

A Middleware for Intelligent Environments and the Internet of Things

Luis Roalter, Matthias Kranz, and Andreas Möller

Technische Universität München

Arcisstr. 21, 80333 Munich, Germany

`roalter@tum.de, matthias.kranz@tum.de, andreas.moeller@tum.de`

Abstract. Interdisciplinary research from the domains of pervasive computing or ubiquitous computing, computer-human-interaction and computer science has led to the development of many intelligent environments, either on lab scale or as live in laboratories. While several middleware have been developed in this field, no standard middleware for intelligent environments or ubiquitous computing has evolved yet.

We consider the lack of a de-facto standard middleware for distributed sensor-actuator environments as one of the key issues limiting research on intelligent environment and the proliferation of intelligent environments from research environments to their deployment in our everyday lives. In addition, we expect the advent of personal robotics for health care and ambient assisted living scenarios in the context of ubiquitous computing in the close future.

In this paper, we report on the successful application of a robotic middleware as glue between sensors, actuators and services and its application in a deployed example scenario. Thereby, we verify by examples the applicability of robotic middleware for complex ubiquitous computing environments.

To foster re-use and potential community-adoption, we share our source code, documentation and data sets (in the future) via <https://vmi.lmt.ei.tum.de/ros/>.

Keywords: Middleware, Internet of Things, Ubiquitous Computing, Intelligent Environments, Sensors, Actuators, Services.

1 Introduction

While novel information and communication technologies are widely available, bandwidth, processing power and storage are no longer restricting factors, so is currently the lack of a common middleware to interconnect heterogeneous distributed systems. While many middleware have been proposed, none has been accepted by the community as standard yet. The availability of a suitable middleware though would allow to focus on the applications and services intelligent environments can provide to humans and to bootstrap the development of distributed ubiquitous computing systems.

The paper is structured as follows. In Sec. 2 we discuss selected middleware from the domain of ubiquitous computing and general requirements for middleware in the field of intelligent environments. In Sec. 3 we report on an example for an intelligent environment using a middleware from the robotics domain and discuss the applicability of this middleware for intelligent environments against the previously identified criteria. We share our experiences with this middleware, the demonstration scenario, and demonstrate its suitability for intelligent environments. We thereby hope to enable other researchers to accelerate their research on applications and services, instead of focusing on middleware development. Our experiences are summarized in Sec. 4. We conclude our paper in Sec. 5 by giving a short overview on future work.

2 Related Work

Yau et al. [1] divide ubiquitous computing middleware by the way of communication and data exchange in two categories: either data is exchanged by applications communicating via a shared space (such as a Blackboard or Tuple Space) or by an RPC or service-style oriented manner by calling functions and receiving processed information. Yau et al. also extend the notion of middleware from connecting heterogeneous distributed sensors and actuators (we subsume anything that “(re-)acts” on information, such as agents, displays, etc., as actuators) to context-awareness. While context-awareness requires more than the mere connection of inputs and outputs, and more application scenarios do require more specialized middleware, the basic problem of meaningfully interconnecting devices and applications so far has not been concludingly addressed. As Nakajima et al. [2] describe their middleware, they state that most middleware does not “offer generic services for building ubiquitous computing applications. They support to develop applications for specific domains to realize ubiquitous computing visions”. A one size fits all solution will most probably not exist, though we think that for *most* scenarios a common basis, as we will later introduce in our example scenario, could be beneficial. The urge of a suitable middleware becomes more pressing as the Internet of Things [3] also demands for a middleware that allows both data management and for interaction with the Internet of Things [4].

The main requirements, characteristics and design issues for middleware in intelligent environments have been discussed extensively in the literature [2, 1, 5, 6, 7, 8, 9, 10]. Summarizing the selected requirements by high-level keywords, an ideal of a middleware would allow for

- abstraction over heterogeneous input and output hardware devices
- abstraction over hardware and software interfaces
- abstraction over data streams (continuous or discrete data or events) and data types
- abstraction over physicality (location, context)
- abstraction over the development process (time of integration of services or devices)

While specific issues have been addressed in various research efforts, no standard middleware for intelligent environments has yet evolved in the area of ubiquitous computing.

Different middleware systems, such as GAIA [10] and MundoCore [9], have been proposed and used in the relatively young research field of pervasive and ubiquitous computing. The challenges of distributed multimodal information processing, connecting heterogeneous input and output technologies have very different demands towards middleware systems. Unfortunately, reuse and finally development in this domain is limited usually to the initial developers of a respective middleware and no community yet evolved to pursue the ambitious goal of a unified middleware.

This vicious circle of no or only limited reuse (both in more projects and by more researchers) and thus the lack of necessary extensions or drivers finally leads to the neglection of available middleware again. Existing middleware also have not been designed to have a long development life cycle beyond the end of the research project and to allow for future integration of demands and upcoming technologies.

We therefore investigated the available middleware especially with a focus on community support, maturity, extent of supported hardware and software, and data management architecture.

As promising candidates we identified two middleware systems from the robotics domain, Player/Stage [11] and ROS (Robot Operating System) [12]. Key factors supporting the idea of using a middleware from the robotics domain are:

- The challenges with respect to the heterogeneous devices and interfaces in robotics seem very similar to those found in the context of intelligent environments.
- Player as example has reached a large the maturity - this middleware is used, supported and further developed in the robotics community for more than 10 years by now.
- Conceptually an intelligent environment is very similar to a static, non-movable robot, a so-called “ImmoBot” [13].

Player so far has already been reported to have been used in the context of intelligent environments [14, 15]. As ROS is intended to be a successor of Player and also is downward compatible, we opted to assess the potential of ROS in the use case presented in Sec. 3.

Therefore, we took the view of a robotic systems developer and investigated the potential of a robotic middleware for distributed, heterogeneous, sensor-actuator-based, communicating intelligent environments. As ROS is downward compatible w.r.t. existing drivers and components, and includes many modern concepts of distributed architectures, we decided to explore the potentials of ROS in more details.

The data management architecture includes decentralized peer-to-peer network concepts, publish-subscribe information distribution or bi-directional services between components. The middleware not only allows for inclusion of an

immense variety of sensing and actuation systems, but also to visualize and simulate, both the information flow and the physical space using e.g. OGRE (open source 3D Graphics Engine, <http://www.ogre3d.org>) and ODE (Open Dynamics Engine, <http://www.ode.org>) open source engines. This allows designing 3D objects in a CAD style manner, such as the different service cores, investigating their interaction and sending the very same information as the deployed sensor-actuator system would do - well before any physical prototyping is done. This reduces the time needed for iterative development and refinement and also costs. Additionally, the physical paths a human would have to take in such an environment can be predicted, calculated and optimized already during the development phase in the middleware. This also has an impact on the prediction of the interaction times with the different digital systems. The inclusion for real world simulation capabilities originates from e.g. robotic SLAM where algorithms have to be re-tested often but a real experiment is quite expensive and thus cannot be re-conducted for each run of a test.

3 Cognitive Office - An Open Test Bed for Ubiquitous Computing Research

We will now report on the application of the chosen robotic middleware ROS in the context of an intelligent environment, the Cognitive Office. This environment is a normal office room in our faculty that has been transformed into an 'intelligent environment'. The room is a one person office space and actively inhabited and used for all kinds of office and research work and thus a "normal" environment. As physical entity in space, the office is considered to be an ImmoBot [13].

Following Mark Weiser's philosophy at Xerox PARC of "Build what you use, and use what you build" this environment serves as live in lab [16]. The 3D Gazebo model of the office space, as seen from the center of the room, is shown in Fig. 1 and Fig. 2.

As for most intelligent environments, a "final" set of devices and services is unlikely to be known in advance and also most probably will never exist, it was important that sensors, actuators, interfaces and services can be added at any point of time in the development phase.

We were interested in integrating a large variety of different hardware devices – from industry components such as an IP networked power switch to research hardware as Phidgets (<http://www.phidgets.com>) to end user prototyping micro-controller-based systems as Arduino (<http://www.arduino.com>). From software side, we wanted to be able to integrate external web based services such as RSS news feeds or social networking platforms such as Twitter. Our intention here was to connect physical and virtual worlds, especially regarding community based services to allow the intelligent environment to share its digital data with human users in a more convenient manner.

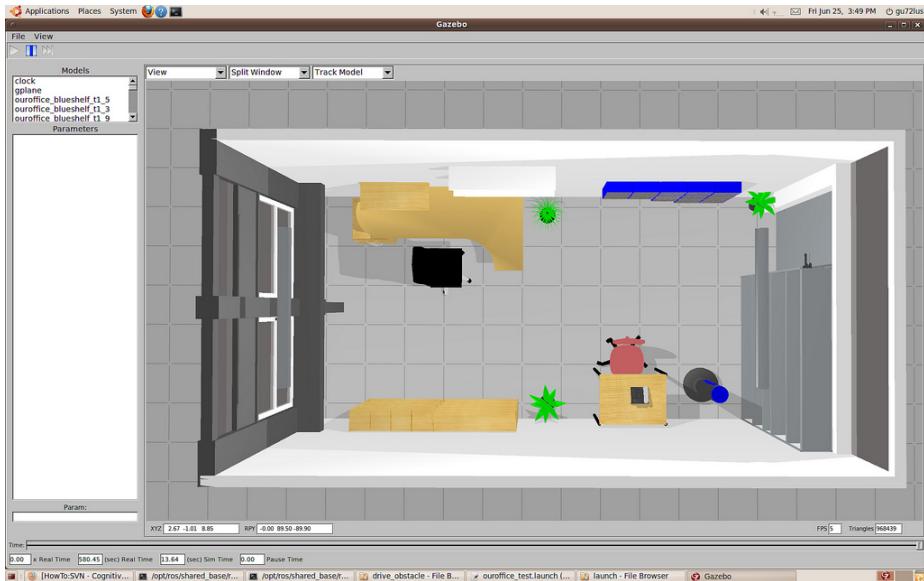


Fig. 1. The Cognitive Office visualized in the Gazebo viewer running on top of the ROS middleware. The ceiling has been made transparent, several light sources have been placed inside the room model. The model should look as realistic as possible.

3.1 Sensors and Actuators

For our example to be realistic, we tried to be as inclusive as possible regarding the set of sensors, devices, actuators, and services and to include what has been used in most comparable intelligent environments. As of now, we have connected the following sensors and actuators via various interfaces to the robotic middleware:

- ultrasonic and PIR sensors for movement detection via Arduino boards
- reed contact switches on the door, windows, and drawers via Phidgets Interface Kit boards
- light and temperature sensors via ZigBee
- RFID sensors via a dedicated web service
- instant messaging data (mood, activity) via plugins
- social networking data (Twitter) via web service
- IP power switches via a HTTP web server
- traffic and weather information via RSS news feeds
- web cam via USB
- moisture, temperature and light sensors for office plants via Twitter (Fig. 3)
- ...

This list could be continued, but already gives an idea of the complexity and diversity of the developed intelligent environment. We have shown that many commonly found devices and systems in ubiquitous computing environments are easily integratable with the ROS middleware. So could an environmental-based

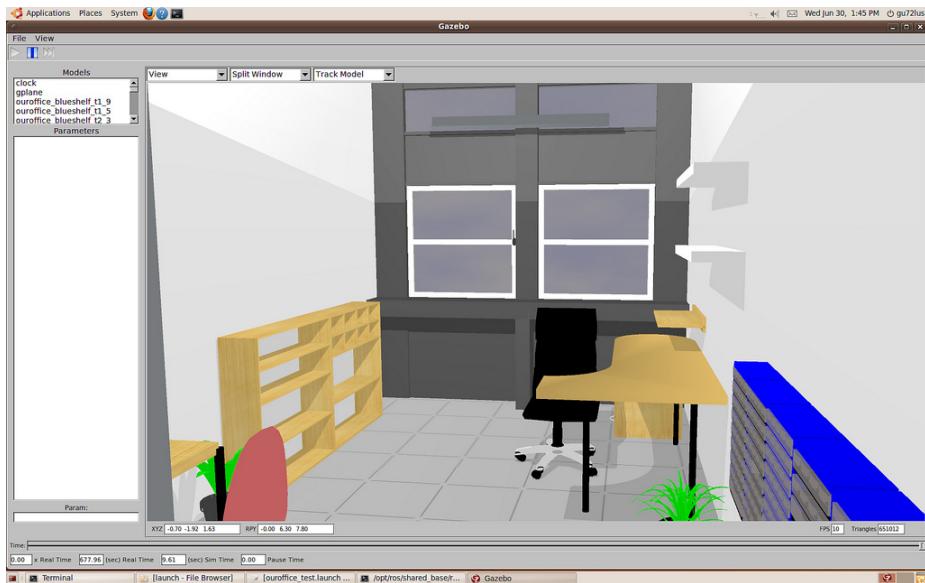


Fig. 2. Part 2 of the Cognitive Office: the workplace with shelves. The 3D model is a 1:1 match of the physical real-world office. All elements – drawers, windows – can be controlled by physical controllers, detecting and visualizing the state of the real, physical world.



Fig. 3. The “Botanicalls” (<http://www.botanicalls.com>) device directly publishes e.g. moisture information to Twitter. This data is imported via a web service using the Twitter API into the ROS middleware and published to the plant care controller.

Thracker [17] be added for detecting pick and place tasks in the shelf, or an tangible user interface for instant messaging [18] be easily added to the ecology of devices commonly found in modern office spaces.

We have proven by constructing examples that ROS makes an interesting candidate middleware for further investigation in the context of ubiquitous and pervasive computing. Future work will e.g. include a comparison on e.g. the lines of code to integrate systems in different middleware or how a complex task it is to support new systems in ROS.

3.2 Event and Service-Based Data Exchange

Data from various sources and heterogeneous sensor and actuator devices is centrally managed by a ROS middleware server instance located in the Cognitive Office. As ROS allows to “connect” several servers, we deployed a second system in another office in another building where co-workers are located. Data from both servers then is available to data consumers, e.g. on a mobile Linux based device for services such as location information. Location information is visualized through our university’s visitor service (see Fig. 5).

Part of the above mentioned sensors deliver event-based data which are then published to registered listeners. This blackboard-like publish/subscribe architectures allows to connect an arbitrary number of information producers and consumers. A naming service allows to “search” for topics of interest, such as location information.

Other information is exchanged via request/response services, using dedicated message exchange formats. Examples in our intelligent environment are for example the day length calculation to obtain the amount of daylight. The service sends the date information and receives an answer containing the time information. This information is used by the plant care service that ensures that the plant receives enough daylight to support optimal growing. This is only computed once a day.

While being a simple service and implementation, it still proves the support of services in general and of closed control loops across different distributed heterogeneous devices and systems in the ROS middleware for intelligent environments as they are commonly found in ubiquitous computing.

3.3 Context Services

As Schmidt et al. [19] state, there is more to context than location. We have implemented a rudimentary set of context inference services on top of the middleware to provide convenient services to the office user. The following list gives an idea of currently implemented context services:

- length of day
- date and time
- weather information
- appointments and calendar information
- status information
- disruptability
- ...

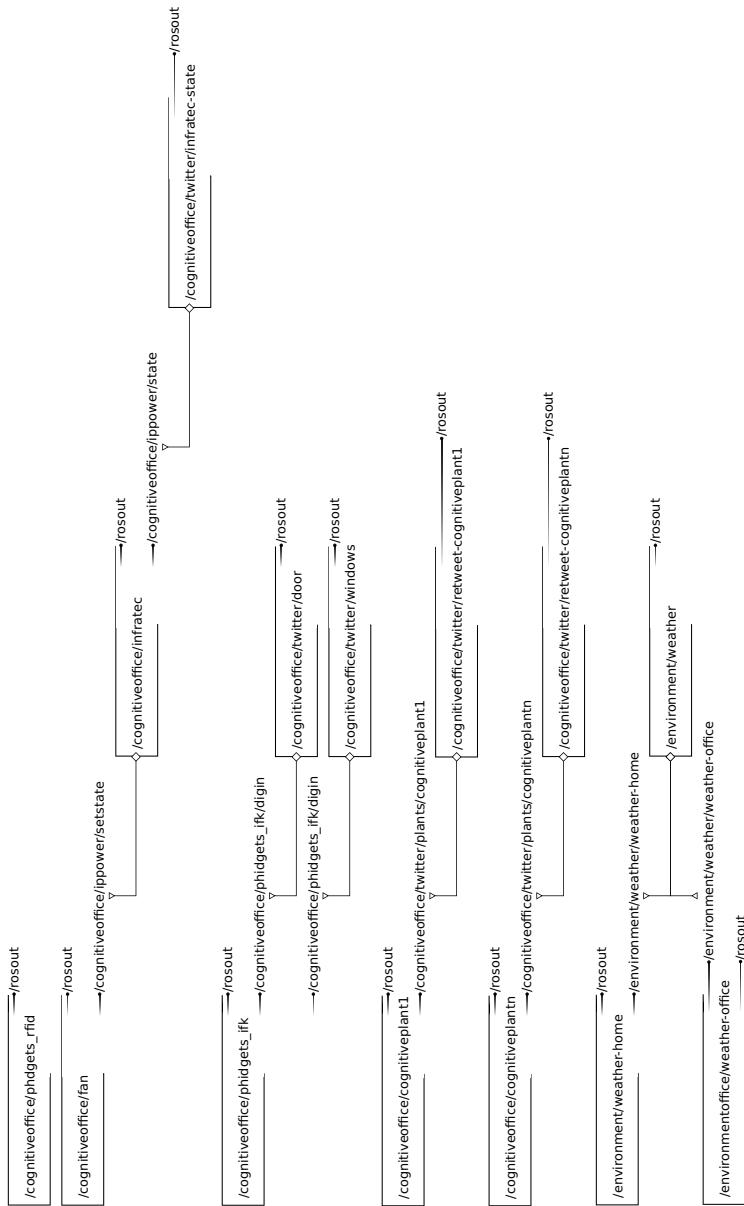


Fig. 4. Event-based data is made available via a publish/subscribe data management mechanism. Examples are open/close events of doors and windows acquired by local sensors or traffic information acquired by remote RSS news feeds.

As location is an important cue to activity and context information, both for the middleware and e.g. office visitors such as students, the computed location information is shared on a public display outside the office. A privacy filter ensures that only abstracted and public information is displayed, e.g. the general information “away for today” or “in lecture”. The latter is augmented with a map of the lecture room using a university map service (see Fig. 5).



Fig. 5. Context information, such as the location of the office user, are obtained by the Cognitive Office’s custom services. The location information is then visualized, e.g. to visitors, on a wireless picture frame outside the office room. The location is, after computation, fed into the university’s room service system, the resulting image downloaded and automatically made available to the picture frame via an UPNP media server connected to the ROS middleware or via a RSS news feed.

As you may have noticed, the room-precision of the point can only be reached if you already know the room where the person is located. This will be automated soon when the annotation tools as shown in Fig. 6 will be finished. We will provide multiple ways for the indoor localization, such as WLAN and DECT [20].

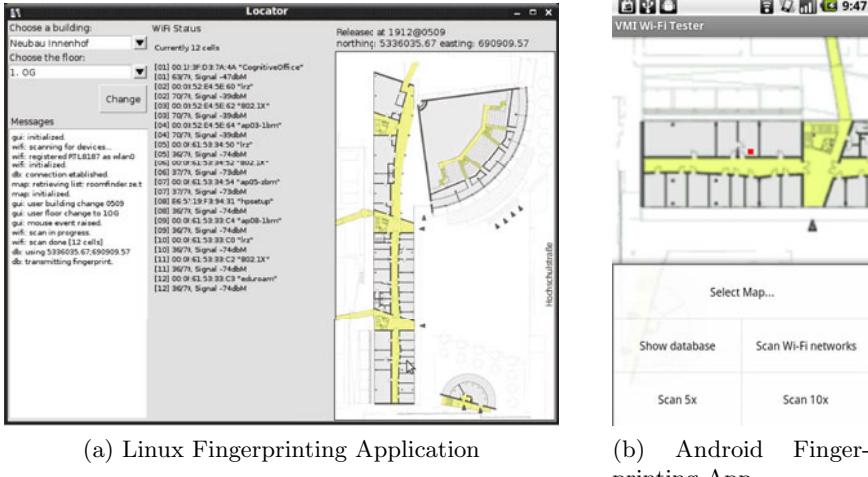


Fig. 6. Creating precise annotation from WLAN fingerprints to obtain the current location of the user. This picture shows the user interface which is used to annotate the WLAN fingerprints on a laptop around the building (a). The locating application will run without a complex UI on an iPhone or Android Smart Phone as well (b).

3.4 End User Services

Other example services, besides location information for colleagues, are travel information. Using data from the user's personal calendar the travel times to meetings and for the drive home are estimated. Depending on the targets (in campus, off campus, conference, ...) and the preferred travel methods (foot, car, ...), a prediction on the time necessary to go from A to B is made, including latest information such as RSS news feeds on the current traffic situation. This is presented to the user, e.g. to notify him to start traveling home earlier as a traffic jam on his route is to be expected.

Simpler services are responsible for automatic lightning of the individual work spaces where presence is detected e.g. by the usage of a computer or by ultrasonic distance sensors. The lights are then automatically switched on via an IP power switch to provide convenient lightning. A last example for a personalized service is the "cold coffee warning". As the office user sometime is busier than expected, a fresh coffee can get cold. To prevent the user to take a sip of an ice cold coffee, an intelligent coffee cup is used. The idea of the cup is similar to Beigl et al. [21]. Our version features temperature sensors inside the cup, presence information is acquired using an RFID augmented cup place holder (see Fig. 7).

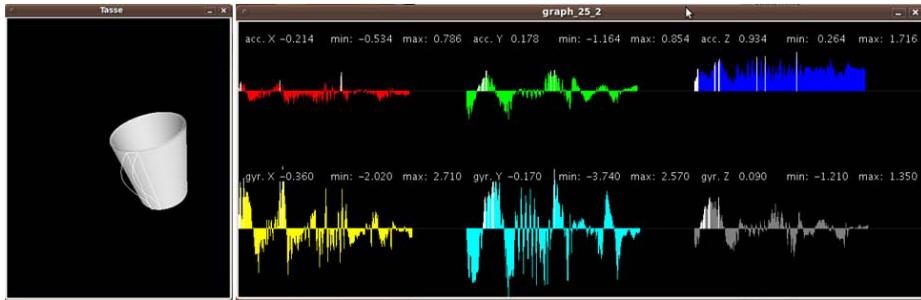


Fig. 7. The Cognitive Cup features an ATMEL microcontroller, a ZigBee RF transceiver, a 6 DOF IMU (3D acceleration and 3D gear rates), a temperature sensor and an embedded RFID tag. The figure shows the orientation information in 3D space calculated from its embedded IMU. The data is wirelessly communicated to the middleware and in case of cold coffee, a warning is displayed on the user's screen when a RFID reader detects the “leave” event equaling to the take event of the cup.

The set of services is currently focused on automated building control, e.g. a temperature and climate control service has been implemented. Future services will also focus on multimodal computer-human interaction.

To achieve a better integration of the digital information available in the technical system of the middleware, we connected all event based outputs (excluding video and audio streams) to a social networking platform. While ROS directly allows for convenient visualization of and direct interaction with all sensors and actuators, we feel that this way of visualization is not appropriate to “normal” users. We here think of using ROS in the context of an independent living project [22] and a care giver wishing to access the data. A development tool will not be appropriate in this scenario.

As Twitter has recently been used e.g. during clinical operations to inform relatives about the progress (<http://scienceroll.com/2009/01/19/twitter-live-surgery-sugarstats-and-100-ways-for-hospitals/>), we chose to also use Twitter and finally make use of the rich set of visualization tools on top of Twitter (see Fig. 8).

A public information display presents selected information to office visitors, such as the current location (e.g. when lecturing) or the next consultation hours. Data is acquired from many sources, text and images are generated using standard Linux tools such as “convert”, and this data is finally published on a connected UPNP media server and displayed on a WLAN picture frame outside the office.

4 Experiences

As our overall experiences were positive, we will begin with the negative impressions and conclude with our positive experiences.

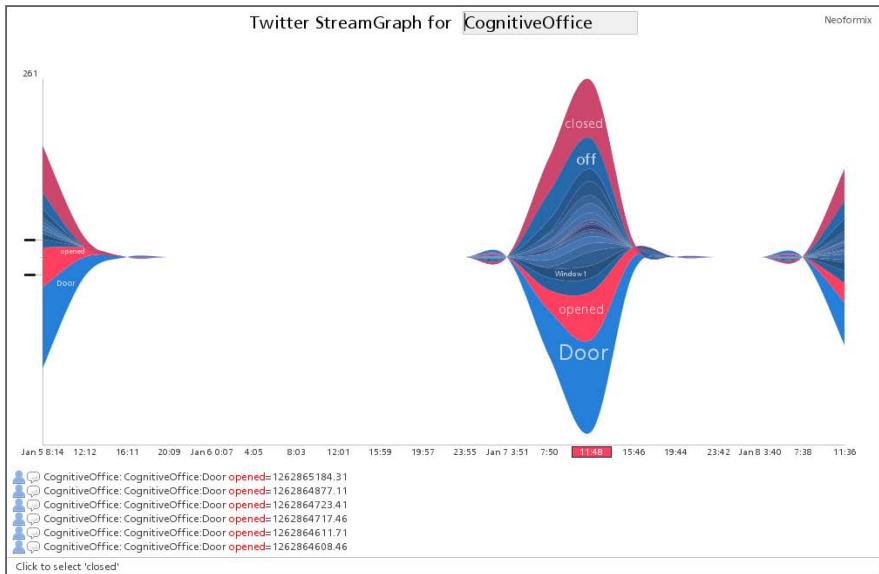


Fig. 8. Visualization of sensor and actuator event data published to a social networking platform. The data here was experimentally sent to a Twitter account and later visualized using Neofomix StreamGraph web-based tool. A normal user can easily see e.g. which events occurred at what time and what information was acquired and processed in the middleware.

The documentation of the middleware is, in parts, in an early state, but fast and constantly improving. This can be attributed to the youth of ROS as successor of Player. The search results of the ROS website often did not contain valuable information and we had to look at the code in order to complete some parts. Though, now tutorials and additional documentation are added at a high rate.

As the middleware is, in our view, very capable and impressive, the initial way into the structure and features was sometimes hard. Initial tutorials exist and provide very valuable starting points but are really required reading to be able to use ROS.

The built system (we used ROS with Ubuntu Linux) was very convenient – missing Linux system packages are downloaded on demand and dependencies with other ROS packages are automatically resolved. The rosmake build tool allows easy compilation of the packages. The middleware, examples and tutorials work out of the box and allow adoption for initial own interfaces. After two weeks the first real interfaces and parts where working, allowing us to dig deeper inside the middleware. A to do for future work for us will include a comparison e.g. of development tools, speed and lines of code of ROS and other ubiquitous computing middleware.

The development tools, besides the build system, include tools for listing available topics and services and to visualize them. Fig. 4 is the output of the

rxgraph tool. Complex sets of topics and services can be stored in configuration files (.launch files), the addition and removal of services is possible at runtime. Logging and playback of all data is possible and support faster algorithm development. The (2D/3D) visualization of the physical sensors and actuators further speeds up the development process. The visualization tool (Gazebo) also includes the possibility of triggering sensor events by clicking – e.g. to click on the door (Fig. 1) and generate the “open” event. For intelligent environments this implies that services and applications can already be prototyped long before an intelligent home would be build as virtual and real events generate the same data.

The possibility of interconnecting several ROS servers, allowing to partition computation, easily supports developers of intelligent environment to build a more complex system in a Lego brick manner.

We developed software using C, C++, Java and Python to implement the presented environment. This choice of languages allowed us to develop efficient code (in C) where necessary and to quickly prototype (in Python) novel ideas.

From first hand experiences with other middleware systems such as the EITool Kit [23] for computer-human interaction in distributed environments or with wireless sensor node based environments featuring Particles or Crossbow hardware, Player and ROS, as well as from discussions with colleagues, we feel that ROS is a very interesting candidate middleware for intelligent environments.

5 Conclusions and Future Work

After the investigation of available middleware systems for intelligent environments, we chose an open source, community-supported middleware from the robotics domain to develop a distributed sensor-actuator-system. We explored the potentials of the ROS middleware in the context of the use case of an intelligent office environment. We connected a diverse set of heterogeneous devices via different physical media and implemented a set of initial services.

We shared our experiences gathered during the incremental development process of the presented intelligent office environment.

Our experiences so far are very positive and we therefore think that this development experience report can be of great value for researchers and developers of intelligent environments.

To foster research, we invite researchers to use our scenario as starting point for using ROS as middleware in ubiquitous computing. Therefore, we share our 3D models and code at <https://vmi.lmt.ei.tum.de/ros/>. We – in addition to the available documentation of ROS at <http://www.ros.org> – share our data, code and documents with the UbiComp community.

Resources

Open source code, documentation and data sets (in the future) are shared via <https://vmi.lmt.ei.tum.de/ros/>.

Acknowledgments

This work has been funded in parts from the German DFG funded Cluster of Excellence ‘CoTeSys – Cognition for Technical Systems’.

We also acknowledge the individual contributions of our students and wish to thank them for all their efforts.

References

1. Yau, S.S., Karim, F., Wang, Y., Wang, B., Gupta, S.K.S.: Reconfigurable context-sensitive middleware for pervasive computing. *IEEE Pervasive Computing* 1(3), 33–40 (2002)
2. Nakajima, T., Fujinami, K., Tokunaga, E., Ishikawa, H.: Middleware design issues for ubiquitous computing. In: *MUM 2004: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, pp. 55–62. ACM, New York (2004)
3. Floerkemeier, C., Langheinrich, M., Fleisch, E., Mattern, F., Sarma, S.E. (eds.): *IOT 2008. LNCS*, vol. 4952. Springer, Heidelberg (2008)
4. Kranz, M., Holleis, P., Schmidt, A.: Embedded interaction: Interacting with the internet of things. *IEEE Internet Computing* 14, 46–53 (2010)
5. Landay, J.A., Borriello, G.: Design patterns for ubiquitous computing. *Computer* 36(8), 93–95 (2003)
6. Aiken, R.J., Abramski, A., Bates, J., Blackadar, T.: Middleware for Ubiquitous Computing. In: Gellersen, H.-W. (ed.) *HUC 1999. LNCS*, vol. 1707, pp. 301–303. Springer, Heidelberg (1999)
7. Nakajima, T.: Case study of middleware infrastructure for ambient intelligence environments. In: Nakashima, H., Aghajan, H., Augusto, J.C. (eds.) *Handbook of Ambient Intelligence and Smart Environments*, pp. 229–256. Springer, New York (2010)
8. Su, H., Fu, X., Li, Z., Yang, Q., Teng, S.: A Service-oriented Middleware for Pervasive Computing Environments. In: *1st International Symposium on Pervasive Computing and Applications*, pp. 36–41 (2006)
9. Aitenbichler, E., Kangasharju, J., Mühlhäuser, M.: Mundocore: A light-weight infrastructure for pervasive computing. *Pervasive Mob. Comput.* 3(4), 332–361 (2007)
10. Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: A middleware infrastructure for active spaces. *IEEE Pervasive Computing* 1(4), 74–83 (2002)
11. Collett, T.H., MacDonald, B.A., Gerkey, B.P.: Player 2.0: Toward a practical robot programming framework. In: *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, Sydney, Australia (2005)
12. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T.B., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: *ICRA Workshop on Open Source Software* (2009)
13. Goldman, R.P., Baral, C.: Robots, softbots, immobots: The 1997 aaai workshop on theories of action, planning and control. *Knowl. Eng. Rev.* 13(2), 179–184 (1998)
14. Kranz, M., Schmidt, A., Rusu, R., Maldonado, A., Beetz, M., Hornler, B., Rigoll, G.: Sensing technologies and the player-middleware for context-awareness in kitchen environments. In: *Fourth International Conference on Networked Sensing Systems, INSS 2007*, pp. 179–186 (June 2007)

15. Kranz, M., Schmidt, A., Maldonado, A., Rusu, R.B., Beetz, M., Hörnler, B., Rigoll, G.: Context-aware kitchen utilities. In: TEI 2007: Proceedings of the 1st International Conference on Tangible and Embedded Interaction, pp. 213–214. ACM, New York (2007)
16. Intille, S.S., Larson, K., Tapia, E.M., Beaudin, J., Kaushik, P., Nawyn, J., Rockinson, R.: Using a live-in laboratory for ubiquitous computing research. In: Fishkin, K.P., Schiele, B., Nixon, P., Quigley, A. (eds.) PERVASIVE 2006. LNCS, vol. 3968, pp. 349–365. Springer, Heidelberg (2006)
17. Wimmer, R., Holleis, P., Kranz, M., Schmidt, A.: Thracker - using capacitive sensing for gesture recognition. In: Proceedings of the 26th IEEE International Conference Workshops on Distributed Computing Systems, ICDCSW 2006, p. 64. IEEE Computer Society, Washington (2006)
18. Kranz, M., Holleis, P., Schmidt, A.: Ubiquitous presence systems. In: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC 2006, pp. 1902–1909. ACM, New York (2006)
19. Schmidt, A., Beigl, M., Gellersen, H.-W.: There is more to context than location. Computers and Graphics 23(6), 893–901 (1999),
<http://citeseer.ist.psu.edu/schmidt98there.html>
20. Kranz, M., Fischer, C., Schmidt, A.: A comparative study of dect and wlan signals for indoor localization. In: PerCom., pp. 235–243. IEEE Computer Society, Los Alamitos (2010)
21. Gellersen, H.-W., Beigl, M., Krull, H.: The mediacup: Awareness technology embedded in a everyday object. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 308–310. Springer, Heidelberg (1999)
22. Kranz, M., Linner, T., Ellmann, B., Bittner, A., Roalter, L.: Robotic service cores for ambient assisted living. In: 4th International Conference on-NO PERMISSIONS, Pervasive Computing Technologies for Healthcare (Pervasive Health), pp. 1–8 (22-25, 2010)
23. Holleis, P., Schmidt, A., Paasovaara, S., Puikkonen, A., Häkkilä, J.: Evaluating capacitive touch input on clothes. In: ter Hofte, G.H., Mulder, I., de Ruyter, B.E.R. (eds.) ACM International Conference Proceeding Series, pp. 81–90. ACM, New York (2008)