Deliverable 5

Dental Office Computerized Billing System Requirements Report

Casey Galloway, Colton Howard, Peter Carew, Glen Fridman, Tyler Wood, and Kelly Wetherbee

Mar. 24, 2016

CIS 427.01

Project Management with Practice

Table of Contents

Deliverable 5

Dental Office Computerized Billing System Requirements Report

Casey Galloway, Colton Howard, Peter Carew, Glen Fridman, Tyler Wood, and Kelly Wetherbee

Introduction

After hearing the receptionist at their dentist's office complain about all the handwritten paperwork required to run the office, your professor offered your class's Microsoft Access database skills to assist with the design and set up of a computerized database billing system for the office. Mavis Walker, the dentist's office receptionist and manager, indicated three main objectives that the new billing system would be required to perform and gave some relevant information about the dentist's office. Besides Mavis the office has one dentist and one dental assistant who gets paid by the hour. Various procedures can be performed at the office and each procedure has a set specific cost. She also noted that a patient may receive more than one procedure during a single visit. According to Ms. Walker the first main objective deals with checking the patient out at the end of the appointment. During check out, the receptionist should be able to enter the procedure name or the procedure ID along with the patient name or patient ID. The second objective is tracking the dental assistant's hours. Lastly, the system must generate monthly bills for all the patients without dental insurance.

Context Diagram

Figure 1 is the context diagram for this particular billing system. The context diagram is the highest level of logical design for a system and shows the data flow interaction between level 0 processes, data stores, and external entities. Further breakdown of the level 0 processes usually occurs, but is not covered in this report. Based on Figure 1, this system has two external entities, three level 0 processes, four data stores, and various data flows. The data flows are broken down in the data dictionary which can be found in Appendix A.

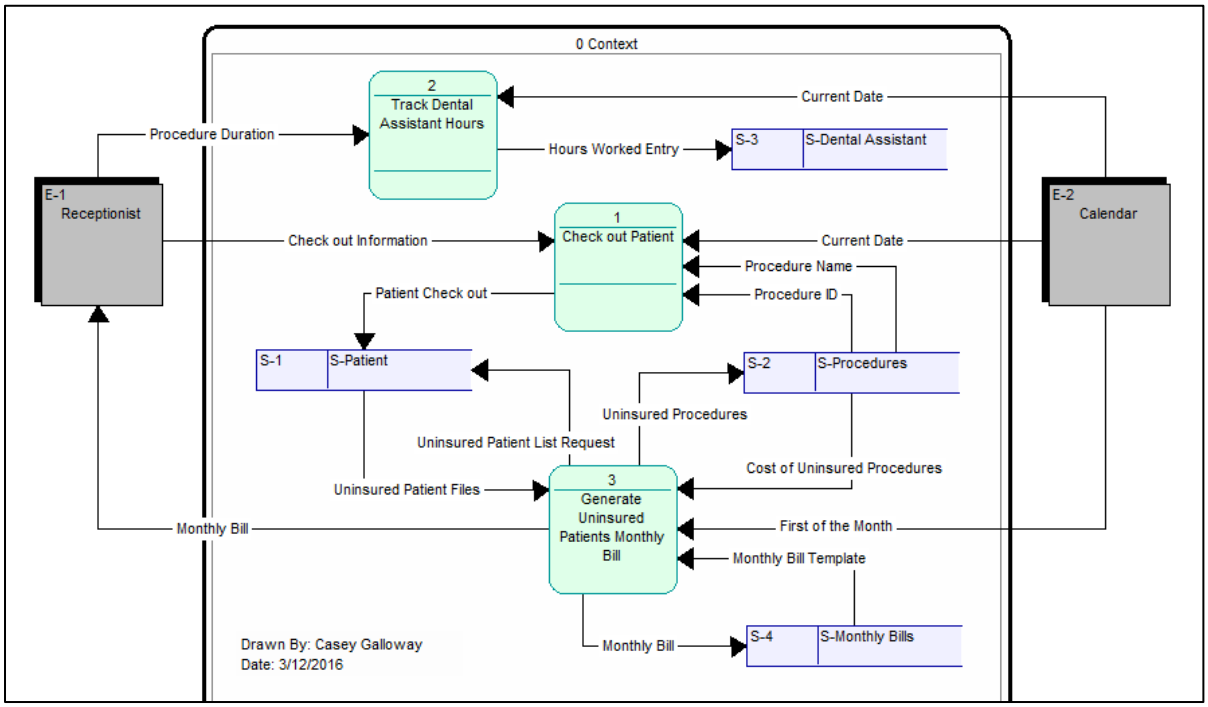

Figure 1: Billing System Context Diagram

External Entities

Figure 1 shows that the only external entities interacting with the system are the receptionist and calendar. The receptionist is the dentist's office employee and is the only actual person who interacts with the system. The patient and the dental assistant interact with the receptionist outside the system who then inputs the necessary information. Calendar is part of the PC's operating system meaning that

it is not actually within the billing system being designed. The billing system is just receiving dates from the calendar.

Level 0 Processes

The level 0 processes correspond to the three main objectives briefly addressed in the introduction. Check out Patient (Process 1) relates to the first main objective. When a procedure name or procedure ID is entered during check out the system must be able to correctly recognize the corresponding procedure name or ID for whichever one was entered. Similarly, the system must be able to correctly recognize the corresponding patient name or ID to match whichever one was entered. With all the check out information entered by the receptionist this process creates a patient check out file and stores it in the Patient data store. The check out information being entered consists of the patient name or patient ID, the time in, the time out, the procedure name or procedure ID performed, and the date of service which is received from the calendar. As stated earlier, a patient may have more than one procedure performed during a visit, a fact that should be taken into consideration while the check out screen is being designed.

Track Dental Assistant Hours (Process 2) relates to the second main objective. The dental assistant reports the start and end times of each appointment to the receptionist who then inputs the reported times into the system. These times are stored together along with the date in the Dental Assistant data store. The start and end times of each appointment are recorded because as previously mentioned the dental assistant gets paid by the hour, so these times are used at a later point to calculate the exact number of hours the dental assistant worked. It is important to note that this specific system does not calculate the actual number of hours, it only records the times.

Generate Uninsured Patients Monthly Bill (Process 3) relates to the third main objective. On the first of every month, the system will automatically filter through the patient files in the Patient data store to find the patient files with no dental insurance and pull the files that had procedures performed in the previous month. The costs of the procedures performed for each of the retrieved files is gotten from

the Procedures data store. A layout template for the monthly bill is retrieved from the Monthly Bills data store and is basically a form that the necessary information can just be filled in to the correct location on the bill. The monthly bills include the patient name, the patient address, the procedures performed, the cost of each procedure, the date of service, the service provider, and the total bill. The service provider is simply the dentist's office and is already included in the template. The monthly bills are then printed and received by the receptionist who mails them out to the uninsured patients. A digital copy of all the monthly bills generated is also stored back into the Monthly Bills data store.

It is also important to note that the addition and/or removal of patient files was not discussed in the letter received from Ms. Walker. Therefore that type of system capability was left out of this report and was not incorporated into the system planning.

Data Stores

The Patient data store (S-1) hosts all data related to the actual patients. All of the patient files are stored there, and the patient files consist of the general patient information and the patient check out created at the end of each visit. One of the most important things included in the patient information is their dental insurance status which is used by Process 3. It also includes patient name, patient ID, and patient address.

The Procedures data store (S-2) hosts all data related to the procedures offered at the dentist's office. It is essentially a list of all the procedure names and their corresponding procedure IDs. This data store also consists of the specific cost associated with each of the procedures.

The Dental Assistant data store (S-3) only interacts with Process 2. This data store simply stores the recorded start and end times of each appointment worked by the dental assistant and the corresponding date. As mentioned earlier, this data is used at some later point to calculate the exact number of hours worked by the dental assistant.

Monthly Bills data store (S-4) only interacts with Process 3. It stores and sends the monthly bill template that is used on every first of the month. This data store also stores copies of all of the monthly bills ever generated for uninsured patients. Even though storing copies of the bills was not explicitly specified, it seemed liked a reasonable addition in case a bill ever had to be reprinted.

Use Cases and Short Descriptions

Use cases are based off of the level 0 processes that can be found in Figure 1. Below is a list of the use cases involved in this system. The name of each use case is followed by the actors that interact with that particular use case and a short description which briefly describes what each use case does.

1. Check out Patient
    a. Actors: Receptionist, Calendar
    b. Short Description: This allows the receptionist to check out a patient at the end of their visit by entering the check out information.
2. Track Dental Assistant Hours
    a. Actors: Receptionist, Calendar
    b. Short Description: This tracks and records start and end times of procedures worked by the dental assistant and the service date. The actual number of hours is not calculated within this use case, but later by the receptionist.
3. Generate Uninsured Patients Monthly Bill
    a. Actors: Calendar, Receptionist
    b. Short Description: This filters out patients with no dental insurance and complies a monthly bill that is to be printed and mailed to the uninsured patients.

Test scenarios were developed in order to test each use case. In each scenario there are multiple test cases that address some possibilities of what may be entered or how information may be entered into the system. The scenarios also contained what the expected result or outcome is supposed to look like and leaves space for

comments of the actual results after the test cases are executed. These test scenarios can be found in Appendix C.

Function Point Analysis

Function points are an interval unit measurement for software, and the quantification of information processing functionality enables a development team to get an idea of the size and complexity of an application may be. They are independent of programming languages and can be figured out based on the context diagram. There are three function types: development, enhancement, and application. This billing system is development because it is a new project that is not maintaining an application and is not being built on top of an already existing application.

Function Types

There are five function types used in a function point analysis. Data functions, or data at rest, are logical data that can be queried and updated and account for two of the function types. The two data function types are internal logical files (ILFs) and external interface files (EIFs). The other three types are transactional functions which set the data in motion and they are external inputs (EI), external outputs (EO), and external inquiries (EQ).

Internal Logical Files (ILFs)

Internal logical files can be defined as a user identifiable group of logically related data that resides entirely within the application boundary. The main point being the ILFs are maintained within the application boundary.

External Interface Files (EIFs)

External interface files are similar to ILFs except they are user identifiable groups of logically related data used for reference purposes only with data that resides entirely outside of the application boundary. EIFs are essentially ILFs within another system, so they are maintained by an external application but are available to the application being counted.

External Inputs (EI)

External inputs is a transactional function meaning it sets data in motion. EI is the elementary process where data crosses the application boundary coming from the outside going inside the boundary.

External Outputs (EO)

External outputs are the opposite of EIs, they are an elementary process where derived data passes across the boundary starting from the inside and going to the outside. Basically data that is exiting the application boundary.

External Inquiries (EQ)

External inquiries can be defined as an elementary process with a combination of input and output components that result in data retrieval from one or more ILFs and EIFs. The EQ inputs do not update any internal or external files, and the EQ outputs do not calculate or derive any data.

Complexity of Function Types

Function point analysis also assess the complexity of each of these function types, and complexity is rating low, average, or high. For data functions their complexity is based on their number of record type elements (RETs) and their number of data element types (DETs). RETs are recognizable subgroups of data elements contained within a data function. DETs are unique, non-recursive fields recognized by the user. The complexity of transactional functions is based on their number of DETs and their number of file types referenced (FTR) which is simply the number of ILFs and EIFs referenced by a transactional function.

Overall System Function Point Analysis

The function point analysis for this system was calculated based on the system as a whole. This function point analysis was calculated by hand without the use of COCOMO. Based on the Figure 1 context diagram the various function types present in this system were determined along with their complexity. It was determined that there are four ILFs, which are the four data stores occurring within the system, and there are no EIFs present in this system. It was found that all four

ILFs have low complexity ratings based on the ILF/EIF complexity matrix. The Patient data store consists of four RETs and thirteen DETs; the Procedures data store consists of no RETs and three DETs; the Dental Assistant data store consists of one RET and three DETs; and the Monthly Bills data store consists of two RETs and seven DETs.

There are a total of five EIs associated with this system. The data flows crossing the application boundary coming into the system are: procedure duration, current date twice, first of the month, and check out information. Using the data dictionary and the context diagram it was determined that four of these EIs have low complexity ratings and the fifth has an average complexity rating. Procedure duration references no FTRs and has two DETs, both current date EIs reference one FTR and contain one DET, and first of the month also references one FTR and contains one DET, therefore these four EIs were rated low complexity. Check out information was rated with average complexity because it references two FTRs and contains seven DETs.

There is only one EO determined to be associated with this system. It is the monthly bill data flow that crosses the application boundary and leaves the system to be received by the receptionist. This EO was rated as having average complexity because monthly bill references three FTRs and consists of seven DETs.

Two EQs were determined to be associated with this system. Both EQs are initiated by Process 3. The one EQ is the getting the uninsured patient files and the EQ is getting the costs of the uninsured procedures. Both EQs were determined to have low complexity ratings. Getting the uninsured patient files EQ references one FTR, the Patient data store, and has six DETs. Getting the cost of uninsured procedures performed EQ also reference one FTR, the Procedures data store, and has two DETs.

Figure 2 illustrates how the unadjusted function point count or UAF for this system was calculated. The figure shows how the degree of complexity of each function type is used to find the total UAF.

	Complexity			
	Low	Average	High	Total
ILF	4 x 7 = 28	0	0	28
EIF	0	0	0	0
EI	4 x 3 = 12	1 x 4 = 4	0	16
EO	0	1 x 5 = 5	0	5
EQ	2 x 3 = 6	0	0	6
				Total UAF = 55

Figure 3: Total UAF Calculation

Figure 3 shows how the total degree of influence or TDI was calculated. The ratings for each of the fourteen General System Characteristics is also displayed in Figure 3. The General System Characteristics are rated on a scale of 0 – 5 where 0 is not present or no influence and 5 is strong influence. These ratings added up is how the total TDI is calculated. The total TDI is then used in the equation displayed in Figure 4. This equation is calculating the value adjustment factor or VAF.

Data Communications	0
Distributed Data Processing	1
Performance	3
Heavily Used Configuration	1
Transaction Rate	3
Online Data Entry	5
End User Efficiency	3
Online Update	3
Complex Processing	3
Reusability	1
Installation Ease	0
Operational Ease	3
Multiple Sites	0
Facilitate Change	2
Total TDI = 28	

Figure 2: GSC Ratings and Total TDI Calculation

Value Added Adjustment Factor = (TDI x 0.01) + 0.65

VAF = (28 x 0.01) + 0.65

= 0.93

Figure 4: VAF Equation and Calculation

In order to calculate the final adjusted function point count the value calculated for the total UAF and the value calculated for the VAF are needed. The total adjusted function points or TAFP results in the final value needed to find the number of lines of code (LOC). The TAFP is converted into the equivalent LOC count by being multiplied by a pre-set language value. The language value for each programming language is different. Figure 5 displays the TAFP equation and the result for this system. Figure 6 is the conversion equation to determine the approximate number of LOC. In this equation the TAFP is multiplied by a certain language value. For this specific system the language it is being converted to is Microsoft Access and the average language value for Access is 35.

Total Adjusted Function Points = UAF x VAF

TAFP = 0.93 x 55

= 51.15

Figure 5: TAFP Equation and Calculation

Lines of Code = TAFP x Average value for Language

LOC = 51.15 x 35

= 1790.25

Figure 6: Converting Function Point to LOC Calculation

According to the calculations, this computerized billing system will approximately end up being 1790.25 LOC in Microsoft Access. This calculation is not a definite amount, but it can be used as a base number. There are more accurate methods of estimating the LOC but this was the method used for this system.

Use Case Scenarios

The use case scenarios elaborate and expand on the list of use cases and their short descriptions. The scenarios provide more detail and are more specific. Use case scenarios include the steps needed to complete each use case, and these steps are then used to decompose the level 0 process in Figure 1. The actual use case scenarios can be found in Appendix B of this report.

Classes

A class is a general template used to define and create specific instances or objects; and an object is a person, place, event, or thing that we want to get information from, basically it is a noun. Classes have operations or methods that enable them to interact with each other by passing messages to and from one another. This system has four classes associated with it. The four classes are: the Patient class, the Procedure class, the Monthly Bill class, and the Dental Assistant class. These classes are used in the class diagram, the CROC cards, and the sequence diagrams all of which are discussed later. The Patient class deals with any information that has to do with the patient specifically. The Procedure class deals with all the information and such specifically related to the procedures performed at the dentist's office. Monthly Bill class encompasses everything that has to do mainly with the monthly bills. The Dental Assistant class is everything related to the dental assistant and their hours.

Class Diagram

The class diagram visually displays how the classes interact with one another. It is a static model that shows the classes and relationships among classes that remain constant in the system over time. Class diagrams indicate the multiplicity of each interaction. Figure 7 is the class diagram for this particular billing system. Based on the figure there are five relationships between the classes. There is a Dental Assistant-Patient class relationship, a Dental Assistant-Procedure class relationship, a Patient-Procedure class relationship, a Patient-Monthly Bill class relationship, and a Procedure-Monthly Bill class relationship. The only classes that

do not interact with each other are the Dental Assistant class and the Monthly Bill class. The numbers seen on the relationship lines between the classes are the multiplicity of that particular class relationship.

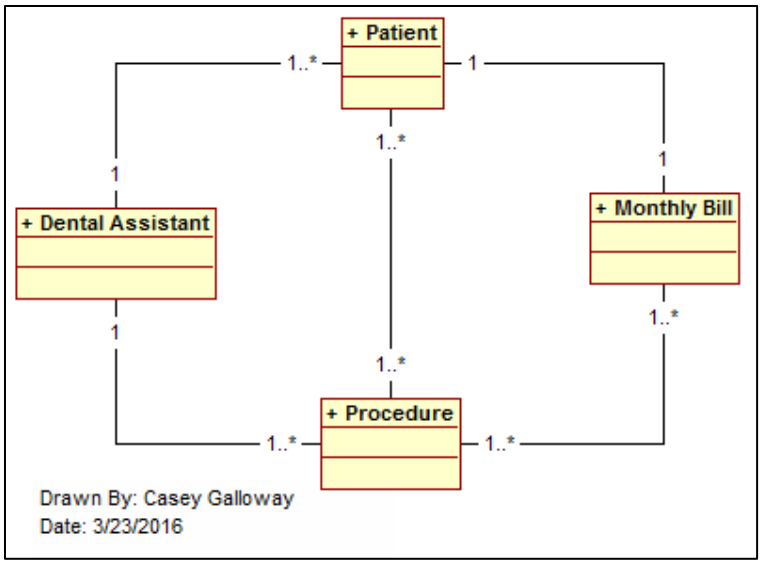

Figure 5: Billing System Class Diagram

Dental Assistant-Patient Class Relationship

This relationship is a one-to-many relationship. In this case, a one-to-many relationship means that the dental assistant can meet with one or more patients, but a patient can only meet with the one dental assistant. This is because there is only the one dental assistant at this dentist's office.

Dental Assistant-Procedure Class Relationship

This relationship is also a one-to-many relationship. This means that the dental assistant can perform one or more procedures, but a procedure can only be performed by the one dental assistant.

Patient-Procedure Class Relationship

This class relationship is a many-to-many relationship. In relation to this system this type of multiplicity means that a patient can receive one or more procedures and a procedure can be received by one or more patients.

Patient-Monthly Bill Class Relationship

These classes have a one-to-one relationship. This means that a patient only receives one monthly bill at a time and a particular monthly bill is only received by one patient. Based on prior knowledge given about the system it is assumed that this relationship only occurs with the patients that do not have dental insurance, otherwise there is no relationship between these classes.

Procedure-Monthly Bill Class Relationship

These classes have a many-to-many relationship. This means that one procedure can be a part of one or more monthly bills being generated and one monthly bill can contain one or more procedures.

CROC Cards

The CROC in CROC cards stands for Class, Responsibilities, Operations, and Collaborators. CROC cards basically define each class that appears on the class diagram. As implied by the acronym CROC cards list out the responsibilities of each class, the other classes that a class interacts with, and the messages that are passed between those interacting classes. Figures 8 – 11 are the CROC cards for each of the classes in this system.

Patient CROC Card

Figure 8 is the CROC card for the Patient class. According to the CROC card, the Patient class has three responsibilities, collaborates with two other classes along with also collaborating with itself, and has nine operations. The nine operations mean that there are nine messages either being sent to the Patient class or being sent from it.

Class: Patient		
Responsibilities	**Operations**	**Collaborators**
Keep a list of all patients and their information		
Keep Dental Insurance Status of all patients	sendUninsuredPatientFiles	Monthly Bill
Check patient out at end of visit	getProcedureName(ProcedureID) getProcedureID(ProcedureName) sendProcedureName sendProcedureID	Procedure
	getPatientName(PatientID) getPatientID(PatientName) sendPatientName sendPatientID	Patient

Figure 6: Patient CROC Card

The first responsibility of the Patient class is to keep a list or record of all patient files for the dentist's office. The patient files include the basic patient information along with any and all patient check outs that a patient might receive based on their visits. This responsibility does not collaborate with any of the other class nor does it have any operations.

The next responsibility is keeping a record of the dental insurance status of each patient. The dental insurance status is actually a part of the patient information, but it has such an important purpose on its own that it was written as a separate responsibility. With this responsibility the Patient class collaborates with the Monthly Bill class. The only operation associated with this interact is sendUninsuredPatientFiles which filters out the patient files of patients with no dental insurance and sends them to the Monthly Bill class.

The third responsibility is checking out the patient at the end of their visit. This responsibility has the remaining eight operations associated with it. With this responsibility the Patient class collaborates with itself and the Procedure class. The collaboration with the Procedure class relates to four of the eight operations. The four operations related to this collaboration are: getProcedureName(ProcedureID),

sendProcedureID, getProcedureID(ProcedureName), and sendProcedureName. These operations address the objective of the receptionist being able to enter the procedure name or procedure ID while checking out a patient and the system being able to match the correct corresponding name or ID. So, if the procedure ID is entered then that gets sent to the Procedure class and the Patient class receives the corresponding procedure name; and if the procedure name is entered then that gets sent to the Procedure class and the Patient class receives the corresponding procedure ID.

The last four operations are related to the Patient class's collaboration with itself. The remaining four operations are: getPatientName(PatientID), sendPatientID, getPatientID(PatientName), and sendPatientName. The purpose of these operations is similar to the prior four procedure related operations, except these operations address the receptionist entering the patient name or patient ID during check out and the system having to get the corresponding name or ID. If the receptionist enters the patient ID then the Patient class will search itself and return the corresponding name, and if the receptionist enters the patient name then the Patient class searches for the correct ID and returns that.

Procedure CROC Card

Figure 9 is the CROC card for the Procedure class. This class has four responsibilities, also collaborates with two other classes and itself, and contains five operations.

Class: Procedure		
Responsibilities	**Operations**	**Collaborators**
Keep a list of procedure names and corresponding procedure IDs		
Keep a list of costs for each procedure		
Fill check out patient information with Procedure Name or Procedure ID	getProcedureName(ProcedureID) getProcedureID(ProcedureName) sendProcedureName sendProcedureID	Patient, Procedure
Inform monthly bill of cost of all uninsured procedures	sendCostOfUninsuredProcedures	Monthly Bill

Figure 7: Procedure CROC Card

The first responsibility is keep a record or list of all the procedures that can be performed at this dentist's office. This list just matches the procedure names to the corresponding procedure IDs. There are no collaborators or operations associated with this responsibility.

The next responsibility is keep track and a record of all the costs for the different procedures offered. Each procedure has a specific cost that is charged to the patient. This list matches the correct cost with the procedure name or ID. This responsibility also has no collaborators or operations associated with it.

The third responsibility pairs with the check out responsibility from the Patient CROC card. It is just returning the correct procedure name or procedure ID to the Patient class. The two get operations, getProcedureName(ProcedureID) and getProcedureID(ProcedureName), relate to the Procedure class's collaboration with itself. The two send operations, sendProcedureName and sendProcedureID, relate to the collaboration with the Patient class. These operations mirror the matching ones described in the Patient CROC Card section.

The final responsibility is informing or basically just sending the costs of the procedures that were performed on uninsured patients to the Monthly Bill class.

There is just the one send operation, sendCostOfUninsuredProcedures, associated with this responsibility. This operation just finds the correct cost for the procedure and sends it to be used by the Monthly Bill class.

Monthly Bill CROC Card

Figure 10 is the CROC card for the Monthly Bill class. This class has two responsibilities, collaborates with two other classes along with also collaborating with itself, and it has six operations associated with it.

Class: Monthly Bill		
Responsibilities	**Operations**	**Collaborators**
Keep record of all monthly bills generated		
Create monthly bills for uninsured patients	getMonthlyBillTemplate createMonthlyBill sendMonthlyBill getUninsuredPatientFiles(UninsuredPatientName, UninsuredPatientAddress,UninsuredProcedures) getCostOfUninsuredProcedures (UninsuredProcedures)	Monthly Bill Patient Procedure

Figure 8: Monthly Bill CROC Card

The first responsibility of this class is to keep a record of every monthly bill that has been generated. These bills are created on the first of every month for all of the patients without dental insurance that had one or multiple procedures performed during the prior month. Once the monthly bills are generated one copy gets printed and received by the receptionist and another digital copy is stored in the Monthly Bills data store. This responsibility is addressing the digital copies being stored in the data store. There are no collaborators or operations associated with this responsibility.

The second responsibility is actually creating those monthly bills that are being stored by the first responsibility. With this responsibility the Monthly Bill class has to collaborate with itself in order to get the template to create the bills. It also has to collaborate with itself to create the bills and the send them once they are created.

The related operations for this collaboration is getMonthlyBillTemplate, createMonthlyBill, and sendMonthlyBill. This responsibility also requires the Monthly Bill class to collaborate with the Patient class in order to get the files of all the uninsured patients. The only operation related to this collaboration is getUninsuredPatientFiles(UninsuredPatientName,UninsuredPatientAddress, UninsuredProcedures). This responsibility also has the Monthly Bill class collaborating with the Procedure class in order to get the costs of all the procedures that were performed on uninsured patients indicated in the uninsured patient files that the Monthly Bill class got from the Patient class. The only operation associated with this collaboration is getCostOfUninsuredProcedures(UninsuredProcedures).

Dental Assistant CROC Card

Figure 11 is the CROC card for the Dental Assistant class. The Dental Assistant class only has one responsibility, collaborates with itself, and therefore only has one operation associated with it.

Class: Dental Assistant		
Responsibilities	**Operations**	**Collaborators**
Track dental assistant hours	getHoursWorkedEntry (ProcedureDuration,CurrentDate)	Dental Assistant

Figure 9: Dental Assistant CROC Card

The Dental Assistant class's only responsibility is to track the hours that the dental assistant works, rather it is keeping a record of the procedure duration times reported by the dental assistant to the receptionist. This class only collaborates with itself because it has to make the hours worked entry that is stored in the Dental Assistant data store. It receives the procedure duration times from the receptionist and the current date from the calendar. These two variables get used by the Dental Assistant class to create the actual entry to be stored with the operation getHoursWorkedEntry(ProcedureDuration,CurrentDate).

Sequence Diagrams

Sequence diagrams illustrate which classes are participating in a particular use case along with the messages getting passed back and forth between the classes. The messages being passed in-between the classes are information being sent in order to trigger a method in another class. There is generally one sequence diagram for every use case, thus there are three sequence diagrams displayed in this report.

Check out Patient Sequence Diagram

Figure 12 is the sequence diagram for the Check out Patient use case. As seen from the diagram only three of the four classes in this system participate with this use case. They are: the Patient class, the Procedure class, and the Monthly Bill class. The sequence starts with the receptionist entering the check out information and the billing system getting the current date from the calendar. Next the system finds the corresponding patient name if a patient ID is entered, or the other way around and the system gets the corresponding patient ID. Then the same process happens but this time with the procedure name and procedure ID. Once a month the stored patient check out files for all the patients without dental insurance are sent to Monthly Bill to create printable bills for those patients.

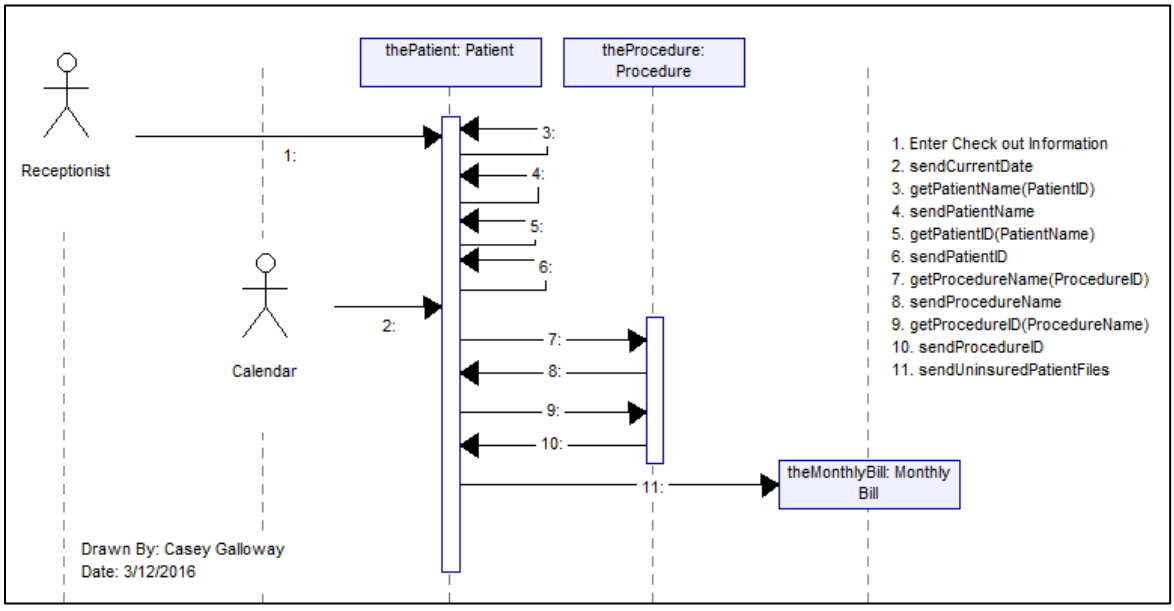

Figure 10: Check out Patient Sequence Diagram

Track Dental Assistant Hours Sequence Diagram

Figure 13 is the sequence diagram for the Track Dental Assistant Hours use case. According to the diagram, this use case only interacts with the Dental Assistant class. The sequence starts with the receptionist entering the procedure duration times reported to her by the dental assistant. Then the calendar sends the current date to the class. The sequence ends by the Dental Assistant class creating the hours worked entry within itself by combining the entered procedure duration times and current date. The Dental Assistant class gets the hours worked entry and sends it back to itself.

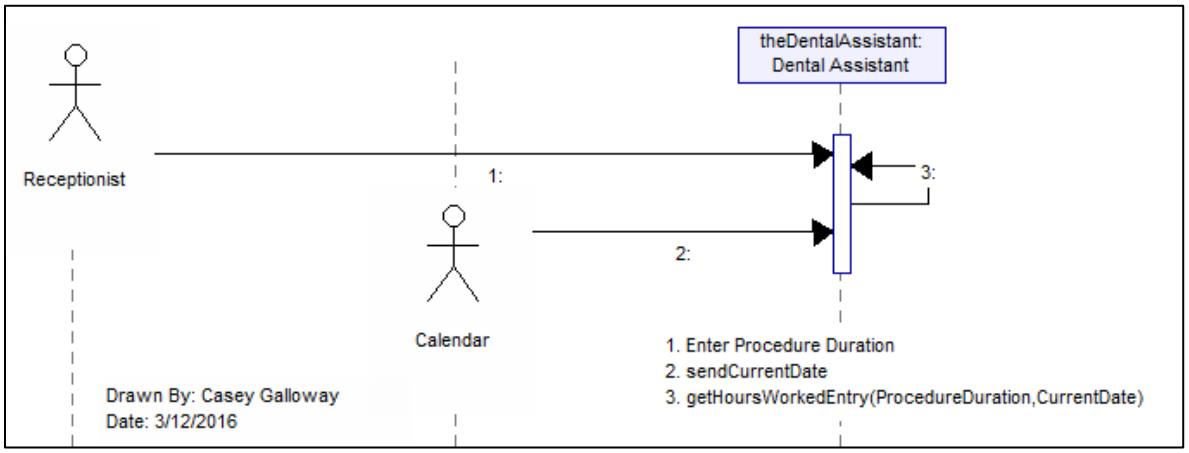

Figure 11: Track Dental Assistant Hours Sequence Diagram

Generate Uninsured Patients Monthly Bill Sequence Diagram

Figure 14 is the sequence diagram for the Generate Uninsured Patients Monthly Bill use case. Just like the Check out Patient sequence diagram, this diagram shows that the Monthly Bill class, Patient class, and Procedure class also participate with this use case. This sequence begins with the calendar sending a message to the Monthly Bill class that it is the first of the month and the bills for the uninsured patients have to be created. Then the Monthly Bill class gets the monthly bill template from itself. Next, the Monthly Bill class requests all of the patient files for the uninsured patients that had procedures performed in the previous month from the Patient class, and the Patient class retrieves those files and sends them back to the Monthly Bill class. Next, the Monthly Bill class gets the costs for each

of the procedures indicated in the uninsured patient files from the Procedures class. The sequence finishes up by actually creating the bills for those patients and then send those created bills to the receptionist by printing them out.

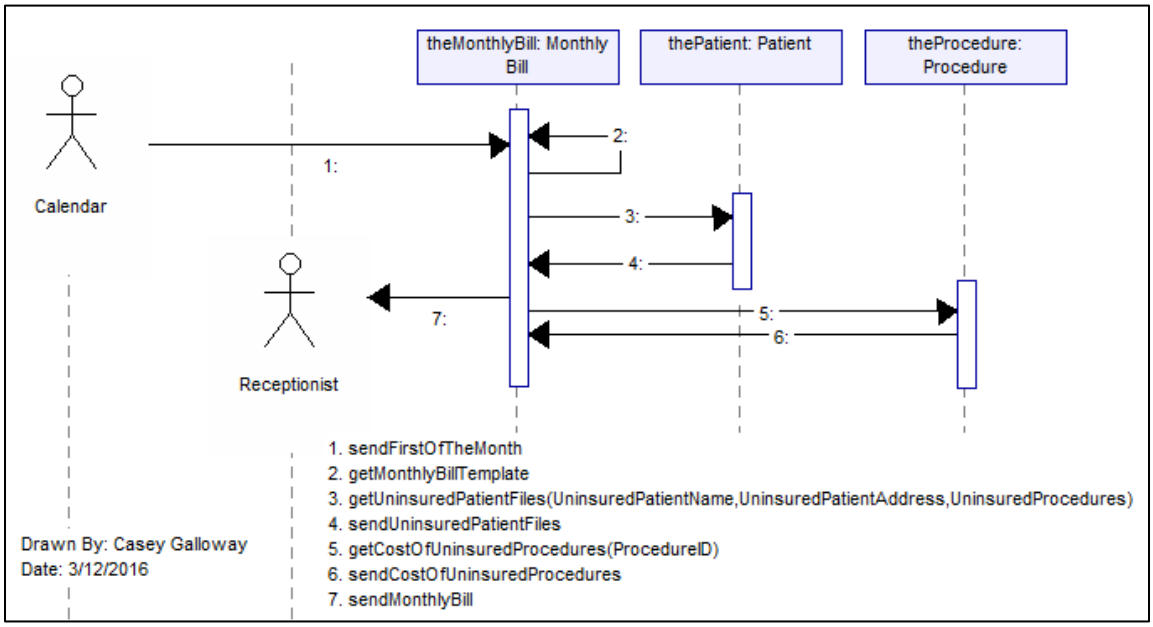

Figure 12: Generate Uninsured Patients Monthly Bill Sequence Diagram

Conclusion

There are three main tasks or objectives that this dentist's office would like this computerized billing system to be able to do. First, the receptionist wants the capability of entering the procedure name or procedure ID and the patient name or patient ID into the system while checking out a patient and having the system be able to the corresponding name or ID. When this objective is addressed during the design phase it is highly recommended that the design team takes into consideration the fact that a patient can have multiple procedures performed within one visit which would need to be recorded during check out. Second, the system must be able to track and keep record of the start and end times of each procedure worked by the dental assistant. Third, the system will have to generate bills once a month with the total cost owed by patients without dental insurance who had procedures performed. This requirements report includes the system's context diagram showing the level 0 processes, data stores, data flows, and external entities; a list of

the three use cases with short descriptions of each; the function point analysis of the system as a whole; the individual use case scenarios; a list of the classes and the class diagram showing how they relate to each other; the CROC cards for each class; the sequence diagrams for each use case; and finally the test scenarios to test each use case.

Appendix A

Data Dictionary

Data Dictionary

The data dictionary lists out the elements and attributes within a system. The data included within each data flow can clearly be seen with this data dictionary.

- Patient File
  - Patient Information
    - Patient Name
    - Patient ID
    - Patient Address
    - Dental Insurance Status
      - Insured
      - Uninsured
  - Patient Check out
    - Check out Information
      - Patient Name
      - Patient ID
      - Procedure Name
      - Procedure ID
      - Time in
      - Time out
      - Current Date
  - Uninsured Patient List Request
- Hours Worked Entry
  - Current Date
  - Procedure Duration
    - Procedure Start Time
    - Procedure End Time
- Monthly Bill
  - Uninsured Patient File
    - Uninsured Patient Name
    - Uninsured Patient Address
    - Uninsured Procedures
    - Cost of Uninsured Procedures
    - Date of Service
  - Total Bill
  - Monthly Bill Template
    - Service Provider

Appendix B

Use Case Scenarios

Use Case Scenarios

The use case scenarios elaborate and expand on the list of use cases and their short descriptions. The scenarios provide more detail and are more specific. Use case scenarios include the steps needed to complete each use case, and these steps are then used to decompose the level 0 process in the Figure 1 context diagram within the main report.

Check out Patient Use Case Scenario

Use Case Name: Check out Patient	ID Number: 1
Short Description: This allows the receptionist to check out a patient at the end of their visit by entering visit information.	
Trigger: Receptionist enters check out information after procedure is completed Type: External / Temporal	

Major Inputs:		Major Outputs:	
Description	Source	Description	Destination
Check out Information	Receptionist	Patient Check out	Patient data store
Current Date	Calendar		
Procedure ID	Procedures data store		
Procedure Name			

Major Steps Performed	Information for Steps
1. The receptionist enters the check out information into the system.	Check out Information, Patient Information
2. If Patient Name is entered with check out information, then the system gets the corresponding Patient ID. If Patient ID is entered with check out information, then the system gets the corresponding Patient Name.	Patient Name, Patient ID
3. If Procedure Name is entered with check out information, then the system gets the corresponding Procedure ID. If Procedure ID is entered with check out information, then the system gets the corresponding Procedure Name.	Procedure Name, Procedure ID
4. Get current date from Calendar to use as service date.	Current Date
5. Generate Patient Check out file and store in Patient data store.	Check out Information, Patient Information, Current Date

Track Dental Assistant Hours Use Case Scenario

Use Case Name: Track Dental Assistant Hours	ID Number: 2
Short Description: This tracks and records start and end times of procedures worked by the dental assistant and the service date. The actual number of hours is not calculated within this use case, but later by the receptionist.	
Trigger: End of the appointment and receptionist enters procedure duration into system. Type: External / Temporal	

Major Inputs:		Major Outputs:	
Description	Source	Description	Destination
Procedure Duration	Receptionist	Hours Worked Entry	Dental Assistant data store
Current Date	Calendar		

Major Steps Performed	Information for Steps
1. The receptionist begins process to track dental assistant hours by clicking the 'Track Dental Assistant Hours' button on the interface.	
2. The receptionist enters Procedure Duration times (start time and end time) as reported by the dental assistant.	Procedure Duration
3. Get current date from Calendar as service date.	Current Date
4. Create Hours Worked Entry.	Procedure Duration, Current Date
5. Store Hours Worked Entry to Dental Assistant data store.	Hours Worked Entry

Generate Uninsured Patients Monthly Bill Use Case Scenario

Use Case Name: Generate Uninsured Patients Monthly Bill	ID Number: 3

Short Description:

This filters out patients with no dental insurance and compiles a monthly bill that is to be printed and mailed to the uninsured patients.

Trigger: The first of the month

Type: External / Temporal

Major Inputs:

Description	Source
First of the Month	Calendar
Uninsured Patient Files	Patient data store
Cost of Uninsured Procedures	Procedures data store
Monthly Bill Template	Monthly Bill data store

Major Outputs:

Description	Destination
Uninsured Patient List Request	Patient data store
Uninsured Procedures	Procedures data store
Monthly Bill	Receptionist
	Monthly Bills data store

Major Steps Performed	Information for Steps
1. Search Patient data store for patients with uninsured dental insurance status.	Uninsured Patient List Request, Uninsured Dental Insurance Status
2. Get list of patient files with uninsured dental insurance status.	Uninsured Patient Files
3. Get Uninsured Procedure(s) from each of the Uninsured Patient Files.	Uninsured Procedures
4. Search Procedures data store for corresponding costs of the Uninsured Procedures.	Uninsured Procedures, Cost of Uninsured Procedures
5. Calculate cost of Total Bill.	Cost of Uninsured Procedures
6. Get Monthly Bill Template from Monthly Bill data store.	Monthly Bill Template
7. Compile Monthly Bill for each uninsured patient.	Monthly Bill Template, Total Bill, Uninsured Patient Files, Uninsured Procedures, Cost of Uninsured Procedures
8. Print and save copy of Monthly Bill.	Monthly Bill

Appendix C

Test Plans for Each Use Case

Test Plans for Each Use Case

The following test plans correspond to each of the use cases used earlier, respectively they follow the same order of the use cases. Each test plan consist of multiple test cases and are only partially completed due to not having run the actual tests yet. The test cases within each test plan consist of the general inputs of that particular use case and its main outputs along with some fake testing information. As mentioned before each test case still needs to be run and then the test plans can be completed.

Check out Patient Test Plan

Test Plan

Program ID: Check out Patient **Version Number: 1**

Tester: Tyler Wood & Peter Carew **Date Designed :** 3/10/2016 **Date Conducted : _____**

Results: **Passed** **Open Items**

Test ID: 1 **Requirement Addressed:** Patient list accessibility

Objective:

To make sure the patient file is filled and updated when patient is checked out

Test Cases

Test Case	Patient Name	Patient ID	Patient Address	Dental Insurance Status	Date	Procedure Name	Procedure ID
1		00078	123 Main St	Uninsured	2/2/16	Root Canal	
2	Bilbo Bagg	00632	44 Elk St		2/4/16	Cavity Filling	27
3			16 Street St	Insured	2/11/16	Cleaning	1
4	Tom Skeret		21 West St	Insured	2/24/16		23, 1

Script

Receptionist enters check out data

Receptionist receives Patient Name or Patient ID from Patient

Receptionist received Procedure Name or Procedure ID from Procedure

Receptionist receives date from calendar

Patient file gets updated with information typed into interface by receptionist

Expected Results/Notes

1. John Doe, 00078, 123 Main St, Uninsured, 2/2/16, Root Canal, 7
2. Error: Empty Field. No Dental Insurance Status.
3. Error: Empty Field. Unable to locate patient file.
4. Tom Skeret, 01200, 21 West St, Insured, 2/24/16, XRay & Cleaning, 23 & 1

Actual Results/Notes

Track Dental Assistant Hours Test Plan

Test Plan

Program ID: Track Dental Assistant Hours **Version Number: 1**

Tester: Kelly Wetherbee & Colton Howard **Date Designed :** 3/10/2016 **Date Conducted : _____**

Results: ☐**Passed** **Op**☐**n Items**

Test ID: 2 **Requirement Addressed:** Storing dental assistant hours

Objective:

Track the date, start and end times of the dental assistant hours worked

Test Cases

Test Case	Date	Start Time	End Time
1	3/10/16	8:00 am	10:00 am
2	3/10/16	1:00 pm	2:00 pm
3	3/11/16	10:00 am	12:00 pm
4	3/12/16	1:30 pm	2:00 pm

Script

Receive date from calendar

Receive procedure duration times (including start time and end time) from receptionist

Generate hours worked entry

Send hours worked entry to Dental Assistant File

Expected Results/Notes

The Dental Assistant file contains that the dental assistant worked on 3/10/16 from 8:00 am to 10:00 am and 1:00 pm to 2:00 pm. The dental assistant also worked from 10:00 am to 12:00 pm on 3/11/16 and 1:30 pm to 2:00 pm on 3/12/16.

Actual Results/Notes

Generate Uninsured Patients Monthly Bill Test Plan

Test Plan

Program ID: Generate Uninsured Patients Monthly Bill **Version Number: 1**

Tester: Casey Galloway & Glen Fridman **Date Designed :** 3/10/2016 **Date Conducted : ____**

Results: ☐**Passed** **Op☐n Items**

Test ID: 3 **Requirement Addressed:** Generate Uninsured Monthly Bills

Objective:

To ensure that monthly bills are correctly being created for only the uninsured patients

Test Cases

Test Case	Patient Name	Patient Address	Procedures	Cost of Procedures	Dental Insurance Status
1	Jane Doe	123 Fourth St	1,2	400,500	Uninsured
2	John Smith	321 Street St	7	700	Insured
3	Bob Henry	456 Something St	4,5	1000,300	Insured
4	Jim James	951 South Ave	3	800	Uninsured

Script

Receive uninsured patient files from patient data store

Receive procedure costs from procedures file

Calculate total bill for uninsured procedures

Generate monthly bills with monthly bill template

Print and save monthly bills

Expected Results/Notes

1. Jane Doe, 123 Fourth St, Procedure 1 - $400, Procedure 2 - $500, Total Bill - $900
2. No bill is saved or printed
3. No bill is saved or printed
4. Jim James, 951 South Ave, Procedure 3 - $800, Total Bill - $800

The system inputs necessary information into the monthly bill template. The monthly bill template includes the service provider which is the dentist office.

Actual Results/Notes