Task Automation Services: Automation for the masses

Miguel Coronado and Carlos A. Iglesias
Universidad Politécnica de Madrid, Spain

Keywords: task automation, mashup, internet service, connected device, Ifttt, automation.

Abstract. A simple model of mashup technology for combining services and connected devices is now becoming popular. This model is frequently known as 'task automation' based on ECA (Event-Condition-Action) rules. The most popular online services that follow this approach are Ifttt and Zapier. In addition, this model is being followed by several mobile frameworks, such as on{x}, AutomateIt or Tasker, to automate how the phone deals with the incoming Internet events and phone sensors. This article outlines the features and components of task automation services, and proposes a generic architecture that supports the current challenges. Finally, as task automation services are a growing trend, this article surveys their characteristics, comparing existing platforms and discussing their evolution and future tendencies.

Introduction

Task automation permeates our daily lives, from the weather forecast that appears when the alarm clock rings, to the smartphone toast-notification that pops up every time we receive an incoming email. These *automations* orchestrate gadgets, Internet services and apps in a way that makes our life easier [1]. We are so accustomed to task automations, that sometimes it is hard to identify them, and even harder to realise that some years ago we used to perform those tasks manually.

While these predefined task automations are spreading across the web, a new user-centred fully-customizable approach is beating them all, the so-called task automation service (TAS). These services are typically web platforms or smartphone apps that provide a visual programming environment, where non-technical users can seamlessly create and manage their own personal automations [2]. The automation in these services takes the form of Event-Condition-Action rules that execute an action upon a certain triggering event i.e. "when *triggering-event* then do *action"*. In the former examples, the alarm clock and the incoming email would be the triggers, whereas querying the weather forecast and displaying a notification are the respective actions.

Some TASs such as Ifttt[1] [3] or AutomateIt[2] have become mainstream. In 2014, Ifttt reported more than 14 million web tasks created by end users. AutomateIt, an Android application, has more than 500,000 users. There are three success factors that explain their growing adoption. The first key factor is usability. TASs provide a simple-yet-powerful intuitive interface for programming task automations. Hence, users experiment almost no learning curve when they start using them. Secondly, customisability. TASs allow their users to program the automations they need. Although simple, automations are powerful. The capability of creating their own rules awakes in them a sense of control and immediacy. They get the automations they need when they need them. Thirdly, integration with existing Internet

[1] http://ifttt.com
[2] http://automateitapp.com

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

services. Thus, users can automate tasks that access the Internet services they already use and are familiar with.

Given the novelty of Tasks Automation Services, this work aims to shed light on them, presenting and exemplifying their main characteristics. Motivated by their relevance and penetration in the market, we survey some of the most prominent TASs and classify them according to different dimensions. The paper also defines a reference TAS architecture that identifies the key elements of TASs as well as their interactions. This architecture provides a common vocabulary and serves as a reference that can accelerate the development, adoption and evolution of TASs.

Scenario and Challenges

To better understand what a TAS might look like, consider the following scenario. Sarah uses a TAS every day, so she has defined a set of useful automations. Some automations notify her when something relevant to her happens, such as "when I'm mentioned on Twitter, send me an email" notify her when something relevant to her happens. Others, save her the bore of repeating a simple task, e.g. "when I'm tagged in a Facebook picture, save it to my Dropbox" or "convert incoming invoice emails to PDF and store them in my Evernote". Furthermore, Sarah also uses the smartphone app provided by the TAS. Once installed, the TAS can access several resources from her smartphone, so she can set up rules involving incoming calls, camera, Bluetooth or GPS among others. Rules such as "when my smartphone's battery level is under 10%, text my parents" or "when I get to work, lower the volume of my ringtone" take advantage of those capabilities. Moreover, the TASs feature Sarah enjoys most is the discovery of compatible services around –using smartphone communication capabilities such as Bluetooth. This feature can automatically integrate her SmartTV or her home automation lighting system with the TAS, allowing her to set rules for home automation such as "when my alarm clock rings, switch on the bedroom lights".

This brief journey with Sarah illustrates how services and sensors can be connected by means of automation rules defined with a single TAS. It aims to give a clear view of the functioning and main features of TASs, and it also outlines some challenges, such as embracing smartphone resources or auto discovery of services; we will address these challenges in the following sections. Now, we are ready to discuss the elements this scenario introduced.

TAS Components: Channels and execution profiles

Our scenario combines events from Internet services, Sarah's smartphone, and connected devices. These services and devices are managed by channels. We define *channels* as abstractions for receiving events or emitting actions to Internet services (i.e. web channels) and connected devices (i.e. device channels). Channels should be registered in a channel directory service provided by the TAS. In this way, users can activate available channels when programming automations.

Web channels

Many TASs rely on third party Internet services to supply a pack of useful, popular, user-tested channels. Hence, users benefit from using TASs to manage the services they are already subscribed to (e.g. Evernote, Gmail). As a result, TASs provide users with a new layer of control to manage their services and they are not required to migrate.

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

By analysing the behaviour of web channels, we identify three characteristic dimensions. Consider a user that wants to define a new automation rule. First of all, that user needs to grant the TAS access to the Internet service, usually by providing access credentials –this is what *privacy paradigm* defines. In addition, some channels require to be configured. This is the case of the weather forecast channel, in which users need to provide a location for the forecast –this is defined within the *configuration paradigm*. Finally, channels may behave differently triggering the rule or being the consequence that takes place, i.e. they may generate events, provide actions or both –this is the *input-output paradigm*.

From the point of view of their privacy paradigm, channels may be public or private. In order to activate a channel, and let the platform act on behalf of the users, they need to grant access to the service. In our former example, Sarah previously had to allow the TAS to access her Dropbox account and email inbox in order to manage her files and emails. When this authentication is required, the channel is private. The privacy paradigm defines who will have access to events and actions provided by the channel. Information regarding private channels are for the user's eyes only, and it is tailored to the user. On the contrary, channels that do not ask for authentication are public channels, and every user gets the same information when using them. This is the case of news feeds or weather forecast channels. The privacy paradigm also covers *private group channels*, where every group member receives all the events the channel generates. These channels are common in scenarios like home automation, where family members are likely to share home channels.

The configuration paradigm defines the setup needed to activate a channel –apart from authentication. Public channels usually require configuration for the sake of better user experience and also as a matter of efficiency. For instance, in our example to activate a weather channel Sarah provided the location where she lives. Hence, she will receive weather events related to that location. In general, private channels don't require configuration since they are already tailored to the user.

Finally, when activated, channels may generate events, provide actions or both. This is what the input-output (IO) paradigm defines. Events are changes in the state of the service, e.g. a new email on Sarah's inbox. On the other hand, channels may also offer action capabilities, e.g. switching on the bedroom light. IO paradigm also covers *pipe channels*: those that process the input to generate a different output, that can be wired to another channel, For instance, the PDF converter channel that saves the content of Sarah's email into a PDF file is then connected to the Evernote channel to store the file.

From an integration perspective, most of the efforts in offering a new web channel are related to implementing the protocol to communicate with the Internet service behind the channel. This is TAS administrators' duty, which depends on the availability of an API for the Internet service. The authorization process involved in accessing the Internet service API determines the privacy paradigm. Besides communication with the service, the TAS administrators define what events and actions will be offered as part of the IO paradigm as well as the configuration paradigm.

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

Device channels

In comparison to web channels, device channels manage data from the connected devices they manage. In Sarah's scenario, her home automation lighting system and the Smart TV provide device channels to control all the switches of her home and her Smart TV, respectively.

From a behavioural approach, device channels respond to the same three dimensions analysed for web channels. In addition, device channels implement two additional dimensions, the communication paradigm and the discovery paradigm. The *communication paradigm* defines how the communication between the devices and the channel will be carried out: wired, wireless, through the Internet, etc. It also defines on what conditions the channel is available. As opposed to web channels, which may be accessed from all around the world through Internet connection, access to device channels may depend on local aspects. These aspects are part of the communication paradigm. For instance, when using wireless communication, availability is subject to the distance between devices, i.e. being under coverage area or not. Finally, device channels may announce themselves so that they are available for automations as in Sarah's scenario. This is what the *discovery paradigm* defines. It provides standard operations and APIs to enable self-identification of devices, capabilities discovery, and access to device data using pre-defined message structures. As a result, the system provides a "plug and play" capability.

From the point of view of implementation, each TAS administrator decides which communication protocols will be supported in the platform. Each protocol has its own restrictions about range, power consumption, number of connected devices, etc.. They also provide mechanisms such as security and device discovery. To communicate two devices, they need to support the same protocol. However, this should not be an issue, as a device may implement several protocols. In fact, many devices are compatible with the most widespread protocols, e.g. WiFi, ZigBee, Z-Wave, even Bluetooth [4].

Rule Execution profiles

This section describes automation rules, which provide the logic to connect channels. As previously stated, TASs automations address simple Event-Condition-Action rules (ECA) [5]. The rule's event and action may come from the same or different channels. However, more complex rules could be devised: *multi-action rules* can execute several actions in parallel when the rule is executed; *multi-event rules* are triggered by a combination of events; and *chain rules* execute a list of actions in sequential order, so the output of an action may be used to trigger the next rule. Complex rules require the TAS to support additional features. For instance, multi-event rules require Complex Event Processing (CEP) support to evaluate complex patterns of events, and chain rules make use of pipe channels, so they must be supported by the TAS.

Group rules are a particular kind of rules that involve several users and are susceptible to collisions with other rules. For instance, Sarah's Smart lighting system is a shared resource managed by a group channel. If Sarah defines a rule to switch off the corridor lights while she is asleep, and her flatmate had a rule that turns them on when the alarm clock rings, both rules collide. It is easy to get to a point where the TAS cannot determine if the lights should be on or off. As a rule of thumb, rules that include group channels are group rules.

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

The rule execution profile defines where the execution of the rule is taking place. Rules may be executed according to different execution profiles to increase performance and enable offline rule execution. In Sarah's scenario, she uses a TAS hosted in the web, but we can imagine other scenarios where a smartphone or a set-top-box performs the automation. Rule execution may be accomplished according to three different execution profiles: entirely on the web, on the mobile client, and mixed execution. A *web driven execution profile* centralises the execution on the server, allowing the existence of lightweight clients at the cost of requiring internet connection. A *mobile driven execution profile* is orchestrated on the client (e.g. smartphone, set-top-box), allowing offline rule execution when only local device channels are involved. A *mixed execution profile* takes the advantages of both profiles. It may shift the execution to the client or to the web, which also reduces communication payload between client and server.

A Reference Task Automation Service Architecture

Once we have described the main components of TASs, we define Task Automation Service as a service that lets users create automation rules that connect channels using a visual editor. A reference TAS architecture must incorporate the features and components previously described (channels, rules, execution profiles) and also provide support for some of the challenges presented in Sarah's scenario. The architecture requirements are:

- To provide a visual rule editor for creating rules;
- to include both web and device channels;
- to feature channel discovery, providing adapters for connected devices so that they are reachable directly by the platform;
- to enable multi-event, multi-action and chain rules;
- to manage group channels and group rules;
- to detect collisions with rules that involve group channels;
- and to support mixed execution profile.

The architecture must provide a visual automation rule editor that users could use to create their automations, but also to activate channels according to the privacy paradigm. To provide access to web and device channels, the architecture must provide the logic needed to connect with Internet services and connected devices. Moreover, the TAS must be notified without delay when an event is generated by the channels, and it must be able to send actions. In addition, it must be able to discover channels according to the discovery paradigm of device channels.

Advanced rule features such as multi-action rules and chained rules do not imply additional requirements, since they can be translated in a set of simpler rules. Multi-event rules involve temporal reasoning of events, and so the TAS requires CEP facilities [6]. Nevertheless, a trade-off among usability for end users and expressivity should be reached for multi-event rules. Group rules require group channels support and rule collision handling. To handle collisions, the architecture must be able to first detect them, and then act when the collision occurs and prevent any unwanted effects.

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

Supporting mixed execution profiles requires some additional logic to coordinate server and device automation rule engines while they orchestrate rule execution, and to guarantee the information about the user and the rules are synchronised on the device and on the server.

Once the requirements to support these novel features are clear, we introduce a reference architecture shown in Fig. 1 which fulfils them.

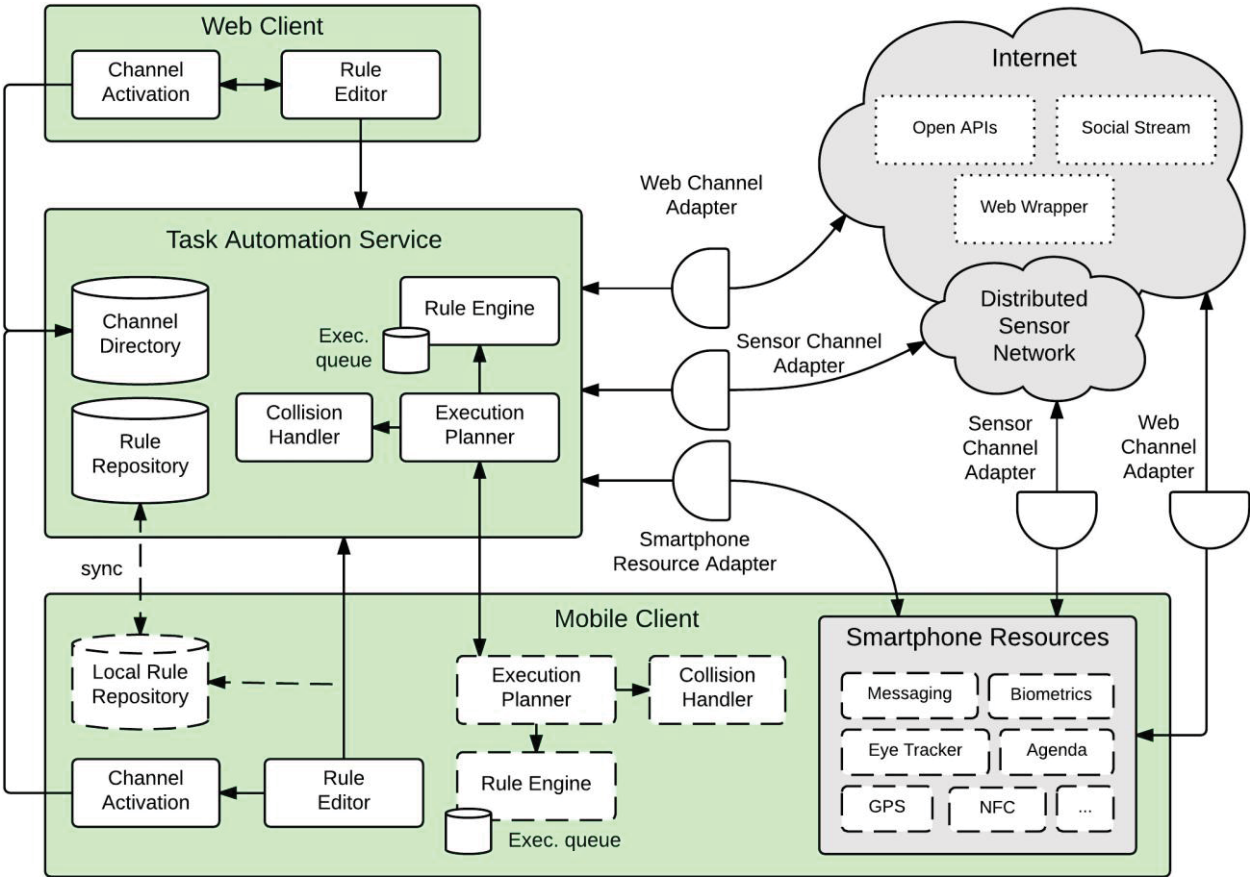

Fig 1. Reference Task Automation Service Architecture general diagram[3].

Rules may be created using an editor on a web client or a mobile client. They are stored in a central rule repository on the TAS server. However, since those rules that can be executed on the client according to the mobile driven execution profile, they are synchronised with a local rule repository for offline access. Mobile and web clients also allow users to activate channels. Once a channel is activated, it is saved in the channel directory together with the authorisation credentials. These credentials are used by the adapters to access the channels.

Adapters provide a uniform access to all kinds of devices. They are responsible for notifying the TAS of incoming events, commanding the execution of actions and taking charge of channel authentication. Adapters can be implemented following a publish-subscribe [7] or polling strategy to get notified of device events depending on their nature. The implementation of adapters can be done by the TAS

[3] Elements in dashed lines are optional in some implementations.

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

administrator or by third parties if the TAS provides an adapter SDK. Adapters to sensor channels provide two different paths: access through a web-protocol, or direct access using access protocols such as ZigBee, Z-wave, Bluetooth or WiFi. They usually expose a set of sensors, i.e. a sensor network, but single device channels are also feasible.

To support a mixed execution profile modules involved in rule execution and channel access must work in coordination. This is the case of the automation rule engine responsible for executing rules and managing rule lifecycles within the execution query. Rule execution consists in fetching the incoming triggering event, extracting the arguments and using those parameters to request the action execution. The Execution Planner guides the orchestration from a higher level according to the active rule execution profile. It manages the state of the channels (within the channel directory), tracking when channels are down and new channels are discovered. For instance, when Sarah arrives at home the Smart TV channel is discovered and added to the channel directory. In turn, when she leaves home, the channel will be down because it is out of range.

Finally, the collision handler analyses the rules in the repository, searching for possible collisions among them. Some patterns of collision are easy to detect e.g. the simplest collision consists of two rules with the same triggering event that try to execute two opposite actions. It takes into consideration which users are currently connected to each channel (as registered in the channel directory), since rules of two users can only collide if they both are connected to the same channel. Recall the example where Sarah wants to have the corridor lights off while she is asleep, and her flatmate set a rule to switch them on when the alarm clock rings. If Sarah's flatmate is not at home, there is no possibility of collision because the channel is not active for Sarah's flatmate. The execution planner consults the collision handler before executing a rule, and in case that rules collides with other rules, its execution is skipped.

Analysis of current TAS platforms

To determine which of the features discussed are supported by the state of the art of task automation, we have analyzed web platforms for general audience (Ifttt), web platforms for business and enterprises (Zapier[4], Cloudwork[5], elastic.io[6], itduzzit[7]), a web platform for cloud storage synchronization (wappwolf[8]), mobile apps (Tasker, Atooma, AutomateIt[9], on{x}[10]), and smart-home systems (wigwag[11], webee[12]). A summary of the results is presented in Fig. 2. and the complete report is available online[13].

[4] https://zapier.com
[5] https://cloudwork.com
[6] http://elastic.io
[7] https://itduzzit.com
[8] http://wappwolf.com
[9] Tasker, Atooma and AutomateIt are available in Google Play
[10] http://onx.ms
[11] http://wigwag.com
[12] http://webeelife.com/
[13] http://bit.ly/TASStudy

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

As expected, web channel support is much larger than device channel support. Although the studied apps manage the resources in the smartphone, only home automation related TASs provide device channels that connect directly to devices. Pipe channels are only fully-supported by elastic.io. It is a powerful concept, and it integrates perfectly in their interface; however, it is barely used. It is worth mentioning that home automation TASs feature channel discovery and group channels. Unfortunately, their support is still very limited.

Regarding different types of rules, few TASs support multi-event rules giving its complexity in comparison to simpler rules. Multi-action rules have wider support, but still many TAS managers do not include them in their platform in order to keep rule editors simple. In the end, a user may achieve the same functionality by implementing a rule for each action in the multi-action rule. Elastic.io, that supports pipe channels, features chain rules, and wigwag and webee, that support group channels, feature group rules –however, none of them handles collisions in group rules. Finally, some of the TASs provide their users with a pack of predefined rules that proved to be useful for previous users, i.e. this is a shortcut for creating those automations.

At the time this study was made, none of these platforms supported a mixed execution profile. Thus, we split them into those with a web-driven execution profile (Ifttt, Zapier, Cloudwork, elastic.io, itduzzit and wappwolf) and those with a device-driven execution profile (Tasker, Atooma, AutomateIt, on{x}, wigwag and webee). By their nature, platforms with a web driven execution profile have more limited access to device channels than those executed on the smartphone. The latter has access to all the smartphone resources. For that reason, Ifttt has already released a smartphone app that grants Ifttt server access to smartphone resources. In turn, Zapier includes Tasker as a channel that effectively grants access to those channels too.

Several TASs offer advanced features for users with programming skills to set up automations using a programming language. This is the case of On{x} (javascript), AutomateIt (bash), elastic.io (javascript), Itduzzit (proprietary), wigwag (Arduino/Raspberry/javascript) and webee (boss). Moreover, Zapier, elastic.io, Tasker and webee provide an API for developing channels (some call them plugins).

		Web						Smartphone				Home	
		Ifttt	**Zapier**	**Cloudwork**	**Elastic.io**	**ItDuzzit**	**Wappwolf**	**On{x}**	**Tasker**	**Atooma**	**AutomateIt**	**WigWag**	**Webee**
Channels	Web Channel support	✓	✓	✓	✓	✓	few	few	✓	✓	✓	✓	✓
	Device Channel support	few	×	×	×	×	×	✓	✓	✓	✓	✓	✓
	Smartphone resources as channels	✓	×	×	×	×	×	✓	✓	✓	✓	×	×
	Public channels support	✓	×	×	✓	✓	×	✓	×	×	×	×	×
	Pipe channel support	×	×	×	✓	×	few	few	×	×	×	×	×

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

	Group channel support	×	×	×	×	×	×	×	×	×	×	few	×
	Device Channel discovery	×	×	×	×	×	×	×	×	×	×	✓	few
Rules	Multi-event rules	×	×	×	×	×	×	✓	✓	×	×	✓	×
	Multi-action rules	×	×	×	✓	×	×	✓	✓	×	✓	✓	✓
	Chain rules	×	×	×	✓	×	few	few	×	×	×	×	×
	Group rules	×	×	×	×	×	×	×	×	×	×	few	×
	Collision handling	×	×	×	×	×	×	×	×	×	×	×	×
	Predefined common rules	×	×	✓	×	✓	✓	×	×	×	✓	✓	✓
	Rule Execution Profile	WD	WD	WD	WD	WD	WD	DD	DD	DD	DD	DD	DD
	Visual rule editor	✓	✓	×	✓	✓	✓	×	✓	✓	✓	✓	✓
TAS	Provides API	×	✓	×	✓	×	×	×	×	×	×	×	×
	Programming language	×	×	×	✓	×	×	✓	few	×	✓	✓	✓

✓= supported; ×= not supported; few= few support
WD=web-driven execution profile; DD=device-driven execution profile; MD=Mixed execution profile

Fig 2. Summary of the comparison of Task Automation Services

Discussion and Outlook

Task Automation Services have gained popularity. While existing TASs increase their number of users, new services appear and compete for them. We identified three main trends in the market. First, web-based TASs are developing smartphone apps in order to integrate smartphone resources as channels of their catalogue. Second, existing Web TASs are starting to offer Rest APIs in order to delegate channel integration to Internet service developers. Thus, Internet service developers interested in integrating their services as channels will have a way to do it themselves. Finally, home automation systems are acquiring this vision, incorporating custom rule automations in their platforms, moving from ‘remote control’ to ‘automate control’.

Nevertheless, this is a novel domain with many challenges to accomplish. First of all, a certain degree of standardization is required. Since each TAS wages war on its own, Internet service developers may find their services available as channels in some TASs, but not in all of them. Moreover, when users define rules within the scope of a TAS, these rules cannot be exported to others, nor shared. Secondly, TASs should evolve into a mixed execution profile that is able to interact with web and device channels. Thirdly, TASs that include device channels to manage shared devices, such as a Smart TV or a smart lighting system should allow group rules and consider the mechanism for detecting and handling collisions. This is really challenging and it will involve usage of complex algorithms. And finally, TASs should include mechanisms for discovering device channels so their configuration is almost transparent to the end user.

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE

Acknowledgements: This work was supported by funds from the Spanish Ministry of Economy and Competitiveness (project CALISTA, TEC2012-32457), and by the Autonomous Region of Madrid through programme MOSI-AGIL-CM (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER).

References

1. K. Parks and D. Watkins, “How to Automate Your Way to Freedom”, 2012.
2. A. R. Meisel, “Optimize, Automate, and Outsource Everything In Your Life: How to Make Email, IFTTT, and Virtual Assistants Your Ultimate Productivity Weapons”, CreateSpace Independent Publishing Platform. 2014.
3. A. Martinez et al., “The Ultimate IFTTT Guide: Use The Web’s Most Powerful Tool Like A Pro”, 2013.
4. H. Labiod et al., “Wi-Fi Bluetooth, Zigbee and WiMax", Springer Science and Business Media, 2007.
5. W. Beer et al., “Modeling Context-Aware Behavior by Interpreted ECA Rules”, Euro-Par 2003 Parallel Processing, Springer Berlin Heidelberg, 2003, pp 1064-1073.
6. M. Eckert et al., “A CEP Babelfish: Languages for Complex Event Processing and Querying Surveyed”, Reasoning in Event-Based Distributed Systems, Springer Berlin Heidelberg, 2011, pp 47-70.
7. P. Th. Eugster et al., “The many faces of publish/subscribe”, ACM Comput. Surv. 35, 2003, pp 114-131.

Miguel Coronado is a PhD student in the School of Telecommunications Engineering at the Technical University of Madrid. His research interests include semantic web, linked data, personal assistants, and Internet of Things. Miguel has an MSc in telecommunications engineering from the Technical University of Madrid.

email: miguelcb@dit.upm.es

Carlos A. Iglesias is an associate professor and head of the Intelligent Systems Group in the Telecommunications Engineering School at the Universidad Politécnica de Madrid, Spain. His research interests include web technologies, agent systems, social intelligent systems, IoT and Big Linked Data analytics. Carlos A. has a PhD in telematics from the Technical University of Madrid.

email: cif@dit.upm.es

Digital Object Indentifier 10.1109/MIC.2015.73 1089-7801/$26.00 2015 IEEE