A Proactive Approach for Context-Aware Self-Adaptive Mobile Applications to Ensure Quality of Service

Md. Shafiuzzaman
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
Email: md.shafiuzzaman.hira@gmail.com

Nadia Nahar
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
Email: nadia.nahar.iit@gmail.com

Md. Rayhanur Rahman
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
Email: rayhan@du.ac.bd

Abstract—**Mobile Applications are rapidly emerging as a convenient medium for using a variety of services. With the high penetration of smart phones in society, self-adaptation has become an essential capability required by mobile application users. In an ideal scenario, an application is required to adjust its behavior according to the current context of its use. This raises the challenge in mobile computing towards the design and development of applications that sense and react to contextual changes to provide a value-added user experience. In its general sense, context information can relate to the environment, the user, or the device status. In this paper, a framework is proposed for building context aware and adaptive mobile applications. Based on the concepts of software requirement specification and probabilistic modeling, this framework guides the modeling of adaptability at design time and supports context awareness and adaptability at runtime to tackle potential issues that can hamper Quality of Service (QoS). In the core of the approach, a probabilistic model has been derived from the software requirement model using the quantitative terms of context dependability and then, it is continuously verified against the runtime operations of a mobile application. Finally, the proposed approach has been used in a mobile application development use case to observe how proactive adaptability can be built into mobile application development to ensure QoS.**

I. Introduction

High penetration of smart phones and tablets in society have created an exponential growth in mobile application development. It is characterized by a high degree of unpredictability and dynamism as mobile applications are installed and used anywhere and anytime by millions of users. Apart from diversified functional requirements from users, development of these applications addresses traditional non-functional requirements regarding security, performance, reliability and some other issues like integration with device hardware, memory and storage limitations. Moreover, it requires considerations of some special factors such as display size, power consumption, available networks, conductivities and their characteristics, availability of software and hardware components, location as well as social, physical and mental state of the users. Hence, factors that impacts on circumstances, situation and environment of both system and users are defined as context [1]. Context characterizes the functional requirements of the system as well as the non-functional requirements at run time [2]. Context of a system can be categorized differently depending on their operational characteristics, though there are three main categories [3] mentioned below:

- *Computing Environment*: CPU and memory usage, screen size, power consumption, available storage, available networks, and their characteristics and availability of software and hardware components
- *User Environment*: Physical and mental state of the users, social context and ongoing activities in device
- *Physical Environment*: Location, weather, light, temperature and noise conditions, sensors and proximity to nearby objects

A context-aware application uses contextual information to provide desired services to the users [2]. For example, a GPS based mobile application utilizes user location to provide map services. Changes to the operational context of a system might affect its runtime behavior, which degrades overall Quality of Service (QoS). However, contextual changes are beyond the control of application as those occur spontaneously and unpredictably [4]. As mobile applications use contextual information more frequently than any other applications [5], there is a growing demand for mechanism that can recognize and withstand contextual changes by dynamically adapting runtime conditions and behavior of the system to satisfy QoS. Thus building self-adaptive capabilities in mobile applications facilitate the application itself being capable of modifying its execution policy at runtime to achieve QoS objectives [6].

Developing self-adaptive software systems has shown to be significantly more challenging than traditional systems [12]. Over the past decade, researchers have made significant progress with methodologies and frameworks [7], [8] that target the development of such systems, though there is hardly any contribution in the domain of self-adaptive mobile application development. Moreover, most of the approaches focus on developing techniques [6], [8] that reconfigure system architecture when adaptability needed. All of these can be categorized as reactive approaches as they apply adaptation

Table I
FEATURE DESCRIPTIONS OF E-HEALTH CARE

Feature	Description	Contexts
F1	Track emergency using GPS	Network, Battery and Location
F2	Display emergencies using locally stored map	Network and Battery
F3	Show addresses of emergencies using text	Network and Battery
F4	Communicate with doctors through video call	Network and Battery
F5	Communicate with doctors through voice call	Network and Battery
F6	Communicate with doctors through real-time chat	Network and Battery
F7	Offline communication	Battery

techniques after it violates the contracted QoS. As we discussed above, QoS of the context-aware mobile application mostly depends on its operational contexts. The software engineers can only predict the quantitative values of these operational contexts while designing the system using their past experiences on observed behavior of similar applications. But these predictions may differ or may not exist at runtime for contextual changes. A mobile application becomes unusable if the contextual changes cannot be determined before the application reaches at failure state. So a proactive approach is needed to recognise the probable failure. In order to address the aforementioned challenge, this paper proposes a proactive adaptation policy that can determine the need for adaptation before entering into unusable state. The approach spans from development time of an application to its run time evolution. While designing the application, probabilistic models are used to quantify the contextual dependability. Traditional software engineering principles insist to follow Model-driven development [9] that stands on requirements models. Requirements models are built to better understand and reason about QoS and also used to support systematic interaction with stakeholders in requirements elicitation. The proposed approach continuously validate the running system comparing with these requirements models to determine whether the system needs to go under adaptation treatment or not to satisfy user requirements and contracted QoS.

As the proposed approach stands on a different motivation than the existing works of this domain, there are hardly any resources to compare with the proposed approach. So, a mission critical mobile application is modeled using the approach to prove the validity of the approach.

The remainder of this paper is organized as follows: Section 2 describes an overview of the related work. Section 3 presents a motivating mobile application used to describe and evaluate the research. Section 4 proposes the overall approach. In section 5, the motivating example is modeled to validate the proposed approach. Finally, Section 6 conclude the paper with future work directions.

II. RELATED WORK

Over the past decade researchers have made significant progress on engineering the adaptation logic. In this section, state of the art research approaches regarding adaptive systems and approaches will be discussed.

The first noteworthy approach [6] in engineering self-adaptive software systems is established on architectural reconfiguration of the system. This approach depends on structural changes, such as adding, removing, and replacing software components. Some other similar approaches provide adaptation by changing system's architectural style [10] and rebinding component's interfaces [11]. All of these approaches depend on structural changes more specifically changes on architecture level which may not be feasible for light-weight applications such as mobile application. Moreover, if architectural changes affect operational context, the application may turn into unusable state. Recently, a three-layer reference model [7] is proposed to support reliability issue of mission-critical systems. Goal management, change management and component control layers of this model ensure self-adaptation on contextual changes for successful service completion. A recent approach proposes a learning-based framework [8] that ensures run-time management of feature selection and deselection to adapt context changes. An interesting contribution was made by Ghezzi et. al [12] where they provide a formal approach that alerts on any divergence on required reliability and performance. An extension of [12] proposed to select implementation variants at run time to support non-functional adaptation by model-checking [13]. The proposed method of [7], [12], [13] target non-functional requirements but context-aware mobile applications must have proper strategic plan to adapt the features (functional requirements) that use affected contexts. Except [12], all of the approaches can be categorized as reactive approaches as they employ adaptation mechanism after detecting any unusual behavior by the system. Though [12] can sense the need of adaptation proactively, their approach only considers non functional requirements.

III. MOTIVATING EXAMPLE

Software applications that are designed for emergency response require a high degree of adaptability. These applications must be developed with a minimum amount of dynamism so that they can adapt run time changes of their environment and other contexts. A medical emergency can occur at any time, in any place, and as no emergency is ever convenient, people are in a fix what to do. A mobile application that locates nearby hospitals or helps to contact with specialised doctors may come with great usefulness during these critical situations. The context of this application is continuously changing because of users mobility. In addition, the computing environment where the application is deployed, also undergo some changes like battery drainage and network disconnection. To improve user experience, the application should be adaptive enough to recognise and tolerate the changes.

The above mentioned application can be a proper example to motivate the concepts of this paper for its adaptive requirements. The application will be used as running example in the remaining sections of this paper to describe and evaluate the proposed approach. It will be called as e-heath care throughout this paper. Features of e-heath care are presented in Table I with their context requirements.

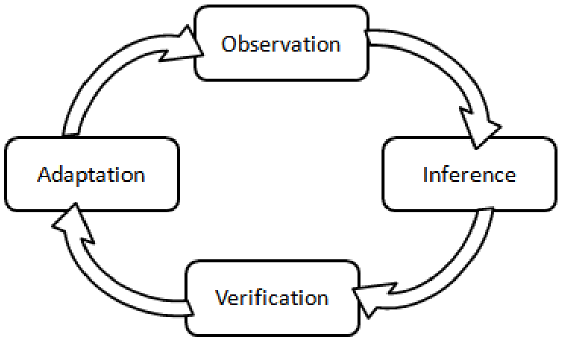

Figure 1. Adaptation Cycle

IV. Approach Overview

Usually a mobile application provides a set of features that manipulate data to fulfill user needs. Such two features might rely on each other or those might be mutually exclusive. When contextual changes occur, adaptation techniques redirect the application to deselect a feature and select an alternative one to meet the overall QoS. To support continuous assurance of overall QoS, we propose an automatic adaptation cycle illustrated in Figure 1. This is a continuous loop comprising with following four steps:

- *Observation*: Observation collects emerging patterns of application behavior through context data collection.
- *Inference*: From the collected data regarding to operational context, it must be inferred which parameters have affected by context change (if any).
- *Verification*: To support requirements satisfaction at run time the running system needs to be verified against the requirements models.
- *Adaptation*: If any failure is predicted in verification step, a suitable adaptation mechanism needs to apply.

As discussed above, a plenty of works have been done on adaptation mechanism. So developing an adaptation algorithm is out of the scope of this paper. Moreover, the impact of any adaptation algorithm can be enhanced in a large scale by implementing proactive detection when an application needs to go under adaptation treatment. This research effort solely focuses on the detection mechanism.

A. Observation

The main challenge of the observation step is to determine the quantitative values of the contexts of an application. These values can be captured by domain knowledge of the software engineers or their past experiences on observed behavior of similar applications. Actually, for these context critical applications, domain knowledge fills the gap between user requirements and specifications [14]. The relation can be formalized using Equation 1. Here, the requirements and the specifications of an application are formally notated as R and S, respectively, and D notates the formal statements that specify the domain assumptions.

$$S, D \models R \tag{1}$$

That is, S ensures satisfaction of user requirements R in the context of the domain properties D. Using the concept of [12] domain properties D are partitioned into three relevant disjoint subsets. D_S captures the set of stable assumptions, which will not change later. For example, D_S may include the known physical environment such as availability of camera. The information regarding D_S should be excluded as these do not change their context value at runtime. D_U and D_E instead capture the domain assumptions that are likely to change over time. D_U denotes the assumptions on usage profiles. These consist of the properties that characterize how the application being designed is expected to be used by its clients, for example, online or offline behavior of a mobile application. D_U properties must be taken into account during design because those may drive architectural decisions that lead to satisfaction of user requirements. However, the assumptions made at design time may turn out to diverge from what the running system eventually exhibits. For example, consider the motivating example that finds the nearest hospital depending on user location. The application requires Internet connection to enable the GPS service. So, an architectural decision is needed for deciding what will happen if user is offline. A static map can be stored as an mutual exclusive feature with GPS feature. D_E denotes the set of assumptions on related actions invoked by the application to perform an action. These related actions are not directly handled by the intended service, so these can be considered as external actions. To understand these domain properties we may go back to the motivating example which offers a feature of real time assistance from the hospital. This feature rely on external service such as vocal or textual response.

B. Inference

In any software application, not all features have equal standings. If the context changes, it might be a better choice to disrupt a non-critical feature if it helps to continue satisfying critical features. Traditional static mobile applications offer all the set of features simultaneously, but we propose to disable some features and enable others when an operational context has changed. As a result of this adaptation, the users will not necessarily deal with a similar pool of features each time they launches the application. For instance, if the battery level

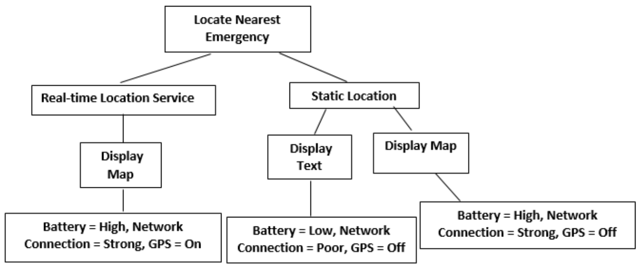
Locate Nearest Emergency
Real-time Location Service
Static Location
Display Map
Display Text
Display Map
Battery = High, Network Connection = Strong, GPS = On
Battery = Low, Network Connection = Poor, GPS = Off
Battery = High, Network Connection = Strong, GPS = Off

Figure 2. Feature Modeling of Locate Nearest Emergency

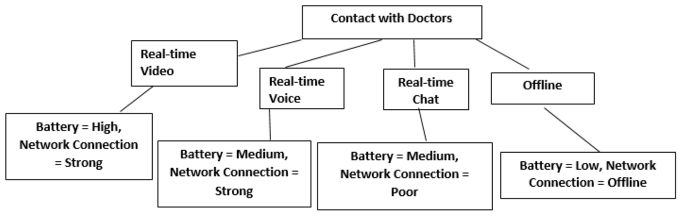
Contact with Doctors
Real-time Video
Real-time Voice
Real-time Chat
Offline
Battery = High, Network Connection = Strong
Battery = Medium, Network Connection = Strong
Battery = Medium, Network Connection = Poor
Battery = Low, Network Connection = Offline

Figure 3. Feature Modeling of Contact with Doctors

of the device is low, non-critical features that are battery intensive may be disabled. Defining rules for disabling features according to contextual changes for a given mobile application is a challenging task. In fact, designers have constraints on what to make unavailable. In this paper, several judicious criteria are inferred for the choice of features to disable.

The traditional requirements engineering approach insists to group features into three priority categories normal, expected and exciting. This approach is commonly used in project development to make trade-offs during project planing according to the available resources. This prioritization may help the designer to perform trade-offs when it comes to infer dynamic adaptation. In case of resource insufficiency for contextual changes, features that have high priority are kept available while those with lower priority are progressively made unavailable according to altered context. Resources consumption estimation is the second criterion for feature selection. When two features have the same priority level, the decision would be based on the estimation of the minimal resources needed to deploy a feature. When the battery of a given device is low, feature implementations that consume less energy would be those deployed ones.

C. Verification

Verification step consists of modeling the application and their runtime management. Here activity diagram is used as reference requirement model as it provides operational description of S. An analytical model as Markov chain is generated by modeling the states of the application with decision parameters that updated through contextual changes. Usually, Markov property of Markov chain models help to determine the distribution of the parameters in current state depending only on the distribution of the previous state. Several approaches are available to support automatic model-to-model transformations, including those from activity diagrams to Markov models. Here, ATOP tool [15] is used to redefine activity diagrams with context values as Markov models. Context annotations on the analytical model are used to express the domain assumptions corresponding to D_U and D_E. To perform run time verification of this analytical models, PRISM [16], a stochastic model checker is used that can check properties of R from running system. Other stochastic model-checkers may also fit here. PRISM takes the analytical models as input and outputs an alert on any divergence from the expected QoS. Adaptation strategy may require Graphical User Interface (GUI) layout change or architectural change at run time. As an example, geo-specific features could be disabled once the location of a user changes. Users need to be aware of such fact and agreement for QoS should be updated accordingly.

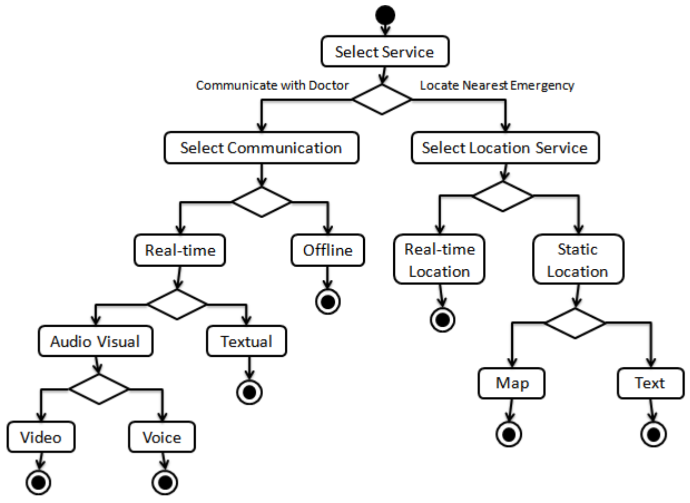

Figure 4. Activity Diagram of e-heath Care

V. Proof of Concept

In this section the motivating example is modeled to show the applicability of the proposed approach. Starting from identification of requirements and uncertain or changeable domain properties, this section covers high-level models of the application developed at design time. The latter part of this section focus on run time verification of those models.

The features described in Table I are intended to provide two specific services: (I) Locate nearest emergency (II) Contact with doctors. Figure 2 and 3 provides an abstract view of those services and their context requirements. From Figure 2, it can be visualized that, real-time map, static map, and static location text, these three mutual exclusive features are responsible for 'Locate nearest emergency' service. And video conferencing, voice communication, textual chatting, offline communication, these four mutual exclusive features provide

Table II
DECISION PARAMETERS

Feature	Hierarchy	Success Requirements
Video Conference (F4)	Communication —>Real-time—>Audio-visual—>Video	Network = Strong, Battery = High and GPS = No
Voice Communication (F5)	Communication—>Real-time—>Audio-visual—>Voice	Network = Strong, Battery = Medium and GPS = No
Textual Chat (F6)	Communication—>Real-time—>Textual	Network = Poor, Battery = Medium and GPS = No
Offline (F7)	Communication—>Offline	Network = Offline, Battery = Low and GPS = No
Track Emergency with GPS (F1)	Location—>Real-time	Network = Strong, Battery = High and GPS = Yes
Stored Map (F2)	Location—>Static—>Map	Network = Strong, Battery = High and GPS = No
Text (F3)	Location—>Static—>Text	Network = Poor, Battery = Low and GPS = No

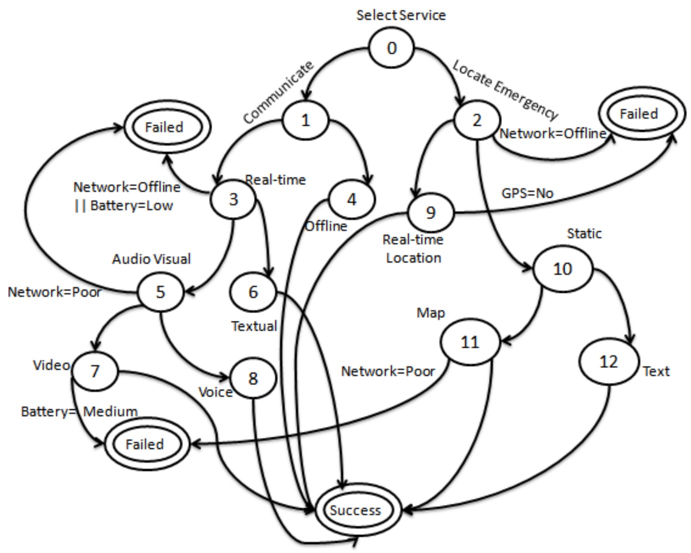

Figure 5. Feature Modeling of Locate Nearest Emergency

the 'Contact with doctors' service (Figure 3). The contextual requirements of these features are also specified.

Using the above feature modeling, an overall activity diagram of the application is generated. Figure 4 illustrates the activity diagram. From feature modeling it is identified that e-heath care has context dependency on network connection, battery and location service (GPS). ATOP tool is used to redefine the activity diagram as Markov model. Here the context dependencies are used as decision parameters of Markov Model. The model contains one state for every operation performed by the system to provide the intended service plus a set of auxiliary states representing potential failures associated with the contextual changes. The failure states are totally absorbing cannot be left once accessed. The analytical model generated by ATOP tool is illustrated in Figure 5. The analytical model can be specified more likely a rule-based model. Table II provides an overview of the rules that used to model this application. PRISM (model-checker) continuously runs the analytical model using these rules against the running system and checks whether there is any possibility of entering into failure state. As this validation gives proactive alert, it provides a scope for adapting the application before entering unusable state. From Table II, it is easily identified when there will be an alert for adaptation. For example, if the device battery is low, there will be only one locate service option - address text; and only one communication option - offline communication. Now, if the Markov model is analyzed, the possible future states can be inferred from the current state based on the device context values. The initial state is State-0, where the decision of the service selection is taken. State-1 is for communicating, and State-2 is for locating. Now, if the network context value is offline, the state cannot propagate to State-3 (Real-time) as it will go to the Failure State for condition 'Network=Offline or Battery=Low'. It must propagate to State-4 (Offline) for getting to the Success State. Similarly, propagation to the other states also depends on these context values which are mentioned in Table II.

The generated rule-based Markov model assures that propagation to the Failure State can be inferred from the device context condition. Thus, these propagation can be stopped by controlling the features' appearance to the user. This eliminates the unexpected shut down of the application, as those failure paths are made unavailable to the application user.

VI. CONCLUSION

This research identifies the necessity of a proactive approach to recognize probable failure of mobile application. Thus, it proposes a proactive adaptation policy to identify the failure states before entering to those. Markov model is used for this to quantify the contextual dependability, and requirements models are used to derive the Markov model. Then the running system is continuously validated against the Markov model for identifying the need of adaptations proactively. To illustrate the proposed model an example application is analysed by generating the requirements model and Markov model. These models are then inspected for getting the failure states that needs to be avoided at runtime. To avoid these failure states, a rule-based approach is followed. The rules are generated by analysing the operational contexts of the application. Analysing the running example, it can be concluded that the proposed approach can be generalised for any context-aware mobile applications. The proposed framework opens a new window to look back to the overall adaptation approach differently. As a result several several important research questions are opened for further research. An alternative approach of the proposed method can investigate the use of interval estimators instead of single-value of the decision parameters. This might lead to the adoption of more expressive models, like Interval Markov Chains. The proposed approach is centered on detection of the need for self-adaptation. The fourth step of the adaptation cycle is the selection of a self-adaptation strategy. Though in this area research is quite active, very few of them have concentrated on light weight applications like mobile application. Another debate-able area is still unsolved what approach an adaptation algorithm should follow between system reconfiguration and parametrised adaptation. Though the proposed work uses parametrised adaptation successfully, further extension of this work can be proper trade-off between these two approaches.

References

[1] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," *Software Engineering Institute*, 1990.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, 1999.

[3] B. Schilit, N. Adams, R. Want, "Context-Aware Computing Applications," *Proceedings of the International Workshop Mobile Computing Systems and Applications*, 1994.

[4] A. Chan et al., "MobiPADS: Reflective Middleware for Context- Aware Mobile Computing," *IEEE Transaction on Software Engineering*, vol. 29, no. 12, pp. 1072-7085, Dec. 2003.

[5] Pessemier, Dooms, Martens,"Context-aware recommendations through context and activity recognition in a mobile environment," Multimedia Tools and Applications, 1–24, 2013.

[6] J. Kramer and J. Magee, "Self-Managed Systems: an Architectural Challenge," *International Conference on Software Engineering*, Minneapolis, Minnesota, 2007.

[7] D. Cooray, E. Kouroshfar, S. Malek, R. Roshandel, "Proactive Self-Adaptation for Improving the Reliability of Mission-Critical, Embedded, and Mobile Software," *IEEE Transactions on Software Engineering*, 2013.

[8] N. Esfahani, A. Elkhodary, S Malek, "A Learning-Based Framework for Engineering Feature-Oriented Self-Adaptive Software Systems," *IEEE Transactions on Software Engineering*, 2013.

[9] J. Bezivin., "Model driven engineering: An emerging technical space in Generative and Transformational Techniques," *Software Engineering (GTTSE)*, volume 4143 of LNCS, pages 36-64. Springer, 2006.

[10] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, Software Architecture: Foundations, Theory, and Practice. Wiley, 2009.

[11] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *Softw. Eng. Notes*, vol. 17, no. 4, pp. 40–52, Oct. 1992.

[12] A. Filieri, C. Ghezzi, G. Tamburrelli, "A Formal Approach to Adaptive Software: Continuous Assurance of Non-Functional Requirements," *Formal Aspects of Computing*, 2012

[13] C.Ghezzi,L.S.Pinto,P.Spoletini,G.Tamburrelli, "Managing non-functional uncertainty via model-driven adaptivity," *International Conference on SoftwareEngineering*,2013.

[14] P. Zave and M. Jackson.,"Four dark corners of requirements engineering," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 1997.

[15] S. Gallotti, C. Ghezzi, R. Mirandola, and G. Tamburrelli, "Quality prediction of service compositions through probabilistic model checking" *Proc. of the 4th International Conference on the Quality of Software Architectures*, Karlsruhe, Germany, 2008.

[16] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "Prism: A tool for automatic verification of probabilistic systems," *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2006.