Resolving limits of organic systems in large scale environments: Evaluate benefits of holonic systems over classical approaches

Claudio Schmidt
University of Passau
schmidcl@fim.uni-passau.de

***Abstract*—With the rapidly increasing number of devices and application components interacting with each other within larger complex systems, classical system hierarchies increasingly hit their limit when it comes to highly scalable and possibly fluctual organic systems. The holonic approach for self-* systems states to solve some of these problems. In this paper, limits of different state-of-the-art technologies and possible solutions to those will be identified and ranked for *scalability*, *privacy*, *reliability* and *performance under fluctuating conditions*. Subsequently, the idea and structure of holonic systems will be outlined, and how to utilize the previously described solutions combined in a holonic environment to resolve those limits. Furthermore, they will be classified in the context of current multi-agent-systems (MAS). The focus of this work is located in the area of smart energy grids and similar structures, however an outlook sketches a few further application scenarios for holonic structures.**

CONTENTS

I. INTRODUCTION

The approach of holonic systems or architectures is not new, in fact the idea of holons has been around for more than 50 years [9]. The name refers to the Greek word "όλος", which means *all*, *whole* [1].

Holonic systems are part of a goal-oriented approach, in which a single node holds two different roles: On the one hand, it is part of a complex system and its hierarchy, being one independent node upon many others offering a set of input goals - definitions of some value or state that should be achieved - to the system it is part of. On the other hand, it acts as a controller encapsulating another complex system of any kind, structure or hierarchy, which is used to propagate, split, and, finally, achieve the defined goals of the system. This refers to the famous *divide-and-conquer principle* which means that splitting up things and handling them separately is much easier than dealing with the problem as a whole ("However, as every parent of a small child knows, converting a large object into small fragments is considerably easier than the reverse process." [13]). Within a holonic system, a single holon recursively consists of multiple sub-holons, that may contain holons themselves. The actual inside implementation of a holonic agent is completely private to the outside. Because of the strict shielding, different architectures can be used for every holon, giving them the possibility to pick the best one suitable for every scenario without any interference with other agents of the system. Given the context of intelligent, dynamic, and self-* Multi-Agent-Systems (MAS) being on the march right now, but still facing some limits regarding scalability, compatibility, privacy or performance, especially in highly fluctuating environments, the demand for a generic and scalable standard is constantly growing. Holonic systems sound very promising, since state-of-the-art architectures (and possibly already existing systems) can be combined to MAS without larger scalability or compatibility problems.

II. LITERATURE REVIEW

The theoretic foundation for holonic architectures is quite clear and prepared by now. Yet, they are still hard to develop,

[1] https://de.pons.com/

like all other MAS, since they are complex self-optimizing systems interacting with each other. The current literature mostly deals with planning problems, "as the coordination of individual agents' planning processes is a hard problem in systems that allow concurrent action between autonomous, rational agents " [7]. One approach to this solution is `MA-STRIPS`, an extension of the `STRIPS` planning language [2] [1]. As well, the development and programming process has been getting pretty much attention over the last years. For example, there are `PASSI` [3] and the adaptation to holonic environments, `HoloPASSI` [4], `MetaMorph` [12], the `SARL Agent-Oriented Programming Language` [2] and the `Janus-Platform` [3]. Other current topics are privacy concerns between agents [1] and self-repairing MAS [10]. "While such issues are well studied in classical planning, the presence of multiple agents makes some known techniques unsuitable" [1].

III. Classification in MAS (Multi-Agent-Systems)

The definition of MAS is very broad, describing multiple agents interacting with each other, either using a central control unit to coordinate them or giving the nodes full control over their own decisions. A system with self-regulating agents is much more complex, because every single node can have different personal goals, possibly even conflicting with each other. So a superior conflict resolution unit (CRU) is necessary that can sort and prioritise conflicting goals. This is only possible if the CRU has a global goal itself, e.g. fairness, performance or system goals like temperature stability of a room. Let's take such a system, defining global goals (e.g. temperature stability) that has three agents: a heating, an air conditioner and a central power unit. The heaters declared goal is to warm up the room, while the air conditioner tries to cool it down. The central power unit wants to minimize the energy consumption, thus holding back the other two. All three goals are conflicting with each other, therefore the CRU has to set a global room temperature and prioritise the agents accordingly to achieve that. This system could be part of another MAS controlling comfort in a house, and this is already the basic idea of holons. Holonic systems are part of a goal-oriented approach, where agents only exist to perform goals, which can, once triggered, subsequently call other goals implemented by the same or other agents.

IV. Challenges for MAS

A. Scalability

One of the most common problems in larger systems is scalability. When having a large number of agents and each of them holding different goals, it becomes impossible to manage them using a central control institution. One possible solution is the *Stigmergy Pattern*, which "describes the way non-rational, autonomous agents (such as termites or ants) collaborate to achieve complex tasks thereby displaying some type of emergent swarm-intelligence" [11] This means, that there is no superior instance monitoring or controlling all agents. Instead, every agent communicates with their direct neighbours (peer-to-peer), making the system extremely scalable while increasing stability (a lot of nodes must fail to impair the overall functionality). However, as all agents are similar to identical, a systematic programming fault could cause all agents to fail at once. The privacy is relatively moderate, as information is scattered over all participants and thus impossible to bundle for one agent, although it is assembled by the supervising structure, if any is in place. Nevertheless, it is hard to gain emergence of the system and reaching a common goal, that has been pre set or may vary during runtime, as actions of a single agent are not directly linked to the emerging system result, which makes the programming of single nodes difficult. The fluctuation performance is rather good, as one single agent has very little logic and thus its meaning to the whole system is limited.

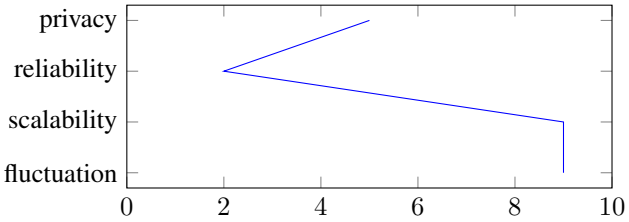

Figure 1. Privacy, reliability, scalability and fluctuation performance representation for the *Stigmergy pattern* on a scale from 0 to 10. 0 means poor, 10 represents excellent performance in an area.

B. Privacy

Whenever systems need to interact with each other, they need to exchange data. That is where privacy becomes an issue. Often, the agents hold information that is either personal or safety-critical and thus must be kept secret from other nodes. Therefore, an agent must classify held information by sensitivity and trade off between the importance of communication and safety. In addition, "the definition of privacy in multi-agent planning is debated, e.g., what agents should kept private information (state variables, actions, goals) and what minimal information they should exchange in order to be able to construct a joint plan remain an open question." [1]. One approach to this topic are *Trust-based or Reputation-based patterns*. In those models, mostly used in peer-to-peer architectures where the agents are unrelated and unknown to each other and privacy plays an important role during interaction, the amount of information shared or the permission to certain roles depends on an agents *trustworthiness* or *reputation* [16]. Reputation can be gained by good performance and respecting the system's rules, or by being vouched for by other trustworthy agents. Given the context of holonic MAS (HMAS), the sub-holons utilized to implement the defined goals could be selected based on their reputation, along with other metrics like performance, value to the system, load, etc. while access control and role distribution could be implemented by trusted entities vouching for each other, as described in [8]. The scalability and thus stability of peer-to-peer networks is very

[2]http://www.sarl.io/docs/official/index.html
[3]http://www.sarl.io/runtime/janus/

high, while the reliability depends on how good agents work together and could be unbalanced by malicious participants. When exposed to high agent fluctuation, the performance will suffer badly, since it takes time to build trust in the agents - and constant joining or leaving of nodes will prevent that.

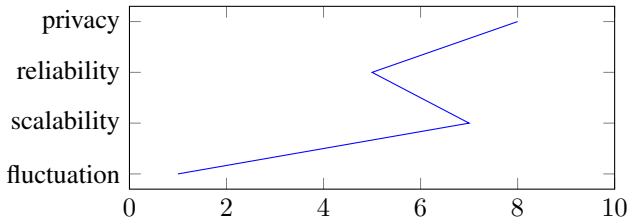

Figure 2. Privacy, reliability, scalability and fluctuation performance representation for the *Trust- and Reputation-based pattern* on a scale from 0 to 10. 0 means poor, 10 represents excellent performance in an area.

C. Fluctuation Performance

Another problem of MAS is their performance under exposure to highly fluctual environments. In a system that manages for example power distribution for electric car charging stations, cars (agents) will be joining and leaving the system constantly, just like a house manager that has to cope with household devices being (un)-plugged constantly. Therefore, the integration of new agents must be (1) possible and (2) extremely time-efficient prevent pulling down the overall performance when adoptions to its structure occur. This requires constant monitoring and optimizing "on a continual basis to accommodate fluctuations in demand" [15], which introduces another bottleneck in the system's structure. Furthermore, every agent joining or leaving possibly triggers a rearrangement in the system or at least some integration effort, which gets costly on larger scales.

V. Description of Holonic Systems

A. Vision & Structure

The efforts for holonic structures are basically driven by practical thoughts: In a theoretical, empty and static environment, an optimal structure could be implemented - in reality, there are already existing structures and systems, that are constantly developed and maintained. On top, smart systems and requirements are constantly growing and developing. Therefore, low coupling between systems is necessary to meet requirements like *scalability*, *reliability* or *performance under fluctuation* and to integrate heterogeneous architectures. The holonic approach defines *goals* for every system, which are tasks it can do or achieve (e.g. "heat up"or "multiply") [5]. If a goal is triggered, the systems tries resolve the goal and break it down into minor goals (e.g. *multiply* $\rightarrow$ *adding*) and either solve primitive goals directly or pass them to sub-systems (*divide and conquer*). A holonic system needs to be able to self-optimize, this means:

- integrating new agents (or sub-holons) on demand, if rational
- continuously optimizing goal resolution functions
- join or leave supra-holons at any point, if sensible
- adapt conflict resolution rules, if pertinent
- be part of multiple supra-holons, if necessary and possible.

This makes them perfectly suitable for smart home applications with high fluctuations (houses consuming and prosuming, varying energy supply) and lots of heterogeneous participants (e.g. power plants, consumers, private prosumers, batteries). However, it is possible that an agent receives multiple instructions that are conflicting. Thus, every holon needs a conflict resolution unit that resolves and prioritises goals and achieves fair results, where possible. On top, in there might occur economical concerns and negotiations, in this scenario about energy prices. A smart house being part of a multigrid will try to absorb as much power as can be stored for a low price and sell spare power when the prices rise. This makes the system state unpredictable, as the price and supply can change within few moments.

B. How challenges for MAS can be issued

1) Scalability: Holonic systems are highly scalable, only limited by the architecture and scale in one single layer. Because of the system encapsulation, the layer depth is (theoretically) unlimited, as all layers beneath are hidden from the next higher level. Although, due to goal resolution and relaying, the performance will suffer when having too many layers between the initial goal and the final goal execution. In self-optimizing systems, this should be automatically detected so that lower holons could step up layers and reduce the amount of relays in between. Per layer, the count of agents is limited by the common holon state, which is provided and managed by the administrating interface. This state needs to be distributed to all participants of the system and is required to be in a consistent state all the time. If too many agents need to synchronize the current state, it would slow down the whole system, again leading to a rearrangement of the holons.

2) Conflict resolution: Every holon has a central conflict resolution function that can prioritise and evaluate conflicting goals. Let's imagine the following scenario: A smart house is configured to start the washing machine at 10pm. At that exact time, the energy price is surprisingly high, so that the houses power manager triggers the goal *energy reduction*. So there are two conflicting goals, *starting the washing machine* and *energy reduction*. The houses CRU has to evaluate the context and make a decision, weather to delay the wash or to use expensive energy. The CRU can be extracted and is independent of the remaining holon implementation or the used architecture. The only requirement is a central container or *membrane* [5] around the holon, that, used in peer-to-peer networks, would have a major impact on performance limits but would still work.

3) Privacy: Due to the membrane surrounding a holon and its encapsulation, private data can easily be kept inside, goal definitions is the only information being exposed

to surrounding agents. This isolation can be weakened, depending on the business case and with security concerns in mind. “Finally, a holonic structure may also help with self-protection and privacy concerns“ [6], as “the state information a holon provides to its supra-organisation can depend on the business context (e.g. collaboration or competition) and may even change during runtime (e.g. threat detection)“ [6]. That means, the holon’s context-awareness capabilities enables it to adapt privacy rules at runtime.

4) Reliability: The concept of holons is that they can join and leave a supra-holon any time or even exist solely in an isolated environment. This constellation makes the system very durable. If the holon membrane fails or the systems collapses, the holons can distribute themselves to other surrounding systems instead of being wrecked. If one sub-holon fails to fulfil his goals, it can be exchanged for another one. If no other node can fulfil its functionality, the goals must be updated eventually causing the system to endure the loss of crucial functionality, eventually causing the system to fail. However, there is some redundancy in comparison to “classical“ MAS, as multiple implementations can exist independently from another. This makes a loosely coupled HMAS much more reliable than hierarchies, while still being more sensitive than peer-to-peer networks.

5) Compatibility: HMAS are highly compatible with other structures and architectures. Since it would be way too costly to reimplement already existing systems or those being under development to adapt them to some common architecture and standard (“The nice thing about standards is that you have so many to choose from“ [14]), the demand for compatible solutions that are easy to adapt to is constantly growing. The holonic approach is a relatively generic one, the only requirement is the definition of goals for the outside, the concrete implementation does not matter, it doesn’t even have to be holonic. A primitive agent without many dependencies that can perform simple goals can be non-intelligent or non-holonic and still be integrated in an existing holon nearly trouble-free.

VI. Flaws of Holonic Systems

A. Supervising instance

Every holon needs some kind of supervising instance, a *membrane* to

- manage the current state and keep it consistent
- offer the goals achievable by the holon
- manage goal implementation(s)
- resolve conflicting goals.

This introduces a new entity and thereby a new single point of failure, that - if it failed - would destroy the whole holon, or at least make it unusable for others. On the other side, this also limits scalability, as this entity can hardly be parallelized. Too many state changes - that need to be distributed to every agent in the system, constant goal adaptations or lots of conflicting goals could cause an overload and thus limit the system’s overall performance. On top of that, every goal defined by the holon passes that entity and must be split into sub-goals that fit the available agent’s goals (divide and conquer).

B. Privacy concerns

The openness of holonic systems also makes them vulnerable to security threats. Since agents can discover and join/leave other unknown and possibly malicious holons, they have to protect themselves against other harmful agents. In some areas, this could be compensated by trust-based implementations, which doesn’t work properly for highly fluctuating environments. Furthermore, data given away to goal implementations of sub-holons can not be traced. This means that either, data given away to unknown sub-holons must be completely pseudonymized and harmless from privacy-perspective, or the supra-holon can only trust certain sub-holons, which limits the advantages of such openly designed systems.

C. Recursion

Special care must be taken when adapting goals during runtime. Let’s imagine the holons A1 and B1 being part of the supra-holons A and B, with A1 defining the goal *addition*. B1 discovers A1 and its *addition* goal and incorporates the goal in its interface (now A1 and B1 offer *addition*). This would be a valid action, because B1 is contributing to discovering better and faster goal implementations for B. Now, A1 discovers B1 and recognizes its *addition* goal. It could adapt the goal implementation and use B1’s *addition* instead of the old one. This means, that A1 and B1 both offer the goal *addition* while depending on each other. Once the goal is triggered, an infinite loop would occur until A1 or B1 decide to change the goal implementation or to remove the goal from its interface. It would need more communication before taking over goals to avoid bad constellations like loops as well as an efficient goal adaptation algorithm to allow quick changes, measure performance accordingly and initiate immediate roll-backs, if the new implementation does not meet performance requirements.

VII. Outlook

Current HMAS research mostly focuses on their use in electricity microgrids, for example to connect appliance-level, house-level and district-level goals [4]. But possible applications reach far beyond that. Systems with deep multi-level architectures are perfectly suitable for those kinds of systems. This counts for every application in the electricity supply system, but is also imaginable for e.g.

- logistic systems, to coordinate good distribution on different layers, from the top down to warehouses and the means of transport
- manufacturing systems, to manage the production steps top down from the end result to every single part and those parts
- house management, to monitor the comfort and control corresponding house appliances to reach abstract goals set by the resident

- large-scale calculations, to split up the formula and calculate every part in a specialized holon for that task, while also being able to optimize by switching implementations

and many more. The performance benefit highly depends on the actual environment and goal implementations, but the structure itself is broad enough to support lots of different scenarios, even though some implementation details are not quite clear yet, which makes further research necessary, especially in the contexts of software engineering and development.

VIII. Conclusion

HMAS offer a rich variety of application scenarios, while being very generic and compatible to state-of-the-art technologies. Furthermore, their clear structures are close to real-world scenarios, which makes them easy to understand, which is, particularly in the development process of highly complex, interlaced, dynamic and intelligent systems, an important thing to note. Regardless of the high potential of such systems, they are still hard to develop. At the current point researches propose a lot of different, potentially promising approaches, yet there is no common standard for the structure of those systems. Furthermore, a lot more research will be necessary to clarify software engineering processes and develop frameworks to make multi-agent-systems in general applicable to the broad majority of companies and developers. Since software development strategy is always a rather sluggish topic, it will probably take some more years of research until HMAS are well-known and initial effort is completely done and usable.

References

[1] A. Bonisoli, "Distributed and multi-agent planning: Challenges and open issues," 01 2013.

[2] R. Brafman and C. Domshlak, "From one to many: planning for loosely coupled multiagent systems." in *Proceedings of ICAPS*, 2008, pp. 28–35.

[3] M. Cossentino, "From requirements to code with the passi methodology," *Agent-Oriented Methodologies*, 04 2012.

[4] M. Cossentino, N. Gaud, S. Galland, V. Hilaire, and A. Koukam, "A holonic metamodel for agent-oriented analysis and design," 01 2007.

[5] A. Diaconescu, S. Frey, C. Müller-Schloer, J. Pitt, and S. Tomforde, "Goal-oriented holonics for complex system (self-)integration: Concepts and case studies," 2016.

[6] S. Frey, A. Diaconescu, D. Menga, and I. Demeure, "A holonic control architecture for a heterogeneous multi-objective smart micro-grid," in *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, Sep. 2013, pp. 21–30.

[7] A. Jonsson and M. Rovatsos, "Scaling up multiagent planning: A best-response approach," in *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011*, F. Bacchus, C. Domshlak, S. Edelkamp, and M. Helmert, Eds. AAAI Press, 6 2011, pp. 114–121.

[8] L. Kagal, T. Finin, and A. Joshi, "Trust-based security in pervasive computing environments," *Computer*, vol. 34, no. 12, pp. 154–157, Dec 2001.

[9] A. Koestler, *The Ghost in the Machine*, 1967.

[10] A. Komenda, P. Novák, and M. Pěchouček, "How to repair multi-agent plans: Experimental approach." in *Proceedings of the First Workshop on Distributed and Multi-Agent Planning.*, 2013, pp. 66–74.

[11] Z. Mason, "Programming with stigmergy: Using swarms for construction," in *ICAL 2003: Proceedings of the eighth international conference on Artificial life*, 01 2002, pp. 371–374.

[12] F. Maturana, "Metamorph: an adaptive multi-agent architecture for advanced manufacturing systems," 1997.

[13] A. Tanenbaum, *Computer Networks*, 4th ed. Prentice Hall Professional Technical Reference, 2002.

[14] ——, *Computer Networks*, 2nd ed. Prentice Hall Professional Technical Reference, 2002.

[15] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White, "A multi-agent systems approach to autonomic computing," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 464–471.

[16] L. Xiong and L. Liu, "Peertrust: supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, July 2004.