Preparing Systems Engineers of Tomorrow

***Ravi Shankar*[1]**

Abstract – The US Bureau of Labor Statistics has documented a definite and remarkable shift in jobs away from emphasis on programming and logic design towards emphasis on systems, applications, and management. We need to reorient our curriculum and our students to develop systems design skills in these students. An engineering product designed in a vacuum without appropriate stakeholder input may fail in the marketplace. Engineering students lack real-life skills and motivation to interact with stakeholders in other domains and to develop relevant commercial products for them. We believe that a top-down system design flow, built around UML (Unified Modeling Language), can provide a well-charted pathway for students to develop these skills with less anguish. We offered such six different projects for our students in a course on software-hardware codesign and helped them to incrementally develop their system designs, using UML. We will present our results in the paper.

Keywords: UML, System Design, Model Driven Architecture (MDA), Co-Design

Background

An engineering product designed in a vacuum without appropriate stakeholder input may fail in the marketplace. Despite ABET's guidance to the contrary, both professors and students in engineering lack the real-life skills and motivation to develop commercially relevant products. Many working engineers are content to focus on engineering issues, such as coding, logic design, and testing. Unfortunately, such pure engineering jobs can be automated, out-sourced, and migrated to technicians, as they are well-defined and really not that challenging anymore – witness the availability of low cost components and the popularity of component-based design, in software and hardware. More unfortunately, the same engineering professional might consider management and systems engineering below his/her intellectual and engineering capabilities. However, as per the US Bureau of Labor Statistics, the net employment increase over the next decade (2006 to 2016) in many computer occupations with focus on systems software, applications, and management will be 40 %, with a median annual salary of about $89K [1]. This may be compared with growths of -4% and 4% for computer programmers and computer hardware engineers, respectively, with respective median annual salaries of $65.5 K and $88.5K. This clearly dictates a shift away from the emphasis on programming and logic design towards systems, applications, and management.

Method

We recently revamped our course on software-hardware codesign to emphasize a top-down flow with UML version 2.0. Key to this was the availability of a well written book on UML (unified modeling language) with a good case study of a wireless handheld computer system [2]. UML is a visual modeling language that enables system designers to express their designs in a standard easy-to-understand way, and communicate these ideas effectively with different stake holders such as the customer, the business manager, and the engineers involved in hardware and software implementation. System design involves communication among people and miscommunication can cause design errors that can cause re-designs, design compromises, and/or field failures, after the product has been prototyped. These are costly mistakes and impact engineering design productivity substantially. An error inadvertently introduced at the requirements level is equivalent to 130 errors introduced at the prototype integration stage, in terms of engineering effort involved to fix it [3]. Such concerns have brought about the development of UML. UML is a collection of diagrammatic views that together provide a model of the system. One does not need all the UML diagrams, to describe a system. UML 2.0, the current standard, has 13 types of diagrams that may be

[1] Professor, Florida Atlantic University, Boca Raton, FL 33431, ravi@cse.fau.edu

divided into three types: 6 are useful for representing structure; 3 others are useful to describe behavior and the remaining 4 others are useful for depicting interactions. Typically, one uses class, object, package, and deployment diagrams from the structure diagram set; activity, use case, and state machine diagrams from the behavior diagram set; and sequence diagram from the interaction set.

The text book [2] details a case study of a wireless unit that serves to connect various workers at a restaurant with the primary objective of providing the patron a satisfying experience of dining out. However, it also helps to enhance the productivity of the restaurant workers and reduce their stress. The system design flow is intended for object oriented systems and follows the process of Requirements Gathering, Analysis, Design, Development, and Deployment. The book covers the first 3 stages; the last two stages are implementation stages that may be addressed with .NET from Microsoft. The stage of Requirements Gathering involves the discovery of the business processes (by interviewing the domain experts and summarizing the process with activity diagrams), domain analysis (identification of classes, their attributes, and operations – the first cut), identification of cooperating systems (a deployment diagram is the product), and discovery of system requirements where all the stakeholders meet to identify high level areas of system functionality (a package diagram is the result, with each package representing a set of use cases that are to be implemented to achieve that system functionality). This is followed by the Analysis stage, which involves the following steps: flesh out use cases, refine class diagrams, identify states in the objects (with a state diagram), define interactions among objects (with a sequence diagram), and integrate with cooperating systems. The next stage of Design involves: object diagram refinement, development of component diagrams, planning for deployment, development of the user interface, and design tests. This is followed by the stages of Development (and testing) of code and Deployment (hardware integration and testing). We covered the book's case study in detail in the class. To ensure that the students had specific goals and applied the concepts in parallel, we identified projects and provided assignments that paralleled the case study.

We derived inspiration from our industry collaborations/ innovations to identify 6 potential team projects for the students. We provide below titles and requirements provided for these projects:

1. Personalized Commencement (or Graduation Ceremony): Provide the students the ability to include a sound clip with proper pronunciation of their names so when they walk over to the podium to pick up their graduation certificate, everyone hears proper pronunciation of their names. Add the ability to showcase a 5 s video strip of their personal life when they accept their certificate.
2. Self-blood pressure management kit: Today's blood pressure monitoring systems are fairly old-fashioned. Today, there are different types of therapies based on drugs, diet, exercise, meditation, etc., to reduce blood pressure. The patient and his doctor want to know which subset of these is helping and should be continued on a long term basis. Design a system for this.
3. University class scheduling: Assume a typical university with 25000 students, 1000 faculty members, 2000 courses, and 500 rooms available Mo-Sa, 8 AM to 7 PM. Classes are to be 50 minutes long, with 10 minute break between class periods. Assume that the classes hold 50 or more students each. Assume that 100 of these class rooms have advanced equipment for projection and internet access. Faculty members have teaching preferences (MWF or TRS). Each faculty member can teach 4 courses, of which he/she will be assigned 2 depending upon their preferred days of teaching, class room availability, and type of room available. Design a system that mimics the functioning of the scheduling department. The numbers given are representative only. Plan on developing a generic scalable version.
4. Sound interface to PC for Pre-K and Kindergarten children: This will be useful for 3 to 7 year olds. They are too young to use the PC, but would like to. Conceptualize a system that can be used to teach these youngsters and help them learn alphabets and numbers (for the really young) and grammar and arithmetic (for the older ones). This may be used at home with the help of a parent or at school with the help of a teacher, who could use it to teach the group as a whole (they use handheld devices in middle and high schools now, to determine how many and which of the students got an answer right).
5. File tracking in a law firm: A successful legal firm has 100 legally trained employees (lawyers, legal clerks, etc.,). They have thousands of cases at various stages of progress. They work in groups of 5 on each case, and not necessarily the same 5 together always. Design a system that can provide PC-based information to all the employees on the current location of the case file. The employee, on a prioritized basis, can check out the file and return it, after a few days, to any of the several convenient central locations.

6. Poll worker Aid: Design a system that can help track the activities at multiple sites in a county so the electoral polling proceeds smoothly and the results are conveyed promptly and fully to a central location. This topic was inspired by polling boxes that were misplaced in the Palm Beach County.

The projects addressed the following: meet stakeholders (typically 3 to 5) in that domain to develop activity diagrams; identify class diagrams and associations (from nouns and verbs of the activity diagrams); conceptualize use cases; chart sequence diagrams and state diagrams; and design the user interface. To bring attention to the business and IT aspects, we added one more step: integrate an on-line business and IT interfaces. The process is completed by evolving a pseudocode/XML/Java/SystemC description of the skeleton code that clearly identifies the components, their interaction, and their internal behavior. Code development and component integration was not part of this course offering.

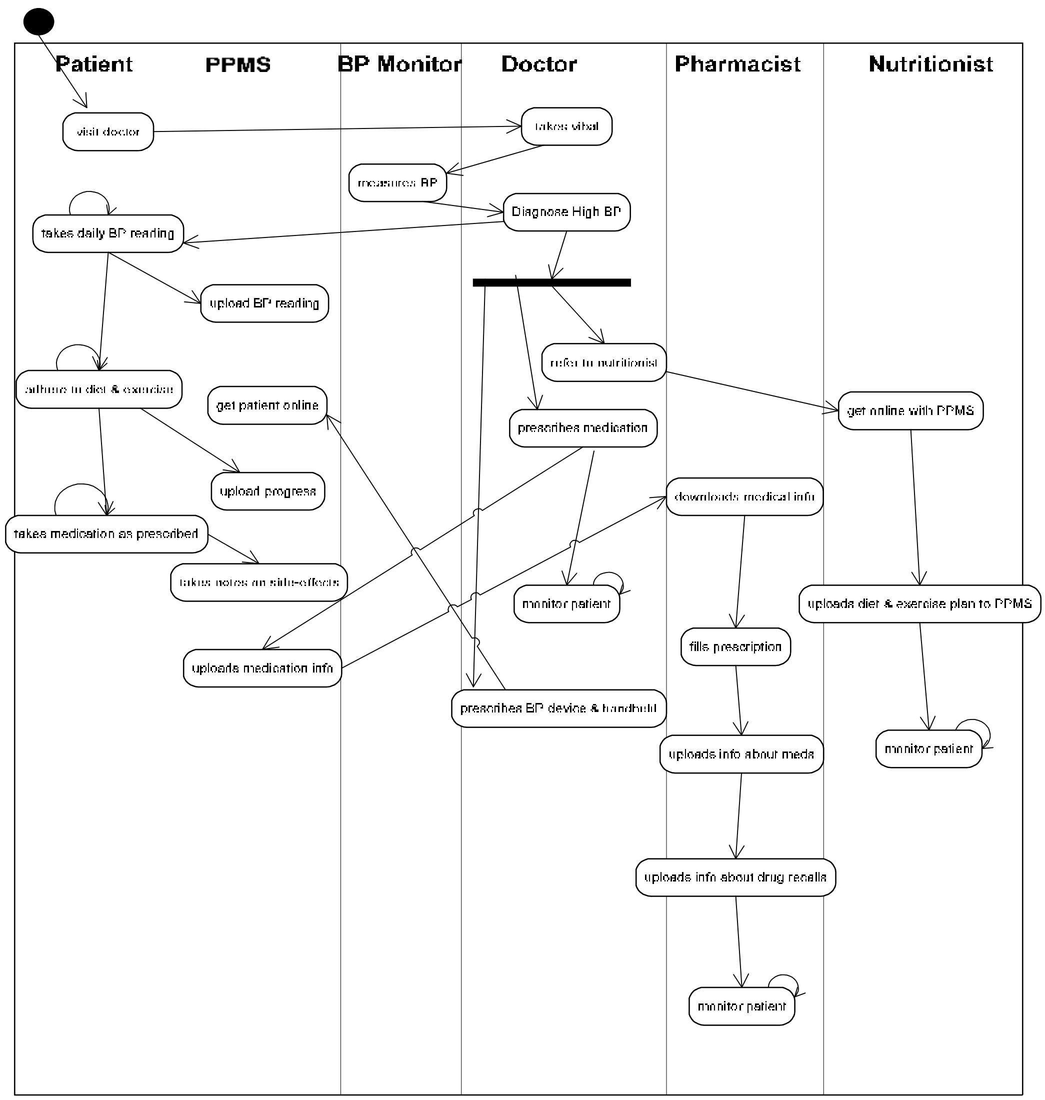

FIGURE 1: ACTIVITY DIAGRAM

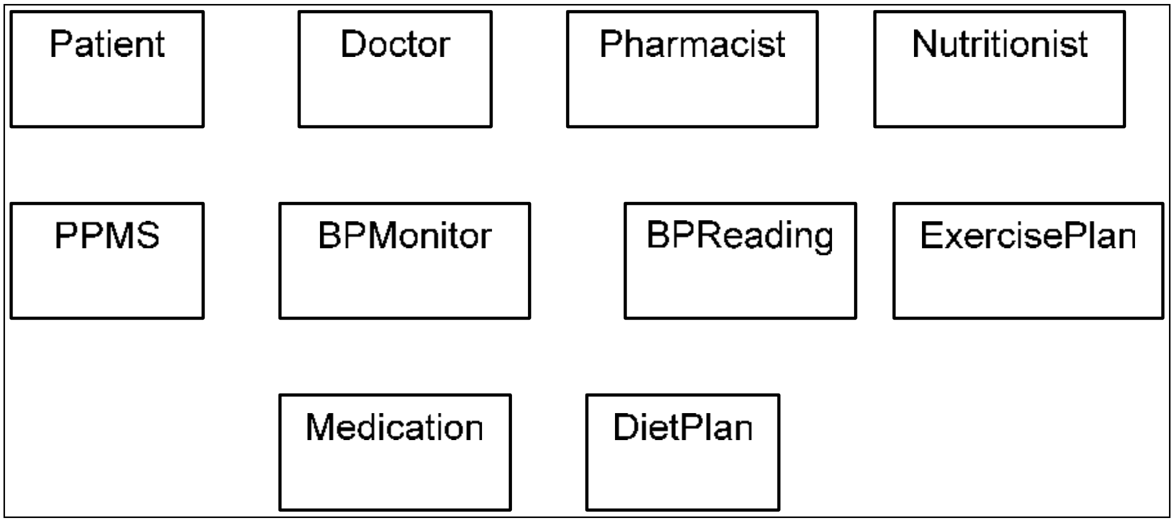

FIGURE 2: CLASSES OF THE PPMS SYSTEM

.

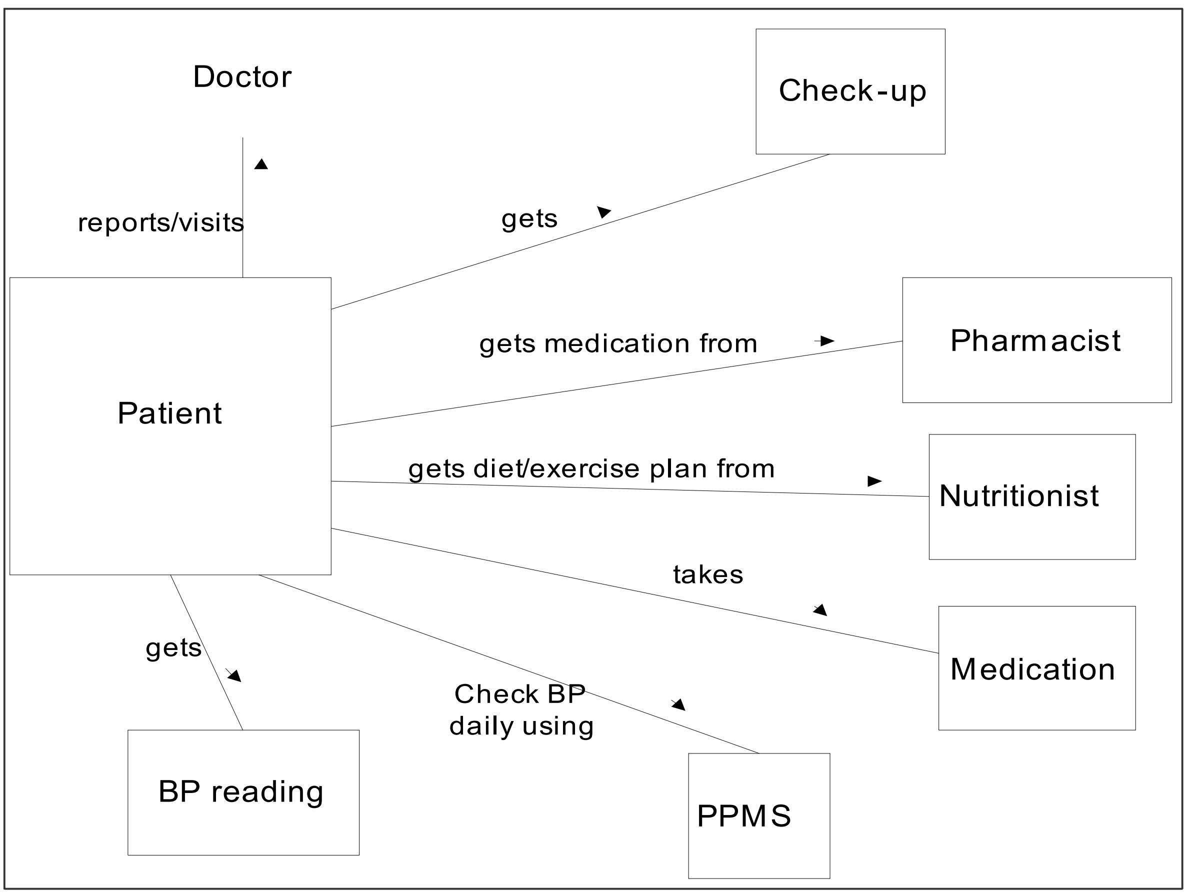

FIGURE 3: PATIENT CLASS AND ITS ASSOCIATIONS

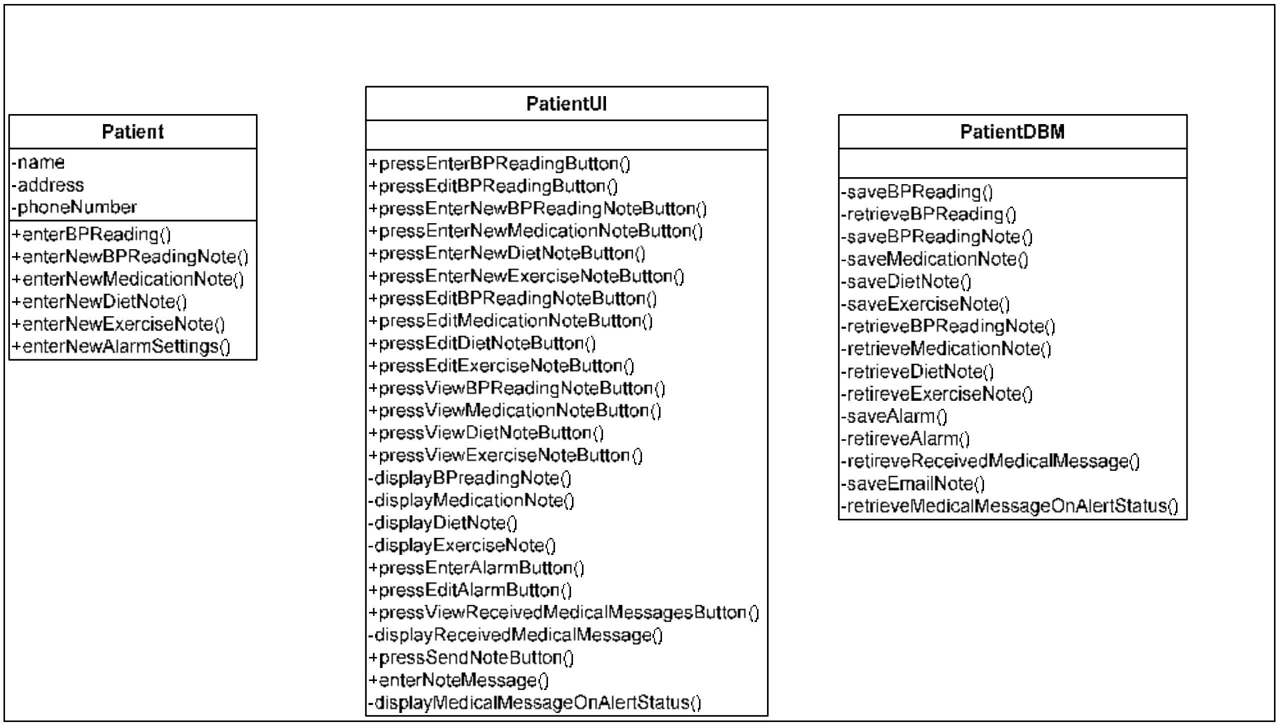

FIGURE 4: THE PATIENT CLASS

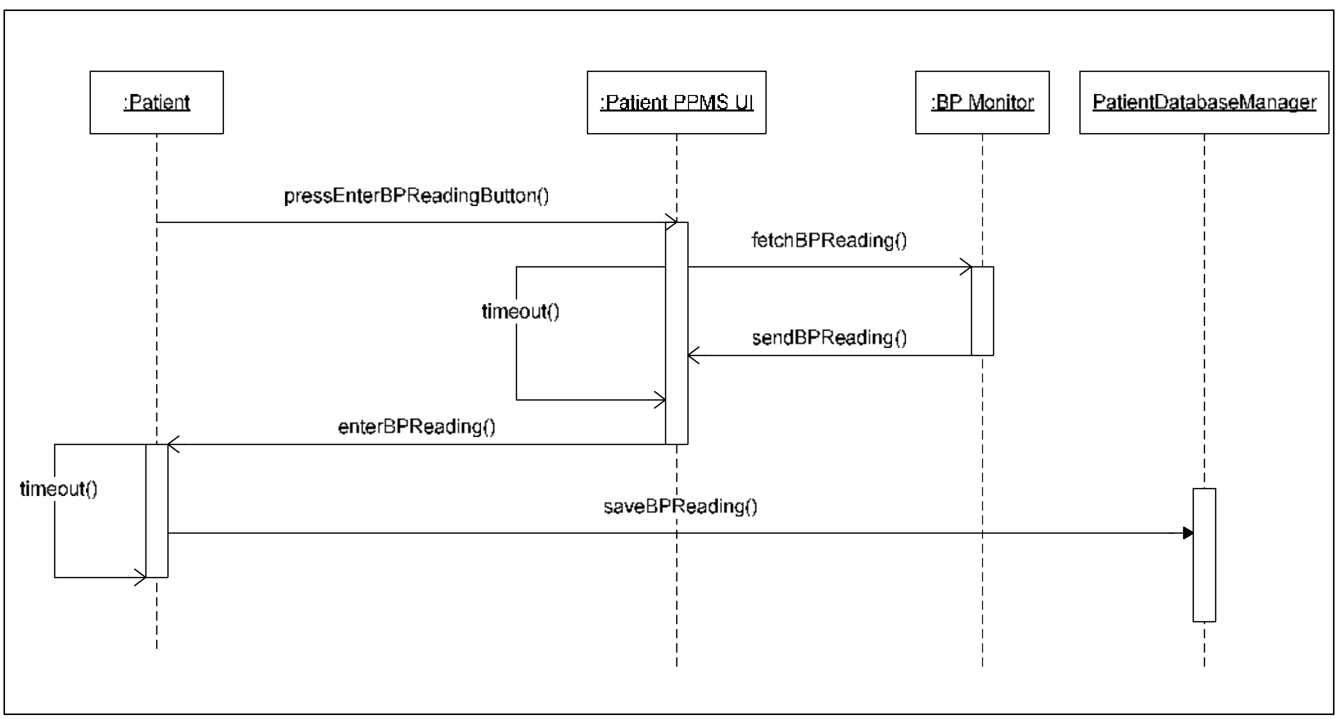

FIGURE 5: SEQUENCE DIAGRAM: PATIENT ENTERS BLOOD PRESSURE READING

RESULTS

All the student groups completed the projects to a reasonable extent. We have published the three good designs at our center's website [4, 5, and 6]. We use documentation of one group, who designed the Pressure (Perfect) Monitoring System, PPMS [4], to illustrate the design.

Figure 1 presents the activity diagram that the students developed at the start of the project. This eventually led to the classes of the PPMS system presented in Figure 2. These individual classes relate to each other (form associations with each other) in very specific ways. The patient class relates to every other class. Although one of the goals of PPMS was to facilitate communication between patients and the medical world, it was important that PPMS focused on the needs of the patient. The patient was assumed to be ultimately responsible for keeping track of his/her own progress; therefore, the student group geared PPMS to be a system which primarily assists the patient's needs. For this reason, the patient class will associate with every other class. See Figure 3. The patient class and its virtual representation in the system are illustrated in Figure 4.

Use cases are detailed in [4]. We present one specific use case below, relevant to Patient Interface in PPMS:

```
Enter Blood Pressure Reading
        Description
                -patient enters their daily blood pressure reading which they received from their in-home
        blood pressure monitor
        Assumptions
                -there is a patient interface on PPMS that allows for input data
                -patient has taken their blood pressure
                -patient has an in-home blood pressure device
        Precondition
                -patient has taken their daily blood pressure reading
        Postcondition
                -Daily blood pressure reading has been entered into the database and saved
        Steps
                -patient brings up user interface for entering blood pressure on PPMS
                -patient enters the reading manually or selects to download it from the electronic in-
                home BP device
                -reading automatically saves to database following a successful entry/download
        Benefiting Actor
                -Patient
```

We next present one specific sequence diagram for the use case of patient entering blood pressure reading. See Figure 5.

Pseudo code for the above sequence diagram is presented below:

```
void EnterBPReading()
{
        int fetch = fetchBPreading();

        If ( fetch =   null)
         {
         enterBPReading();
        connect_database;
        sendBPReading();
        saveBPReading();
        disconnect_database;                    // disconnect and return
        }
        else
```

```
            {
             connect_database;
            int send_reading = sendBPReading();
             if( send_reading = null)
                    {
                            exit;                       // return back to UI
                    }
            saveBPReading();
            disconnect_database;              //disconnect and return
             }
    }
```

FIGURE 6: PSEUDO CODE: PATIENT ENTERS BLOOD PRESSURE READING

The design projects ended with documentation of pseudo code and user interface design (not shown here). Most students did not implement the business and IT interfaces. The course is being offered again this semester with 10 different projects. Students will be asked to build in these additional interfaces.

We expect to implement two of these projects in .NET with web services. This will also be posted at the website. All the designs involve multiple types of users with their own unique interfaces. Thus, the self-blood pressure management kit will have web interfaces for the patient, doctor, pharmacist, and the nutritionist, in addition to the required business/social and IT interfaces. The group will help the patient manage his/her blood pressure with different interventions and with few side effects, as appropriate. The business/social interface may help the patient connect socially with other patients, while the IT interface will be used by the business to maintain and improve the system.

DISCUSSION

Some of the groups tried to bypass the step of interviewing domain experts. This resulted in use cases such as expecting a 4 year child to launch his own lesson from the PC (a teacher or a parent should have set it up for the child); and expecting a poll worker to wait to hand over the ballot box to the transporter (we told the team that automation also meant less manual supervision). Sequence diagrams use arrows to point to the object being invoked and the arrow should be labeled with that object's operation being invoked. Some students did not appreciate the concept and labeled the arrows with respect to the source of the arrow. This may be more intuitive, but does not help in helping the code developer implement the code using object oriented design. In sequence diagrams, time moves down along the life line; so, one follows the sequence from top to bottom to understand the flow of control. The students showed sequences going in opposite directions. By having students make presentations to the class and pointing out these errors politely, we believe, the group progressed as a whole. There is no automatic way to check completeness of the diagrams. For example, Patient UI is missing a function call represented in Figure 5.

UML is a big step forward. On the other hand, UML is not intuitive, because it spreads the information across many diagrams, making it difficult to track the system design and to gain an intuitive feel. We have started exploring OPM (Object Process Methodology) which combines objects and processes in the same diagram which can be hierarchically refined [7]. It provides the diagram in both visual and textual format, thus facilitating brainstorming and automatic code generation (at least at the level of skeleton code). Its biggest advantage may be at the level of requirements capture to extract the process flow from domain experts. One may wish to add this to model class diagrams obtained after the initial step of the use of UML's activity diagrams (to brainstorm with domain experts). Result will be a specifications document that can be translated automatically. We plan to explore its inclusion as it is very intuitive and students may find it useful to better integrate their thought processes.

We have had to hold several face-to-face meetings with the teams to get students to make progress. We found the teams to be uniformly distributed among motivated, reluctant, and disinterested. This is understandable, because the top-down design process requires time and effort up front to get the user requirements and engineering specifications right. Our own experience with companies shows that both managers and engineers feel that engineers are not being productive during this period. However, without this methodical approach, there is significant effort involved in development, integration and debugging. We have seen this personally in our ABET accreditation documentation [8]. We developed the CE Self-Study report with a disciplined top-down approach which has led to program accreditation without any revisions. This was not the case with other programs' self-study reports. Net result is that the methodical top-down approach actually reduces the development period and increases success of the developed product.

A few success stories will convince the future generations of graduates to take this process seriously. With that in view, we expect to evolve collaboration with the college of business to involve their BBA and MBA students in taking these specific projects forward as their business ventures. For that, we would have to implement the design and test it. We plan to integrate .NET in next semester's course, albeit with simpler class projects that can be taken all the way from UML to implementation with .NET. A unified approach to go from concept to commercialization will hopefully prove useful and worthwhile.

CONCLUSION

We have evolved a course that uses UML for top-down system design. There were six unique team projects that the students worked on. We expect to implement a few of them with .NET. Subsequent course offerings will integrate both UML and .NET so the students can fully appreciate the consequences of the decisions they made at the requirements level, and also to provide them with a sense of accomplishment by demonstrating their real implementations.

REFERENCES

[1] Bureau of Labor Statistics, http://www.bls.gov/, accessed on December 08, 2008

[2] Schmuller, J., *SAMS Teach Yourself UML in 24 Hours,* 3rd edition, 2004.

[3] Bennett, T.L., and Wenberg, P.W., Eliminating Software Defects Prior to Integration Test, *CrossTalk,* December 2005, pg. 13-18

[4] Mendolla, M., and Pillai, V., Top-Down Design with UML: Pressure Perfect Monitoring System, Student Report-3, http://www.csi.fau.edu/display/pub/Home#Home-studentReports

[5] Brown, A., and Baptiste, L., Top-Down Design with UML: File Tracking System, Student Report-5, http://www.csi.fau.edu/display/pub/Home#Home-studentReports

[6] Norona, C., and Templetron, R., Top-Down Design with UML: University Class Scheduling**,** Student Report-4, http://www.csi.fau.edu/display/pub/Home#Home-studentReports

[7] Dori, D., *Object Process Methodology: A Holistic Systems Paradigm*, Springer, 2002

[8] Shankar, R. and Agarwal, A., KISMET: An Open Source Process For Faculty Participation in ABET Accreditation, submitted to *2009 ASEE Southeastern Section Annual Conference*, Marietta, GA, April 2009

Ravi Shankar

Ravi Shankar is a professor in the computer science and engineering department at Florida Atlantic University (FAU). He is the director of a college-wide center on systems integration. He holds a PhD from the University of Wisconsin, Madison, WI, and an MBA from FAU. He is a registered PE and a Fellow of AHA. Over the past six years, he has coordinated the efforts of a large group of faculty members and students from computer engineering, computer science, and electrical engineering, to address Motorola's goal to radically increase their engineering design productivity. Motorola has provided $1.1 M in grants towards this goal. Significant progress has been made. Most of the publications are available at the center's website: www.csi.fau.edu. He has now started applying principles learnt in the research project to model and improve other complex processes.