Towards a Functional Control of a Context-Aware Systems

Meriem Zaiter[1], Salima Hacini[2], Zizette Boufaida[2]
Larbi Ben M'hidi University of Oum El-Bouaghi
LIRE Laboratory at Mentouri University of Constantine Algeria
[2]Mentouri University of Constantine
LIRE Laboratory, Algeria
m.zaiter@univ-oeb.dz ,{meriem.zaiter, Salimahacini, zboufaida}@gmail.com

ABSTRACT: *Distributed systems, can have a large number of independent hardware components that cooperate and collaborate to achieve a given objective. However, the assurance of the reliability is not evident because each of these components may cease to operate at any time, which result probably error and a failure of the whole system. The goal of this paper is to propose a way to control the continuous operating of a distributed system via a contextaware mechanism.*

Keywords: Context, Context Aware, Fault, Distributed System, Reliability

Received: 12 November 2011, Revised 8 December 2011, Accepted 14 December 2011

1. Introduction

In the different areas of computing science, reliability is the most required propriety for the availability of systems. For example, the reliable industrial system is a system that increases the productivity of an enterprise and resulting an increased profit. Moreover, in critical systems ensuring a zero risk is indispensable for preventing the catastrophic consequence in the case of the occurrence of fault. So, the guarantee of such propriety in a given system and in particular on context-aware system is a real challenge. The reason is that this type of system is characterized by a set of information that can be removed or inserted implicitly or explicitly. Therefore, the occurrence of a fault in the system leads to a difficulty to clarify and understanding the context. To overcome this problem, this paper focuses, as a first step, on the notion of context that is largely studied in the literature [1-8], and also on the control of system to ensure the detection of fault before its extensions. This detection enables the insurance of a good management in the case of a fail in a context-aware system.

This paper is organized as follows: next section presents the state of the art on the notion of context. Section III presents the concepts used by our fault detection mechanism. This section is concluded with a summary table setting out the categories of fault and its various sources. Section IV describes the architecture of the proposed fault detection mechanism that is based mainly on a set of local controllers and one global controller. Section V details the operational task of the local controller. Section VI specifies the operations performed by the global controller. Finally, a conclusion and perspectives achieve this paper.

2. The Notion of Context and its Categories

The context is a concept that has been discussed by several researchers. The first sub-section presents and discusses the different definitions in the literature while the second subsection provides an overview of its different categories.

2.1 Context definitions

Schilit and Theimer [1] are talking about the context to the location, identity of people and nearby objects, and changes on these objects. In their definition, Brown et al. [9] explain the context around the place, the identity of people surrounding the user, the time of day, season, and temperature. Ryan et al. [10] define context as well as the location of the user, environment, identity and time. Dey [11] described the context as, the emotional state of the user, the center of attention, the location and orientation, date and time, objects and the persons located in the same environment of the user. Other definitions provide synonyms for context, as well as the environment or the situation. Some consider the context associated to the user's environment, while others see it as the application's environment. For their part, Franklin and Flaschbart [2] consider the situation of the user.

In the field of intelligent ambient the context can be identified by an exhaustive list of parameters collected in a box and the reasoning on this box takes into account only the values of the parameters whose are inside the box, disregarding the rest. [12]. For Gaetan [13], the context at given time t is the accumulation of the situations between t0 and t for the achievement of the user task. While Brezillon sees the context as: an evolving concept, a dynamic process in a dynamic environment and not as a set of states [14]. On the other hand, in multi-agent systems, Bucur [15] defined context as a pair consisting of a purpose, and a set of factors (contextual information) called contextual attributes to satisfy a given goal. More recently, In Markus et al.' work [7], the authors consider that context can be subdivided into user, system and environmental context. The definition of context proposed by Strassner and his colleagues [8] confirms that the context of an entity is a collection of actions and knowledge deduced which describe the state and the environment in which an entity exists or will be existed. We note also that, in Kramer and his colleagues research [5] they divided the context of the user into personal, local and service context…

We observe in this set of definitions, the implication of the concept of parameters and aspects that can clarify the context in more explicit manner.

Detailing the definitions presented by Schilit et al. [16], Dey et al. [17] and Pascoe [18], we note that they are the nearest in spirit of the operational definition by mentioning the dynamic of the context in the system. Schilit et al. describe important aspects of context as: where are you, who are you and what resources are nearby? They define the context in terms of the execution environment which is constantly evolving. They consider the following elements of environment:

• The available computer components: processors, network capacity, connectivity, cost calculation, and devices accessible by the user (input and display).

• The location of the user in the environment, the collection of the user neighbors, and social status (as friend in social network, group favorites...).

• Lighting of the physical environment and the noise level (for the hardware side).

We note that these definitions are specific to the applications based on physical devices.

On the other hand, Dey and Abowd [19] define context as any information that can be used to characterize the situation of an entity: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".*

So the context is dynamic, and its values may change from one period of time to another, and even during the same period itself. Therefore, if an information can be used to characterize the situation of a participant in an interaction, then this information is part of the context.

According to the definitions presented above, we note that they vary depending on the context and the nature of the future use of the context. In addition, all these definitions use the context to solve a given problem related to an entity.

The definition proposed by Dey and Abowd in [19] appears the most interesting one because it is an abstraction of the types of entities involved in an application. Thus, our definition is inspired from that of Dey and Abowd. We add another abstraction level on the interaction of objects since the user is not the only one which interacts with the application. In other words, even components of an application can communicate with each other. So our context definition becomes*: "Context is any internal or external information, related to an entity, could be used and have an impact on the future state of the application. This information can be linked to one or more entities. The latter, regardless of their nature (hardware / software / human), can trigger events that affect the global state system. To this end, the occurrence of a fault causes certainly an immediate dysfunction to the global system».*

2.2 Context categories

Different categories have been introduced to characterize context-aware information. Schilit et al. [16] have identified four categories of context-aware applications that vary along two axes: if the task of the application is to collect information or to perform an action (command); or if a given task will be executed, manually or automatically, due to a user or a component request, on which the manual manner is employed when the relevant elements of context are easily specified. It is also called the proximity selection applications. This technique is based on the interaction between a set of system components. However, the automatic one creates an automatic link to the available components in the current context [19] it is also called automatic contextual reconfiguration. As an example of the automatic information application we can give the touristic guide like Campus Aware [20].

Moreover, the applications that execute commands for components manually depending on the available context are classified as contextual commands applications. In addition, the applications executing commands automatically for a component based on the available context consist to generate actions guided, of course, by the context. On which those actions are based on the rule if, then.

On the other hand, Brown et al. [21] divide these applications into two groups, continuous applications and discrete one; in a continuing application, the information delivered is constantly changing with the context. While in a discrete application the information provided is organized into several segments, and for each segment a context is associated context. So, when the context is detected, the information is delivered. In this case we no longer consider here whether the information provided has been requested explicitly or not (as in the categories shown above), but only if that information is continuously available or not.

Our context definition, presented in the end of the previous sub-section indicates that the context is specified according to its elements and its future use. We complete this definition by adding that the context depends also on how these elements are presented and managed.

We specify that the characteristics of a context are used in the presence of a context-aware application in order to recover the various errors that may arise during its execution. At this point it seems relevant to note that a fault is the hypothesized cause of an error. And an error is the manifestation of a fault in the system, which will certainly cause a failure of the global system.

The next point specifies the needed concept for our functional control mechanism.

3. The Used Concepts

Before this section aims to enumerate and define the concepts used to determine the context and the fault detection system stem from a context-aware application. In our study, we selected a set of concepts:

• The global system: represents the physical system components that cooperate and collaborate to achieve a given objective. These elements can be summarized in: entity, the position and the temporal aspect.

• The entity that is noted (ei) represents an element belonging to the global system. (ei) is any element: hardware, software or human (user) that can be a part of the system.

Locality and the temporal aspect concern, for their part, respectively the characteristics of the temporal and the geographic location of entities belonging to the global system (neighboring entities, synchronization between entities ... etc).

Moreover, to elaborate our fault detection mechanism we must:

1) Define the elements of functional dependency denoted D (ei) representing a subsystem of the global system.
This subsystem contains one or more entities which are functionally dependent on the entity (ei).

2) Develop a local controller noted (LCi). It is associated with every entity (ei) and it is responsible for processing events emitted to the entity (ei). An event can be:

a) External: it concerns an event or request to the entity (ei) associated with the controller (LCi);

b) Internal: it concerns all communication between the controller (LCi) and the entity (ei) itself.

We can give at this stage our formal definition to put in a nutshell our context definition; so the context noted “C” can be defined by the set:

$$C = \{I, D_i, E\}$$

Where:
I: is the set of information (state, services ...) characterizing an entity (ei) belonging to the global system

Di: is the set of entities on which the entity (ei) depends functionally.

E: is the set of events that may occur in the system.

Thus, each controller (LC_i) has a dual role: receiving external events and carrying out the periodical control (local test) of the functioning of the entity (ei). The role of (LCi) will be detailed at Sectiond V.

Make available a global controller noted (Gc) which deals with the detection of a dysfunction of an entity ei of the global system (see Section VI).

Based on the definition of context proposed at the end of Section II, the table below (see Table 1) specifies, the element of the context, the source of the fault and its category.

Elements of Context		**Categories of faults**	
		external faults	***internal faults***
Entity	Hardware/user	An error of interaction (such as error identification, or an input mistake...)	/
	hardware	/	Error referencing of an internal component (processor, memory ...) - internal hardware failure
	Software/user	An entry outside the domain specification of the application	/
	Software	/	Design fault in the application itself
Temporal aspects (date, time)		Fault in scheduling and synchronization of messages among system components	the local Clock is not synchronized - physical error due to a transmission problem
Location		Localization problem of neighboring entities (effect of the environment)	Fault in the physical controller *s* of component

Table 1. Fault's Source and Category

4. The Fault Detection Mechanism

The proposed fault detection mechanism concerns the context-aware applications and it is based on the already presented concepts. It aims to set the global controller, the local controller, the entities and functional dependencies of these entities in order to locate the fault, so:

1) Each entity has its own local controller which controls its behavior. It :

- Send and / or treat requests
- Manage all the anomalies presented to or from its entity (see Section V).

2) All local controllers are related to the global controller.

The functional architecture showed in "Figure 1", summarizes the failure detection process.

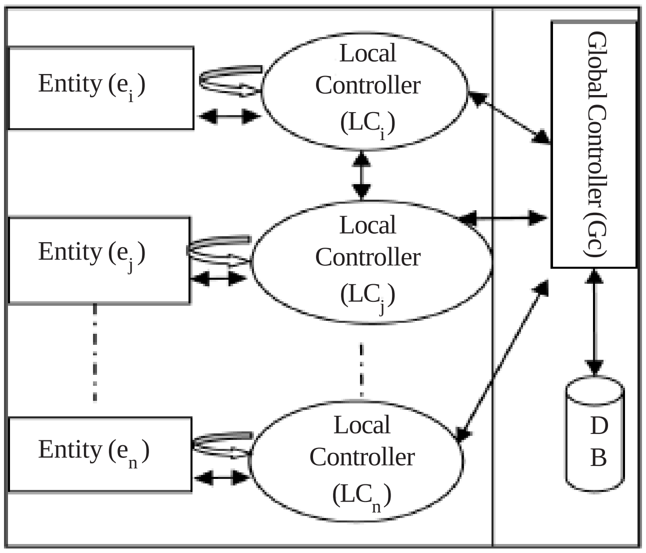
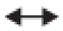

Entity (e_i)
Local Controller (LC_i)
Entity (e_j)
Local Controller (LC_j)
Entity (e_n)
Local Controller (LC_n)
Global Controller (Gc)
D B

Figure 1. The Global system functional architecture

: Test local

 : Events/ request

BD: a data base that contains the information about each entity ei (state, set of D (ei)…).

The two main tasks in this detection mechanism are endorsed by the local controller and the global controller. They are described in the following sections.

5. The Local Controller

A local controller module (LCi) is attributed to each entity ei.

5.1 The Local Controller activities

Each local controller carries out the following activities:

1) It receives the external events (the requests) arriving to the entity ei. In this case an external event can be:

- a simple request from an entity ej;
- a negative feedback from an entity ej that is resulting from an eventual previous interaction with the entity ei;
- An inquiry concerning the entity ei, or an information failure of an entity ej if ei depends functionally from the defective entity ej (this event is raised by the global controller).

2) It indicates the failure of the entity ei due to:

- a negative feedback in the case of an incorrect result or a signal of no-response (ei) by the controller LCj of an entity ej (after a checking) or;
- a no-response to the periodical test of inspection performed by the controller LCi itself, so the controller can signal a failure of ei.
- a no-response from another entity ej in case of its failure.

```
Input: event;
Begin
Repeat
  case (event) of:
    RS(ej ,ei) :
        save (RS(ei,ej)) ;
        Treat (RS(ei,ej)) ;
        send (RP(ei)) ;
    fault (ej ,Gc) :
        If (Verify_Rq (ei, ej)) then
                Cancel (RS(ei,ej))
                search (E,R)
        else
                state (ej) ← bad
        End if
    alert (LCj, ei):
        if (state (ei)=bad) then
                send alert (LCi ,ei) to Gc
                send alert(LCi ,ei) to LCj
        else
        if ((check_ local _ state (ei,0))=0) then
                send alert(LCi ,ei) to Gc
                send alert(LCi ,ei) to LCj
        else
                send good_state(LCi ,ei) to LCj
                send good_state(LCi ,ei) to Gc
        End if
        End if
    RS(ei, ej) :
        save (RS(ei ,ej)) ;
        wait (RP(ej));
    good_state(LCj, ej) and not RP(ej):
        send RS(ei,ej)
    good_state(Gc, ej) and not RP(ej):
        search (E, R)
untile (false)
End;
```

Figure 2. The local controller (LCi) Algorithm

5.2 The local controller task

At the reception of an external message the local controller processes it according to its objective by the extraction of the event related to the interaction. The handling of the external events is given by the algorithm presented in “Figure 2”:

- **Terminology**

RS (ei, ej): it is a simple request send by ei to ej. It denotes a normal case.

RP(ei): it is an answer for a simple request sent by the entity (ei).

Fault (ej, Gc): it indicates a failure of an entity ej reported by the global controller.

Search (E, R): this function returns the set of entities that provide a service R.

Rep (ei): this is a boolean function. It represents the answer or not of ei to the local test triggered by the local controller LCi

Check_ local _state (ei, t): it is a function that represents the local test triggered after a **time t**. This function has a value 0 if the entity ei did not answer, and 1 otherwise see “Figure 3”.

Time: it represents the duration between two periodical tests.

Alert (LCi , ei): It denotes a failure of an entity ei reported by the controller LCi .

good_state (LCi , ei): it is an event emanated from the controller LCi. It shows that its entity ei is in a good state.

good_state (Gc, ej) it is an event emanated from the global controller. It shows that the entity ej is in correct state.

Verify_Rq (ei, ej): the role of this function is to check if a request emitted by entity ei is being processed by an entity ej or not.

Save (RS(ei ,ej)) **: its role is to save the request (sends by the entity (ei)) that will be processed by the entity (ej)**

Wait (RP(ej)): it aims to start the control of the duration of the response of an entity ej to the ei request’s.

```
Input: entity ei;
Begin
   if (not (Rep (ei))) then
      state (ei) ← bad
      send alert(LCi ,ei) to Gc
      t ← time
      Return (0)
   else
      state (ei) ← good
      t ← time
      Return (1)
   End if
End
```

Figure 3. The function check_ local _state (ei , t)

6. The Global Controller

The role of the global controller (Gc) is to supervise the global system and ensure the management of the fault when receiving a signal failure.

6.1 The Global Controller activities:

The global controller manages the following events:

1) Upon receipt of a fault signal of dysfunction of an entity ei from its controller LCi, it will inform the set D (ei) to prevent its expansion.

2) It is responsible for the decision regarding the fault of no-response type ei declared by another controller LCj. In this case, the global controller validates the fault if it has already received a warning from LCi; otherwise it asks LCi to estimate the state of the entity ei.

We note that the signal of a failure of an entity ei by a controller LCi consists to send an alert message to the global controller. In the case of a declaration of fault reported by a controller LCi, which involves another entity ej, another alert will be sent to the controller LCj. This is the case of noresponse presented above.

6.2 The Global Controller task

As the local controller the global controller treat the received event by executing the following algorithm in “Figure 4”:

```
Input:  event
Begin
Repeat
        case (event) of:
        nw_elt:
                Define the set D (ei)
                Creat the controller LCi
                Load _DB (ei)
        alert (LCi , ei) :
                State( ei ) ⟵ bad
                Send fault (ei, Gc) to D (ei)
        alert (LCi , ej) or good_state(LCj ,ej) :
                Check_state (ej)
Until (false)
End
```

Figure 4. The global controller

```
Input : entity : ej
Output: state of ej
Begin
        State ej ⟵ Extract_state (ej);
        if (state ej =bad) then
        send fault (ej,Gc) to LCi
        else
        send Rv (Gc,ej)
        if (Rep (ej)) then
                send good_state (Gc, ej) to LCi
        else
                state ej ⟵ bad,
                send fault (ej, Gc) to D(ej)
        End if
        End if
```

Figure 5. The Function: check_state (ej)

• **Terminology**

nw_elt: means that a new entity has connected to the global system.

Load _DB (ei): load an entry in the Data Base for saving the information of the entity ei (state (ei), D(ei)…).

Extract_state (ej): this function retrieves from the data base of the global controller the state of the entity ej.

Rv (Gc,ej): it is a verification request sent by the global controller Gc to ej; The objective is the test of the state of the entity ej.

7. Conclusion

In this paper we have considered that the failure of a context-aware system is due to an internal or external fault where an internal fault is related to its internal state and the external one is caused by an interaction with the environment or an incorrect input. To this end, the fault will certainly affect the communication system components and can even produce a cascading effect on all entities of the system that will generate a failure of the global system. As a first step, we developed a mechanism that can control and detect the fault before its extension. We have also provided a formal definition of the notion of context, and have presented the faults categories related to the different elements of context. As an extension of this work, we plan to (i) build a model of faults, (ii) to establish all the functional entities dependencies in a given system and (iii) to develop a mechanism which enables the guarantee of the reliability of the global system.

References

[1] Schilit, B. N., Theimer, M. M. September/October (1994). Disseminating active map information to mobile hosts, *IEEE Network*, 8 (5) 22-32.

[2] Franklin, D., Flaschbart, J. (1998). All Gadget and No Representation Makes Jack a Dull Environment. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02, 155-160.

[3] Mari Korkea-aho: Apr.(2000). Context-Aware Applications Survey, Internetworking Seminar.

[4] Dey, A. K. (2001). Understanding and Using Context. Personal and Ubiquitous Computing Journal 5(1) 4–7 .

[5] Ronny Kramer, Marko Modsching, Joerg Schulze, Klaus ten Hagen, (2005). Context-Aware Adaptation in a Mobile Tour Guide CONTEXT 2005, LNAI 3554, p. 210- 224. Springer-Verlag Berlin Heidelberg.

[6] Seon-Ho Park, Young-Ju Han, Tai-Myoung Chung, (2006). Context-Role Based Access Control for Context- Aware Application; HPCC 2006, LNCS 4208, p. 572– 580. Springer-Verlag Berlin Heidelberg.

[7] Markus Ablaßmeier, Tony Poitschke, Stefan Reifinger, Gerhard Rigoll, (2007). *C*ontext-Aware Information Agents for the Automotive Domain Using Bayesian Networks; Human Interface, Part I, HCII, LNCS 4557, p. 561–570. Springer-Verlag Berlin Heidelberg.

[8] Strassner, J., Liu, Y., Jiang, M., Zhang, J., Van der Meer, S., Foghlu, M., Fahy, C., Donnelly, W. (April 2008). Modeling context for autonomic networking. MUCS 2008 p. 10.

[9] Brown, P. J., Bovey, J. D., Chen, X., october (1997). Context-aware applications: from the laboratory to the *marketplace, IEEE Personal Communications*, 4 (5) 58-64..

[10] Ryan, N., Pascoe, J., Morse, D. (1997). Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. Gaffney, V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology .

[11] Dey, A. K., (1998). Context-Aware Computing: The CyberDesk Project. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02, 51- 54.

[12] McCarthy, J. (1993). Notes on Formalizing Context. *In:* Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France, p. 555–560.

[13] Gaetan, R. (2001) Systemes interactifs sensibles au contexte. Unpublished doctoral dissertation, Polytechnic National Institute of Grenoble ENSIMAG, France.

[14] Brezillon P. (2003). Representation de pratiques dans le formalisme des graphes contextuels. *In*: Bastien, J.M.C.

(ed.) Actes des Deuxemes Journees d'etude en Psychologie Ergonomique.

[15] Bucur, O., Beaune, P., Boissier, O. (2005). Representing Context in an Agent Architecture for Context-Based Decision Making. In: Serafini L., Bouquet P. (eds.) Proceedings of the CRR'05 Workshop on Context Representation and Reasoning.

[16] Schilit, B. N., Adams, N., Want, R., Décember (1994). Context-Aware Computing Applications, *In:* Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, USA, p. 85-90.

[17] Dey, A. K., Abowd, G. D., Wood, A. CyberDesk, (1999). A Framework for Providing Self-Integrating Context- Aware Services. Knowledge-Based Systems, 11, 3- 13,.

[18] Pascoe, J. (1998). Adding Generic Contextual Capabilities to Wearable Computers. 2nd International Symposium on Wearable Computers, p. 92-99.

[19] Dey, A. K., Abowd, G. D. (2000). Towards a Better understanding of context and context- Awarness. *In:* At the CHI 2000 Workshop on the What, Who, Where, When, Why, How of context-Awareness.

[20] Burrell, J., Gray, G. K., Kubo, K., Farina, N., (2002). Context-aware computing: a text case, *In:* Borriello, G., Holmquist, L. E. (eds.), LNCS 2498 - 4th International Conference on Ubiquitous Computing, Springer-Verlag, p. 1-15.

[21] Brown, P. J., Bovey, J. D., Chen, X. (october 1997). Context-aware applications: from the laboratory to the marketplace, *IEEE Personal Communications*, 4 (5) 58-64.