Self-Adaptive Quality Requirement Elicitation Process for Legacy Systems: A Case Study in HealthCare

Nour Ali
School of Computing, Engineering and Mathematics
University of Brighton
Lewes Road, Brighton-UK
n.ali2@brighton.ac.uk

Alfonso Martínez-Martínez, Lorena Ayuso-Pérez, Angelina Espinoza
Universidad Autonoma Metropolitana
Iztapalapa-Mexico City
{Almm,cbi210343101,aespinoza}@xanum.uam.mx

ABSTRACT
Legacy systems need to be continuously maintained and re-engineered to improve their provision of services and improve quality attributes. An approach that promises to improve quality attributes and reduce human maintenance tasks is the self-adaptive approach, where software systems modify their own behaviour. However, there is little guidance in the literature on how to migrate to a self-adaptive system and evaluate which features should be designed/implemented with self-adaptive behaviour. In this paper, we describe a process called Self-Adaptive Quality Requirement Elicitation Process (SAQEP), a process that allows eliciting quality attribute requirements from legacy system stakeholders and specify which of these requirements can be taken account to be implemented in a self-adaptation system. The SAQEP has been applied to elicit the self-adaptive quality requirements of a legacy system in a Mexican hospital. We also discuss our experience applying this approach.

CCS Concepts
• Software and its engineering → Software organization and properties → Extra-functional properties.

Keywords
Self-adaptive requirements; self-adaptive scenario; quality attribute; quality attribute scenario; self-property.

1. INTRODUCTION
Legacy software systems commonly require continuous maintenance tasks. The improvements can be in terms of: 1) providing new functionality and 2) improving the quality attributes provided by a system. If these improvements are driven by runtime faults and administrators/developers/operators have to manually perform these fixes continuously and repetitively, then providing self-adaptive capabilities to a system could be a plausible solution. A self-adaptive system is defined as [1]: "*software which modifies its own behavior in response to changes in its operating environment. By operating environment, we mean anything observable by the software system, such as end-user input, external hardware devices and sensors, or program instrumentation"*. The common goal of self-adaptive system approaches is to tackle the system evolution in an autonomic fashion, i.e., with minimum human intervention. Therefore, reducing administrators' efforts in technical tasks to maintain the system operation and improve system quality attributes.

There have been several approaches that provide guidelines on how to design and implement self-adaptive solutions and architectures such as [1] [5]. However, there has not been enough approaches to provide guidelines on how to elicit the requirements to be taken into account developing self-adaptive systems. For example, the authors in [12] present the re-implementation of a self-adaptive legacy system but do not describe how they have elicited the quality attributes and decided on the self-adaptive properties to include in the re-engineering process. Therefore, there is no synthesized and systematic process that provides guidelines on how to come up with self-adaptive requirements of a legacy system that need to be included in the re-design and re-implementation. This systematic process is needed as software engineers need guidelines to elicit requirements to re-engineering software for self-adaption. Self-adaptation solutions can require new costs such as investing in new software platforms, middleware, developers and hardware equipment. However, software engineers need to consider that not all challenges need to have a self-adaptation solution.

In this paper, we introduce Self-Adaptive Quality Requirement Elicitation Process (SAQEP), a process to elicit the new legacy quality attribute requirements and to analyse which of them can be self-adaptive scenarios. Our process initially identifies a set of Quality Attribute Scenarios [3] that could potentially be self-adaptive ones, and finally produces a set of self-adaptive scenarios that will be taken into account in re-engineering the system to include a self-adaptive behaviour. To illustrate SAQEP and perform an initial evaluation, we have used a health-care case study, in the context of a legacy system currently in operation in a Mexican hospital.

The paper is organized as follows: Section 2 presents de background to follow the rest of the paper, Section 3 presents the process to elicit the self-adaptive scenarios. Section 4 details a case study in the health-care domain to evaluate our approach. Section 5 presents related work. Finally, section 6 explains the general conclusions and the on-going and future work.

2. Background
Two of the concepts that we have used in our process are:

- *Quality Attribute Scenario (QAS)* [4]. It is a quality-attribute-specific requirement described in a template which consists of six parts: 1) *Source of Stimulus* - This is some entity (a human, a computer system, or any other actuator) that generated the stimulus, 2) *Stimulus* - It is a condition that needs to be considered when it arrives at a system, 3) *Environment* – The system condition when the stimulus occurred, 4) *Artifact* – The stimulated system artifact, 5) *Response* - It is the activity undertaken after the arrival of the stimulus, and 6) *Response Measure* - When the response

occurs, it should be measurable in some fashion so that the requirement can be tested.

- *Self-Properties (also called as self-* properties):* They are the characteristics that allow self-adaptive software to respond to changes at runtime [5] [6]. Several self-properties proposed in [5], but not limited to, are:
    - Self-configuration*:* Automated configuration of components and systems that follow high-level policies.
    - Self-optimization: Components and systems continually seek opportunities to improve their own performance and efficiency.
    - Self-healing: A system automatically detects, performs diagnoses, and repairs localized software and hardware problems.
    - Self-protection: A system automatically defends itself against malicious attacks or cascading failures.

Several authors argue that self-properties are related to quality attributes. Salehei et al. [6] states that there is a relationship between self-properties and software quality factors [10], and the existent knowledge on quality factors, metrics and requirements support self-adaptive software development. For example, also Salehei et al. in [6] relate self-configuring to maintainability, functionality, portability, usability and reliability (depending on the reconfiguring definition); self-healing to availability, survivability, maintainability, and reliability; self-optimization to efficiency and functionality; self-protecting to reliability and functionality. Also, Ganek and Corbi [13] state that availability, maintainability and reliability are maximized by self-healing. This relationship between quality factors/requirements and self-properties is important for identifying the legacy system's requirements that can be implemented with an adaptive approach; and this is the backbone for the process that we propose.

3. A Process for Eliciting Self-Adaptive Quality Requirements of Legacy Systems

In this section, we define a Self-Adaptive Quality Requirement Elicitation Process (SAQEP), which is based on Quality Attribute Scenarios proposed by the SEI [4]. This process allows the requirements of a legacy system to be analysed to be reengineered for self-adaptation.

STAGE 1: **Specify Quality Attribute Scenarios (QAS) from the Legacy System**

TASK 1.1: Eliciting Challenging Situations

This step considers answering the question of why do we need to reengineer the system, what are the problems that the current system has? Usually, as this is a legacy system, stakeholders will describe problems/challenging situations that they are encountering in the current system instead of only needs. To be able to answer these questions, the following can be performed:

- Interview Stakeholders
    - Users. This activity involves interviewing the users to describe the problems they currently are experiencing from the system.
    - System Administrators, Operators or Developers. This activity implies to prepare, and carry out the meetings with the legacy system administrators and operators in order to collect the current system challenging situations.
- Analyse System Logs. Many software systems generate logs to monitor their execution. These system logs need to be analysed in order to identify the challenging situations (issues) that a legacy system is currently undergoing.
- Describe the Challenging Situation. From the interviews and system log analysis, a list of problems or challenging situations is identified. The challenging situations can be described by indicating *the elements* of the system involved and *the* possible *reasons* or/and the implications e.g., "The server fails every 2 days due to high requests". The first work product generated in this process is the list containing the identified system challenging situations. This activity implies to prepare such list, and to validate it with the stakeholders involved.

Outcome: List containing the challenging situations..

TASK 1.2: Formalize the Challenging Situations into QAS

1) For each challenging situation, identify which QA describes it. To identify the QA, the SEI Quality attribute (e.g. testability) list can be used to classify them. If not found in the SEI, you can use the SQuaRE model in ISO 25010 [2].
2) Define the QAS: Once you know the QA, you start complementing the challenging situation with additional information to define a complete QAS. This can be performed by either directly translating the challenging situations into a QAS based on the guidelines provided in [3] and/or by following Generic QAS templates associated to quality attributes, if applicable, as presented in [4]. In this step, several of the information of the QAS sections could be incomplete as they have not been collected in the list of challenging situations and therefore, it is recommendable to iteratively work with stakeholders to complement this information.
    a. Define the **Stimulus section**, for this purpose, we recommend dividing this section into three parts:
        i. Challenging Situation: A description of the event that is causing the problem.
        ii. Current Actions: List the current actions that are being performed in order to solve the current problem and indicate who is performing them. There are several cases where administrators/developers/operators are manually (or semi-automatically) fixing the problems.
        iii. Current Measure: List the costs in efforts (e.g., person per task and time per person), money, current quality attribute measures (e.g., response time,), etc. from currently having the problem and using the existing solution to the problem.
    b. Define the Response Measure. While defining the response measure, evaluate whether the proposed response will improve the Current Measure section of the Stimulus. If not, keep on defining responses until the Response Measure improves the Current Measure.
3) Refine the QAS: In this step, the initial QAS is validated with Stakeholders.

Outcome: QAS List

STAGE 2: Identify Self-Adaptive Quality Attribute Scenarios (SAQAS)

In this stage, the QAS produced are analysed to determine whether self-adaptation is the appropriate solution or not. Several QAS will be already describing self-adaptive solutions, while others could be considered to become self-adaptive. We define

QAS that describe self-adaptive solutions as *Self-Adaptive Quality Attribute Scenarios (SAQAS)*. The following steps can be performed:

TASK 2.1: Identify Potential Self-Adaptive Quality Attribute Scenario (SAQAS)

In this task, we will only select a subset of the QAS list which could potentially be SAQAS. For each QAS, review their sections to identify potential SAQAS. All Potential SAQAS are ones that have:

1) The **Environment** section describes a runtime condition. For example, it is indicated that the stimulus occurs at runtime or a runtime situation e.g., when the system is overloaded.

In addition, a Potential SAQAS can have one of the following:

1) The **Response** indicates actions to be performed by the system.
2) The **Response** states that a set of actions are to be conducted repeatedly by stakeholders.

Outcome: List of Potential SAQAS.

TASK 2.2: Determine SAQAS

For each potential SAQAS obtained in Task 2.1, an analysis is made in order to determine whether they can be performed through self-adaptation or not. In this Task, the Response and Response Measure are analysed. The Response section of a SAQAS should define the two characteristics that define a self-adaptive situation, according to [1]. These characteristics are that the system at runtime needs to 1) observe parts of its behaviour and 2) modify its own behaviour. Therefore, we propose the following to select SAQAS:

CHOICE 2.2.1 Check if the Response already contains these two characteristics. The first characteristic of *system* observation could have been written by having verbs that are synonymous for observation such as "detects", "checks", "identifies", "discovers", etc. The second characteristic is related to the *system* performing actions such as "configuring", "removing", "adding", etc. If this is already satisfied, then this is a SAQAS.

CHOICE 2.2.2 If the current Response does not indicate these two, then the Response section is analysed to determine if it can be redefined in this form or not. For example, a Response indicating that stakeholders detect an issue and/or perform the solution manually, can be determined to be redefined by replacing these actions with ones performed by the system. If this is not possible then this is not a SAQAS. If this is possible, then:

a. Write the Potential Response and Potential Response Measure. To define the Potential Response, stakeholders are involved to help in determining the potential response measurements.
b. If the new Potential Response Measure is an improvement to the Response Measure in QAS, then this is a SAQAS that replaces the QAS. The SAQAS will have the Potential Response and Potential Response Measure replacing the QAS Response and Response Measure, iteratively.

Outcome: Selected SAQAS

Table 1. Self-Adaptive General Scenario

Source	Administrator, Developer, Operator, System, User	
Stimulus	Challenging Situation	
	Current Actions	
	Current Measure	
Artifact	Locate which architectural element(s) of the legacy system are affected by the stimulus. These elements can be components (servers, software, etc.), connectors, services, subsystems, hardware.	
Environment	Runtime	
Response	Actions	Indicate the actions that the *system* will perform in: 1) observing its own behaviour and 2) self-adapting its behaviour.
	Self-Adaptive Response	List the self-adaptive properties
Response Measure	Effort, QAS measurements, Expenditure, etc.	

STAGE 3: Rewrite the Selected SAQAS

In this step, the Potential SAQAS are rewritten to follow **Table 1**. To do this, the Source, Stimulus, Artifact and Environment sections will not change from the QAS. For all SAQAS add in the Response, a section that lists the Self-Properties applicable in this scenario. These self-properties list can come from the traditional self-* derived from IBM [5]. In several cases, the stakeholders can come up with self-properties from their experience and contribute towards their definition and context. In this step, some guidance can be used to identify self-properties. As mentioned in the background section, there has been research that relates Quality Attributes to corresponding Self-Properties [6].

For only those SAQAS that have been chosen from CHOICE 2.2.2, include the new Response and Response Measure.

Outcome: List of SAQAS.

STAGE 4: Prioritization Stakeholders will prioritize by voting the SAQAS and the regular QAS based on Difficulty and Importance. The self-properties and response measures can influence on the stakeholders' decisions. The final outcome is a final set of prioritized SAQAS which will drive the architectural decisions.

4. Case Study

This section presents a case study which we have used to evaluate our approach. The description of the case study is presented as well as the application of the process described in Section 3. Also, a discussion section is included to analyse the case study results.

4.1 Description

Our case study was performed in the context of a Mexican public hospital: Instituto Nacional de Rehabilitación (INR), located in Mexico City. It is an institution dedicated to rehabilitation medicine attending physical disabilities. Currently, INR has several information systems. One of the most important systems is the Picture and Archiving and Communication System (PACS) [7], which is responsible for transferring, storing, and displaying medical images (X-ray, tomography or ultrasound). INR have a deployed and running PACS implementation called the PACS-INR [8]. This presents a 3-tier architecture: Client, Business Logic, and Data Management.

PACS-INR currently has several functionalities that are impacting negatively on 1) the delivery of services to end-users such as doctors and patients, and 2) the effort invested by system

administrators in technical and maintenance tasks. Therefore, the PACS-INR administration area, including the system administrators are very interested in an approach that could automatize several of these tasks, with no human intervention if possible and improve the experience of end-users.

The PACS-INR subsystem can be re-engineered to include self-adaptive scenarios. Therefore, we have analysed its functionalities and technical tasks by applying our process described in section 3 and we have been able to specify several SAQAS.

Table 2. Challenging situations and their Quality Attribute

ID	Challenging Situation	Quality Attribute
1	A failure is detected in the application, file system or database servers. This failure prevents the normal PACS-INR operation.	Reliability Sub-attribute: Availability
2	A new version of the visualization component is released which must be installed manually in each client PC of the doctors.	Portability Sub-attribute: Installability Replaceability Adaptability
3	Each time a new equipment for visualizing medical images is installed to be part of the PACS-INR system, the DICOM (Digital Imaging and Communications in Medicine) compatibility is assured in the new server since the PACS-INR system protocol to transfer images is specified in DICOM.	Compatibility Sub-attribute: Co-existence Interoperability

4.2 Applying the Process

In the following, we explain how we have applied the different stages of the process presented in section 3 to the PACS-INR subsystem.

4.2.1 STAGE 1: Specify Quality Attribute Scenarios (QAS) from the Legacy System

TASK 1.1: Eliciting Challenging Situations

For conducting this task, we interviewed Users and *System Administrators and Operators*. We also analysed log files with the stakeholders. We interviewed 1 doctor, the responsible Administrator and 2 Operators. For this purpose, a workshop was carried out with all the stakeholders of PACS-INR to identify functionality to improve the PACS behaviour. As a result, a list containing the challenging situations was obtained. Table 2 presents three out of the 13 challenging situations that we captured during this task.

TASK 1.2 Formalize the Challenging Situations into QAS

The first step is to identify the corresponding quality attributes for each challenging situation. For this purpose, we have used the SQuaRE model in ISO 25010 to allocate each challenging situation according to the software quality model in the standard. Table 2 shows this identification.

Step 2 involves defining the QAS for each challenging situation. For this purpose, we used the general templates provided in [4], here we mapped the ISO 2500's SQuaRE model attributes with the quality attributes considered by the SEI.

For example, to define the QAS for the challenging situation 1 in Table 2, the Availability SQuaRE sub-attribute corresponds to the Availability one in SEI [4]. The availability attribute has a general template to specify the specific availability scenarios [4].

Table 3**. QAS for the challenging situation 1 from Table 2**

Source	Internal to the system	
Stimulus	Challenging Situation	A crash is detected in the application, file system or database servers. This failure prevents the normal PACS-INR operation.
	Current Actions	The administrator manually sets up a mirror server by using the same parameters as the failed server. Once the mirror server is configured, the administrator performs the following reliability checks: - To verify that all the application, file system and database servers are in normal operation. - Several transactions are launched from the application server to the database server. Once the above checks are performed, the administrator publishes and activates the servers to be online to provide services to the end-users.
	Current Measure	The effort of one administrator takes 60 minutes
Artifact	- Application server - File system server - Database server	
Environment	Runtime	
Response	The Administrator is notified that there is a failure in any of the servers, and then he/she launches an automatic process that consists of a) configuring a mirror server and b) checking that the mirror server has been properly configured to ensure that doctors will be able to save, retrieve and visualize medical images, c) Publishes the new mirror server.	
Response Measure	-The repair time in executing the automatic process takes 5 minutes. -The effort of developing this automatic process is 1 developer during two months.	

For challenging situation 1, we also included the Source which is the internal system as the indication of the fault comes internally. We then separated the *Stimulus* into three sections. By following Step 4 we have written in conjunction to the stakeholder the Response measure section in order to specify an improvement compared to the Current Measure from the Stimulus section. We have validated this QAS with the stakeholder according to Step 5. We finally obtained 13 QAS corresponding to the 13 challenging situations. Table 3 shows the final QAS for the challenging situation 1 stated in Table 2.

4.2.2 STAGE 2: Identify Self-Adaptive Quality Attribute Scenarios (SAQAS)

TASK 2.1: Identify Potential Self-Adaptive Quality Attribute Scenario (SAQAS)

In this section, we selected all the QASs which have Runtime in the Environment section. As a result, one of the selected QAS is the one shown in Table 3. We then reviewed the response section of the QAS from Table 3. It can be noticed that the Response includes both actions performed by the Administrator and the system. Therefore, we conclude that this QAS is a Potential SAQAS. Also, the QAS for challenging situation 2 and 3 were selected to be potential SAQAS. In this task, we have identified 7 Potential SAQAS for this case study.

TASK 2.2: Determine SAQAS

In this task, we analysed our Potential SAQAS to determine if they are SAQAS. For each Potential SAQAS, we reviewed the two choices. For the QAS defined in Table 3 we analysed the Response section. According to the choices, we apply Choice 2.2.2. This is because there is an observation action but the administrator has to get the notification and then he manually/she launches the automatic system instead of the system itself does these actions itself. We analysed if this action can be performed by the system. We identified that it can. The Response already states that the system automatically performs actions. We then wrote the new Potential Response. Then we worked with the stakeholder to identify a new Potential Response.

Therefore, this QAS is determined as a SAQAS, and then it can be implemented with a self-adaptation approach.

For the 7 Potential SAQAS, we have selected 4 SAQAS.

For the Challenging Situations 2 and 3 in Table 3 they were not determined to be SAQAS. For Situation 2, the Potential Response Measure was considered to not improve the Current Measure since the installation time will be the same as if it is to be launched by an administrator or automatically by the software itself. Also, since the frequency of the needed installation is once per year, the effort to re-engineer the system to a self-adaptive scenario is big compared to the obtained benefits.

4.2.3 STAGE 3: Rewrite the Selected SAQAS

In this stage, we rewrote the selected SAQAS according to our analysis in the previous stage. Also, all SAQAS should have the self-properties section. For the QAS defined in Table 3, the SAQAS response and response measure are rewritten as in Table 4. We also include the self-properties section. For this purpose, we have analysed the quality attribute from Table 2 which is Reliability with the sub-attribute Availability from ISO 25000, then we have concluded that the self-properties are: *self-configuration, self-healing and self-awareness*. We also explored self-healing property. However, we did not consider it applying to this scenario as the scenario is not repairing the failure. It is only creating a temporal state to allow the availability of the system.

4.2.4 STAGE 4: Prioritization

Finally, we prioritized all the scenarios with the stakeholders. We will use these prioritized scenarios to develop the architecture in an iterative process.

Table 4. SAQAS for the challenging situation 1 from Table 2

Response	The PACS-INR system: 1) Detects that one of the server fails. 2) Automatically a) configures a mirror server and b) checks that the mirror server has been properly configured to ensure that doctors will be able to save, retrieve and visualize medical images, c) Publishes the new mirror server.
	Self-Awareness Self-Healing Self-Configuration
Response Measure	- The PACS-INR takes 6 minutes to detect and repair the failure. -The effort of developing this automatic process is 1 developer during two months.

4.3 Discussion

In this section, we discuss several of the lessons that we have learnt when applying SAQEP in the different stages of the process.

Close Interaction with Stakeholders to Define QAS

Stage 1: One of the issues we faced when we applied SAQEP in the case study is that the stakeholders did not directly communicate a challenging situation. Many would just describe a situation but without identifying the core problem. The software engineer who is applying SAQEP, needs to advice the stakeholder to correctly be able to identify the problem. In addition, the software engineer needs to work very closely with the stakeholder to be able to write the QAS. Stakeholders do not provide the information needed to specify the sections of the template.

Special Attention to Costs in Determining SAQAS

Stage 1: Special attention has to be made in deciding which measurement costs e.g., efforts man, speediness of task realization, response time, etc., to include in the response measure and the current measure when defining the QAS. This is highly important as it can have later implications to decide whether the response measure improves from the current measure when determining SAQAS. If several measurements are ignored or not considered, wrong decisions could be made in choosing SAQAS.

Stage 2: Another aspect to consider related to taking the decision to include a SAQAS or not is that there are cases when some of the measurements in the response measure improve from the current measure whereas others do not. In these cases, it is not directly clear if the QAS is more suitable to be a SAQAS or not. Therefore, stakeholders and analysts have to work out a trade-off between these measurements and prioritise which are more important for an organization or a task.

Self-Adaptive Expertise needed to apply SAQEP

Stage 2: Another issue, but not especially a drawback, is that the SAQEP application requires the software engineer in charge to have expertise in the self-adaption paradigm to properly identify the self-adaptive quality attribute scenarios. This expertise is needed specifically to identify which manual activities in the Response section can be automatically performed by the system and to identify the events to be observed. This is totally oriented to model a self-adaptive behaviour.

Stage 3: When re-writing the SAQAS and defining the self-adaptive properties, the software engineer who applies SAQEP has to have a deep understanding of the different self-adaptive properties available and know how to identify if there is a mapping between the self-property and the quality attribute.

Guidance in identifying self-properties from mapping quality attributes literature

Stage 3: A fundamental task is to properly re-write the QAS into SAQAS and identify the self-adaptive properties. As part of our SAQEP, we mentioned that literature exists that attempts to map self-properties to quality attributes. After applying our case study,

we searched for formal approaches to make this mapping. For example, for our SAQAS in Table 4 we identified 3 self-properties for a reliability/availability quality attribute. The Self-configuration property has been in line with Salehei et al. [6], which states that self-configuring can be related to reliability. For self-healing, Ganek and Corbi [13] state that reliability is maximized by self-healing. For self-awareness, we did not found a previous approach stating the relationship between this self-property and any quality attributes. As a result, we believe that more work can be made in relating quality attributes and different self-properties.

5. Related Work

We have not directly found a systematic process for eliciting requirements to re-engineer legacy systems with self-adaptation. Similar research to SAQEP is the one presented in [14]. Even though they do not present a systematic process to allow software engineers to apply it, they describe how they have used QASs to consider self-adaptive properties in the design of an architecture. In SAQEP, we provide a set of steps to guide the software engineer in identifying potential self-adaptive scenarios. For example, SAQEP indicates to compare current measures against response measures of a self-adaptive solution. SAQEP also extends the QAS template to include self-adaptive features in order to more explicitly drive the architectural design.

Another approach at the requirements stage is [11]. In this paper, the author provides a brief process for modelling adaptation requirements based on the goal approach. However, this process is not quality attribute driven and does not provide guidance for eliciting the self-adaptive requirements.

6. Conclusions and Further Work

We have introduced a process called SAQEP which elicits new quality attribute requirements from stakeholders of a legacy system to evaluate which to include as self-adaptive requirements. These requirements will be used to re-engineer the system. We believe it is one of the first systematic processes that provides guidelines for conducting a quality attribute requirement elicitation for re-engineering a legacy system to become self-adaptive. We have applied our process to identify the self-adaptive requirements to be considered for re-engineering the PACS system at the INR. From 13 quality attribute challenging situations elicited from INR stakeholders, our process selected 7 self-adaptive quality attribute scenarios. These scenarios specify the quality requirements to be considered in re-engineering the architecture in further stages. By applying our process at INR, we have discussed several lessons learnt.

Our further work includes evaluating the SAQEP after re-engineering the software architecture of the healthcare legacy system with the new self-adaptive properties. We also plan to refine our process with the several lessons learnt such as including more guidelines in defining costs, risks, and mapping quality attributes to self-properties. In addition, we will apply our refined process in several other systems to be re-engineered to become self-adaptive.

7. ACKNOWLEDGMENTS

We would like to thank Instituto Nacional de Rehabilitacion (INR) for providing access to their system and staff, specially to Marco Antonio Nuñez Gaona. This work was partially funded by The Royal Academy of Engineering (NRCP1516/1/39) and The Royal Society (NI150203) through the Newton fund scheme.

8. REFERENCES

[1] Oreizy, P., Gorlick, M. M., Taylor, R. N., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D. S., and Wolf, A. L.; An architecture-based approach to self-adaptive software; IEEE Intelligent Sys. Vol. 14-3, pp 54-62; 1999

[2] ISO/IEC 25010.4 Software engineering- Software product Quality Requirements and Evaluation (SQuaRE) Quality model, 2009

[3] Bass, L., Klein, M., Moreno, G., Applicability of General Scenarios to the Architecture Tradeoff Analysis Methods, CMU/SEI-2001-TR-014 ESC-TR-2001-014 Oct 2001.

[4] Bass, L., Clements, P., Kazman, R., Software Architecture in Practice, 3rd Edition, Addison Wesley, 2012.

[5] Kephart, J.O., Chess, D.M.; The vision of autonomic computing; Computer, vol. 36-1, pp. 41-50; 2003

[6] Salehie, M., and Tahvildari, L.; Self-Adaptive Software: Landscape and Research Challenges; ACM Transactions on Autonomous and Adaptive Systems; Vol 4-2, pp 2-39; 2009.

[7] Huang, HK: PACS and Imaging Informatics. Basic Principles and Applications, New Jersey: Wiley Blackwell 2nd Edition, 2010

[8] Gutiérrez-Martínez J., Núñez-Gaona M.A., Aguirre-Meneses H.; Business Model for the Security of a Large-Scale PACS, Compliance with ISO/27002:2013 Standard; Journal of Digital Imaging; Vol. 28-4, pp 481-491; 2015.

[9] Pianykh, O: Digital Imaging and Communications in Medicine (DICOM) Cap 11. DICOM Media and Security, Springer 2nd Edition, 2012.

[10] ISO/IEC 9126-1 2001. ISO/IEC 9126-1 Standard: Software Eng. -Product quality - Part 1: Quality model, Int. Standard Organization, 2001.

[11] Amoui, M.; "Evolving Software Systems Towards Adaptability"; 16th Working Conference on Reverse Engineering; 2009.

[12] Mulcahy, J.J. and Huang, S.; Autonomic Software Systems - Developing for Self-Managing Legacy Systems; IEEE International Conference on Software Maintenance and Evolution; 2014.

[13] Ganek, A. G. and Corbi, T. A. The dawning of the autonomic computing era. IBM Sys. Journal, Special Issues on Autonomic Computing 42, 5-18; 2003.

[14] Zhu Y., Huang G. and Mei H.; Quality Attribute Scenario Based Architectural Modeling for Self-Adaptation - Supported by Architecture-based Reflective Middleware; 11th Asia-Pacific Software Engineering Conference, 2004.