The Need of Proper Programming Models for CPS

Martin Richter, Christine Jakobs, Matthias Werner
Operating Systems Group
Chemnitz University of Technology
09111 Chemnitz, Germany
email: {martin.richter, christine.jakobs, matthias.werner}@informatik.tu-chemnitz.de

***Abstract*—Mobile cyber-physical systems consist of possibly moving heterogeneous execution units, which interact with their environment through sensors and actuators. Programming such systems without taking motion into account has already proven to be error-prone and complex, as challenges like communication or programming multiple different devices have to be considered by the developer. Corresponding programming models abstract from these challenges through the provision of transparencies. This allows the programmer to focus on describing the behavior of the system instead of managing its infrastructure. When mobility is taken into account, the devices tasks may depend on their positions in space. Therefore, location and motion awareness have to be supplied. This impedes the provision of distribution transparency, as the programmer has to consider the movement and positioning of certain objects. In contrast to supporting awareness, providing motion and location transparency allows to maintain distribution transparency. In exchange, this limits the developers ability to consider the positioning and movement of the devices. Therefore, a programming model, which bridges the gap between maintaining transparencies and providing awareness is required to enable the developer to focus on describing the behavior of the possibly mobile system as a whole. This paper aims to show that there is a need for research on such models. To achieve this goal, a systematic literature review is performed. Its main target is the assessment of existing programming models, regarding their provided types of awareness and transparency. To classify on which aspects of the system the considered programming models focus, an architectural model for mobile cyber-physical systems is introduced. Additionally, desired programming model properties are defined with respect to the presented architectural model. This allows to determine, in which way the considered approaches fail or succeed in handling the described challenges. Therefore, a conclusion on the need of programming models for mobile cyber-physical systems can be drawn.**

Keywords*—cyber-physical systems; distribution; mobility; programming models; context awareness.

I. Introduction

In the modern world, an increasing number of diverse computational units are interconnected. Especially in the context of the Internet of Things and Industry 4.0, this plays an important role. These devices communicate with each other in order to exchange data or coordinate their actions. An integration of sensors and actuators leads to the emergence of Cyber-Physical Systems (CPS), which consist of heterogeneous execution units, interacting with their physical environment. Sensors allow them to recognize the systems surroundings digitally. Based on this information, the devices control actuators to influence physical objects or phenomena. Therefore, a control loop is created, which incorporates the execution units (including sensors and actuators) as well as their physical environment.

CPS are used for a vast array of different tasks. Automated production lines [1], field fertilization by drones [2] and warehouse logistics management by robots [3] are just a few examples for this. In this context, considerations on mobility become increasingly important, as execution units (e.g., robots) and physical objects in their surroundings (e.g., goods being carried) may move.

In classic distributed systems, movements and locations of the execution units are hidden from the programmer through motion and location transparency, which are provided by the operating system or middleware. The deployment of these transparencies may become less beneficial, when the interactions of moving devices with their physical surrounds are considered. This makes maintaining distribution transparency more difficult. Positions and movements of single execution units and physical objects may have to be observed or even controlled by the programmer to obtain the intended behavior of the system. In this case, distribution transparency is still desired, as error-prone tasks like inter-device communication and coordination should be abstracted. This implies, that approaches have to be considered, which maintain distribution transparency, while providing support for motion and location awareness.

Programming models allow to take an abstract view on how these properties may be implemented. They describe the developers view on the system as well as its internal interactions. These interactions depend on the systems architecture. For that reason, we present an architectural model, which incorporates classic properties of CPS as well as the mobility of devices and physical events. This allows us to evaluate on which aspects of the system current programming models focus. On the basis of the architectural model, we define desired programming model properties to assess how present proposals tackle the mentioned challenges. We conduct this assessment through a Systematic Literature Review (SLR). Therefore, we are able to methodically inspect existing programming models with respect to the proposed architectural model and the desired programming model properties.

In Section II, the architectural model for mobile CPS is described. The desired programming model properties are presented in Section III. The methodology of the SLR is described in Section IV. In Section V, its results are analyzed and presented. Finally, a conclusion is drawn and topics for future work are proposed in Section VI.

 ISBN: 978-1-61208-811-2

II. Architectural Model

A layered architectural model for mobile CPS (see Figure 1) allows to view them in an abstract way. This is necessary, as the system encompasses the interaction of various fundamentally different entities, such as execution units, applications and physical phenomena.

Its lowest layer represents the execution units. Sensors allow them to continuously gather data about physical objects and phenomena in their proximity. Actuators enable them to influence their environment corresponding to the collected information. Heterogeneity is of utter importance in this context, as the devices possess different capabilities and therefore might have to cooperate to solve a given problem. Additionally, the gathered data of different execution units may correspond to the same observed physical object or phenomenon, but differs due to varying sensors being available on the physical nodes.

The environmental data layer abstracts from this heterogeneous view of the single execution unit. This is achieved by aggregating the gathered data based on its location and the corresponding physical phenomenon. Therefore, logical representations of multiple phenomena may be created, which leads to a global, more precise view on the physical world. This makes it possible to coordinate different devices to react to physical phenomena, even if they do not possess the capabilities to observe them.

The systems reaction to its observations is described in the application layer. It represents applications being executed on the different physical nodes. Those use the environmental data layer as a foundation for their calculations. The main goal of an application is to control the devices actuators to influence the environment. This is done according to the existing virtual image of the physical surroundings. Therefore, coordination and considerations on the heterogeneity of execution units are essential on this layer, as multiple different nodes might have to act synchronously to solve a common problem.

III. Programming Model Properties

The described architectural model (see Section II) allows to view CPS in an abstract way. Programming models describe the interactions between the different architectural layers in an implementation-independent way. These models may be used as a basis for implementing a corresponding middleware or operating system and therefore, are well suited to be used as a first step towards further considerations.

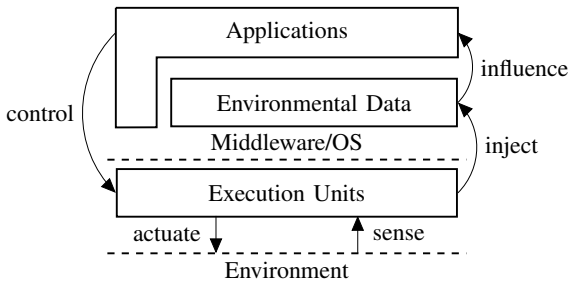

Figure 1. Overview of the architectural model.

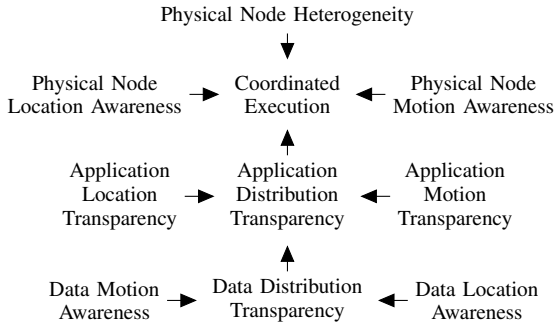

Figure 2. Relations of the relevant programming model properties.

As already stated in Section I, the focus of this work lies on mobility and distribution in CPS. Programming models may view both properties in different ways (i.e., through transparency or awareness). The provided architecture allows to consider them separately on the distinct layers of the system. Thus, the conflict of providing distribution transparency as well as location and motion awareness to the programmer (see Section I) can be circumvented.

When considering the discussed architecture, different kinds of motion have to be regarded for physical phenomena, execution units and applications. Through the implemented programming model, the programmer may not recognize the movement of some of these entities at all through motion transparency. Motion awareness contrasts this. It allows the developer to either observe the movement of these entities through passive motion awareness or to influence it through active motion awareness. When considering motion, location awareness has to be given, as motion is a change of location over time.

When regarding the different architectural layers, support of motion awareness (actively or passively) on the execution unit and environmental data layers has to be provided. This has to be considered to ensure correct interaction between possibly moving physical phenomena and execution units. On the application layer, motion and location transparency have to be deployed to provide distribution transparency. The programmer should not have to address execution units based on their locations and motion but rather on their properties and whether they are located close to physical phenomena of interest. This is achieved through allowing the developer to create one application, which is distributed to the corresponding nodes at run-time, depending on their capabilities and information gathered in the environmental data layer. Distribution transparency is required on the environmental data layer as well, as data has to be aggregated to a digital representation of the physical world. This allows all nodes to have an identical view on the observed entities. Therefore, a collective decision on which execution units tackle the corresponding tasks can be made. Location and motion awareness are necessary on this layer to differentiate between multiple different phenomena taking place simultaneously at different locations. Figure 2 illustrates the described programming model properties and their relationship to each other.

 ISBN: 978-1-61208-811-2

IV. Research Methodology

In this section, the approach for planning and conducting the SLR is presented. The SLR is based on the proposals of Kitchenham [4] as well as Biolchini, Mian, Natali and Travassos [5]. Its main goal is to inspect, whether there is a need for research on programming models for mobile CPS with respect to the presented challenges (see Section I). Therefore, contributions focusing on properties regarding distribution, location and motion awareness or transparency are examined as a first step. Figure 3 gives an overview of the research methodology. The need for a SLR is discussed in Section I. Additionally, a search for existing reviews on the presented topic is performed on *Google Scholar*. The search query encompasses keywords to identify SLRs on programming models for mobile distributed systems with regard to motion and location awareness as well as transparency.

None of the found articles are related to the presented topic. Therefore, the conduction of a SLR is needed. The research questions, defined in Table I, lay the foundation for finding, selecting and analyzing relevant contributions. They directly refer to the presented properties of programming models in correspondence to our architectural model (see Sections II and III). These research questions allow us to decide, whether the inspected approaches sufficiently tackle the described challenges. Thus, we can determine whether a need for research exists on programming models for mobile CPS.

A. Search Strategy

The selection of an initial set of papers is performed on basis of the following search strategy. Multiple trial searches are carried out on *Google Scholar* to identify, whether the results contain relevant articles. The research questions are taken as a basis for the keywords, used in the corresponding search string. If not enough relevant papers are included in the results, the search string is altered accordingly. Additionally, negative keywords are added to exclude articles regarding other domains, which may use identical technical terms with meanings

TABLE I
Research questions for performing the SLR.

ID	Research Question	Target
RQ 1	What are the used abstractions to provide distribution, location and/or motion transparency on the different architectural layers?	To get an overview over the used abstractions for choosing one or more of them for an incorporation into a new or existing programming model.
RQ 2	What approaches are used to provide location and/or motion awareness on the different architectural layers?	To get insight into the means to provide location and motion awareness for an incorporation into a new or existing programming model.
RQ 3	What kind of problems are solved by the programming models?	To identify, whether the solved problems are correlated to the challenges discussed in this work.

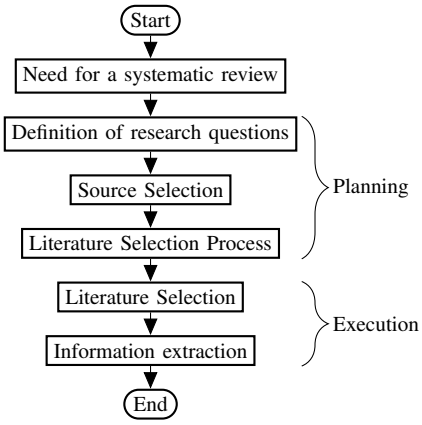

Figure 3. Approach for conducting the SLR [4], [5].

differing from the presented definitions. The following search string is the final result of the trial searches and alterations:

("programming model" AND ("distributed computing" OR "distributed systems") AND ("mobility" OR "mobile systems") AND "computer science" -android -ios -"artificial intelligence" -"high performance computing" -multimedia -web -openmp -mpi -energy)

The initial set of articles, obtained through the use of this search string on *Google Scholar*, is used for the conduction of the SLR.

B. Review Protocol

The review protocol is constituted of the following aspects:

- *Google Scholar* is selected as a resource, as in a comparison with searches on *IEEE Xplore*, *Springer Link*, *ACM Digital Library* and *Science Direct*, *Google Scholar* provided the same results in addition to articles only being available in smaller digital libraries.
- The search method is based on a keyword search through the provided web search engine of *Google Scholar*.
- Papers focusing on programming models for mobile distributed systems constitute the population.
- Goal of the intervention is the comparison of existing programming models with respect to the presented architectural model and the desired programming model properties (see Sections II and III).

The article inclusion criteria are shown in Table II. They are directly connected to the research questions, presented in Table I. Articles are excluded, if they focus on optimization, energy efficiency, latency or other topics not directly related to the programming model, regarding distribution, location and motion.

C. Article Selection Procedure

The initial set of papers is filtered on the basis of the presented inclusion and exclusion criteria. First, all titles are read and it is decided, if the articles match the described domain (see Section I). For the remaining papers the abstract is read. If it is obvious, that the article matches the presented criteria, it is kept for the final set of papers. Otherwise, the

 ISBN: 978-1-61208-811-2

TABLE II
DEFINITION OF LITERATURE INCLUSION CRITERIA.

ID	Inclusion criteria
IC 1	The article describes the used abstractions of a programming model or middleware for mobile distributed systems.
IC 2	The article considers abstracting from or being aware of location.
IC 3	The article considers abstracting from or being aware of motion.
IC 4	The article considers distribution transparency.

introduction and conclusion are examined. If no clear decision can be deduced from the introduction and conclusion, the whole article is read and kept for or removed from the final list accordingly.

The initial set of papers is constituted of 452 articles, which is reduced by 236 through reading their titles and removing duplicates. The remaining results are reduced to a quantity of 19 articles by examining their abstracts and possibly their introductions and conclusions. Three more papers are removed by reading the whole text. Therefore, 16 articles remain in the final set of contributions to be reviewed.

D. Information Extraction

The information inclusion and exclusion criteria (see Table II) directly correspond to the presented research questions. Thus, the papers are examined for approaches to distribution transparency, location transparency and awareness, and motion transparency and awareness. This is done with respect to the presented architectural model (see Section II), as varying properties may be regarded on different architectural layers. Additionally, the discussed problems are inspected to decide in which way the contributions are related to the proposed challenges of our paper. The following paragraphs elaborate on the articles and the programming model properties.

Four proposals focus on the application layer. They mainly differ in their approach to providing distribution transparency. In [9] and [10], the developer takes the view of programming a given spatial region itself, instead of different nodes. In [9], this is achieved through creating an automaton for a static

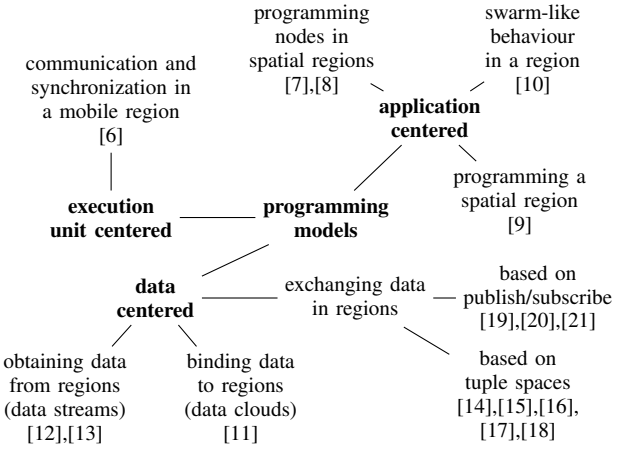

Figure 4. Categorization of the different research articles.

spatial region, which is emulated round-robin wise by the execution units, residing in it. Location awareness is provided on the application layer as well, as the programmer decides, in which spatial region a program executes. This implies, that location awareness is deployed on the execution unit level as well, as devices have to decide in which region they are situated. As they may move arbitrarily between regions (i.e., without control of the programmer), passive motion awareness is supported on this layer as well. In [10], swarm-like behavior in a static given region is employed, as one program is distributed over the corresponding physical nodes. Those perform calculations, based on their own state and the state of neighboring devices. Therefore, the execution of the different program instances converges. Distribution transparency and location awareness are provided on the application and execution unit layers for the same reasons as in [9]. Motion transparency is deployed on the application and execution unit layer, as movement of devices is not considered to be impactful. In [10], the environmental data layer is also regarded. Data is spreading from device to device through their neighborhood-based calculations. Therefore, distribution transparency, location transparency and motion transparency are maintained on this level. In [8] and [7], an application is created, which migrates between devices, based on their positions in space and their capabilities. Therefore, distribution transparency, location awareness and active motion awareness are provided on the application layer. On the execution unit layer active location awareness and passive motion awareness are deployed, as devices may move arbitrarily but have to exchange information about their positioning to decide, which computational unit executes the application.

Eleven approaches focus directly on the environmental data layer. In [13] and [12], the gathering of data from spatial regions is discussed. In [13], an application is executed distributively on spatially distributed execution units with the goal of aggregating data, situated on them. This implies the provision of distribution transparency on the application and environmental data layer. The information is accumulated depending on the positioning of devices. Therefore, location awareness is deployed on all layers. As positions of execution units are viewed as static information, motion transparency is maintained on every architectural level. In [12], an approach to represent data from spatially distributed devices as data streams is presented. As data is not aggregated, and distribution of it is maintained, no distribution transparency is provided on the environmental data layer. Location awareness is supported on the environmental data and the execution unit layers, as information is accessed based on its location and the point in time it was gathered. Since devices may move arbitrarily and the corresponding motion of information can be seen as a change of its location over time, passive motion awareness is maintained on both lower layers as well.

In [11], an approach to bind data to regions, instead of devices, is presented. Information corresponds to physical phenomena, which are sensed by surrounding execution units. The data is aggregated to create a more precise virtual im-

age of the corresponding phenomena. Therefore, distribution transparency is maintained on the environmental data layer. Devices may move arbitrarily and spread the aggregated data according to the locations of the observed phenomena and agendas provided by the programmer. Therefore, location awareness and active motion awareness are deployed on the environmental data layer. The former is also provided on the execution unit level to enable the devices to decide, where physical phenomena are located. Passive motion awareness is maintained on this layer, as devices may move arbitrarily without the influence of the programmer.

Eight suggestions focus on the exchange of information inside spatial regions. They can be divided into approaches, based on the publish/subscribe paradigm or tuple spaces. In [20], [19] and [21], the publish/subscribe paradigm is extended to allow to constrain publishers and subscribers to only exchange data, if they are in a given proximity to each other. As no information about the location of information itself is given, location and motion transparency are supported on the environmental data layer. On the execution unit layer, location and passive motion awareness are deployed, as devices have to decide, whether they are in proximity to each other. Motion can be observed, but not influenced by the devices through the change of their locations and subscriptions to given topics.

Five approaches use tuple spaces to exchange information in spatial regions. In [14] and [15], each device hosts a tuple space and tuples may spread to other execution units based on given rules concerning directions and distances in space. As data is viewed in the form of single distributed tuples, spread across all devices, no distribution transparency is maintained on the environmental data layer. Location and active motion awareness are provided on this level, as the spread of information can be controlled, based on the positioning of devices. Therefore, location awareness is deployed on the execution unit layer as well. Additionally, passive motion awareness is provided, as execution units perceive their change of location without having influence on it. In [17], every device hosts a tuple space, similar to [14] and [15]. Additionally, tuples and reading operations are associated with geometric shapes. Only if the shape of the reading operation and the shape of the corresponding tuple intersect, information may be exchanged. Thus, location awareness is provided on the environmental data layer. Motion transparency is maintained on this level, as information may only be read from devices with the required relative positioning. Since devices may move in formation, the motion of data cannot be observed. On the execution unit layer, location and passive motion awareness are provided, as devices have to perceive their location in order to decide whether the geometric shapes of reading operations and tuples intersect. Additionally, they perceive their motion as a change of location, without having influence on it. In [18], tuple spaces are attached to logical mobile units, which may migrate between devices. Tuples may be exchanged between them, based on tuple space operation annotations regarding the logical units identities, regardless of their location. Therefore, location and motion transparency are provided on all layers. In [16], the developer creates multiple logical mobile units (i.e., agents), which perceive their environment through views. Each view contains data from other devices in a given radius around the current execution unit of the corresponding agent. Therefore, location and motion transparency are given on the application layer, as agents move with their devices or migrate between them arbitrarily and do not perceive their own location. On the environmental data layer distribution transparency is provided, as data from different devices is accessed similarly through the described views. Through the view constraints, regarding the positioning of devices, location awareness is maintained as well. Motion transparency is provided, as agents may migrate between devices, taking their views with them. Therefore, the set of data in its proximity changes, without actual information on whether the device moved, the agent migrated or other execution units or logical mobile units in its proximity moved.

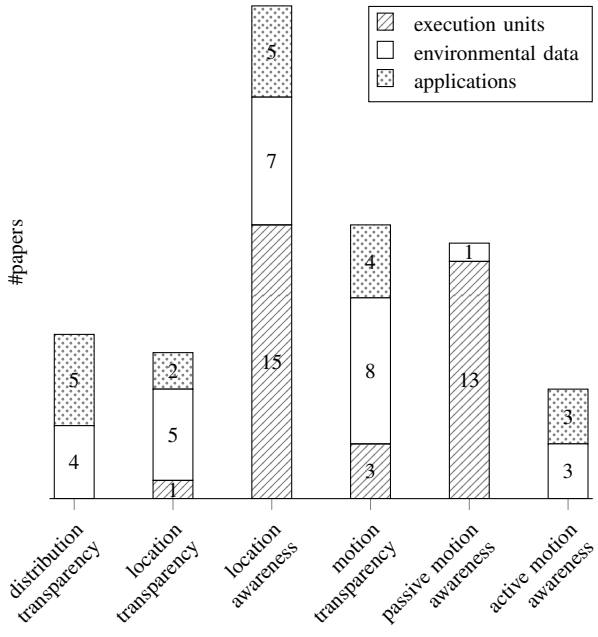

Figure 5. Discussed programming model properties in the reviewed papers.

One approach focuses directly on the execution unit layer. In [6], the programmer determines a region in space, whose extend and location may change, based on given functions. Inside the region devices may exchange messages. Programs are executed step-wise on the execution units. Whenever an according message is received, the execution of a step is triggered. Based on this, active motion awareness is provided on the application layer, as the programmer influences the movement of the region. Devices perceive their location to decide, whether they are located in a given region, hence location awareness is deployed on the execution unit layer. Additionally, passive motion transparency is provided, as execution units move arbitrarily between regions.

Figure 5 shows the final results of the information extraction. It illustrates the number of papers focusing on the given programming model properties for each architectural layer. The considered problems and which architectural level the proposal directly concentrates on are illustrated in Figure 4.

 ISBN: 978-1-61208-811-2

V. Analysis

As already described in Section IV-D, the reviewed programming models focus on different layers of the presented architectural model (see Section II). None of them is concerned with the interaction across all levels of the system, according to the given requirements (see Section III). Therefore, a composition of programming models or the creation of a new model is required. It is evident, that most approaches do not suite the presented needs, as they provide motion or location transparency on the execution unit or environmental data layers, or do not support distribution transparency on these layers.

On the execution unit layer *Autonomous Virtual Mobile Nodes* by Dolev, Gilbert, Schiller, Shvartsman and Welch [6] fulfills all requirements, as it supports location aware communication and synchronization between devices, residing in a mobile region. Therefore, it provides a first building block for further considerations on the higher layers. On the environmental data layer *Hovering Data Clouds* by Ebers et al. [11] fits the described needs. It provides distribution transparency through location aware data aggregation, regarding mobile physical events or phenomena. This directly corresponds to the described programming model properties for this layer. No programming model fulfills the requirements for the application layer, as they only support the programming of devices in static spatial regions. Therefore, if physical objects or phenomena leave the region, devices stop interacting with them according to the developers intentions. This implies, that alterations regarding mobile spatial regions have to be performed or a new model considering the application layer has to be created.

VI. Conclusion and future work

An architectural model for mobile CPS was presented in this paper. On its basis programming model properties regarding distribution, location and motion of physical phenomena, execution units and applications were defined. These considerations laid the foundation for the inspection of existing contributions, regarding the provision of the mentioned programming model properties. The examination of current approaches was performed through the conduction of an SLR.

The obtained results provide evidence, that existing programming models do not bridge the gap between providing distribution transparency to the programmer and supporting the motion and location aware execution of applications. Therefore, a need for research exists on how to incorporate these properties into one programming model.

As a next step, the results of this paper will be used to create an architecture in which further research will be applied. Topics for future work include considerations on the composition of existing models or the creation of a new programming model, as this may provide a more complete view on CPS to the developer. Further inspections on the formalization of such models are required to ensure the correct behavior of the system under any circumstances. In the context of CPS, the heterogeneity of devices in combination with concurrency is another subject for future work, as multiple different execution units might have to cooperate synchronously to interact with their physical environment to reach a common goal.

References

[1] G. Fragapane, D. Ivanov, M. Peron, F. Sgarbossa, and J. O. Strandhagen, "Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics," *Ann. of Operations Res.*, pp. 1–19, 2020.

[2] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *2015 Int. Conf. on Intell. Environments*, 2015, pp. 140–143.

[3] F. Basile, P. Chiacchio, and J. Coppola, "A cyber-physical view of automated warehouse systems," in *2016 IEEE Int. Conf. on Automat. Science and Eng. (CASE)*, 2016, pp. 407–412.

[4] B. Kitchenham, "Procedures for performing systematic reviews," Keele Univ., Tech. Rep. TR/SE-0401, 2004.

[5] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, "Systematic review in software engineering," COPPE/UFRJ PESC, Tech. Rep. RT-ES 679-05, 2005.

[6] S. Dolev, S. Gilbert, E. Schiller, A. A. Shvartsman, and J. Welch, "Autonomous virtual mobile nodes," in *Proc. of the 2005 Joint Workshop on Found. of Mobile Comput.*, ser. DIALM-POMC '05. ACM, 2005, pp. 62–69.

[7] C. Borcea, C. Intanagonwiwat, P. Kang, U. Kremer, and L. Iftode, "Spatial programming using smart messages: design and implementation," in *24th Int. Conf. on Distrib. Comput. Syst.*, 2004, pp. 690–699.

[8] Y. Ni, U. Kremer, and L. Iftode, "Spatial views: Space-aware programming for networks of embedded systems," in *Lang.s and Compilers for Parallel Comput.* Springer, 2004, pp. 258–272.

[9] S. Dolev, S. Gilbert, L. Lahiani, N. Lynch, and T. Nolte, "Virtual stationary automata for mobile networks," MIT CSAIL, Tech. Rep. MIT-LCS-TR-979, 2005.

[10] J. Beal and J. Bachrach, "Infrastructure for engineered emergence on sensor/actuator networks," in *IEEE Intell. Syst.*, vol. 21, 2006, pp. 10–19.

[11] S. Ebers, S. P. Fekete, S. Fischer, H. Hellbrück, B. Hendriks, and A. Wegener, "Hovering data clouds for organic computing," in *Organic Comput. — A Paradigm Shift for Complex Syst.*, 2011, pp. 221–234.

[12] S. Imai and C. A. Varela, "A programming model for spatio-temporal data streaming applications," in *Procedia Comput. Science*, 2012, pp. 1139–1148.

[13] R. Newton, G. Morrisett, and M. Welsh, "The regiment macroprogramming system," in *2007 6th Int. Symp. on Inf. Process. in Sensor Networks*, 2007, pp. 489–498.

[14] M. Mamei, F. Zambonelli, and L. Leonardi, "Tuples on the air: a middleware for context-aware computing in dynamic networks," in *23rd Int. Conf. on Distrib. Comput. Syst. Workshops, 2003. Proc..*, 2003, pp. 342–347.

[15] M. Viroli, D. Pianini, and J. Beal, "Linda in space-time: An adaptive coordination model for mobile ad-hoc environments," in *Coordination Models and Lang.* Springer, 2012, pp. 212–229.

[16] C. Julien and G.-C. Roman, "Egocentric context-aware programming in ad hoc mobile environments," in *Proc. of the 10th ACM SIGSOFT Symp. on Found.s of Softw. Eng.* ACM, 2002, pp. 21–30.

[17] J. Pauty, P. Couderc, M. Banatre, and Y. Berbers, "Geo-linda: a geometry aware distributed tuple space," in *21st Int. Conf. on Adv. Inf. Networking and Appl.s (AINA '07)*, 2007, pp. 370–377.

[18] A. Murphy, G. Picco, and G.-C. Roman, "LIME: a middleware for physical and logical mobility," in *Proc. 21st Int. Conf. on Distrib. Comput. Syst.*, 2001, pp. 524–533.

[19] P. T. Eugster, B. Garbinato, and A. Holzer, "Location-based publish/subscribe," in *Fourth IEEE Int. Symp. on Network Comput. and Appl.*, 2005, pp. 279–282.

[20] R. Meier and V. Cahill, "On event-based middleware for location-aware mobile applications," in *IEEE Trans. on Softw. Eng.*, vol. 36, 2010, pp. 409–430.

[21] L. Fiege, F. C. Gartner, O. Kasten, and A. Zeidler, "Supporting mobility in content-based publish/subscribe middleware," in *Middleware 2003*. Springer, 2003, pp. 103–122.

 ISBN: 978-1-61208-811-2