International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 7 (2015) pp. 16563-16574

https://dx.doi.org/10.37622/IJAER/10.7.2015.16563-16574

# An Approach For Prioritising Soft and Hard Goals In Self Adaptive Software Systems

**Dr.Ananthi Sheshasaayee**
*Associate Professor and Head, Dept of Computer Science,*
*Quaid-E-Millath College for Women, Chennai – 600 002, INDIA*

**Brinda Ramanujam**
*Research Scholar, Bharathiar University, Coimbatore*
*Assistant Professor, Department of Computer Science,*
*M.O.P Vaishnav College for Women,*
*Chennai – 600 034, INDIA*

## Abstract

**Background:** Self-adaptive systems are software-intensive systems having the capability to respond to a variety of changes that occur in their operating environment. These systems modify their behavior by adapting their structure at run-time in an autonomous way [1].
**Method**: This paper focuses on requirements gathering and prioritization of requirements in self adaptive software systems. There are several requirements gathering approaches that exist for such systems, our approach considers an integration of several techniques to gather requirements of self adaptive systems and also prioritise the soft and hard goals of the system.
**Findings :** The extent to which the several requirements influence the system and change the behavior of the system is crucial and leaves an uncertain state of the system. Prioritisation of hard goals is obtained by calculating the normalized priority vector.

**Keywords:** Self adaptive software systems, adaptability, requirements elicitation, soft goal, hard goals, prioritization.

## Introduction

Computing has become a very integral part of our daily lives. In recent years, several steps have been taken to build self adaptive software systems (SAS). Self adaptive systems are those systems that take increasing responsibility for dynamically detecting problems and repairing themselves. The complexity of current software systems has led the researchers in the software engineering community to investigate

number of ways of developing and managing self adaptive systems. The features that make the Software systems self adaptable are dependable, configurable, versatile, energy efficient, recoverable, customizable, resilient, and self optimizing. Software systems, when adapting to the changes that take place in their operational environment, become self adaptive in nature[2]. The behavior of these systems is often undeterminable and uncertain due to the ever changing requirements [3][4][5].

Self adaptive systems are able to modify their behavior or structure in response to their perception of the environment [6]. Requirements for building such systems is an important research topic. Researchers have taken numerous efforts to explore self adaptation. The most common initiative that triggers the design of self adaptive systems is "software".

This paper may serve as a blueprint for analyzing the requirements of SAS. Today there are several techniques for gathering requirements for the Systematic construction of self adaptive software systems. The various approaches handled to study the requirements for modeling self adaptive systems are discussed briefly.

## Related Work

In the following section we present the related work classified according to the approach used and convert the same to a model based learning SAS- REQ.

[2010] Nelly Bencomo, J.Whittle [7] compares requirements reflection with computational reflection and proposes that requirement reflection enables the creation of self managed software systems which will continuously change due to the changing environmental conditions.

[2011] Victor E Silva, Souza [8] proposes a Goal Oriented Requirements Engineering (GORE) model. This approach classifies goals as hard goals, softgoals, quality constraints (QCs) and domain assumptions (DAs). This model captures the goals of different stakeholders of the system.

[2012] Dhaminda. B. Abeywickrama, Nicola Bicocchi, Franco Zambonelli [9] proposes an adaptive SOTA model checker which is a general purpose and effective framework. This model checker serves as the key point and a building block for self adaptation.

[2013] Nelly Bencomo [10] explains that the key challenge posed by Self-Adaptive Systems (SAS) is the need to handle changes to the requirements and corresponding behavior of a SAS in response to varying environmental condition during runtime. This paper discusses the role of uncertainty in such systems, research challenges and present results. Different modeling techniques for the development of self-adaptive systems with specific emphasis on goal-based techniques are also studied.

[2014] Manzoor Ahmad [11] Model driven engineering techniques enable in the identification of requirements of self adaptive software system. Requirements are classified into FR and NFR. Processes are then identified to clearly define which of the FR or NFR can be adaptable in order that the system behaves in the desired manner.

[2014] Mohammad Dabbagh and Sai Peck Lee [12] emphasise why it is essential to prioritise software requirements. Prioritising the requirements results in the ordering of requirements which have to necessarily be considered first during the development.

## Research Contribution

In our requirements gathering approach we propose to create a SAS-REQ, which a combines all the methods proposed by several researchers to gather the requirements for self adaptive systems. Our model based approach aims to reduce uncertainty that may occur at runtime which is a result of lack of proper requirements gathering.

Requirements analysis, using our model based approach is integrated by

1. Identifying the requirements and classifying them as runtime entities.
2. Identifying the goal of the system to be deployed and developed and deriving the Quality Constraints **(QC)** and Domain attributes **(DA)**.
3. Creating a model checker to validate if the requirements gathered are correct.
4. Identifying goal based techniques for refining the requirements for self adaptive systems and classifying them into hard and soft goals.

Finally we prioritise the requirements using **REQ-PA** (Requirements priority and assimilation and approach)

## Proposed Solution

The traditional software engineering life cycle includes the five phases of software development namely Requirements gathering, design, implementation, testing and maintenance. When the same phases are revisited for a self adaptive software system, the same phases are slightly modified and the phases are well established within a boundary and adhere to the collect, adapt and relate stages.

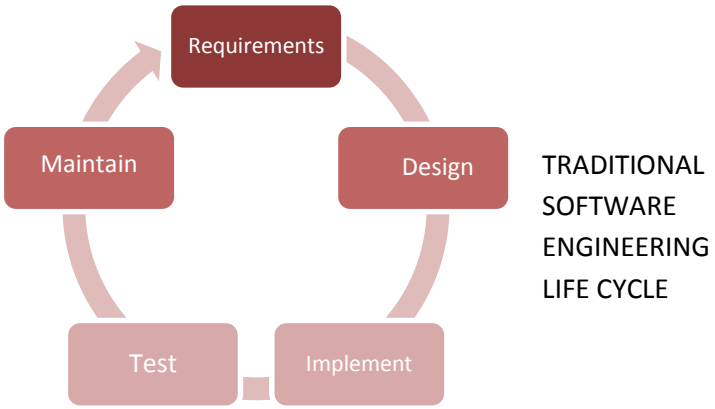


**Figure 1:** Traditional Software Engineering Life Cycle

The traditional software engineering life cycle (Fig :1), follows the waterfall model and consists of the requirements, design, implementation, testing and maintenance

phases. The life cycle of an SAS system needs to be revisited, since all the phases happen simultaneously, hand in hand hence making the entire system adaptable to the changes that occur. However a boundary is included within which the entire system operates. Responsibilities of the various phases of the system get transferred from the software engineer's side to the system side [13]. This transformation causes the traditional boundary between development time and run time to vanish. Hence the traditional software engineering model needs to be reconceptualised. The SPEM model (Software and Systems Process Engineering and Meta Model) [14] is mainly used by most of the developers to conceptualize their product.

The Software life cycle of self adaptive systems can hence be modified as shown (Fig: 2) below

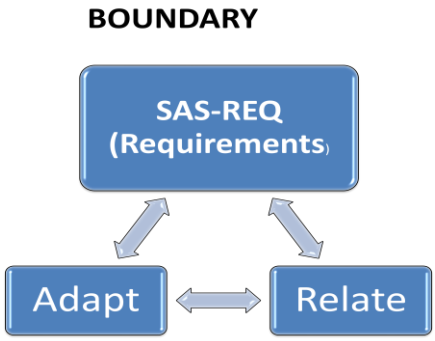


**Figure 2:** SAS-REQ Model for SAS

The self adaptive system's requirements phase **SAS-REQ** phase(Fig : 3), the first and the important phase of the life cycle of software which we hence forth call as the **SAS-REQ** can be described to be as follows

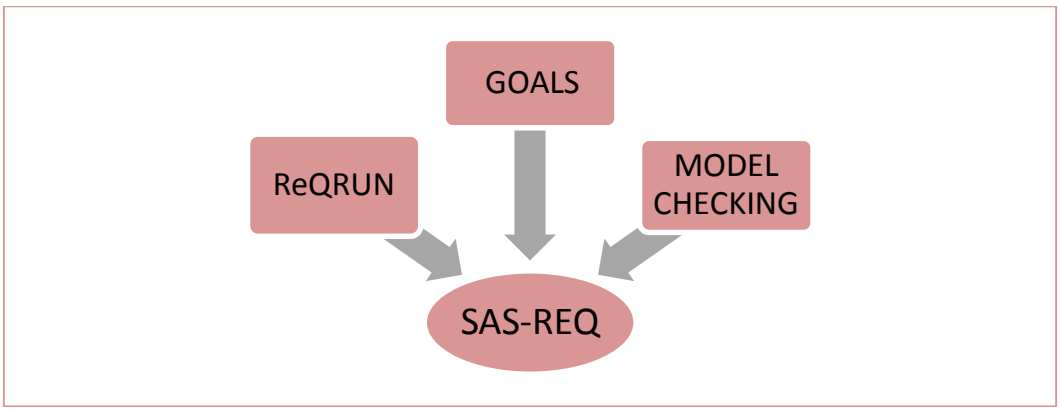


**Figure 3:** SAS-REQ Phase

The SAS-REQ phase combines all the techniques for requirements gathering. The next phase is the prioritisation of the gathered requirements. Our method proposes to

assimilate and prioritise these requirements using **REQ-PA**(Requirements Priority and Assimilation Approach)

**Requirements as Runtime entities (ReQ_RUN)**

Extracting the information from runtime system models to optimize the generation of target system models is an important issue in self adaptive systems [15]. Software engineering methods do not support dynamic appraisal of requirements as required by a Self adaptive system [16]. In most of the software that is created to be self adaptive, the definition of the structure of the requirements gets diluted as the requirements get transformed into implementation. Domain independent evaluation functions analyse the structure of the, to be generated target system model and compare them against a model of the executing system. Genetic algorithms are used to tradeoffs between functional and non functional requirements [17]. Gathering of requirements at runtime can be performed using the following algorithm (Algorithm:1)

---

**Algorithm 1: ReQ_RUN (**finding the requirements at runtime)

---

***Require**: ReQ_Run*
***Ensure:** ReQ_Run≠0;*
*1: def find_max ReQ_run (Req):*
*2: ReQ_Run= 0;*
*3: **for** x in Req:*
*4: **if** x>ReQ_Run*
*5: max ReQ_Run =x*
*6: **return** max ReQ_Run*

## Goal Based Modeling and Goal Realisation Strategies

Goal based models (Fig.4) help in deciding whether the requirements are partially or totally fulfilled.

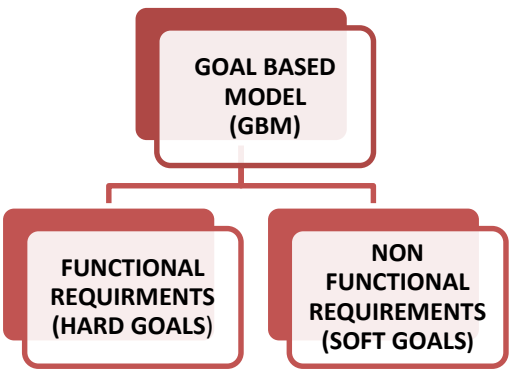


**Figure 4:** Goal Models

Goal realization strategies are the techniques used to build a GBM. There are various strategies to build these goals. The best strategy to be used is based on a utility function. This utility function is calculated taking into account the sum of the different effects of soft goals. In a SAS the goal realization strategy is based on DDN [18]

Soft goals are difficult to measure due to the vague and fuzzy nature. They belong to the non functional requirements. These provide an approach to make rational decisions in the phase of uncertainty due to change in environments [12,19]. Goal based modeling enables the developer to realize ***"why the user wants this?"*** and ***"why the user wants it in this way?"***

## Hard and Soft Goals

Requirements of any SAS are characterized by hard goals and soft goals. Hard goals refer to *"what the system should do?" w*hereas soft goals refer to *"How well the system should do it?"*. Soft goals reflect the optimization of the system's response time, performance, efficiency, security and satisfaction of the given specifications. Hard and soft goals have a serious effect on each other which are very essential for the development of software. Prioritising hard and soft goals for self adaptive systems is a separate research area [20]. Prioritization of functional and non functional requirements has been done using the different techniques such as Analytical hierarchy process (AHP) [21], Value Oriented Prioritisation (VOP) and Integrated Prioritisation Approach (IPA) [22].

Prioritization of functional and non function requirements has been done using Integrated Prioritisation Approach(IPA).

We propose to identify and prioritise **sas-HG** (self adaptive systems' **HARD GOALS**) and **sas-SG**(self adaptive systems' **SOFT GOALS**) using the **REQ-PA** (Requirements Priority and Assimilation ) approach

The following are the steps to be performed for the **REQ-PA(**Requirements Priority and Assimilation Approach)( Fig.5)

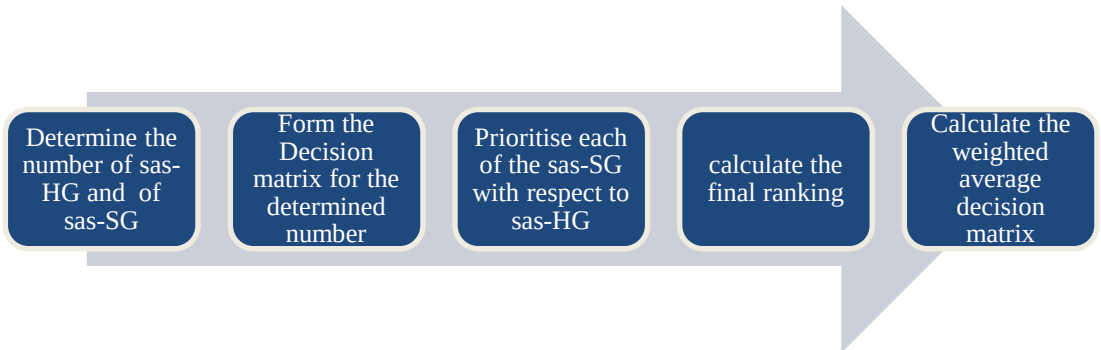


**Figure 5:** REQ-PA Phase

The first step is to identify and prioritise the sas-HG and sas-SG to be included in building the SAS system. We assume *"m"* to be the number of sas-HG and "*n*" the number of sas-SG. Let us presume there are three sas-HG( hard goals) i.e., HG1, HG2,HG3 and four sas-SG (soft goals) i.e., SG1,SG2,SG3, SG4.

The second step is to construct the decision matrix DM for the identified sas-HG and sas-SG. The HG and SG identified in the step 1 are introduced into the rows and columns of the decision matrix DM.

$$DM = \begin{array}{c|cccc} & SG1 & SG2 & SG3 & SG4 \\ HG1 & - & - & - & - \\ HG2 & - & - & - & - \\ HG3 & - & - & - & - \end{array}$$

---------- (1)

The next step is to assign priorities of each HG with respect to the SG. Priorities can be assigned based on the table given below. We use two scales the qualitative scale and convert the same to the quantitative scale to prioritise or rank each HG with respect to the SG.

**Table 1:** Prioritisation Actual scale and nominal scale

| **Qualitative scale** | **Quantitative Scale** |
|---|---|
| Very important (VI) | 1 |
| Important(I) | 0.5 |
| Less important (LI) | 0.25 |
| Not important (NI) | 0 |

The importance of each HG with respect to the SG is determined. Given a pair of HG and SG (selected from the matrix DM) it is the choice of the decision maker to determine the importance of each SG associated with each HG

The next step is to calculate SG final ranking with respect to all HG. Triangular fuzzy numbers and alpha cut approach is used for ranking. When all pairs have been evaluated, REQ-PA carries out the priority assessment of SGs with respect to all HGs, which represent the weights for Step 5, using triangular fuzzy numbers and alpha cut approach. During this step, REQ-PA creates a prioritized list of sas-SGs by calculating the total importance degree of each SG with respect to all HGs. The rationale behind this idea is that an SG which achieves the highest total importance degree among all HGs could be assigned as being a high-priority SG.

The following subtasks illustrate a stepwise process of computing the priority vector of SGs

***Subtask 1*** (conversion of the elements of matrix *D*M into numerical values).
First, REQ-PA converts all values of the decision matrix *D*M, which were specified according to the nominal scale, into corresponding actual scales, resulting in the matrix DM'

***Subtask 2*** (setting up the triangular fuzzy number).
In order to aggregate the different importance degrees of each SG for different HGS, the Triangular fuzzy number (TFN) is calculated. TFN is capable of aggregating the subjective opinions of a decision maker through fuzzy set theory. In this study, we use TFN since it is the most popular fuzzy number among the various shapes of fuzzy numbers.

The triangular fuzzy number $T_{yi}$ is represented using the following equations:

$$T_{yi} = (L_y, M_{yi}, H_{yi}), i = 1.....N, L_{yi}, M_{yi}, H_{yi} \in [0.001, 1] \quad \text{--------} \quad (2)$$

$$M_{yi} = n\sqrt{DM_{yia} \cdot DM_{yib} \cdot DM_{yic} \cdot \cdot \cdot DM_{yin}}, \quad \text{--------} \quad (3)$$

where $T_{yi}$ indicates the triangular fuzzy number of soft goal "y$i$"; $L_{yi}$ and$H_{yi}$ represent the lowest and highest value of the softgoal "y$i$ ," respectively;

$M_{yi}$ is generated by calculating the geometric mean of all values belonging to the softgoal "y$i$"; "n" is the total number of SGs and "m" is the total number of HGs. $D_{yia}$ specifies an opinion of a decision maker toward the importance degree of the SG "$y_i$" for achieving the Hard goal "a".

***Subtask 3*** (constructing the fuzzy priority vector, Table:2 ) After calculating the TFN value for each sas-SG, the fuzzy priority vector, namely, $F_y$, is generated,

**Table: 2 Fuzzy priority vector**

| $y_1$ | $y_2$ | $y_3$ | $y_4$ | …. | … | …. | $y_m$ |
|---|---|---|---|---|---|---|---|
| $Ty_1$ | $Ty_2$ | $Ty_3$ | $Ty_4$ | | | | $Ty_m$ |

Notice that the values of $F_y$ are derived from (2)

***Subtask 4 (*** Defuzzification process). REQ-PA uses the alpha cut approach, proposed by Lious ndWang [ 23 ], as shown in (4), to perform the defuzzification process. The defuzzification is accomplished in order to convert the calculated TFN values into quantifiable values, leading to the priority vector $W$

$$\mu_{\alpha,\beta}(\tilde{F}_{yi}) = [\beta \times f_\alpha(L_{yi}) + (1-\beta) \times f_\alpha(H_{yi})], 0 \le \alpha, \beta \le 1, \quad \text{----------} \quad (4)$$

where $f_\alpha(L_{yi}) = (M_{yi} - L_{yi}) \times \alpha + L_{yi}$, which represents the left-end boundary value $f_\alpha(H_{yi}) = H_{yi} - (H_{yi} - M_{yi}) \times \alpha$, which indicates right-end boundary value.

In this context, $\alpha$ and $\beta$ hold the preferences and tolerance of risk as assumed by the decision maker. The values range between 0 and 1, in such a way that a lesser value indicates greater uncertainty in decision making. Preferences and risk tolerance are not the focus of this paper. A value of 0.5 is used for both $\alpha$ and $\beta$ to represent a

balanced environment. This indicates the decision maker to be neither optimistic nor pessimistic about the judgment. Normalizing the calculated priority vector, NW of normalized weights is obtained using the following equation:

$$NW_k = \frac{W_k}{\sum_{k=1}^{N} W_k} \quad \text{----------(5)}$$

Performing the steps stated above, a decision maker is provided with a prioritized list of sas-SGs along with their corresponding importance values with respect to all existing sas-HGs

The final step is the computation of final ranking using weighted average decision matrix and weights from Step 4.

During the previous steps, we obtained the priority value of each sas-SG with respect to all sas-HGs. Furthermore, the importance degree of each sas-SG regarding every individual sas-HG was elicited (i.e., elements of the matrix). By gathering such data, the weighted average decision matrix, as indicated in Table 3, is generated in order to assist the process of calculating the priority vector of sas-HG (according to their relations with sas-SGs). The aggregation method determines the prioritization of sas-HG by calculating the geometric means but by using the obtained normalized priority vector of sas-HGs for its weights, leading to the priority vector V as represented using the following equation

$$V = \prod_{1}^{n} DM_{ij}^{NW_K} \; i = 1\ldots\ldots\ldots m \quad \text{----------} \quad (6)$$

The obtained vector R is normalized giving the normalized priority vector of sas-HG to ensure that the final ranking values will be between 0 and 1.

$$NV_i = \frac{r_i}{\sum r_i} \quad \text{----------} \quad (7)$$

The decreasing ordered hard goal indicates the final ranking, and the most important one is the one which has highest NV value

## Model Checking

Model checking is a technique to validate the nature of self adaptive systems against properties that relate to time (temporal properties). This technique inspects the target systems' desirable properties and identifies those that are temporal. To identify a temporal property of the system, the user needs a deep understanding of the behavior of the system and sophisticated knowledge of the entire formulation of the system [24].

Given a model of a system M, and identifying a temporal property Ø, a model checking algorithm determines whether or not M satisfies Ø. M |= Ø. [25] A Labelled transition system can be used as model. Model checking requires a specification of the target system's desirable properties, some of which are temporal .

Temporal properties of any software system can be converted into temporal operators as shown below (Table: 3)

**Table 3:** Temporal properties of SAS

| TEMPORAL PROPERTY | FUNCTIONALITY |
|---|---|
| ***EVENTUALLY*** | Requirement must hold eventually |
| ***UNTIL*** | Requirement must hold until the future position |
| ***AFTER , BEFORE*** | Requirement must hold after or before a particular event |
| ***IN*** | Requirement must hold during a particular event |
| ***AS TIMELY, AS DELAYED POSSIBLE*** | Requirement specifies something that should hold as soon as possible or should be delayed as long as possible |

## Progress and Current Status

The model, SAS-REQ gathers requirements of SAS using requirements as runtime entities, classifying the goals into hard goals (sas-HG) and soft goals(sas-SG). Model checking enables in identifying the temporal properties and whether the right requirements are gathered. Goal based modeling is also used to categorize the goals into functional and non functional ones. Using such a model, integrates several research approaches to requirements gathering in SAS. Integrating the identified requirements or a SAS and prioritizing the same is the current work.

This model aims to reduce any uncertainty as it combines several ways to gather requirements. The requirements of SAS systems will definitely fall into one of the phases used for modeling the SAS.

## Conclusion and Future Direction

In this paper we have discussed the different ways by which the requirements phase of a SAS can be gathered. Applying the approaches used by different researchers, a model has been built (SAS-REQ). This model expects to handle uncertainty that arises due to lack of proper requirements gathering Prioritising the hard and soft goals using REQ-PA approach has also been discussed. As part of the future work, it would be of interest to carry out a controlled experiment to combine the two models and compare and verify the experimental results.

## References


[1] Silva Souza, Vítor E., et al. "Awareness requirements for adaptive systems."*Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems*. ACM, 2011.

[2] Perez-Palacin, Diego, and Raffaela Mirandola. "Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation."Proceedings of the 5th ACM/SPEC international conference on Performance engineering. ACM, 2014.

[3] Russell, S. J. "Norvig. 2003." Artificial intelligence: a modern approach (2003).

[4] Souza, Vıctor E. Silva, Alexei Lapouchnian, and John Mylopoulos. "(Requirement) evolution requirements for adaptive systems." *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*. IEEE, 2012.

[5] Abeywickrama, Dhaminda B., Nicola Bicocchi, and Franco Zambonelli. "SOTA: Towards a general model for self-adaptive systems." *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*. IEEE, 2012.

[6] Bencomo, Nelly, et al. "Requirements reflection: requirements as runtime entities." Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2. ACM, 2010.

[7] Bencomo, Nelly. "Requirements for Self-adaptation." Generative and Transformational Techniques in Software Engineering IV. Springer Berlin Heidelberg, 2013. 271-296.

[8] Souza, Vıctor ES, and John Mylopoulos. "From awareness requirements to adaptive systems: A control-theoretic approach." *Requirements@ Run. Time (RE@ RunTime), 2011 2nd International Workshop on*. IEEE, 2011.

[9] Abeywickrama, Dhaminda B., and Franco Zambonelli. "Model checking goal-oriented requirements for self-adaptive systems." *Engineering of Computer Based Systems (ECBS), 2012 IEEE 19th International Conference and Workshops on*. IEEE, 2012.

[10] Bencomo, Nelly, Amel Belaggoun, and Valerie Issarny. "Dynamic decision networks for decision-making in self-adaptive systems: a case study." Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on. IEEE, 2013.

[11] Ahmad, Manzoor. DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE. Diss. Universidade Nova de Lisboa, Portugal, 2014.

[12] Dabbagh, Mohammad, and Sai Peck Lee. "An Approach for Integrating the Prioritization of Functional and Nonfunctional Requirements." The Scientific World Journal 2014 (2014).

[13] Abeywickrama, Dhaminda B., Nicola Bicocchi, and Franco Zambonelli. "SOTA: Towards a general model for self-adaptive systems." *Enabling*

*Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*. IEEE, 2012.

[14] Völter, Markus, et al. Model-driven software development: technology, engineering, management. John Wiley & Sons, 2013..

[15] Yang, Zhuoqun, et al. "A Systematic Literature Review of Requirements Modeling and Analysis for Self-adaptive Systems." *Requirements Engineering: Foundation for Software Quality*. Springer International Publishing, 2014. 55-71.

[16] Welsh, Kristopher, Pete Sawyer, and Nelly Bencomo. "Towards requirements aware systems: Run-time resolution of design-time assumptions." *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*. IEEE, 2011.

[17] Ahmad, Manzoor. DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE. Diss. Universidade Nova de Lisboa, Portugal, 2014.

[18] Bencomo, Nelly, et al. "Software Engineering Processes for Self-adaptive Systems." Software Engineering for Self-adaptive Systems 2 7475 (2012).

[19] Cordy, Maxime, et al. "Model checking adaptive software with featured transition systems." *Assurances for Self-Adaptive Systems*. Springer Berlin Heidelberg, 2013. 1-29.

[20] Souza, Vıtor E. Silva, Alexei Lapouchnian, and John Mylopoulos. "(Requirement) evolution requirements for adaptive systems." *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*. IEEE, 2012.

[21] T. L. Saaty, Fundamentals of the Analytical Hierarchy Process,RWS Publications, 1994.

[22] Ejnioui, Abdel, Carlos E. Otero, and Luis D. Otero. "Prioritisation of software requirements using grey relational analysis." International Journal of Computer Applications in Technology 47.2 (2013): 100-109.

[23] T.S. Liou and M.-J. J. Wang, "Ranking fuzzy numbers with integral value," Fuzzy Sets and Systems, vol. 50, no. 3, pp. 247– 255, 1992.

[24] Yang, Jinlin, and David Evans. "Dynamically inferring temporal properties."*Proceedings of the 5th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*. ACM, 2004.

[25] Bencomo, Nelly, Amel Belaggoun, and Valerie Issarny. "Dynamic decision networks for decision-making in self-adaptive systems: a case study." Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on. IEEE, 2013.