The University of Melbourne

SWEN90010: High Integrity Software Engineering

Assignment 1

Due Date: 11:59pm, Wednesday 21 March, 2018

1 Introduction

This assignment is worth 10% of your total mark.

The aim of this assignment is to provide you with an opportunity to implement a simplified safety- and security-critical system in Ada. Doing so will help to explore the properties of Ada, and how they relate to safe programming.

You will work in pairs for the assignment. Each pair will submit only one solution, produced jointly by both partners. Working in pairs is necessary since a non-trivial part of this assignment involves carefully understanding the requirements of the system, including discussing and resolving any ambiguities therein, and designing and implementing careful functionality against a set of critical requirements. Part of the assignment will be for you to devise and implement strategies in your pair (see later on) to ensure your solution meets the requirements.

Tip: Perhaps the most difficult part of this assignment is getting your head around the system itself, and the packages already supplied. Be sure to download the provided code early, and to understand the example scenario and how it uses the interfaces of the supplied packages.

2 System overview

The system to be produced as part of the assignment is part of a *simulation* of a real medical device called an *implantable cardioverter-defibrillator* (ICD).

In ICD is a battery-powered device implanted into patients who are at risk of cardiac arrest (heart attack) due to ventricular fibrillation and ventricular tachycardia. The device is battery powered, and monitors electrical impulses from the right ventricle of patient's heart, delivering powerful electric pulses if an anomaly is detected. The device is similar to a pacemaker.

Figure 1[1] shows an illustration of an ICD as it would be implanted in a person. The small box labelled *Implantable Cardioverter-Defibrillator* is where the critical decision making occurs. It consists of several components, however in this project, we will consider only four: (1) a heart rate monitor; (2) a decision-making system (software controlled); (3) an impulse generator; and (4) a wireless network interface.

Figure 2 shows the closed-loop control for these four components. From this figure, one can see that the patient's heart rate is measured using the heart-rate monitor in the device. The ICD software must determine whether the heart is behaving normally, or whether a critical event, such as ventricular fibrillation, is occurring. If it has determined this, it much decide whether to deliver a pulse into the heart using the impulse generator, and if so, how much, thereby directly influencing the beat of the heart. At the same time, the ICD software receives and

[1] Image from Wikimedia Commons

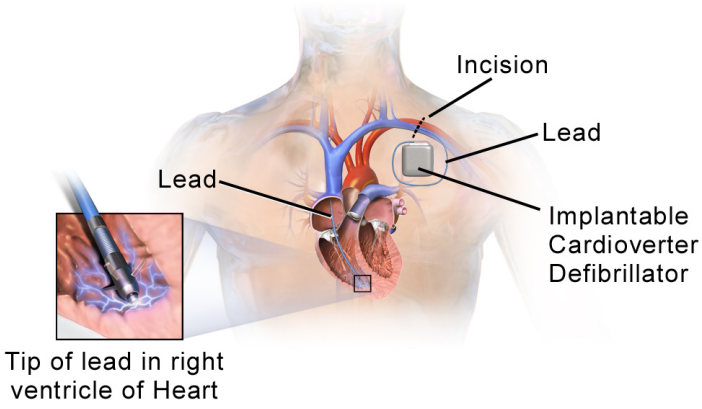

Figure 1: Implantable cardioverter-defibrillator (ICD)

sends messages on the network, for instance to allow the device's settings to be changed or to query its current status.

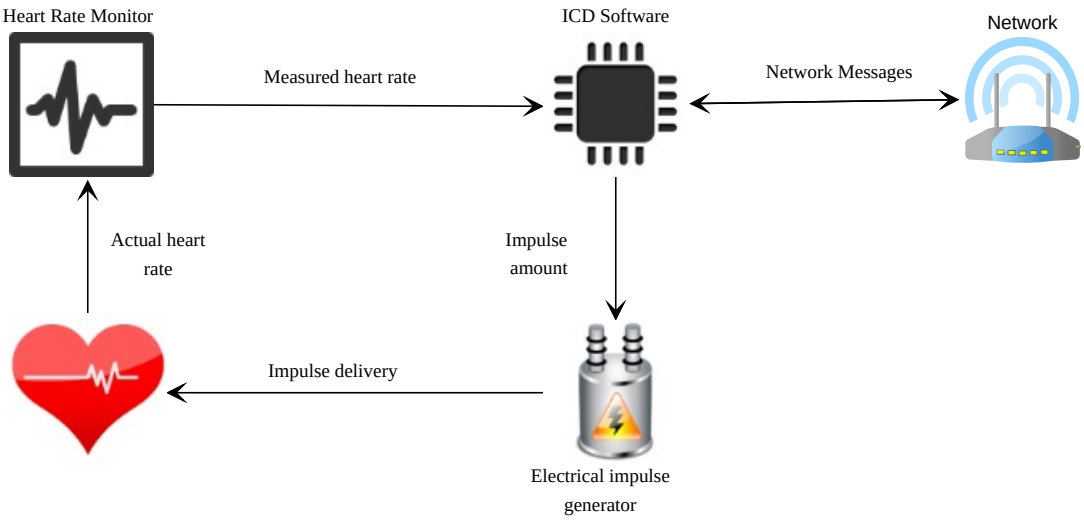

Figure 2: ICD closed loop.

Note the difference between the signal and the actual in this figure. That is, there is an actual heart rate, and a *measured* heart rate, which may be incorrect due to malfunction or low accuracy of the monitor. Similarly, the impulse amount specified to the impulse generator may not be what the generator ultimately delivers, should the generator have problems.

The measured heart rate from the heart rate monitor, which influences how the ICD Software controls the heart rate via the impulse generator, creates a feedback loop in the system, making

this system a simple example of a closed-loop control system[2].

The safety-critical aspects of this system are clear: if the impulse delivered to the patient is wrong, it can cause significant health risks, including, at worst, death.

3 User requirements

3.1 Definitions

D1 *bpm* (*beats per minute*): a unit of measuring heart rate.

D2 *Tachycardia*: A fast heart rate, typically defined as above 100 bpm at rest. A more complete definition: "A rapid life-threatening rhythm originating from the lower chambers of the heart. The rapid rate prevents the heart from filling adequately with blood, and less blood is able to pump through the body"[3].

D3 *Ventricle fibrillation*: "An erratic, disorganised firing of impulses from the ventricles. The ventricles quiver and are unable to contract or pump blood to the body"[2]. Typically leads to cardiac arrest.

D4 *Joules*: The unit of energy used for delivering electrical impulses. Formally defined as the amount of energy required to apply a force of one Newton through one metre.

D5 *ds (decisecond)*: One tenth of a second.

3.2 Principals and Roles

The network interface for the device allows external *principals* to interact with it. Principals may be associated with different *roles*. The following roles are defined:

1. *Patient*: A person who has an ICD implanted into their chest.
2. *Cardiologist*: A medical doctor specialising in cardiology, who treats patients with an implanted ICD. A cardiologist is able to perform certain actions on an implanted ICD, such as reading data and changing settings.
3. *Clinical Assistant*: A medically-trained person who supports treatment of a patient with an implanted ICD. A clinical assistant is able to perform certain actions on an implanted ICD, such as reading data.

Each patient has some particular cardiologist and clinical assistant assigned to care for them. Each device is implanted in a specific patient, who is the *authorised* patient for the device. Likewise, the patient's assigned cardiologist is the *authorised* cardiologist for that patient's device, and similarly for the patient's assigned clinical assistant who is the *authorised* clinical assistant for the patient's device.

[2] `http://www.electronics-tutorials.ws/systems/closed-loop-system.html`
[3] `http://www.webmd.com/heart-disease/heart-disease-glossary-terms`

4 Network Messages

The network component transmits network messages to and from the ICD software component, and so provides the external interface to the closed-loop system. These messages are used to control the device (e.g. turning it on and off), change and read its settings, and to query its recent readings. Besides the device's authorised patient, cardiologist and clinical assistant, other principals might also transmit messages on the network; however those other principals are not authorised to affect the device.

The following message types are supported:

ModeOn : when sent from an authorised principal, instructs the device to transition to the On mode (see Section 4.1 below). The *Source* field indicates the principal that sent the message.

ModeOff : when sent from an authorised principal, instructs the device to transition to the Off mode (see Section 4.1 below). The *Source* field indicates the principal that sent the message.

ReadRateHistoryRequest : when sent from an authorised principal, requests the device to respond with a *ReadRateHistoryResponse* message detailing the recent history of the measured heart rate. The *Source* field indicates the principal that sent the message.

ReadRateHistoryResponse : the type of message sent by the device in response to a *ReadRateHistoryRequest* message from an authorised principal. The *Destination* field identifies which principal the response message is for (i.e. the Source of the ReadRateHistoryRequest message). The *History* field is an array, each of whose elements is a pair containing the measured heart rate and the time at which each measurement was taken. This field should contain the 5 most recent readings and their associated times.

ReadSettingsRequest : when sent from an authorised principal, requests the device respond with a *ReadSettingsResponse* message detailing the device's current settings. The *Source* field indicates the principal that sent the message.

ReadSettingsResponse : the type of message sent by the device in response to a *ReadSettingsRequest* message from an authorised principal. The *Destination* field identifies which principal the response message is for (i.e. the Source of the ReadSettingsRequest message). The *TachyBound* field specifies the current upper bound for tachycardia, while the *JoulesToDeliver* field specifies the current number of joules to deliver for ventricle fibrillation.

ChangeSettingsRequest : when sent from an authorised principal, instructs the device to update its current settings and to respond with a *ChangeSettingsResponse* message. The *Source* field indicates the principal that sent the message. The *TachyBound* field specifies the new upper bound for tachycardia, while the *JoulesToDeliver* field specifies the new number of joules to deliver for ventricle fibrillation.

ChangeSettingsResponse : the type of message sent by the device in response to a *ChangeSettingsRequest* message from an authorised principal. The *Destination* field identifies which principal the response message is for (i.e. the Source of the ChangeSettingsRequest message).

4.1 Modes

The system operates in one of two modes: *off* and *on*:

R1.1 *Off*: In the *off* mode, the closed-loop functionality of the monitor is off to allow authorised medical personnel to change settings on the device.

R1.2 *On*: In the *on* mode, the closed-loop functionality of the device is on, meaning that impulses may be delivered to the patient, and settings cannot be changed.

R1.3 An authorised Cardiologist or Clinical Assistant can switch between the modes.

R1.4 The initial mode is *off*.

4.2 Off mode

The system must support the following requirements for the off mode:

R2.1 In the off mode, the heart rate monitor and impulse generator must be switched off at all times.

R2.2 The patient must receive no electric impulse.

R2.3 An authorised Cardiologist or Clinical Assistant can read the settings; that is, the upper bound for tachycardia, and number of joules to deliver.

R2.4 An authorised Cardiologist can increase or decrease the upper bound for tachycardia.

R2.5 An authorised Cardiologist can increase or decrease the number of joules delivered for a ventricle fibrillation.

R2.6 The initial upper bound for tachycardia when the system starts is 100 bpm.

R2.7 The initial number of joules to deliver in the case of a ventricle fibrillation is 30.

4.3 On mode

The system must support the following requirements for the on mode:

R3.1 If a tachycardia is detected (the heart rate is above the specified upper bound), the impulse generator must deliver ten (10) signals at two (2) joules at the rate of 15bpm *above* the current heart rate.

For example, if a tachycardia is detected and current heart rate is 110bpm, the impulse generator must deliver 10×2 joule signals at 125bpm.

NOTE: the treatment for tachycardia is to *increase* the heart rate. The shocks are designed to short-circuit the ventricular rhythms, attempting to disrupt the rhythm and halt the tachycardia.

R3.2 If a ventricle fibrillation is detected, the impulse generator must deliver one (1) single shock at the specified number of joules. See Section 4.6 for a definition of ventricle fibrillation.

R3.3 If no tachycardia or ventricle fibrillation is detected, the impulse generator should deliver no signal (zero joules).

The system must supporting the following requirements in all modes:

R3.4 An authorised Patient, Cardiologist, or Clinical Assistant can read the last 5 heart rate measurements, and the time they were measured (where "time" is just the number of clock ticks that have elapsed).

4.4 Impulse generator

The system must support the following requirements for regarding the impulse generator:

1. The impulse generator can be switched on or off.

Constraints The impulse generator has the following constraints/properties:

1. The maximum number of joules the pump is able to administer is 45.

4.5 Heart rate monitor

The system must support the following requirements for regarding the heart rate monitor:

1. The heart rate monitor can be switched on or off.

The heart rate monitor has the following constraints/properties:

1. To simplify the assignment, the heart rate monitor provides a heart rate in *beats per minute* (bpm) every decisecond, rather than a wave form, as is common for real heart rate monitors.

   This reading is based on the time between the *previous two beats of the heart.*

4.6 Detecting ventricle fibrillation

Detecting ventricle fibrillation is a non-trivial task, and there are many algorithms available for this – most work on wave patterns, which is far beyond the scope of this subject.

In this assignment, we will use just a rough approximation. A heart is defined to be in ventricle fibrillation if and only if: the average change in heart rate per reading over the previous six readings, is greater than or equal to 10 bpms. That is:

$$\frac{\Sigma_{t=now-6}^{now} \, |(bpm(t) - bpm(t-1))|}{6} \geq 10$$

in which now is the most recent tick, and $bpm(t)$ returns the heart rate at time t.

5 Existing packages

The following packages are available for download from the LMS. The first four simulate the heart, heart rate monitor, impulse generator, and network respectively:

1. `Heart`: A (very crude) simulation of a patient's heart. A heart has heart rate, which can be affected electric impulses. The patient's heart rate changes over time, even if no impulse is sent from the generator.
2. `ImpulseGenerator`: A simulation of an impulse generator. The generator is connected to a patient's heart, receives a target number of joules, and shocks the patient's heart with this amount.
3. `HRM`: A simulation of a heart rate monitor. The monitor is connected to the patient's heart, can monitor the heart rate, and reports the current heart rate; if it could detect one. The monitor has a small error margin in its measures, meaning it is not completely accurate.
4. `Network`: A simulation of the network. Each clock tick the network can possibly deliver a new message to the device. In this simulation, its behaviour is random.
5. `Principal`: A package for representing principals.
6. `RandomNumber`: A package used to generate some random values for simulation purposes. You do not need to use this directly as part of your submission.
7. `Measures`: A package containing the Ada types used to represent beats per minute (BPM) and joules.
8. `ManualOperationExample`: A procedure containing an example simulating a scenario in which the various components are used, but not in closed-loop mode.
9. `ClosedLoop.ads`: A package specification (only) that is the spec for the top-level package (see below) whose body you will implement. (You will also specify and implement a second package `ICD`; see below.)

Note that the first four of these each contain a procedure called `Tick`, which is used to simulate a one decisecond passage of time. So, e.g. when a shock is delivered to a patient's heart, the heart's `Tick` procedure must be called to simulate the delivery of that shock.

6 Your tasks

The tasks for the assignment are listed below:

1. Implement an Ada package called `ICD`, which implements the functionality to provide the necessary calculations of the impulse to be delivered based on the measured heart rate.

   Note: Your solution should be more general than simply adjusting the impulse for the single heart that is implemented in the `Heart` package. That is, it should support hearts with different behaviour.

2. Implement the package body for the `ClosedLoop.ads` package spec. This package must encapsulate instances of the `Network`, `ICD`, `Heart`, `HRM`, and `ImpulseGenerator` packages to fulfil the behaviour of an ICD device. The package interface must adhere to the supplied `ClosedLoop.ads` file. See the comments in that file for further information.

   The functions should use the packages outlined in Section 5 to implement the behaviour of the network, heart, impulse generator, and monitor respectively.

3. Devise strategies within your pair to help to ensure your code correctly implements the requirements (e.g. code reviews, paired programming, requirements analysis and tracing, etc.). Include in your submission a `README.txt` file of **no more than 250 words** that explains the strategies you employed.

You are encouraged to modify the code for the purpose of testing etc., however, we will use our own implementations of `Network`, `Heart`, `ImpulseGenerator`, and `HRM` to run tests as part of the assessment.

7 Criteria

Criterion	Description	Marks
Design	The design of the system is of high quality. The correct components have been included, the design is loosely coupled, and suitable information hiding strategies have been used.	2 marks
Correctness	The implementation behaves correctly with respect to the user requirements. The automated mode detects anomalies correctly and acts on them appropriately.	2 marks
Completeness	The implementation is complete. All components have been implemented and all user requirements have been addressed.	1 marks
Clarity	The design and implementation are clear and succinct.	1 marks
Code formatting	The implementation adheres to the code format rules (Appendix A).	1 marks
Tests	The implementation passes our tests.	2 marks
Strategy	Your pair employed appropriate strategies to help ensure the correctness of your implementation (as documented in your submitted `README.txt` file)	1 marks
Total		10 marks

8 Submission

One of your pair should create a zip file called *your_username*`.zip` or *your_username*`.tgz`. The file should contain your code for the *complete* ICD system, including code provided by subject staff, as well as your `README.txt` file. Your code and `README.txt` should be in the top-level of the zip/tgz file (i.e. should be visible in the current directory immediately after uncompressing your file). Failure to adhere to this requirement might cause your submission to receive 0/2 marks for the Tests criteria.

Submit the zip/tgz file to the LMS.

9 Academic Misconduct

The University misconduct policy applies to this assignment. Students are encouraged to discuss the assignment topic, but all submitted work must represent the individual's understanding.

The subject staff take plagiarism very seriously. In the past, we have successfully prosecuted several students that have breached the university policy. Often this results in receiving 0 marks for the assessment, and in some cases, has resulted in failure of the subject.

A Code format rules

The layout of code has a strong influence on its readability. Readability is an important characteristic of high integrity software. As such, you are expected to have well-formatted code.

A code formatting style guide is available at `http://en.wikibooks.org/wiki/Ada_Style_Guide/Source_Code_Presentation`. You are free to adopt any guide you wish, or to use your own. However, the following your implementation must adhere to at least the following simple code format rules:

- Every Ada package must contain a comment at the top of the specification file indicating its purpose.
- Every function or procedure must contain a comment at the beginning explaining its behaviour. In particular, any assumptions should be clearly stated.
- Constants and variables must be documented.
- Variable names must be meaningful.
- Significant blocks of code must be commented.

  However, not every statement in a program needs to be commented. Just as you can write too few comments, it is possible to write too many comments.
- Program blocks appearing in if-statements, while-loops, etc. must be indented consistently. They can be indented using tabs or spaces, and can be indented any reasonable number of spaces (three is default for Ada), as long as it is done consistently.
- Lines must be no longer than 80 characters. You can use the Unix command "`wc -L *.ad*`" to check the maximum length line in your Ada source files.