for agent-based systems.

There are different definitions for the concept of an "agent":

- Michael Woolridge defines agents in most of his publications as the following [WC01]: An angent is a system. It is (1) autonomous, encapsulates a state and makes decisions, there is no external intervention. Agents are (2) reactive, they are situated in an environment, perceive it through sensors and respond to changes happening in it with actuators. An agent is (3) pro-active and exercises goal-directed behaviour by "taking the initiative". Additionally it is (4) social, communication with other agents happens through an "agent-communication language". Social activities include negotiation and cooperative problem solving.

- The Friend of a Friend ontology *FOAF* defines an agent class [foa14]: Agents are "things that do stuff". There are definitions for the agent subclasses "Person", "Organization" and "Group".

- The Dublin Core ontology defines an agent class [dc:20]: An agent is a "resource that acts or has the power to act."

- The website "Academic Dictionaries and Encyclopedias" has a definition in its law dictionary section [aca]: An agent is "someone or something that acts or exerts power: a moving force in achieving some result".

The very basic definition for *agent* is "something that does stuff or acts". This is a very broad view for a term. According to the FOAF ontology agents can be persons or organizations [foa14].

The FOAF ontology gives a short explanation for when to use "agent" over "person": The term "person" might be too specific, when properties associated with non-humans are used to describe things - for example software bots - the term "agent" might be more fitting [foa14].

COSMO, the "Common Semantic Model" [cos21], is described as a "foundation ontology" which provides a broad vocabulary for any application or domain. It defines several RDF classes related to agents and makes interesting statements about this concept. The "GenericAgent" class is defined as "the causative subject of an action". (Generic) agents can be inanimate objects or living organisms. There is the special case of organizations which can be seen as agents too: They are not physical objects and therefore cannot directly change the physical world but organizations are talked about as if they perform actions themselves. This indicates that organizations consist of one or more physical components – agents – which perform actions. The class "IntentionalAgent" is described as a type of agent which "has the capacity to form a plan of action", some purpose might be assigned to it. A computer program (a software agent) can be categorized as IntentionalAgent if it possesses a rather high degree of autonomy. "Autonomy" in this case means being "able to reason about situations and form plans that were not directly preprogrammed".

The COSMO ontology [cos21] references the "OpenCyc" [1] ontology [ope12], in which generic agents are described as things "that have desires and intentions and the presumed ability to act on them".

An additional external reference to the "SUMO" (Suggested Upper Merged Ontology) [sum22] outlines a "CognitiveAgent" as an equivalent to the aforementioned "IntentionalAgent", it "may or may not have responsibilities and the ability to reason".

The referenced "DOLCE" [dol] ontology gives agents the following properties: "They internally represent descriptions, and in particular plans, goals and possible actions, but they do not necessarily act"; the word "agent" is used to "tell

[1]Access to the most recent version of OpenCyc seems not possible, OpenCyc seems abandoned, an older version can be found online.

that something has acted in a certain way" or to "say that sinething has an initiator or leading rile in some action".

The fourth reference in the COSMO ontology is to FIBO (Financial Industry Business Ontology) [fib21]. In FIBO an "(autonomous) agent" is defined as "any entity which is able to act on its own part and embraces all such things, including people, animals, software agents organizations and all forms of legal persons".

FIBO's definition goes more into detail regarding software agents, it is said that agents in IT systems are only useful if they have the following three properties:

- *Autonomy:* An agent acts without direct external intervention. It has at least some control over its internal state and acts based on its experience. It has internal responsibilities and processing which enables acting without external choreography. An agent embodies the goals of an external person or thing if it acts as a proxy on behalf of this person or thing.
- *Interactivity:* An agent communicates with other agents and the environment. They can exchange messages with other entities which are part of their environment. Messages may have different uses, for example request services and resources or be used as event detection and notification. Interaction can also be more complex for negotiating contracts or making bids in a marketplace.
- *Adaptivity:* An agent responds to its environment or other agents. Responses are made appropiately to incoming messages and events. Agents can be specifically designed to make difficult decisions and modify their own behaviour based on their experience. They are capable of learning and evolving.

It is worth mentioning that in most ontologies agents are a very high-level concept which is used to describe more specific real-world concepts relating to life-forms and non-life-forms which have the qualities and features referenced in this article. This includes for example humans, animals, plants, organizational forms and computer programs.

The article [WC01] gives more insight about an agent's properties:

- *Pro-activeness* is goal-directed behaviour. A goal is achived when an intended action is performed. Actions rely on *assumptions* before it is performed and their result is an *effect* after their execution. Such kind of action can be seen as a function with a pre-condition and a post-condition.
- Functions do not work when pre-conditions change, this might break functions or lead to errors when the function is executed. Additionally changing pre-conditions can lead to goals becoming invalid which means functions can become unnecessary if they are not part of the process of reaching a goal. As a result an agent must be *reactive*, this means that changing pre-conditions (a changing environment) must trigger a modification of an agent's goals or inner functions so it keeps functioning.
- An agent's reactiveness must not overshadow its goal-directed behaviour. Being reactive by constantly monitoring the environment and adapting

to it does not allow performing actions for reaching the intended goal. This means some kind of *balance* must be achieved between performing goal-directed actions and adapting to new pre-conditions.

- Agents are *social.* While exchanging information is a basic social act a real meaning is given to it when this exchange is goal-driven. Agents exchange information to achieve their own goals, they *cooperate.* This leads to agents being able to actively negotiate and coordinate. In a agent-based system each agent following personal goals results in reaching the overall system's goal.

In the context of software systems most of those properties are difficult to realize. Most literature about software-based agents can be found in the infomation technology branch of artificial intelligence.

5 Agents Versus Objects

Since the agent concept can be difficult to understand the article [WC01] describes what makes agents different from intuitively more understandable *objects*: Objects are computational entities which encapsulate a state and perform actions on this state, they communicate by passing messages between them. When talking about objects there is a tendency to anthropomorphisize them, to give them additional abilities so an object might *decide* to do somethig or *ask* a different object about something.

Both agents and objects communicate and are autonomous, they control their inner state, but objects do not control their behaviour, it cannot be modified by themselves whereas an agent is able to adapt to changes in its environment.

An additional difference is that in a object-based system which is designed to achieve a goal all its parts i.e. objects also share this goal. In contrast to this a (heterogenic) multi-agent system's overall goal is not necessarily shared by its parts. For agents a goal is an intrinsic value which drives its desire to perform actions for achieving their goal. Each agent might have a different goal. This means that when an agent requests an action to be performed by a different agent, the latter might ignore this request because the action is not in his interest.

As [WC01] points out objects lack some features which are important for agents: Objects are not reactive, pro-active and social. Of course those abilities could be programmed into a object-oriented software but generelly the abilities are not part of the basic object-oriented programming model.

Another distinction between agents and objects is that the latter do not control their own execution. They are only active when an action is requested by a different object, they are triggered. Agents are always active. Since they continuously monitor their environment and react to it while updating their internal state, they can perform actions without an explicit external trigger [WC01].

A very illustrating statement in [WC01] to summarize the basic difference between the two entity types is: *"Objects do it for free; agents do it because they* *want to."* An object *is essentially an agent* which cannot exhibit flexible and autonomous behaviour.

It is clear that the act of *wanting* is difficult to implement in technical systems and software which behave deterministically.

6 Agent-Based Systems

A system is defined as "agent-based" if for its description or design the key abstraction concept of an "agent" is being used [WC01]. Such systems can contain a single agent, for example a continuously running program that labels e-mails according to their subjects or senders. Designing a system to use multiple agents opens up new possibilities for effective problem solving.

Why does it make sense to design or analyze a (software) system with the help of the agent concept? The article [WC01] gives some ideas. Depending on the observed problem domain in which a system exists its parts might be seen as rather passive but still interacting *objects* or active and purposeful *agents*. For software systems it is difficult to find a single point which takes control of the whole system. In complex systems control is distributed across different programs running on a number of geographically distributed computers. Autonomous interaction between those computers is necessary so they migth be modelled as agents. Some systems might be very flexible and open so its components change during their execution. For their effective operation it is crucial to decide autonomously which components shall exist.

7 Designing and Analyzing Agent-Based Systems

The article [WC01] references different methologies to talk about, design and explore agent-based systems.

The *AAII methodology* defines an *internal* and *external* system model. The system's components are agents, the system-level *external* view describes the relationships between those agents. This external model consists of a class hierarchy to describe agents, their interactions and instances. Agents are explicitly gien attributes for beliefs, desires and intentions. In contrast to this the *internal* view is concerned with the inner workings of each agent. The AAII Methodology aims to identify roles, responsibilities and goals in a agent-based system.

The *Gaia methodology* proposes a blueprint for desiging agent-based system. A problem's requirements are translated into concepts which in turn allow implementing a system matching the requirements. In this process a set of abstract concepts is used to describe the system: Roles, permission, responsibilities, protocols, activities. From this description concrete agent implementations are derived.

The *Cassiopeia methology* is a *bottom-up* approach which concentrates on identifying the necessary *behaviours* to achieve a goal. The three proposed steps are the identification of elementary behaviours to reach the altogether system task, the identification of the relationship between those behaviours, and the identification of how the behaviours are organized in the system.

The *Z language* is an agent spcification framework. Here a set of different kinds of agents is used to model a system: *Entities* are simple objects without functionality, they only hold attributes like position or color. *Objects* are like entities with the addition of (primitive) functionality. *Agents* in turn are objects which possess goals and try to achieve them actively. Finally *autonomous agents* are agents with motivations. By using the entities defined above relationships between the different types can be modelled and implemented. The hierarchy translates well to object-oriented frameworks.

[WC01] gives an introduction to the specification of agent-based systems. A system specification allows the implementation of said system. The specification is made with the help of an agent specification framework. Different frameworks and approaches exist. The predominant way to specify agents is to give them human-like *mental states.* Specifically those are beliefs, desires and intentions. This is called *Belief–desire–intention model* or simply *BDI*, the page [wbd22] gives an overview about implementations using this model. Beliefs are the information the agents have about their environment which can be wrong or incomplete. Desires are the objectives and agent wants to accomplish. In relation to this a goal is a desire with a special focus and might be an amalgation of multiple desires. Other desires might be of secondary nature. Intentions are the actions an agent has decided to perform to fulfill its desires.

8 Agents In Businesses

Processes in businesses organizations suggest an implementation as agent-bases systems [J$^+$96] with the support of information technology.

For making decisions managers rely on a plethora of information sources. The information gathering and concentration can be improved with the help of computer systems. There is a list of requirements for such a system: The information provided should be available in the whole organization. Information should be requestable at any time from internal departmens or external organizations. Information from departments should be delivered even if the manager did not explicitly ask for it because its existence was not known. The manager should be informed by the system if important information changes which affects the decision making context.

Businesses and their processes possess properties which are suited for a translation to agent-based systems: Multiple organizations are involved in the processes. Each follows its own goals and agenda with the primary goal of maximizing profits. Organizations are physically distributed, this is epecially true too for very large businesses. Tasks inside the organization are decentralized, the informations and resources to perform them is also distributed. There are different groups in organizations which act rather autonomously. Most tasks are executed concurrently. The overall business process must be monitored and managed; although control and resources are not centralized there is a need to centrally guide the organization or put constraints on the process. While the general processes in an organization can be clearly defined there is still a need to deal with uncertainty, unaniticipated events and delays happen.

Given the organization's properties above it makes sense to explore the inner workings of an organization with the help of agents: They are autonomous and work without external intervention. Agents are social and interact with other agents to achieve their own problem solving with the idea to reach an overall goal. They are pro-active and respond to changes in their environment.

An organization's domain involves distributed responsibilities and and problem solving skills which matches the concept of agents. The given organization's structure consitsing of autonomous parts suits the idea of agents being autonomous. Since agents are social and have the ability to communicate, negotiate and coordinate they fit the real-world processes in an organization. Just like organizations having to adapt to unforeseen events agents react to their environment.

For applying the agent concept to business organizations the article [J$^+$96] introduces the concept of *services*: "A service corresponds to some unit of problem solving activity." Services allow performing tasks and which in turn produce a result. Those rather atomic tasks can run in parallel, multiple services can be nested and become complex. Taking all services together results in the overall business process. Each service is associated with one or more agents. When a complex service is performed multiple agents can be involved. The agents act autonomously and must communicate with each other to request services, the execution of a service is not simply "instructed" or "ordered", it is *asked* in a social sense. This includes that an agent might deny a service's execution because internal dependencies are not met or it opposes the internal goal. The negotiation between agents for requesting and executing services between agents is called *service level agreement.* Negotiation and communication happens through a defined protocol for message exchanges. On a higher level there needs to be a mutual understaning about terms used in the agent's domain. An organization's parts use different terms to describe things which means that semantic mappings must be performed to enable a meaningful communication. Here it makes sense to create an ontology all agents can access or share.

Classical management theories focus on hierarchical organizational structures in which managers with authority make decisions. The manager acts as a leader and communicates decisions downward to employees which follow them. The manager, as a single individual, bases his decision making on a plethora of information from marketing, sales, research, development, manufacturing, finance and other departments. This is a complex task and might be overwhelming for larger and complex goals and thus not feasible [J$^+$96].

Human-based organizations could be modelled with agents. The agents are realized by single persons or groups of persons, it is appropriate to call those agents “teams”. Each team provides a specialized service so there might be a marketing team, a sales team, a manufacturing team and so on. Each team acts autonomously and tries to reach its own goal. To fulfill goals interaction between teams is necessary, for example the sales team must reach out to the manufacturing team for the supply of products to sell. The interaction between teams is enabled by a common interaction protocol. There is an agreement between the teams on how interaction is implemented, for example communication happens through a ticket system.

The agent-based business system is self-organizing, each team tries to reach their own goals which in turn allows the whole business to reach its goal. The need for a centralized top-down management is eliminated.

In theory a human agent-based could work but it is to be expected that the actual persons do not behave in the sense of actual agents. They might ”leave” the boundaries of their own agents or fully change their agents original purpose and undermining the company system's inner workings and goals.

The article [J$^+$96] proposes a specific architecture for designing an agent-based system in a business environment called ”ADEPT”.

All agents have the basic architecture consisting of different modules with special purposes.

An agent's resources are *tasks* or access to *sub-agents*, they are embedded in the agent's *agency.* Agents on this same level are called *peers.*

The use of sub-agents allows to create a hierarchy of agents for performing more complex services. In this case it is stated that when an agent requests services from sub-agents those sub-agents are not allowed to refuse the service request. A top-level agent cannot access the sub-agents of peer agent.

Top-level agents have a *communication* module for talking to peers and sub-agents. It routes messages to a service execution module and the agent's tasks and is concerned with service initiation requests.

The *interaction management module* handles service requests and their execution. It activates a situation assessment module which takes care of gathering additional external services the agent might need for performing its own services or tasks. The interaction management module has the power to propose and negotiate service interaction with other agents. For this the module has access to knowledge about itself, its own domain and peer agents.

The ability of an agent to privide services is monitored by a *situation assessment module.* For this it keeps record of the agent's resources. Additionally it handles service exceptions and can reschedule or terminate running service executions.

The *service execution module* manages an agent's executed services. Specifically it starts services and routes information between services, tasks and agents. Service exceptions are routed to the situation assessment module which responds with the necessary action.

Each agent has *Acquaintance Models* which is a list of peer agent and their provided services.

The agent's *self model* stores its own service level agreements which are offered to other peers.

This proposed model allows designing an agent's service lifecycles. Services are described in a *service description language.*

The agent's provided services are linked together by the ability to negotiate dependencies between agents for fulfilling those services. This is described in a specific *negotiation model.*

For the negotion model and communication to work a basic language and protocol is defined which all agents use.

The description of the architecture of an agent shows that it is rather uncomplicated to model a system with agents. In contrast to this designing how an agent works and translating it to a functioning program is not a trivial task.

9 Advanced Topics

While the concept of agents is understood the research around it goes a lot deeper.

Some questions to investigate are *how* the transition from an abstract model or specification of an agent-bases system to an actual software implementation is made. In [WC01] it is often talked about using object-oriented approches but at which point does a derterministic program become an agent with properties like beliefs and desires and an adaptive behaviour?

The article [J^+96] talks about agents which interact with each other by providing services. What is the advantage of this concept in contrast to a classical (micro-) service-based architecture? It is suspected that at this point because of talking in a software context the line between "service" and "agent" becomes blurry.

Agents adapt to their environment. To which degree can this be allowed? Are intense changes allowed to change an agent's purpose or a system's overall goal? Can agents "evolve"?

Most basic research around agents seems to have happened around 2000. What role do agents play now in research and real-world applications?

Agents are thought of to be somewhat "intelligent". What impact does the emergence of artificial intelligence have on the concept?

Agents have a very wide area of application. What specific uses do they have and are they better that other models in the respective research fields?

10 Seminar remarks

During the presentation of the topic "Agent-based business systems" some questions emerged which shall be answered in this section.

Agents and agent-based systems are designed to serve a purpose, they have an external goal. In reference to [WC01] in which agents are said to *want* to do things, they try to achieve inner goals. Those inner goals do not necessarily match the external one. Agents have an internal state, beliefs, desires and intentions. Fulfilling the desires and intentions means reaching internal goals with the result of achieving the external ones. As an example an external goal is "build a house". This goal requires multiple steps performed by multiple agents which each have their own internal goals which are not exposed to other agents.

The article [WC01] gives an overview about how agents can be modelled and specified. The most important statement is that agents are autonomous, proactive, responsive and social. The concept of agents in system design and software development exists to deal with complexity. It is a different kind of abstraction for solving problems, just like object-orientation is an alternative. There are benefits of using agents for modelling systems. Some domains can be better described with the help of agents than through simple objects. Modern IT systems are distributed, their components must communicate with each other. Ideally this communication should be effective and goal-oriented with the option to negotiate and cooperate. Agents are a feasible metaphor for this. Since technical agents are given human-like (mental) capabilities working with them should be rather intuitive. The transition from an agent-based specification to a working implementation can be problematic since programming beliefs, desires and intentions into a software is not trivial.

When talking about agents the term *goal* is the main focus. A goal justifies an agent's existence, it is basically the one thing it *wants*. An agent's purpose is to achieve a goal. This means that the pourpose is something that is viewed from the outside, goals are what drives the agent's internals. In [WC01] the term *objective* is used on a higher level: An agent is designed to satisfy an objective. This means the objective is not a part of the agent-based system but rather something that the creator of the system wants to fulfill.

An agent serves a purpose, it tries to achieve an ermergent function. Agents themselves are systems: They have components like states, desires and beliefs. When composing a more complex system from single agents, they become components in this system. Using agents is very feasible for engineering a system.

In traditional software engineering design, modelling and specification precede the implementation of a system. This is not necessarily true for building agent-based software. In [WC01] multiple approches are given. Some try to dissect a problem domain into rough abstract concept (e.g. a role) wich are refined into smaller concepts (e.g. responsibilities, activities). The agent concept allows understanding domains on a high level.

In general it can be assumed that the agents of an agent-based system exist in the same environment. This environment allows agents to communicate, negotiate und share information. For a real-world IT system this environment could be a network which enables the software agents running on computers to communicate. The connections should be rather dynamic, each agent should be able to exchange infomation with any other agent. Of course this is very idealistic. In real-world applications some communication ways might be limited because of technical boundaries or the intended design. With the thought of ideal agents being social and self-organizing, any communication should be made possible.

A business environment and its actors can be modelled as agents. This might work well when talking about businesses on a level of companies and organizations. Organizations are well-defined and do not change from an external view, no matter the changes in the internal structure or the staff. It makes sense to model the actors in a value network as agents. Companies do have inner goals, beliefs and intentions with an external goal of making a profit. Business organizations match the definition of agents: They are social, pro-active, reactive, adaptive and autonomous. Businesses reach out to other businesses to achieve personal goals, for example requesting services or resources for internal production. When viewing an agent-based system consisting of businesses, it might be possible that there is no top-level overall goal, each business only tries to achieve personal optimization. Concepts like a joint venture or an alliance possess a greater goal.

References

[aca] Academic dictionaries and encyclopedias. `https://law.en-academic.com/157/agent`. [online; accessed 03.01.2022].

[cos21] Cosmo: Common semantic model. `http://www.micra.com/COSMO/`, 2021. [online; accessed 11.01.2022].

[dc:20] Dublin core metadata initiative metadata terms. `https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms/Agent`, 2020. [online; accessed 02.01.2022].

[dol] Dolce: Descriptive ontology for linguistic and cognitive engineering. `http://www.loa.istc.cnr.it/dolce/overview.html`. [online; accessed 13.01.2022].

[fib21] Fibo: Financial industry business ontology. `https://github.com/edmcouncil/fibo`, 2021. [online; accessed 13.01.2022].

[foa14] Foaf vocabulary specification 0.99. `http://xmlns.com/foaf/spec/#term_Agent`, 2014. [online; accessed 29.12.2021].

[Grä21] Hans-Gert Gräbe. Systems and their development, 2021.

[Grä22] Hans-Gert Gräbe. Modelling sustainable systems and semantic web. `https://github.com/wumm-project/Seminar-W21`, 2022. [online; accessed 30.01.2022].

[J$^+$96] N. R. Jennings et al. Agent-based business process management. 1996.

[ope12] OpenCyc. `https://github.com/asanchez75/opencyc`, 2012. [online; accessed 11.01.2022].

[sum22] Sumo: Suggested upper merged ontology. `http://www.ontologyportal.org`, 2022. [online; accessed 11.01.2022].

[wbd22] Belief–desire–intention software model. `https://en.wikipedia.org/wiki/Belief%E2%80%93desire%E2%80%93intention_software_model`, 2022. [online; accessed 10.03.2022].

[WC01] Michael Wooldridge and Paolo Ciancarini. Agent-oriented software engineering: The state of the art. pages 1–28. Springer-Verlag, 2001.