Extending the GeoJSON Standard with Deontic Logic Policies

Benjamin Aziz[a]
School of Creative and Digital Industries, Buckinghamshire New University, High Wycombe HP11 2JZ, U.K.

Keywords: Deontic Logic, Geofencing, GeoJSON, Policies.

Abstract: GeoJSON is a widely used format for encoding geographic data structures using the JSON format. It enables easy integration of spatial data in Web applications and supports various geometries like points, lines and polygons. However, its openness and simplicity can introduce safety and security challenges. Amongst these is the lack of the ability to express policy rules related to some geometry, therefore leading to potential security and safety risks for any critical geofencing solutions that use the standard. In this paper, we propose an extension to the GeoJSON standard that includes the addition of a *policy* element to geometric features, such that the policy consists of a set of rules that will be evaluated when the geometry is activated (i.e. stepped on, crossed over or entered/existed). We use deontic logic to express such rules, and demonstrate the usefulness of such approach to a couple of potential real-world examples.

1 INTRODUCTION

GeoJSON (Butler et al., 2016) is an open standard for encoding various geographic data structures, based on the JSON (JavaScript Object Notation) data exchange standard (Bray, 2017). In recent years, GeoJSON has revolutionised the handling of spatial data in Web-based Geographic Information System (GIS) applications by dramatically simplifying the way geographic data is represented, moved around and used on the Web. This simplicity and flexibility of the GeoJSON schema enables seamless integration with modern Web technologies, as it fits well with the RESTful API design and Cloud-based approach, making it indispensable for real-time mapping and spatial analysis nowadays. However, despite its popularity, the GeoJSON standard lacks built-in mechanisms for specifying security and access control policies. This omission presents significant challenges (Tarameshloo and Fong, 2014), especially as geospatial data becomes increasingly integral to various applications, many of which could have security and safety requirements.

In this paper, we address the above deficiency by proposing to extend the specification of the GeoJSON standard to incorporate a new Feature element called `policy`, which is a placeholder for including *any policies* associated with a geofence. We demonstrate how this placeholder can be used to include *deontic policies* in order to control behaviour around three types of geometry: a point, a line and a polygon. Note that the approach we follow here is different from the framework defined in RFC6772 (Schulzrinne et al., 2013) in that the latter describes how access to location information can be protected using authorisation policies, and not how activities triggered by geolocation behaviour are controlled. We give in the paper examples of the kind of scenarios where such policies might be useful in the real world.

First formalised by von Wright in (Von Wright, 1951), the deontic logic contains three main operators: permissions, prohibitions and obligations. Permissions express events or actions that may happen, prohibitions express events or actions that are not allowed to happen, and finally, obligations express events or actions that must happen. The idea of combining normative reasoning, such as deontic logic reasoning, with geographic information is not new. In (Winkels et al., 2010), for example, the authors demonstrated how normative reasoning can be integrated with geospatial data using Semantic Web technologies, thus allowing users to query geospatial information in combination with legal norms or spatial regulations, for example, determining whether specific actions, like building construction or waste disposal, are permitted or prohibited. Deontic logic has also other important applications, in particular within the legal domain, where it is used to model statutes and regulations by expressing obligations (e.g., "Tax must be paid by April 15") and permissions (e.g., "Cit-

[a] https://orcid.org/0000-0001-5089-2025

Aziz, B.
Extending the GeoJSON Standard with Deontic Logic Policies.
DOI: 10.5220/0013673500003985
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Web Information Systems and Technologies (WEBIST 2025)*, pages 133-140
ISBN: 978-989-758-772-6; ISSN: 2184-3252
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

izens may appeal within 30 days"), therefore enabling consistency checks and compliance verification.

The rest of the paper is structured as follows. In Section 2, we discuss a few works from literature related to our paper. In Section 3, we define a couple of example scenarios as the starting point for end-user goals. In Section 4, we propose our extension of the GeoJSON standard by introducing the `policy` element. We demonstrate in the section how this new element can be used to express deontic policies and link these with the end-user goals. Finally, in Section 5, we conclude the paper and discuss future work.

2 RELATED WORK

One of the earliest attempts to standardise the model of geofences was provided by W3C's Geofencing API model (W3C, 2017). However, this model was limited in its definition of a geofence to the regional geometry, including circular geometries. Another popular API model is TomTom's Geofencing API (TomTom, 2019). The API permits the definition of circles, polygons, corridors and two-point rectangle geofences, and provides for a rich geofence management API that allows for the setting up of projects (collections of geofences) with various configurations for each project. Despite the lack of generic polices, TomTom's API allows for an access control list-style policy to be set up in association with various concepts (a fence, a project, an object etc.), which in turn provides permissions for the administrators of those concepts and objects interacting with them to carry out certain operations such as reading of reports, updating of the attributes of a geofence, deletion of a geofence or a project and creation of a new geofence or a project. TomTom's Geofencing API does not implement holes in polygons, as defined in RFC 7946. Moreover, we found that the only shapes mentioned in TomTom's API are circles, rectangles, corridors (i.e. lines) and polygons (we did not find support for single point-based geometries.)

As we mentioned in the Introduction, the idea of a geofence-related policy is not new. In fact, the recent version of NIST's guidelines on managing the security of mobile devices in enterprises (Howell et al., 2023) alludes to the idea of a "geofence policy" in scenarios where devices belonging to an enterprise might be granted different permissions, depending on their geographical locality. There is also version of XACML 3.0 (Rissanen, 2013) called GeoXACML 3.0 (Matheus, 2023), which extends XACML with geospatial information.

Several applications for geofences have been suggested in literature, in almost every walk of life. Of particular relevance are applications of geofencing in the healthcare domain, where a recent literature review (Hill et al., 2024) highlighted geofencing as one of the main intervention technologies for people suffering from dementia, for example. In (Ullah et al., 2021), the authors suggested the use of geofencing technologies as a means for monitoring and subsequently containing the spread of dangerous and highly infectious viruses (e.g. COVID-19 variants). The authors in (Nguyen et al., 2017) proposed the use of geofencing as a solution for the problem of the ascertainment of hospitalisations, where smartphones and geolocations are used to identify and record hospitalisations. The approach they followed is based on a quantitative methodology, demonstrating improvement in the quality of care within the limited sample used in their study. In (Helmy and Helmy, 2017), the authors introduced Alzimio, an application that uses IoT devices in conjunction with geofences to implement safe zones, activity-based alarms, "take-me-home", "navigate to nearest friend" and "check-on-me" functions for patients living alone with Alzheimer's disease. Similar approach was proposed by (Chantaweesomboon, 2021), however using Bluetooth-based geofencing. In (Gilmore, 2020), geofencing has been proposed as a measure for protecting children, while away from parents.

More recently, the scoping review presented in (Tobin et al., 2023) found that more than half of the studies covered used direct intervention with geofencing in scenarios of healthcare and care at home. Moreover, the review concluded that one of the most important missing links in the geofencing research landscape is the problem of the *derivation of geofences* themselves based on *end-user requirements*. Such missing link would explain how geofences work to achieve the desired outcomes in various scenarios. Our approach in this paper advocates the control of geofence events using *policies* that are themselves derived from end-user *scenarios*, encompassing the user's requirements and therefore providing one answer for this missing link between end-user requirements and geofences.

Finally, in (Van Riemsdijk et al., 2015), the authors discussed the problem of *social norms* when designing technology, and highlighted the issue of the specification of ethical challenges that could arise from imposing those norms, particularly, in the example context of geofencing. Part of such challenges are what is known as *contrary-to-duty* obligations as formulated by (Hansen et al., 2007), which define what *ought to be done* if something goes wrong. Works such as these provide a reasonable motivation for the adoption of

deontic logic as a formal language for defining and specifying geofence-related policies.

3 SCENARIOS

Scenarios (sometimes also called *user stories*) (Kannan et al., 2019; Turner et al., 2013) are detailed descriptions of what happens in reality in regards to some problem at hand. For example, the process for caring for an elderly or disabled person while living at home or in an assisted-living environment, or for a patient who is currently recovering in a hospital ward, along with the context of that environment in terms of the various resources available and the actions the carers, nurses, doctors or any other personnel might need to perform to achieve the objectives. Scenarios, which can be regarded as firsthand statements of stakeholders' and end-users' needs, are usually described using free-style natural languages and will likely contain technical or specialised terminology.

We consider, in our case, a scenario to be such a natural language statement, which further contains a list of goals of the following format (incorporating deontic notions):

- *"[A user] must/ought/should not [do something] [with reference to some geo-location]."* (Prohibition)
- *"[A user] may [do something] [with reference to some geo-location]."* (Authorisation)
- *"[A user] must/ought/should [do something] [with reference to some geo-location]."* (Obligation)

This approach promotes user-centric design while at the same time, expressing user requirements and goals within some locality. Let's look at some examples from real world scenarios.

Example 1 (Care at home for the elderly)**.** *Jane is an elderly lady who suffers from Dementia, and she lives on her own in an apartment that she owns. Due to her age, she also suffers from frailty and a few other minor ailments. Jane is vulnerable to a few risks, including the risk of wondering off outside of the perimeter of the building in which her apartment is situated, the risk of developing weaker muscles due to lack of movement and the risk of dehydration, specially during hot weather periods. While the perimeter of the building is secured and monitored by a resident manager. Therefore, the following two goals must still be satisfied continuously in order to ensure Jane maintains a healthy daily routine:*

G3 *Jane* must not *switch off the lights in her bedroom at evening time when she exits the bedroom, so that she can find her way back*

G4 *Jane* must use *the bathroom at least three times per day to demonstrate that she is drinking enough quantity of water*

□

As we can see, the scenario mentions two goals: a prohibition (G3) and an obligation (G4).

Example 2 (Shopping at a local market)**.** *Mark runs a small business stall selling farm food products at a local market every Saturday. He would like to make use of geofencing technologies to achieve some objectives that would enhance his product sales. In particular, Mark would like to achieve the following two goals:*

G1. *Mark* may send *discount promotions to customers entering the geographic perimeter of the market, where his business stall is based, via a registration app. This would help attract potential customers to his stall*

G2. *Mark* may receive *feedback from his customers who exit the perimeter of the market in order to improve his future products*

□

In this second example, we have two goals, (G1) and (G2), both of which are permissions. We will demonstrate in Section 4.2 how deontic logic policies can be used to link to the goals of the above two scenarios, using our new extension of *GeoJSON policies*.

4 GEOJSON POLICIES

We propose here an extension to the GeoJSON standard by including a new "foreign member" (Butler et al., 2016, §6.1), which we use to express a *policy associated with some geofence*. This foreign member is expressed as the following new name-value pair:

$$\texttt{"policy"} : \{\theta_1, \ldots, \theta_n\}$$

where $\theta_1, \ldots, \theta_n$ is a set of deontic logic rules of the following form:

$$\theta = C \vdash op(f)$$

such that whenever a necessary logical condition C becomes valid (i.e. true), then $op(f(x_1, \ldots, x_m))$ becomes activated, where f is some *functionality* offered by either the geofence controller or the entity interacting with the geofence, $(x_1, \ldots, x_m)$ are m number of data points (i.e. parameters) that f is

applied to or requires and finally, *op* is a deontic logic operator of one of the following three forms:

- $\mathcal{O}$ (Obligation): where $\mathcal{O}(f)$ means "it ought to be f". Obligations are the only fundamental operators, through which both permissions and prohibitions can be derived using the principle of duality in logic (e.g. see (McCarty, 1983))
- $\mathcal{P}$ (Permission): where $\mathcal{P}(f)$ means that "f is permitted if not f is not obligatory"
- $\mathcal{F}$ (Prohibition): where $\mathcal{F}(f)$ means that "f is forbidden if not f is obligatory"

So, for example, $C \vdash \mathcal{P}(f(x_1, \ldots, x_m))$ means that function "$f(x_1, \ldots, x_m)$ is *permitted* whenever condition C is true" and that this statement is true. We refer to the data $(x_1, \ldots, x_m)$ as X_f.

From the point of view of geofencing, when a host enters/exits, crosses or steps over a geofence, the deontic logic policy associated with that geofence becomes activated, and various functions f stated in the policy become permitted, prohibited or obligated, depending on the deontic semantics associated with those functions and whether their necessary logical conditions are true.

We associate the proposed policy member in GeoJSON to the `"Feature"` object, where the `"policy"` member value will apply to the `"geometry"` member of that specific Feature object. Hence:

```
{
 "type": "Feature",
 "geometry": { ... },
 "properties": { ... },
 "id": "...",
 "policy": { ... }
}
```

Depending on the type of the geometry that defines the geofence, the specified policy will trigger each time an entity enters/exits, crosses over or steps on the geofence. In the case that the geometry of the geofence has multiple parts (i.e. it is a multi-point, a multi-line string, a multi-polygon or indeed multiple geometries), the policy will apply equally to all the individual parts stated under the specific Feature. We do not deal with geometric collections in this paper, and assume for simplicity a single type geometry. For an explanation of all the standard elements of the GeoJSON syntax, we refer the reader to the original specification document (Butler et al., 2016).

4.1 Linking Goals

We mentioned in the previous section that a scenario's aim is to produce a number of end-user goals, as in Jane or Mark's goals above. Here, we redefine the notion of a goal as a *logical formula* (e.g. as in the formalisation of KAOS (van Lamsweerde, 2000) goals in (Darimont and Van Lamsweerde, 1996)), where we express a single goal as $g(X_g)$, defined over a subset $X_g \subseteq X_f$ of the same data points (parameters) as the ones to which policy rule functions apply. Naturally, when $g(X_g)$ evaluates to $\mathbf{T} \in \mathbb{B}$, then that signifies that the goal has been satisfied, but when $g(X_g) = \mathbf{F} \in \mathbb{B}$, then that signifies that the goal has *not* been satisfied. This set-up of using the same data points as inputs to the policy rules and to goals at the same time provides the important *expressive link* between the end-user requirements and the policy rules.

4.2 Examples

We now give a couple of examples of the use of deontic logic policies to express desirable bahaviour when an entity interacts with a geofence. We focus only on basic geometric shapes: a point, a line and a polygon.

4.2.1 Stepping on a Point

Our first example of the usage of the new `policy` member in GeoJSON is in association with the Point (i.e. one-dimensional) geometry. We use this policy to express Jane's toilet usage goal ($G4$). First, let's assume our geofence has a single point geometry:

```
{
 "type": "Feature",
 "geometry": {
 "type": "Point",
 "coordinates": [
        -51.91843809141983,
         70.90835554088363]
         },
 "policy": {G4theta}
}
```

where:

$\texttt{G4theta} = \mathbf{T} \vdash \mathcal{O}(point\_step\_on\_counter{+}{+})$

The policy has an *obligation* rule that obliges the counter called *point_step_on_counter* to be incremented each time the point at coordinates (-51.91843809141983, 70.90835554088363) (somewhere on Appat island, Greenland) is stepped on by some entity (Jane here). Stepping on this point will trigger the policy and the rule within it. As a result, this policy can be used to fulfill Jane's goal $G4$, by counting the number of times a dot sensor placed near the toilet seat on the ground is being stepped on. Note that the definition of `G4theta` does not require C to

be of anything else other than true, since the rule is assumed to be activated whenever the point is stepped on (the only necessary condition.)

Now, we can rewrite $G4$ as an MTL-$\int$ logic (Lakhneche and Hooman, 1995) formula as follows:

$$G4 \quad \Box \Diamond_{\leq 24} (point\_step\_on\_counter \geq 3)$$

Note that $G4$ is directly linked with the policy rule through the '*point_step_on_counter*' data point. The above formula states that it is always the case that within any 24hr period, that the pointer will eventually have at least a value of 3 (i.e. Jane has used the toilet at least 3 times per day).

Point-based policies are also useful in several other scenarios in the real world. For example, in structural wear-and-tear testing, manufacturers of flooring materials (e.g. tiles, carpets) can place sensors at single points to measure wear and tear in high-traffic areas and assess material longevity. In grocery stores or fast-food lines, sensors at single points can track customer footfalls to measure line congestion and waiting times. In athletics, such as high and long jumps or sprinting, tracking exactly where an athlete steps can be critical for improving technique and performance.

4.2.2 Crossing a Line

For our second example, we assume the geofence has a line (i.e. two-dimensional) geometry, near the same location on Appat island where we had our point in the previous section. For example, let's consider a policy to express Jane's goal $G3$:

```
{
 "type": "Feature",
 "geometry": {
 "type": "LineString",
 "coordinates": [
        [-51.91844745606619,
          70.90835709478952],
        [-51.91846539333858,
          70.90835177617504],
    ]},
 "policy": {G3theta,G3theta1,G3theta2,
            G3theta3}
}
```

In this case, the policy is defined by four rules, as follows:

$$\texttt{G3theta} = out(person, bedroom) \wedge on(bedroom, lights) \vdash \mathcal{F}(switch\_off(bedroom, lights))$$

$$\texttt{G3theta1} = out(person, bedroom) \wedge off(bedroom, lights) \vdash \mathcal{O}(switch\_on(bedroom, lights))$$

$$\texttt{G3theta2} = in(person, bedroom) \vdash \mathcal{P}(switch\_on(bedroom, lights))$$

$$\texttt{G3theta3} = in(person, bedroom) \vdash \mathcal{P}(switch\_off(bedroom, lights))$$

The first rule, `G3theta`, prohibits the lights of the bedroom from being switched off (using the function *switch_off*) when the condition of a person, being *out* of the bedroom and the lights being on, is satisfied. On the other hand, rule `G3theta1` considers the same scenario but where the lights are off, in which case it is obligated that that the lights be switched on. Finally, both rules `G3theta2` and `G3theta3` consider the scenario when the person is *in* their bedroom, in which case the lights are permitted to be either switched on or switched off, respectively. We must note here that *in* and *out* are predicates on the state of the person (being in or out of the bedroom), and not on the fact that the line is crossed in some direction as there is no *in* or *out* concepts for one-dimensional geometries.

Using the *bedroom* and *lights* data points, we can rewrite $G3$ for Jane, as follows:

$$G3 \quad \Box_{[19:00,07:00]} (out(Jane, bedroom) \rightarrow on(bedroom, lights))$$

The goal states that during evening time (defined here as the period between 7pm and 7am), it is always the case that if Jane is *out* of her bedroom, then this means that the bedroom lights will be *on*. Naturally, it is always the case that: $on(bedroom, lights) \oplus off(bedroom, lights)$ is true, i.e. that the lights are either on or off, but not both or neither at the same time.

Policies for line-crossing geometries in GeoJSON can be useful in other scenarios in the real world in addition to the above example. In urban planning, line-crossing policies can be used to prevent roads or railways from incorrectly intersecting or issuing an alarm when crossed, ensuring safety of drivers and pedestrians. In utility infrastructure mapping, such policies can help to avoid dangerous overlaps between power lines, pipelines or water systems. Finally, in the context of maritime shipping, ship lanes can be maintained separate using policies, and ensure these do not cross underwater cables or marine protected areas.

4.2.3 Entering and Existing a Polygon

Finally, let's consider the example of a polygon geofence setup. Somewhere not far away from the previous location on Appat island in Greenland where the point and the line geofences were located inside Jane's apartment, we now define a polygon geofence around the local market where Mark has his stall:

```
{
 "type": "Feature",
 "geometry": {
 "type": "Polygon",
 "coordinates": [
        [
            [-51.918944960471464,
              70.90845379535514],
            [-51.91907047121319,
              70.90846268556521],
            [-51.91899532465419,
              70.90835312652129],
            [-51.918805859393046,
              70.90843679935406],
            [-51.918944960471464,
              70.90845379535514]
        ]
    ]},
 "policy": {G1theta,G2theta}
}
```

The geofence has two policy rules, one for each of Mark's two goals above. These are defined as follows:

$\texttt{G1theta} = \mathit{enters}(\mathit{customer}, \mathit{market}) \vdash \mathcal{P}(\mathit{send}(\mathit{discount}, \mathit{customer}))$
$\texttt{G2theta} = \mathit{exits}(\mathit{customer}, \mathit{market}) \vdash \mathcal{P}(\mathit{receive}(\mathit{feedback}, \mathit{customer}))$

The first rule, `G1theta`, permits Mark to *send* a promotion code to customers *entering* the polygon geofence of the market. The second rule, `G2theta`, permits Mark to receive feedback from customers *exiting* the polygon geofence of the market. Note that the rules use an *enters* predicate and an *exits* predicate as additional conditions to determine the directionality of the polygon event, which is not possible to determine simply based on GeoJSON's definition.

We can use these rules and the data points, *discount*, *market* and *customer*, to rewrite Mark's goals using temporal logic (Prior, 1957) as follows:

$G1 \quad \mathit{sent}(\mathit{discount}, \mathit{customer}) \rightarrow \blacklozenge\, \mathit{enters}(\mathit{customer}, \mathit{market})$

$G2 \quad \mathit{received}(\mathit{feedback}, \mathit{customer}) \rightarrow \blacklozenge\, \mathit{exists}(\mathit{customer}, \mathit{market})$

Both rules rely on the $\blacklozenge$ temporal operator to state that sending of promotions or receiving of feedback implies that the customer *once entered or exited, in the past, at some point in time,* the market's perimeter.

In other contexts, polygon entering and exiting policies can be used to ensure a wide range of desirable properties and regulations. For example, in wildlife conservation, tracking animal movement in and out of protected zones helps researchers understand migration patterns and enforce anti-poaching regulations. In logistics, delivery route optimisation relies on detecting entry and exit points into delivery areas to ensure timely and legal routing. In security applications, geofencing can trigger alerts when vehicles or individuals enter or leave restricted areas, such as military bases or private facilities. Similarly, in agriculture, monitoring machinery or drone movement in and out of specific crop zones helps manage operations and ensure efficient resource usage.

4.3 Consistency of Deontic Logic Rules

The deontic logic policies associated with geofences must obey a number of basic consistency properties, in order to ensure that no conflicts arise from combining the rules within these policies.

Property 1 (Permitted Functionality)**.** *A permitted functionality associated with a geofence must not be prohibited under the same condition. This is expressed as follows:*

$$(C \vdash \mathcal{P}(f)) \Rightarrow \neg(C \vdash \mathcal{F}(f)) \qquad \square$$

This is a sort of *liveness* property, which ensures that under a certain condition, C, a permitted action, f, can not stopped.

Property 2 (Prohibited Functionality)**.** *A prohibited functionality associated with a geofence must not be permitted under the same condition. This is expressed as follows:*

$$(C \vdash \mathcal{F}(f)) \Rightarrow \neg(C \vdash \mathcal{P}(f)) \qquad \square$$

This is a sort of safety property, which ensures that a forbidden action, f, is not allowed under the same condition, C.

Property 3 (Obligated Functionality)**.** *An obligated functionality associated with a geofence must be permitted (and therefore, by implication, not prohibited) under the same condition. This is expressed as follows:*

$$(C \vdash \mathcal{O}(f)) \Rightarrow (C \vdash \mathcal{P}(f)) \qquad \square$$

This final property is also related to ensuring liveness, being ensuring that an obliged action, f, is also a permitted action, under the same condition, C.

In all of the above, we write $P \Rightarrow Q$ to mean $\square\,(P \rightarrow Q)$.

5 CONCLUSION

We proposed in this paper an extension of the RFC7946 GeoJSON standard (Butler et al., 2016) to incorporate an element expressing policies. We used this element to model deontic logic-based policies for three kinds of geometries: a point, a line and a polygon. We showed how the extension can be useful in controlling behaviour for a couple of example scenarios related to care at home and local market shopping.

Our extension of the GeoJSON standard with the `policy` element is generic enough to encompass other (i.e. non-deontic) forms of policies, e.g. access control or usage control policies, as well as other (i.e. non-set-theoretic) formal languages for expressing such policies. Our approach is also general in the sense that it can apply to other geofencing and geo-location standards, such as TopoJSON (TopoJSON, 2018) and GPX (GPX, 2023), without the need to change the model of the policies used.

There are several directions we are planning to investigate in the future related to this work. First, we plan to formalise more rigorously the language used for defining deontic policies, and also to expand this to other models of policies with focus on safety versus liveness properties (Alpern and Schneider, 1987). Another direction is to embed and evaluate the applicability of the current Policy element in the context of a healthcare software and conduct a real-world case study to validate the proof-of-concept. This would allow us to understand better the limitations of the model and how best to improve it.

REFERENCES

Alpern, B. and Schneider, F. B. (1987). Recognizing safety and liveness. *Distributed computing*, 2(3):117–126.

Bray, T. (2017). The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259.

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., and Hagen, S. (2016). The GeoJSON Format. RFC 7946.

Chantaweesomboon, W. (2021). Bluetooth geo-fence for elderly and patient care. In *2021 25th International Computer Science and Engineering Conference (ICSEC)*, pages 252–255. IEEE.

Darimont, R. and Van Lamsweerde, A. (1996). Formal refinement patterns for goal-driven requirements elaboration. *ACM SIGSOFT Software Engineering Notes*, 21(6):179–190.

Gilmore, J. N. (2020). Securing the kids: Geofencing and child wearables. *Convergence*, 26(5-6):1333–1346.

GPX (2023). GPX: the GPS Exchange Format . `https://www.topografix.com/gpx.asp`. Last accessed: 12.05.2025.

Hansen, J., Pigozzi, G., and Van Der Torre, L. (2007). Ten philosophical problems in deontic logic. In *Dagstuhl Seminar Proceedings 07122*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Helmy, J. and Helmy, A. (2017). Demo abstract: Alzimio: A mobile app with geofencing, activity-recognition and safety features for dementia patients. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 994–995. IEEE.

Hill, J. R., Min, E. E., Abebe, E., and Holden, R. J. (2024). Telecaregiving for dementia: A mapping review of technological and nontechnological interventions. *The Gerontologist*, 64(1):gnad026.

Howell, G., Franklin, J. M., Sritapan, V., Souppaya, M., and Scarfone, K. (2023). Guidelines for managing the security of mobile devices in the enterprise. Technical report, National Institute of Standards and Technology.

Kannan, V., Basit, M. A., Bajaj, P., Carrington, A. R., Donahue, I. B., Flahaven, E. L., Medford, R., Melaku, T., Moran, B. A., Saldana, L. E., et al. (2019). User stories as lightweight requirements for agile clinical decision support development. *Journal of the American Medical Informatics Association*, 26(11):1344–1354.

Lakhneche, Y. and Hooman, J. (1995). Metric temporal logic with durations. *Theoretical Computer Science*, 138(1):169–199.

Matheus, A. (2023). OGC Geospatial eXtensible Access Control Markup Language (GeoXACML) 3.0. Open Geospatial Consortium.

McCarty, L. T. (1983). Permissions and obligations. In *International Joint Conferences on Artificial Intelligence*, volume 83, pages 287–294.

Nguyen, K. T., Olgin, J. E., Pletcher, M. J., Ng, M., Kaye, L., Moturu, S., Gladstone, R. A., Malladi, C., Fann, A. H., Maguire, C., et al. (2017). Smartphone-based geofencing to ascertain hospitalizations. *Circulation: Cardiovascular Quality and Outcomes*, 10(3):e003326.

Prior, A. N. (1957). *Tme and Modality*. OUP Oxford.

Rissanen, E. (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard.

Schulzrinne, H., Tschofenig, H., Cuellar, J. R., Polk, J., Morris, J., and Thomson, M. (2013). Geolocation Policy: A Document Format for Expressing Privacy Preferences for Location Information. RFC 6772.

Tarameshloo, E. and Fong, P. W. (2014). Access control models for geo-social computing systems. In *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies*, SACMAT '14, page 115–126, New York, NY, USA. Association for Computing Machinery.

Tobin, K., Heidari, O., Volpi, C., Sodder, S., and Duncan, D. (2023). Use of geofencing interventions in population health research: a scoping review. *BMJ Open*, 13(8).

TomTom (2019). TomTom's Geofencing API. https://developer.tomtom.com/geofencing-api/documentation/product-information/introduction. Last accessed 01.05.2025.

TopoJSON (2018). The TopoJSON Format Specification.

https://github.com/topojson/topojson-specification. Last accessed 12.05.2025.

Turner, A. M., Reeder, B., and Ramey, J. (2013). Scenarios, personas and user stories: User-centered evidence-based design representations of communicable disease investigations. *Journal of biomedical informatics*, 46(4):575–584.

Ullah, F., Haq, H. U., Khan, J., Safeer, A. A., Asif, U., and Lee, S. (2021). Wearable iots and geo-fencing based framework for covid-19 remote patient health monitoring and quarantine management to control the pandemic. *Electronics*, 10(16):2035.

van Lamsweerde, A. (2000). Requirements Engineering in the Year 00: A Research Perspective. In *International Conference on Software Engineering*, pages 5–19.

Van Riemsdijk, M. B., Jonker, C. M., and Lesser, V. (2015). Creating socially adaptive electronic partners: Interaction, reasoning and ethical challenges. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pages 1201–1206. Citeseer.

Von Wright, G. H. (1951). Deontic logic. *Mind*, 60(237):1–15.

W3C (2017). Geofencing API. `http://w3c.github.io/geofencing-api/`. Last accessed 01.05.2025.

Winkels, R., Hoekstra, R., and Hupkes, E. (2010). Normative reasoning with geo information. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, New York, NY, USA. Association for Computing Machinery.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS