Building Smart Spaces on the Home Manager platform

Roberta Calegari and Enrico Denti

Dipartimento di Informatica, Scienza e Ingegneria (DISI)
Alma Mater Studiorum–Università di Bologna
viale Risorgimento 2, 40136, Bologna, Italy
{roberta.calegari, enrico.denti}@unibo.it

Abstract. Smart Spaces refer to application scenarios where people are immersed in time and space in an augmented environment, exploiting ubiquitous computing technologies, space and time awareness, and pervasive intelligence.
This paper focuses on Home Manager, an agent-based platform for the implementation of Smart Spaces in Smart Home contexts, rooted in the *Butlers for Smart Spaces* framework – the specialisation to Smart Spaces of the *Butlers* vision, aimed at supporting smart services to users immersed and interacting with their surrounding environment. The goal is to anticipate the users' needs whenever possible, reasoning on potentially any kind of relevant data grabbed from a plurality of sources, as well as from the users's preferences.
Taking the case of a Smart Kitchen as our running example, we first discuss how the *Butlers for Smart Spaces* framework can be specialised to the Home Manager case and exploited to define smart appliances, then present the Home Manager technology in full, and show by a case study how a smart space can be deployed on its top.
Home Manager's support to intelligence is provided both by the TuCSoN coordination infrastructure, which enables intelligence to be spread on agents and on coordination artifacts, and by the tuProlog technology, which not only powers TuCSoN artifacts but also – being light-weight and Java-based – effectively supports the development of declarative (Prolog), imperative (Java,C#), and hybrid agents.

1 Introduction

Smart spaces [1, 2] refer to a wide variety of possible environments (apartments, offices, museums, hospitals, schools, malls, university campuses, outdoor areas) enabled both for the cooperation of smart objects and systems, and for the ubiquitous interaction with visitors. In this pervasive vision, computer systems seamlessly integrate into people's everyday lives, providing services and information "anywhere, anytime" [3, 4]. As such, they combine dimensions and behaviour from *pervasive*, *distributed*, *situated* and *intelligent* computing—altogether [5].

The dramatically growing development of the Internet of Things [5–8] is further pushing scenarios where appliances and devices of virtually any sort are

networked together, possibly embedding some form of (usually limited) intelligence. Not surprisingly, the major players are promoting their architectures and technologies – from Google's 'Works With Nest' [9] to Samsung SmartThings [10], Apple's Home Kit [11], Windows 10 IoT Core [8] – up to Amazon's Smart Home shop [12]; moreover, personal assistants with natural language capabilities, like Google Now [13], Siri [14], Cortana [15]), are developing further capability of giving suggestions based on the user's current context and habits.

Of course, several challenges need to be addressed [16–21]: still, some common aspects and needs are starting to emerge.

In order to effectively design and build smart spaces that go beyond the mere remote-control of appliances and more generally the "gadget"-like approach of some prototypes presented also by the major vendors, two key preconditions seem to be *(i)* the availability of an effective interaction and coordination middleware, going beyond the basic support to interoperability, and *(ii)* an effective support to *distributed intelligence* [16] and *situatedness*, intended as awareness of the environment and possibly reactivity to changes. The goal is, on the one side, to effectively "put together" the distributed intelligence but also, on the other, to support its spreading "when and where needed", including the middleware—which, in turn, calls for an "intelligence-ready" coordination layer. The result is a special kind of *Socio-Technical System* (STS)—the kind of systems that arise "when cognitive and social interaction is mediated by information technology, rather than by the natural world (alone)" [22]: they are by nature heterogeneous, distributed, made both of software agents, (sensors, actuators) and humans with their capabilities and social organisations [23, 24]. Because of their characteristics, Multi-Agent Systems (MAS) [25] are a natural reference in this scenario, since they provide models, infrastructures and methodologies that fit well the modelling needs of Smart Spaces – in particular, as concerns the support of the so-called *infrastructure intelligence*.

In this paper, we focus on *Home Manager* [26, 27], an agent-based platform for the implementation of Smart Spaces in Smart Home contexts, and specifically in IoT-aware environments, rooted in the *Butlers for Smart Spaces* framework [28]. This framework is conceived according to the *Butlers* vision [29], and aims to support advanced services to users *immersed* and *situated* in time and space and interacting with their surrounding environment—in particular, trying to anticipate the users' needs whenever possible, reasoning on potentially any kind of relevant data grabbed both from users's preferences and from other sources. The platform is built on top of TuCSoN [30, 31], a multi-agent coordination infrastructure enabling intelligence to be spread not only on agents, but also on coordination artifacts, thanks to the underlying tuProlog [32, 33] technology.

Being light-weight and Java-based, tuProlog effectively supports the development of both declarative (Prolog) and imperative (Java,C#) agents, as well as hybrid Java+Prolog agents [34]—in particular, making it easy to embed intelligence in Java agents. At the same time, it constitutes the powering technology of TuCSoN artifacts (Subsection 2.1), both as concerns the social intelligence – ReSpecT *tuple centres* are declaratively-programmable coordination media – and

the boundary artifacts – *ACC*s and *transducers*. As a result, the coordination laws can be expressed in a declarative way, paving the way to bring reasoning also at the infrastructure level.

In the following we discuss why Home Manager can be well suited for the design and development of Smart STS, with a focus on the reasoning aspects in the *Butlers for Smart Spaces* framework and, more generally, on the clear separation among agent intelligence, infrastructure intelligence and system policies promoted by the underlying TuCSoN model. Our goal is to show that this approach not only makes it natural to spread and coordinate distributed, situated intelligence in smart spaces where and when necessary, but also supports the integration of heterogeneous (possibly legacy) components in spite of the differences in design paradigm, implementation language and platform, data representation, communication and coordination language, thus helping to bridge the gap between the infrastructure level and the situated interaction patterns that are often found in Smart Spaces contexts.

2 The Home Manager platform

Home Manager [26, 27] is an open source platform [35] for Smart Spaces, inspired to the above architecture and explicitly conceived to be open, deployable on a wide variety of devices (PCs, smartphones, tablets, up to Raspberry PI 2), and – thanks to the underlying TuCSoN [31] infrastructure and the suitable integration with tuProlog – suitable to accommodate "as much intelligence as the system needs, where the system needs".

Its purpose is to provide advanced services to users immersed in / interacting with the surrounding environment—in particular, the ability to reason on potentially any kind of relevant data, both extracted from the users's preferences and grabbed from other sources, so as to anticipate the users' needs.

Before discussing the platform, we shortly summarise below the main features of TuCSoN infrastructure and the *Butlers for Smart Spaces* approach.

2.1 The TuCSoN infrastructure in a nutshell

TuCSoN [31, 36] is a tuple-based agent coordination infrastructure for open distributed MAS, rooted on ReSpecT *tuple centres* [37]. Moreover, two middleware abstractions play the role of boundary artefacts—namely, *Agent Coordination Contexts* (ACC) [38] for agents, and *transducers* for resources.

Tuple centres are enhanced logic tuple spaces, distributed over a network of TuCSoN nodes, and programmable via the (Turing-equivalent) ReSpecT [39] logic language: in fact, the ReSpecT virtual machine is itself built on top of tuProlog. In the TuCSoN vision, tuple centres embed the coordination laws, enabling MAS designers to govern the interaction space and reifying the "social intelligence". Moreover, ReSpecT support to *situatedness* [40, 41] makes it possible to associate events in the interaction space – occurring as a consequence of agents activities or environment changes – to appropriate handlers (computations).

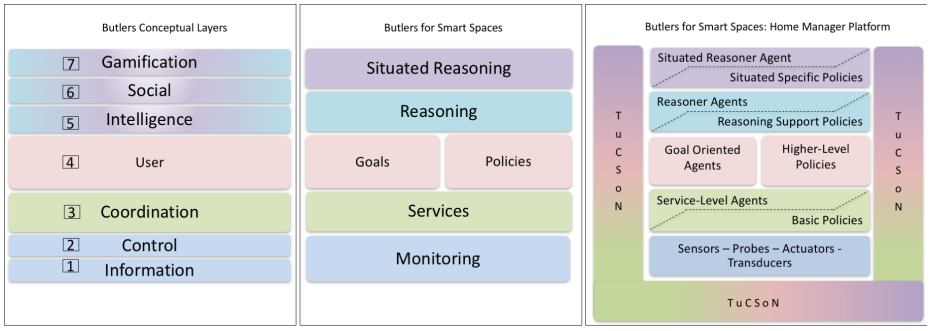


Fig. 1. Butlers, Butlers for Smart Spaces, and the Home Manager specialisation.

Each agent is also associated to an ACC, the security and organisation abstraction in charge of mediating the agent interactions with the MAS—in particular, providing the agent with available operations, based on its role and task. Transducers [42] represent individual resources, with their own peculiar ways of interacting: each transducer is capable of two-way interaction, to map meaningful resource events upon admissible MAS events.

The TuCSoN technology is light-weight, open source [30], and Java-based—although some (limited) interaction with Windows 10-IoT is also possible [43].

2.2 Butlers for Smart Spaces

The Butlers architecture [29] defines a technology-neutral framework which identifies seven conceptual layers and relates technologies with the corresponding features and value-added for users. *Butlers for Smart Spaces* [28] is the specialisation of that framework to the Smart Spaces context: Fig. 1 shows Butlers, Butlers for Smart Spaces as well as its re-shaping to the Home Manager case.

Leaving the full discussion of the Butlers layers to [29], the bottom layers concern the enabling technologies – communication-enabled sensors, meters, actuators, etc. – while the infrastructural / middleware layers provide coordination and geographical information services, and the top layers focus on intelligence, sociality, gamification aspects (not necessarily to be taken in the sequence).

In Smart Spaces, lower-level functionalities are typically provided by the underlying infrastructure, while some envisioned upper functionalities are either too far from the foreseeable future or from the current state of the art: so, their layers can be conveniently collapsed. Accordingly, in *Butlers for Smart Spaces*:

- the information (1) and control (2) Butlers layers are grouped together in a single *Monitoring* layer;
- the new *Services* layer, in-between information (Butlers layer 1) and coordination (layer 3), pre-processes raw information into exploitable knowledge;
- coordination (3) and user-aware (4) Butlers layers are on the one hand grouped into a single layer, because coordination in a Smart Space must

necessarily take users – the main actors – into account; on the other, the foreseeable complexity of such coordination leads to split such a layer into *Goals* and *Policies* side-by-side, for both practical and conceptual reasons;
- the intelligence (5) Butlers layer is also split into two *Reasoning* and *Situated reasoning* on top of each other, to separate the reasoning which exploit only the local/user knowledge from the ones which exploit also the surrounding environment—which features the very nature of a Smart Space.

2.3 Specialisation to the Home Manager case

In Home Manager, *Butlers for Smart Spaces* layers are concretised into a TuCSoN-based MAS. This is why the TuCSoN infrastructure surrounds all layers, enabling the seamless integration of heterogeneous entities, bridging among technologies and agents' perceptions, and supporting situated intelligence.

From the architectural viewpoint, each device is supposed to be equipped with an agent, acting as a sort of "proxy" to interface the physical device to the agent society that powers the Smart Space. Then:

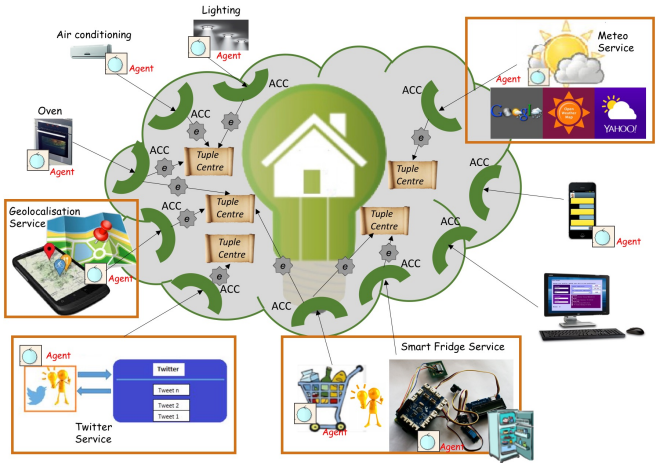
- in the *Monitoring* layer, proxy agents provide for the device monitoring and (possibly) remote control, interacting via TuCSoN to grab the required information and operate on the environment;
- the *Services* layer is concretely split into *Service-Level Agents* and *Basic Policies*, following the idea that information is elaborated in this layer via mechanisms that do not require sophisticated reasonings (e.g. grabbing weather info based on the simple user policies, like the preferred weather sites);
- analogously, *Goals* and *Policies* take the concrete form of *Goal-Oriented Agents* and *Higher-Level policies*: at this stage, policies concern everyday life habits and aspects, and are generally rather stable; the corresponding agents handle the autonomous decisions which refer to such policies;
- the *Reasoning* layer also splits into *Reasoner Agents* and related *Reasoning Support Policies*, reasoning on user-related knowledge (profile, habits, preferences) and corresponding rules;
- similarly, the *Situated Reasoning* layer concretises into *Situated Reasoner Agents* (which take into account the user location, movement, etc. to provide real-time suggestions and pro-active actions) and related policies.

2.4 The technology

The Home Manager platform is built on top of three main technologies: TuCSoN and its artifacts – tuple centres and ACCs – for MAS coordination, the ReSpecT logic language to bring situated intelligence to TuCSoN nodes, and tuProlog to build light-weight intelligent agents.

The logical architecture is shown in Fig. 2: each device is assumed to be equipped with an agent, which is connected to a TuCSoN ACC defining its admissible operations and roles in the agent society; as such, they embed the individual intelligence. Tuple centres, on the other hand, embed the "social intelligence",

Fig. 2. The Home Manager logical architecture



further supported by tuple centres' *linkability*—the ability to trigger reactions in other tuple centres as a consequence of a local event [44].

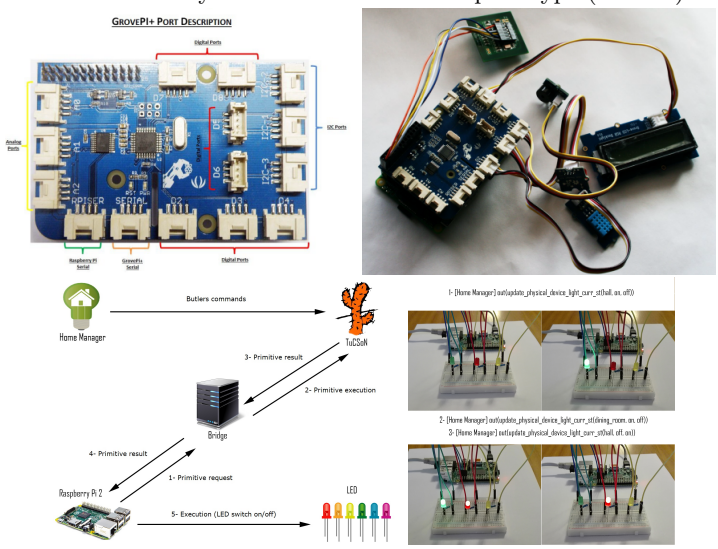
The infrastructure embeds and enforces the coordination laws to mediate among agents, governing the agent-agent and the agent-environment interaction. Heterogeneous entities, such as legacy agents, can be integrated by charging the infrastructure of bridging the gap between the common ontology and the specific agents' representations and ontologies [45]: agents can thus be heterogeneous in nature, implementation language, etc.—the only requirement being that they coordinate via the TuCSoN APIs, adhering to their intended semantics.

The prototype enables users to control the system configuration and interaction, yet with no need to know or operate directly on the underlying machinery—i.e., the inner tuple-based representation of data and policies.

It is worth highlighting that the declarative, tuple-based approach is what makes it easy to evolve the system incrementally from a purely-simulated environment hosted on a personal computer (with simulated house, inhabitants, and sensors) to an "increasingly-real" system, interfaced – for instance – to an Android smartphone as shown in the following sections, so that users can interact with the system in mobility via a suitable app—which, by the way, takes care of geo-localising the user and to support advanced services based on the user's situatedness in space and time (Subsection 3.2).

Going farther, the system can also be interfaced to handling actual hardware devices, up to possibly run "out of the box" on low-cost stand-alone platforms like a Raspberry PI 2 Model B. The latter choice provides *i)* an independent installation on a dedicated software+hardware platform, dropping the requirement of a personal computer for the hosting environment, and *ii)*, perhaps more relevantly, the chance to exploit the many Raspberry sensors and devices.

Fig. 3. Home Manager out-of-the-box on a Raspberry Pi 2 + GrovePi kit (top); interaction with a Windows-10 system – architecture and prototype (bottom).



2.5 The Raspberry porting

For our intended application context (Subsection 3.1), the support of displays and RFID readers is particularly relevant, to simulate the presence and movement of items, users, etc.; but several other (low cost) sensors and actuators are necessary for a reasonable simulation. This is why we developed our prototype on top of the GrovePi [46] board, which comes with many nice sensors and actuators (displays, switches, temperature sensors, etc.) in an all-in-one pack with customized Raspbian version: the resulting platform [47] is shown in Fig. 3 (top).

Even more interestingly, and orthogonally, the Raspberry can be exploited as an implementation platform for smart devices – Smart Fridge, Smart Oven, etc. Although Java and a Raspbian-based Raspberry is the most obvious choice, a multi-platform, multi-language environment could be another, challenging, perspective. To this end, we explored the Microsoft Windows 10-IoT Core [8, 48] platform, which enables UWP (Universal Windows Platform) applications to be designed in Visual Studio and then deployed to the Raspberry PI, supporting remote executing and debugging.

In order to integrate it into Home Manager, an ad-hoc bridge has to be set up to interface the (Java-based) TuCSoN primitives, used by Home Manager, with the Windows 10 platform (Fig. 3, bottom left): to ensure that the communication with TuCSoN is seamless, the bridge itself is written in Java. The (C#-coded) client agent and the bridge communicate via UTF-8 strings: a suitable XML configuration file specifies the data required by the client agent (bridge IP and port numbers, target tuple centre name, etc.) and maps the client coordination

language primitives onto the TuCSoN one. Fig. 3 (bottom right) shows a trivial demo application, where some LED are controlled via such a bridge.

3 Building Smart Spaces on Home Manager

According to Fig. 1, and following the logical architecture in Fig. 2, building a smart space on top of Home Manager means *i)* to identify the device and service categories that are relevant for that specific space; *ii)* to define a suitable tuple-based representation of the relevant knowledge, and *iii)* the desired agent interaction protocols – which, by the way, need not match the knowledge representation 1-1, since tuple centres can be suitably programmed to bridge the gap; *iv)* to develop an agent for each device category and for each external service to interact with – possibly testing and debugging them separately from the rest of the system, thanks to the data-driven approach. The whole design process is aimed at keeping the social/individual intelligence, on the one hand, and mechanisms/policies, on the other, clearly separate.

3.1 The reference scenario

Home Manager refers to a smart house immersed in a smart living context, with devices (air conditioners, lights, etc.) and users of different categories and (RBAC-based) roles [27].

At the basic operation level, the goal is to satisfy the users' desires (e.g. room light, temperature) while respecting some global constraints (e.g. energy saving, temperature range, etc.): as an example, Table 1 shows the ReSpecT reactions that control the temperature of a room, averaging the preferred temperatures of users in case two or more people are present.

At a higher level, however, the goal is more ambitious—to anticipate the user's needs by reasoning on the user's habits and on any user-related information, including the environment where he lives, travels, purchases goods, etc. The idea is to go beyond the mere monitoring and remote control of house appliances via app, as it is often found today, towards:

- exploiting the user's location, tracked by the smartphone GPS, to enable an intelligent reasoner agent to take autonomous "situated" decisions;
- explore the environment around the user's location, extracting information about shops, services, etc, to be taken as a further reasoning knowledge base;
- getting information about the surrounding environment (e.g. weather) so as to tailor decisions to the user's habits and needs;
- interacting with selected social networks (e.g. Twitter) to grab information that could later be exploited fur further reasonings;
- tracking the human presence for intrusion detection or elderly applications (e.g. to detect falls, stand or walk status, etc.);
- overall, providing novel, integrated services by coupling smart appliances (smart fridge, smart ove, etc.) with environment and user information.

```
% 2+ Users
reaction(
    in(new_temperature(L,Users,T)),
    (request),
    ( Users > 1,
        out(avg_temp(L,L,Users,0))
    )
).
% Calculate the temperature..
reaction(
    out(avg_temp(L,[user_pref(X,WarmTemp,W,Z)|OtherPrefs],Users,Sum)),
    (internal),
    ( in(avg_temp(L,[user_pref(X,WarmTemp,W,Z)|OtherPrefs],Users,Sum)),
        rd(temp_mode(heat)),
        NewSum is Sum+WarmTemp,
        out(avg_temp(L,OtherPrefs,Users,NewSum))
    )
).
reaction(
    out(avg_temp(L,[user_pref(X,Y,CoolTemp,Z)|OtherPrefs],Users,Sum)),
    (internal),
    ( in(avg_temp(L,[user_pref(X,Y,CoolTemp,Z)|OtherPrefs],Users,Sum)),
        no(temp_mode(heat)),
        NewSum is Sum+CoolTemp,
        out(avg_temp(L,OtherPrefs,Users,NewSum))
    )
).
reaction(
    out(avg_temp(L,[],Users,Sum)),
    (internal),
    ( in(avg_temp(L,[],Users,Sum)),
        Average is Sum/Users,
        out(new_temperature(L,Users,Average))
    )
).
```

Table 1. Excerpt of ReSpecT code for temperature management.
ReSpecT reactions take the form `reaction(`*`operation, guard, body`*`)`, meaning that the reaction is triggered by the occurrence of *`operation`*, causing the execution of *`body`* provided that *`guard`* is true; if either of these fails, the reaction as a whole also fails, yielding no effect at all. The *`body`* can perform other tuple centre operations, thus triggering other reactions in a chain.
Here, the top reaction is triggered when an agent asks for the new temperature `T` given the list of preferences `L` of a given numbers of `Users`. The guard checks that the operation is in its `request` phase, then the body checks that there are at least two users (otherwise a different reaction set, not shown, will be executed) and emits the service tuple `avg_temp(L,L,Users,0)` to trigger the actual average computation in chain. In its turn, this latter `out` operation triggers the three next reactions—whose guard, `internal`, guarantees that they are only executed as a consequence of another, previous reaction. These reactions are, by design, mutually exclusive: in particular, the first two consider either the heat season – signalled by the tuple `temp_mode(heat)`, in which case the preferred "warm" temperature is taken – or the winter season (in which case the preferred "cool" temperature in taken instead), respectively, while the latter handles the termination case. Basically, the reaction chain sums the preferred user temperatures, then the closing reaction divides by `Users` and emits the desired `new_temperature/3` tuple, awaking the agent suspended on the original `in` operation.

3.2 Space situatedness via geo-localisation

As a first step in supporting the user's situatedness in the environment in time and space, [26] presented a simple scenario exploiting the geo-localisation facility embedded in any modern smartphone to:

- extend the system intelligence, recognising places and services based on their position, via the Google Places API;
- provide user-location-related information e.g. about surrounding services.

There, the user location is monitored and conceptually used to control a Smart Oven, switching it on automatically if the user buys a take-away pizza in her way back home: Fig. 4 shows the exploration of the surrounding services in the Android app at different user's positions [49]. As a further consequence, the smart house switches on the oven, so that the user find it hot enough for warming the pizza as she comes back. The system decision is notified to the user (Fig. 4, right), so that she always remains in control and has the last word.

A possible evolution, envisioned in [29], could be to exploit such info not to overcome the electrical power threshold, e.g. postponing the washing machine to switch on the oven without causing a blackout—which would be particularly dangerous after dusk, in presence of elderly people or children, etc.

3.3 Space and time situatedness based on weather

Weather info is by nature situated in time and space, and – even more relevant – suitable to inherently and continuously condition everyone's life: many "micro-decisions" we take every day depends on weather—doing laundry, to (not) go out for shopping, switching on heating, closing windows, etc.

This is why a Home Manager agent has been developed for that purpose (Fig. 5) [50]: once retrieved, weather info can be later exploited in several ways—from intelligent appliances scheduling (e.g., avoid scheduling the washing machine in a raining day, so as to avoid the dryer), to the automatic control of rolling shutters based on sunrise and sunset times, to just taking into account the user's mood, etc. For instance, intelligent shutter control could be

Fig. 4. Exploration of surrounding services, and notification for autonomous actions

obtained by embedding a small piece of intelligence – a ReSpecT reaction – in the infrastructure, so as to intercept the addition of new relevant weather data (independently of the specific tuple format) and generate the shutter actions.

Weather sources can be Google's [51], Yahoo's [52] or OpenWeatherMaps [53]'s web services—but they all require the house location. This datum can either be statically stored in some tuple centre or, better, be dynamically extracted from the house's IP number, thanks to services like `IP-API.com`, and then turned onto a physical city name, via Flicker or similar services. The retrieved info is reified in form of suitable tuples (Fig. 5, bottom right) in the `weather-tc` tuple centre: the history is maintained, so as to enable any potential reasoning.

Fig. 5. Weather agent and the respective tuples

3.4 Let's be social: Twitter integration

As recalled in Fig. 1, the social aspects have been included in the Butlers framework from its very origin—mainly as a source of further knowledge about the user (habits, interests, traffic, special offers, government warnings, etc.), but also, more in the long-term perspective, of butlers networks [29].

The first choice has been to develop a Twitter agent (Fig. 6) [54] – both because Twitter's text-based nature makes it easier to parse and extract data, and because messages are typically more informative than, say, Facebook posts, which are more often oriented to closed groups of friends commenting on their own lives.

To keep things simple, this early prototype considers a single Twitter user, representing the home butler, with read-only capabilities: the goal is to monitor selected "interesting" users, and – like the weather info above – store the related tweets in a suitable tuple centre, for further reasoning (Fig. 6, bottom right). Technically, Twitter is accessed via REST API, with OAUth [55] to handle the user authentication and Twitter4J [56] for Java inter-operability. Due to its nature of early implementation, severe limitations apply—in particular, Twitter credentials are currently stored in a tuple centre; security issues are to be engineered in a future release.

In the longer-term perspective, however, the butler account could also post messages—e.g. to share result of its reasonings with its own followers.

3.5 The Smart Kitchen scenario

As a more comprehensive example we take the case of the smart kitchen, made of a *Smart Oven*, a *Smart Fridge*, a *Smart Pantry*, a *Smart Mixer*, integrated with a *Smart Shopper* butler aimed at managing the food supply.

The Smart Oven is aimed at supporting the user's food cooking—in principle, exploiting any available technology to identify and cook the food; the user profile is supposed to include information about his/her dietary requirements. The Smart Fridge, on its side, is capable of monitoring the availability and quantity of food in the fridge, while the Smart Pantry is supposed to to the same for the whole kitchen. The Smart Mixer manages the recipe instructions, interacting with both the Smart Fridge – to check that the ingredients for the selected recipe are actually available – and with the Smart Oven – to check its ability to cook that food and potentially synthesise the proper control instructions. To this end, the fridge content is checked and, based on user-defined policies, the shopping list is produced; this is then handled by the Smart Shopper butler, in charge of opportunistic behaviour—e.g. sending the order to the "appropriate" supermarket for home delivery (Fig. 7, bottom).

This scenario clearly calls for "proper chunks" of intelligence spread where needed—in particular, to reason on the available information based on specific policies, which can be as complex as desired. For instance, policies (currently under development) could aim at guaranteeing e.g. that *i)* at least 2 bottles of milk and 3 cans of beer are always present, *ii)* the total of the shopping list reaches a minimum threshold to exploit free home delivery, *iii)* the list is compared against multiple markets to find the most convenient, taking into account fidelity cards and special offers; and so on. Opportunistic behaviour could also exploit the user's location to alert her of a nearby market, so as to avoid another subsequent drive and minimise the fuel consumption and cost, or suggest an alternative market to avoid traffic jams; and so on.

In order to contextualise the Butlers for Smart Spaces framework, as specialised for Home Manager, to this application case, the methodological approach discussed at the beginning of this Section should be applied: the complete discussion of this aspect is outside the scope of this paper, but the interested reader can find the Smart Oven case discussed in [28]. The resulting architecture has to be

Fig. 6. The Twitter agent

Fig. 7. Smart Fridge hardware

implemented as a TuCSoN MAS. Accordingly, TuCSoN agents take care of the Basic Fridge/Pantry/Oven Services, while ReSpecT reactions implement both the basic mechanisms (services layer) and the social intelligence—for instance, whenever some milk is taken, to check the remaining quantity and possibly generate the "buy milk" order tuple; and so on.

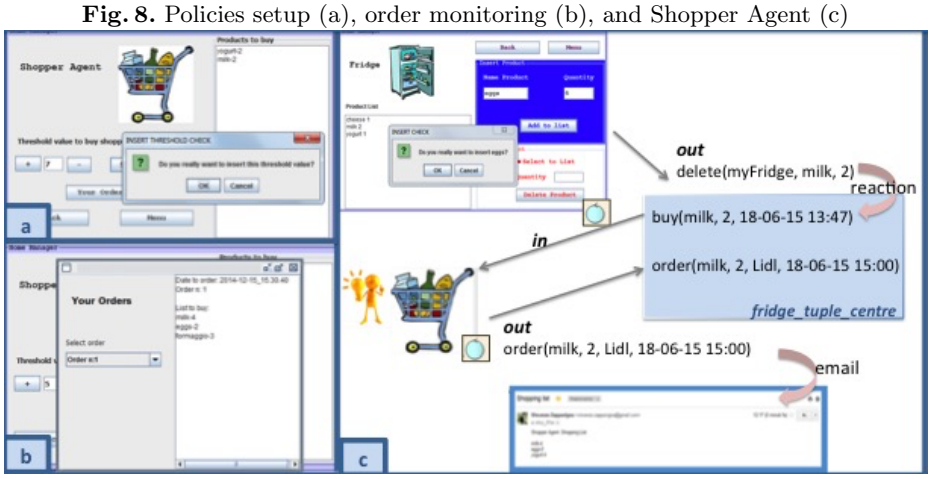
In the current (highly experimental) prototype, the Smart Oven and the Smart Pantry are simulated in software (the GUI supports recipe insertion/update/removal), while the Smart Fridge [57] integrates software-only and software+Raspberry PI hardware. The Raspberry is coupled with the GrovePI display, sensors and LEDs, and with an RFID tag reader to track the fridge content (Fig. 7). Every time that an RFID tagged product is moved through the fridge, a read event is created: an acoustic sound is produced by the fridge, a led is lit and information about the product appear on the LCD display.

The information about the fridge content is reified in a tuple centre so that other agents can exploit it. In particular, the Smart Fridge monitors each product availability, reified as `fridge_content(`*`fridgename,product,quantity`*`)` tuples: whenever a product is low, it is added to the shopping list. The scarcity check is based on user-defined, per-product thresholds, reified as `user_pref(`*`userID,product,threshold`*`)` tuples: accordingly, whenever a product is taken from the fridge, a suitable ReSpecT reaction checks its level, generating a `scarcity(`*`userID,product,actual_quantity,threshold,timestamp`*`)` tuple if this is the case; of course, more complex policies could be defined.

This information is then exploited – based on other suitable policies – to generate the buy orders (Fig. 8, b): currently, the trivial policy is just to generate an order when "enough" `scarcity` tuples are present—the threshold value being configured by the user (Fig. 8, a) and reified as a `threshold(`*`userID, value`*`)`. The result is a suitable `buy(`*`product,quantity,timestamp`*`)` tuple for that product (Fig. 8, c): of course, it is up to the policy to define how many items to purchase.

In its turn, the Smart Shopper – based on its own shopping policies – consumes the `buy` information, compiles the shopping list and contacts the "proper" (currently: the pre-defined) vendor by any means (currently: by email), with the list of products and quantities to be purchased (Fig. 8, c).

Fig. 8. Policies setup (a), order monitoring (b), and Shopper Agent (c)



4 Conclusions

Smart spaces systems are one of today's most promising, and challenging, application areas: increasing user expectations call for suitable integration of skills, components, services from several fields—mobile and ubiquitous systems, intelligent systems, pervasive systems, software infrastructures, multi-agent systems, information security, and inter-operability issues, to name just a few.

At the same time, such a wideness come with drawbacks and weaknesses—e.g., interoperability among different hardware and software components from different manufacturers and among different home automation technologies, the complexity of configuration, the lack of universal service consideration, security issues [58]. As shown above, the declarative approach plays a key role in this context, both to provide a *"lingua franca"* – to bridge among the different forms of heterogeneity – and to support agent uncoupling, the separation between policies and mechanisms, and the base for further reasoning. In our case, suitably-programmed tuple centres bridge among different interaction protocols and knowledge representations, effectively supporting interoperability; moreover, any information retrieved by any agent is represented declaratively in form of tuples, creating a knowledge base for subsequent elaboration.

Other issues in the literature concern the complexity and cost of the architectures, the lack of unifying frameworks and affordable infrastructures, the intrusiveness of the system installations, sometimes poor user interfaces, the complexity of configuration (or inadequate configurability), the insufficient or inconsistent approaches towards security and safety, etc. This is why stronger emphasis is being devoted to the development of standard frameworks, APIs, infrastructures, and interoperability protocols, as shown by the recent proposals by the major stakeholders.

Our approach explores a different starting point: instead of moving from a specific need or application area, the Butlers for Smart Spaces framework tries to define a technology-neutral reference for Smart Spaces in pervasive IoT contexts, which can work both as design guidelines and possibly as a suggester for new application scenarios/niches in a whole stack of smarter possibilities.

By suitably integrating three technologies – TuCSoN for MAS coordination, ReSpecT for situated intelligence, tuProlog for light-weight intelligent agents –, the Home Manager platform concretises the above approach while safeguarding designers' freedom, by delegating the underlying infrastructure to bridge among the diverse agents' ontologies, APIs, knowledge representations, etc.

The result shows the essential role of a declarative approach in supporting *i)* the coordination of heterogeneous agents, *ii)* the injection of intelligence in the infrastructure level (with high-level symbolic capabilities), and more generally *iii)* in building and natively integrating intelligent agents in the system nodes. More in the perspective, since TuCSoN policies are encoded in the form of (ReSpecT) tuples, they can be modified as easily and uniformly as any other data, potentially opening the way to meta-reasoning agents, capable of changing/tuning the policies and coordination laws according to some meta-level rules.

References

1. Wang, X., Dong, J.S., Chin, C., Hettiarachchi, S., Zhang, D.: Semantic space: An infrastructure for smart spaces. IEEE Pervasive Computing **3**(3) (2004) 32–39
2. Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., Chakraborty, D.: Intelligent agents meet the semantic web in smart spaces. Internet Computing, IEEE **8**(6) (Nov 2004) 69–79
3. Satyanarayanan, M.: Pervasive computing: vision and challenges. Personal Communications, IEEE **8**(4) (Aug 2001) 10–17
4. Saha, D., Mukherjee, A.: Pervasive computing: a paradigm for the 21st century. Computer **36**(3) (Mar 2003) 25–31
5. Ricci, A., Piunti, M., Tummolini, L., Castelfranchi, C.: The mirror world: Preparing for mixed-reality living. Pervasive Computing, IEEE **14**(2) (Apr 2015) 60–63
6. Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., Oliveira, A.: The future internet. Springer-Verlag, Berlin, Heidelberg (2011) 431–446
7. Rothensee, M.: User acceptance of the intelligent fridge: Empirical results from a simulation. In Floerkemeier, C., Langheinrich, M., Fleisch, E., Mattern, F., Sarma, S., eds.: The Internet of Things. Volume 4952 of LNCS. Springer (2008) 123–139
8. Microsoft: The internet of your things. https://dev.windows.com/en-us/iot/ (2015)
9. Google: Works with Nest. http://techcrunch.com/2014/06/23/google-makes-its-nest-at-the-center-of-the-smart-home/ (2014)
10. Samsung Smart Things. https://www.smartthings.com (2015)
11. Apple Home Kit. https://developer.apple.com/homekit/ (2014)
12. Amazon Smart Home. https://www.amazon.com/smarthome-home-automation/b?ie=UTF8&node=6563140011 (2016)
13. Google Now. https://www.google.com/landing/now/ (2015)
14. Apple: Siri home page. http://www.apple.com/ios/siri/ (2015)
15. Microsoft Cortana. https://microsoft.com/en-us/mobile/experiences/cortana/ (2016)

16. Parker, L.: Distributed intelligence: Overview of the field and its application in multi-robot systems. Journal of Physical Agents **2**(1) (2008)
17. Baresi, L., Shahzada, A.: An architecture-centric approach for dynamic smart spaces. In Weyns, D., Mirandola, R., Crnkovic, I., eds.: Software Architecture. Volume 9278 of LNCS. Springer Int Pub (2015) 277–284
18. Jeng, T.: Designing a ubiquitous smart space of the future: The principle of mapping. Springer Netherlands (2004) 579–592
19. Jeng, T.: Toward a ubiquitous smart space design framework* (2009)
20. Ricci, A., Omicini, A., Denti, E.: Engineering agent societies: A case study in smart environments. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3. AAMAS '02, New York, NY, USA, ACM (2002) 1064–1065
21. Zambonelli, F., Omicini, A., Anzengruber, B., Castelli, G., Angelis, F.L.D., Serugendo, G.D.M., Dobson, S., Fernandez-Marquez, J.L., Ferscha, A., Mamei, M., Mariani, S., Molesini, A., Montagna, S., Nieminen, J., Pianini, D., Risoldi, M., Rosi, A., Stevenson, G., Viroli, M., Ye, J.: Developing pervasive multi-agent systems with nature-inspired coordination. Pervasive and Mobile Computing **17, Part B** (2015) 236 – 252
22. Whitworth, B.: Socio-technical systems. In: Encyclopedia of Human Computer Interaction. Hershey: Idea Group (2006) 533–541
23. Zambonelli, F.: Toward sociotechnical urban superorganisms. Computer **45**(8) (2012) 76–78
24. Jennings, N.R., Moreau, L., Nicholson, D., Ramchurn, S., Roberts, S., Rodden, T., Rogers, A.: Human-agent collectives. Com. ACM **57**(12) (November 2014) 80–88
25. Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems **1**(1) (January 1998) 7–38
26. Denti, E., Calegari, R.: Butler-ising HomeManager: A pervasive multi-agent system for home intelligence. In Loiseau, S., Filipe, J., Duval, B., Van Den Herik, J., eds.: 7th Int. Conf. on Agents and Artificial Intelligence (ICAART 2015), Lisbon, Portugal, SCITEPRESS (10–12 January 2015) 249–256
27. Denti, E., Calegari, R., Prandini, M.: Extending a smart home multi-agent system with role-based access control. In: 5th Int. Conf. on Internet Tech & Society, Taipei, Taiwan, IADIS Press (10–12 December 2014) 23–30 Best Paper Award.
28. Calegari, R., Denti, E.: The Butlers framework for socio-technical smart spaces. In Bagnoli, F., Satsiou, A., Stavrakakis, I., Nesi, P., Pacini, G., Welp, Y., Tiropanis, T., DiFranzo, D., eds.: Internet Science. 3rd International Conference (INSCI 2016). Volume 9934 of LNCS., Springer (2016) 306–317
29. Denti, E.: Novel pervasive scenarios for home management: the Butlers architecture. SpringerPlus **3**(52) (January 2014) 1–30
30. TuCSoN: Home page. `http://tucson.apice.unibo.it/` (2008)
31. Omicini, A., Zambonelli, F.: Coordination for Internet application development. Autonomous Agents and Multi-Agent Systems **2**(3) (September 1999) 251–269
32. tuProlog: Home page. `tuprolog.apice.unibo.it` (2001)
33. Denti, E., Omicini, A., Ricci, A.: tuProlog: A light-weight Prolog for Internet applications and infrastructures. In Ramakrishnan, I., ed.: Practical Aspects of Declarative Languages. Volume 1990 of LNCS. Springer (2001) 184–198
34. Denti, E., Omicini, A., Calegari, R.: tuProlog: Making Prolog ubiquitous. ALP Newsletter (October 2013)
35. Home Manager. http://apice.unibo.it/xwiki/bin/view/Products/HomeManager (2014)

36. Omicini, A., Zambonelli, F.: Tuple centres for the coordination of Internet agents. In: 1999 ACM Symposium on Applied Computing (SAC'99), ACM (1999) 183–190
37. Omicini, A., Denti, E.: Formal ReSpecT. Electronic Notes in Theoretical Computer Science **48** (June 2001) 179–196
38. Omicini, A.: Towards a notion of agent coordination context. In Marinescu, D.C., Lee, C., eds.: Process Coordination and Ubiquitous Computing. CRC Press, Boca Raton, FL, USA (October 2002) 187–200
39. Omicini, A.: Formal ReSpecT in the A&A perspective. Electronic Notes in Theoretical Computer Science **175**(2) (June 2007) 97–117
40. Casadei, M., Omicini, A.: Situated tuple centres in ReSpecT. In Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M., eds.: 24th Annual ACM Symposium on Applied Computing (SAC 2009). Volume III., ACM (8–12 March 2009) 1361–1368
41. Casadei, M., Omicini, A.: Programming agent-environment interaction for mas situatedness in respect. The Knowledge Engineering Review (January 2010)
42. Casadei, M., Omicini, A.: Situated tuple centres in respect. In: Proceedings of the 2009 ACM symposium on Applied Computing, ACM (2009) 1361–1368
43. Marzaduri, L.: Windows 10 IoT su Raspberry Pi 2: multi-paradigm programming tra Java e C# (2016) Bachelor's Thesis – Scuola di Ingegneria e Architettura, Alma Mater Studiorum-Università di Bologna. http://amslaurea.unibo.it/10299/.
44. Omicini, A., Ricci, A., Zaghini, N.: Distributed workflow upon linkable coordination artifacts. In Ciancarini, P., Wiklicky, H., eds.: Coordination Models and Languages. Volume 4038 of LNCS. Springer (June 2006) 228–246
45. Omicini, A., Denti, E.: From tuple spaces to tuple centres. Science of Computer Programming **41**(3) (November 2001) 277–294
46. GrovePi Home. http://www.dexterindustries.com/grovepi/
47. Carano, M.: Sperimentazione di tecnologie raspberry in contesti di home intelligence. Bachelor's Thesis – Scuola di Ingegneria e Architettura, Alma Mater Studiorum-Università di Bologna (2015)
48. Microsoft Projects. https://microsoft.hackster.io/en-US (2015)
49. Paolini, D.: Geolocalizzazione di servizi in un sistema di home intelligence. Bachelor's Thesis – Scuola di Ingegneria e Architettura, Alma Mater Studiorum-Università di Bologna (2015)
50. Celi, A.: Smart home: reasoning e proattivit applicate ad un caso di studio. Bachelor's Thesis – Scuola di Ingegneria e Architettura, Alma Mater Studiorum-Università di Bologna (2015)
51. Google Meteo. http://www.androidcentral.com/google-now (2014)
52. Yahoo Meteo. https://www.yahoo.com/news/weather (2014)
53. OpenWeatherMap. https://openweathermap.org (2014)
54. Bevilacqua, S.: Home intelligence & social network in the butlers perspective. Bachelor's Thesis – Scuola di Ingegneria e Architettura, Alma Mater Studiorum-Università di Bologna (2015)
55. OAuth: Home page. http://oauth.net
56. Twitter4J: Home page. http://twitter4j.org/en/index.html
57. Zappavigna, V.: Butler vision nella home intelligence: un caso di studio. Bachelor's Thesis – Scuola di Ingegneria e Architettura, Alma Mater Studiorum-Università di Bologna (2015)
58. Ahmad, A., Paul, A., Rathore, M.M., Chang, H.: Smart cyber society: Integration of capillary devices with high usability based on cyber–physical system. Future Generation Computer Systems **56** (2016) 493 – 503