Modular Approach for Expert System toward Anomaly: N-Layers

Etienne Pardo, David Espes, Philippe Le Parc

Modular Approach for Expert System toward Anomaly: N-Layers

Étienne Pardo*†, David Espes*‡, Philippe Le-Parc*§,

* Université de Bretagne Occidentale — LabSTICC, 20 Avenue Le Gorgeu, 29200 Brest, France,
†etienne.pardo@univ-brest.fr,
‡david.espes@univ-brest.fr,
§philippe.le-parc@univ-brest.fr

***Abstract*—Smart cities and smart homes are booming fields of development of pervasive systems. With the high stakes these systems have to manage, and their sheer complexity, anomalies have to be considered. In these complex systems are many connected components with computing capacities. They can manage anomalies, even if partially, and can act as some kind of expert systems. These expert systems can be relied upon to provide anomaly management.**

The complexity to manage generic expert systems brings us to suggest another approach. In this paper, the N-layers is discussed. It layers the various existing expert systems to organize them as a more generic expert system. It also aims to correct some difficulties to develop and integrate generic expert systems into various scales complex systems, such as smart cities or smart homes.

I. Introduction

Pervasive systems, as smart cities (SC) or smart homes (SH), are more and more relied upon. These systems provide efficient usage of their infrastructure, to serve best their users. Among the functionalities that target SH and SC, one can cite the ambient assisted living (AAL) or the smart grid (SG). In AAL, the users' living standard is improved via assisting technology. In SG, the overall resource consumption and distribution is optimized to lessen wastes.

No matter its origin –be it attrition, design, malevolence, and so on–, an anomaly may occur in any system. Such occurrence is more likely the more the system is complex. More so if the stakes are high, as is the case when daily life may be affected. Smart cities and smart homes are no exception. These complex systems are made of many connected components, some of which are highly specific, while others are more generic.

It is sensible to consider that among the various components, there are expert systems (ES). These are the intelligence provider of the system. From the information they gather, they analyze the world to provide appropriated reactions of the system. A perfect expert system would offer: a wide scope; an easy maintenance; and a high reliability. Although there are steady improvement, the choice and configuration of an expert system boils down to the selection of at most two properties.

978-1-5090-0284-9/16/$31.00

When an anomaly occurs, a few steps are commonly done. Their exact number and precise ordering may change, but the broad idea remains the same.

1) Discovery: the anomaly must be detected, and then reported, to alert the system of its occurrence.
2) Reasoning: information about the anomaly must be gathered, and analyzed to improve the system's knowledge of the situation.
3) Termination: with enough knowledge, a solution to improve the overall situation can be tried. Of its outcome depend whether the anomaly is closed, or left open.

To ease this management, it may be useful to use a generic, anomaly related ontology [4], [10], [11]. Thus, the relevant information can be properly formulated and transmitted.

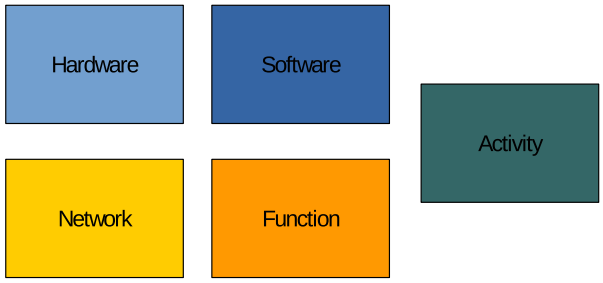

Fig. 1. The various kinds of anomaly

Once the anomaly is declared: it is detected and the system is alerted; it has to be analyzed in order to be properly resolved. Throughout literature, the anomaly is related to a few types, each with their specific solutions. Usually, the distinction (figure 1) is made [15] between hardware related anomalies, software related anomalies, network related anomalies, and functioning related anomalies.

Hardware and software anomalies are mainly related to "*WHAT* in the system" is affected, i.e.: the affected components. Network and function anomalies are more related to "*HOW* the system" is affected, i.e.: the relation between components.

In addition to these, any user oriented technology (i.e.: SH and AAL) requires anomalies related to the users' daily activities [12]. Such anomalies are centered on the user.

Therefore, they may not be related to which component is affected, nor how is affected the system. In fact, the system and its components may as well be perfectly fine.

No matter the anomaly, when it occurs, it has to be managed and resolved. This is the role of the expert system, as discussed in section II.To improve their usage, out proposal, the N-Layers, is detailed in section III. A test implementation is described in section IV.

II. Expert System

Expert systems (ES) are products of the artificial intelligence research. These are components of a system, built to resolve certain situations. Initially, the role of expert systems is to provide feedback analysis on large quantity of data.

In smart cities, smart homes and similar fields, it is sensible to expect that many ES are scattered inside the infrastructure. Each one of them may be unique and required. But there are many chances that many ES are redundant one toward another.

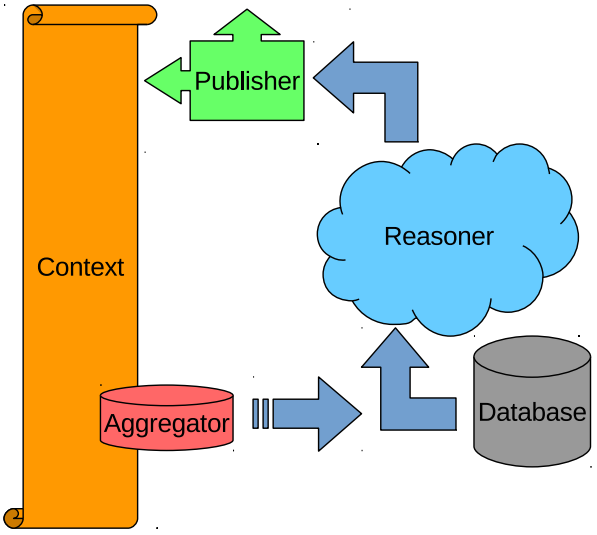

Fig. 2. Schematic description of expert systems

These decision makers/helpers can be perceived as black-boxes, more or less specific to a field. They receive a context –a situation description– as input and emit a decision as output. Though the focus is mostly on the reasoner part, a complete expert system is built (figure 2) from an aggregator, a reasoner, a database and a publisher.

The aggregator receives various information –from the "context"– and produces a meaningful input for the reasoner. Depending on the peculiarity of the reasoner, the aggregator might process more or less its inputs. It may as well store temporarily the input context.

The database, or knowledge base, stores the "expert's knowledge" used by the reasoner. Depending on the expert system, it may be fused within the reasoner, or be an inert database. It should not be mistaken with the configuration. The configuration relates to the expert system's state, and is used before its usage, by setting parameters in the aggregator, the publisher... The database is relevant during the usage of the expert systems, and relates to the context processing.

The reasoner is, with the knowledge base, the core of the expert system. It is the reasoner that produces, from the context and its database, outputs information.

The publisher is almost the mirror of the aggregator. From the output of the reasoner, it publishes the relevant messages. The improved situation description is added to the context, while the required action, is sent to the relevant components. For example, if the reasoner's output is that there is an anomaly, the publisher emits an appropriated alert.

The various parts of an expert system are tightly coupled. They are either deployed as a whole; or the reasoner and the database are provided, and the publisher and aggregator have to be built ad-hoc.

All things being equal, an expert system is a compromise between:

- its upkeep, the ease of maintenance (update, deploy...);
- its reliability, the trust the system can give to its responses;
- its scope, whether it is generic or specific to a given field.

An expert system may be made to change "on-the-fly" this compromise. But for most expert systems, the reliability and the scope is directly linked to the knowledge base, whereas the upkeep is linked to the cost of changes the knowledge base.

Historically, the first kinds of ES –the reasoners based on rules, models and cases– are imitation of human experts' strategies, hence the name. Each follows an iterative reasoning process explicated by experts in a given field. More recent approaches move away from these explicit processes to focus on the modelization of either the field, or the expert reasoning.

Basically, rule-based reasoners (RBR) and model-based reasoners (MBR) are similar. Once received, the context is matched against each rule's condition or requirement. The items whose rules/requirement are selected have their actions triggered [9], [16]. RBR use set of "if-then" rules as knowledge base. Whereas MBR use set of "for-requires" rules as knowledge base.

RBR and MBR each expresses a different condition-action relation. This boils down to the expression of models as logical rules (RBR) or set of properties (MBR). In RBR, it is easier to express one condition for many actions, whereas in MBR, it is more pragmatic to express many conditions for one action.

Case-based reasoners (CBR) use set of "situation-response" cases as knowledge base. The context is matched against each case's situation. The more relevant cases are selected, and their responses mixed to produce what is expected to be the correct solution [8], [14]. CBR relies on the idea that similar situations expect similar responses, and that the similarity at both ends can be automatically computed and produced.

Bayesian networks (BN) (and Markov networks) can be used for the modelization of the knowledge of the field [6], [13]. In such networks, the knowledge is not stored as factual, definitive rules. It is expressed as probabilistic relations

between properties. Reasoners can rely on such networks to try and predict evolution of the situation.

Neural networks (NN), as BN, tend to fuse their database with their reasoner [3], [5], [17]. In NN, the reasoner does not emulate conscious, human reasoning process, but tries to emulate the brain's knowledge processing.

On the whole, expert systems are well adapted to specific cases management. Still, each has some flaws yet to overcome. As effective as they are, ES lack the genericity and the plasticity of a human being. They may be more efficient than a human in their specific field. But their upkeep is related to the complexity (size and peculiarity of the cases) of their database[1]. Moreover, each technology of ES has its own strength, and is more suited for certain aspects of problematic.

RBR and MBR suffer from the complexity to improve their knowledge base. Easy to define for a small knowledge set, each addition is trickier than the previous one; new knowledge may interfere with existing, validated knowledge. Hence, conflict resolution may require human intervention, as the meaning and worth of the rules rely on human metrics.

CBR is less impacted, but may still suffer from a "too big" case base. The more complex and the more cases there are, the more time is required to select the relevant cases. Moreover the similarity cases and the responses fusing algorithms have to be written, and are often ad-hoc. Worse, some cases with similar situation may require contradictory responses. Algorithms have to be adapted if such a possibility is expected to arise.

NN may be perceived as an "easy to configure solution": sending in inputs and outputs, until the desired outputs are reached. A special care has to be given to avoid over-fitting: losing the genericity of NN for matching exactly the learning set. In addition, each evolution of the NN requires to re-run the full validation process, as new knowledge may alter the previously valid responses.

Expert systems require careful configuration, especially when their (or their knowledge's) complexity increases. Yet when their field is correctly constrained, and not expected to change, ES are "perfect generic ad-hoc solutions".

On a side note, it may be possible to classify some ES by their processing time. For a similar complexity, RBR and MBR performs in less time than CBR[2], as they have less computational complexity. This is a gross approximation, as each expert system is more suited for a set of situations. And to exactly compare the process time, one should define metrics to compare situations' complexity.

III. N-Layer

Pervasive systems, as smart cities or smart homes, are complex systems that rely on distributed components of varying "intelligence". These components are parts of their integrating systems, and usually independent from the rest of the whole system. Among these components, some may be considered as "expert systems" in a broad sense.

The more a system is: complex; relied upon; deployed, and the higher the stakes are, the more anomalies' occurrence has to be taken into account. Pervasive systems –smart cities, smart homes…– and even expert systems, are no exception.

Anomalies can be managed by an expert system. In spite of steady progress, conceiving a perfect, generic expert system is a nigh impossible task. A well-suited, specific ES is easier to toughen against its own errors, to the detriment of its genericity. Easier to design and manage, and of a lesser scope, such an expert system can be grouped with other similar ones. In a pervasive, connected world, such as SC or SH, there may be many redundant, "small" expert systems, as well as some domain-specific ones.

For example, a smart city can be built from inter-connected smart homes, each with their own set of expert systems. With cooperation, a dysfunction in an ES of one smart home can be mitigated while repairs are under way, by switching to the same ES of the next home. Or some functionalities can be improved by sharing information between homes; taking into account information otherwise not accessible at this level [7].

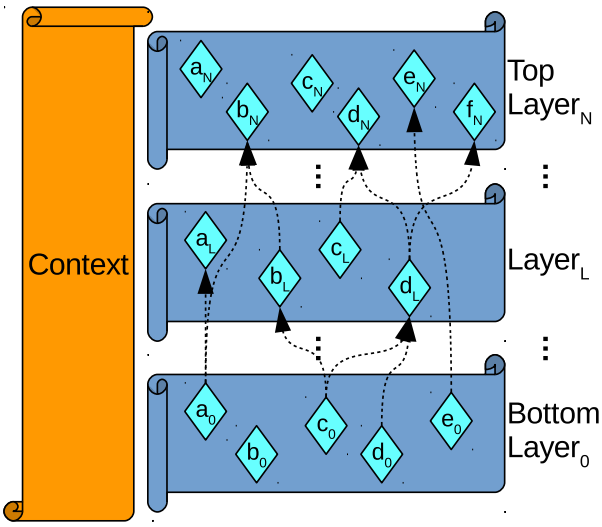

Fig. 3. Schematic overview of the N-Layer architecture
The light blue diamonds represent expert systems, of various kinds. The arrows denote possible dependencies. The aggregation from and publication to the context are implied for every expert system.

As pictured by figure 3, the solution suggested in this paper, the N-Layer, makes hierarchical layers of expert systems. In a similar fashion to [2], the various expert systems improve the context, and provide their responses to the rest of the system. However, the expert systems are organized as layers, based on their authority and their dependencies.

These dependencies are service inspired: any expert system similar enough can be relied upon. SH and SC are expected to have many ES, of varying redundancy. Hence, in a system of systems, the various specific expert systems can be used

[1] or the equivalent, as the learning set for NN

[2] BN and NN (and others) are more dependant on more complex parameters than "database size" and "activation or response fusing"

together to provide improved efficiency.

The dependency only goes upward: layers are independent form the layers above them, to avoid looping dependency. However, layers may depend on the ones under them. They benefit from the enriched context and the decisions of these lower layers. As the knowledge is improved with the level of the layer, a layer may preempt or invalidate its lower layers outputted information.

An event is processed by every relevant expert systems in the N-Layer's layers, from the bottom to the top. Each expert system managing the event benefits from the previous ES' processing. As the process occurs, the information is increasingly complete: the enriched context and decisions are expected to be more reliable. Hence the downward preemption.

The resulting layers of ES are organized with increasing complexity and genericity. Each additional expert system can be more high-level than the previous ones, as it can rely on these "lower layers".

The reliability of a whole depends on the reliability of its parts, as well as how the mistakes are carried away. This is why the N-Layer relies on many highly specific, efficient expert systems in the lower layers. Each is reliable in its domain only. Above these, more generic expert system can discriminate between the false positive and false negative, as they dispose of more knowledge. By focusing on small subset of functionalities, the higher layers are of increasingly genericity, while still staying reliable.

At the lowest layers, the components are barely expert systems, and at most RBR with as few rules as possible, to minimize false categorization of the context. Depending on the criticalness and the requirement to distinguish these errors, false positive or false negative are avoided. For example, it may be better to falsely suspect the batteries of a device are dead than to falsely suspect they are full.

Relying on these layers, the next layers consolidate and improve the knowledge from the bottom layers. The ES of these layers, which becomes more and more "real" expert systems[3], perform more complex analysis. They check the lower layers' context for false positive or false negative. They add complementary information. They even provide new context and decisions.

IV. Implementation / Test

An implementation of the N-Layer has been created on the UniversAAL middleware. This "proof of concept" relied on some functionalities of the middleware to ease the development.

A. UniversAAL

UniversAAL [1] is a European Union project to create a reference among the AAL oriented middlewares. Ended in 2014, though still active, UniversAAL (uAAL) is the follow-up of half a dozen previous projects. This modular middleware is a service oriented architecture written in Java and OSGi. It provides miscellaneous functionalities, in regard to security, privacy… as well as various tools to ease application/service development.

In UniversAAL, applications communicate with each other by the means of ontologies, abstracting their APIs. To this end, 3 buses are provided: one dedicated to the user interface (UI bus); one for service request (Service bus), where each request is answered by a response; and one for context exchange (Context bus). To make use of a bus, applications/services must register on it, specifying communication properties, to the middleware. Then, the security modules can check whether the registration is accepted or refused.

UniversAAL's buses filter inputs to and outputs from services. From the context (cf. figure 2), the aggregator does only access information it specifically required. Thus, its job is to sort ambiguous data and to remove sensible part of the messages, on a per case basis. This implementation of the solution, which relies on the application developer to do the right job, is far from perfect. A more concrete solution would be to ask the user whether each registration to a bus is valid or not.

UniversAAL ensures that each context message is transmitted to each interested service. This eases the dependency management. The context is broadcasted to any component authorized to access it, and the services are left to organize themselves. By tagging each response with the set of its related inputs' IDs, it is possible to tackle the problem of the delay management while limiting the additional weight of the message. This relies on a service which provides storage for every context message transmitted.

B. Implementation

The example presented in this section simulates 3 rooms: a storage room; a break room; and a corridor linking the both together. Each room is geared with various sensors which monitors the temperature and the presence of a user. To manage the various reasoners, a 4 layers N-Layer is used.

Both the storage room and the break room have environment control, to increase or decrease the temperature. For each room, when the temperature goes beyond a given range, an anomaly is raised. The storage room's temperature is kept ice-cold. The break room's temperature is luke-warm. Whereas the corridor's temperature is monitored, but left unmanaged.

In addition, the temperature of the room in which the user is is monitored. A too long exposition to "harmful" temperature triggers an anomaly alert.

To manage the user's well-being, and the rooms' temperature, the N-Layer is built of 4 layers of increasing worth. The first layer processes the sensors' context, to try and extract revelant properties. The second layer tries and extracts from this enriched context some obvious anomalies.

[3] compared to the bottom layers' ES Though these are still expert systems, even if they are not as complex as more common cases

The third layer processes the available context to submit mitigation measures. These measures are vetted by the fourth layer. This last layer emits the validated outputs.

The context is then enriched by services and related expert systems. The events ("temperature in storage is too hot", "user is not found", etc.) are expressed with their own ontology, as the message is destined to whichever can read them. Anomalies are expressed using a common ontology, focusing on the system's components and their relations. This implies that each expert system knows their relevant components' relations.

When the application starts, so do the services, which perform their duty. Once initialized, the sensors provide grounds for the context. From there, the N-Layer's layers can perform their work.

Periodically, the sensors update their share of the context. These events may trigger updates in the N-Layer. When an event occurs, the relevant sensors emit new context.

When triggered by the second layer, the anomaly alert reaches the fourth layer with potential, quick solutions. The solutions' discrimination (or combination) is then performed.

The selected action –or lack thereof– triggers new notifications from the sensors. This new step may further improve the situation's knowledge, closing in to its resolution. Or it may start a new situation to resolve.

The presented example, aimed to check the validity of the N-Layer, is minimalist. This is due to the amount of work required to develop various expert systems. Their integration requires either translation modules (at the level of the aggregator and the publisher), or additional fine-tuning.

An improved example is pending. It is expected to be bigger: with many more devices and expert systems. In addition, it'll be independent of the middleware (even though UniversAAL has some interesting features): no functionality of the N-Layer requires specifically UniversAAL. When this example will be finished, it should scale from AAL to smart homes.

V. CONCLUSION

The anomaly management, in the field of ubiquitous intelligence such as smart cities, smart homes… has been focused on increasingly efficient expert systems. However, it is difficult to benefit from these expert systems, as they are hard to manage (design, use and maintain). Our idea is to focus on existing solutions, and to integrate them together, by using a "KISS" (Keep It Simple) approach.

Some of the related problematic still have to be tackled. The various expert systems have to express information in a sensible way for the whole system. This can be done by translating to and from, or even using, a common language, which can be achieved by using ontologies.

The N-Layer relies on many "low level" expert systems, to provide a ground context. On top of these layers, more abstract expert system are deployed. The goal is to ease the expression of high level anomaly management, by ensuring the lower levels exist and obey. This is done through a redundancy tolerant, hierarchically layered expert system.

Off course, the solution is not perfect. Its upward dependency puts constraints on how to express generic solutions. A possible improvement could be to rely on subscriber/publisher, though this trades the dependencies restrictions with the time (and reactivity) management.

REFERENCES

[1] Universaal. http://www.sintef.no/universaal.

[2] Bridget Beamon and Mohan Kumar. HyCoRE: Towards a generalized hierarchical hybrid context reasoning engine, 2010.

[3] Guangyou Cai, Quansong Xie, and Guangming Li. Neural network expert system and its application in command. In *Computer Science and Service System (CSSS), 2011 International Conference on*, pages 1399–1402, June 2011.

[4] Liming Chen, Chris Nugent, and George Okeyo. An ontology-based hybrid approach to activity modeling for smart homes. *IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS*, 44, February 2014.

[5] Xingchao Gong and Xin Guan. Intrusion detection model based on the improved neural network and expert system. In *Electrical Electronics Engineering (EEESYM), 2012 IEEE Symposium on*, pages 191–193, June 2012.

[6] Ting Han, Bo Li, and Limei Xu. A universal fault diagnostic expert system based on bayesian network. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 260–263, Dec 2008.

[7] Xu Li, Rongxing Lu, Xiaohui Liang, Xuemin Shen, Jiming Cheng, and Lin Xiaodong. Smart community: an internet of things application. *IEEE Communications Magazine*, November 2011.

[8] Hongbo Ni, Xingshe Zhou, Daqing Zhang, and Kejian Miao. Discovering inhabitant's context-dependent task with case-based reasoning. In *Future Generation Communication and Networking, 2008. FGCN '08. Second International Conference on*, volume 2, pages 453–456, Dec 2008.

[9] N. Noury, G. Virone, and T. Creuzet. The health integrated smart home information system (his2): rules based system for the localization of a human. In *Microtechnologies in Medicine amp; Biology 2nd Annual International IEEE-EMB Special Topic Conference on*, pages 318–321, 2002.

[10] Ioannis Panagiotopoulos, Lambrini Seremeti, and Achilles Kameas. A policy enforcement framework for ubiquitous computing applications. *Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering*, 2011.

[11] Fabrício Henrique Rodrigues, Cecília Dias Flores, and Liane Nanci Rotta. An ontology design pattern to represent universal relationships. *Proceedings of the 2015 IEEE* 9^{th} *International Conference on Semantic Computing*, 2015.

[12] Ted Selker. Understanding considerate systems - UCS (pronounced: You see us). 2010.

[13] Yang Shunkun. Bayesian network based software diagnosis expert system. In *Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on*, pages 189–192, Jan 2012.

[14] T. Van Nguyen, Yi Chang Woo, and Deokjai Choi. Ccbr: Chaining case based reasoning in context-aware smart home. In *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, pages 453–458, April 2009.

[15] Allan Venceslau, Raphaela Lima, Luiz Affonso Guedes, and Silva Ivanovitch. Ontology for computer-aided fault tree synthesis. *IEEE Emerging Technology and Factory Automation*, 2014.

[16] B. Walek, J. Zacek, M. Janosek, and R. Farana. Adaptive fuzzy control of thermal comfort in smart houses. In *Control Conference (ICCC), 2014 15th International Carpathian*, pages 675–678, May 2014.

[17] Yingying Wang, Ming Chang, Hongwei Chen, Yueou Ren, and Qiuju Li. Research on fault diagnosis expert system fusing the neural network knowledge. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2011 International Conference on*, volume 1, pages 196–200, Aug 2011.