SDPS-2021
Printed in US, December 2021

SOFTWARE ARCHITECTURE FOR SITUATION AWARENESS IN HUMAN-MACHINE INTERACTIONS FOR APPLICATIONS IN HUMANITARIAN EMERGENCIES

**Sirajuddin Asjad, Kristian Alfheim*, Bjørn-Ivar Bekkevold,
Karoline Moholth McClenaghan and Radmila Juric
University of South-Eastern Norway, * Kongsberg Gruppen
216988@usn.no kristian.alfheim@kongsberg.com 146164@usn.no
karolinem@usn.no rju@usn.no**

ABSTRACT

Since the early 2000 there has been repeated interest in developing context aware software applications, which can react to and understand the environment, and sometimes become self-tuned and situation aware. The idea of context awareness is now speeding towards mobile and wireless computing and have applications from Internet of Things and Wearable Healthcare to Pervasive Cyber Physical Spaces. This paper looks at possibilities of applying the same paradigm in situations of machine-human interactions and focuses on cases where teamwork between humans and devices, which collectively operate in risky and dangerous real-life environments, is essential for accomplishing a given task. It is important to address the issue of moving towards the edges of computational networks, utilizing possible fog/cloudlets solution and move away from cloud computing. The example used for the illustration of the proposal is in the field of the robotic explosive hazard detonation in war thorned countries.

INTRODUCTION

This paper proposes a software architecture (SA) model (Cervantes, 2016) which defines a building block of software applications suitable for managing cyber physical spaces created for explosive hazard detonations in war affected countries. Considering that the semantic of situations, in which such applications would run, include expected and unexpected moments, it is important that the SA captures all possible data and defines which types of computations we may run, on a selection of devices and computational servers. This research is a step forward for debating what would this move, moving towards the edges of computational networks (López et al., 2015; Seal & Mukherjee, 2018; Shi & Dustdar, 2016) and away from cloud computing solutions, bring in creating such software applications. Software technologies needed for deploying the SA model range from traditional transactional processing, applicable in mobile environments such as Android, to learning and predictive technologies, which can be used for image/speech recognition and potential predictions for assisting in decision making. The design of a conceptual solution, which adheres to the proposed SA, must underline the impact of an ad-hoc and sensor generated data and human involvement in decision making, on the successful deployment of this type of software.

The paper is organized as follows. In the Scenario, we illustrate a real-life situation within which we would wish to place our software application. In the section which follows we model the functionality of the application through Use Case models, which clearly specify the main functionality of the application: defining the content in which application runs and making sure that decision making is based on this context. The building blocks of software architectures are derived from sequence diagrams, and they are essential for proposing component based and layered SA style. The outcome of the research in debated in Conclusions.

THE SCENARIO

A situation in which we place the software application is in explosive hazard detonation in war affected urban and rural locations. It is assumed that (a) a potential explosive has been detected (b) it must be detonated (c) there is a team of humans available for managing software application which assists in the detonation and (d) there are numerous devices with variable computational power within the environment where detonation takes place. This situation is a typical example of a risk averse environment in which humans play important role from both side: running and making decisions using software applications and,6,7 detonating explosive hazard. The environment is data rich, equipped with devices and could resemble instances of internet of things (R. Juric, 2019; R. Juric, Madland, O.,, 2020; Sheth, 2016) Consequently. The semantic of data/computations typical of this situation could require contextualization (Barrett & Power, 2003; Bettini et al., 2010; Dey, 2001), in terms of understanding “what is happening” within the environment and which decisions are to be made. Decision making, such as “shall we detonate the object and how” could be either solely supported by automated decision making through software application or supported by human involvements in terms of adding more semantic to the detected context and putting risk averse aspect of the situation into human’s hands. It is expected that the software application would react on the context and had impact on decision making according to the semantic stored on the context.

It is important that software modelling, for such applications, start with generic solutions of the SA and proceed further, towards the deployment of SA and the implementation of the application following the most suitable framework for computing. It will not be prudent to speculate on possibilities of using clouds, cloudlets, fog and

edge computing, before there is a clear picture on the nature of computing programs (to be develop) and the size, type and structure of data we need to run them.

MODELLING SOFTWARE FUNCTIONALITY

Figure 1 is a typical model of pervasive software application functionalities, and it can fit into many cyber physical spaces for decision making, where (i) enormous amount of sensor generated data exists, (ii) the same data is being collected and used in decision making across the application (iii) the powerful data may affect physical devices which might not be able to compute locally and (iv) we must address constant changes in the environment through contextualization. Therefore, the reusability of the diagram in Figure 1 is striking. It allows for refinement in further exploration of use cases and the use case diagrams by using UML stereotyped dependencies. An example is in Figure 2 where Configure Context use case has been itemized. It is important to note that Figure 2 is one of many decisions on "what constitutes a context" and it mirrors a design decision for this configuration. Figure 2 should change if we place Figure 1 in a different context.

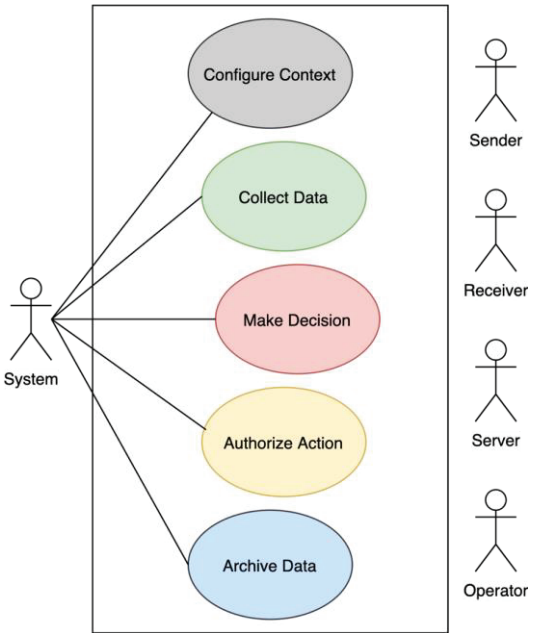

Figure 1 Use Case model

Figure 1 also shows that it would be prudent not to overload safety critical computations with a heavy programming code which would need constant attention due to contextual changes and constant need for decision making. Therefore, it is expected that the most complex computation would be Created Context and the lightest should be Authorize Actions. The former would be based on the extensive semantic of data collected though Collect Data and Configure Context and as such would not need heavy programming code. The most complex computations and coding would appear for Creating Context because it would depend on the nature and type of data which define context. Therefore, this use case would be crucial in determining which data we need for the core of computing: Make Decisions and Authorize Action.

The complexity of Use Case diagrams in further itemization of functionalities, as depicted in Figure 2 may have impact on the further software development, through modeling with abstractions. This will affect the generation of UML sequence diagram and therefore there should be a clear balance between the excessive use of stereotyped dependencies in the use case model and the role sequence diagrams play in creating software architectures. This is difficult to judge. It is hard to keep the balance between the two, but at least SA might be able to highlight if we have unnecessary over-itemized the generic functionality of the proposed software solution.

Figure 2 also signals something else as a part of SA design. Despite having "Configure Context" as a base use case in Figure 1, Figure 2 actually explains what "Configure Context" exactly means. It is self-explanatory from Figure 2 that each context definition must include "Planning Mission", "Creating Team" and "Setting Access" to devices and computations. From this perspective, it is obvious that functionality behind all these three use cases in Figure 2 depends on the data which has been collected through some other use cases (such as from "Collect Data") and the data which has been recorded as a result of decision making (such as from "Make Decision"). Consequently, the data which resulted from computations from Figure 2 are also allowed to be shared and thus would create semantically rich computational space for defining and manipulating context.

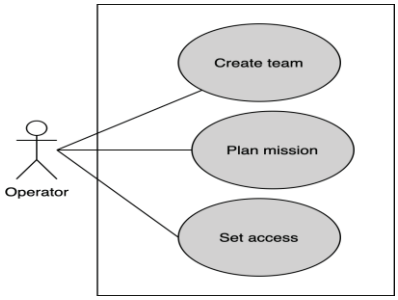

Figure 2 Configure Context Use Case Model

Building Blocks of Software Architectures

Figure 3 is one of numerous sequence diagrams developed from the use case model in Figure 1. It shows one aspect of contextualization where "the team for detonation has been created", as outline din Figure 2. It shows "Create Team" and defines if the data is involved in either update or retrievals. The top row of such sequence diagrams is the first glimpse of SA building blocks, which guarantee the existence of user interfaces (UI), computational software component, as CT.C and followed by a set of data of any length, type and generated by any use case from the itemized base uses cases from Figure 1.

Due to space restrictions, we cannot show numerous variations of the sequence diagram from Figure 3, but it is

obvious that this use case generates or updates the data for creating teams, operator data and the data for securing access in decision making (and add data for securing safe detonations!). The data which is needed before these updates are being made are simple "readings" of relevant data which could create a "context" in a moment when decision on detonation is being made. The sources of the data are numerous and should be shown in the proposed SA model. The abstractions of such data should be revealed in any component based and layered SA style within its bottom layer.

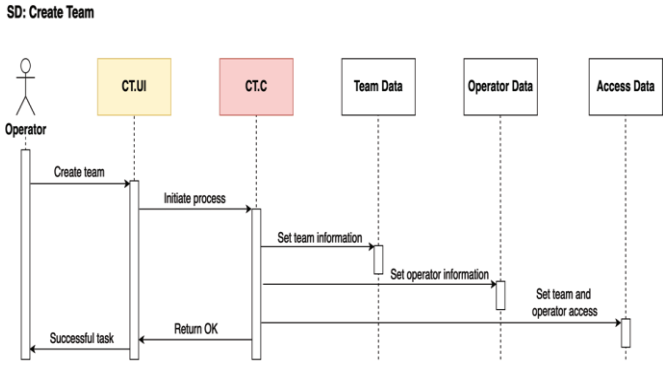

Figure 3 Illustration of Sequence Diagram "Creating Team"

PROPOSING THE SOFTWARE ARCHITECTURE

The SA from Figure 4 has been created from 20+ sequence diagrams. It shows three distinctive aspects of the software application it will generate.

Firstly, it is extremely data intensive and as such it should affect the choice of computational framework for its deployment. However, this is the first signal that we need to rethink where the computations of this software application will take place. We must determine how close we should go when deciding to move computations near the places where data originate, i.e., do we wish to compute locally, at the edge of computational networks. Figure 4 is also supposed to show a generic model of a pervasive computing software application, which has computational power clearly defined in the middle layer. However, due to constant data generation and the use of existing and persistent data storages, as visible in the data layer, it sways towards heavy data management application as opposed to be centered around computation algorithms which manage controlled detonation of explosive objects in a particular situation.

Second, three's a high reusability of data and which a double edge sward. It is welcome because it shows that the SA model maximizes the presence of data, does not create separation of data according to their role in computing programs and keeps an eye on "separation of concerns" which is essential in the deployment of SA across clouds edges. On the other side, it triggers numerous questions:

(a) Have we "found well defined functionalities through use cases"?
(b) Are the functionalities from Figure 4 generic enough to allow changes during the life of the software application?
(c) What does the data layer "say"?
(d) Could we plug and play with all these software components from Figure 4.
(e) Does the data layer allow the freedom in determining which data would become persistent data storage, which data will be archived for further analysis, possibly with learning and predictive technologies and which data we will simply discard after computations?

Thirdly, the contextualization is essential in pervasive software application and our SA clearly shows that the data created for a particular instance of the "context" feeds decision making and influences authorization of actions. This is one of the most precious parts of the SA and overall design. However, storing contextual data in persistence and archiving it, in constantly changing environment, which can differ from one moment to another, puts enormous burden on software applications. We need to balance such decisions. The SA form Figure 4 shows a clear picture of the level of efficiency we may have when running it. We see the re-use of data, and the amount of data needed for running each of these functionalities depicted on Figure 1. Considering that the software application should run in restrictive, risk averse and possibly dangerous environments, then we must pay attention to its flexibility when running it and efficiency in terms of decision making. There is no silver-bullet answer in this case.

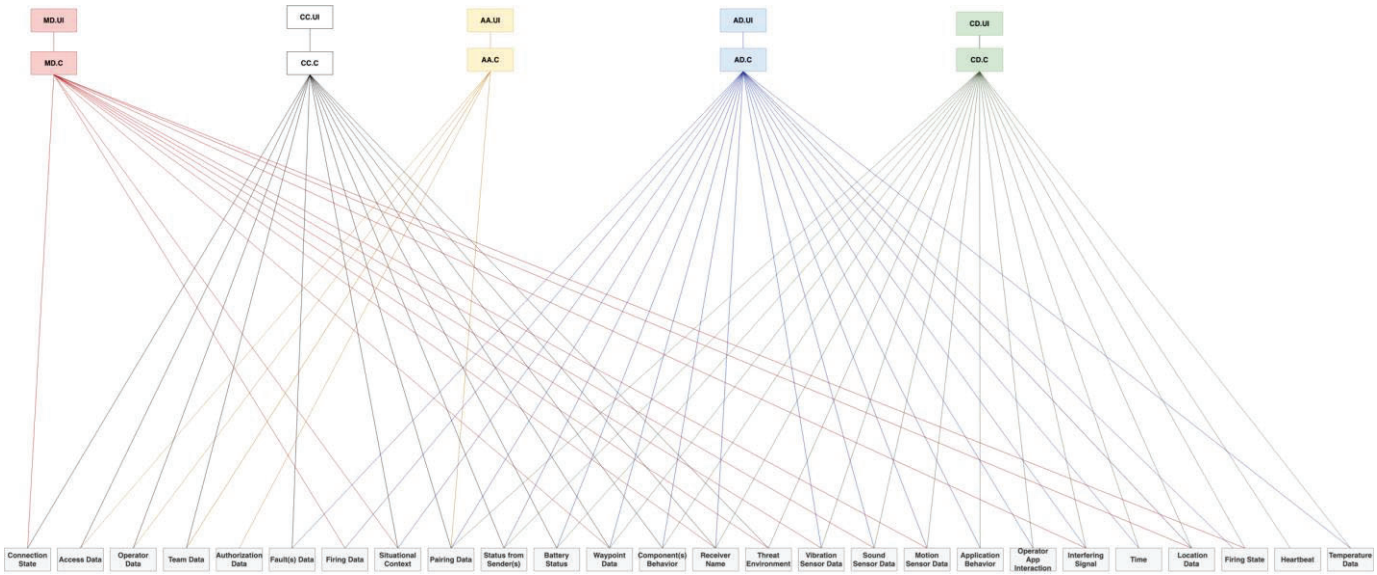

Figure 4 The Software Architecture

CONCLUSIONS

This paper is written by computer scientists and placed directly in the environment, typical of cyber physical spaces, where decision making, potentially supported by human involvement, manages risk averse situations, in which hazardous explosive objects were detected and detonated.

Software modelling illustrated in the paper adheres to the main principles for creating dynamic and constantly changeable software solutions which would react to the environment where the application is run. Also, using component based and layering SA style we adhere to the main principle of sound software engineering, and keep "separation of concerns" alive through layering. Therefore, it would be difficult to depart from these principles without affecting the deployment, implementation and efficiency and safety of the software application we need.

However, there is something else more important as an outcome of the research. In the introduction we promised to debate strategies of moving computations in these applications away from the cloud and inspect possibilities of pushing forward localized and edge computing. It appears that this is not as straightforward tasks as one could hope for.

Considering that physical items cannot be modeled when designing software applications which will create instances of cyber physical spaces (we use abstractions instead), we have discovered strong level of data sharing (including the data generated by devices in real time) which in turn have direct impact on the choice of technologies we can use in implementation and the choice of computer/communication frameworks we must have to safely run the application.

Figure 4 shows that there is no way that we can determine whether we will run the application at the either edges (locally on the small devices) or on the cloud or on both or fog frameworks, or on all. The deployment of the model is as critical as running computer program which impalements "Authorize Action". This is because we need to act according to the "context" (i.e., data values stored in the software application) which we find in the moment when we have to "detonate a hazardous object". Deciding in advance on the exact configuration of the implemented model would be extremely risky.

Where do we go from here?

Modern pervasive spaces are our reality, but software applications which create them and make these spaces "useful" are very specific. If we do not address the special characteristics of pervasive spaces, then we will not be able to create them. The solution in which we always look at "instances of either IoT or Internet of Everything (Barrett & Power, 2003; Bettini et al., 2010)" are promising. If we consider that any software component available in the layered SA style in Figure 4 can be on any node, any server, any cloud and on any device, including wearables, robots, vehicle, drone, and cyborg, then it is obvious that we need careful thinking before we deploy the solution. The deployment should not be commercially dictated.

At the time of writing this paper, no publications have been found which would comprise related work.

REFERENCES

Barrett, K., & Power, R. (2003). *State of the Art: Context Management.* Paper presented at the Ambient Intelligence: European Conference, AmI, Germany.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., & Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput., 6*, 161-180.

Cervantes, H., Kazman, R.,. (2016). *Designing Software Architectures: A Practical Approach (SEI Software Engineering)* (1 ed.): Addison-Wesley Professional.

Dey, A. K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing, 5*(1), 4-7. doi:10.1007/s007790170019

Juric, R. (2019) Semantic Model for Creating an Instance of the IoT, in Proceedings of the IEEE CyberScieTech conference, Fukuoka, Japan August 2019.

Juric, R., Madland, O.,. (2020). *Semantic Framework for Creating instance of IoE in Urban Transport: A Study of Traffic Management with Driverless Vehicles*. Paper presented at the under review for HICSS 52, Hawaii, USA.

López, P. G., Montresor, A., Epema, D. H. J., Datta, A., Higashino, T., Iamnitchi, A., . . . Riviere, E. (2015). Edge-centric Computing: Vision and Challenges. *Comput. Commun. Rev., 45*, 37-42.

Seal, A., & Mukherjee, A. (2018, 2018). *On the Emerging Coexistence of Edge, Fog and Cloud Computing paradigms in Real-Time Internets-of-EveryThings which operate in the Big-Squared Data space*.

Sheth, A. (2016). Internet of Things to Smart IoT Through Semantic, Cognitive, and Perceptual Computing. *IEEE Intelligent Systems, 31*(2), 108-112. doi:10.1109/mis.2016.34

Shi, W., & Dustdar, S. (2016). The Promise of Edge Computing. *Computer, 49*(5), 78-81. doi:10.1109/mc.2016.145