Chapter 8 – Approaches to System Development

Solutions to End-of-Chapter Problems

Review Questions

1. What is a project?

As defined in Chapter 1, a project is a planned undertaking, with a beginning and end, that produces a well-defined result or product.

2. What is the range of sizes of an information system development project?

Some system development projects are very large, requiring thousands of hours of work by many people and spanning several calendar years. Many smaller projects still require several months to deploy.

3. What is the system development life cycle (SDLC)?

SDLC is the term used to describe the process and methodology for developing, deploying, and maintaining a software system. A particular SDLC will include a description of the methods, the tools, techniques, models, and project management processes that are used in the development of a new system. It is called a life cycle because it can cover the entire life of a system from initial idea to deployment to ongoing maintenance.

4. What characteristics of a project call for a predictive approach to the SDLC?

Predictive: Projects are well understood—technology is well known; user requirements are well known; development methodology is well known; project team is experienced and familiar with the system; and there are few known risks.

5. What characteristics of a project call for an adaptive approach to the SDLC?

Adaptive: Projects are not well understood—technology is new or unfamiliar; requirements are not very clear; team is not experienced with the type of system or methodology.

6. What are the seven phases of the traditional predictive SDLC?

1. Project initiation
2. Project planning
3. Analysis
4. Design
5. Implementation (programming)
6. Deployment
7. Support

7. What are the objectives of the support phase?

The objective of the support activities is to keep the system running productively during the years following its initial deployment. Three major activities occur during support:

- Maintaining the system
- Enhancing the system
- Supporting the users

8. Explain how the waterfall model of the SDLC controls the changes that occur during a project.

The pure waterfall attempts to specify completely the requirements during the analysis phase, and then tries to "freeze" those requirements so that things do not change. Another solution is to have a modified waterfall so that additional analysis (i.e. specifying new requirements) can be done as the project progresses. However, neither situation is well suited to handle big changes during the life of a project.

9. Explain the advantages of having the phases of a predictive SDLC overlap.

As indicated in Question 8, having overlap allows the project to handle ongoing analysis and requirements definition even after design and programming have begun. Another advantage is that the development team can work more efficiently by doing design and even programming on those requirements and specifications that are being worked on (i.e. in analysis).

10. What organizing concept is included in all adaptive SDLCs?

Adaptive SDLCs are all oriented to being able to handle changing requirements during the life of the project. All include iterations as the method to be able to integrate newly discovered requirements into the project work.

11. What is considered the first adaptive SDLC? Sketch it.

The Spiral model was one of the very first adaptive SDLCs. (Sketch see Figure 8-5)

12. For an adaptive SDLC, explain what goes on during each iteration.

Each iteration is like a mini-project. Withing each iteration, the activities related to the phases of the predictive SDLC occur, e.g. planning, analysis, design, implementation, and deployment.

13. The SDLC used in this text is based on what adaptive SDLC?

The SDLC used in the this text is a simplification of and variation on the Unified Process (UP), which is a well known and well accepted formal iterative approach.

14. What are the core processes in the SDLC used in this book, and what traditional predictive SDLC phase corresponds to each process?

1. Identify the problem and obtain approval – Initiation Phase
2. Plan and monitor the project – Planning Phase
3. Discover and understand details – Analysis Phase
4. Design system components – Design Phase
5. Build, test, and integrate system components – Implementation Phase
6. Complete system tests and deploy the solution – Deployment Phase

15. What is the iterative approach that involves completing and deploying part of an application over a few iterations and then completing and deploying another part of that application after a few more iterations?

This is usually referred to as incremental development. A variation of incremental development that builds the entire structure at first is called a walking skeleton.

16. Why do adaptive SDLCs not explicitly include the support phase?

Support does not fit into an "iteration" project model. Hence, the Support Phase is either treated as separate follow-on projects, or it reverts back to the predictive model and becomes a Support Phase.

17. What are the three activities of the support phase?

Three major activities occur during support:

- Maintaining the system
- Enhancing the system
- Supporting the users

18. What is a popular support technique used to answer users' questions and help them increase productivity?

One way to support users is to provide a "help desk" type of capability. A help desk is a standard place, with procedures and staff, to provide both technical and application support for users.

19. What is a system development methodology?

A methodology is more than just a method. It is an entire set of guidelines and procedures to develop a new system. Included in the components of a methodology would be what modeling approach to use, what tools, what techniques to do analysis and design and programming. A methodology provides the guidelines for the entire development process.

20. What are some examples of models included in a methodology?

In a systems development methodology there would be graphical models, such as class diagrams and use case diagrams. There would also be descriptive models such as use case descriptions. There are also models associated with the management of the project such as Gantt chart or NPV spreadsheet. (See Fig 8-9)

21. What are some examples of techniques included in a methodology?

Techniques have to do with completing specific tasks or activities. Examples might be how to model use cases, or how to develop a class diagram. User-interface design uses techniques such as story boards. Other techniques might be how to interview users etc. (See Fig 8-11)

22. What are some examples of tools included in a methodology?

Tools might include drawing tools, such as Microsoft Visio, or an Integrated Development Tool (IDE), such as Struts, Eclipse, or Netbeans. Testing tools are also often used for program tests. (See Fig 8-10)

23. What are the two approaches to software construction and modeling?

The two primary approaches to software construction are the Structured Approach (sometimes called the Traditional approach), and the Object-oriented Approach.

24. What are the basic characteristics of the traditional approach?

The traditional approach is Structured development. Its basic approach was to break a problem into smaller components and write programs a set of modules in a hierarchical structure. By dividing a solution into these smaller modules, developers could focus on one thing at a time, which was easier to write programs and to maintain systems.

25. What are the basic characteristics of the object-oriented approach?

All processes and data can be thought of as "objects." A program consists of interacting objects that work together by sending messages to each other and executing internal functions (methods).

26. What are the three main structured techniques?

The structured approach includes:

- Structured programming – one entry, one exit point; sequence, iteration, decision
- Structured analysis – DFDs, flowcharts; flows of data across processes and agents
- Structured design – Structure charts; boss modules call submodules

27. What are three diagrams created by the structured approach?

The three primary diagrams are the:

- Data flow diagram
- Structure chart
- Entity Relationship diagram (ER)

28. What are the main object-oriented techniques?

Object oriented analysis – OOA
Object oriented design – OOD
Object oriented programming – OOP

29. What is Agile development?

Agile development is a philosophy and set of guidelines for developing information systems in an unknown, rapidly changing environment, and consists of a statement of core philosophy and a set of values or guidelines for how to build models.

30. What are the four "values" reflected in Agile development?

The four basic values of the core philosophy are:

- Value responding to change over following a plan
- Value individuals and interactions over processes and tools
- Value working software over comprehensive documentation
- Value customer collaboration over contract negotiation

31. What is Agile modeling (AM)?

Agile modeling is a guiding philosophy about modeling. Only models that are necessary to move the project forward and that are really needed for the development of the software.

32. What are the 11 Agile modeling principles?

The 11 Agile modeling principles are:

1. Develop software as your primary goal.
2. Enable the next effort as your secondary goal.
3. Minimize your modeling activity—few and simple.
4. Embrace change, and change incrementally.
5. Model with a purpose.
6. Build multiple models.
7. Build high-quality models and get feedback rapidly.
8. Focus on content rather than representation.
9. Learn from each other with open communication.
10. Know your models and how to use them.
11. Adapt to specific project needs.

Problems and Exercises

1. Write a one-page paper that distinguishes among the fundamental purposes of the analysis phase, the design phase, and the implementation phase of the traditional predictive SDLC.

Answers will vary. For analysis, the primary focus is on discovery and understanding. The purpose is to understand at a detailed level the needs of the users. This can be based on existing systems and procedures or based purely on the desired business processes. The focus should not be on the solution system but instead on understanding the business need.

For design, the focus changes from understanding to thinking about a viable solution. The purpose of design is to synthesize or develop a solution. The two layers of design are the architecture layer and the detailed module layer. At the architecture layer, the purpose is to think of the entire solution system and how it is structured. At the detailed design layer, the purpose is to think about the individual modules or classes and define the details of the executable code.

For implementation, the focus is on building a robust system. The activities of this phase are oriented towards taking the design and building a high quality working system. Also included in this phase are the support or ancillary activities needed to train users, convert data, and fine-tune the working environment.

2. Describe an information system project that might have three subsystems. Discuss how three iterations might be used for the project.

An iteration is a min-project in that it often takes a smaller component, or subsystem of a larger system, does analysis, design, and implementation within the iteration. So if there are three subsystems, AND if they are small enough, each subsystem might be done in a single iteration. (Larger subsystems might require multiple iterations each.)

Another approach might to verify that the more complex or risky parts of the system are handled within the first iteration. Some work should be done within each subsystem for each iteration because doing so reduces schedule dependencies.

3. Why might it make sense to teach analysis and design phases and activities sequentially, like a waterfall, even though iterations are, in practice, used in nearly all development projects?

Usually analysis and design and implementation are taught separately, as though in a waterfall project because the knowledge and skills for each is a separate topic. It is also valid to teach them in a waterfall fashion because within each iteration of an adaptive project a modified waterfall mini-project often occurs.

Also teaching analysis and design sequentially like a waterfall helps to organize the course and helps to explain the specific work that is done in each SDLC phase. Teaching an iteration approach first can confuse the work that is done in analysis and design.

4. List some of the models that architects create to show different aspects of a house they are designing. Explain why several models are needed.

Answers will vary. Blue prints show floor plan, front and rear elevation, roof framing plan, plumbing layout, electrical layout, foundation, and placement of house on lot. A scale 3D model shows what the house looks like. Mathematical calculations define stresses and forces affecting the roof or walls. Each model "abstracts" or highlights certain aspects of the house. No one model can contain everything needed. All models must fit together to define a house. Example models might include:

- Floor plan layouts
- Wiring diagrams
- Plumbing diagrams
- Detailed materials diagrams
- Detailed blueprint diagrams
- Landscaping diagrams

Different models are included to be able to explain and specify the various aspects of the home. Obviously the plumbing diagram does no good for the electrician.

5. What models might an automotive designer use to show different aspects of a car?

Answers will vary. Again, graphical models show what the car looks likes from different views, and there are many calculations and specifications for the different car components. Some look like the car, and some, like engineering models, are schematic diagrams or tables of specifications. Some models might include:

- Body frame specifications
- Electrical wiring diagram
- Fuel flow specification model
- Suspension specification model
- Braking system specification
- Body specifications
- Accessory option specifications
- Engine specifications
- Exhaust specifications

6. Sketch and write a description of the layout of your room at home. Are both the sketch and the written description considered models of your room? Which is more accurate? More detailed? Which would be easier to follow by someone unfamiliar with your room?

Answers will vary. Yes, both are models. They might be equally accurate and detailed, or one could be more accurate and/or detailed than the other. The sketch would be easier to follow for people, hence the use of graphical models in system development.

7. Describe a technique you use to help you complete the activity "Get to class on time." What are some of the tools you use with this technique?

Answers will vary. A technique is a strategy for completing an activity. So a strategy for getting to class on time might include setting the alarm, getting to bed early, checking the class schedule each night, checking the bus schedule over breakfast, and so on. The tools in this example are the alarm, the class schedule, and the bus schedule.

8. Describe a technique you use to make sure you get assignments done on time. What are some of the tools you use with this technique?

Answers will vary. The technique might include reading assignments right away, getting started early, making sure you have the materials needed ahead of time, checking with instructor as soon as you have a question about the assignment, using the appropriate computer tools to complete the assignment, making a backup copy of your work as you go, and getting to class on time the day the assignment is due. Tools might include the assignment, a calendar, e-mail, phone, computer tools used to create the assignment (word processor, spreadsheet), and online access to order materials needed.

9. What are some other techniques you use to help you complete activities in your life?

Answers will vary, probably widely. Career selection techniques, job search techniques, spouse selection techniques, house buying techniques, job survival techniques, health maintenance techniques, retirement planning techniques, and grief coping techniques.

10. There are at least two approaches to the SDLC, two approaches to software construction and modeling, and a long list of techniques and models. Discuss the following reasons for this diversity of approaches: The field is young; the technology changes quickly; different organizations have different needs; there are many types of systems; developers have widely different backgrounds.

Answers will vary. All of the listed reasons are valid in some ways. No one approach or technique is appropriate for all system projects. We need to be able to select the approach and the techniques that are right for the problem at hand.

The field is young: Even though it may not appear so to many young developers, information systems and software development is rather young. (Compared to other disciplines such as accounting or engineering or manufacturing.) In a young field there is always a lot of experimentation and change to discover better ways to do work.

The technology changes quickly: The technology in both hardware and software tools changes rapidly. Faster computers, mobile computing, new programming languages, and so forth, all enable new techniques and tools to develop systems. This rapidly changing technology forces systems developers to change to try to take advantage of these changes.

Different organizations have different needs: Some organizations have rapidly changing needs to meet a dynamic, changing market place. Other organizations, and within an organization, have a slower pace of work and of change. New systems may not be necessary and older systems, using older technology maybe adequate for the processing and workflow requirements.

There are many types of systems: Systems also are very different. From mobile systems, to business systems, to engineering systems, to scientific systems. Some systems have a high usage of detailed data and are very database sensitive. Other systems must be real time and respond rapidly to external inputs. Different specifications require different types of development techniques.

Developers have widely different backgrounds: Obviously developers use the tools that they are familiar with. So even though this does affect the tools and techniques used, developers should also be amenable to learning new and more powerful techniques and tools.

11. Go to the campus placement office to gather some information on companies that recruit information systems graduates. Try to find any information about the companies' approaches to developing systems. Is their SDLC described? Do any mention an IDE or a visual modeling tool? Visit the companies' Web sites to look for more information.

Answers will vary. Larger companies might post information on methodologies used as part of the recruiting information. Otherwise, companies sometimes tend to keep this information confidential. Students might suggest to recruiters that this information could help students understand the way development is done at the company.

12. Visit the Web sites of a few leading information systems consulting firms. Try to find information about their approaches to developing systems. Are their SDLCs described? Do the sites mention any tools, models, or techniques?

Answers will vary. If it is a consulting firm, they might publish a lot about their methodology (or even sell their methodology to clients). Some will brag about the state-of-the-art approach that they use. Some will be vague about it.

Solutions to End-of-Chapter Cases

Case Study: A "College Education Completion" Methodology

1. What are the phases that your college education completion life cycle might have?
2. What are some of the activities included with each phase?
3. What are some of the techniques you might use to help complete those activities?
4. What models might you create? Differentiate the models you create to get you through college from those that help you plan and control the process of completing college.
5. What are some of the tools you might use to help you complete the models?

Answers will vary. One approach involves planning for going to college, analyzing the requirement for what you want out of college, designing a specific program of study, and implementing the schedule and completing the classes for the program of study until college is complete. Another approach is to make up more specific phases: getting in, moving away from home, meeting new friends, deciding on a major, establishing a social life, completing courses, planning for the job hunt, hunting for a job, participating in final graduation activities (social and academic), and moving on after graduation. Remember:

Technique: A strategy or guidelines for getting an activity or task completed.
Model: Something created as an outcome of a task or an activity.
Tool: Something that is used by the technique to create the model.

Running Cases: Community Board of Realtors

1. Compared to the Tradeshow application described in Chapter 1, how long might this project take, and which approach to the SDLC would be most appropriate?

Answers will vary.

In the Tradeshow system in Chapter 1 we illustrated an iteration as a week-long mini-project. In reality it will probably require a normal iteration of 3 or 4 weeks. The other subsystem will probably also require an iteration of 3 or 4 weeks. The Community Board of Realtors is a comparable sized system – maybe a little larger with about 12 to 15 use cases. It will probably require three or four iterations.

The Community Board of Realtors system is also a well defined requirement. So rather than do it as an adaptive project with iterations, it could very easily be done as a single predictive type of project. In fact, probably most developers would proceed with a normal modified waterfall approach to this system.

2. If you use a predictive SDLC, how much time might each phase of the project take? How much overlap of phases might you plan for? Be specific about how you would overlap the phases.

Answers will vary.

Using a predictive approach with a modified waterfall approach some estimates (guesses) of time required:

Planning: Schedule, cost estimate, work team – probably a week.

Analysis: Use cases and domain models – probably around three or four weeks.

Design: Packages, network, domain model, user interfaces, use case realization (use case design) – probably another three or four weeks.

Implementation: Programming and testing – probably about six to eight weeks.

Overlapping the last three phases is desirable. There are three primary users (from chapter 3), and working on the new system can be partitioned by those use cases. The analysis phase could focus first on the domain model, which impacts all use cases. Then it could focus on the use cases associated with the MLS office and begin designing and implementing those use cases. Additional analysis, design and implementation could then follow for the Real Estate Office and the RE Agent.

3. If you use an adaptive SDLC, how many iterations might you plan to include? What use cases would you analyze, design, and implement in the first iteration? What use cases would you work on in the second iteration? In additional iterations? Think in terms of getting the core functionality implemented early and then building the supporting functionality.

Answers will vary.

Assuming about 12 to 15 use cases, this project would probably be handled in three or four iterations. (The list of use cases developed in Chapter 2 will need to be revisited and probably expanded. CRUD analysis will indicate if enough use cases have been defined to maintain and report data. There may need to be some more 'R'eporting use cases defined.) The most important data for this application is the property listing data. However, the property listing information is dependent on real estate offices and real estate agents. The interdependencies are quite tight and it does not make sense to try to deploy part of the system early. The core functionality is to build and maintain the database. The supporting functionality is to report and view the data. Hence a possible solution might be:

Iteration 1: Use cases to create and update real estate offices and agents.
Iteration 2: Use cases to create and update property listings
Iteration 3: Use cases to produce views and reports of the data

4. Let us say this project focused on Web access to the MLS. If you also plan to deploy a smartphone application for use by the public and by the agents and brokers, how might this affect your choice of the approach to the SDLC? What are the implications for including the smartphone application in the initial project versus having a separate project for wireless later?

Answers will vary.

Including a smart phone application will add considerable complexity to the system. The current approach is to use desktop and web based systems. Smart phone is an entirely new requirement with new user interface issues, communication issues, and usability issues. There are also a lot of unknowns with smart phones because the capability of smart phones tends to change rapidly. By including a smart phone app, it would appear that an adaptive approach to the system development would be best.

If the smart phone requirement was included in the original project, it could be isolated into its own iteration(s). However, the deployment of the entire system might be delayed if it is included in the original project. It might make more sense to go ahead and deploy desktop and Web solution, and then add smart phone functionality as a separate project. The danger with that approach, is that the original solution should be designed and constructed so that it does not have obstacles to later implementing smart phone capability.

5. Consider using incremental development to include the Web application and the wireless support. Describe what would be included in the first and second deployments of the project. Take into consideration that you might want to work on some initial problem solving for requirements, design, and implementation of the wireless support at the same time you are working on the Web application.

Answers will vary.

As indicated in Problem 4, the original design should consider the implications and issues related to smart phone applications.

The core functionality is to maintain the database, e.g. create, update, delete offices, agents, and property listings. This functionality is best done via desktop and web applications (typing critical data on a smart phone tends to be error-prone). Hence there is not as much need to implement the core database maintenance functions on a smart phone.

Viewing and reporting should be available for desktop, tablet, and smart phone devices. Hence after the basic database update is constructed, it might make sense to address the smart phone technical issues in a short iteration early in the project. Any new requirements for network, communications, user interfaces, etc. can then be addressed early in the project. Follow-on iterations could then complete the implementation of all the viewing and reporting functions.

Running Cases: The Spring Breaks 'R' Us Travel Service

1. The SBRU information system includes four major subsystems: Resort relations, Student booking, Accounting and finance, and Social networking. Although you have only worked with the domain model class diagram for the social networking subsystem, list as many of the domain classes that would probably be involved in each of the subsystems. Note which classes are used by more than one subsystem.

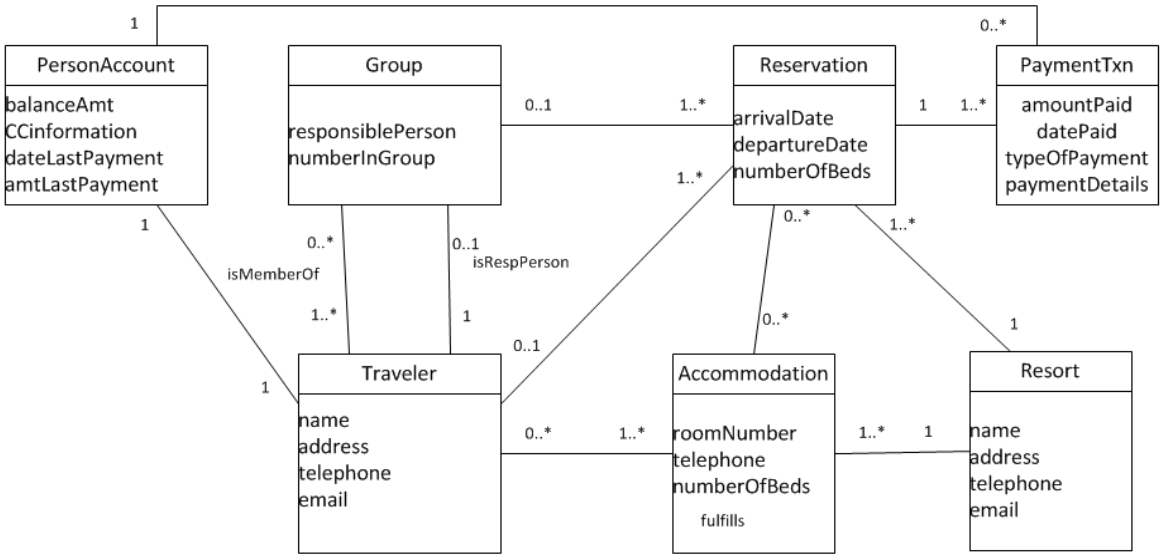
Note: In Chapter 5 we enhanced the domain model from Chapter 4 in order to discuss Student Reservations. We will use that domain model as our starting point and add other classes as necessary.

Subsystems	**Classes**
Resort relations	Resort Accommodation Facility Activity Traveler TravelerInRoom Comment (read)
Student booking	Reservation Traveler PersonAccount Group Resort (read) PaymentTxn
Accounting and finance	PaymentTxn PersonAccount (read) Traveler (read) Resort (payments to resorts)
Social networking	Traveler (read) Group (read) Interest** Comment Resort (read) TravelerInRoom (read) Accomodation (read) Facility (read) Activity (read) MultimediaInfo**

Note: ** denotes class NOT used by multiple subsystems. All other classes are accessed by multiple subsystems.

2. Based on the overlapping classes, what domain classes seem to be part of the core functionality for SBRU? Draw a domain model class diagram that shows these classes and their associations.

Core functionality will be to allow students to make reservations at resorts. That is the first and most important functionality. That requires resort information and student information.



3. Let us say you plan to implement the basic uses cases that create and maintain the classes that are part of the core functionally you just modeled. Describe what domain classes you would focus on in each iteration if you assumed that you would need two iterations for the initial core functionality and two additional iterations to complete each of the subsystems.

The above (Question #2) domain classes cross two subsystems, Student booking and Resort relations. So we will have two iterations on the core functionality in these two subsystems, and then the remainder of the use cases (from Chapter and domain classes to finish those two subsystems.

Iteration	Subsystem	Use Cases	Domain Classes
#1	Resort relations	Sign up with SBRU Edit resort info Create SBRU website Post availability of rooms	Resort Accomodation
#2	Student booking	Join SBRU View Resort info Make a reservation	Traveler PersonAccount Reservation PaymentTxn
#3	Resort relations	Create facility info Create Activity info View comments Assign traveler to accommodation	Facility Activity TravelerInRoom Comment
#4	Student booking	Create group Make payment Cancel reservation	Group PaymentTxn

4. How might you use incremental development to get some core functionality or some subsystems deployed and put into use before the project is completed?

These first two subsystems are the most critical and could be deployed first. Even assigning travelers to rooms does not need to occur until the students arrive. So given the above (Question #3) approach these subsystems could be deployed prior to the Accounting and Finance and the Social Networking subsystems.

Running Cases: On the Spot Courier Services

1. In what order would you develop the four subsystems? Support your answer.

1. Customer account subsystem (like customer account)
2. Pickup request subsystem (like sales)
3. Package delivery subsystem (like order fulfillment)
4. Routing and scheduling subsystem

The above list would allow On the Spot system to start supporting picking up and delivering packages. Although all of the Customer account system is not needed at first, at a minimum the customer class is required to be able to assign requests and packages to customers. So either the entire Customer account system can be implemented first, or at least a single Customer class can be implemented and next the Pickup request followed by the Package delivery subsystem.

After the pickup and delivery subsystems are working, the rest of customer account subsystem is highest priority. The routing and scheduling subsystem is purely internal and could be done by hand if necessary until the system was completed.

2. Reviewing your work from Chapter 3, assign each of your use cases to a particular subsystem. Does this change your answer or does it strengthen your original premise? Support your answer.

Note: As this case unfolds, it becomes evident that additional use cases can be defined from a CRUD analysis of the domain classes in Chapter 4. The table below includes some of those additional use cases.

Subsystem	**Use Case**
Pickup request	Enter/update request for pickup information Cancel request for pickup Enter package pickup information (scan) Update package pickup information Print label
Package delivery	Enter package delivery information (scan) Enter customer signature (delivery) Get package tracking information
Customer account	Add/update a customer Delete a customer View customer account information Print invoices Enter payment information
Routing and scheduling	Enter package event information (scan) Create trip manifest Update trip manifest Print trip manifest (packages to be delivered) Display trip manifest (real time)

	Assign package to trip (truck route)

3. Reviewing your work from Chapter 4, assign each of your classes to a subsystem. (Note: Some classes may be in multiple subsystems. The primary subsystem is the one that "creates" the objects in that class.) Does this change your answer or does it strengthen your original premise? Support your answer.

Subsystem	**Class**
Pickup request	PickupRequest Package Customer CustomeAccount Payment MovementEvent RouteTrip
Package delivery	Package Customer MovementEvent Employee RouteTrip
Customer account	Customer Individual Customer Business Customer CustomerAccount Payment
Routing and scheduling	RouteTrip Employee Package MovementEvent

4. Considering the Agile modeling principles, discuss each of the following:
(a) In Chapter 3, you developed a list of use cases and a use case diagram. If you follow the Agile modeling philosophy, how much or how little of this model do you think is necessary? Support your answer.

Since many activities of analysis and design are "use case" driven it is absolutely necessary to have a list of use cases. The Use Case diagram is simply a way to organize the use cases and to show the actor that invokes the use cases. So even though a list of use cases will contain the same information as the Use Case diagram, the diagram is extremely useful to communicate relationships and utility. It is useful as a monitor for later work such as detailed design. Keep it.

(b) In Chapter 4, you developed a class diagram. If you follow the Agile modeling philosophy, how much or how little of this model do you think is necessary? Support your answer.

A class diagram has essential business rules expressed in the relationships and cardinality constraints. It is also heavily used by developers in the development of the database. This model is absolutely necessary.

(c) In Chapter 5, you developed some use case descriptions, activity diagrams, and system sequence diagrams. If you follow the Agile modeling philosophy, how many or how few of these models do you think are necessary? Support your answer.

Minimizing the models, as proposed by the Agile philosophy will enable us to eliminate some of the models used in Chapter 5. For simple use cases, there may be no need for any of the three models.

Also there is some duplication between the fully developed use case description and the activity diagrams. For those that are visually oriented, an activity diagram may be more helpful. However a compbination of fully developed use case description and a system sequence diagram will provide a strong base of understanding. In many cases not all three models are required, even for complex use cases.

(d) In Chapter 6, you developed a network diagram and specified hardware requirements. If you follow the Agile modeling philosophy, how many or how few of these models do you think are necessary? Support your answer.

The modeling requirements from Chapter 6 would depend heavily on the complexity of the system. A simple system may not require a network diagram and the model can be eliminated. Complex solutions, however, may require a model in order to know how to configure the solution.

Running Cases: Sandia Medical Devices

1. Given the system goals, requirements, and scope as they are currently understood, is the project schedule reasonable? Why or why not?

Although the technical issues related to this project are complex, the number of use cases and the application functions is limited. A six month development period should be adequate. The three month testing period is good given that this is a health system and could have serious consequences if it did not work correctly.

2. How well understood are the system requirements at the start of the project? What are the implications of your answer for using a predictive, adaptive, or mixed SDLC? What are the implications of your answer for using Agile techniques?

Answers will vary.

Given the use cases and problem description provided, it would appear that the overall view of the requirements is quite good. However, the details are very much unknown at this time. And the technical difficulties may not be understood. Hence probably an adaptive approach would be best in this case. Agile techniques would fit well in this environment – working closing with the users and building code without a lot of modeling.

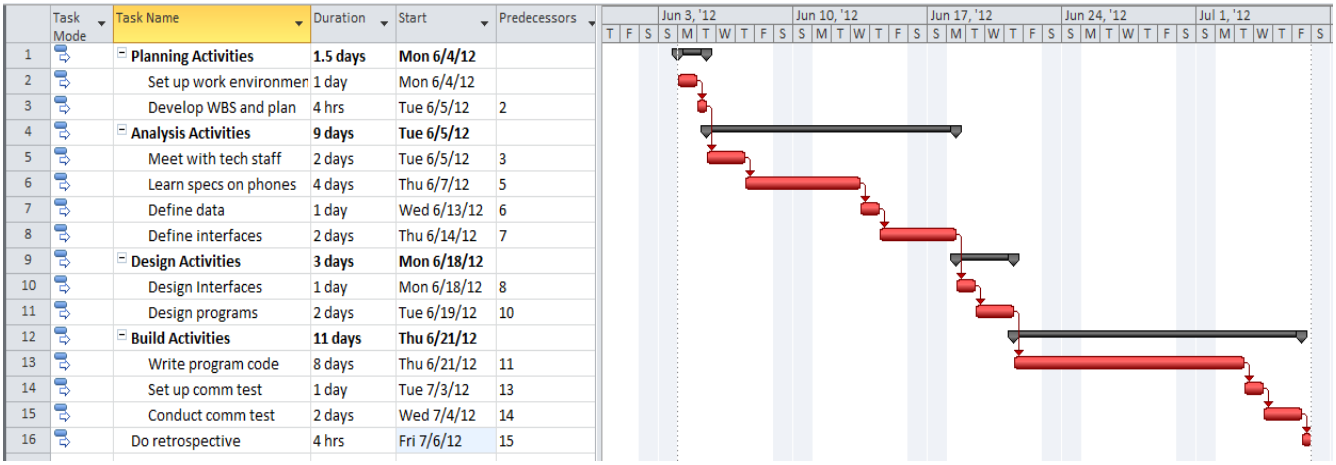
3. Medical personnel at NMHS have very busy schedules. NMHS's decision to assign two medical practitioners to the project for one day a week represents a significant investment in salary and lost revenue. How should project iterations be structured to ensure rapid progress to completion, high quality, and efficient use of medical practitioner time?

Answers will vary.

4. Prepare a detailed work schedule for the first iteration. If you have access to project management software, prepare the schedule and a Gantt chart by using the software.

Answers will vary. This project is highly technical, and especially this iteration requires little user involvement. It addresses technical areas.

Discipline	Activity	Effort	Dependency
Planning	Set up work environment	1 day	
	Develop WBS and plan work	1/2 day	
Analysis Activities	Meet with tech staff on specs of monitoring device	2 days	
	Learn specs on selected phone devices	4 days	
	Define data	1 days	
	Define interface requirements	2 days	
Design Activities	Design detailed interfaces	1 days	
	Design program structure	2 days	
Build Activities	Write program code	8 days	
	Set up for communication/integration test	1 day	
	Conduct communication & integration test	2 days	
	Do retrospective	1/2 day	
Total Days		**25 days = 5 wks**	



	Task Mode	Task Name	Duration	Start	Predecessors
1		**Planning Activities**	**1.5 days**	**Mon 6/4/12**	
2		Set up work environmen	1 day	Mon 6/4/12	
3		Develop WBS and plan	4 hrs	Tue 6/5/12	2
4		**Analysis Activities**	**9 days**	**Tue 6/5/12**	
5		Meet with tech staff	2 days	Tue 6/5/12	3
6		Learn specs on phones	4 days	Thu 6/7/12	5
7		Define data	1 day	Wed 6/13/12	6
8		Define interfaces	2 days	Thu 6/14/12	7
9		**Design Activities**	**3 days**	**Mon 6/18/12**	
10		Design Interfaces	1 day	Mon 6/18/12	8
11		Design programs	2 days	Tue 6/19/12	10
12		**Build Activities**	**11 days**	**Thu 6/21/12**	
13		Write program code	8 days	Thu 6/21/12	11
14		Set up comm test	1 day	Tue 7/3/12	13
15		Conduct comm test	2 days	Wed 7/4/12	14
16		Do retrospective	4 hrs	Fri 7/6/12	15