Different approaches for testing body sensor network applications

Citation for the original published paper (version of record):
Silva, S., Diniz Caldas, R., Pelliccione, P. et al (2025). Different approaches for testing body sensor network applications. Journal of Systems and Software, 223.
http://dx.doi.org/10.1016/j.jss.2025.112336

The Journal of Systems and Software 223 (2025) 112336

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Different approaches for testing body sensor network applications ☆

Samira Silva [a,*], Ricardo Caldas [b], Patrizio Pelliccione [a], Antonia Bertolino [a,c]

[a] Gran Sasso Science Institute, L'Aquila, Italy
[b] Chalmers University of Technology, SE 417 56, Gothenburg, Sweden
[c] ISTI-CNR, Via Moruzzi, 1, Pisa, Italy

ARTICLE INFO

Dataset link: https://github.com/samirasilva/Paper_JSS

Keywords:
Body sensor networks
Model-based testing
Combinatorial testing
Discrete-time Markov chain

ABSTRACT

Body Sensor Networks (BSNs) offer a cost-effective way to monitor patients' health and detect potential risks. Despite the growing interest attracted by BSNs, there is a lack of testing approaches for them. Testing a Body Sensor Network (BSN) is challenging due to its evolving nature, the complexity of sensor scenarios and their fusion, the potential necessity of third-party testing for certification, and the need to prioritize critical failures given limited resources. This paper addresses these challenges by proposing three BSN testing approaches: PASTA, ValComb, and TransCov. These approaches share common characteristics, which are described through a general framework called GATE4BSN. PASTA simulates patients with sensors and models sensor trends using a Discrete Time Markov Chain (DTMC). ValComb explores various health conditions by considering all sensor risk level combinations, while TransCov ensures full coverage of DTMC transitions. We empirically evaluate these approaches, comparing them with a baseline approach in terms of failure detection. The results demonstrate that PASTA, ValComb, and TransCov uncover previously undetected failures in an open-source BSN and outperform the baseline approach. Statistical analysis reveals that PASTA is the most effective, while ValComb is 76 times faster than PASTA and nearly as effective.

1. Introduction

Wireless Sensor Networks (WSNs) are spatially dispersed networks of sensors that track their surroundings and send the information they gather to a central processor. Body Sensor Networks (BSNs) are a particular kind of WSN that resulted from a recent push on wearable biosensors technology development (Gravina and Fortino, 2020; Guk et al., 2019). BSNs attract much interest, mostly from the healthcare industry (Alrige and Chatterjee, 2015; Aziz et al., 2006; Lee and Lee, 2015), but also from the military-industrial sector (Tatbul et al., 2004), sports and entertainment (Conroy et al., 2009; Pansiot et al., 2010; Burchfield and Venkatesan, 2010), and the social public field (Osmani et al., 2007; Wai et al., 2010).

BSNs have also steadily emerged as a hotbed of research. Developing comprehensive BSN applications requires an interconnection between physical signals with cybernetic devices, middleware components, and data fusion algorithms. In the medical domain, for example, vital signals (such as heartbeat rate and blood pressure) are captured by a variety of physiological parameter sensors positioned in or around the human body, on the body surface (Lai et al., 2013), or even by mobile devices, including smartwatches or smartphones. The collected data are then processed in real-time and are finally distributed to e.g., caregivers, emergency personnel, or medical servers that may assist the patient.

In the current literature, though, there is a shortage of proper Software Engineering (SE) approaches focused on WSN applications (Picco, 2010), or even more specifically on BSNs. Besides, notwithstanding the extensive body of research on testing in SE, relatively little of this knowledge has been applied to WSNs or BSNs.

Concerning BSNs testing, since the application consists of elaborating the vital signals as inputs that come from different sensors, testers need to consider different combinations of values that could be read. This scenario seems to be the typical case for using Combinatorial Testing (Nie and Leung, 2011). However, combinatorial testing provides a static approach to test case generation, whereas BSNs work by monitoring and analyzing some vital measures that are captured over time. In particular, the sensors could at any moment detect some alarming values and react accordingly, e.g., by issuing an alarm system. Hence, we would like to continue testing dynamically a BSN over differing combinations of parameters values. Testing a BSN also involves other challenges, e.g.: (i) ensuring the proper testing of data fusion, (ii) since

☆ Editor: Yan Cai.

* Corresponding author.
E-mail addresses: samira.silva@gssi.it (S. Silva), ricardo.caldas@chalmers.se (R. Caldas), patrizio.pelliccione@gssi.it (P. Pelliccione), antonia.bertolino@isti.cnr.it (A. Bertolino).

https://doi.org/10.1016/j.jss.2025.112336
Received 17 June 2024; Received in revised form 13 November 2024; Accepted 8 January 2025
Available online 28 January 2025

third-party testing could be required for testing at system level, e.g., to certify the compliance to medical ISO standards, testing should be feasible also without code accessibility, and (iii) given potential limited physical and human resources, testing should also prioritize the most critical failures.

In this work, we fill this gap by proposing three different testing approaches to test a BSN application:

- *PASTA* (PAtient Simulation for Testing of bsn Applications),
- *ValComb* (Sensor Values Combination), and
- *TransCov* (Sensor Transitions Coverage).

Since these approaches share many characteristics, for presentation clarity, we first introduce a generic approach, which we refer to with the shortcut *GATE4BSN* (Generic Approach to TEsting for BSN applications). *GATE4BSN* is then specialized into the three concrete methods mentioned above. In addition to enhancing readability, the definition of the generic approach also brings the benefit of allowing other researchers to build new approaches by proposing new instantiations of *GATE4BSN*.

GATE4BSN consists of the abstract steps to be taken for testing a BSN. The first of which is collecting a set of sensor data, to be used as the test inputs. Then, we identify different criteria that could be used to collect such data, by applying the basic notion of combinatorial testing and also by leveraging a model of the sensors.

PASTA is a model-based testing approach that mixes Combinatorial Testing and Markov Chains-based simulation to test a BSN in a novel way. Specifically, we use a Discrete Time Markov Chain (DTMC) to simulate the trend of each sensor, which is determined by its transition probabilities. These probabilities are defined by applying combinatorial testing to the set of sensors that are monitored by the BSN under test. *ValComb* makes use of all possible combinations of sensor risk levels to explore different behaviors of a patient. Finally, *TransCov* employs a dummy patient in which all the transitions in the DTMCs are labeled with equal probability. Then, their DTMCs are executed until all the transitions are covered.

The three proposed approaches are generic and could be applied to test any type of BSN application For the sake of evaluation, in this paper, we consider BSNs that are used in the health domain. Thus, for the purpose of testing, we simulate a patient by considering the vital signs read by a whole set of sensors, and a test run consists of monitoring the risk levels for the patient's health.

For evaluation, we used *PASTA*,*ValComb*, and *TransCov* to test a self-adaptive system from the literature, the SA-BSN (Self-Adaptive Body Sensor Network) (Gil et al., 2021). We assessed the Passing Test Case Rate (PTCR) and the Execution Time (ET) of each approach, and compared them one against the other and against a random baseline. Our approaches could detect failures in the system that has been in academic use for 2 years now, and all three of them were shown statistically to be more effective than the random baseline. *PASTA* is the most effective proposed approach, presenting a PTCR that is 5.38% higher than the baseline. In terms of time, the fastest proposed approach is *ValComb* which is approximately 29 times faster, and also more effective (PTCR 2.71% higher), than the baseline.

In summary, our work provides the following original contributions:

- We introduce *GATE4BSN* as a generic approach composed of the main abstract steps needed to test BSNs;
- We provide formal definitions for *GATE4BSN* and instantiate it in three different approaches: *PASTA*, *ValComb* and *TransCov* as potential alternative solutions;
- We include an extensive experimentation (including the three approaches, and replicating each experiment 25 times);
- We conduct a rigorous statistical comparison of the results (using both the Kruskal–Wallis test and the Vargha–Delaney A measure);
- We investigate the failures to understand whether they might be due to the same faults, by clustering the non-passing test cases from *PASTA* and *ValComb*;
- Last, but not least, we test a new fixed version of the SA-BSN.

In Section 2, we briefly overview background concepts; Section 3 describes our generic approach to test BSNs, the *GATE4BSN*; Section 4 describes the three instances of *GATE4BSN*, which are *PASTA*, *ValComb*, and *TransCov*; Section 5 presents the study setting; in Section 6 we discuss the results and threats to validity; in Section 7 we review the related literature; finally, in Section 8 we wrap up and hint at promising future research directions.

2. Background

In this section, we provide the concepts on which this work relies, i.e., Discrete Time Markov Chains (DTMCs) and Combinatorial Testing. Then, we also introduce what the testing of BSN entails.

2.1. Discrete time Markov chains

A Markov process refers to a stochastic process for which the future probabilistic behavior depends only on the present state and is not affected by how this state has been reached (Trivedi, 2016). When the state space is finite, the Markov process is said a Markov Chain, and if evolving in discrete steps (or, we observe the process at a discrete set of time points), it becomes a Discrete-Time Markov Chain (DTMC) (Trivedi, 2016, Ch.7). In this paper, since a BSN samples sensors readings at discrete time intervals, we use DTMCs to simulate the patient health status and depict for each sensor a DTMC as a transition system in which transitions are annotated with probabilities.

A DTMC is a tuple (S, P) where S is a set of states, and $P : S \times S \rightarrow [0, 1]$ is a function that given $s, s\prime \in S$ as input, calculates a probability p with $0 \leq p \leq 1$. Moreover, for each $s \in S$

$$\sum_{s\prime \in S} P(s, s\prime) = 1 \tag{1}$$

A finite DTMC, containing finite s ∈ S, is typically represented by a probability transition matrix (TM),

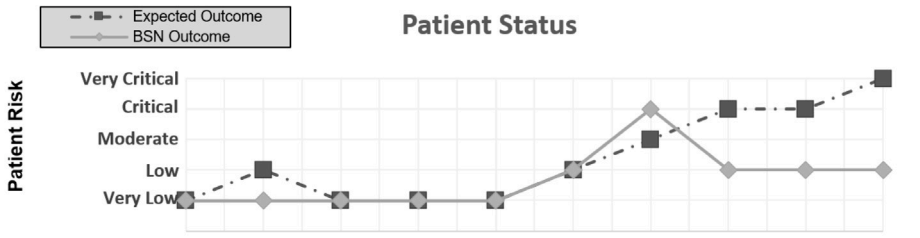
$$TM = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0m} \\ p_{10} & p_{11} & \cdots & p_{1m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m0} & p_{m1} & \cdots & p_{mm} \end{bmatrix}$$

where $p_{ij} = P(s_i, s_j)$. Each row $p_{i1}, \ldots, p_{im}$ in TM corresponds to the probabilities labeling all arcs that exit from state i, and according to Eq. (1) above, their sum is equal to 1.

The work in Whittaker and Thomason (1994) provides a method for statistical testing based on a Markov chain model of software usage. In their work, Markov chains have two important applications. First, compared to many other methods currently in use, it is more general since it enables test input sequences to be created from numerous probability distributions. Second, an additional Markov chain is created to capture the test history, including any observed failure information. The test input sequences that are formed from the chain and applied to the program constitute a stochastic model in and of themselves. Analytical calculations on this chain are used to evaluate the impact of the failures.

2.2. Combinatorial testing

As a BSN receives as input a set of values provided by independent sensors, a proper testing strategy should examine systematically the combinations of sensor values; this is the basic concept behind Combinatorial Testing (CT). CT is a well-known testing strategy providing a good balance between cost and effectiveness (Nie and Leung, 2011). To cover specific parameter value combinations, it samples the vast



Time	13:45	14:00	14:15	14:30	14:45	15:00	15:15	15:30	15:45	16:00
Heart beat rate [bpm]	90	85	92	88	87	100	99	114	120	120
Oxygenation [%]	99	98	99	98	98	98	97	95	93	88
Temperature [°C]	36.5	36.5	36.5	37	37	37.2	38.2	36.5	36.1	36.5

Fig. 1. Representation of the vital signs of a patient.

combination space using a condensed test suite. T-way CT is a strategy in this sense, relying on the empirically proven intuition that most failures are caused by a single parameter value or by interactions between a small number of parameters (Kuhn et al., 2010), because not every parameter affects every failure. Indeed, empirical studies showed that faults are generally caused by the interaction among at most six parameters (Kuhn et al., 2004).

A t-way test set satisfies the property that for any subset of t parameters, all possible combinations of their values are covered by at least one test in the test set. In particular, when t = 2 (which is commonly named *pairwise testing*), for any two input parameters, the test set includes at least one test case that covers each combination of their relevant values.

Various tools exist that automate test generation for CT. In our approach we adopted the Advanced Combinatorial Testing System (ACTS) (Yu et al., 2013), a well-known test generation tool for constructing t-way combinatorial test sets.

2.3. Testing body sensor networks

Testing BSNs is the activity of finding (test) cases that describe a set of vital signs (i.e., test inputs) provided by the sensors, which, once processed and fused, give a diagnostic of the patient's health status (i.e., test outcomes).

To exemplify an instance of the problem, Fig. 1 illustrates the Patient Risk levels over time for a fictitious patient from 13:45 to 16:00. The patient's status is characterized by a combination of streamed vital signs that vary over time (a.k.a. vital signals). Each signal is captured by a sensor designated to measure the specific sign in terms of key–value pairs. Keys are timestamps (e.g., Time = 13:45) and values are real numbers indicating the intensity of the signal (e.g., Heartbeat rate = 90 bpm). Thus, representing the behavior of a patient is a matter of describing a composite of streamed values of vital signs and how they evolve through time. The continuous line provides the actual diagnosis provided by the BSN, while the dashed line refers to the expected one.

Fig. 1 shows a strong decrease in blood oxygenation (≤95%) after 15:30. Accordingly, the expected outcome for the BSN would be to reach a very critical risk for the patient's health status. This oracle verdict is, e.g., the diagnosis given by the experts (doctors) based on the observed vital signs. However, the patient risk level diagnosed by the BSN under test (depicted as BSN Outcome) stays at low risk: such behavior evidences a failure.

3. Testing a body sensor network

In this section we present our generic approach, called *GATE4BSN*, to test body sensor network applications. It will be then instantiated in three different approaches (see Section 4). Before presenting *GATE4BSN* (Section 3.2), we make use of a running example, which is introduced in Section 3.1.

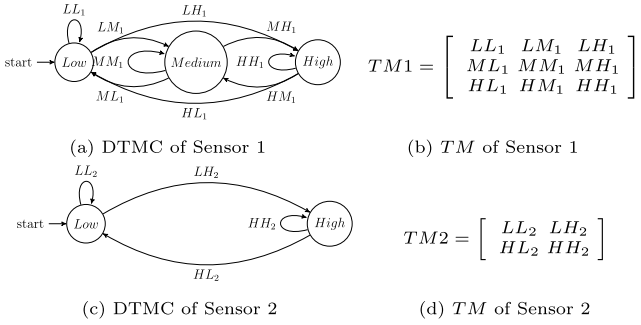


Fig. 2. Example of DTMCs and transition matrices (TMs) for Sensor 1 and Sensor 2.

3.1. Example of a simple BSN

As a running example, we consider a simple BSN containing two sensors, Sensor 1 and Sensor 2. We also suppose that the possible risk levels for Sensor 1 are *Low*, *Medium*, or *High-risk*, while Sensor 2 can range from *Low* to *High-risk*. The behavior of these sensors may be graphically described by DTMCs and their respective transition matrices, as shown in Figs. 2(a) and 2(c), and Figs. 2(b) and 2(d) for Sensor 1 and Sensor 2, respectively. The states in these DTMCs are labeled with the risk level they are representing and the transitions with the probability of a determined sensor moving from one risk level to another. In the BSN domain, matrix indexes represent risk levels, and cells represent the probability of transition from one risk level to another. For instance, in Fig. 2(b), $TM1_{3,2} = HM_1$ indicates that Sensor 1 varies from *High* to *Medium* risk with a probability equal to the value of HM_1.

The same risk levels for different sensors may also be associated with different risk value ranges. In this running example, to make it simple, let us suppose that: for Sensor 1, the risk levels *Low*, *Medium*, and *High* correspond to the risk value ranges [0,10], [11,20], and [21,30], respectively; for Sensor 2, the risk levels *Low* and *High* correspond to the risk value ranges [0,15] and [16,30], respectively. Thus, for each sensor, each risk level corresponds to a single risk value range.

The representation of the sensor behavior as a DTMC is employed in *TransCov* and *PASTA*, while for *ValComb*, there is no need to use DTMCs. The idea behind the representation of a patient through DTMCs is that patients more prone to diseases may have higher probability values in the arcs that enter the "High-risk" state. On the other hand, healthy patients may be represented by higher probabilities in the arcs to the "Low-risk" state.

3.2. A generic approach to TEsting for a BSN application (GATE4BSN)

Fig. 3 provides an overview of *GATE4BSN*, which takes 4 main stages. First, in stage ①, we either collect data from sensors that are

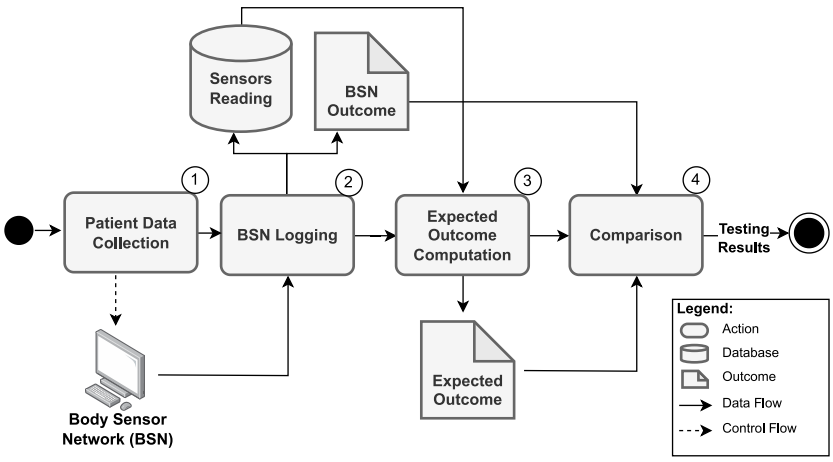


Fig. 3. Overview of the generic approach to TEsting for a BSN application (GATE4BSN).

connected to a real patient or simulate the patient's behavior through the use of some strategy. Then, this data is sent to a running BSN that in ② logs at predefined intervals the patient's situation (Sensors Reading) and the patient's diagnosis provided by the BSN (BSN outcome). Next, based on the Sensor Reading, the expected diagnosis (Expected Outcome) is computed in ③, and compared to the BSN outcome in ④. The testing results are finally provided. In the following, we walk through all stages while referring to the running example in Fig. 2.

Patient data collection. Stage ① corresponds to the collection of sensor data from a patient. This patient may be either a real patient to which there are some sensors connected or a determined model or structure that simulates the behavior of a patient for a determined period of time. This data is then sent to a running BSN. Sections 4.1, 4.2, and 4.3 provide a detailed description of this stage for the three proposed approaches, *PASTA*, *ValComb* and *TransCov*, respectively. Note that even if in the proposed approaches we allow the simulation of unusual and edge cases, aiming for a more comprehensive evaluation of the BSN, for all of them the Patient Data Collection stage may be easily adapted by adding a constraint to it to avoid the generation of unrealistic inputs for the BSN.

BSN logging. Stage ② consists of logging the BSN while it is being executed to collect sensor readings, which are the data about the different patients at every instant of time, and the BSN Outcome. We provide a more formal definition for Sensor Reading as follows.

Definition 1 (*Sensors Reading*)**.** The reading of all the sensors of a patient $p \in P$ at time t is denoted as $SR_t^p = (SV_t^1, SV_t^2, \ldots, SV_t^n)$, where P is the set of patients, and SV_t^i is the value of sensor i at time t, with $1 \leq i \leq n$ and n is the number of sensors.

Then, we can define BSN outcome. Let $Patient_{Risks} = \{o_1, \ldots, o_k\}$ be the set of possible outcomes of the BSN, that is, the possible patient risk levels. The definition of BSN outcome is formally provided as follows.

Definition 2 (*BSN Outcome*)**.** BSN outcome $BO(p, t, SR_t^p) = o$ is a function that takes as input a patient $p \in P$, an instant of time $0 \leq t$ and SR_t^p, i.e., the sensors reading of patient p at time t, and returns the BSN outcome $o \in Patient_{Risks}$.

Table 1 provides some fictitious instances of Sensors Readings and BSN Outcomes for the running example previously described in Section 3.1. The columns indicate the patient ID, the instant of time, the sensor values, and the BSN Outcome based on the sensor values, respectively. The first four columns compose Sensors Readings. For instance, the first row refers to patient 0 and time 0, and the values of sensors reading by SR as follows: $SR_0^0 = (5, 10)$. The last column contains the labels which are the BSN outcomes. Still referring to the first row, "Low Patient Risk" is the value calculated by $BO = (0, 0, (5, 10))$.

Table 1
Examples of fictitious Sensors Readings and BSN outcomes.

Sensors reading				BSN Outcome
Patient	Time	Sensor 1	Sensor 2	Label
0	0	5	10	Low patient risk
0	1	15	18	Moderate patient risk
1	0	22	30	Critical patient risk
1	1	21	28	Critical patient risk

This stage is very relevant since when simulating a patient through a set of DTMC's, for example in *PASTA* and *TransCov*, to understand what the current patient risk level is, we may need to know at each instant of time which state each of the DTMC's is in. By logging in this set of information from BSN, we are able to understand the patient's situation according to the simulated sensors.

Expected outcome computation. In Stage ③, the Expected Outcome, that is, the Oracle, is computed based on the Sensors Readings. In other words, this phase provides the expected overall risk level of a patient at a determined instant of time according to the data provided by each Sensors Reading entry. The Expected Outcome is formally defined as follows.

Definition 3 (*Expected Outcome*)**.** Expected outcome $E\_BO(p, t, SR_t^p) = e\_o$ is a function that gets as input a patient $p \in P$, a time instant $0 \leq t$, and a sensors reading for the patient p at time t, i.e., SR_t^p, and returns the expected outcome $e\_o \in Patient_{Risks}$.

The computation of the $E\_BO(p, t, SR_t^p)$ function should be performed by using a rule created by the domain expert (i.e., a doctor). That is, the expert, based on her/his knowledge about the domain, creates a function that takes as input the sensor values and outputs the expected outcome. The Expected Outcome is computed for all the Sensors Readings generated in the previous stage. An example of this function for the running example could be:

$$E\_BO(p, t, SR_t^p) = \begin{cases} \text{Low PR} & \text{, if } \forall SV_t^i, SV_t^i \leq 10, \\ \text{Moderate PR} & \text{, if } \forall SV_t^i, 11 \leq SV_t^i \leq 20, \\ \text{Critical PR} & \text{, if } \forall SV_t^i, 21 \leq SV_t^i \leq 30. \end{cases}$$

where $1 \leq i \leq n$, PR stands for "Patient Risk" and the set of possible patient risk levels $Patient_{Risks}$ is defined as {"Low PR", "Moderate PR", "Critical PR"}.

This function states that when all sensors provide values lower than or equal to 10, the expected outcome is "Low PR". If the sensors

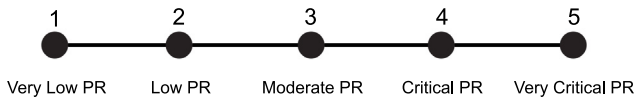


Fig. 4. Example of possible patient risk levels.

provide values from 11 to 20, the expected outcome is "Moderate PR". Otherwise, the expected outcome is "Critical PR". These rules must be created so that they cover all possible values within the sensor value ranges RL_i.

One may think that manually crafted rules, used as the test oracle, could potentially be incorporated directly into BSNs to ensure correctness. However, the role of an oracle differs from that of writing correct code. While an oracle may effectively identify bugs, it is not intended to be embedded within the system. As discussed in Barr et al. (2015), specified oracles often face challenges, such as representing results in a partial or oversimplified manner, which limits their usefulness as guidance for system development.

Comparison. Finally, in stage ④ the BSN outcome and the Expected Outcome (i.e., the Oracle), for a determined Sensors Reading, are compared. For each test case, it is decided if the test passed or not, whereby we define a Test Case as follows:

Definition 4 (*Test Case*)**.** A test case $TC = (SR_t^p,\ E\_BO(p,t,SR_t^p))$ with a patient $p \in P$, the set of patients, and $0 \leq t$, is a tuple where:

- SR_t^p is a Sensors Reading as described in Definition 1;
- $E\_BO(p,t,SR_t^p)$ is the Expected Outcome as described in Definition 3.

Then, for each TC so defined, we compare its Expected Outcome against the actual BSN outcome using the function $Comp$.

Definition 5 (*Comparison Function*)**.** Given a patient $p \in P$, an instant of time $0 \leq t$ and SR_t^p, $Comp$ is a comparison function that takes as input the BSN outcome $BO(p,t,SR_t^p)$ and the Expected Outcome $E\_BO(p,t,SR_t^p)$, and returns a Testing Result $\in (Pass, Fail)$.

We implement the $Comp$ function based on a generic notion of distance between expected and actual outcomes. Let us suppose that the possible patient risk levels $Patient_{Risks}$ is {"Very Low PR", "Low PR", "Moderate PR", "Critical PR", and "Very Critical PR"}. We then assign to each of these risk levels an integer number, as for instance in Fig. 4. To account for possible low oscillations of sensor values we assess that a test case passes if the difference between the Expected Outcome number and the BSN Outcome number is lower than 2. Otherwise, it does not pass. Thus, for example, if the BSN provides as outcome "Low PR" (2) and the Expected Outcome is "Moderate PR" (3), we consider the test successful $(3-2=1)$; if instead the BSN provides "Low PR" (2) and the Expected Outcome is "Critical PR" (4), the test fails $(4-2=2)$.

4. Three instances of the generic approach

In this section, we present three instances of *GATE4BSN*, namely *PASTA* (Section 4.1), *ValComb*(Section 4.2), and *TransCov* (Section 4.3). They all have in common the stages ②, ③, and ④ of *GATE4BSN*, but present a different strategy for stage ①, Patient Data Collection. While steps ②, ③, and ④ have been already explained in the previous section, in the following we describe how the three instances implement step ①. *PASTA* employs combinatorial testing and DTMCs to generate patients with different behaviors. In *ValComb*, the data coming from the patient are the possible combinations of sensor risk levels. Finally, *TransCov* makes use of a dummy patient with equal probabilities in all the transitions of DTMCs, aiming at covering all their arcs.

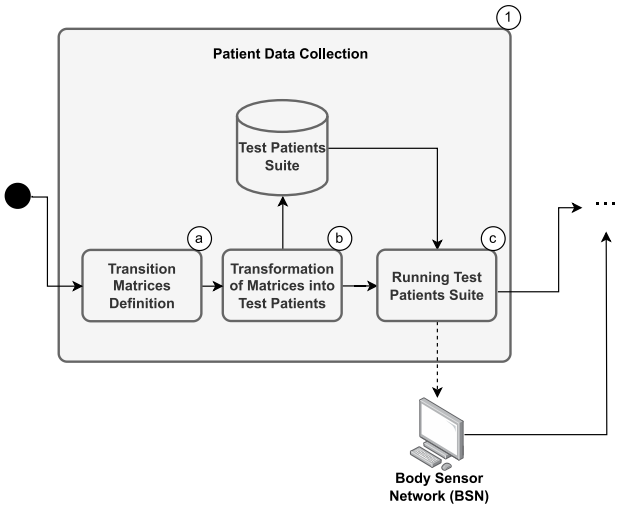


Fig. 5. Patient data collection for PASTA.

4.1. PASTA approach

In this section, we show how *PASTA* implements the stage ① of *GATE4BSN* described in Section 3.2, the Patient Data Collection. This stage is composed of the following substages: ⓐ Transition Matrices Definition, ⓑ Transformation of Matrices into Test Patients, and ⓒ Running Test Patient, as shown in Fig. 5. We describe them as follows.

Transition matrices definition. In stage ⓐ testers using *PASTA* define a set of transition matrices, which we use to simulate, during testing, the vital signs that would be read by the sensors while monitoring one person. Formally, in *PASTA* we define a sensor as follows.

Definition 6 (*Sensor*)**.** A Sensor is defined as a quadruple $S = (Freq, m, RL, TM)$ where:

- $Freq$ is the change frequency in the states of the sensor DTMC;
- m is the number of states in the DTMC;
- $RL = ((Min^1, Max^1), \ldots, (Min^m, Max^m))$ are the Risk Levels of the sensor. Each risk level is defined as a range, i.e., minimum and maximum values, for each state in the DTMC of the sensor;
- $TM = (p_{11}, p_{12}, \ldots, p_{mm})$ are the values assigned to the transition matrix of the sensor.

To control the probability of going to a determined state regardless of the state of origin, we restrict the possible variations of each transition matrix TM of a Sensor S, by requiring that all the rows of the same matrix are equal. In other words, the probability Pc_j, with $1 \leq j \leq m$, of each column j in the transition matrix TM is:

$$Pc_1 = p_{11} = p_{21} = \cdots = p_{m1}$$
$$Pc_2 = p_{12} = p_{22} = \cdots = p_{m2}$$
$$\ldots$$
$$Pc_m = p_{1m} = p_{2m} = \cdots = p_{mm}$$

Thus, referring to the example in Fig. 2, for Sensor 1 we assume that $Pc_1 = LL_1 = ML_1 = HL_1$, $Pc_2 = LM_1 = MM_1 = HM_1$ and $Pc_3 = LH_1 = MH_1 = HH_1$. In this way, the transition matrix TM of Sensor 1 is fully defined by three values: Pc_1, Pc_2, and Pc_3. Note that this is a practical restriction we made to better control our approach, but that can be easily removed.

Stage ⓐ aims at defining the values of TM (i.e., of $Pc_1, \ldots, Pc_m$). As a test strategy, we considered three different scenarios:

(i) All arcs in the DTMC have the same probability, i.e., from the current sensor risk level, any risk level can be reached next, or intuitively the patient has no specific medical profile.
(ii) All arcs in the DTMC that enter one determined state have a probability equal to 100% and all the remaining arcs are set to 0. In other words, from wherever in the DTMC, the same risk level is always reached, or the patient has a very clear diagnosis.
(iii) All arcs in the DTMC that enter one determined state have a probability equal to 0, i.e. this risk level is never reached, and all the other remaining arcs have equal probabilities.

As the values of TM are transition probabilities, we also define a constraint to ensure that the sum of values referring to the transitions from the same sensor state is equal to 100%:

$$Pc_1^i + Pc_2^i + \cdots + Pc_m^i = 100 \quad (2)$$

where i denotes the sensor and m the number of states in the DTMC of sensor i. Considering the example in Fig. 2, the constraint states that $Pc_1^1 + Pc_2^1 + Pc_3^1 = 100$ and $Pc_1^2 + Pc_2^2 = 100$.

Transformation of matrices into test patients. Stage ⓑ consists of generating and transforming transition matrices describing the behavior of all sensors into a "Test Patient": by this term, we consider the description of the vital signs' behavior collected by a whole set of sensors that are all applied to one person. Formally, we define a Test Patient as follows.

Definition 7 (*Test Patient*)**.** A test patient $TP = \{S_1, \ldots, S_n\}$ is a set of n sensors.

Considering the example in Fig. 2, the Test Patient there defined is $TP = \{S_1, S_2\}$, where $S_1 = (Freq_1, TM1, RL_1)$ and $S_2 = (Freq_2, TM2, RL_2)$. The frequencies $Freq_1$ and $Freq_2$ must be defined as well as the pairs of risk values RL_1 and RL_2. For RL_1, we should define the minimum and maximum values for the *Low*, *Medium*, and *High* risk levels. As for RL_2, it is necessary only to define the *Low* and *High* risk levels. Finally, the matrices $TM1$ and $TM2$ are the outcomes of this process.

In particular, a test patient is characterized by the combination of transition probabilities in the TMs associated with the sensors applied to the patient's body, which we refer to as a t-way combination.

Definition 8 (*T-way Combination*)**.** A t-way combination $\mathbf{C} = (PC^1, PC^2, \ldots, PC^n)$ is a tuple containing the probability of each column in the transition matrix TM of the n sensors connected to a Test Patient. In turn, for $1 \leq i \leq n$, $PC^i = (Pc_1^i, Pc_2^i, \ldots, Pc_{m_i}^i)$ is a tuple containing the probability of each column in the transition matrix TM of a sensor i, and m_i the number of states in the DTMC of sensor i.

We apply combinatorial testing techniques to derive a minimum set of t-way combinations of the parameters in $\mathbf{C}$. Considering again the example in Fig. 2, its t-way combination is $\mathbf{C} = (Pc_1^1, Pc_2^1, Pc_3^1, Pc_1^2, Pc_2^2)$. The combinations generated by the combinatorial strategy are then converted into transition matrices (as the ones in Fig. 2) by positioning the corresponding Pc_j^i values in the matrices as previously explained. Then, by combining the different TMs that are generated for each sensor we can obtain the data of a patient.

In *PASTA*, to test the BSN behavior, we assume that the same BSN can be used to monitor the health status of different individuals with differing health characteristics. Hence, we generate a collection of test patients that share the same set of sensors equally configured (the same frequency and risk levels) but may expose different health statuses; this is reflected in defining different values for the transition matrices across different persons. To denote this set of test patients we introduce the term "Test Patients Suite".

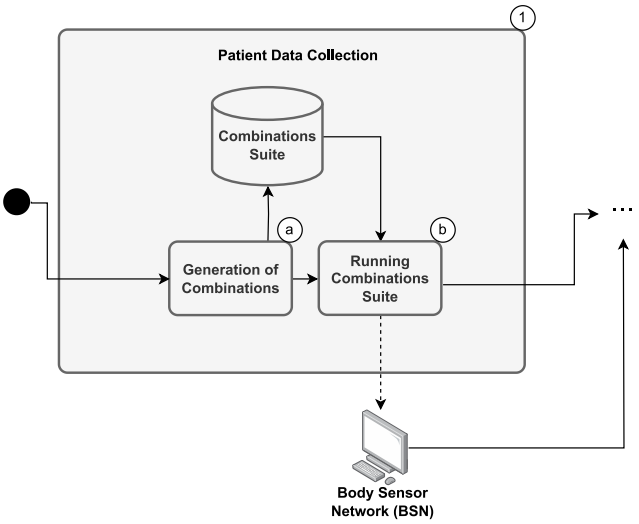


Fig. 6. Patient data collection for *ValComb*.

Table 2
Combinations for a simple running example of BSN.

Combination	Sensor 1	Sensor 2
#1	Low	Low
#2	Low	High
#3	Medium	Low
#4	Medium	High
#5	High	Low
#6	High	High

Running test patients suite. In stage ⓒ, the BSN is executed by taking each of the test patients in the test patient suite as input. *PASTA* makes use of the test patient data to simulate a patient. In this sense, it creates and executes the DTMCs corresponding to the sensors as defined for a certain amount of time, which is a parameter set by the tester. It is important to highlight that the simulation of a patient works by randomly sampling values within the range of the current state (risk level) of a DTMC. For example, if the current state of Sensor 1 DTMC is "Low", and the range (Min^L, Max^L) for the RL "Low" has been set to [65,100], a possible number that could be generated for this sensor is 70. Then, from time to time, the BSN provides an overall patient risk level (*BSN Outcome*, as explained before) based on the combination of these values for all the sensors.

4.2. ValComb approach

In this section, we show how the *ValComb* (Sensor Values Combination) approach implements the stage ① of our generic approach described in Section 3.2, the Patient Data Collection. This stage is composed of the following substages: ⓐ Generation of Combinations and ⓑ Running Combinations Suite, as shown in Fig. 6. We describe them as follows.

Generation of combinations. In *ValComb*, the stage ⓐ consists of generating all the possible combinations between sensors' risk levels. If we consider, for instance, the example provided in Section 3.1, for Sensor 1 the possible risk levels are: Low, Medium and High, while for Sensor 2 they are: Low and High. Table 2 depicts the set of combinations for that simple example of BSN. All the combinations are stored in what we call the "Combinations Suite".

Running combinations suite. In *ValComb*, stage ⓑ consists of walking through the Combinations Suite and executing each combination individually on the BSN under test. This is done by sampling values within

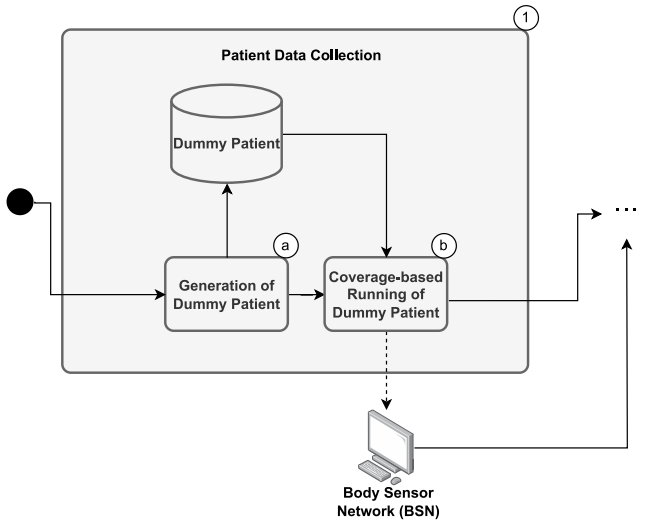


Fig. 7. Patient data collection for *TransCov*.

the ranges of each risk level in the combination. For example, if we consider Combination #1 in Table 2, for Sensor 1 a value within its Low-level range should be randomly sampled. The same applies to Sensor 2. Then, the BSN provides an overall patient risk level (*BSN Outcome*, as explained before) based on the combination of these values for all the sensors.

4.3. TransCov approach

In this section, we show how the *TransCov* (Sensor Transitions Coverage) approach implements the stage ① of our generic approach described in Section 3.2, the Patient Data Collection. This stage is composed of the following substages: ⓐ Generation of a Dummy Patient and ⓑ Coverage-based Running of Dummy Patient, as shown in Fig. 7. We describe them as follows.

Generation of a dummy patient. In *TransCov*, the stage ⓐ comprises the generation of what we call a Dummy Patient. It consists of defining a Testing Patient TP, as presented in Definition 7, in which regardless of the number of sensors, all their Transition Matrices TM contain equal probabilities, that is, the probability of going from one state to another is always the same, no matter what is the current state.

If we consider, for instance, the example provided in Section 3.1, for Sensor 1, the transition matrix TM1 contains the same probability values at each of its cells, that is, $LL_1 = LM_1 = LH_1 = ML_1 = MM_1 = MH_1 = HL_1 = HM_1 = HH_1$. The same goes for TM2, corresponding to Sensor 2.

Coverage-based running of a dummy patient. In *ValComb*, stage ⓑ consists of running the Dummy Patient taking into consideration the percentage of transition coverage as a stopping criterion. In other words, the Dummy Patient is run until all the arcs/transitions of the transition matrices TM of all the sensors are covered. The arcs in a transition matrix are randomly covered since the Dummy Patient has equal probability for all of the transitions of its transition matrices, but once one arc is covered, we check this arc as already covered and distribute its probability among the remaining uncovered arcs that share the same origin as this arc. We stop running the patient when all the arcs of the DTMCs of all sensors have been covered, that is, once we achieve 100% of coverage.

5. Study setting

To evaluate how effective are the three approaches (instances of *GATE4BSN*) in failure detection we performed some experiments. Specifically, we measure the Passing Test Case Rate (PTCR) and the Execution Time (ET) for the three instances of *GATE4BSN* and the baseline (defined in the next subsection). PTCR measures the percentage of test cases that pass in the SUT. Thus, the smaller the better, that is, the more effective in finding failures. In this case, we apply the function $Comp$ (Definition 5) that results in *Fail* for $difference \geq 2$ and *Pass* for $difference < 2$ when comparing the BSN Outcome (Definition 2) and the Expected Outcome (Definition 3). ET comprises the total time spent to perform one execution of a testing approach. To compute ET, we calculate the time taken for running the approach in the case of the three instances of *GATE4BSN*, or run the baseline for an equivalent amount of time.

In the remainder of this section, we describe the baseline to which we compare the three approaches, the System Under Test (SUT), i.e., Self-Adaptive Body Sensor Network (SA-BSN), and the experimental setup and procedure.

5.1. Baseline

To the best of our knowledge, there is no testing approach in the literature for BSNs to which we can compare the three instances of *GATE4BSN*. As discussed later in Section 7, we identified some approaches in the literature that are related to our proposal. However, these approaches would need significant adaptation to effectively handle BSN testing. Additionally, most of them do not provide a replication package, making it difficult to extract the source code and perform experiments. For this reason, we propose the random generation of data from a patient as a baseline. To develop the random approach, we make use of a function that generates random numbers following a normal distribution. Therefore, whenever the BSN requests vital sign values from the patient, our function provides random numbers. These random numbers are generated respecting the limits of each sensor (vital sign), as shown in Table 3. For example, Oxygen Saturation values range from 0 to 100. Accordingly, we generate values within this range anytime the BSN requests values coming from the Oxi sensor. On the other hand, if the BSN requires data from the Ecg sensor, the random value generated is within the range [0,300].

The simulation of a patient in the random approach runs for the same amount of time as *PASTA*, since PASTA is the approach among the proposed ones that takes more time. Notice that this time is different from the Execution Time (ET) that is explained and computed later in this work. This time is a parameter for the BSN and corresponds to the amount of time for which the simulation of the patient from which we collected data will run. On the other hand, ET comprises the entire time spent by the approach, including, for example, reading and writing operations and delays in the script. As for *PASTA* we run 278 patients, each of them for 30 s, we end up simulating patients for 8340 s. Thus, we also run Random using the number 8340 as a parameter for the amount of patient simulation time. To guarantee the integrity of the results, we run the Random approach 25 times.

5.2. Self-Adaptive Body Sensor Network (SA-BSN)

The implementation of our SUT, i.e., the Self-Adaptive Body Sensor Network (SA-BSN) (Gil et al., 2021) is publicly available.[1] The SA-BSN's objective is to identify emergencies by continuously checking on the patient's health. It contains six sensors including an electrocardiograph sensor (Ecg) for heartbeat rate and electrocardiogram curve, a pulse oximeter (Oxi) for blood oxygen saturation measurement, a

[1] https://github.com/rdinizcal/sa-bsn.

Table 3
Data ranges for the sensors of SA-BSN.

Sensor	Data ranges
Oxygen Saturation (Oxi)	100 > low > 65 > medium > 55 > high > 0
Heart Beat Rate (Ecg)	300 > high > 115 > medium > 97 > low > 85 > medium> 70 > high > 0
Temperature (Term)	50 > high > 41 > medium > 38 > low > 36 > medium> 32 > high > 0
Systolic Body Pressure (Abps)	300 > high > 140 > medium > 120 > low > 0
Diastolic Body Pressure (Abpd)	300 > high > 90 > medium > 80 > low > 0
Glucose (Glc)	200 > high > 120 > medium > 96 > low > 55 > medium > 40 > high > 20

thermometer (Term), which measures body temperature in Celsius, a sphygmomanometer for measuring diastolic (Abpd) and systolic (Abps) arterial blood pressure, and a glucose sensor (Glc) for measuring blood glucose levels. Table 3 presents the ranges for the risk levels of the SA-BSN sensors.

We chose to structure it this way because it made the oracle easier to understand and align with the way the SUT operates. However, this can be easily adapted for other BSNs that may use different ranges, sensors, or even no specific ranges at all.

In this work, we employed a version of the SA-BSN fixed by us and that is made available in our already mentioned replication package.[2] The original version contained two bugs related to the sensor readings printing function and the function that moves a DTMC to the next state. Since these bugs compromised the accuracy of the results, we reported them to the developers and used the fixed version in these experiments.

For the SA-BSN, three are the risk level ranges (i.e., Low, Medium-1, and High-1) presented by each of the Oxi, Abps, and Abpd sensors. The risk level ranges of the Ecg, Term, and Glc sensors are five (i.e., Low, Medium-0, Medium-1, High-0, and High-1). This distinction is taken into account in the experimental procedure that follows.

5.3. Experimental setup and procedure

The experiments have been carried out on a computer with Ubuntu 20.04.6 LTS, 16 GB of RAM Memory, and an Intel i7-1165G7 processor. To run the SA-BSN, it is also necessary to install the ROS Noetic[3] for Ubuntu 20.04. To install the SA-BSN, we followed the instructions available here.[4] Because we employ DTMCs and generation of random numbers, each run of *GATE4BSN* exhibits non-deterministic behavior. Therefore, to ensure the validity of the experiments, we performed 25 runs of each instance of *GATE4BSN*. The experimental procedure has been defined according to the guidelines in Wohlin et al. (2012) and it follows the same four stages described in Section 3.

Patient data collection for PASTA. We first define the transition matrices for the SA-BSN. We report the parameters used to the ACTS tool in Table 4. Its first column contains the sensor to which the parameter is associated. In the second column, we present the parameter name, which comprises the vital sign name combined with the specific risk level associated with the parameter. The possible risk levels for the SA-BSN, which are High-0, Medium-0, Low, Medium-1, and High-1, are represented here as State0, State1, State2, State3, and State4, respectively. We create a parameter for each combination of vital signs (or sensors) and possible risk levels (or states). Then, the "Parameter Value", in the third column, provides the values that may be combined in the t-way combinations for each specific parameter. Notice that, since the sensors Oxi, Abps, and Abpd present solely 3 risk level ranges,

Table 4
Parameters of the SUT in the ACTS tool for *PASTA*.

Sensor	Parameter name	Parameter value
Oxygen Saturation (Oxi)	oxi_State0	[0]
	oxi_State1	[0]
	oxi_State2	[0, 33, 34, 50, 100]
	oxi_State3	[0, 33, 34, 50, 100]
	oxi_State4	[0, 33, 34, 50, 100]
Heart Beat Rate (Ecg)	hr_State0	[0, 20, 33, 34, 50, 100]
	hr_State1	[0, 20, 33, 34, 50, 100]
	hr_State2	[0, 20, 33, 34, 50, 100]
	hr_State3	[0, 20, 33, 34, 50, 100]
	hr_State4	[0, 20, 33, 34, 50, 100]
Temperature (Term)	temp_State0	[0, 20, 33, 34, 50, 100]
	temp_State1	[0, 20, 33, 34, 50, 100]
	temp_State2	[0, 20, 33, 34, 50, 100]
	temp_State3	[0, 20, 33, 34, 50, 100]
	temp_State4	[0, 20, 33, 34, 50, 100]
Systolic Body Pressure (Abps)	abps_State0	[0]
	abps_State1	[0]
	abps_State2	[0, 33, 34, 50, 100]
	abps_State3	[0, 33, 34, 50, 100]
	abps_State4	[0, 33, 34, 50, 100]
Diastolic Body Pressure (Abpd)	abpd_State0	[0]
	abpd_State1	[0]
	abpd_State2	[0, 33, 34, 50, 100]
	abpd_State3	[0, 33, 34, 50, 100]
	abpd_State4	[0, 33, 34, 50, 100]
Glucose (Glc)	gluc_State0	[0, 20, 33, 34, 50, 100]
	gluc_State1	[0, 20, 33, 34, 50, 100]
	gluc_State2	[0, 20, 33, 34, 50, 100]
	gluc_State3	[0, 20, 33, 34, 50, 100]
	gluc_State4	[0, 20, 33, 34, 50, 100]

for the parameters related to High-0 and Medium-0, that is, State0 and State1, the only value that they can assume is zero. It means that these states will not be considered during the SA-BSN execution. Besides the parameters, in the ACTS tool, we may need to define possible constraints. We restrict the parameters according to Eq. (2) by providing the list of constraints in Eqs. (3)–(8).

$$100 = oxi\_State0 + oxi\_State1 + oxi\_State2 + oxi\_State3 + oxi\_State4 \quad (3)$$

$$100 = hr\_State0 + hr\_State1 + hr\_State2 + hr\_State3 + hr\_State4 \quad (4)$$

$$100 = temp\_State0 + temp\_State1 + temp\_State2 + temp\_State3 + temp\_State4 \quad (5)$$

$$100 = abps\_State0 + abps\_State1 + abps\_State2 + abps\_State3 + abps\_State4 \quad (6)$$

$$100 = abpd\_State0 + abpd\_State1 + abpd\_State2 + abpd\_State3 + abpd\_State4 \quad (7)$$

$$100 = gluc\_State0 + gluc\_State1 + gluc\_State2 + gluc\_State3 + gluc\_State4 \quad (8)$$

Finally, in generating the t-way combinations, we choose t = 2 (i.e., pairwise), as detailed in Section 2.2. For the remaining setup in the ACTS tool, we use the default values.

We then run the ACTS tool with the parameters and constraints previously defined. In total, the ACTS generates 278 combinations. An excerpt of the combinations generated is illustrated in Table 5. For example, the table indicates that the rows of the transition matrix for Oxi of the first combination ("C.#1") contain the sequence: [0 0 33 33 34],[5] which are the values, for this combination, of the parameters: *oxi_State0*, *oxi_State1*, *oxi_State2*, *oxi_State3*, and *oxi_State4*, respectively.

[2] https://github.com/samirasilva/Paper_JSS.
[3] http://wiki.ros.org/noetic.
[4] https://github.com/lesunb/bsn.

[5] The sequence 33, 33, 34 is an integer approximation for the case of all transitions having the same probability.

Table 5
Combinations generated by ACTS for *PASTA*.

Parameter	C.#1	C.#2	C.#3	C.#4
oxi_State0	0	0	0	0
oxi_State1	0	0	0	0
oxi_State2	33	34	50	100
oxi_State3	33	33	50	0
oxi_State4	34	33	0	0
hr_State0	0	0	0	0
hr_State1	0	33	34	50
hr_State2	33	34	0	50
hr_State3	33	0	33	0
hr_State4	34	33	33	0
temp_State0	20	33	34	50
temp_State1	20	33	0	50
temp_State2	20	34	0	0
temp_State3	20	0	33	0
temp_State4	20	0	33	0
abps_State0	0	0	0	0
abps_State1	0	0	0	0
abps_State2	33	34	50	100
abps_State3	33	33	50	0
abps_State4	34	33	0	0
abpd_State0	0	0	0	0
abpd_State1	0	0	0	0
abpd_State2	33	34	50	100
abpd_State3	33	33	50	0
abpd_State4	34	33	0	0
gluc_State0	20	33	34	50
gluc_State1	20	33	0	50
gluc_State2	20	34	0	0
gluc_State3	20	0	33	0
gluc_State4	20	0	33	0

```
<launch>
<node name="patient" pkg="patient" type="patient" output="
    screen" />

<param name="frequency" value="10" />

<param name="vitalSigns" value="oxigenation,heart_rate,
    temperature,abps,abpd,glucose" />

<!-- Change frequency in states of each Markov -->
<param name="oxigenation_Change" value="0.2"/>
<param name="heart_rate_Change" value="0.1"/>
<param name="temperature_Change" value="0.1"/>
<param name="abps_Change" value="0.1"/>
<param name="abpd_Change" value="0.1"/>
<param name="glucose_Change" value="0.1"/>
```

Fig. 8. Header of the configuration file for *PASTA*.

Next, we perform the conversion of transition matrices into a test patient TP. The patient is the input for SA-BSN and is represented by a configuration file with a predefined header as shown in Fig. 8. It may contain information concerning: whether the sensor values will be shown on the screen or saved in a "log file"; the node scheduling frequency; the vital sign names; and for each DTMC (i.e., sensor) the change frequency.

Following the header, we define the transition matrix (TM) and the ranges of risk levels (RL) associated with each of the 5 states in the sensor DTMC. The current state represents the current status of the patient concerning that sensor, that is, if the DTMC for the Oxygen Saturation is in the "LowRisk" state, the patient presents low-risk values according to this specific sensor.

Let us consider the first combination ("C.#1") of Table 5 as an example from now on. Each transition matrix defined in the previous stage and generated here is converted into a fragment of the input configuration file. Fig. 9 shows an excerpt of the configuration file containing the DTMC transition matrix for the Oxygenation sensor and the risk level ranges of each state in the DTMC. Notice that we do not

```
<!-- Markov chain for oxigenation -->
<param name="oxigenation_State0" value="0,0,0,0,0" />
<param name="oxigenation_State1" value="0,0,0,0,0" />
<param name="oxigenation_State2" value="0,0,33,33,34" />
<param name="oxigenation_State3" value="0,0,33,33,34" />
<param name="oxigenation_State4" value="0,0,33,33,34" />

<!-- Risk values for oximeter -->
<param name="oxigenation_HighRisk0" value="-1,-1"/>
<param name="oxigenation_MidRisk0" value="-1,-1"/>
<param name="oxigenation_LowRisk" value="65,100" />
<param name="oxigenation_MidRisk1" value="55,65" />
<param name="oxigenation_HighRisk1" value="0,55" />
```

Fig. 9. Oxygenation setup in the configuration file for *PASTA*.

```
<!-- Markov chain for heart frequency -->
<param name="heart_rate_State0" value="0,0,33,33,34" />
<param name="heart_rate_State1" value="0,0,33,33,34" />
<param name="heart_rate_State2" value="0,0,33,33,34" />
<param name="heart_rate_State3" value="0,0,33,33,34" />
<param name="heart_rate_State4" value="0,0,33,33,34" />

<!-- Risk values for heart frequency -->
<param name="heart_rate_HighRisk0" value="0,70" />
<param name="heart_rate_MidRisk0" value="70,85" />
<param name="heart_rate_LowRisk" value="85,97" />
<param name="heart_rate_MidRisk1" value="97,115" />
<param name="heart_rate_HighRisk1" value="115,300" />
```

Fig. 10. Heart beat rate setup in the conf. file for *PASTA*.

use State0 and State1 since there is only one range for the medium and high-risk levels, as previously shown in Table 3. So, we fill with zeros the rows in the transition matrix corresponding to these states. Concerning the risk levels, we set their value to −1. These states are useful for sensors with multiple ranges for the same risk level.

In Fig. 10 we report the DTMC transition matrix for the Heartbeat Rate sensor and the risk level ranges of each state in the DTMC. Each row of the transition matrix consists of the sequence {0, 0, 33, 33, 34} corresponding to the heartbeat rate parameters of "Combination #1" in Table 5. Notice that, for this sensor, we assign values to State0 and State1 since there are two possible ranges for the medium and high-risk levels, as previously shown in Table 3.

We repeat this process for all 6 sensors of the SA-BSN, and the transition matrices corresponding to the same t-way combination are converted into excerpts that together make the configuration file, that is, the test patient. The combinations generated by the ACTS tool become test patients, and this leads to 278 patients.

Each of the patient configuration files generated is used as input to the SA-BSN execution. Besides this file, it is also necessary to inform how long the SA-BSN should run. In our study, we leave each test patient to run for 30 s (since this amount of time has proved to be sufficient for the SA-BSN to produce a substantial amount of data).

Patient data collection for ValComb. To generate all the possible combinations of sensor risk levels we make use of the ACTS tool. The parameters we use in the tool are shown in Table 6. They may be either 100, meaning that in the combination the sensor is in that state (risk level), or 0 otherwise.

We then run the ACTS tool with the parameters and constraints previously defined, and t = 6. In total, the ACTS generates 3375 combinations. An excerpt of the combinations generated is illustrated in Table 7. All of these combinations are stored in a file that we call "Combinations Suite" and, individually, they are sent to the running SA-BSN, which randomly samples values within a determined range for each sensor depending on the state (risk level) that contains the value 100.

Table 6
Parameters of the SUT in the ACTS tool for *ValComb*.

Sensor	Parameter name	Parameter value
Oxygen Saturation (Oxi)	oxi_State0	[0]
	oxi_State1	[0]
	oxi_State2	[0, 100]
	oxi_State3	[0, 100]
	oxi_State4	[0, 100]
Heart Beat Rate (Ecg)	hr_State0	[0, 100]
	hr_State1	[0, 100]
	hr_State2	[0, 100]
	hr_State3	[0, 100]
	hr_State4	[0, 100]
Temperature (Term)	temp_State0	[0, 100]
	temp_State1	[0, 100]
	temp_State2	[0, 100]
	temp_State3	[0, 100]
	temp_State4	[0, 100]
Systolic Body Pressure (Abps)	abps_State0	[0]
	abps_State1	[0]
	abps_State2	[0, 100]
	abps_State3	[0, 100]
	abps_State4	[0, 100]
Diastolic Body Pressure (Abpd)	abpd_State0	[0]
	abpd_State1	[0]
	abpd_State2	[0, 100]
	abpd_State3	[0, 100]
	abpd_State4	[0, 100]
Glucose (Glc)	gluc_State0	[0, 100]
	gluc_State1	[0, 100]
	gluc_State2	[0, 100]
	gluc_State3	[0, 100]
	gluc_State4	[0, 100]

Table 7
Combinations generated by ACTS for *ValComb*.

Parameter	C. #1	C. #2	C. #3	C. #4
oxi_State0	0	0	0	0
oxi_State1	0	0	0	0
oxi_State2	0	100	0	100
oxi_State3	0	0	100	0
oxi_State4	100	0	0	0
hr_State0	0	100	0	0
hr_State1	0	0	100	0
hr_State2	100	0	0	0
hr_State3	0	0	0	100
hr_State4	0	0	0	0
temp_State0	0	100	0	0
temp_State1	0	0	100	0
temp_State2	100	0	0	0
temp_State3	0	0	0	100
temp_State4	0	0	0	0
abps_State0	0	0	0	0
abps_State1	0	0	0	0
abps_State2	100	0	0	0
abps_State3	0	100	0	100
abps_State4	0	0	100	0
abpd_State0	0	0	0	0
abpd_State1	0	0	0	0
abpd_State2	100	0	0	100
abpd_State3	0	100	0	0
abpd_State4	0	0	100	0
gluc_State0	0	100	0	0
gluc_State1	0	0	100	0
gluc_State2	0	0	0	100
gluc_State3	100	0	0	0
gluc_State4	0	0	0	0

Patient data collection for TransCov. The generation of a dummy patient for the Patient Data Collection of *TransCov* consists of giving the same probability for all the transitions of transition matrices representing sensors. In Fig. 11, we report the DTMC transition matrix for the Heartbeat Rate sensor and the risk level ranges of each state in the DTMC.

```
<!-- Markov chain for heart frequency -->
<param name="heart_rate_State0" value="20,20,20,20,20" />
<param name="heart_rate_State1" value="20,20,20,20,20" />
<param name="heart_rate_State2" value="20,20,20,20,20" />
<param name="heart_rate_State3" value="20,20,20,20,20" />
<param name="heart_rate_State4" value="20,20,20,20,20" />

<!-- Risk values for heart frequency -->
<param name="heart_rate_HighRisk0" value="0,70" />
<param name="heart_rate_MidRisk0" value="70,85" />
<param name="heart_rate_LowRisk" value="85,97" />
<param name="heart_rate_MidRisk1" value="97,115" />
<param name="heart_rate_HighRisk1" value="115,300" />
```

Fig. 11. Heart Beat rate setup in the configuration file for *TransCov*.

We do the same for all the sensors of the SA-BSN and create a dummy patient configuration file. This configuration file is used as input to the SA-BSN execution. The SA-BSN runs until all the arcs of transition matrices are covered.

Logging the BSN. During the execution of the SA-BSN, we collect and store all the values generated by the sensors (the sensor readings SRs). We also gather the BSN outcome BO, that is, the patient risk level provided by the SA-BSN according to the sensors' values. The possible outcomes ($Patient_{Risks}$) for the SA-BSN are: "Very Low PR", "Low PR", "Moderate PR", "Critical PR" and "Very Critical PR". Table 8 reports some examples of the collected data for *PASTA*.

Expected outcome computation and comparison. We also compute the oracle, that is, the expected outcome $E\_BO(p, t, SR_t^p)$, for each of the sensor readings SR_t^p collected. The oracle, in this work, has been defined as Table 9 describes.

Each sensor reading SR and its corresponding expected outcome E_BO make a test case TC. Note that to compute the oracle, instead of using the SR purely, we convert the values into risk levels (i.e., low, moderate, or high risk) based on the ranges considered by the SA-BSN, as previously shown in Table 3. Then, we count the number of sensors at each risk possible level. We opted for this conversion because it made the oracle simpler to understand and better aligned with the way the SA-BSN functions. However, this approach can be easily adapted for other BSNs that use different sensor types, ranges, or even no specific ranges at all. The 10th row of Table 8 reports some examples of expected outcomes for sensor readings collected from the SA-BSN. In the 11th row, we provide the BSN outcomes BO. Then, as shown in the following row of the table, we present the result of the function $Comp$ that computes the absolute difference between the BSN outcome and the expected outcome corresponding numbers, and decides if the TC passed or not. We recall, as stated in Definition 5, that a TC passes if the difference is lower than 2, otherwise, it fails.

The complete collection of data we gathered in each stage of *GATE4BSN* can be found on Github.[6]

6. Experimental results

This section encompasses the experimental findings (Section 6.1), the statistical analysis we performed (Section 6.2), and the potential threats to the experiments' validity (Section 6.3). The experiments measured the Passing Test Case Rate (PCTR) and the Execution Time (ET) of our three proposed approaches, showing that in comparison to the random baseline they provide an effective means for testing Body Sensor Networks. Specifically:

[6] https://github.com/samirasilva/Paper_JSS.

Table 8
Examples of Test Cases for the SA-BSN and the results using *PASTA*.

	TC #1	TC #2	TC #3
Patient	0	0	1
Time	0	1	0
Oxi	41.8212 (High risk)	72.5934 (Low risk)	7.03972 (High risk)
Ecg	99.2647 (Moderate risk)	97.9146 (Moderate risk)	95.6635 (Low risk)
Term	37.3749(Low risk)	37.3749 (Low risk)	38.1103 (Moderate risk)
Abps	23.6058 (Low risk)	66.6665 (Low risk)	118.236 (Low risk)
Abpd	25.6534 (Low risk)	18.6827(Low risk)	95.0196 (High risk)
Glc	86.6656 (Low risk)	90.227 (Low risk)	65.6257 (Low risk)
Expected Outcome (Risk)	Critical PR	Low PR	Very Critical PR
BSN Outcome (Risk)	Critical PR	Very Low PR	Moderate PR
Difference	0	1	2
Passed?	Yes	Yes	No

Table 9
Oracle definition.

Risk level	Definition
Very Low Risk	All the sensors are "Low Risk".
Low Risk	One sensor is "Moderate Risk" and the remaining are "Low Risk".
Moderate Risk	At least two sensors are "Medium Risk" and no sensor is "High Risk".
Critical Risk	Exactly one sensor is "High Risk".
Very Critical Risk	More than one sensor is "High Risk".

- *PASTA* resulted in an average of 92.80% ± 0.59 of PTCR and presented an ET on average of 22014.44 seconds.
- *ValComb* resulted in an average of 95.47% ± 0.32 of PTCR and presented an ET on average of 288.88 seconds.
- *TransCov* resulted in an average of 94.96% ± 5.84 of PTCR and presented an ET on average of 339.24 seconds.
- The Random approach (Baseline) resulted in an average of 98.18%± 0.03 of PTCR and presented an ET on average of 8389.16 seconds.

The complete experiment results are presented in Table 10. To analyze the data, we followed the guidelines in de Oliveira Neto et al. (2019) and Arcuri and Briand (2014). For reproducibility, the R scripts, and datasets are made available[6].

6.1. Empirical data

The data for analysis is collected by logging sensor data, the BSN outcome, and timestamps in the format of execution traces, as well as computing the expected outcome. For the three approaches and the baseline, the difference between the BSN outcome and the expected outcome is calculated for each test case. Table 8 shows some examples of test cases for SA-BSN, the expected outcome, the outcome obtained using, for example, *PASTA*, and the calculated difference. Note that, given the different methods applied to derive the test cases, the number of test executions performed for the three proposed approaches differ widely. For *PASTA*, each of the 25 experiments consisted of 278 patients resulting in approximately 88 227.52 test cases in each experiment. The experiments for the baseline were not guided by patients, but by execution time intending to pair with *PASTA*, which is the approach that takes more time, for a fair comparison, resulting in 25 samples of 495 888.12 test cases on average. *ValComb* presents 1 test case for each possible combination of sensor risk label. For SA-BSN, three sensors have 3 possible risk levels and three sensors have 5 possible risk levels. Thus, the total number of combinations of risk labels is 3375 which is the amount of test cases in *ValComb*. Finally, *TransCov* runs until the complete transition coverage is achieved. This approach presents an average of 4234.2 test cases per experiment.

For the approaches that employ DTMCs to represent sensors, i.e., *PASTA* and *TransCov*, we also compute the percentage of DTMC's arcs that have been covered. Table 11 reports the percentage of coverage for these approaches considering each sensor and the average coverage. By calculating coverage for *PASTA*, we were able to observe that the arcs/transitions of sensor DTMCs are being little explored. This fact served as inspiration for proposing an approach that guaranteed full transition exploration, i.e., the *TransCov* approach.

6.2. Data analysis and discussion

To choose an approach to statistical analysis, we carefully examine the properties of the dataset (de Oliveira Neto et al., 2019). After ascertaining that our data does not follow a normal distribution, we apply the Kruskal–Wallis test (Kruskal and Wallis, 1952) to assess at a significance level of 5% the null hypothesis that the differences in PTCR for the different approaches are not statistically significant. The Kruskal–Wallis test is a non-parametric test used to determine whether there are statistically significant differences between three or more independent groups when the assumptions of normality and homogeneity of variances are violated. When considering the baseline and the proposed approaches, the resulting p-value for the test was 1.491e−11, with H = 53.42 and freedom degree equal to 3, meaning that we can reject the null hypothesis that no significant difference in effectiveness exists, at least at the 95% confidence level.

A significant Kruskal–Wallis test indicates that at least one testing approach stochastically dominates another one, but does not identify the dominance relationship among pairs of techniques. To determine which testing approaches are different, we performed pairwise comparisons with Vargha–Delaney effect size (A_{12} measure) (Vargha and Delaney, 2000) after the Kruskal–Wallis test. The results are shown in Table 12. If the value in cell (i, j) is lower than 0.5, it means that the approach i dominates the approach j. Contrariwise, if the value in cell (i, j) is greater than 0.5, the approach j dominates the approach i. Finally, if the value in the cell (i, j) is exactly 0.5, this means that there exists no difference between the scores of the two approaches. By looking at the data in the table, we can conclude that the order of domination is: *PASTA*≫*ValComb*≫*TransCov*≫Random, where the symbol ≫ means "dominates".

According to the statistical analysis, *PASTA* is the most effective approach in terms of the capability to detect failures, as measured by PTCR. However, among the instances of *GATE4BSN*, according to the ET analysis shown in Table 10, it is also the approach that takes the longest execution time. Therefore, *PASTA* should be used if effectiveness is the most important variable when testing a BSN. Also considering statistical analysis, the second most effective approach is *ValComb*, which is still the fastest one. Therefore, in cases where a quick approach to testing BSNs is desired, *ValComb* is a good option, because it achieves a good balance between PCTR and ET. Statistically speaking, *TransCov* is in third place in terms of effectiveness, being a very unstable approach, which can be confirmed if we observe the PTCR value throughout the 25 experiment repetitions. In our interpretation, this might be due to the fact that in addition to the generation of random numbers when the DTMC is in a given state, the transition

Table 10
Passing Test Case Rate (PTCR) and Execution Time (ET) for the three instances of *GATE4BSN* and the Random Approach (Baseline).

	PASTA		ValComb		TransCov		Random	
	PTCR (%)	ET (s)	PTCR (%)	ET (s)	PTCR (%)	ET (s)	PTCR (%)	ET (s)
Ex1	93.59	22 010	95.41	253	100.00	341	98.15	8390
Ex2	93.10	22 013	95.38	342	79.67	336	98.18	8389
Ex3	93.34	22 009	95.67	281	98.77	337	98.15	8389
Ex4	93.29	22 015	95.55	226	99.19	338	98.19	8389
Ex5	93.36	22 011	94.72	292	97.50	335	98.15	8389
Ex6	93.02	22 012	95.67	278	98.56	332	98.16	8389
Ex7	93.23	22 017	96.24	313	84.12	348	98.19	8389
Ex8	93.59	22 012	95.88	245	97.33	330	98.19	8389
Ex9	93.79	22 008	95.70	335	87.89	347	98.14	8389
Ex10	93.03	22 013	95.41	155	100.00	340	98.25	8389
Ex11	93.24	22 019	95.32	312	99.11	327	98.20	8389
Ex12	93.22	22 018	95.49	324	95.75	335	98.15	8389
Ex13	92.16	22 011	95.20	297	99.61	340	98.16	8389
Ex14	92.18	22 022	95.49	260	98.85	358	98.23	8389
Ex15	92.45	22 021	95.20	301	96.86	343	98.13	8389
Ex16	91.44	22 010	95.47	313	94.88	327	98.16	8389
Ex17	92.22	22 025	95.11	381	99.13	337	98.20	8390
Ex18	92.71	22 026	95.79	155	83.70	353	98.21	8389
Ex19	92.47	22 016	95.08	260	97.33	335	98.19	8389
Ex20	92.16	22 016	95.82	307	90.41	338	98.20	8389
Ex21	92.24	21 999	95.14	361	95.83	350	98.22	8389
Ex22	92.54	22 011	95.64	283	88.00	355	98.18	8390
Ex23	92.59	22 013	95.32	284	93.79	335	98.18	8390
Ex24	92.86	22 018	95.70	325	98.69	327	98.20	8389
Ex25	92.12	22 016	95.32	339	99.11	337	98.16	8389
Avg	**92.80**	**22 014.44**	**95.47**	**288.88**	**94.96**	**339.24**	**98.18**	**8389.16**
Std	**0.59**	**5.80**	**0.32**	**54.34**	**5.84**	**8.54**	**0.03**	**0.37**

Table 11
Coverage of DTMC's transitions for each sensor and the average coverage for *PASTA* and *TransCov*.

Approach	Oxi	Hr	Temp	Abps	Abpd	Gluc	Average	Std
PASTA	0.91	0.63	0.65	0.85	0.86	0.66	0.76	0.13
TransCov	1	1	1	1	1	1	1	0.00

Table 12
Vargha–Delaney effect size (A_{12} measure) for the baseline and instances of *GATE4BSN*.

	PASTA	ValComb	TransCov	Random
PASTA	–	0	0.2408	0
ValComb	1	–	0.328	0
TransCov	0.7592	0.672	–	0.44
Random	1	1	0.56	–

coverage is also carried out completely randomly. Therefore, as this is the approach with more non-deterministic elements, there is a large variation in PTCR compared to other approaches. Finally, the random generation of values coming from the sensors in the Baseline is the least effective approach.

As the effectiveness values, i.e., PTCR, of *PASTA* and *ValComb*, even though the former dominates statistically the latter, are anyhow not very distant, given their respective execution time values ET are very different, with ($ET_{PASTA} > ET_{ValComb}$), we tried to investigate whether the failures raised by *PASTA* and *ValComb* could be associated to the same faults or descend from different causes. In fact, if by running *PASTA* for a longer time we continue to detect again and again failures that are due to one same fault, then the additional effort would not be worthwhile. However, due to the huge amount of collected data and the complexity of the system under test, we have not (yet) been able to localize the faults that are responsible for the observed failures, neither so the team of SA-BSN developers to whom we reported the failures. Therefore, we decided to focus on the "symptoms" and tried to cluster the cumulative set of failing test cases from both approaches to verify whether the two approaches identified the same or different clusters of failures. Our intuition was that if we were able to observe some clusters of failing test cases belonging to only one approach, then we could hypothesize that these are symptoms of possible faults that the other approach did not reach.

With this objective, we used a clustering algorithm well known in the literature, K-means (Lloyd, 1982). However, to use K-means it is necessary to define the value of K, that is, the number of clusters. To choose a good value for K, we use two approaches, the Silhouette Score (Rousseeuw, 1987) and the Davies–Bouldin Score (Davies and Bouldin, 1979). The silhouette score is a measure used to evaluate the quality of clusters created by a clustering algorithm. It measures how well-separated the clusters are, with scores ranging from −1 to 1. A score closer to 1 indicates that the data points are well-clustered and far away from neighboring clusters, while a score closer to −1 indicates that data points have been assigned to the wrong clusters. A score around 0 indicates overlapping clusters. It is commonly used in clustering analysis to determine the optimal number of clusters. The silhouette score threshold is commonly set at 0.5. A score of more than 0.5 indicates a high-quality cluster, whereas a score of less than 0.5 indicates a low-quality cluster. After systematically computing the Silhouette Score with increasing values of K, we chose K = 360 since it is the smallest value of K that produces a Silhouette Score greater than 0.5, which is equal to 0.51 and indicates high-quality clustering.

To confirm that K = 360 is a good choice, we also compute the Davies–Bouldin Score. The value of the Davies–Bouldin Score varies from 0 to infinity, the lower the value, the better the quality of the clustering. An ideal value for the Davies–Bouldin Score is between 0 and 1, where lower values indicate better separation of clusters. In our analysis, the value of Davies–Bouldin Score is equal to 0.78 for K = 360, which confirms that 360 is a good choice.

Considering K = 360, we run K-means with all non-passing test cases from *PASTA* and *ValComb*. We found that there are many clusters containing only test cases from *PASTA* and a few containing test cases only from *ValComb*. Both approaches also shared some clusters. Then, we investigated the clusters containing only *PASTA* test cases. Since we clustered the raw sensor values of test cases, we then converted them into risk labels to make the analysis of the clusters easier. In this way, we could check that the clusters are meaningful since they do not present much discrepancy among the combinations of risk level

Table 13
Examples of risk level combinations that fail for *PASTA* and pass for *ValComb*.

	Combination #1		Combination #2	
	PASTA	**ValComb**	**PASTA**	**ValComb**
Experiment	#1	#1	#1	#1
Patient	233	0	67	0
Time	71 421	129	20 539	488
Oxi	low risk (69.2052)	low risk (73.714222)	high risk (7.74788)	high risk (42.542467)
Ecg	high risk (137.158)	high risk (282.742014)	moderate risk (70.8799)	moderate risk (104.522312)
Term	high risk (31.8519)	high risk (16.608634)	low risk (37.3749)	low risk (37.721862)
Abps	high risk (159.522)	high risk (202.050148)	low risk (96.6736)	low risk (3.913506)
Abpd	low risk (76.5002)	low risk (67.233322)	high risk (127.512)	high risk (278.977966)
Glc	moderate risk (46.5979)	moderate risk (44.084136)	moderate risk (52.7125)	moderate risk (54.968508)
Expected Outcome	Moderate PR	Critical PR	Moderate PR	Critical PR
BSN Outcome	Very Critical PR	Very Critical PR	Very Critical PR	Very Critical PR
Difference	2	1	2	1
Passed?	No	Yes	No	Yes

labels belonging to the same cluster. Finally, we noticed that there are some test cases belonging only to PASTA clusters that correspond to risk level combinations that fail in PASTA but pass in *ValComb*. This may be explained by the fact that *ValComb* contains exactly one test case instance for each possible risk level combination, while in *PASTA*, even if we do not have the guarantee of exploring all possible combinations, for the same combination we may have more than one example. Table 13 provides an excerpt of the test case suites of *PASTA* and *ValComb*. As we can notice, even if combination#1 is present in both test case suites, in *ValComb* it passes while in *PASTA* it fails. The same applies to Combination #2.

Hence we can conclude that – when time permits – it can be useful to continue testing for longer time employing *PASTA*, because this might reveal issues that *ValComb* may skip. This is even more important for safety-critical applications, as SA-BSN is. On the other hand, under tight time constraints, *ValComb* already provides good effectiveness.

6.3. Threats to validity

Notwithstanding our greatest efforts, there are potential threats to the validity of the presented results. We take into consideration four aspects following the classification in Runeson and Höst (2009).

Construct validity: whether the experiments we design are suitable to assess the effectiveness of the three proposed instances of *GATE4BSN*'s, particularly in comparison to Random. Our validation shows that *GATE4BSN* identified unknown failures in SA-BSN and has better effectiveness than Random. We selected two metrics (PTCR and ET), which are widely used to assess effectiveness, but we cannot exclude that other metrics might have led to different results. Also, we compared *GATE4BSN* only with the Random baseline. This limitation stems from the scarcity of testing approaches tailored for BSNs. Another possible threat to validity is the fact that in the approaches that make use of DTMC's, that is, *PASTA* and *ValComb*, the initial state is always the same, the one representing the low risk level. However, this comes from the original implementation of SA-BSN.

Internal validity: whether, in reality, the "treatment" is what caused the observed results rather than other factors. We followed guidelines for performing experiments and the statistical analysis. A frequent internal threat involves the precision of measures influenced by random elements. To address this, we replicated the experiments 25 times, aiming to mitigate this potential threat.

External validity: whether and how far the observations lend themselves to generalization. *GATE4BSN* is general, even though, as we explained in the introduction, for the sake of exposition, we consider a BSN in the healthcare domain. However, so far our results only rely on the SA-BSN case study, and more experiments are needed to better support our conclusions on *GATE4BSN* effectiveness, and also to ascertain if we can use it in other BSN application domains.

Reliability: whether and how much other researchers can replicate the observations. To guarantee reproducibility, we make available all data and settings information.

7. Related work

In this section, we first explore the key characteristics that are essential for effectively testing BSNs. We then proceed to a comparative analysis of the existing literature.

7.1. Key characteristics of BSN testing approaches

When evaluating strategies for effectively testing BSNs, several key characteristics must be considered to ensure comprehensive and reliable assessment. These characteristics form the foundation for developing robust testing strategies for BSNs in real-world scenarios.

7.1.1. Dynamic testing with time window

BSNs involve sensors monitoring real-time physiological data, which are sensitive to dynamic changes in the body and can vary significantly over relatively short periods. Thus, the Dynamic Testing of BSNs with a Time Window ensures that the system is tested under real-world conditions that may fluctuate over time. Testing approaches with this characteristic validate the BSN's ability to respond accurately and adaptively to varying physiological and environmental changes, ensuring reliable performance in real-time scenarios. When testing Self-Adaptive BSNs, this characteristic becomes even more important, due to the adaptive nature of these systems.

7.1.2. Black-box

Black-box testing approaches focus on evaluating the functionality of a system or component based on its inputs and outputs, without any knowledge of its internal code or structure. In many cases, BSNs are tested by third parties or users who may not have access to or knowledge of the system's internal code. Black-box testing approaches allow these external testers to evaluate the system's functionality and ensure that it meets user requirements and medical standards without needing to understand or access the code behind it.

7.1.3. Combinatorial input

Since each sensor in a Body Sensor Network (BSN) measures a distinct vital sign and can generate multiple possible readings, the resulting combinations of sensor inputs multiply rapidly, leading to a vast and complex array of potential scenarios. Combinatorial testing is especially valuable for BSNs as it systematically manages this complexity, enabling efficient testing across diverse conditions to ensure accurate and reliable testing. By examining various combinations of sensor values, it addresses both predictable and unexpected interactions among multiple sensors, identifying potential issues that might only be triggered under specific combinations of conditions.

7.1.4. Risk-based

Risk-based approaches prioritize tests based on the potential risks associated with the software, focusing on areas where failures would have the most significant impact. This strategy aims to identify, assess, and mitigate the highest-priority risks first, ensuring that limited testing resources are applied where they are most needed to protect against critical failures. Risk-based approaches are particularly interesting when testing BSNs since they are often used in critical health-related applications where failures can have serious consequences. By focusing on the highest-risk scenarios, this approach ensures that the BSN's most essential and failure-prone components are rigorously tested, providing a higher level of assurance in its reliability.

7.1.5. Fusion of sensors (Input)

Testing the data fusion function in BSNs, rather than sensors individually, is critical for enhancing the accuracy of health assessments and improving decision-making. By integrating information from multiple sensors, this function provides a comprehensive view of a patient's condition, helping to identify trends or anomalies that might be missed when analyzing individual sensor data. Thus, it must be extensively tested. The correct operation of this function reduces possible noise from the sensors, increases the reliability of the BSN, and ensures resilience against sensor failures, allowing the BSN to maintain functionality even if one sensor malfunctions.

7.2. Analysis of prior studies

Since we did not find testing approaches for BSNs, we discuss three related research areas: (i) gaining confidence through fault diagnosis of BSNs, (ii) model-based testing using Markovian Models (M.M.), and (iii) approaches combining models with combinatorial testing.

Table 14 provides a summary of the comparison between the proposed approaches and related studies in the literature. The symbols ✓and ✗ indicate whether or not a strategy possesses a specific characteristic, respectively. The symbol ∼ denotes that the approach can be readily adapted to address this characteristic, while — signifies that the evaluation is not applicable, as the work is a secondary study and thus does not introduce new approaches. Given the importance of making available the source code or a replication package when proposing an approach for reproducibility, we also evaluate the related works based on this attribute. As shown in the table, only one secondary study provides a replication package (Barbosa et al., 2022), along with one paper that offers a tool (Cai, 2001), which significantly limits the use of most approaches.

When gathering confidence in BSNs, a prominent approach is to use fault diagnosis during system execution. The work in Mahapatro and Khilar (2012) uses multi-sensor fusion approaches to identify faulty sensors by assuming correlations between sensorial measurements and diseases. Their algorithm identifies permanent faults, intermittent faults, and transient faults, by re-executing the test cases with a set frequency. Based on the comparison between readings obtained from correlated sensors (e.g., Heart Beat Rate and Body Pressure), it decides whether a sensor is faulty. Instead Zhang et al. (2017) describe a method using hidden Markov models (HMMs) to identify faulty readings of individual sensors. Even if both approaches do not constitute a testing process, they still look for faults in the sensors over time (dynamic testing with time window). Neither of the two previously mentioned approaches is risk-based or consider combinatorial inputs. Also, they are both black-box since they evaluate a sensor based on input data. Finally, the work in Mahapatro and Khilar (2012) assesses the fusion of sensors by considering their spatial correlation, that is, the correlation between different sensors, to determine if a sensor is faulty. Differently, the work in Zhang et al. (2017) considers temporal correlation, that is, the correlation between the current value of a sensor and its history. Therefore, sensor fusion is not evaluated to any extent. None of these approaches address the data fusion algorithm at the application layer level, but at the body level, as opposed to ours.

A taxonomy for model-based testing approaches is proposed in Utting et al. (2012) and their work discusses Markov chains for statistical testing, which can be used for test case generation or specifying oracles. One of the tools collected in their secondary study is JUMBL (Prowell, 2003), a Java class library that supports all phases of model-based statistical testing and offers a command-line interface designed to interact with Markov chain usage models. Also, making use of Markov models as usage models, Prowell (Prowell, 2005) defends the application of concurrency operators to the test cases generated from simple Markov chain usage models, to create sophisticated test cases when testing complex systems. Markov models are also employed to test case prioritization as shown by Barbosa et al. (2022) in their systematic literature review. This study shows that Markovian models are used for test case prioritization in six contexts: controlled Markov chains, usage models, model-based testing, regression testing, statistical testing, and random testing.

Finally, Markov chains have been used in the application of cybernetic principles to software testing (Cai, 2001, 2002). Building on the concept of software cybernetics, the Controlled Markov Chains (CMC) approach treats software testing as a control problem. The work in Cai (2001) discusses how to do so and how the CMC approach to software testing determines an optimal testing strategy. In their subsequent work (Cai, 2002), the same authors examine the behavior of the optimal test profile identified through the CMC approach to software testing and introduce the concept of adaptive software testing. Adaptive software testing adjusts the software testing strategy online by using testing data collected during software testing in response to changes in the understanding of the software under test. Among these works, the works in Utting et al. (2012) and Barbosa et al. (2022) are secondary studies, and therefore they do not propose novel testing approaches. On the other hand, the works in Prowell (2003) and Prowell (2005) do not employ dynamic testing nor combinatorial input. Test cases are derived from a probabilistic usage model, making the approach black-box, and non-risk-based. Test cases are a fusion of input data since they are paths in the usage model. Finally, the works in Cai (2001) and Cai (2002) consider a dynamical system and the testing should also be dynamic to either adapt to changes in the system or to consider the result of previous test cases. They are also black-box since test cases are extracted from the optimal test profile. With respect to the other key characteristics, these approaches may be adapted to consider them.

Intertwining models and combinatorial test case generation to the systematic collection of safety evidence has shown effectiveness in the work by Nguyen et al. (2012). In this paper, test sequences are derived from inferred models and complemented with selective test input combinations. Dynamic testing is not addressed, and their approach does not incorporate risk-based strategies. However, if the inferred model is based on the specification, their method can be adapted to become a black-box approach. Additionally, they explore the t-way combinatorial fusion of input data as in our approach. Following this line of research, Gannous and Andrews (2019) employ a previously proposed framework for testing safety-critical systems, known as Model-Combinatorial based testing (MCbt). MCbt is a framework that proposes an integration of model-based testing, fault analysis, and

Table 14
Comparison of our approaches and the existing literature.

Characteristics	Related work								GATE4BSN		
	Fault diagnosis of BSN		Model-based testing with M.M.				Models and Comb. testing				
	Mahapatro and Khilar (2012)	Zhang et al. (2017)	Utting et al. (2012)	Prowell (2003, 2005)	Barbosa et al. (2022)	Cai (2001, 2002)	Nguyen et al. (2012)	Gannous and Andrews (2019)	PASTA	ValComb	TransCov
Dynamic testing with time window	✓	✓	–	✗	–	✓	✗	✗	✓	✗	✓
Black-box	✓	✓	–	✓	–	✓	~	✓	✓	✓	✓
Combinatorial input	✗	✗	–	✗	–	~	✓	✓	✓	✓	✗
Risk-based	✗	✗	–	✗	–	~	✗	~	✓	✗	✗
Fusion of sensors (Inputs)	✓	✗	–	✓	–	~	✓	✓	✓	✓	✓
Replicable work	✗	✗	✗	✗	✓	[✓, ✗]	✗	✗	✓	✓	✓

combinatorial testing to produce the maximum number of evidences for an efficient safety certification process but was never actually used to derive a specific testing approach. In this paper, they present a concrete application of MCbt with an application to a case study. As in the previous work, dynamic testing is not addressed, but their approach allows the incorporation of risk-based strategies by deriving failure paths for behavioral models of each component in the system. These paths may be used to generate high-risk test cases. Their approach is also black-box since the tester creates behavior models using system requirements and specifications. Test paths are also combined for two or more components making this strategy combinatorial and based on fusion of input data. T-way combinatorial testing is mentioned as future work to overcome the state space explosion problem.

8. Conclusion and future works

This paper proposed *GATE4BSN*, a novel approach to test BSN applications. *GATE4BSN* is a generic and abstract approach that is instantiated in three approaches, namely: (i) *PASTA*, which simulates patients by considering a set of sensors and by mimicking the trend of each sensor via a DTMC; (ii) *ValComb*, which makes use of all possible combinations of sensor risk levels to explore different behaviors of a patient; and (iii) *TransCov*, which employs a dummy patient in which all the transitions in the DTMCs are labeled with equal probability. Then, their DTMCs are executed until all the transitions are covered.

We evaluated the three approaches by using them to test a self-adaptive BSN system from the literature, and they were able to detect some unknown failures. Through experiments, we compared the three approaches among themselves and with a random baseline approach. We could then conclude that our approaches outperform the random baseline in terms of effectiveness. Also, statistical analysis showed that *PASTA* is the most effective instance of *GATE4BSN*. *ValComb* is in second place with effectiveness value close to *PASTA* but with ET approximately 76 times smaller than for *PASTA*. By clustering non-passing test cases from *PASTA* and *ValComb* we noticed that there exist many test cases containing a combination of risk labels that are failing in *PASTA* but are passing in *ValComb*. This indicates that *PASTA* is effective in finding faults that are not reported by *ValComb*. Finally, *TransCov* is also fast but due to more than one non-deterministic elements in its implementation, its PTCR is very unstable, presenting large variations from one execution to another.

In future works, we plan to test other BSNs in different domains. Also, we will explore how *GATE4BSN* and its instances can be used for testing other systems that would benefit from our idea of simulating the input (our patients' simulation), such as state-based systems (e.g., CPSs and autonomous systems), which often depend on a specific order of inputs that eventually lead to failures (Kuhn et al., 2023). An example of these systems is autonomous drone navigation systems (Lindvall et al., 2017) that may benefit from risk-based testing since it focuses on high-risk situations, such as flying near obstacles or in inclement weather, which could lead to loss of control or collision. Industrial robotics for manufacturing (Gotlieb et al., 2021) is another example of an application that could take advantage of dynamic testing with a time window, since real-time adjustments are necessary for robots to respond to moving parts, changing positions, or unexpected obstacles. Testing within a time window ensures that robots can make adjustments quickly to avoid collisions or errors.

CRediT authorship contribution statement

Samira Silva: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Ricardo Caldas:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Patrizio Pelliccione:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Antonia Bertolino:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Patrizio Pelliccione reports financial support was provided by Italian Ministry of University and Research (MUR). Patrizio Pelliccione reports financial support was provided by Knut and Alice Wallenberg Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially funded by (a) the European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem (i) grant ECS00000041 - VITALITY – CUP: D13C21000430001 and (ii) PNRR Missione 4 Componente 2 Investimento 1.3, grant PE0000020 – CHANGES – CUP: D53C22002560006; (b) the MUR (Italy) Department of Excellence 2023–2027 for GSSI; (c) the PRIN project P2022RSW5 W - RoboChor: Robot Choreography; (d) the PRIN project 2022JKA4SL - HALO: etHical-aware AdjustabLe autOnomous systems; (e) the European HORIZON-KDT-JU research project MATISSE "Model-based engineering of Digital Twins for early verification and validation of Industrial Systems", HORIZON-KDT-JU-2023-2-RIA, Proposal number: 101140216-2, KDT232RIA _00017, and (f) the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Data availability

The replication package for the paper is available at https://github.com/samirasilva/Paper_JSS.

References

Alrige, M., Chatterjee, S., 2015. Toward a taxonomy of wearable technologies in healthcare. In: New Horizons in Design Science: Broadening the Research Agenda: 10th International Conference, DESRIST 2015, Dublin, Ireland, May 20-22, 2015, Proceedings 10. Springer, pp. 496–504.

Arcuri, A., Briand, L., 2014. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Softw. Test. Verif. Reliab. 24 (3), 219–250.

Aziz, O., Lo, B., King, R., Darzi, A., Yang, G.-Z., 2006. Pervasive body sensor network: an approach to monitoring the post-operative surgical patient. In: International Workshop on Wearable and Implantable Body Sensor Networks. BSN'06, IEEE.

Barbosa, G., de Souza, É.F., dos Santos, L.B.R., da Silva, M., Balera, J.M., Vijaykumar, N.L., 2022. A systematic literature review on prioritizing software test cases using Markov chains. Inf. Softw. Technol. 147, 106902.

Barr, E.T., Harman, M., McMinn, P., Shahbaz, M., Yoo, S., 2015. The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering 41 (5), 507–525. http://dx.doi.org/10.1109/TSE.2014.2372785.

Burchfield, R., Venkatesan, S., 2010. A framework for golf training using low-cost inertial sensors. In: 2010 International Conference on Body Sensor Networks. IEEE, pp. 267–272.

Cai, K.-Y., 2001. Optimal test profile in the context of software cybernetics. In: 2nd Asia-Pacific Conference on Quality Software. IEEE, pp. 157–166.

Cai, K.-Y., 2002. Optimal software testing and adaptive software testing in the context of software cybernetics. Inf. Softw. Technol. 44 (14), 841–855.

Conroy, L., Ó Conaire, C., Coyle, S., Healy, G., Kelly, P., Connaghan, D., O'Connor, N.E., Smeaton, A.F., Caulfield, B., Nixon, P., 2009. TennisSense: a multi-sensory approach to performance analysis in tennis. In: 27th International Society of Biomechanics in Sports Conference, 17-21,Limerick, Ireland.

Davies, D.L., Bouldin, D.W., 1979. A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. (2), 224–227.

de Oliveira Neto, F.G., Torkar, R., Feldt, R., Gren, L., Furia, C.A., Huang, Z., 2019. Evolution of statistical analysis in empirical software engineering research: Current state and steps forward. J. Syst. Softw. 156, 246–267.

Gannous, A., Andrews, A., 2019. Integrating safety certification into model-based testing of safety-critical systems. In: 2019 IEEE 30th International Symposium on Software Reliability Engineering. ISSRE, IEEE, pp. 250–260.

Gil, E.B., Caldas, R., Rodrigues, A., da Silva, G.L.G., Rodrigues, G.N., Pelliccione, P., 2021. Body sensor network: A self-adaptive system exemplar in the healthcare domain. In: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, IEEE, pp. 224–230.

Gotlieb, A., Marijan, D., Spieker, H., 2021. Testing industrial robotic systems: A new battlefield!. Softw. Eng. Robot. 109–137.

Gravina, R., Fortino, G., 2020. Wearable body sensor networks: state-of-the-art and research directions. IEEE Sens. J. 21 (11), 12511–12522.

Guk, K., Han, G., Lim, J., Jeong, K., Kang, T., Lim, E.-K., Jung, J., 2019. Evolution of wearable devices with real-time disease monitoring for personalized healthcare. Nanomaterials 9 (6), 813.

Kruskal, W.H., Wallis, W.A., 1952. Use of ranks in one-criterion variance analysis. J. Amer. Statist. Assoc. 47 (260), 583–621.

Kuhn, D.R., Kacker, R.N., Lei, Y., et al., 2010. Practical combinatorial testing. NIST Spec. Publ. 800 (142), 142.

Kuhn, D.R., Raunak, M., Kacker, R.N., 2023. Ordered t-way combinations for testing state-based systems. In: 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops. ICSTW, IEEE, pp. 246–254.

Kuhn, D.R., Wallace, D.R., Gallo, A.M., 2004. Software fault interactions and implications for software testing. IEEE Trans. Softw. Eng. 30 (6), 418–421.

Lai, X., Liu, Q., Wei, X., Wang, W., Zhou, G., Han, G., 2013. A survey of body sensor networks. Sensors 13 (5), 5406–5447.

Lee, T.-G., Lee, S.-H., 2015. Design of wearable bio-patch system platform in human healthcare environment. Indian J. Sci. Technol. 8 (17).

Lindvall, M., Porter, A., Magnusson, G., Schulze, C., 2017. Metamorphic model-based testing of autonomous systems. In: 2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing. MET, IEEE, pp. 35–41.

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Trans. Inform. Theory 28 (2), 129–137.

Mahapatro, A., Khilar, P.M., 2012. Fault diagnosis in body sensor networks. Int. J. Comput. Inf. Syst. Ind. Manag. Appl. 5.

Nguyen, C.D., Marchetto, A., Tonella, P., 2012. Combining model-based and combinatorial testing for effective test case generation. In: International Symposium on Software Testing and Analysis. pp. 100–110.

Nie, C., Leung, H., 2011. A survey of combinatorial testing. ACM Comput. Surv. 43 (2), 1–29.

Osmani, V., Balasubramaniam, S., Botvich, D., 2007. Self-organising object networks using context zones for distributed activity recognition. In: 2nd International ICST Conference on Body Area Networks.

Pansiot, J., Lo, B., Yang, G.-Z., 2010. Swimming stroke kinematic analysis with BSN. In: 2010 International Conference on Body Sensor Networks. IEEE, pp. 153–158.

Picco, G.P., 2010. Software engineering and wireless sensor networks: Happy marriage or consensual divorce? In: FSE/SDP Workshop on Future of Software Engineering Research. FoSER '10, ACM, New York, NY, USA, ISBN: 9781450304276, pp. 283–286.

Prowell, S.J., 2003. JUMBL: A tool for model-based statistical testing. In: 36th Annual Hawaii International Conference on System Sciences, 2003. IEEE, pp. 9–pp.

Prowell, S.J., 2005. Using Markov Chain usage models to test complex systems. In: 38th Annual Hawaii International Conference on System Sciences. IEEE, p. 318c.

Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. 20, 53–65.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14, 131–164.

Tatbul, N., Buller, M., Hoyt, R., Mullen, S., Zdonik, S., 2004. Confidence-based data management for personal area sensor networks. In: 1st International VLDB Workshop on Data Management for Sensor Networks. pp. 24–31.

Trivedi, K.S., 2016. Probability and Statistics with Reliability, Queueing and Computer Science Applications, second ed. John Wiley and Sons, New York, ISBN: 9780471460817.

Utting, M., Pretschner, A., Legeard, B., 2012. A taxonomy of model-based testing approaches. Softw. Test. Verif. Reliab. 22 (5), 297–312.

Vargha, A., Delaney, H.D., 2000. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. J. Educ. Behav. Stat. 25 (2), 101–132.

Wai, A.A.P., Fook, F.S., Jayachandran, M., Biswas, J., Lee, J.-E., Yap, P., 2010. Implementation of context-aware distributed sensor network system for managing incontinence among patients with dementia. In: 2010 International Conference on Body Sensor Networks. IEEE, pp. 102–105.

Whittaker, J.A., Thomason, M.G., 1994. A Markov Chain model for statistical software testing. IEEE Trans. Softw. Eng. 20 (10), 812–824.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Springer Science & Business Media.

Yu, L., Lei, Y., Kacker, R.N., Kuhn, D.R., 2013. ACTS: A combinatorial test generation tool. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation. IEEE, pp. 370–375.

Zhang, H., Liu, J., Li, R., Le, H., 2017. Fault diagnosis of body sensor networks using hidden Markov model. Peer- To- Peer Netw. Appl. 10, 1285–1298.

Samira Silva is a PhD candidate at the Gran Sasso Science Institute (GSSI) in L'Aquila, Italy. Her current research focuses on self-adaptive testing, particularly in the context of Self-Adaptive Body Sensor Networks (SA-BSN). In addition to her research, she has worked as a Lecturer at the Federal University of Ouro Preto (UFOP), the University of Itaúna (UIT), and the State University of Minas Gerais (UEMG) in Brazil, between 2016 and 2021. Although she has a strong background in Machine Learning and Computer Vision, over the past four years, she has redirected her efforts towards software engineering, with a specific emphasis on software testing. For more information, please visit her Lattes curriculum: http://lattes.cnpq.br/3321124706549203.

Ricardo Caldas is a postdoctoral researcher at Gran Sasso Science Institute (GSSI, Italy). His research topic is software design and assurance provision for autonomous systems. He focuses on resilience as a key enabler to long-living autonomous systems by contributing to software architecture and testing for multi-agent heterogeneous applications, control-based self-adaptation, and automated explanation of violated properties. His contributions extend to software applications implemented in the robot operating system (ROS), in several domains such as robotics, autonomous driving, and healthcare. He received his Ph.D. in Computer Science and Engineering from the Chalmers University of Technology in Sweden in 2024. More information is available at https://ricardocaldas.me.

Patrizio Pelliccione is a Full Professor and Director of the computer science area at Gran Sasso Science Institute (GSSI), Italy, and also an Adjunct Professor at the University of Bergen, Norway. His research topics are mainly in software engineering, software architecture modeling and verification, and autonomous systems. He received his PhD in computer science from the University of L'Aquila in Italy. Thereafter, he worked as a senior researcher at the University of Luxembourg in Luxembourg, then assistant professor in the University of L'Aquila in Italy, then Associate Professor at Chalmers|University of Gothenburg in Sweden and University of L'Aquila. He has been on the organization and program committees for several top conferences, and he is a reviewer for top journals in the software engineering domain. He is very active in European and National projects. In his research activity, he has collaborated with several companies. More information is available at http://www.patriziopelliccione.com.

Antonia Bertolino is currently an Adjunct Professor at the Gran Sasso Science Institute (GSSI), L'Aquila, Italy. Previously, she has been a Research Director of the Italian National Research Council (CNR), at the Institute for Information Science and Technologies "Alessandro Faedo" (ISTI), in Pisa, Italy. Her research interests cover software and services validation, testing, and monitoring, and on these topics she has published 250 papers in international journals, conferences and workshops. She has led or participated in several collaborative European and national projects. Currently she is the Software Testing Area Chair for Elsevier Journal of Systems and Software, and an Associate Editor of Wiley Journal of Software: Evolution and Process. She serves regularly in the Program Committee of the most renowned conferences in the field of Software Engineering, such as ESEC-FSE, ASE and ICSE, and in software testing and analysis, such as ISSTA and ICST. She was the General Chair of the 2015 edition of the flagship ACM/IEEE ICSE in Florence (Italy), and General/Program Co-Chair, among others, of AST2020, QUATIC2018, ICST2012, CBSE2011, ESEC/FSE2007.