



**HAL**  
open science

## Analyzing Ambient Assisted Living Solutions: A Research Perspective

Ashalatha Kunnappilly, Axel Legay, Tiziana Margaria, Cristina Seceleanu,  
Bernhard Steffen, Louis-Marie Traonouez

► **To cite this version:**

Ashalatha Kunnappilly, Axel Legay, Tiziana Margaria, Cristina Seceleanu, Bernhard Steffen, et al.. Analyzing Ambient Assisted Living Solutions: A Research Perspective. 12th International Conference on Desig & Technology of Integrated Systems In Nanoscale Era (DTIS), Apr 2017, Palma de Mallorca, Spain. pp.1 - 7, 10.1109/DTIS.2017.7930168 . hal-01591347

**HAL Id: hal-01591347**

**<https://hal.science/hal-01591347>**

Submitted on 21 Sep 2017

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Analyzing Ambient Assisted Living Solutions: A Research Perspective

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

Axel Legay  
INRIA, Rennes, France  
axel.legay@inria.fr

Tiziana Margaria  
Univ. Limerick and Lero, Limerick, Ireland  
tiziana.margaria@lero.ie

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

Bernhard Steffen  
TU Dortmund, Germany  
steffen@cs.tu-dortmund.de

Louis-Marie Traonouez  
INRIA, Rennes, France  
Louis-marie.traonouez@inria.fr

**Abstract**—Typical AAL solutions rely on integrating capabilities for health monitoring, fall detection, communication and social inclusion, supervised physical exercises, vocal interfaces, robotic platforms etc. Ensuring the safe function and quality of service with respect to various extra-functional requirements like timing and security of such AAL solutions is of highest importance. To facilitate analysis, latest system development platforms provide underlying infrastructures for model-driven design (e.g., via the DIME tool), timing and resource-usage specification (e.g., via the REMES tool), security features (e.g., by employing SECube), and statistical model-checking techniques (e.g., via Plasma).

In this paper, we discuss the challenges associated with analyzing complex AAL solutions, from relevant properties to semantic interoperability issues raised by employing various frameworks for modeling and analysis, and applicability to evolving architectures. We take as examples two of the prominent existing AAL architectures and our own prior experience.

## I. INTRODUCTION

The worldwide trend towards an ageing demographic is globally making the Ambient Assisted Living (AAL) domain a research focus. A worthy AAL solution should offer many functions, e.g., health monitoring, fall detection, social inclusion and connectivity, physical exercise monitoring, physiotherapy support, home monitoring, robotic platform support etc. Apart from these functional attributes, many extra-functional attributes like timeliness, security, resource usage, reliability, etc. are equally important to ensure the success of AAL solutions.

Modern AAL systems are increasingly complex, with many modules that cater for different functionalities and should act in a highly dynamic environment, at the core of IT Connected Health platforms ?? A patchwork of partial solutions across providers has been so far adopted mostly in projects, but never really at a large scale, but the first workable solutions are starting to emerge. Philips' newly launched HealthSuite platform, for example, is an open, cloud-based platform that collects, compiles and analyzes clinical and other data from a wide range of devices and sources, based on open APIs. Its computational core, CareCatalyst<sup>1</sup>, enhances collaborative

<sup>1</sup><http://www.usa.philips.com/healthcare/innovation/about-health-suite/carecatalyst>

care between healthcare consumers and healthcare providers by collecting and connecting data across the health ecosystem including PHR, Consumer and Medical Devices, and EMRs.

Ensuring correctness, security, safety, and proper real-time behavior of AAL systems is of utmost importance as they need to tackle many critical scenarios that can even result in life loss if mismanaged. For instance, fall incidents of elderly people are critical indicators also of other conditions, and may even lead to death if proper aid is not given within a prescribed time and proper follow-up is not provided. In case of a fall event, the AAL system should raise a fall alarm and inform the care givers, potentially in an escalation chain, to guarantee that adequate care will be given in due time.

In our research experience, architectures, development frameworks, and tools play a significant role in the application of our research to AAL projects. In this paper, we reconsider our experience so far, including diverse prominent examples of AAL architectures (Section II), system design and development platforms (Section III), and analysis and verification tools (Section IV). As we employ a wide variety of tools, mostly in subsets according to our expertise, we observe that their interoperability by syntactic and semantic compatibility is essential for performing proper verification of critical functionalities. In Section V we provide a conceptual sketch for the integration of the design and analysis methods and tools, to exploit synergies and complementarities. In this way, we wish to provide a holistic and well-rounded design and analysis platform that uses the tools listed in the following sections, and ensures the needed syntactic and semantic compatibility.

## II. AAL ARCHITECTURES FOR ANALYSIS

The many challenges associated with the analysis of AAL solutions span the scope of *model-driven design* (MDD) (e.g., via the DIME [1] tool), timing and resource-usage specification (e.g., via REMES language [2]), security features (e.g., using SECube [3]), and statistical model-checking techniques (e.g., via Plasma [4]). To facilitate the discussion, we refer to two concrete AAL architectures, a *cloud-based* one and a *service-oriented* one, both sharing a set of *functional* and *extra-functional* aspects (e.g., timing, resource-usage, security

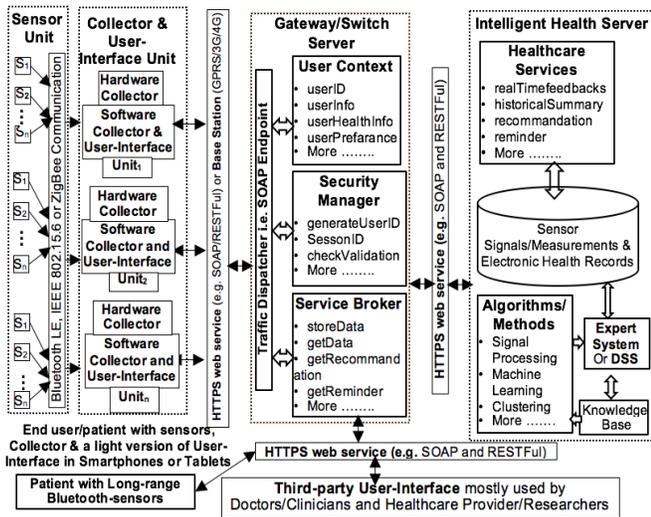


Fig. 1. ESS-H: a cloud-based architecture

and privacy etc.) that need to be analyzed. Since the design of adequate AAL architectures is based on user feedback, an MDD approach that involves users in the design and development loop would be extremely beneficial. However, the different paradigms used in the two architectures may lead to differences with respect to meeting quality requirements.

a) **ESS-H: a cloud-based architecture:** The ESS-H (Embedded Sensor Systems for Health) architecture [?] is shown in Fig. 1. Its major components are the *collector* and *user-interface* unit, the *gateway and switch server*, and the *intelligent health server*, the last two being cloud based. The ESS-H is a *centralized* solution, as the intelligent control is embedded within the intelligent health server that decides the actions to take in a centralized manner.

b) **FUSION: a service-oriented architecture:** The distributed multi-agent-based system developed for supporting people affected by the Alzheimer disease shown in Fig. 2 is based on a *Flexible User and Service-Oriented multi-agent Architecture (FUSION@)* [5], with agents based on deliberative Belief-Desire-Intention (BDI) paradigm. Applications and services communicate with the agent platform using the SOAP protocol, and the inter-agent communication happens through FIPA ACL. A distributed artificial intelligence (AI) unit is implemented through *Case-Based Reasoning* and *Case-Based Planning* techniques. The implementation framework is JADE.

### A. Analysis and comparison

The design aspects that deserve consideration and proper analysis for both architectures are as follows.

a) **Resource usage:** In the centralized ESS-H cloud-based solution, the Decision Support System associated with the cloud server takes the intelligent decisions. In contrast, the service-oriented architecture foresees multiple agent systems, with all the agents collaborating to deliver the required intelligent decisions. In both cases, *resource-usage* analysis

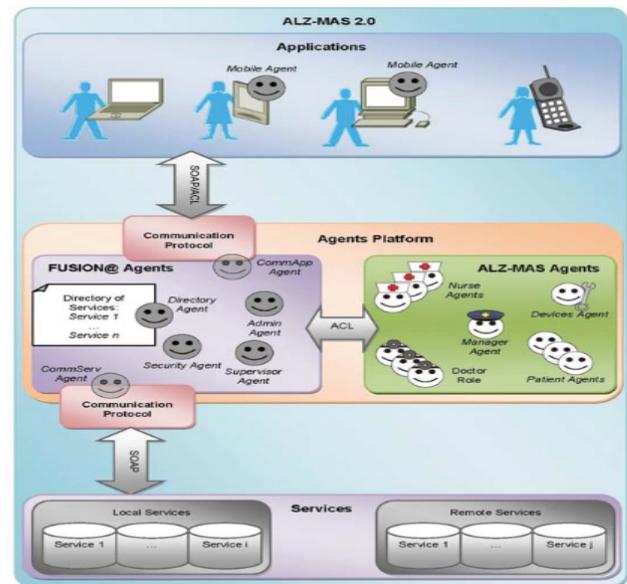


Fig. 2. FUSION: a service oriented MAS architecture for Alzheimer care

is relevant for the design and implementation stages. For instance, in the distributed case each agent is basically a processor with its own *memory*, so a *feasibility* analysis should ensure that the required memory for each agent does not exceed the provided one. Similarly, a proper load balancing when distributing computing tasks among agents requires a *CPU usage* analysis of each agent.

The communication protocol choice is a further crucial dimension on which the success of any AAL system largely depends. While the cloud-based AAL solution leaves space for a flexible choice, for the distributed MAS-based architecture the inter-agent communication protocol choice is restricted to Agent Communication Languages (ACL), hence *bandwidth* can become a constraint if large amounts of data need to be transmitted between agents.

b) **Security:** Security properties, e.g., protecting sensitive medical data or ensuring authorized access to data by a third party, are essential extra-functional features for any AAL solution. Since the information flow in the cloud-based AAL solution is centralized, one can intuitively argue that it is easier to encrypt and protect the data from unauthorized third party attacks than in the agent-based solution: there, each agent should incorporate a security aspect of its own in order to be safe in cases of third party attacks.

c) **Real-time properties:** An AAL system should guarantee known real-time performance even in a highly dynamic environment. Many functionalities, like the health parameters variations, falls etc., need immediate assistance from care givers. They are thus hard real-time, that is, the respective deadlines and other timing constraints must be met. In the distributed agent-based system, communication and synchronization between the agents adds time overheads when taking a decision after a critical event, therefore real-time property

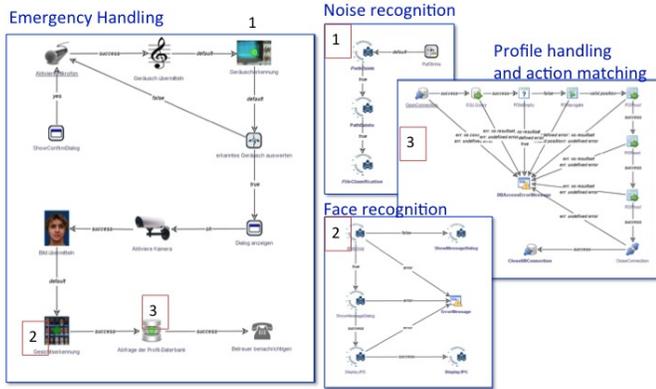


Fig. 3. AAL noise and face recognition system: Top model (left), submodels (right)

analysis is a must to still guarantee timely operation.

d) **Fault tolerance and Reliability:** Fault tolerance has a strict connection with reliability that refers to how long the system can deliver its desired functionality successfully. The centralized cloud-based AAL architecture has a higher single point of failure risk, hence essential units need to be replicated in order to ensure reliability. The inherently distributed MAS architecture should be less exposed to fault tolerance risks due to the distribution of functionalities.

Modern AAL architectures are highly complex systems and provide continuous multi-functionality support that is time and quality critical. To design, represent, analyze, and then guarantee the various aspects of functions and quality of service outlined above, we envision the use of a multitude of tools specialized to support design and analysis for such properties. This tool landscape however needs to work coherently, and be semantically interoperable.

In the following section we present the design platform that one could consider for developing AAL solutions with the user-in-the-loop, to which various existing tools can be added to achieve the analysis along the various dimensions mentioned above.

### III. DESIGN PLATFORMS: DESIGN AND ANALYSIS WITH LIVING MODELS

In the modern connected world, agile and prototype-driven design is rapidly emerging as the paradigm of choice for the co-creation of applications and systems that really serve the needs of the users and healthcare practitioners. In contrast to the traditional software development process, collaborative approaches that include AAL users and professionals in the co-production of executable models do not start with a lengthy analysis to produce textual specifications, distinct and detached from the design and implementation. Agile design platforms allow action-based design from inception,

involving the user/application expert continuously throughout the whole systems' life-cycle. Developing systems with the eXtreme Model-Driven Development (XMDD) paradigm [6], [7], for example, adopts a user-in-the-loop and expert-in-the-loop model-driven philosophy that works by successively enriching and refining a single artefact that is a rich multi-aspect and multi-faceted formal and executable model, as in the One-Thing Approach (OTA) [8].

Fig. 3 shows the service logic of a noise and face recognition application for the AAL Smart Home showcased at the CeBIT 2009 and IFA 2009. This lightweight nighttime surveillance system has been designed in collaboration with a residential care home for dementia patients. In this design approach application models are at the center of the design activity and first class entities of the global system design process.

**Domain specific libraries** of models establish a design language familiar to both IT and non-IT stakeholders, where building blocks are (elementary) units of behavior rather than software components. In this example, the top level model handles emergencies by 1) analyzing and classifying the noises that microphones capture in the surveilled rooms, detecting anomalies, and 2) switching on a camera and notifying the nurse in charge, in case of peril or inconclusive classification. To recognize the people in the room, it 3) compares the image to a database of known potential individuals (patients, staff, family members) and sends an appropriate message or alarm to the nurse in charge.

Systems are specified by **model assembly**. Here we use **orchestration** in each model, **hierarchy** for behavioral refinement, and **configuration** as composition techniques. The top level model includes in this example three submodels, one for the noise recognition, one for the face recognition, and one for the situation profile handling and action matching (i.e., what to do in which case).

Knowledge and requirements are expressed as **properties**, via constraints formulated in an automatically verifiable fashion. Actually, some of the constraints happen to be domain-independent, and already taken care of at design time by the jABC or more recently DIME [1] design environment. Here, this covers both the functional correctness of each model element (Action), but also the patterns of usage inside processes and workflows, i.e., behavioral constraints expressed in temporal logics (typically CTL and LTL) and verified by model checking.

As our models are immediately executable, first in animation mode that proposes a walk through the system, then with real code (for simulation or for implementation), they are enactable from the very beginning. Hence the "living models" name [8] that distinguishes them from the usual software design models, which are purely descriptive and illustrative, and do not provide immediate feedback on their own. In most cases, such models get refined in this style until the atomic actions get implemented, in source code or reusing previous assets like a database, components via an API, or services. In this case, there is no inherent design/implementation gap between the initial prototype and the final product: the finished

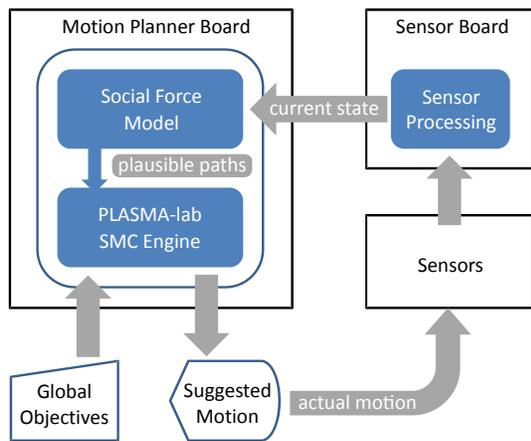


Fig. 4. Architecture of the DALi motion planner with Plasma Lab

running system is co-created incrementally along the design process, and grows from the model through prototypes into the fully implemented and running system.

To design the system behavior, we use domain-appropriate design tools like DIME [1], a Cinco [9] product that is adequate for the design of web-based applications, as most AAL applications are. The XMDD design approach is however so far architecture-agnostic, and its property support is geared towards workflow-style properties easily expressible in Computation Tree Logic (CTL) for model checking with the GEAR [10] game-based model checker for modal mu-calculus, and in Linear Temporal Logic (LTL) for workflow (i.e., subprocess) synthesis with the PROPHETS plugin [11]. Timing properties, resources, and stochastic behavior could and should be added by a proper integration with REMES IDE [12], an environment that provides automated support to the resource-aware modeling and analysis in the REMES [2] language, UPPAAL model checker [13], and Plasma tools [4]. Also, architecture-awareness might be a useful addition, especially in case of distribution, which often leads to *feature interaction* problems. We have seen in the past how incremental formalization [14] turned out beneficial in managing interferences in telecommunication platforms[15]. We expect the connected healthcare aspect of AAL to share many traits with those applications, thus profiting from that experience in evolution-friendly design.

#### IV. STATISTICAL MODEL CHECKING AND AAL

In the context of motion planning for assisted living [16], [17], the Plasma Lab platform for Statistical Model checking (SMC) was integrated with robotic devices in the DALi and ACANTO EU projects. There, a novel online motion planning application of SMC helps those with impaired ability to negotiate complex crowded environments like shopping malls and museums. While DALi is focused on helping a single user reach a number of specific locations, ACANTO concerns therapeutic activities of groups of users, where group cohesion, social interaction, and exercise are the metrics of interest.

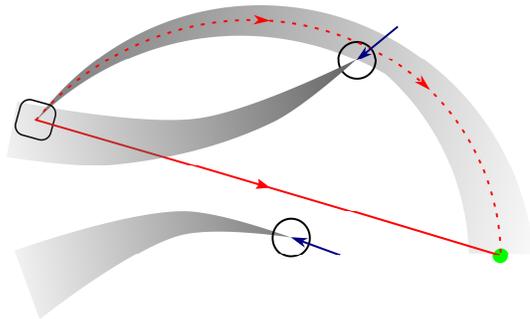


Fig. 5. Operation of the DALi motion planner

In the basic system architecture shown in Fig. 4, sensors like fixed cameras and cameras on robotic devices locate fixed and moving objects in the environment. A predictive stochastic model of human motion (the “social force model”) constructed from this information is used to generate plausible future trajectories of all the detected moving agents, given initial deviations from their current trajectories. After hypothesizing different initial directions, Plasma Lab estimates the probability that future trajectories will satisfy path constraints and objectives expressed in temporal logic. The best deviation is suggested to the user.

In Fig.5, a user (the rectangle) walking to the next local waypoint (green dot) in straight line (in red) would with high probability collide with other pedestrians (circles), whose position and velocity are indicated by vectors. By making a deviation to the user’s trajectory (dashed red line), Plasma Lab predicts that the pedestrians will avoid each other with high probability (shaded areas).

The planner can include additional constraints like desired zones for the pedestrian (e.g., “keep within 5 minutes walk from a restroom”), undesired zones to avoid, and anomalies like temporary obstacles over the path. The global planner starts from a pre-calculated global plan that visits the user’s objectives in an a priori optimal way, considering all things known in advance, and calculates a local way point  $w$  as the user’s point of greatest straight line progress along the global plan within the sensor range.

Then, the motion planner assumes the user will follow the global plan, but needs to temporarily deviate to avoid collisions. A short term planning algorithm uses SMC to suggest a deviation to the user’s direct path.

##### A. Plasma Lab: a Statistical Model Checking platform

SMC is based on the notion that sample runs of a stochastic system are drawn according to the distribution defined by the system, and can therefore be used to estimate the probability measure on executions. The properties that we handle by such approach include BLTL [18], a bounded version of LTL.

PLASMA is a compact, efficient and flexible SMC platform that offers a series of SMC algorithms, including classical SMC algorithms and specialized ones for rare events. Being a platform, PLASMA is designed for API-based integration of external simulators, input languages, and SMC algorithms.

This ability reduces the effort of integrating new algorithms and allows us to create direct plugin interfaces with industry-used specification tools, without using extra compilers, for example with Simulink.

## V. AN ENCOMPASSING SYNERGETIC APPROACH

As just described, many dimensions of analysis and verification cooperate to ensure the high quality of AAL solutions. At each level, the one or other platform, approach, or tool offers means to express, check, or enforce some characteristics of the system under design that are crucial to a high quality AAL solution. Our proposal is to join forces and combine the strengths of each approach into an integrated design and verification platform with strong holistic description and validation capabilities. For each dimension, we sketch the possible synergies and how we envisage to achieve them.

*a) Design and Functional Correctness:* We can analyze functional properties by model checking behavioral models like, e.g., in DIME, and synthesize functionally correct sub-models with PROPHETS. We use CTL and LTL, respectively, directly on our behavioral models. At the metamodel level, we can formulate correctness criteria in Cinco and ensure that the tool generated from that Cinco model (e.g., DIME) will automatically enforce model compliance to them.

Probabilistic properties can be covered via SMC through Plasma Lab [4], most likely using it as a plugin to a design environment like DIME or REMES. In fact, the PLASMA API and plugin-based architecture makes it easy to add new simulators, checkers, or algorithm components. This is compatible with both the Cinco tool generation philosophy and the DIME design environment: the PLASMA GUI is itself created using the Cinco metamodeling environment [19].

Functionalities can also be mapped to REMES models, achieving a design environment that coherently integrates at the model level the different aspects to describe, and then validates and tests the systems underlying the AAL solutions.

*b) Security, Privacy, Confidentiality:* Security is layered inside the activities, at the model level, and at the global level including the run-time environment.

Equipping the system with security features via the security Domain Specific Language and security processes [20] offered by the SEcube security design platform [3] is an attractive possibility, especially as SEcube is already integrated with DIME. We can map also privacy and confidentiality to general temporal logic properties once we have a role model (e.g. as RBAC) [1] and a characterization of the security primitives. If the properties are probabilistic, the SMC capabilities can again be used.

The entire SEcube<sup>TM</sup> platform is ready to support security-agnostic application designers in their need to add security aspects to their models, by leveraging predefined abstract security primitives, which they might theoretically not even know nor understand in detail, as in [21]. For these reasons, DIME includes by design the support of properties and model manipulations that are foundational for the OTA-based XMDD. DIME focuses on application experts, who

are typically non programmers, and its versatility is a key characteristic.

*c) Performance and Real-time properties:* The hard real-time constraints of AAL solutions can be analyzed in our integrated framework by model checkers like UPPAAL [13], a state-of-the-art tool for verifying real-time systems. End-to-end deadlines, response times and synchronization constraints can be encoded as (timed) CTL properties and model checked with UPPAAL, assuming the model of the system as a network of timed automata.

PLASMA can be used through its GUI, but also via the command line or embedded in other software as a library. The PLASMA GUI, itself created using the Cinco metamodeling environment [19], supports its use as a standalone SMC with multiple ‘drop-in’ modeling languages, and provides an SMC engine and a source template to create custom simulator classes. A plugin system makes adding a simulator, a checker or an algorithm component pretty straightforward. To benefit from massive distribution of the simulations, the PLASMA API provides generic methods to define distribution algorithms. We have used these functionalities to distribute large number of simulations over a computer grid <sup>2</sup>.

REMES is already integrated with UPPAAL, and the latter can also be added to DIME, and easily interfaced with Plasma via a dedicated plugin.

*d) Resource Analysis:* Resource guarantees and optimizations are kept as much as possible distinct from design issues, in order to maintain information on the structure and the design decisions independent of the considerations that lead to a particular optimized implementation.

Resource-related properties of AAL solutions, like CPU, bandwidth and memory usage, can be expressed and analyzed in REMES (REsource Model for Embedded Systems) [2], [12], which is a resource-aware behavioral language of interacting components, called modes, which communicate with one another and the environment via shared variables. Each of the considered abstract resources, that is, memory, CPU, energy, bandwidth, have a dedicated type in the language: mem, CPU, eng, bdw, respectively. The semantics of REMES modes is given in terms of (priced) timed automata, so feasibility analysis as well as optimal and worst-case resource usage of various AAL solutions can be checked as (weighted) computation tree logic properties with the UPPAAL suite.

In the future, one could equip the DIME models with the ability to express resource models and characteristics as in the REMES model, in order to also encompass resource-aware behavior, and be able to reason about possible trade offs between quality-of-service attributes of AAL solutions.

*e) Fault tolerance and Reliability:* The reliability of an AAL solution for a specified period of time under specified environmental conditions can be modeled probabilistically in Plasma or UPPAAL SMC (Statistical Model Checker) [22], and the probability of successful operation can be checked by

<sup>2</sup><https://project.inria.fr/plasma-lab/documentation/tutorial/igrida-experimentation/>

hypothesis testing with Plasma for untimed models, and with UPPAAL SMC for timed models, or estimated via probability evaluation.

UPPAAL SMC can be interfaced with Plasma for statistical analysis of complex timed models, and with REMES to provide statistical model checking of resource-related properties. Replicated AAL models can be abstracted into networks of stochastic timed/hybrid automata that can be statistically model checked to analyze fault tolerance.

**f) Evolution and Tools Interoperability:** Changes in models and properties are inevitable for dynamic AAL systems, to which a new health service can be added at a later time, or hardware can be replaced depending on the user's needs. Such changes need to be supported by a potential integrated framework for modeling and analyzing AAL systems.

In DIME, system changes (e.g., upgrades, customer-specific adaptations, new versions) occur only, or at least primarily, at the model level, with a subsequent global re-verification, and re-compilation (or re-synthesis, in the future).

Evolution can be handled in REMES by depicting intra- and inter-component dependencies via dependency analysis [23] that traces the impact of some modification in the model. Smooth changes are facilitated by tools interoperability, which means that we can exchange model artefacts between tools.

Moreover, in order to ensure semantic compatibility between the involved tools (DIME, SECube, REMES, UPPAAL and Plasma), that is, guarantee that a model preserves the original execution semantics from one tool to the other, we might need to define a “semantic translator” that implements the model-to-model translations based on a mapping metamodel, either in a constructive way as in Cinco or in a transformational manner [24].

## REFERENCES

- [1] Steve Boßelmann, Johannes Neubauer, Stefan Naujokat, and Bernhard Steffen. Model-Driven Design of Secure High Assurance Systems: An Introduction to the Open Platform from the User Perspective. In T.Margaria and Ashu M.G.Solo, editors, *The 2016 International Conference on Security and Management (SAM 2016). Special Track "End-to-end Security and Cybersecurity: from the Hardware to Application"*, pages 145–151. CREA Press, 2016.
- [2] Cristina Secoleanu, Aneta Vulgarakis Feljan, and Paul Pettersson. Remes: A resource model for embedded systems. In *14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2009)*, pages 84–94. IEEE Computer Society, 2009.
- [3] Antonio Varriale, Giorgio di Natale, Paolo Prinetto, Bernhard Steffen, and Tiziana Margaria. SECube<sup>TM</sup>: An open security platform: General Approach and Strategies. In T.Margaria and Ashu M.G.Solo, editors, *The 2016 International Conference on Security and Management (SAM 2016). Special Track "End-to-end Security and Cybersecurity: from the Hardware to Application"*, pages 131–137. CREA Press, 2016.
- [4] Benoît Boyer, Kevin Corre, Axel Legay, and Sean Sedwards. PLASMA-lab: A Flexible, Distributable Statistical Model Checking Library. In *Proceedings of QEST*, volume 8054 of LNCS, pages 160–164. Springer, 2013.
- [5] Dante I Tapia, Sara Rodriguez, and Juan M Corchado. A distributed ambient intelligence based multi-agent system for alzheimer health care. In *Pervasive Computing*, pages 181–199. Springer, 2009.
- [6] Tiziana Margaria and Bernhard Steffen. Agile IT: Thinking in User-Centric Models. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation*, volume 17 of *Communications in Computer and Information Science*, pages 490–502. Springer Berlin / Heidelberg, 2009.
- [7] Tiziana Margaria and Bernhard Steffen. Service-Oriented: Conquering Complexity with XMDD. In Mike Hinchey and Lorcan Coyle, editors, *Conquering Complexity*, pages 217–236. Springer London, 2012.
- [8] Tiziana Margaria and Bernhard Steffen. Business Process Modelling in the jABC: The One-Thing-Approach. In Jorge Cardoso and Wil van der Aalst, editors, *Handbook of Research on Business Process Modeling*. IGI Global, 2009.
- [9] Stefan Naujokat, Michael Lybecait, Dawid Kopetzki, and Bernhard Steffen. CINCO: A Simplicity-Driven Approach to Full Generation of Domain-Specific Graphical Modeling Tools. 2016. to appear.
- [10] Marco Bakera, Tiziana Margaria, Clemens Renner, and Bernhard Steffen. Tool-supported enhancement of diagnosis in model-driven verification. *Innovations in Systems and Software Engineering*, 5:211–228, 2009.
- [11] Anna-Lena Lamprecht, Stefan Naujokat, Tiziana Margaria, and Bernhard Steffen. Synthesis-Based Loose Programming. In *Proc. of the 7th Int. Conf. on the Quality of Information and Communications Technology (QUATIC 2010)*, Porto, Portugal, pages 262–267. IEEE, September 2010.
- [12] Dinko Ivanov, Marin Orlić, Cristina Secoleanu, and Aneta Vulgarakis. Remes tool-chain: A set of integrated tools for behavioral modeling and analysis of embedded systems. In *Proc. of the IEEE/ACM International Conference on Automated Software Engineering, ASE '10*, pages 361–362. ACM, 2010.
- [13] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1):134–152, 1997.
- [14] Bernhard Steffen, Tiziana Margaria, Andreas Claßen, and Volker Braun. Incremental Formalization: A Key to Industrial Success. *Software - Concepts and Tools*, 17(2):78–95, 1996.
- [15] Bengt Jonsson, Tiziana Margaria, Gustaf Naeser, Jan Nyström, and Bernhard Steffen. Incremental requirement specification for evolving systems. *Nordic J. of Computing*, 8:65–87, March 2001.
- [16] Alessandro Colombo, Daniele Fontanelli, Axel Legay, Luigi Palopoli, and Sean Sedwards. Motion planning in crowds using statistical model checking to enhance the social force model. In *IEEE Conference on Decision and Control (CDC)*, pages 3602–3608, 2013.
- [17] Alessio Colombo, Daniele Fontanelli, Axel Legay, Luigi Palopoli, and Sean Sedwards. Efficient customisable dynamic motion planning for assistive robots in complex human environments. *Journal of ambient intelligence and smart environments*, 7:617–633, 2015.
- [18] P. Zuliani, C. Baier, and E. Clarke. Rare-event verification for stochastic hybrid systems. In *Hybrid Systems: Computation and Control (part of CPS Week 2012), HSCC'12, Beijing, China*, pages 217–226. ACM, 2012.
- [19] Stefan Naujokat, Louis-Marie Traonouez, Malte Isberner, Bernhard Steffen, and Axel Legay. Domain-Specific Code Generator Modeling: A Case Study for Multi-faceted Concurrent Systems. In *Proc. of the 6th Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation, Part I (ISoLA 2014)*, volume 8802 of LNCS, pages 463–480. Springer, 2014.
- [20] Giorgio di Natale, Alberto Carelli, Pascal Trotta, and Tiziana Margaria. Model driven design of crypto primitives and processes. In T.Margaria and Ashu M.G.Solo, editors, *The 2016 International Conference on Security and Management (SAM 2016). Special Track "End-to-end Security and Cybersecurity: from the Hardware to Application"*, pages 152–158. CREA Press, 2016.
- [21] Tiziana Margaria, Bernhard Steffen, and Manfred Reitenspieß. Service-Oriented Design: The Roots. In *Proc. of the 3rd Int. Conf. on Service-Oriented Computing (ICSOC 2005)*, Amsterdam, The Netherlands, volume 3826 of LNCS, pages 450–464. Springer, 2005.
- [22] Alexandre David, K.G. Larsen, A. Legay, M. Mikućionis, and D.B. Poulsen. Uppaal smc tutorial. *STTT Journal*, 17(4):397–415, 2015.
- [23] Raluca Marinescu, Saad Mubeen, and Cristina Secoleanu. Pruning architectural models of automotive embedded systems via dependency analysis. In *Proc. of IEEE Euromicro SEAA Conference (SEAA 2016, to appear)*. IEEE Computer Society, 2016.
- [24] Gabor Karsai, Andras Lang, and Sandeep Neema. Design Patterns for Open Tool Integration. *Software and Systems Modeling*, 4(2):157–170, 2005.