Research Journal of Computer and Information Technology Sciences ISSN 2320 – 6527
Vol. **5(1),** 1-5, February **(2017)** *Res. J. Computer and IT Sci.*

Requirements engineering framework for automated emergency departments of hospitals

Rasika Mallya[1,2] and Snehalata Kothari[1*]
[1]PAHER University, Udaipur, Rajasthan, India
[2]Navinchandra Mehta Institute of Technology and Development, Dadar (W), MS, India
skothariudr@gmail.com

Available online at: www.isca.in
Received 30th November 2016, revised 5th January 2017, accepted 10th February 2017

Abstract

In Emergency departments of hospitals, users have to take critical decisions related to patients and resources within short time. The increased demand of quality resources at critical time in emergency department is a major challenge to provide quick medical assistance. To automate and improve emergency healthcare process, autonomic computing framework is used in both intra and inter-organizational services. This framework gives dynamic nature to emergency department processes. Emergency department environment has to deal with sudden changes in workloads due to emergencies like natural disasters, fire or terrorist attacks. Sometimes, in ICU, patients have to be monitored continuously. The emergency department has to ensure care continuity and Quality of Service (QoS) at optimum cost. The framework of autonomic computing is appropriate for emergency department environment since it has self-management capabilities with self-configuring, self-optimizing, self-protecting and context awareness features. Hence, according to current scenario, autonomic computer software adapts its behavior at run-time. This framework proposes dynamic requirements engineering using the self-adaptive software approach with model driven architecture (MDA) to gather functional and non-functional requirements. As the level of autonomy increases, various requirements can be identified at runtime. Autonomy architectures like (MAPE-K) Monitor-Analyze-Perform-Evaluate cycle and (IMD) Intelligent Machine Design can be used to bring requirements engineering autonomy during software development.

Keywords: Autonomic computing, Self-management, Requirement engineering, Level of autonomy, MAPE-K cycle.

Introduction

The emergency departments are the most critical units of healthcare organization where proper medical services are to be provided as fast as possible to the patients. The services start from admitting the patient, allocating bed, consulting specialized doctor till making the patient comfortable by giving suitable treatment. The hospital staff is also under stress because of giving quality services to patients and to make efficient use of resources.

The healthcare professionals are finding new dynamic ways to provide better medical services to admitted patients at optimum cost and to reduce stress of hospital staff by providing automated equipments.

The dynamic nature of emergency department makes the automation of medical service complex. There can be suddenly more workload because of natural disaster, terrorists attack or spreading of epidemic disease. Such sudden changes in workload are difficult to predict[1].

Emergency department consist of multiple actors like doctors, pathology persons, nurses, technicians, social workers etc. These actors are highly dependent on various medical softwares to make their tasks easy and fast. Some medical softwares help health agents to make decisions in various complicated scenarios. But due to increase in complexity of medical cases, there is a need of runtime decisions given by medical softwares based on complexity of scenario.

Such system will make Emergency department self-managed and it will improve QoS. This paper explains level autonomy required during requirements gathering. This will help to increase autonomy of Emergency department systems.

Requirements Engineering: Requirements engineering is a set of tasks for identification of purpose of a software-intensive system. Hence, Requirements engineering acts as bridge between real-world needs of users, customers and other actors affected by software system and opportunities afforded by software intensive technologies.

Requirement engineering covers elicitation, analysis, specification and validation. Requirements change happen because of business conditions, user needs, resource availability and environment. Requirements change management is an important activity in self-adaptive software. Requirements engineering activities are performed by the analyst/designer at design time. But in context of self-adaptive software, changes in environment are expected continuously. Thus, self-adaptive software must be able to cope up with these changes and adapt new candidate solution to resolve dynamic needs. Hence, the CARE (Continuous Adaptive Requirements Engineering) framework suggests adaptation types for requirements engineering of self-adaptive software at run-time with the involvement of end-user[2]. i. Type 1 requirements engineering handles set of requirements which are implemented at design-time and alternate specifications are also mentioned in domain itself. So it is similar to traditional requirements engineering. ii. Type 2 requirements engineering considers changes in context or preferences of requirements. The self-adaptive software must monitor such changes. In such case, it adapts to the most feasible alternative solution using different available services. Here, according to latest input, new solution is adapted. iii. Type 3 requirements engineering considers the changes which are unknown to user and analyst. The self-adaptive software can ask questions to the end-user for unanticipated events and it will add them dynamically into domain. Self-adaptive software uses ontologies to adapt these unanticipated events into system. Type 3 requirements engineering is most difficult to implement because it creates representations for possible adaptive reactions to new inputs. iv. Here, the software gets automated nature as once an appropriate solution is found for any type 3 requirement, this new requirement is updated in original specification at run-time. If no solution is found, the system goes for type-4. v. Type 4 requirements consist of type 3 events for which no possible solution or specification is identified, so here completely new functionality or ontology is to be developed[3].

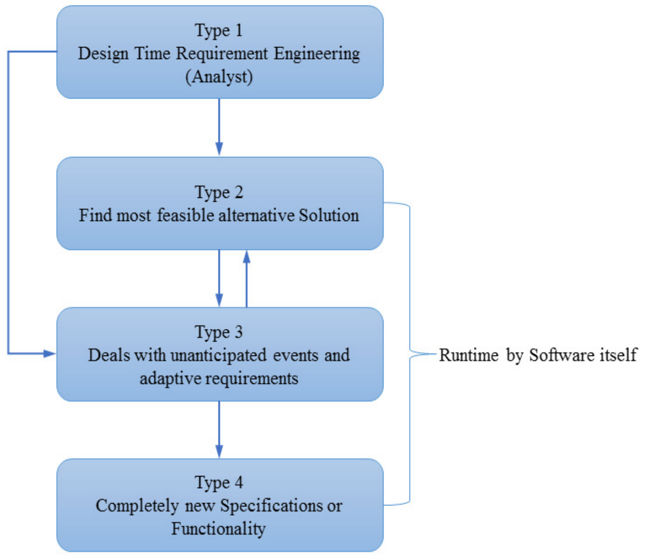
Type 1
Design Time Requirement Engineering
(Analyst)
Type 2
Find most feasible alternative Solution
Type 3
Deals with unanticipated events and adaptive requirements
Type 4
Completely new Specifications or Functionality
Runtime by Software itself

Figure-1: Types of adaptations during RE.

Goal-Oriented Requirements engineering and Ontologies: Goals are high-level objectives of business or organization. The goal can be objective which can be completed by multiple agents of system. Agents are active components in system[4]. GORE identifies scenarios for different context of system. A scenario is a temporal sequence of interactions among different agents to achieve some purpose. Ontologies is a representation or artifact for specifying the knowledge in a certain domain. Following ontologies are used for requirements engineering[5]: i. Requirements Ontology: This ontology can be used for requirements elicitation. Different types of requirements like functional requirements, non-functional requirements, organizational requirements can be elicitated using this ontology. ii. Requirements Specification Document Ontology: Here, different scenarios are specified. These ontologies create requirement specification document. iii. Application Domain Ontology: This ontology represents application domain knowledge and business information required for software development. This ontology can identify dynamic and changing requirements i.e. this ontology is useful for level 3 adaptation of self adaptive software. Hence, this type of ontology is important.

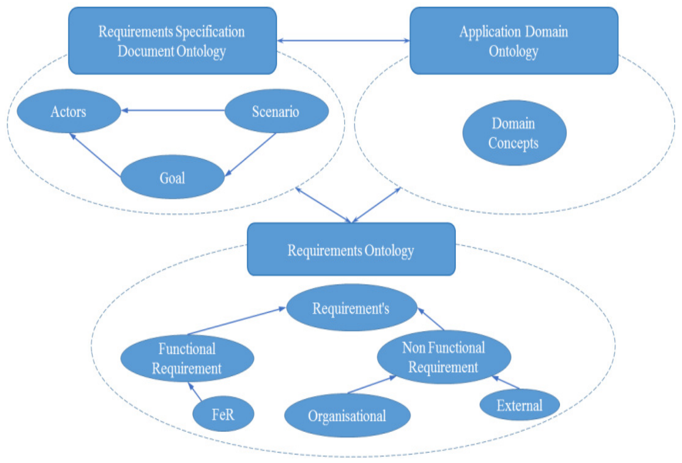

Figure-2: Types of ontologies for RE.

Autonomic Computing: Autonomic computing is the computing environment with the ability to manage itself and dynamically adapt to change according to business policies and objectives. If due to external or internal event, the system goes out of state, the system will always work towards returning it to its original state i.e. called self-healing[6].

Autonomic applications and systems are composed of autonomic elements. Their main characteristics are: i. Self-awareness: An autonomic application "knows itself" and is aware of its state behavior (Type 1 adaptation). ii. Self-configuring: Application should be able to configure and reconfigure itself under unpredictable conditions (Type 2 & 3adaptation). iii. Self-optimizing: Application should be able to detect suboptimal behaviors and optimize itself to improve its execution. iv. Self-Healing: An application should be able to detect and recover from potential problems and continue to function smoothly. v. Self-protecting: An application should be able to detect and protect its resources from internal and external attacks to maintain system security and integrity. vi. Context-aware: The system should be aware of its execution environment and be able to react to changes in environment.

Self-managing system/ application shall install component itself when it detects that system is missing some event (self-configuration), restarting a failed event after modification (self-healing), balancing the workload when an increase in load is observed (self-optimization) and taking resources offline if invalid event is detected (self-protecting). Such software is called self-managed software[7].

Research Objective: The research objective is to explore level of autonomy of automated system for emergency department when requirements change runtime. Ontologies and MAPE-k cycle can be applied to deal with runtime scenario changes.

Literature Review

Autonomic computing framework is introduced by IBM in 2001. An autonomic system provides autonomous computing environment which is self-managing and can hide its complexity from user. It will constantly check and optimize its status, automatically adapt itself to changing conditions. Self-management is achieved through self-governing, self-adaptation, self- organization, self-diagnosis of faults[6]. The structure of autonomic element is as shown below. An autonomic element consists of managed component and autonomic manager. The autonomic element operates in following manner:

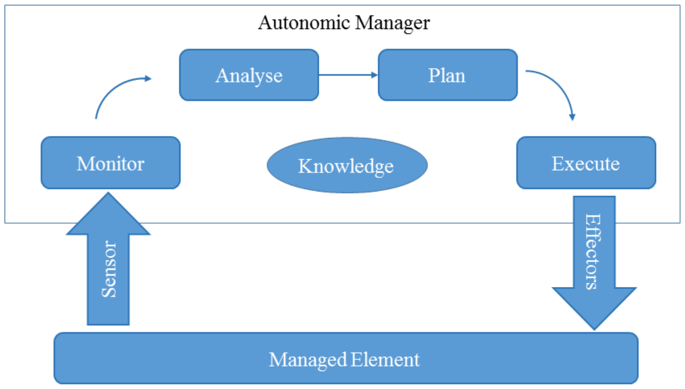

Figure-3: Structure of autonomic manager.

The managed element represents the software which is implementing autonomic behavior along with autonomic manager. Sensors collect information i.e. requirements about the managed element. Effectors carry out changes to the managed element. Autonomic manager is the agent who implements MAPE-K cycle in autonomic element[7]. The sensors collect data which helps autonomic manager to monitor the managed element. Based on the analysis done by autonomic manager, it executes changes in autonomic element through effectors.

To develop self-managed emergency department, MAPE-K cycle can be implemented in emergency department environment. The emergency department environment is set of managed elements. The sensors look at environment and pass information to autonomic manager. The autonomic manager can be triage nurse, physician and pathology lab technician. The autonomic manager monitors the changes, analyzes the scenarios, plan the necessary adaptations and execute those. To implement such automated softwares, all types of events or goals should be identified. For this, requirements engineering is required to be implemented at design time as well as run time. To perform requirements engineering dynamically some scenarios can be identified like triage process.

The Emergency Severity Index (ESI) is used to maximize efficiency at the ED, ESI is used to classify the patients coming into the ED. ESI is a five-level emergency department (ED) triage algorithm that provides medical classification of patients into five groups from 1 (most urgent) to 5 (least urgent) on the basis of acuity and resource needs. A triage nurse is the actor who is responsible for assigning ESI level to the patients coming to emergency department. The ESI level determines the waiting time of patients and urgency of treatment. e.g. ESI level 1 patients have no slack time and they should get treatment immediately. Emergency Departments are divided into care areas or emergency zones based on ESI levels categories. Consequently, the ESI level also determines the zone the patient will occupy. Following research design implements level of autonomy for requirements engineering based on ESI level and EHR of patient[1].

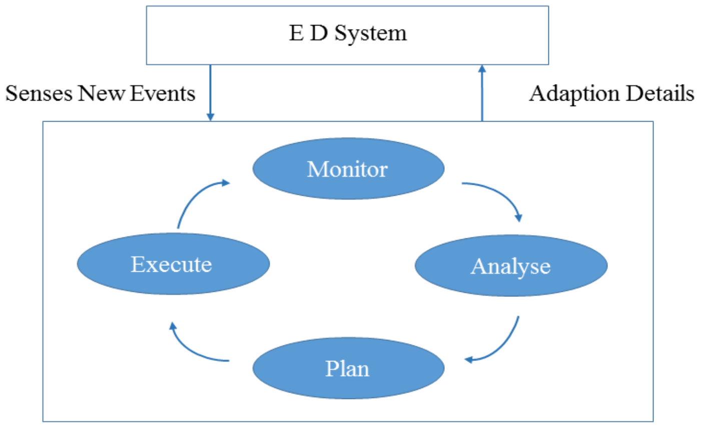

Figure-4: MAPE-K cycle for ED.

Research Experiment

For the smooth working flow at emergency department the common obstacles are delay or lack of adequate patient information, errors in patient details, assigning resources to patient as per need and many more. Hence, hospitals have adopted new technologies to enable automation, quick response, improved clinical decision making, efficient healthcare services. The general flow for emergency department is as follows:

When an emergency occurs, the ability to respond to uncontrolled event with highest speed and with coordination of resources, a big loss or damage to life can be prevented.

Responding to any uncontrolled or new event is possible with autonomic software due to its nature of self-management. The approach of self-management and context awareness of automated software will help in triage for prioritization, minimal wait time, fastest clinical decisions.

If the patient's health information is linked with EHR (Electronic Health Record) then patient's real-time information will be available to nurse or doctor for triage process. EHR contains patient's medical history, diagnoses, medications, treatment plans, immunization dates, allergies and lab reports.

Consider the ontology for triage process. This ontology will be having design time requirements specification about triage process. This ontology with the help of MAPE-k cycle will be useful to gather runtime requirements related to triage process. Thus, this ontology will help the system to bring autonomy in triage process though some unanticipated event will occur. During triage if the patient's EHR is found, requirements adaptation type 1 and 2 are implemented. If the patient's EHR not found then there should be one person with relative or friend who will provide information about patient. In this case, for triage process, type 3 requirement adaptation will be implemented[8]. For above description, following are the levels of autonomy in self-managed emergency department software. Related level of adaptation of requirement is also shown in Table-1.

Decision tree describes level of autonomy along with EHR and symptoms of patient's disease.

Table-1: Four Autonomy levels with adaptation levels.

Autonomy level	Activity	System	Adaptation level
Basic	Monitor the patient	EHR record found	Level 1
Managed	Analyze information, Make decisions	EHR record found + narration of disease by relative	Level 1
Predictive	Make decisions	EHR record found + symptoms of disease recognized	Level 2
Adaptive	Observe and patient's information	EHR record not found but symptoms of disease are known	Level 3
Autonomic	Monitor patient's symptoms, consult the specialized physician and take his opinion	EHR record found/ not found but symptoms are unknown and not matching with patient's history.	Level 4 (new functionality will be added in ontology)

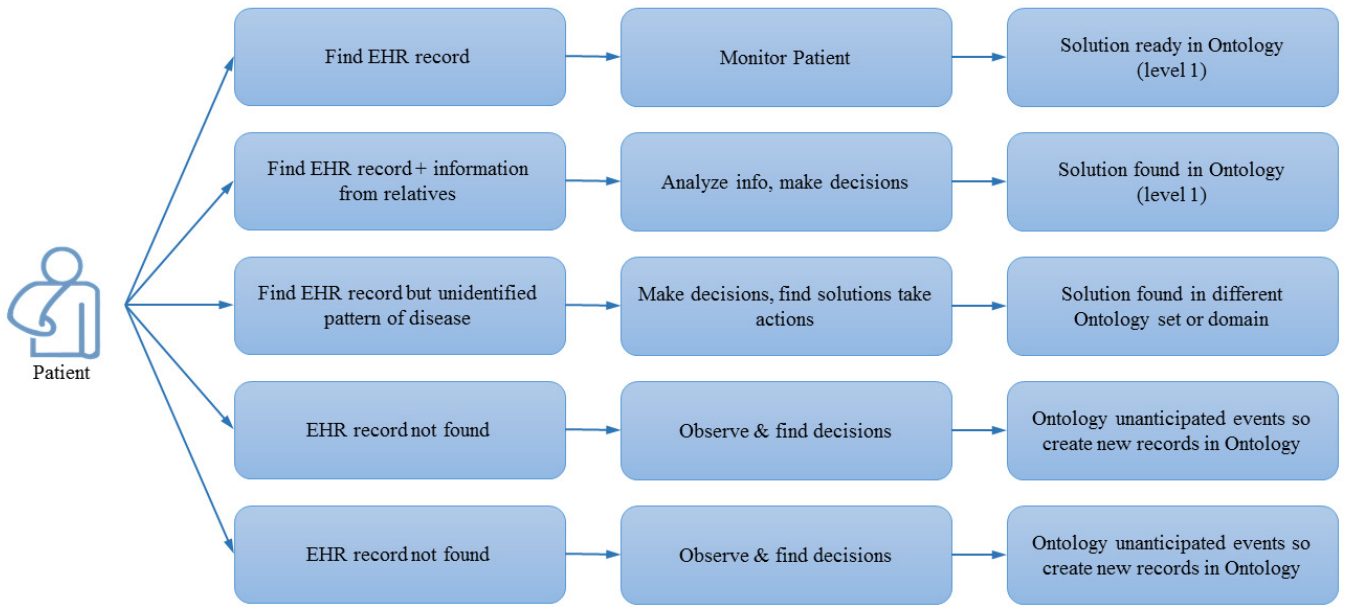
Patient
Find EHR record
Monitor Patient
Solution ready in Ontology (level 1)
Find EHR record + information from relatives
Analyze info, make decisions
Solution found in Ontology (level 1)
Find EHR record but unidentified pattern of disease
Make decisions, find solutions take actions
Solution found in different Ontology set or domain
EHR record not found
Observe & find decisions
Ontology unanticipated events so create new records in Ontology
EHR record not found
Observe & find decisions
Ontology unanticipated events so create new records in Ontology

Figure-5: Decision tree describing level of adaptation in triage process.

Implementation

To implement self- management of emergency department, level of autonomy during requirements engineering phase will be identified. The ontology specification will be created for basic set of requirements i.e. level 1 adaptation. So autonomy levels like basic, managed are implemented with the existing ontology specification[5] (Table-1). When predictive level of autonomy comes, requirement ontology can be searched since it may happen that the requirement and its specification is mentioned in ontology but not used before. When adaptive level of autonomy comes, application domain ontology can be used since the solution may not exist in existing requirement specification ontology. So using application domain ontology the requirement and its specification will be found and existing ontology will be updated i.e. level 3 adaptation. If the unidentified or unanticipated event occurs then new requirement specification will be added in application domain ontology as well as requirement specification ontology. This is autonomic level with level 4 adaptation. This is possible by implementation of MAPE-k cycle[9].

Conclusion

Autonomic computing is a promising approach for the development of computing systems that aim to self- manage the emergency department to offer fast service to patient and to reduce load of emergency department staff. It is known that ontologies are used to model requirements at design time. This paper proposes use of ontology to model requirements at runtime. Hence, Ontologies can be used in autonomic software to manage level of autonomy. In this paper, specific set of ontologies are used for triage process which helps triage nurse to take decisions fast about the treatment of patient. In the same direction, set of ontologies can be used for all transactions of emergency department which will help for self-management of activities during department. With the help of such self-managed autonomic software hospital staff can offer better and fast medical services to emergency patients.

References

1. Serene Almomen and Daniel Menascé (2011). An autonomic computing framework for self-managed emergency departments. HEALTHINF 2011 - International Conference on Health Informatics, 52-60.

2. Qureshi Nauman Ahmed (2011). Requirements engineering for self-adaptive software: bridging the gap between design-time and run-time. University of Trento, 59-84.

3. Berry Daniel M., Cheng Betty H.C. and Zhang J. (2005). The Four Levels of Requirements Engineering for and in Dynamic Adaptive Systems. *11th International Workshop on Requirements Engineering Foundation for Software Quality (REFSQ),* 5.

4. Lapouchnian Alexei (2005). Goal-Oriented Requirements Engineering: An Overview of the Current Research. *University of Toronto,* 30.

5. Castañeda Verónica, Ballejos Luciana, Caliusco Ma. Laura and Galli Ma. Rosa (2010). The Use of Ontologies in Requirements Engineering. *Global Journal of Researches in Engineering*, 10(6), 1-7.

6. Parashar Manish and Hariri Salim (2005). Autonomic Computing: An Overview. J.P. Banˆatre et al. (Eds.): UPP 2004, LNCS 3566, 247-259.

7. Huebscher Markus C. and McCann Julie A. (2008). A survey of Autonomic Computing - degrees, models and applications. *ACM Computing Surveys* (CSUR), 40(3), 7.

8. Yahyaa Mona A., Yahyaa Manal A. and Dahanayake Dr. Ajantha (2013). Autonomic Computing: A Framework to Identify Autonomy Requirements. *Procedia Computer Science,* 20, 235-241.

9. Sterritt Roy, Parashar Manish, Tianfield Huaglory and Unland Rainer (2005). A concise introduction to autonomic computing. *Advanced Engineering Informatics*, 19(3), 181-187.