A Novel Integrated Architecture for Ambient Assisted Living Systems

Ashalatha Kunnappilly
Mälardalen University, Sweden
ashalatha.kunnappilly@mdh.se

Alexandru Sorici
Politehnica University, Romania
alex.sorici@gmail.com

Imad Alex Awada
Politehnica University, Romania
awadaalex@hotmail.com

Irina Mocanu
Politehnica University, Romania
irinag.mocanu@gmail.com

Cristina Seceleanu
Mälardalen University, Sweden
cristina.seceleanu@mdh.se

Adina Madga Florea
Politehnica University, Romania
adinamagdaflorea@yahoo.com

***Abstract*—The increase in life expectancy and the slumping birth rates across the world result in lengthening the average age of the society. Therefore, we are in need of techniques that will assist the elderly in their daily life, while preventing their social isolation. The recent developments in Ambient Intelligence and Information and Communication Technologies have facilitated a technological revolution in the field of Ambient Assisted Living. At present, there are many technologies on the market that support the independent life of older adults, requiring less assistance from family and caregivers, yet most of them offer isolated services, such as health monitoring, reminders etc; moreover none of current solutions incorporates the integration of various functionalities and user preferences or are formally analyzed for their functionality and quality-of-service attributes, a much needed endeavor in order to ensure safe mitigations of potential critical scenarios. In this paper, we propose a novel architectural solution that integrates necessary functions of an AAL system seamlessly, based on user preferences. To enable the first level of the architecture's analysis, we model our system in Architecture Analysis and Design Language, and carry out its simulation for analyzing the end-to-end data-flow latency, resource budgets and system safety.**

***Index Terms*—Ambient Intelligence, Ambient Assisted Living, Architecture Analysis and Design Language.**

I. INTRODUCTION

According to the statistics of the World Population Ageing Report 2015, the world's elderly population is predicted to reach 2.1 billion by 2050, which is more than double of the population of elderly adults in 2015 [1]. The ageing society entails coping with an increased number of diseases, increased health-care costs, shortage of caregivers [2], etc. Assisted living systems can help in supporting elderly persons in their daily activities and their independent living, with limited risks.

Nowadays, there are numerous Ambient Assisted Living (AAL) solutions available, ranging from a large variety of health monitoring and fall detection sensors, smart homes and assisted robots [3]. However, most current systems are not very effective in critical situations due to not sufficient support of integration of functionalities, difficulty of usage and low acceptance rates [4][5]. One such scenario that supports this claim and that we also analyze in this paper is the occurrence of "fire" and "fall" events simultaneously. When both these events occur together, a safe mitigation of the scenario is achieved only when both these events are communicated to caregivers and firefighters; which is not guaranteed by independent systems working side by side. Assuming that the fire alarm communicated to the firefighters is verified for confirmation by a phone call to the user's home, it follows that the elderly who has fallen that has been communicated to the caregivers only cannot answer in due time, so the fire alarm may be deemed false and discarded, triggering a potential catastrophe. The fact that the existing AAL products do not integrate the modules targeted towards the particular needs of a user, namely health monitoring, home appliances control, report and communication with health professionals, telepresence module etc., offering a solution that could safely resolve potential combined critical situations, also confirmed by model-based behavioral analysis of the system, serves as the motivation for the research presented in this paper. We propose a novel modular architecture for AAL that can be seen as a fully integrated solution, with functionalities selected based on user choices. Our proposed architecture is designed by taking into account the pros and cons of existing prominent AAL architectures in the literature. Since our solution should operate appropriately in critical situations also, it is important that its quality-of-service (QoS) is analyzed. To achieve this, we model the proposed architecture in the Architecture Analysis and Design Language (AADL) [6], and carry out simulations in AADL to estimate the end-to-end flow latency, resource budgets and system safety.

The remainder of the paper is organized as follows. In Section 2, we discuss some of the prominent architectural frameworks developed for AAL and underline their advantages and disadvantages by simulating these architectures in AADL. Section 3 describes our integrated architecture, whereas in Section 4 we present the run-time architecture model in AADL. The simulation results depicting the end-to-end flow latency, resource budgets and safety analysis in AADL are presented and discussed in Section 5. In Section 6, we conclude the paper and outline future lines of research.

II. LITERATURE REVIEW

Ambient Assisted Living and Ambient Intelligence (AmI) techniques are some of the most researched areas in the past few years due to an increasing amount of elderly population across the world [2]. At present, there are no market solutions

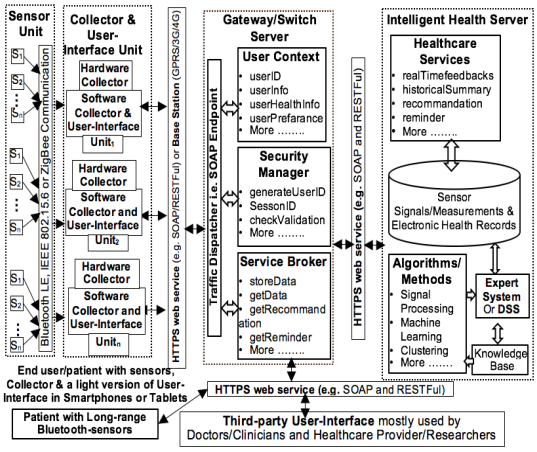

Fig. 1. ESS-H architecture.

that offer a complete integrated solution for AAL. Consequently, in the following, we survey the existing literature on AAL architectural solutions, and identify advantages and shortcomings through AADL simulations in OSATE 2.2.1. AADL has been chosen for architecture analysis due to its architecture-centric and model-based engineering approach, sound specifications and large acceptance in the industry for modeling embedded systems.

This section is organized as follows: in subsection A, we give a brief introduction to AADL and in subsection B, we describe in detail the existing architectures in literature and the results of their AADL simulation.

A. Architecture Analysis and Design Language

AADL [6] models a system's architecture in terms of hierarchies of components at various levels of abstractions, whose interaction is represented by connections via ports (data, event and event data ports). There are three categories of component abstractions in AADL - *Software, Execution Platform and System*. Application software comprises the process, data, subprogram, thread, and thread group components. The execution platform is made up of computation and communication resources, consisting of processor, memory, bus, and device components. System components are composites that can consist of other systems as well as software or hardware components. The major components are: 1) *Process* - a unit of protected address space, 2) *Data* represents a type, local data subcomponent, or parameter of a subprogram 3) *Thread* - unit of concurrent execution based on various protocols (periodic, aperiodic, sporadic, server and background), 4) *Processor* - a virtual machine that schedules and executes threads, 5) *Memory* - a storage abstraction that can hold data or code, 6) *Bus* - a connector abstraction between execution platform components, and 7) *Device* - an abstraction of an active component that an application system can interact with, and a processor executing software that requires access to, via a bus.

B. Prominent AAL architectures in literature

By examining the AAL literature, we identify some architecture types that address the construction of integrative AAL applications (that is, those that focus on creating a holistic user experience, not just the development of a specific functionality such as health data management or social interaction). In what follows, we investigate two commonly-used architecture types: Cloud based and Multi-Agent System (MAS), showing an example for each.

1) **Cloud-based AAL architectures.**
In this category, we describe the ESS-H (Embedded Sensor Systems for Health) [7], shown in Fig.1. Although the architecture supports multiple functionalities like health monitoring, fall detection, communication to caregiver etc., there is no support for home monitoring (with fire detection systems), robots etc. Hence, inorder to analyze the scenario of simultaneous fire and fall events, the only choice would be to use an independent fire detection system along with the ESS-H. In a related work [5] , the authors argue, based on sequence diagram simulations, that the timing constraint of taking a correct decision could not be met by two independent systems working side by side. However, sequence diagram simulations cannot analyze flows through the components, while considering the sensor sampling times, component execution times and communication bus latencies. Hence, we describe the architecture, systems requirements and flow latency analysis in AADL, for both the ESS-H fall detection system, and the separate fire detection system, respectively.
a) ESS-H Fall Detection: Architecture Description in AADL: The major components of ESS-H architecture are sensor unit, collector and user-interface unit, the gateway and switch server, and the intelligent health server (IHS), with the servers being cloud based as shown in Fig. 1. The data flow during the occurrence of a fall event is described as three end-to-end flows in AADL as shown in Fig. 2. Flow 1 describes the data flow from issuing the fall event, until its sending to the caregiver through the data collector (modeled as a process with a thread for analyzing data), gateway (modeled as a device) and cloud (modeled as a process with a thread for communication); Flow 2 is models the caregiver's confirmation of the fall event through the gateway and data collector (which in this case is a tablet); Flow 3 describes the caregiver's action on the elderly in case of a genuine alarm. As we cannot model human behaviors accurately, we model them as devices with maximum allowed response times.
b) ESS-H Fall Detection: System Requirements: The system's latency requirements are described in Table I, in accordance with values based on previous work [5], except the split of the human response times into the time needed to register the information and the time needed to take the action. For instance, the maximum allowed response time of the caregiver from the time of being

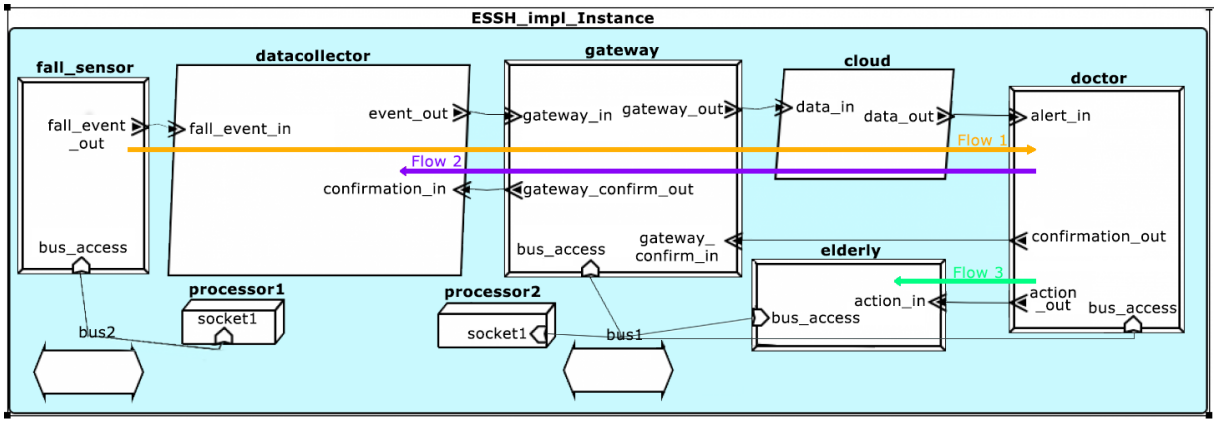

Fig. 2. ESS-H architecture in AADL describing data flow during fall event.

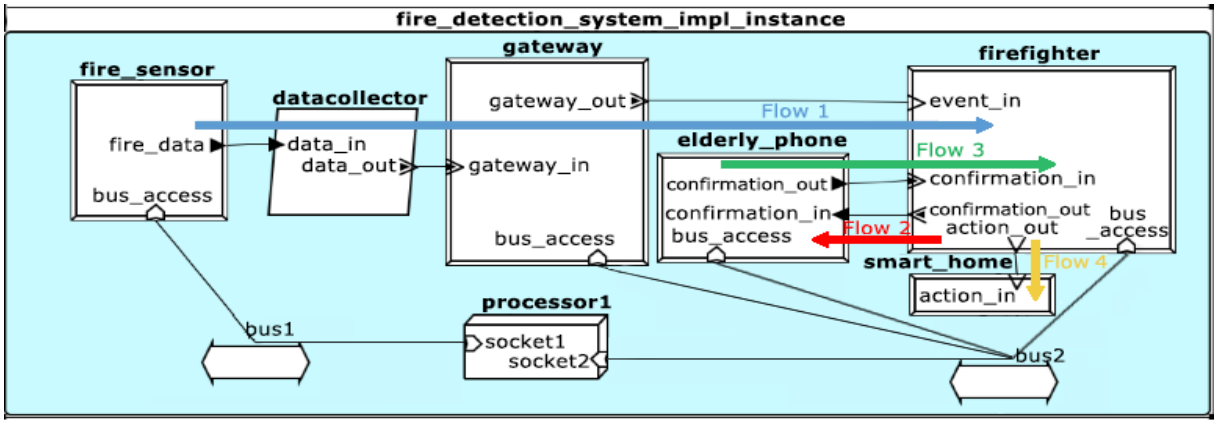

Fig. 3. Fire detection system architecture in AADL describing data flow during fire event.

informed of a fall event is 2,96 min [8]; we have divided the response time arbitrarily as 1 min for registering the data (Flow 1) and 1.96 min for the response (Flow 2). The maximum latency values for the end-to-end flows are shown in Table I. For the analysis, we assume periodic sampling of the fall sensor, every 30 sec. All other components are also activated at 30 sec to effectively handle the sensor data. The communication, except that of caregiver's action occurs via two buses - Bluetooth (Latency: 150 to 200 ms) and Internet (Latency: 50 to 100 ms).

c) ESS-H Fall Detection: Analysis Results: The AADL flow latency analysis results are generated based on considering the processing time of the tasks, processing delay due to queueing, transfer time of information between connections etc. The end-to-end latency analysis results are shown in Table I. All of the three flows meet their maximum end-to-end latencies.

Next, we analyze the independent fire detection system that detects the fire event and communicates it to the firefighters.

d) Fire Detection: Architecture Description in AADL: The fire detection system has a fire detection sensor, a data analyzer module to process the sensor data, a gateway device and a mobile phone to communicate with the firefighters. The fire sensor, gateway, mobile phone and firefighters are modeled as devices, however the data collector is modeled as a process with an associated

TABLE I
SYSTEM REQUIREMENTS AND FLOW LATENCY ANALYSIS RESULTS OF ESS-H.

Name	System latency	Max Latency (AADL)
Fall detection system	1233600 ms	928755 ms
Flow1	156255 ms	97755 ms
Flow2	177600 ms	30800 ms
Flow3	900000 ms	800200 ms

TABLE II
SYSTEM REQUIREMENTS AND FLOW LATENCY ANALYSIS RESULTS OF FIRE DETECTION SYSTEM.

Name	System latency	Max Latency (AADL)
Fire detection system	741600 ms	811400 ms
Flow1	91000 ms	69300 ms
Flow2	28000 ms	12000 ms
Flow3	25000 ms	30100 ms
Flow4	600000 ms	700000 ms

TABLE III
FLOW LATENCY ANALYSIS RESULTS OF AGENT BASED SYSTEM.

Flows	System Latency Requirements	Max Latency (AADL)
Flow1	91000 ms	170100 ms

thread that deals with processing fire sensor data. The AADL model of the fire detection system contains 4 flows: Flow 1 for communicating the fire event to the fire fighter, Flow 2 for confirmation of the fall event, Flow 3 for modeling the response to the confirmation call and Flow 4 for capturing the firefighter's action. The AADL model and associated flows are illustrated in Fig. 3.

e) Fire Detection: System Requirements: The system's latency requirements are tabulated and described in Table II [5]. For the flow analysis, we model two different device implementations for mobile phone and firefighter - one showing the normal behavior (in case of Flow 1 and Flow 2) and the other showing the delayed behavior (in case of Flow 3 and Flow 4). Moreover, the Flow 2 is dependent on data from Flow 1, we divide 10 sec response time as 7 sec for registering the fire data and 3 sec for the response. The maximum end-to-end latencies of the flows are thus 1.5, 0.46, 0.4 and 10 min respectively (Table II). We assume that the confirmation call is not answered by the elderly within 5 min. For the analysis, we assume periodic sampling for the fire sensors every 20 sec and all the communication, except firefighter's action occurs via two buses - Bluetooth and Internet with the same prescribed latencies as before.

f) Fire Detection: Analysis Results: The end-to-end flow latency results show that Flow 3 and 4 miss their deadlines. The simulation results are also shown in Table II. Our analysis shows that the ESS-H solution that consists of fall detection capabilities working side by side with an independent fire detection system might not cover our critical scenario safely, that is, the response actions are not completed by their deadlines. However, later in the paper, we show that if the fire and fall detection capabilities are integrated into the same system, such critical scenarios can be handled in due time. Moreover, the ESS-H solution also runs the risk of single point of failure due to the centralized IHS. The dependency of the architecture on Internet connectivity is high, and there is no local processing of the data, and so the system is exposed to a complete failure in the absence of a working Internet connection, or when the IHS is not able to respond in real-time.

2) **Distributed agent-based AAL architectures.** Next, we investigate whether a distributed solution serves the purpose effectively. Thus, we hereby analyze a distributed multi-agent system architecture proposed to support people suffering from the Alzheimer disease [9].

g) Agent architecture: Architecture Description in AADL: The architecture uses a Flexible User and a Service-Oriented multi-ageNt Architecture (FUSION@) [10] and is depicted in Fig. 4. Though this architecture does not

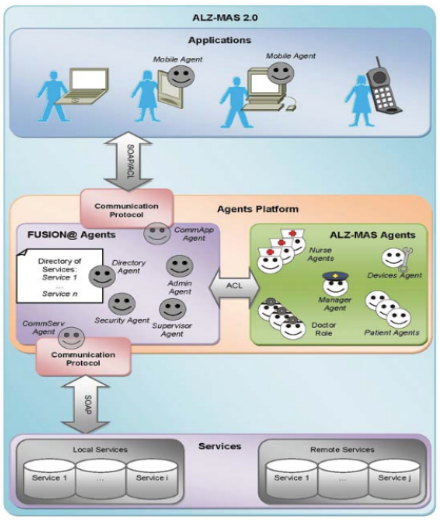

Fig. 4. A service oriented MAS architecture for Alzheimer health care.

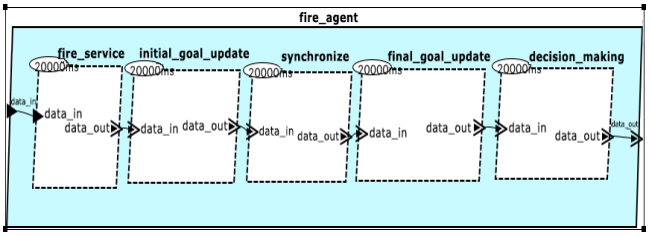

Fig. 5. Fire agent behavior as AADL threads.

offer a fully integrated functionality, it can be easily extended to support the intended functionalities owing to its distributed nature. Let us assume that the agent-based architecture contains a fire-agent and a fall-agent to deal with fire and fall events respectively. The actions for a fire agent include detecting the fire event, updating its event list, synchronizing this event with fall agent, updating the event list again and finally taking the decision. We map all these sequences as separate threads in AADL as shown in Fig. 5. In this system model, we analyze whether fire and fall events are effectively communicated to the firefighter in due time taking into account the agent synchronization and communication delays.

h) Agent architecture: System Requirements: We assume the system's requirements similar to those of the fire detection system described earlier and we assume communication latency of 50 ms to 100 ms for sending synchronization messages.

i) Agent architecture: Analysis Results: The results are tabulated in Table III. As shown, Flow 1 clearly misses its deadline. Hence, even though distributed systems offer higher performance due to resource sharing and has higher reliability and fault tolerance when compared to centralized counterparts, data synchronization becomes a new problem and so does the unpredictable nature of the

system (the response times are dependent on the system organization and network load), as shown also by the flow latency analysis. Moreover, if we consider the above system, the solely local deployment of agents makes the system's maintainability difficult. In addition, as the system becomes more distributed, security challenges are higher, and the system's maintainability becomes difficult. As a result of the carried analysis, we opt for a centralized solution, with necessary fault tolerance to deal effectively in real-time, especially in scenarios where multiple events occur together.

Based on the above, we conclude that none of these architectures can be directly used as a framework for building fully integrated AAL systems. Moreover, we have not found any evidence ensuring various QoS attributes. To alleviate such inconveniences, we propose a new architectural solution, named CAMI (Companion with Autonomously Mobile Interface), which is a nominal mix of the studied solutions. The CAMI architecture is described in detail in the following sections.

III. Proposed Architecture

In this section, we describe our novel integrated architecture for AAL, named CAMI[1]. CAMI offers a fully integrated AAL solution by providing services for health monitoring, fall detection, supervised physical exercises, home management and wellbeing as well as telepresence support. CAMI builds an artificially intelligent ecosystem that allows the seamless integration of any number of ambient and wearable sensors, with a mobile robotic platform endowed with multimodal interaction (touch, voice, person detection), including a telepresence robot with manipulation capabilities. As compared to existing solutions, the functionalities that we have chosen to integrate in CAMI are based on user preferences recorded via a multi-national survey with 108 primary and 58 secondary users from Denmark, Romania and Poland.

The architecture is based on the following underlying assumptions: 1) Biometric data must be handled with extreme caution due to privacy laws, 2) The end result of the project has to be feasible from a commercial point of view, 3) Due to the need of a business model, the CAMI system as a whole needs a cloud-based infrastructure.

The CAMI architecture is based on microservices, and has a clean and robust skeleton, onto which several plug-in modules can be coupled ensuring **modularity** and **reuse**. Two distinct features of CAMI, as compared to other AAL architectures, are: (i) the presence of both **local and cloud-based** processing schemes, and (ii) the **continuity of services** even in the absence of Internet. The CAMI architecture is depicted in Fig. 6. The major components of the architecture are: Sensor unit, Data collector unit, CAMI Gateway, Mobile phone unit, Telepresence, and the CAMI Cloud, which are discussed below.

[1]http://www.aal-europe.eu/projects/cami/

(i) *Sensor unit*: The CAMI system includes various health monitoring sensors, environmental sensors, physical exercise monitoring sensors and fall sensors.

(ii) *Data collector unit*: The interfacing of a wide range of specific sensors/devices with the CAMI ecosystem is achieved by the *Data collector* unit. The unit acts as an intermediate layer aiming at clearly separating the devices from the rest of the CAMI ecosystem, thus increasing its modularity and loose-coupling character.

(iii) *CAMI Gateway*: The CAMI Gateway is a collection of software modules that implement the core infrastructure of the CAMI system. Its purpose is to enable the easy interconnection of the micro services that provide the main functionality of CAMI, like the sensor data collection, intelligent short-term event processing, forwarding of shareable data to CAMI cloud services, etc. At the OS level, there is a switch that can shift the box's operation from Internet to GSM, if needed, in order to ensure that the CAMI system carries out its critical functionalities even in the absence of the Internet connection.

A typical CAMI gateway deployment hosts the following micro services:

1) *Event Stream Manager:* It is a part of the core infrastructure solution enabling message-based interconnection between all the other micro services.
2) *Local Data Storage:* Local data storage offers short-term storage for data collected from sensors, and user information inferred by the decision support algorithms.
3) *Decision Support System (DSS):* The DSS provides a collection of symbolic and data-driven reasoning algorithms that continuously monitor the short-term state of the user (current health status and mood, current and planned daily activities, required reminders, etc).
4) *Voice command manager:* It is offered as a service implementing voice-based interaction with the CAMI system.
5) *Communication with 3rd party health platforms:* This service allows the sharing of selected health measurements and physical exercise sessions with primary and secondary caregivers.
6) *Connection with the CAMI cloud:* Ensures the replication of locally-collected data to the CAMI cloud platform for longer term and higher level processing, as well as the communication with the CAMI cloud to retrieve the results of performed analyses or suggested actions (e.g., context-aware rescheduling of planned activities, and physical exercise recommendations).

(iv) *Robotic telepresence unit*: The CAMI architecture is equipped with an integrated robotic support that is missing from many of the existing AAL frameworks [3]. The robotic teleprsence unit in CAMI can be used for both input and output interactions, and is furthermore capable of actuation.

(v) *Mobile phone unit*: The mobile phone carried by the user acts as an intelligent, friendly collaborator that provides suggestions, advice or reminders. It is also equipped with automatic facilities of sending SMS to third party users like doctors or firefighters in case of emergency situations.

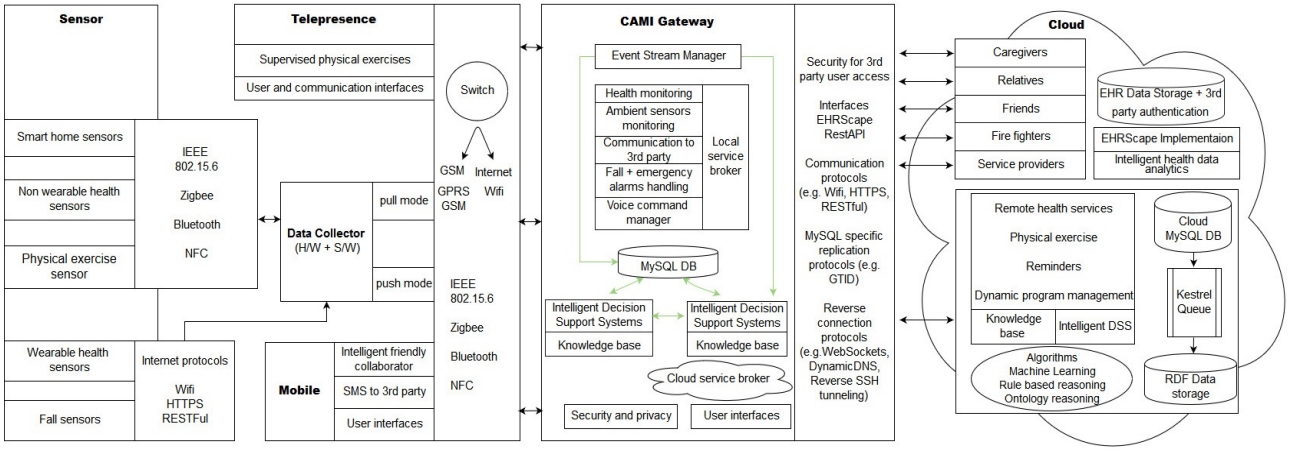

Fig. 6. CAMI - An integrated architecture for AAL.

(vi) *Cloud Services*: The CAMI cloud services enable the communication to secondary caregivers (family and friends), healthcare professionals, firefighters, and other CAMI instances. The unit also enables intelligent analysis of user data, collected over a prescribed period of time. Further, it supports the modeling of user data using Semantic Web Technologies (e.g., ontologies for Activities of Daily Living). Finally, cloud services enable the clear/easy administration of each CAMI user account.

TABLE IV
FLOW LATENCY ANALYSIS RESULTS OF CAMI SYSTEM.

Flows	Specified Max Latency	Max Latency (AADL)
Flow1(Fall)	156255 ms	48055.6 ms
Flow2(Fall)	177600 ms	40851.3 ms
Flow3(Fall)	900000 ms	800200 ms
Flow4(Fire)	91000 ms	90400.6 ms
Flow5(Fire)	28000 ms	22501.3 ms
Flow6(Fire)	25000 ms	19001 ms
Flow7(Fire)	600000 ms	402000 ms

IV. AADL MODEL OF CAMI ARCHITECTURE

By their nature, AAL systems are safety-critical real-time systems that need to mitigate possible real-time scenarios of high criticality, which could endanger the life of the elderly. Examples of such situations include scenarios when a person is having a cardiac arrest, or a fire at home etc. Usually, most errors are introduced at design stage of embedded systems, but they are discovered very late, leading to increased rework costs and re-engineering efforts. Therefore, modeling and analyzing AAL architectures at early stages of development can be used to ensure their real-time performance, schedulability, reliability and safety [6]. Consequently, we model the CAMI architecture in the architecture description language, AADL [6] using OSATE 2.2.1, and analyze the end-to-end data flow latency of sensor event flows. We also analyze the the system's resource feasibility, its safety and reliability.

The AADL model is shown in Fig. 7 and we use this model further to analyze the above attributes. The AADL model of CAMI specifies the communication and data flow between components, through data and event ports. We model all the sensors, the CAMI cloud (caregiver and firefighter), the telepresence, the mobile phone, the smart home and the elderly person as *devices*, and the data collector and the CAMI Gateway as *processes* (Fig. 7). The data collector process contains one thread called the "Data_analyzer_module", and the CAMI Gateway contains two threads, "Event_stream" and "DSS". The process components with their threads and port connections for communication are shown in Fig. 8 and 9. The "Data _analyzer_module" thread in the data collector preprocesses and analyzes the sensor data before it is passed to the CAMI Gateway. All the normal data is passed to the database through the data port of the data collector; however, if any deviations from normal values occur, the data is passed through the output event port, which is then fed to the input event port of the CAMI Gateway. The "Event_ stream" thread in the CAMI Gateway records all the generated events, and passes them to the "DSS" thread for determining the actions in case of events. All the threads except the "Event_stream" are assumed to be periodic, to ensure the continuous monitoring of active events generation, given that the environment is highly dynamic. In comparison, the "Event_stream" is aperiodic and is invoked each time an event occurs. The execution of the modules is controlled by 3 *processors* - "CAMI_Data_processor", "CAMI_Main_processor" and "CAMI_Cloud_processor" with access to various *buses* - "Bluetooth", "Internet" and "GSM".

V. CAMI ARCHITECTURE ANALYSIS IN AADL

In the subsections below, we outline the details of various AADL analyses of the CAMI architecture.

A. Flow latency analysis

In this subsection, we model the end-to-end flows to determine if the CAMI architecture can successfully mitigate critical scenarios involving simultaneous fall and fire events within the respective deadlines. In an integrated system, the occurrences and associated data of concurrent fire and fall

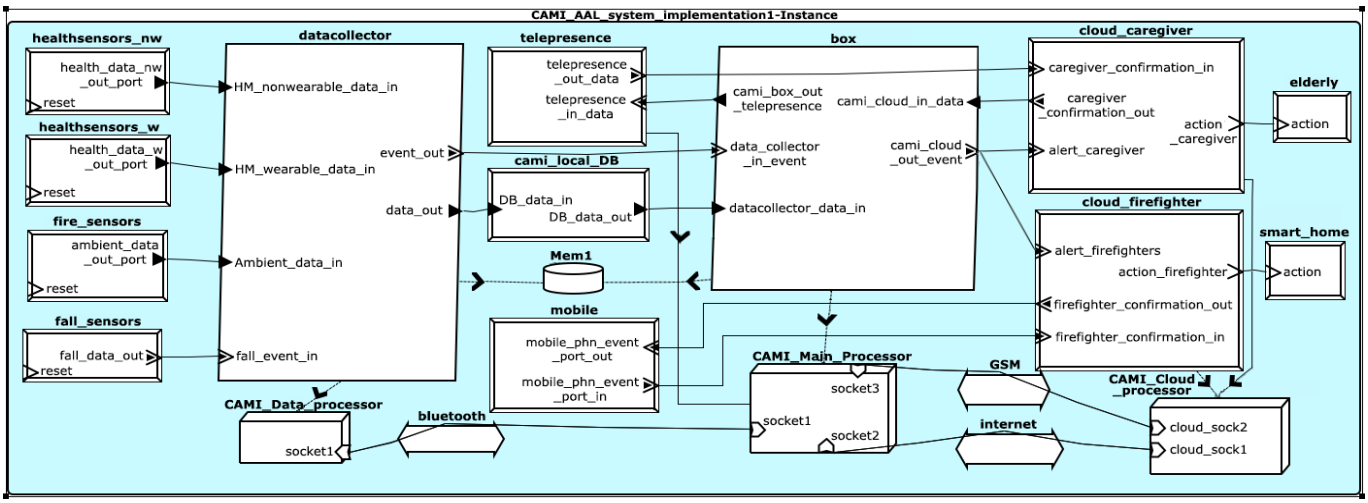

Fig. 7. CAMI system architecture in AADL showing component inter-connections.

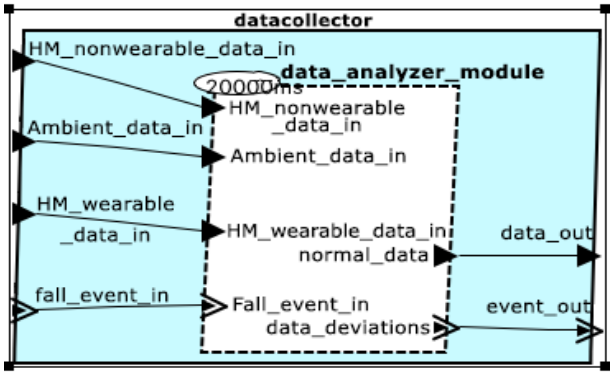

Fig. 8. Data collector process and its thread.

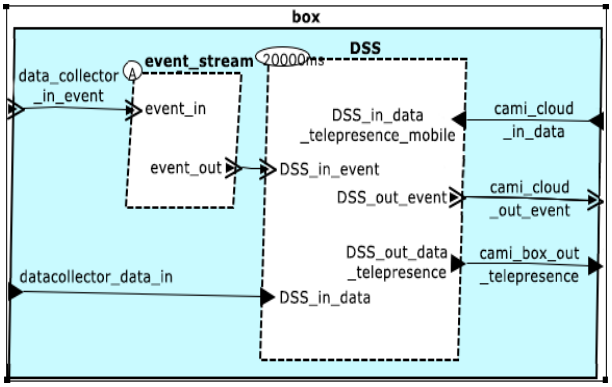

Fig. 9. CAMI Gateway process and its threads.

events are communicated to both caregivers and firefighters, rendering an immediate rescue action from the firefighters who do not need a phone-based confirmation anymore. The fall event data flow has the same 3 end-to-end flows as for the ESS-H architecture detailed in Section II. Similarly, the fire event data flow is as for ESS-H, adapted to CAMI. The end-to-end latency analysis results are summarized in Table IV, and the analysis concludes that all the flows meet their respective deadlines.

B. Resource analysis

AADL has resource analysis plugins also, to analyze resource budgets and allocation during the earlier stages of system development. Resource budgeting can be done for processors, memory, and network bandwidth and can be analyzed to determine whether budgets exceed the allocated sizes (feasibility analysis). We can allocate application components to the execution platform and reconsider the resource budgets in terms of those allocations.

Since our model is designed to illustrate that it could effectively handle the situation with simultaneous occurrence of fire and fall events, only the application components belong to this data flow. Therefore, we associate existing threads to "CAMI_Main_Processor" assigned with a capacity of 200 MIPS and memory capacity of 100 Kbyte, to analyze the respective resource usage. The analysis results shown in Table V illustrate that the tasks' resource usage exceeds the processor capacity and memory capacity, hence we add, more processor to our system "CAMI_Data_processor" with capacity 50 MIPS and increase the memory capacity to 150 Kbyte. The process "Data_collector" is associated with "CAMI_Data_Processor", the process "box" is associated with the "CAMI_Main_Processor" and a memory of 150 Kbyte is associated with the processors. In this case, all the resource budgets are met as shown in Table VI.

C. Safety analysis

In the following, we outline the safety analysis of CAMI architecture using the Error Annex (EA) V2 [11]. AADL facilitates different types of safety analysis like the fault hazard analysis (FHA), fault tree analysis (FTA), fault impact analysis etc.

```
annex EMV2{**
use types cami::error_library;
use behavior cami::error_library::simple;

error propagations
fall_data_out: out propagation{NoValue,InvalidValue};
flows
    ef0: error source fall_data_out{NoValue,InvalidValue};
    end propagations;
    component error behavior
    events
    Reset: recover event;
    Fault: error event;
    transitions
    t0: Operational-[fall_data_out{NoValue} or fall_data_out{InvalidValue} ]->Failed;
    t1: Operational-[Fault]->Failed;
    t2: Failed-[Reset]->Operational;
    end component;
    properties
        emv2::hazards =>
([crossreference =>"N/A";
    failure =>"NoValue";
    phases =>("all");
    description =>"No value from fall detection sensor";
    comment =>"Would impact detecting any fall events";
])
    applies to fall_data_out.novalue;
        emv2::hazards =>
([crossreference =>"N/A";
    failure =>"InvalidValue";
    phases =>("all");
    description =>"Invalid value from fall detection sensor";
    comment =>"Would impact detecting any fall events";
])
    applies to fall_data_out.invalidvalue;
    **};
```

Fig. 10. Error Annex specification of fall sensor.

TABLE V
RESOURCE ALLOCATION ANALYSIS RESULTS OF CAMI SYSTEM WITH SINGLE PROCESSOR.

Components	Resource Capacity	Resource Usage
"CAMI_Main_processor"	200 MIPS	203.5 MIPS
"Mem1"	100 Kbyte	140 Kbyte

a) Error Modeling: As a first step towards analyzing the safety of CAMI system, we define the error model of the individual components. The CAMI sensor devices are associated with two types of failure: 1) Value Error: When they have no value ("No Value" error) or when they have wrong value ("Invalid Value" error), or 2) Other failure events: E.g., internal failure due to system malfunction ("Fault"). We also define two states of operation of the devices, "Operational" and "Failed". Initially the system is in operational mode, i.e., it performs its required functionality without any errors. If any value error or other fault events occur, the system moves to the failed mode. To return from a failed mode, we defne a system self recovery event called "reset". Upon "reset", the system moves back to operational from the failed mode.

b) Safety Analysis: The FHA analysis of the architecture generates an excel report of all potential faults in the system. Fault impact analysis is used show how faults propagate in the system. For this, we assign all the sensor devices as the error flow sources. An example of EA of fall sensor is depicted in Fig. 10. Any of the errors in sensors propagate through the data collector and CAMI gateway to the cloud (error sink). FTA also helps us to analyze the failure effects by combining various failure events.

VI. CONCLUSIONS

In this paper, we have proposed an innovative integrated architecture with local and cloud-based processing for AAL systems. In order to validate the performance of our proposed architecture, we have modeled it in AADL, and analyzed the data-flow latency, resource feasibility and system safety. The end-to-end latency analysis has helped in deciding on the combined local and cloud-based centralized architectural solution. The resource analysis in AADL has effectively determined the processor and memory capacities required for executing the application components, facilitating the design decision of resource increase to remove the potential resource usage overflow. Safety analysis in AADL is vital to identify the potential system faults, and analyze their propagation within the system, such that one can recognize what components need back-up and devise error mitigation strategies later.

TABLE VI
RESOURCE ALLOCATION ANALYSIS RESULTS OF CAMI SYSTEM WITH TWO PROCESSORS.

Components	Resource Capacity	Resource Usage
"CAMI_Main_processor"	200 MIPS	200.0 MIPS
"CAMI_Data_processor"	50 MIPS	3.5 MIPS
"Mem1"	150 Kbyte	140 Kbyte

As future work, we intend to formally verify the CAMI architecture, including the internal behavior of components, especially the DSS behavior under critical scenarios. We envision to produce a full working prototype that will be deployed in the market in the near future.

REFERENCES

[1] D. of Economic and S. A. P. Division, "World Population Ageing 2015," United Nations, NewYork, Tech. Rep., 11 2015.

[2] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults," *IEEE journal of biomedical and health informatics*, vol. 17, no. 3, pp. 579–590, 2013.

[3] R. Li, B. Lu, and K. D. McDonald-Maier, "Cognitive assisted living ambient system: A survey," *Digital Communications and Networks*, vol. 1, no. 4, pp. 229–252, 2015.

[4] H. Sun, V. De Florio, N. Gui, and C. Blondia, "The missing ones: Key ingredients towards effective ambient assisted living systems," *Journal of ambient intelligence and smart environments*, vol. 2, no. 2, pp. 109–120, 2010.

[5] A. Kunnappilly, C. Seceleanu, and M. Lindén, "Do we need an integrated framework for ambient assisted living?" in *Ubiquitous Computing and Ambient Intelligence: 10th International Conference, UCAmI 2016, San Bartolomé de Tirajana, Gran Canaria, Spain, November 29–December 2, 2016, Part II 10*. Springer, 2016, pp. 52–63.

[6] P. H. Feiler, B. Lewis, S. Vestal, and E. Colbert, "An overview of the sae architecture analysis & design language (aadl) standard: a basis for model-based architecture-driven embedded systems engineering," in *Architecture Description Languages*. Springer, 2005, pp. 3–15.

[7] M. U. Ahmed, M. Björkman, and M. Lindén, "A generic system-level framework for self-serve health monitoring system through internet of things (iot)," *Studies in health technology and informatics*, vol. 211, pp. 305–307, 2015.

[8] H.-M. Tzeng and C.-Y. Yin, "Nurses' response time to call lights and fall occurrences," *Medsurg Nursing*, vol. 19, no. 5, p. 266, 2010.

[9] D. I. Tapia, S. Rodrıguez, and J. M. Corchado, "A distributed ambient intelligence based multi-agent system for alzheimer health care," in *Pervasive Computing*. Springer, 2009, pp. 181–199.

[10] D. I. Tapia, S. Rodríguez, J. Bajo, and J. M. Corchado, "Fusion@, a soa-based multi-agent architecture," in *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*. Springer, 2009, pp. 99–107.

[11] J. Delange and P. Feiler, "Architecture fault modeling with the aadl error-model annex," in *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*. IEEE, 2014, pp. 361–368.