Knowledge Aggregation with Subjective Logic in Multi-Agent Self-Adaptive Cyber-Physical Systems

Ana Petrovska
Technical University of Munich
Munich, Germany
petrovsk@tum.de

Sergio Quijano
Technical University of Munich
Munich, Germany
sergio.quijano@in.tum.de

Ilias Gerostathopoulos
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
i.g.gerostathopoulos@vu.nl

Alexander Pretschner
Technical University of Munich
Munich, Germany
pretschn@in.tum.de

ABSTRACT

Modern software systems, such as cyber-physical systems (CPSs), operate in complex and dynamic environments. With the continuous and unanticipated change in the operational environment, these systems are subjected to a variety of uncertainties. Self-adaptive CPSs (SACPSs) can adjust their behavior or structure at run-time as a response to the changes in their perceived environment. Namely, self-adaptation is commonly realized through a MAPE-K feedback loop incorporating newly derived knowledge obtained by the sensed data from the run-time monitoring, during the operation of decentralized SACPSs. However, to build the knowledge, the need for run-time observations' aggregation and reasoning emerges, since the observations made by the decentralized systems might be conflicting. In this paper, we propose an approach for observations aggregation and knowledge modeling in SACPSs that is domain-independent and can deal with inaccurate, partial, and conflicting observations, based on the formalisms of Subjective Logic.

KEYWORDS

subjective logic, knowledge aggregation, reasoning, self-adaptive systems, cyber-physical systems

ACM Reference Format:
Ana Petrovska, Sergio Quijano, Ilias Gerostathopoulos, and Alexander Pretschner. 2020. Knowledge Aggregation with Subjective Logic in Multi-Agent Self-Adaptive Cyber-Physical Systems. In *IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '20), October 7–8, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3387939.3391600

1 INTRODUCTION

In recent years, the fast growth of cost-effective embedded systems with continuously increasing computation power has created a solid foundation for the emergence and expansion of omnipresent Cyber-Physical Systems (CPSs) across different domains, with growing socio-economic influence. CPSs are embedded systems that are distributed, networked, and interconnected. Examples range from environmental monitoring systems to robotic fleets and self-driving cars. Their close connection to the physical world means that they are exposed to high uncertainty during their operation. Namely, modern CPSs need to be able to operate efficiently and reliably within a continually changing, uncertain, and unanticipated environment (execution context) [25, 27, 29]. The context is the relevant part of the environment for a particular system. When the system under consideration is a CPS, then the relevant objects contained in the context can be other homogeneous and heterogeneous systems, entities, and processes in the physical world or cyberspace, including humans.

A common approach to deal with run-time changes and uncertainties is to make the CPSs self-adaptive. Self-adaptation is traditionally realized by an adaptation logic based on closed feedback loop—MAPE-K, with four consecutive functions, i.e., **M**onitor, **A**nalyse, **P**lan, **E**xecute with shared **K**nowledge among all the elements of the loop [11, 12, 23]. The Knowledge component comprises models of the CPSs and the context where they are operating.

Due to the dynamicity of the CPSs and their execution context, these models cannot be created based on assumptions made at design time, but instead they need to be models at run-time [5, 6, 15, 34]. Concretely, the run-time context model should get continuously updated in response to the changes in the context, reflecting the actual operational context during the execution of the CPSs. To update the model—and accoringly the Knowledge component—the need for run-time information aggregation and reasoning, of what each CPSs individually observes, emerges.

When the systems are complex and heterogeneous, like CPSs, a single MAPE loop for managing all adaptations in the system may not be sufficient [28, 36]. Instead, self-adaptive CPSs (SACPSs) typically feature more than a single MAPE loop. Having multiple control loops in SACPSs also raises a number of challenges in their development and operation. A main challenge is how to coordinate the operation of the different MAPE loops. To address this, previous works have proposed the use of design patterns for decentralized MAPE control loops in self-adaptive systems [36].

In this work, we focus on SACPSs in which MAPE-K loops are structured according to the Master-Slave pattern [36]. In this pattern, one or more MAPE-K loops at a lower level perform decentralized monitoring and execution of the adaptation actions. However, centrally, there is a single high-level MAPE-K loop that performs analysis on the monitored data, updates the knowledge, and does the planning accordingly (Figure 1). In particular, the high-level MAPE-K loop needs to reason on the uncertain, inaccurate, and potentially conflicting observations coming from the decentralized monitoring, which, once aggregated, they become knowledge.

SEAMS '20, October 7–8, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7962-5/20/05…$15.00
https://doi.org/10.1145/3387939.3391600

Knowledge aggregation in SACPSs that follows the Master-Slave pattern or any other pattern with decentralized monitoring and centralized analysis is essential since it can have a significant influence on the subsequent phases of planning and executing. Consider, for example, the case of a route planning algorithm that relies on knowledge aggregation from distributed sensors from multiple agents, which may give conflicting readings. Knowledge aggregation in SACPSs is also challenging, since, as mentioned above, it needs to take into account the inherent uncertainties related to each monitor and provide ways to synthesize and consolidate knowledge in different cases where conflicts arise.

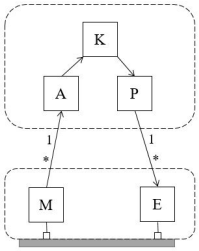

Figure 1: Modified Master-Slave pattern from [36].

Although knowledge and its aggregation is an important concern for MAPE-K patterns as also acknowledged by Weyns et al. [36], there is a scarcity of approaches for modeling self-adaptation knowledge that allows for capturing uncertainty at a local level and for effectively aggregating knowledge for decision making at a global level. Instead, knowledge modeling is typically treated as a domain-specific task, leading to ad-hoc solutions to knowledge aggregation.

In response, in this paper, we present an approach for observations aggregation and knowledge modeling in SACPSs that is domain-independent and can deal with reasoning on inaccurate, partial, and conflicting observations. Concretely, our approach uses Subjective Logic to build knowledge by aggregating partial observations of the context made by each agent in a decentralized multi-agent SACPSs.

Subjective Logic is an enriched probabilistic logic-based framework for artificial reasoning, based on which Subjective Opinions about a knowledge item from the different monitors are created. Additionally, Subjective Logic proposes different fusion operators that allow aggregating the opinions to a final actionable set of knowledge items in the analysis phase, based on which the runtime context model in the Knowledge component is accordingly updated, and later utilized to plan the next adaptation actions.

In short, the main contributions of the paper are the following:

- A subjective logic-based approach for knowledge aggregation in decentralized monitoring of partial context observation in SACPSs.
- An open-source implementation of a ROS-based simulated multi-robot system, based on the robotics use case, further explained in Section 2.

2 RUNNING EXAMPLE

To motivate the need for knowledge aggregation in self-adaptive systems and illustrate our approach, we introduce a reference problem from the domain of CPSs, in particular from the robotics domain, which is also used as a running example throughout the paper.

The reference problem is comprised of several cleaning robots operating in the same context, e.g., a room. Each robot is able to autonomously move to a destination while avoiding 1) static obstacles (e.g., walls, furniture, etc.) and 2) dynamic obstacles (e.g., other robots, humans) along its way. New dirt tasks continuously appear in the room, and the robots discover them in a distributed manner with a 2D laser LIDAR scanner, capable of sensing 360 degrees radius. The robot's observation space is determined by the scanning distance of the laser scanner, which is less than the room's dimensions. Hence, each robot can only observe partially the room in which it operates. Consequently, the robots can detect the newly appearing dirt tasks only if they are within their range of observation.

The mission of the robots is to keep the room as clean as possible by discovering the dirt tasks and then cleaning them most efficiently. However, the fact that the robots have only a partial observation of the room brings inefficiency to the overall performance, for example, when one part of the room is getting dirtier than the other.

Therefore, the partial observations made by each robot are sent periodically to a `Cleaning Controller`, whose responsibility is to aggregate the received observations and assign the discovered dirt tasks to the robots, while respecting the optimality criteria. Once a dirt task is assigned to the robot, it navigates and moves to the location of the corresponding task, accomplishes it, and then navigates to the next task in its queue.

We model the context as a grid map with a size equal to the size of the room (Figure 3). The cells in the grid are either occupied by static obstacles, or by dynamic obstacles: robots or dirt tasks. A context variable models whether a specific cell of the grid map has a dirt task or not, so there are as many context variables as cells in the map of the room. Dirt tasks appear per cell; therefore, in the paper, we use dirt task and dirt cell interchangeably.

The multi-robot cleaning system is subject to external (contextual) and internal (system) uncertainties, manifested via the continuous appearance of tasks in the room and different sensor uncertainties, respectively. Occasionally, each robot will mistakenly sense a dirt task when there is none, and on the contrary, fail to sense an actual task. This potentially results in different robots holding different opinions regarding the space they observe, which requires appropriate conflict resolution during the aggregation process.

In this setting, each robot is an agent of the SACPS and can be modeled via a low-level MAPE-K loop (with respect to the Master-Slave pattern). Each robot independently *monitors* its surroundings and *executes* actions to accomplish its assigned tasks. `Cleaning Controller` has the role of the high-level MAPE-K loop performing the centralized *analysis* and *planning*.

3 BACKGROUND ON SUBJECTIVE LOGIC

Subjective Logic (SL) [19, 20] is a framework for artificial reasoning, in which the general idea is to enrich probabilistic logic by explicitly including (1) uncertainty about probabilities and (2) subjective belief ownership. It allows to express a degree of (un)certainty about a subjective belief (called *opinion*).

To reason with propositions whose truth values are uncertain, Bayesian probability and statistics can also be employed [18]. However, this type of probabilistic logic does not allow to seamlessly model situations where different agents express their beliefs about

the same proposition. SL explicitly integrates the subjective nature and ownership of beliefs in its formalism, allowing the combination of different beliefs about the same proposition. Nevertheless, the interpretation of a SL opinion in the Bayesian perspective is possible by mapping opinions into probability distributions [19].

SL is based on the Dempster-Shafer Theory of evidence (DST), a flexible theoretical framework to represent uncertainty introduced by Dempster [13] and Shafer [32]. In particular, DST's rule of combination, originally proposed for merging sources of evidence in DST, is also used in SL where it represents a method of preference combination embodied in SL's *belief constraint operator* [21]. Moreover, the idea of explicit representation of ignorance is inherited from the Dempster-Shafer belief theory [19, 32].

3.1 Subjective Logic Opinions

The fundamental building block of SL is a *subjective opinion* that represents the amount of uncertainty on the degree of truth about a proposition. The representation of a subjective opinion is a composite function consisting of belief masses, uncertainty mass and base rate. An opinion expresses a belief about the state of a variable which takes its values from a domain (t.e., a state space).

A domain represents all the possible states of a variable situation. Domains can be binary or n-ary. A binary domain can be denoted $\mathbb{X} = \{x, \overline{x}\}$, where $\overline{x}$ is the complement of x. Binary domains are typically used when modeling situations that have only two alternatives, such as the case of our running example, where a location in the map can either have dirt or not. Situations with more than two alternatives have n-ary domains. If $\mathbb{X}$ denotes a binary or an n-ary domain, X can be a random variable which takes its values from $\mathbb{X}$. For instance, we model a situation in our running example using binary random variables X that take their values from the binary domain $\mathbb{X} = \{dirt, no\_dirt\}$.

Binomial Opinion Representation. In SL, the notation w_X^A is used to denote opinions, where X indicates the target variable or proposition to which the opinion applies, and A indicates the agent who holds the opinion. Opinions on binomial variables (e.g. variables with domain $\mathbb{X} = \{x, \overline{x}\}$) are called binomial opinions, and a special notation is used for their mathematical representation.

Definition 1 (Binomial Opinion [20])**.** Let $\mathbb{X} = \{x, \overline{x}\}$ be a binary domain with binomial random variable $X \in \mathbb{X}$. A binomial opinion about the truth/presence of value x is the ordered quadruplet $w_x = (b_x, d_x, u_x, a_x)$, where the additivity requirement $b_x + d_x + u_x = 1$ is satisfied, and where the respective parameters are defined as

b_x: belief mass in support of x being TRUE (i.e. $X = x$),
d_x: disbelief mass in support of x being FALSE (i.e. $X = \overline{x}$),
u_x: uncertainty mass representing the vacuity of evidence,
a_x: base rate, i.e., prior probability of x without any evidence.

Opinions with $u_x = 1$ and $u_x = 0$ are called *vacuous* and *dogmatic*, respectively. Finally, the expected probability of a binomial opinion about value x is defined by: $P(x) = b_x + a_x u_x$

Illustration on the Running Example. In our running example, each robot R issues an opinion for each cell (i, j) that they are able to observe. A binomial opinion about the presence of dirt on a cell is the ordered quadruplet $w_{i,j}^R = (b_x, d_x, u_x, a_x)$ with

b_x: belief mass in support of a tile being dirty,
d_x: disbelief mass in support of no dirt,
u_x: uncertainty of the sensor observation,
a_x: 1/2 (taking an unbiased viewpoint).

The belief mass distribution is calculated as a function of the robot's sensor range and the distance to a detected object. In Section 4 we discuss the details of the detection process and the corresponding belief/disbelief and uncertainty masses calculation.

3.2 Combination of Subjective Logic Opinions

When there are more than one opinions for a proposition, there is often the need to merge or combine them into a single collective opinion. Such knowledge aggregation with Subjective Logic can be realized through a process called *Belief fusion* [20]. Multiple distinct agents, denoted A_1, A_2, ... A_N, can produce different and possibly conflicting opinions $w_X^{A_1}$, $w_X^{A_2}$,...$w_X^{A_N}$ about the same variable X. Multi-source fusion consists of merging the different sources into a single source that can be denoted $\diamond(A_1, A_2, ..., A_N)$, and mathematically fusing their opinions into a single opinion denoted $w_X^{\diamond(A_1, A_2, \ldots, A_N)}$.

Subjective logic provides a variety of operators, generalizing and extending operators from binary logic and probability calculus, including different belief fusion operators: *averaging belief fusion, cumulative belief fusion, weighted belief fusion, consensus & compromise fusion*, and *belief constraint fusion* [33]. Each of these fusion operations is designed to determine the shared belief and uncertainty of a group of evidence sources, with different applications depending on how evidence should be combined.

In the rest of the section, we detail on the Cumulative Belief Fusion and Consensus & Compromise Fusion operators, which we have experimented with in the running example.

Cumulative Belief Fusion (CBF). CBF is suitable when it is assumed that the amount of independent evidence increases with the inclusion of more independent sources [20, 22]. If no dogmatic opinion is present, CBF cumulates the evidence parameters of all opinions. Alternatively, only dogmatic opinions are considered in the cumulation of evidence. Vacuous opinions have no influence on the result. Applying CBF to non-conflicting, uncertain opinions reduces the uncertainty of the resulting opinion. On the other hand, applying CBF to conflicting opinions with the same uncertainty mass has the effect of canceling them out. The CBF operator is associative, commutative, and non-idempotent. The last property means that the fusion of equal opinions will in general produce an opinion that is different from the initial ones.

Consensus & Compromise Fusion (CCF). CCF is suitable when there is a need for keeping shared beliefs from different sources and transforming conflicting beliefs into compromise belief [20]. Conflict resolution is achieved by first computing a consensus, conserving the agreed weight of all sources, and then computing a weighted compromise for the residue belief mass based on the relative uncertainty and the corresponding base rates [33]. Similar to CBF, vacuous opinions are neutral elements in CCF fusion. Contrary to CBF though, CCF is idempotent, meaning that fusing equal opinions produces the same opinion. Technically, the calculation of CCF consists of three phases, namely the consensus phase, the compromise phase, and the normalization phase[22, 33].

Illustration on the Running Example. The different robots in our running example are the agents that hold independent opinions

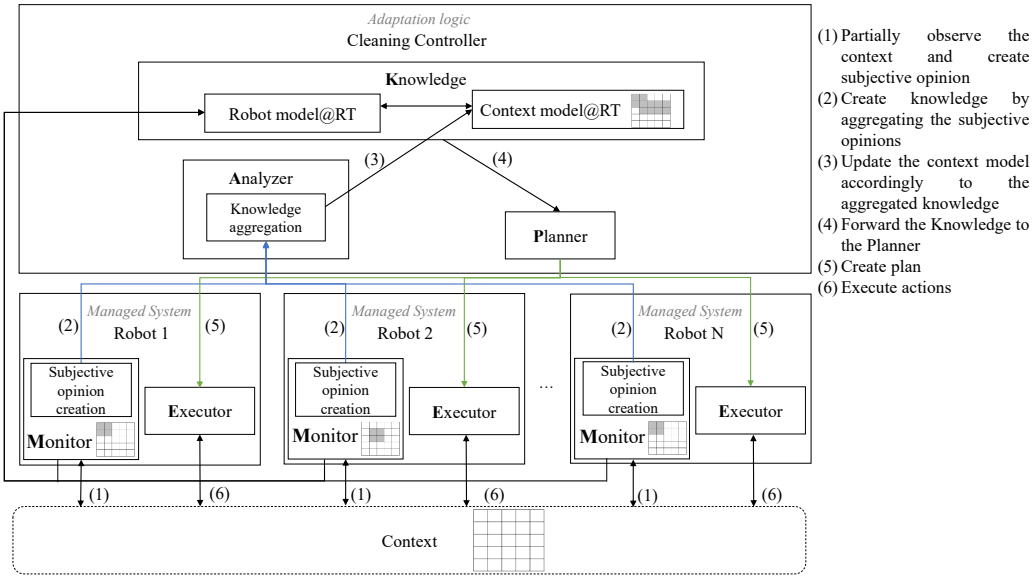

Figure 2: Overview of the approach.

about the cells that they observe being dirty or not. CBF is known to be well suited for fusing opinions coming from sensor-generated evidence [22], while CCF is well suited for fusing opinions coming from different experts. Both operators can be used in the running example, since they can cope with shared beliefs (e.g. robots detecting dirt in a cell with low confidence), conflicting beliefs (e.g. when a robot detects dirt in a cell due to sensor noise while another does not), and vacuous opinions (e.g. a robot does not observe a cell at all since it is out of its LIDAR range). Choosing CCF over CBF allows us to better deal with totally conflicting opinions at the cost of higher uncertainty in the merged opinion.

4 APPROACH

4.1 Overview of the Approach

Our approach assumes that there are several MAPE-K loops in the adaptation logic of the SACPS under study and that the MAPE-K loops are structured accordingly to the Master-Slave pattern [36]. In particular, there are several decentralized Monitor and Execute components, and single centralized Analysis and Plan components, as depicted in the overview of the approach in Figure 2.

Monitor. Monitor components belong to the lower-level MAPE-K loops. They make independent observations about the SACPS itself and the context in which the SACPSs operate. These observations are partial—only cover part of the context, and incorrect—they may include mistakes due to sensor noise and inaccuracies. Each Monitor has a Subjective Opinion Creator entity, which is responsible for creating Subjective Logic opinions from each agent about different context variables. Once the Subjective Logic opinions are created, they are independently forwarded to the Analysis component. The Subjective Opinion Creator is further explained in Section 4.2.

Running example. Each robot in our running example represents a monitor component. It periodically senses its own position and the presence of dirt tasks in the room. Its observations are both partial, due to limited range of its LIDAR sensors, and sometimes incorrect, due to noise in its LIDAR sensors. For illustration, Figure 3 shows that each robot, at a point in time, can only observe part of the context.

Analysis. The analyzer is a centralized component in our approach that collects the Subjective Logic Opinions from the distributed Monitor components and updates the run-time context model in the Knowledge component. In particular, the Knowledge Aggregator entity is responsible for combining the opinions from different agents about different context variables using a Subjective Logic operator.

Running example. In our running example, the Analyzer is a component housed in `Cleaning Controller`. It aggregates the partial observations by fusing the opinions made by robots for the cells that they are observing. The Knowledge Aggregator is further explained in Section 4.3.

Plan. The Planner is another centralized component in our approach that is responsible for selecting adaptation actions or plans, which are later being executed by each lower-level MAPE-K loop. We assume that the Planner relies on the run-time context model represented by the context variables, upon which agents issue opinions in the previous step. However, we do not prescribe how to perform planning: any planning approach (e.g., rule-based, goal-oriented) can be used within our approach.

Running example. In our running example, the Planer is also part of the centralized `Cleaning Controller`. It takes as input the aggregated knowledge in terms of fused opinions about the appearance of dirt tasks in the cells. The discovered, unassigned tasks are assigned to the robots, as exlained in Section 2.

Execute. Executor components belong to the lower-level MAPE-K loops. They obtain adaptation actions or plans from the Planner of the higher-level MAPE-K loop and independently execute them.

Running example. In the running example, each robot represents an Executor component. It keeps a self-adaptive priority queue of the locations of the dirt tasks that a robot needs to accomplish. The queues of the robots are modified at run-time, based on the distance of the robot closest to the newly appeared cleaning task. As long

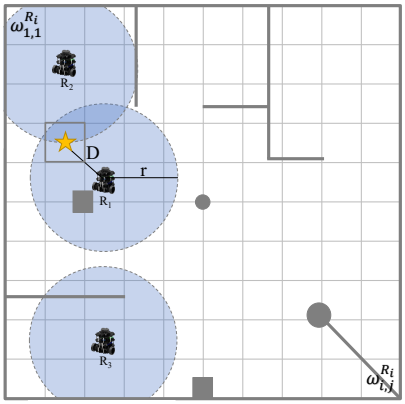

Figure 3: Grid map and partial context observation

as there are tasks in its queue, each robot picks the next task and navigates autonomously to the corresponding cell to clean the task.

4.2 Subjective Opinion Creator

A robot R issues a subjective opinion $w^R_{i,j}$ for each cell (i, j) in the grid map that it is within its detection range r (Figure 3). This opinion depends on the Euclidean distance D of the cell from the robot and models whether that cell contains dirt or not. In particular, the parameters of an opinion $w^R_{i,j} = (b_X, d_X, u_X, a_X)$ are calculated by the following formulas:

$$b_X = \begin{cases} 1 - u_X, \text{for } X = dirt \\ 0.0, otherwise \end{cases} \qquad d_X = \begin{cases} 1 - u_X, \text{for } X = no\_dirt \\ 0.0, otherwise \end{cases}$$

$$u_X = min(0.99, \frac{D}{r}) \qquad a_X = 1/2$$

The further a cell is from the robot, the higher the uncertainty mass u_X of the robot's opinion. Observations at the edge of LIDAR sensor range are considered highly uncertain, but can still contribute during knowledge aggregation; hence, we assign an uncertainty value of 0.99 instead of a totally uncertain opinion (i.e., $u_X = 1$). If a robot detects dirt on a cell, the belief mass b_X becomes the complement of u_X and the disbelief mass d_X for that cell becomes zero. On the contrary, if no dirt is detected, b_X becomes zero and d_X becomes the complement of u_X. The base rate a_X is always considered to be the default base rate for a binary domain, i.e. $a_X = 1/2$. Finally, no subjective opinions are issued for (i) cells which are occupied by static obstacles (e.g. walls) since these are assumed to be accurately detected, (ii) cells that lie outside of the detection range of the robot.

4.3 Knowledge Aggregator

When the system is first initialized, the run-time context model does not contain information that the Analyzer can use to carry out its tasks. We model this situation by creating a vacuous subjective opinion for each cell of the context grid map. This initialization serves two purposes in our subjective logic approach: (1) a vacuous opinion will inform the analyzer that there is no previous knowledge about a context variable and (2) when the first knowledge aggregation is executed, the existing vacuous opinions do not influence in the final result.

Opinion	No conflict				Conflicting observations			
	Inputs		Aggregation		Inputs		Aggregation	
	R_1	R_2	CBF	CCF	R_1	R_2	CBF	CCF
b_x	0.630	0.010	0.631	0.634	0.000	0.010	0.004	0.004
d_x	0.000	0.000	0.000	0.000	0.630	0.000	0.628	0.630
u_x	0.370	0.990	0.369	0.366	0.370	0.990	0.369	0.366
a_x	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
$P(x)$	0.815	0.505	0.816	0.817	0.185	0.505	0.188	0.187

Table 1: Knowledge aggregation of two robots' observations

The knowledge aggregation takes place each time a robot informs the Knowledge Aggregator of a made partial observation of the context. For the cells covered in the partial observation, subjective opinions are issued. Additionally, the Knowledge Aggregator extracts the previously-stored opinions for the corresponding cells from the Knowledge. For each cell, both opinions are then fused and aggregated, and the run-time context model in the Knowledge is updated accordingly. This process is executed with the same frequency for all the robots scanning their surrounding area.

This aggregation step aims to solve potential conflicting observations, and at the same time, increase the confidence of the observations, according to the chosen subjective logic operator.

As an example, let us consider the situation illustrated in Figure 3, where robots R_1 and R_2 hold overlapping partially observed contexts and a dirt cell (depicted with a yellow star) is detected. When the knowledge aggregator receives the independent partial observations sent by the robots, it initiates the aggregation process. Let us further consider two different scenarios: (1) R_1 and R_2 detect dirt in the same cell and, (2) R_1 does not detect dirt but R_2 does. Using the formulas described in Section 4.2, Table 1 shows the calculated opinions for both scenarios, and the knowledge aggregation results using CBF and CCF operators.

In both scenarios, we observe improvement in the confidence of the aggregated observations, i.e., the uncertainty mass in the aggregated opinion decrease compared to the individual uncertainty masses of each robot's opinion (blue cells in Table 1). However, in the presence of conflicting observations, the resulting belief masses reflect a consensus and compromise of the individual robots' opinions (yellow cells in Table 1); this compromise belief is reflected in the expected probability $P(x)$ calculated as explain in Section 3.1. The resulting $P(x)$ of the aggregated opinions is then used by the Analyzer to decide when the detected dirt should be considered as a goal to be assigned to one of the participating robots.

5 IMPLEMENTATION

In this section, we discuss the implementation of our testbed based on the reference problem, which was previously described in Section 2. Robotics is an inherently complex domain, particularly when considering not only one but multiple agents. So merely creating and setting up a realistic multi-robot system presents a challenge by itself. By providing a *simulated*, yet physically correct representation of the robots with their sensors and actuators, and the context where they are operating, our current implementation provides a foundation for various applications and experiments, which can also be easily modified accordingly to the individual needs of other researchers.

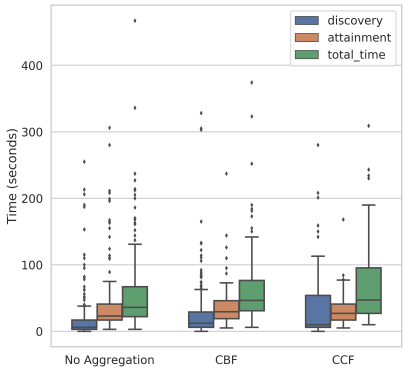

Figure 4: Result for a simulation of 2 robots with and without knowledge aggregation.

In our implementation, the entire communication is based on the Robot Operating System (ROS). For simulating the robotics system we use Gazebo [1, 24], and the robots we simulate are *TurtleBot 3 Burger*[1]. With our implementation, one can simulate as many robots, as long the number of robots makes sense in the given room. Additionally, different room maps, from the one currently considered in our system, can be used. Gazebo relies on a well-established physics engines, which enables high physical [10], functional [2] and visual [31] fidelity. Simulations with high fidelity closely resemble the real world, meaning that the model and even the physics of the robots, the environment, including the static and dynamic objects are simulated as realistically as possible. The use of ROS allows the system to be deployed on real, physical robots without any modification. The source code of the implementation, together with complete documentation, and installation instructions are available on the following link: https://github.com/squijanor/knowledge-aggregation-subjective-logic.git. For subjective logic, we have used an open-source Java library[2].

6 PRELIMINARY EXPERIMENTS

6.1 Setup

The testbed provides a simulation of n robots deployed in a room spanning 10 x 10m. To evaluate the knowledge aggregation approaches, we conducted five series of experiments, 10 minutes each. For the simulation of the appearance of new tasks, we used a random seed in each experiment, and to guarantee a better replication of the test scenarios, we used a fixed frequency with which the new tasks are created. We compare these results with a base scenario without knowledge aggregation. In this scenario, the `Cleaning Controller` does not aggregate nor solve conflicts in the observations made by different robots; instead, it proceeds to create goals directly from the observations received from every individual robot. This might result in goals at locations where dirt tasks do not exist or locations that are different from the real locations of the tasks.

6.2 Preliminary Results

Figure 4 shows first results from experiments comparing no aggregation, and aggregation with CBF and CCF. CBF requires the fusion of different sources to increase the confidence in the observations, whereas CCF trades-off conflict resolution for a higher uncertainty in the merged result. We used 0.5 as expected probability threshold in the aggregated subjective opinions to determine when the detected task should be assigned as a goal to a robot. By adjusting this parameter, the effect on the discovery time can be controlled. In our first experimental results, we see that the discovery time increases for both CBF and CCF. This is expected, since they need more evidence (and hence time) to create cleaning goals. The attainment time is roughly the same in all cases.

7 RELATED WORK

Models@Runtime Models@Runtime [5, 6, 15, 34] are a promising approach to managing complexity in run-time environments, based on software models. They are considered as adaptation mechanisms, or rather support for realizing self-adaptive systems. According to [6] the run-time models provide "abstractions of run-time phenomena" and they can be used in a various ways by different stakeholders. Additionally, the models should represent the system by reflecting the system and its current state and behavior. Namely, if the underlying system changes, then the representations of the system—the models—should also change. Floch et al. in [15] and Bennaceur et al. in [5] identify the need for mechanisms to reason about the system and its environment in models@runtime, without proposing any concrete solution.

Uncertainties in Self-adaptive systems. The term uncertainty has been broadly discussed across many disciplines and sciences. However, in the field of self-adaptive systems, uncertainties have a central role as the main triggers for a system to self-adapt so that the business continuity of the system can be preserved during runtime. Across the literature, there have been many proposed works that have tried to understand the scope and the effects that the uncertainties have on the dynamic systems[8, 14, 16, 25, 26, 30, 35]. Ramirez et al. [30] classify uncertainties on three different levels, based on the uncertainties' sources: uncertainties on the requirement level, design level and runtime level. According to the authors, the potential sources of runtime uncertainties are mainly related to interactions between the system and its context, including sensor noise, inaccuracy of sensor measurements, or an unpredictable system environment. Besides the proposed mitigation techniques for unpredictable environment [3, 7, 9, 17, 37], sensor failure [7, 17] and incomplete information [4, 9, 37], there are no identified techniques to mitigate the rest of the run-time uncertainties including sensor noise, sensor imprecision and inconsistency [30]. We hope that with the proposed approach in this paper, we are contributing towards narrowing this gap.

8 CONCLUSION

In this paper, we propose an approach that uses subjective logic to collaboratively aggregate the partial observations of the context made by each CPS (robot) in a multi-agent setup. Based on the aggregated observations, we accordingly update the run-time context model of the context in the knowledge of the adaptation logic, which is later utilized by the adaptation logic to analyze and plan the next adaptation actions.

[1] https://www.turtlebot.com/
[2] https://github.com/vs-uulm/subjective-logic-java

Through preliminary experiments, we demonstrated that Subjective logic is a flexible framework to merge knowledge from different agents. The provided testbed sets the basis to experiment with different methods for knowledge aggregation. In this work we only evaluated the application of CBF and CCF operators. In future work, we plan to evaluate the remaining set of operators, which will provide a complete insight into the full capabilities of subjective logic as a means for knowledge aggregation. Also, to guarantee the internal validity of our testbed, we will conduct different and multiple trials to determine the statistical soundness of our results. Namely, we plan to increase the number of experiments, testing different room distributions and the simulation of multiple robots.

ACKNOWLEDGMENTS

This research has in part been funded by the *German Federal Ministry of Education and Research* under the grants no. 01IS16043A.

REFERENCES

[1] Carlos E. Aguero, Nate Koenig, Ian Chen, Hugo Boyer, Steven Peters, John Hsu, Brian Gerkey, Steffi Paepcke, Jose L. Rivero, Justin Manzo, Eric Krotkov, and Gill Pratt. 2015. Inside the Virtual Robotics Challenge. 12 (2015), 494–506. Issue 2. https://doi.org/10.1109/TASE.2014.2368997

[2] John A. Allen, Robert T. Hays, and Louis C. Buffardi. 1986. Maintenance Training Simulator Fidelity and Individual Differences in Transfer of Training. 28 (1986), 497–509. Issue 5. https://doi.org/10.1177/001872088602800501

[3] Luciano Baresi, Liliana Pasquale, and Paola Spoletini. 2010. Fuzzy goals for requirements-driven adaptation. In *2010 18th IEEE International Requirements Engineering Conference*. IEEE, 125–134.

[4] Nelly Bencomo, Jon Whittle, Pete Sawyer, Anthony Finkelstein, and Emmanuel Letier. 2010. Requirements reflection: requirements as runtime entities. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, Vol. 2. IEEE, 199–202.

[5] Amel Bennaceur, Robert France, Giordano Tamburrelli, Thomas Vogel, Pieter J Mosterman, Walter Cazzola, Fabio M Costa, Alfonso Pierantonio, Matthias Tichy, Mehmet Akşit, et al. 2014. Mechanisms for leveraging models at runtime in self-adaptive software. In *Models@ run. time*. Springer, 19–46.

[6] Gordon Blair, Nelly Bencomo, and Robert B France. 2009. Models@ run. time. *Computer* 42, 10 (2009), 22–27.

[7] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. 2009. Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*. Springer, 48–70.

[8] Javier Cámara, David Garlan, Won Gu Kang, Wenxin Peng, and Bradley Schmerl. 2017. Uncertainty in Self-Adaptive Systems Categories, Management, and Perspectives. (2017).

[9] Betty HC Cheng, Pete Sawyer, Nelly Bencomo, and Jon Whittle. 2009. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 468–483.

[10] Jeff Craighead, Robin Murphy, Jenny Burke, and Brian Goldiez. 2007. A Survey of Commercial & Open Source Unmanned Vehicle Simulators. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*. IEEE, 852–857. https://doi.org/10.1109/ROBOT.2007.363092

[11] Rogério De Lemos, David Garlan, Carlo Ghezzi, Holger Giese, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Danny Weyns, Luciano Baresi, Nelly Bencomo, et al. 2017. Software engineering for self-adaptive systems: Research challenges in the provision of assurances. In *Software Engineering for Self-Adaptive Systems III. Assurances*. Springer, 3–30.

[12] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*. Springer, 1–32.

[13] A. P. DEMPSTER. 1968. A Generalization of Bayesian Inference Author (s): A . P . Dempster Source : Journal of the Royal Statistical Society . Series B (Methodological), Vol . 30 , No . 2 Published by : Wiley for the Royal Statistical Society Stable URL : http://www.jstor.o. 30, 2 (1968), 205–247.

[14] Naeem Esfahani and Sam Malek. 2013. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II*. Springer, 214–238.

[15] Jacqueline Floch, Svein Hallsteinsen, Erlend Stav, Frank Eliassen, Ketil Lund, and Eli Gjorven. 2006. Using architecture models for runtime adaptability. *IEEE software* 23, 2 (2006), 62–70.

[16] David Garlan. 2010. Software engineering in an uncertain world. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 125–128.

[17] David Garlan, S-W Cheng, A-C Huang, Bradley Schmerl, and Peter Steenkiste. 2004. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer* 37, 10 (2004), 46–54.

[18] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2014. Part I Fundamentals of Bayesian Inference. In *Bayesian Data Analysis* (3. ed. ed.). CRC Press, Chapter 1, 4–29.

[19] Audun Jøsang. 2001. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9, 3 (2001), 271–311.

[20] Audun Jøsang. 2016. *Subjective Logic*. Springer International Publishing Switzerland. https://doi.org/10.1007/978-3-319-42337-1

[21] Audun Jøsang and Simon Pope. 2012. Dempster's rule as seen by little colored balls. *Computational Intelligence* 28, 4 (2012), 453–474. https://doi.org/10.1111/j.1467-8640.2012.00421.x

[22] Audun Josang, Dongxia Wang, and Jie Zhang. 2017. Multi-source fusion in subjective logic. *20th International Conference on Information Fusion, Fusion 2017 - Proceedings* (2017). https://doi.org/10.23919/ICIF.2017.8009820

[23] Jeffrey O. Kephart and David M. Chess. 2003. The Vision of Autonomic Computing. *Computer* 36, January (2003), 41–50. https://doi.org/10.1046/j.1365-2745.2002.00730.x arXiv:arXiv:astro-ph/0005074v1

[24] Nathan Koenig and Andrew Howard. 2004. Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. IEEE, 2149–2154. https://doi.org/10.1109/IROS.2004.1389727

[25] Sara Mahdavi-Hezavehi, Paris Avgeriou, and Danny Weyns. 2017. A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements. In *Managing Trade-Offs in Adaptable Software Architectures*. Elsevier, 45–77.

[26] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*. ACM, 3–14.

[27] Ana Petrovska and Alexander Pretschner. 2019. Learning Approach for Smart Self-Adaptive Cyber-Physical Systems. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 234–236.

[28] Mariachiara Puviani, Giacomo Cabri, and Franco Zambonelli. 2013. A taxonomy of architectural patterns for self-adaptive systems. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*. ACM, 77–85.

[29] Federico Quin, Thomas Bamelis, Singh Buttar Sarpreet, and Sam Michiels. 2019. Efficient analysis of large adaptation spaces in self-adaptive systems using machine learning. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 1–12.

[30] Andres J Ramirez, Adam C Jensen, and Betty HC Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 99–108.

[31] Edward J. Rinalducci. 1996. Characteristics of Visual Fidelity in the Virtual Environment. 5 (1996), 330–345. Issue 3. https://doi.org/10.1162/pres.1996.5.3.330

[32] Glenn Shafer. 1976. A mathematical theory of evidence. *Princeton University Press* (1976).

[33] Rens W. Van Der Heijden, Henning Kopp, and Frank Kargl. 2018. Multi-Source Fusion Operations in Subjective Logic. *2018 21st International Conference on Information Fusion, FUSION 2018* (2018), 1990–1997. https://doi.org/10.23919/ICIF.2018.8455615 arXiv:arXiv:1805.01388v1

[34] Thomas Vogel, Andreas Seibel, and Holger Giese. 2010. The role of models and megamodels at runtime. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 224–238.

[35] Kristopher Welsh and Pete Sawyer. 2010. Understanding the scope of uncertainty in dynamically adaptive systems. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2–16.

[36] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. GÃűschka. 2013. On Patterns for Decentralized Control in Self-Adaptive Systems. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, RogÃŕrio de Lemos, Holger Giese, Hausi A. MÃijller, and Mary Shaw (Eds.). Springer, Berlin, Heidelberg, 76–107. https://doi.org/10.1007/978-3-642-35813-5_4

[37] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty HC Cheng, and Jean-Michel Bruel. 2009. Relax: Incorporating uncertainty into the specification of self-adaptive systems. In *2009 17th IEEE International Requirements Engineering Conference*. IEEE, 79–88.