Aggregate Drone Monitoring of Wildfires

Aggregate Drone Monitoring of Wildfires

1st Giorgio Audrito
Department of Computer Science
Università degli Studi di Torino
Turin, Italy
giorgio.audrito@unito.it

2nd Marco Ottina
Department of Computer Science
Università degli Studi di Torino
Turin, Italy
marco.ottina@unito.it

***Abstract*—Wildfires are an ever-increasing problem, exacerbated by the current global warming trends. Accordingly, it is becoming more and more relevant to monitor factors influencing their outbreaks and spreading, in order to both preemptively act on the most risky areas, and guide interventions in case an outbreak occurs. Different approaches have been proposed during the decades tackling this issue, which however require large datasets that are difficult and expensive to gather.**

In this paper, we propose to address the management of wildfires by empowering existing centralised models with a decentralised component. Leveraging dedicated monitoring drones together with smartphones held by experts and intervention corps, a decentralised system could both enhance data collection and assist interventions. As conditions near wildfires require strong fault-tolerance guarantees, we propose to develop such an application through *aggregate programming*, a novel approach to the resilient programming of decentralised systems.

***Index Terms*—fire prevention, monitoring, aggregate programming, edge intelligence**

I. Introduction

Wildfires are an ever-increasing problem [1] and their rates and intensities are influenced by many different factors, such as global warming [2], [3], weather conditions in general, the interested areas properties and human activities [4]. For example, the humidity level interferes by reducing the probability of fire outbreaks. The vegetation distribution and type gives relevant contributions to fires development, especially during hot and dry seasons. The landscape and the presence of water in form of lakes, rivers or vapour are important factors that also need to be taken into account. Human activity and urbanization play a role in wildfire ignition as well [5]. This multitude of different relevant factors contributes significantly to the complexity of the wildfire monitoring problem.

Thus, collecting an increasing amount of data may highlight some important factors that can by themselves spark the ignition of fires, or be crucial actors in their propagation. It is then becoming more and more relevant to monitor at a large scale factors such as weather, concentrated elements like patches of fuel-rich plants, and human activities [6] in order to achieve a wider and deeper understanding of the wildfires outbreaks and to preemptively act on the most risky areas.

The coordination of both the monitoring and intervention processes is a non-trivial task, which insofar has been carried out through centralised approaches requiring a reliable and efficient network. However, in real-world settings the connection between agents deployed in wild areas and the Internet may be unstable, weak or absent, thus limiting or impeding a centralised communication and orchestration.

This paper proposes to manage the interconnection between deployed devices (monitoring drones) and intervening humans (through their smartphones) according to a decentralised approach. Thanks to the *aggregate programming* model [7] and its implementation FCPP [8], [9], a resilient ad-hoc network can be maintained even in absence of Internet connection, while handling message losses due to adverse weather conditions. Exploiting the aggregate programming framework, we delineate an approach specifically designed at the maintenance of such a resilient network, with some devices identified as *workers* performing the necessary monitoring and intervention tasks, while other devices called *postmen* focus on bridging messages in order to keep the network connected to ground *stations*, that are meant to operate as higher-level orchestrators, also interacting with the Internet if possible.

Section II introduces the necessary background concepts on collective adaptive systems in general, on the aggregate programming model and its implementation FCPP. Section III describes the wildfire monitoring case study, focusing on the issues and obstacles that a solution to this problem needs to address. Then, Section IV presents the proposed technological solution, first at a higher level and then outlining the aggregate programming patterns to be leveraged in order to ease its design. Finally, Section V concludes proposing a roadmap to the practical implementation of the proposed solution.

II. Background

A. Collective Adaptive Systems

Collective Adaptive Systems (CAS) are collections of intelligent agents able to automatically change their internal structure (i.e., the connections between their components) and/or their function in response to external inputs.

Due to such characteristics, programming CAS presents peculiar challenges, that must be addressed in ways that depend on the given context and goals. At a sufficient level of abstraction, the various approaches can be classified in four categories [10]:

- methods that abstract devices and/or networks, such as TOTA [11] and MapReduce [12];

978-1-6654-6297-6/22/$31.00

- methods that provide geometrical or topological pattern languages, such as the *self-healing geometries* in [13];
- methods that provide languages for information retrieval and routing, such as TinyLime [14];
- general space-time computing models, such as StarLisp [15] and Aggregate Programming [7].

B. Aggregate Programming

Among the approaches previously mentioned, in this paper we focus on *Aggregate Programming* (AP). AP is characterized by the definition of a single program that is executed asynchronously in each node of a whole network. The networked system is thus modelled as a single aggregate machine, which manipulates collections of distributed data called *computational fields*.

Communication between devices is realised at low level through proximity-based broadcasts, which at a higher level generate *neighbouring fields*, i.e., maps from neighbour device identifiers to their relative values. Neighbouring fields cannot be accessed directly; instead, they are manipulated through "map" operations that produce a new field, and "fold" operations that synthesize a single value from a field.

Such aggregate computations can be expressed through the *Field Calculus* (FC) [16], a minimal functional language providing the necessary mechanism to express and functionally compose distributed computations, neglecting low-level aspects such as synchronisation, delivery of messages between devices, and the number and physical positions of devices in the network. The FC language has been proved to be able to express any computable distributed program [17], while being particularly effective in the monitoring of space-time properties [18]–[21], and allowing integration with traditional multi-agent planning techniques [22], [23].

An FC program P running on every device i of the network, executes the following steps periodically:

- the device perceives contextual information, i.e.:
  - data provided by local sensors,
  - local (state) information stored in the previous round,
  - messages received from neighbours after the previous round.

  As said above, the latter are made available to the program P as a *neighbouring field* ϕ;
- the device evaluates the program P considering as input the contextual information gathered as described above. Note, therefore, that P is not only executed by each device, but also at each round: when needed, different behaviors are obtained by branching statements in P based on the input context;
- the result of the local computation is stored locally (as local state), sent to neighbours and may produce outputs fed to local actuators.

The above steps, executed across space by different devices, and across time at different rounds, give rise to a global behaviour at the overall network-level [24] that can thus be viewed as a single aggregate machine. While the neighbouring relation is usually based on spatial proximity, it is possible to define it as a logical relationship, for example as a master-slave relationship among devices independently of their position.

C. FCPP

FCPP (FieldCalc++) is a library written in the C++ language that implements the Field Calculus (FC). Given the goal of being able to deploy FCPP on as many platforms as possible, C++ has been chosen as the implementation language not only for its power and efficiency, but also because it can target most platforms, from microcontrollers to GPUs. Beside providing an internal DSL for expressing FC programs within C++, the library provides several features:

- a component-based software architecture, suitable to be extended and customised for different application scenarios, such as deployments on IoT devices, simulation, and HPC;
- an efficient implementation exploiting compile-time optimisations through advanced template programming [25];
- support for parallel execution of a simulated system or self-organising cloud application;
- tools for executing FC programs on simulations of distributed systems.

The only scenario that is currently fully supported by the implemented components of FCPP is the simulation of distributed systems. Compared to the alternative implementations of FC (Protelis [26] and Scafi [27]), it features additional simulation capabilities (3D environments, basic physics, probabilistic wireless connection models), with a significant reduction of the simulation cost, and a corresponding speedup of the development and test of new distributed algorithms. Moreover, thanks to the extensible architecture, it is much easier to address additional scenarios than with previous FC implementations. Two such scenarios are of particular practical interest:

- deployments on microcontroller-based systems typically used in IoT applications, which have limited computing power and memory;
- deployments as self-organising cloud applications, which require fine-grained parallelism in order to be able to scale with the resources allocated in the cloud.

Prototype components addressing these scenarios are currently under development, and are already available in the main FCPP distribution.

Figure 1 shows the architecture of the FCPP library, partitioned in three main conceptual layers:

1) *C++ data structures of general use*. Some are needed by the components of the second layer either for internal implementation or for the external specification of their options; other data structures are designed for implementing the aggregate functions of the third layer.
2) *components*. They define the abstractions for representing single devices (*node*s) and the overall network (*net*), which is fundamental in scenarios where there is no physical network, such as simulations and cloud-oriented applications. It is worth noting that, in an

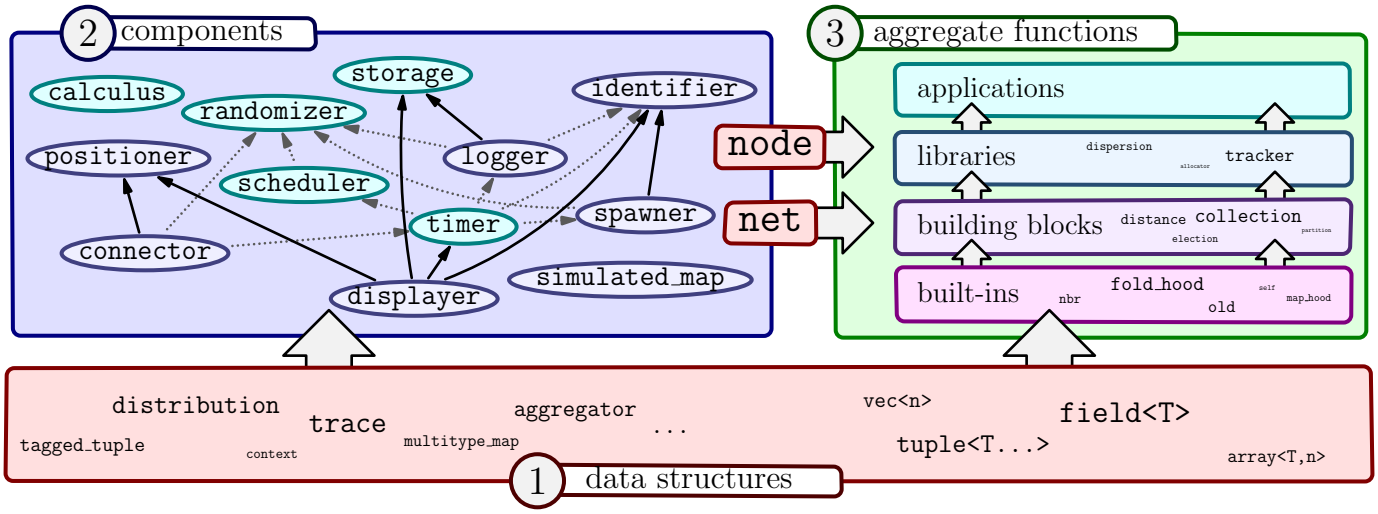

Fig. 1. Representation of the software architecture of FCPP as the combination of three main layers: *data structures* for both other layers, and *components* which provide node and network abstractions to *aggregate functions*. Components are categorized as general purpose (cyan), used across different domains, and simulation-specific (violet), with variations for different domains. The new *displayer* and *simulated map* (simulation-specific) components are highlighted in magenta. Dependencies between them can be either *hard* (solid), for which the pointed component is required as an ancestor of the other; or *soft* (dotted), for which the pointed component is not required, but if present, it should be an ancestor of the other component.

application based on FCPP, the two types `node` and `net` are obtained through template programming by combining a chosen sequence of components [28], each of them providing a needed functionality, in a *mixin*-like fashion [29], [30]. The role of the component system is thus that of enabling the reuse of specific functionalities across different application scenarios.

3) *aggregate functions*. Actual implementations of FC programs, as templated functions with a `node` parameter; note that also these functions are partitioned in several layers, starting from the built-ins that implement the core of FC, up to the applications written by the users of the FCPP library.

Figure 1 shows the dependencies between components, i.e., whether a component needs another component as its ancestor in the mixin composition. The number of such dependencies has been kept as low as possible, and it is always possible to substitute a "required" component for another offering an analogous interface in the composition.

III. Wildfire Monitoring

A. Forecasting Strategies

Different approaches have been proposed during the decades for the monitoring and prevention of wildfires. Almost all of them involve a system aimed at predicting at least one feature. For instance: [31] provides models to predict the wildfire scale, like the fire's duration and the area burned; [32] analyses spatial data to provide natural hazard likelihoods estimations; and [33] combines conditions like wind and vegetation in a cellular automata model to describe the evolution in space and time of the wildfire. Statistics are heavily involved in every model, in particular in [34], which implies that large datasets are needed in order to extract accurate descriptions without suffering from errors due to sample fluctuations.

B. Intervention Guidance

All of these models can be valuable tools, supporting a decision process managed and actuated by agents in a CAS, which monitor susceptible areas for wildfires. Such a CAS may include fixed sensors, drones, and other agents able to both scan the environment to provide data to wildfire prevention models, as well as possibly interacting with their surroundings to mitigate the situation whenever possible. Real-time monitoring, through modelling systems, may be capable to indirectly guide the intervention of firefighters and other experts in order to increase efficiency and reduce risks and wastes of resources. For example, changes in wind direction and speed, or in air composition like in the levels of humidity and oxygen, may impact the flames' evolution, by redirecting, feeding or weakening them. Also, flames exhibiting chaotic-like patterns may hurt the firefighters or set up new things on fire, including hazardous substances as fuel tanks and explosives. The analysis of a current situation and environment may thus redirect the firefighters' actions, for instance, prioritizing some areas and objects over others, or by avoiding being engulfed in flames if the model predicts it is about to happen.

C. Data Acquisition

As discussed in [2], many different data can be of interest for wildfire management: from weather conditions as humidity and strong winds, to human activities and presence of flammable fuels in vegetation or soil. The high variability in type and geographical displacement of the interested data pose difficult challenges in the collection of the desired information. This becomes particularly problematic as the areas of interest

are often poorly served by existing infrastructure, and internet connectivity may be difficult if at all possible.

Various devices and strategies to collect data have been considered, each differing in terms of area coverage and granularity. Satellites may supply huge images depicting large areas, providing overview and esteems to the models mentioned above, but often lack in granularity and fine-detailed resolution due to the satellite's own limitations. Meteorological data or images can be collected during overflight of particular areas [35], such as helicopters, hot air balloons, or unmanned aerial vehicles. In that cases, GPS data may also be used to tag perceived values and to help the orchestration and movements of the involved devices and humans [36]. Fixed sensing infrastructure such as video-cameras and specialised observatories can also be used, but their usefulness and deployment cost is often limited by the availability of internet connection.

Additionally, a new paradigm of data acquisition is emerging in the recent years: collaborative data collection, exploiting social media and the capabilities of modern mobile devices [37]. Large scale disaster events can often be detected and monitored by combining information gathered from different social media networks, obtaining more copious, consistent in time and widespread amount of data. This data can feed forecasting systems, enabling more targeted and localized intervention. For example, some drones can be dispatched in the areas where a dangerously high likelihood of wildfire ignition is suspected [38]. The drones can then reliably provide high-quality data of various types, from images to hygroscopic data or temperature, in selected regions of interest, thus increasing the granularity and precision of the available information, which is needed to feed forecasting and analysis models [6]. The currently limited amount of drones available does not allow a widespread and dense usage of those devices, so they are usually delivered only when strictly needed and within the regions where wildfires are already present or highly likely to spawn in the near future.

D. Difficulties upon Intervention

Many difficulties may arise during interventions of the firefighters and forestry corps, which may be consequently slowed down, limited or disrupted. Wind conditions may hinder the mobility of devices and personnel dispatched, as well as reducing the effectiveness of sensors and water streams aimed to deal with fires. Untracked dry vegetation and fuels [39] may unexpectedly feed the wildfires, increasing their damage potential, and possibly allowing flames to engulf humans, animals or valuable objects within burning areas. The damages and erosion made by fire to huge plants and buildings may lead to their sudden fall, resulting in high risks to everything and everyone in the surrounding area. Clearing the potential diffusion path of a fire is thus a crucial precaution, that however requires careful monitoring.

In current practice, this is usually handled by human experts, based on their subjective perception of the situation. However, fatigue and mental overloading in such critical situations has been shown to often lead to sub-optimal or erroneous decision-making and planning operations. Furthermore, human senses may be limited or imprecise, limiting the forecasting abilities of domain experts. Overall, this considerations demand for the support of machines in the decision process, including sensing drones and artificial intelligence assistants.

On the other hand, drones and other sensing machines have their own limitations. Excessive temperature or air conditions may corrupt the gathered data, requiring to reconstruct information by combining data coming from different but close devices. Connectivity may be absent or scarce in some areas, especially during wildfires due to signal dispersion induced by airborne particles and physical obstacles between drones as trees and hills. Temporarily isolated drones may fail to behave properly and become useless, possibly ending up crashing, requiring orchestration efforts on the drone location and communication patterns to solve or mitigate those issues.

IV. Proposed Technological Solution

We propose to tackle the problem of wildfire monitoring through a combination of drones running Linux-based operative system (OS) and smartphones running Android OS. Other connected devices may also be part of the network (e.g., main stations). We assume that drones will perform most of the tasks, while smartphones would still be helpful both in keeping the network connected, and in allowing friendly human interaction with the system to the dispatched personnel. The interaction between the involved devices could be provided through the *Aggregate Programming* (AP) framework and the FCPP library implementing it, allowing for a resilient and adaptive orchestration. The proposed solution focuses then in the devices management capable to build up a communication infrastructure that is fault and approximation tolerant, robust to an evolving context, flexible under small changes and automatically readjusts itself in real-time in case of changes, disruptions, and connection losses.

A. Network Architecture

In order to program the aforementioned decentralised system of drones and smartphones, we propose to sort devices into three main categories, defining their role in the computation based on their capabilities: workers, postmen, stations. A *worker* is a device having crucial tasks to perform in given locations (called *Points of Interest*), including human support or image acquisition and measuring. A *station* is a device with stable network connectivity (e.g., a truck-like vehicle, or a drone deploy point), able to interact with centralised forecasting services on the cloud. Finally, a *postman* acts as a bridge-like proxy by forwarding messages between workers and stations. A single device may take on multiple roles simultaneously, if its capabilities allow it and tasks are compatible; although we expect most devices to take one role only.

The proposed strategy aims to create a network resilient to disconnection, by assigning the *postman* role to a sufficient number of drones. This has to be accomplished while maintaining the overall effectiveness of the systems, which poses a lower bound on the number of *workers* that have to be

appointed. In the meanwhile, access to cloud-based services through few available *stations* should be granted whenever possible, as it can enhance the system's predictive capabilities. On the other hand, the network should be able to operate even in case of a complete lack of stations. Roles should be initially preassigned, with possibility of varying them over time to react to unexpected environmental changes.

A possible aggregate approach ensuring robust communication in an hostile environment can be the following. A forest structure could be created by an adaptive aggregate algorithm executed on all devices, where *stations* are roots and *parent* devices are assigned in order to ensure connectivity between every node in the network and a root. As stations are connected to the internet, we can assume that all of them are able to communicate with each other, in turn connecting the whole network. If no stations are available, an aggregate algorithm could elect a leader device to act as a station surrogate, to drive connectivity inside the local network, even though in that case internet connectivity will not be available.

The postman role may be adaptively assigned only to devices with movement capabilities (usually drones). A general coordinated policy may assign tasks to the nearest devices that can handle them and are not busy yet, making them *workers*. Devices that are not burdened by tasks may act as postmen, striving to ensure network connectivity. A postman should keep track of their children and parent positions, obtained for instance through a GPS sensor. If the children are sufficiently close to the parent, it may simply strive to stand stationary in a middle-way position, in order to maintain the connection with both children and the parent. If this is not possible, the postman should move back and forth between a position closer to children and one closer to the parent, in order to allow communication to proceed (although slower). While performing the postman task, sensing activities could still proceed and feed some data into the network.

B. Application of Aggregate Programming and FCPP

The AP framework and consequently the FCPP library already provides all necessary prerequisites and algorithms required to develop the proposed strategy. According to the AP framework (cf. Section II-B), drones and smartphones periodically evaluate a same program in cyclic rounds. By tuning the round duration, the whole system can be reasonably reactive and adaptive to changes, while keeping resource consumption under control. Information about the devices states and environment can be easily shared across devices through basic constructs like *nbr* and *share*. The AP framework ensures fault-tolerance by default, as well as algorithms that are guaranteed adaptive, which can build the proposed system.

The selection of *workers* may leverage existing *leader election* algorithms [40] which are already available in FCPP. Furthermore, aggregate functions computing distances between devices [41] can be used to generate the spanning forest and calculate shortest paths from *workers* to their parent *station*. Resilient dispatching of single messages has already been address through the *spawn* construct [42].

V. Conclusions

In this paper, we presented a roadmap towards a resilient technological solution for the problem of wildfire monitoring. A system of drones may be deployed in a critical area for data acquisition and general assistance tasks, while experts and intervention corps may also be present and interact with the system through smartphones. The Aggregate Programming paradigm, incarnated into the FCPP library, could be exploited to ensure resilient coordination of the network. In particular, a dynamic spanning forest construction and task allocation system could ensure consistent network connectivity even with limited drones and adverse meteorological conditions.

In order to concretely realise the proposed system, two independent ingredients need to be available. First, build configurations and networking drivers have to be developed for FCPP on the Android and Linux OSs. Simultaneously, the decentralised task allocation policy and "postman" behaviour should be defined and optimised through testing in the FCPP simulation framework. With these two preconditions met, a concrete deployment on a fleet of drones and smartphones could be made and assessed in a test environment, such as a plain field. After these steps of successful validation, more realistic tests could be performed, opening to the possible use in real emergency situations.

References

[1] S. E. Mueller, A. E. Thode, E. Q. Margolis, L. L. Yocom, J. D. Young, and J. M. Iniguez, "Climate relationships with increasing wildfire in the southwestern us from 1984 to 2015," *Forest Ecology and Management*, vol. 460, p. 117861, 2020.

[2] A. L. Westerling, H. G. Hidalgo, D. R. Cayan, and T. W. Swetnam, "Warming and earlier spring increase western u.s. forest wildfire activity," *Science*, vol. 313, no. 5789, pp. 940–943, 2006.

[3] D. McKenzie, Z. Gedalof, D. L. Peterson, and P. Mote, "Climatic change, wildfire, and conservation," *Conservation Biology*, vol. 18, no. 4, pp. 890–902, 2004.

[4] J. T. Abatzoglou and A. P. Williams, "Impact of anthropogenic climate change on wildfire across western us forests," *Proceedings of the National Academy of Sciences*, vol. 113, no. 42, pp. 11 770–11 775, 2016.

[5] M. Calviño-Cancela, M. L. Chas-Amil, E. D. García-Martínez, and J. Touza, "Interacting effects of topography, vegetation, human activities and wildland-urban interfaces on wildfire ignition risk," *Forest Ecology and Management*, vol. 397, pp. 10–17, 2017.

[6] G. Amatulli, F. Peréz-Cabello, and J. de la Riva, "Mapping lightning/human-caused wildfires occurrence under ignition point location uncertainty," *Ecological Modelling*, vol. 200, no. 3, pp. 321–333, 2007.

[7] J. Beal, D. Pianini, and M. Viroli, "Aggregate programming for the internet of things," *IEEE Computer*, vol. 48, no. 9, pp. 22–30, 2015.

[8] G. Audrito, "FCPP: an efficient and extensible field calculus framework," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2020, Washington, DC, USA, August 17-21, 2020*. IEEE, 2020, pp. 153–159.

[9] G. Audrito, L. Rapetta, and G. Torta, "Extensible 3d simulation of aggregated systems with FCPP," in *24th International Conference on Coordination Models and Languages (COORDINATION)*, ser. Lecture Notes in Computer Science, vol. 13271. Springer, 2022, pp. 55–71.

[10] J. Beal, S. Dulman, K. Usbeck, M. Viroli, and N. Correll, "Organizing the aggregate: Languages for spatial computing," in *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*. IGI Global, 2013, pp. 436–501.

[11] M. Mamei and F. Zambonelli, "Programming pervasive and mobile computing applications: The TOTA approach," *ACM Transactions on Software Engineering Methodologies*, vol. 18, no. 4, pp. 15:1–15:56, 2009.

[12] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[13] A. Kondacs, "Biologically-inspired self-assembly of two-dimensional shapes using global-to-local compilation," in *18th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 2003, pp. 633–638.

[14] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco, "Mobile data collection in sensor networks: The TinyLime," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 446–469, 2005.

[15] C. Lasser, J. Massar, J. Miney, and L. Dayton, *Starlisp Reference Manual*. Thinking Machines Corporation, 1988.

[16] G. Audrito, M. Viroli, F. Damiani, D. Pianini, and J. Beal, "A higher-order calculus of computational fields," *ACM Trans. Comput. Log.*, vol. 20, no. 1, pp. 5:1–5:55, 2019.

[17] G. Audrito, J. Beal, F. Damiani, and M. Viroli, "Space-time universality of field calculus," in *20th International Conference on Coordination Models and Languages (COORDINATION)*, ser. Lecture Notes in Computer Science, vol. 10852. Springer, 2018, pp. 1–20.

[18] G. Audrito and G. Torta, "Towards aggregate monitoring of spatio-temporal properties," in *VORTEX 2021: Proceedings of the 5th ACM International Workshop on Verification and mOnitoring at Runtime EXecution, Virtual Event, Denmark, 12 July 2021*. ACM, 2021, pp. 26–29.

[19] G. Audrito, F. Damiani, V. Stolz, G. Torta, and M. Viroli, "Distributed runtime verification by past-ctl and the field calculus," *J. Syst. Softw.*, vol. 187, p. 111251, 2022.

[20] G. Audrito, R. Casadei, F. Damiani, V. Stolz, and M. Viroli, "Adaptive distributed monitors of spatial properties for cyber-physical systems," *J. Syst. Softw.*, vol. 175, p. 110908, 2021.

[21] G. Audrito, F. Damiani, G. M. D. Giuda, S. Meschini, L. Pellegrini, E. Seghezzi, L. C. Tagliabue, L. Testa, and G. Torta, "RM for users' safety and security in the built environment," in *VORTEX 2021: Proceedings of the 5th ACM International Workshop on Verification and mOnitoring at Runtime EXecution, Virtual Event, Denmark, 12 July 2021*. ACM, 2021, pp. 13–16.

[22] G. Audrito, R. Casadei, and G. Torta, "Fostering resilient execution of multi-agent plans through self-organisation," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2021, Companion Volume, Washington, DC, USA, September 27 - Oct. 1, 2021*. IEEE, 2021, pp. 81–86.

[23] ——, "Towards integration of multi-agent planning with self-organising collective processes," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2021, Companion Volume, Washington, DC, USA, September 27 - Oct. 1, 2021*. IEEE, 2021, pp. 297–298.

[24] M. Viroli, G. Audrito, J. Beal, F. Damiani, and D. Pianini, "Engineering resilient collective adaptive systems by self-stabilisation," *ACM Transactions on Modelling and Computer Simulation*, vol. 28, no. 2, pp. 16:1–16:28, 2018.

[25] D. Abrahams and A. Gurtovoy, *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series)*. Addison-Wesley Professional, 2004.

[26] D. Pianini, M. Viroli, and J. Beal, "Protelis: practical aggregate programming," in *30th ACM Symposium on Applied Computing (SAC)*. ACM, 2015, pp. 1846–1853.

[27] M. Viroli, R. Casadei, and D. Pianini, "Simulating large-scale aggregate MASs with Alchemist and Scala," in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, ser. Annals of Computer Science and Information Systems, vol. 8. IEEE, 2016, pp. 1495–1504.

[28] M. D. McIlroy, J. Buxton, P. Naur, and B. Randell, "Mass-produced software components," in *1st international conference on software engineering*, 1968, pp. 88–98.

[29] H. I. Cannon, "Flavors: A non-hierarchical approach to object-oriented programming," Artificial Intelligence Laboratory, MIT, USA, Tech. Rep., 1979.

[30] G. Bracha and W. R. Cook, "Mixin-based inheritance," in *International Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA) / European Conference on Object-Oriented Programming (ECOOP)*. ACM, 1990, pp. 303–311.

[31] H. Liang, M. Zhang, and H. Wang, "A neural network model for wildfire scale prediction using meteorological factors," *IEEE Access*, vol. 7, pp. 176 746–176 755, 2019.

[32] A. Jaafari, E. K. Zenner, M. Panahi, and H. Shahabi, "Hybrid artificial intelligence models based on a neuro-fuzzy system and metaheuristic optimization algorithms for spatial prediction of wildfire probability," *Agricultural and Forest Meteorology*, vol. 266-267, pp. 198–207, 2019.

[33] A. Alexandridis, D. Vakalis, C. Siettos, and G. Bafas, "A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through Spetses island in 1990," *Applied Mathematics and Computation*, vol. 204, no. 1, pp. 191–201, 2008.

[34] S. W. Taylor, D. G. Woolford, C. B. Dean, and D. L. Martell, "Wildfire Prediction to Inform Fire Management: Statistical Science Challenges," *Statistical Science*, vol. 28, no. 4, pp. 586 – 615, 2013.

[35] J. Mandel, M. Chen, L. P. Franca, C. Johns, A. Puhalskii, J. L. Coen, C. C. Douglas, R. Kremens, A. Vodacek, and W. Zhao, "A note on dynamic data driven wildfire modeling," in *Computational Science - ICCS 2004*, M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 725–731.

[36] O. Ghorbanzadeh, K. Valizadeh Kamran, T. Blaschke, J. Aryal, A. Naboureh, J. Einali, and J. Bian, "Spatial prediction of wildfire susceptibility using field survey gps data and machine learning approaches," *Fire*, vol. 2, no. 3, 2019.

[37] V. Slavkovikj, S. Verstockt, S. Van Hoecke, and R. Van de Walle, "Review of wildfire detection using social media," *Fire Safety Journal*, vol. 68, pp. 109–118, 2014.

[38] M. T. Rashid, Y. Zhang, D. Zhang, and D. Wang, "Compdrone: Towards integrated computational model and social drone based wildfire monitoring," in *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2020, pp. 43–50.

[39] P. F. M. Ellis, "The likelihood of ignition of dry-eucalypt forest litter by firebrands," *International Journal of Wildland Fire*, vol. 24, no. 2, pp. 225–235, 2015.

[40] Y. Mo, G. Audrito, S. Dasgupta, and J. Beal, "A resilient leader election algorithm using aggregate computing blocks," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3336–3341, 2020, 21st IFAC World Congress.

[41] G. Audrito, F. Damiani, and M. Viroli, "Optimal single-path information propagation in gradient-based algorithms," *Science of Computer Programming*, vol. 166, pp. 146–166, 2018.

[42] R. Casadei, M. Viroli, G. Audrito, D. Pianini, and F. Damiani, "Engineering collective intelligence at the edge with aggregate processes," *Eng. Appl. Artif. Intell.*, vol. 97, p. 104081, 2021.