Spaces-Based Communication: an Extension to Support Infinite Spatial Models

Diego Bernini, Francesco Fiamberti, Daniela Micucci, Francesco Tisato, Alessio Vertemati
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Milano, Italy
Email: {diego.bernini, francesco.fiamberti, daniela.micucci, francesco.tisato, alessio.vertemati}@disco.unimib.it

***Abstract*—Space Integration Services (SIS) is a software communication platform that enables the seamless integration of sensors, actuators, and application-logic components through a multi-space model and a spaces-based publish/subscribe mechanism. The underlying model is based on finite spaces only. SIS has been extended in order to add support for spaces with an infinite number of locations, (e.g., spaces described by continuous coordinates, such as geodetic or Cartesian spaces). The existing conceptual model, valid for finite spaces, is reviewed and generalized to infinite spaces. The obtained model is tested using a prototype implementation realized by means of an additional layer on top of the finite-space version. Finally, the performance of the prototype is then compared to the one of the finite-space version in a series of experimental tests.**

Keywords*—*Infinite spatial models; spaces-based communication; software architecture; component-based architecture.

I. INTRODUCTION

Pervasive computing [1] aims at simplifying the everyday life through digital environments that are sensitive, adaptive, and responsive to the user's needs. A pervasive computing system requires the perception of the context in which the user operates to provide a richer and expanded mode of interaction, in addition to an intelligence for performing actions on the environment. From a technological point of view, pervasive computing relies on responsive environments. The term responsive environment [2] refers to physical environments enhanced by input devices (e.g., sensors or cameras) and output devices (e.g., displays, lights, motors). Input devices capture stimuli from the environment, whereas output devices execute actions on the environment given a predefined set of commands.

Responsive environments are, therefore, able to perceive and respond to users thanks to the presence of a computer system that receives data from the sensors (input stream) and sends commands to the actuators (output stream). For example, an application may locate the user onto the cartographic representation of the city and may also receive data from light sensors (input stream). On the basis of established rules, it could send commands to the street lights (output stream). This simple example emphasizes how responsive environments require establishing information flows from the devices to the applications and vice versa.

Space Integration Services (SIS) [3] is a software communication platform that enables the seamless integration of sensors, actuators, and application-logic components through a multi-space model and a spaces-based publish/subscribe mechanism. It provides various spatial models that can be used by applications to represent location-related information in order to support complex representations of the environment with a focus on location-aware systems. In this regard, different spatial representations can be put in appropriate correspondence to describe the localization according to different visions of the environment. The spatial representations supported by SIS are symbolic models (graphs and name spaces) and grid models (bi- and tri-dimensional). With the increase of accuracy and precision of localization sensors [4] and with the need to include geographical-related spatial models, a revision of the platform was required in order to deal with spatial representations that contain a potentially infinite number of locations, for example those described by continuous coordinates, such as geodetic and Cartesian spaces, but also unbounded grids described by discrete coordinates.

The paper is organized as follows. Section II presents the SIS conceptual model and the proposed extension for the inclusion of infinite spaces. Section III discusses the application of the model to a simplified case study. Section IV describes a prototype implementation used to test the extension. Section V presents the results of several tests aimed at estimating the performance of the proposed extension with respect to the existing SIS implementation. Section VI reviews related works. Finally, conclusions and future developments are presented in Section VII.

II. CONCEPTUAL MODEL

The previous conceptual model was based on the assumption that *spaces* are finite sets of *locations* built from *spatial models* (e.g., graph spatial model). Non- empty sets of locations belonging to a space are named *spatial contexts*. In such a model, locations play a crucial role since the existence of both a space and a context is subject to the existence of the set of locations that constitutes them (space and context). In order to handle infinite spaces, that is, spaces that contain a potentially infinite number of locations, the conceptual model has been revised around the concept of spatial context.

A. Space

A *space* is a set of *potential location*s, that are all the locations that could be theoretically considered in that space, not only the ones actually used explicitly. For example, in a Cartesian space only a finite number of points (i.e., locations) can be actually used explicitly by a real application, but theoretically every point in $\mathbb{R}^2$ could be used by the application.

A *spatial model* defines 1) the types of locations that may belong to the spaces created by that model, 2) the way in which locations are arranged, and 3) at least one premetric that can be applied to a pair of locations. The premetric defines the distance between two locations as a positive, non-zero number if the two locations are distinct, and zero if the locations are the same. A single location might be, for example, a node of a graph, a symbolic label or a numeric coordinate. Two examples of spatial models for infinite spaces are (1) the Geodetic spatial model, defining locations as geodetic coordinates (i.e., latitude and longitude) in a geodetic coordinate reference system [5]; (2) the n-dimensional Cartesian spatial model, that models an Euclidean space $\mathbb{R}^n$. In this model, locations are represented by ordered tuples of real numbers in a orthogonal Cartesian reference system. These examples also highlight that a location in an infinite space can be identified with a potentially infinite precision, as in the nature of the coordinate system based on real numbers.

B. Spatial Context

In order to handle the selection of subsets of locations in an infinite space, the concept of spatial context (context in the following), already present in the finite space-based SIS model, has been refined. A *spatial context* C_S is a subset of potential locations of a space S. It is defined by a set of *characteristic locations* in S and by a membership function that states if a given location of S belongs to the context. Essentially, the membership function is a boolean function that is true when a location is in the spatial context. According to the membership function used, the following kinds of contexts have been identified: *enumerative*, *premetric declarative*, *polygonal*, and *pure functional*.

In a *enumerative context* the set of characteristic locations is non-empty and the membership function is based on the standard belonging relationship defined in set theory. For example, given a space S, an enumerative context is defined by the set $C_S = \{l_1, l_2, ..., l_5\}$ of characteristic locations.

In a *premetric declarative context* the members are all the locations within a given distance from a given characteristic location in terms of a premetric function (see Figure 1a).

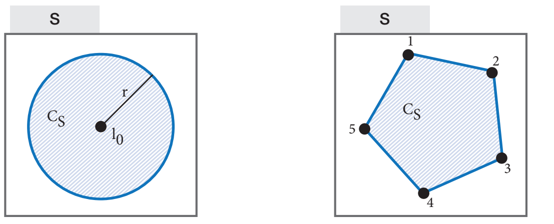
In a *polygonal context* the characteristic locations are the vertices of a polygon, and the membership function indicates the inclusion of a location in the region corresponding to the polygon itself (see Figure 1b).

Finally, in a *pure functional context* the set of characteristic locations is empty and the membership function is defined by using mathematical expressions defined in terms of the space coordinate system. For example, consider a Cartesian bi-dimensional space S. A context can be defined by the following membership function: for all locations (x, y) in S and for any given location (x_0, y_0) and (x_1, y_1),

$$f(x,y) = \begin{cases} \text{true} & \text{if } x_0 < x < x_1 \text{ and } y_0 < y < y_1 \\ \text{false} & \text{otherwise} \end{cases} \quad (1)$$

C. Projection

Related to the concept of space, is the concept of *projection*, which is widely used in geometry and in algebra.



(a) A premetric context (b) A polygonal context

Fig. 1: Different types of spatial context

Given a source spatial context C_S defined in a source space S and given a target space T, the result of a projection is a spatial context C_T on the target space T containing the locations that are obtained by applying a projection function f to all the locations in the source context C_S. The target space can be defined according to the same spatial model of the source, or to a different one. For example, a planimetry of a building is the projection of the tri-dimensional environment onto a bi-dimensional surface. Another typical example of the application of a projection is the transition from a geodesic representation of the Earth to any cartographic representation.

D. Mapping

Mappings relate different spaces. For example, a mapping can relate an area of a Cartesian space (representing the plant of a building) to a node of a graph (representing a synoptic view of the same building). Mappings are key concepts because, as it will be explained afterward, enable the communication among components, even if they rely on different spaces. Three kind of mappings have been defined: *explicit*, *projective* and *implicit*.

An *explicit mapping* is an ordered pair of contexts defined in different spaces (possibly based on different spatial models): given the two spaces S_1 and S_2 with $S_1 \neq S_2$, the ordered pair $\langle SC_1, SC_2 \rangle$ is an explicit mapping between the contexts $SC_1 \subseteq S_1$ (source) and $SC_2 \subseteq S_2$ (target).

Figure 2 shows two examples of explicit mappings between the context SC_m of the `Geo` space (a Geodetic spatial model) and the context SC_k of the `LocationNames` space (a graph spatial model), and between the context SC_i of the `PeopleIds` space (a name spatial model) and the context SC_m of `Geo`.

The target context may be defined independently of the source context. But when the target context is the result of the application of a projection to the source context, the mapping is termed *projective mapping* and is fully determined by the source context and the projection function.

Let SM be the set of all the defined explicit and projective mappings, and let SC_a and SC_b be contexts defined in different spaces. The *implicit mapping* $\langle SC_a, SC_b \rangle$ is derived if there exist k contexts $SC_1, \ldots, SC_k$ such that $\langle SC_a, SC_1 \rangle, \langle SC_1, SC_2 \rangle, \ldots, \langle SC_k, SC_b \rangle \in SM$ for $k > 1$.

In Figure 2 the dotted arrow represents an implicit mapping between the contexts SC_i of `PeopleIds` and SC_k of `LocationNames`.

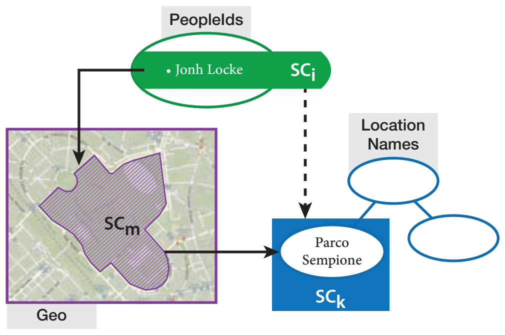


Fig. 2: Explicit and implicit mappings

E. Matching

A *direct matching* occurs when the intersection between two spatial contexts defined in the same space is not empty.

Let SC_1 and SC_2 be spatial contexts defined in different spaces. An *indirect match* between SC_1 and SC_2 occurs when there exists a mapping (explicit or implicit) $\langle SC_a, SC_b\rangle$ such that the intersections between SC_1 and SC_a and between SC_2 and SC_b are both not empty (see Figure 3).

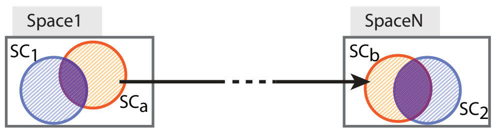


Fig. 3: Indirect matching

F. Publication and subscription

As previously introduced, SIS enables information flows relying on the publish and subscribe mechanism. A *publication* includes a thematic information (whose semantics is up to the application domain) and one or more spatial contexts; a *subscription* includes one or more spatial contexts.

When an application performs a publication, the enclosed thematic information is received by all the applications that previously performed a subscription such that at least one of its contexts *matches*, either directly or indirectly, a context of the publication. The thematic information is enriched with all the contexts that contributed to the matching.

III. Applicative Scenario

This section will apply the concepts introduced in Section II considering an exemplified scenario. Consider a parcel distribution company with warehouses distributed in Europe (for the sake of simplicity, we consider only six warehouse distributed in Italy and Spain). The company exploits vehicles to distribute parcels. Each vehicle is equipped with a device (mounting a GPS) that periodically notifies its position. Before reaching the final warehouse, vehicles can pass through intermediate warehouses. Each time a vehicle enters a warehouse, different operations have to be performed according to the country's rules, including the decision of the next warehouse the vehicle has to reach.

As depicted in Figure 4, the required spaces are: S_1, a Geodetic spatial model covering the involved countries; S_2, a graph spatial model where each node corresponds to a specific warehouse (identifiers $wh_1, wh_2, ..., wh_6$) and each arc connects the warehouses that can be reached without any intermediate stop; finally, S_3, a name spatial model containing the identifiers of the two countries $(Italy, Spain)$.

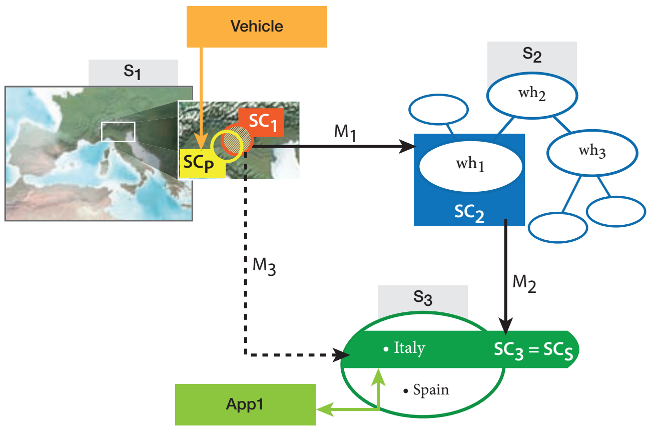


Fig. 4: Example scenario

Explicit mappings are required from S_1 to S_2 with the aim of localizing each warehouse in the Geodetic space. Mappings are in the form $M = \langle(latitude, longitude, radius), \{wh_i\}\rangle$, where the target is an enumerative context containing a node of the S_2 space (i.e., a warehouse wh_i) and the source is a premetric declarative context specifying the area in S_1 where the warehouse wh_i is located. For example, the mapping $M_1 = \langle SC_1, SC_2\rangle$, where $SC_1 = (45.523653, 9.219436, 50)$ and $SC_2 = \{wh_1\}$. In Figure 4 spatial contexts on S_1 have been hugely enlarged for visualization purposes. Moreover, explicit mappings are required from S_2 to S_3 with the aim of localizing each warehouse in its country. Mappings are in the form $M = \langle\{wh_i\}, \{country\}\rangle$, where the source is an enumerative context containing a node of the graph (i.e., the identifier of a warehouse) and the target is an enumerative context containing the identifier of the country. For example, the mapping $M_2 = \langle SC_2, SC_3\rangle$, where $SC_2 = \{wh_1\}$ and $SC_3 = \{Italy\}$. Finally, six indirect mappings are derived: their source contexts are the source contexts of the explicit mappings defined between S_1 and S_2, and their target contexts are the target contexts of the explicit mappings defined between S_2 and S_3. For example, $M_3 = \langle SC_1, SC_3\rangle$.

Two applications are required (App_1 for Italy and App_2 for Spain), each implementing the local rules. Each application subscribes to the appropriate country to be notified when a vehicle enters a warehouse in the country of competence. For example, App_1 performs a subscription to the enumerative context SC_S in S_3 containing the location $Italy$ (i.e., $SC_S = \{Italy\}$). Periodically, the vehicles make publications in the S_1 space, thus sharing their position with all the interested applications. Publications are in the form $[vehicleID, \{(latitude, longitude, radius)\}]$, where $vehicleID$ is the thematic information that identifies the vehicle, and $(latitude, longitude, radius)$ is a premetric declarative context specifying the position of the vehicle identified by $vehicleID$ in the Geodetic space. For example, publication $Pub_i = [12345, \{SC_P\}]$, where 12345 is the identifier of the vehicle that performs the publication and $SC_P = (45.51788, 9.214071, 20)$ is the location in which it has been localized.

When the vehicle 12345 makes the publication Pub_i, App_1 is notified because an indirect match occurs. Indeed, the following conditions result true: SC_P intersects SC_1, SC_1 is indirectly mapped to SC_3 (M_3 mapping), and SC_S intersects SC_3. When notified, App_1 receives the thematic information 12345 enriched with all the contexts that enabled the matching, that is, SC_P, SC_1, SC_2, SC_3, SC_S. This way, App_1 is aware of the warehouse in which the vehicle 12345 is, and, if it is able to manage S_2, it can inspect the graph in order to decide the next stop for the vehicle.

Form the above example, it is clear that spaces can be used by applications not only to enable information flows, but also to reason about spatial configurations.

IV. Implementation

A. Prototype

The developed prototype is based on the implementation of the finite-space version of Space Integration Services.

The infinite-space extension of SIS is organized according to two different software layers [6] on top of the finite-space SIS core layer. Figure 5 shows the resulting structure.

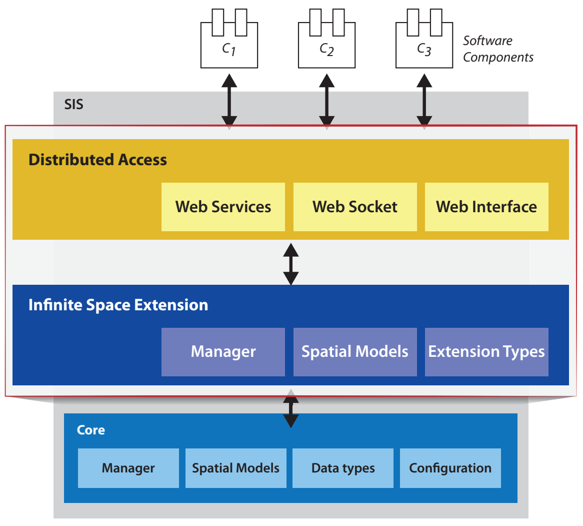


Fig. 5: The extended SIS Structure

The *Distributed Access* layer exhibits three different mechanisms allowing applications to interact with the platform. The Web Services interface provides access to the platform features by means of the Web Service Definition Language (WSDL) and of the Representational State Transfer (REST). Web Sockets [7] are a recent W3C standard for two-way asynchronous communication in the context of web applications; this technology makes available asynchronous communication of notifications to the applications. Finally, a Web interface visually exposes all the primitives of the platform and allows for the configuration of the users with their access permissions.

The *Infinite Space Extension* layer encloses the management of the new spatial models, the new data structures and the business logic to handle publications and subscriptions, the definition of contexts and the creation of mappings between contexts of different spaces.

The *Core* layer contains the finite spatial models (i.e., graph space, name space and grid space) with related primitive types and the manager in charge of monitoring instances of SIS itself. The Core layer uses the JESS rule engine in order to handle the operations of transitive closure and matching. This layer grants the full compatibility of the SIS extension with the previous versions of SIS.

The prototype has been developed in Java because the current Core layer is implemented in this language.

V. Performance evaluation

Several performance tests have been performed on the SIS prototype. For such tests, a SIS instance has been used running on an Intel Core i5 2.8 GHz PC with 4GB of RAM, running Windows(R) 7 64bit with the 64bit version of the Java Runtime Environment 1.6.33. Publications were generated by a mobile client and sent to the SIS server. The tests measured the mean reasoning time and the maximum RAM occupation. The mean reasoning time is the mean time between the reception of a publication and the moment at which notifications are made available to interested applications.

A. Mean reasoning time vs. Mappings

The first experimental setup allows studying the dependence of the mean reasoning time on the number of mappings. In this test, the mappings are created in the space configuration phase and do not change dynamically.

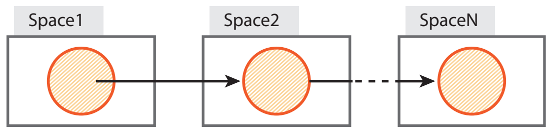


Fig. 6: Space configuration for the mapping test

The space configuration for this test consists of n Cartesian spaces, each containing a single premetric declarative context with a radius of 2 units. Every context is directly mapped onto a context in the next space, thus realizing a chain of $n \quad 1$ explicit mappings, as shown in Figure 6. Considering the implicit mappings, the total number of mappings is therefore equal to $n(n \quad 1)/2$. Publications occur on the context in the first space with a frequency of 50 Hz (i.e., every 20 ms), and a subscription is made on the context of the last space. With this generic configuration, the mean reasoning time for a publication can be measured as a function of the number of spaces n.

Figure 7 shows the measured mean reasoning time, expressed in milliseconds, for n varying from 2 to 300. Two different behaviors can be identified in the graph: for n between 2 and 90, the mean reasoning time is mostly constant (about 0.9 ms), whereas for $n > 90$ the reasoning time depends

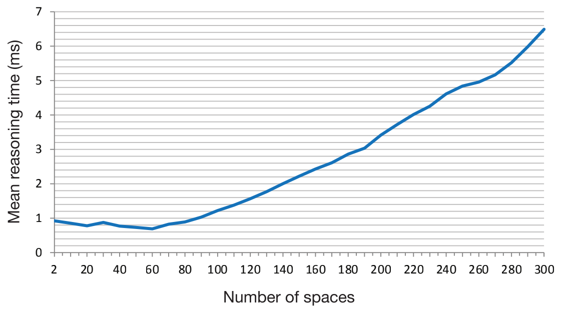


Fig. 7: Mean reasoning time vs number of spaces

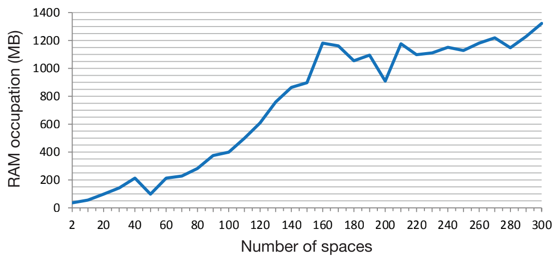


Fig. 8: Memory footprint vs number of spaces

more or less linearly on n. Figure 8 shows the RAM footprint, showing a constant increase for about half of the test and next an oscillatory behavior until the maximum value around 1.3 GB for 300 spaces. Such an amount of memory is probably due to the current implementation, which is not optimized and thus exploits a huge number of objects. The oscillatory behavior is probably due to the Java memory management.

Given these experimental results, a comparative test has been conducted against the finite-space implementation. In particular, when using the finite-space version of SIS (SIS 1), the Cartesian space has been approximated with a grid of regular cells.

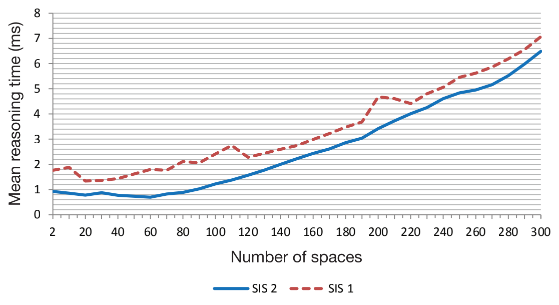


Fig. 9: SIS 2 vs SIS 1: mean reasoning time comparison

As pointed out by the graph in Figure 9, the prototype (SIS 2) is 0.7 ms faster than SIS 1 using a grid approximation. Considering the prototypical stage, the memory footprint is at acceptable levels when compared to SIS 1. Overall, an average gain of 50 MB was observed on the use of grid spaces as approximation of a Cartesian space.

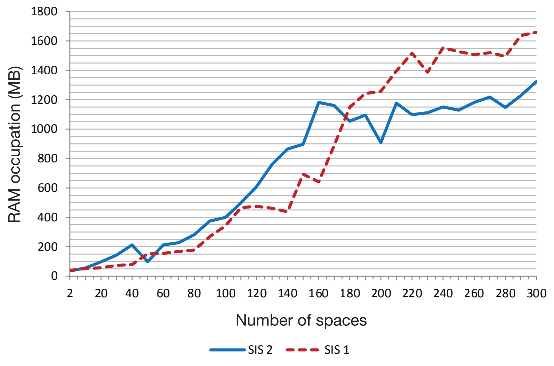


Fig. 10: SIS 2 vs SIS 1: memory footprint comparison

B. *Mean reasoning time vs. context size*

The second test aims to investigate the dependence of the mean reasoning time on the size of the contexts. The space configuration in this case includes two Cartesian spaces. One premetric declarative context was defined in each space, the first one for publication and the second one for subscription. During the test, the number of mapped spaces has been maintained fixed at two, whereas the radius of the premetric declarative contexts was varied according to the values 1, 5, 10, 15, 20 and 25 units.

To get a better idea of the results, a comparative test has been conducted between the SIS 2 prototype and the SIS 1 implementation with Cartesian spaces approximated by regular sized grid. As pointed out by the graph in Figure 11, the SIS 1 prototype has a constant mean reasoning time about 2 ms, while the SIS 1 implementation behaves like $O(n^2)$.

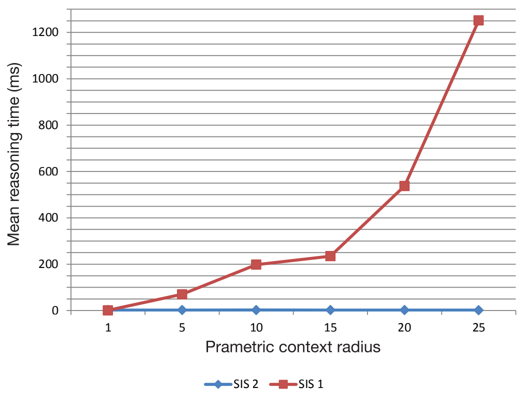


Fig. 11: Mean reasoning time vs. size of context

C. *Discussion*

The experimental tests show how the implemented techniques allow for a better management of different context sizes in infinite spaces, maintaining a constant time of reasoning together with a moderate use of the available RAM memory.

Moreover, the prototype has a better scaling behavior with respect to the finite-space implementation, both for increasing number of mappings and for increasing size of publication and subscription contexts. Such improved scalability can be understood in terms of the number of facts that form the knowledge base of the rule engine managing the operations of transitive closure and matching; the infinite-space implementation reduces the number of facts to one per context, and thus to two facts for a single mapping, whereas the finite-space implementation creates a fact for each location involved in the mapping.

The memory footprint globally resembles the one characterizing the finite-space implementation, but without the peak that has been evidenced during the tests. This memory occupation is caused by the supplementary data structures needed by the infinite-space implementation to correctly handle situations which involve both finite and infinite spaces. It is noticeable that these structures have a relevant weight when a reduced number of spaces and mappings is involved but become negligible as the number of spaces increases.

VI. Related work

Location-aware computing has been an active area of research. Different platforms at the state of art enable location-aware applications focusing on sensor fusion and reasoning with the help of a multi-spatial model, or hybrid model (as called by Becker et al. [8]).

Location Stack [9] defines a layered modeled for fusing location information from multiple sensors and reasoning about an object's location. It, however, does not incorporate a spatial model of the physical world and does not support representations of immobile objects. This leads to a lack of support for spatial reasoning relative to stationary entities such as rooms or corridors.

Loc8 [10], on the other hand, extends the Location Stack layered architecture by considering only high level position and data instead of low level sensor data. Reasoning is applied to that position data, enriched by the knowledge given by a base ontology, to infer additional spatial relationships.

The *Aura Space Service* [11] combines coordinate and hierarchical location models into a single hybrid model, which supports spatial queries. The focus of the Aura Space Service is only on modeling the physical space and supporting spatial queries. It does not address location inferencing and does not provide a framework for spatial reasoning.

MiddleWhere [12] uses the hybrid location model introduced by the Aura Space Service and enables the fusion of different location sensing technologies. MiddleWhere introduces also probabilistic reasoning techniques to resolve conflicts and deduce the location of people given different sensor data. The model of the world is stored in a spatial-enabled database.

Semantic Spaces from Microsoft Research decomposes the physical environment into a hierarchy of spaces. The locations of moving users or devices are correlated to actual physical spaces, thus it is capable of answering "containment" queries. However, because of its inherent lack of metric attributes and precision, it is unable to compute distance accurately or represent locations precisely, which are requirements for some ubiquitous computing applications [13].

Semantic Spaces and Location Stack lack any support for infinite spaces and in general spaces with a coordinate system, while Loc8 and MiddleWhere have at least one spatial model with a Cartesian coordinate system and can handle different levels of precision on that space model. These two platforms substantially treat infinite spaces by using different granularities for location representation on a local and a global coordinate system.

VII. Conclusion and Future Work

The paper proposed an extension to the conceptual model of the SIS platform. This refinement comes in order to enable the use of infinite spatial models like the geodetic or the Cartesian ones. The approach that has been presented has involved the extension of the concept of spatial context and the use of that concept as the elementary unit at the basis of all the spatial operations enabled by the platform itself (i.e., mapping and matching). With the help of a prototypal implementation the revised conceptual model has been tested for performance evaluation. The tests that have been conducted have shown an overall increase of performance and capacity to handle spatial contexts with large extent as needed when using the geodetic space. The main future work consists in the deep integration of the Infinite Space Extension layer in the Core layer. This will enable a more efficient use of the rule engine.

References

[1] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[2] L. Bullivant, *Responsive environments: architecture, art and design*. Victoria & Albert Museum, 2006.

[3] D. Bernini, F. Fiamberti, D. Micucci, and F. Tisato, "Architectural abstractions for spaces-based communication in smart environments," *Journal of Ambient Intelligence and Smart Environments*, vol. 4, no. 3, pp. 253–277, 2012.

[4] C. A. Patterson, R. R. Muntz, and C. M. Pancake, "Challenges in location-aware computing," *Pervasive Computing, IEEE*, vol. 2, no. 2, pp. 80–89, 2003.

[5] ISO, *Spatial referencing by geographic identifiers*. International Organization for Standardization, Geneva, Switzerland, 2003, no. ISO 19112.

[6] D. Garlan and M. Shaw, "An introduction to software architecture," *Advances in software engineering and knowledge engineering*, vol. 1, pp. 1–40, 1993.

[7] IETF. The WebSocket Protocol - RFC 6455. [retrieved: July, 2013]. [Online]. Available: http://datatracker.ietf.org/doc/rfc6455/?include_text=1

[8] C. Becker and F. Dürr, "On location models for ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 9, no. 1, pp. 20–31, 2005.

[9] D. Graumann, W. Lara, J. Hightower, and G. Borriello, "Real-world implementation of the location stack: The universal location framework," in *Mobile Computing Systems and Applications, 2003. Proceedings. Fifth IEEE Workshop on*. IEEE, 2003, pp. 122–128.

[10] G. Stevenson, J. Ye, S. Dobson, and P. Nixon, "Loc8: a location model and extensible framework for programming with location," *Pervasive Computing, IEEE*, vol. 9, no. 1, pp. 28–37, 2010.

[11] C. Jiang and P. Steenkiste, "A hybrid location model with a computable location identifier for ubiquitous computing." Springer, 2002, pp. 307–313.

[12] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. Mickunas, "Middlewhere: a middleware for location awareness in ubiquitous computing applications," in *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Springer-Verlag New York, Inc., 2004, pp. 397–416.

[13] B. Brumitt and S. Shafer, "Topological world modeling using semantic spaces," in *Proceedings of the Workshop on Location Modeling for Ubiquitous Computing, UbiComp*, vol. 2001, 2001, pp. 55–62.