Háskólinn á Akureyri
University of Akureyri

Viðskipta- og raunvísindasvið
School of Business and Science

Automated triage

Final Year Project
2010

Unnar Freyr Hlynsson

Háskólinn á Akureyri
University of Akureyri

Automated Triage

Final Report

Unnar Freyr Hlynsson

Supervisor: Andrew Brooks

Computer Science Division
Faculty of Natural Resource Science

School of Business and Science
University of Akureyri

Submitted April 2010, in partial fulfilment of
the conditions of the award of the degree BSc.

I hereby declare that this final report is all my own work,
except as indicated in the text:

Signature ______________________

Date _____/_____/_____

Table of Contents

Table of figures 4
Table of tables 5
Introduction 6
History of triage 6
Modern day triage 6
Motivation 7
Suggested solution 7
Related Work 9
Description on current triage systems 9
Emergency Severity Index 9
Computer systems 12
Emergency Centre Organization and automated triage system 12
An Integrated Computerized Triage System in the Emergency Department 15
Benefits of Computer Assisted triage 17
The Effects of Computerized Triage on Nurse Work Behaviour 17
Emergency Triage: Comparing a Novel Computer Triage Program with Standard Triage 18
Requirement Analysis and Design 20
Introduction 20
Purpose 20
Scope 20
Tools 20
Overview 21
General Description 21
Product Perspective 21
Product Requirements 21
User Characteristics 22
Requirements 24
Use Case Requirements 24
External Interface Requirement 31
Implementation 35
Technology platform 35

Description of key features 37

Architecture 37

Database design and access 38

Changes in design 38

Static analysis tools 39

Evaluation 40

Use case testing 40

Testing the use case: Triage a patient 40

Additional Observations 44

Code evaluation 44

HCI testing 44

Cognitive walkthrough 44

Conclusion 55

Personal reflections 55

Skills gained 55

Future work 56

References 57

Appendix A - Code listing 58

NoListener 58

RadioButtonListener 59

ResourceConfirmationListener 60

TableSelectionListener 61

TriageController 62

TriageInitiationListener 66

VitalsConfirmListener 67

ESITriage 69

Patient 75

TriageDialog 81

TriageFrame 90

Appendix B – Project plan (not updated) 114

Appendix C – ESI triage algorithm 115

Table of figures

Figure 1 - High level System overview (Golding et al., 2008, p. 2) 12
Figure 2 - Fitness function used in genetic algorithm (Golding et al., 2008, p. 4)..... 13
Figure 3 - Presented screenshots of the GUI. a) is the nurse view and b) is the doctor’s (Golding et al., 2008, p. 5)..... 14
Figure 4 – Main triage screen(Aronsky et al., 2008, p. 17)..... 16
Figure 5 - Triage Nurse Work and Communication Activity (Aronsky et al., 2006, p. 1)..... 17
Figure 6 - Comparing paper-based and computer-based triage to expert panel (Adapted from original) (Blitz et al., 2004, p. 5) 19
Figure 7 – Main system components 21
Figure 8 – Use case diagram..... 22
Figure 9 – Main Frame looking at table of waiting patients..... 31
Figure 10 – Main Frame viewing triaged patients..... 32
Figure 11 – step d in triage process..... 33
Figure 12 – sted d, confirming triage score..... 34
Figure 13- project package diagram, implementing MVC architecture 37
Figure 14 - Competence cases loaded into triage software..... 40
Figure 15 - ESI triage dialog - short labels 44
Figure 16 – ESI triage algorithm (Gilboy et al., 2005, p. B-3)..... 115

Table of tables

Table 1 - patient_record table schema 38
Table 2 - Checkstyle problems in original classes.. 39
Table 3 - Comparing answers from triage exercises from chapter 9 41
Table 4 - Comparing answers from triage competency cases from chapter 10.................................. 43

Introduction

In general term, the word triage refers to the act of prioritizing resource allocation when said resources cannot be used all at once. This is often applied in the field of medical treatment when medical resources, such as doctors and medical facilities, are not sufficient enough to treat every patient needing medical care at the same time.

History of triage

The term “triage” is derived from the word “trier” in French which mean “to sort”. It was originally used to describe the act of sorting agricultural products. In contemporary times, the word “triage” almost exclusively refers to the act in its medical context. The term first acquired widespread use during World War 1.

The concept of medical triage has in most of its history been tied in military medicine, used on sick and wounded soldiers to indicate if they could be treated back to a fighting state or not. The earliest documentation of this is from the 18th century, in which Western military surgeons were starting to develop protocols in sorting wounded soldiers. The French military surgeon Baron Dominique-Jean Larrey is generally credited for the first formal battlefield triage system.

Before triage was formulated and when it was not used, soldiers were generally treated on a “first come first served” basis, and not necessarily before a battle was over. Larrey, on the other hand, reasoned that the most dangerously wounded soldiers, despite of rank, should have greater priority of treatment than others.

There have been different views on the prioritisation in triage since then, noticeably in World War 1. One strategy from that era focuses the priority of a patient based on the complexity of treatment, that time should not be spent on patient whose wounds would take too long to treat when there were others with less time-consuming treatments. Other strategy from same era purely focuses on priority of wounded soldiers that would be easier to get into fighting shape, placing little or no prioritisation on those that are more difficult to treat.

While application of triage in a military context is well documented, relatively little is documented about civilian triage, such as used in disasters and in emergency departments. It is likely that it was simply adapted from military triage. It is not until 1964 that Weinerman et al releases the first published systematic specification of civilian triage. The further development and refinements of such triage systems were done by individual institutions, local and regional emergency medical systems and federal agencies.

Modern day triage

The most common categories of triage in use are Emergency department (ED) triage, military (battlefield) triage and disaster (mass casualty) triage. The focus of this project will be on ED triage.

Modern ED triage is mostly performed by nurses in the ED which utilise it to prioritize and sort all patients that present there for treatment. Of all environments of medical treatment, EDs usually have the highest resource-to-patient ratio. In daily ED triage there are usually enough resources available to treat every patient that arrives, so that the focus lies on identifying which patients must be treated immediately and who can be allowed to wait.

Many different types of ED triage exist that can also have different number of priority categories associated with them. However, 5-level systems, with 5 priority categories, seem to be proving themselves as being the most reliable today. Example of widely accepted ED triage systems are, amongst others, the American Emergency Severity Index (ESI), the English Manchester Triage Scale, the Canadian Triage and Acuity System (CTAS) and the Australasian Triage Scale (Iserson & Moskop 2007).

In later years, research and effort have gone into computerising the triage process within the different triage categories, ED triage included. Triage is often a complex process, in which the person conducting it is often faced with difficult on the spot decisions. There are arguable benefits in incorporating a computer based triage system, and that doing so can further improve this critical process. The aid of computerized triage has been demonstrated with studies to assist in the decision process

Motivation

A study made to research the effect of the effects of using computerised triage assistance supported that while computer triage did not necessarily shorten the triage time, it did significantly cut down interruption time and task jumping when triaging (Aronsky et al., 2006).

Another, Canadian, study which compared a computer triage system with a standard paper based one in a real working environment, further supports the benefit of computerisation. The study revealed that triage scoring of same cases differed greatly between nurses that used a paper based system. In contrast, nurses that used computer assisted triage had more consistency in their scoring as well as with the scoring made by an expert review panel (Blitz et al., 2004).

Suggested solution

The goal of this project will be to create ED triage decision support software. It will be assumed that it is being developed for a newly built hospital in Reykjavík. This software should be able to categorise patients when they first arrive to the ED. A patient is categorised into a priority class based on a triage nurse’s input, through an implementation of an already established triage system. The triage system that the software implements is the American Emergency Severity Index. The software is only meant to serve as assistance, it should never take control from the user, as he/she should be able to override the software actions through his/her own professional judgement.

The system will be programmed in the Java programming language. The version of java will be Java Standard Edition 6. One reason is the experience I have with the language, which makes it more feasible to finish the implementation in time. Additionally, the system is considered to being developed for a newly developed hospital, in which the operating systems for the computer terminals have not been decided. This would make it less risky to base the system in Java, as it can easily run on many different operating systems.

One of Java’s shortcomings is how slow it can be to develop user interfaces. Therefore the user interface of the system will be developed with Jigloo GUI Builder version 4.5.3. I did not manage to spend time working around with the different user interface builders, but, Jigloo seems to be getting favourable user-reviews.

The triage system is planned to be integrated into the fictional hospital‘s information system. It should be able to aquire patient medical history to aid the triage process. To simulate this, I will develop a small medical database using mySQL. SQL is one of the most popular type of databases and it would be a likely candidate.

Related Work

Description on current triage systems

Emergency Severity Index

The Emergency Severity Index is a five-level ED triage scale. It was developed by ED physicians Richard Wuerz and David Eitel in the United States. The ESI was developed around a new conceptual model of ED triage. It not only considers what patient should be treated first, but also what resources are necessary to get the patient through an ED disposition. An image representation of the algorithm can be found in appendix B (Gilboy et al., 2005).

Decision point A: Is the patient dying (requires immediate life saving intervention)?

In its most simplified implementation, the triage nurse asks if the patient is dying. If the answer is yes, then the algorithm is complete and the patient is given ESI level 1. If the answer is no, the nurse proceeds to decision point B in the ESI algorithm.

According to the handbook, research supports that triage nurses can accurately predict the need for life-saving interventions. Life-saving interventions are intended to support circulation, securing an airway or maintaining breathing in a patient.

In order for the nurse to decide if a patient fulfils the level 1 criteria, there are some questions that are useful to consider:

- Does the patient require an immediate airway, medication, or other hemodynamic intervention?
- Does the patient meet any of the following criteria: already intubated, apneic, pulseless, severe respiratory distress, SpO2 < 90 percent, acute mental status changes, or unresponsive?
- Does this patient have a patent airway?
- Is the patient breathing?
- Does the patient have a pulse?
- Is the nurse concerned about the pulse rate, rhythm, and quality?
- Was this patient intubated pre-hospital because of concerns about the patient's ability to maintain a patent airway, spontaneously breathe, or maintain oxygen saturation?
- Is the nurse concerned about this patient's ability to deliver adequate oxygen to the tissues?

Decision point B: Should the patient wait?

The triage nurse moves to decision point B once it has been determined the patient does not meet the criteria for ESI level 1.

At point B the nurse asks if the patient can afford to wait. If the answer is no then the algorithm is complete and the patient is assigned ESI level 2. If the answer is yes the nurse moves to point C in the algorithm.

To determine the answer, the problem can be split up into three broad questions that should be considered in orderly fashion:

1. Is this a high risk situation?
2. Is the patient confused, lethargic or disoriented?
3. Is the patient in severe pain or distress?

Is this a high risk situation?
The first question in the step is if the patient is in a high risk situation. A patient is at high risk if his/her condition could easily deteriorate or a patient who presents with symptoms suggestive of a condition that need time-sensitive treatment. This includes great risk of death or organ loss. A hih risk patient does not need all vital signs measured in all cases to be identified.

Is the patient confused, lethargic or disoriented?
This is the second question to be asked to determine if the patient meets level-2 criteria. This is meant to detect if the patient has displayed a serious change in level of consciousness. If so, it can indicate that the patient's brain may be structurally or chemically compromised.

- Confused: Inappropriate response to stimuli, decrease in attention span and memory.
- Lethargic: Drowsy, sleeping more than usual, responds appropriately when stimulated.
- Disoriented: The patient is unable to answer questions correctly about time, place or person.

Is the patient in severe pain or distress?
The third and final question the triage nurse needs to consider in step B is if the patient is in pain or distress. If the answer is no, then the nurse can immediately go to step C. If the answer is yes, the patient's pain and/or distress needs to be evaluated.

A nurse rates a patient's pain level on a scale from 0 to 10. A patient scoring 7 or higher is qualified to be assigned ESI level 2, but it is not required.

The measuring of severe distress is more qualitative. Examples of severe distress can be found in patients that are:

- Distraught after experiencing a sexual assault.
- Exhibiting behavioral outbursts at triage.
- Combative.
- Victims of domestic violence.
- Experiencing an acute grief reaction.

Decision point C: Resource needs
If step C is reached in the algorithm, the triage nurse needs to ask "How many different resources this patient is going to consume in order for the physician to reach a disposition decision?" Resources can be, for example, hospital services, tests, procedures, consults or even simple interventions such as applying a bandage.

The nurse is required to draw on past experiences in handling similar cases. The nurse needs to consider what is usually done for patients arriving at the ED with similar complaints. The nurse needs to be familiar with the ED's standards of care .

If a patient is considered to need no medical resources, the algorithm is complete and the patient is assigned the ESI level 5. If a patient is considered to need one resource, the algorithm is also complete and the patient is assigned the ESI level 4. If more medical resources are considered to be needed, the nurse proceeds to step D in the algorithm.

Decision point D: The patient's vital signs

At step D in the algorithm, the nurse needs to consider a selection of the patient's vital sign. If none of the patient's vital signs are in a critical condition the algorithm is complete and the patient is assigned ESI level 3. If some of the vital signs are in a critical state, the nurse can consider upgrading the ESI level to 2 instead.

The vital signs that need to b considered in this step are:

- Heart rate (HR)
- Respiratory rate (RR)
- Oxygen saturation (Sp02)
- Body temperature

The critical range of the vital signs, with the exclusion of oxygen saturation, is affected by the patient's age. While blood pressure (BP) is an important vital sign, it is not required for this step in the ESI algorithm. Pain has already been considered in step B. Body temperature needs to be measured for patients younger than three years, but is not necessary for older patients (Gilboy et al., 2005).

Computer systems

Emergency Centre Organization and automated triage system

This paper was made by Dan Golding, Linda Wilson and Tshilidzi Marwala at the School of Electrical & Information Engineering department at the University of the Witwatersrand, Johannesburg South Africa. It describes their design and implementation suggestions of a computerized triage system meant to triage and schedule patients as they arrive to the ED, declaring the order in which patients receive treatments (Golding et al., 2008).

The specific solution that is produced is made with the ED of the Johannesburg General Hospital in South Africa in mind and implements the Cape Triage System. The problem is that the country does not have any national standards in triaging and patient queue lengths in ED departments are dangerously high, far exceeding the number of doctors and resources available. Triage documents from the hospital also showed that triaging was riddled with inconsistencies. Despite being designed around the particular problems of the hospital, it should be possible to adapt the system design to different hospital situations (Golding et al., 2008).

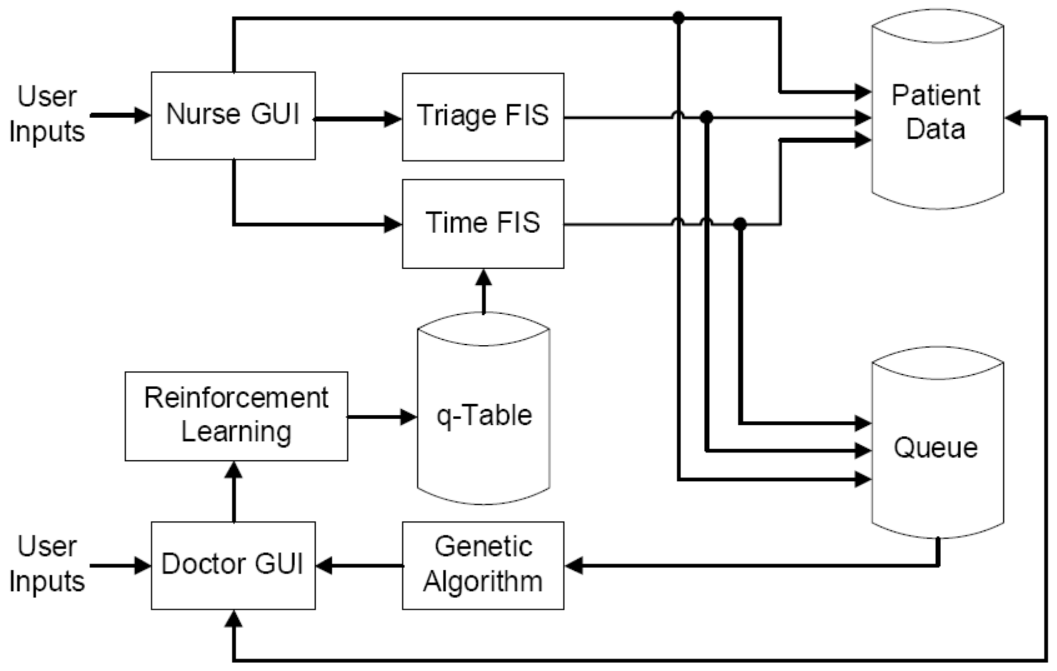


Figure 1 - High level System overview (Golding et al., 2008, p. 2)

The goal of the proposed system is the automation and optimization of the triaging and queuing processes and provides easy access to the system by a separated GUI, one for the triage nurse and another for the doctor receiving the triaged patients. This triage tool implements a fuzzy inference system to aid with the triage procedure as well as in estimating the time a patient might spend occupying the doctor. The result of both components are used to create an optimized patient queue in which high priority patients are treated as soon as possible while patients with middle to low priority still do not have to wait too long (Golding et al., 2008).

The purpose of the Fuzzy inference system is to simulate human medical expert reasoning. After contacting with the heads of triage in the Johannesburg General Hospital and going over the Cape Triage System, the researchers decided that using a Fuzzy Inference System was the best solution. It uses a rule base but is more precise than the Cape Triage System. They further support this by stating that some input used in the triage, such as pain, are already quite fuzzy and are a natural fit into a Fuzzy Inference System. They also say that Fuzzy Inference Systems have low computational complexity and are relatively easy to understand and implement (Golding et al., 2008).

A patient schedule can be represented as a sequence of patients, and the goal would be to generate the optimal sequence of patients at a given time. Genetic algorithm was chosen to be responsible for the creation of a close to optimal sequence. The reasons given are its simplicity and widely available programming toolboxes.

In order to make the optimal sequence the factors that need to be considered for each patient are their urgency and waiting times. Those two factors can be used together to define the optimal sequence through implementation in a fitness function.

When activated, the genetic algorithm generates a population of possible patient sequences, representing each sequence as a single value. A mapping function is responsible for mapping a value put out by the genetic algorithm to a sequence of a given number of patients. The sequence that is found is then evaluated by the fitness function, seen in figure 2.

$$\sum_{i=1}^{n}\left((T_i+1)(t-t_{ai}+\sum_{k=1}^{i-1}t_{ek})\right) \qquad (2)$$

Where: n = number of patients in queue
T_i = TS of patient i
t = current time
t_{ai} = time that patient i arrived
t_{ek} = time that patient k is expected to spend with the doctor

Figure 2 - Fitness function used in genetic algorithm (Golding et al., 2008, p. 4)

All the potential sequences produced by the Genetic Algorithm are measured in this same way. The best scoring sequence can then be used to generate a new generation of potential sequences until a satisfactory number of generations have passed and a locally optimal sequence of patients is proposed (Golding et al., 2008).

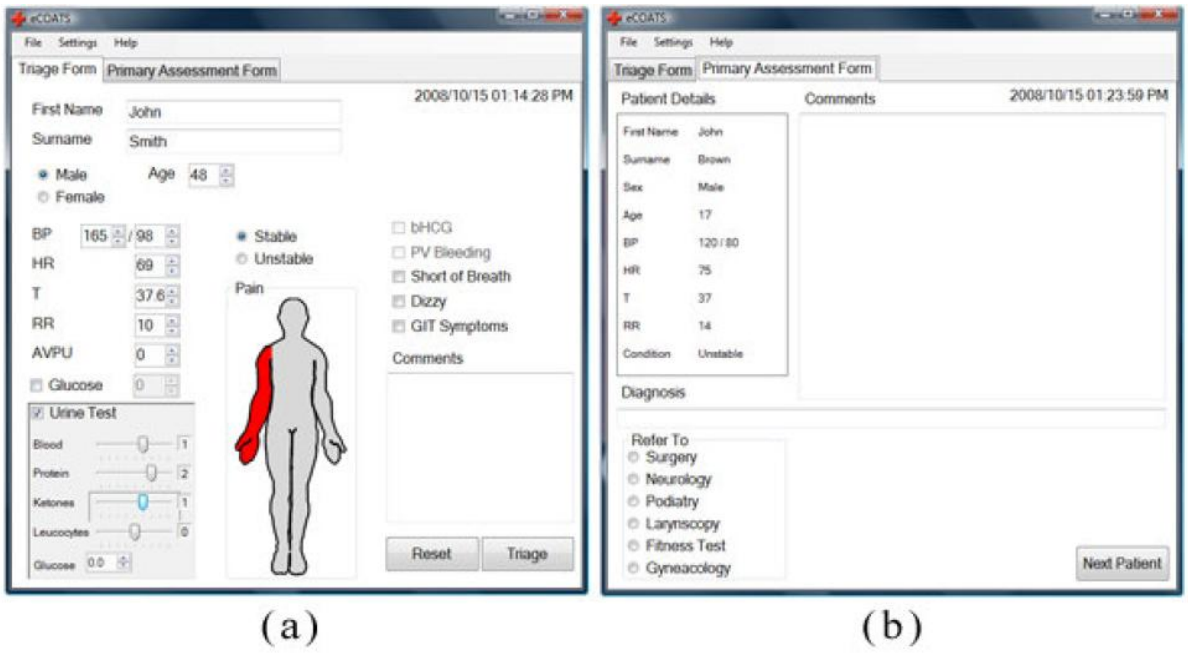
The drawback of using a genetic algorithm is that it is relatively slow. For a queue of 100 patients it can take about 30 seconds to find the best result. The researchers claim that this should not be too much of a hindrance within the particular context. It would take a patient longer time than that to leave the doctor's office and it is therefore acceptable (Golding et al., 2008).

Along with the system solution the researchers also present a simple GUI. It is to be used by the triage nurses and the doctors and is split into separate screens for each user. The interface's main goal is to be user friendly and easily understood as it is vital that it does not hinder the triage nurse in any way.

The nurse interface attempts to minimize the number of times its user enters input through the keyboard, relying more on mouse type input. This includes using sliders and increment/decrement buttons for inputting values for vital signs. The researchers claim that this is easier to use than free typing through the key board.

To input pain measurement, the nurse clicks the ragdoll image to specify the body location of the pain felt. Repeated clicks on the same body part increase the level of pain intensity on it, represented visually through a colour scheme from yellow to red.

The doctor gets his own form that provides him with information on his next patient, such as name and vital signs. It also provides an area for the doctor's comments. A button can be pressed to proceed to the next patient, uploading the doctor's comments and triggering the triage system to create a new patient queue (Golding et al., 2008).



(a)

2008/10/15 01:14:28 PM

First Name John
Surname Smith
● Male Age 48
○ Female

BP 165 / 98
HR 69
T 37.6
RR 10
AVPU 0
☐ Glucose 0
☑ Urine Test
Blood 1
Protein 2
Ketones 1
Leucocytes 0
Glucose 0.0

● Stable
○ Unstable

☐ bHCG
☐ PV Bleeding
☐ Short of Breath
☐ Dizzy
☐ GIT Symptoms

(b)

2008/10/15 01:23:59 PM

First Name	John
Surname	Brown
Sex	Male
Age	17
BP	120 / 80
HR	75
T	37
RR	14
Condition	Unstable

Figure 3 - Presented screenshots of the GUI. a) is the nurse view and b) is the doctor's (Golding et al., 2008, p. 5)

An Integrated Computerized Triage System in the Emergency Department

This triage application was designed, developed and implemented as a collaborative effort between ED and biomedical informatics, nursing staff, directors of the adult and pediatric ED, registration staff, trauma team members, a biomedical informaticist and software engineers. The system is still being supported and improved upon (Aronsky et al., 2008).

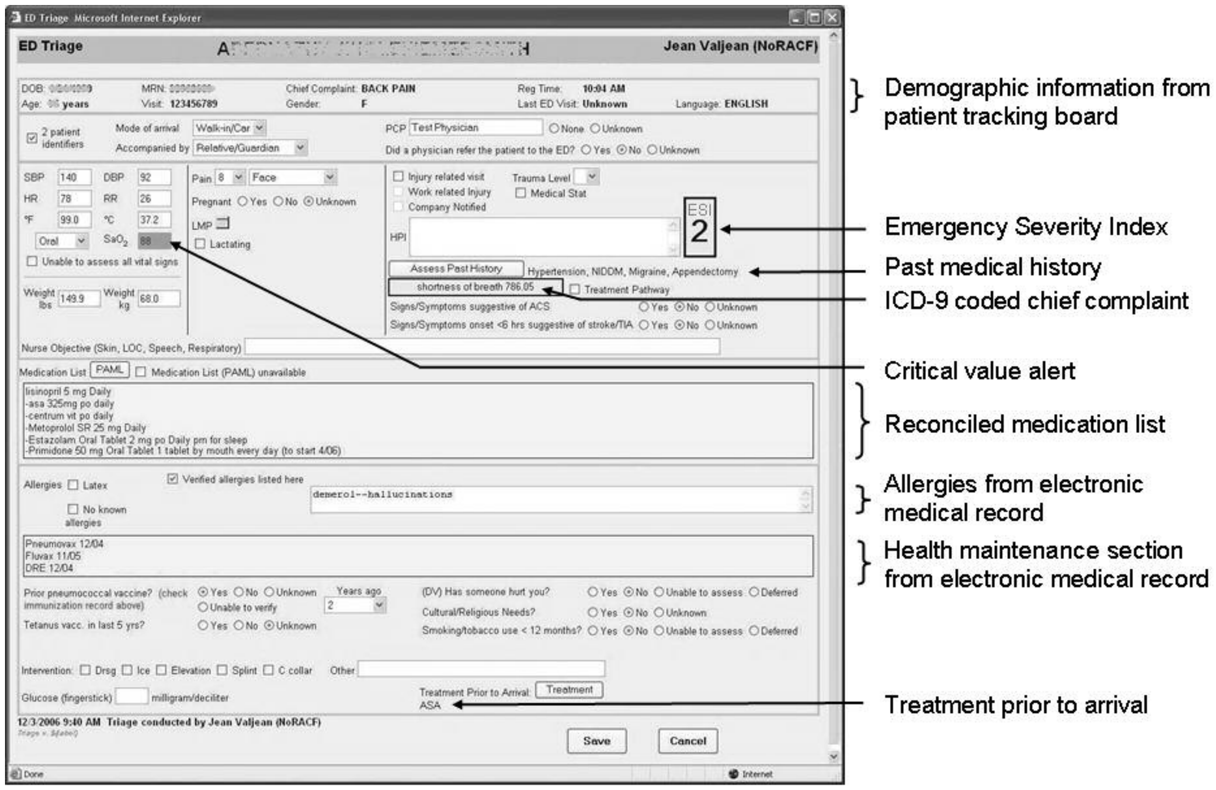
This triage application provides the triage nurse using it with decision support, which include the calculation of a patient's ESI category as well as providing reminders about his medical history and pre-existing conditions. The system is reported to have supported the triage documentation of at least 290.000 paediatric and adult patients (Aronsky et al., 2008).

The triage system is used in both paediatric and regular ED. It is accessed through clinical work stations that are installed throughout the ED's. The system is accessed through the ED patient tracking system and requires user identification and password (Aronsky et al., 2008).

What is special about this particular system is that it is integrated with pre-existing medical systems used in the institution. This is done in attempt to reduce the number of redundant data entries in the triage application as well as in the systems that use data from triage assessments. The triage application exchanges patient data with the many different institutional information systems already in place. The information systems it communicates with include the ED patient tracking board, the longitudinal electronic medical record, the computerized provider order entry, and the medication reconciliation application used in the institution (Aronsky et al., 2008).

This triage application also uses age based triaging, adjusting required input on whether the user needs paediatric or adult care. As an example of paediatric data elements are head circumference and a field telling whether the child was crying both of which can impact vital signs, age-driven ESI criteria, the inclusion of paediatric comorbidities and age-driven alerts for critical vital signs (Aronsky et al., 2008).

After logging in the user is presented by the main triage assessment screen, which is split up into a number of different sections. This is the screen where the user doing the triage enters quantitative patient data such as heart rate and respiratory rate and also qualitative entries such as pain assessment and cultural need assessment.


ED Triage Microsoft Internet Explorer

ED Triage [redacted] Jean Valjean (NoRACF)

DOB: [redacted] MRN: [redacted] Chief Complaint: BACK PAIN Reg Time: 10:04 AM
Age: [redacted] years Visit: 123456789 Gender: F Last ED Visit: Unknown Language: ENGLISH

☑ 2 patient identifiers Mode of arrival: Walk-in/Car Accompanied by: Relative/Guardian
PCP: Test Physician ○ None ○ Unknown
Did a physician refer the patient to the ED? ○ Yes [✓ No] ○ Unknown

SBP 140 DBP 92
HR 78 RR 26
°F 99.0 °C 37.2
Oral SaO$_2$ 88
☐ Unable to assess all vital signs
Weight lbs 149.9 Weight kg 68.0

Pain 8 Face
Pregnant ○ Yes ○ No [✓ Unknown]
LMP
☐ Lactating

☐ Injury related visit Trauma Level
☐ Work related Injury ☐ Medical Stat
☐ Company Notified
ESI 2
HPI
Assess Past History: Hypertension, NIDDM, Migraine, Appendectomy
shortness of breath 786.05 ☐ Treatment Pathway
Signs/Symptoms suggestive of ACS ○ Yes [✓ No] ○ Unknown
Signs/Symptoms onset <6 hrs suggestive of stroke/TIA ○ Yes [✓ No] ○ Unknown

Nurse Objective (Skin, LOC, Speech, Respiratory)

Medication List PAML ☐ Medication List (PAML) unavailable
lisinopril 5 mg Daily
-asa 325mg po daily
-centrum vit po daily
-Metoprolol SR 25 mg Daily
-Estazolam Oral Tablet 2 mg po Daily prn for sleep
-Primidone 50 mg Oral Tablet 1 tablet by mouth every day (to start 4/06)

Allergies ☐ Latex ☐ No known allergies ☑ Verified allergies listed here
demerol--hallucinations

Pneumovax 12/04
Fluvax 11/05
DRE 12/04

Prior pneumococcal vaccine? (check immunization record above) [✓ Yes] ○ No ○ Unknown ○ Unable to verify Years ago 2
Tetanus vacc. in last 5 yrs? ○ Yes ○ No [✓ Unknown]
(DV) Has someone hurt you? ○ Yes [✓ No] ○ Unable to assess ○ Deferred
Cultural/Religious Needs? ○ Yes [✓ No] ○ Unknown
Smoking/tobacco use < 12 months? ○ Yes [✓ No] ○ Unable to assess ○ Deferred

Intervention: ☐ Drsg ☐ Ice ☐ Elevation ☐ Splint ☐ C collar Other
Glucose (fingerstick) milligram/deciliter
Treatment Prior to Arrival: Treatment
ASA

12/3/2006 9:40 AM Triage conducted by Jean Valjean (NoRACF)

Save
Cancel
Done
Internet
Demographic information from patient tracking board
Emergency Severity Index
Past medical history
ICD-9 coded chief complaint
Critical value alert
Reconciled medication list
Allergies from electronic medical record
Health maintenance section from electronic medical record
Treatment prior to arrival

Figure 4 – Main triage screen(Aronsky et al., 2008, p. 17)

"... The main triage screen is partitioned into a) patient demographic information, b) arrival information, c) initial assessment including vital signs, history of present illness (HPI), past medical history, chief complaint, Emergency Severity Index (ESI) and nurse objective, d) medication information, and e) allergy, health maintenance information, screening questions, interventions at triage, and treatment prior to arrival."(Aronsky et al., 2008, p. 17).
From the main page, several secondary pages can be accessed. Those pages show patients past history, the chief complaint, and both treatment and ESI assessment before patient's arrival to the ED (Aronsky et al., 2008).

On the ESI assessment page the user can see rules for vital signs and helps the user in following the ESI algorithm and how it chooses the correct acuity level. The acuity levels can be manually overridden, although the user is required to input his reasons for it (Aronsky et al., 2008).

Much of the data entries are mandatory in order to continue through the process, to maintain data quality, completeness and compliance. Also, notably, when an entered patient attribute is abnormal, a critical value alert shows up, highlighting the attribute in question (Aronsky et al., 2008).

Once the triage nurse decides to end the triage assessment, the application does a check to see if all the mandatory fields have been filled correctly after which it produces a summary page of the triage evaluation along with the time that was used.

Benefits of Computer Assisted triage

The Effects of Computerized Triage on Nurse Work Behaviour

This research was made by Scott Levin, Daniel France, R. Scott Mayberry, Shannon Stonemetz, Ian Jones and Dominik Aronsky at the Vanderbilt University in USA.

The goal of this research was to measure and examine the efficiency of ED triage process before and after the implementation of a computerized triage system. This was conducted in June 2005, when an undisclosed, level 1 trauma centre was switching over from a paper-based triage system to a computerized system.

To measure nurse work and communication efficiency during the triage process, two trained observers conducted fourteen time-and-motion primary task analyses of each triage.

Results of the measurement can be seen in figure 5. The number of triage cases observed, for both pre- and post-implementation of the computer triage system, (n) is seven. The differences between the two samples were tested for significance using the Student's t-test. The results are presented as mean of measurements ± 95% confidence interval. They are based on aggregate information gathered within each of the observation sessions.

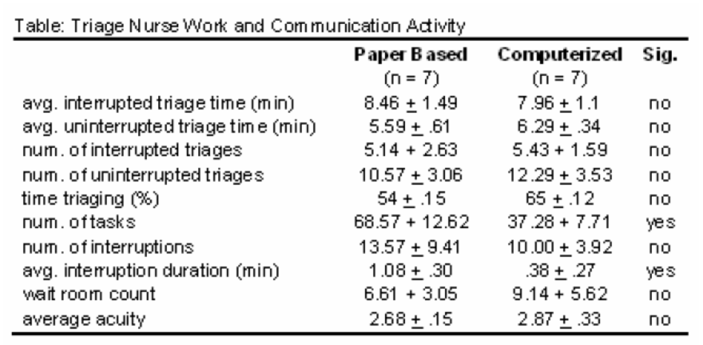


Table: Triage Nurse Work and Communication Activity

	Paper Based (n = 7)	Computerized (n = 7)	Sig.
avg. interrupted triage time (min)	8.46 ± 1.49	7.96 ± 1.1	no
avg. uninterrupted triage time (min)	5.59 ± .61	6.29 ± .34	no
num. of interrupted triages	5.14 + 2.63	5.43 + 1.59	no
num. of uninterrupted triages	10.57 ± 3.06	12.29 ± 3.53	no
time triaging (%)	54 ± .15	65 ± .12	no
num. of tasks	68.57 + 12.62	37.28 + 7.71	yes
num. of interruptions	13.57 ± 9.41	10.00 ± 3.92	no
avg. interruption duration (min)	1.08 ± .30	.38 ± .27	yes
wait room count	6.61 + 3.05	9.14 + 5.62	no
average acuity	2.68 ± .15	2.87 ± .33	no

Figure 5 - Triage Nurse Work and Communication Activity (Aronsky et al., 2006, p. 1)

The researchers state that, according to the data, the change to computerized triaging did not alter the duration of the triage process in a significant way. Triage nurse experience did not have any effect on triage times. Additionally, a patient's chief complaint, age and acuity score did not affect the triage times.

The computer system did, however, reduce significantly the average time duration of interruptions. It also significantly reduced the number of tasks performed by each triage nurse during triaging. It did also increase the length of triage processes that were interrupted (Aronsky et al., 2006).

There are a couple of things debatable about this research. The number of triage cases measured pre- and post- the computerised implementation is only 7. Usually for a statistical measurement to be valid, it requires at least around 15 measured cases. Additionally, they do not define what they mean by a task, so it is difficult to imagine what the number of tasks actually mean.

Emergency Triage: Comparing a Novel Computer Triage Program with Standard Triage

This study was made by Sandy L. Dong, Michael J. Bullard, David P. Meurer, Ian Colman, Sandra Blitz, Brian R. Holroyd and Brian H. Rowe in the University of Alberta. The objective of the study was to find out the agreement between a computerized triage application and memory-based triage.

The study compares a traditional, paper-based application of the five-level Canadian Triage and Acuity Scale with a computerized implementation based on the same triage system. The computerized triage tool in the study is eTriage. It is a Web-based triage decision support tool that assists the user in assigning an appropriate triage score to a patient.

The study was a prospective observational study conducted in real time in a working ED environment. It was conducted in a large Canadian urban tertiary care teaching hospital which had about 67000 ED visits annually. Experienced emergency triage nurses volunteered to take part in the study, all of which were familiar with the memory based methods of triaging. Two out of 37 nurses used in the study were required to use the eTriage application for triaging while the others used traditional methods. Patients' triage scores from both categories of nurses were recorded. For each category the triage score interrater reliability was calculated using kappa statistics. The nurse's triage scoring was compared to the triage scoring from an assembled expert panel reviewing a sample of 97 patients from the same triage cases.

The study showed a significant discrepancy among nurses using the memory-based triage process. The researchers state several things as possible reasons for this. One reason is triage drift, which is when triage nurses subjectively down- or up-stratify patient's triage score, mostly based on the current situation of the ED environment. This has important implications to patient safety, as physician evaluation may be delayed for high-risk patients as a result.

The researchers also discuss that while the original documentation for the Canadian Triage and Acuity Scale is comprehensive its application is reliant on memory and experience. Due to the often busy and frenetic working environment, a nurse cannot be expected to constantly refer to the paper version during triage. This in turn can lead to subjectivity and inconsistency within the triage process.

They also state however, that this should not mean that clinical judgment should be discarded for total dependence on clinical tools. The tools should not take the control from its user but rather encourage its user to override it when his/her clinical impressions require it. The override instance and the reasons for the override should be recorded to be possibly used to adjust the tool. Electronic triage tools, such as decision support tools, could be applied as memory enhancements. They are able to display the key elements for each complaint in such a manner to identify a patient's triage score (Blitz et al., 2004).

It is also interesting to note, that when compared with the triage scoring from the expert review panel, the software assisted scoring showed a generally better agreement to the panel than the memory-based scoring. This further supports the benefits of computerised triage tools, as the expert panel most likely has the most success in triage scoring. This can be observed in figure 6.

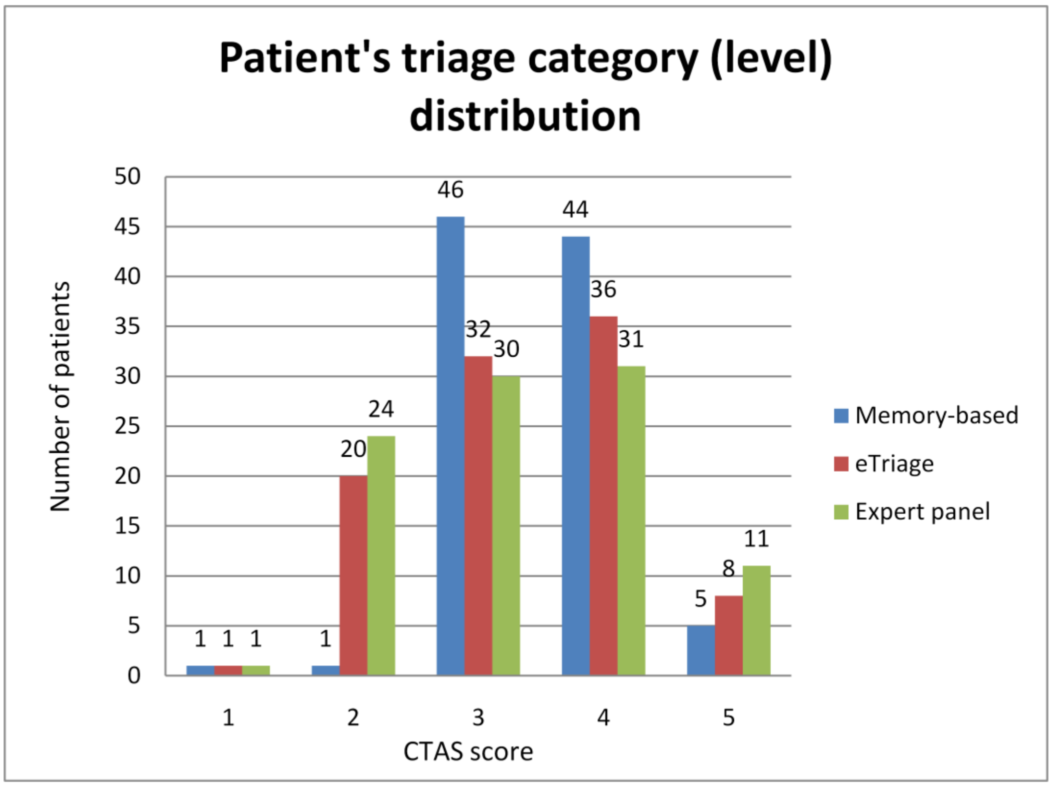


Figure 6 - Comparing paper-based and computer-based triage to expert panel (Adapted from original) (Blitz et al., 2004, p. 5)

Requirement Analysis and Design

Introduction

Purpose

What will be described in this part of the report are the software requirements for ED triage decision support software. This will be done by written description of requirements and supported by UML design diagrams. It is mainly intended for aiding development within the final year project university course as well as to be evaluated for course grade. It could also be used by any developer that might want to continue or add to the project after its original development.

Scope

For the purpose of the final year project, The ED triage decision support software is designed to be used within a theoretical, new Icelandic hospital in a densely populated zone, such as Reykjavík. It would be used within the accident- and emergency department by triage nurses through computer terminals in different rooms within the department. This includes specialized triage rooms equipped with computers having a dedicated access to the software.

The software implements the ESI triage scale and is used to guide nurses through the triage process and to give an appropriate triage score to a patient. It will allow quick access to patient data relevant to the ESI process, such as medical history and medication. The software allows patients triage score to be quickly re-evaluated if the nurse decides to. The software is only meant as a support tool and will allow for the overriding of triage score by the nurse if his/her expert opinion deems it necessary.

The software will be integrated into the theoretical, established system in the hospital. Patient data will be fetched from a medical database within the institution and triage data saved in a dedicated triage database.

Tools

The system will be programmed in the Java programming language. The version of java will be Java Standard Edition 6. One reason is the experience I have with the language, which makes it more feasible to finish the implementation in time. Additionally, the system is considered to being developed for a newly developed hospital, in which the operating systems for the computer terminals have not been decided. This would make it less risky to base the system in Java, as it can easily run on many different operating systems.

One of Java's shortcomings is how slow it can be to develop user interfaces. Therefore the user interface of the system will be developed with Jigloo GUI Builder version 4.5.3. I did not manage to spend time working around with the different user interface builders, but, Jigloo seems to be getting favourable user-reviews.

The triage system is planned to be integrated into the fictional hospital‘s information system. It should be able to aquire patient medical history to aid the triage process. To simulate this, I will develop a small medical database using mySQL. SQL is one of the most popular type of databases and it would be a likely candidate.

Overview

General Description

Product Perspective

Major System Components

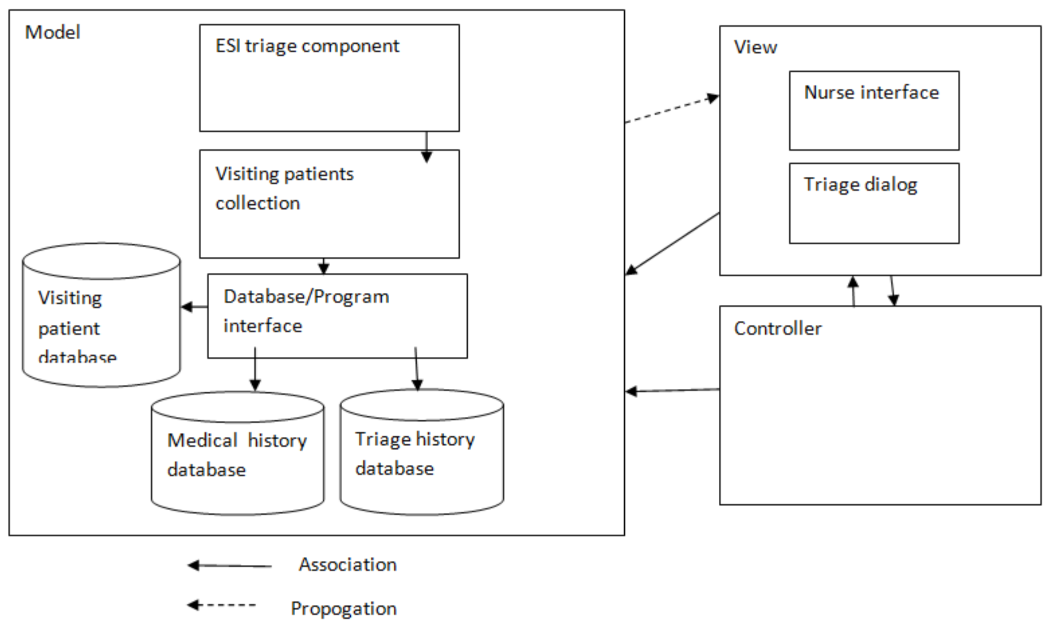


Figure 7 – Main system components

Product Requirements

These use case requirements are described in section 3.3 Requirements. They will be described in the terms of the Unified Modelling Language (UML)

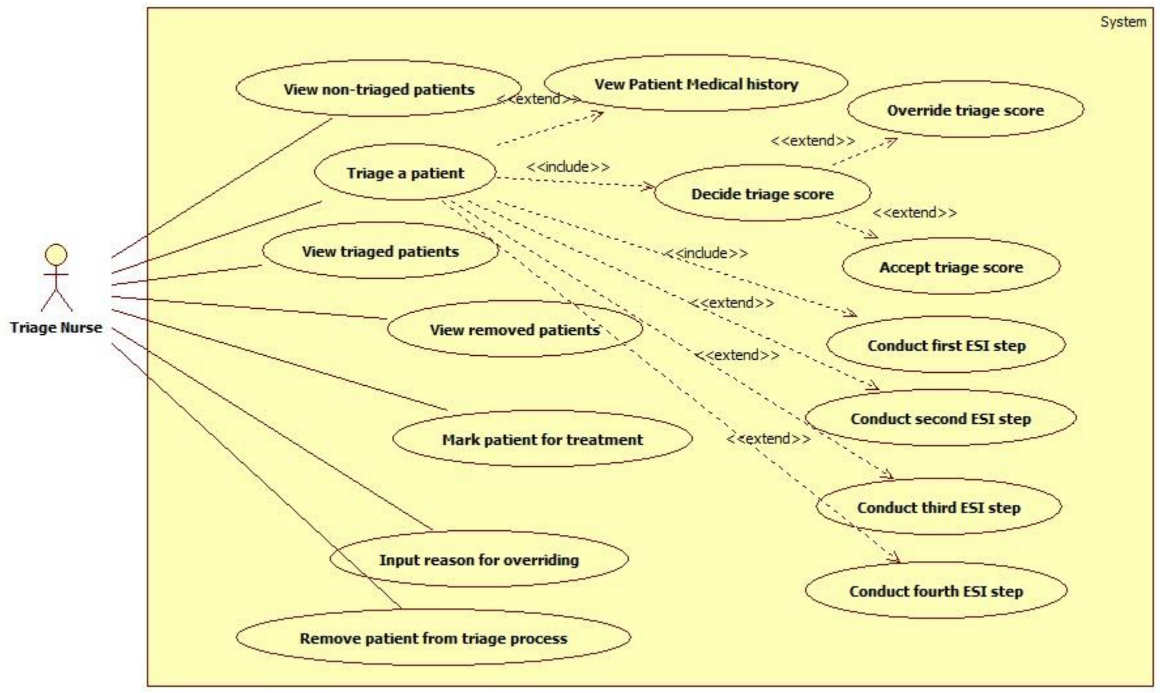


Figure 8 – Use case diagram

1. View non-triaged patients
2. View triaged patients
3. View removed patients
4. Mark patient for treatment
5. Input reason for overriding
6. Remove patient from triage process
7. Triage a patient
8. View patient medical history
9. Conduct first ESI step: “Is the patient dying?”
10. Conduct second ESI step: “Is the patient at high risk?”
11. Conduct third ESI step: “How many resources are needed?”
12. Conduct fourth ESI step: “Danger zone vitals?”,
13. Decide triage score
14. Override triage score
15. Accept triage score

User Characteristics

The software will be used by triage nurses of the hospital’s accident- and emergency department. It can be used by on-duty triage nurses but it could also be used by nurses studying and exercising the ESI triage system.

The on-duty triage nurses are expected to be well learned in the ESI process as well as having studied how to use this decision support tool. They should be familiar with working on software within a graphical windowed environment such as the MS Windows OS incorporates.

The nurses studying the ESI system might obviously not have as much experience in the ESI system, but otherwise share the same characteristics as the on-duty nurses.

Requirements

Use Case Requirements

Use case: 1 View non-triaged patients

CHARACTERISTIC INFORMATION

Goal in Context: A triage nurse will need to be able to quickly view what patients have entered the accident and emergency department and have not been triaged.

Scope: System

Level: Primary task

Preconditions: system displays list of waiting, non-triaged patients

Success End Condition: system displays the table of waiting, non-triaged patiets

Failed End Condition: nothing happens

Primary Actor: Triage nurse

Trigger: A triage nurse is not in the process of triaging and wants to see what patients are waiting to be triaged.

MAIN SUCCESS SCENARIO

0. Triage nurse clicks the “waiting patients” tab
1. System displays the table of waiting patients.

EXTENSIONS

none

SUB-VARIATIONS

none

RELATED INFORMATION

Priority: Core functionality that must be implemented

Performance Target: none

Frequency: Very frequent. Repeated after any patient has been triaged.

Superordinate Use Case: none

Subordinate Use Cases: none

Channel to primary actor: interactive

Secondary Actors: none

Channel to Secondary Actors: none

____

OPEN ISSUES

none

SCHEDULE

Due Date: 15/2/2010

Use case: 2 View triaged patients

CHARACTERISTIC INFORMATION
Goal in Context: Due to how a patient's condition can change with time, a nurse might want to find a recently triaged patient to reassess his triage score.
Scope: System
Level: Primary task
Preconditions: nurse want to see triaged patients
Success End Condition: system displays table of triaged patients
Failed End Condition: nothing happens
Primary Actor: Triage nurse
Trigger: The triage nurse is not conducting a triage. A doctor is available and she wants to find out which triaged patient should receive the doctor's attention first. The nurse might also want to reassess the triage score of a triaged patient.

MAIN SUCCESS SCENARIO

0. Triage nurse clicks the "triaged patients" tab
1. System displays the table of triaged patients.

EXTENSIONS
none

SUB-VARIATIONS
none

RELATED INFORMATION
Priority: Core use case that must be implemented
Performance Target: none
Frequency: Often. Patient condition frequently fluctuate.
Superordinate Use Case: none
Subordinate Use Cases: none
Channel to primary actor: interactive
Secondary Actors: none
Channel to Secondary Actors: none
_ _
OPEN ISSUES
none

SCHEDULE
Due Date: 15/2/2010

Use case: 3 View removed patients

CHARACTERISTIC INFORMATION

Goal in Context: Nurses might want to reconsider patients that they have removed from other patient lists. This is to provide some reversibility in case they had removed a patient by mistake.

Scope: System

Level: Primary task

Preconditions: Triage nurse might have chosen to remove or treat a patient by accident

Success End Condition: system displays the table of removed patients.

Failed End Condition: nothing happens

Primary Actor: Triage nurse

Trigger: The triage nurse is not conducting triage. She thinks she has removed a patient from the triage process by mistake and wants to reconsider bringing him/her back into the triage process.

MAIN SUCCESS SCENARIO

0. Triage nurse clicks the "removed patients" tab
1. System displays the table of removed patients.

EXTENSIONS

none

SUB-VARIATIONS

none

RELATED INFORMATION

Priority: Should be implemented to improve usability.

Performance Target: none

Frequency: Estimated to be occasional. User mistakes can always happen.

Superordinate Use Case: none

Subordinate Use Cases: none

Channel to primary actor: interactive

Secondary Actors: none

Channel to Secondary Actors: none

OPEN ISSUES

none

SCHEDULE

Due Date: 15/2/2010

Use case: 7 Triage a patient

CHARACTERISTIC INFORMATION
Goal in Context: The main use of the system is to assist triaging of patients. The triage nurse will want to either triage a new waiting patient or reassess a previously triaged patient.
Scope: System
Level: Summary
Preconditions: The triage nurse has finished triaging a patient or needs to reassess previous triage of a patient due to changes in his condition.
Success End Condition: The system exits the dialog and moves the patient from the list of waiting patients to the list of triaged patients.
Failed End Condition: The triage nurse cancels the triaging of the patient, exiting the triage dialog.
Primary Actor: Triage nurse
Trigger: Triage nurse has gone through a list of patients and decided which one to triage next.

MAIN SUCCESS SCENARIO (Based on practice triage case 21, page 64 in ESI handbook)

2. Triage nurse looks at the table displaying the un-triaged patients queue on the main triage screen of the GUI.
3. Triage nurse sees no patient described being in a potential life threatening state. The first patient in the queue is a 68-year old female with her complaint being: “Right arm in a sling. Possibly fractured bone.”
4. Triage nurse selects the patient by clicking on him on the waiting patient list.
5. The system highlights the patient. The system activates the “triage” button.
6. The nurse clicks the triage button.
7. The system launches the triage dialog for the patient.
8. The triage nurse navigates the triage dialog.
9. The System assigns an ESI level of 3 to the patient. The system exits the dialog and moves the patient from the list of waiting patients to the list of triaged patients.

EXTENSIONS
None

SUB-VARIATIONS

3. The system highlights the patient. The system activates the “triage” button. A new patient with an apparent critical condition is added to the waiting patient list.
4. The nurse decides to triage the new patient instead of the current patient. The nurse clicks on the new patient in the patient list

---

RELATED INFORMATION
Priority: The central performance of the software. This must be implemented.
Performance Target: the process must take at maximum 5 minutes.
Frequency: Very often. At least once for every patient that does not leave the ED by themselves.
Superordinate Use Case: none
Subordinate Use Cases: 9 Conduct first ESI step: “Is the patient dying?”, 10 Conduct second ESI step: “Is the patient at high risk?”, 11 Conduct third ESI step: “How many resources are needed?”, 12 Conduct fourth ESI step: “Danger zone vitals?”, 13 Decide triage score
Channel to primary actor: interactive
Secondary Actors: none
Channel to Secondary Actors: none
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

OPEN ISSUES
none

SCHEDULE
Due Date: 15/2/2010

Use case: 12 Conduct fourth ESI step: “Danger zone vitals?”

CHARACTERISTIC INFORMATION

Goal in Context: The fourth step of the algorithm requires the nurse to consider particular vital signs of the patient. The goal is to find out whether or not any of those vitals are in a critical state, which relies on the patients age. The nurse inputs the values of the vital signs to let the software quickly identify which ones are in a critical state.

Scope: System

Level: Subfunction

Preconditions: In step C of the triage, the nurse had estimated more than one resource.

Success End Condition: Patient is assigned ESI level 2 or 3 and the system asks the nurse for confirmation.

Failed End Condition: Exit the triage dialog.

Primary Actor: Triage nurse.

Trigger: When asked to confirm the choice of more than one resource, the triage nurse clicks confirm button.

MAIN SUCCESS SCENARIO

0. The system displays the step D panel on the triage dialog screen. The system prompts for heart rate, respiratory rate and oxygen saturation values to be entered.
1. Triage nurse enters heart rate value of 96.
2. The system prompts for respiratory rate and oxygen saturation values to be entered.
3. Triage nurse enters respiratory rate of 15.
4. The system prompts for oxygen saturation value to be entered.
5. Triage nurse enters oxygen saturation ratio of 98%.
6. The system states that no vital signs are in a critical state. The system asks the nurse for confirmation.
7. The triage nurse clicks the confirmation button.
8. The system states the assignment of ESI level 3 to the patient and asks the nurse for confirmation.

EXTENSIONS

none

SUB-VARIATIONS

5. Triage nurse enters oxygen saturation ration of 85%
6. The system marks the entered oxygen saturation. The system states that there is a vital sign in a critical state. The system states that the nurse should consider raising the ESI level of patient to 2.
7. After evaluating patient history, the nurse clicks the “raise ESI to 2” button.
8. The system states the assignment of ESI level 2 to the patient and asks the nurse for confirmation.

RELATED INFORMATION

Priority: Must be implemented. Part of the core functionality.

Performance Target: None defined.

Frequency: Estimated to be somewhat frequent.

Superordinate Use Case: 7 Triage a patient

Subordinate Use Cases: none

Channel to primary actor: interactive

Secondary Actors: none

<u>Channel to Secondary Actors: </u>none
____

OPEN ISSUES

SCHEDULE

Due Date: 15/2/2010

External Interface Requirement

User interfaces

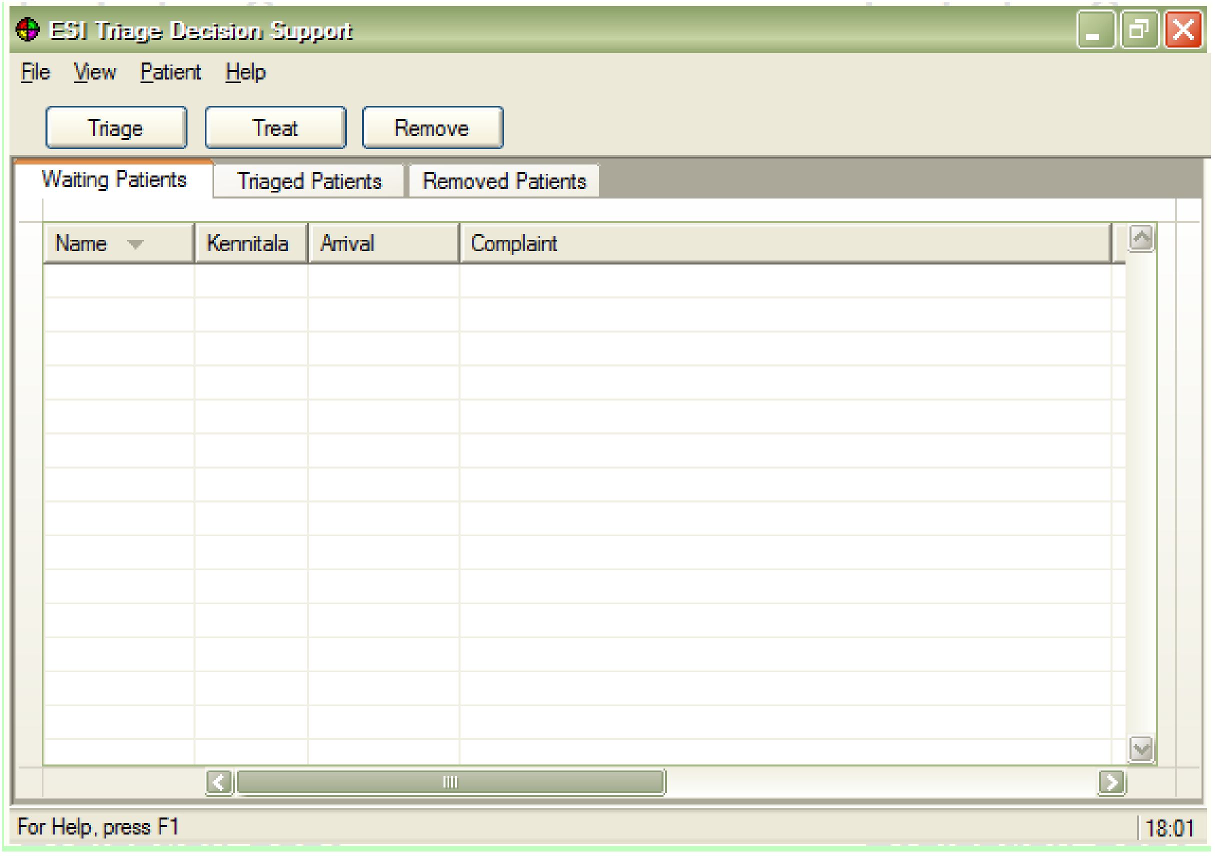


Figure 9 – Main Frame looking at table of waiting patients

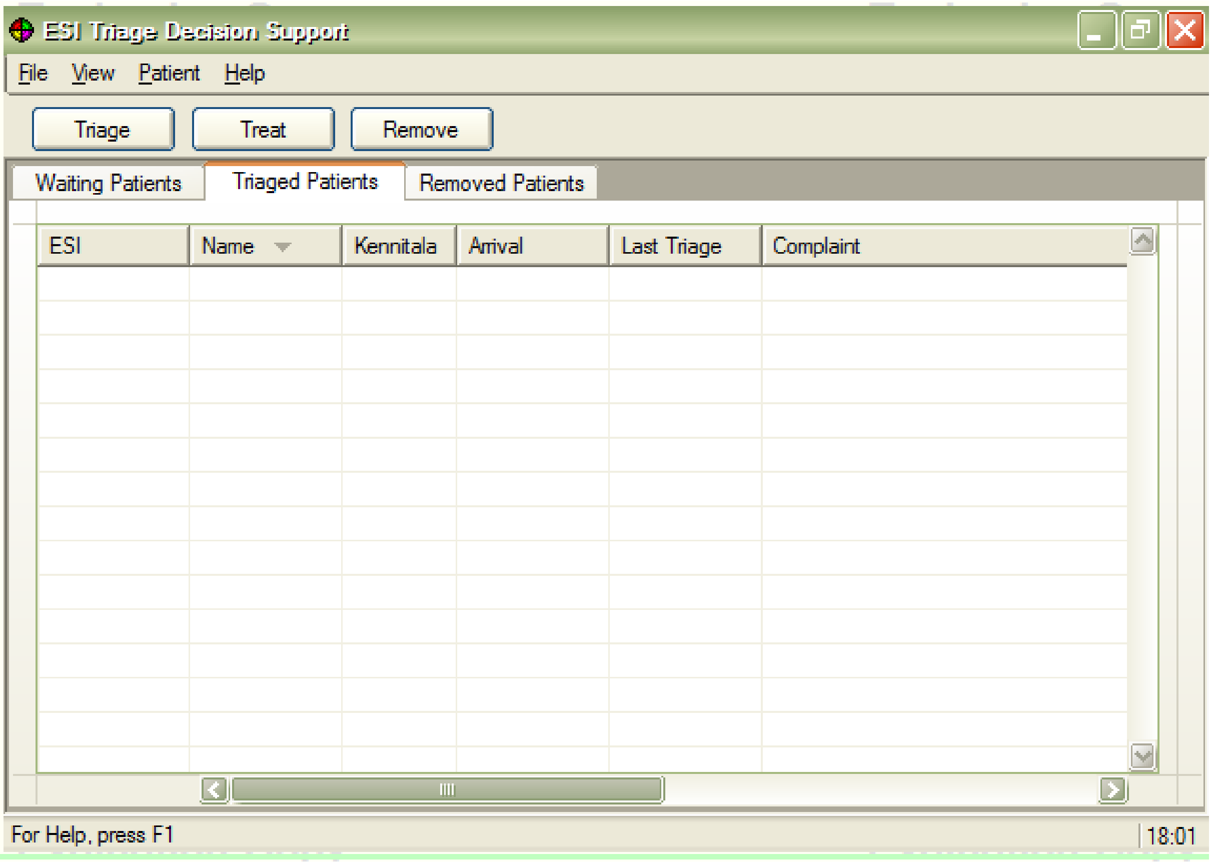
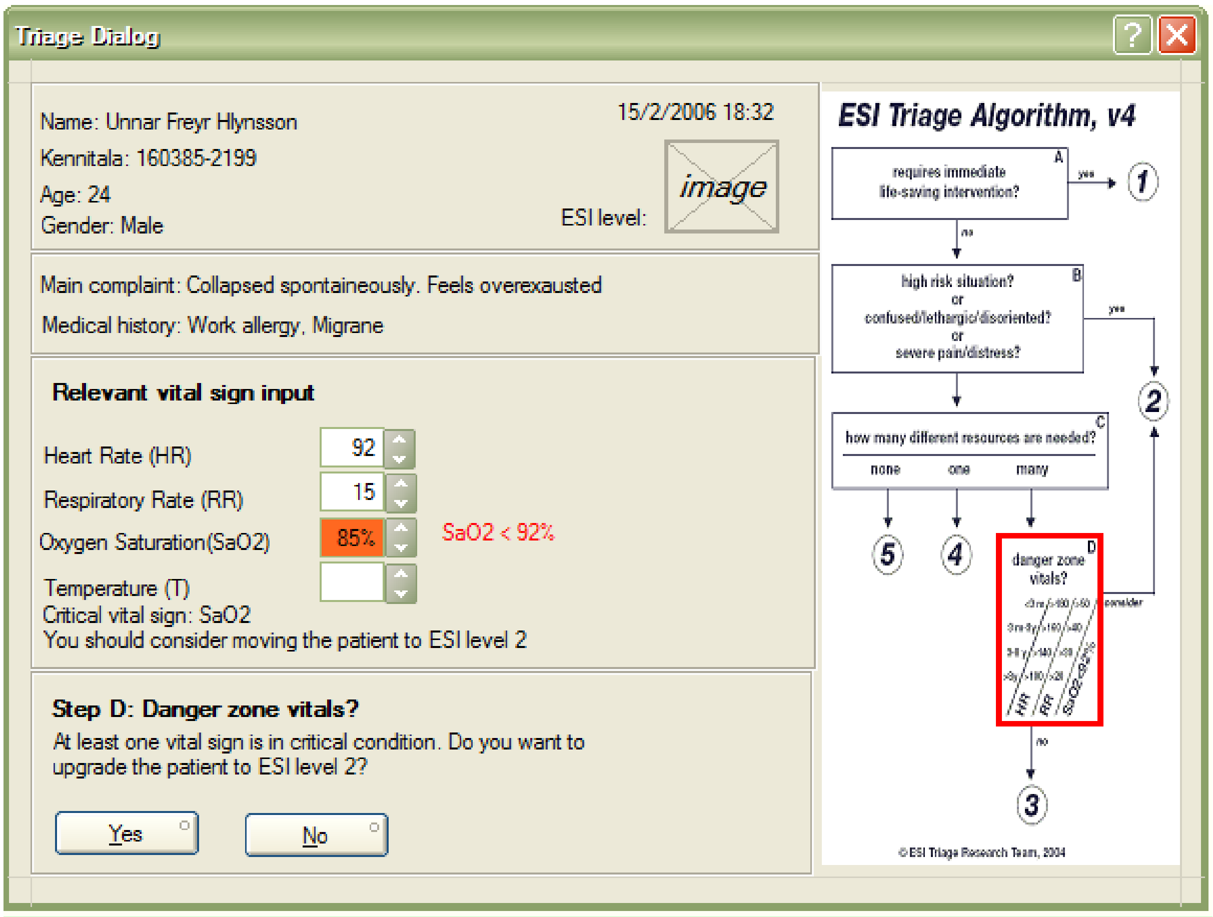


Figure 10 – Main Frame viewing triaged patients


Triage Dialog

Name: Unnar Freyr Hlynsson
Kennitala: 160385-2199
Age: 24
Gender: Male

15/2/2006 18:32

ESI level:

Main complaint: Collapsed spontaineously. Feels overexausted

Medical history: Work allergy, Migrane

Relevant vital sign input

Heart Rate (HR) 92
Respiratory Rate (RR) 15
Oxygen Saturation(SaO2) 85% SaO2 < 92%
Temperature (T)
Critical vital sign: SaO2
You should consider moving the patient to ESI level 2

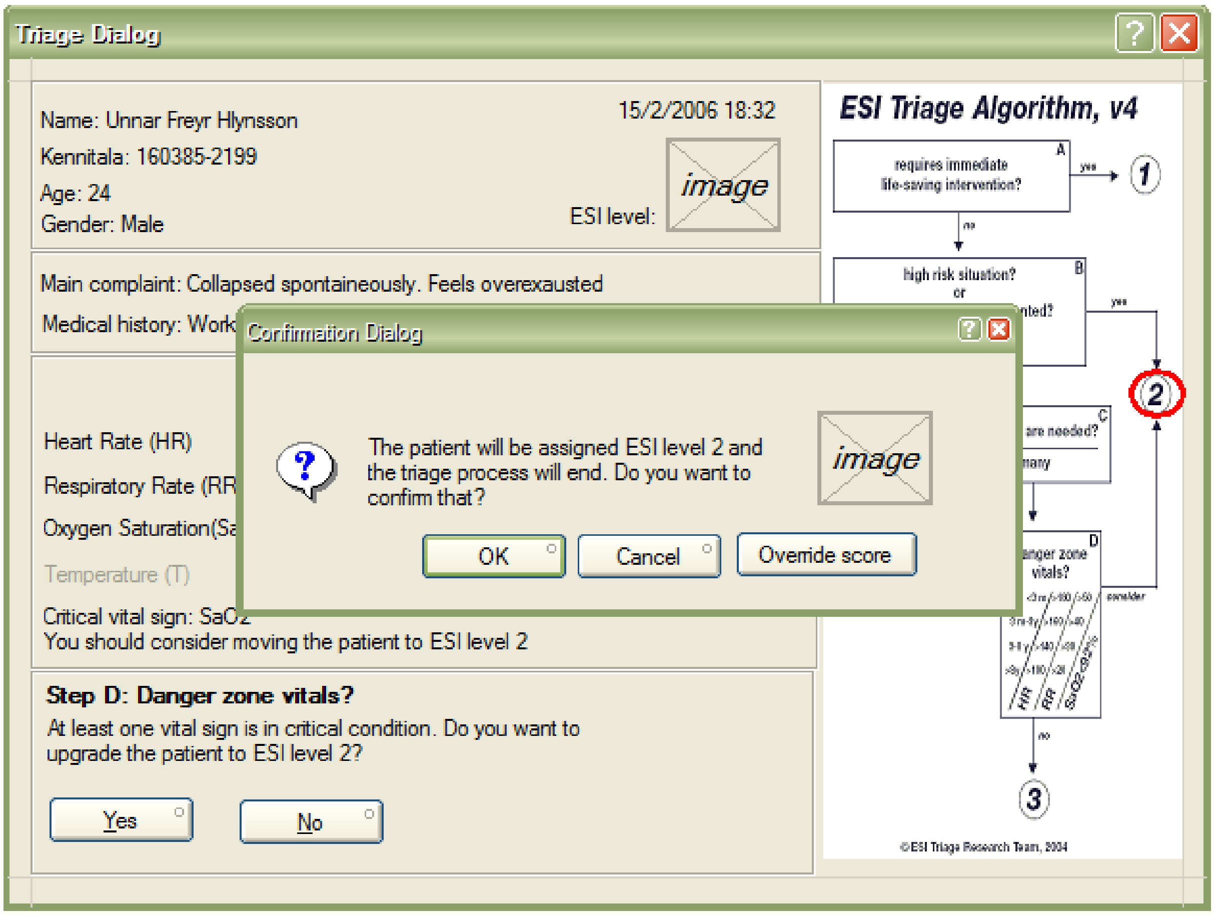
Step D: Danger zone vitals?

At least one vital sign is in critical condition. Do you want to upgrade the patient to ESI level 2?

Yes No

image
ESI Triage Algorithm, v4
A
requires immediate life-saving intervention?
yes
1
no
B
high risk situation? or confused/lethargic/disoriented? or severe pain/distress?
yes
2
C
how many different resources are needed?
none
one
many
5
4
D
danger zone vitals?
<3 m
>180
>50
3 m-3 y
>160
>40
3-8 y
>140
>30
>8 y
>100
>20
HR
RR
SaO2 <92%
consider
no
3
© ESI Triage Research Team, 2004

Figure 11 – step d in triage process


Triage Dialog

Name: Unnar Freyr Hlynsson
Kennitala: 160385-2199
Age: 24
Gender: Male

15/2/2006 18:32

ESI level:

Main complaint: Collapsed spontaineously. Feels overexausted

Medical history: Work

Heart Rate (HR)

Respiratory Rate (RR

Oxygen Saturation(Sa

Temperature (T)

Critical vital sign: SaO2
You should consider moving the patient to ESI level 2

Step D: Danger zone vitals?
At least one vital sign is in critical condition. Do you want to upgrade the patient to ESI level 2?

Confirmation Dialog

The patient will be assigned ESI level 2 and the triage process will end. Do you want to confirm that?

Figure 12 – sted d, confirming triage score

Implementation

Technology platform

The project was developed on a single machine, a Packard Bell EasyNote laptop running on the Windows Vista operating system.

For the implementation of the project I chose Java Standard Edition 6, which is used to implement the system logic and the GUI. I chose Java because of how relatively easy it is to develop network connected applications on it, which the triage software was originally intended to be.

The Java code was implemented and compiled in the Eclipse development environment. The GUI was created with Jigloo GUI editor plugin for Eclipse.

The programming code was compiled into java bytecode by javac 1.6

To implement an incoming patient database, I used a local MySQL database server, edited with MySQL Workbench database design tool.

Technology platform overview:

- Machine
    - Packard Bell EasyNote_MT85-T-149NC laptop
- Operating System
    - Windows Vista™ Home Premium
- Java development Environment
    - Java Eclipse
    - Jigloo GUI editor plugin - Jigloo version 3.9.0
- Static analysis tools
    - Checkstyle (Eclipse plugin)
    - FindBugs (Eclipse plugin)
- GUI/System programming language
    - Java SE 6
    - Javac compiler1.6.0_18
- UML
    - StarUML 5.02.1570
- Database system

- MySQL Server 5.1
- MySQL Workbench 5.2 OSS

Description of key features

Architecture

The system implements the model view controller architecture.

triagemodel

The model is the component representing the core of the architecture, the part that implements the logic and data of the application domain. This is implemented by the triagemodel package in the ESI software code. The model is supposed to contain the classes that support and model the base problems, being able to stand as long as the problem itself.

In this project, the triagemodel contains the classes that access the patient database, keeps a list of all the patients loaded from the database and the execution of the ESI triage process in the software (Bennet et al., 2006).

triageview

The view components in MVC give a user some representative view of the model's current state. In this project it is implemented by the triageview packet. This is where the GUI classes are implemented, as is the case in this project (Bennet et al., 2006).

triagecontroller

The controller component handles the input made in the view, used to manipulate the view and the state of the model. In this project it is implemented through the triagecontroller package. This package contains all of the classes that handle events generated by the view component (Bennet et al., 2006).

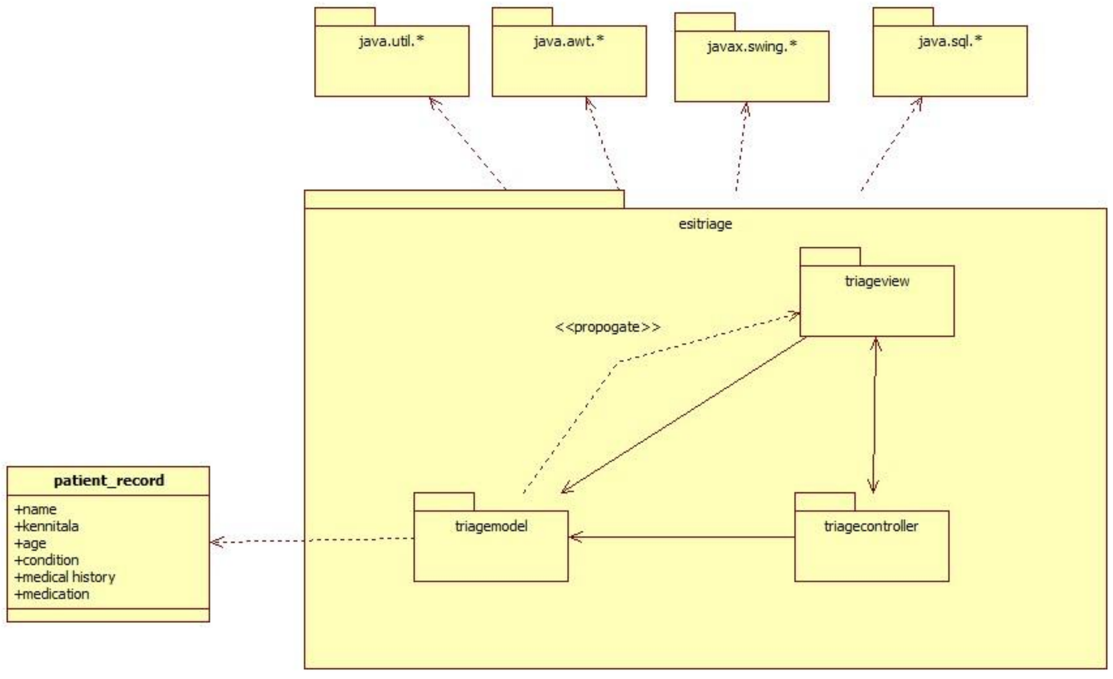


Figure 13- project package diagram, implementing MVC architecture

Database design and access

The database implemented for this project is rather simple, and was meant to be a simple prototypical representation of the incoming patient data that would potentially be accessable from a real-life hospital database. Each tuple in the database represents an incoming patient in to the emergency department, waiting for triage.

To uniquely identify patients, kennitala is used as a key attribute. The name attribute is useful for addressing the patient. There are also attributes of the patient's age, known medical history, known medication as well as his medical complaint. These are all important attributes for a triage nurse to consider during different phases during the ESI triage (Gilboy, N. et al., 2005).

The age category attribute was implemented later on in the implementation process. Not a required part of the original ESI algorithm, it is a makeshift addition to indicate in which of five age ranges that are considered by the algorithm a patient is in. The age group a patient is in alters the limit of critical vital sign range considered in the fourth ESI phase.

patient_record	Kennitala	Name	Age	Complaint	Medical history	Medication	Age category

Table 1 - patient_record table schema

The prototype database was populated by using the MySQL Workbench tool. During development and testing it was run locally on the development machine.

In order to establish a MySQL database connection, I needed to download and import the mysql-connector-java version 5.1.12, a Java database connection (JDBC) driver into the project. To query the database and generate a data structure that could be displayed in the GUI, I imported two classes from online, ResultSetTableModelFactory and ResultSetTableModel. The author of these classes is David Flanagan.

ResultSetTableModelFactory works by encapsulating a JDBC database connection, takes a SQL query string as input and in return outputs a ResultSetTableModel object that can be displayed in a JTable GUI component (Flangan D., 2000).

Changes in design

Not all of the use cases specified in the requirement section (see figure 8) could be implemented before the project deadline. The use cases that were implemented are:

1. View non-triaged patients

7. Triage a patient

8. View patient medical history

5. Conduct first ESI step: "Is the patient dying?"

6. Conduct second ESI step: "Is the patient at high risk?"

7. Conduct third ESI step: "How many resources are needed?"

8. Conduct fourth ESI step: “Danger zone vitals?”

9. Accept triage score

These cases are representative of the core functionality of the triage software and were therefore prioritised above others. They exist as a proof of concept for the software, and their implementation gives a protypical demonstration on how a final version of the software could implement the ESI triage process. Additional benefits of having implemented these use cases is that the implementation of triage process is on a form that can be tested and evaluated, as will be done in the evaluation chapter of this document.

Static analysis tools

In order to enforce coding conventions, the static analysis tool Checkstyle which is a plug-in for the Eclipse IDE. The coding convention used was Sun Checks (Eclipse), which is a slightly altered version of Sun Checks coding conventions.

The Sun Checks (Eclipse) coding convention was enforced on all classes in the triagecontroller and the triagemodel packages. They were not enforced on the triageview package, which contains the GUI code. This is because the GUI code was generated by the Jigloo GUI creator, which automatically generates Java code via a graphical development environment. Changing the GUI code can often disrupt the editing tool.

Additionally, the classes that I acquired online for database access, ResultSetTableModel and ResultSetTableModelFactory have not been altered by enforcing of checkstyle.

Table 2 displays the checkstyle problems that are still unresolved in all of the original classes.

ESI triage CheckStyle problems

Description	Resource	Path	Location	Type
Missing package documentation file.	triageview	/EsiTriage/src	line 0	Checkstyle Problem
Missing package documentation file.	triagemodel	/EsiTriage/src	line 0	Checkstyle Problem
Missing package documentation file.	triagecontroller	/EsiTriage/src	line 0	Checkstyle Problem

Table 2 - Checkstyle problems in original classes

Evaluation

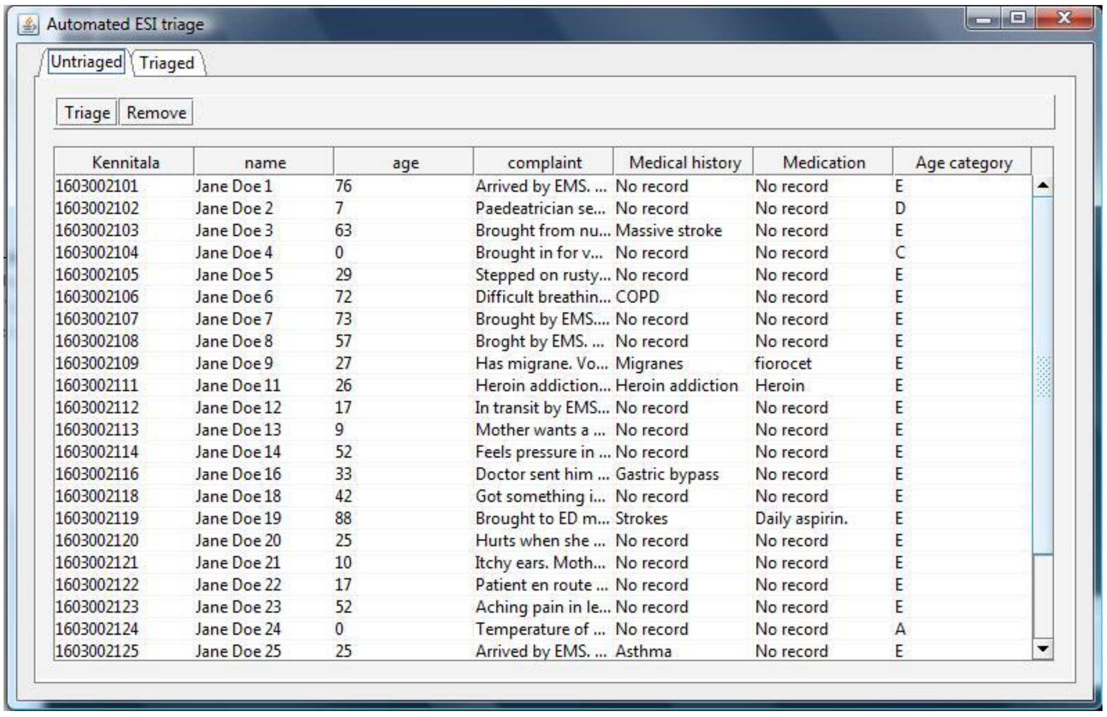
Use case testing

Testing the use case: Triage a patient

The central use case that needs to be carried out by the triage software is to triage a patient. It critical that the ESI category that the software suggests is correctly generated based on the ESI v4 triage.

The official handbook for the ESI triage algorithm contains 60 examples of patients coming into a hospital‘s emergency department. Those include 30 triage nurse exercises in chapter 9 and 30 triage cases for testing a triage nurse‘s competency in the use of the algorithm in chapter 10. Both set of triage cases expect the nurse conducting them to apply the ESI triage algorithm in order to assign to each case an appropriate ESI triage category, 1-5. Both the exercises and the competence cases are followed by what the authors consider to be the correct ESI triage scores for each case, along with a detailed justification (Gilboy, N. et al., 2005).

To test whether or not the ESI triage software would assign correct ESI triage category, given that the triage nurse using doing the triage has expert knowledge in doing ESI triage, I did all of the 60 cases using the triage software as aid. To simulate triage nurse’s expert knowledge I used the descriptive justification for the correct answer in each case as a guide through the algorithm. This includes judging patient’s symptoms and deciding on the number of resources needed for patient treatment during appropriate triage phases. Note that for cases where a vital sign was missing from a case description and vital signs need to be considered in ESI phase 4, I assume that the vital sign is within regular range.



Kennitala	name	age	complaint	Medical history	Medication	Age category
1603002101	Jane Doe 1	76	Arrived by EMS. ...	No record	No record	E
1603002102	Jane Doe 2	7	Paedeatrician se...	No record	No record	D
1603002103	Jane Doe 3	63	Brought from nu...	Massive stroke	No record	E
1603002104	Jane Doe 4	0	Brought in for v...	No record	No record	C
1603002105	Jane Doe 5	29	Stepped on rusty...	No record	No record	E
1603002106	Jane Doe 6	72	Difficult breathin...	COPD	No record	E
1603002107	Jane Doe 7	73	Brought by EMS....	No record	No record	E
1603002108	Jane Doe 8	57	Broght by EMS. ...	No record	No record	E
1603002109	Jane Doe 9	27	Has migrane. Vo...	Migranes	fiorocet	E
1603002111	Jane Doe 11	26	Heroin addiction...	Heroin addiction	Heroin	E
1603002112	Jane Doe 12	17	In transit by EMS...	No record	No record	E
1603002113	Jane Doe 13	9	Mother wants a ...	No record	No record	E
1603002114	Jane Doe 14	52	Feels pressure in ...	No record	No record	E
1603002116	Jane Doe 16	33	Doctor sent him ...	Gastric bypass	No record	E
1603002118	Jane Doe 18	42	Got something i...	No record	No record	E
1603002119	Jane Doe 19	88	Brought to ED m...	Strokes	Daily aspirin.	E
1603002120	Jane Doe 20	25	Hurts when she ...	No record	No record	E
1603002121	Jane Doe 21	10	Itchy ears. Moth...	No record	No record	E
1603002122	Jane Doe 22	17	Patient en route ...	No record	No record	E
1603002123	Jane Doe 23	52	Aching pain in le...	No record	No record	E
1603002124	Jane Doe 24	0	Temperature of ...	No record	No record	A
1603002125	Jane Doe 25	25	Arrived by EMS. ...	Asthma	No record	E

Figure 14 - Competence cases loaded into triage software

For both cases I uploaded all of the cases from both chapters into the locally hosted prototype database patient_record from which they could be loaded into the triage software on launch.

For each case I would diligently follow the ESI triage algorithm within the software and write down the ESI triage category suggested by the software for that case. Finally I would compare the ESI category suggested by the software to the category given in the answer for that case, checking if they are the same or not.

Table 3 shows the result from conducting the 30 triage exercises from chapter 9 of the ESI v4 handbook using the ESI triage software.

Comparison of practise case answers and ESI software result

Practise case index	Correct ESI category	Triage software answer	Equal?
1	1	1	TRUE
2	2	2	TRUE
3	3	3	TRUE
4	2	2	TRUE
5	4	4	TRUE
6	2	2	TRUE
7	4	4	TRUE
8	2	2	TRUE
9	2	2	TRUE
10	5	5	TRUE
11	3	3	TRUE
12	1	1	TRUE
13	2	2	TRUE
14	4	4	TRUE
15	5	5	TRUE
16	5	5	TRUE
17	5	5	TRUE
18	5	5	TRUE
19	1	1	TRUE
20	1	1	TRUE
21	3	3	TRUE
22	5	5	TRUE
23	2	2	TRUE
24	1	1	TRUE
25	1	1	TRUE
26	2	2	TRUE
27	4	4	TRUE
28	3	3	TRUE
29	3	3	TRUE
30	3	3	TRUE

Table 3 - Comparing answers from triage exercises from chapter 9

The first column from the left refers to index of the triage case from chapter 9. The second column is the correct ESI category for each case given by the handbook. The third column is the ESI category given by the software. The fourth and final column indicates if the answer from the book is the same as the answer from the software.

The table demonstrates that for each exercise case, the ESI triage software was able to produce answers matching those given by the handbook. All of the 30 exercises resulted in a correct triage score. Given that all the right decisions are made for each ESI phase in the software, as instructed by the handbook, the software appears to give the correct ESI category when the nurse makes the right decisions.

Comparison of competency case answers and ESI softare result

Competency case index	Correct ESI category	Triage software answer	Equal?
1	1	1	TRUE
2	3	3	TRUE
3	4	4	TRUE
4	5	5	TRUE
5	5	5	TRUE
6	1	1	TRUE
7	2	2	TRUE
8	4	4	TRUE
9	3	3	TRUE
10	2	No result	FALSE
11	4	4	TRUE
12	1	1	TRUE
13	5	5	TRUE
14	1	1	TRUE
15	2	No result	FALSE
16	3	3	TRUE
17	2	No result	FALSE
18	4	4	TRUE
19	2	2	TRUE
20	4	4	TRUE
21	5	5	TRUE
22	1	1	TRUE
23	3	3	TRUE
24	2	2	TRUE
25	3	3	TRUE
26	3	3	TRUE
27	4	4	TRUE
28	3	3	TRUE
29	5	5	TRUE
30	1	1	TRUE

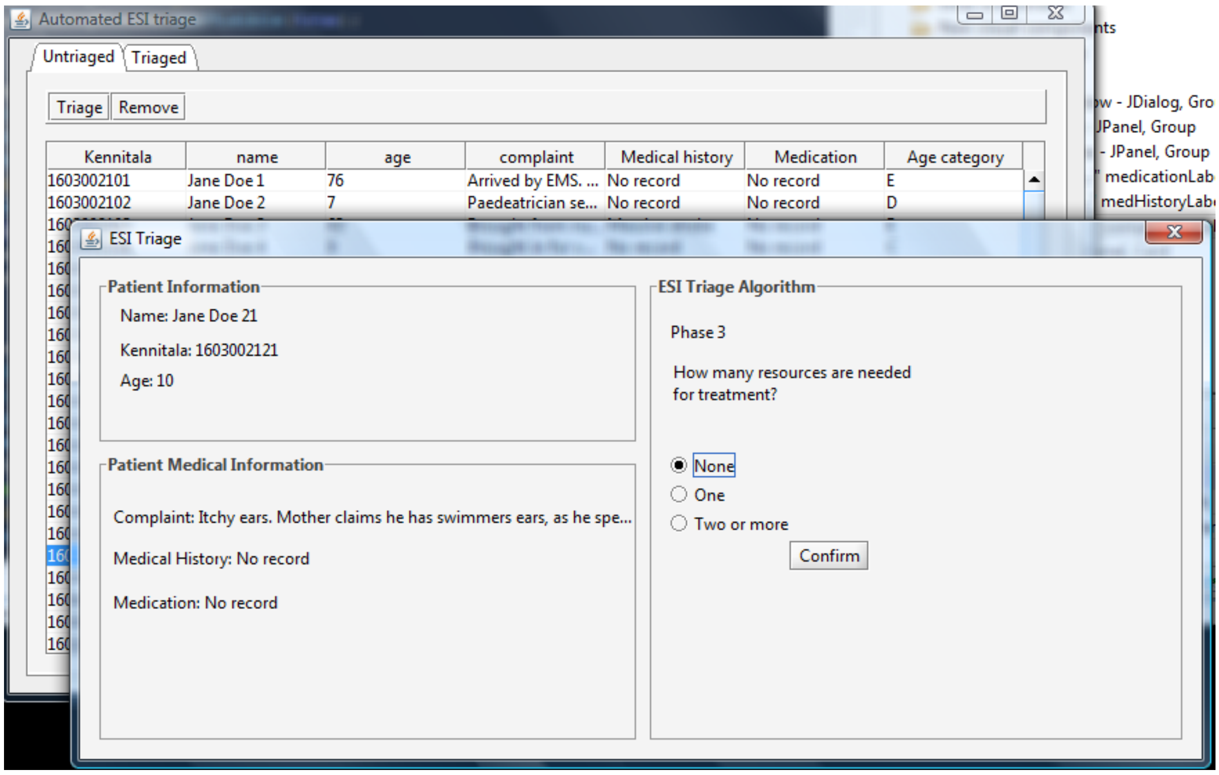
Table 4 - Comparing answers from triage competency cases from chapter 10

In table 4 the result from going through the 30 competency cases from chapter 10 with the ESI triage software. The first column from the left refers to index of the triage competency case from chapter 10. Again, the second column is the correct ESI category for each case given by the handbook. The third column is the result ESI category from the ESI software. The fourth column tells if the answer from the book matches the answer from the software.

Except for cases 10, 15 and 17, the resulting ESI triage categories are the same as the ESI categories given by the handbook. Out of 30 competence cases 27 gave correct result. This demonstrates that in most competency cases the ESI triage software could lead the nurse to a correct conclusion.

However, cases 10, 15 and 17 reveal a flaw in the triage software. In these cases, the patient's age is not given. All the cases have that in common that they are critical emergency cases, in which the patient require immediate attention when he arrives at the emergency department. A likely explanation of the lack of age would be that there was simply not enough time to register all of the patient's information when he arrived. When the age data was omitted, the software crashed as it loaded the prototype patient database, due to an unhandled exception. This should, of course, be amended during any future development.

Additional Observations


Automated ESI triage

Untriaged | Triaged

Triage | Remove

Kennitala	name	age	complaint	Medical history	Medication	Age category
1603002101	Jane Doe 1	76	Arrived by EMS. ...	No record	No record	E
1603002102	Jane Doe 2	7	Paedeatrician se...	No record	No record	D

ESI Triage

Patient Information

Name: Jane Doe 21

Kennitala: 1603002121

Age: 10

Patient Medical Information

Complaint: Itchy ears. Mother claims he has swimmers ears, as he spe...

Medical History: No record

Medication: No record

ESI Triage Algorithm

Phase 3

How many resources are needed for treatment?

[✓ None]
One
Two or more

Confirm

Figure 15 - ESI triage dialog - short labels

I noticed during the test that during the ESI triage dialog, the labels conveying information on the patient being triaged are faulty. This can be seen in figure 14. When complaints, medical history or medication information becomes sufficiently long, the labels used to display those runs out of character space. Since this is often crucial information for the triage it should be a priority to fix the patient medical information panel during future iterations.

Code evaluation

HCI testing

Cognitive walkthrough

Cognitive walkthroughs are a useful way of finding HCI related flaws within a software's user interface. This chapter gives an analysis of the task necessary to complete an ESI triage process in the system at its current state and the cognitive walkthrough conducted by myself in order to detect the flaws in the current user interface. Some recommended changes are then presented to improve the interface given the faults that were captured.

The tasks

Main triage frame

The following are the tasks associated with the main triage frame, the central part of the GUI where the nurse decides which patient to triage.

(1.1)View untriaged patients

To decide which patient should be triaged, the nurse will want to view the list of all currently incoming, untriaged patients. On the main triage frame, there is already a table displaying the previously mentioned patients along with specific attribute related with them.

(1.2) Select a patient

If the nurse has decided what patient to triage next or he/she wants to put a focus on a particular patient on the table (a visual aid for reading the patient attributes) she can highlight a single patient on the untriaged patient table.

(1.3) Start triage

If a patient on the untriaged patient table has been highlighted, the nurse can then initiate the triage process by pressing the triage button located on the interface. This opens up the triage dialog screen and starts the ESI triage process for that patient.

(1.4) Terminate the software

At any given point when the triage program is running, the nurse can terminate the program by pressing the default windows close button on the main triage frame.

ESI triage dialog

The following tasks are associated with the ESI dialog screen, where the nurse will be guided through the ESI V4 triage process in triaging a particular patient.

(1.5) Conduct triage

Once the ESI triage process has been initiated, the nurse will answer a series of questions that are a part of the triage algorithm until it assigns a triage category 1 to 5 to the patient (1 being higest priority, 5 the least). In order to answer these questions the nurse will rely on her skills and knowledge of identifying patient status, both from actual patient feedback as well as the patient information displayed on the triage dialog screen.

(1.5.1) View patient detail

As a part of the decision process associated with the questions asked by the ESI, the nurse will likely have to make a quick study of the patient's medical history, his current/past medication and the patient's stated complaint when he arrived at the patient registration. These can always be seen at the bottom left side of the triage dialog screen. The nurse can also view reminders of the patient's age, gender, kennitala and name as well at the top left of the triage dialog screen.

(1.5.1.1) View patient medical history

In order to evaluate the patients current condition, the nurse often wants to check on the patient's previously recorded medical conditions. The nurse can view this information by looking at the patient medical information panel on the bottom left side of the triage dialog window. It is always visible while the dialog is open.

(1.5.1.2) View patient medication

In order to evaluate the patient's current condition, the nurse wants to check on the patient's medication, both past and present. The nurse can view this information by looking at the patient medical information panel on the bottom left side of the triage dialog window. It is always visible while the dialog is open.

(1.5.1.3) View patient complaint

When a patient enters the hospital's accident- and emergency department, the patient's reason for arriving is recorded. This can be done for example by a secretary in the department 's lobby or recorded by a nurse when paramedics bring in a patient from a medical transport. This is valuable information that the nurse can view on the bottom left side of the triage dialog screen in the patient medical information panel. It is always visible while the dialog is open.

(1.5.2) Answer triage questions

(1.5.2.1) ESI phase 1

In the first panel of the ESI wizard, representing the first phase of the ESI triage process, the nurse is asked: "Does the patient require immediate life-saving intervention?" Using the information presented, the nurse answers either yes or no. This is done by pressing either a "yes" button or a "no" button respectively. If the nurse answers no, the dialog will display ESI phase 2.

(1.5.2.2) ESI phase 2

In the second panel of the ESI wizard, representing the second phase of the ESI triage process, the nurse is asked: "Is the patient in a high-risk situation? Is he confused/lethargic/disoriented? Is he in severe pain or distress?" Using the information presented, the nurse answers with either a single yes or a single no to all of the questions. This is done by pressing either a "yes" button or a "no" button respectively. If the nurse answers no, the dialog will display ESI phase 3.

(1.5.2.3) ESI phase 3

In the third panel of the ESI wizard, representing the third phase of the ESI triage process, the nurse is asked: "How many resources are needed?" Using the information presented, and drawing on experience and knowledge of medical cases, the nurse makes a judgement on how many different medical resources will be needed to treat the patient or make further diagnosis. If the nurse answers "two or more resources", the dialog will display ESI phase 4.

(1.5.2.3.1) Enter resource count

The nurse is presented with a set of radio buttons through which the nurse can input how many different resources the patient will need for further diagnosis or treatment. The choices are: "No resource", "one resource" and "two or more resources". The nurse presses the appropriate radio button.

(1.5.2.3.2) Confirm resource count

After choosing the expected number of resources, the nurse presses the confirm button to confirm his/her choice to the triage process.

(1.5.2.4) ESI phase 4

In the fourth panel of the ESI wizard, representing the fourth phase of the ESI triage process, the nurse is asked: “Are there any danger zone vital signs?” In most cases, the triage nurse measures the patient’s vital signs and inputs them to a standard medical paper form. The vital signs that are significant to this particular ESI phase are heart rate, respiration rate, blood oxygen saturation and body temperature. Body temperature, however, is only required for patient three years or younger. Whether or not a vital sign has a critical value is different depending on the age group of the patient. After confirmation, if no vital sign was detected as critical, the patient is assigned ESI category 3. If any of the signs are critical, the nurse must consider moving the patient to ESI category 2.

(1.5.2.4.1) Input vital signs

The nurse inputs each of the patient’s requested vital sign into that vital signs designated input field. The requested vital signs can be input in any order.

(1.5.2.4.2) Confirm vital signs

After the nurse considers that the vital sign input is correct, he/she presses the confirm button to confirm the values to the ESI process.

(1.5.2.4.2) Consider uptriaging

After confirmation, if at least one vital sign has been detected as critical, the nurse is asked to judge weather or not the patient should stay at ESI category 3, or if the patient’s case is severe enough for him to be moved up to ESI category 2. An input dialog will pop up, stating this question to the nurse and requesting answer. The nurse can press a button for ESI category 3 or ESI category 2. She can also cancel the confirmation by pressing the default close button of the dialog.

(1.5.3) Accept ESI triage score

At the end of each ESI phase, depending of the input of the nurse, the system might assign the patient an ESI category of 1-5 and present the this choice in the form of a information dialog. The nurse then acknowledges this by pressing the OK button which closes the ESI dialog screen, returning to the main triage frame.

(1.5.4) Cancel ESI triage

At any given point when the ESI triage dialog is open for a given patient, the nurse can cancel the ESI triage process by pressing the default windows close button on the dialog screen. This should be possible no matter what phase of the triage is active.

Cognitive Walkthrough

The Questions

For each action required to finish each task, the following four questions are asked in the following order:

Action 1 View incoming patient table

1. Will users be trying to produce whatever effect the action has?

2. Will users see the control (button, menu, switch, etc.) for the action?
3. Once users find the control, will they recognize that it produces the effect they want?
4. After the action is taken, will users understand the feedback they get, so they can go on to the next action with confidence?

The answers are given in either yes or no. Every "no" answer is followed by an explanation.

Main triage frame

(1.1)View untriaged patients

Action 1 View incoming patient table

1. Yes
2. Yes
3. Yes
4. Yes

(1.2) Select a patient

Action 1 Click on patient

1. Yes
2. Yes
3. Yes
4. Yes

(1.3) Start triage

Action 1 Click on patient on untriaged patient table

1. Yes
2. Yes
3. Yes
4. Yes

Action 2 Click on triage button

1. Yes
2. Yes
3. No, the title of the button is a bit ambiguous. It is always enabled even though no patient is selected.

4. No. The button is always enabled even if no patient is selected. No feedback occurs if the user clicks the button with no patient selected.

(1.4) Terminate the software

Action 1 Click on default windows close button

1. Yes
2. Yes
3. Yes
4. Yes, the program terminates

ESI triage dialog

(1.5) Conduct triage

(1.5.1) View patient detail

(1.5.1.1) View patient medical history

Action 1 look at patient medical history label

1. No. Experienced users will, but new users might potentially forget that this information is important.
2. Yes
3. Yes
4. Yes

(1.5.1.2) View patient medication

Action 1 look at patient medication label

1. No. Experienced users will, but new users might potentially forget that this information is important.
2. Yes
3. Yes
4. Yes

(1.5.1.3) View patient complaint

Action 1 look at patient complaint label

1. No. Experienced users will, but new users might potentially forget that this information is important.
2. Yes
3. Yes
4. Yes

(1.5.2) Answer triage questions

(1.5.2.1) ESI phase 1

Action 1 click on “Yes” button

1. Yes
2. Yes
3. Yes
4. No. The feedback message is unclear. The dialog that pops up only states the message “ESI – 1” but does not state to whom that applies to. Can be potentially confuse new users.

Action 2 click on “No” button

1. Yes
2. Yes
3. Yes
4. Yes

(1.5.2.2) ESI phase 2

Action 1 click on “Yes” button

1. Yes
2. Yes
3. Yes
4. No. The feedback message is unclear. The dialog that pops up only states the message “ESI – 2” but does not state to whom that applies to. Can be potentially confuse new users.

Action 2 click on “No” button

1. Yes
2. Yes
3. Yes
4. Yes

(1.5.2.3) ESI phase 3

(1.5.2.3.1) Enter resource count

Action 1 click resource count radio button

1. Yes
2. Yes

3. Yes

4. Yes

(1.5.2.3.2) Confirm resource count

Action 1 click on confirm button

1. Yes

2. Yes

3. No. The button is always enabled so its purpose can potentially be confusing.

4. No. A bug has been found. If no resource count has been selected and the confirmation button is pressed, then ESI category 5 is assigned to the patient. For the cases when an ESI category is assigned, the feedback message is unclear. The dialog that pops up only states the message "ESI – 5" or "ESI – 4" but does not state to whom that applies to. Can potentially confuse new users.

(1.5.2.4) ESI phase 4

(1.5.2.4.1) Input vital signs

Action 1 click on a vital sign spinner text field

1. No. Experienced users will, but there is no message stating that you really have to input the vital signs into the GUI. There is also no explanation for when the temperature spinner is disabled, the result of the patient being older than three.

2. Yes

3. Yes

4. Yes

Action 2 type vital sign value into spinner text field

1. No. Experienced users will, but there is no message stating that you really have to input the vital signs into the GUI.

2. Yes

3. Yes

4. Yes

Action 3 click on increment button of a vital sign spinner

1. No. Experienced users will, but there is no message stating that you really have to input the vital signs into the GUI.

2. Yes

3. Yes

4. Yes

Action 4 click on decrement button of a vital sign spinner

1. No. Experienced users will, but there is no message stating that you really have to input the vital signs into the GUI.

2. Yes

3. Yes

4. Yes

(1.5.2.4.2) Confirm vital signs

Action 1 click on the confirm button

1. No. Experienced users will, but there is no message stating that you really have to input the vital signs into the GUI.

2. Yes

3. No. The button is always enabled so its purpose can potentially be confusing.

4. No. The GUI does not prevent the user from continuing if he has not edited at least one of the vital signs. This can lead to confusion of why the feedback will be the way it will be.

(1.5.2.4.2) Consider uptriaging

Action 1 click on "Uprade to ESI-2" button

1. Yes (However, it would be better if the dialog states which vital sign(s) are in a danger zone.

2. Yes

3. Yes

4. No. The feedback message is unclear. The dialog that pops up only states the message "ESI – 2" but does not state to whom that applies to. Can potentially confuse new users.

Action 2 click on "Keep at ESI-3" button

5. Yes (However, it would be better if the dialog states which vital sign(s) are in a danger zone.

6. Yes

7. Yes

8. No. The feedback message is unclear. The dialog that pops up only states the message "ESI – 3" but does not state to whom that applies to. Can potentially confuse new users.

Action 3 click on default close button

1. Yes.

2. Yes

3. No. It is not stated that clicking the button will cancel the confirmation of the vital signs.

4. No. There is no feedback that states that the confirmation has been aborted and that the user can readjust the vital sign values.

(1.5.3) Accept ESI triage score

Action 1 click on "OK" button

1. Yes

2. Yes

3. Yes

4. No. Currently the message can be considered to be not completely clear. It does not unambiguously state that the ESI score has been assigned to the patient. Also, the removal of the patient from the untriaged patient table has not been implemented.

Action 1 click on default close button

1. Yes

2. Yes

3. No. There is no clue that it will have exactly the same results as clicking the OK button.

4. No. The user might potentially think that he has cancelled the assignation of the ESI triage score, since there is no feedback stating otherwise.

(1.5.4) Cancel ESI triage

Action 1 click on default windows close button

5. Yes.

6. Yes

7. No. The user might want to terminate the program and not just the ESI dialog window.

8. No. There is no feedback that tells the user that the ESI triaging has been aborted, and it has to be started from scratch. There is also no confirmation dialog for confirming if the user really intends to cancel the triage process.

Recommendations

To clarify the purpose of the triage button, change the button text from “triage” to “triage selected patient”. The button should also be disabled at all times except when a patient has been selected on the untriaged patient table.

(1.5.1.1) View patient medical information

In order to highlight the importance of the information found on the patient info panel, let the border of the panel flash red for a brief time.

(1.5.3) Accept ESI triage score

The current message on the information dialog should be changed from “ESI – X” to “The patient has been assigned category ESI – X”, where X is the appropriate ESI category. There should also be a cancel button in order to backtrack in case input confirmation was made by mistake. If either this cancel button or the default windows button is pressed, then there should be a clear message that the assignment of ESI triage category has been cancelled and the user is free to alter the input.

(1.5.2.3.2) Confirm resource count

The confirm button for the confirmation of the medical resource count should be disabled at all times except when a resource count has been selected.

(1.5.2.4.2) Confirm vital signs

On the fourth ESI phase there should be message clearly stating that it is necessary to input all requested vital signs into their appropriate spinners to continue. The message should also state which vital signs are needed.

On the same panel, the confirm button should always be disabled unless all of the input spinners have had their values altered. There should be clear labelling of which vital signs inputs have not been edited.

(1.5.2.4.2) Consider uptriaging

In the dialog asking the user for uptriaging, it should be clearly stated what vital signs where in a danger zone when the input was confirmed.

In the same dialog, there should be a cancel button, which should have the same functionality as pressing the default exit windows exit button. If a user decides to cancel the input confirmation this way, there should be a clear message on the panel stating that the input has been cancelled.

(1.5.4) Cancel ESI triage

To make it clear that the user can cancel the triage process, a button named “Cancel this triage” should be added to the ESI triage dialog screen. Whenever this button or the default windows close button is pressed a confirmation dialog should pop up asking the user if he/she really meant to cancel the triage process.

Conclusion

Personal reflections

Several reasons why I chose this project originally. First of all it seemed like an interesting problem of organization that could be solved by some kind of a sorting algorithm. I originally wanted to see if I could apply my studies of algorithms, from a previous university course, to this particular problem.

It was also very appealing to be developing a solution to something that could have direct benefits to other people. In medical emergencies, every second counts. Any methods applied to slice seconds off the time from when a patient enters an emergency department and gets treatment could save lives. It gave the project the warm glow of a higher purpose to the study of computer science, which felt rather motivating.

While I was not successful in implementing everything that I had originally intended, I did nonetheless manage to create a working prototype of a triage assisting software with functional core functionality. The software is capable of aiding nurses through ESI triage of patients and assign ESI categories to them. Use case evaluation was also very promising. Discounting cases when some input information was missing from the test cases, the software produced correct categories rather consistently.

There were regardless some failures in this project. I did manage to follow the implementation schedule rigorously, resulting in the implementation phase of the development to become bloated. This had detrimental effect on the rest of the development process, shortening the time I had for evaluation of the software.

It would have been better if I had been able to spend larger portion of my development time on the software evaluation. More time spent to get usability feedback from the target users, ED nurses or nurse students would have been invaluable information for improving the software. Some rigorous unit testing would also have been necessary for each class developed in this project, this being a safety critical software.

In hindsight, it might have been better to develop the GUI of the software in Java. Even if Jigloo provides a rather easy to use graphical development environment for developing a Java GUI, adjustments of the code generated by the tool sometimes resulted in bugging the Jigloo development environment. These bugs tended to be hard to locate. If I would be to go through another development cycle with this project, I would want to develop the GUI in a programming language known to be better suited for such tasks, for example Visual Basic.

I consider that I have gained valuable insight and appreciation of the process of emergency medical treatment. I have had the fortune of not yet having to experience the emergency department of a hospital first hand,

I gained more insight into the workings of another profession. It is necessary for software developers to be able to work with people of other professions, such as nurses, in order to solve problems within the profession’s domain.

Skills gained

I was already rather familiar with the Java programming language, which I chose to use to implement this software, from previous courses taken at the university. Nevertheless, up until this project I had only used it in the context of soling relatively small, isolated projects. Through the development of this project I consider that I at least trained myself in applying my knowledge of the Java language to a real-life project, on that exists in an actual profession domain.

As an extension of that, this was the first time I was using Java to connect with another technology platform, in this case a MySQL database. Implementing the Java to MySQL connection was very problematic at first, but worked out in the end, giving me practical insight and concrete idea of how such connectivity works.

As for non-technical skills, many of them are related with doing the research and requirements capture and analysis phases of the project. Emergency medicine was an unfamiliar field of study to me, requiring me to research a domain pretty much from scratch. This is as I understand it, not very uncommon in real-life software development, as the trade often requires you to work in problem domains you might be unfamiliar with.

Future work

The ESI triage system current version could be considered a prototype, as it has not implemented all of the originally planned use cases. It would be useful as a starting ground for a new iteration, one that could implement the rest of the use cases. The next development cycle could use the evaluation information gained from this project to enhance usability, security and effectiveness of the software.

However, It would be interesting to integrate the triage software with an actual hospital database. Interacting with the institution's system, patient data could be retrieved from stored patient medical and medication records. The result from the triage could in return be stored within a special triage database.

Additionally, the triage software could be developed further into the direction of being a training tool for nurses studying the ESI triage algorithm. It could implement tutorials, a help system and tooltip assistance for them to use.

References

Aronsky, D. et al., (2008). *An Integrated Computerized Triage System in the Emergency Department.* [Internet] AMIA. Available at: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2656061/ . [Accessed 6 Desember 2009].

Bennet, S. McRobb, S. Farmer, R. (2006). Object-Oriented Systems Analysis and Design. *The Architecture of the Presentation Layer*. 464-465. McGraw-Hill Education. Maidenhead Berkshire.

Dong, S. L. et al., (2008). *Emergency Triage: Comparing a Novel Computer Triage Program with Standard Triage.* [Online] Society for Academic Emergency Medicine. Available at: http://www3.interscience.wiley.com/cgi-bin/fulltext/119821459/PDFSTART. [Accessed 6 Desember 2009].

Flangan, David. (2000). *Making SQL Queries with JDBC and Displaying Results with Swing*. Available at: http://www.oreillynet.com/pub/a/oreilly/java/news/javaex_1000.html. Last accessed 15 April 2010.

Gilboy, N. et al., (2005) *Emergency Severity Index, Version 4: Implementation Handbook*. Rockville MD: AHRQ Publication.

Golding, D. Wilson, L. & Marwala, T., (2008). *Emergency Centre Organization and Automated Triage System.* [Online] Available at: http://arxiv.org/ftp/arxiv/papers/0810/0810.3671.pdf. [Accessed 7 Desember, 2009].

Iserson, K. V. & Moskop, J. C., (2007). *Triage in Medicine, Part I: Concept, History, and Types.* [Online] Mosby, Inc. Available at: http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6WB0-4KCGHM7-2&_user=5914913&_rdoc=1&_fmt=&_orig=search&_sort=d&_docanchor=&view=c&_rerunOrigin=scholar.google&_acct=C000068840&_version=1&_urlVersion=0&_userid=5914913&md5=6de9ba578ec4a71a15d93d0bafd4e5e8. [Accessed 6 Desember 2009].

Levin, S. et al., (2006). *The Effects of Computerized Triage on Nurse Work Behavior.* [Internet] Amia. Available at: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1839482/. [Accessed 6 Desember 2009].

Appendix A - Code listing

NoListener

```
package triagecontroller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * The listener interface for receiving no events.
 * The class that is interested in processing a no
 * event implements this interface, and the object created
 * with that class is registered with a component using the
 * component's <code>addNoListener</code> method. When
 * the no event occurs, that object's appropriate
 * method is invoked.
 *
 * @see NoEvent
 */
public class NoListener implements ActionListener {
      /** The access controller to this actionlistener. */
      private TriageController controllerInterface;

      /**
       * Instantiates a new no listener.
       *
       * @param controllerInterfaceIn the controller interface
       */
      public NoListener(final TriageController controllerInterfaceIn) {
            this.controllerInterface = controllerInterfaceIn;
      }

      /** {@inheritDoc} */
      public final void actionPerformed(final ActionEvent e) {
            this.controllerInterface.getTriage().recieveAnswerNo();
      }
```

RadioButtonListener

```
package triagecontroller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * The listener interface for receiving radioButton events.
 * The class that is interested in processing a radioButton
 * event implements this interface, and the object created
 * with that class is registered with a component using the
 * component's <code>addRadioButtonListener</code> method. When
 * the radioButton event occurs, that object's appropriate
 * method is invoked.
 *
 * @see RadioButtonEvent
 */
public class RadioButtonListener implements ActionListener {

	/** The access controller to this actionlistener. */
	private TriageController controllerInterface;

	/** The resource number. */
	private int resourceNumber;

	/**
	 * Instantiates a new radio button listener.
	 *
	 * @param controllerInterfaceIn the controller interface
	 */
	public RadioButtonListener(final TriageController 
controllerInterfaceIn) {
		this.controllerInterface = controllerInterfaceIn;
		this.resourceNumber = 0;
	}

	/** {@inheritDoc} */
	public final void actionPerformed(final ActionEvent e) {
		// medical resource count
		resourceNumber = Integer.parseInt(e.getActionCommand());
		System.out.println("resources: " + resourceNumber);
	}

	/**
	 * Gets the resource number.
	 *
	 * @return the resource number
	 */
	public final int getResourceNumber() {
		return resourceNumber;
	}
}}
```

ResourceConfirmationListener

```
package triagecontroller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * The listener interface for receiving resourceConfirmation events.
 * The class that is interested in processing a resourceConfirmation
 * event implements this interface, and the object created
 * with that class is registered with a component using the
 * component's <code>addResourceConfirmationListener</code> method. When
 * the resourceConfirmation event occurs, that object's appropriate
 * method is invoked.
 *
 * @see ResourceConfirmationEvent
 */
public class ResourceConfirmationListener implements ActionListener {
      /** The access controller to this actionlistener. */
      private TriageController controllerInterface;  //

      /**
       * Instantiates a new resource confirmation listener.
       *
       * @param controllerInterfaceIn the controller interface
       */
      public ResourceConfirmationListener(final TriageController

      controllerInterfaceIn) {
            this.controllerInterface = controllerInterfaceIn;
      }

      /** {@inheritDoc} */
      public final void actionPerformed(final ActionEvent e) {
            int resourceCount = this.controllerInterface

      .getRadioButtonListener().getResourceNumber();
            this.controllerInterface.getTriage()
                  .recieveResourceCount(resourceCount);
      }
}
```

TableSelectionListener

```
package triagecontroller;

import javax.swing.ListSelectionModel;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import triagemodel.ESITriage;

/**
 * The listener interface for receiving tableSelection events.
 * The class that is interested in processing a tableSelection
 * event implements this interface, and the object created
 * with that class is registered with a component using the
 * component's <code>addTableSelectionListener</code> method. When
 * the tableSelection event occurs, that object's appropriate
 * method is invoked.
 *
 * @see TableSelectionEvent
 */
public class TableSelectionListener implements ListSelectionListener  {

      /** The triage. */
      private ESITriage triage;
      /** The access controller to this actionlistener. */
      private TriageController controllerInterface;  //

       /**
       * Instantiates a new table selection listener.
       *
       * @param controllerInterfaceIn the controller interface in
       */
      public TableSelectionListener(final TriageController
                                                     controllerInterfaceIn)
{
             this.controllerInterface = controllerInterfaceIn;
       }

       /** {@inheritDoc} */
     public final void valueChanged(final ListSelectionEvent e) {
       System.out.println("event caught!");
       ListSelectionModel lsm = (ListSelectionModel) e.getSource();
       int selectedIndex = lsm.getMinSelectionIndex();
       this.controllerInterface.getTriage()
                                         .setPatIndex(selectedIndex);
     }
}
```

TriageController

```
package triagecontroller;

import triagemodel.ESITriage;
import triageview.TriageFrame;

/**
 * The Class TriageController.
 */
public class TriageController {

	/** listens for "yes" answers from gui. */
	private YesListener yesListener;

	/** listens for "no" answers from gui. */
	private NoListener noListener;

	/** listens for table selection from gui. */
	private TableSelectionListener selectionListener;

	/** listen for triage initiation action from gui. */
	private TriageInitiationListener triageInitListener;

	/** listen for confirmation of number of resources. */
	private ResourceConfirmationListener resourceListener;

	/** listens for radio buttons clicked in gui. */
	private RadioButtonListener radioButtonListener;

	/** listens for confirmation of vital signs. */
	private VitalsConfirmListener confirmListener;

	/** the triage view (gui). */
	private TriageFrame triageFrame;

	/** the triage logic (model). */
	private ESITriage triage;

	/**
	 * Instantiates a new triage controller.
	 *
	 * @param triageFrameIn the triage frame in
	 * @param triageIn the triage in
	 */
	public TriageController(final TriageFrame triageFrameIn,
				    final ESITriage triageIn) {
		this.triageFrame = triageFrameIn;
		this.triage = triageIn;

		initiateListeners();
	}

	/**
	 * Initiate listeners.
	 */
	public final void initiateListeners() {
		selectionListener = new TableSelectionListener(this);
		yesListener = new YesListener(this);
		noListener = new NoListener(this);
		triageInitListener = new TriageInitiationListener(this);
		resourceListener = new ResourceConfirmationListener(this);
		radioButtonListener = new RadioButtonListener(this);
		confirmListener = new VitalsConfirmListener(this);
	}

	/**
	 * Gets the triage.
	 *
	 * @return the triage
	 */
	public final ESITriage getTriage() {
		return this.triage;
	}

	/**
	 * Gets the yes listener.
	 *
	 * @return the yes listener
	 */
	public final YesListener getYesListener() {
		return yesListener;
	}

	/**
	 * Sets the yes listener.
	 *
	 * @param yesListenerIn the new yes listener
	 */
	public final void setYesListener(final YesListener yesListenerIn) {
		this.yesListener = yesListenerIn;
	}

	/**
	 * Gets the selection listener.
	 *
	 * @return the selection listener
	 */
	public final TableSelectionListener getSelectionListener() {
		return selectionListener;
	}

	/**
	 * Sets the selection listener.
	 *
	 * @param selectionListenerIn the new selection listener
	 */
	public final void setSelectionListener(final TableSelectionListener

	selectionListenerIn) {
		this.selectionListener = selectionListenerIn;
	}

	/**
	 * Gets the no listener.
	 *
	 * @return the no listener
	 */
	public final NoListener getNoListener() {
		return noListener;
	}
```

```
      /**
       * Sets the no listener.
       *
       * @param noListenerIn the new no listener
       */
      public final void setNoListener(final NoListener noListenerIn) {
            this.noListener = noListenerIn;
      }

      /**
       * Gets the triage init listener.
       *
       * @return the triage init listener
       */
      public final TriageInitiationListener getTriageInitListener() {
            return triageInitListener;
      }

      /**
       * Sets the triage init listener.
       *
       * @param triageInitListenerIn the new triage init listener
       */
      public final void setTriageInitListener(final
TriageInitiationListener

      triageInitListenerIn) {
            this.triageInitListener = triageInitListenerIn;
      }

      /**
       * Gets the triage frame.
       *
       * @return the triage frame
       */
      public final TriageFrame getTriageFrame() {
            return triageFrame;
      }

      /**
       * Sets the triage frame.
       *
       * @param triageFrameIn the new triage frame
       */
      public final void setTriageFrame(final TriageFrame triageFrameIn) {
            this.triageFrame = triageFrameIn;
      }

      /**
       * Terminates the current triage process.
       */
      public final void endCurrentTriage() {

      }

      /**
       * Gets the resource listener.
       *
       * @return the resource listener
       */
      public final ResourceConfirmationListener getResourceListener() {
            return resourceListener;
      }

      /**
       * Sets the resource listener.
       *
       * @param resourceListenerIn the new resource listener
       */
      public final void setResourceListener(final
ResourceConfirmationListener

      resourceListenerIn) {
            this.resourceListener = resourceListenerIn;
      }

      /**
       * Gets the radio button listener.
       *
       * @return the radio button listener
       */
      public final RadioButtonListener getRadioButtonListener() {
            return radioButtonListener;
      }

      /**
       * Sets the radio button listener.
       *
       * @param radioButtonListenerIn the new radio button listener
       */
      public final void setRadioButtonListener(final RadioButtonListener

      radioButtonListenerIn) {
            this.radioButtonListener = radioButtonListenerIn;
      }

      /**
       * Gets the confirm listener.
       *
       * @return the confirm listener
       */
      public final VitalsConfirmListener getConfirmListener() {
            return confirmListener;
      }

      /**
       * Sets the confirm listener.
       *
       * @param confirmListenerIn the new confirm listener
       */
      public final void setConfirmListener(final VitalsConfirmListener

      confirmListenerIn) {
            this.confirmListener = confirmListenerIn;
      }
}
```

TriageInitiationListener

```
package triagecontroller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * The listener interface for receiving triageInitiation events.
 * The class that is interested in processing a triageInitiation
 * event implements this interface, and the object created
 * with that class is registered with a component using the
 * component's <code>addTriageInitiationListener</code> method. When
 * the triageInitiation event occurs, that object's appropriate
 * method is invoked.
 *
 * @see TriageInitiationEvent
 */
public class TriageInitiationListener implements ActionListener {
      /** The access controller to this actionlistener. */
      private TriageController controllerInterface;

      /**
       * Instantiates a new triage initiation listener.
       *
       * @param controllerInterfaceIn the controller interface
       */
      public TriageInitiationListener(final TriageController
                                                     controllerInterfaceIn)
{
            this.controllerInterface = controllerInterfaceIn;
      }

      /** {@inheritDoc} */
      public final void actionPerformed(final ActionEvent e) {
            if (this.controllerInterface.getTriage().getPatIndex() >= 0) {
                  // only initiate dialog if a patient has been selected

      this.controllerInterface.getTriageFrame().resetTriageInputs();

      this.controllerInterface.getTriageFrame().displayTriageDialog();
                  this.controllerInterface.getTriage().triagePatient();
            }
      }
}
```

VitalsConfirmListener

```
package triagecontroller;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * The listener interface for receiving vitalsConfirm events.
 * The class that is interested in processing a vitalsConfirm
 * event implements this interface, and the object created
 * with that class is registered with a component using the
 * component's <code>addVitalsConfirmListener</code> method. When
 * the vitalsConfirm event occurs, that object's appropriate
 * method is invoked.
 *
 * @see VitalsConfirmEvent
 */
public class VitalsConfirmListener implements ActionListener {
      /** The access controller to this actionlistener. */
      private TriageController controllerInterface;

      /**
       * Instantiates a new vitals confirm listener.
       *
       * @param controllerInterfaceIn the controller interface
       */
      public VitalsConfirmListener(final TriageController
controllerInterfaceIn) {
            this.controllerInterface = controllerInterfaceIn;
      }

      /** {@inheritDoc} */
      public final void actionPerformed(final ActionEvent e) {
            // retrieve vital signs from GUI
            double[] vitals = this.controllerInterface

      .getTriageFrame().retrieveVitals();
            this.controllerInterface.getTriage().recieveVitals(vitals);
            this.controllerInterface.getTriage().processVitals();
      }
}

package triagecontroller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * The listener interface for receiving yes events.
 * The class that is interested in processing a yes
 * event implements this interface, and the object created
 * with that class is registered with a component using the
 * component's <code>addYesListener</code> method. When
 * the yes event occurs, that object's appropriate
 * method is invoked.
 *
 * @see YesEvent
 */
public class YesListener implements ActionListener {
	/** The access controller to this actionlistener. */
	private TriageController controllerInterface;

	/**
	 * Instantiates a new yes listener.
	 *
	 * @param controllerInterfaceIn the controller interface
	 */
	public YesListener(final TriageController controllerInterfaceIn) {
		this.controllerInterface = controllerInterfaceIn;
	}

	/** {@inheritDoc} */
	public final void actionPerformed(final ActionEvent e) {
		this.controllerInterface.getTriage().recieveAnswerYes();
	}

}
```

ESITriage

```
package triagemodel;

import java.util.ArrayList;
import java.util.Observable;

/**
 * The Class ESITriage. An interface class for the triagemodel
 * component to the view and controller.
 * Implements Observable interface, propogating updates of the
 * model to any classes registered as its observers.
 */
public class ESITriage extends Observable {

      /** The Constant KENNITALA_INDEX. */
      public static final int KENNITALA_INDEX = 0;

      /** The Constant NAME_INDEX. */
      public static final int NAME_INDEX = 1;

      /** The Constant AGE_INDEX. */
      public static final int AGE_INDEX = 2;

      /** The Constant COMPLAINTS_INDEX. */
      public static final int COMPLAINTS_INDEX = 3;

      /** The Constant MEDICAL_HISTORY_INDEX. */
      public static final int MEDICAL_HISTORY_INDEX = 4;

      /** The Constant MEDICATION_INDEX. */
      public static final int MEDICATION_INDEX = 5;

      /** The Constant AGEGROUP_INDEX. */
      public static final int AGEGROUP_INDEX = 6;

      /** The number of patient's recorded characteristics. */
      public static final int PATIENT_ATTRIBUTE_COUNT = 8;

      /** The number of patient's triage necessary vital signs. */
      public static final int VITAL_COUNT = 4;

      /** The untriaged patients. */
      private ArrayList<Patient> untriagedPatients;

      /** The pat index. */
      private int patIndex;

      /** The triage dialog. */
      private TriageDialog triageDialog;

      /**
       * Instantiates a new eSI triage.
       */
      public ESITriage() {
            initialize();
      }

      /**
       * Initialize.
       */
      final void initialize() {
            // no patient is selected initially
            patIndex = -1;
            untriagedPatients = new ArrayList<Patient>();
            triageDialog = null;

            // load patient data into the program
            updatePatients();
      }

      /**
       * Update patients.
       */
      void updatePatients() {
            // has not been implemented
      }

      /**
       * Populate.
       *
       * @param patientModel the patient model
       */
      public final void populate(final ResultSetTableModel patientModel) {
            // the number of untriaged patients
            int numberOfPatients = patientModel.getRowCount();
            if (numberOfPatients > 0) {
                  Patient patient;
                  String kennitala = "";
                  String name = "";
                  int age = 0;
                  String complaints = "";
                  String medicalHistory = "";
                  String medication = "";
                  String ageGroup = "";
                  int index = 0; // the column index
                  for (int i = 0; i < numberOfPatients; i++) {
                        // acquire patient information
                        kennitala = (String) (patientModel
                                          .getValueAt(i,
KENNITALA_INDEX));
                        name = (String) (patientModel.getValueAt(i,
NAME_INDEX));
                        // parsing the string to int
                        age = Integer.parseInt((String)
                                                      (patientModel
                                                      .getValueAt(i,
AGE_INDEX)));
                        complaints = (String) (patientModel
                                                      .getValueAt(i,
COMPLAINTS_INDEX));
                        medicalHistory = (String)
                                                (patientModel
                                                      .getValueAt(i,
MEDICAL_HISTORY_INDEX));
                        medication = (String)
                                          (patientModel
                                                .getValueAt(i,
MEDICATION_INDEX));
                        ageGroup = (String)
                                    (patientModel
                                                .getValueAt(i,
AGEGROUP_INDEX));

                        // create and store a patient
                        patient = new Patient(kennitala, name, age,
                                                        complaints,
medicalHistory,
                                                        medication,
ageGroup);
                        untriagedPatients.add(patient);
                  }
            }
      }

      /**
       * Triage patient.
       */
      public final void triagePatient() {
            if (patIndex >= 0) {
                  // only start the dialog if a patient is selected
                  Patient selectedPatient =
untriagedPatients.get(patIndex);
                  triageDialog = new TriageDialog(selectedPatient, this);
                  this.broadCastUpdate();
            } else  {
                  // do nothing
                  System.out.println("No patient selected!");
            }

      }

      // pass an answer to the currently active dialog
      /**
       * Recieve answer yes.
       */
      public final void recieveAnswerYes() {
            if (triageDialog != null) {
                  // only recieve answer if a dialog has been initiated
                  triageDialog.recieveAnswerYes();
            }
      }

      /**
       * Recieve answer no.
       */
      public final void recieveAnswerNo() {
            if (triageDialog != null) {
                  // only recieve answer if a dialog has been initiated
                  triageDialog.recieveAnswerNo();
            }
      }

      /**
       * Retrieve all vital sign states. Tells if any vitals are critical.
       *
       * @return the boolean[]
       */
      public final boolean[] retrieveVitalStates() {
            // fetch vital sign data from
            boolean[] vitalStates = new boolean[VITAL_COUNT];
            if (this.triageDialog != null) {
                  vitalStates = this.triageDialog.retrieveVitalStates();
            }
            return vitalStates;

      }

      /**
       * Process vitals.
       */
      public final void processVitals() {
            if (this.triageDialog != null) {
                  this.triageDialog.processVitals();
            }
      }

      /**
       * Recieve resource count.
       *
       * @param resourceCount the resource count
       */
      public final void recieveResourceCount(final int resourceCount) {
            triageDialog.processResourceCount(resourceCount);
      }

      /**
       * Recieve vitals.
       *
       * @param vitals the vitals
       */
      public final void recieveVitals(final double[] vitals) {
            this.triageDialog.recieveVitals(vitals);
      }

      /**
       * Broad cast update.
       */
      public final void broadCastUpdate() {
            this.setChanged();
            this.notifyObservers();
      }

      /**
       * Retrieve proposed score.
       *
       * @return the int
       */
      public final int retrieveProposedScore() {
            if (triageDialog != null) {
                  int proposedScore =
triageDialog.getProposedTriageScore();
                  return proposedScore;
            } else {
                  return -1;
            }
      }

      /**
       * Retrieve selectet patient info.
       *
       * @return the string[]
       */
      public final String[] retrieveSelectetPatientInfo() {
            // fetch the currently selected patient
            if (patIndex >= 0) {
                  // only fetch info if a patient has been selected
                  Patient selectedPatient =
this.untriagedPatients.get(this.patIndex);
                  String[] output = new String[PATIENT_ATTRIBUTE_COUNT];
                  int attributeIndex = 0;
                  output[attributeIndex++] = selectedPatient.getName();
                  output[attributeIndex++] =
selectedPatient.getKennitala();
                  output[attributeIndex++] = selectedPatient.getName();
                  // integer converted into string
                  output[attributeIndex++] = "" + selectedPatient.getAge();
                  output[attributeIndex++] =
selectedPatient.getComplaints();
                  output[attributeIndex++] =
selectedPatient.getMedicalHistory();
                  output[attributeIndex++] =
selectedPatient.getMedication();
                  output[attributeIndex++] = selectedPatient.getAgeGroup();

                  return output;
            } else {
                  // no patient has been selected
                  String[] output = new String[PATIENT_ATTRIBUTE_COUNT];
                  for (int i = 0; i < PATIENT_ATTRIBUTE_COUNT; i++) {
                        // set output as undefined
                        output[i] = "undefined";
                  }
                  return output;
            }
      }

      /**
       * Retrieve current phase.
       *
       * @return the int
       */
      public final int retrieveCurrentPhase() {
            if (triageDialog != null) {
                  int phase = triageDialog.getCurrentPhase();
                  return phase;
            } else {
                  return -1;
            }
      }

      /**
       * Gets the pat index.
       *
       * @return the pat index
       */
      public final int getPatIndex() {
            return patIndex;
      }

      /**
       * Sets the pat index.
       *
       * @param patIndexIn the new pat index
```

```
         */
        public final void setPatIndex(final int patIndexIn) {
                this.patIndex = patIndexIn;
                System.out.println(untriagedPatients.get(patIndexIn));
        }

}
```

Patient

```java
package triagemodel;

/**
 * The Class Patient. Reptresent a patient waiting in
 * ED to be triaged.
 */
public class Patient {

	/** a patient's ID. */
	private String kennitala;

	/** patient's full name. */
	private String name;

	/** patient's aquired. */
	private int age;

	/** patient's description of his ailment. */
	private String complaints;

	/** patient's previously registered medical occurrences. */
	private String medicalHistory;

	/** patient's current medication. */
	private String medication;

	/** patient's last measured heart rate. */
	private double heartRate;

	/** patient's last measured respiration rate. */
	private double respiratoryRate;

	/** patient's last measured oxygen saturation. */
	private double oxySat;

	/** patient's last measured body temperature. */
	private double temp;

	/** age group of the patient. */
	private String ageGroup;

	/**
	 * Instantiates a new patient.
	 *
	 * @param kennitalaIn the kennitala input
	 * @param nameIn the name input
	 * @param ageIn the age input
	 * @param complaintsIn the complaints input
	 * @param medicalHistoryIn the medical history input
	 * @param medicationIn the medication input
	 * @param ageGroupIn the age group input
	 */
	public Patient(final String kennitalaIn, final String nameIn,
			final int ageIn, final String complaintsIn,
			final String medicalHistoryIn, final String
medicationIn,
			final String ageGroupIn) {

		this.kennitala = kennitalaIn;
		this.name = nameIn;
		this.age = ageIn;
		this.complaints = complaintsIn;
		this.medicalHistory = medicalHistoryIn;
		this.medication = medicationIn;
		this.ageGroup = ageGroupIn;
	}

	/** {@inheritDoc} */
	public final String toString() {
		return "Patient [age=" + age + ", complaints=" + complaints
				+ ", kennitala=" + kennitala + ", medicalHistory="
				+ medicalHistory + ", medication=" + medication +
", name="
				+ name + "]";
	}

	/** {@inheritDoc} */
	public final int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result;
		// check if a kennitala has been assigned
		if (kennitala == null) {
			result = result + 0;
		} else {
			result = result + kennitala.hashCode();
		}
		return result;
	}

	/** {@inheritDoc} */
	public final boolean equals(final Object obj) {
		if (this == obj) {
			return true;
		}
		if (obj == null) {
			return false;
		}
		if (getClass() != obj.getClass()) {
			return false;
		}
		Patient other = (Patient) obj;
		if (kennitala == null) {
			if (other.kennitala != null) {
				return false;
			}
		} else if (!kennitala.equals(other.kennitala)) {
			return false;
		}
		return true;
	}

	/**
	 * Gets the kennitala.
	 *
	 * @return the kennitala
	 */
	public final String getKennitala() {
		return kennitala;
	}
```

```
    /**
     * Sets the kennitala.
     *
     * @param kennitalaIn the new kennitala
     */
    public final void setKennitala(final String kennitalaIn) {
          this.kennitala = kennitalaIn;
    }

    /**
     * Gets the name.
     *
     * @return the name
     */
    public final String getName() {
          return name;
    }

    /**
     * Sets the name.
     *
     * @param nameIn the new name
     */
    public final void setName(final String nameIn) {
          this.name = nameIn;
    }

    /**
     * Gets the age.
     *
     * @return the age
     */
    public final int getAge() {
          return age;
    }

    /**
     * Sets the age.
     *
     * @param ageIn the new age
     */
    public final void setAge(final int ageIn) {
          this.age = ageIn;
    }

    /**
     * Gets the complaints.
     *
     * @return the complaints
     */
    public final String getComplaints() {
          return complaints;
    }

    /**
     * Sets the complaints.
     *
     * @param complaintsIn the new complaints
     */
    public final void setComplaints(final String complaintsIn) {
```

```
		this.complaints = complaintsIn;
	}

	/**
	 * Gets the medical history.
	 *
	 * @return the medical history
	 */
	public final String getMedicalHistory() {
		return medicalHistory;
	}

	/**
	 * Sets the medical history.
	 *
	 * @param medicalHistoryIn the new medical history
	 */
	public final void setMedicalHistory(final String medicalHistoryIn) {
		this.medicalHistory = medicalHistoryIn;
	}

	/**
	 * Gets the medication.
	 *
	 * @return the medication
	 */
	public final String getMedication() {
		return medication;
	}

	/**
	 * Sets the medication.
	 *
	 * @param medicationIn the new medication
	 */
	public final void setMedication(final String medicationIn) {
		this.medication = medicationIn;
	}

	/**
	 * Gets the heart rate.
	 *
	 * @return the heart rate
	 */
	public final double getHeartRate() {
		return heartRate;
	}

	/**
	 * Sets the heart rate.
	 *
	 * @param vitals the new heart rate
	 */
	public final void setHeartRate(final double vitals) {
		this.heartRate = vitals;
	}

	/**
	 * Gets the respiratory rate.
	 *
	 * @return the respiratory rate
 */
public final double getRespiratoryRate() {
      return respiratoryRate;
}

/**
 * Sets the respiratory rate.
 *
 * @param vitals the new respiratory rate
 */
public final void setRespiratoryRate(final double vitals) {
      this.respiratoryRate = vitals;
}

/**
 * Gets the oxy sat.
 *
 * @return the oxy sat
 */
public final double getOxySat() {
      return oxySat;
}

/**
 * Sets the oxy sat.
 *
 * @param vitals the new oxy sat
 */
public final void setOxySat(final double vitals) {
      this.oxySat = vitals;
}

/**
 * Gets the temp.
 *
 * @return the temp
 */
public final double getTemp() {
      return temp;
}

/**
 * Sets the temp.
 *
 * @param vitals the new temp
 */
public final void setTemp(final double vitals) {
      this.temp = vitals;
}

/**
 * Gets the age group.
 *
 * @return the age group
 */
public final String getAgeGroup() {
      return ageGroup;
}

/**
 * Sets the age group.
```

```
         *
         * @param ageGroupIn the new age group
         */
        public final void setAgeGroup(final String ageGroupIn) {
                this.ageGroup = ageGroupIn;
        }
}
```

TriageDialog

```
package triagemodel;

/**
 * The Class TriageDialog. This is the implementation
 * of the Emergency Severity Index version 4 triage.
 */
public class TriageDialog {

      /** The number of considered vital signs in ESI. */
      public static final int VITAL_SIGN_COUNT = 4;

      /** The Constant OXY_SATURATION_LIMIT. */
      public static final double OXY_SATURATION_LIMIT = 0.92d;

      /** The Constant UNLIMITED_TEMPERATURE. */
      public static final double UNLIMITED_TEMPERATURE = Double.MAX_VALUE;

      /** The Constant HR_LIMIT_A. */
      public static final double HR_LIMIT_A = 180.0d;

      /** The Constant RR_LIMIT_A. */
      public static final double RR_LIMIT_A = 50.0d;

      /** The Constant TEMP_LIMIT_A. */
      public static final double TEMP_LIMIT_A = 38.0d;

      /** The Constant HR_LIMIT_B. */
      public static final double HR_LIMIT_B = 180.0d;

      /** The Constant RR_LIMIT_B. */
      public static final double RR_LIMIT_B = 50.0d;

      /** The Constant TEMP_LIMIT_B. */
      public static final double TEMP_LIMIT_B = 38.0d;

      /** The Constant HR_LIMIT_C. */
      public static final double HR_LIMIT_C = 160.0d;

      /** The Constant RR_LIMIT_C. */
      public static final double RR_LIMIT_C = 40.0d;

      /** The Constant TEMP_LIMIT_C. */
      public static final double TEMP_LIMIT_C = 39.0d;

      /** The Constant HR_LIMIT_D. */
      public static final double HR_LIMIT_D = 140.0d;

      /** The Constant RR_LIMIT_D. */
      public static final double RR_LIMIT_D = 30.0d;

      /** The Constant TEMP_LIMIT_D. */
      public static final double TEMP_LIMIT_D = Double.MAX_VALUE;

      /** The Constant HR_LIMIT_E. */
      public static final double HR_LIMIT_E = 100.0d;

      /** The Constant RR_LIMIT_E. */
      public static final double RR_LIMIT_E = 20.0d;
```

```
/** The Constant TEMP_LIMIT_E. */
public static final double TEMP_LIMIT_E = Double.MAX_VALUE;

/** ESI category 1 - highest priority. */
public static final int ESI_CATEGORY_1 = 1;

/** ESI category 2. */
public static final int ESI_CATEGORY_2 = 2;

/** ESI category 3. */
public static final int ESI_CATEGORY_3 = 3;

/** ESI category 4. */
public static final int ESI_CATEGORY_4 = 4;

/** ESI category 5 - lowest priority. */
public static final int ESI_CATEGORY_5 = 5;

/** ESI phase A - is patient dying. */
public static final int ESI_PHASE_A = 1;

/** ESI phase B - is patient in high risk. */
public static final int ESI_PHASE_B = 2;

/** ESI phase C - how many resources. */
public static final int ESI_PHASE_C = 3;

/** ESI phase D - any critical vitals. */
public static final int ESI_PHASE_D = 4;

/** the patient being triaged. */
private Patient patient = null;

/** current phase of the triage algorithm. */
private int currentPhase = 1;

/** the triage management class. */
private ESITriage triageModel = null;

/** triage score proposed by the ESI algorithm. */
private int proposedTriageScore = -1;

/** allocated triage score. */
private int triageScore = -1;

/** number of resources needed for treatment. */
private int resourceCount = -1;

/** heart rate danger zone limit. */
private double heartRateLimit;

/** respiration rate danger zone limit. */
private double respiratoryRateLimit;

/** oxygen saturation danger zone limit. */
private double oxySatLimit;

/** body temperature danger zone limit. */
private double tempLimit;

/** The heart rate critical value. */
```

```
private boolean heartRateCritical = false;

/** The respiratory rate critical value. */
private boolean respiratoryRateCritical = false;

/** The oxy sat critical value. */
private boolean oxySatCritical = false;

/** The temp critical value. */
private boolean tempCritical = false;

/**
 * Instantiates a new triage dialog.
 *
 * @param patientIn the patient
 * @param triageModelIn the triage model
 */
public TriageDialog(final Patient patientIn,
                              final ESITriage triageModelIn) {
      this.patient = patientIn;
      this.triageModel = triageModelIn;
      this.initVitalsignLimits();
}

/**
 * Aquires input from the gui that is relevant to the current phase.
 *
 * @return true, if successful
 */
final boolean fetchInput() {
      boolean success = false;

      return success;
}

/**
 * Execute phase1, is the patient dying?.
 */
final void executePhase1() {
      if (this.currentPhase == 1) {
            // set the proposed score as the highest triage level
            proposedTriageScore = 1;
            triageModel.broadCastUpdate();
      }
}

/**
 * Initiates the critical vital sign limits.
 */
private void initVitalsignLimits() {
      // age group of the patient
      String ageGroup = this.patient.getAgeGroup();
      if (ageGroup.equalsIgnoreCase("A")) {
            //patient is 0 - 28 days old

            /*
            * heart rate danger zone limit
            * respiration rate danger zone limit
            * oxygen saturation danger zone limit
            * body temperature danger zone limit
            */
            heartRateLimit = HR_LIMIT_A;
            respiratoryRateLimit = RR_LIMIT_A;
            oxySatLimit = OXY_SATURATION_LIMIT;
            tempLimit = TEMP_LIMIT_A;
      } else if (ageGroup.equalsIgnoreCase("B")) {
            // patient is 1 - 3 months old

            /*
            * heart rate danger zone limit
            * respiration rate danger zone limit
            * oxygen saturation danger zone limit
            * body temperature danger zone limit
            */
            heartRateLimit = HR_LIMIT_B;
            respiratoryRateLimit = RR_LIMIT_B;
            oxySatLimit = OXY_SATURATION_LIMIT;
            tempLimit = TEMP_LIMIT_B;
      } else if (ageGroup.equalsIgnoreCase("C")) {
            // patient is 3 months - 3 years old

            /*
            * heart rate danger zone limit
            * respiration rate danger zone limit
            * oxygen saturation danger zone limit
            * body temperature danger zone limit
            */
            heartRateLimit = HR_LIMIT_C;
            respiratoryRateLimit = RR_LIMIT_C;
            oxySatLimit = OXY_SATURATION_LIMIT;
            tempLimit = TEMP_LIMIT_C;
      } else if (ageGroup.equalsIgnoreCase("D")) {
            // patient is 3-8 years old

            /*
            * heart rate danger zone limit
            * respiration rate danger zone limit
            * oxygen saturation danger zone limit
            * body temperature not a concern in this age group
            */
            heartRateLimit = HR_LIMIT_D;
            respiratoryRateLimit = RR_LIMIT_D;
            oxySatLimit = OXY_SATURATION_LIMIT;
            tempLimit = TEMP_LIMIT_D;
      } else if (ageGroup.equalsIgnoreCase("E")) {
            // patient is older than 8 years old

            /*
            * heart rate danger zone limit
            * respiration rate danger zone limit
            * oxygen saturation danger zone limit
            * body temperature not a concern in this age group
            */
            heartRateLimit = HR_LIMIT_E;
            respiratoryRateLimit = RR_LIMIT_E;
            oxySatLimit = OXY_SATURATION_LIMIT;
            tempLimit = TEMP_LIMIT_E;
      }
}

/**
 * Recieve answer yes.
```

```
         */
        final void recieveAnswerYes() {
                if (currentPhase == 1) {
                // the patient needs an immediate life saving intervention
                        proposedTriageScore = 1;
                        this.triageModel.broadCastUpdate();
                } else if (currentPhase == 2) {
                // the patient is in a high risk situation
                        proposedTriageScore = 2;
                        this.triageModel.broadCastUpdate();
                }
        }

        /**
         * Recieve answer no.
         */
        public final void recieveAnswerNo() {
                if (currentPhase == ESI_PHASE_A) {
                        // the patient does not need immediate life saving
intervention
                        currentPhase = ESI_PHASE_B;
                        this.triageModel.broadCastUpdate();
                } else if (currentPhase == ESI_PHASE_B) {
                        // the patient is not in a high risk situation
                        currentPhase = ESI_PHASE_C;
                        this.triageModel.broadCastUpdate();
                }
        }

        /**
         * Process resource count.
         *
         * @param resourceCountIn the resource count
         */
        public final void processResourceCount(final int resourceCountIn) {
                this.resourceCount = resourceCountIn;
                if (this.currentPhase == ESI_PHASE_C) {
                        // we are, correctly, in the third phase
                        if (this.resourceCount == 0) {
                                this.proposedTriageScore = ESI_CATEGORY_5;
                                this.triageModel.broadCastUpdate();
                        } else if (this.resourceCount == 1) {
                                this.proposedTriageScore = ESI_CATEGORY_4;
                                this.triageModel.broadCastUpdate();
                        } else if (this.resourceCount >= 2) {
                                this.currentPhase = ESI_PHASE_D;
                                this.triageModel.broadCastUpdate();
                        }
                }
        }

        /**
         * Recieve vitals.
         *
         * @param vitals the vitals
         */
        public final void recieveVitals(final double[] vitals) {
                int vitalIndex = 0;     // index iterating through the input
vital columns
                // assign vital signs to patient
                this.patient.setHeartRate(vitals[vitalIndex++]);
```

```
            this.patient.setRespiratoryRate(vitals[vitalIndex++]);
            this.patient.setOxySat(vitals[vitalIndex++]);
            this.patient.setTemp(vitals[vitalIndex++]);

            this.verifyVitals();

            this.triageModel.broadCastUpdate();
      }

      /**
       * Retrieve vital states.
       *
       * @return the boolean[]
       */
      public final boolean[] retrieveVitalStates() {
            int vitalIndex = 0;     // index iterating through the input 
vital columns
            boolean[] vitalStates = new boolean[VITAL_SIGN_COUNT];
            vitalStates[vitalIndex++] = this.heartRateCritical;
            vitalStates[vitalIndex++] = this.respiratoryRateCritical;
            vitalStates[vitalIndex++] = this.oxySatCritical;
            vitalStates[vitalIndex++] = this.tempCritical;

            return vitalStates;
      }

      /**
       * Process vitals.
       */
      public final void processVitals() {
            boolean isCritical = false;
            if (this.currentPhase == ESI_PHASE_D) {
                  // it is the phase conserning the vital signs (phase 4)
                  this.verifyVitals();
                  System.out.println("hr: " + patient.getHeartRate());
                  System.out.println("rr: " + 
patient.getRespiratoryRate());
                  System.out.println("oxysat: " + patient.getOxySat());
                  System.out.println("temp: " + patient.getTemp());

                  // are there any critical vital signs?
                  isCritical = (this.heartRateCritical || 
this.respiratoryRateCritical
                                          || this.oxySatCritical || 
this.tempCritical);
                  if (isCritical) {
                        // a critical vital sign - nurse should consider 
uptriage
                        this.proposedTriageScore = ESI_CATEGORY_2;
                  } else {
                        // no vitals critical
                        this.proposedTriageScore = ESI_CATEGORY_3;
                  }
                  this.triageModel.broadCastUpdate();
            }
      }

      /**
       * Verify vitals.
       */
      public final void verifyVitals() {
            double patientHeartRate = this.patient.getHeartRate();
            double patentRespiratoryRate =
this.patient.getRespiratoryRate();
            double patientOxySat = this.patient.getOxySat();
            double patientTemp = this.patient.getTemp();

            // test if any vitals are lower than critical limit
            heartRateCritical = (patientHeartRate >= this.heartRateLimit);
            respiratoryRateCritical = (patentRespiratoryRate
                                                  >=
this.respiratoryRateLimit);
            oxySatCritical = (patientOxySat < this.oxySatLimit);
            tempCritical = (patientTemp >= this.tempLimit);
      }

      /**
       * Send warning.
       */
      void sendWarning() {

      }

      /**
       * Send vital sign warning.
       */
      void sendVitalSignWarning() {

      }

      /**
       * Gets the proposed triage score.
       *
       * @return the proposed triage score
       */
      public final int getProposedTriageScore() {
            return proposedTriageScore;
      }

      /**
       * Sets the proposed triage score.
       *
       * @param proposedTriageScoreIn the new proposed triage score
       */
      public final void setProposedTriageScore(final int
proposedTriageScoreIn) {
            this.proposedTriageScore = proposedTriageScoreIn;
      }

      /**
       * Gets the triage score.
       *
       * @return the triage score
       */
      public final int getTriageScore() {
            return triageScore;
      }

      /**
       * Sets the triage score.
       *
       * @param triageScoreIn the new triage score
```

```
	 */
	public final void setTriageScore(final int triageScoreIn) {
		this.triageScore = triageScoreIn;
	}

	/**
	 * Gets the current phase.
	 *
	 * @return the current phase
	 */
	public final int getCurrentPhase() {
		return currentPhase;
	}

	/**
	 * Sets the current phase.
	 *
	 * @param currentPhaseIn the new current phase
	 */
	public final void setCurrentPhase(final int currentPhaseIn) {
		this.currentPhase = currentPhaseIn;
	}
}
```

```
package triageview;
import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Component;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Observable;
import java.util.Observer;

import javax.swing.AbstractAction;

import javax.swing.BorderFactory;
import javax.swing.ButtonGroup;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollBar;
import javax.swing.JScrollPane;
import javax.swing.JSpinner;
import javax.swing.JTabbedPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextPane;
import javax.swing.JToggleButton;
import javax.swing.JToolBar;
import javax.swing.LayoutStyle;
import javax.swing.ListSelectionModel;
import javax.swing.SpinnerListModel;
import javax.swing.SpinnerNumberModel;
import javax.swing.SwingConstants;

import javax.swing.WindowConstants;
import javax.swing.border.BevelBorder;
import javax.swing.border.LineBorder;
import javax.swing.border.SoftBevelBorder;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;
import javax.swing.SwingUtilities;

import triagecontroller.TableSelectionListener;
import triagecontroller.TriageController;
import triagemodel.ESITriage;
import triagemodel.ResultSetTableModel;
import triagemodel.ResultSetTableModelFactory;
```

TriageFrame

```
/**
* This code was edited or generated using CloudGarden's Jigloo
* SWT/Swing GUI Builder, which is free for non-commercial
* use. If Jigloo is being used commercially (ie, by a corporation,
* company or business for any purpose whatever) then you
* should purchase a license for each developer using Jigloo.
* Please visit www.cloudgarden.com for details.
* Use of Jigloo implies acceptance of these licensing terms.
* A COMMERCIAL LICENSE HAS NOT BEEN PURCHASED FOR
* THIS MACHINE, SO JIGLOO OR THIS CODE CANNOT BE USED
* LEGALLY FOR ANY CORPORATE OR COMMERCIAL PURPOSE.
*/
public class TriageFrame extends javax.swing.JFrame implements Observer{

      {
            //Set Look & Feel
            try {

      javax.swing.UIManager.setLookAndFeel("com.jgoodies.looks.plastic.Plas
tic3DLookAndFeel");
            } catch(Exception e) {
                  e.printStackTrace();
            }
      }

      private JScrollPane jScrollPane1;
      private JButton triageButton;
      private JScrollPane jScrollPane2;
      private JPanel medicalStatePane;
      private JPanel patientInfoPane;
      private JLabel jLabel3;
      private JLabel jLabel2;
      private JLabel jLabel1;
      private JLabel warningHR;
      private JSpinner temSpinner;
      private JSpinner HRSpinner;
      private JSpinner RRSpinner;
      private JSpinner OSSpinner;
      private JLabel medHistoryLabel;
      private JLabel complaintLabel;
      private JLabel ageLabel;
      private JLabel ktLabel;
      private JLabel nameLabel;
      private JPanel messagePane;
      private JButton jButton4;
      private JButton treatButton;
      private JButton jButton2;
      private JToolBar jToolBar1;
      private JTable jTable1;
      private JPanel jPanel2;
      private JToolBar triageToolBar;
      private JTabbedPane jTabbedPane1;
      private JPanel jPanel1;
      private JTable patientTable;

      // the DB attributes
      private static final String driver = "org.gjt.mm.mysql.Driver";
      private static final String url =
"jdbc:mysql://localhost/medical_prototype";
      private static final String username = "root";
      private static final String password = "localhost1";

      private static Connection con;
      private JLabel medicationLabel;
      private JDialog jDialog1;
      private JLabel vitalSignLabel;
      private JLabel HRLabel;
      private JLabel RRLabel;
      private JLabel OSLabel;
      private JLabel tempLabel;
      private JTextArea jTextArea3;
      private JTextArea jTextArea2;
      private JTextArea jTextArea1;
      private JTextArea phase1Explanation;
      private JButton vitalSignConfirmButton;
      private JButton confirmResourcesButton;
      private JRadioButton twoPlusRadioButton;
      private JRadioButton oneRadioButton;
      private JRadioButton noneRadioButton;
      private ButtonGroup resourceButtonGroup;
      private JButton jButton9;
      private JButton jButton8;
      private JLabel phase4Title;
      private JPanel phase4Panel;
      private JLabel phase3Title;
      private JPanel phase3Panel;
      private JLabel phase2Title;
      private JPanel phase2Panel;
      private JLabel phase1Title;
      private JPanel phase1Panel;
      private JDialog esiTriageDialogWindow;
      private JButton jButton3;
      private JButton jButton1;
      private static Statement st;
      private static ResultSet result;
      private static ResultSetMetaData metadata;      // Additional
information about the results
      private static ResultSetTableModelFactory factory;
      private static String query = "select * from patient_record";

      private ESITriage triage;
      private TriageController triageController;
      private static String PHASE_1_DIALOG = "phase1Dialog";
      private static String PHASE_2_DIALOG = "phase2Dialog";
      private static String PHASE_3_DIALOG = "phase3Dialog";
      private static String PHASE_4_DIALOG = "phase4Dialog";

      /**
      * Auto-generated main method to display this JFrame
      */
      public static void main(String[] args) {
            SwingUtilities.invokeLater(new Runnable() {
                  public void run() {
                        ESITriage triage = new ESITriage();
                        TriageFrame inst = new TriageFrame(triage);
                        TriageController controller = new
TriageController(inst, triage);
                        inst.setController(controller);
                        try{
                              factory = new ResultSetTableModelFactory(
                                          driver, url, username, password);
                        }
                        catch(Exception e){
                              System.out.println(e);
                        }
                        try {
                              // This is the crux of it all.  Use the 
factory object
                              // to obtain a TableModel object for the query 
results
                              // and display that model in the JTable 
component.

                              triage.addObserver(inst);

                              ResultSetTableModel model = 
factory.getResultSetTableModel(query);
                              inst.jTable1.setModel(model);

                              ListSelectionModel listSelectionModel = 
inst.jTable1.getSelectionModel();

      listSelectionModel.addListSelectionListener(controller.getSelectionLi
stener());
                              triage.populate(model);
                              // We're done, so clear the feedback message
                            }
                            catch (SQLException e) {
                              System.out.println(e);
                            }
                        inst.setLocationRelativeTo(null);
                        inst.setVisible(true);
                  }
            });
      }

      public TriageFrame(ESITriage triage) {
            super();
            this.triage = triage;
            esiTriageDialogWindow = this.getJDialog1();
            jDialog1.setTitle("ESI triage");
            jDialog1.setTitle("ESI Triage");
            initGUI();
      }

      private void initGUI() {
            try {
                  GroupLayout thisLayout = new 
GroupLayout((JComponent)getContentPane());
                  getContentPane().setLayout(thisLayout);

      setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
                  this.setTitle("Automated ESI triage");
                  {
                        jTabbedPane1 = new JTabbedPane();
                        {
                              jPanel2 = new JPanel();
                              jTabbedPane1.addTab("Untriaged", null, 
jPanel2, null);
```

```
                              GroupLayout jPanel2Layout = new
GroupLayout((JComponent)jPanel2);
                              jPanel2.setPreferredSize(new
java.awt.Dimension(740, 340));
                              jPanel2.setLayout(jPanel2Layout);
                              {
                                    jScrollPane2 = new JScrollPane();
                                    {
                                          TableModel jTable1Model =
                                                new DefaultTableModel(
                                                            new String[][] {
{ "One", "Two" }, { "Three", "Four" } },
                                                            new String[] {
"Column 1", "Column 2" });
                                          jTable1 = new JTable();


      jScrollPane2.setViewportView(jTable1);
                                          jTable1.setModel(jTable1Model);
                                    }
                              }
                              {
                                    jToolBar1 = new JToolBar();
                                    {
                                          jButton2 = new JButton();
                                          jToolBar1.add(jButton2);
                                          jButton2.setText("Triage");
                                    }
                                    {
                                          jButton4 = new JButton();
                                          jToolBar1.add(jButton4);
                                          jButton4.setText("Remove");
                                    }
                              }

      jPanel2Layout.setVerticalGroup(jPanel2Layout.createSequentialGroup()
                                    .addContainerGap()
                                    .addComponent(jToolBar1,
GroupLayout.PREFERRED_SIZE, 25, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                                    .addComponent(jScrollPane2, 0, 267,
Short.MAX_VALUE)
                                    .addContainerGap());

      jPanel2Layout.setHorizontalGroup(jPanel2Layout.createSequentialGroup(
)
                                    .addContainerGap()

      .addGroup(jPanel2Layout.createParallelGroup()
                                        .addComponent(jScrollPane2,
GroupLayout.Alignment.LEADING, 0, 621, Short.MAX_VALUE)
                                        .addComponent(jToolBar1,
GroupLayout.Alignment.LEADING, 0, 621, Short.MAX_VALUE))
                                    .addContainerGap());
                        }
                        {
                              jPanel1 = new JPanel();
                              jTabbedPane1.addTab("Triaged", null, jPanel1,
null);
```

```
                              GroupLayout jPanel1Layout = new
GroupLayout((JComponent)jPanel1);
                              jPanel1.setLayout(jPanel1Layout);
                              jPanel1.setPreferredSize(new
java.awt.Dimension(645, 147));
                              {
                                    jScrollPane1 = new JScrollPane();

      jScrollPane1.setRequestFocusEnabled(false);
                                    {
                                          TableModel patientTableModel =
                                                new DefaultTableModel(
                                                            new String[][] {
{ "One", "Two" }, { "Three", "Four" } },
                                                            new String[] {
"Column 1", "Column 2" });
                                          patientTable = new JTable();
                                          patientTable.setLayout(null);

      jScrollPane1.setViewportView(patientTable);

      patientTable.setModel(patientTableModel);
                                          patientTable.setPreferredSize(new
java.awt.Dimension(618, 32));

      patientTable.getTableHeader().setBounds(0, 0, 618, 20);
                                    }
                              }
                              {
                                    triageToolBar = new JToolBar();
                                    {
                                          triageButton = new JButton();
                                          triageToolBar.add(triageButton);
                                          triageButton.setText("Retriage");
                                    }
                                    {
                                          treatButton = new JButton();
                                          triageToolBar.add(treatButton);
                                          treatButton.setText("Remove");
                                    }
                              }

      jPanel1Layout.setHorizontalGroup(jPanel1Layout.createSequentialGroup(
)
                                          .addContainerGap()

      .addGroup(jPanel1Layout.createParallelGroup()

      .addComponent(jScrollPane1, GroupLayout.Alignment.LEADING, 0, 621,
Short.MAX_VALUE)

      .addComponent(triageToolBar, GroupLayout.Alignment.LEADING, 0, 621,
Short.MAX_VALUE))
                                                      .addContainerGap());

      jPanel1Layout.setVerticalGroup(jPanel1Layout.createSequentialGroup()
                                          .addContainerGap()
                                          .addComponent(triageToolBar,
GroupLayout.PREFERRED_SIZE, 25, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
```

```
                                          .addComponent(jScrollPane1, 0, 
279, Short.MAX_VALUE)
                                          .addContainerGap());
                        }
                  }

      thisLayout.setVerticalGroup(thisLayout.createSequentialGroup()
                        .addComponent(jTabbedPane1, 0, 462, 
Short.MAX_VALUE)
                        .addContainerGap());

      thisLayout.setHorizontalGroup(thisLayout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jTabbedPane1, 0, 753, 
Short.MAX_VALUE)
                        .addContainerGap());
                  pack();

                  pack();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      private JPanel getMedicalStatePane() {
            if(medicalStatePane == null) {
                  medicalStatePane = new JPanel();
                  GroupLayout medicalStatePaneLayout = new 
GroupLayout((JComponent)medicalStatePane);
                  medicalStatePane.setLayout(medicalStatePaneLayout);

      medicalStatePane.setBorder(BorderFactory.createTitledBorder("Patient 
Medical Information"));

      medicalStatePaneLayout.setHorizontalGroup(medicalStatePaneLayout.crea
teSequentialGroup()
                        .addGap(7)

      .addGroup(medicalStatePaneLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING, 
medicalStatePaneLayout.createSequentialGroup()
                                .addComponent(getMedicationLabel(), 0, 363, 
Short.MAX_VALUE)
                                .addGap(13))
                            .addComponent(getMedHistoryLabel(), 
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 376, 
GroupLayout.PREFERRED_SIZE)
                            .addComponent(getComplaintLabel(), 
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 376, 
GroupLayout.PREFERRED_SIZE)));

      medicalStatePaneLayout.linkSize(SwingConstants.HORIZONTAL, new 
Component[] {getMedHistoryLabel(), getComplaintLabel()});

      medicalStatePaneLayout.setVerticalGroup(medicalStatePaneLayout.create
SequentialGroup()
                        .addContainerGap()
                        .addComponent(getComplaintLabel(), 
GroupLayout.PREFERRED_SIZE, 26, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
```

```
                        .addComponent(getMedHistoryLabel(),
GroupLayout.PREFERRED_SIZE, 22, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getMedicationLabel(),
GroupLayout.PREFERRED_SIZE, 31, GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(81, Short.MAX_VALUE));
            }
            return medicalStatePane;
      }

      private JPanel getMessagePane() {
            if(messagePane == null) {
                  messagePane = new JPanel();
                  CardLayout messagePaneLayout = new CardLayout();
                  messagePane.setLayout(messagePaneLayout);


      messagePane.setBorder(BorderFactory.createTitledBorder("ESI Triage
Algorithm"));
                  messagePane.setVerifyInputWhenFocusTarget(false);

                  messagePane.add(getPhase1Panel(), PHASE_1_DIALOG);
                  messagePane.add(getPhase2Panel(), PHASE_2_DIALOG);
                  messagePane.add(getPhase3Panel(), PHASE_3_DIALOG);
                  messagePane.add(getPhase4Panel(), PHASE_4_DIALOG);
            }
            return messagePane;
      }

      private JLabel getNameLabel() {
            if(nameLabel == null) {
                  nameLabel = new JLabel();
                  nameLabel.setText("Nafn:");
            }
            return nameLabel;
      }

      private JLabel getKtLabel() {
            if(ktLabel == null) {
                  ktLabel = new JLabel();
                  ktLabel.setText("Kennitala:");
            }
            return ktLabel;
      }

      private JLabel getAgeLabel() {
            if(ageLabel == null) {
                  ageLabel = new JLabel();
                  ageLabel.setText("Age");
            }
            return ageLabel;
      }

      private JLabel getComplaintLabel() {
            if(complaintLabel == null) {
                  complaintLabel = new JLabel();
                  complaintLabel.setText("Patient complaint:");
            }
            return complaintLabel;
      }
```

```
	private JLabel getMedHistoryLabel() {
		if(medHistoryLabel == null) {
			medHistoryLabel = new JLabel();
			medHistoryLabel.setText("Medical history:");
		}
		return medHistoryLabel;
	}

	private JSpinner getOSSpinner() {
		if(OSSpinner == null) {
			SpinnerNumberModel osMdel = new SpinnerNumberModel(0.0d, 
0.0d, 100.0d, 0.1d);
			OSSpinner = new JSpinner();
			OSSpinner.setModel(osMdel);
		}
		return OSSpinner;
	}

	private JSpinner getRRSpinner() {
		if(RRSpinner == null) {
			SpinnerNumberModel rrMdel = new SpinnerNumberModel(0.0d, 
0.0d, 100.0d, 1.0d);
			RRSpinner = new JSpinner();
			RRSpinner.setModel(rrMdel);
		}
		return RRSpinner;
	}

	private JSpinner getHRSpinner() {
		if(HRSpinner == null) {
			SpinnerNumberModel hrMdel = new SpinnerNumberModel(0.0d, 
0.0d, 200.0d, 1.0d);
			HRSpinner = new JSpinner();
			HRSpinner.setModel(hrMdel);
		}
		return HRSpinner;
	}

	private JSpinner getTemSpinner() {
		if(temSpinner == null) {
			SpinnerNumberModel tempMdel = new 
SpinnerNumberModel(0.0d, 0.0d, 100.0d, 0.1d);
			temSpinner = new JSpinner();
			temSpinner.setModel(tempMdel);
			temSpinner.setEnabled(false);
		}
		return temSpinner;
	}

	private JLabel getWarningHR() {
		if(warningHR == null) {
			warningHR = new JLabel();

	warningHR.setBorder(BorderFactory.createBevelBorder(BevelBorder.LOWER
ED));
		}
		return warningHR;
	}

	private JLabel getJLabel1() {
```

```
            if(jLabel1 == null) {
                  jLabel1 = new JLabel();

      jLabel1.setBorder(BorderFactory.createBevelBorder(BevelBorder.LOWERED
));
            }
            return jLabel1;
      }

      private JLabel getJLabel2() {
            if(jLabel2 == null) {
                  jLabel2 = new JLabel();

      jLabel2.setBorder(BorderFactory.createBevelBorder(BevelBorder.LOWERED
));
            }
            return jLabel2;
      }

      private JLabel getJLabel3() {
            if(jLabel3 == null) {
                  jLabel3 = new JLabel();

      jLabel3.setBorder(BorderFactory.createBevelBorder(BevelBorder.LOWERED
));
            }
            return jLabel3;
      }


      private JButton getJButton1() {
            if(jButton1 == null) {
                  jButton1 = new JButton();
                  jButton1.setText("Yes");
            }
            return jButton1;
      }

      private JButton getJButton3() {
            if(jButton3 == null) {
                  jButton3 = new JButton();
                  jButton3.setText("No");
            }
            return jButton3;
      }

      public void setController(TriageController controller){

            this.triageController = controller;

      this.jButton2.addActionListener(controller.getTriageInitListener());


      this.getJButton1().addActionListener(controller.getYesListener());

      this.getJButton3().addActionListener(controller.getNoListener());

      this.getJButton8().addActionListener(controller.getYesListener());

      this.getJButton9().addActionListener(controller.getNoListener());
      this.getConfirmResourcesButton().addActionListener(controller.getReso
urceListener());

      this.getNoneRadioButton().addActionListener(controller.getRadioButton
Listener());

      this.getOneRadioButton().addActionListener(controller.getRadioButtonL
istener());

      this.getTwoPlusRadioButton().addActionListener(controller.getRadioBut
tonListener());

      this.getVitalSignConfirmButton().addActionListener(controller.getConf
irmListener());
      }

      public double[] retrieveVitals(){
            double[] vitals = new double[4];    // the patient vital sign
user input
            SpinnerNumberModel numberModel;           // value model of the
spinner gui components
            double hrValue = 0.0d;
            double rrValue = 0.0d;
            double oxSatValue = 0.0d;
            double tempValue = 0.0d;

            // retrieve heart rate value from user input
            numberModel =
(SpinnerNumberModel)(this.getHRSpinner().getModel());
            hrValue = ((Double)(numberModel.getValue())).doubleValue();
            vitals[0] = hrValue;

            // retrieve respiration rate value from user input
            numberModel =
(SpinnerNumberModel)(this.getRRSpinner().getModel());
            rrValue = ((Double)(numberModel.getValue())).doubleValue();
            vitals[1] = rrValue;

            // retrieve oxygen saturation ratio from user input
            numberModel =
(SpinnerNumberModel)(this.getOSSpinner().getModel());
            oxSatValue = ((Double)(numberModel.getValue())).doubleValue();
            vitals[2] = oxSatValue;

            // retrieve temperature value from user input
            numberModel =
(SpinnerNumberModel)(this.getTemSpinner().getModel());
            tempValue = ((Double)(numberModel.getValue())).doubleValue();
            vitals[3] = tempValue;
            return vitals;
      }

      public void displayTriageDialog() {
            displayPatientInformation();
            this.esiTriageDialogWindow.pack();
            this.esiTriageDialogWindow.setVisible(true);
      }

      private void displayPatientInformation(){
            String[] currentPatientInfo = 
this.triage.retrieveSelectetPatientInfo();
            nameLabel.setText("Name: " + currentPatientInfo[0]);
            ktLabel.setText("Kennitala: " + currentPatientInfo[1]);
            //genderLabel.setText("Gender: " + currentPatientInfo[2]);
            ageLabel.setText("Age: " + currentPatientInfo[3]);
            complaintLabel.setText("Complaint: " + currentPatientInfo[4]);
            medHistoryLabel.setText("Medical History: " + 
currentPatientInfo[5]);
            medicationLabel.setText("Medication: " + 
currentPatientInfo[6]);
      }

      public void update(Observable o, Object arg) {
            int proposedScore = triage.retrieveProposedScore();
            int esiPhase = triage.retrieveCurrentPhase();
            System.out.println("proposed score: " + proposedScore);
            displayPatientInformation();
            if(proposedScore == -1){
                  System.out.println("esi phase: " + esiPhase);
                  // display appropriate gui components for a given phase
                  CardLayout layout = 
(CardLayout)(this.messagePane.getLayout());
                  switch (esiPhase) {
                        case 1:
                              layout.show(this.messagePane, PHASE_1_DIALOG);
                              break;
                        case 2:
                              layout.show(this.messagePane, PHASE_2_DIALOG);
                              break;
                        case 3:
                              layout.show(this.messagePane, PHASE_3_DIALOG);
                              break;
                        case 4:
                              layout.show(this.messagePane, PHASE_4_DIALOG);
                              temperatureCheck();     // enable/disable 
temperature input
                              break;
                        default:
                              // invalid phase value
                              break;
                  }
            }
            else {
                  int triageScore = this.triage.retrieveProposedScore();
                  switch(triageScore){
                        case 1:
                              JOptionPane.showMessageDialog(this, "ESI-1");
                              this.esiTriageDialogWindow.setVisible(false);
                              break;
                        case 2:
                              if(esiPhase == 4){
                                    Object[] options = {"Upgrade to ESI-2", 
                     "Keep at ESI-3"};
                                    int selected = 
JOptionPane.showOptionDialog(this,
                                                "One or more vital sign is 
critical!",
                                                "Critical vital sign",
                                                JOptionPane.YES_NO_OPTION,
                                                JOptionPane.QUESTION_MESSAGE,
                                             null,     //do not use a
custom Icon
                                             options,  //the titles of
buttons
                                             options[0]); //default button
title
                                   if(selected == 0){
                                         // chose to upgradet the triage
level

      JOptionPane.showMessageDialog(this, "ESI-2");

      this.esiTriageDialogWindow.setVisible(false);
                                   }
                                   else if(selected == 1){
                                         // chose to keep current triage
level

      JOptionPane.showMessageDialog(this, "ESI-3");

      this.esiTriageDialogWindow.setVisible(false);
                                   }

                             }
                             else{
                                   JOptionPane.showMessageDialog(this,
"ESI-2");

      this.esiTriageDialogWindow.setVisible(false);
                             }

                             break;
                       case 3:
                             JOptionPane.showMessageDialog(this, "ESI-3");
                             this.esiTriageDialogWindow.setVisible(false);
                             break;
                       case 4:
                             JOptionPane.showMessageDialog(this, "ESI-4");
                             this.esiTriageDialogWindow.setVisible(false);
                             break;
                       case 5:
                             JOptionPane.showMessageDialog(this, "ESI-5");
                             this.esiTriageDialogWindow.setVisible(false);
                             break;
                 }
                 // ask for confirmation of the triage score
                 // then pass the answer to the model
           }

      }

      public void resetTriageInputs(){
            // resetting vital sign input components
            this.getHRSpinner().setValue(0.0d);
            this.getRRSpinner().setValue(0.0d);
            this.getOSSpinner().setValue(0.0d);
            this.getTemSpinner().setValue(0.0d);

            // resetting resource count input components
            this.getResourceButtonGroup().clearSelection();
      }
      private void temperatureCheck(){
            // enaple temperature spinner based on the patient category
            String ageCategory =
this.triage.retrieveSelectetPatientInfo()[7];
            boolean tempMatters = (ageCategory.equalsIgnoreCase("A") ||

      ageCategory.equalsIgnoreCase("B") ||

      ageCategory.equalsIgnoreCase("C"));
            if(tempMatters){
                  // enable temperatuer input
                  this.getTemSpinner().setEnabled(true);
                  this.getTempLabel().setEnabled(true);
            }
            else{
                  // disable temperatuer input
                  this.getTemSpinner().setEnabled(false);
                  this.getTempLabel().setEnabled(false);
            }
      }

      private JPanel getPatientInfoPane() {
            if(patientInfoPane == null) {
                  patientInfoPane = new JPanel();
                  GroupLayout patientInfoPaneLayout = new
GroupLayout((JComponent)patientInfoPane);
                  patientInfoPane.setLayout(patientInfoPaneLayout);

      patientInfoPane.setBorder(BorderFactory.createTitledBorder("Patient
Information"));

      patientInfoPaneLayout.setVerticalGroup(patientInfoPaneLayout.createSe
quentialGroup()
                        .addComponent(getNameLabel(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getKtLabel(),
GroupLayout.PREFERRED_SIZE, 22, GroupLayout.PREFERRED_SIZE)
                        .addComponent(getAgeLabel(),
GroupLayout.PREFERRED_SIZE, 22, GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(31, Short.MAX_VALUE));

      patientInfoPaneLayout.setHorizontalGroup(patientInfoPaneLayout.create
SequentialGroup()
                        .addContainerGap()

      .addGroup(patientInfoPaneLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
patientInfoPaneLayout.createSequentialGroup()
                                .addComponent(getNameLabel(),
GroupLayout.PREFERRED_SIZE, 261, GroupLayout.PREFERRED_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 18,
Short.MAX_VALUE))
                            .addGroup(GroupLayout.Alignment.LEADING,
patientInfoPaneLayout.createSequentialGroup()
                                .addComponent(getKtLabel(), 0, 325,
Short.MAX_VALUE)
```

```
                                .addGap(33))
                            .addGroup(GroupLayout.Alignment.LEADING,
patientInfoPaneLayout.createSequentialGroup()
                                .addComponent(getAgeLabel(),
GroupLayout.PREFERRED_SIZE, 285, GroupLayout.PREFERRED_SIZE)
                                .addGap(0, 73, Short.MAX_VALUE)))
                        .addContainerGap());
            }
            return patientInfoPane;
      }


      private JPanel getPhase1Panel() {
            if(phase1Panel == null) {
                  phase1Panel = new JPanel();
                  GroupLayout phase1PanelLayout = new
GroupLayout((JComponent)phase1Panel);
                  phase1Panel.setLayout(phase1PanelLayout);

      phase1PanelLayout.setHorizontalGroup(phase1PanelLayout.createSequenti
alGroup()
                        .addContainerGap()
                        .addGroup(phase1PanelLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
phase1PanelLayout.createSequentialGroup()

.addGroup(phase1PanelLayout.createParallelGroup()
                                    .addComponent(getJButton1(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 52,
GroupLayout.PREFERRED_SIZE)

.addGroup(GroupLayout.Alignment.LEADING,
phase1PanelLayout.createSequentialGroup()
                                        .addComponent(getPhase1Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                        .addGap(12)))

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(getJButton3(),
GroupLayout.PREFERRED_SIZE, 57, GroupLayout.PREFERRED_SIZE)
                                .addGap(0, 131, Short.MAX_VALUE))
                            .addComponent(getPhase1Explanation(),
GroupLayout.Alignment.LEADING, 0, 245, Short.MAX_VALUE)));

      phase1PanelLayout.setVerticalGroup(phase1PanelLayout.createSequential
Group()
                        .addContainerGap()
                        .addComponent(getPhase1Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(getPhase1Explanation(),
GroupLayout.PREFERRED_SIZE, 63, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

      .addGroup(phase1PanelLayout.createParallelGroup(GroupLayout.Alignment
.BASELINE)
```

```
                            .addComponent(getJButton1(),
GroupLayout.Alignment.BASELINE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                            .addComponent(getJButton3(),
GroupLayout.Alignment.BASELINE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE))
                        .addContainerGap(212, 212));
            }
            return phase1Panel;
      }

      private JLabel getPhase1Title() {
            if(phase1Title == null) {
                  phase1Title = new JLabel();
                  phase1Title.setText("Phase 1");
            }
            return phase1Title;
      }

      private JPanel getPhase2Panel() {
            if(phase2Panel == null) {
                  phase2Panel = new JPanel();
                  GroupLayout phase2PanelLayout = new
GroupLayout((JComponent)phase2Panel);
                  phase2Panel.setLayout(phase2PanelLayout);

      phase2PanelLayout.setHorizontalGroup(phase2PanelLayout.createSequenti
alGroup()
                        .addContainerGap()
                        .addGroup(phase2PanelLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
phase2PanelLayout.createSequentialGroup()

.addGroup(phase2PanelLayout.createParallelGroup()
                                    .addComponent(getJButton8(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 52,
GroupLayout.PREFERRED_SIZE)

.addGroup(GroupLayout.Alignment.LEADING,
phase2PanelLayout.createSequentialGroup()
                                        .addComponent(getPhase2Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                        .addGap(12)))

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(getJButton9(),
GroupLayout.PREFERRED_SIZE, 59, GroupLayout.PREFERRED_SIZE)
                                .addGap(0, 129, Short.MAX_VALUE))

.addGroup(phase2PanelLayout.createSequentialGroup()
                                .addComponent(getJTextArea1(),
GroupLayout.PREFERRED_SIZE, 245, GroupLayout.PREFERRED_SIZE)
                                .addGap(0, 0, Short.MAX_VALUE)))
                        .addContainerGap(122, 122));

      phase2PanelLayout.setVerticalGroup(phase2PanelLayout.createSequential
Group()
                        .addContainerGap()
                        .addComponent(getPhase2Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(getJTextArea1(),
GroupLayout.PREFERRED_SIZE, 92, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)

      .addGroup(phase2PanelLayout.createParallelGroup(GroupLayout.Alignment
.BASELINE)
                            .addComponent(getJButton8(),
GroupLayout.Alignment.BASELINE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                            .addComponent(getJButton9(),
GroupLayout.Alignment.BASELINE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE))
                        .addContainerGap(147, 147));
            }
            return phase2Panel;
      }

      private JLabel getPhase2Title() {
            if(phase2Title == null) {
                  phase2Title = new JLabel();
                  phase2Title.setText("Phase 2");
            }
            return phase2Title;
      }

      private JPanel getPhase3Panel() {
            if(phase3Panel == null) {
                  phase3Panel = new JPanel();
                  GroupLayout phase3PanelLayout = new
GroupLayout((JComponent)phase3Panel);
                  phase3Panel.setLayout(phase3PanelLayout);
                  resourceButtonGroup = getResourceButtonGroup();
                  resourceButtonGroup.add(getNoneRadioButton());
                  resourceButtonGroup.add(getOneRadioButton());
                  resourceButtonGroup.add(getTwoPlusRadioButton());

      phase3PanelLayout.setHorizontalGroup(phase3PanelLayout.createSequenti
alGroup()
                        .addContainerGap()
                        .addGroup(phase3PanelLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
phase3PanelLayout.createSequentialGroup()

.addGroup(phase3PanelLayout.createParallelGroup()
                                        .addComponent(getTwoPlusRadioButton(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)

.addGroup(GroupLayout.Alignment.LEADING,
phase3PanelLayout.createSequentialGroup()
                                            .addComponent(getPhase3Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                            .addGap(46))
.addGroup(GroupLayout.Alignment.LEADING,
phase3PanelLayout.createSequentialGroup()
                                        .addComponent(getNoneRadioButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                        .addGap(39))

.addGroup(GroupLayout.Alignment.LEADING,
phase3PanelLayout.createSequentialGroup()
                                        .addComponent(getOneRadioButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                        .addGap(46)))
                                .addComponent(getConfirmResourcesButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                .addGap(0, 101, Short.MAX_VALUE))
                            .addComponent(getJTextArea2(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 245,
GroupLayout.PREFERRED_SIZE)));

      phase3PanelLayout.setVerticalGroup(phase3PanelLayout.createSequential
Group()
                        .addContainerGap()
                        .addComponent(getPhase3Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(getJTextArea2(),
GroupLayout.PREFERRED_SIZE, 63, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getNoneRadioButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getOneRadioButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getTwoPlusRadioButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getConfirmResourcesButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(153, Short.MAX_VALUE));
            }
            return phase3Panel;
      }

      private JLabel getPhase3Title() {
            if(phase3Title == null) {
                  phase3Title = new JLabel();
                  phase3Title.setText("Phase 3");
            }
            return phase3Title;
      }

      private JPanel getPhase4Panel() {
            if(phase4Panel == null) {
                  phase4Panel = new JPanel();
                  GroupLayout phase4PanelLayout = new
GroupLayout((JComponent)phase4Panel);
                  phase4Panel.setLayout(phase4PanelLayout);
                  phase4Panel.setPreferredSize(new java.awt.Dimension(399,
343));

      phase4PanelLayout.setHorizontalGroup(phase4PanelLayout.createSequenti
alGroup()
                        .addContainerGap()
                        .addGroup(phase4PanelLayout.createParallelGroup()

.addGroup(phase4PanelLayout.createSequentialGroup()

.addGroup(phase4PanelLayout.createParallelGroup()
                                    .addComponent(getTempLabel(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 146,
GroupLayout.PREFERRED_SIZE)

.addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                        .addComponent(getVitalSignLabel(),
GroupLayout.PREFERRED_SIZE, 105, GroupLayout.PREFERRED_SIZE)
                                        .addGap(41))

.addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()

.addComponent(getVitalSignConfirmButton(), GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                                        .addGap(62))

.addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                        .addComponent(getPhase4Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                        .addGap(106))
                                    .addComponent(getOSLabel(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 146,
GroupLayout.PREFERRED_SIZE)
                                    .addComponent(getRRLabel(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 146,
GroupLayout.PREFERRED_SIZE)
                                    .addComponent(getHRLabel(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 146,
GroupLayout.PREFERRED_SIZE))

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(phase4PanelLayout.createParallelGroup()
                                    .addComponent(getHRSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
```

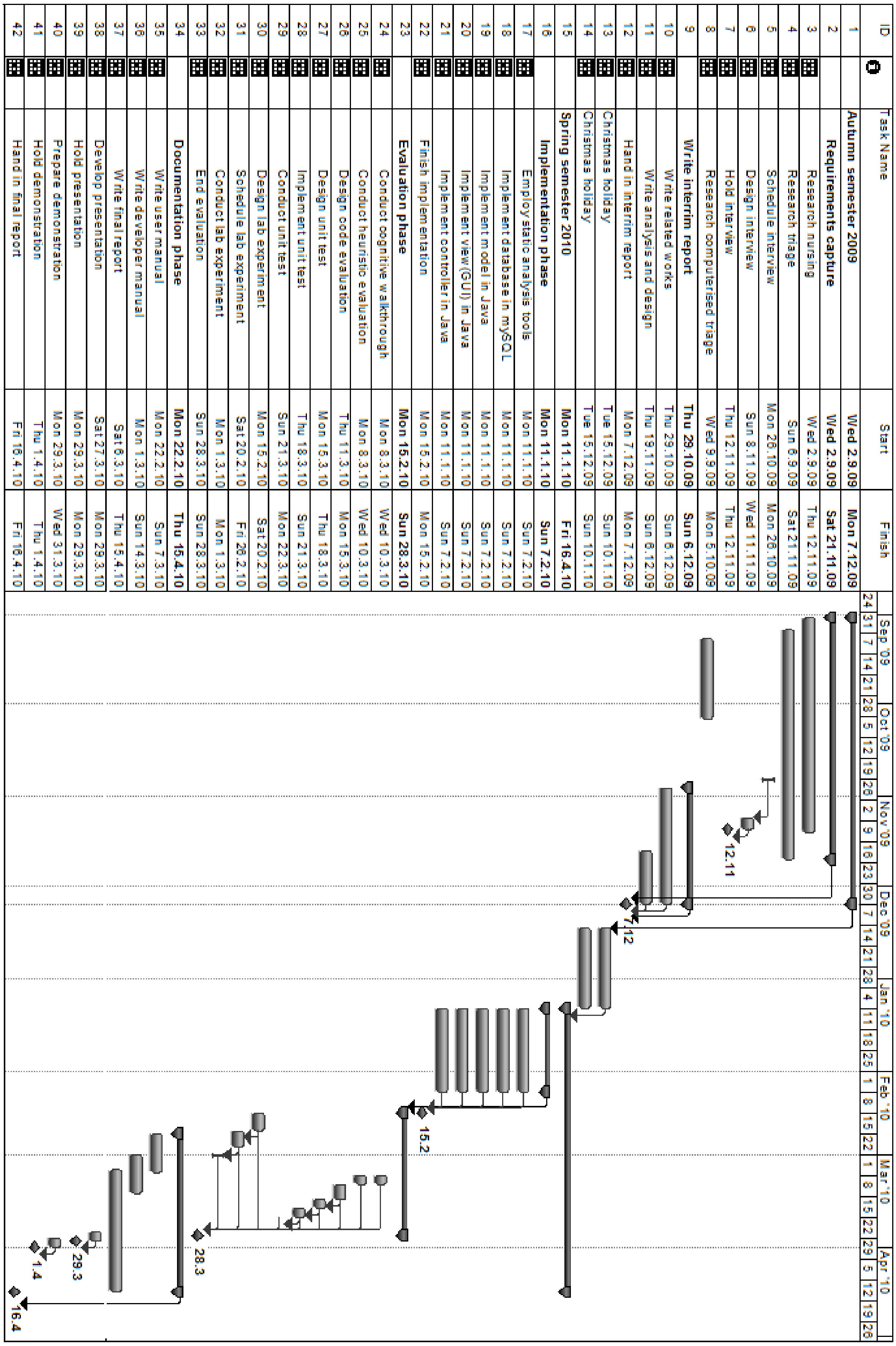
```
                                    .addComponent(getRRSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                                    .addComponent(getOSSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                                    .addComponent(getTemSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE))

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addGroup(phase4PanelLayout.createParallelGroup()
                                    .addComponent(getJLabel2(),
GroupLayout.Alignment.LEADING, 0, 104, Short.MAX_VALUE)
                                    .addComponent(getJLabel3(),
GroupLayout.Alignment.LEADING, 0, 104, Short.MAX_VALUE)
                                    .addComponent(getJLabel1(),
GroupLayout.Alignment.LEADING, 0, 104, Short.MAX_VALUE)
                                    .addComponent(getWarningHR(),
GroupLayout.Alignment.LEADING, 0, 104, Short.MAX_VALUE))
                                .addGap(43))
                            .addComponent(getJTextArea3(),
GroupLayout.Alignment.LEADING, 0, 373, Short.MAX_VALUE)));

      phase4PanelLayout.setVerticalGroup(phase4PanelLayout.createSequential
Group()
                        .addContainerGap()
                        .addComponent(getPhase4Title(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(getJTextArea3(),
GroupLayout.PREFERRED_SIZE, 63, GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getVitalSignLabel(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(phase4PanelLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                .addComponent(getWarningHR(),
GroupLayout.PREFERRED_SIZE, 16, GroupLayout.PREFERRED_SIZE)
                                .addGap(6))
                            .addComponent(getHRSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                .addComponent(getHRLabel(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                .addGap(6)))
                        .addGroup(phase4PanelLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
```

```
                                .addComponent(getJLabel1(),
GroupLayout.PREFERRED_SIZE, 16, GroupLayout.PREFERRED_SIZE)
                                .addGap(6))
                            .addComponent(getRRSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                .addComponent(getRRLabel(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                .addGap(6)))
                        .addGroup(phase4PanelLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                .addComponent(getJLabel2(),
GroupLayout.PREFERRED_SIZE, 16, GroupLayout.PREFERRED_SIZE)
                                .addGap(6))
                            .addComponent(getOSSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                .addComponent(getOSLabel(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                .addGap(6)))
                        .addGroup(phase4PanelLayout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                .addComponent(getJLabel3(),
GroupLayout.PREFERRED_SIZE, 16, GroupLayout.PREFERRED_SIZE)
                                .addGap(6))
                            .addComponent(getTemSpinner(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE)
                            .addGroup(GroupLayout.Alignment.LEADING,
phase4PanelLayout.createSequentialGroup()
                                .addComponent(getTempLabel(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                                .addGap(6)))
                        .addGap(16)
                        .addComponent(getVitalSignConfirmButton(),
GroupLayout.PREFERRED_SIZE, GroupLayout.PREFERRED_SIZE,
GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(78, Short.MAX_VALUE));
            }
            return phase4Panel;
      }

      private JLabel getPhase4Title() {
            if(phase4Title == null) {
                  phase4Title = new JLabel();
                  phase4Title.setText("Phase 4");
            }
            return phase4Title;
      }

      private JButton getJButton8() {
            if(jButton8 == null) {
```

```
			jButton8 = new JButton();
			jButton8.setText("Yes");
		}
		return jButton8;
	}

	private JButton getJButton9() {
		if(jButton9 == null) {
			jButton9 = new JButton();
			jButton9.setText("No");
		}
		return jButton9;
	}

	public ButtonGroup getResourceButtonGroup() {
		if(resourceButtonGroup == null) {
			resourceButtonGroup = new ButtonGroup();
		}
		return resourceButtonGroup;
	}

	public JRadioButton getNoneRadioButton() {
		if(noneRadioButton == null) {
			noneRadioButton = new JRadioButton();
			noneRadioButton.setText("None");
			noneRadioButton.setActionCommand("0");
		}
		return noneRadioButton;
	}

	private JRadioButton getOneRadioButton() {
		if(oneRadioButton == null) {
			oneRadioButton = new JRadioButton();
			oneRadioButton.setText("One");
			oneRadioButton.setActionCommand("1");
		}
		return oneRadioButton;
	}

	private JRadioButton getTwoPlusRadioButton() {
		if(twoPlusRadioButton == null) {
			twoPlusRadioButton = new JRadioButton();
			twoPlusRadioButton.setText("Two or more");
			twoPlusRadioButton.setActionCommand("2");
		}
		return twoPlusRadioButton;
	}

	private JButton getConfirmResourcesButton() {
		if(confirmResourcesButton == null) {
			confirmResourcesButton = new JButton();
			confirmResourcesButton.setText("Confirm");
		}
		return confirmResourcesButton;
	}

	private JButton getVitalSignConfirmButton() {
		if(vitalSignConfirmButton == null) {
			vitalSignConfirmButton = new JButton();
			vitalSignConfirmButton.setText("Confirm vitals");
		}
```

```
            return vitalSignConfirmButton;
      }

      private JTextArea getPhase1Explanation() {
            if(phase1Explanation == null) {
                  phase1Explanation = new JTextArea();
                  phase1Explanation.setText("Does the patient require 
\nimmediate life-saving intervention?");
                  phase1Explanation.setEditable(false);
                  phase1Explanation.setLocale(new java.util.Locale("en", 
"GB"));
            }
            return phase1Explanation;
      }

      private JTextArea getJTextArea1() {
            if(jTextArea1 == null) {
                  jTextArea1 = new JTextArea();
                  jTextArea1.setText("Is the patient in a high risk 
situation? \n"
                                          + "- Or - \n"
                                        + 
"Confused\\lethargic\\disoriented?\n"
                                        + "- Or - \n"
                                        + "In Severe pain\\distress?");
                  jTextArea1.setEditable(false);
                  jTextArea1.setLocale(new java.util.Locale("en","GB"));
            }
            return jTextArea1;
      }

      private JTextArea getJTextArea2() {
            if(jTextArea2 == null) {
                  jTextArea2 = new JTextArea();
                  jTextArea2.setText("How many resources are needed \nfor 
treatment?");
                  jTextArea2.setEditable(false);
                  jTextArea2.setLocale(new java.util.Locale("en","GB"));
            }
            return jTextArea2;
      }

      private JTextArea getJTextArea3() {
            if(jTextArea3 == null) {
                  jTextArea3 = new JTextArea();
                  jTextArea3.setText("Are any vital signs \nin a danger 
zone?");
                  jTextArea3.setEditable(false);
                  jTextArea3.setLocale(new java.util.Locale("en","GB"));
            }
            return jTextArea3;
      }
      private JLabel getTempLabel() {
            if(tempLabel == null) {
                  tempLabel = new JLabel();
                  tempLabel.setText("Body Temperature (°F)");
                  tempLabel.setEnabled(false);
            }
            return tempLabel;
      }
      private JLabel getOSLabel() {
            if(OSLabel == null) {
                  OSLabel = new JLabel();
                  OSLabel.setText("Oxygen Saturation (SaO2)");
            }
            return OSLabel;
      }
      private JLabel getRRLabel() {
            if(RRLabel == null) {
                  RRLabel = new JLabel();
                  RRLabel.setText("Respiration Rate (RR)");
            }
            return RRLabel;
      }
      private JLabel getHRLabel() {
            if(HRLabel == null) {
                  HRLabel = new JLabel();
                  HRLabel.setText("Heart Rate (HR)");
            }
            return HRLabel;
      }
      private JLabel getVitalSignLabel() {
            if(vitalSignLabel == null) {
                  vitalSignLabel = new JLabel();
                  vitalSignLabel.setText("Lífsmörk");
                  vitalSignLabel.setFont(new java.awt.Font("Segoe
UI",1,12));
            }
            return vitalSignLabel;
      }

      private JDialog getJDialog1() {
            if(jDialog1 == null) {
                  jDialog1 = new JDialog(this);
                  GroupLayout jDialog1Layout = new
GroupLayout((JComponent)jDialog1.getContentPane());
                  jDialog1.getContentPane().setLayout(jDialog1Layout);

      jDialog1Layout.setHorizontalGroup(jDialog1Layout.createSequentialGrou
p()
                        .addContainerGap()
                        .addGroup(jDialog1Layout.createParallelGroup()
                            .addComponent(getPatientInfoPane(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 392,
GroupLayout.PREFERRED_SIZE)
                            .addComponent(getMedicalStatePane(),
GroupLayout.Alignment.LEADING, GroupLayout.PREFERRED_SIZE, 392,
GroupLayout.PREFERRED_SIZE))

      .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(getMessagePane(), 0, 389,
Short.MAX_VALUE)
                        .addContainerGap());

      jDialog1Layout.setVerticalGroup(jDialog1Layout.createSequentialGroup(
)
                        .addContainerGap()
                        .addGroup(jDialog1Layout.createParallelGroup()
                            .addGroup(GroupLayout.Alignment.LEADING,
jDialog1Layout.createSequentialGroup()
                                .addComponent(getPatientInfoPane(),
GroupLayout.PREFERRED_SIZE, 123, GroupLayout.PREFERRED_SIZE)
.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(getMedicalStatePane(), 0, 
210, Short.MAX_VALUE))
                            .addComponent(getMessagePane(), 
GroupLayout.Alignment.LEADING, 0, 339, Short.MAX_VALUE))
                        .addContainerGap());
            }
            return jDialog1;
      }

      private JLabel getMedicationLabel() {
            if(medicationLabel == null) {
                  medicationLabel = new JLabel();
                  medicationLabel.setText("Medication:");
            }
            return medicationLabel;
      }

}
```

Appendix B – Project plan (not updated)



ID	Task Name	Start	Finish
1	**Autumn semester 2009**	**Wed 2.9.09**	**Mon 7.12.09**
2	**Requirements capture**	**Wed 2.9.09**	**Sat 21.11.09**
3	Research nursing	Wed 2.9.09	Thu 12.11.09
4	Research triage	Sun 6.9.09	Sat 21.11.09
5	Schedule interview	Mon 26.10.09	Mon 26.10.09
6	Design interview	Sun 8.11.09	Wed 11.11.09
7	Hold interview	Thu 12.11.09	Thu 12.11.09
8	Research computerised triage	Wed 9.9.09	Mon 5.10.09
9	**Write interrim report**	**Thu 29.10.09**	**Sun 6.12.09**
10	Write related works	Thu 29.10.09	Sun 6.12.09
11	Write analysis and design	Thu 19.11.09	Sun 6.12.09
12	Hand in interrim report	Mon 7.12.09	Mon 7.12.09
13	Christmas holiday	Tue 15.12.09	Sun 10.1.10
14	Christmas holiday	Tue 15.12.09	Sun 10.1.10
15	**Spring semester 2010**	**Mon 11.1.10**	**Fri 16.4.10**
16	**Implementation phase**	**Mon 11.1.10**	**Sun 7.2.10**
17	Employ static analysis tools	Mon 11.1.10	Sun 7.2.10
18	Implement database in mySQL	Mon 11.1.10	Sun 7.2.10
19	Implement model in Java	Mon 11.1.10	Sun 7.2.10
20	Implement view(GUI) in Java	Mon 11.1.10	Sun 7.2.10
21	Implement controller in Java	Mon 11.1.10	Sun 7.2.10
22	Finish implementation	Mon 15.2.10	Mon 15.2.10
23	**Evaluation phase**	**Mon 15.2.10**	**Sun 28.3.10**
24	Conduct cognitive walkthrough	Mon 8.3.10	Wed 10.3.10
25	Conduct heuristic evaluation	Mon 8.3.10	Wed 10.3.10
26	Design code evaluation	Thu 11.3.10	Mon 15.3.10
27	Design unit test	Mon 15.3.10	Thu 18.3.10
28	Implement unit test	Thu 18.3.10	Sun 21.3.10
29	Conduct unit test	Sun 21.3.10	Mon 22.3.10
30	Design lab experiment	Mon 15.2.10	Sat 20.2.10
31	Schedule lab experiment	Sat 20.2.10	Fri 26.2.10
32	Conduct lab experiment	Mon 1.3.10	Mon 1.3.10
33	End evaluation	Sun 28.3.10	Sun 28.3.10
34	**Documentation phase**	**Mon 22.2.10**	**Thu 15.4.10**
35	Write user manual	Mon 22.2.10	Sun 7.3.10
36	Write developer manual	Mon 1.3.10	Sun 14.3.10
37	Write final report	Sat 6.3.10	Thu 15.4.10
38	Develop presentation	Sat 27.3.10	Mon 29.3.10
39	Hold presentation	Mon 29.3.10	Mon 29.3.10
40	Prepare demonstration	Mon 29.3.10	Wed 31.3.10
41	Hold demonstration	Thu 1.4.10	Thu 1.4.10
42	Hand in final report	Fri 16.4.10	Fri 16.4.10

Appendix C – ESI triage algorithm

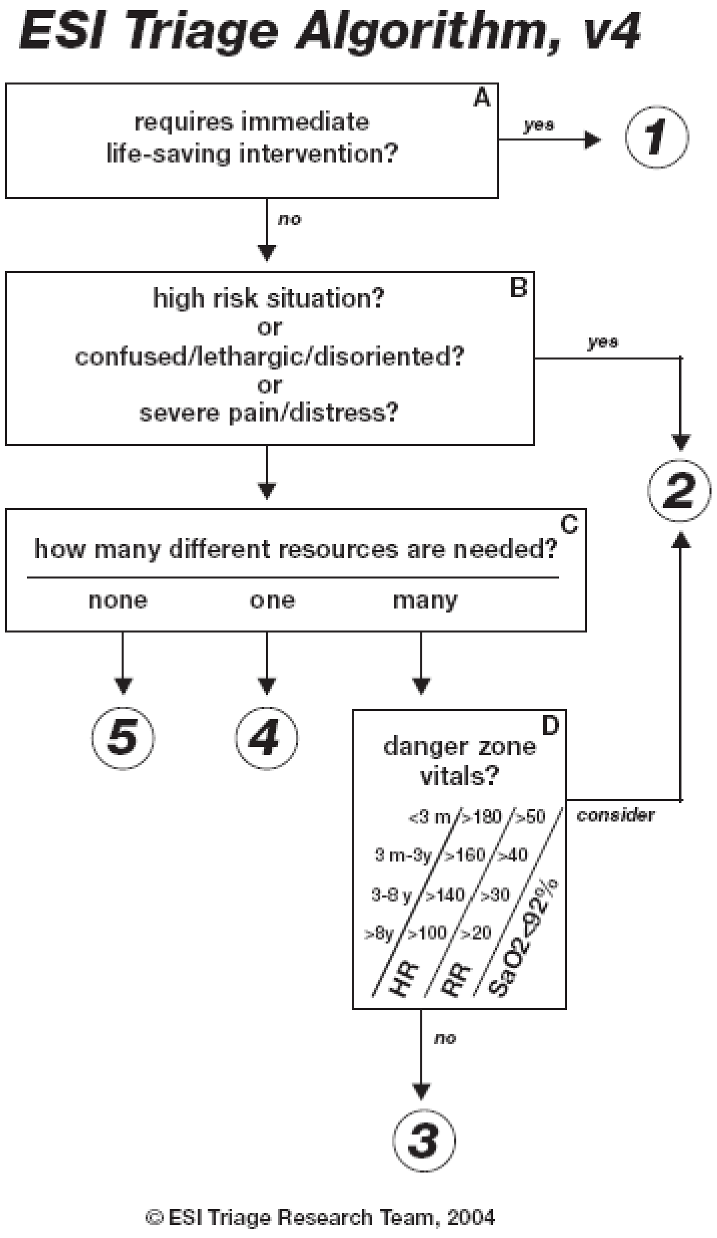

ESI Triage Algorithm, v4
A
requires immediate life-saving intervention?
yes
1
no
B
high risk situation? or confused/lethargic/disoriented? or severe pain/distress?
yes
2
C
how many different resources are needed?
none
one
many
5
4
D
danger zone vitals?
consider
<3 m >180 >50
3 m-3y >160 >40
3-8 y >140 >30
>8y >100 >20
HR
RR
SaO2<92%
no
3
© ESI Triage Research Team, 2004

Figure 16 – ESI triage algorithm (Gilboy et al., 2005, p. B-3).