A Context-Aware System to Communicate Urgency Cues to Health Care Workers

Qi Wei

Master of Telematics - Communication Networks and Networked Services (2

Submission date:	May 2014
Supervisor:	Rolv Bræk, ITEM
Co-supervisor:	Joakim Klemets, ITEM

Norwegian University of Science and Technology
Department of Telematics

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING

THESIS ASSIGNMENT

Student’s name:	Qi Wei
Course:	TTM4905, master thesis
Thesis title:	A Context-Aware System to Communicate Urgency Cues to Health Care Workers

Thesis description:
When nurses receive calls on their mobile phones, they need to decide whether to engage in the call or not. Several contextual factors play roles in making this decision. For example, if a nurse is busy with another patient when the phone rings, she is reluctant leave that patient. Research shows, for example, that the urgency of a call is an important factor. Another important factor is the availability of colleagues.

The objective of this thesis is to develop a proof-of-concept system that automatically, or semiautomatically, detects and communicates context clues to health care workers. The system could utilize various information sources, such as sensors and patient records, to capture relevant context clues. These context clues are used by the health care workers to make more informed decisions.

Department:	Department of Telematics
Supervisor:	Joakim Klemets
Responsible professor:	Rolv Braek

NTNU – Trondheim
Norwegian University of
Science and Technology

A Context-Aware System to Communicate Urgency Cues to Health Care Workers

Qi Wei

Submission date:	May 2014
Responsible professor:	Rolv Braek
Supervisor:	Joakim Klemets

Norwegian University of Science and Technology

Department of Telematics

Abstract

In hospital, patients use nurse call system to call the nurse when they need help. On the other hand, a nurse could be busy with other patient when the nurse call is delivered. In this case, the nurse needs to decide if he should stop what he is doing and answer the nurse call. This decision normally depends on how urgent this nurse call is. However, the nurse only knows who is call, but do not know the current situation of the patient. In this paper, a context-aware urgency cue system is proposed in order to resolve the problem. This system is based on context-aware and case-based reasoning. It collects patients' newest context and provides a particular patient's urgency cue to the nurse call system, so the nurse call system can provide the urgency cue with the nurse call to the nurse.

Contents

List of Figures

List of Tables

List of Acronyms

oNCS	ontology-based Nurse Call System
EPR	Electronic Patient Record
FT	Finger Temperature
CBR	Case-Based Reasoning
RFID	Radio Frequency, Identification
XML	Extensible Markup Language
RDF	Resource Description Framework
NTNU	Norwegian University of Science and Technology
EPR	Electronic Patient Record
W3C	World Wide Web Consortium
OWL	Web Ontology Language
DAO	Data Access Object
JDBC	Java Database Connectivity
API	Application Interface
GUI	Graphical User Interfaces

Chapter 1 Introduction

Now communication and electronic technologies are widely used in hospitals. For instance, wireless nurse call system is quite common in hospitals. As described in [1], a typical wireless nurse call system includes call buttons, a central console and wireless phones. Call buttons are usually placed in patients' rooms, toilets and so on. Central console is a big screen set up at the nurse station. Each nurse holds a wireless phone. When a patient needs nurse's help, he (or she) presses the button that is the nearest one to him. After the button is pressed, a nurse call will be generated and delivered to a wireless phone that is held by a nurse. At the meantime, there will be information printed out on the central console, which tells which room or which patient is calling.

As mentioned in [2], nurse call system also brings interruption problem for nurses while it makes it easy for patients to call the nurses. For example, a patient could call a nurse when the nurse is treating another patient. In this case, the nurse would be interrupted by the call. The reason that the patient calls the nurse could be different. For example, the patient could feel terrible and need some treatment, the patient could just need some water, or the patient could want to ask for some information. Normally, the nurse does not know why the patient makes the call. When the nurse is busy to treat a patient, it is hard for him to decide whether he should answer the phone, since he does not know which patient is in a more urgent situation. So if there are some urgency cues of the caller when the nurse call is delivered, the nurse could have better idea to decide if he should answer the call immediately.

In this paper, a context-aware based urgency cue system is proposed to provide urgency cues to hint nurses while they make the decision if they should answer the nurse call immediately. In the following chapters, chapter 2 describes the case in St. Olav's University Hospital in Trondheim where it has been equipped a wireless nurse call system. In chapter 3, several former works will be discussed. These works are related to case-based reasoning and context-aware system in the health care field. Chapter 4 describes the methodology while doing this project. In chapter 5, the context-aware based urgency cue system will be presented in details, including the system architecture, communication between different components and details of each component. Followed by chapter 5, chapter 6 tests and evaluates the system. In chapter 7, we discuss the benefits and disadvantages of this context-aware based urgency cue system. In the end, conclusion will be given in chapter 7.

Chapter 2 Background

2.1 Case

In St. Olav University Hospital, Trondheim, Norway, there are different departments. Each department contains a few nurses and patient rooms. The rooms are relatively small, so there is only one patient in each room. The whole hospital is covered by wireless network, and also has deployed nurse call system. Although all the departments in the hospital use the same nurse call system, each department has its own nurse call system instance, which is separated from the instances in other departments. Patients use the nurse call system to call nurses when they need help.

As described in [3], the nurse call system in St. Olav University Hospital contains: A telephony system (delivered by Cisco), a fixed nurse call system (delivered by BEST [4]), and a wireless nurse call system delivered by Imatis). The nurse call system overview is showed in figure 1 (The telephony system is omitted for simplification). This paper will discuss the Imatis part in figure 1, and details about other parts can be found in [3].

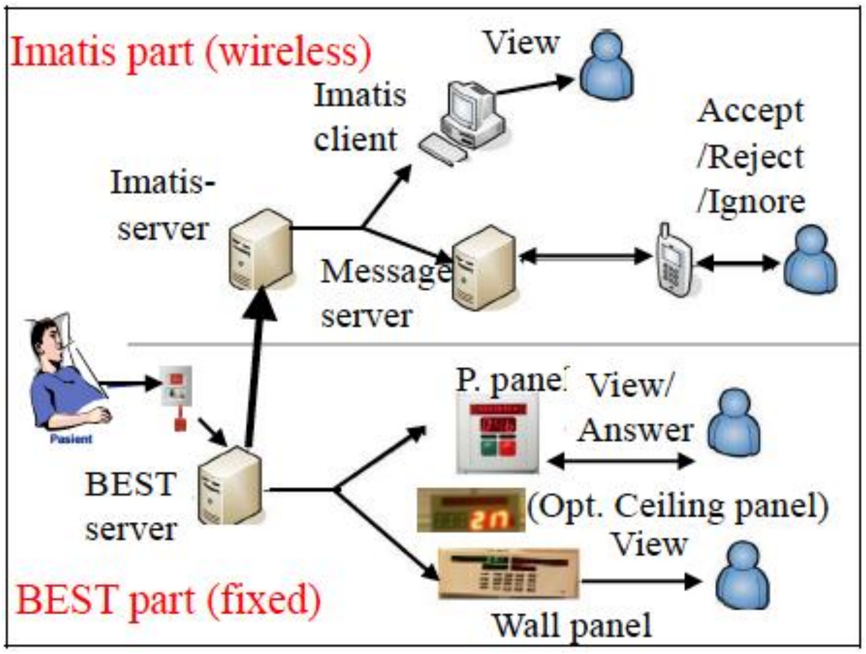


Figure 1 Overview of nurse call system in St. Olav Hospital [3]

When a patient presses the “Call” button in his room, a signal will be transmitted to the Imatis server. This server will further generate a request in order to generate a nurse call. This request will be sent to the message server. The message server receives the request, and then looks up the call plan in order to find the responsible nurse for the calling patient. The call plan stores the nurse-patient mapping. Usually a nurse is responsible for a few patients. After the message server found the responsible nurse, it will generate a nurse call to the specific nurse. When the nurse call is delivered to the nurse, the nurse can see who is calling on the nurse phone. A nurse phone looks like a normal mobile phone, as shown in figure 2. If the nurse rejects or ignore the call in 15 seconds, the call will be redelivered to another nurse.

Figure 2 Nurse phone in St. Olav Hospital

2.2 Problem

In the hospital, a nurse is responsible for more than one patient. Thus, when a nurse call is delivered to a nurse, the nurse is likely to be occupied by another patient. In this case, it is hard for the nurse to decide whether he should stop what he is doing and answer the call. Basically, the nurse’s decision is made based on the urgency of the calling patient and the on-treating patient. In other words, knowing the urgency situation of both patients is helpful for the nurse. Since the nurse is treating the on-treating patient, he knows for sure how emergent the on-treating patient is. On the other hand, there is no sign for him to know the situation of the calling patient. The calling patient could have a high fever or low heart beat, which means the patient is in danger. Or everything is fine with the calling patient, and he calls the nurse just because he wants some water.

For example:

1. Patient A has diabetes. When he calls the nurse, his blood pressure is 120, body temperature is 36, heart beat is 90 and is located in his patient room. Everything is fine for him, and he calls the nurse because he needs water.

2. Patient B has diabetes. When he calls the nurse, his blood pressure is 180, body temperature is 36.5, heart beat is 120 and he is located in his room. He fell on the floor, so he calls the nurse to ask for help.

3. Patient C has diabetes. When he calls the nurse, his blood pressure is 180, body temperature is 39, heart beat is 150 and he is in his room. He calls the nurse because he feels chest pain.

In the above examples, it is obvious that the first patient is not in an urgent situation at all. If the nurse is busy when he receives the call, he can finish what he is doing before checking this patient. In the second case, the patient is in a medium urgent situation. And in the last case, the patient is in a dangerous situation, and the nurse needs to response to the call as soon as possible.

Unfortunately, the nurse only knows which room the call is from, but do not know the calling patient's context. So it is difficult for the nurse to decide if he should response to the call immediately.

In order to solve the problem, a context-aware urgency cue system is proposed in this thesis. This system focuses on the architecture that collects context and dynamically provides urgency cue about a patient. Data sources are not the emphasis of this thesis. The system in this thesis does not implement any data source. It provides interfaces for different data sources so that they can publish new data. The system collects the data and maintains it. It can also provide urgency cue about a patient based on the data received from different data sources. In order to provide dynamically urgency cue, case-based reasoning is used.

Chapter 3 Related Work

This chapter discusses some former works in order to propose a solution for the problem in 2.2. In addition context-aware and case-based reasoning will also be introduced in this chapter.

3.1 Related Technologies

In order to understand the rest parts of the thesis better, it is necessary to introduce context-aware and case-based reasoning here. They are also core technologies for the system proposed in this thesis.

3.1.1 Context-aware

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."[5]

When human talks to human, it is easy to understand each other without explaining background, because human shares implicit situational information (context). This is based on human's common knowledge about the world we are living in. For example, when you are talking with your mother about your father, you do not have to explain why he would react to something in some particular way, because both of you knows his personality well.

Unfortunately, when human interact with computer, this common knowledge does not exist. Thus, human needs to input extra information so that computer can understand and provide further services. In order to make computer provide better services, we can try to make computer know context automatically, i.e. context-aware.

"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."[5]

A system that wants to be context-aware, it must have the ability to capture the state of the environment and collect information about the surroundings that is related to the human computer interactions. The awareness is often achieved by collecting data from sensors, devices, other applications and user predefining [6]. However, these data sources provide various data types, and the data is diverse in practice. Thus, context modeling is important in context-aware. Context-aware system needs a formal form to share and interpret context.

In [7], the author suggests the following way to model context:

Key-value pair context models

This is the most simple approach to represent context. RFID (Radio Frequency, Identification) [8] tag is a typical usage of this approach. A RFID tag has a unique ID that can be read by a RFID reader, and the RFID often contains a simple message. This approach is more common for items with limited memory.

Markup scheme context models

Markup scheme can express more complex information and typically has a fixed structure. This model type is often serialized in XML (Extensible Markup Language) or RDF (Resource Description Framework). This approach is more often used to capture static information, such as user agent profile, device profile, user profile and configuration.

Ontology-based context models

Ontologies provide a specification of a conceptualization [9]. And also ontologies are well known to model concepts and the relationships that hold among them [7]. The Semantic Web has published specifications and tools about using ontologies to model context. Because ontologies have explicit support for semantic reasoning, they are used frequently to model context [7]. Such examples can be found in [10, 11].

Table 1 Comparative analysis of context models [7]

	Key-value Pairs	Markup Scheme	Ontology Based
Expressiveness	*	**	***
Efficiency	***	**	*
Programming	***	**	**
Reasoning	*	*	***

Ambiguity	*	*	***

Comparison about the previous context models is presented in table 1. The stars mean score (from 1 to 3), the more, the better. From table 1, it is easy to conclude that ontology-based context model can express most complex information and is best for reasoning. However, it has biggest overhead. Markup scheme context model can express relatively complicated information while it has medium overhead, and markup scheme does not fit reasoning scenarios. Key-value context model is most efficient and has smallest overhead, but it has limited ability to express information and deal reasoning.

3.1.2 Case-based Reasoning

According to [24], when human wants to resolve a problem, we first try to apply previous experience, and then adjust to the current situation in order to come out a solution for the new problem. Case-based reasoning is inspired by it.

As described in [12], case-based reasoning approaches the reasoning process by using old memory, instead of using rules or general knowledge. Knowledge in a case-based reasoning system always contains plenty of old experience (cases) and solution to each case. Case-based reasoning finds out the solution to a new problem by reusing the solution of a similar previous case.

As shown in figure 3, case-based reasoning solves a problem by going through a cycle. It contains four steps:

1. **Retrieve:**

   When a case-based system faces a new problem, it tries to retrieve the most similar case(s) by matching the previous cases with the new case.

2. **Reuse:**

   If the system finds suitable previous case that is similar to the new case, the solution of the previous case will be reused and proposed to the user.

3. **Revise:**

   The user may revise the selected old case with new solution if he is not satisfied with the old solution.

4. Retain:

In the end, the user could retain the revised old case and the new case into the case base. This is the learning process.

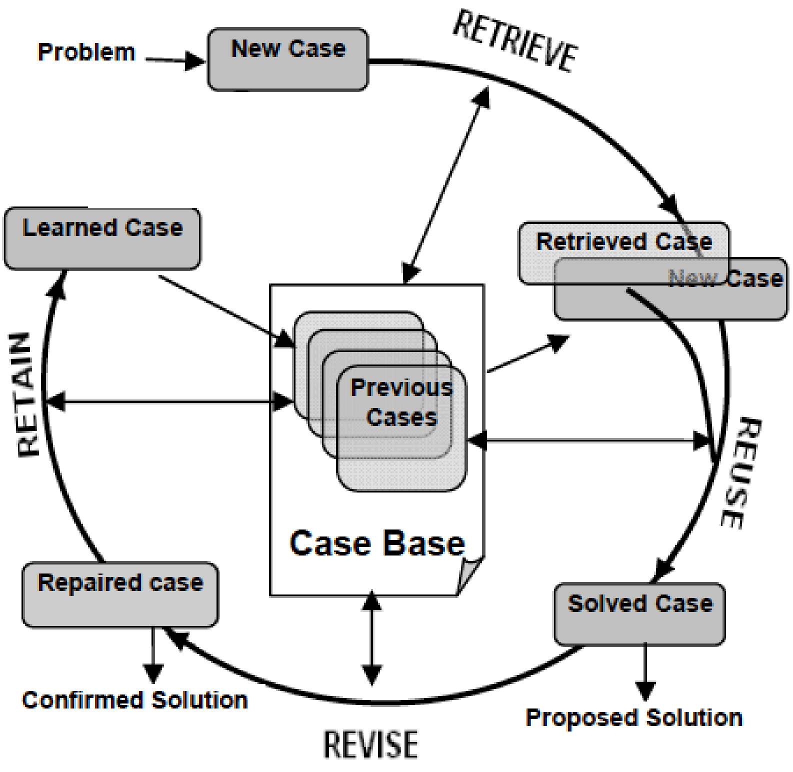


Figure 3 CBR cycle [12]

The challenge of CBR system is the design of similarity computation. How the similarities are computed is a very important factor that affects the performance of a CBR system. And the similarity computation is diverse in different use scenarios. In addition, academic knowledge is also quite important while design the similarity computation. Normally, it is the computer scientists who implement the CBR systems. However, the computer scientists have limited knowledge in the system's use scenario, such as hospital. Thus, a CBR system should be flexible and provide a user-friendly mechanism to let the user define similarity computation function.

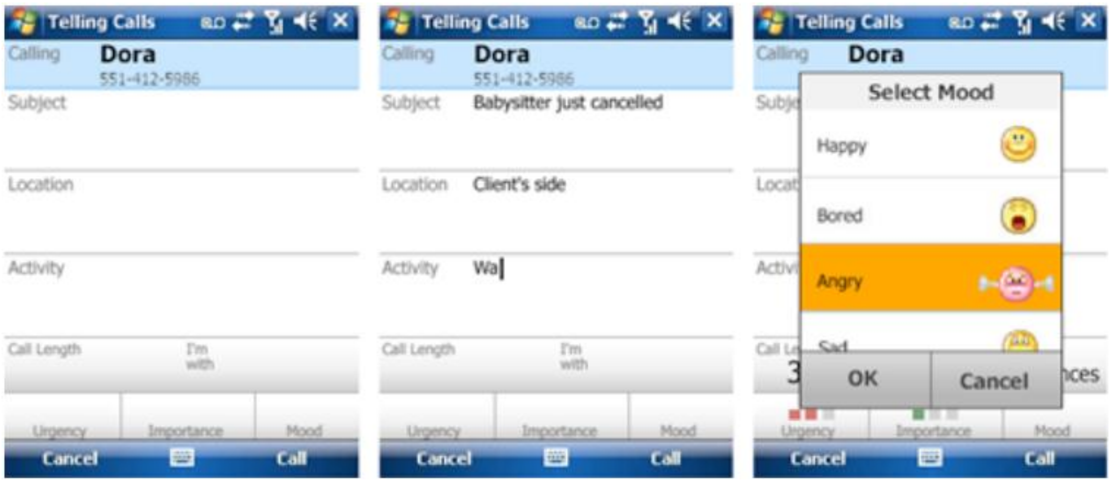
3.2 Former Work

3.2.1 Calling with Text

In [14], the authors mentioned that when a phone call is coming, the receiver is facing the task to make the decision whether to answer or reject it. Normally, the decision is made by knowing limited information (caller name). In this situation, the receiver could think the call is important and answer it, but actually this call is not important. Sometimes this does not affect the receiver. However, if the receiver is doing something else, he might have been interrupted by the call and wasted time to answer it. This could be harmful. On the other hand, the receiver could also think the call is not important and therefor ignore it. In this situation, it could be harmful when the caller really needs help, and this is an emergent call.

Thus, the authors in [14] proposed to use context to help the receiver to decide whether he answers the call. They suggest that additional information could be used to provide hints to the receiver. In addition, a mobile application was developed to evaluate the concept.

The mobile application's name is telling calls. As shown in figure 4, the caller needs to fill up the blanks before he makes the call. On the other side, the receiver will receive not only the call, but also the information that has been sent from the caller, As shown in figure 5.



Subject: Babysitter just cancelled
Location: Client's side
Activity: Wa

Figure 4 Caller interface of telling calls [14]

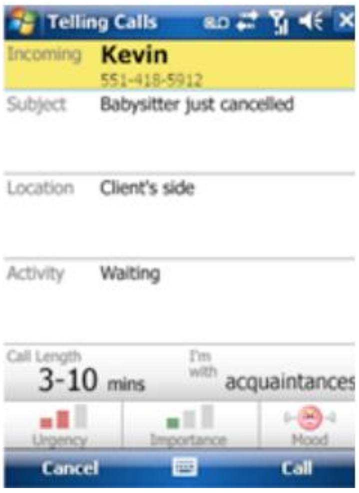


Figure 5 Receiver interface of telling calls [14]

[15] had investigation about factors that affect receiver's decision to answer the call. It found out that call reason, caller's estimated call length, activity of the caller, location of the caller, caller's mood, the urgency and importance of the call, people the caller is with are the factors the receiver highly expected when he receives a call. Based on the investigation, the authors in [14] decided to use the information in figure 4 to help the receiver to decide whether he should answer the call.

After the application was implemented, en evaluation was made as well. The authors asked 37 individuals to use the application with their friends, lover or families. The 37 users were interviewed after they had used the application for 1 to 2 weeks. The 37 users included adults and students, and all of them had used mobile phone more then 3 years.

As the result, most participants liked the concept that they got extra information when they received the calls. They thought this was helpful when they made decision to answer it or not. The following are two participants experience about telling calls.

"I know her very well. I could almost deduce everything from the location and activity. I knew if she was in the supermarket shopping that – this would be a quick call and that it was probably something important." --Anonymous user in [14]

"The other day, she was waiting for me to pick her up. She gave me a call. I didn't have a hands free <set> on and I didn't want to pick the call up for the fear of getting a ticket. And I saw all the information <on Telling Calls> - she was finished at the gym and was waiting at so and so location. So, I knew what it was about and I was heading there, so <I thought> it's okay if I let this call go."--Anonymous user in [14]

On the other side, [14] also mentioned that the participants preferred to use the application to answer calls other than to make the calls. They thought it was annoying that they had to type various information before they made the calls.

Back to our issue in 2.2, the nurses as call receivers might like to have additional information when they receive nurse calls, just like the receivers in this research. Inspired by this research, we could also add information with the nurse call and present them together on the nurse phone. And a nurse could use the information to know how the calling patient is, if he should answer the call immediately.

However, the way to provide information in this research is unacceptable in hospital scenario. As we can see in this research, normal phone users do not like to type a bunch of information before they make calls. In hospital, patients can be in various situations. Therefor, it is not practical to ask them to type things before they call.

Thus, we could use the idea to send extra information while the call is made, but the patients should be able to call the nurses without doing any pre-work.

3.2.2 Context-aware system in Heath Care Environment

In [16], the authors concerned that the current nurse call systems in the hospitals are place-oriented, which means that the systems are very static. However, the nurse call system could be improved to be person-oriented. In this way, the system could be context-aware.

In study [17], they tested two nurse call systems in a hospital. As in the left part of figure 6, it is the traditional place-oriented nurse call system. Call buttons only exist in the patient rooms, so when a patient wants to call a nurse, he has to be in his room. And when the call is generated, the controller will decide which nurse the call should go to according to some algorithm. However, it is possible that a patient needs help when he is not in his room, for example, in the hall or in other patient’s room. If the patient is in the hall, he could not even be able to make the call. If the patient is in other patient’s room, the wrong nurse could receive the call because this nurse is assigned to the room. And these situations could be harmful.

On the other hand, in the right part of figure 6, the person-oriented nurse call system is different. In this system, the call buttons are not longer fixed in the rooms. They are portable and assigned to specific patients. Each patient has a portable call button (typically a mobile phone). In this case, patients can carry their call buttons moving around. Whenever they press the call button, the system will know who is calling, and deliver the call to the right nurse.

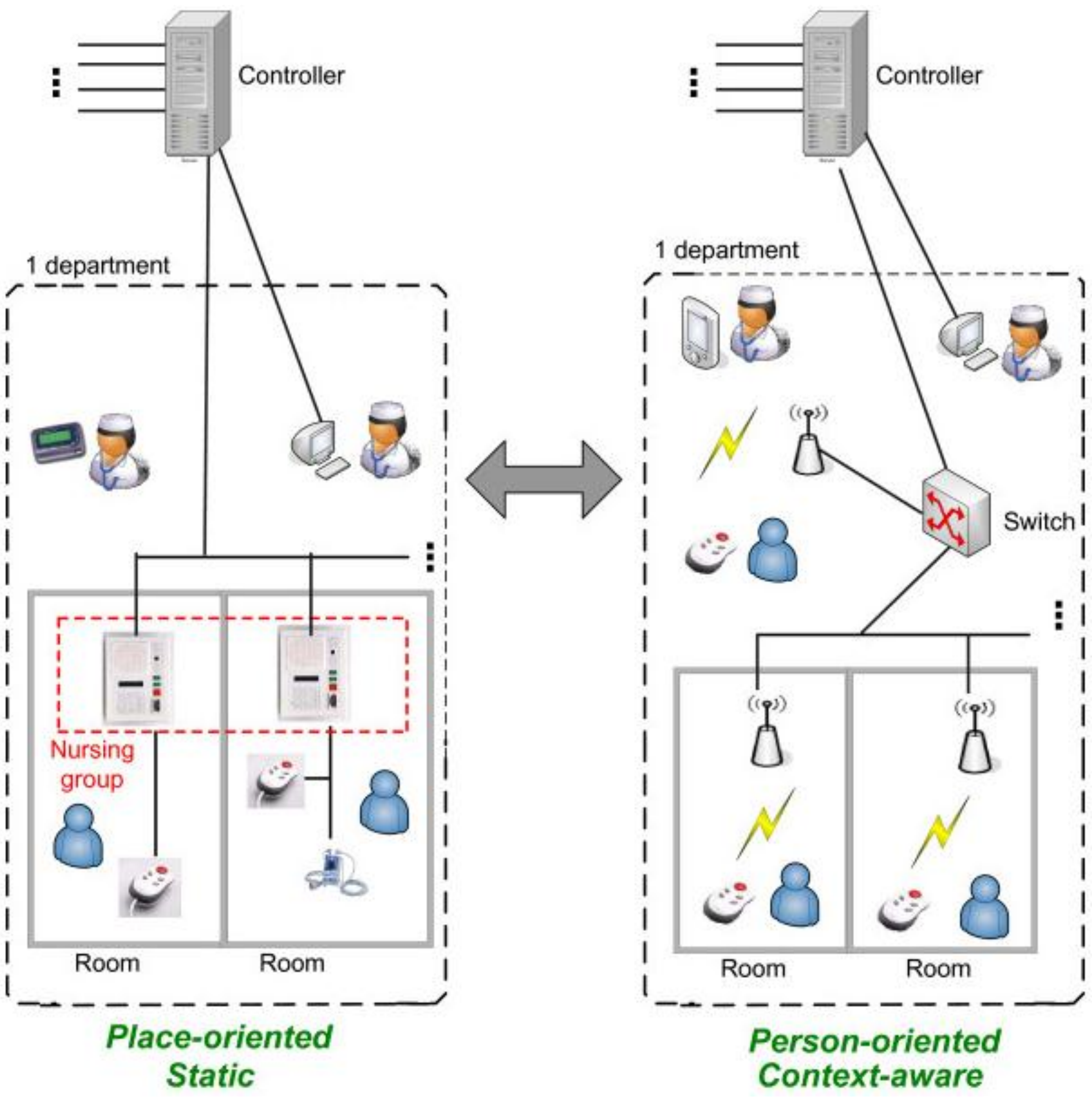


Figure 6 Comparison of two nurse call systems [16]

According to [17], 80% patients preferred to use the person-oriented nurse call system, because they can make the nurse call wherever they are.

In addition, [16] also suggested that the nurse call system should be “smarter”. According to [18, 19], there are already various systems in hospital scenario. These systems manage staff tasks, patient data, EPR (Electronic Patient Record) and so on. And these systems provide various data about he patients and health care staff. However, these systems work separately. It is a waste that the data is used separately [20]. So [16] proposed to use context-aware technology to exploit the medical context in the nurse call system. Further more, the authors of [16] developed a new nurse call system to evaluate their concept.

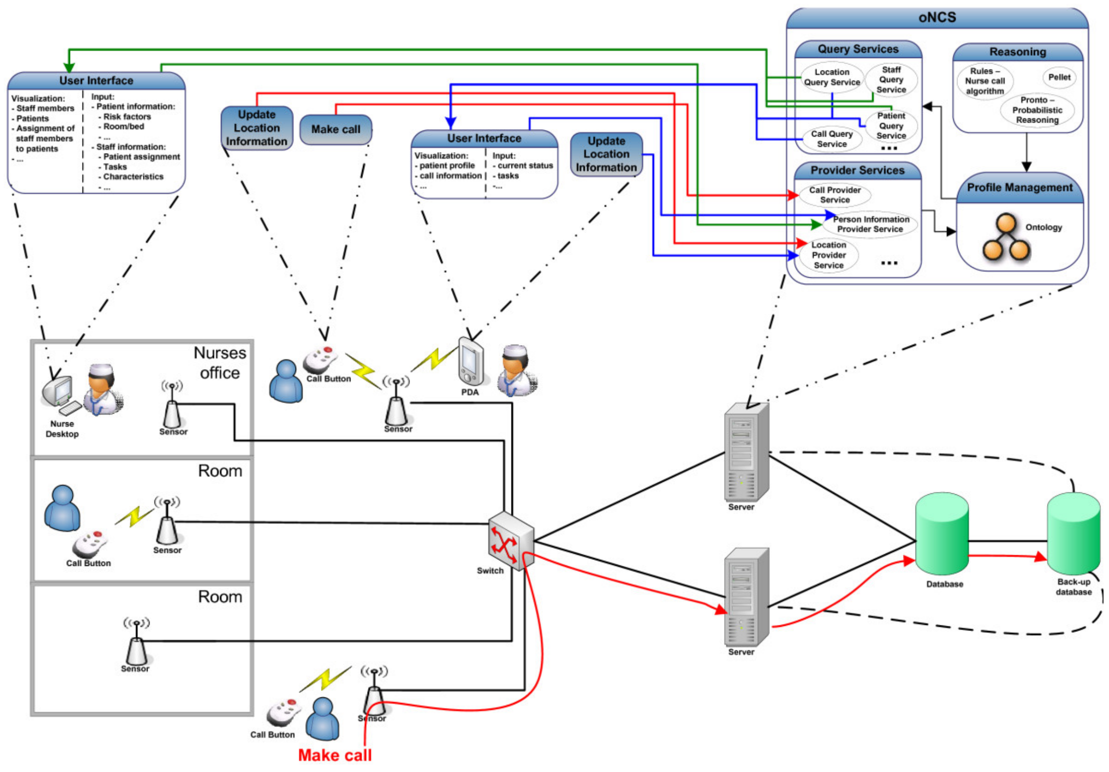


Figure 7 General concept of the oNCS platform with probabilistic risk assessment and profile management [16]

The system built in [16] is oNCS (ontology-based Nurse Call System). Compared to traditional nurse call system, oNCS contains a profile management component. As shown in figure 7, this component manages context about patients and health care staff. Context includes location information, patient profile, assignment of staff members to patients, staff task, patient risk factors and so on. And the context is collected through various devices and servers, such as location badge, EPR server. Based on the profile management, oNCS provides (1) “smart” nurse call assignment and (2) priority of nurse call.

“Smart” nurse call assignment aims to find the correct staff member to handle a call. When a call request is received, the system uses information stored in the ontology to determine the receiver of the call. The information includes location of nurses, availability of nurses and so on. Details about this algorithm can be found in [16].

Priority of a call in oNCS is determined using probabilistic reasoning algorithm, and it was concluded by Pronto [21]. The details can be found in [16].

In the person-oriented context-aware nurse call system, when a patient presses his own call button, the system will notice which patient is calling. And then the system can find all context related to this patient. Based on the patient's information and other relevant information, the system can not only deliver the call to a right nurse but also provide the probabilistic priority of this call.

The evaluation of the system was done with realistic simulations with data collected from the Ghent University Hospital. And the results showed that the oNCS system improved the nurse assignment. The priorities of the calls were also dynamically determined according to the current situation of the patients. At the meantime, the execution time is negligible. Unfortunately, the authors did not evaluate this system in a practical environment.

Context-aware technology can be used to provide patient's current context, such as patient's location, diagnosis, heart beat, body temperature. This could help the nurse to determine the urgency of the patient. Thus, it can be used to resolve the problem in 2.2.

The authors mentioned in [22] that the systems in hospital should not decide too much, because cases in hospital are often live-related. If the systems decide too much instead of health care staff themselves, it could be harmful to the patients. Thus, the systems in the hospital should provide advices to health care staff, but not decide for them. This is the shortage of the system in [16].

In addition, the situation and environment changes in the hospital. The system in [16] cannot adapt itself automatically according to the new environment, because its strategy is static. Thus, we need a system that is more flexible and is able to "learn" the new environment by itself.

3.2.3 Case-based Reasoning in Health Care Environment

It is risky to live with severe stress in a long period. This could cause disease such as heart attack and high blood pressure [23]. According to [23], it is identified by medical investigation that FT (Finger Temperature) has a strong correlation with human's stress status. In practice, clinicians always need to do a lot of measurements in order to capture the patients' symptom. After this, the clinicians need to understand the measurements. Considering the large variation of measurements from diverse patients, it requires knowledge and experience for the clinicians to determine the patients' situation correctly. However, there are many clinicians who are lack of experience. So they could misjudge the patients' situation, which could be harmful.

Thus, the authors of [23] proposed a case-based decision support system to help the clinicians to determine the patients' situation. Diagnosing is given out based on doctor's knowledge and experience, which comes from previous study. And this process is similar to the concept of case-based reasoning. That is the reason why the authors chose to use case-based reasoning to solve the problem.

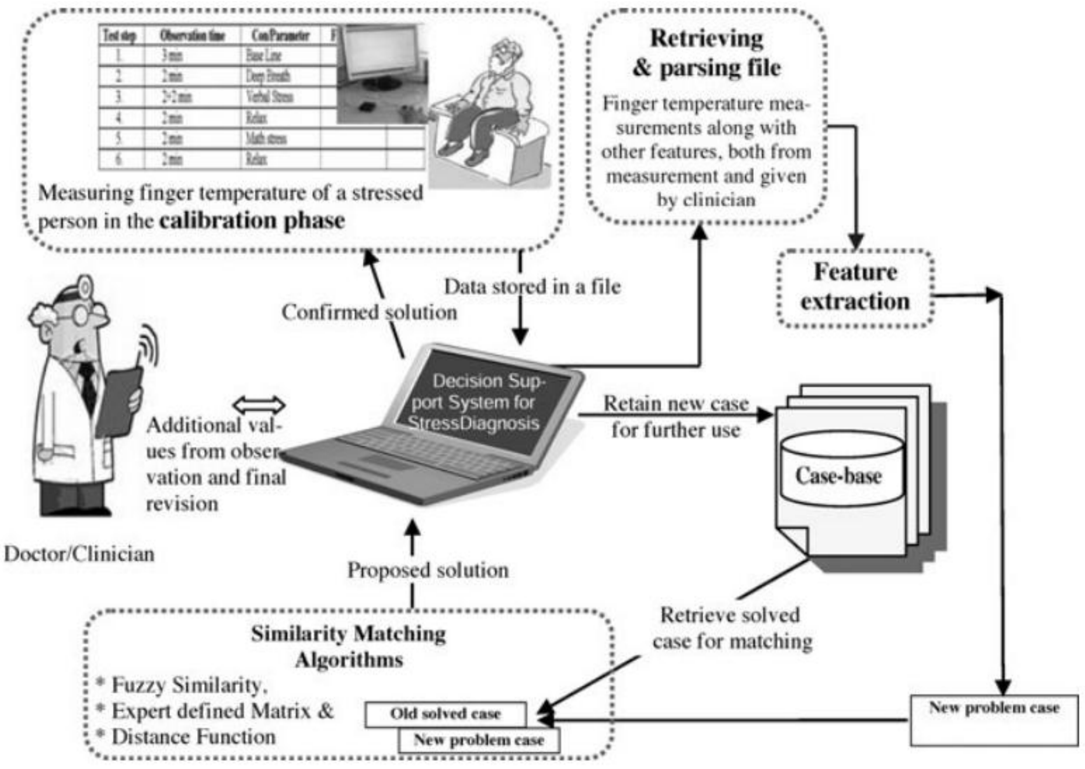

Test step | Observation time | ConParameter
1 | 3min | Base Line
2 | 2min | Deep Breath
3 | 2+2min | Verbal Stress
4 | 2min | Relax
5 | 2min | Math stress
6 | 2min | Relax
Measuring finger temperature of a stressed person in the calibration phase
Retrieving & parsing file
Finger temperature measurements along with other features, both from measurement and given by clinician
Feature extraction
Data stored in a file
Confirmed solution
Decision Support System for StressDiagnosis
Retain new case for further use
Case-base
Additional values from observation and final revision
Doctor/Clinician
Proposed solution
Retrieve solved case for matching
Similarity Matching Algorithms
* Fuzzy Similarity,
* Expert defined Matrix &
* Distance Function
Old solved case
New problem case
New problem case

Figure 8 System overview of case-based decision support system for the stress diagnosis [23]

As shown in figure 8, the case-based decision support system works in the following steps:

1. A patient needs to take FT measurements in order to setup his stress profile, and the measurements can be done by using temperature sensor.

2. After the patient's stress profile is stored, the system will extract relevant features.

3. Based on the features, a new problem case will be formulated.

4. The system compares the new problem case with old problem cases, and selects the most similar old problem case as best matching for this new problem.

5. The clinician can then revise the best matching case and approve it to solve the new problem. The clinician may need to adjust the solution to the old problem case when he uses it to solve the new problem case, because the new problem case is not always as the same as the old ones.

6. After the new problem case is solved, this case will be stored in the case base in order to solve a future problem. This is the learning process in CBR (Case-Based Reasoning).

Evaluation has been done to this case-based decision support system. As in [24], the authors compared the diagnostic performance between this system and clinicians. The clinicians had different experience levels, including expert, senior and junior clinicians. The result showed that the system could correctly classify more 80% of cases. That number for junior clinicians is between 57% and 69%. For a senior clinician, the correct rate is 73%.

Case-based reasoning suits health care environment, and this has been identified in [25]. It can also be used to solve the problem mentioned in 2.2. Because in hospital, a patient usually has a specific responsible nurse, and this nurse knows the patient's situation best. When the nurse is away, another nurse will temporarily take over the responsibility for the patient, and the new nurse does not know the patient's situation well. In this case, it could be useful that a system can provide some urgency cues based on previous experience.

3.2.4 Summary

Through section 3.2.1, 3.2.2 and 3.2.3, we can use the concepts in the three works. To solve the problem that nurse does not have clues to decide whether he should answer the nurse call when he is busy with another patient. The concept in 3.2.1 can be used. We can add extra information about the calling patient sending with the nurse call to the nurse, so the nurse could know how urgent the patient's situation is.

Instead of asking the caller to type in information as in 3.2.1, the concept in 3.2.2 could be used. We can use context-aware technology to provide patient information, including patient's diagnosis, location, heart-beat, body temperature and so on. In this case, the system can know patients' context. When a patient calls a nurse, the system can quickly give urgency clues to the nurse based on the patient's context. Since the patients' context includes a lot of information, it is not practical to send all the information to nurses. Otherwise it will be hard for the nurse to know the situation quickly. So the system should only give short urgency cues instead of long and complex information. Further more, the concept person-oriented nurse call is needed, since the system needs to know who the caller is.

In addition, case-based reasoning can also be used to improve the accuracy of the urgency cues. As mentioned in 3.2.3, case-based reasoning fits health care environment. We can let the system remember the old cases, which includes the current patient context and the urgency cues. When a patient tries to call the nurse, the system can compare the patient's current context with old cases, and find out the most similar one. The urgency cues of the most similar old case will be sent to the nurse as well as the nurse call.

Chapter 4 Methodology

4.1 Design Science

Design science is applied though the whole project. Design science provides research rigor, doing project in this way makes the outcome more acceptable by the scientific community.

In order to make the output of this project more valuable and more acceptable by the scientific community, the project is processed following the guidelines, which are mentioned in [26]:

- ✓ **"Design as an artifact: Design-science research must product a viable artifact in the form of a construct, a model, a method, or an instantiation." [26]**

  In this project, a runnable context-aware urgency cue system is built in order to prove the concept.

- ✓ **"Problem relevance: The objective of design-science research is to develop technology-based solutions to important and relevant business problems." [26]**

  Firstly, the case in St. Olav University Hospital is literally studied and the problem that the nurse call is lack of urgency cues is explained. Further, related former works are studied in order to find a solution for this problem. All the works are based on the problem.

- ✓ **"Design evaluation: The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation." [26]**

  In order to evaluate the outcome of this project, tests are done after developing the system.

- ✓ **"Communication of Research: Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences." [26]**

This report as part of the output of the project presents the whole concept and important details of this project. Both technology-oriented and management-oriented audiences can learn something from this report.

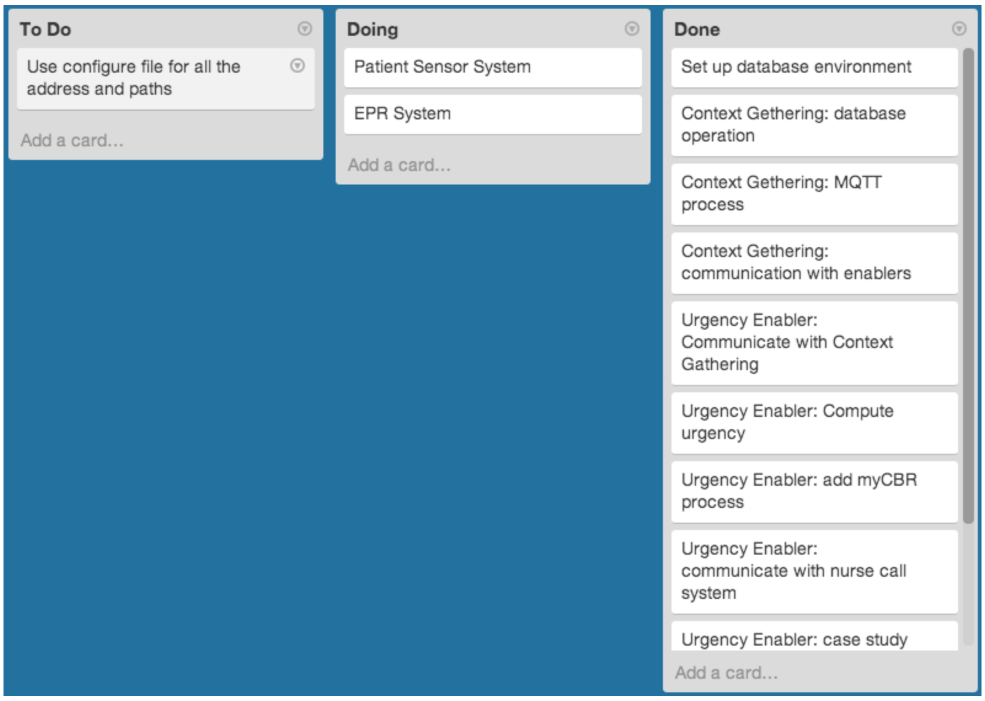
The entire project is processed following the design science guidelines. Firstly, a case in St. Olav University Hospital is studied and an issue has been focused. The issue is that the nurse call in the hospital does not have urgency cues, so it is hard for the nurse who receives the call to decide which patient is more urgent, the calling one or the on-treating one (assuming that the nurse is treating a patient when he receives a nurse call from another patient). Secondly, several former works has been studied in order to find useful technologies to resolve the problem. Thirdly, a context-aware urgency cue system is built in order to resolve the problem. After that, tests and evaluation has been done to evaluate the solution. In the end, the report is proposed to present the outcome of this project.

During the development of the system, Agile development was used.

4.2 Agile Development

In order to make sure the development is flexible, Agile software development is used. Agile development is different from traditional waterfall development. According to [27], Agile uses short iterations. The reasons to use short iterations are: increasing feedback opportunities and make course corrections. Developing with Agile, developers have more small demos to present to the customers after short periods. As in return, the customers can give feedback after the demos, so the developers can react and modify the system according to the feedback immediately. In this way, the final system can fit the customers' requirement better.

Runnable is an important factor in Agile development. The project is divided into small tasks, and all the tasks come out with runnable demos. After all the tasks, the whole system should be close to the final produce. In this project, Trello [28] was used to manage the tasks.


To Do

Use configure file for all the address and paths

Add a card...

Doing

Patient Sensor System

EPR System

Add a card...

Done

Set up database environment

Context Gethering: database operation

Context Gethering: MQTT process

Context Gethering: communication with enablers

Urgency Enabler: Communicate with Context Gathering

Urgency Enabler: Compute urgency

Urgency Enabler: add myCBR process

Urgency Enabler: communicate with nurse call system

Urgency Enabler: case study

Add a card...

Figure 9 Task management

As shown in figure 9, There are 3 columns to present the states for the tasks: To Do, Doing and Done. In the beginning, all the tasks are in To Do column, because all of them need to be done. The on-going task will be move to Doing column, and there should not be so many tasks in this column, because it is better to focus on one or two tasks once. After the on-going task is achieved, it will be moved to Done column. Trello is very helpful in Agile development, and it can also be used in group development.

During the development, the codes are managed on GitHub [29]. GitHub is a useful code management tool. It provides code storage, and also version control. If one is not satisfied with the newly modification to the codes, GitHub provides rollback. So the codes can be rollback to the last version.

Chapter 5 Design and Implementation of the Context-aware urgency cue system

In order to solve the issues discussed in [30], Telematics department at NTNU is developing a new nurse call system prototype to research and resolve the problem regarding the current nurse call system in St. Olav University Hospital. This new nurse call system has similar architecture as Imats part in figure 1. In this new nurse call system, Android-based smart phones are used as nurse phones. Although smart phones are not used in the current nurse call system in this hospital, it is reasonable to assume that they will be used in the future.

In this thesis, a context-aware urgency cue system is built in order to solve the problem described in 2.2. This system will provide urgency cues to the new nurse call system so it can be sent with the nurse call to the nurse.

The context-aware urgency cue system is inspired by the former works introduced in chapter 3. This system is based on context-aware and case-based reasoning. It has a context-gathering server to collect data from diverse sources, such as sensors, profiles and other systems. In addition, this system also includes an urgency server that approaches case-based reasoning. The system details will be described in 5.2.

This chapter introduces the context-aware urgency cue system that is the main work of this thesis, including system main functionality, system architecture, protocol and two main components in the system. The system is implemented in Java.

5.1 System Functionality

As shown in figure 10 is the use cases figure. The system provides three main functions to outer instances:

1. Diverse data sources should be able to update context to the system.

2. Nurse call system should be able to get an urgency cue from the system.

3. Nurse call system should be able to give feedback about an urgency cue, and this

feedback should be learned as a new case by the system.

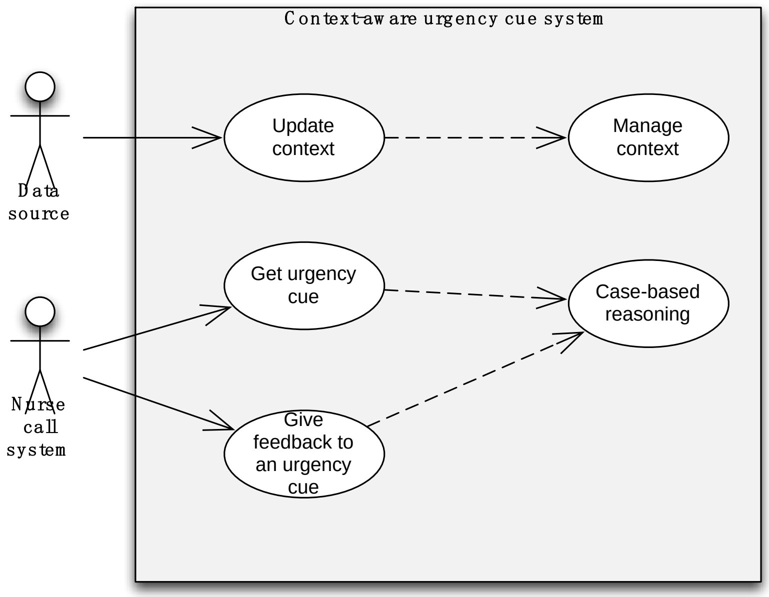


Figure 10 User cases figure

According to figure 10, the system has two main functions: (1) manage context and (2) approach case-based reasoning.

For the first function, the system should be able to get the latest data from every data source and keep the data consistent and formal in the system.

For the second function, the system should be able to use the data from function (1), and dynamically match the new case with previous cases in order to find the most similar case to get urgency cue for the new case. In addition, the system should also be able to accept the nurse's feedback towards a case, and retain the case with the feedback into the case base.

An example will be introduced bellow in order to describe the system.

There are three data sources: location system, patient sensor and EPR system. Location system can provide which room the patient is in. Patient sensor can provide patients' blood pressure, body temperature and heart beat. EPR system can provide patients' diagnosis. All the three data sources are able to publish new status about a patient to the context-aware urgency cue system, and the system maintains the latest patients' context.

Three previous cases have already been stored in the context-aware urgency cue system. They are:

1. In case A, the patient had diabetes. When he called the nurse, his blood pressure was 120, body temperature was 36, heart beat was 90 and was located in his patient room. And urgency number for this patient was 1.

2. In case B, the patient had diabetes. When he called the nurse, his blood pressure was 180, body temperature was 36.5, heart beat was 120 and he was located in his room. And urgency number for this patient was 5.

3. In case C, the patient had diabetes. When he called the nurse, his blood pressure was 180, body temperature was 39, heart beat was 150 and he was in his room. And urgency number for this patient was 10.

There is another patient, and he has diabetes. When he calls the nurse, his blood pressure is 110, body temperature is 36, heart beat is 89 and is located in his patient room.

When the patient calls the nurse, the nurse call system will send a request to the context-aware urgency cue system. This system will compare the patent's context, which has been collected from the data sources before, to the previous cases. After comparison, the system will pick case A as the most similar case and return the urgency number in case A, which is 1, to the nurse call system.

The nurse call system delivers the urgency number 1 along with the nurse call to the nurse. After the nurse checked the patient and assessed the urgency by himself, he can assign a new urgency number (for example 2) as feedback to the call. This feedback will be delivered to the context-aware urgency cue system through the nurse call system. After this, the context-aware urgency cue system will store this case with the urgency number (in this case, 2) as previous case for future use.

5.2 System Architecture

Figure 11 shows the system architecture. The context-aware urgency cue system needs to cooperate with other systems in order to provide fully functionality. This system provides interfaces to nurse call system so that nurse call system could get urgency cue and give feedback towards an urgency cue. On the other side, this system also provides interfaces to data sources, such as location sensors, patient sensors and EPR (Electronic Patient Record) system. These data sources can use the interfaces to update corresponding data.

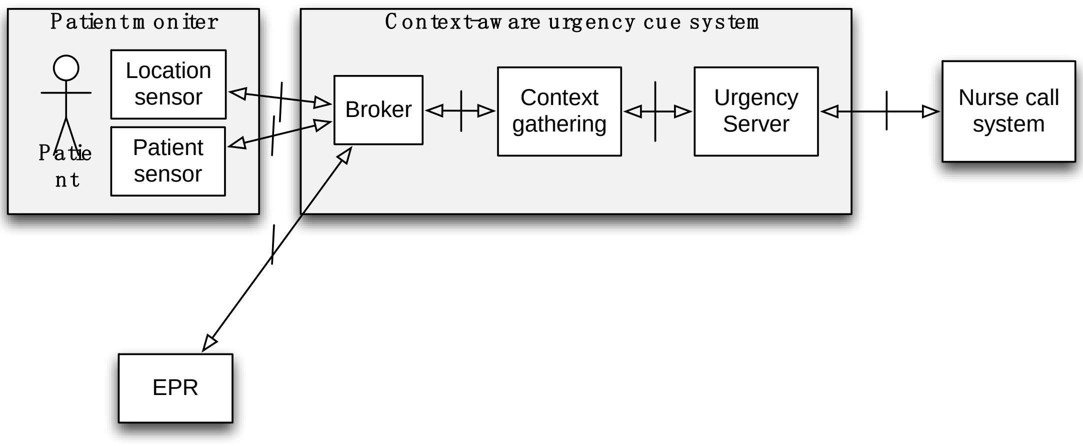


Figure 11 System architecture

As shown in figure 11, there are three components in this context-aware urgency cue system: broker, context gathering server and urgency server.

In order to keep the context-aware urgency cue system having the latest status about patients, a publish/subscribe pattern is used. The broker in figure 11 as an intermediary message broker, receives published new status from data sources (For example, “Mike has_location patient_room_1” from location sensor) and sends subscribed data to the context gathering server. In this thesis, Moquitto [35] is chosen as the broker.

Context gathering server collects data from diverse data sources and manages the context. It also provides context to urgency server based on urgency server’s request.

Urgency server waits for nurse call system’s request, and ask for relevant context from context gathering server according to the request. After gotten relevant context, urgency server retrieves the most similar case in the case base compared to the relevant context gotten from context gathering server, and then send the most similar case’s urgency solution to the nurse call system as response. In addition, the urgency server also accepts feedback from nurse call system in order to learn a new case. The detailed process will be described in section 5.6

In this thesis, the context-aware urgency cue system is proposed to provide architecture in order to collect the context about patients and provide an urgency number about a patient based on the patient’s context. Only the context-aware urgency cue system is implemented in this thesis. This system aims to provide flexible architecture so that more data sources can be included. Although there is only three data sources are included in figure 11, more data sources can be added as long as they follow the protocol described in 5.4

5.3 Context Model

As discussed in 3.1.1, context-aware system needs a model to share and interpret the context. According to the comparison in 3.1.1, ontology-based context model is chosen for this project, because ontology-based context model is most expressive. Since the data sources in hospital are diversity, we need expressive model to cover all the data types.

W3C (World Wide Web Consortium) Web Ontology Working Group proposed ontology language OWL (Web Ontology Language) to express ontology-based context model [32]. OWL is a very expressive language. Therefore it is very complicated, including several specifications and pre-knowledge, such as RDF scheme. Since the time to accomplish this thesis is limited, only the basic model of OWL is used in this project. According to [33], The basic model of OWL is:

Tuple (subject, predicate, object)

The above model is used in this project. Context is presented in this form. Each tuple presents a statement. For example: **(Anna, has location, operation room 1)**. In this example, “Anna” is the subject, “is located in” is the predicate and “operation room 1” is the object.

5.4 Communication protocol

As shown in figure 11, there are both external communication and internal communication for the context-aware urgency cue system. The internal communication are between the urgency server and the context gathering server, and between context gathering server and broker. This system also has external communication with both nurse call system and data sources.

5.4.1 External Communication

The context-aware urgency cue system needs to communicate with the nurse call system and diverse data sources. This section will discuss the communication protocols used in the communications.

Figure 12 shows the context-aware urgency cue system’s external interfaces. The system has three external interfaces: (1) cnu, (2) cnf and (3) cd.

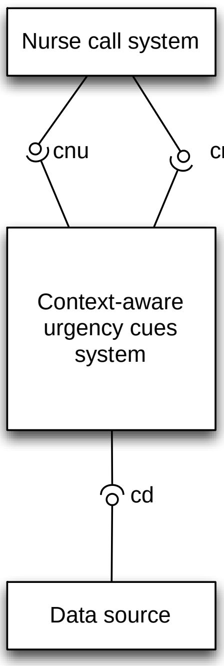


Figure 12 External interfaces

Interface cnu and cnf:

They are the interfaces between the context-aware urgency cue system and the nurse call system. Request-response protocol is used in these interfaces. Since there is not so much data need to be transmitted in these two interfaces.

As discussed in 3.2.4, it is not practical to send all the context about a patient to a nurse, since it is hard for the nurse to read a lot of information in a short time. Thus, the context-aware urgency cue system will provide an urgency number based on a patient’s context. The urgency number indicates an emergency degree. From 1 to 10, 1 is not emergent at all, and 10 is extremely emergent. However, it is possible to add a few more information about the patient in the future, such as heart beat and body temperature. Therefore, the messages in these interfaces are presented in XML scheme in order to be extensible.

Interface **cnu** is used for the nurse call system to send request to ask for urgency cues, and the request and response should be in the following form. The nurse call system must send the urgency request as the following request form, and the context-aware urgency cue system will send the response in one of the responses’ form.

```
<request>
    <callId>XXXXX</callId>          //the nurse call's idtification
    <patient>XXX</patient>          //patient's identification
</request>
```

```
<response status="200">              //status code, 200 means success
    <urgency>XX</urgency>            //urgency number
</response>
----------------------------------------------------------------------------------------------------
<response status="XXX"> //status code, XXX is as the same as HTTP status
code
    <message>XXX</message>           //reason for failure
</response>
```

Interface **cnf** is used for the nurse call system to send request to give feedback towards an urgency cue, and the request and response should be in the following form. The nurse call system must send the feedback request as the following request form, and the context-aware urgency cue system will send the response in one of the responses' form.

```
<request>
    <callId>XXXX</callId>            //the nurse call's idtification
    <urgency>XXX</urgency>           //new urgency number
</request>
```

```
<response status="200">              //status code, 200 means success
</response>
----------------------------------------------------------------------------------------------------
<response status="XXX"> //status code, XXX is as the same as HTTP status
code
    <message>XXX</message>           //reason for failure
</response>
```

Interface cd:

This interface is used by the data sources to update their new information. Considering that some of the data sources do not have big bandwidth or power, the communication protocol in this interface is designed to be as simple as possible. Thus, MQTT (MQ Telemetry Transport) is used in this interface. MQTT is a lightweight broker-based publish/subscribe messaging protocol [34]. It contains only 2 bytes header, which makes this protocol lightweight and less overhead.

In addition, in order to reduce the delay, a publish/subscribe pattern is reasonable in this interface. If the system needs to collect the data from diverse data sources every time when the nurse call system sends an urgency request, the system performance will be affected due to various communications must be accomplished. However, if the data sources update their information whenever they have changed data, the system can collect the context before it receives an urgency request. This will make the system response to the request faster. Thus, MQTT is a proper choice in this interface.

MQTT is broker-based protocol. The context-aware urgency cue system needs to subscribe data sources to the broker. And all the data sources will send their new data to the broker. Whenever the broker receives a new publish, it will send a message to the context-aware urgency cue system if the system have subscribed data from this data source.

In this interface, the data sources need to publish new information as a statement in order to follow the context model described in 5.3. A data source can only publish one statement at once, and the statement must be in this form:

Subject Predicate Object

A statement contains three elements: subject, predicate and object. They are separated by space. To make sure there is no ambiguity, there must not have space in each element.

5.4.2 Internal Communication

There is one interface between the urgency server and the context gathering server. Urgency server uses this interface to get current context of a patient. In order to make the protocol extendable, XML is used to present messages in this interface. And the communication is a request-response pattern.

The following is the message pattern used in this interface. The urgency server sends the request to ask for context of one or more subject. If there is predicate node under an item node, it means it only asks for the listed predicates of this subject. Otherwise it asks all the context of a subject. The context gathering server sends one of the responses to the urgency server.

```
<request>
        <item subject="Anna">
       //if an item has no predicate, it means that it asks all context about the
subject, otherwise means only ask for the listed predicates
                <predicate>has_location</predicate>
                <predicate>has_diagnosis</predicate>
       </item>
        <item subject="Peter">
                <predicate>has_location</predicate>
       </item>
</request>
```

```
<response status="200">            //status code, 200 means success
       <item subject="Anna">
                <predicate>has_location</predicate>
                <object>room_1</object>
        </item>
        <item subject="Anna">
                <predicate>has_diagnosis</predicate>
                <object>diabetes</object>
        </item>
        <item subject="Peter">
                <predicate>has_location</predicate>
                <object>room_2</object>
        </item>
</response>
------------------------------------------------------------------------------------------------
<response status="XXX"> //status code, XXX is as the same as HTTP status
code
     <message>XXX</message>         //reason for failure
</response>
```

Another internal interface is between the broker and the context gathering server. In this interface, MQTT protocol is used, and the message pattern is as the same as it is in interface cd in section 5.4.1

5.5 Context Gathering Server

Context gathering server subscribes data sources and collects latest data. It maintains the context collected from every data source. When the urgency server asks for context of a patient, the context gathering server prepares the relevant context according to the request content.

5.5.1 Context Gathering Server Structure

As shown in figure 13, context gathering server includes five modules: urgency server connector, request handler, broker connector, context DAO (Data Access Object) and database.

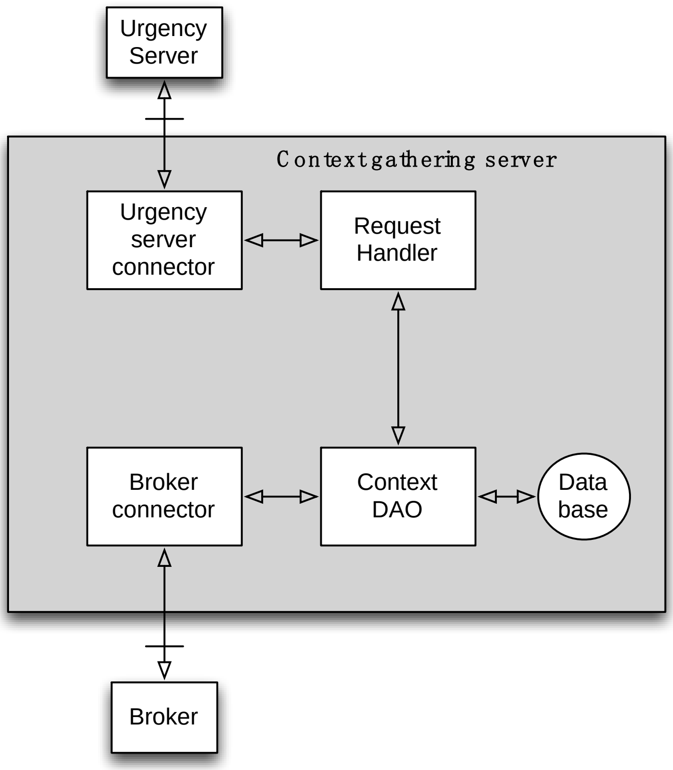


Figure 13 Context gathering server structure

Urgency server connector communicates with urgency server, gets request from urgency server and sends response back.

Request handler processes the messages from the urgency server, extracts information from a message and encapsulates response information in a message.

Broker connector subscribes data from data sources and receives publish. Broker connector also handles MQTT protocol.

Context DAO is the module handles database operation, such as store and read.

Database is the instance that stores context.

5.5.2 Context Gathering Server Process

There are two types of process in context gathering system: (1) Urgency server sends request to get context about specific patient and (2) Data source publish data to update context.

The first process is shown in figure 14:

1. Urgency server sends a request to ask for context;
2. Urgency server connector receives the request message and hands it to request handler;
3. After request handler gets a message from urgency server connector, it extracts information from the message and asks context DAO to read relevant data;
4. Context DAO forms correct SQL query according to the information extracted in step 3 and asks database to return data;
5. Database executes the query in step 4 and returns data to context DAO.
6. Context DAO returns the data gotten in step 5 to request handler.
7. Request handler encapsulates data as a message that follows the protocol and delivers it back to urgency server connector.
8. Urgency server connector sends this message back to urgency server as a response.

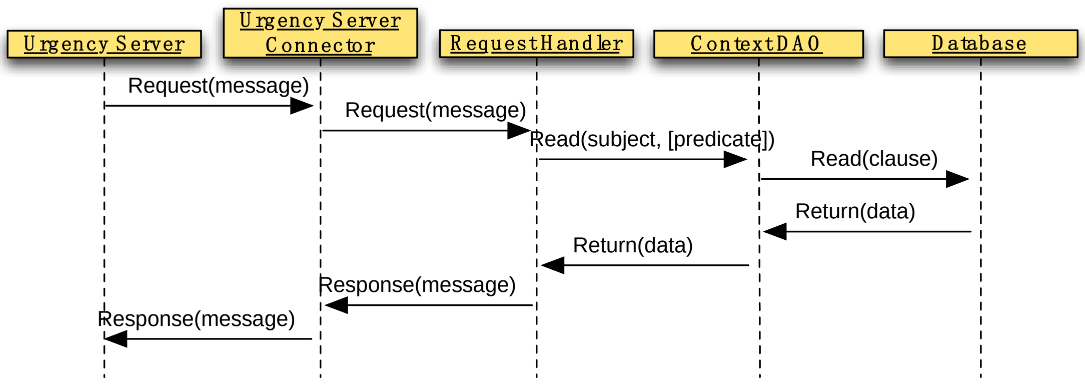


Figure 14 Process of responding urgency server's request

The second process is shown in figure 15:

1. Data source publishes new data to the broker, and the broker sends a new message to broker connector.

2. Broker connector receives the new message and extracts the payload of the MQTT message. After gotten the context statement, broker connector asks context DAO to store it.

3. Context DAO updates the database with new data.

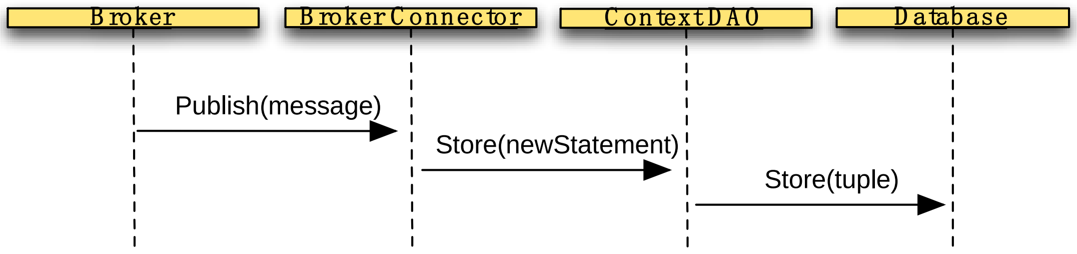


Figure 15 Process of updating context

5.5.3 Context Gathering Server Implementation

The server is programmed in Java. There are two key processes in context gathering server's implementation: (1) database process and (2) MQTT process.

Database process:

MySQL is used as database management, since it is free. In addition, this is a proof-of-concept system, so there is no need to use a commercial database management. In Java code, JDBC (Java Database Connectivity) API (Application Interface) is used to operate MySQL.

The table for the context in the context gathering server is **context_item** and is described in table 2.

Table 2 Table context_item

Field	Type	Null	Key	Description
id	int(11)	NO	PRI	ID of a context item
subject	varchar(255)	NO		subject of a context item
predicate	varchar(255)	NO		predicate of a context item
object	varchar(255)	NO		object of a context item

A class ContextDAO is implemented in order to store and read context items. This class has the following methods:

public void store(ContextItem contextItem)

public ArrayList<ContextItem> read(String subject)

The read method is simple. It reads the relevant items in table context_item. The sql for this method is:

```
select * from context_item

where subject = ?
```

When the data source publish new status about a subject. The object of the statement is the new status, subject and predicate do not change. For example, when a location system publishes a new status about Anna, it publishes “Anna has_location operation_room_1”. In this statement, “operation_room_1” is the new status. “Anna” and “has_location” is the same as before. So the store method first tries to find the context item in table context_item with the sql:

```
select * from context_item

where subject = ? and predicate = ?
```

If the query returns an instance, then updates this instance with new object. Otherwise insert this new context item as a new instance into the database.

MQTT process:

In order to save time, Moquitto [35] is chosen as the broker in the MQTT communication. In addition, Eclipse Paho [36] is used as MQTT API in Java code. With Paho, it is very easy to handle MQTT messages. Paho provides listeners. In order to capture the new messages from the broker, only one callback function is needed to implement, this function is:

public void messageArrived(String topicName, MqttMessage message)

5.6 Urgency Server

The urgency server runs the case-based reasoning. It receives urgency request from the nurse call system, asks current context from the context gathering server, retrieves the most similar case in the case base and returns the urgency number back to the nurse call system. It also receives feedback request from the nurse call system and updates the case base with new context and urgency number.

5.6.1 Urgency Server Structure

As shown in figure 16, urgency server contains 4 modules: analyzer, adjustor, casebase and temporary cases.

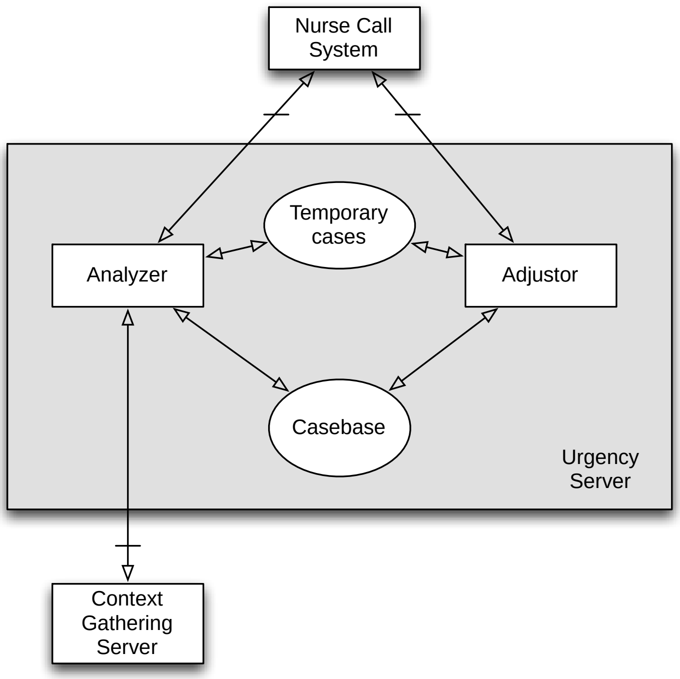


Figure 16 Urgency server structure

Analyzer manages urgency request process. It receives urgency request from the nurse call system, asks context from the context gathering server, retrieves the most similar case from the casebase and sends the urgency number as response to the nurse call system. It also stores the context (case) gotten from the context gathering server in the temporary cases, so the adjustor can retrieve it and store it into casebase.

Adjustor manages feedback request process. It receives feedback request from the nurse call system, retrieves the correspondent context (case) from the temporary cases and stores it with the new urgency number gotten from the feedback request into casebase.

Temporary cases is the case library that stores the context (case) without urgency number. These cases are formed in previous urgency request, and they are waiting for the nurse to give feedback urgency number. This will be discussed in details in the following section.

Casebase is the case library that stores the learned cases. A new case will be compared to the cases in this library in order to find a most similar case.

5.6.2 Urgency Server Process

Urgency server needs to handle two processes: (1) urgency request process and (2) feedback request process

The first process is shown in figure 17:

1. Analyzer gets urgency request from nurse call system with callId and patientId;

2. Analyzer extracts the information from the request and asks context gathering server for the patient’s context;

3. Context gathering system sends the current context about patient with the patientId to the analyzer;

4. Analyzer stores the current context gotten in step 3 with the callId gotten in step 1 into the temporary cases;

5. Analyzer asks the casebase to compare the current context (case) gotten in step 3 with the cases in the casebase;

6. Casebase compares the new case with the previous cases and returns a most similar case.

7. Analyzer gets the urgency number of the most similar case in step 6 and sends it

as response to the nurse call system.

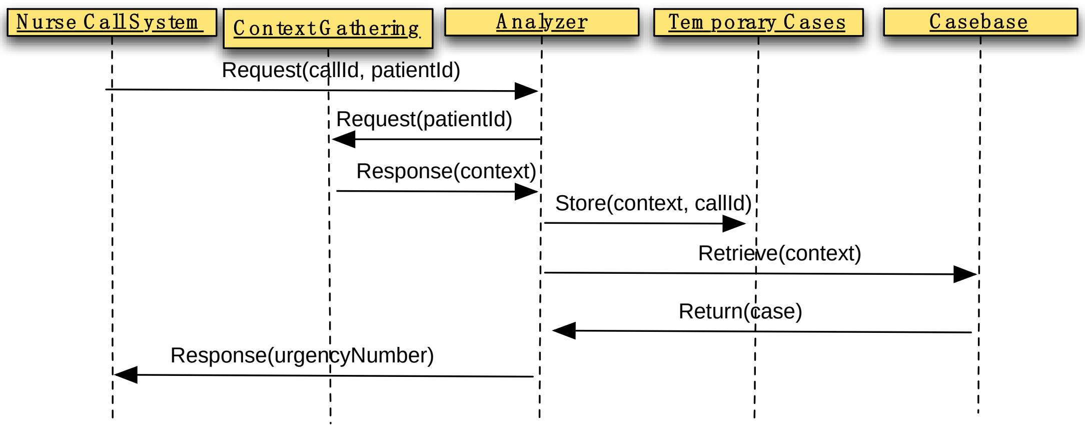


Figure 17 Urgency request process

The second process is shown in figure 18:

1. Adjustor receives feedback request from the nurse call system with callId and new urgency number;

2. Adjustor gets information from the feedback request and asks temporary cases to retrieve the case mapped to the callId in step 1;

3. Temporary cases gets the case according to the callId in step 1 and returns it back to adjustor;

4. Adjustor stores the case (context) gotten in step 3 with the new urgency number in step 1 into the casebase;

5. Adjustor returns response to the nurse call system.

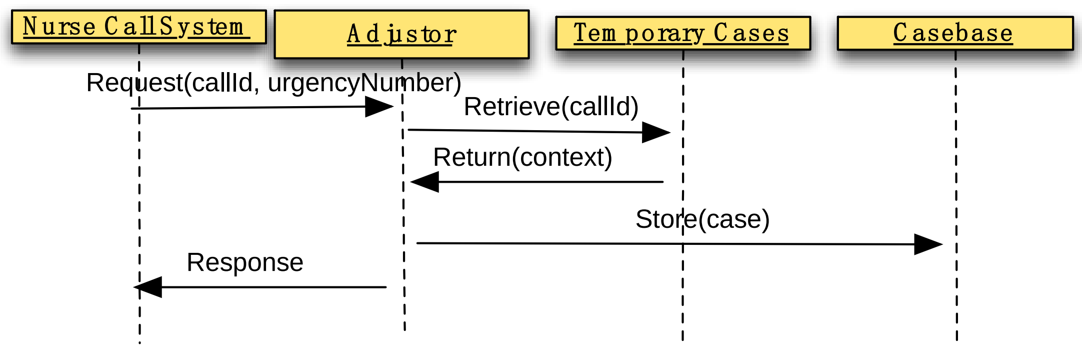


Figure 18 Feedback request process

5.6.3 Urgency Server Implementation

The urgency server is implemented in Java. There are two key processes in urgency server's implementation: (1) database process and (2) case-based reasoning.

Database process:

MySQL is still used in the urgency server implementation. The table for the temporary cases in the urgency server is **context_temp** and is described in table 3.

Table 3 Table context_temp

Field	Type	Null	Key	Description
callId	varchar(255)	NO	PRI	callId is used to map the context with a previous call
context	varchar(1000)	NO		A current context

TempContextDAO is implemented in order to store and read temporary context. I provides the following methods:

public void store(String callId, ArrayList<ContextItem> context)

public ArrayList<ContextItem> read(String callId)

Case-based reasoning:

According to 3.1.2, case-based reasoning includes four steps: retrieve, reuse, revise and retain. Urgency server runs case-based reasoning. Analyzer executes retrieve and reuse while adjustor executes retain. It is not safe to let a normal nurse revise the old cases frequently in practical, so this step can only be done by administer. And administer can revise old cases though extra tool.

The challenge for case-based reasoning is how to store and compare cases. Firstly, a self-defined approach was implemented. After found some issues in the self-defined approach, it was stopped, and an approach based on an open source case-based reasoning project was used in the end.

1. Self-defined approach

In this approach, XML file is used to store cases, and the cases are stored in this form:

```
<cases>
    <case urgency=”1”>
        <context>
            <predicate>has_diagnosis</predicate>
            <object>diabetes</object>
        </context>
        <context>
            <predicate>has_location</predicate>
            <object>patient_room</object>
        </context>
    </case>
    <case urgency=”5”>
        <context>
            <predicate>has_diagnosis</predicate>
            <object>diabetes</object>
        </context>
        <context>
            <predicate>has_location</predicate>
            <object>patient_room</object>
        </context>
        <context>
            <predicate>has_blood_pressure</predicate>
            <object>high</object>
        </context>
    </case>
</cases>
```

Each “case” node represents a case, and a case can have several context items. If a new case contains all the context items in one case, then the case is called matched to the new case. There could be several cases that match the new case, the case with highest urgency number will be chosen as the most similar case.

For example, A with diabetes, high blood pressure, high heart beat rate, and located in patient room. According to the upper cases, both case 1 and case 2 match the new case, but case 2 with higher urgency number, so the urgency number for this patient is 5.

However, this approach is not flexible. Normally some context items have numeric data, such as heart beat, blood pressure and body temperature. For these context, it is difficult determine if two cases are similar. For example, patient with heart beat 92 and patient with heart beat 90 could be similar, but this is difficult to determine based on the algorithm above.

In order to reuse existing tools, myCBR is used instead of the algorithm above.

2. MyCBR approach

According to [37], myCBR is free open source CBR project. It is developed at German Research Center for Artificial Intelligence. MyCBR is suitable for research, because it is free, open source and easy to develop with. Although it does not provide as much functionality as other commercial CBR implementation, it is still enough for research.

In order to reduce the complexity of developing CBR application, myCBR provides Java API as well as GUI (Graphical User Interfaces). The graphical client can be used to setup, modify and manage a CBR project. Java API can also be used to do the same thing.

Since myCBR provides GUI, it has very good flexibility and extendibility. Both Java API and GUI store and retrieve project related data in a common file. User can setup and manage a project using GUI. And urgency server can use Java API to read data from the common file, define a new case to compare to the previous cases, get similarity of each case and add a new case to the casebase.

The GUI usage example can be found in appendix A.

The urgency server uses myCBR Java API to do three tasks: (1) retreive previous cases, (2) compare the new case with previous cases and (3) add new case to casebase.

The first task can be done by using the following API. When the project is retrieved from the common file, all the data related to the project will be retrieved, including the casebase.

```
Project project = new Project(data_path+projectName);
```

The second task can be done by using the following codes.

```
Retrieval retrieval = new Retrieval(concept, casebase);
retrieval.setRetrievalMethod(RetrievalMethod.RETRIEVE_SORTED);
Instance query = retrieval.getQueryInstance();
setAttributesToInstance(query, newCase);
retrieval.start();
List<Pair<Instance, Similarity>> result = retrieval.getResult();
```

The third task can be done by using the following codes.

```
Instance instance = new Instance(concept, NewCaseName);
setAttributesToInstance(instance, newCase.getContext());
AttributeDesc desc = concept.getAttributeDesc("urgency");
instance.addAttribute(desc, newCase.getUrgency());
casebase.addCase(instance);
project.save();
```

Although myCBR is used in the urgency server in the end, the self-defined approach is also kept in the urgency server project. In addition, an interface was defined in order to exploit Java's polymorphism. The interface is defined as:

```
public interface PersistentStore {
    public Case getMostSimilarCase(ArrayList<ContextItem> subjectContext);
    public void addCase(Case newCase);
}
```

Each CBR implementation approach implements this interface, and the system administrator can modify the server configuration file to choose between different approaches. Implementing in this way, the server's extendibility is also improved. If there is need to use a new CBR approach in the future, the new approach just needs to implements the PersistentStore interface, and no other code in the existing system needs to modify.

Chapter 6 Demo Evaluation

In order to evaluate the context-aware urgency cue system described in chapter 5, some tests were done. This chapter will introduce the tests based on the system functionality described in 5.1, and evaluation will also be presented in this chapter.

6.1 Test Objective

The system main functionality has been discussed in 5.1, and they are:

1. Data source should be able to publish new status to the system;

2. It should be able to get an urgency cue from the system by sending an urgency request;

3. It should be able to add an new case to the system by sending a feedback request;

6.2 Test Environment

As described in chapter 5, the context-aware urgency cue system needs data update from data sources in hospital. However, the data sources are not accessible due to practical reason. In addition, the data sources were not designed to publish updates to this system, so an adapter is needed for every data sources. And there is not enough time to develop adapters. So faked data sources are used to test the system.

On the other side, the nurse call system needs to be modified in order to use the context-aware urgency cue system. This thesis focus on providing urgency cues to the nurse call system, but not the nurse call system itself. Thus, a tester was developed to test the context-aware urgency cue system instead of modifying the nurse call system. In fact, the tester sends the requests to the context-aware urgency cue system following the protocols in 5.4. If the real nurse call system sends the requests in the same form, it should have the same test result as the tester.

Figure 19 shows the test architecture. Basically it is the same architecture as in 5.2, but the nurse call system is replaced by tester. Location system, patient sensor and EPR system are replaced by dummy ones individually.

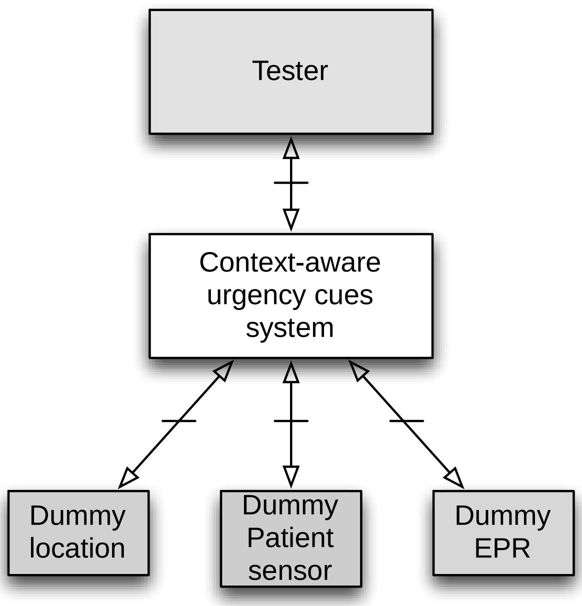


Figure 19 System test architecture

In the tests, all the instances (servers) are executed in the same computer, but different ports. And a myCBR project has already been setup by using myCBR GUI. The setup process can be found in appendix A.

6.3 Functional Test

In order to test the functional requirement, the tests in table 4 are done.

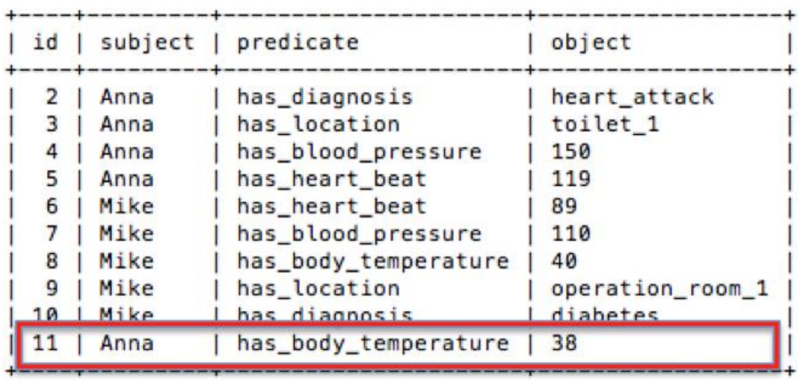
Table 4 Test cases

Test case	Test procedure	Expected result
Patient sensor publishes a new patient's body temperature (refer to functionality 1)	1. The dummy patient sensor publish a statement "Anna has_body_temperature 38"	There is a new context item about "Anna has_body_temperature 38" in the context_item table
Patient sensor updates a previous patient's body temperature (refer	1. The dummy patient sensor publish a statement "Anna has_body_temperature	The context item in context_item table is updated to "Anna

to functionality 1)	36”	has_body_temperature 36”
The system provides the urgency number of the most similar case to the tester (refer to functionality 2)	1. Two previous cases has been in casebase, and Mike’s current context is more similar to the first case 2. The tester sends the urgency request to the system to ask about Mike’s urgency	The system response the tester with the urgency number of the first case
The system stores the new urgency number with the new case when it receives feedback request (refer to functionality 3)	1. The system had processed the procedure in last test 2. The tester sends the feedback request with a new urgency number	The system stores a new case in the casebase that records Mike’s current context in last test with the new urgency number

- Test 1: Patient sensor publishes a new patient’s body temperature

As shown in figure 20, after the patient sensor publish a new patient (Anna)’s body temperature, there is a new context item in the context_item table that shows “Anna has_body_temperature 38”.

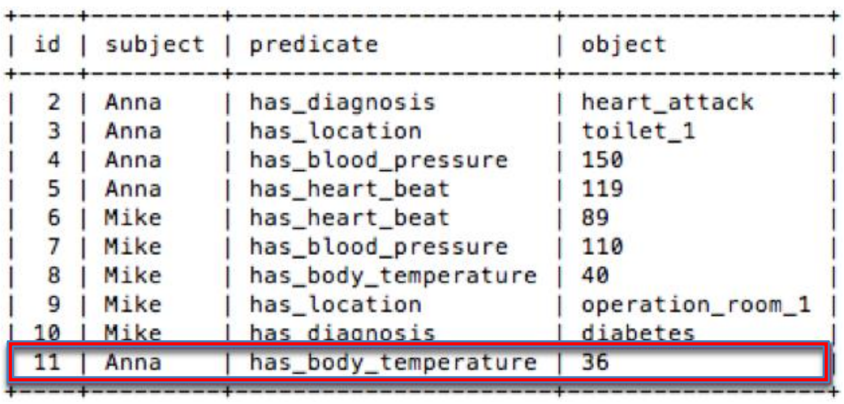


```
+----+---------+----------------------+------------------+
| id | subject | predicate            | object           |
+----+---------+----------------------+------------------+
|  2 | Anna    | has_diagnosis        | heart_attack     |
|  3 | Anna    | has_location         | toilet_1         |
|  4 | Anna    | has_blood_pressure   | 150              |
|  5 | Anna    | has_heart_beat       | 119              |
|  6 | Mike    | has_heart_beat       | 89               |
|  7 | Mike    | has_blood_pressure   | 110              |
|  8 | Mike    | has_body_temperature | 40               |
|  9 | Mike    | has_location         | operation_room_1 |
| 10 | Mike    | has_diagnosis        | diabetes         |
| 11 | Anna    | has_body_temperature | 38               |
+----+---------+----------------------+------------------+
```

Figure 20 Dummy patient sensor publishes a new patient’s body temperature

- Test 2: Patient sensor updates a previous patient’s body temperature

As shown in figure 21, after the patient sensor published a new statement, Anna’s body temperature changed to 36 in the context_item table.

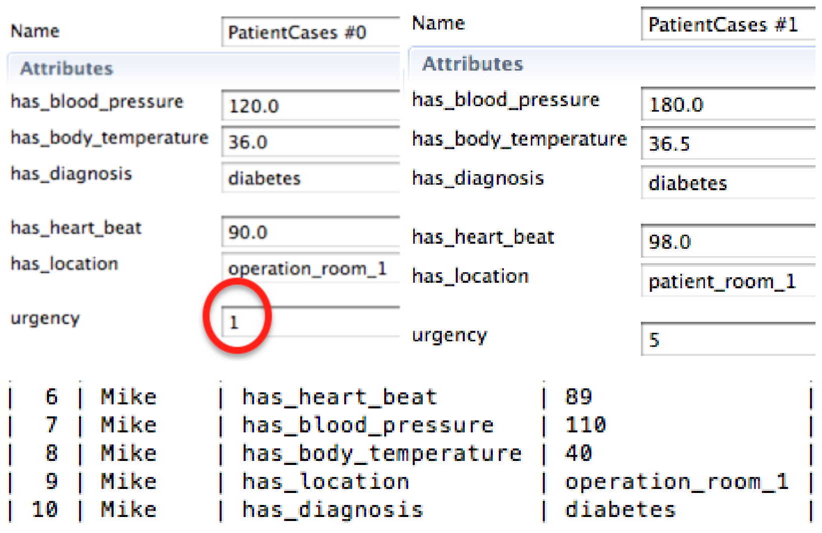


```
+----+---------+----------------------+-------------------+
| id | subject | predicate            | object            |
+----+---------+----------------------+-------------------+
|  2 | Anna    | has_diagnosis        | heart_attack      |
|  3 | Anna    | has_location         | toilet_1          |
|  4 | Anna    | has_blood_pressure   | 150               |
|  5 | Anna    | has_heart_beat       | 119               |
|  6 | Mike    | has_heart_beat       | 89                |
|  7 | Mike    | has_blood_pressure   | 110               |
|  8 | Mike    | has_body_temperature | 40                |
|  9 | Mike    | has_location         | operation_room_1  |
| 10 | Mike    | has_diagnosis        | diabetes          |
| 11 | Anna    | has_body_temperature | 36                |
+----+---------+----------------------+-------------------+
```

Figure 21 Dummy patient sensor updates a previous patient's body temperature

- Test 3: The system provides the urgency number of the most similar case to the tester when the tester sends the urgency request

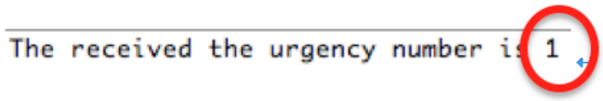
As shown in figure 22, the left upper part is the first case, the right upper part is the second case, and the bottom part is the current context about Mike. It is easy to determine that Mike's current context is more similar to the first case. After the tester sent the urgency request to ask for Mike's urgency, the system responses with the first case's urgency number, which is "1". The tester receives the response and prints it out, as shown in figure 23.



Name	PatientCases #0	Name	PatientCases #1
Attributes		Attributes	
has_blood_pressure	120.0	has_blood_pressure	180.0
has_body_temperature	36.0	has_body_temperature	36.5
has_diagnosis	diabetes	has_diagnosis	diabetes
has_heart_beat	90.0	has_heart_beat	98.0
has_location	operation_room_1	has_location	patient_room_1
urgency	1	urgency	5

```
|  6 | Mike    | has_heart_beat       | 89                |
|  7 | Mike    | has_blood_pressure   | 110               |
|  8 | Mike    | has_body_temperature | 40                |
|  9 | Mike    | has_location         | operation_room_1  |
| 10 | Mike    | has_diagnosis        | diabetes          |
```

Figure 22 Situation of casebase and Mike's context before urgency request

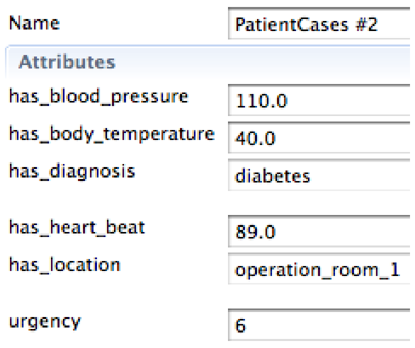


```
The received the urgency number is 1
```

Figure 23 Response for urgency request

- Test 4: The system stores the new urgency number with the new case when it receives feedback request

Based on the procedure of the last test, the tester sends the feedback request to give the new urgency number “6” to the system. After the system received the feedback request, it stores Mike’s context as in the bottom part of figure 22 with the new urgency number “6” into the casebase. As shown in figure 24, the new case can be seen from the myCBR GUI.



Name	PatientCases #2
Attributes	
has_blood_pressure	110.0
has_body_temperature	40.0
has_diagnosis	diabetes
has_heart_beat	89.0
has_location	operation_room_1
urgency	6

Figure 24 A new case is stored after the system received a feedback request

6.4 Evaluation

Based on the dummy data and tester, the system has been proved to be able to work as we expected. It fulfills the functionality stated in 5.1.

Although the systems passed the tests that have been described in this chapter, it is not proved that the system is qualified to use in practice. Since the system was tested without real data and real environment, the system still needs much more tests in a real hospital environment.

Chapter 7 Discussion

Functionally, the context-aware urgency cue system works as it is designed. It listens to the data sources' change and updates the context of the patients. It realizes context-aware. On the hand, the context-aware urgency cue system is able to execute the case-based reasoning. To achieve this, myCBR is used in the system. myCBR is enough for research, but it may not work as well as desired in practice. So it may be necessary to use a commercial myCBR implementation in real hospital.

The system built in this thesis is flexible and extendible. As shown in appendix A, users can use the GUI to set up a new project and define how to calculate the similarities by themselves. And according to appendix B, the system is easy to add other data sources by modifying the configuration file. Thus, this system can be used in different departments in different hospitals.

The tests in chapter 6 are actually far less than enough. In order to test case-based reasoning system, an amount of cases are needed. However, there is not enough time to create lots of cases. So the case base is not big enough to test the accuracy of the case-based reasoning result.

In addition, there is no real data to test with. Since there is no healthcare staff involved in this thesis, the test cases are too naive. So if it is possible, it is better to test the system with real data in hospital.

In order to use this system, the existing data sources need to be modified or add extra adapters. That is the same situation for the nurse call system. The nurse call system needs to be modified as person-oriented architecture in order to exploit the benefits of context-aware and case-based reasoning.

Another issue with case-based reasoning is that it needs a relatively long time to setup the case base at the first beginning. Because there is almost no case in the case base when the project is first set up. The nurses may suffers in the first beginning, because they cannot get valuable urgency cues, but have to give a lot of feedback in order to add more cases into the case base. However, this could be done in the testing stage.

Chapter 8 Conclusion

In this thesis, a context-aware urgency cue system was built up in order to provide clues to nurses along with the nurse calls. The context-aware urgency cue system includes context-aware and case-based reasoning technologies. It requires the nurse call system has a person-oriented architecture. It also requires data sources in the hospital, such as location system and EPR, has ability to publish new updates following a specific protocol. So adapters for each existing data source may be needed in the future. This system has been tested roughly, and real data from hospital is needed to test the system. If it is possible, it is better to test it in a hospital.

References

[1] C.B. Jensen, ”The Wireless Nursing Call System: Politics of Discourse, Technology and Dependability(…)”, CSCW, 15 (5-6), pp 419-441, 2006.

[2] Scholl J, Hasvold P, Henriksen E, Ellingsen G. Managing communication availability and interruptions: a study of mobile communication in an oncology department. Proc PERVASIVE; 2007.

[3] L. Kristiansen, “Nurse Calls via Personal Wireless Devices: Some Challenges and Possible Design Solutions”. In: Proc CBMS 11.2011: pp. 1-6.

[4] Best, BEST IQ , Product Information, http://www.best.se/?id=647, (Accessed in May, 2014)

[5] Anind K. Dey. Understanding and using context. Personal and Ubiquitous Computing (2001) 5: 4-7

[6] Davy Preuveneers (2010). Context-aware adaptation for ambient intelligence. ISBN 978-3-8383-4477-5, pp. 1-9

[7] Davy Preuveneers (2010). Context-aware adaptation for ambient intelligence. ISBN 978-3-8383-4477-5, pp. 15-19

[8] Sen Dipankar, Sen Prosenjit, Das Anand M. (2009), RFID For Energy and Utility Industries, PennWell, ISBN 978-1-59370-105-5, pp. 1-48

[9] Thomas R. Gruber. A translation approach to portable ontology specifications. Knowl. Acquis., 5 (2): 199-220, 1993.

[10] H. Chen, F. Perich, T. Finin, A. Joshi. Soupa: standard ontology for ubiquitous and pervasive applications. International conference on mobile and ubiquitous systems: networking and services. pp. 258-267. 2004.

[11] X. H. Wang, D. Q. Zhang, T. Gu, H. K. Pung. Ontology based context modeling and reasoning using OWL. pp. 18-22, 2004.

[12] Anders Kofod-Petersen. A case-based approach to realizing ambient intelligence

among agents. Doctoral theses at NTNU, 2007:97.

[13] Aamodt A, Plaza E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7, 1994, pp 39-59.

[14] S. A. Grandhi, R. P. Schuler, Q. Jones. Telling Calls: Facilitating Mobile Phone Conversation Grounding and Management. CHI 2011, May 7-12, 2011.

[15] Grandhi, S. A., Schuler, R. P., & Jones, Q. (2009). To answer or not to answer: that is the question for cell phone users. CHI '09 (EA). ACM Press.

[16] F. Ongenae, D. Myny, T. Dhaene, T. Defloor, D. V. Goubergen, P. Verhoeve, J. Decruyenaere, F. D. Turck. An ontology-based nurse call management system (oNCS) with probabilistic priority assessment. BMC Health Services Reserch 2011.

[17] Miller ET, Deets C, Miller R. Nurse call and the work environment: lessons learned. J nurs Care Qual 2000, 15 (3): 7-15.

[18] Van Hoeche S, Decruyenaere J, Danneels C, Taveirne K, Colpaert K, Hoste E, Dhoedt B, De Turck F. Service-oriented subscription management of medical decision data in the intensive care unit. Methods of information in Medicine 2008, 47 (4): 364-380.

[19] Colpaert K, Van Belleghem S, Benoit D, Steurbaut K, Van Hoecke S, De Turck F, Decruyenaere J. Has information technology finally been adopted in intensive care units? Proceedings of the 22nd Annual Congress of the European Sciety of Intensive Care Medicine: Oct 11-14 2009. Vienna, Austria 2009, 235.

[20] Tentori M, Segura D, Favela J. Monitoring Hospital Patients Using Ambient Displays. Mobile Health Solutions for Biomedical Applications New York. Medical Information Science Reference. 2009, 143-158.

[21] Klinov P. Pronto: a non-monotonic probabilistic description logic reasoner. Proceedings of the 5th European Semantic Web Conference: June 1-5 2008. Tenerife, Spain 2008. 822-826.

[22] J. Klemets, L. Kristiansen. A pervasive system for communicating urgency cues to health care workers. 24th International Conference of the European Federation for Medical Informatics Quality of Life through Quality of Information.

[23] Begum S, Ahmed M. U, Funk P, Xiong N, Von Scheele B. A case-based decision support system for individual stress diagnosis using fuzzy similarity matching. Computational Intelligence, 25, 180-195.

[24] C. Marling, S. Montani, I. Bichindaritz, P. Funk. Synergistic case-based reasoning in medical domains. Expert Systems with Applications 41 (2014). 249-259.

[25] S. Begum, M. U. Ahmed, P. Funk. Case-based systems in health sciences – a case study in the field of stress management. Wseas transactions on systems, issue 3, volume 8, March 2009.

[26] A. R. Hevner, S. T. March, J. Park, S. Ram. Design Science in Information System Research. MIS Quarterly Vol. 28 No. 1, pp. 75-105. March 2004

[27] C. Hibbs, S. Jewett, M. Sullivan. The Art of Lean Software Development. ISBN 978-0-596-51731-1. pp 1-24

[28] Joel Spolsky. How Trello is different [Online]. From: http://www.joelonsoftware.com/items/2012/01/06.html [Accessed: May 2014]

[29] Wikipedia. GitHub [Online]. From: http://en.wikipedia.org/wiki/GitHub [Accessed: May 2014]

[30] J. Klemets, T.E. Evjemo, L. Kristiansen, “Designing for Redundancy: Nurses Experiences with the Wireless Nurse Call System”, MEDINFO, pp 328-332, 2013.

[31] X. Franch, P. Botella. Putting Non-functional Requirements into Software Architecture. Software Specification and Design. April 1998. pp. 60-67

[32] W3C OWL Working Group. OWL 2 Web Ontolog Language Document Overview (Second Edition). 11 December 2012.

[33] T. Gu, H. K. Pung, D. Q. Zhang. A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 28 (2005) 1-18.

[34] IBM, Eurotech. MQTT V3.1 Protocol Specification.

[35] Mosquitto [Online]. From: http://mosquitto.org/. [Accessed: May 2014]

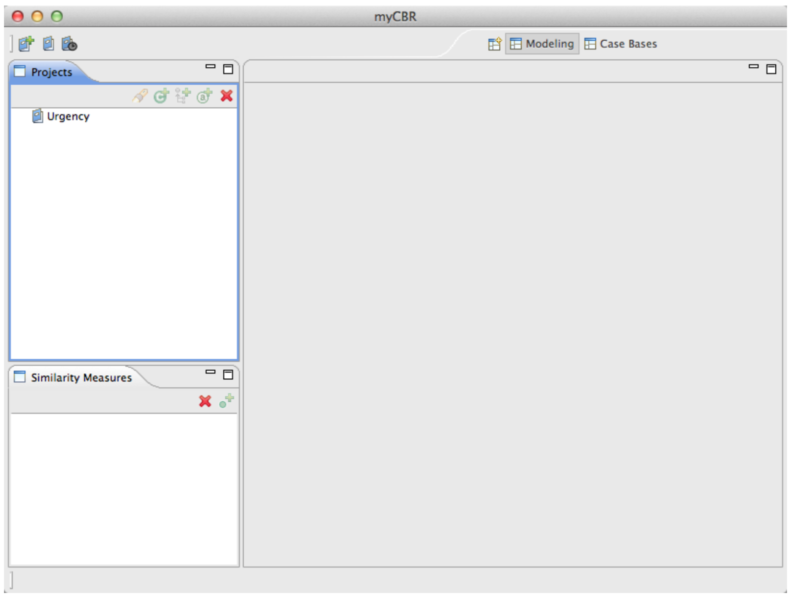
[36] Paho [Online]. From: http://eclipse.org/paho/. [Accessed: May 2014]

[37] A. Stahl, T. T. Roth-Berghofer. Rapid prototyping of CBR applications with the open source tool my CBR. Computer Science Volume 5239, 2008, pp 615-629

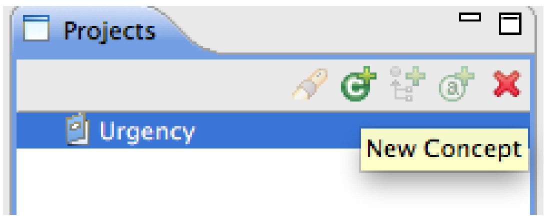
Appendix A Project Setup Example by using myCBR GUI

myCBR provides GUI to setup and manage a CBR project. There are different GUI clients based on Linux, MAC and Windows platform. The clients can be done on myCBR's website: http://www.mycbr-project.net/download.html. On this website, detailed documentation about the GUI can also be found. This appendix will introduce the setup of the project that is used in chaper 6.

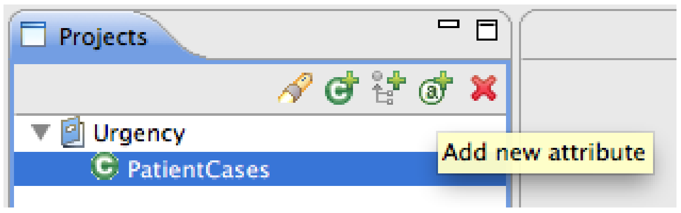
1. Create a new project. This will create a .prj file, and this will be shared between the GUI and Java API;

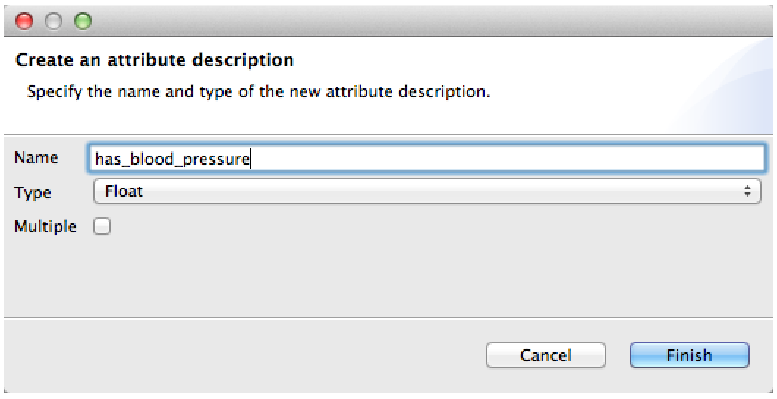


2. Create a new concept by clicking the "New concept" button;



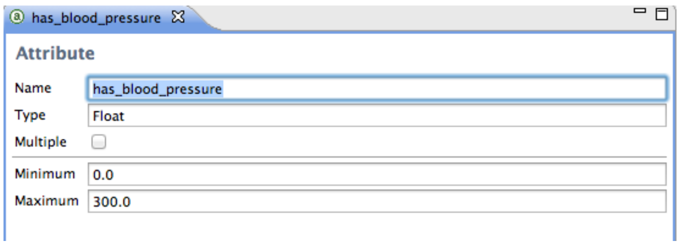
3. Add new attribute “has_blood_pressure” by clicking the “Add new attribute” button, set the attribute type as “Float”, set minimum as 0 and maximum as 300;




Create an attribute description

Specify the name and type of the new attribute description.

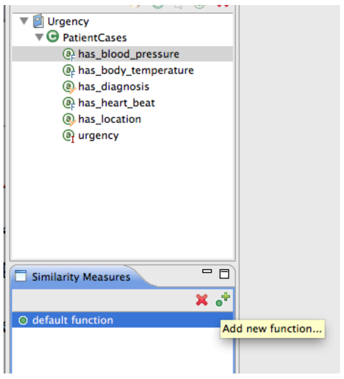
Name	has_blood_pressure
Type	Float
Multiple	☐

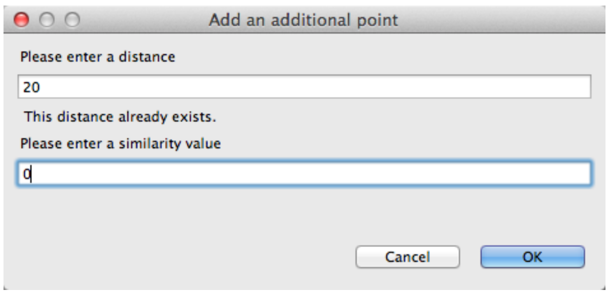


Attribute

Name	has_blood_pressure
Type	Float
Multiple	☐
Minimum	0.0
Maximum	300.0

4. Add new similarity function to this attribute by clicking “Add new function”. Because “has_blood_pressure” is a numeric attribute, it needs to be set a similarity function. Add a similarity point, set distance to be 20 and similarity value to be 0. This defines a linear function as the following figure. The similarity of this attribute is computed based on this function. And similarity of case is computed based on all its attributes’ similarities.





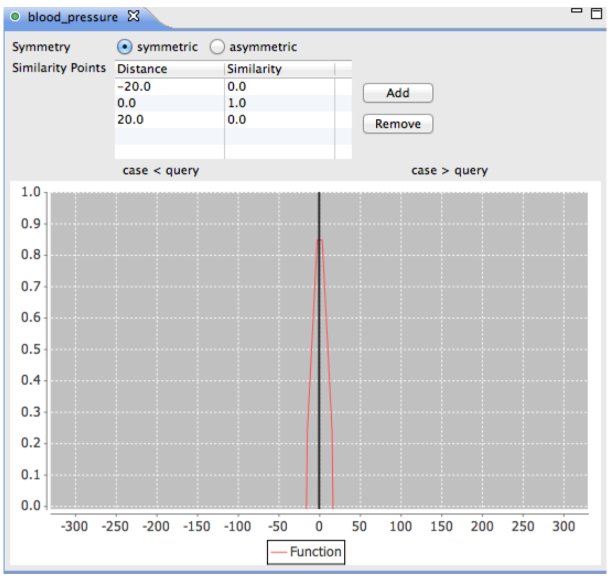
Please enter a distance

20

This distance already exists.

Please enter a similarity value

0



Symmetry [✓ symmetric] asymmetric

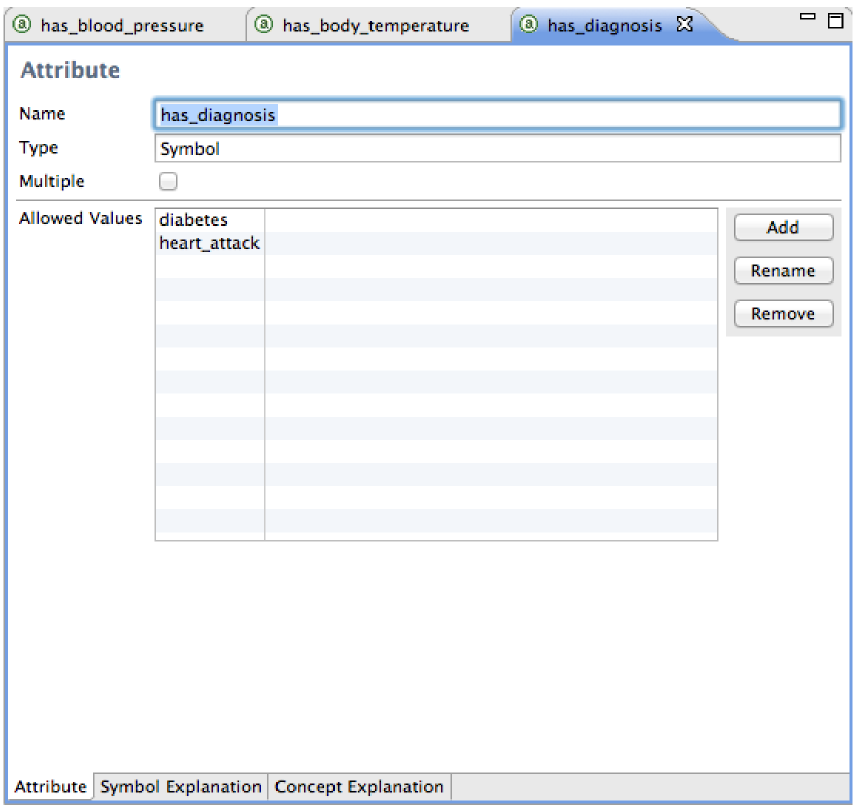
Similarity Points

Distance	Similarity
-20.0	0.0
0.0	1.0
20.0	0.0

5. Add new attribute “has_body_temperature”, set it as type “Float”, set minimum as 0 and maximum as 50;

6. Follow the same procedure in step 4, add similarity function for attribute “has_body_temperature”. Add similarity point “distance 1, similarity 0”;

7. Add new attribute “has_diagnosis”, set it as type “Symbol” and add “diabetes” and “heart_attack” as allowed values; There is no need to add similarity function for symbol type attributes. If two cases have the same value of “has_diagnosis” attribute, then the similarity for this attribute is 1. Otherwise is 0.



Attribute

Name has_diagnosis
Type Symbol
Multiple
Allowed Values diabetes
heart_attack

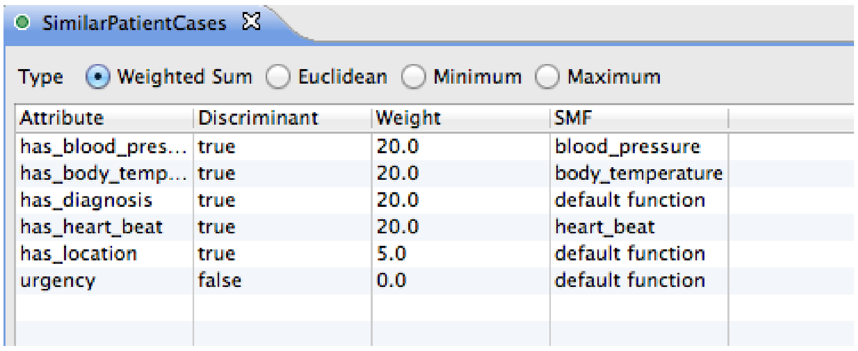
8. Add new attribute “has_heart_beat”, set it as type “Float”, set minimum as 0 and maximum as 200;

9. Follow the same procedure in step 4, add similarity function for attribute “has_heart_beat”. Add similarity point “distance 20, similarity 0”;

10. Add new attribute “has_location”, set it as type “location” and add “operation_room_1”, “patient_room_1” and “toilet_1” as allowed value;

11. Add new attribute “urgency”, set it as type “Integer”, set minimum as 0 and maximum as 10;

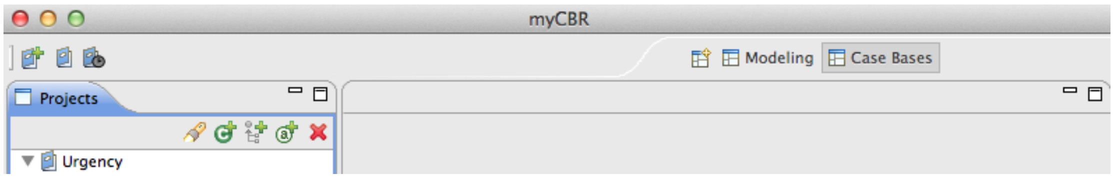
12. Add similarity function to the concept “PatientCases”. Set the fields as the following figure. “Discriminant” determines if this attribute should be considerred when myCBR computes the similarity of cases. “Weight” determines how important this attribute is when myCBR computes the similarity of cases. SMF is the similarity function that has been defined in the previous steps for each attribute.

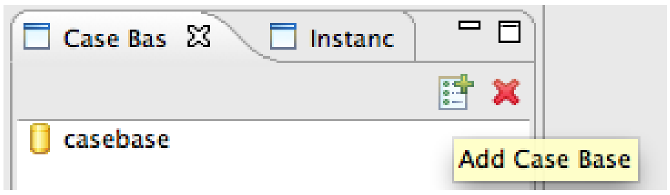


Type [✓ Weighted Sum] Euclidean Minimum Maximum

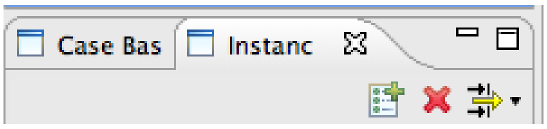
Attribute	Discriminant	Weight	SMF
has_blood_pres...	true	20.0	blood_pressure
has_body_temp...	true	20.0	body_temperature
has_diagnosis	true	20.0	default function
has_heart_beat	true	20.0	heart_beat
has_location	true	5.0	default function
urgency	false	0.0	default function

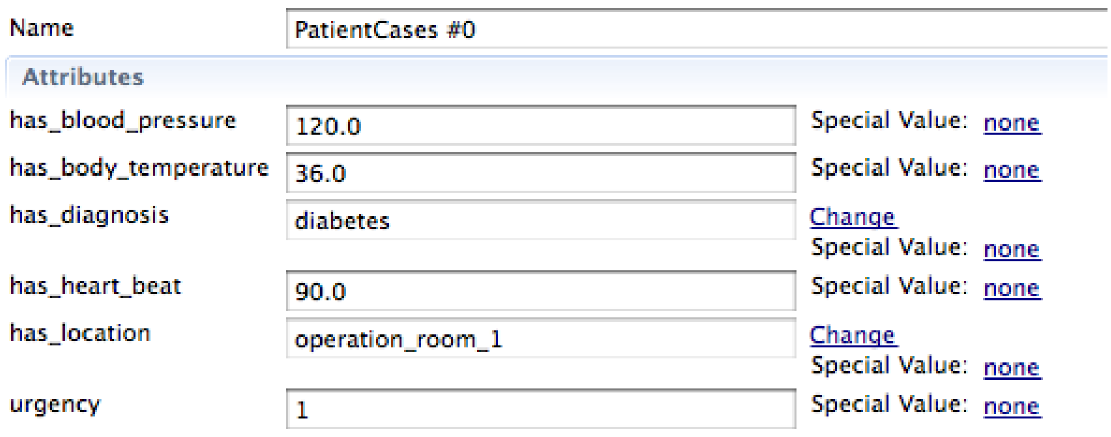
13. Add case base. Firstly select “Case Bases” label. And then add case base by clicking the “Add case base” button;





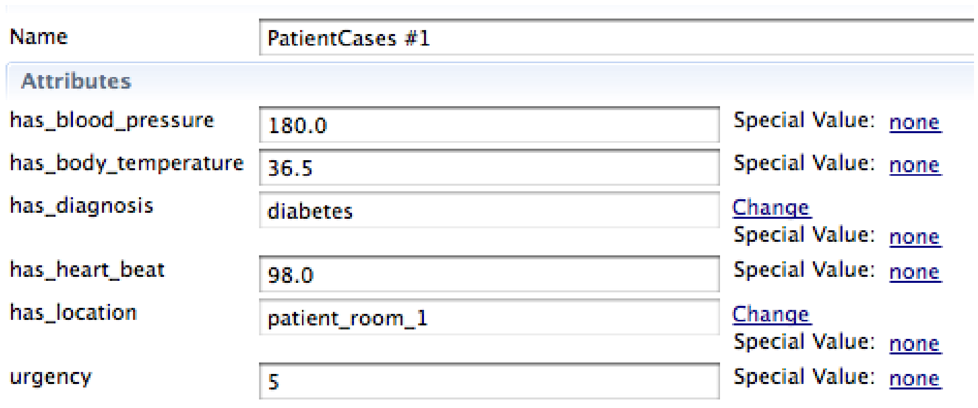
14. Add the first case instance. Firstly select “Instance” label. And then click “Add case base” button. In the following view, fill in the information as the following figure;





Name	PatientCases #0	
Attributes		
has_blood_pressure	120.0	Special Value: none
has_body_temperature	36.0	Special Value: none
has_diagnosis	diabetes	Change Special Value: none
has_heart_beat	90.0	Special Value: none
has_location	operation_room_1	Change Special Value: none
urgency	1	Special Value: none

15. Add the second case instance. Click "Add case base" button again. In the following view, fill in the information as the following figure;



Name	PatientCases #1	
Attributes		
has_blood_pressure	180.0	Special Value: none
has_body_temperature	36.5	Special Value: none
has_diagnosis	diabetes	Change Special Value: none
has_heart_beat	98.0	Special Value: none
has_location	patient_room_1	Change Special Value: none
urgency	5	Special Value: none

16. Save the project. And the project setup is accomplished.

Appendix B System deployment

The context-aware urgency cue system includes context gathering server, urgency server and broker. All the three servers need to be deployed.

1. Moquitto is used as the broker. It can be downloaded on the Internet, and it is free. Other brokers can also be used in stead of Moquitto, but they have to follow MQTT.

2. Context gathering server is implemented in Java, so it can be executed by running the .jar file. Before running this file, the configuration file needs to be modified.

In the following configuration, “mqtt_address” is the broker’s address; “mqtt_name” is the name of context gathering server in the MQTT communication; “mqtt_subscribe” is the list of subscribing data sources, they are separated by “;”; “port” is the port this server listens to, and the urgency server should ask context of a patient though this port; “db_url” is the url of database that stores context items; “db_user” is the user name of database; “db_passwork” is the passwork of the database; “db_table_context_item” is the table that describes context item;

```
mqtt_address=tcp://localhost:1883
mqtt_name=server
mqtt_subscribe=location;patientSensor;epr
port=7979
db_url=jdbc:mysql://localhost/context
db_user=root
db_password=
db_table_context_item=context_item
```

3. Urgency server is implemented in Java, so it can be executed by running the .jar file. Before running this file, the configuration file needs to be modified.

In the following configuration file, “urgency_port” is the port that is used by the nurse call system to send urgency request to; “feedback_port” is the port that is used by the nuse call system to send feedback request to; “context_gathering_ip” is the IP of the context gathering server; “context_gathering_port” is the port of the context gathering server; “db_url” is the url of data base that stores the temporary cases; “db_user” is the user name of database that stores the temporary cases; “db_password” is the password of the database; “db_table_context_temp” is the table describes the temporary cases; “myCBR_path” is the path of the myCBR project; “myCBR_project” is the name of the myCBR project; “myCBR_concept” is the concept name; “myCBR_casebase” is the casebase name;

```
urgency_port=7980
feedback_port=7981
context_gathering_ip=127.0.0.1
context_gathering_port=7979
db_url=jdbc:mysql://localhost/context
db_user=root
db_password=
db_table_context_temp=context_temp
myCBR_path=/Users/which/Documents/workspace/Urgency/
myCBR_project=Urgency.prj
myCBR_concept=PatientCases
myCBR_casebase=casebase
```

A myCBR project has to be set up before executing the urgency server. The way to set up a myCBR project can be found in appendix A.