Positioning-Based Domain-Specific Modelling through Mobile Devices

Alberto Sebastián-Lombraña
Universidad Autónoma de Madrid
Madrid, Spain
Alberto.Sebastian@uam.es

Esther Guerra
Universidad Autónoma de Madrid
Madrid, Spain
Esther.Guerra@uam.es

Juan de Lara
Universidad Autónoma de Madrid
Madrid, Spain
Juan.deLara@uam.es

***Abstract*—Modelling is a central activity in many disciplines. It is typically performed with the support of modelling tools that run on desktop computers or laptops, i.e., in static settings. However, some modelling scenarios require a faithful representation of the position of the model elements in the physical world. Such scenarios would benefit from the ability to model in mobility and exploit the data obtained from the sensors embedded in mobile devices. For this purpose, we propose a conceptual approach to positioning-based modelling based on the combination of a physical dimension (as provided by the sensors of mobile devices) and an ontological one (as provided by domain meta-models). We showcase different scenarios for these ideas, and present a prototype app – called METAPHORE – that runs on iOS devices and realizes these concepts.**

***Index Terms*—Model-driven engineering, domain-specific modelling, positioning-based modelling, mobile apps**

I. INTRODUCTION

Modelling is an essential task in any engineering discipline. Models are abstractions of the system under study, built for a given purpose, and often used to design, understand, analyse, verify, test or simulate (existing or new) systems, among other activities [1].

Computer-supported modelling is typically performed on desktop computers or laptops, in static environments. However, mobile devices and their integrated sensors have opened the door to modelling in mobility [2]. This is useful when modelling needs to take place close to the systems being modelled, such as in the on-site maintenance of factories or data-centers [3], when designing touristic itineraries or hiking routes, in construction sites, or to help the placement of IoT devices in smart buildings or cities [2].

In previous work, we developed DSL-Comet, an iOS tool for domain-specific modelling on mobile devices [2]. DSL-Comet allows geo-positioning model elements by *manually* placing them on a map. In this paper, we continue these ideas by enabling positioning-based modelling by means of external elements like Bluetooth low energy (BLE) beacons [4] and sensors within the mobile device like the compass or the Bluetooth. This way, we *automatically* detect the position of real-world elements (e.g., via beacons) to include them in the model. This is useful in scenarios involving indoor positioning [5], or requiring the detection or monitoring of the position of elements of different types (e.g., to help people find their way in airports, hospitals, museums, or track senior citizens with health conditions [4]).

This paper makes the following contributions. First, we propose the novel concept of *positioning-based modelling* on mobile devices, and identify application scenarios. Second, we present a conceptual approach for positioning-based modelling based on the combination of physical and ontological dimensions. This way, modelling is performed by first detecting the element of interest in the physical world using the sensors of the mobile device, and then assigning the element an ontological type and domain properties. Finally, we demonstrate the feasibility of the approach by a prototype called METAPHORE, which runs on iOS devices.

Paper organization. Section II motivates our work and outlines usage scenarios; Section III describes our approach; Section IV details our prototype implementation; Section V compares with related work; and Section VI concludes.

II. MOTIVATION, SCENARIOS AND RUNNING EXAMPLE

In this section, we first motivate positioning-based modelling by identifying application scenarios (Section II-A), and then we introduce a running example (Section II-B).

A. Motivation and application scenarios

Modelling tools enable domain experts to create models within their domain of interest, at a desired level of abstraction. Since domain experts may not be skilled in modelling languages like the UML, these tools must provide user-friendly mechanisms to permit their use by non-technical users, including personalization to specific domains [6], [7].

Traditionally, modelling tools have been developed for desktop computers or laptops, to be used in a static environment. However, some scenarios can profit from modelling in mobility using mobile devices [2]. For example, creating models of hiking tracks, scavenger/treasure hunting games and gymkhanas, or designing domotic buildings, are scenarios that would benefit from modelling on-site. In these cases, the position of the objects in the model is relevant as it has a physical meaning. However, most modelling tools permit arranging the objects arbitrarily in the model, either automatically or manually, and so the relation between the physical element position in reality and its arrangement in the model can be lost. More generally, when modelling a physical reality, we may

TABLE I: Categorization of use case scenarios

Category	Description	Examples
Monitoring	Detect proximity of users to items	Access control, context-aware broadcast, physical internet
Tracking	Get user's movement path	Commercial planning
Routing	Get direction towards items	Checkpoints, navigation in airports, hospitals, museums, shopping malls
Ranging	Get users or items distances	Gymkhanas, navigation help for blind people, lost item's location
Self-positioning	Get user's position using items	Modelling buildings or areas (convention centers, stand distribution in fairs); locate model items using position determination calculations
Others-positioning	Get items' position using users	

need to accurately represent both physical object properties (e.g., position, temperature) and conceptual properties (e.g., use, owner, state), and their relations. However, in classical modelling, abstraction is often used to represent the reality, and little attention has been paid to obtain the physical information. If necessary, physical properties are modelled as attributes of the objects (e.g., as defined in a domain meta-model) and filled manually by the modeller.

We claim that there is a need for modelling tools able to bridge model objects with their physical counterpart. To this aim, we propose using mechanisms to obtain the physical information of the represented reality, both of real objects and of abstract entities – areas, paths, users – to create physically realistic and dynamic models (since the physical properties may change). Moreover, we realize our vision by the use of mobile devices, as they integrate sensors to detect the physical properties of interest (positions in our case).

Table I classifies and illustrates use case scenarios that fit in this modelling approach. We describe them in the following.

First, *monitoring* applications have some behaviour based on detecting when a user is close to a given set of positions or areas. Detecting proximity can be realised using mechanisms such as BLE beacons, GPS, cameras, etc. The purpose of monitoring can be for access control, to provide context-aware content (e.g., extra information in front of a museum exhibitor), or to deploy a "physical internet" [8].

In *tracking* and *routing* applications, the models represent paths where elements are located. Specifically, *tracking* extends monitoring to supervise different locations in space, from which a physical path is created (e.g., the sequence of promotion stands a client has been in front of). Instead, *routing* applications typically direct the user along a path of items (e.g., to help passengers find their way in an airport).

Ranging involves the calculation of the distance between two elements, such as the user's mobile device and a beacon. This distance can be used, e.g., to assign different types to the model's elements, like in hot-cold games, or in applications for lost item's location [9].

Finally, *positioning* applications use calculated positions to locate physical elements in space (e.g., the distribution of stands in a fair). While ranging applications are just interested in finding whether an element is far or close, positioning needs to use more sophisticated techniques (e.g., lateration [10]) to locate the element. We distinguish between *self-positioning*, which locates a measuring device using a reference system, and *others-positioning*, which does the opposite, i.e., it uses measuring devices as a reference to locate other objects.

B. Running example

As a running example, assume we aim at developing an app to configure and manage spaces for conventions and fairs. The envisioned tool should support three usage scenarios:

1) A manager of a convention center can define a model of the conference areas using the tool.
2) A security expert can plan for setting maximum capacities of the different areas, as well as locating emergency exits and evacuation paths.
3) An attendant can consult the space distribution, and its data is used to update the area model (e.g., reflecting the number of attendants in each area to limit the entrance).

The first scenario can use *others-positioning* to build the area model; the second one can use *routing* to direct users to the exit; and the last one can use *monitoring* to help controlling user access.

III. Approach

Our approach relies on mobile devices for domain-specific modelling, and on their sensors to extract physical properties of the model elements of interest. This enables a faithful physical representation of the elements. Fig. 1 shows this working scheme.

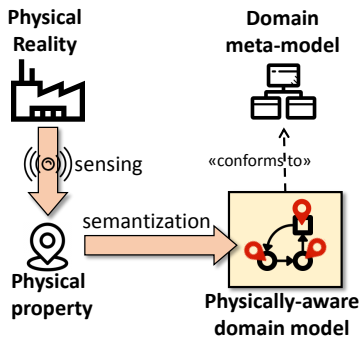


Fig. 1: Approach overview.

In a first step, we detect the physical property of interest, like the position of an element, by using the sensors of mobile devices. Mobile devices are equipped with technologies such as GPS, compass, inertial sensors – accelerometers and gyroscopes – or the telephone network, which can be used to position other objects. In addition, there is a growing number of technologies compatible with mobile devices (e.g., BLE beacons) that extend the device capabilities.

Next, an object representing the physical element is created and assigned a type from a domain meta-model, and domain attributes and relations can be established, in addition to the physical ones. We call this process *semantization* [11]. Models so created can be used for design activities (e.g., to create a building blueprint), or be the basis of model-based applications enriched with context-based behaviour (e.g., in models@run.time applications [12]).

In the following, Section III-A details our approach to positioning-based modelling, Section III-B explains our use of sensors to locate external objects, and Section III-C focuses on the customization of positioning-based modelling apps.

A. Positioning-based modelling

In model-driven engineering, domain-specific modelling is enabled by the provision of domain-specific languages tailored to an application domain. These languages are defined by means of a meta-model (normally a class diagram) holding the primitives, attributes, relations and constraints of the domain.

Fig. 2(b) shows an excerpt of a meta-model for the running example. Conventions have registered Attendants and take place in Buildings. These latter are divided in one or several Sections inter-communicated through Gates. EmergencyExits are a special kind of gates. In this scenario, we are interested in providing an accurate representation of the conference areas, and using this model to monitor the attendants present in each area.

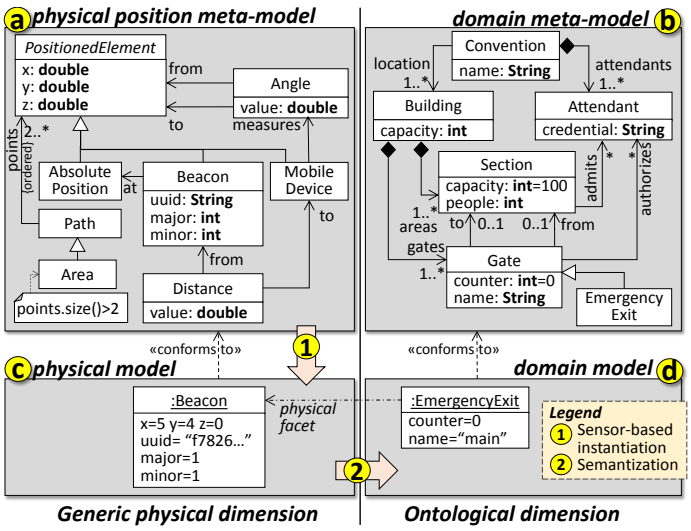


Fig. 2: Physical-based modelling scheme.

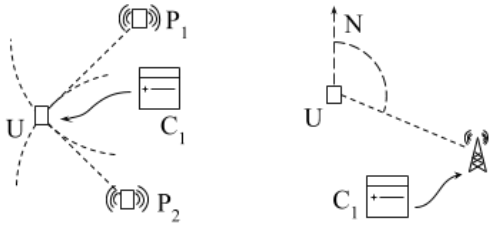
To deal with this scenario, we distinguish two dimensions for objects: a domain-specific or ontological dimension concerned with Gates and Attendants; and a physical dimension concerned with Positions, Distances and devices like Beacons. This way, objects in the ontological dimension may have associated objects from the physical dimension. For this purpose, we rely on the notion of facet-based modelling [13], whereby a host object can transparently acquire the attributes and type of other objects, called its *facets*. As an example, in Fig. 2(d), the EmergencyExit object in the ontological dimension hosts a Beacon facet from the physical dimension. Our solution is inspired by the Orthogonal Classification Architecture [14]. This separates the ontological dimension of model elements from a linguistic dimension, which refers to the (meta-)element used to create them (e.g., Class or Attribute). Instead, our physical dimension accounts for intrinsic object properties (e.g., position), the devices used to obtain them (e.g., beacon, mobile device), location-based information (e.g., angles, distances), and higher-level concepts likes paths and areas[1]. Fig. 2(a) shows the meta-model of the physical dimension.

[1]An area is a special case of closed path of length bigger than 2.

While we currently focus on positions, the same idea could be applied to other physical properties detectable with sensors, like temperatures.

To guarantee fidelity of the model to its physical properties, we instantiate the physical dimension first, likely aided by sensors and positioning methods (step 1 in Fig. 2). For example, we can use the distance of a mobile device to two beacons to determine the device position, provided that the position of the beacons is known. Section III-B will explain the positioning methods that we use. Next, the physical facet is added domain information through a process called model semantization (step 2 in Fig. 2). This consists in assigning a type from the domain meta-model to the physical facet, creating an instance of the type, and filling-in its attributes and relations. For example, we may assign the type EmergencyExit to a positioned beacon and inform its name, as done in Fig. 2.

Elements in the physical and ontological dimensions can have many-to-many relations. For example, there may be several beacons distributed in a building, and a modeller would place her device close to an element of interest (e.g., an EmergencyExit) to locate it in relation to the other beacons. Conversely, attendants may receive in their badge a beacon assigned to a host Attendant object, hence enabling the monitoring of the attendant position based on the beacon position. These two examples, illustrated in Fig. 3, correspond to the others- and self-positioning cases of Table I.



(a) (b)

Fig. 3: (a) In *self-positioning*, a user U uses the location of other partners or beacons P_1 and P_2 as a reference system to locate a model's element C_1. (b) In *others-positioning*, a user U triangulates an element C_1 using a beacon.

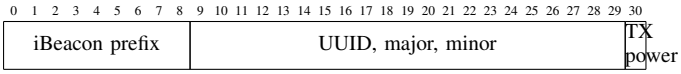
B. Sensors and RF technologies for modelling

Mobile devices continuously use location methods in their applications, services and internal operation to locate themselves. For this purpose, they use technologies such as GPS, compass, inertial sensors (e.g., accelerometers, gyroscopes) or other compatible technologies. However, using location methods to determine the position of other objects is less common. This requires from mathematical methods traditionally found in topography, astronomy or radiodetermination. Given a reference system, these methods allow determining the position and speed of an object by measuring angles, distances or times [15]. Today's mobile devices integrate or use technologies that allow the following measurements:

1) While rotating, the device sweeps an angle that the compass or the inertial system's gyros can measure.
2) Distances can be measured using accelerometers [16], images or BLE beacons [17] [18].
3) The recent Bluetooth 5.1 implements *angle of arrival* and *angle of departure* features [19]. New mobile devices released with ultra-wide band (UWB) capabilities will be able to find near phones or IoT devices using RF propagation times.

Many indoor positioning systems (IPS) [4], [5], [20] use these methods, in particular trilateration or triangulation depending on whether they are based on distances or angles. Next, we overview the technologies and mathematical methods used in our proposal, and refer to [15], [21] for a detailed description. Depending on the employed method and technology, different location modes are enabled.

1) BLE beacons: Bluetooth low energy (BLE) beacons broadcast Bluetooth advertisements to announce their presence to near devices. They use low-energy transmission at 2.4 GHz band – like the WiFi system – and most mobile devices support their capabilities. The dominant protocols are Google's Eddystone for Android, and Apple's iBeacon for iOS [22]. The broadcasted iBeacon frame (shown below) includes a unique identifier of type UUID, as well as two values called minor and major in case the signals need to be redundant. This way, mobile devices can unequivocally identify and register each beacon.



2) Distance ranging using RSSI: Mobile devices continuously calculate the received signal strength indicator (RSSI) of the collected signals for multiple purposes. This value measures the signal power loss, and is proportional to its travelled distance in some scenarios. By electromagnetism, the received signal strength is inversely proportional to the square of the distance covered by the transmission. However, in mobile and indoor scenarios, this relationship cannot be assumed due to phenomena such as reflection and interference. Thus, indoor positioning methods often use corrective methods [23], calibration processes, Kalman or particle type filtering [24], machine learning [24], or other evaluation methods such as fingerprints [22]. This way, the accuracy of indoor positioning based on BLE technologies is between 2 and 5 meters [20].

3) Trilateration using beacons ($\rho - \rho$ location mode): To calculate distances, a user can place BLE beacons on known positions, and then use RSSI values measured with a mobile device to locate the device in reference to the beacons. The opposite is also possible, i.e., placing a beacon on a spot of interest, and using RSSI to locate the beacon out of known positions. As Fig. 4 shows, trilateration permits solving both scenarios.

Trilateration allows determining absolute or relative locations of points by measurement of distances. Specifically, if the distance ρ_i and position (x_i, y_i) of a set of beacons B_i are

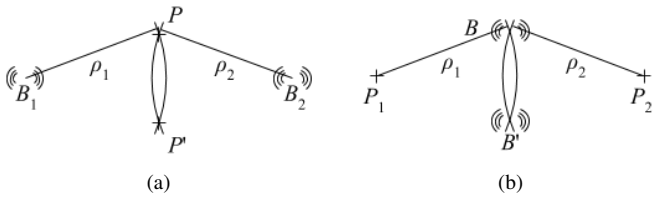


Fig. 4: (a) The $\rho - \rho_i$ *self-location* mode locates an object using beacons in known positions as reference system. (b) The $\rho - \rho$ *location* mode locates a beacon using known positions as reference system.

known, as in Fig. 4(a), a circumference:

$$(x - x_i)^2 + (y - y_i)^2 = {\rho_i}^2 \quad \forall i \tag{1}$$

can be defined for the measures of the i-th beacon, where positions P and $P' = (x, y)$ are the circumference intersections. To avoid solving non-linear equations, we can take more measurements, and by subtracting pairs of them, we obtain:

$$(x_i - x_1)x + (y_i - y_1)y - \frac{1}{2}({\rho_1}^2 - {\rho_i}^2 + {d_{1i}}^2) = 0 \quad \forall i \tag{2}$$

where d_{1i} is the distance between the first and i-th beacon. If there are two equations, then the system yields two solutions, and we can either take a third measurement or heuristically choose between P and P' (e.g., select the one inside the building). Taking three or more measurements over-determines the system and permits solving it applying the least squares method. Besides, if the error deviation is known, each measurement can be weighted to calculate an optimum solution.

We can use the same method if the beacons' positions are unknown but the measurement positions are known, as in Fig 4(b).

4) Triangulation ($\theta - \theta$ location mode): Triangulation is based on measuring the angle between two known positions to locate another place. Again, there are two options: using two known positions to locate the user's mobile device, or using the known device position as part of the reference system. Fig. 5 shows a scheme of both possibilities.

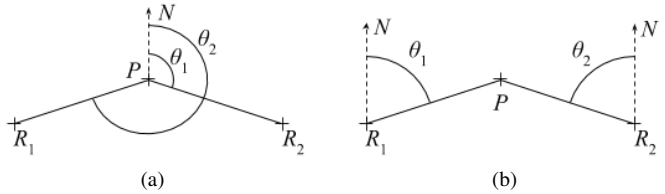


Fig. 5: (a) The $\theta - \theta$ *self-location* mode locates an object P using the angles between the reference system points R_i. (b) The $\theta - \theta$ *location* mode locates an object P using the angles from some known positions of a reference system.

If a reference system of positions $R_i = (x_i, y_i)$ is known[2],

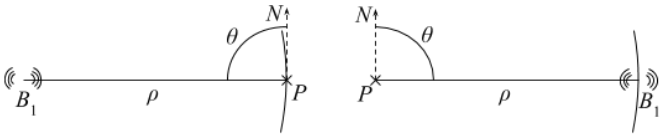
[2]Fig. 5 uses the North for simplicity. In this case, since the model elements' position depends on their position on the Earth, the "model's North" must be subtracted from the measurement.

then we can describe the lines from them to an unknown point $P = (x, y)$ as the equation:

$$(y - y_i) = \tan(\theta_i)(x - x_i) \quad \forall i \tag{3}$$

Once again, this system can be solved either directly or using least squares depending on the number of equations.

5) $\rho - \theta$ location mode: This mode combines angles and distances. In this case, the user places a beacon close to the object to be positioned, and then uses RSSI values and relative angles measured from a unique known position to locate the beacon. Alternatively, the user can place a beacon in a known position, and then use RSSI and relative angles measured from unknown positions to determine such positions. Fig. 6 shows a scheme of these scenarios.



(a) (b)

Fig. 6: (a) The $\rho - \theta$ *self-location* mode locates an object placed in the vertex of the angle between a reference point and a beacon. (b) The $\rho - \theta$ *location* mode finds an object close to a beacon.

Given a beacon in the known position $B_i = (x_i, y_i)$, and an unknown measuring point $P = (x, y)$ at distance ρ_i and angle θ_i of the beacon, we can describe the line between both as:

$$\begin{cases} x - x_i = \rho_i \cos\theta_i \\ y - y_i = \rho_i \sin\theta_i \end{cases} \quad \forall i \tag{4}$$

The opposite is similar, as angles are defined counterclockwise and so signs are equalised. Thus, we can solve the equation:

$$\begin{cases} x = \rho_i \cos\theta_i + x_i \\ y = \rho_i \sin\theta_i + y_i \end{cases} \quad \forall i \tag{5}$$

Note that only two measurements are needed to determine a position over a plane: the angle θ and the distance ρ; therefore, this location mode only requires one beacon.

6) Geo-positioning: The position and magnetic heading of a mobile device on Earth can be obtained via GPS multi-lateration. This information can be exploited to geo-position a model and display it over maps. However, GPS is generally not accurate enough to locate model elements in indoor scenarios.

C. Customization levels for positioning-based modelling

Positioning-based modelling tools may need to support one or more of the usage scenarios presented in Table I. Each scenario requires customizing different elements. In particular, we identify the four tool customization levels shown in Fig. 7, where higher label values denote more complex scenarios.

In the simplest case (label 1), a "raw" positioning-based modelling tool just creates physical models (e.g., conforming to the physical meta-model of Fig. 2(a)) using the sensors of the mobile device.

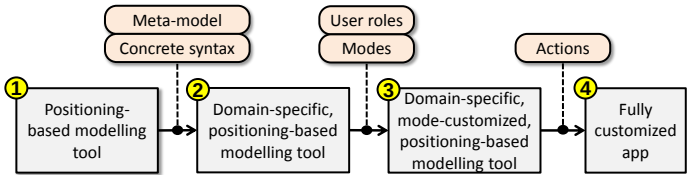


Fig. 7: Customization levels for positioning-based modelling tools.

The subsequent level (label 2) corresponds to tools that need to assign a domain-specific meaning to physical models. For this purpose, these tools must provide a domain meta-model, in order to allow the model semantization process proposed in Fig. 2. In our running example, this customization is achieved using the domain meta-model for convention centers shown in Fig. 2(b). Moreover, the concrete syntax of the domain meta-model can be customized as well. This concrete syntax could react to changes on domain and physical properties of the model elements (e.g., highlighting areas with high occupancy).

The next level (label 3) permits a modelling tool to configure the used measurement methods and location modes (cf. Section III-B) as well as defining triggers from one mode to another. Transitioning between modes does not change the physical/domain models but only the measurement technique. Tools can exploit this to assign different modes to different user roles. For instance, the tool for the running example can define one mode for each scenario and user role (cf. Section II-B). The first scenario, performed by the convention manager, requires creating a model of the convention area, for which any of the location modes $\rho - \rho$, $\theta - \theta$ or $\rho - \theta$ can be selected (cf. Sections III-B3 to III-B5). The second scenario, performed by the security expert, requires setting the emergency exits and evacuation paths. These can be located using beacons and the $\rho - \rho$ location mode.

Finally, the most complete tool customization level (label 4) permits triggering actions upon certain events detected by the mobile sensors (e.g., coming close to a beacon). This enables integrating positioning-based models as part of applications. For instance, the last scenario of the running example requires an action that increases the number of people in a section when an attendant goes through a gate.

IV. Tool Support

We have realized our proposal on a prototype tool called METAPHORE. Next, Section IV-A describes the tool capabilities, Section IV-B shows how to customize the tool by defining domain meta-models and location modes, and Section IV-C illustrates the use of the tool for creating positioned models.

A. The METAPHORE tool

METAPHORE[3] offers a simple and versatile way to model the space around the users. It supports the location modes

[3]https://metaphore.herokuapp.com/web/

described in Section III-B via inertial sensors, compass, GPS and iBeacon devices. This permits the creation of domain-specific, mode-customized, positioning-based modelling tools (customization level 3 in Fig. 7). Fig. 8 shows the tool deployed on a tablet, and several (white round) iBeacon devices in the bottom-right.

Fig. 8: Using METAPHORE on a tablet, and some iBeacon devices.

METAPHORE is an iOS application for mobile devices. It is implemented in Objective-C following the model-view controller pattern. In the foreground, a series of event-oriented views allow users to control the application, while in the background, the delegated modules listen to the system interruptions. This makes any location measurement and positioning computation transparent to the users.

Regarding its operation, METAPHORE itself manages the privacy requirements of the operating system to use location services; the users, credentials, roles and access permissions to information; the work sessions; the available iBeacon devices; and the storage of models in memory.

Fig. 9 shows the high-level architecture of METAPHORE. As the figure shows, it can be used by modelling tool creators to customize positioning-based modelling tools, and by end-users modellers to build positioning-based models using those tools. Such modelling tools also run atop METAPHORE, enabling the creation of physical models featuring the position of real-world elements. Objects from this physical model can be assigned ontological types from a domain meta-model, which may bring domain attributes and relations that the end-user can fill in. The next subsection explains how modelling tool creators can configure this domain meta-model.

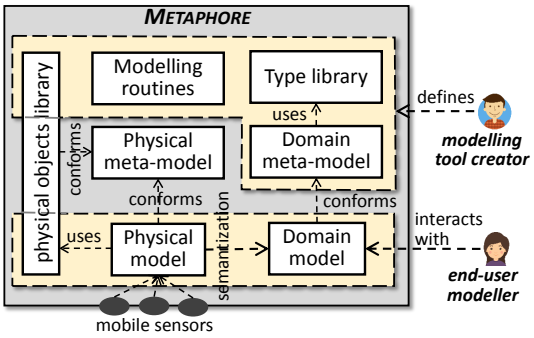


Fig. 9: METAPHORE at a glance.

B. Domain customization by modelling tool creators

METAPHORE allows defining domain meta-models, and using them in the semantization of physical models. To promote reuse, we enable the construction of type/object libraries at the meta-model/model levels. As an example, Fig. 10 shows how to create types and their attributes to yield a type library, and Fig. 11 shows how to reuse the defined types to create the meta-model of convention centers in our running example.

Fig. 10: Defining an ontological type and its attributes.

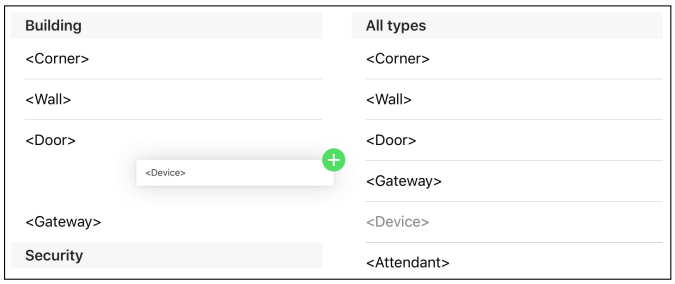


Fig. 11: Defining a meta-model by reusing ontological types.

At the model level, METAPHORE permits defining libraries of physical objects, including absolute positions and iBeacon devices (cf. Fig 2(a)). These objects can be reused in different physical models, and can be assigned positions via location methods, and domain types and attributes using semantization. Typically, these libraries are created by the tool creator, but can be extended by the end-user modeller, e.g., to configure additional iBeacon devices, as shown in Fig. 12.

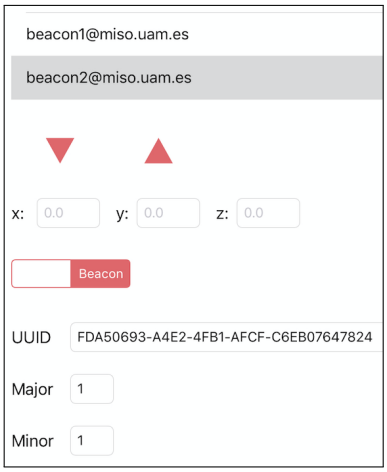


Fig. 12: Adding an iBeacon to the library of physical objects.

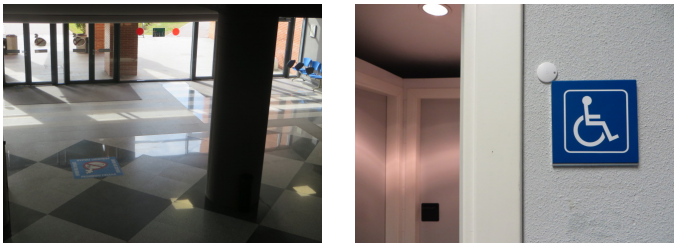
The tool also supports the specification of *modelling routines*: sequences of consecutive positioning modes for specific user roles and a given domain meta-model.

C. Positioning-based modelling by end-users

Once a modelling tool has been customized with a domain meta-model, a library of physical objects, and modelling routines, end-user modellers can use it for positioning-based domain-specific modelling.

For illustration purposes, next we report on the modelling tool for our running example. This considers three user roles: the *manager* of the convention center who uses the tool to build the conference area model; the *security expert* who can add emergency exits and evacuation paths to the area model; and the *attendant* who uses the area model as a reference map, and whose positioning data is retrieved by the modelling tool to update the area model.

First, we use an iBeacon-based reference system to model the convention center, placing one beacon in each hall corner to register its position. Fig. 13(a) shows our University hall, which will be used as one of the sections of the convention center. Then, the manager can add gates, windows, columns, etc. to the model by using the *self-location* $\rho - \rho$ mode. Specifically, the manager places the mobile device on the point of interest, and upon its location, assigns the position a domain type (e.g., window). The manager can also use beacons to help attendants find significant spots in the hall, as Fig. 13(b) shows.



(a) (b)

Fig. 13: (a) University hall used as a convention center. (b) An iBeacon device used to help attendants find services within the building.

Next, to enable access control, the security expert uses the modelling tool and some beacons to locate the gates. The beacons have been previously assigned a domain type (e.g., entrance, emergency exit), so that the gate is automatically semantized and related to a beacon UUID when added to the area model. Likewise, the event planning expert can position stands and other objects by using the iBeacon-based reference system. Finally, several signboards placed in the columns inform attendants about the location of the reference system.

Fig. 14 shows the area model while being constructed. The right part contains the model of the hall, and its elements were placed using positioning techniques. The pop-up window to the left allows the semantization of elements, i.e., assigning a domain type to the elements created by sensor-based positioning, and giving a value to the attributes of the type.

In a last step, attendants use the modelling tool to find where places stand at. They can also use the signboards placed in the building columns to locate their position in the event: using

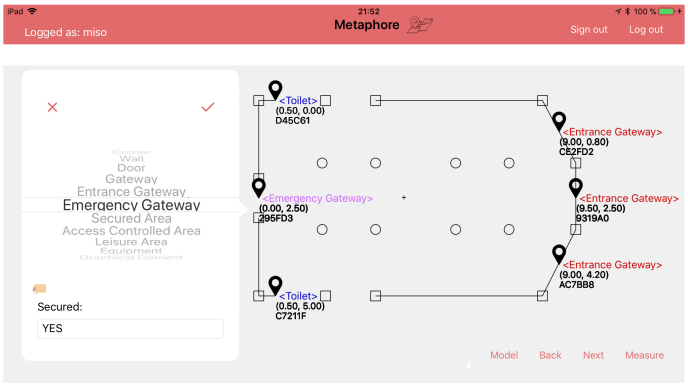


Fig. 14: Semantization process within METAPHORE.

the *self-location* $\theta - \theta$ mode, their device can be set to point to each signboard, and the angle measured between them is used to locate the device.

METAPHORE does not support yet setting triggers and actions (customization level 4 in Fig. 7). Once supported, the position of attendants can be used to dynamically update the area model. For example, the attendant devices can notify when they pass through gates in order to update the capacity ratio, and the entrances' UUID advertisements can trigger an interface to submit the credentials and secure the access.

V. RELATED WORK

BLE beacons have been generally used for localization; proximity detection and interaction; and activity sensing [4]. Next, we review some applications that could have applied METAPHORE in their development.

Regarding localization, BLE beacons are especially useful for indoor positioning, as other systems like GPS are not effective [5]. For example, several airports (e.g., Hong Kong, Gatwick) have developed beacon-based apps to aid passengers in finding the way to terminals [4]. These applications normally use a model of the environment, as in [25], where building information models (BIM) and beacons were combined for indoor positioning using mobile devices. Commercial products like air-go (https://air-go.es) offer technologies for indoor positioning, targeting industries like hospitals, shopping malls, museums and airports. Other applications like [9] use beacons to locate objects instead of users.

Regarding proximity detection, BLE beacons have been used to send effective notifications (e.g., in art galleries [26]) leveraging from the user context and location [4]. BLE beacons have also been used to detect and monitor user activity, e.g., to track senior citizens for daycare report generation [27].

Some works aim at facilitating the creation of apps that rely on beacons. A beacon management system was proposed in [28] to facilitate the registration of beacon UUIDs and the creation of apps that employ user positions. Instead, our approach provides a modelling environment with support for localization and proximity, which uses several sensors, and that can be customized with domain-specific concepts.

Since BLE beacons have limited precision, several techniques have emerged to improve their accuracy [22]–[24]. We have not used these techniques up to now, as the obtained precision is enough for our current purposes. However, in the future, we may study optimization techniques.

The model-driven engineering community has barely explored the use of devices and sensors for modelling. In [29], the authors argue for the need of models of user interaction within modelling editors, which includes the selection of inputs from devices (e.g., MIDI keyboards). The authors do not consider a physical modelling dimension, but their statechart-based method could complement our approach. In [30], the authors present a DSL to prototype executable models of gestural interactions, and use various devices (e.g., 3D sensors, accelerometers) to obtain the real-time position of body parts and update the model in consequence. Instead, we consider positioning and not gestures, and expand the type and attributes of model objects based on this information. Semantization [11] is a technique to provide additional information to sensor data. While it has been used in IoT applications, we are not aware of approaches providing model-based semantization, i.e., adding types and attributes of a meta-model to sensor data.

Overall, we are not aware of other approaches enabling positioning-based domain-specific modelling on mobile devices. We believe that this opens the door to a plethora of new applications for modelling technology.

VI. Conclusions and Future Work

We have proposed a positioning-based modelling approach to bridge the system under study and its virtual representation. It is based on the coordinated use of physical and ontological dimensions, as modelling is driven by the physical signals provided by devices (e.g., beacons and sensors within the mobile). We have presented a prototype tool and an example.

In the future, we plan to improve the tool to support actions and triggers, and hence achieve level 4 in Fig. 7. We would like to explore the use of other sensors (e.g., accelerometer, gyroscope) and devices (e.g., camera), as well as to combine the approach with augmented reality and collaboration, in the style of [31]. Finally, we would like to connect our approach with systems supporting BIM technology [32].

Acknowledgment

Work funded by the Spanish Ministry of Science (RTI2018-095255-B-I00) and the Madrid Region (S2018/TCS-4314).

References

[1] H. Vangheluwe, E. J. H. Kerckhoffs, and G. Vansteenkiste, "Computer automated modelling of complex systems," in *Proc. European Simulation Multi-conference*, 2001, pp. 1–12.

[2] D. Vaquero-Melchor, J. Palomares, E. Guerra, and J. de Lara, "Active domain-specific languages: Making every mobile user a modeller," in *Proc. MODELS*. IEEE CS, 2017, pp. 75–82, see also https://diagrameditorserver.herokuapp.com/.

[3] D. Vaquero-Melchor, A. Garmendia, E. Guerra, and J. de Lara, "Domain-specific modelling using mobile devices," in *ICSOFT, Revised Selected Papers*, ser. CCIS, vol. 743. Springer, 2017, pp. 221–238.

[4] K. E. Jeon, J. She, P. Soonsawad, and P. C. Ng, "BLE beacons for internet of things applications: Survey, challenges, and opportunities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 811–828, 2018.

[5] R. F. Brena, J. García-Vázquez, C. E. Galván-Tejada, D. M. Rodríguez, C. V. Rosales, and J. F. Jr., "Evolution of indoor positioning technologies: A survey," *Sensors*, vol. 2017, pp. 2 630 413:1–2 630 413:21, 2017.

[6] S. Abrahão, F. Bourdeleau, B. H. C. Cheng, S. Kokaly, R. F. Paige, H. Störrle, and J. Whittle, "User experience for model-driven engineering: Challenges and future directions," in *Proc. MODELS*. IEEE CS, 2017, pp. 229–236.

[7] J. J. López-Fernández, A. Garmendia, E. Guerra, and J. de Lara, "An example is worth a thousand words: Creating graphical modelling environments by example," *SoSyM*, vol. 18, no. 2, pp. 961–993, 2019.

[8] Google's Physical Web, "https://google.github.io/physical-web/," 2020.

[9] S. Mirri, C. Prandi, P. Salomoni, and L. Monti, "Social location awareness: A prototype of altruistic IoT," in *Proc. NTMS*. IEEE, 2016, pp. 1–5.

[10] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, 2001.

[11] F. Shi, Q. Li, T. Zhu, and H. Ning, "A survey of data semantization in internet of things," *Sensors*, vol. 18, no. 1, p. 313, 2018.

[12] G. S. Blair, N. Bencomo, and R. B. France, "Models@run.time," *IEEE Computer*, vol. 42, no. 10, pp. 22–27, 2009.

[13] J. de Lara, E. Guerra, J. Kienzle, and Y. Hattab, "Facet-oriented modelling: open objects for model-driven engineering," in *Proc. SLE*. ACM, 2018, pp. 147–159.

[14] C. Atkinson and T. Kühne, "Rearchitecting the UML infrastructure," *ACM Trans. Model. Comput. Simul.*, vol. 12, no. 4, pp. 290–321, 2002.

[15] B. Forssell, *Radionavigation systems*. Artech House, 2008.

[16] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Foundations and Trends in Signal Processing*, vol. 11, no. 1-2, p. 1–153, 2017.

[17] F. Zafari, I. Papapanagiotou, M. Devetsikiotis, and T. J. Hacker, "Enhancing the accuracy of ibeacons for indoor proximity-based services," in *Proc. ICC*. IEEE, 2017, pp. 1–7.

[18] H. Zou, Z. Chen, H. Jiang, L. Xie, and C. Spanos, "Accurate indoor localization and tracking using mobile phone inertial sensors, wifi and ibeacon," in *Proc. INERTIAL*, 2017, pp. 1–4.

[19] N. B. Suryavanshi, K. Viswavardhan Reddy, and V. R. Chandrika, "Direction finding capability in bluetooth 5.1 standard," in *Ubiquitous Communications and Network Computing*. Springer, 2019, pp. 53–65.

[20] M.-S. G. Martín, J. Torres-Sospedra, and J. Huerta, "A meta-review of indoor positioning systems," *Sensors*, vol. 19, no. 20, p. 4507, 2019.

[21] F. Seco, A. R. Jiménez, C. Prieto, J. Roa, and K. Koutsou, "A survey of mathematical methods for indoor localization," *WISP*, pp. 26–28, 2009.

[22] P. C. Ng, J. She, and R. Ran, "A compressive sensing approach to detect the proximity between smartphones and BLE beacons," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7162–7174, 2019.

[23] A. D. Blas and D. López-de Ipiña, "Improving trilateration for indoors localization using BLE beacons," in *Proc. SpliTech*, 2017, pp. 1–6.

[24] C. H. Lam, P. C. Ng, and J. She, "Improved distance estimation with BLE beacon using kalman filter and SVM," in *ICC*, 2018, pp. 1–6.

[25] J. C. Ferreira, R. Resende, and S. Martinho, "Beacons and BIM models for indoor guidance and location," *Sensors*, vol. 18, no. 12, p. 4374, 2018.

[26] P. C. Ng, J. She, and S. Park, "Notify-and-interact: A beacon-smartphone interaction for user engagement in galleries," in *Proc. ICME*. IEEE CS, 2017, pp. 1069–1074.

[27] Y. Kashimoto, T. Morita, M. Fujimoto, Y. Arakawa, H. Suwa, and K. Yasumoto, "Sensing activities and locations of senior citizens toward automatic daycare report generation," in *Proc. AINA*. IEEE CS, 2017, pp. 174–181.

[28] A. Barriga, A. Rodríguez, J. García-Alonso, J. Berrocal, R. Flores, and J. Murillo, "Using beacons for creating comprehensive virtual profiles," in *Proc. UCAmI*, ser. LNCS, vol. 10070. Springer, 2016, pp. 295–306.

[29] V. Sousa, E. Syriani, and K. Fall, "Operationalizing the integration of user interaction specifications in the synthesis of modeling editors," in *Proc. SLE*. ACM, 2019, pp. 42–54.

[30] R. Deshayes and T. Mens, "GISMO: a domain-specific modelling language for executable prototyping of gestural interaction," in *Proc. EICS*. ACM, 2015, pp. 34–43.

[31] R. Seiger, M. Gohlke, and U. Aßmann, "Augmented reality-based process modelling for the internet of things with HoloFlows," in *Proc. BPMDS/EMMSAD*, ser. LNBIP, vol. 352. Springer, 2019, pp. 115–129.

[32] R. Sacks, C. Eastman, G. Lee, and P. Teicholz, *BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers, 3rd Edition*. Wiley, 2018.