Formal Reliability Analysis of a Typical FHIR Standard based E-Health System using PRISM

Usman Pervez*, Osman Hasan*, Khalid Latif*, Sofiène Tahar[§], Amjad Gawanmeh[¶] and Mohamed Salah Hamdi[‖]
*School of Electrical Engineering and Computer Engineering, National University of Sciences & Technology, Islamabad, Pakistan
Email: {usman.pervez,osman.hasan,khalid.latif}@seecs.nust.edu.pk
[§]Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada
Email: tahar@ece.concordia.ca
[¶]Department of Electrical and Computer Engineering, Khalifa University, UAE
Email: amjad.gawanmeh@kustar.ac.ae
[‖]Information Systems Department, Ahmed Bin Mohammed Military College, Doha, Qatar
Email: mshamdi@abmmc.edu.qa

Abstract—**Fast Health Interoperable Resources (FHIR) is the recently proposed standard from HL7. Its distinguishing features include the user friendly implementation, support of built-in terminologies and for widely-used web standards. Given the safety-critical nature of FHIR, the rigorous analysis of e-health systems using the FHIR is a dire need since they are prone to failures. As a first step towards this direction, we propose to use probabilistic model checking, i.e., a formal probabilistic analysis approach, to assess the reliability of a typical e-health system used in hospitals based on the FHIR standard. In particular, we use the PRISM model checker to analyze the Markov Decision Process (MDP) and Continuous Time Markov Chain (CTMC) models to assess the failure probabilities of the overall system.**

I. Introduction

There is growing interest in e-health activities in all healthcare centers around the globe. However, one of the biggest challenges in this regard is the interoperability of the underlying e-health systems. To overcome this problem, various predefined e-health standards have been developed and they have to be followed at the national and international levels to facilitate the patient information and diagnosis exchange between healthcare professionals. Health Level Seven (HL7) [1], a non-profitable organization that is accredited by the American National Standards institute (ANSI), developed standards which are widely used for the exchange, integration, sharing, and retrieval of electronic health information. These standards help in delivery, clinical management and evaluation of health services and further improve the knowledge transformation among healthcare providers, vendor community, patients, government agencies and other stake holders.

HL7 launched version 2.x Messaging Standard in 1989 for health services to support hospital workflows. In 1995, the version 3.x was launched, which is primarily based on the Reference Information Model (RIM), HL7 Development Framework (HDF) and object oriented principles [1]. Despite a lot of effort by HL7 as well as the community around it, version 3 was never fully implemented by any major vendor and thus the earlier version (2.4 to 2.7) is still widely used by many providers. The HL7 community has now moved on and is now working on a new version of the standard called the Fast Health Interoperable Resources (FHIR) [2], which is easily implementable; has built-in clinical and administrative terminologies and is based on web standards like HTTP, Atom, XML and JSON. Based on these unique features, FHIR is expected to attract a lot of interest from health practitioners in the near future.

The fact that the information shared using the FHIR standards is used for interpreting results of blood and MRI tests and diagnosing diseases, like cancer and brain hemorrhage, makes the reliability of the the e-health system using this standard extremely important. However, despite an enormous amount of work available on the design of efficient and novel schemes for e-health systems, in particular the HL7 standard, very little attention is paid to their reliability and security. Most of the testing of healthcare systems is based on simulation-based methods, which cannot guarantee accurate analysis results due to their inherent incompleteness. This makes existing health-care systems, based on HL7, prone to errors, which is an extremely undesirable condition considering the safety-critical nature of health related systems.

Formal methods [3], which use mathematical analysis methods in a computer, are capable of overcoming the inaccuracy limitations of simulation. Some prominent examples of formal methods based verification of healthcare systems include the verification of collaborative and agent based workflows in healthcare [4], [5], software components in medical devices [6], [7], ambient assisted systems [8], or healthcare requirements [9]. The Communicating Sequential Processes (CSP) have been adopted as a formal method language to extensively formalize the system specifications and utilize it as a useful extension in the specification refinement in the system engineering lifecycle [10]. Similarly, a probabilistic model checker has been used to model and verify the treatment therapies of Tuberculosis and HIV [11]. Recent work shows some trials on using formal methods in the verification of software components in health care systems. For instance, model checking has been used to verify the reliability of software used in medical devices for an infusion pump [12]. Model-Based Testing has also been used to generate test cases for healthcare systems [13]. A formal model for e-Healthcare readiness assessment was also proposed in [14]. Despite the above-mentioned formal methods work in ascertaining the correctness of healthcare systems, their usage for analyzing the functionality and performance of health-care standards, like HL7, has been very rare. Similarly, to the best of our knowledge, formal methods have never been used to assess the recently proposed FHIR standard.

Given the safety-critical nature of the FHIR standard, it is a dire need to assess its functionality, reliability and performance using formal methods. As a first step towards this direction, we present in this paper a formal reliability analysis of a typical e-health system, based on the the FHIR standard. The main idea is to develop a Markovian model of the FHIR standard and analyze it using the probabilistic model checker PRISM [15]. In particular, we use Markov Decision Processes (MDP) [16] in PRISM to find the probability of occurrence of wrong results (failures) in the considered system following the FHIR standard. Similarly, we use Continuous time Markov Chain (CTMC) [17] in PRISM to find the frequency of wrong results (failures) among a certain number of requests. The proposed approach provides more accurate results than traditional counterparts due to the exploration of an exhaustive state-based model of the FHIR standard.

The rest of paper is organized as follows: Section II describes some preliminaries about model checking and PRISM to facilitate the understanding of the paper. The FHIR standard for the typical hospital scenario is described in Section III. This is followed by the proposed MDP and CTMC models for this scenario and the corresponding formally verified properties in Section IV and V, respectively. Finally, Section VI concludes the paper.

II. Model Checking and PRISM

Model checking [18] is primarily used in the verification of reactive systems, i.e., the systems whose behavior is dependent on time and their environment, like controllers of digital circuits and communication protocols. The inputs to a model checker include the finite-state model of the system that needs to be analyzed along with the intended system properties, which are expressed in temporal logic. The model checker automatically and exhaustively verifies if the properties hold for the given system while providing an error trace in case of a failing property. The state-space of a system can be very large, or sometimes even infinite. Thus, it becomes computationally impossible to explore the entire state-space with limited resources of time and memory. This problem, termed as state-space explosion, is usually resolved by developing abstract, less complex, models of the system. Moreover, many efficient techniques, like symbolic and bounded model checking, have been proposed to alleviate the memory and computation requirements of model checking.

Probabilistic model checking is a variant of traditional model checking, where the probabilistic behavior of the given system is described using a Markovian model. This model can then be used to verify probabilistic properties. Many probabilistic model checkers, including PRISM [15], YMER [19], MRMC [20], VESTA [21] and ETMCC [22] are available. The main objective of the proposed work in this paper is to find steady state probabilities of failures, which are usually modeled as continuous time Markov chains (CTMC). Both, YMER and VESTA do not support steady state probabilities and thus cannot be used for our purpose. The PRISM model checker has been reported as the most efficient one in terms of memory consumption compared to MRMC and ETMCC. It also supports a wider range of models, including Markov Decision Processes (MDPs), Discrete Time Markov Chains (DTMCs) and Continuous time Markov Chains (CTMCs) and thus has been selected for analyzing the FHIR standard in this paper.

The models in PRISM are described using a state-based language called the PRISM language. Modules and variables are basic components of the modeling language. A model can consist of a number of modules whose state at a given time is represented by the values of local variables defined in those modules. The values of local variables of all the modules define the overall state of the system. A set of guarded commands describe the behavior of each module:

$$[\,]\ guard \rightarrow prob_1 : update_1 + \cdots + prob_n : update_n;$$

The *guard* is a predicate over all variables and a command is enabled when its guard becomes true. Each $update_i$ defines a possible transition with probability $prob_i$. PRISM provides support for a variety of property specifications such as PCTL, CSL, LTL and PCTL*. For example:

$$S_{\geq 0.99}[\text{``}normal\text{''}]$$

is the steady state probability of *normal* state $\geq$ 0.99. PRISM supports verification and analysis of time based properties which we use for the time based analysis of Markovian models. These properties are analyzed by associating a certain reward with each state of the model through a reward structure. PRISM also allows the use of customized properties using the *filter* operator: $filter(op, prop, states)$, where op represents the filter operator (min, avg, max), $prop$ represents the PRISM property and $states$ (optional field) represents the set of states over which to apply the filter.

III. A Typical Hospital Scenario using the FHIR Standard

For analysis purposes, we consider a typical FHIR based e-health system that is used in a hospital setting. This system provides the means of communication between various stake holders. The overall architecture of the system scenario is depicted in the Fig. 1. In the hospital, there is a doctor who examines the patient, a Lab that takes the medical tests of the patient, and an information resource, which is actually called as a Diagnostic Resource [23]. All information exchanged between the doctor and the patient will be done via the Diagnostic Resource.

It is basically a Client/Server [24] system in which the doctor is a client and the Lab is a server. Whenever a patient visits the doctor, the doctor examines the patient and refers the patient to the Lab in order to undergo some medical tests [25] like blood test, urine test, sugar test etc. In the meanwhile the doctor makes a request to the Lab through the FHIR standard that a patient is coming for some tests. When the patient visits the lab, the Lab staff takes the samples and finally sends the medical reports to the doctor according to the FHIR standard, after performing diagnostic tests.

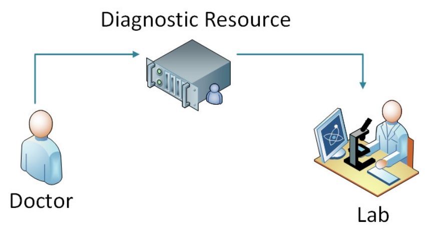

Fig. 1. A Typical FHIR Standard based System Architecture

According to the FHIR standard, the doctor's request can be termed as accepted, partially accepted, rejected, suspended, failed or completed depending on the action taken by the Lab staf. The state representation of this behaviour is depicted in Fig. 2. The request is *Accepted* if the Lab staff accepts the request and is willing to take all the tests of the patient. The request is *Partially Accepted*, if the Lab staff agrees to take at least one test and does not perform all the tests requested by the doctor. The request is termed as *Rejected* if the Lab staff is not willing to take the tests due to some reasons. The request is called *Suspended* if either the doctor or the Lab staff suspends the request due to some reasons. The request is termed as *Failed* if the Lab staff fails to perform the test. The request is called to be *Completed* if the Lab staff successfully performs the test and provides the medical reports to the doctor.

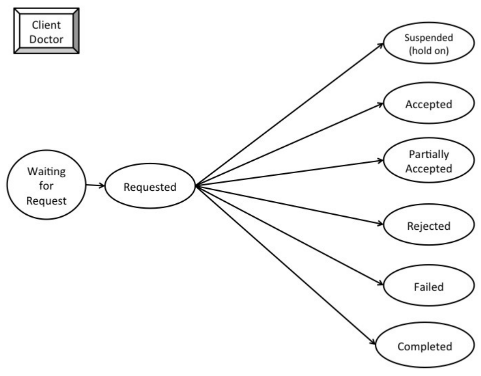

Fig. 2. State Machine depicting the Doctor's Behavior

The state machine that depicts the working of the Lab staff is given in Fig. 3, which is an adaptation of the FHIR standard [26] for our context. The state machine starts from the initial state *Waiting for Request*, when there is no request sent by the doctor. When the request arrives, the state machine will move to the *Requested* state. In the meanwhile, if the doctor or the Lab staff suspends the request due to some reason, the state machine will move to *Suspended* state, from where it will either go to the *Received* state or to the initial state, based on the decision of the doctor or the Lab staff. When the request is received, the Lab staff will take decision that either they can perform all or some or none of the tests. Based on the decision taken by the Lab staff, the state machine will either move to the *Accepted*, *Partially Accepted* or *Rejected* state. There is a chance that the Lab staff, incidentally marks the request as completed without undergoing the tests, and if it happens, then the state machine will move directly from *Accepted* state to the *Completed* state. The request will be processed in the *In Progress* state. In this state the medical equipment performs the test of the patients blood or urine sample. If the equipment fails to test the sample, then the state machine will move to the *Failed* state, if equipment successfully completes the test, then the state machine will move to the *Actual Results Review* and if the equipment generates an error due to some reason, then the state machine will move to the *Error in Results Review* state. It is possible that the medical equipment fails to perform the test but provides the results of a previous test from the cache. Under such circumstances, the state machine will move to the *False Results Review* state. In the review states, the Lab staff will finally review the medical reports of the tests and the state machine will eventually move to the *Completed* state. When the state machine moves to the states *Suspended*, *Accepted*, *Partially Accepted*, *Rejected*, *Failed* or *Completed*, the Lab staff state machine automatically updates the status of the state machine, corresponding to the doctor's behavior, accordingly as well.

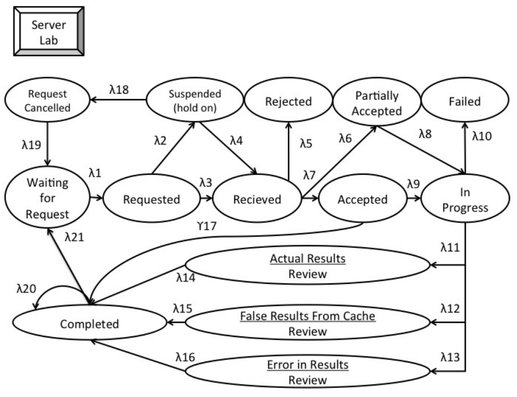

Fig. 3. State Machine depicting the Lab Staff Behavior

Based on the above-mentioned behavior, there are two critical situations in the state machine of the Lab staff that need special attention. The first one is when the Lab staff incidentally marks the request as completed without performing the tests, and the second one is, when the medical equipment gives false results from the cache. The probability of occurrence of these two events would be the main focus of our formal reliability analysis of the given system.

IV. Single Request Processing using MDPs

In this section, we use MDPs to model the scenario where a doctor generates a single request and the request is processed by the Lab staff. This model is then used to calculate the probability of occurrence of false results or failures.

When the doctor generates a request, the request will be processed by the Lab staff based on the state machine, given in Fig. 3. The choices in this behavior are probabilistic and thus the real-world behavior of the Lab staff can be captured using a Markov Decision Process (MDP), which allows probabilistic decisions by including the appropriate state transition probabilities. In a MDP, the transition from state ***S*** to a state ***S'*** is based on a probabilistic decision maker, and the transition only depends on the present state ***S***. The transition probabilities for going from state ***S*** to state ***S'*** are governed by the following equation

$$P_a(S, S') = P_r(S_{t+1} = S'|S_t = S, a_t = a) \tag{1}$$

where ***Pr*** is the transition probability and ***a*** is the action performed by the decision maker. The system is modeled in its mathematical form in terms of a Transition Probability Matrix ***P*** [27], which represents various transition rates. To calculate the probability of next state ***S'***, the Transition Probability Matrix ***P*** is multiplied with the current state probability as follows

$$P_r(S') = P_r(S) * P \tag{2}$$

Based on our MDP model of the behavior of the system, described in the previous section, the following probabilistic properties related to the system failure are verified:

$$Pmin = ?[F\ x = K] \tag{3}$$

where ***Pmin*** is the output probability, ***F*** indicates future, ***x*** is one of the 14 states, i.e., [0..13], according to our system behavior given in Fig. 3, and ***K*** is a variable whose values lies in [0..13]. For example, if ***K=1***, then the output will be the transition probability to state ***S'=1***. Similarly, we can find other state transition probabilities by changing the values of variable ***K***. The second main property that we verified is as follows:

$$Pmin = ?[F\ var = 1] \tag{4}$$

where ***var*** is a variable with an initial value ***0***. In our MDP model, when the transition from state *Accepted* to state *Completed* takes place, the value of the variable ***var*** is updated to ***1***. Thus the property, expressed in Equation (4), allows us to find the transition probability from state *Accepted* to state *Completed.*

The Transition Probability Matrix for the analysis of the FHIR based typical hospital e-health system, given in Fig. 3, is provided in Table I. The probabilities in this matrix have been chosen based on various statistical analysis done on the health care professionals and systems.

Based on the above-mentioned, transition probabilities, we find the probabilities of events associated with the *Suspension, Rejection, Partially Acceptance, Failure* and *Completed* states by changing the value of the variable **K** in Equation (3). Similarly, by using the property in Equation (4), we find the probability of marking the request as completed without performing the tests. The results of these verifications are summarized in Table. II

TABLE I
State Transition Properties

State Transition	Probability	State Transition	Probability
λ_1	1.0	λ_{12}	0.1
λ_2	0.1	λ_{13}	0.2
λ_3	0.9	λ_{14}	1.0
λ_4	0.3	λ_{15}	1.0
λ_5	0.1	λ_{16}	1.0
λ_6	0.2	λ_{17}	0.1
λ_7	0.7	λ_{18}	0.7
λ_8	1.0	λ_{19}	0.0
λ_9	0.9	λ_{20}	1.0
λ_{10}	0.1	λ_{21}	0.0
λ_{11}	0.6		

TABLE II
Verification Results for the MDP model

Events	Probability	Events	Probability
Suspension	1.0	Completion	0.2
Rejection	0.1	Actual Results	0.7
Partially Acceptance	0.9	False results	1.0
Acceptance	0.3	Error in Results	0.9
Failure	0.1	Request Marked as Completed Without Performing Tests	0.1

V. Continuous Request Generation using CTMCs

Now, we consider the case when the doctor is continuously generating requests, with some rate, to be processed by the Lab staff. This behavior can be described using the Continuous Time Markov Chain (CTMC) [17] and based on this model we can find the frequency of false results. The CTMC includes the total number of states ***S***, initial probability distribution of states and the transition rate matrix ***Q***. The next transition state probabilities are calculated in the CTMC as follows:

$$P_t' = P_t * Q \tag{5}$$

Based on the above-mentioned CTMC model, we verified the following properties using the PRISM model checker.

$$P = ?\ [F = T\ x = K] \tag{6}$$

where ***P*** is the output probability, ***F*** indicated future, ***T*** is the time in hours, ***x*** represents one of the possible states from [0..13] and ***K*** is a variable whose value lies in [0..13]. The second formally verified property is

$$P = ?\ [F = T\ var = K] \tag{7}$$

where ***var*** is the variable that acts like a counter and counts the total number of path transitions.

We executed the CTMC model for the state-transition rates, given in Table III, for ***1 hour***, and calculated how many

TABLE III
STATE TRANSITION PROBABILITIES

State Transition	Rate	State Transition	Rate
λ_1	10	λ_{12}	1
λ_2	2	λ_{13}	1
λ_3	8	λ_{14}	1
λ_4	1	λ_{15}	1
λ_5	1	λ_{16}	1
λ_6	1	λ_{17}	1
λ_7	8	λ_{18}	1
λ_8	1	λ_{19}	1
λ_9	9	λ_{20}	0
λ_{10}	1	λ_{21}	10
λ_{11}	7		

requests produced false results. When the counter variable is placed in the path from state *Accepted* to the state *Completed*, it acts like a path transition counter for this transition when the requests were marked as completed without performing the tests. So by using Equation (7), we can find the probability of ***K*** requests that were marked as completed without performing results. The verification results are summarized in Table IV

TABLE IV
PENDING REQUESTS

No. of Completed Requests Without Getting Service/hour	Probability	No. of Completed Requests Without Getting Service/hour	Probability
1	0.07042	6	5.250E-14
2	0.00302	7	0.00000
3	3.863E-5	8	0.00000
4	1.299E-7	9	0.00000
5	1.381E-10	10	0.00000

Table IV represents the probabilistic results of the requests that were marked as completed without getting serviced. For example, if the system runs for ***1 hour***, then the probabilities of making a single request and two requests as completed without getting processed are ***0.07042*** and ***0.00302***, respectively.

Next, the CTMC model is executed for ***10 hours*** and by keeping ***var=1*** we observed how the first Request, that was marked as completed, changed over time. Similarly, we can observe the next request probabilities by changing the values of the underlying variables ***var***. Our results are summarized in Fig. 4.

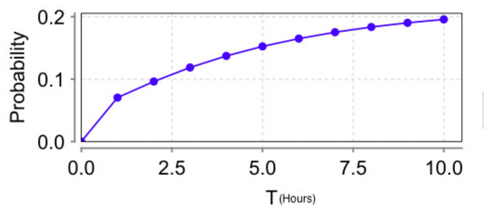

Fig. 4. Time Effect on Probability of First Pending Request

To find the probabilities of the number of requests that produced false results, we placed the variable ***var*** on the path moving from state *In Progress* to state *False Results From Cache Review*. The findings are summarized in Table V.

TABLE V
PROBABILITIES OF NUMBER OF FALSE RESULTS PRODUCED IN 1 HOUR

False Results Produced/hour	Probability	False Results Produced/hour	Probability
1	0.06873	6	0.00000
2	2.741E-4	7	0.00000
3	4.534E-8	8	0.00000
4	6.65E-13	9	0.00000
5	0.000000	10	0.00000

Table V summarizes the probabilistic results for the requests that produced false results. For example, the probabilities that a single request and two requests would lead to false results in a ***1 hour*** execution of the system are ***0.06873*** and ***2.741E-4***, respectively.

After that, we calculated the probabilities of the number of requests, that were partially accepted by placing the variable ***var*** between the corresponding required path. The results are summarized in Table. VI. By executing the CTMC model for ***10 hours***, the probabilities associated with the first request that was partially accepted are given in Fig. 5.

TABLE VI
PROBABILITIES OF NUMBER OF PARTIALLY ACCEPTED REQUESTS IN 1 HOUR

No. of Partially Accepted Requests/hour	Probabilities	No. of Partially Accepted Requests/hour	Probabilities
1	0.099660	6	0.00000
2	2.356E-4	7	0.00000
3	1.197E-8	8	0.00000
4	4.048E-14	9	0.00000
5	9.718E-22	10	0.00000

Table VI shows the probabilistic results for the requests that were partially accepted. For example, the probabilities for the partial acceptance of a single request and two requests in a ***1 hour*** execution of the system are found to be ***0.0996600*** and ***2.356E-4***, respectively.

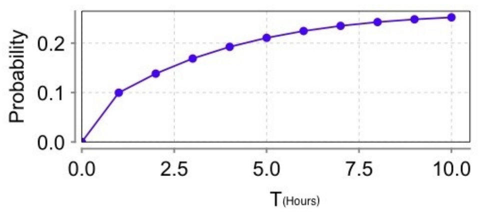

Fig. 5. Time Effect on Probability of First Request that was Partially Accepted

It is known that the systems based on the FHIR standard are susceptible to failures. The most critical failures include the identification of request completion without performing tests, false results generated by medical equipment and partial acceptance of request. These flaws are very important and need serious attention because these issues may lead to serious consequences, including life threatening situations. The main contributions of this paper include the formal probabilistic analysis of these flaws by using the Markov Chain theory and the evaluation of the numerical probabilistic values associated with these failures. Table II presents the absolute probabilistic values of the failure of request as well as the communication system. Similarly, the results obtained from the CTMC provide statistics about real-time failures of the requests and the system. These results can facilitate to make the FHIR standard more reliable as additional checks can be added to minimize the probabilities of these events.

To the best of our knowledge, the verification of properties, like the probabilities of completed requests without performing service, false results produced and partially accepted requests per hour and state specific probabilities, can only be done using probabilistic model checking due to its exhaustive state-exploration. Thus, despite providing useful information, these properties, to the best of our knowledge, have not been verified in the context of any health care standard, including HL7. Moreover, traditional techniques, like numerical methods and simulation, cannot match the precision of results obtained by the proposed approach for systems with such a large number of state-transitions, like the one analyzed in this section.

The complexity of the analysis can be judged by the fact that the verification of some of the properties, mentioned above, required exploring up to tens of thousands of state-transitions. However, PRISM handled these verification problems successfully and its user interface was found to be very easy to work with. Finally, the plotting capabilities of PRISM were also found to be very handy.

VI. Conclusion

This paper presented a formal reliability analysis for a typical FHIR Standard based e-health system in a hospital setting. The main contributions of the paper include the development of the Markovian models for the doctor and the Lab staff. These behaviors are then used to develop the MDP and CTMC models, using the PRISM language, and check various probabilistic properties of our system. The paper also identifies some key reliability assessment properties of the FHIR standard that can be formally verified by PRISM. The proposed approach has been found to be more accurate and scalable and it also allows us to verify many novel reliability aspects compared to other existing reliability analysis approaches for health care standards. We plan to use the proposed approach to analyze the reliability of other standards as well. We also plan to see the impact of having redundancy in the models on the overall reliability of the system.

References

[1] HL7. http://http://www.hl7.org/, 2014.

[2] FHIR. http://www.hl7.org/implement/standards/fhir/, 2014.

[3] J.R. Abrial. Faultless Systems: Yes We Can! *IEEE Computer Journal*, 42(9):30–36, 2009.

[4] C. Bertolini, Z. Liu, M. Schaf, and V.Stolz. Towards a Formal Integrated Model of Collaborative Healthcare Workflows. In *Foundations of Health Informatics Engineering and Systems*, volume 7151 of *LNCS*, pages 57–74. Springer, 2012.

[5] M. Hoogendoorn, M. C. Klein, Z. A. Memon, and J. Treur. Formal Verification of an Agent-Based Support System for Medicine Intake. 25:453–466, 2009.

[6] S. M. Babamir and M. Borhani. Formal Verification of Medical Monitoring Software Using Z Language: A Representative Sample. *Journal of Medical Systems*, 36(4):2633–2648, 2012.

[7] Z. Daw, R. Cleaveland, and M. Vetter. Formal Verification of Software-Based Medical Devices considering Medical Guide-lines. *International Journal of Computer Assisted Radiology and Surgery*, 9(1):145–53, 2014.

[8] K. Benghazi, M. V. Hurtado, M. L. Rodrguez, and M. Noguera. Applying Formal Verification Techniques to Ambient Assisted Living Systems. volume 5872 of *LNCS*, pages 381–390. Springer, 2009.

[9] A. Gawanmeh. An Axiomatic Model for Formal Specification Requirements of Ubiquitous Healthcare Systems. In *IEEE Consumer Communications and Networking Conference*, pages 898–902, 2013.

[10] O. Faust, U. R. Acharya, and T. Tamura. Formal Design Methods for Reliable Computer-Aided Diagnosis: A Review. *IEEE Revisions in Biomedical Engineering*, 5:15–28, 2012.

[11] R. Jetley, S. PurushothamanIyer, and P. L. Jones. A Formal Methods Approach to Medical Device Review. *IEEE Computer Journal*, 39(4):61–67, 2006.

[12] J. F. Groote, A. Osaiweran, and J. H. Wesselius. Analyzing the Effects of Formal Methods on the Development of Industrial Control Software. *IEEE International Conference on Software Maintenance*, pages 467–472, 2011.

[13] M. Vieira, X. Song, G. Matos, S. Storck, R. Tanikella, and Bill B. Hasling. Applying Model-Based Testing to Healthcare Products: Preliminary Experiences. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, pages 669–672. ACM, 2008.

[14] S.O. Oio, O.O. Olugbara, G. Ditsa, M.O. Adigun, and S.S. Xulu. Formal Model for e-Healthcare Readiness Assessment in Developing Country Context. In *International Conference on Innovations in Information Technology*, pages 41–45, 2007.

[15] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *International Conference on Computer Aided Verification*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[16] C.C. White III and D.J. White. Markov Decision processes. *European Journal of Operational Research*, 39(1):1–16, 1989.

[17] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-Checking Continuous-Time Markov Chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[18] C. Baier and J. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[19] H. Younes. Ymer: A Statistical Model Checker. In *International Conference on Computer Aided Verification*, volume 3576 of *LNCS*, pages 429–433. Springer, 2005.

[20] M. Khattri J. Katoen and I.S. Zapreev. A Markov reward Model Checker. In *International Conference on the Quantitative Evaluation of Systems*, pages 243–244, 2005.

[21] M. Viswanathan K. Sen and G. Agha. Vesta: A statistical Model-Checker and analyzer for probabilistic systems. In *International Conference on the Quantitative Evaluation of Systems*, pages 251–252, 2005.

[22] J. Meyer-Kayser H. Hermanns, J. Katoen and M. Siegle. Etmcc: Model Checking performability properties of Markov Chains. In *International Conference on Dependable Systems and Networks*, pages 673–673, 2003.

[23] A. Fong and T. Motoyama. Method and system of remote diagnostic, control and information collection using a shared resource, June 24 2008. US Patent 7,392,307.

[24] A. Berson. *Client/Server Architecture*. McGraw-Hill, Inc., 1996.

[25] J. T. Barry, K. J. Casey, M. Lachman, D.G. McGinnis, C. Niedmann, V. Santarsieri, M. Serra, S.M. Tenner, W.M. Tilton, et al. Methods for generating patient-specific medical reports, 1999. US Patent 5,991,729.

[26] FHIR Diagnostic Order Status State Machine. http://www.hl7.org/implement/standards/fhir/diagnostic-order-status.html, 2014.

[27] O.O. Aalen and S. Johansen. An Empirical Transition Matrix for Non-Homogeneous Markov Chains based on Censored Observations. *Scandinavian Journal of Statistics*, 5(3):141–150, 1978.