Software Engineering Sequence
SE1 / E&CE451 / CS445 / CS645
SE2 / E&CE452 / CS446 / CS646
SE3 / E&CE453 / CS447 / CS647
Overview of the Course Project

School of Computer Science, and Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, ON, Canada, N2L 3G1

Last Revision: January 14th 2004

1 Introduction

For each of the three courses SE1, SE2, and SE3, you will will have a term-long project that involves a different stage in the development of one software system. The software system that is common to the three courses is for a small telephone exchange and its associated information system. The project has been designed to allow you to apply the software engineering principles and techniques discussed in lecture to the problems of specifying, implementing, verifying and validating the implementation of, and enhancing a nontrivial software system.

To give better coverage of various specification techniques and notations, we have designed the course project to incorporate techniques used to develop real-time software for embedded systems and object-oriented techniques used to develop information systems.

For the SE1 (Software Requirements and Specification) course project, you will write a Software Requirements Specification (SRS) for the software system described herein. For the SE2 (Software Design and Architecture) course, you will design and implement the system you specify in SE1. For the SE3 (Software Testing, Maintenance, and Quality Assurance) course, you will test your implementation and will enhance your system with new features.

In each of the three courses, you will work in a group of size three or four. No, you may not form a group of two! Start thinking now about whom you want to work with. Keep in mind that the smaller the group, the more work each member will have to do. However, the larger the group, the more difficult it is to co-ordinate, to reach consensus, and even to find a time when everyone can meet. Since the same project is used in all three courses, it's wise to have the same group members throughout these courses.

The remainder of this document summarizes the essential features of the project. The *Hardware Interface Description* document describes the hardware application program interface (API) provided to you. Lastly, though not essential, if you would like to learn more about Voice over Internet Protocol (IP), there are numerous resources available on the Internet and several books in the Davis Centre library.

Each course provides its own description of the deliverables for that course.

1.1 Definitions

Callee	Call recipient.
Caller	Call originator.
User	1) The people who use the phones in your system. 2) Your client's customers.
Client	1) The company for whom you are developing your system. 2) The TA assigned to your group.

2 System Features

This section describes the essential features of your system. You will likely find the descriptions unclear, and you are expected to ask your client for further details, just like in the real world.

2.1 System Structure

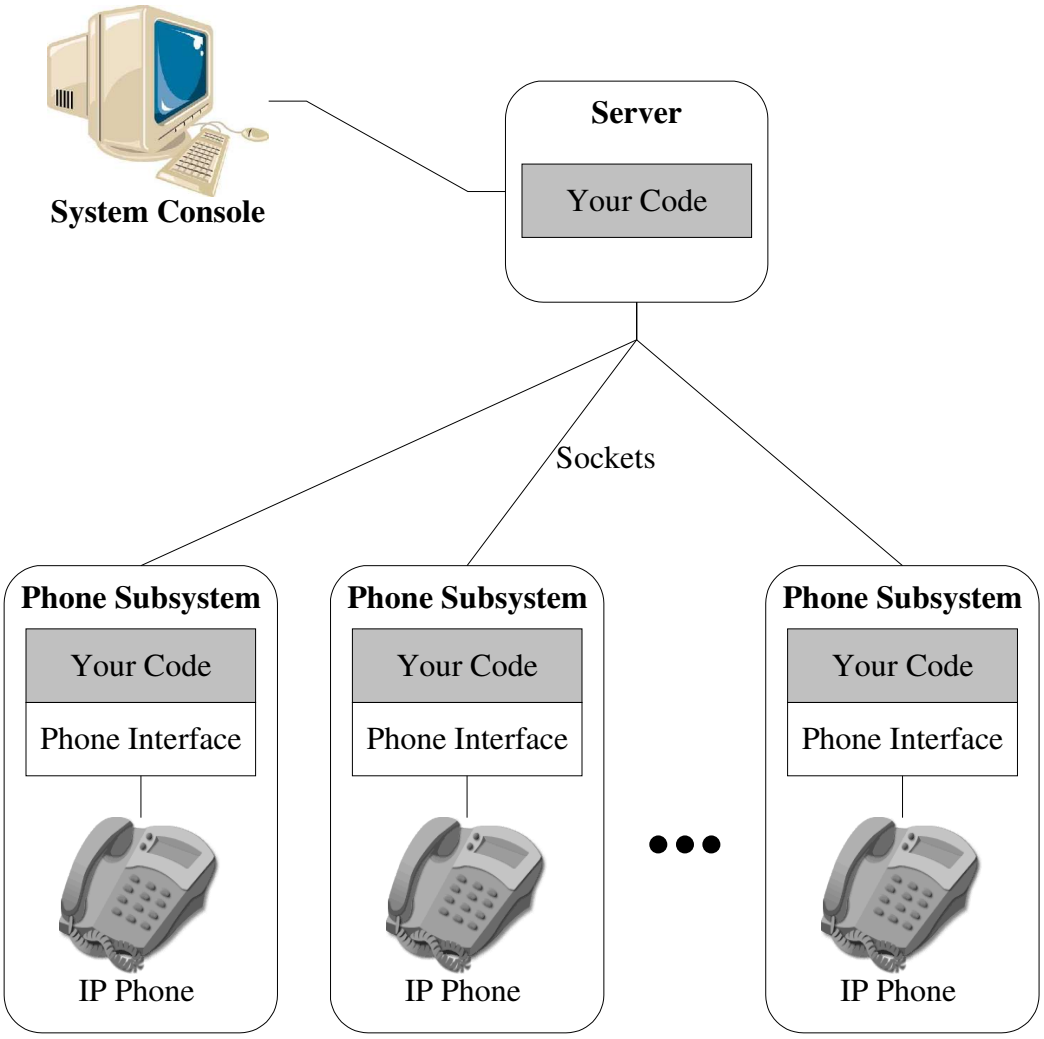

Figure 1: System Configuration – The shaded areas correspond to student written code

At the highest level, your system must consist of a Server and several Phone Subsystems. The Server consists of one or more processes and possibly a database. Each Phone Subsystem amounts to a process that uses the API

described in the Hardware Interface Description document to interface with a single phone.

You are free to distribute or centralize the responsibilities of your system over your Server processes and the Phone Subsystems as you see fit. However, communication between your Server and each Phone must occur over sockets. The reason for this restriction is that, conceptually, the Phone Subsystem is connected to your system via an IP network. Thus, sockets are the default form of inter-process communication.

2.2 Basic Call Processing

Users should be able to send and receive calls. Sending and receiving calls is done by having your system monitor each phone for events, e.g., off-hook, numbers being dialed, etc., and in the case of a successful call, having your system establish an audio connection.

As an example, your system allows a user, Alice, to pick up her phone, and dial Bob's phone number. Bob's phone then starts ringing, causing him to answer his phone. At this point, an audio connection is established so that Alice and Bob can speak to one another. The audio connection is terminated when either Alice or Bob hangs up.

Note that the above scenario describes a successful call. Your specification must also describe scenarios that deviate from this sequence, e.g., the caller dials an invalid phone number, the call recipient's line is busy, the caller hangs up before the connection is established, the caller is on a phone that cannot originate calls, etc. Your specification must also describe what happens if resources are not available for the call, e.g., the maximum number of calls has been reached.

2.3 Dialing Plan

To facilitate basic call processing your system needs to construct and maintain a mapping from dialed numbers to (IP address, port) pairs. Your software uses this mapping to translate the number dialed by a caller, e.g., the digits dialed by Alice, into the IP address and port of the call recipient, i.e., Bob.

Note that this mapping needs to be accessible to the software that does Call Processing as well as the software that handles Administration. You should consider the potential conflicts this may cause, e.g., should a request from the software that does Call Processing have higher priority than a request from the

software that handles Administration? Should caching be used somewhere?.

Phone numbers, also called extensions, are 4 digits long. You may make certain numbers special if you wish. For example, if you want to allow users to access advanced features, they could specify a certain prefix as the indicator for the advanced feature. A concrete example from the real phone system is dialing 1 to call long-distance. A side-effect of this special prefix is that no phone numbers can begin with a 1.

2.4 System Console

The System Console is a Graphical User Interface used by your client's personnel to monitor and control a network of phones. These personnel, called Administrators, must login to the System Console using a login name and a password. You may assume that every Administrator has access to all the features of the System Console.

You may also assume that only one Administrator accesses your system at a time. Despite this assumption, some functions, e.g., find an available phone number, should nevertheless be atomic. This will make it easier to extend your system to accommodate multiple Administrators.

Your software is responsible for checking the Administrator's input to ensure that it does not cause your system's database to become inconsistent. For example, requests to map a single phone number to two different IP addresses should not be allowed.

Additional features of the System Console are described, sometimes implicitly, in the following sections.

2.5 User Accounts

When a user is added to the system, an Administrator must find an unassigned phone number, find an available IP address, and associate the phone number with the IP address.

Each user account has some combination of the following privileges:

- originate calls
- receive calls

Note that if you chose to implement these privileges using filters, then you would have a more powerful feature. For example, the CEO of a company could set up a filter to ignore incoming calls from everyone except high ranking executives and their secretaries.

To cancel a user's telephone service, an Administrator must disassociate the IP address from the phone number, and disassociate both from the user's account, thereby making the IP address and phone number available to future users. User records should be retained indefinitely.

2.6 System Maintenance

The System Console can display hardware status information, such as which phones (read IP addresses) have been allocated to a user, which devices, if any, are out of service, as well as any other information you deem relevant.

An Administrator may request hardware tests at any time. For example, if a user complains that his or her phone does not work, the Administrator may run a connectivity test to verify that the user's phone is connected to the system. If a problem is detected, the Administrator may request that the phone be taken out of service. How an out of service phone is handled in your system is left up to you. Similarly, if a new or repaired hardware device is added to the system hardware, the Administrator may request that the device be enabled.

In addition to requests from the Administrator, the System Console must be able to receive error messages, e.g., Hardware Failure. The System Console must alert the Administrator whenever there is an error, e.g., new negative test results, so that the operator can take immediate action, e.g., request additional tests, take the device out of service, or both.

2.6.1 Automatic Hardware Fault Detection

Your system should monitor all phone processes and periodically run some tests to verify that the phone is operating normally. For example, a message could be sent to each phone process every 30 seconds. If a phone process does not respond within a reasonable length of time, then the system can assume that an error has occurred.

If a phone is turned off, unplugged, or suffers a catastrophic software failure then the system must detect this and alert the operator of the failure. The phone

software may fail in any number of ways, like any piece of software. The system's automatic hardware fault detection needs to detect only that a phone has become completely unresponsive. To prevent resource leakage, any resources in the system associated with an unresponsive phone must be released.

When the problem with a previously unresponsive phone is corrected, the system will detect the correction and make the phone available for use again, restoring any resources the phone was previously using.

In more concrete terms of what you will be implemented in SE2, we can simulate a phone's crashing or hanging by killing or suspending, respectively, the corresponding phone process. Likewise by restarting or unsuspending the phone process we can simulate a phone's coming back up.

2.6.2 Call Processing Reset

Call processing software tends to be very complicated, largely due to its concurrent nature. If the call processing stalls due to some concurrency bug or any other reason (something that happens in real phone systems at times), the system needs to be able to gracefully recover to some default state, that is, without completely stopping the process and restarting it, and without shutting the system down and then restarting it.

The Administrator must have the ability to reset the call processing for any phone. Depending on the architecture of the system, this ability may mean resetting the call logic in the affected phone process, the server, or both.

Note the difference between this Call Processing Reset and the Automatic Hardware Fault Detection. When the call processing is reset, the affected process or thread may still be responsive. However if another thread is waiting on an impossible event, the resetting must be able to recover the call processing from the waiting.

2.6.3 Load Balancing

The Administrator must be able to set a value for the maximum number of calls allowed in your system. When a user tries to place a call, your system should verify that the maximum number of calls has not been exceeded. If you're feeling ambitious, you may display the system load. You may even develop a means of calculating a suggestion for the Administrator.

It is important to consider when to check or increment this number. That is, should it be incremented when the user picks up the handset, returning a dial tone only if he or she can place a call, or should it be incremented when the system attempts to establish an audio path? The latter might annoy the user whose call is not allowed, i.e., he or she wonders, “Why didn’t you tell me before?”.

It is also important to consider what happens if a phone crashes during a call or if a phone is unplugged when the system decrements the number of calls. It might be helpful for you to look at a book on distributed systems to help you develop an algorithm to handle this case, although creating your own algorithm would not be too difficult.

2.7 Billing

Telephone calls are not free. The cost of a call depends on the number dialed, the duration of the call, when the call was established, i.e. the time of the day and the day of the week, and the caller’s plan[1].

To facilitate billing, your system needs to keep a record of every established call. This record indicates who the caller was, so that he or she can be billed; the destination phone number; the time and duration of the call; as well as any other information you deem necessary.

Each user is sent a bill showing all charges incurred during a defined billing period. The System Console can display a user’s bill for any billing period. The default billing period is the current billing period. For each call a bill shows, at a minimum, the number dialed, the day of the week and the time of the day when the call started, the duration of the call, the rate per minute for the call, and the charge for the call; the bill also shows the total charge for all calls and the sum of all charges incurred during the relevant billing period.

At the end of every billing period, your software automatically issues a bill to each user who has an outstanding balance. In addition, a bill is immediately issued to a user if the user’s service is cancelled.

Bill Payments are recorded by an Administrator using the System Console. If a user fails to make a payment, a warning appears on his or her next bill. An

1 Depending on the features that your system supports and how you implement them, the charge may also depend on the callee’s plan. For example, although it is not required, you may choose to support toll-free calling, wherein the call’s recipient is charged instead of the caller. This feature could be implemented using billing plans.

Administrator has the authority to suspend or cancel a user's telephone service if the user fails to pay his or her bill.

An Administrator can change amounts charged for calls by adding new billing plans or editing existing ones. The Administrator can also change which plan a user subscribes to.

Each billing plan specifies:

- the regular charge rate for calls
- one or more periods, i.e. days of the week and times of the day
- the discount rate for each of these periods; for simplicity, you may use a percentage of the regular charge rate.

Your software is responsible for checking that the dates and times of the discount periods don't overlap. Changes to a user's billing plan take effective immediately.