DESIGN AND IMPLEMENTATION OF MOBILE PATIENT DATA COLLECTION AND TRANSMISSION SYSTEM FOR AN EMERGENCY AMBULANCE

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF INFORMATICS

OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMRE KOSEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF INFORMATION SYSTEMS

APRIL 2004

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this wok.

Emre Kösen

Approval of the Graduate School of Informatics

____

Prof. Dr. Neşe YALABIK

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

____

Assoc. Prof. Dr. Onur Demirörs

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

____

Asist. Prof. Dr. Erkan Mumcuoğlu

Supervisor

Examining Committee Members

Prof. Dr. Semih BİLGEN ________________________

Prof. Dr. Bülent CELASUN ________________________

Assoc. Prof. Dr. Nazife BAYKAL ________________________

Asist. Prof. Dr. Erkan MUMCUOĞLU ________________________

Dr. Alper ÖZKOÇAK ________________________

ABSTRACT

DESIGN AND IMPLEMENTATION OF MOBILE PATIENT DATA COLLECTION AND TRANSMISSION SYSTEM FOR AN EMERGENCY AMBULANCE

Kösen, Emre

MS., Department of Information Systems
Supervisor: Assist. Prof. Erkan Mumcuoğlu

April 2004, 102 pages

In this thesis, a low-cost system, called Mobile Ambulance, is designed and implemented that provides patient's medical data collection and transmission from a moving ambulance. The aim of the system is to decrease the waiting time for critical care patients to be seen at the emergency department (ED) at the same time to equip the emergency physician with the essential medical data before the patient arrives the ED. Mobile Ambulance is a multi-tiered distributed application composed of three components: ambulance component to capture patient's essential medical data (EMD) and to transmit it to the ED (transmission is wireless via General Packet Radio Service, GPRS), synchronization component (synch for short) to persist incoming data into the back-end database and to warn the emergency physician, and service component to analyze the patient's EMD.

Keywords: Ambulance, Emergency Department, Essential Medical Data Set, Multi-tiered Distributed Application Model, Wireless Communication

ÖZ

ACİL SERVİS AMBULANSI İÇİN TAŞINABİLİR HASTA VERİ TOPLAMA VE İLETİMİ SİSTEMİNİN TASARIMI VE UYGULAMASI

Kösen, Emre

Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. Erkan Mumcuoğlu

Nisan 2004, 102 sayfa

Bu tezde hareket halindeki bir ambulanstan hasta temel verilerinin toplanması ve acil service iletilmesini sağlayan Mobil Ambulans adlı düşük maliyetli bir sistem tasarlanmış ve geliştirilmiştir. Bu sistemin amacı kritik durumdaki hastaların acil serviste bekleme zamanlarını azaltmak aynı zamanda hasta acil servise varmadan doktoru gerekli tıbbi veriyle donatmaktır. Mobil Ambulans üç bileşenden oluşan çok katmanlı dağıtık bir uygulamadır: ambulans bileşeni hastanın temel tibbi verilerini almak, ve acil servise yollamak için kullanılır (verilerin yollanması kablosuz olarak GPRS ağı vasıtasıyla sağlanır), senkronizasyon bileşeni (kısaca synch) gelen verinin veritabanında saklanması ve nöbetci doktorun uyarılması için kullanılır, servis bileşeni ambulans seferlerinin listelenmesi ve hastanın temel tıbbi verilerinin incelenmesi için kullanılır.

Anahtar kelimeler: Ambulans, Acil Servis, Temel Tıbbi Veri Seti, Çok Katmanlı Dağıtık Uygulama Modeli, Kablosuz İletişim.

To my mother,

For your endless support and love

ACKNOWLEDGEMENTS

I am grateful to many individuals for the cooperation, support and encouragement they gave me while preparing this thesis. My special thanks go to my supervisor Assit. Prof. Dr. Erkan Mumcuoğlu.

Prof. Dr. Bülent Celasun and Dr. Yıldırım Karslıoğlu, Department of Pathology, Gülhane Military Medical Academy, prepared essential medical data set which is interpreted by emergency physician for a patient at a single emergency case.

I really appreciate Dr. Alper Özkoçak from Medline and Dr. Nazmiye Koyuncu from Bayındır Hospital for their supports while preparing an environment for the field tests. I also would like to thank to the Medline ambulance crew for their intensive contribution during the field tests.

Finally, I would like to thank my parents since they started me off with a good education, from which all else springs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DOM	Document Object Model
DTD	Document Type Definition
ECG	Electrocardiogram
ED	Emergency Department
EIS	Enterprise Information System
EMD	Essential Medical Data Set
ER	Entity Relationship
GHz	Giga Herz
GMMA	Gülhane Military Medical Academy
GSM	Global System for Mobile Communications
HEAL	Hospital and Emergency Ambulance Link
HIS	Hospital Information System
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
IT	Information Technology
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
JDBC	Java Database Connectivity
JDK	Java Development Kit
JRE	Java Runtime Environment
JSP	Java Server Page
JVM	Java Virtual Machine
MB	Mega Byte
MHz	Mega Herz
MSS	Main Success Scenario
PC	Personal Computer
PDA	Personal Digital Assistant
PJAE	Personal Java Application Environment

PSTN	Public Switched Telephone Network
RMI	Remote Method Invocation
SAX	Simple API for XML
SDK	Software Development Kit
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
UML	Unified Modeling Language
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Motivations of the Study

Ambulances were for many decades regarded primarily as means for transportation of the sick and wounded from homes, work sites and public places to medical institutions where real treatment usually began. Unfortunately, hospital staff would have known about ambulance transported patients only on arrival at the emergency department or sometimes beforehand via unstructured and prone to error voice communication with cellular phone or radio between ambulance and ED staff, if such equipment are available. This situation causes delay for setting the proper diagnosis and beginning the treatment. Delay is the enemy for patients within vital conditions. So evaluating the patient's essential medical data as soon as possible in an emergency case would be helpful for the hospital staff to make the necessary preparations before the patient's arrival.

A number of evolving health care issues have shown the need for diagnosis to be made as early as possible and treatments initiated earlier:

- Identification of an increasing array of time-sensitive diseases and treatments and the realization that earlier diagnosis and earlier treatments create the potential for better outcomes and lesser mortality.
- The acceptance that health care is a continuous process from the point of injury/sickness to the hospital and that prompt structured and accurate knowledge of care provided earlier can only enhance the level of care to be given at the subsequent level.
- The realization that ambulance crew (paramedics) can be trained to provide many forms of urgent medical care in ambulances and other pre-hospital environments has opened up the potential for earlier medical interventions to the patient's eventual benefit.

With the increasingly useful role of ambulances and paramedics in the provision of emergency medical care, receiving hospitals are being forced to polish up their acts in ensuring that they review their investigational and treatment processes so as to maximize the benefits of patients arriving in hospital in a medically better condition than previously.

1.2 Problem Statement

While at one time, hospital staff would have known about ambulance transported patients only on arrival at the emergency department, the advancements in voice communication devices has meant that ambulance crew are now able to verbally alert the hospital of impending arrivals. Such verbal communications (radio and cellular phone) have had their limitations in terms of quality of voice transmission. Also, the descriptive capacity of speech is limited compared with vision. Given a visual or pictorial representation of an incident scene, an injured person or a document, speech is not necessary to supply the general background, but can be used instead to elaborate the detail.

Fax transfer is to allow remote interpretation of electrocardiograms (ECGs) acquired in hospital [1]. Use of the mobile telephone to transfer ECGs from ambulance to hospital is also well established [2]. A chief advantage is the immediate availability of a hard copy of a document or a photograph at a remote location.

Video-recording has been used successfully, as an adjunct to supervision and training, to discover variation between reported and observed practice in a hospital resuscitation room [3]. It was possible to determine deviations from the standard resuscitation protocols that were thought to have been introduced successfully. An early experiment with ambulance television took the form of simple video-recording without the transmission of pictures in real time [4]. This gave hospital staff a realistic understanding of pre-hospital care and ambulance practice, and also provided material that could be used repeatedly for teaching and training, of both hospital staff and paramedics.

The main problem tackled in this study is compiling the essential medical data set for emergency patients and conveying them to the receiving healthcare institution as soon as possible using a low-cost system designed.

1.3 Proposed Solution

None of the methods introduced at Section 1.2 permits the rapid information transfer anticipated when radio conversation is coupled with essential medical data transmission. In this thesis, a solution is provided to capture and transmit increasing variety of information during an ambulance run, and record communication between ambulance and emergency department.

This thesis proposes a multi-tiered distributed application, Mobile Ambulance, as a solution to address the requirements above. Mobile Ambulance has been developed as a part of METU project, AFP 2002-07-04-02. Mobile Ambulance is differentiated from its currently available counterparts in modeling essential medical data set that is necessary for an emergency physician to set preliminary diagnosis and to begin definitive therapy at an emergency case. Mobile Ambulance has been designed and developed with particular emphasis on using open source software and vendor independence as expected in an academic study. Another fundamental aim was to keep Mobile Ambulance within the boundaries of a low-cost system.

1.4 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 discusses technologies and industry standards enable us to implement the Mobile Ambulance system. It also summarizes the related systems previously established.

Chapter 3 summarizes the underlying business model of an emergency service. It also discusses the requirements of the Mobile Ambulance system.

Chapter 4 and 5 gives the design and implementation details of the Mobile Ambulance system respectively. Chapter 5 also summarizes the testing efforts.

Finally Chapter 6 concludes the thesis with a discussion issue about integration with legacy hospital information systems and points out several development challenges such as ensuring the data security.

CHAPTER 2

RELATED WORK

This chapter provides the details of literature survey. Some existing related ambulance systems are briefly reviewed at Section 2.1. The technologies and industry standards that are used to realize Mobile Ambulance system are summarized at Section 2.2.

2.1 Related Ambulance Systems

In this section, related ambulance systems, in terms of idea, technology and implementation are presented. They are mostly research projects which are currently on the field. These systems can be categorized into three parts:

- Vital signs transmission systems: the ambulance crew captures the patient’s vital signs and transmits this information from a moving ambulance to the ED based station via a wireless public data network.
- Video images transmission systems: patient's live video images are captured by video cameras attached in the ambulance and video stream is delivered

real-time to the ED based station via wireless network such as satellite communication or mobile telephone network.

- ECGs transmission systems: ambulance is equipped with an ECG recorder, which is connected to a portable computer and coupled to a cellular telephone. The ambulance crew record patient's ECG and transmit this information to the ED based station via mobile telephone network.

2.1.1 Hospital and Emergency Ambulance Link (HEAL)

Hospital and Emergency Ambulance Link (HEAL) [5] was developed to make records of communication between ambulance crew and emergency physician and to capture and transmit the increasing variety of information that ambulance crew were able to obtain.

Before starting the HEAL project, the following problems had been addressed: data entry in a moving ambulance, electromagnetic interference by communication equipment in the ambulance on vital signs and ECG equipment, and whether it would be possible to capture all or only part of the information, required by ambulance crew and hospital staff, electronically.

Before starting the HEAL project, the following objectives had been set: conveying patient's medical information to the ED, providing ambulance paramedics with a simpler and quicker way to communicate with emergency physicians, improving the documentation of patient management events during the ambulance run, allowing a mechanism to improve the care of patients during ambulance runs to implement better quality control.

There were four modules in the HEAL system:

- **Advance patient details module**: captures patient demographic information, vital signs and other medical management information, processes these and conveys the processed information to emergency physicians before the arrival of patient to the ED.
- **Ambulance incident management module**: processes and archives information received from the ambulances.
- **Drug request and authorization module**: facilitates the process of seeking approval from the emergency physician to administer certain special drugs to patients.
- **Text communication module**: facilitates the exchange of interactive messages between the ambulance and the ED.

Ambulance workflow with the HEAL system can be described as follows:

- At the start of every shift of work, the ambulance paramedic logs into the system. Upon confirmation, the paramedic enters the ambulance number.
- At the start of an ambulance run, the paramedic keys in the relevant data of the incident into the system.
- After reaching the patient and providing initial care, the paramedic captures vital signs such as blood pressure, heart rate, pulse oximetry and ECG data by using a portable monitor cum defibrillator.

- On the way back to the ED with the patient, the paramedic downloads these data into a notebook. This notebook was equipped with a user-friendly touch screen interface. The paramedic also enters all clinical information that is obtained from or about the patient into the notebook. The paramedic would transmit these data to the ED as initial pre-arrival information. Thereby providing ED staff with valuable indices on the medical condition of the patient prior to arrival at the department.
- When a drug authorization is needed, the paramedic enters the drug request information and transmits it to the ED. Physicians at the ED than transmit the authorization to the requester.
- Treatment protocols are built into the system, the ambulance paramedic can be prompted to perform actions required by the protocol, based on detection of a number of fields that satisfied given conditions.
- A siren alarm is used to warn ED personnel of such an incoming patient.
- The text communication module facilitates the exchange of interactive electronic messages between the ambulance crew and the ED.

A three-month review and evaluation was conducted with respect to three criteria: technical, operational and clinical effectiveness. The results of the evaluation are addressed below:

- Technical effectiveness: three aspects of the technical effectiveness were examined: *System availability*, *Transmission times* and *amount of data transmitted*.

    - *System availability*: initially only 69.1% of ambulance runs used the HEAL system. By the third month, this increased to 93.9%.
    - *Transmission times*: air transmission time for data was ~ 4 s unless ECG waveforms were also transmitted resulting in transmission times exceeding 60 s.
    - *Amount of data transmitted*: at least 68% of data was transmissible in 75% of HEAL ambulances as opposed to only 25% in less than 5% of non-HEAL ambulances.
- Operational effectiveness: two indices of operational effectiveness were measured: *time spent by the paramedic in the ED*, *time for data entry by the paramedic*.
    - *Time spent by the paramedic in the ED*: paramedics' time in the ED decreased from 15 to 8 min as a result of HEAL.
    - *Time for data entry by the paramedic*: it was possible to capture a complete ambulance case record electronically at a mean time of 94 s vs. 7 min 7 s for the traditional written record.
- Clinical effectiveness: two aspects of the clinical effectiveness were examined: *waiting time to be attended by the doctors*, *treatment protocol discipline*.
    - *Waiting time to be attended by the doctors*: the waiting time for critical care patients to be seen at the ED decreased from 35 to 17 min if bought by HEAL ambulances.

- *Treatment protocol discipline*: the HEAL system was able to effectively prompt paramedics in carrying out critical aspects of treatment in close to 100% of instances.

2.1.2 HECTOR

The HECTOR research project [6] was an integrated system based on existing technology and new technologies in multimedia and telecommunication. It focused to improve coordination and management in health emergencies. The project also sought to verify that the emergency health care could benefit from innovative telecommunication. A virtual presence of medical expertise at the point of care could minimize intervention time.

In the scope of the project, an emergency ambulance was equipped with three video cameras and a system for transmitting slow-scan video pictures through a GSM cellular telephone link to a hospital emergency department. In addition, a helmet-mounted camera was used with a wireless transmission link to the ambulance and thence the hospital. By this way, remote supervision was provided by a senior emergency physician present at the same time the pictures were transmitted.

Video pictures were transmitted at a resolution of 320x240 pixels and a frame rate of 15 pictures per minute. Although this speed does not allow the reproduction of continues, fluid movement, it allows detailed inspection of each frame before the next received.

Speech was transmitted using a separate hand-held telephone. Researchers added that an obvious next step would be to complement the helmet-mounted camera with an ear piece and microphone to allow hands-free conversation. Thus the picture and speech will be transmitted by a single telephone.

Further developments are to superimpose, on the hospital video screen, the output from monitoring devices such as those for non-invasive blood pressure measurement, pulse rate, oxygen saturation and respiratory rate. Finally it is intended to transmit the remote observer's image so that the paramedic or the patient can see it in the ambulance [6].

2.1.3 SendECG

In the scope of this project [7], an ambulance was equipped with a portable digital ECG recorder and ECG analysis software, a notebook computer, a PC GSM data card, a cellular telephone, and transmission software. SendECG used data compression algorithms and error-correction protocols for fast, error-free transmission over the GSM network and PSTN lines with the store-and-forward method to a base station at emergency department. The base station consisted of a desktop computer, running the ECG analysis software, a modem and a printer.

In 90% of the cases successful transmission was achieved on the first attempt. A second or third attempt was required in six cases and in one case data transmission was impossible. The most common reason for transmission failure (50%) was ambulance speed, while weather and interference caused by large structures were also important reasons.

The average time for ECG acquisition by paramedics was 2 min ± 0.5, depending on the patient's cooperation and vehicle speed. Transmission via GSM took 34 s ± 14 at 9600 bits per second. Thus ECG data would be available to be interpreted by emergency physicians in approximately 2.5 min while the ambulance was en route to hospital.

As maximum benefit from thrombolytic therapy in acute myocardial infarction (AMI) is obtained with early therapy, it is important for treatment strategies to decrease time delays. The time elapsed for seeking medical care depends on the patient, thus they can only try to minimize the pre-hospital and in-hospital delays.

Researchers concluded that ECG transmission from a moving ambulance is fast and feasible, can be performed easily by paramedics and significantly decreases the time needed for in-hospital evaluation [7].

2.1.4 Kaan

Kaan project [8] was implemented by Başarı Elektronics in Turkey. Inherently the project depends on vehicle tracking rather than serving for conveying patient's demographic and medical data to a healthcare institution. In the scope of the project, a portable onboard computer is manufactured and coupled with a GSM phone and a GPS receiver. Embedded software, running on the portable computer, retrieves location data (latitude, longitude, speed and direction) from the GPS receiver and sends it to the server via the GSM phone by this way vehicle tracking is achieved.

2.2 Enabling Technologies and Standards

Mobile Ambulance system requires capturing patient's essential medical data at the point-of-injury/sickness and to transmit it to the ED based station for persistence and further evaluation. Furthermore, the system requires running on commercially available hardware and operating system without need of any proprietary system or technology. Finally it requires running in a wide area network once the network coverage and power supply are available.

For instance, to implement such a system, we need some sort of infrastructure containing fundamental services for communication and persistence. This infrastructure composed of wireless network and a relational database management system. Furthermore, we need an industry standardized application model. By the help of this model, we distribute application components to different machines and divide the components into different tiers for the ease of design, implementation and even maintenance.

In this section, the emerging technologies and industry standards, which constitute the building blocks of Mobile Ambulance system, are briefly summarized. In this respect, Multi-tiered Distributed Application Model and Wireless Mobile Computing are covered.

2.2.1 Multi-tiered Distributed Application Model

Distributed software applications are difficult to develop. They are large and complex – but they are important and increasingly employed. Terms such as Internet, intranet and extranet, e-commerce and virtual private network prove that this is the case. Developers, however, struggle with the heterogeneity of hardware, high transmission times during remote communication, failure and unreliability of system components, and many other problems.

At the end of the 80's, Sun Microsystems wrote the motto "The Network is the Computer" on their banners. Their vision is to make networks and distributed systems as simple and reliable as we have become accustomed to from normal computers.

An essential corner stone on the way to this goal is Java Programming Language [9]. This language breathes the spirit of this vision. Java helps to reduce the complexity of distributed systems by integrating important mechanisms for programming of distributed systems directly into the language, and it overcomes the heterogeneity through its platform independence.

Concurrency, Distribution and Persistence

Development and programming of distributed systems are substantially more complex than the programming of local applications, as they are typically used on standard workstations. Of course, even the development of sequential programs is quite complex, but through distribution, several dimensions of complexity are added. While local sequential programs can be executed only on exactly one computer, computers in the context of this section are understood as only one single user control devices on which at least one Java Virtual Machine exists and which are capable of communicating over a network and their data can be stored in a file without major difficulties. Distributed applications, on the other hand, run on several spatially distributed computers, need to communicate over a relatively slow and potentially insecure network, are used by several users simultaneously, and access and security of shared data need to be secured through transactions, and a consistent central or distributed data management needs to be guaranteed.

Three of these aspects in particular: *concurrency*, *distribution* and *persistence* are addressed below.

Concurrency is the actual or apparent parallelism of control flows. Concurrency can occur either between different processors, as in a multi-processor system, or on a single-processor. Modern processor architectures allow several processes to be executed simultaneously on one processor. Obviously, these processes do not really run simultaneously, but are allocated staggered time slices of the processor's computing time. For the user, however, these processes appear to be running in parallel. There are two types of such processes. On the one hand, they can be completely isolated from each other, so that they each have a separate execution environment. In this case they are called heavyweight processes. On the other hand, several control flows can be processed within one execution context, the so-called lightweight processes which can cooperate with each other through a common address space. In Java, lightweight processes, known here under the name of threads, have received user-friendly integration: programmers can immediately produce and manage them within the language.

Distribution is the logical or physical spatial distance of objects from each other. Two objects, which cannot use the ordinary method invocation for their communication, but need to employ mechanism of remote communication, are distributed with respect to each other. This is the case when they are located on different computers and therefore are spatially separated. But also when they are located on the same computer, but in different address spaces (in different heavy weight processes), they are -logically- distributed. Obviously, different objects that have nothing to do with each other, but are located on several computers, are also distributed with respect to each other, but are of no great significance for us. Here, distribution shall be understood as the separation of objects that stand in relation to each other. Between these, there is the problem of how they can locate each other, access each other, and communicate with each other.

Persistence is the long-term storage of data or objects on non-volatile media. Data or objects, which are not explicitly saved in some form or the other, no longer exist after a program is terminated. They are described as transient. However, much data needs to be retained even when the program is terminated or the computer is shut down. Therefore, such data needs to be stored in some way or other in order to be available again when the computer is re-started. In this case we speak about persistent data or objects. Persistence achieves the distribution of data and objects in time.

These three aspects of distributed programming are relatively independent from each other and are dealt with totally different techniques. Therefore, we could describe them as being orthogonally related to each other and talk about dimensions. These three dimensions are shown in Figure 2.1.

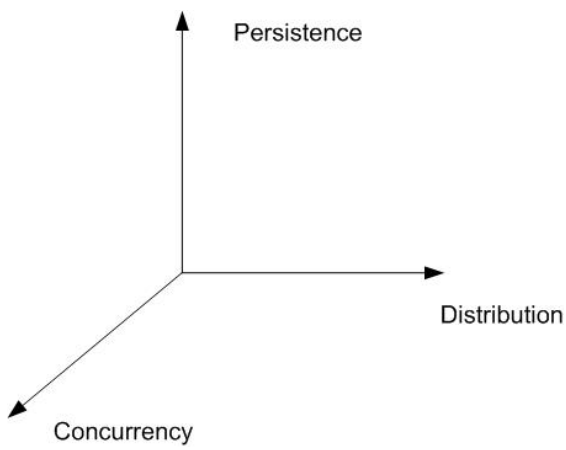


Figure 2.1 Dimensions of distributed programming [10]

With regard to concurrency Java threads are used in this work. Java supports concurrency in the language itself with the aid of lightweight processes. In most languages used previously, this was possible only by direct access to operating system libraries, which made working difficult and portability almost impossible. Java enables concurrency not only independently from the platform and with immediate language support, but also makes things fairy simple.

The communication of programs in distributed systems can be solved in extremely different ways, which in the end all boil down to the same thing: bits and bytes are shifted across the network. However, this simple data communication can be abstracted on different levels. On the lowest level we find the mechanism of transmitting data streams via sockets. Here, data is transferred from one computer to another without further interpretation. All subsequent mechanisms built on this one; however, they confer specific semantics to the transmitted data and relieve programmers from a large part of work, so they can concentrate better on their real tasks. From the programmer's point of view it is preferable not to have to communicate from one computer to another, but from object to object, as it happens in object-oriented programming. This mechanism is Remote Method Invocation (RMI) [11]. However this mechanism only works when the same semantics is used on both sides, which means that the same language must be used and the location of the remote object is known. When you abstract further and conceal location and language of a remote object, you arrive at Common Object Request Broker Architecture (CORBA) [12]. Here, the programmer can use a remote object without knowing where it is and in which language it has been implemented. The next step of abstraction consists in the possibility of changing the position of an object at runtime, which is described as migration. And when objects not only let themselves be moved but do so on their own (autonomously), we talk about mobile agents [10].

Fundamentally, there are three techniques for permanent storage of data, that is, for persistence. The most basic one, the simple storage of serialized data streams into files. The most widely diffused and reliable one is represented by relational databases which store data in tables that can be connected with each other. These permit the search of data through structured enquiries with Structured Query Language (SQL). They also permit the coordination of access by several users and the compliance with the so-called ACID principles (Atomicity, Consistency, Isolation, and Durability). For Java, access to relational databases is standardized by Java Database Connectivity (JDBC) [13]. However, the storage of data in tables means a rupture with object-oriented programming languages which no longer deal with data, but with encapsulated objects. To avoid this, object-oriented databases were developed which fit much more seamlessly into languages such as Java.

These three dimensions and their respective techniques that were chosen in this work are pointed out in Figure 2.2.

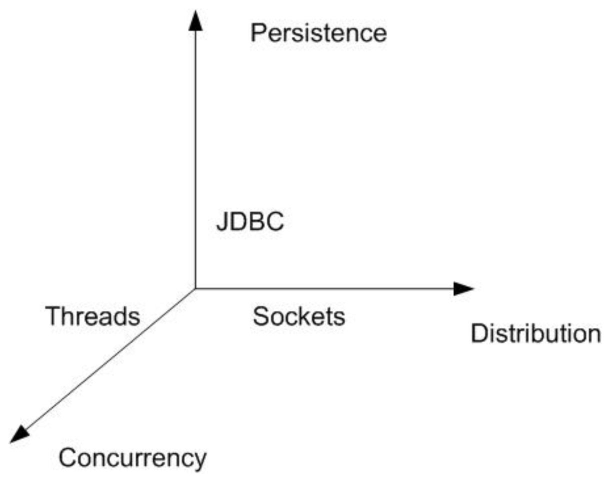


Figure 2.2 Distributed programming techniques chosen in the thesis

Java2 Enterprise Edition

In the thesis, Java2 Enterprise Edition (J2EE) platform [14] is used as an application model for Mobile Ambulance system's service component. In J2EE platform, Application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multitiered J2EE environment to which the application component belongs. Figure 2.3 shows multitiered J2EE application divided into the tiers described in the following list.

- Client-tier components run on the client machine.
- Web-tier components run on the J2EE server.
- Business-tier components run on the J2EE server.
- Enterprise information system (EIS)-tier software runs on the EIS server.

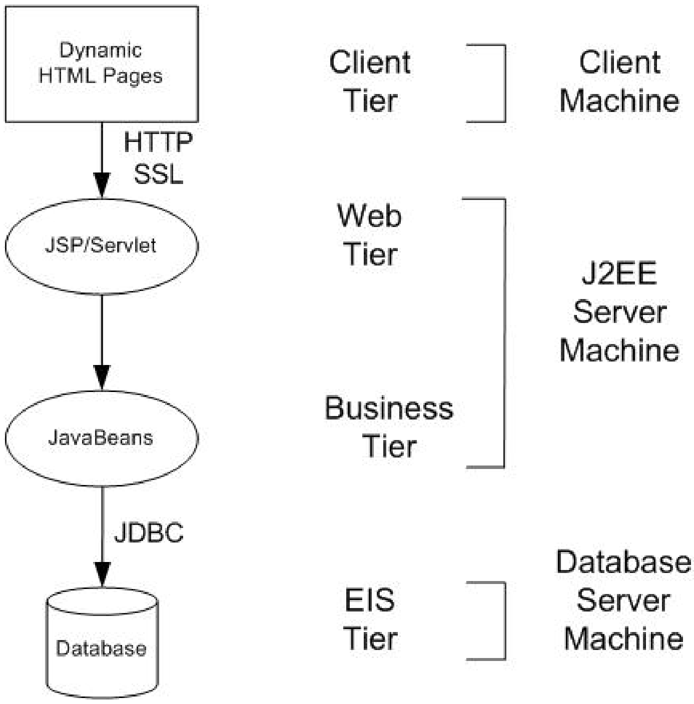


Figure 2.3 Multitiered J2EE application model used in the thesis

Although a J2EE application can consist of the three or four tiers shown in Figure 2.3, J2EE multitiered applications are generally considered to be three tiered applications because they are distributed over three different locations: client machines, the J2EE server machine, and the database machine at the back end. Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage.

Previously, we have described distributed software applications from three independent dimensions: concurreny, persistance and distribution respectively, and we have chosen threads, JDBC and sockets as the implementation techniques we used in the thesis. Fortunately, J2EE architecture follows this rule. Since J2EE server is a multithreaded application server on which many client requests are handled concurrently, J2EE platform requires a relational database accessible with JDBC Application Programming Interface(API) for persistance. Client/server communication is established via Hyper Text Transfer Protocol (HTTP) which is the protocol of web communication over Transmission Control Protocol/Internet Protocol (TCP/IP) and Secure Socket Layer (SSL) [15].

2.2.2 Wireless Mobile Computing

First we should define a mobile computing device, which is simply a small computing device that has limited processor speed and available memory when compared to a desktop or server computer. Then we will understand the basics of wireless networking that enable mobile devices to access external resources and services. Finally we will combine these two aspects and define wireless mobile computing with the advent of Java technology.

Mobile Computing Devices

Mobile computing device in the context of the thesis is understood as a PDA, Personal Digital Assistant, such as Palm VII or Compaq IPAQ. Admittedly, the line separating a mobile computing device from the capabilities of a full-fledged desktop or server computer is rather blurry as thinner notebook computers, web pads, and other devices come onto the market. In general, however we can use the following rule of thumb to separate the two camps. If you never worry about the speed of your application or about the amount of memory that it requires (within normal limits), you are not computing with a mobile device.

Four of the aspects of mobile devices in particular: memory and storage capacity, processor power, input/output methods, and the form factor are addressed below.

Memory capacity is an important measuring stick, and at first glance, a mobile device might not seem that limited. It is not unusual, for example, for a modern PDA to have 32 MB or more of memory. While this amount is less than the 256 MB or 512 MB that you will get in a typical desktop system, it is not a debilitating amount.

Total storage capacity is the sum of a device's online and offline storage capacity. Online storage capacity is simply another term for memory capacity – the amount of memory available to store runtime application data (stack and heap), system data, built-in applications and operating system. Online storage is characterized by instant availability and might or might not be persistent. Offline storage capacity, on the other hand, is the capacity of secondary, persistent storage modules such as memory sticks. Offline storage is usually slower to access and is not normally available for storing runtime application data.

Mobile devices are typically designed as self-contained units without a dedicated electrical power connection. In other words, they are battery-operated. One of the primary goals of a battery-operated system is to conserve battery power. The longer that a device can function without replacing or recharging its batteries, the more useful the device is. To conserve battery power, then, manufacturers of mobile devices use central processing units (CPUs) that are optimized for low power scenarios. These processors are slower than normal CPUs and often based on older chip models. A processor running at a clock speed of 200 MHz is not unusually slow for such devices. Of course 200 MHz is quite slower than the 2.4 GHz clock speed that you will find in today's desktop computers.

Mobile devices are more likely to have alternative input/output (I/O) methods than desktop computers. For portable devices the form factor, which we discuss next, often determines what I/O methods are acceptable. Size constraints make the regular I/O methods of a desktop computer (full size keyboard, monitor and mouse) impractical or impossible for mobile devices.

The touch-sensitive screen, perhaps the most common input method for PDAs, also doubles as an output method. They are most often used in conjunction with some kind of writing or stroking device (usually referred to as pen or stylus) and on-board character recognition software. A conventional keyboard is the most accurate and fastest way for most users to enter text data. On mobile devices a full size keyboard is out of the question. Instead, most devices that have keyboards reduce both the number and the size of the keys. A more sophisticated input method is voice recognition. Full voice recognition requires a lot of processing power and is still far from perfect, but it is perhaps the most promising input method for mobile devices [16]. And of course there are push buttons, Compaq IPAQ, for example, has four buttons below its screen. Specialized input methods, such as barcode readers, are also available with some devices.

The most common output method is obviously the display screen. Just like a desktop computer, mobile devices often support some kind of display screen. The screen is quite small and is monochrome or color. Apart from the screen, indicator lights can also be used to communicate with the user. Individual lights can provide a visual indication of hardware events such as network activity and power usage or software events such as message availability. If a speaker included, a device can use simple audio signals such as rings and beeps. More sophisticated devices can play back messages and music or synthesize voices.

The form factor of a device refers to its physical size, shape and weight. The intended use of the device places absolute limits on these values, but in general, a smaller form factor is better. The form factor of a hand-held device such as Compaq IPAQ requires it to fit comfortably in one hand, freeing the other hand to use stylus on the touch-sensitive screen [17].

Wireless Networks

We can broadly classify portable devices into two groups based on the type of network connection that they support. The first group includes devices without a permanent network connection, which we can refer to as occasionally connected devices. Most hand-held devices support serial communication with a desktop computer, for example, through the use of a cradle or a simple cable that is permanently connected to the computer. You place the device in its cradle to do its networking (primarily data synchronization). The second group includes devices with a permanent network connection, which we refer to as always-connected devices. If portable, these devices use some form of wireless communication either infrared or radio- based to exchange data with other devices and computers. But none of these communication methods are currently as fast as the typical Ethernet connection that you will find on a desktop computer, although new technologies such as GPRS will narrow the gap.

GPRS is a new nonvoice value added service that allows information to be sent and received across a mobile telephone network. It supplements today's Circuit Switched Data. Theoretical maximum speeds of up to 171.2 kilobits per second (kbps) are achievable with GPRS using all eight timeslots at the same time. This is about three times as fast as the data transmission speeds possible over today's fixed telecommunications networks and ten times as fast as current Circuit Switched Data services on GSM networks. By allowing information to be transmitted more quickly, immediately and efficiently across the mobile network, GPRS may well be a relatively less costly mobile data service compared to Circuit Switched Data. GPRS facilitates instant connections whereby information can be sent or received immediately as the need arises, subject to radio coverage. No dial-up modem connection is necessary. This is why GPRS users are sometimes referred to be as being “always connected”. Immediacy is one of the advantages of GPRS when compared to Circuit Switched Data [18]. High immediacy is a very important feature for time critical applications such as, in our case, tranmission of patient's essential medical data to ED based station where it would be unacceptable to keep the paramedic waiting for even thirty extra seconds.

Java Technologies for Embedded Devices

The PersonalJava Application Environment (PJAE) [19] is a Java application runtime environment which allows Java-based applications to run. In the case of a desktop computer environment, Java Runtime Environment (JRE) is utilized to run Java-based software. PJAE, on the otherhand, is an application development environment specifically designed for set-top boxes, WebPhones, SmartPhones, high-end PDAs, and other consumer electronic devices with built-in networking capability.

Due to the compact size of these mentioned devices, they are provided with limited hardware resources, such as smaller memory size and slower CPU processing cycle, when compared with the desktops as we mentioned previously. Additionally, these devices are equipped with a smaller display which may be limiting to programmers while working with graphical user interfaces (GUI). For these portable devices, working with JRE would not be suitable.

PersonalJava is introduced so that the characteristics of these portable networking devices can be enhanced while providing a Java development environment to programmers. Meanwhile, the Java framework has been changing since the emergence of Java2, PersonalJava is soon to be integrated as part of Java2 Micro Edition (J2ME), and the specification and package reconfigurations are being worked on; it will be reorganized as J2ME Personal Profile [20].

The PJAE specification is based on the PersonalJava API, which differs from JRE in its specification. Although PersonalJava API is created with Java API in its base, PJAE is organized by its target device subsets. The PersonalJava Virtual Machine is then a compact version of the familiar Java Virtual Machine (JVM), to meet the limited memory capacity of these portable devices (JVM and PersonalJava Virtual Machine share the same Java language / Virtual Machine specification).

PJAE includes class libraries which support both the PersonalJava Virtual Machine and PersonalJava API. Although each PJAE installation is custom made for each target device, application portability among PJAE machines is guaranteed as long as the code supports the API. However if you include the optional API, machine-specific libraries, and applications with native methods, the portability is not guaranteed.

CHAPTER 3

REQUIREMENTS ANALYSIS OF THE MOBILE AMBULANCE SYSTEM

This chapter provides the details of requirement analysis. Business model, which illustrates the business domain of a generic emergency service to clarify the feasibility, is summarized in Section 3.1. The requirements and system architecture are summarized in Section 3.2.

3.1 Business Model

In this section, business processes of a generic emergency department are discussed. This discussion will be beneficial to grasp the ambience of emergency department and to understand the feasibility of our study.

Business processes of an emergency department can be separated into four parts:

- Taking the emergency call
- Reaching to the patient
- Providing initial care

- Transporting the patient to the ED
- Treatment at ED

Taking the Emergency Call

While an emergency incident occurs, someone calls the call center and this is just the start of the whole chain. During the emergency call phase, duty officer, who is responsible to answer the calls and inform the ambulance crew, captures the necessary information about call such as the address of the incident, complaint of the patient etc. These information are valuable for the ambulance crew to reach to the patient and to provide initial care to the casualty at site. Unfortunately these information cannot be validated at the emergency call phase, thus the ambulance crew can easily be misguided.

Reaching to the patient

Since the address of the incident has been known, it is the driver's responsibility to deliver the paramedics to the patient as soon as possible. Unfortunately this process may not be as easy as we guess. There are different situations influencing the time passed on reaching to the patient, for example, the address of the incident may be too far, the ambulance can get caught in the traffic jam or weather/road conditions may be unexpectedly worse, or even the ambulance can be broken down or can have an accident and make the ambulance crew get stuck on route to the patient.

Providing Initial Care

After the ambulance crew reaches to the patient, paramedics provide initial care. Of course the steps of the emergency care are the subjects of totally different discipline, Emergency Medicine. Thus it is not proper to discuss these steps in detail here, but it will be beneficial to illustrate briefly what the situation is at emergency care phase.

It is not possible to define a generic treatment for the emergency patients since the reasons of injury or sickness change widely. However it will be useful for paramedics to check vital parameters listed below to decrease time delay at an emergency case: Consciousness, Respiration, Circulation, Bleeding, Shock, Cracked, Burnt and KİBAS (Turkish acronym, Kafa İçi Basınç Artışı Sendromu).

Eventually checking these physiological parameters gives valuable information to the emergency physician about the patient, and saves significant amount of time for setting the diagnosis and begining the treatment at the ED. For some specific cases, patient's ECG is also recorded and faxed to the ED in case ambulance is equipped with the necessary equipment.

Transporting the Patient to the ED

After the patient was carried into the ambulance, it is again driver's responsibility to transport the patient to the ED as soon as possible. Unfortunately this process may not be easy as discussed before.

During the transportation, paramedics may often continue initial care process in the ambulance and communicate with the call center (via a radio or cellular phone). They inform the duty officer about the situation of the patient, and duty officer conveys the information from the call center to the ED. However verbal communications have their limitations in terms of quality of voice transmission. Also, the descriptive capacity of speech is limited compared with vision. More importantly, there may be misunderstandings between the ambulance-call center and call center-ED voice communication.

Treatment at ED

After the patient is transported to the ED, patient is checked in and treatment process begins. During the treatment process emergency physicians often have to make critical decisions very quickly with little or no background information. It may also be possible to directly transfer the patient to another department. For example, patient may have an acute myocardial infarction thus may need to be attended in the cardiology department.

In the thesis, we do not intend to propose solutions for handling the exceptions we discussed above. Moreover we do not intend to decrease the time elapsed for seeking medical care. However, we concentrate on how to decrease the in-hospital delay for the achievement of early therapy.

3.2 Requirements and System Architecture

For determination of system requirements, an ambulance run scenario is written based on the assumptions and constraints. Subsequently, use cases for capturing the functional requirements of the system are described. Unified Modeling Language (UML) is used for modeling [21]. With this study, we intend to describe the typical interactions between the user of the system and the system itself. The rest of the section describes logical database requirements and the system architecture.

3.2.1 Assumptions and Constraints

Assumptions

- Paramedics are experienced in using PDAs.
- Emergency physician is experienced in using desktop computers.
- System components communicate via GPRS network and network coverage is always available.
- Ambulance crew is composed of doctor, paramedic and driver.
- Ambulance is equipped with a radio or cellular phone to communicate with the call center.
- Hospital has an existing Hospital Information System (HIS) that provides patients' medical history information including past treatments and existing conditions.

Constraints

- Client-server communication is based on TCP/IP.
- PDA has limited memory (64 MB).
- PDA has limited display (240 x 320 pixels).
- PDA is battery-powered (Lithium Polymer).
- In an emergency incident, it is unacceptable to take significant time of the paramedic during her interaction with the system. Thus the system should provide an easy to use GUI.
- Our aim is not to provide a desktop environment in ED with an emergency personnel always standing by. The very nature of mobility implies that the personnel may typically be performing another task when the information is received about the impending arrival of the ambulance. Thus the system should use some kind of alert to warn the emergency personnel.

3.2.2 Ambulance Run Scenario

To clarify requirements of the system, we provide an ambulance run scenario which captures the key characteristics of an emergency incident. This scenario is derived from our observation of ED personnel's behavior during the incident. It is intended to be brief and domain-specific.

The Scenario

Ambulance run starts with an incoming call to the call center. The duty officer captures the relevant data of the incident and conveys this information to the related ambulance crew. After reaching the patient and providing initial care, ambulance crew returns to the ambulance with the patient. Paramedic activates the Mobile Ambulance system. The patient's demographic information (obligatory information to register the patient into the HIS) is entered first. Then the patient's medical information is entered by checking the vital parameters discussed in *providing initial care* part of Section 3.1. Finally this information is transmitted to the ED based station. Upon the completion of transmission, the system uses an alert message to warn the ED staff about the impending arrival. Based on the patient's medical information, emergency physician immediately starts preparations prior to the arrival of the ambulance. Furthermore, if the situation necessitates an early treatment, emergency physician directs the ambulance crew via the call center and give directions to them. Based on the patient's demographic information, patient is registered into HIS. If patient already has an existing record in HIS then emergency physician may retrieve complete patient information including past treatments and existing conditions. After the ambulance arrives to the ED, emergency encounter phase starts. During emergency encounter, patient's incoming essential medical data will definitely be useful for emergency physician to efficiently treat the patient and improve the quality of care.

3.2.3 Functional Requirements

Use cases and actors of the system are derived from the scenario mentioned in the previous section. Use case diagram is illustrated in Figure 3.1.

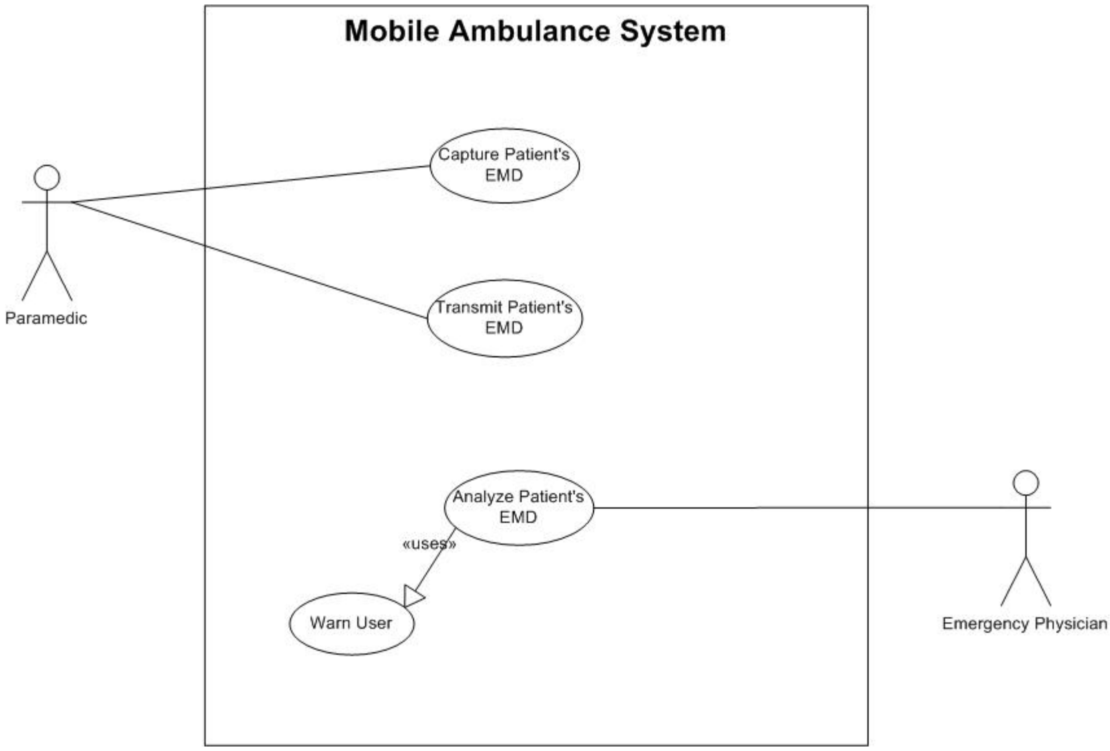


Figure 3.1 Use case diagram

Capture Patient's EMD

Main Success Scenario(MSS):

1. Paramedic fills in patient's demographic information (name, surname, birthdate, gender and contact phone number).
2. Paramedic fills in patient's medical information (blood pressure, pulse rate, consciousness, respiration, circulation, bleeding, shock, cracked, burnt, KİBAS).

3. Ambulance component stores patient's EMD in a file.

Transmit Patient's EMD

Main Success Scenario:

1. Paramedic goes to transmit.
2. Ambulance component sends patient's EMD to the ED based station.
3. Ambulance component displays confirmation message to paramedic.
4. Synch component persists EMD into the database.

Extension:

2a. Ambulance component fails to send patient's EMD.

.1: Paramedic retransmit patient's EMD or may cancel.

Warn User

Main Success Scenario:

1. Synch component generates a warning alert.

Analyze Patient's EMD

Main Success Scenario:

1. Synch component <u>warns user</u>.
2. Emergency physician refreshes daily ambulance run list.
3. Emergency physician browses the list and selects incoming patient.

4. Service component presents incoming patient's demographic and medical information.

Extension:

2a. Emergency physician has not logged on yet.

.1: Service component prompts user name and password.

.2: Emergency physician fills in authorization information.

.3: If authorization is confirmed, emergency physician returns to MSS at step 3. If not, service component displays error message. Emergency physician may reenter his/her authorization information or may cancel.

3.2.4 Logical Database Requirements

EMD is the minimum required data set for an emergency incident. This data set was determined by the highly contribution of Prof. Dr. Bülent Celasun and Dr. Yıldırım Karslıoğlu, Department of Pathology, Gülhane Military Medical Academy (GMMA). They were the mentors during the thesis study, and they gave the original idea of this thesis. They prepared this data set and the question structure using their medical knowledge and sources in the hospital.

EMD is composed of two parts: *demographic* and *medical information*. Two of these aspects are addressed below.

Demographic information (name, surname, birth date, gender, phone number) is used to register the patient into HIS. Demographic information is also checked with hospital records to examine whether the patient has a previous record at that hospital or not. In that case, all medical history information is available instantly. Otherwise, gender and age (computed from birth date) information are used to limit types of risks can the patient have and to decide on emergency treatment methods.

Medical information is the physiological parameters (blood pressure, pulse rate, consciousness, respiration, circulation, bleeding, shock, cracked, burnt, KİBAS) that should be checked in an emergency incident. In the thesis, these parameters and their corresponding values are modeled as questions and choices respectively. Thus paramedic selects choice(s) (values) for each question (parameter) while entering the medical information, then this selection list is evaluated in the ED as patient's medical information. The question-choice (parameter-value) coupling is listed (in Turkish) at Appendix A.

Mobile Ambulance system stores patient's EMD into a relational database. Its related ER diagram is shown in Figure 3.2.

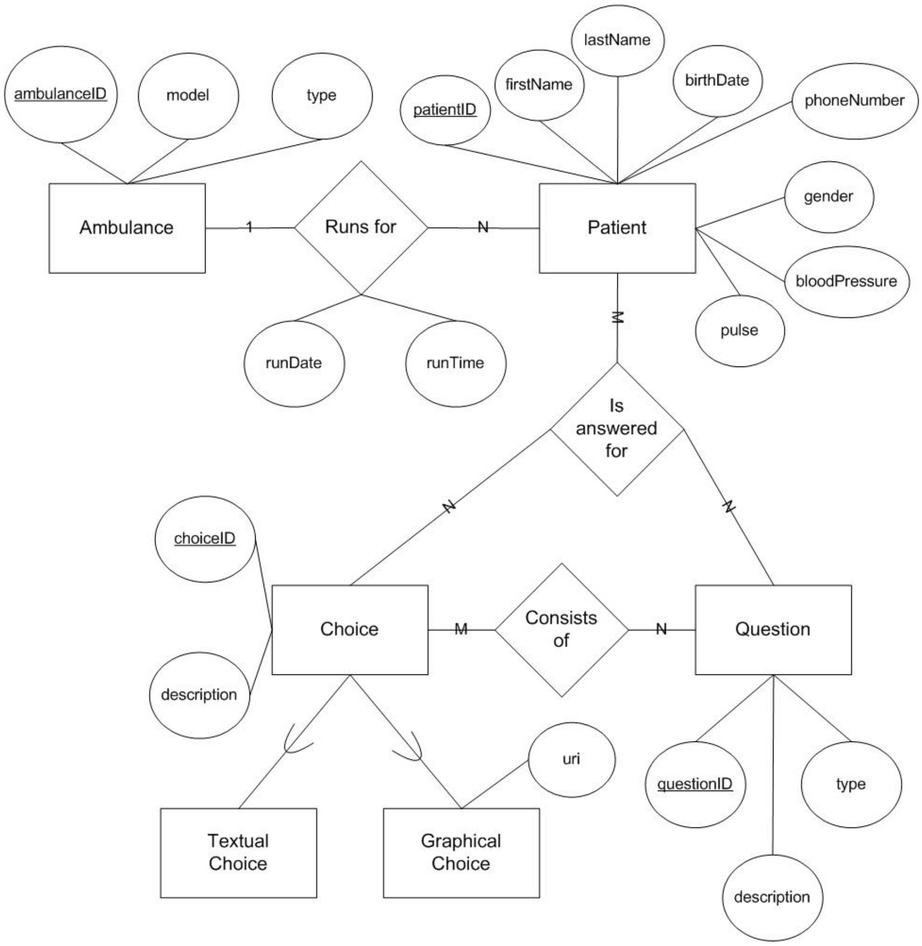


Figure 3.2 ER diagram

3.2.5 System Architecture

As a conceptual model, system architecture is shown in Figure 3.3.

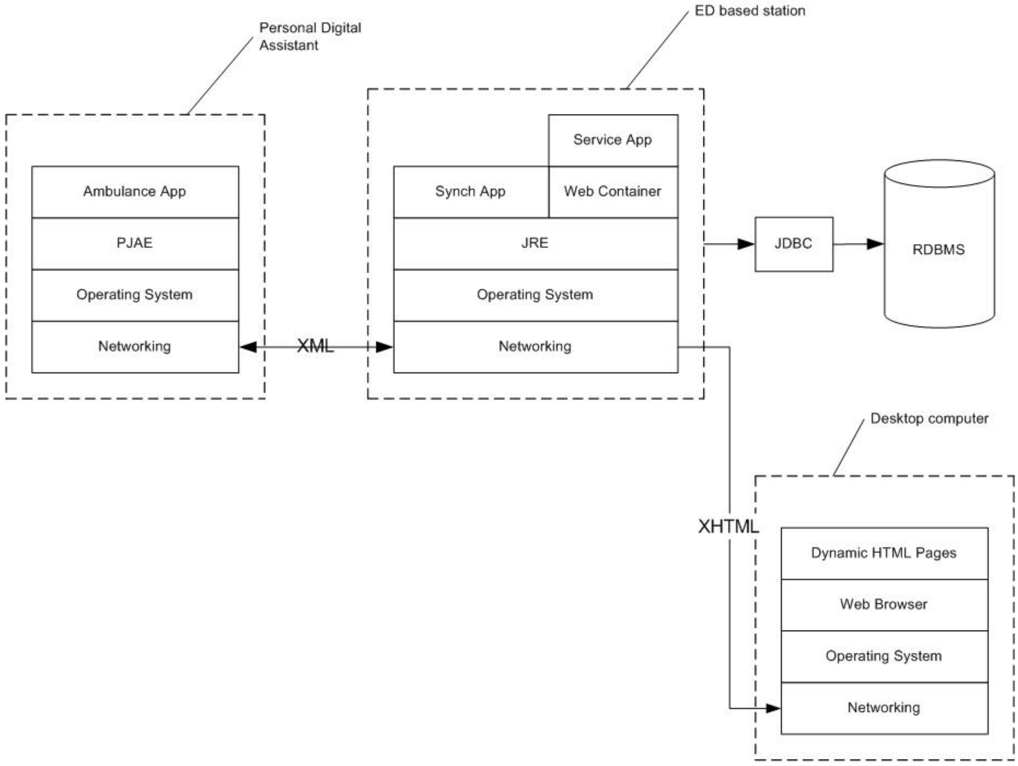


Figure 3.3 System Architecture

Conceptually the system is separated into two parts. First one is for paramedic to capture and transmit the patient's essential medical data. Second one is for emergency physician to analyze incoming data before the arrival of ambulance. However technically, Mobile Ambulance System is distributed into three applications: ambulance, synch and service for the accomplishment of these tasks.

Ambulance application is deployed on the PDA. Since it runs on PJAE, it does not require any specific operating system or mobile device. Thus it is platform independent. Its main functionalities are to have the paramedic enter patient's EMD and transmit these data to the ED based station. It provides an easy-to-use GUI not to keep paramedic suffering complex menus and high amount of text input. Paramedic sets each of vital signs by just a few pen strokes and completes the data entry with a minimum effort and time. Ambulance application stores the patient's EMD in an Extensible Markup Language (XML) [22] file. Upon request, it loads this file into memory, opens a socket connection to the ED based station and transmits data stream via this connection.

Synch application is deployed on the ED based station. Actually it is platform independent too. Since it is a Java application and runs on JRE, thus not to depend on the underlying operating system or computer architecture. Its main functionalities are persistence of patient's EMD into the database for back-end storage and warning the emergency physician. It waits for socket connections. Upon connection is established, it stores incoming data stream in an XML file, unmarshal this file (binds XML nodes to Java objects) and persists these objects into the database via JDBC. It then plays back a sound alert to warn the emergency physician.

Service application shares the same platform with synch application. However service application is a web application and implemented according to JSP 1.2 specification. Thus it runs on J2EE web container. Its main functionality is to have the emergency physician analyze the patient's EMD. It provides an easy-to-use GUI to have the emergency physician quickly browse the ambulance run list, easily select the incoming run and view the patient's EMD. It is a thin client application thus it does not require a separate installation on a desktop computer other than a XHTML [23] complaint web browser.

CHAPTER 4

DESIGN OF THE MOBILE AMBULANCE SYSTEM

This chapter provides the details of design. Software architecture is summarized at Section 4.1. It illustrates a top view of the system. Mobile Ambulance system is decomposed into three subsystems. Detailed design of each of these subsystems is given at Section 4.2, 4.3 and 4.4 respectively.

4.1 Software Architecture

Software architecture involves packages as design entities, and provides an object-oriented design of the system under these packages. The Mobile Ambulance system is decomposed into four packages:

- tr.mil.gata.emergency.ambulance(ambulance)
- tr.mil.gata.emergency.synch(synch)
- tr.mil.gata.emergency.service(service)
- tr.mil.gata.emergency.util(util)

Packages and dependencies are shown in Figure 4.1. For each package subordinates, design entity attributes are given in the following sections. Subordinates of the package are the classes encapsulated in that package.

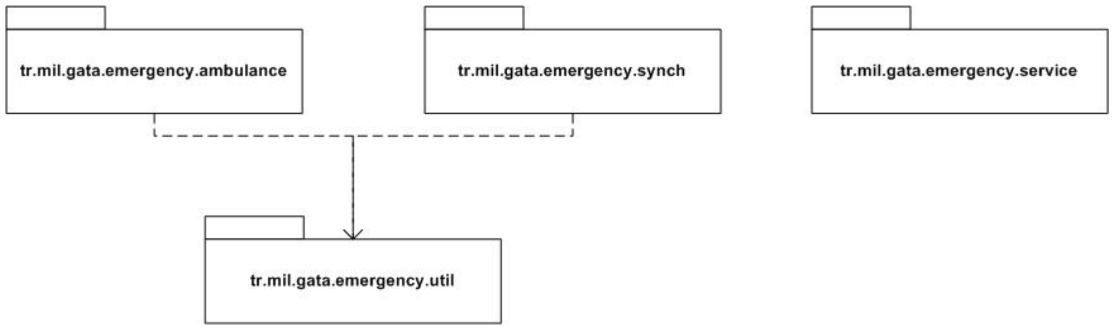


Figure 4.1 Software Architecture

Actually, the packages shown in Figure 4.1 are dependent to some other APIs, many of which are Java's standard libraries. These dependencies include Java2 SDK 1.4.1, PJAE 1.2, Castor, Xerces and JDBC 2.0 APIs.

4.2 Util Package

Util package contains classes that serve common functionalities for other packages (ambulance and synch). Package diagram, and associations and dependencies between class diagrams are shown in Appendix C.

4.3 Ambulance Package

Ambulance package is the one which is deployed on the mobile device. It is developed according to the PJAE 1.2 specification. PJAE 1.2 uses Java Development Kit (JDK) 1.1.8 as its base. However PJAE 1.2 adds security as specified in Java 2

SDK, Standard Edition version 1.2.2 (referred to JDK 1.2.2). Therefore, this specification contains some APIs based on JDK 1.2.2 APIs. Thus using PJAE 1.2 APIs was a major design constraint for development of the ambulance package.

As we mentioned in Section 3.2.5, ambulance and synch applications uses XML documents for transmission of patient's EMD. Actually, XML documents are serialized via the socket connection between ambulance (client) and synch (server) applications over TCP/IP. The power of XML is that it maintains the separation of the user interface from structured data, allowing the seamless integration of distributed applications. Moreover XML is text-based format that lets us describe, deliver and exchange structured data between a range of applications and clients for local display and manipulation. Today with XML, Document Type Definitions (DTDs) [24] may accompany a document, essentially defining the rules of the document, such as which elements are present and the structural relationship between the elements. DTDs help to validate the data when the receiving application does not have a built-in description of the incoming data. However, DTDs is not the only schema language. In fact, the W3C XML Schema Language [25] is the other way to express a schema. The XML Schema Language is much richer than DTDs. For example, schemas written in the XML Schema Language can describe structural relationships and data types that can't be expressed (or can't easily be expressed) in DTDs. XML Schema documents using in data exchange between ambulance and synch applications are listed at Appendix B.

The most typical way to access and use an XML document through Java Programming Language is through parsers that conform to the Simple API for XML (SAX) [26] or the Document Object Model (DOM) [27]. Developers can invoke a SAX or DOM parser in an application to parse an XML document that is, scan the document and logically break it up into discrete pieces. The parsed content is then made available to the application. In the SAX approach, the parser starts at the beginning of the document and passes each piece of the document to the application in the sequence it finds it. Nothing is saved in memory. The application can take action on the data as it gets it from the parser, but it can't do any in-memory manipulation of the data. For example, it can't update the data in memory and return the updated data to the XML file. On the other hand, in the DOM approach, the parser creates a tree of objects that represents the content and organization of data in the document. In this case, the tree exists in memory. The application can then navigate through the tree to access the data it needs, and if appropriate, manipulate it.

None of these methods are simpler and more maintainable than XML Data Binding [28] approach. Data Binding simplifies access to an XML document from a Java program by presenting the XML document to the program in a Java format. Data Binding allows developers to access and process XML data without having to know XML or XML processing. For example, there's no need to create or use a SAX parser or write callback methods. Thus, in our study, we chose XML Data Binding approach to access and process XML data. For instance we bind the schemas at Appendix B for the XML documents into a set of Java classes encapsulated in inbound and outbound packages.

Ambulance, inbound and outbound packages are shown in Appendix C. Associations and dependencies between class diagrams are also shown in Appendix C. Classes encapsulated in inbound and outbound packages are not involved in diagrams since they are dynamically generated by Castor source-generator according to the schemas at Appendix B.

4.4 Synch Package

Synch application, designed as a multi-threaded server, handles each ambulance socket connection (client request) in a separate thread. That means, in the physical world, the Mobile Ambulance system can handle more than one ambulance-to-ED communication concurrently. It uses the same approach, Data Binding, with the ambulance application to access and process XML data. Finally it depends on JDBC 2.0 API for persistence of patient's EMD into the database.

synch, inbound and outbound packages are shown in Appendix C. Associations and dependencies between class diagrams are also shown in Appendix C. Classes encapsulated in inbound and outbound packages are not involved in diagrams since they are dynamically generated by Castor source-generator according to the schemas at Appendix B.

4.5 Service Package

Service application is designed as a web application. For instance, service package conforms to JSP 1.2 specification [29]. JSPs simplify the delivery of dynamic web content. They enable web developers to create dynamic content by reusing and by interacting with components using server side scripting.

JSP is text-based format. In many ways, it looks like XHTML or XML documents. In fact JSPs normally include XHTML or XML markup. JSPs execute as part of a Web server. The server is often is referred to as the JSP container. When the JSP-enabled server receives the first request for a JSP, the JSP container translates that JSP into a Java Servlet that handles the current request and future requests to the JSP.

Presentation tier of service package consists of two JSPs: Patient and PatientList respectively. They use other classes in business tier of the package such as data access classes for accessing and querying database and data classes for encapsulation of data.

Service package containing business classes and default package containing JSPs are shown in Appendix C. Associations and dependencies between class diagrams are also shown in Appendix C.

CHAPTER 5

IMPLEMENTATION AND TESTING OF MOBILE AMBULANCE SYSTEM

This chapter provides the details of implementation which is based on the design in Chapter 4. Tools and facilities used during implementation are briefly introduced at Section 5.1. Ambulance Run Scenario mentioned at Section 3.2.2 is implemented at Section 5.2. Finally actual usage and system testing is introduced at Section 5.3. The Mobile Ambulance system's physical layout, revealing which pieces of software run on what pieces of hardware is shown at Figure 5.1.

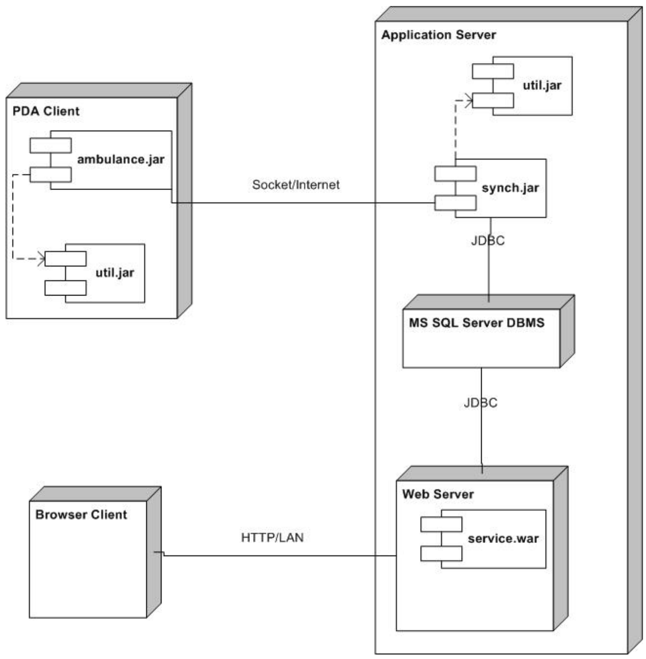


Figure 5.1 Deployment diagram

5.1 Tools and Facilities

The Mobile Ambulance system is written in Java Programming Language. Borland JBuilder is used as an integrated development environment. Ambulance application is compiled with JDK 1.2.2 however the code is written according to PJAE 1.2 API Specification for PersonalJava compatibility. For the rest of the system Java2 SDK 1.4.1 is used.

For XML Data Binding, Castor API is used [30]. Actually, Castor depends on Xerces, a free XML parser, for parsing XML documents. Inherently XML documents can be created and edited by any text editor however we used XMLSpy tool, which is an award-wining tool for easy creation and management of XML documents, XML schemas.

For persistence JDBC 2.0 API is used. Database implementation is made on MS SQL Server 2000 DBMS.

In addition to software development tools, Microsoft Visio tool is used for drawing the UML diagrams extensively.

5.2 Scenario Implementation

Ambulance run scenario mentioned at Section 3.2.2 is implemented with the Mobile Ambulance system. Since the system has multi-tiered distributed architecture, it is decomposed into three applications. Ambulance application runs on Insignia's Jeode Runtime that is an embedded virtual machine for Pocket PC devices [31]. Jeode Runtime is a fully certified implementation of PJAE 1.2 Specification. We installed Jeode Runtime on a Compaq IPAQ 3970 device for implementation and system testing. Synch application runs on Sun Microsystems' JRE 1.4.1 which is a Java virtual machine for desktop machines. Service Application is deployed on Tomcat 4.1 server that is a certified JSP container. Finally database is created on MS SQL Server 2000 RDBMS. JRE, Tomcat and SQL Server are all installed on a Windows (x86) machine, which is also not required so.

Ambulance run scenario is simulated at Appendix D. Description of screenshots follows in sequential order.

5.3 Actual Usage and Testing

For actual usage and testing of the Mobile Ambulance system, we need the contribution of two parties: an emergency aid service owning emergency ambulances and the emergency department of a hospital. After we came to an agreement for the usage of the system, client components (a PDA front-end application and a GSM phone for wireless data transmission) would be given to one of the ambulances of emergency aid service, and server components would be deployed to a server machine, having internet connection behind a firewall, in the hospital.

Exactly one-month period, March 2004, was passed to find out a proper environment for the Mobile Ambulance system's field tests. Negotiations were done with four different organizations from both public and private sector. At the end, most feasible party was chosen and field tests were started on April 05, 2004 and continued until April 17, 2004.

First, we thought to implement field tests of the Mobile Ambulance system with the contribution of GMMA emergency ambulance team and ED. However during interviews with the ED staff, we realized that GMMA emergency ambulances were being used for mostly transportation of stable patients from homes to hospital or from hospital to homes or even from one building to another in the hospital campus. With the help of IT staff we found a desktop computer having internet connection for deployment of the server components. However this computer was located in a separate room, not at the ED. Therefore someone should inform the ED staff about an incoming patient when s/he is coincidentally in that room and notices system generated emergency alert. Therefore we thought there was a high possibility that the Mobile Ambulance system would not be tested properly for emergency incidents in a short time period at GMMA.

Second, we contacted with 112 Emergency Aid Service Ankara Branch. We presented our study and made a demo to administrator staff including Vise Manager, Dr. Altuğ Aysun on March 09, 2004. They really appreciated our study however stressed some difficulties for actual usage of the system by 112 Emergency Aid Service. The client of Mobile Ambulance system would be used by only one ambulance and this corresponds to a station in the context of emergency aid service. The station serves for a certain region such as Etlik, Aşağı Ayrancı or Dikmen. The ambulance runs for emergency calls from that region, but transports the patient to any hospital, which is found suitable by the ambulance control center. For the actual usage of the Mobile Ambulance system, this business model is unacceptable. Since it requires the deployment of server components to every potential hospital in the city of Ankara, which is almost impossible.

Lastly, by the reference of Dr. Altuğ Aysun we contacted with Medline, a company providing emergency aid service nationwide. We presented our study to Dr. Alper Özkoçak, Ankara Branch Manager of Medline on March 19, 2004. He appreciated our study and promised for full contribution to the project. Medline has three stations in Ankara. Two of them are at Bayındır Hospital Branches (Kavaklıdere and Söğütözü). Dr. Alper Özkoçak offered us to use the Mobile Ambulance system's client components at Söğütözü station and to deploy server components to Bayındır Hospital. Actually this model is acceptable for us. Since there would be a single station serving for the patients distributed all over Ankara and transporting them to a single hospital, which is Söğütözü Bayındır in our case. By the help of Dr. Alper Özkoçak, we contacted with Bayındır Hospital's administrative staff and ED Head Dr. Nazmiye Koyuncu. They all supported the project and promised contribution during field tests. This completed our effort to find an environment for the actual usage and testing of the Mobile Ambulance system.

The field tests were launched for two weeks period from April 05, 2004 to April 17, 2004. During the tests, the following parameters were evaluated:

Technical Effectiveness: Two aspects of technical effectiveness of the Mobile Ambulance system were examined.

1. System availability.
2. Data transmission times for which the system availability was ensured.

Operational Effectiveness: Two indices of operational effectiveness that were measured are

1. Time spent by the paramedic in the ED, calculated from the time of arrival in the ED to the time patient was accepted by the ED staff, including the time to register the patient in the ED.
2. Time for data entry by the paramedics.

Clinical Effectiveness: One aspect of clinical effectiveness that was measured is

1. Waiting time to be attended by the doctors, calculated from patients' time of arrival in the ED to the time they were attended to by the ED doctor.

To measure the parameters discussed above, we prepared two information forms one for the ambulance crew, the other for the ED staff. During the tests, these forms had to be filled for each emergency incident for the sake of data collection. These forms are shown at Appendix E.

Results

During the test period, six emergency incident cases were reported, three of them by the Medline ambulance team and four of them by the Bayındır Hospital ED staff (one case was common in both the parties). Other than the reported ones, two of the unreported incidents' data were collected during the ambulance runs.

Technical effectiveness

1. System availability: 25% (2 out of 8) emergency ambulance run used the Mobile Ambulance system. One reason was that Mobile Ambulance system's server components were often not running while the ambulance team trying to transmit the data. Since every time the server restarted after an auto-update all running programs were shut down so did the server components. Another reason was the preliminary resistance of ambulance team against using the Mobile Ambulance system. Since they thought that while they were doing their job, using a handheld device was a burden for them.
2. Data transmission times: data transmission time was ~7 s for the emergency ambulance runs at which the Mobile Ambulance system was used.

Operational effectiveness

1. Time spent by the paramedic in the ED: actually, the net savings in ED time by the ambulance crew could not be measured. Since during the test period, ambulance crew reported only three emergency ambulance runs and in all

these cases, they could not achieve to convey the collected data to ED based station. Thus we cannot compare the two (one with the Mobile Ambulance system, the other without Mobile Ambulance system). Based on the reports provided by the ambulance team the ED staff accepts the patient in less than 1 min after s/he arrives in ED.

2. Data entry time by paramedics: during the test period, ambulance crew attempted to use the Mobile Ambulance system in two reported cases out of three. In one case, they reported the data entry time as 1.5 min, in the other as 5 min. In three cases, radio communication time was reported as 15 s., 30 s. and 90 s. In radio communication patient's demographic information (name, surname, age, gender), chief complaint, blood pressure, pulse rate and preliminary diagnosis were conveyed to the ambulance control center. However the time passed during conveying the information from the ambulance control center to hospital was not measured. Therefore overall voice communication time is more than the given values above.

Clinical effectiveness

1. Waiting time to be attended by the ED doctor: actually, the net savings in waiting time by the patients could not be measured. Because, during the test period, ED staff reported only four of the emergency ambulance runs and in all these cases the Mobile Ambulance system was not used. Thus we cannot compare the two (one with the Mobile Ambulance system, the other without Mobile Ambulance system). Based on the reports provided by the ED staff, the patient is seen by the ED doctor in less than 1 min after s/he arrives in ED.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Mobile Ambulance is neither a video images transmission system nor an ECG transmission system. It can also not be used for tracking and routing ambulances on a digital map as in the case of vehicle tracking systems. Mobile Ambulance is essentially a vital sign transmission system. Therefore we can only compare it with the systems in this category.

In the HEAL system, mentioned at Section 2.1.1, there were four functional modules: patient details module, ambulance incident management module, drug request and authorization module and text communications module. Mobile Ambulance meets the functionality of first two of these modules: patient details module by enabling to capture and convey patient demographic information and ambulance incident management module by enabling to archive information received from the ambulances. Mobile Ambulance system does not provide the functionality of drug request and authorization module of the HEAL system. However in our country there is always one doctor available in the ambulance and the doctor has full authority in an emergency incident thus there is no need for approval. Mobile Ambulance system does also not provide the functionality of text communication module. Since we thought it was not possible to use a chatting module by ambulance crew in all that rush and hurry. Additionally it may be uncomfortable for the users of Mobile Ambulance system to create and modify electronic messages by using PDA's soft keyboard or character recognition software. Thus the process may consume users' significant amount of time and hence leave less time for other duties, which may be more critical.

In the HEAL system, data transmission is established via the wireless MOBITEX network which is a packet-switched, mobile data network. Air transmission time for data is ~4 s unless ECG waveforms were also transmitted. Otherwise it results in transmission times exceeding 60 s. In the Mobile Ambulance system, GPRS, which is also publicly available, wireless and packet-switched network, is used. Size of the message transmitted from the ambulance to ED is ~2 KB. Air transmission time for data is ~7 s.

In the HEAL system and other related systems, common hardware and facilities were used. The clients were provided as PC front-end applications for the ambulance mobile computer/notebook. In the HEAL system, mobile computer was attached on board the ambulance. In the HECTOR system, mentioned at Section 2.1.2, emergency ambulance was even equipped with video cameras. In the SendECG system, mentioned at Section 2.1.3, data transmission was established over circuit-switched GSM network. Eventually, all these equipment and facilities increase the total cost of the ownership. However, in the Mobile Ambulance system, we aimed for low-cost property. Thus the client of the system comprises only an application running on a moderate PDA that costs ~600 USD. There is no need to attach the PDA on board the ambulance; it can easily be carried in paramedic's pocket. Moreover, data transmission is established over the packet-switched GPRS network in which you are billed only for the amount of data that are sent or received. During the test, we used TurkCELL's GPRSCELL service for the aim of data transmission and TurkCELL charges 1500 TL per 1 KB of data. In the Mobile Ambulance system, size of the message transmitted from the ambulance to ED is ~2 KB thus, the communication cost of the system per patient is only 3000 TL.

Mobile Ambulance system was tested from April 05, 2004 to April 17, 2004 by the contribution of Medline emergency ambulance and Bayındır Hospital. Further test data are needed for better accessing the effectiveness of the system. However, during the following period of time, the server we used in Bayındır Hospital Computer Center was out of service due to a hardware failure. As a result, this inhibited our data collection. We measured the system's effectiveness with respect to three criteria: technical, operational and clinical effectiveness. Eventually two weeks analysis of the pilot run revealed the following.

Totally six emergency incidents reported during the two weeks of evaluation. Ambulance team reported three of them, ED staff reported four (one incident was common in both parties). Unfortunately no data transmission was achieved by the Mobile Ambulance system in these cases. However two of the unreported incidents' data were conveyed to the hospital by ambulance team. Therefore system availability was measured as 25% (2 out of 8) and data transmission time was found ~7 s based on these two cases.

Data entry time was found to be greater than the one in radio communication. However when radio communication is used, data is transmitted from the ambulance to the control center then from the control center to hospital via a dedicated telephone. Thus the total transmission time with radio including ambulance control center to hospital and airtime busy is greater than the reported ones. So we cannot do a proper comparison. Additionally, all the data transmitted by using the Mobile Ambulance system is archived which is not so using radio communication.

In two cases it is reported that ambulance team could not achieve to convey patient's information to the hospital although they attempted. After a short examination, we found the reason that Mobile Ambulance system's server components were not running while the team was trying to transmit the data. Since every time the server restarts after an auto-update all running programs are shutdown so do the server components, and someone from hospital's IT staff should manually rerun them which in our case is hard to apply. This problem was specific to the server we used in Bayındır Hospital Computer Center. To overcome this problem, we moved the server components' .bat files in Startup directory. By this way, every time server's operating system (Windows 2003 Server in our case) starts, it looks for the application programs to automatically start up from this directory.

In all the cases ambulance team reported that the patient was accepted by the ED staff in less than 1 minute after s/he arrived in ED. Therefore we can conclude that the Mobile Ambulance system does not offer a considerable timesavings for ambulance team in ED, since there are registration counter clerks available at 24/7 in the ED, and their main responsibilities are to welcome and register the patient immediately.

In all the cases the ED doctor saw the patient in less than 1 minute after s/he arrived in ED. Therefore we can conclude that the Mobile Ambulance system does not offer a considerable timesavings by patients in ED. Since there is always one doctor and one or more nurses on duty in the ED, and they attend to the patient immediately while ambulance team or patient's relatives help the clerks to register the patient.

6.2 Future Work

Our future work will focus on collecting more data, evaluating the system in state hospitals which have more patient throughput, security issues and the way to integrate the Mobile Ambulance system with legacy Hospital Information Systems. Further investigation of the cost-effectiveness of Mobile Ambulance is also necessary.

Further test data are needed for better accessing the effectiveness of the Mobile Ambulance system. Moreover, it will be beneficial to test the system in state hospitals which have more patient throughput than the private healthcare institutions.

The state hospitals have limited resources allocated per patient when compared with the private hospitals. Therefore, we suppose that the Mobile Ambulance system will be more useful to improve the quality of service in state hospital ED.

In the Mobile Ambulance system, further interoperability between distributed applications is achieved by realizing all data exchanges through XML, which is a universal standard for electronic data exchange. However these messages are exchanged in plain text, without an encryption protocol. Therefore a standard encryption protocol should be used to enable the Mobile Ambulance system to encode and decode the XML messages for secure transmissions.

Traditionally, patients being brought by ambulances were registered only after arrival at the ED. The process of registration takes several minutes and may be carried out by ambulance crew, hospital staff, patients themselves or their relatives. With the Mobile Ambulance system, it was aimed to complete registration before the patient arrives at the ED. Since the information required for patient registration at ED may have already been captured by ambulance crew, it only remains for this information to be transferred to patient registration component of HIS to result in automated pre-arrival patient registration. It remains to be seen whether such an enhanced feature can be integrated into the Mobile Ambulance system and result in further timesavings for ED staff.

REFERENCES

[1] Chamberlain, D. (1997). Fax machines for thrombolysis?. Heart, 78, 108.

[2] Editorial. (1990). From pain to reperfusion: what role for the prehospital 12-lead ECG?. Annals of Emergency Medicine, 19, 1343-1346.

[3] Michaelson, M., Levi, L. (1997). Videotaping in the admitting area: a most useful tool for quality improvement of the trauma care. European Journal of Emergency Medicine, 4, 94-96.

[4] Audette, L. G., Keggi, K. J. (1970). Portable television for emergency vehicles. Medical and Biological Illustration, 20, 1-5.

[5] Anantharaman, V., Han, L. S. (2001). Hospital and emergency ambulance link: using IT to enhance emergency pre-hospital care. International Journal of Medical Informatics, 61, 147–161.

[6] Curry, G. R., Harrop, N. (1998). The Lanchashire telemedicine ambulance. Journal of Telemedicine and Telecare, 4, 231-238.

[7] Giovas, P., Papadoyannis, D., Thomakos, D., Pazachos, G., Rallidis, M. , Soulis, D., Stamatopoulos, C., Mavrogeni, S., Katsilambros, N. (1998) Transmission of electrocardiograms from a moving ambulance. Journal of Telemedicine and Telecare, 4 (Suppl 1), 5-7.

[10] Boger, M. (2001). Java in Distributed Systems. John Wiley & Sons.

[17] Giguere, E. (2000). Java 2 Micro Edition. John Wiley & Sons.

[21] Fowler, M. (2004). UML Distilled. Addison Wesley Third Edition.

[28] Sosnoski, D. M. (2002). XML in Java: Data Binding with Castor. IBM developerWorks.

[8] Kaan project. (2004). http://www.essed.com.tr/kaan_401.htm, April 22

[9] Java Programming Language. (2004). http://java.sun.com, April 22

[11] RMI, Remote Method Invocation. (2004). http://java.sun.com/products/jdk/rmi/, April 22

[12] CORBA, Common Object Request Broker Architecture. (2004). http://www.corba.org/, April 22

[13] JDBC, Java Database Connectivity. (2004). http://java.sun.com/products/jdbc/, April 22

[14] J2EE, Java 2 Enterprise Edition. (2004). http://java.sun.com/j2ee/, April 22

[15] HTTP, Hyper Text Transfer Protocol. (2004). http://www.w3.org/Protocols/, April 22

[16] MMI, Multimodal Interaction. (2004). http://www.w3.org/2002/mmi/, April 22

[18] GPRS, General Packet Radio Service. (2004). http://www.gsmworld.com/technology/gprs/intro.shtml, April 22

[19] PersonalJava. (2004). http://java.sun.com/products/personaljava/, April 22

[20] J2ME Personal Profile. (2004). http://java.sun.com/products/personalprofile/, April 22

[22] XML, Extensible Markup Language. (2004). http://www.w3.org/XML/, April 22

[23] XHTML, Extensible Hypertext Transfer Protocol. (2004). http://www.w3.org/TR/xhtml1/, April 22

[24] DTD, Document Type Definition. (2004). http://www.w3.org/TR/2004/REC-xml-20040204/, April 22

[25] XML Schema. (2004). http://www.w3.org/XML/Schema, April 22

[26] SAX, Simple API for XML. (2004). http://www.saxproject.org, April 22

[27] DOM, Document Object Model. (2004). http://www.w3.org/DOM/, April 22

[29] JSP, Java Server Pages. (2004). http://java.sun.com/products/jsp/, April 22

[30] Castor. (2004). http://castor.exolab.org/xml-framework.html, April 22

[31] Jeode Java Virtual Machine. (2004). http://www.cs.unc.edu/~lindsey/7ds/notes/jeode/, April 22

APPENDICES

APPENDIX A

Structured Question Set for an Emergency Case

1- Bilinç durumu:

a) Açık
b) Yarı açık
c) Kapalı

2- Solunum:

a) Var
    i) Düzenli
    ii) Düzensiz
b) Yok

3- Dolaşım:

a) Var
    i) Periferik damarlarda nabız alınabiliyor.
    ii) Nabız yalnızca büyük damarlarda hissediliyor.
b) Yok

4- Kanama:

a) Var
    Şekil üzerinde gösterilecek.
    - Baş-boyun
    - Sağ üst ekstremite
    - Sol üst ekstremite
    - Sağ alt ekstremite
    - Sol alt ekstremite

- Göğüs
- Karın

b) Yok
  i) Arteriyel kan basıncı ve nabız normal sınırlarda
  ii) Nabız yüksek

5- Preşok veya şok:

a) Arteriyel kan basıncı düşük mü?
b) Nabız hızlı ve yüzeysel mi?
c) Deri, soğuk, soluk ve terli mi?
d) Genel durum bozuluyor mu?

6- Açık veya kapalı kırık

Şekil üzerinde gösterilecek.

- Kafa ve yüz kemikleri
- Üst ekstremiteler
- Alt ekstremiteler ve pelvis
- Omurga ve/veya göğüs kafesi

7- Yanık

Şekil üzerinde yanık olan bölgeler işaretlenecek. Toplam 11 bölge var, her biri %9 değerinde:

- Vücut yüzeyinin % 10'undan daha azı yanık durumda.
- Vücut yüzeyinin % 10-19'u yanık durumda.
- Vücut yüzeyinin % 20-29'u yanık durumda.
- Vücut yüzeyinin % 30-39'u yanık durumda.
- Vücut yüzeyinin % 40-49'u yanık durumda.
- Vücut yüzeyinin % 50-59'u yanık durumda.
- Vücut yüzeyinin % 60-69'u yanık durumda.
- Vücut yüzeyinin % 70-79'u yanık durumda.
- Vücut yüzeyinin % 80-89'u yanık durumda.
- Vücut yüzeyinin % 90 veya daha fazlası yanık durumda.

8- KİBAS:

a) Ani ve fışkırır biçimde kusma oldu mu?
b) Hastanın bilinç düzeyi hızla kötüleşti mi?
c) Arteriyel kan basıncı yüksek mi?
d) Nabzı normalin alt sınırında veya daha aşağıda mı?
e) Konvülsiyon (nöbet) geçirdi mi?

APPENDIX B

XML Schemas for XML Documents Used in Data Exchange between Ambulance and ED Based Station


```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="AmbulanceSync">
		<xs:annotation>
			<xs:documentation>Ambulans Seferi Ana
Veriler</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="QuestionList"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="Question">
		<xs:annotation>
			<xs:documentation>Soru Bilgisi</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="Choice" minOccurs="0"
maxOccurs="unbounded"/>
			</xs:sequence>
			<xs:attribute name="ID" type="xs:byte" use="required"/>
			<xs:attribute name="Type" use="required">
				<xs:simpleType>
					<xs:restriction base="xs:string">
						<xs:enumeration value="text"/>
						<xs:enumeration value="image"/>
						<xs:enumeration value="multi"/>
					</xs:restriction>
				</xs:simpleType>
```

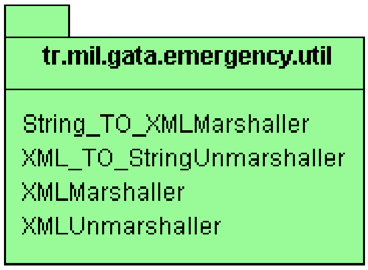
```
                        </xs:attribute>
                        <xs:attribute name="Description" type="xs:string"
use="required"/>
                </xs:complexType>
        </xs:element>
        <xs:element name="Choice">
                <xs:annotation>
                        <xs:documentation>SeÃ§enek Bilgisi</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                        <xs:attribute name="ID" type="xs:byte" use="required"/>
                        <xs:attribute name="Description" type="xs:string"
use="required"/>
                        <xs:attribute name="URI" type="xs:anyURI"
use="optional"/>
                </xs:complexType>
        </xs:element>
        <xs:element name="QuestionList">
                <xs:annotation>
                        <xs:documentation>Soru Listesi</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Question"
maxOccurs="unbounded"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="AmbulanceTrx">
                <xs:annotation>
                        <xs:documentation>Ambulans Sefer
Bilgileri</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="Ambulance" type="xs:int"/>
                                <xs:element ref="PatientList"/>
```

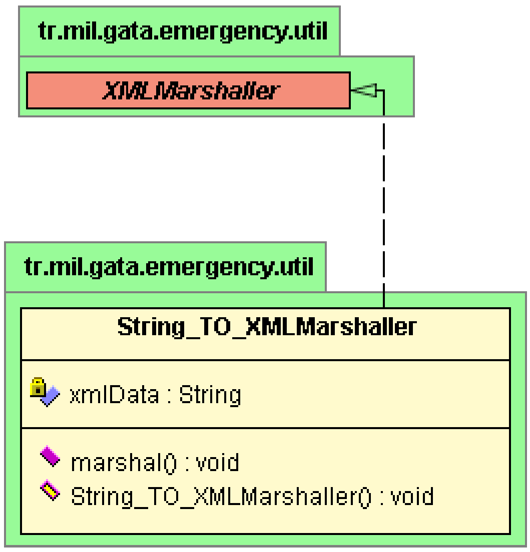
```
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="Patient">
		<xs:annotation>
			<xs:documentation>Hasta Bilgileri</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="FirstName" type="xs:string"/>
				<xs:element name="LastName" type="xs:string"/>
				<xs:element name="BirthPlace" type="xs:string"/>
				<xs:element name="BirthDate" type="xs:date"/>
				<xs:element name="PhoneNumber" type="xs:string"/>
				<xs:element name="Gender" type="xs:string"/>
				<xs:element name="BloodPressure_B" type="xs:int"/>
				<xs:element name="BloodPressure_S" type="xs:int"/>
				<xs:element name="Pulse" type="xs:int"/>
				<xs:element ref="AnswerList"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="AnswerList">
		<xs:annotation>
			<xs:documentation>Cevap Listesi</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="Answer" maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="PatientList">
		<xs:annotation>
			<xs:documentation>Hasta Listesi</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="Patient" maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
```

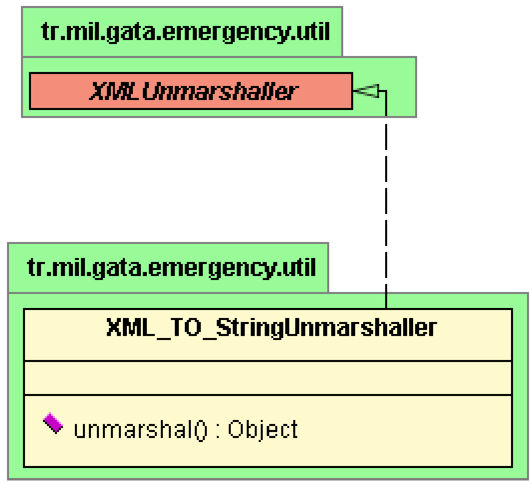
```
		</xs:element>
		<xs:element name="Answer">
			<xs:annotation>
				<xs:documentation>Cevap Bilgileri</xs:documentation>
			</xs:annotation>
			<xs:complexType>
				<xs:sequence>
					<xs:element ref="Question"/>
					<xs:element ref="Choice" minOccurs="0"
maxOccurs="unbounded"/>
				</xs:sequence>
			</xs:complexType>
		</xs:element>
		<xs:element name="Question">
			<xs:annotation>
				<xs:documentation>Soru</xs:documentation>
			</xs:annotation>
			<xs:complexType>
				<xs:attribute name="ID" type="xs:byte" use="required"/>
			</xs:complexType>
		</xs:element>
		<xs:element name="Choice">
			<xs:annotation>
				<xs:documentation>SeÃ§enek</xs:documentation>
			</xs:annotation>
			<xs:complexType>
				<xs:attribute name="ID" type="xs:byte" use="required"/>
			</xs:complexType>
		</xs:element>
</xs:schema>
```

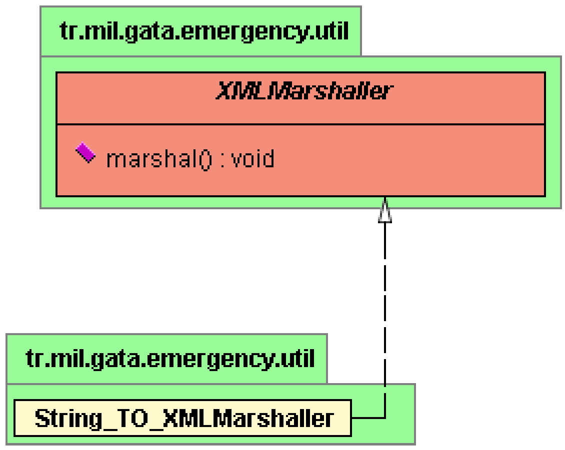
APPENDIX C

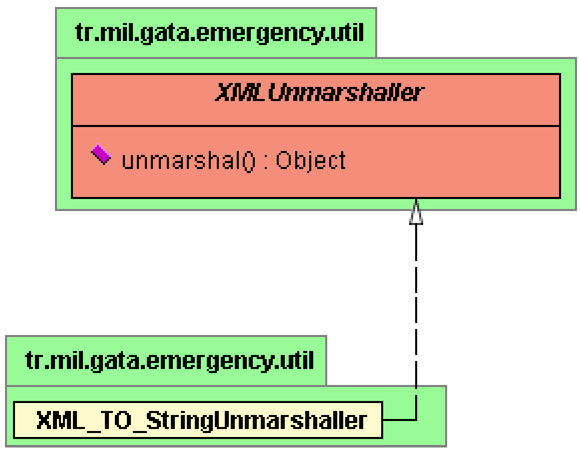
UML Package and Class Diagrams of the Software Components

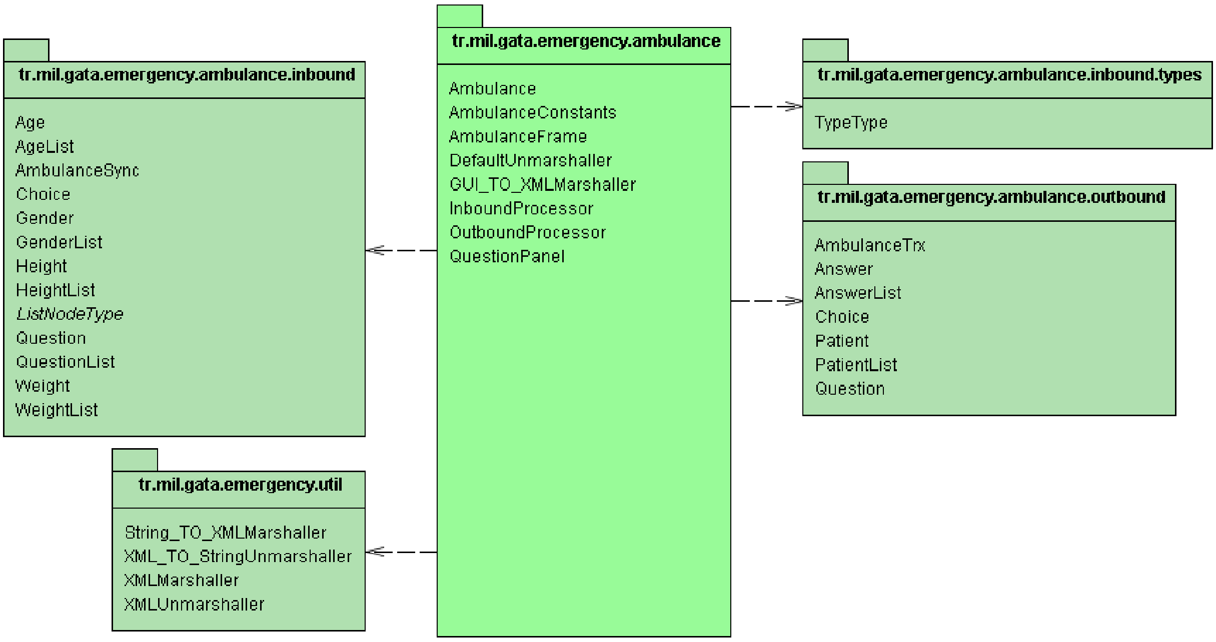




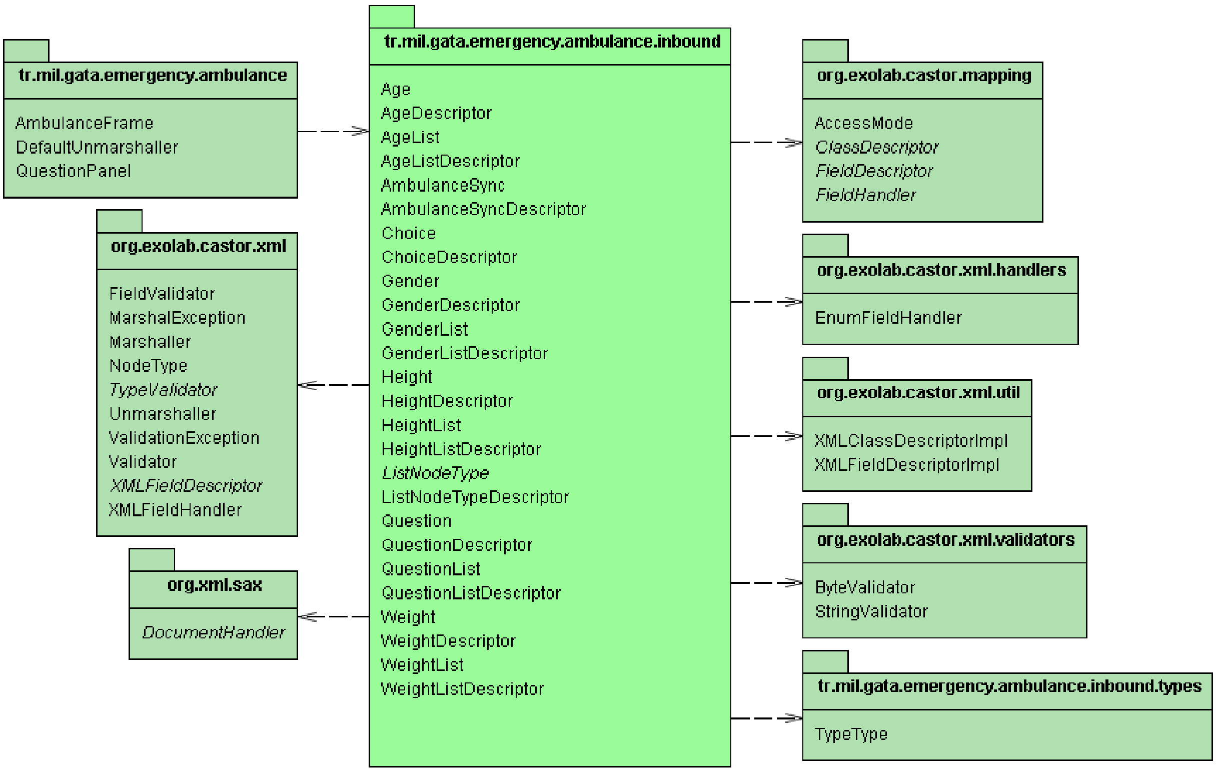

tr.mil.gata.emergency.util
XMLUnmarshaller
tr.mil.gata.emergency.util
XML_TO_StringUnmarshaller
unmarshal() : Object


tr.mil.gata.emergency.util
XMLMarshaller
marshal() : void
tr.mil.gata.emergency.util
String_TO_XMLMarshaller

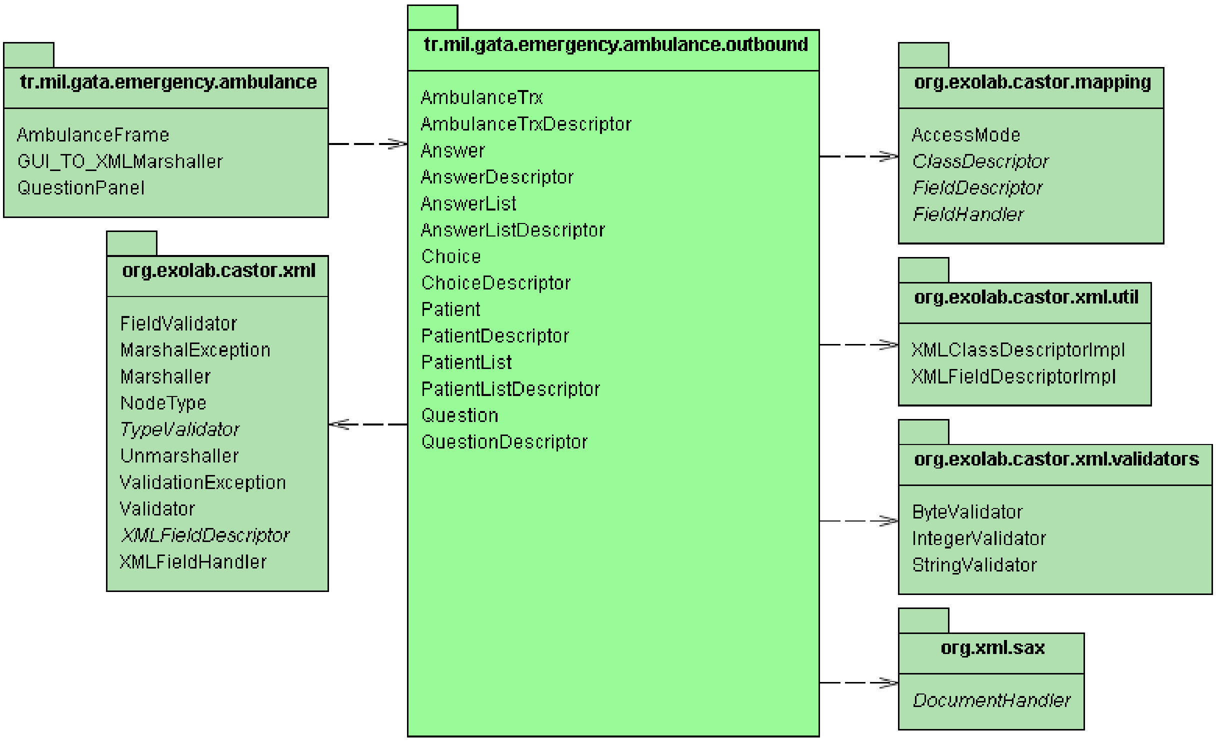

tr.mil.gata.emergency.util
XMLUnmarshaller
unmarshal() : Object
tr.mil.gata.emergency.util
XML_TO_StringUnmarshaller

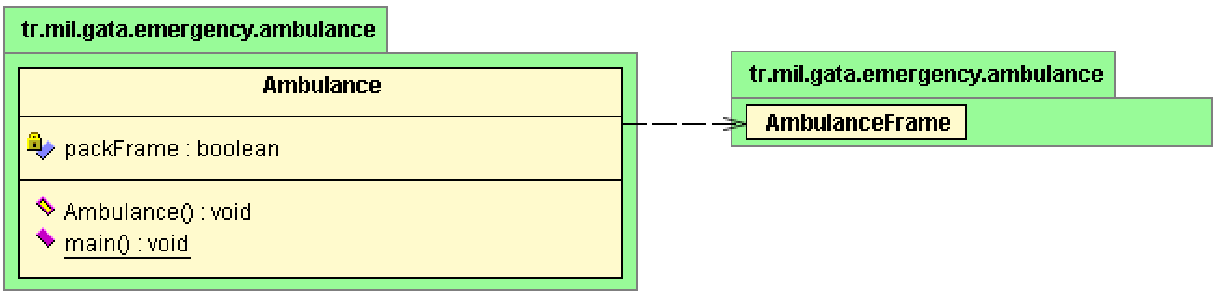


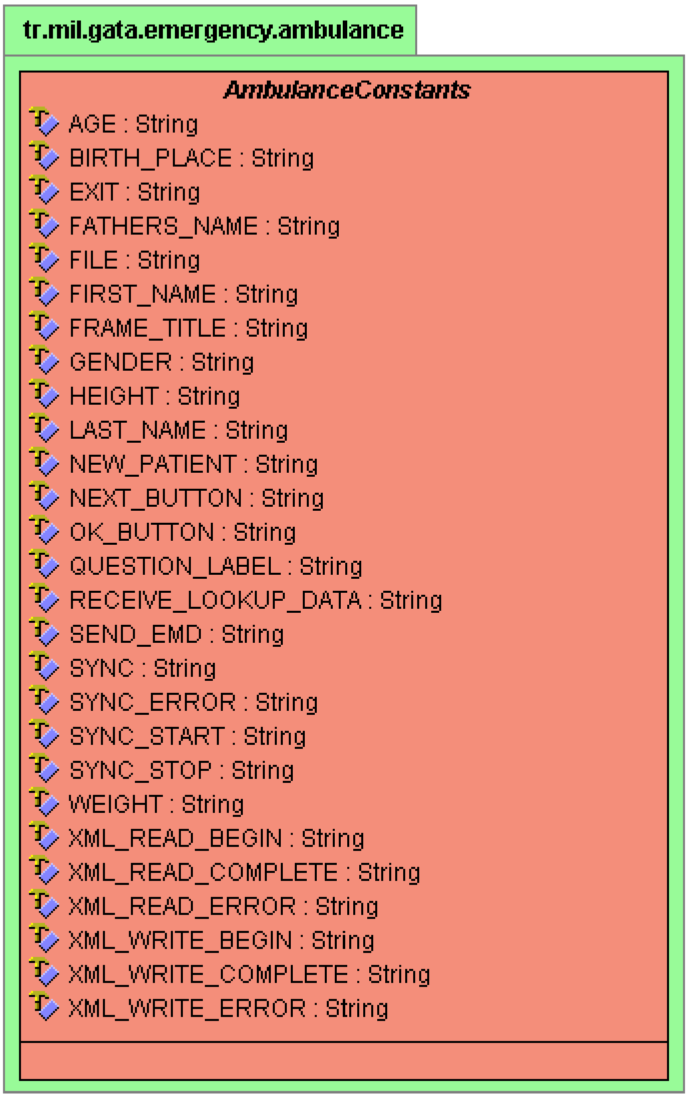
tr.mil.gata.emergency.ambulance.inbound
Age
AgeList
AmbulanceSync
Choice
Gender
GenderList
Height
HeightList
ListNodeType
Question
QuestionList
Weight
WeightList
tr.mil.gata.emergency.ambulance
Ambulance
AmbulanceConstants
AmbulanceFrame
DefaultUnmarshaller
GUI_TO_XMLMarshaller
InboundProcessor
OutboundProcessor
QuestionPanel
tr.mil.gata.emergency.ambulance.inbound.types
TypeType
tr.mil.gata.emergency.ambulance.outbound
AmbulanceTrx
Answer
AnswerList
Choice
Patient
PatientList
Question
tr.mil.gata.emergency.util
String_TO_XMLMarshaller
XML_TO_StringUnmarshaller
XMLMarshaller
XMLUnmarshaller

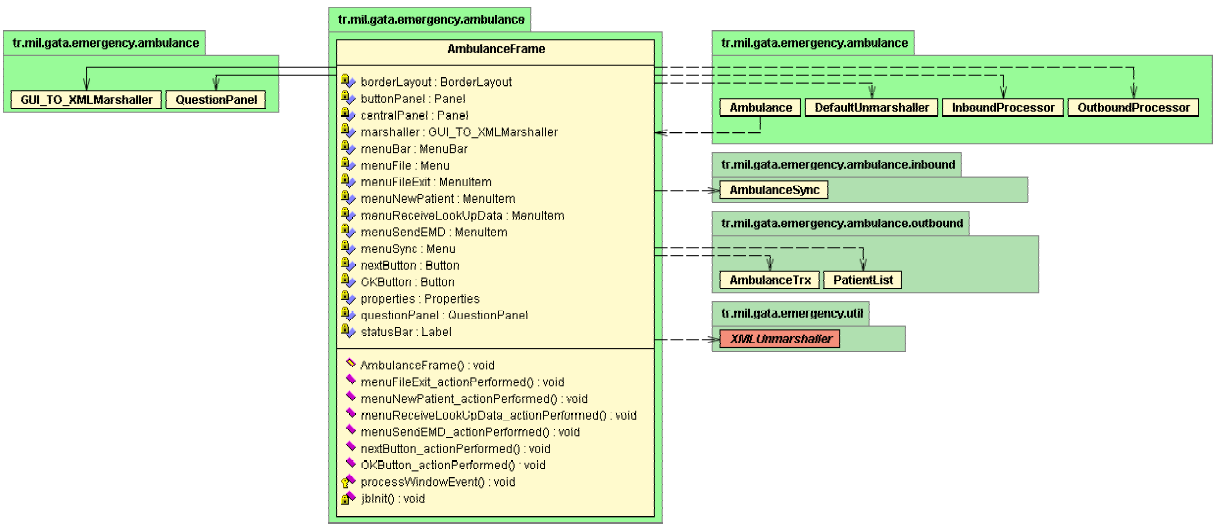


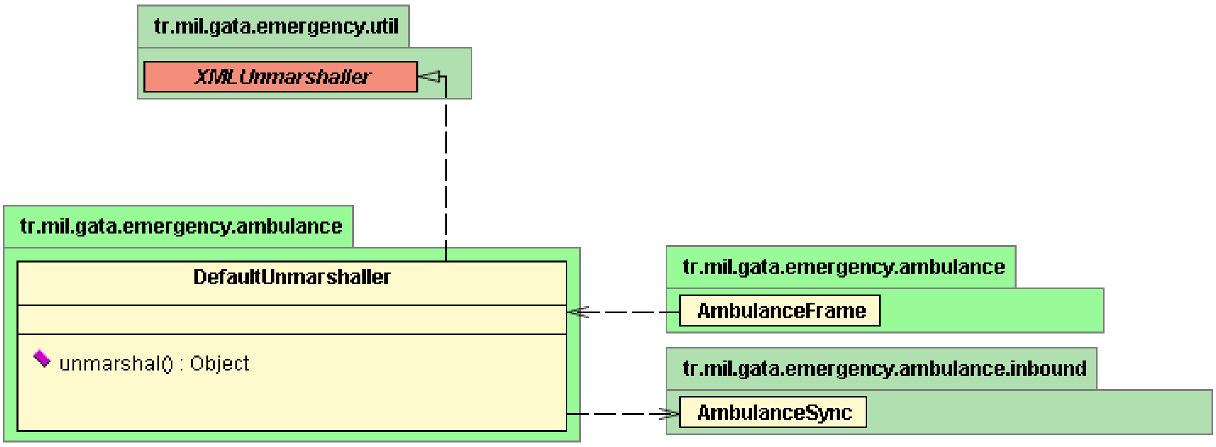
tr.mil.gata.emergency.ambulance
AmbulanceFrame
DefaultUnmarshaller
QuestionPanel
tr.mil.gata.emergency.ambulance.inbound
Age
AgeDescriptor
AgeList
AgeListDescriptor
AmbulanceSync
AmbulanceSyncDescriptor
Choice
ChoiceDescriptor
Gender
GenderDescriptor
GenderList
GenderListDescriptor
Height
HeightDescriptor
HeightList
HeightListDescriptor
ListNodeType
ListNodeTypeDescriptor
Question
QuestionDescriptor
QuestionList
QuestionListDescriptor
Weight
WeightDescriptor
WeightList
WeightListDescriptor
org.exolab.castor.mapping
AccessMode
ClassDescriptor
FieldDescriptor
FieldHandler
org.exolab.castor.xml
FieldValidator
MarshalException
Marshaller
NodeType
TypeValidator
Unmarshaller
ValidationException
Validator
XMLFieldDescriptor
XMLFieldHandler
org.exolab.castor.xml.handlers
EnumFieldHandler
org.exolab.castor.xml.util
XMLClassDescriptorImpl
XMLFieldDescriptorImpl
org.xml.sax
DocumentHandler
org.exolab.castor.xml.validators
ByteValidator
StringValidator
tr.mil.gata.emergency.ambulance.inbound.types
TypeType

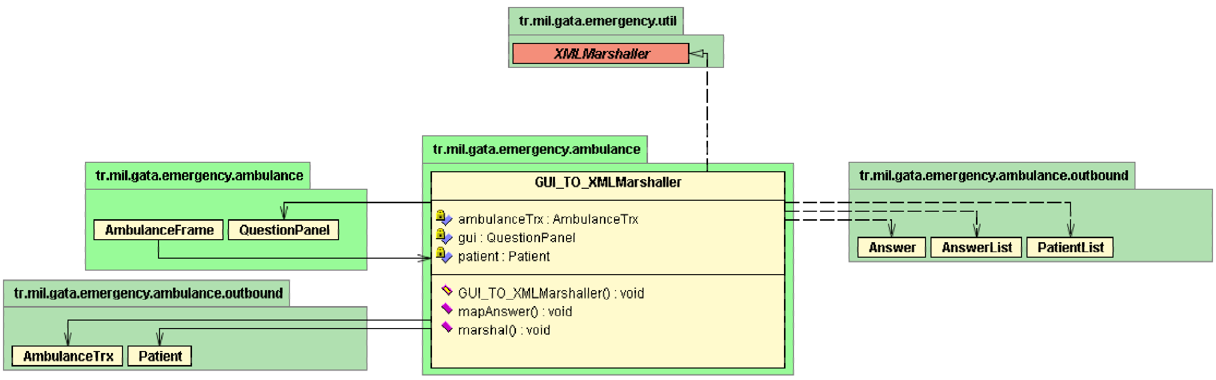

tr.mil.gata.emergency.ambulance
AmbulanceFrame
GUI_TO_XMLMarshaller
QuestionPanel
tr.mil.gata.emergency.ambulance.outbound
AmbulanceTrx
AmbulanceTrxDescriptor
Answer
AnswerDescriptor
AnswerList
AnswerListDescriptor
Choice
ChoiceDescriptor
Patient
PatientDescriptor
PatientList
PatientListDescriptor
Question
QuestionDescriptor
org.exolab.castor.mapping
AccessMode
ClassDescriptor
FieldDescriptor
FieldHandler
org.exolab.castor.xml
FieldValidator
MarshalException
Marshaller
NodeType
TypeValidator
Unmarshaller
ValidationException
Validator
XMLFieldDescriptor
XMLFieldHandler
org.exolab.castor.xml.util
XMLClassDescriptorImpl
XMLFieldDescriptorImpl
org.exolab.castor.xml.validators
ByteValidator
IntegerValidator
StringValidator
org.xml.sax
DocumentHandler

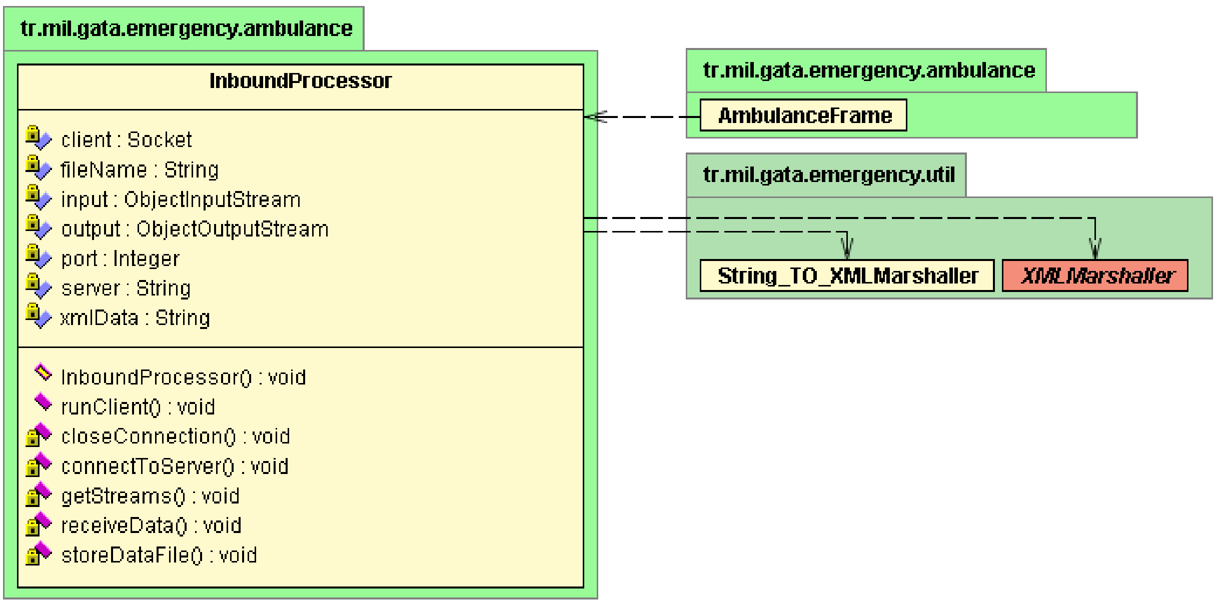

tr.mil.gata.emergency.ambulance
Ambulance
packFrame : boolean
Ambulance() : void
main() : void
tr.mil.gata.emergency.ambulance
AmbulanceFrame

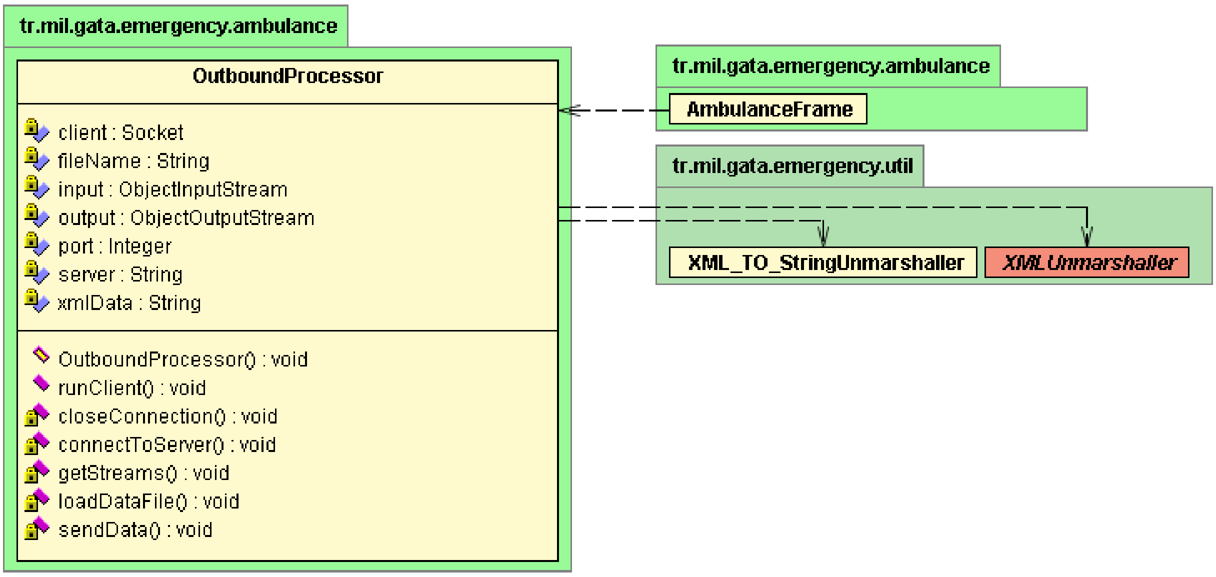

tr.mil.gata.emergency.ambulance
AmbulanceConstants
AGE : String
BIRTH_PLACE : String
EXIT : String
FATHERS_NAME : String
FILE : String
FIRST_NAME : String
FRAME_TITLE : String
GENDER : String
HEIGHT : String
LAST_NAME : String
NEW_PATIENT : String
NEXT_BUTTON : String
OK_BUTTON : String
QUESTION_LABEL : String
RECEIVE_LOOKUP_DATA : String
SEND_EMD : String
SYNC : String
SYNC_ERROR : String
SYNC_START : String
SYNC_STOP : String
WEIGHT : String
XML_READ_BEGIN : String
XML_READ_COMPLETE : String
XML_READ_ERROR : String
XML_WRITE_BEGIN : String
XML_WRITE_COMPLETE : String
XML_WRITE_ERROR : String

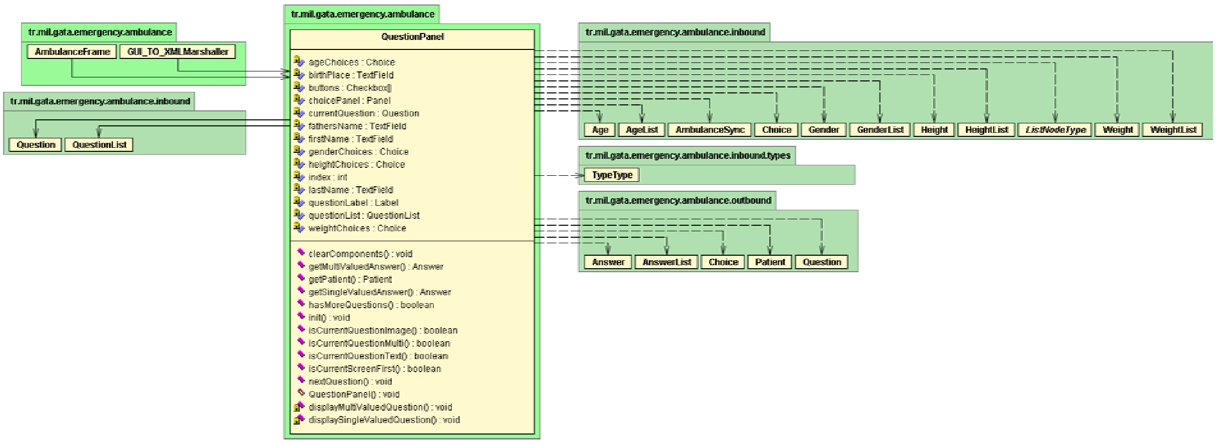

tr.mil.gata.emergency.ambulance
GUI_TO_XMLMarshaller
QuestionPanel
tr.mil.gata.emergency.ambulance
AmbulanceFrame
borderLayout : BorderLayout
buttonPanel : Panel
centralPanel : Panel
marshaller : GUI_TO_XMLMarshaller
menuBar : MenuBar
menuFile : Menu
menuFileExit : MenuItem
menuNewPatient : MenuItem
menuReceiveLookUpData : MenuItem
menuSendEMD : MenuItem
menuSync : Menu
nextButton : Button
OKButton : Button
properties : Properties
questionPanel : QuestionPanel
statusBar : Label
AmbulanceFrame() : void
menuFileExit_actionPerformed() : void
menuNewPatient_actionPerformed() : void
menuReceiveLookUpData_actionPerformed() : void
menuSendEMD_actionPerformed() : void
nextButton_actionPerformed() : void
OKButton_actionPerformed() : void
processWindowEvent() : void
jbInit() : void
tr.mil.gata.emergency.ambulance
Ambulance
DefaultUnmarshaller
InboundProcessor
OutboundProcessor
tr.mil.gata.emergency.ambulance.inbound
AmbulanceSync
tr.mil.gata.emergency.ambulance.outbound
AmbulanceTrx
PatientList
tr.mil.gata.emergency.util
XMLUnmarshaller

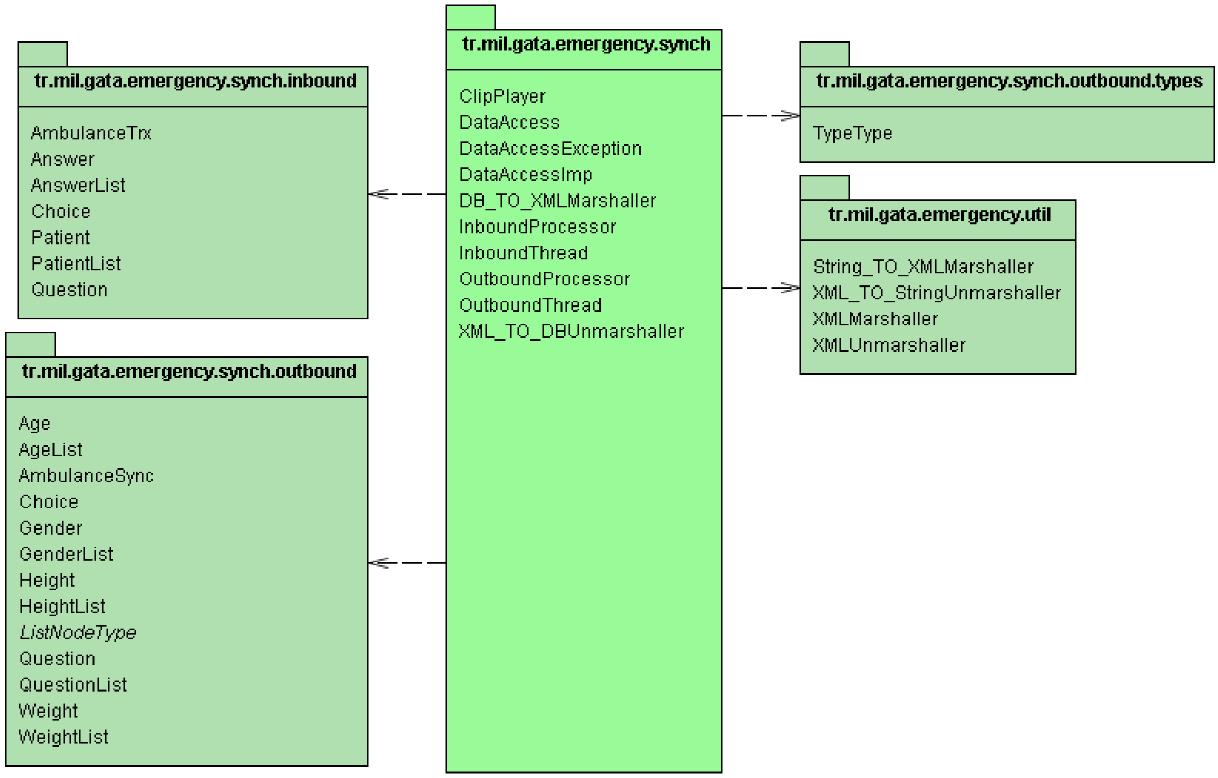

tr.mil.gata.emergency.util
XMLUnmarshaller
tr.mil.gata.emergency.ambulance
DefaultUnmarshaller
unmarshal() : Object
tr.mil.gata.emergency.ambulance
AmbulanceFrame
tr.mil.gata.emergency.ambulance.inbound
AmbulanceSync

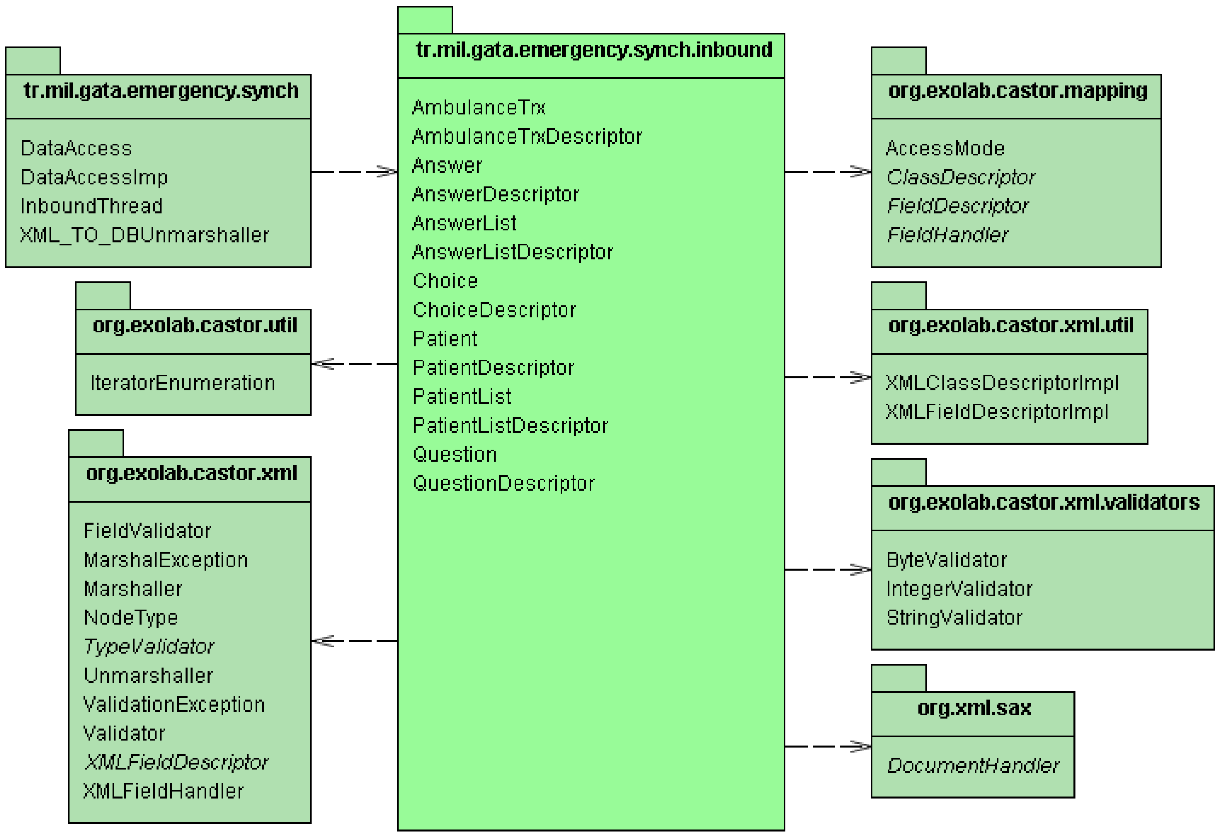

tr.mil.gata.emergency.util
XMLMarshaller
tr.mil.gata.emergency.ambulance
GUI_TO_XMLMarshaller
ambulanceTrx : AmbulanceTrx
gui : QuestionPanel
patient : Patient
GUI_TO_XMLMarshaller() : void
mapAnswer() : void
marshal() : void
tr.mil.gata.emergency.ambulance
AmbulanceFrame
QuestionPanel
tr.mil.gata.emergency.ambulance.outbound
AmbulanceTrx
Patient
tr.mil.gata.emergency.ambulance.outbound
Answer
AnswerList
PatientList

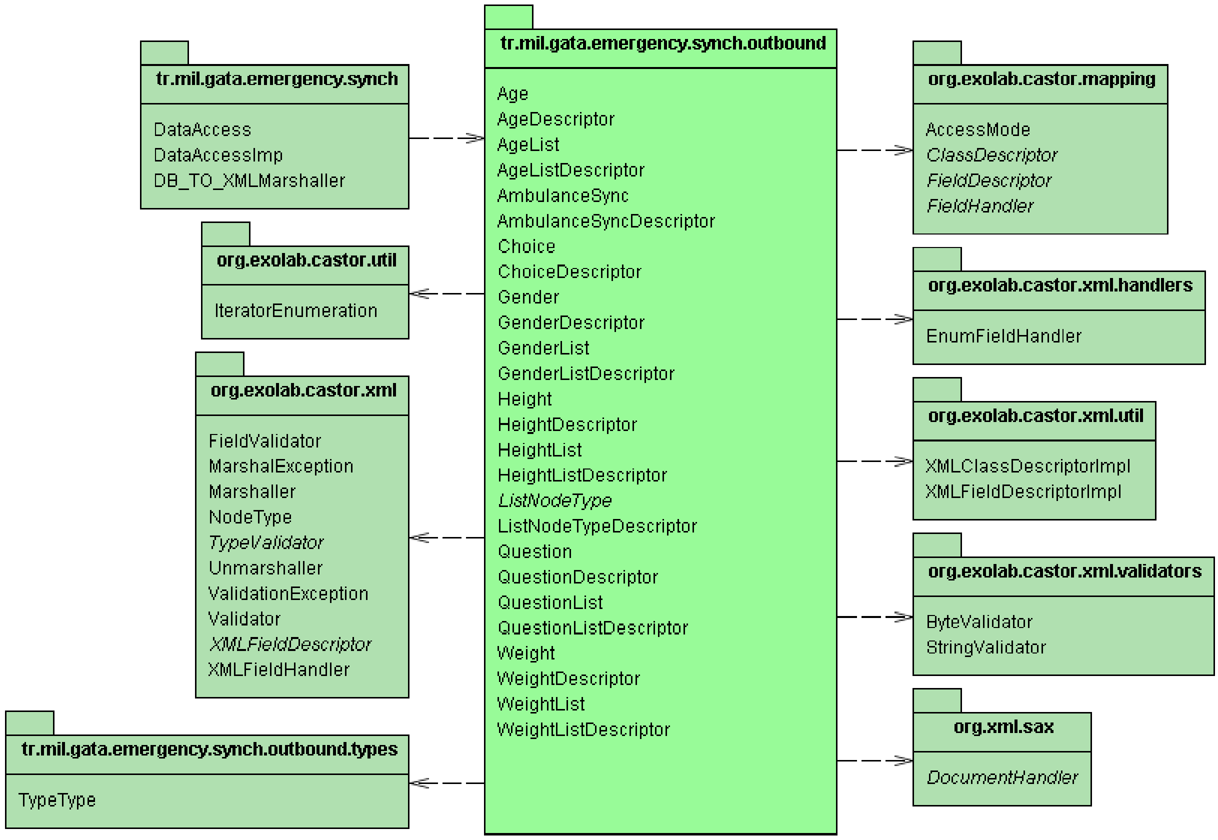

tr.mil.gata.emergency.ambulance
InboundProcessor
client : Socket
fileName : String
input : ObjectInputStream
output : ObjectOutputStream
port : Integer
server : String
xmlData : String
InboundProcessor() : void
runClient() : void
closeConnection() : void
connectToServer() : void
getStreams() : void
receiveData() : void
storeDataFile() : void
tr.mil.gata.emergency.ambulance
AmbulanceFrame
tr.mil.gata.emergency.util
String_TO_XMLMarshaller
XMLMarshaller

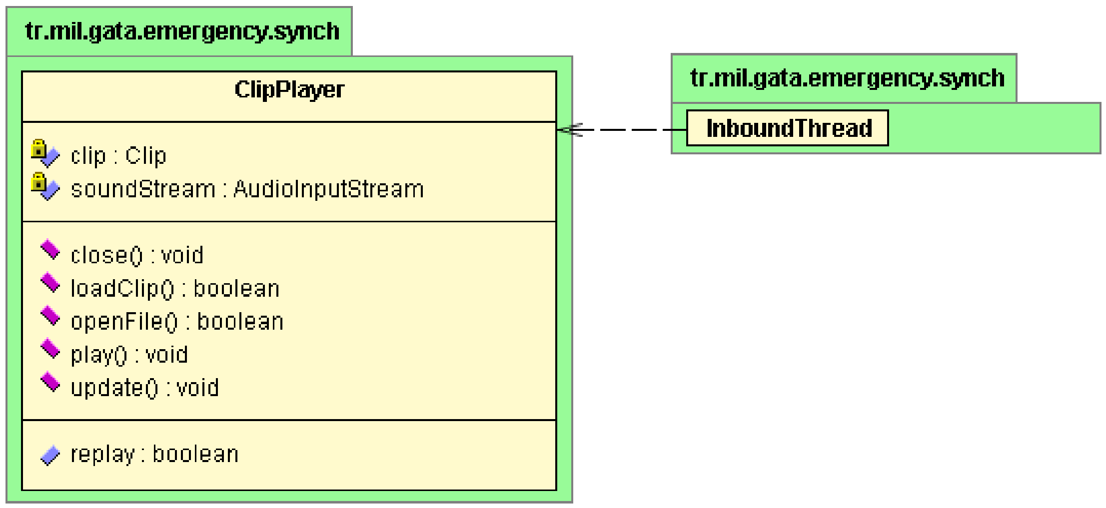

tr.mil.gata.emergency.ambulance
OutboundProcessor
client : Socket
fileName : String
input : ObjectInputStream
output : ObjectOutputStream
port : Integer
server : String
xmlData : String
OutboundProcessor() : void
runClient() : void
closeConnection() : void
connectToServer() : void
getStreams() : void
loadDataFile() : void
sendData() : void
tr.mil.gata.emergency.ambulance
AmbulanceFrame
tr.mil.gata.emergency.util
XML_TO_StringUnmarshaller
XMLUnmarshaller

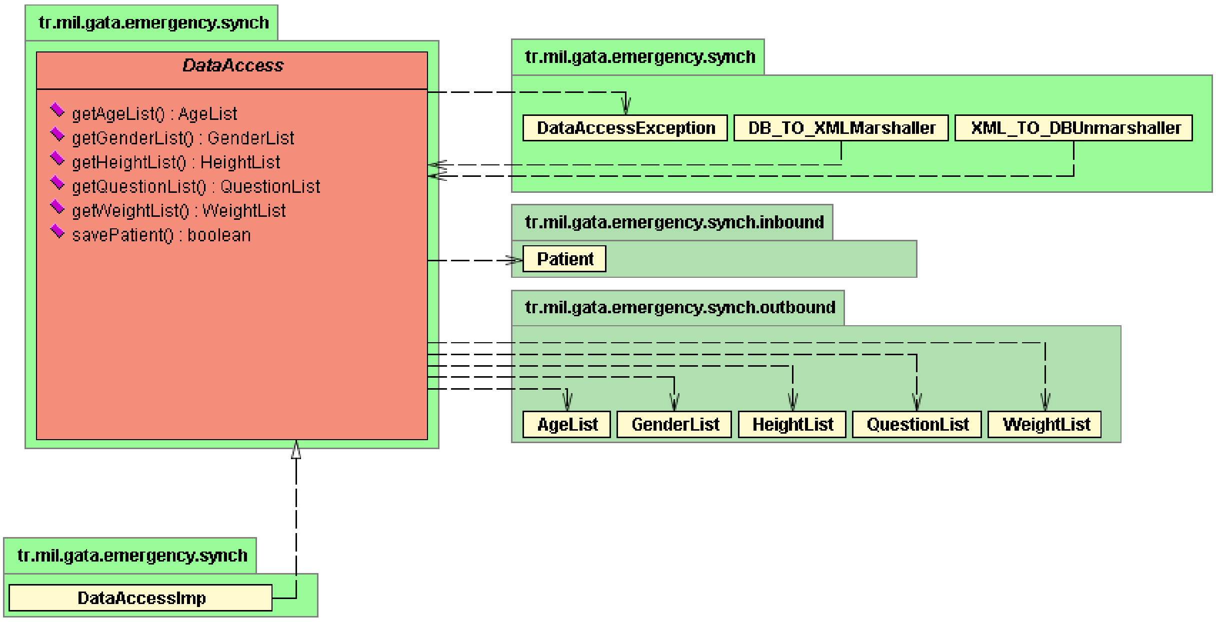

tr.mil.gata.emergency.ambulance
AmbulanceFrame
GUI_TO_XMLMarshaller
tr.mil.gata.emergency.ambulance.inbound
Question
QuestionList
tr.mil.gata.emergency.ambulance
QuestionPanel
ageChoices : Choice
birthPlace : TextField
buttons : Checkbox[]
choicePanel : Panel
currentQuestion : Question
fathersName : TextField
firstName : TextField
genderChoices : Choice
heightChoices : Choice
index : int
lastName : TextField
questionLabel : Label
questionList : QuestionList
weightChoices : Choice
clearComponents() : void
getMultiValuedAnswer() : Answer
getPatient() : Patient
getSingleValuedAnswer() : Answer
hasMoreQuestions() : boolean
init() : void
isCurrentQuestionImage() : boolean
isCurrentQuestionMulti() : boolean
isCurrentQuestionText() : boolean
isCurrentScreenFirst() : boolean
nextQuestion() : void
QuestionPanel() : void
displayMultiValuedQuestion() : void
displaySingleValuedQuestion() : void
tr.mil.gata.emergency.ambulance.inbound
Age
AgeList
AmbulanceSync
Choice
Gender
GenderList
Height
HeightList
ListNodeType
Weight
WeightList
tr.mil.gata.emergency.ambulance.inbound.types
TypeType
tr.mil.gata.emergency.ambulance.outbound
Answer
AnswerList
Choice
Patient
Question

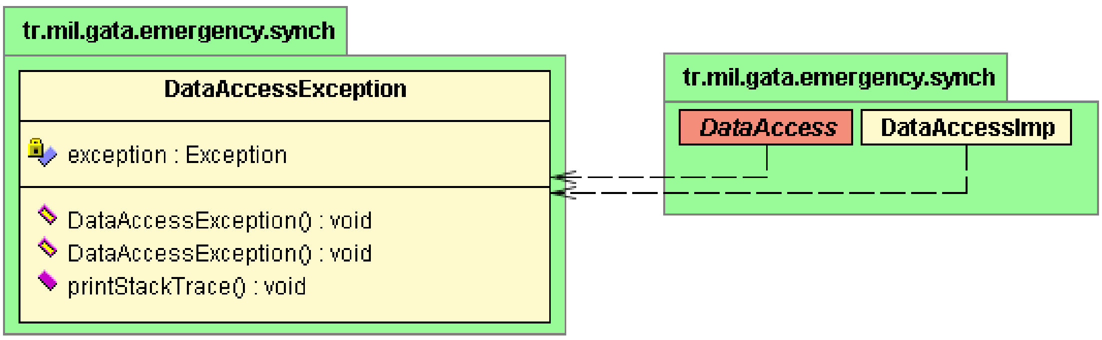

tr.mil.gata.emergency.synch.inbound
AmbulanceTrx
Answer
AnswerList
Choice
Patient
PatientList
Question
tr.mil.gata.emergency.synch
ClipPlayer
DataAccess
DataAccessException
DataAccessImp
DB_TO_XMLMarshaller
InboundProcessor
InboundThread
OutboundProcessor
OutboundThread
XML_TO_DBUnmarshaller
tr.mil.gata.emergency.synch.outbound.types
TypeType
tr.mil.gata.emergency.util
String_TO_XMLMarshaller
XML_TO_StringUnmarshaller
XMLMarshaller
XMLUnmarshaller
tr.mil.gata.emergency.synch.outbound
Age
AgeList
AmbulanceSync
Choice
Gender
GenderList
Height
HeightList
ListNodeType
Question
QuestionList
Weight
WeightList

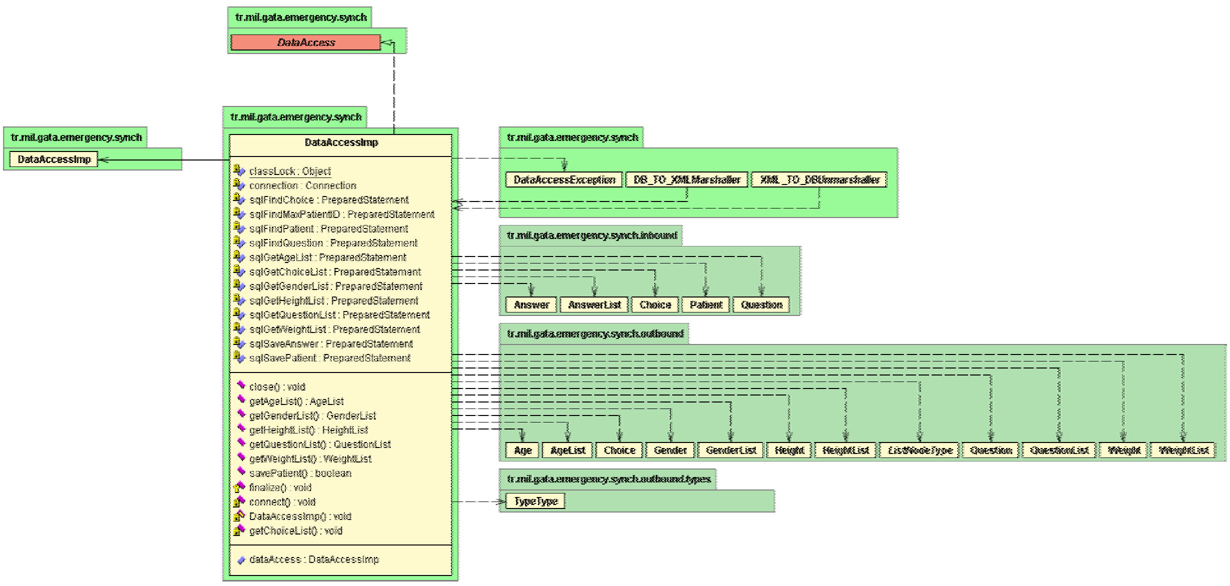

tr.mil.gata.emergency.synch
DataAccess
DataAccessImp
InboundThread
XML_TO_DBUnmarshaller
tr.mil.gata.emergency.synch.inbound
AmbulanceTrx
AmbulanceTrxDescriptor
Answer
AnswerDescriptor
AnswerList
AnswerListDescriptor
Choice
ChoiceDescriptor
Patient
PatientDescriptor
PatientList
PatientListDescriptor
Question
QuestionDescriptor
org.exolab.castor.mapping
AccessMode
ClassDescriptor
FieldDescriptor
FieldHandler
org.exolab.castor.util
IteratorEnumeration
org.exolab.castor.xml.util
XMLClassDescriptorImpl
XMLFieldDescriptorImpl
org.exolab.castor.xml
FieldValidator
MarshalException
Marshaller
NodeType
TypeValidator
Unmarshaller
ValidationException
Validator
XMLFieldDescriptor
XMLFieldHandler
org.exolab.castor.xml.validators
ByteValidator
IntegerValidator
StringValidator
org.xml.sax
DocumentHandler

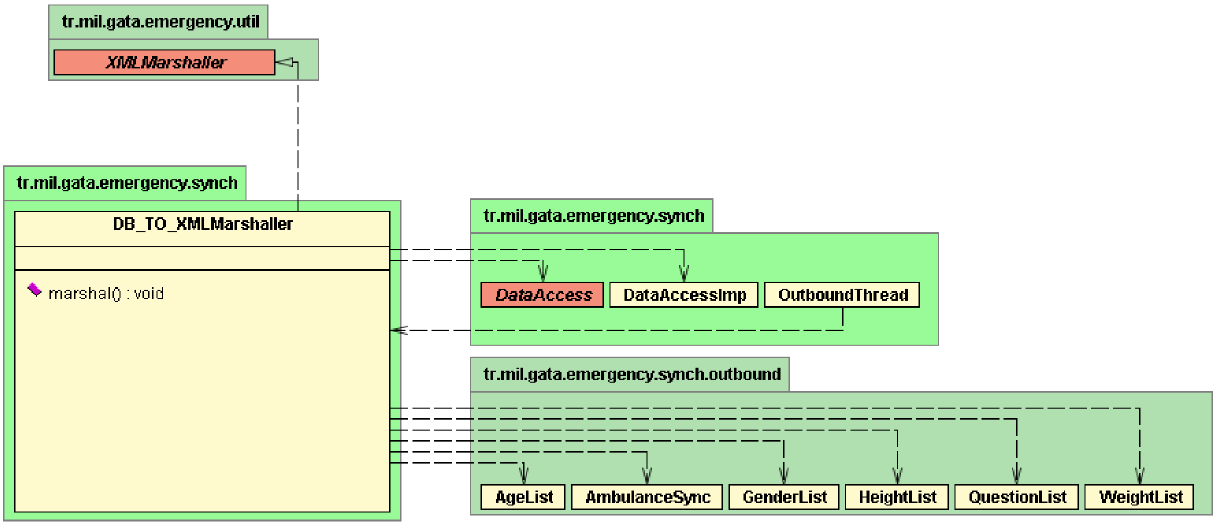

tr.mil.gata.emergency.synch
DataAccess
DataAccessImp
DB_TO_XMLMarshaller
org.exolab.castor.util
IteratorEnumeration
org.exolab.castor.xml
FieldValidator
MarshalException
Marshaller
NodeType
TypeValidator
Unmarshaller
ValidationException
Validator
XMLFieldDescriptor
XMLFieldHandler
tr.mil.gata.emergency.synch.outbound.types
TypeType
tr.mil.gata.emergency.synch.outbound
Age
AgeDescriptor
AgeList
AgeListDescriptor
AmbulanceSync
AmbulanceSyncDescriptor
Choice
ChoiceDescriptor
Gender
GenderDescriptor
GenderList
GenderListDescriptor
Height
HeightDescriptor
HeightList
HeightListDescriptor
ListNodeType
ListNodeTypeDescriptor
Question
QuestionDescriptor
QuestionList
QuestionListDescriptor
Weight
WeightDescriptor
WeightList
WeightListDescriptor
org.exolab.castor.mapping
AccessMode
ClassDescriptor
FieldDescriptor
FieldHandler
org.exolab.castor.xml.handlers
EnumFieldHandler
org.exolab.castor.xml.util
XMLClassDescriptorImpl
XMLFieldDescriptorImpl
org.exolab.castor.xml.validators
ByteValidator
StringValidator
org.xml.sax
DocumentHandler

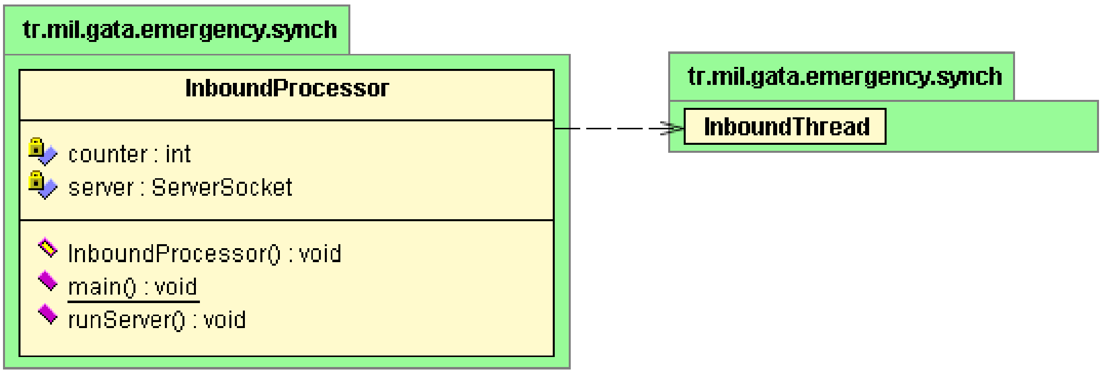

tr.mil.gata.emergency.synch
ClipPlayer
clip : Clip
soundStream : AudioInputStream
close() : void
loadClip() : boolean
openFile() : boolean
play() : void
update() : void
replay : boolean
tr.mil.gata.emergency.synch
InboundThread

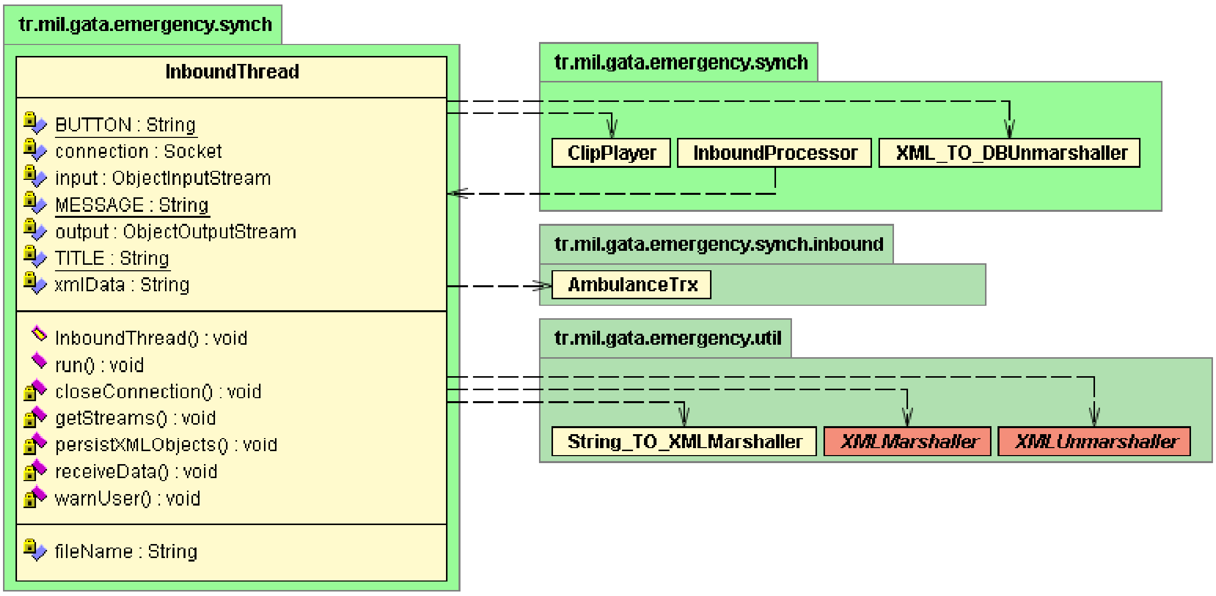

tr.mil.gata.emergency.synch
DataAccess
getAgeList() : AgeList
getGenderList() : GenderList
getHeightList() : HeightList
getQuestionList() : QuestionList
getWeightList() : WeightList
savePatient() : boolean
tr.mil.gata.emergency.synch
DataAccessException
DB_TO_XMLMarshaller
XML_TO_DBUnmarshaller
tr.mil.gata.emergency.synch.inbound
Patient
tr.mil.gata.emergency.synch.outbound
AgeList
GenderList
HeightList
QuestionList
WeightList
tr.mil.gata.emergency.synch
DataAccessImp


tr.mil.gata.emergency.synch
DataAccessException
exception : Exception
DataAccessException() : void
DataAccessException() : void
printStackTrace() : void
tr.mil.gata.emergency.synch
DataAccess
DataAccessImp

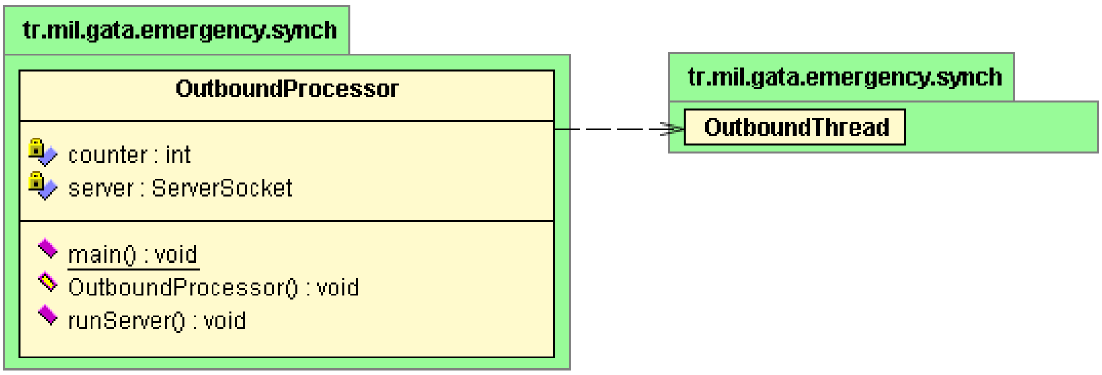

tr.mil.gata.emergency.synch
DataAccess
tr.mil.gata.emergency.synch
DataAccessImp
tr.mil.gata.emergency.synch
DataAccessImp
classLock : Object
connection : Connection
sqlFindChoice : PreparedStatement
sqlFindMaxPatientID : PreparedStatement
sqlFindPatient : PreparedStatement
sqlFindQuestion : PreparedStatement
sqlGetAgeList : PreparedStatement
sqlGetChoiceList : PreparedStatement
sqlGetGenderList : PreparedStatement
sqlGetHeightList : PreparedStatement
sqlGetQuestionList : PreparedStatement
sqlGetWeightList : PreparedStatement
sqlSaveAnswer : PreparedStatement
sqlSavePatient : PreparedStatement
close() : void
getAgeList() : AgeList
getGenderList() : GenderList
getHeightList() : HeightList
getQuestionList() : QuestionList
getWeightList() : WeightList
savePatient() : boolean
finalize() : void
connect() : void
DataAccessImp() : void
getChoiceList() : void
dataAccess : DataAccessImp
tr.mil.gata.emergency.synch
DataAccessException
DB_TO_XMLMarshaller
XML_TO_DBUnmarshaller
tr.mil.gata.emergency.synch.inbound
Answer
AnswerList
Choice
Patient
Question
tr.mil.gata.emergency.synch.outbound
Age
AgeList
Choice
Gender
GenderList
Height
HeightList
ListNodeType
Question
QuestionList
Weight
WeightList
tr.mil.gata.emergency.synch.outbound.types
TypeType


tr.mil.gata.emergency.util
XMLMarshaller
tr.mil.gata.emergency.synch
DB_TO_XMLMarshaller
marshal() : void
tr.mil.gata.emergency.synch
DataAccess
DataAccessImp
OutboundThread
tr.mil.gata.emergency.synch.outbound
AgeList
AmbulanceSync
GenderList
HeightList
QuestionList
WeightList

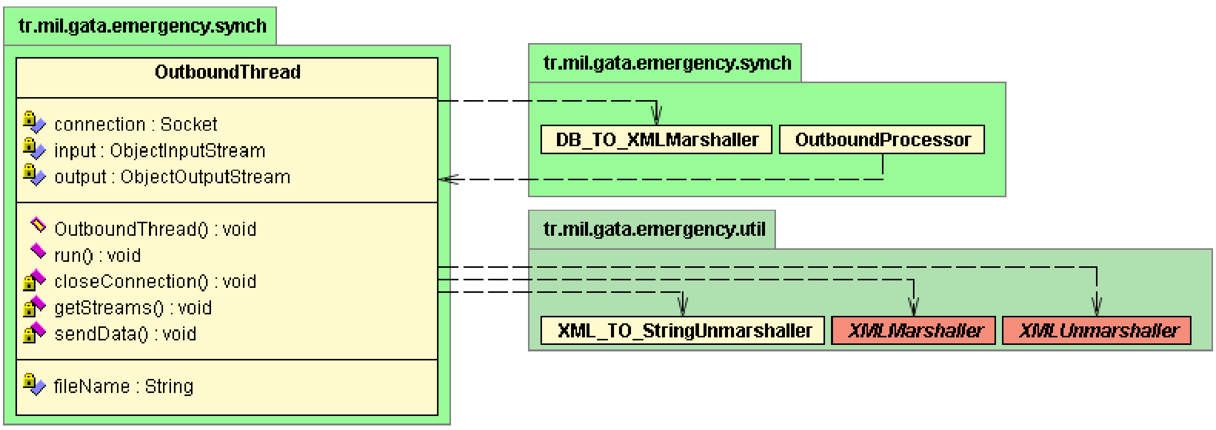

tr.mil.gata.emergency.synch
InboundProcessor
counter : int
server : ServerSocket
InboundProcessor() : void
main() : void
runServer() : void
tr.mil.gata.emergency.synch
InboundThread



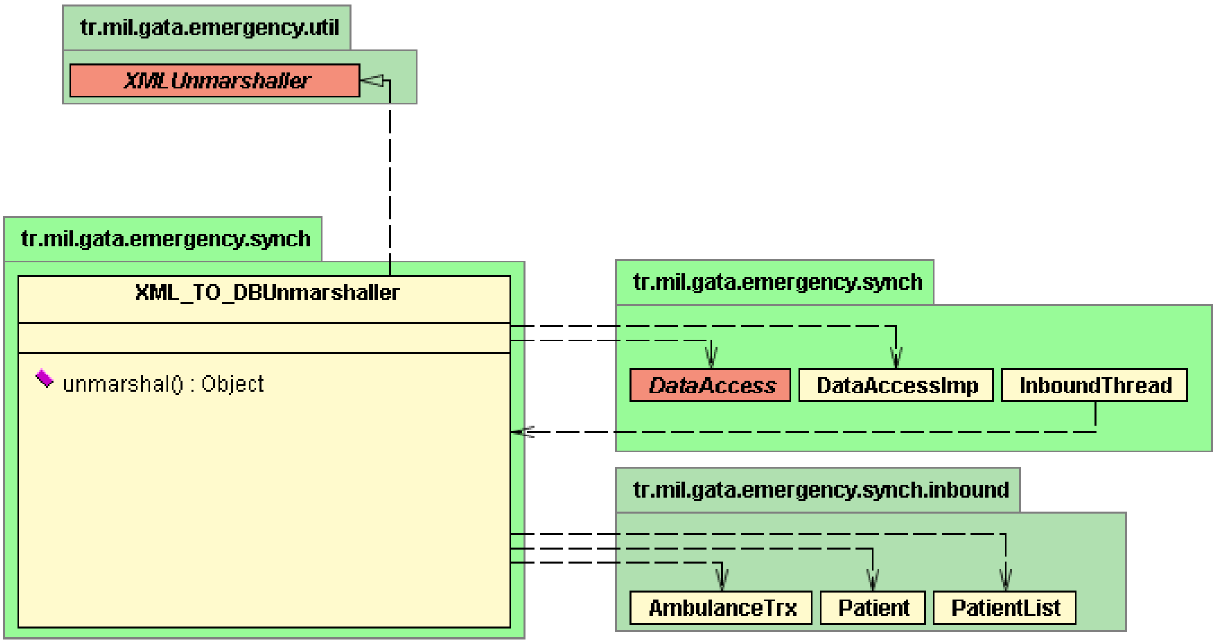
tr.mil.gata.emergency.synch
InboundThread
BUTTON : String
connection : Socket
input : ObjectInputStream
MESSAGE : String
output : ObjectOutputStream
TITLE : String
xmlData : String
InboundThread() : void
run() : void
closeConnection() : void
getStreams() : void
persistXMLObjects() : void
receiveData() : void
warnUser() : void
fileName : String
tr.mil.gata.emergency.synch
ClipPlayer
InboundProcessor
XML_TO_DBUnmarshaller
tr.mil.gata.emergency.synch.inbound
AmbulanceTrx
tr.mil.gata.emergency.util
String_TO_XMLMarshaller
XMLMarshaller
XMLUnmarshaller

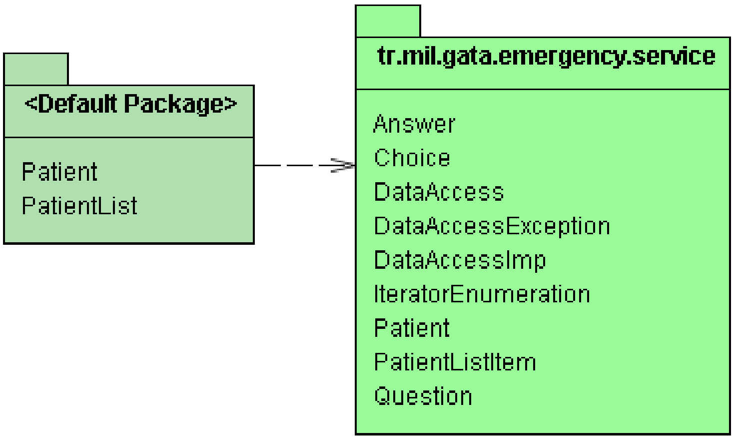


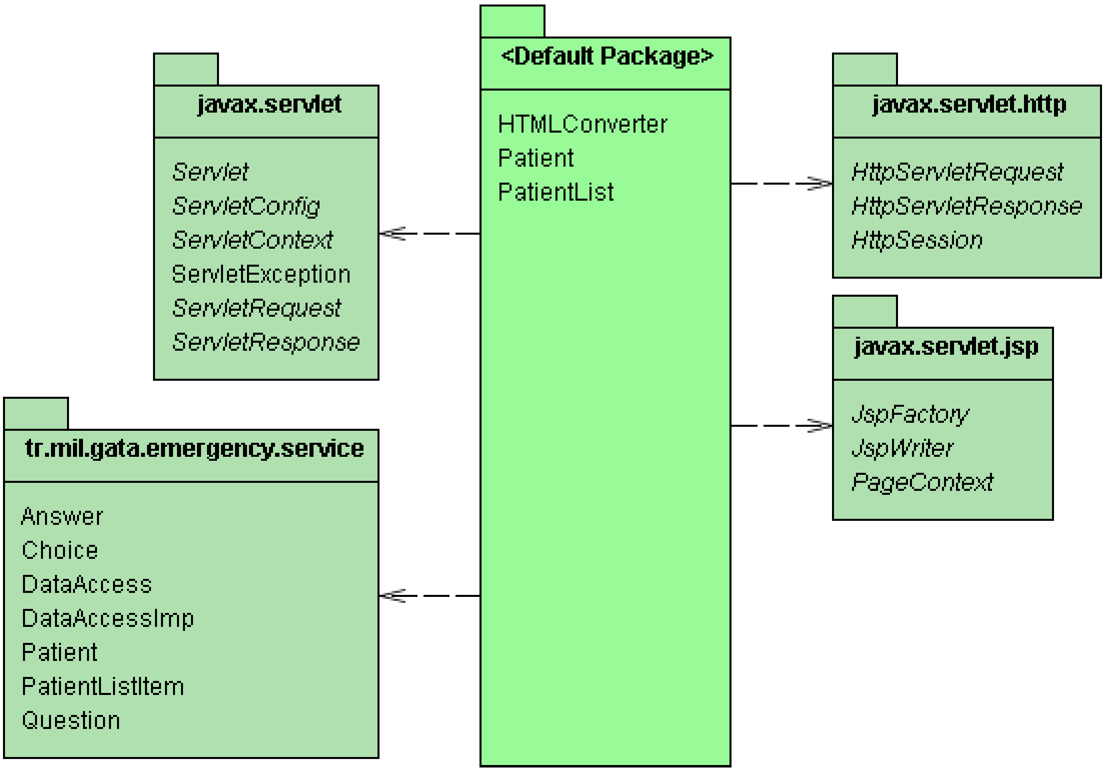
tr.mil.gata.emergency.synch
OutboundProcessor
counter : int
server : ServerSocket
main() : void
OutboundProcessor() : void
runServer() : void
tr.mil.gata.emergency.synch
OutboundThread

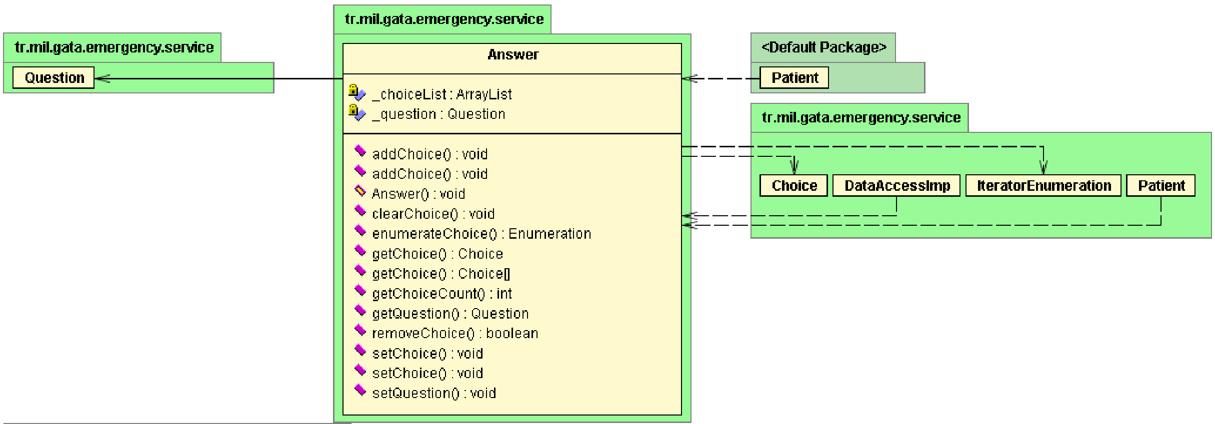


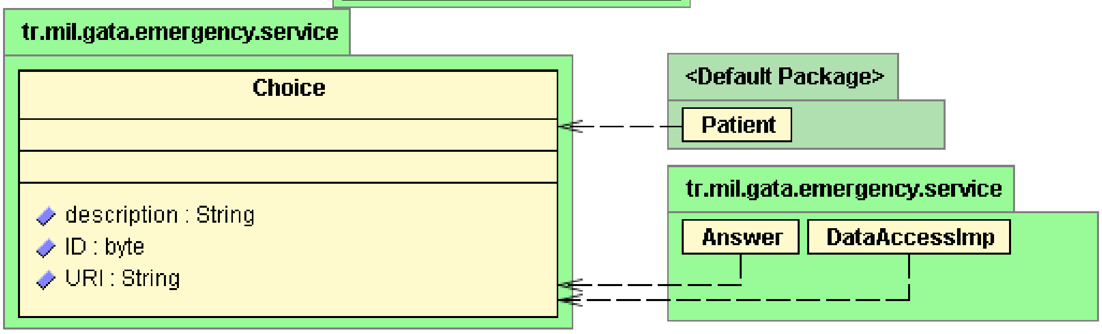
tr.mil.gata.emergency.synch
OutboundThread
connection : Socket
input : ObjectInputStream
output : ObjectOutputStream
OutboundThread() : void
run() : void
closeConnection() : void
getStreams() : void
sendData() : void
fileName : String
tr.mil.gata.emergency.synch
DB_TO_XMLMarshaller
OutboundProcessor
tr.mil.gata.emergency.util
XML_TO_StringUnmarshaller
XMLMarshaller
XMLUnmarshaller

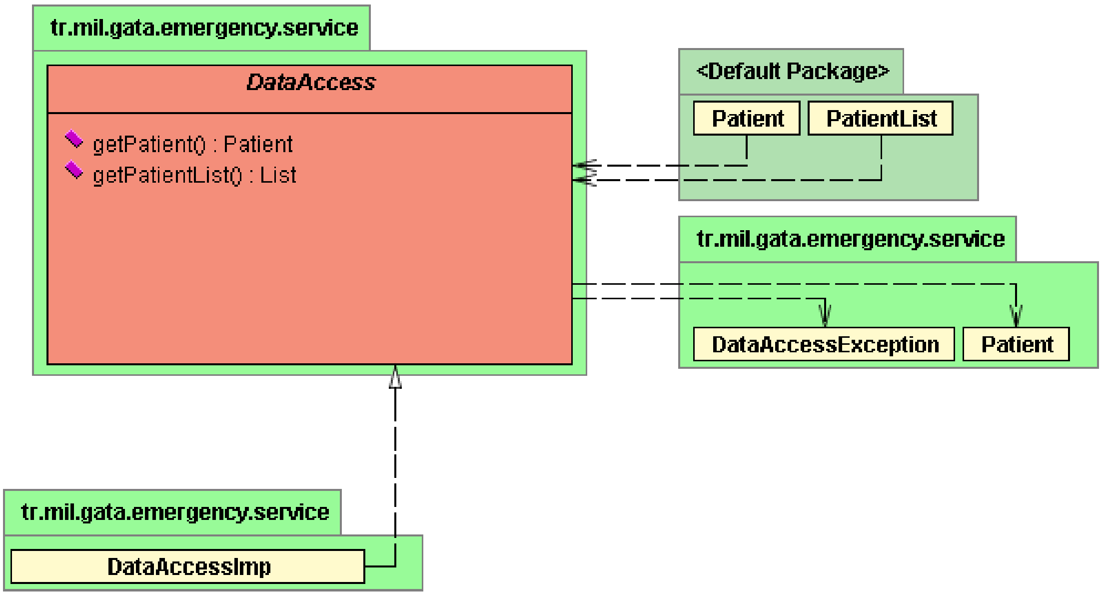

tr.mil.gata.emergency.util
XMLUnmarshaller
tr.mil.gata.emergency.synch
XML_TO_DBUnmarshaller
unmarshal() : Object
tr.mil.gata.emergency.synch
DataAccess
DataAccessImp
InboundThread
tr.mil.gata.emergency.synch.inbound
AmbulanceTrx
Patient
PatientList

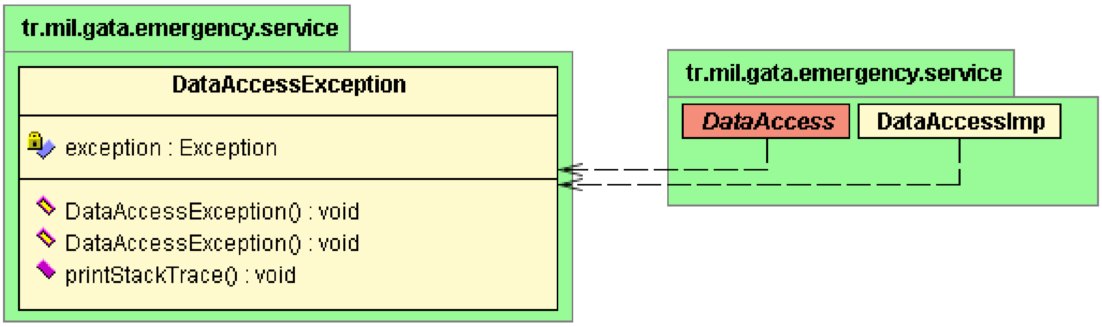

<Default Package>
Patient
PatientList
tr.mil.gata.emergency.service
Answer
Choice
DataAccess
DataAccessException
DataAccessImp
IteratorEnumeration
Patient
PatientListItem
Question

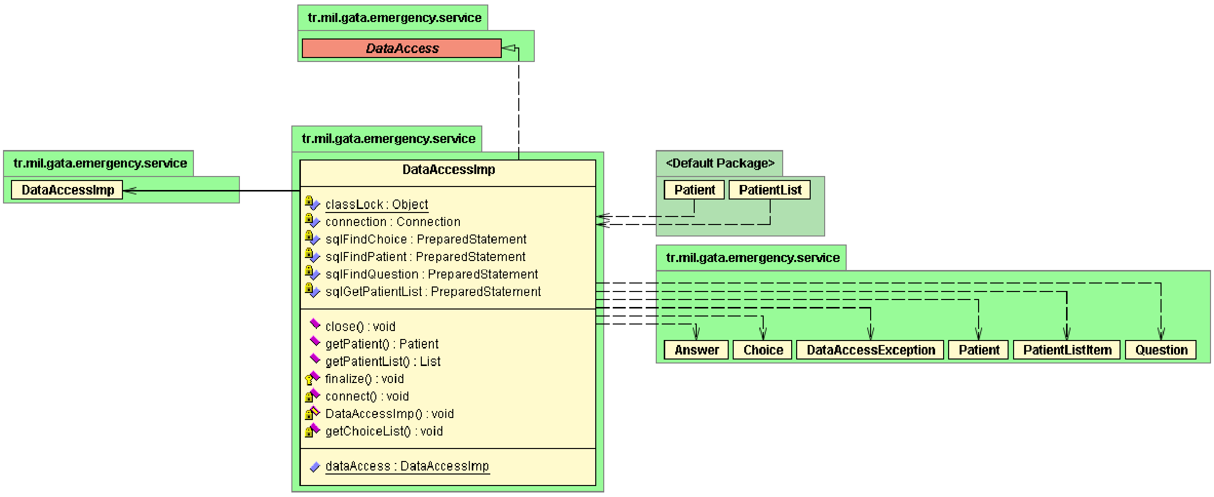

<Default Package>
HTMLConverter
Patient
PatientList
javax.servlet
Servlet
ServletConfig
ServletContext
ServletException
ServletRequest
ServletResponse
javax.servlet.http
HttpServletRequest
HttpServletResponse
HttpSession
javax.servlet.jsp
JspFactory
JspWriter
PageContext
tr.mil.gata.emergency.service
Answer
Choice
DataAccess
DataAccessImp
Patient
PatientListItem
Question

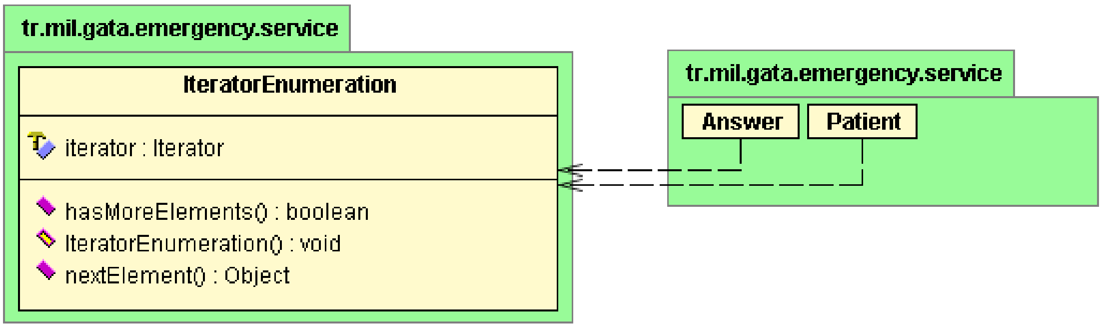

tr.mil.gata.emergency.service
Question
tr.mil.gata.emergency.service
Answer
_choiceList : ArrayList
_question : Question
addChoice() : void
addChoice() : void
Answer() : void
clearChoice() : void
enumerateChoice() : Enumeration
getChoice() : Choice
getChoice() : Choice[]
getChoiceCount() : int
getQuestion() : Question
removeChoice() : boolean
setChoice() : void
setChoice() : void
setQuestion() : void
<Default Package>
Patient
tr.mil.gata.emergency.service
Choice
DataAccessImp
IteratorEnumeration
Patient

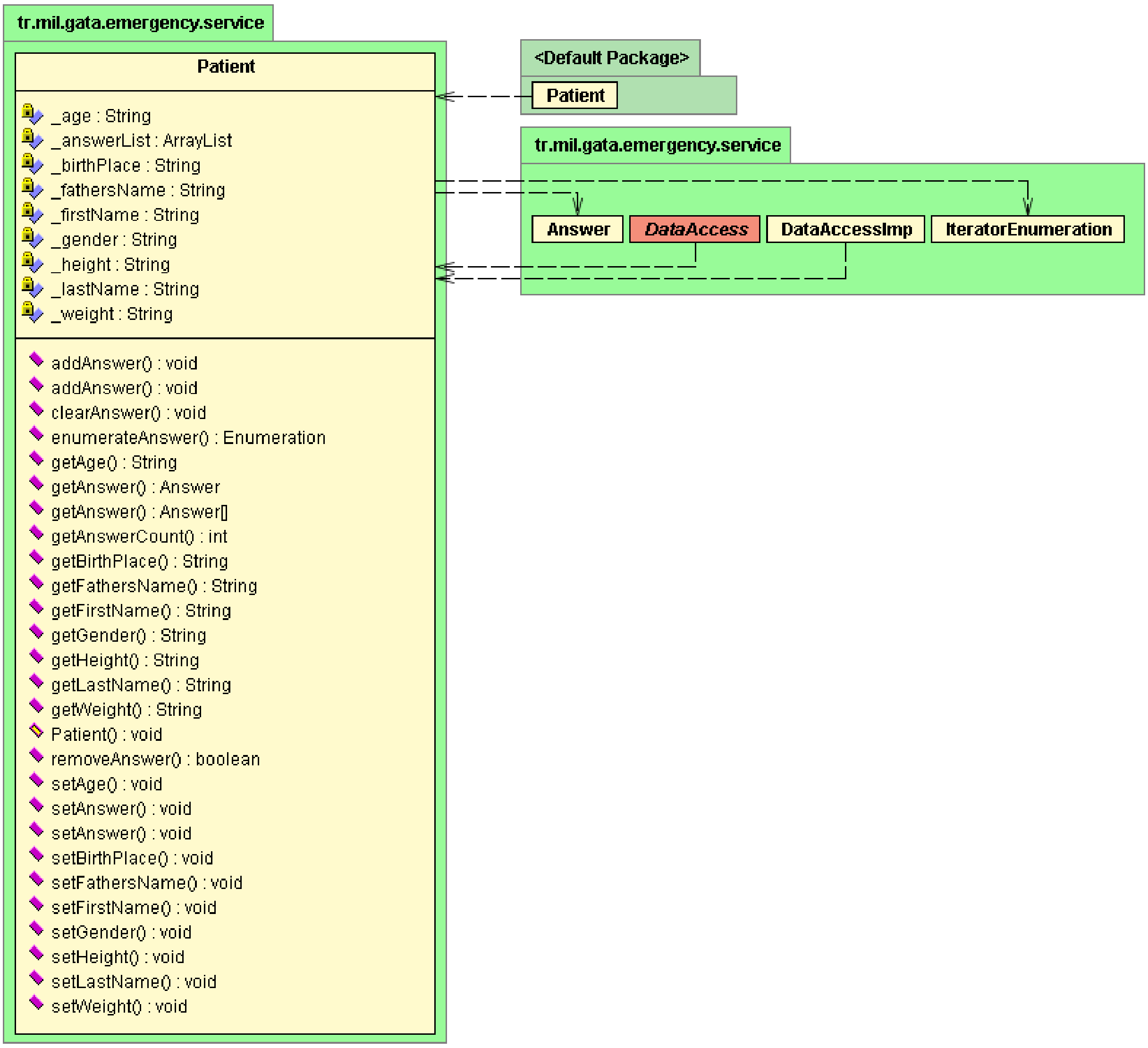

tr.mil.gata.emergency.service
Choice
description : String
ID : byte
URI : String
<Default Package>
Patient
tr.mil.gata.emergency.service
Answer
DataAccessImp

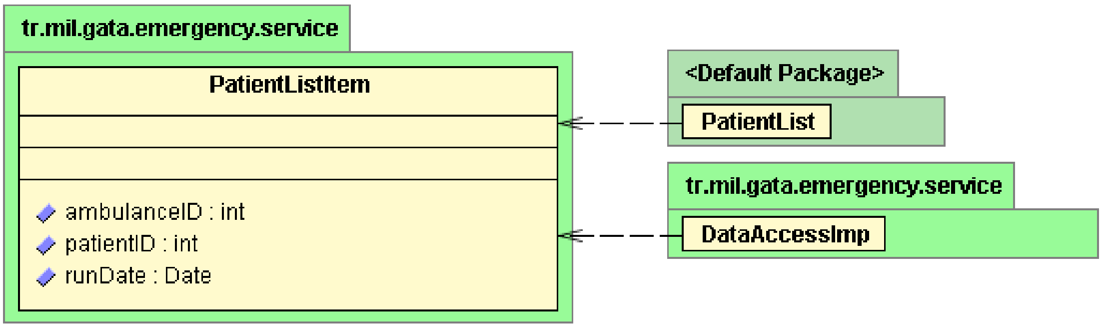

tr.mil.gata.emergency.service
DataAccess
getPatient() : Patient
getPatientList() : List
<Default Package>
Patient
PatientList
tr.mil.gata.emergency.service
DataAccessException
Patient
tr.mil.gata.emergency.service
DataAccessImp


tr.mil.gata.emergency.service
DataAccessException
exception : Exception
DataAccessException() : void
DataAccessException() : void
printStackTrace() : void
tr.mil.gata.emergency.service
DataAccess
DataAccessImp

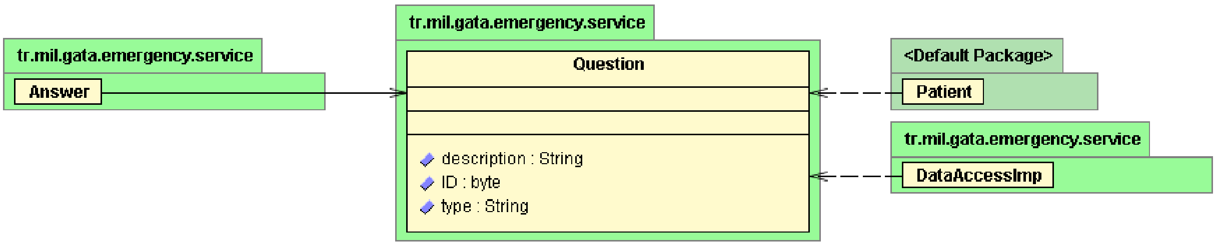

tr.mil.gata.emergency.service
DataAccess
tr.mil.gata.emergency.service
DataAccessImp
tr.mil.gata.emergency.service
DataAccessImp
classLock : Object
connection : Connection
sqlFindChoice : PreparedStatement
sqlFindPatient : PreparedStatement
sqlFindQuestion : PreparedStatement
sqlGetPatientList : PreparedStatement
close() : void
getPatient() : Patient
getPatientList() : List
finalize() : void
connect() : void
DataAccessImp() : void
getChoiceList() : void
dataAccess : DataAccessImp
<Default Package>
Patient
PatientList
tr.mil.gata.emergency.service
Answer
Choice
DataAccessException
Patient
PatientListItem
Question


tr.mil.gata.emergency.service
IteratorEnumeration
iterator : Iterator
hasMoreElements() : boolean
IteratorEnumeration() : void
nextElement() : Object
tr.mil.gata.emergency.service
Answer
Patient


tr.mil.gata.emergency.service
Patient
_age : String
_answerList : ArrayList
_birthPlace : String
_fathersName : String
_firstName : String
_gender : String
_height : String
_lastName : String
_weight : String
addAnswer() : void
addAnswer() : void
clearAnswer() : void
enumerateAnswer() : Enumeration
getAge() : String
getAnswer() : Answer
getAnswer() : Answer[]
getAnswerCount() : int
getBirthPlace() : String
getFathersName() : String
getFirstName() : String
getGender() : String
getHeight() : String
getLastName() : String
getWeight() : String
Patient() : void
removeAnswer() : boolean
setAge() : void
setAnswer() : void
setAnswer() : void
setBirthPlace() : void
setFathersName() : void
setFirstName() : void
setGender() : void
setHeight() : void
setLastName() : void
setWeight() : void
<Default Package>
Patient
tr.mil.gata.emergency.service
Answer
DataAccess
DataAccessImp
IteratorEnumeration



tr.mil.gata.emergency.service
PatientListItem
ambulanceID : int
patientID : int
runDate : Date
<Default Package>
PatientList
tr.mil.gata.emergency.service
DataAccessImp



tr.mil.gata.emergency.service
Answer
tr.mil.gata.emergency.service
Question
description : String
ID : byte
type : String
<Default Package>
Patient
tr.mil.gata.emergency.service
DataAccessImp

APPENDIX D

Ambulance Run Scenario

AmbulanceSync.xml file is shown in Figure D.1. This file contains the look up data – including structured question set for an emergency case (see Appendix A)-. Ambulance application constructs its GUI dynamically based on this file.

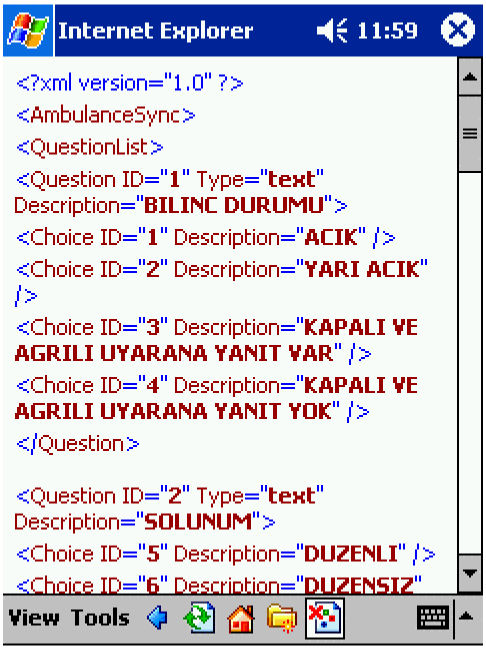


Figure D.1 Ambulance GUI - AmbulanceSync XML file

When paramedic activates ambulance application, main screen is displayed as shown in Figure D.2.

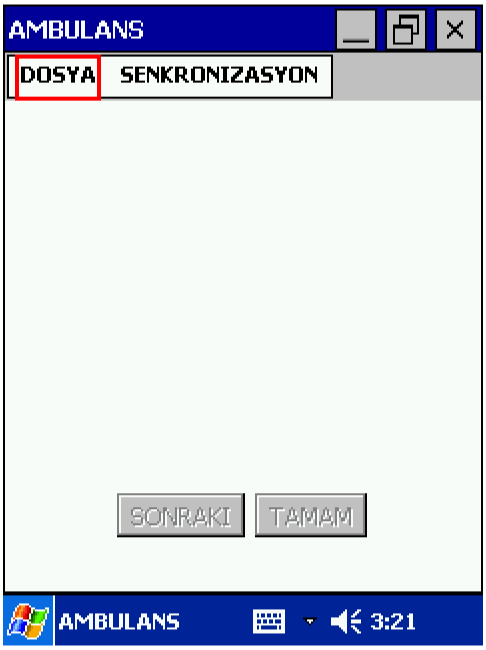


Figure D.2 Ambulance GUI – Main Screen

Upon Paramedic presses "YENI HASTA" menu item under "DOSYA" menu as shown in Figure D.3, she jumps to the patient's demographic fields (Figure D.4).

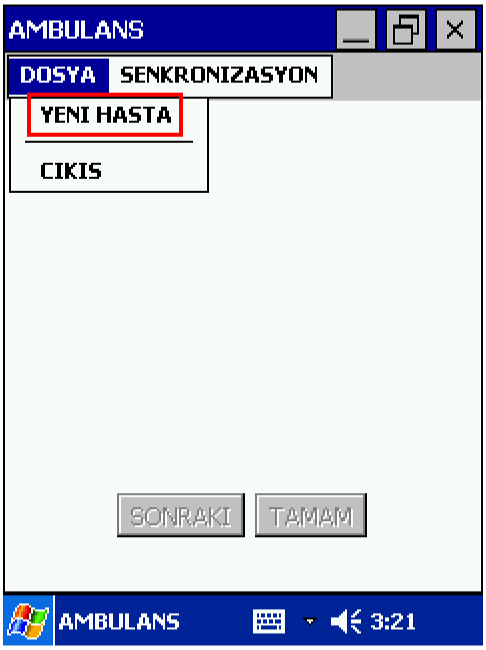


Figure D.3 Ambulance GUI – "DOSYA" Menu

Paramedic enters patient's demographic information using soft keyboard and presses "SONRAKI" button to go to the medical fields as shown in Figure D.4.

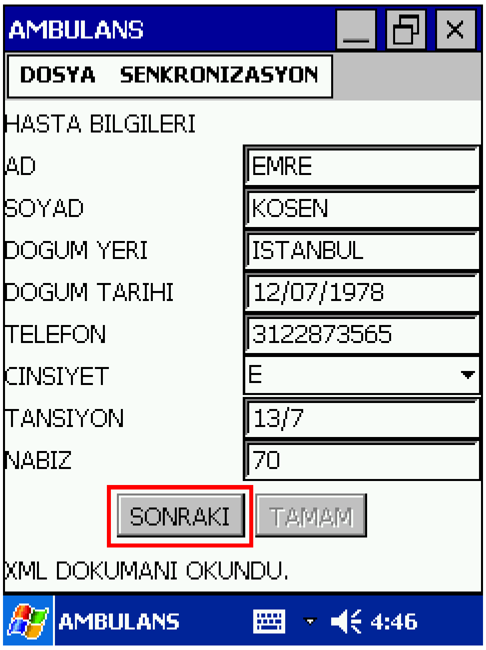


Figure D.4 Ambulance GUI –Demographic Fields Screen

Paramedic selects a value for consciousness parameter and presses "SONRAKI" button to go to the next parameter screen as shown in Figure D.5.

Figure D.5 Ambulance GUI - Consciousness Screen

Paramedic selects a value for respiration parameter and presses “SONRAKI” button to go to the next parameter screen as shown in Figure D.6.

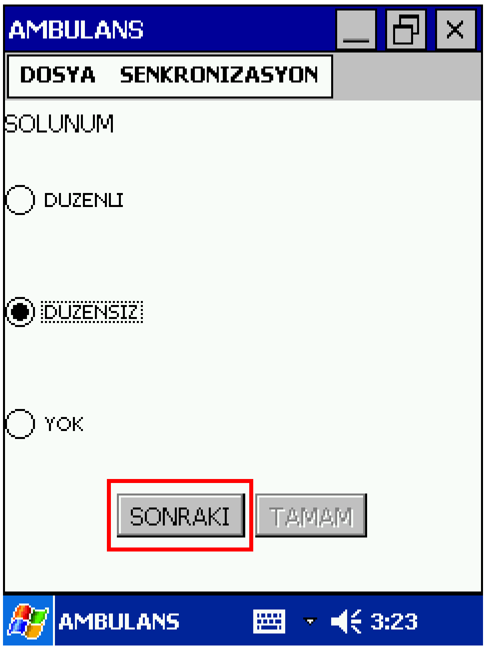


Figure D.6 Ambulance GUI – Respiration Screen

Paramedic selects a value for circulation parameter and presses “SONRAKI” button to go to the next parameter screen as shown in Figure D.7.

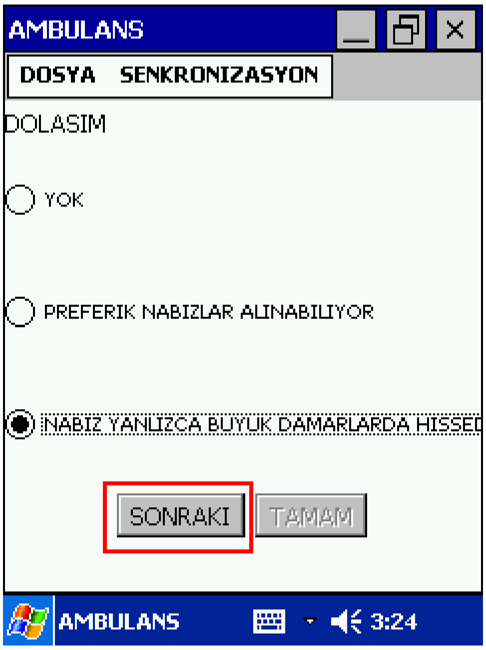


Figure D.7 Ambulance GUI – Circulation Screen

Paramedic may select more than one value for bleeding parameter and presses "SONRAKI" button to go to the next parameter screen as shown in Figure D.8.

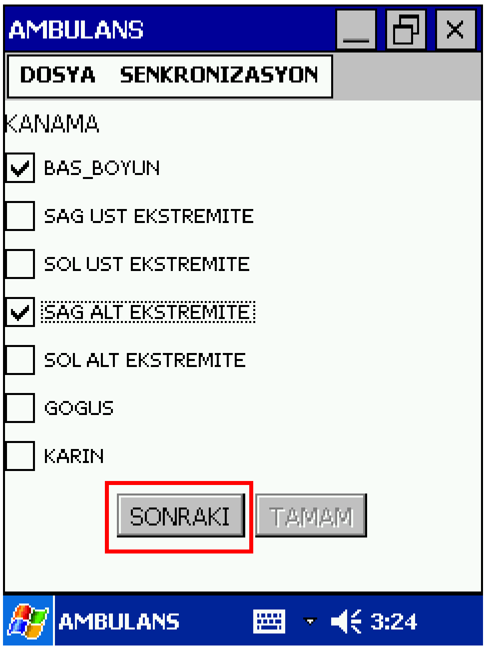


Figure D.8 Ambulance GUI – Bleeding Screen

Paramedic selects more than one value for shock parameter and presses "SONRAKI" button to go to the next parameter screen as shown in Figure D.9.

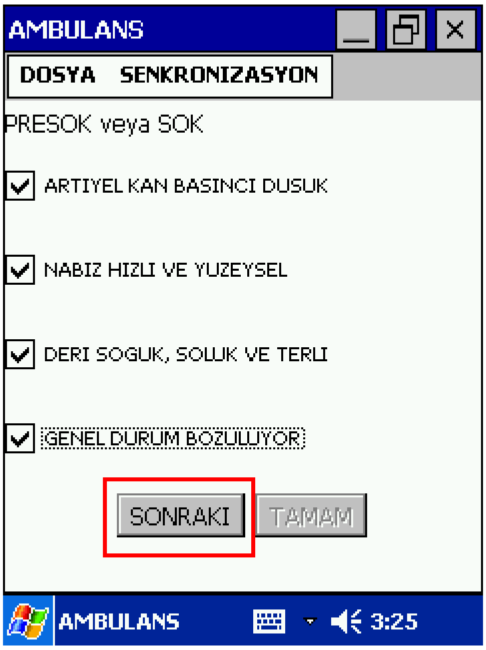


Figure D.9 Ambulance GUI – Shock Screen

Paramedic selects more than one value for cracked parameter and presses "SONRAKI" button to go to the next parameter screen as shown in Figure D.10.

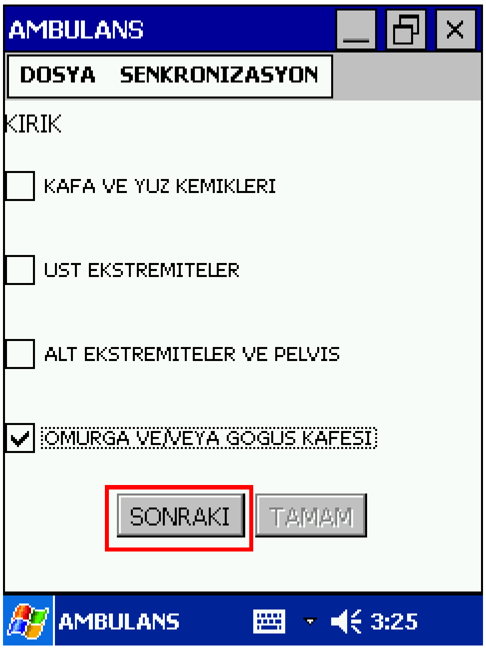


Figure D.10 Ambulance GUI – Cracked Screen

Paramedic selects a value for burnt parameter and presses "SONRAKI" button to go to the next parameter screen as shown in Figure D.11.

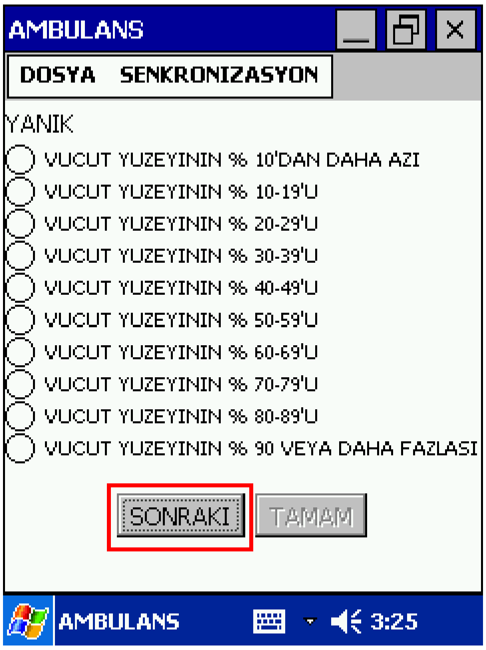


Figure D.11 Ambulance GUI – Burnt Screen

Paramedic selects more than one value for KIBAS parameter and presses "TAMAM" button to complete the question set as shown in Figure D.12.

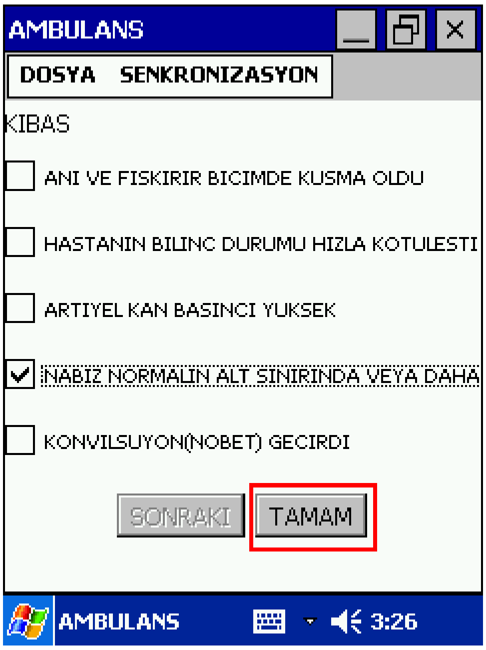


Figure D.12 Ambulance GUI – KIBAS Screen

Upon pressing "TAMAM" button in Figure D.12, patient's EMD is saved in an XML file, AmbulanceTrx.xml, and a status message is displayed to the user as shown in Figure D.13.

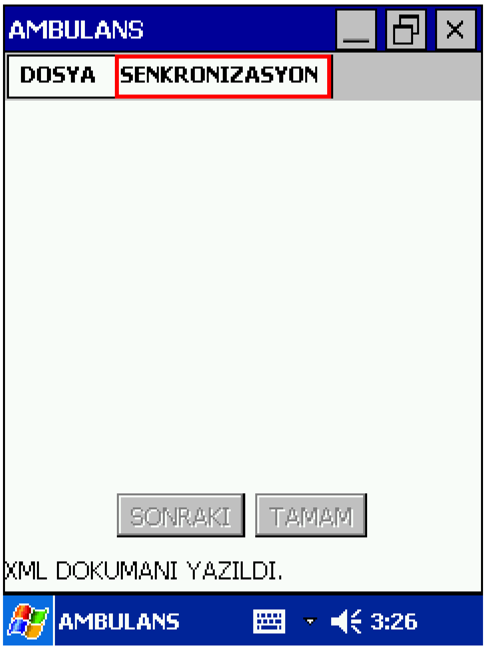


Figure D.13 Ambulance GUI – XML Persistence Screen

AmbulanceTrx.xml file is showing in Figure D.14. This file contains the patient's EMD based on the paramedic's input.

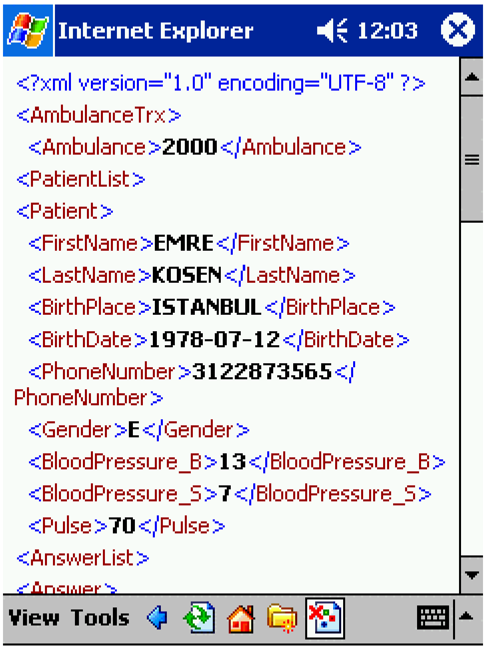


Figure D.14 Ambulance GUI – AmbulanceTrx XML File

Upon pressing "HASTA BILGILERINI GONDER" menu item under "SENKRONIZASYON" menu as shown in Figure D.15, ambulance application transmits patient's EMD to the ED server by serializing AmbulanceTrx.xml file via a socket connection over TCP/IP.

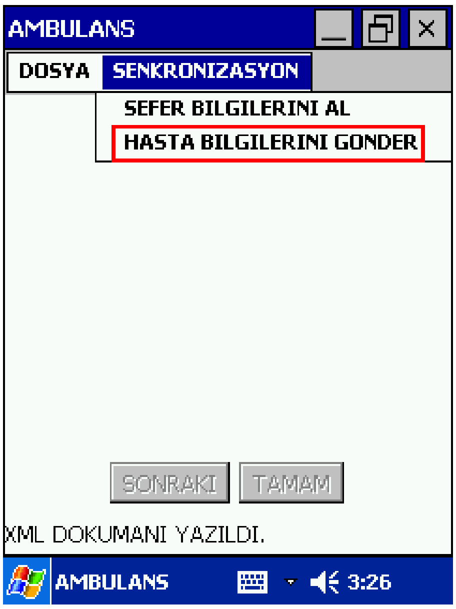


Figure D.15 Ambulance GUI – Synchronization Screen

Upon completion of EMD transmission (XML serialization), synch application binds XML tree nodes to corresponding Java objects and persists these objects into the database via JDBC. If no exception is thrown, synch application plays back a sound message and displays an information message to inform the emergency physician about the impending arrival as shown in Figure D.16.

Figure D.16 Synch GUI - Information message screen

Emergency physician starts browser application running on her desktop PC and requests PateintList.jsp page from the web server. Eventually she goes to the patient list screen displaying the ambulance run info as shown in Figure D.17.

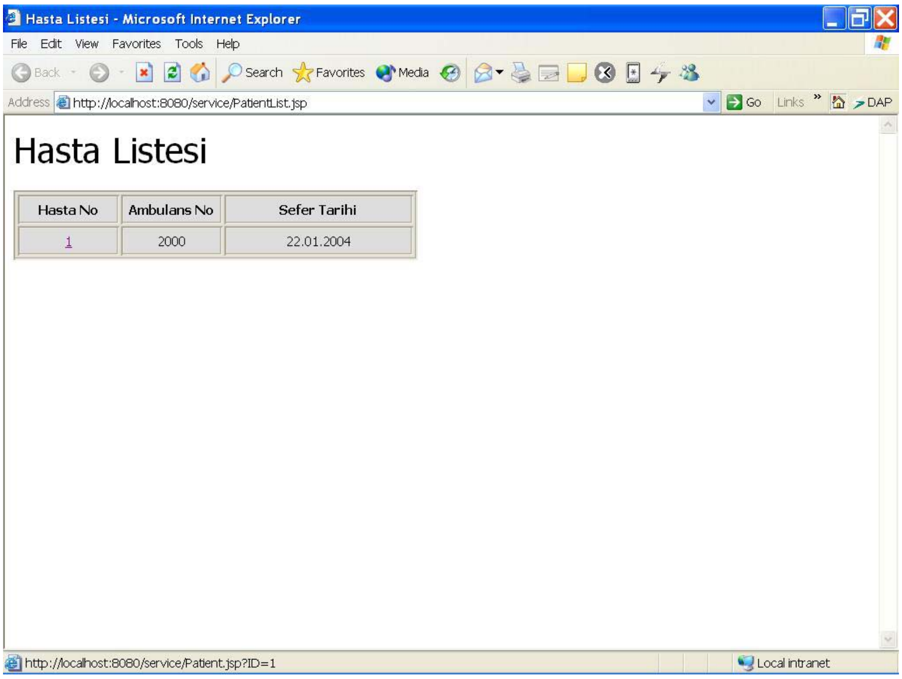


Figure D.17 Service GUI – Patient List Screen

Emergency physician clicks on "Hasta No" field in Figure D.17 and jumps to the Patient.jsp page displaying patient's EMD as shown in Figure D.18.

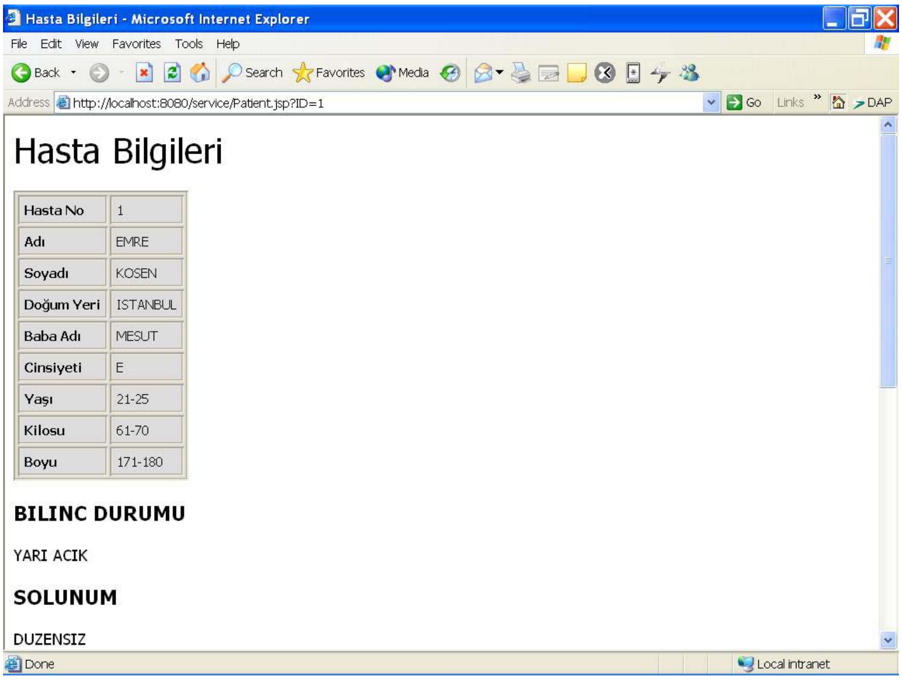


Figure D.18 Service GUI – Patient Screen

APPENDIX E

Ambulans Ekibi Bilgi Formu

(MEDLINE ambulans personeli [paramedik/doctor] tarafından her sefer için doldurulacak, mümkünse)

Tarih: ....../....../.........

Sıra no: ....

Hasta kimlik/adres: ....

Hayati tehlike var mı? ☐ **Evet** ☐**Hayır**

Normal iletisim

Hangi yöntem kullanıldı? ☐ **Telsiz** ☐**Telefon**

Hangi bilgiler aktarıldı? ☐**Kimlik** ☐**Şikayet** ☐ **Tansiyon** ☐**Nabız**
☐ **Öntanı** ....

Normal iletişim ne kadar zaman aldı? **............ dakika**

Elektronik iletişim

Elektronik iletişim sistemi kullanıldı mı? ☐**Evet** ☐**Hayır**

Veri girişi ne kadar zaman aldı? **............ dakika**

Veri aktarımı kaçıncı denemede sağlandı? **..............**

Ambulansın hastane aciline varış saati ve dakikası ? **.......... : ....**

Hastanın acil servise kabul saati ve dakikası? **.......... : ....**

Hastane Acil Ekibi Bilgi Formu

(Ambulasla getirilen bütün hastalar icin acil ekibince doldurulacak, mümkünse)

Tarih: **……/……/………**

Sıra no: **..........**

Hasta kimlik/adres: **...**

Ambulans yoldayken hasta bilgileri acile hangi yöntemle iletildi?

☐**İletilmedi** ☐ **Telefon** ☐**Telsiz** ☐**Bilgisayar**

Ön hazırlık yapıldı mı?

☐ **Ön hazırlık yapılmadı** ☐ **Hasta kayıtları bulundu**
☐ **İlgili ekipman hazırlandı** ☐ **İlgili doktor/resütaston timi çağrıldı**

Hayati tehlike var mı? ☐**Evet** ☐ **Hayır**

Hastanın acile varış saati ve dakikası? **.......... : ....**

Hastanın ilgili doktor / resütaston timi tarafından müdahaleye başlama saati ve dakikası? **.......... : ....**