Toward a Behavioral Decomposition for Context-awareness and Continuity of Services

Nicolas Ferry and Stéphane Lavirotte and Jean-Yves Tigli and Gaëtan Rey and Michel Riveill

Abstract Many adaptative context-aware middleware exist and most of them rely on so-called vertical architectures that offer a functional decomposition for context-awareness. This architecture has a weak point: it does not allow the system handling both dynamics of the changing environment and applications. To avoid this, we propose an approach for context-awareness based on a behavioral decomposition, and because each behavior must complete all functionalities necessary for context-awareness, we introduce an hybrid decomposition. It consists in a functional decomposition into a behavioral decomposition. This approach derives benefits from both decomposition, first allowing to handle environment and application's dynamics, second introducing reusability and modularity into behaviors.

1 Introduction

Nowadays, with the miniaturization of computer hardware, many objects with computational capabilities are dissolving in our daily lives. Thus, the idea of personal computer as the single smart object or as a universal digital assistant is fading away.

Nicolas Ferry
I3S (UNS - CNRS) and CSTB, 290 route des Lucioles - BP209 06904 Sophia-Antipolis France, e-mail: nicolas.ferry@unice.fr

Stéphane Lavirotte
I3S (UNS - CNRS), 930 Route des Colles - BP 145 06901 Sophia-Antipolis France, e-mail: stephane.lavirotte@unice.fr

Jean-Yves Tigli
I3S (UNS - CNRS), e-mail: tigli@polytech.unice.fr

Gaëtan Rey
I3S (UNS - CNRS), e-mail: gaetan.rey@unice.fr

Michel Riveill
I3S (UNS - CNRS), e-mail: riveill@unice.fr

Ambient systems consist of two categories of entities: (1) living entities, as the user, and (2) systems with computational capabilities. A system relies on a hardware infrastructure which can provide a software infrastructure. Under these constraints, the editable software part of the system allows to implement new functionalities for new applications.

Ambient systems are characterized by using devices and objects of everyday life, and can take many forms. The technological heterogeneity of those devices and objects is still the biggest challenge to overcome in order to enable them to interact with the system. Moreover, we must consider the semantic heterogeneity of these entities that can be introduced into ambient system. Another major feature of ambient system is the high variability of software infrastructure. These infrastructures evolve dynamically led by appearances and disappearances of objects or devices. The topologies of this infrastructure are dynamic due to arbitrary node mobility, failures or energy saving.

Because environment's nature is highly variable, even the corresponding software infrastructure, pervasive systems have to handle those variations and offer such dynamicity. These systems have to be able to adapt to their context; they must be context-aware. And adaptations to these changes have to be done quickly to follow those variations but not to often to preserve the application's stability and not to divert the user.

2 Related works: Context and context-awareness

Generally and specially in the field of ambient computing, the context appears as the supplement of all editable software entities. So this supplement can be:

- All hardware or software infrastructures,
- The environment,
- All the entities without any computational capabilities like users or raw objects.

2.1 Context-awareness

Classical mechanisms for context-awareness rely on a functional decomposition to provide reusability and evolution facilities. This decomposition is usually based on key functionalities [6]. The first stage of sensing is to gather contextual information also called observables. These observables are then transformed as symbolic observables about the state of the environment, for example using ontologies.

Middleware such as the Context Toolkit [9], SOCAM [11], Contextors [14], providing mechanism to collect, store and sometimes process these observables, have emerged. They offer some mechanisms to discover heterogeneous entities, but also for deduction and filtering of data collected using some centralized mechanisms.

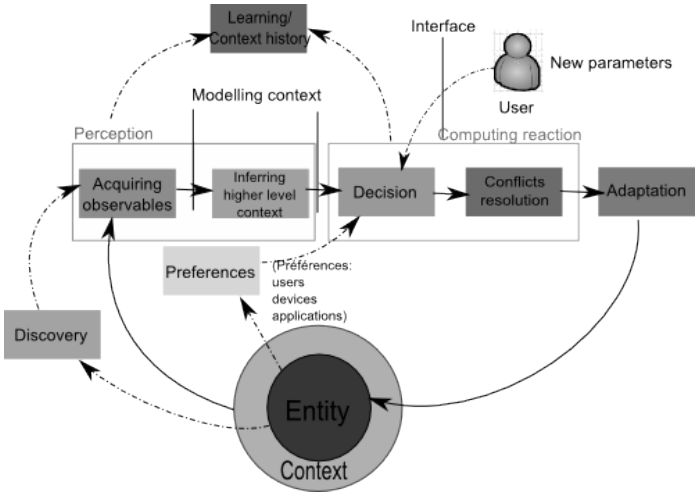

Fig. 1 Functional decomposition of context-aware mechanism

However, some middleware as COWSAmI [1] or Construct [7] are based on a decentralized architecture.

Context observation is a sub-problem of context-awareness. Indeed, as shown in SAFRAN [8], application's behavior adaption is also a crosscutting concern and a part of context-awareness mechanism [3]. Hence, as a result of contextual information collection, data will be used during a decision stage, also called situation identification stage. It will produce an action plan which will be implemented by the adaptation mechanism also called the control mechanism. Of course, these steps can be refined as shown in Fig 1.

2.2 Limitations

The functional decomposition often results in the development of so called "vertical" architecture, consisting in a set of various layered functionalities.

Most of context-aware middleware are based on those architectures. They have a weak point: the data centralization in at least one of the functionality of the decomposition. This means that they rely on a common representation or a global model of context. For example, in Gaia [15], a context-aware middleware, observables are stored in a single entity: the "context file system". We found the same problem with SOCAM [11] or CARMEN [2]. For example, CARMEN uses a specialized LDAP containing user profiles for the decision stage. However, centralizing data remains hardly imaginable in an ambient system where you have to manage a throng of heterogeneous entities and data. In fact the environment is not known a priori and is constantly evolving, so considering the whole environment is similar to considering

the universe, which is not possible. A centralized approach involving potentially a throng of information then comes with a bottleneck of performance and does not allow the system to scale.

But these architectures for context-awareness have a central role in an ambient system since they interact with both environment and applications. Then, they should respect application and environment's dynamics. So reactivity is a key concern of ambient computing, both for adaptation triggering and for adaptation time. But reactivity defined as follows can hardly be proposed by middleware using a vertical architecture.

We consider that adaptive applications are always in one of the three states presented in Fig 2. States (1) and (3) are normal execution states of the application, where it is consistent with its environment. It means that the application's behavior is based on what is relevant in its environment and this is the expected behavior for a particular situation. During the transitional state (2) the application is in its adaptation phase and unavailable. It is considered in an inconsistent state because the application is not in line with its environment.

Fig. 2 The three states of an adaptive application

Consequently, Adaptation's dynamic has to be consistent with the dynamic of the changing environment [10]. In other words, it is essential that:

- The system is not unavailable (2) for too long while adapting, so adaptation has to be as fast as possible in order to obtain a consistent application (3). Otherwise, the system could become unstable and may never reach a normal execution state before new evolutions occur in its environment.
- The system does not stay in the previous state (1) too much time before reacting to environment changes.

Moreover, adaptation's dynamic has to be consistent with application and/or user operation (application's dynamic). In other words, it is essential that:

- The system does not go too frequently from state (1) to (2) and produce an unstable and inconsistent application.
- Too many lag can disrupt and divert the user from the system [12].

Vertical architectures does not allow handling all the dynamics described above. Indeed system reactions to change in context depend on the reactivity of each part of treatment (each layer). Changes are all evaluated simultaneously using the same processed data. This leads to a system with a fixed dynamic.

Therefore, it appears that the complexity of processing and representation of contextual information impact the dynamic of the mechanism of context-awareness. Then it is necessary to study another kind of decomposition to handle these dynamics. This line of study is not unlike the general approach to problem solving by

a human operator. Indeed the Rasmussen's model [13] called SSRK defines three levels of cognitive processes:

- Skill Based: the skill level corresponds to actions that are done automatically.
- Rule Based: the rules' level corresponds to procedures (algorithmic) implemented by human to define its action.
- Knowledge Based: the level of declarative knowledge implements a process that requires many cognitive resources and so that take time to make a decision and implement its action.

We then see that it is a decomposition of various processes with their own dynamic working separately.

3 Reactivity and behavioral decomposition of context-awareness mechanism for application's adaption

Similar architectures exist in the field of robotics, where they are called "horizontal" architecture. The aim of this approach is to specialize a generic minimal core of an application with some specific modules called horizontal layers [16]. These layers are independent of each other and work in parallel. Each layer is connected to the world via a set of sensors and can act through actuators. This kind of architecture introduces a new type of decomposition: the *behavioral decomposition*.

According to Bryson [5], the behavioral decomposition is an architectural approach that decomposes the intelligence in terms of behavior such as eating or walking, rather than generic processes like planning or observing. A behavior corresponds to an horizontal layer and activities may consist of sets of managed behavior. A major contribution of these approaches is not to see the system as a sequence of processes and thus functionality, but rather as a parallelization of processes which can produce a coherent activity. More complex behaviors are achieved with simple ones like for divide and conquer strategies. These architectures comply to the following characteristics:

- Each layer captures relevant information from the environment, sensing the surrounding to a degree sufficient to achieve the necessary [4].
- There is no need for a representation of the environment in a decentralized approach: "The world is its own best model" [4]. The only accurate environmental data are obtained immediately by the sensors.
- They are based on many small behaviors of low complexity for the best possible reactivity.

Horizontal architectures need a coordination mechanism to combine the output data of each layer in order to obtain a rational and coherent global behavior. A simpler mechanism induces more reactivity.

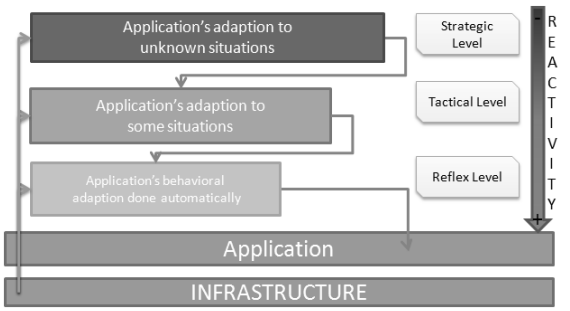

Fig. 3 The proposed CONTINUUM architecture. This architecture relies on three levels. Levels closest to the infrastructure are the most reactive. These levels are similar to the three levels of SSRK. The level N is based on the mechanisms offered by the level N-1 and may act on it.

In the field of context-awareness, this behavioral decomposition relies on autonomous and separated processes of context-awareness. They produce adequate changes in the application from relevant gathered information in order to provide a coherent global behavior to the application. We call such processes a *basic adaption to context behavior* (BACB).

This kind of architectures enables a better control of environment and application's dynamics. Indeed, treatment directed by a BACB is made only when necessary. Moreover, it is as fast as possible since each BACB computes only what is relevant to them. In fact, because these BACB are independent of each other, they have their own dynamic which is not anymore dependent of other treatments. Then we are able to write BACB with various levels of complexity to best fit the dynamics imposed by the environment and the application. Indeed, we will be able to study for each BACB their dynamics (time of response) and their compatibility with the evolving environment (not enough or too reactive). On the other hand, we can also study interactions between these behaviors and their good management. Each BACB is scheduled with a late manager, it may be a mechanism to resolve adaptation conflicts.

In the CONTINUUM French national project, we propose a context-aware mechanism for continuity of services in ambient systems, see Fig 3. This architecture is based on a behavioral decomposition. And each level has its own dynamic according to their priority and the complexity of their process.

4 Towards an hybrid approach

As we saw earlier, a BACB gather some contextual information and act on it in return. A behavior identifies an action to perform according to a relevant context, so it must respect the classical steps of the functional decomposition: perception, decision, and reaction. Such a BACB can then be decomposed into several functionalities. So we introduce an hybrid approach consisting of a functional decomposition into a behavioral decomposition. A behavior is itself composed of a set of functionalities that, applied to context-awareness, spreads from perception to reaction.

The hybrid approach derives some benefits from both decomposition, first to control its dynamics and independence thanks to behavioral decomposition, secondly improving reusability and modularity into behaviors thanks to functional decomposition. More precisely, hybrid decomposition offers, compared to behavioral decomposition, modularity into behavior and therefore maintenance facilities and thus evolving facilities. Moreover it induces a better separation of concerns into behaviors. The modularity introduced by functional decomposition could enable adaption of behaviors. For example, in the CONTINUUM project, a scenario applied to the hydrant man job defines a set of services to help them. One part of the job is to stop water valves when necessary. In this project a service of the system indicates, in the car, the localization of those valves. According to the brightness, these information are visual or voiced. Moreover, the system help them to manage their intervention (description, ...) from their car. This is not allowed when the car moves too fast.

In this small part of the scenario, we can define three behaviors: (1) visual or (2) voiced indications of the valve's localization, (3) intervention management. We can see that those three behaviors must respect various dynamics. So, blocking the use of the help system is critical, it has to be done quickly (environment's dynamic). Otherwise changing the kind of interaction with the localization service must be done more slowly not to pass from an interaction to another incessantly (application's dynamic). Here appears the advantages of the behavioral decomposition. On the other hand, the perception module of behavior (1) and (2) can be reused, thanks to the functional decomposition included in the hybrid decomposition.

Then it appears that this approach requires an inversion in design methodology of mechanisms provided by a middleware. Indeed, it is not to decompose behaviors in a set of features but to write a set of behavior from reusable features. Thus, the major stages of designing an hybrid architecture are: (1) identifying behaviors, (2) specifying their inputs and outputs, (3) identifying features, (4) specifying the feature chaining, (5) implementation. When creating behavioral architectures, specification of inputs / outputs is particularly important. Unlike in classical functional decomposition, aggregation of data is not given as a pre-requisite of the system (hence the need for data representation in functional approaches) but between behavior's outputs.

5 Conclusion

Many context-aware middleware exist and most of them rely on vertical architectures that offer a functional decomposition for context-awareness. This architecture has a weak point: it does not allow the system to handle both dynamics of the changing environment and applications. Handling these dynamics is a key concerns in the field of ambient intelligence since ambient systems must consider a throng of informations and devices. To avoid this, we proposed in this paper to adapt an approach from robotics called behavioral decomposition to context-awareness. To do this, we introduced the concept of basic adaptation to context behavior (BACB). Because

each BACB relies on the classical steps found in the functional decomposition from perception to reaction, we introduce an hybrid decomposition. It consists in a functional decomposition into a behavioral decomposition. This approach derives benefits from both decomposition, allowing to handle environment and application's dynamics, and introducing reusability and modularity into behaviors.

Acknowledgements This work is part of the Continuum Project (French National Research Agency) *ANR-08-VERS-005*.

References

1. Athanasopoulos, D., Zarras, A., Issarny, V., Pitoura, E., Vassiliadis, P.: CoWSAMI: Interface-aware context gathering in ambient intelligence environments. Pervasive and Mobile Computing **4**(3), 360–389 (2008)
2. Bellavista, P., Corradi, A., Montanari, R., Stefanelli, C.: Context-aware middleware for resource management in the wireless Internet. Software Engineering, IEEE Transactions on **29**(12), 1086–1099 (2003)
3. Bottaro, A., Bourcier, J., Escoffier, C., Lalanda, P.: Context-aware service composition in a home control gateway. International Conference on Pervasive Services **0**, 223–231 (2007). DOI http://doi.ieeecomputersociety.org/10.1109/PERSER.2007.4283920
4. Brooks, R.: Elephants Don't Play Chess. Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back pp. 3–15 (1991)
5. Bryson, J.: Intelligence by design: Principles of modularity and coordination for engineering complex adaptive agents. Ph.D. thesis (2001)
6. Coutaz, J., Crowley, J., Dobson, S., Garlan, D.: Context is key. Communications of the ACM **48**, 49–53 (2005)
7. Coyle, L., Neely, S., Stevenson, G., Sullivan, M., Dobson, S., Nixon, P., Rey, G.: Sensor fusion-based middleware for smart homes. International Journal of Assistive Robotics and Mechatronics **8**(2), 53–60 (2007)
8. David, P., Ledoux, T.: Towards a framework for self-adaptive component-based applications. Lecture Notes in Computer Science pp. 1–14 (2003)
9. Dey, A., Salber, D., Futakawa, M., Abowd, G.: An architecture to support context-aware applications. submitted to UIST (99)
10. Ferry, N., Hourdin, V., Lavirotte, S., Rey, G., Tigli, J.Y., Riveill, M.: Models at Runtime: Service for Device Composition and Adaptation. In: MRT'09, p. 10 (2009)
11. Gu, T., Pung, H., Zhang, D.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications **28**, 1–18 (2005)
12. MacKenzie, I., Ware, C.: Lag as a determinant of human performance in interactive systems. In: Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems, pp. 488–493. ACM New York, NY, USA (1993)
13. Rasmussen, J.: Information processing and human-machine interaction: An approach to cognitive engineering. Elsevier Science Inc.,NY, USA (1986)
14. Rey, G., Coutaz, J.: The Contextor Infrastructure for Context-Aware Computing. In: ECOOP 04, Workshop on Component-Oriented Approach to Context-Aware Systems. Citeseer (2004)
15. Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R., Nahrstedt, K.: Gaia: a middleware platform for active spaces. ACM SIGMOBILE Mobile Computing and Communications Review (2002)
16. Zhang, C., Jacobsen, H.: Resolving feature convolution in middleware systems. ACM SIGPLAN Notices (2004)