Do ”Web of Things Platforms” Truly Follow the Web of Things?

Andreas Kamilaris and Muhammad Intizar Ali

Insight Centre for Data Analytics
National University of Ireland, Galway
Emails: andreas.kamilaris@insight-centre.org, ali.intizar@insight-centre.org

Abstract—The Web of Things (WoT) aspires to bring interoperability at the application layer, on top of the Internet of Things. Many state of the art platforms and frameworks claim to support the WoT, following its principles towards the seamless integration of heterogeneous physical devices and real-world services at the web. But do these platforms truly comply to the concepts of the WoT or only follow some of its characteristics? Do designers understand the WoT when claiming that their products follow the WoT specifications? This paper lists the main elements of the WoT, as defined by pioneering works in the field and examines 26 popular platforms and frameworks, aiming to shed light on how the WoT is understood and applied, both in academia and commerce.

Keywords—Web of Things; Platforms; Review;

I. INTRODUCTION

While the Internet of Things (IoT) [1] enables communication between heterogeneous devices at the network layer, the Web of Things (WoT) envisions interoperability at the application layer, by reusing well-accepted and understood web standards [2], [3], [4], [5]. Since 2007, when the WoT was initially proposed [2], numerous platforms and frameworks have appeared, both in academia and in business, claiming to support and apply the concepts of the WoT, demonstrating web-based operation in real-world environments such as smart homes, buildings and smart cities [6], [7], [8] employing various embedded devices which become enabled to the web.

But do these platforms truly comply to the WoT principles, or just adopt some of its characteristics? Which are the foundational elements that allow some projects or initiatives to be WoT-enabled or based on the WoT? Do researchers and entrepreneurs understand the WoT when claiming that their products follow WoT specifications?

In this paper, we aim to shed light on how the WoT is used in platforms declaring to support the WoT, by examining 26 popular relevant projects and initiatives. Our contribution is to show how the WoT is perceived in research and in commerce, how it is being applied and which are the trends of this practice.

Rest of the paper is organized as follows: Section II describes the methodology we followed while Section III presents and analyzes our findings. Then, Section IV discusses our overall results and Section V concludes the paper.

II. METHODOLOGY

Our methodology consists of four steps:
1) Identify the basic elements of the WoT, based on the state of art literature and pioneering work in the field.
2) Locate the most popular platforms and frameworks which claim to be ready for the WoT.
3) Analyze these platforms according to each of the foundational WoT elements, as identified in Step 1.
4) Consider each of the basic elements of the WoT one by one, examining how they have been addressed/satisfied by those platforms.

In the first step, we examined the main bibliography describing the WoT, its elements, principles and characteristics [2], [3], [4], [5], [9]. Based on this, we identified the most important elements of the WoT, which are indicated to be followed by any WoT-based initiative. Twelve such elements were identified, listed in Section III.

In the second step, we performed a keyword-based search in well-known scientific databases (e.g. IEEE Xplore, ACM, DBLP, ScienceDirect, CiteSeerX) and search engines, using keywords such as “Web of Things”, “platforms”, “frameworks” and combinations of these. We selected the most popular papers, where popularity was defined as those papers having more than 15 citations. We focused on works with actual implementations, not just theoretical papers. In this way, we collected 21 papers in total. Then, by including well-known large initiatives from the enterprise world [7], [10], [11], [12], [13] that claim to provide WoT-based services, we increased our list to 26 initiatives, frameworks, platforms and projects, referred as platforms in the rest of the paper.

In the third step, we examined each platform one by one, observing and recording how it addresses each of the basic elements of the WoT, as identified in the first step.

In the final step, we considered how the basic principles of the WoT are applied and realized by these platforms, discussing protocols and technologies adopted, as well as missing elements and incomplete characteristics.

III. ANALYSIS

By examining the main bibliography on the architecture of the WoT [2], [3], [4], [5], [9], we identified 12 elements which need to be followed by WoT platforms:
E1 **Device discovery**, either locally at the same physical space or globally through the web.

E2 **Integration of things to the web**, either directly by embedding web servers on them [14] or indirectly by means of gateways [4].

E3 **RESTful interaction with the things**, based on the REST architectural style [15], [16] and the four main operations, GET, PUT, POST, and DELETE.

E4 **RESTful interaction with the platform**, from clients and end-users.

E5 **Data formats** which are well-known, accepted and widely used on the web.

E6 **Multiple representations** of the responses provided by the things and the platform, in various data formats.

E7 **Security** regarding authorization on the platform, message encryption and integrity among thing-platform, thing-user and platform-user.

E8 **Service semantics**, for describing services offered by physical things and interaction with them through REST.

E9 **Data semantics**, for understanding the semantics of the data provided by the things, facilitating parsing and integration with other web data sources.

E10 **Physical mashups**, defined as web mashups involving and combining physical entities together with web services and web content [17].

E11 **Sharing** of physical things and their services among end users, based on trusted online contacts or online social media relationships.

E12 **Syndication techniques and/or web messaging.** Syndication can be a RESTful model for interacting with collections of things in a pull basis. Web messaging can also take the form of push notifications based on long-lasting HTTP interactions and web hooks\(^1\) or RESTful publish/subscribe infrastructures.

According to bibliography, some elements are more important for a true WoT-based interaction (E2-E6), while some elements (E10-E12) can be considered more as additional features than as core functionality. Table I lists the 26 identified platforms (first column) in relation to the 12 elements (first row) characterizing the WoT as defined above. In the following subsections, we analyze how each of the aforementioned WoT basic elements has been applied and used by these platforms.

**E1. Device discovery:** Device discovery seems to be the “Achilles heel” of WoT, as the large majority of these platforms (21 out of 26) do not implement or adopt any discovery protocol. From the few platforms dealing with discovery of devices, almost all focus on local device discovery at the same physical space. TinyREST [18] and Dynamix [19] use UPnP\(^2\), while SOCRADES [20] uses WS-Discovery\(^3\). HomeWeb [6] has implemented a lightweight equivalent of WS-Discovery, more appropriate for the WoT. It is remarkable that none of the platforms employed the Service Location Protocol\(^4\) (SLP), Sun’s Jini, Apple’s Bonjour, mDNS\(^5\) or DNS-SD\(^6\), while are popular local discovery protocols at the IoT realm. The only platform focusing on global discovery of devices on the web is SpitFire [8], which employs an interesting discovery technique based on web crawlers. A recent survey on search techniques at the WoT [21] lists numerous (global) approaches proposed for discovering things and their data on the web. However, these approaches have not yet been adopted by any of the 26 platforms, for dynamic discovery of devices on the web.

**E2. Integration of things to the web:** The requirement of enabling things to the web, either by embedding web servers on them or by means of gateways, is satisfied only by a subset of the platforms (10 out of 26). pREST [22], SOCRADES, Akribopoulos et al. [23] and the VO Framework [24] employ embedded HTTP servers on SunSPOTs and Arduinos. HomeWeb and SpitFire use the 6LoWPAN [25] and CoAP [26] protocols to enable things to the web directly. SenseWeb [27] and Social Access Controller (SAC) [28] use gateways while Evrything [10] has developed its own embedded device toolkits for helping developers build their own WoT applications.

**E3. RESTful interaction with the things:** Ten out of the 26 platforms suggest a RESTful interaction with WoT devices, either directly or through (dedicated) gateways. These are mainly the platforms mentioned in the previous paragraph for integrating things to the web. Apparently, gateways are the default option for constrained things such as RFID tags, QR codes and barcodes. SenseWeb supports interaction through WS-* services [29] and SOCRADES offers also DPWS-based support. The rest 16 platforms offer only indirect ways to interact with physical devices through RESTful interfaces.

**E4. RESTful interaction with the platform:** The requirement most satisfied by the platforms (18 out 26), as they provide RESTful APIs for users to interact with WoT devices/services and data collected. SenseWeb and SensorBase [30] offer interaction based on WS-*. There is an ongoing discussion whether REST or WS-* is more appropriate for WoT [31], with pioneers in the field [2], [3] indicating REST as the suggested style to follow, being a lightweight alternative of WS-*, more suitable for resource-constrained environments.

**E5. Data formats:** Most platforms employ data formats which are well-known, accepted and widely used on the web, such as XML, JSON, CSV and HTML. XML is the most widely used (17 out of 26 platforms). Other data formats used include Atom (Sensorpedia [32], WebPlug [33] and Xively [7]), XHTML (SOCRADES), GeoRSS (Sensorpedia), KML (WoTKit [34]), GeoML (WoT framework for CPS [35]), EEML and RSS (Xively). RDF is employed in Sense2Web [36], in the WoT framework for CPS and in SpitFire. This data format is used when employing Semantic Web technologies [37]. Finally, we note that SenseWeb, Open.Sen.se [11] and ThingWorx [12] use only plain text, which is not the indicated way in WoT for data exchange.

---

<table>
<thead>
<tr>
<th>No.</th>
<th>Platform / WoT Element</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
<th>E5</th>
<th>E6</th>
<th>E7</th>
<th>E8</th>
<th>E9</th>
<th>E10</th>
<th>E11</th>
<th>E12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>TinyREST [18]</td>
<td>UpnP</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Unknown</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2.</td>
<td>pREST [22]</td>
<td>No</td>
<td>IP stack for 8-bit architectures</td>
<td>Yes</td>
<td>Yes</td>
<td>XML, text</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>RDF, OWL</td>
<td>No</td>
<td>No</td>
<td>SUBSCRIBE requests</td>
</tr>
<tr>
<td>3.</td>
<td>SOCRADES [20]</td>
<td>WS-Discovery</td>
<td>Web server on SunSPOT's</td>
<td>Yes (+ DPWS)</td>
<td>No</td>
<td>XHTML</td>
<td>No</td>
<td>No</td>
<td>DPWS, WSOL</td>
<td>No</td>
<td>No</td>
<td>WS-Notification</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>HomeWeb [38], [6], [39]</td>
<td>WS-Discovery</td>
<td>6LoWPAN</td>
<td>Yes</td>
<td>Yes</td>
<td>XML, JSON</td>
<td>Yes</td>
<td>Basic authentication</td>
<td>WADL, HTML links</td>
<td>No</td>
<td>Mashup Editor</td>
<td>Yes</td>
<td>Web hooks</td>
</tr>
<tr>
<td>5.</td>
<td>SenseWeb [27]</td>
<td>No</td>
<td>No (Use of gateway)</td>
<td>No (WS-*)</td>
<td>WS-* Text</td>
<td>No</td>
<td>Basic authentication</td>
<td>No</td>
<td>No</td>
<td>Sensor Map</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Sense2Web [36]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>XML, RDF</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>RDF, SPARQL</td>
<td>Sensor Map</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Sensor.Network [40]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>CSV, XML, JSON</td>
<td>Yes</td>
<td>API key, HTTPS</td>
<td>API Documentation</td>
<td>XML descriptors</td>
<td>LivePlots</td>
<td>No</td>
<td>Web hooks</td>
</tr>
<tr>
<td>8.</td>
<td>SensorBase [30]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>WS-* CSV, XML, JSON</td>
<td>Yes</td>
<td>Basic authentication</td>
<td>No</td>
<td>SensorML</td>
<td>Line plots, mapview</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Sensorpedia [32]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Atom, GeoRSS</td>
<td>No</td>
<td>OpenID, OAuth</td>
<td>No</td>
<td>No</td>
<td>Sensor Map</td>
<td>No</td>
<td>ATOMPub</td>
</tr>
<tr>
<td>10.</td>
<td>SAC [28]</td>
<td>No</td>
<td>No (Use of gateway)</td>
<td>Yes</td>
<td>Yes</td>
<td>CSV, XML, JSON</td>
<td>Yes</td>
<td>OAuth</td>
<td>HTML links</td>
<td>No</td>
<td>Sensor Map</td>
<td>Yes</td>
<td>ATOMPub</td>
</tr>
<tr>
<td>11.</td>
<td>WoT framework for CPS [35]</td>
<td>No</td>
<td>Unknown</td>
<td>Yes</td>
<td>Yes</td>
<td>CSV, XML, RDF, GeoML</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>WebPlug [33]</td>
<td>No</td>
<td>Yes</td>
<td>No (No implementation)</td>
<td>Yes</td>
<td>Yes</td>
<td>Atom, JSON, HTML, CSV</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Collections, URL callbacks</td>
</tr>
<tr>
<td>13.</td>
<td>AutoWoT [41]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>JSON, XML, HTML</td>
<td>Yes</td>
<td>No</td>
<td>Microformats</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>SenseBox [42]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>XML</td>
<td>No</td>
<td>No</td>
<td>OGC SWE</td>
<td>OGC O&amp;M</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>Akribopoulos et al. [23]</td>
<td>No (basic broadcast) Nano HTTP server</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>XML</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>WiseML</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>SemSense [43]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>XML</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>RDF, SPARQL</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>17.</td>
<td>VO Framework [24]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>EML, CSV, JSON, RSS, Atom</td>
<td>Yes</td>
<td>Basic authentication, API key</td>
<td>SDK, API Documentation</td>
<td>No</td>
<td>Widgets, mashup pipes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>18.</td>
<td>WoTKit [34]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>CSV, KML, HTML, JSON</td>
<td>Yes</td>
<td>No</td>
<td>API Documentation</td>
<td>No</td>
<td>Widgets, mashup pipes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>19.</td>
<td>GaaS [44]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>JSON, ATOM, XML</td>
<td>Yes</td>
<td>No</td>
<td>WS-DL</td>
<td>No</td>
<td>WS-BPEL adaptation</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>20.</td>
<td>Dynamix [19]</td>
<td>UpnP, AirPlay</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Images</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td>Xively [7]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>EML, CSV, JSON, RSS, Atom</td>
<td>Yes</td>
<td>Basic authentication, API key</td>
<td>SDK, API Documentation</td>
<td>No</td>
<td>Line plots, mapview</td>
<td>No</td>
<td>MQTT broker</td>
</tr>
<tr>
<td>22.</td>
<td>Evrything [10]</td>
<td>No</td>
<td>No</td>
<td>Embedded device toolkit</td>
<td>Yes</td>
<td>Yes</td>
<td>CSV, XML, JSON</td>
<td>Yes</td>
<td>OAuth, TLS, DTLS, API key</td>
<td>SDK, API Documentation</td>
<td>No</td>
<td>Visual tool, Node.js rules</td>
<td>No</td>
</tr>
<tr>
<td>24.</td>
<td>ThingWorx [12]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Text</td>
<td>No</td>
<td>Basic auth., API key</td>
<td>No</td>
<td>Mashup Builder</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>25.</td>
<td>Parainput [13]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>CSV, XML, CSS</td>
<td>Yes</td>
<td>HTTPS, API key</td>
<td>API Documentation</td>
<td>No</td>
<td>Mashup Composer</td>
<td>Yes</td>
<td>Web hooks</td>
</tr>
<tr>
<td>26.</td>
<td>SpitFire [8]</td>
<td>Web crawlers</td>
<td>6LoWPAN, CoAP</td>
<td>Yes</td>
<td>No</td>
<td>XML, text, RDF</td>
<td>Yes</td>
<td>No</td>
<td>Sensor meta-data</td>
<td>RDF, OWL, SPARQL</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**TABLE I**

WoT-based platforms and their relationship to the elements characterizing the WoT.
E6. Multiple representations: A powerful element of the WoT is the opportunity to negotiate among multiple representations of the HTTP responses from things, their gateways or the platforms themselves. Negotiation should be based on various well-known data formats used on the web, such as the ones listed in the previous paragraph. From the 26 platforms, only 16 satisfy this important requirement.

E7. Security: The WoT suggests to follow well-understood web security protocols. However, 14 platforms do not consider security at all, while HomeWeb, SenseWeb, SensorBase and Open.Sen.se offer only basic user authentication. Sensor.Network [40], Xively, Evrything, ThingWorx and Paraimpu [13] require an API key on the HTTP header, in order to call any RESTful service offered by them. Moreover, Sensor.Network and Paraimpu support HTTP Secure (HTTPS) and Sensorpedia supports OpenID7. OAuth 2.08, the open standard for authorization used as a way to log in to third party websites, is adopted by Sensorpedia, SAC and Evrything. A general observation here is that security is considered important mostly by the commercial platforms. Xively is the platform with the most complete security profile, supporting among others TLS and DTLS for privacy and data integrity during message communication.

E8. Service semantics: More than half (14 out of 26) platforms provide semantics for interacting with their APIs and devices. SOCRADES and GaaS [44] adopt WSDL9, which is the default service description language used in WS-*. HomeWeb proposes WADL10, which is a description language more suitable for HTTP-based applications. HomeWeb and SAC also use HTML links to navigate between services offered by WoT devices. Furthermore, AutoWoT [41] adopts microformats11, which is about meta-content which can be parsed and understood by machines, web crawlers and search engines. SenseBox [42] prefers the OGC Sensor Web Enablement12 (SWE) standards while SpitFire uses sensor meta-data to describe sensors’ services. Few research-based platforms (Sensor.Network, WoTKit), as well as all commercial platforms (Xively, Evrything, Open.Sen.se, ThingWorx and Paraimpu) provide detailed API documentations. Xively and Evrything move one step further, offering SDKs in various programming languages, facilitating end-user development of WoT apps. A general observation here is that description languages such as WSDL and WADL have expressive limitations for real-world objects while microformats require web crawling and website parsing. This is one of the reasons why business initiatives chose to provide clean API documentations and SDKs.

E9. Data semantics: Sixteen out of 26 platforms do not provide any semantic descriptions of the data exchanged between the things/platform and the users. SensorBase describes sensor data by SensorML13, Akribopoulos et al. use WiseML and Sensor.Network employs XML descriptors. The rest platforms (pREST, Sense2Web, SenseBox, SemSense, VO Framework, Evrything and SpitFire) employ Semantic Web technologies [37] such as RDF14, OWL15, SPARQL16 as well as the OGC Observations and Measurements17 (O&M) standard. The commercial platforms do not employ semantics, with the exception of Evrything. This indicates that semantic technologies are not yet mature to be largely used in business initiatives. The authors believe though that the Semantic Web is an important element of the WoT, towards advanced interoperability and seamless data and software integration, eventually leading to a Semantic Web of Things (SWoT) [45].

E10. Physical mashups: An appealing aspect of WoT is that web mashups can be created in any programming language supporting HTTP communication, including RESTful WoT services. Fifteen platforms provide some mashup functionality (e.g. sensor maps, plots, widgets), however, only HomeWeb, WoTKit, GaaS, Evrything, Open.Sen.se, ThingWorx and Paraimpu offer mashup builders/composers. Mashup editors seem to be recognized by the commercial initiatives as an integral feature provided to their end users.

E11. Sharing: Sharing of WoT devices and services among online contacts is a practice not adopted by most platforms, except from HomeWeb (see [46], [38]), SAC and Paraimpu. HomeWeb shares things and their services among Facebook friends, SAC delegates access to things based on Facebook and Twitter APIs (to locate trusted contacts) while Paraimpu shares things among other Paraimpu users. The authors argue that for a true WoT experience, things should be shared based on web authentication APIs and/or social network APIs, using protocols such as OpenID and OAuth.

E12. Syndication and messaging: Fifteen platforms employ syndication and/or publish/subscribe messaging techniques for interacting with things. Sensorpedia and SAC use ATOMPub18 as a syndication protocol for publishing WoT resources. TinyREST and pREST use dedicated subscription services on things, but TinyREST violates REST by including the extra verb SUBSCRIBE in HTTP requests. The concept of Web hooks is employed in HomeWeb, Sensor.Network, SenseBox (push notifications), WebPlug, Open.Sen.se and Paraimpu. SOCRADES and GaaS select WS-Notification from WS-*) while VO Framework proposes the OSGi Eventing Service. Commercial platforms prefer more scalable (and still lightweight) publish/subscribe messaging brokers such as MQTT19 (Xively and Evrything) and XMPP20 (Open.Sen.se). An observation here is that most platforms understand the importance of push-based notifications when interacting with...

\[^{7}\text{OpenID. http://openid.net/}\]
\[^{8}\text{OAuth 2.0. http://oauth.net/2/}\]
\[^{9}\text{WSDL. https://www.w3.org/TR/wsdll}\]
\[^{10}\text{WADL. https://www.w3.org/Submission/wadl/}\]
\[^{11}\text{Microformats. http://microformats.org/}\]
\[^{13}\text{SensorML. http://www.opengeospatial.org/standards/sensorml}\]
\[^{14}\text{RDF. https://www.w3.org/RDF/}\]
\[^{15}\text{OWL. https://www.w3.org/OWL/}\]
\[^{16}\text{SPARQL. https://www.w3.org/TR/sparql11-query/}\]
\[^{17}\text{OGC O&M. http://www.opengeospatial.org/standards/om}\]
\[^{19}\text{MQTT. http://mqtt.org/}\]
\[^{20}\text{XMPP. https://xmpp.org/}\]
From the 12 basic WoT elements, as identified in Section III, most platforms satisfy E4-E6, which are the critical ones, but less than half support E2-E3 and E7, which are also quite important. From the less critical elements, E10 is supported mostly by commercial approaches, dealing with customers and novice end users. E12 is recognized as an important feature, supported by the majority of platforms, mostly through various publish/subscribe messaging techniques. E11 is provided by three platforms only, and is not considered as important by researchers and entrepreneurs. E1 is addressed by five platforms only, all of those dealing with local discovery of devices. Global discovery on the WoT is still an open issue [21]. Regarding description of services (E8), it is addressed mainly by the commercial approaches. E9 (data semantics) is supported mostly by the research initiatives, by means of Semantic Web technologies. This indicates that data semantics are still at a research stage, not mature yet for the real world.

Everythign is the most complete platform from the ones reviewed, even though it does not support a discovery protocol for the WoT nor sharing of things among users. It seems that none of the 26 platforms fully adopt more than 10 of the 12 basic elements identified. SemSense and Dynamix develop only two elements, TinyREST and Akribopoulos et al. support only three, while the WoT framework for CPS, Sense2Web, SensorBase, Sensorpedia, AutoWoT and WoTKit provide only five of the 12 elements. Hence, we claim that there do not exist any complete WoT platforms yet, covering the whole cycle from physical things to web clients, addressing effectively all the issues of discovery, interaction, service/data description, messaging, security, sharing and mashup creation. Hence, we suggest that academia and enterprise should use the “Web of Things” label more moderately, with some caution.

Table II lists the WoT Devices used by the 26 platforms under review, together with their supported wireless protocols. Most academic efforts employ sensor motes (Telosb, Micaz, SunSPOTs), smart outlets (Ploggs), Arduinos21 and proprietary low-cost hardware, mainly for platform testing and proof of concept. Enterprise approaches include more complete and elegant case studies and solutions in domains such as smart homes, lighting systems and smart farms, including support of commercial WoT-ready products (e.g. Nabaztag Rabbit22, Chumby23, Bromton24 bikes and Apple TV25). Concerning wireless protocols, the most popular ones in both research/commercial platforms are Bluetooth, Near-Field Communications (NFC), IEEE 802.15.4 and ZigBee, while telecommunications protocols (UMTS, 3G, GPRS) have been used as well in three cases. The trend seems to be towards more and more powerful.

23 Chumby. https://www.chumby.com/
The lack of standards in the field encourages the phenomenon of applying WoT concepts only partly. Nevertheless, we acknowledge efforts in the direction of standardization, such as the W3C WoT Interest Group and the Sensor Web interface for IoT SWG. There is still a long distance to cover though, before WoT basic elements become standardized (especially E1, E8 E9, E12). Hence, some misuse of the WoT label is expected to continue in the near future.

V. CONCLUSION

In this paper, we have examined how WoT is used in academia and commerce. To achieve this, we have identified 12 basic elements of WoT, as defined by pioneering work in academia and commerce. To achieve this, we have identified such as the W3C WoT Interest Group. We then selected 26 of the most popular platforms we acknowledge efforts in the direction of standardization, and which are the trends of this practice.

ACKNOWLEDGMENT

This research has been supported by Science Foundation Ireland (SFI) under grant No. SFI/12/RC/2289 and EU FP7 CityPulse Project under grant No.603095.

REFERENCES