Software Engineering & Project Management

Engr. Abdul-Rahman Mahmood
MS, PMP, MCP, QMR(ISO9001:2000)

armahmood786@yahoo.com
alphapeeler.sf.net/pubkeys/pkey.htm
pk.linkedin.com/in/armahmood
www.twitter.com/alphapeeler
www.facebook.com/alphapeeler
abdlmahmood-sss alphasecure
armahmood786@hotmail.com

alphasecure@gmail.com
http://alphapeeler.sourceforge.net
http://alphapeeler.tumblr.com
armahmood786@jabber.org
alphapeeler@aim.com
mahmood_cubi 48660186
alphapeeler@icloud.com

http://alphapeeler.sf.net/me
http://alphapeeler.sf.net/acms/

Microsoft Certified Professional
ALISON Certified
Teaching skills, Project Management

VC++, VB, ASP
Course portals

- [http://alphapeeler.sf.net/me](http://alphapeeler.sf.net/me)

**PAFKIET-SEPM-Fall2016**

PAFKIET Students registered in the course of Software Engineering and Project Management, during Fall 2016 session, please download your lectures, tutorials and web resources from here:


Facebook page for the course is here:

[https://www.facebook.com/PAFKIETSEPMFa16/](https://www.facebook.com/PAFKIETSEPMFa16/)

Group Email: pafkietsepmfa16@wigiomail.com
Engr. Abdul Rahman is expecting students to follow Classroom / Lab Policies, & Procedures listed below:

A. Note from Engr. Abdul Rahman: These policies to lead a respectful & disciplined classroom. You are responsible to comply with policies. If you fail to comply, there will be serious consequences.

B. Class / Lab Rules:

1. **Strict attendance policy:** Students are required to maintain 100% attendance throughout the session. 5 Minutes margin will be given after that student will be marked absent.

2. **No space for plagiarism:** In case, if any of the assignment/project deliverables found plagiarized, the whole assignment/project will be marked ‘ZERO’.

3. **Late submission:** Within 1 day after the deadline => 25% marks will be deducted. After 1 day => 50% marks will be deducted. After 2 days, ‘ZERO’ credit will be given.

4. **Submission of Assignment:** Students will submit their assignments within due date. If a student has an excused absence from class he or she is responsible for the assignments / homework that missed. It is up to the student to inquire about missed work and tests. Zero will be given if a student fails to make up work within an acceptable period. Following elements are mandatory for an assignment file:
   1. Assignment must be submitted in a proper file cover, and must be labeled properly.
   2. On cover page following items should be printed: Student name, Roll no, Date of submission.
   3. Attach print of the assignment question paper issued by the instructor after cover page.
   4. Attach hand written assignment after question paper.

5. **Consultation Time:** Students are advised to meet Engr. Abdul Rahman during the consultation time of the course only with prior appointment. Refer to the procedure for consulting hours from this url: http://alphapeeler.sourceforge.net/me/?page_id=158

6. **Project Submission:** The course required a proper project which will be submitted in Week 13. In this project, a proper report of at least 40 pages will be submitted after which a viva will be conducted in front of Engr. Abdul Rahman / HoD.

7. **Hand-held devices:** It is generally not acceptable to use cell phones, pagers, IPod/MP3 players, computers, etc. during lectures, except with the permission of Teacher and for reasons directly related to class activity.

8. **Lab assignments:** Assignments are checked only within lab timings. Lab files will not be checked after lab is over.

9. **Courtesy and respect to all:** Students will exhibit courtesy and respect to all other students at all times. Hateful comments concerning race, gender, sexuality, political views, appearance, or of any other type will not be tolerated; this applies to serious as well as “joking” comments.

10. **Food / Drinks:** Students may not eat / drink in classroom. This includes gum and candy.

11. **Make-Up Tests:** There is no official policy defined for make-up tests, if you are absent or have not appeared in test then zero marks will be given to you.

12. **Final Year Students:** Students who are engaged in FYP, are responsible to demonstrate their work at least twice a week in FYP lab, otherwise I may send unsatisfactory report to the FYP coordinator.
13. **Leave policy:** Application of leave is not entertained by the class teacher, it should be notified to the HoD, and CC to Director Academics / Examination & Manager Student Affairs. Even if the leave is approved, your class teacher will not mark you present on the basis of sick leave or any other type of leave. If you fail to maintain 75% attendance, you may not be eligible to sit in exams.

14. **Class compensation:** Engr. Abdul Rahman will notify the CR of the class in case of any class missed due to holidays or extra class required for students. It is the responsibility of class CR to schedule extra class by after reviewing the time table of class and teacher’s time table and book the classroom from administration block.

15. **Late arrival application:** No application will be considered for late arrival after the attendance has been marked. In case of genuine reason, students need to submit their late arrival application 1 week earlier along with the documented proof. Teacher has the right to dismiss the apology application in case of regular late arrivals.

16. **Asking Important questions for exam:** It is strictly prohibited to ask important questions from your course teacher. Teacher may report such students to management.

17. **Gifts:** It is strictly prohibited to present any kind of gift to your course teacher. Strict actions will be taken.

18. **Entering the Classroom Procedure:** Enter the classroom quietly and in advance of class starting time. Class start time means that you are in your seat and working on your exercise. Class CR is responsible to turn on the multimedia projector before the class starts.

19. **Classroom Exit Procedure:** Wait for me to dismiss you.

**C. Exam policies:**

1. Read all questions carefully first and then ask for clarifications.
2. Question paper related queries will not be entertained after 30 minutes after start of paper.
3. Do not write anything on question paper unless until specifically asked for.
4. Fill the required information and return the question paper along with the answer script.
5. Write your name, and enrollment number, otherwise you may not remain eligible for exam.
6. Get your paper signed from invigilator against your enrollment number; else your paper will not be checked.
7. Only attempt questions assigned to your column, otherwise you may disqualify from exam.
8. In case of MCQs, only circle one choice, otherwise you may disqualify from exam.
9. Any kind of miss-conduct/miss-behavior/cheating will disqualify the candidate.
10. Warning will be issues only once, along with -1 score, after that you will lose your eligibility for exam.

**D. If YOU CHOOSE to Break a Rule:** Punishments will always fit the crime. Of course there are behaviors that will warrant a Vice Principal’s Referral immediately. Examples of this include gross insubordination or violent behavior. Behaviors that are less severe, but in violation of the basic rules of the class will be dealt with in the manner described below. This format is in no way all inclusive and is subject to change:

1. **1st Incident** — Teacher/Student Conference
2. **2nd Incident** — Teacher/Student Conference, Parent Notification by phone or email, review behavior grade per grading policy.
3. **3rd Incident** — Referral to Administration / discipline committee.
What Is Software Construction?

- In common usage, "construction" refers to the process of building. The construction process might include some aspects of planning, designing, and checking your work, but mostly "construction" refers to the hands-on part of creating something.
- Problem definition
- Requirements development
- Construction planning
- Software architecture, or high-level design
- Detailed design
- Coding and debugging
- Unit testing
- Integration testing
- Integration
- System testing
- Corrective maintenance
Construction

- Problem Definition
- Requirements Development
- Construction Planning
- Detailed Design
- Coding and Debugging
- Unit Testing
- Integration Testing
- System Testing
- Corrective Maintenance
Why Is Software Construction Important?

- Construction is a large part of software development.
- Construction is the central activity in software development.
- With a focus on construction, the individual programmer's productivity can improve enormously.
- Construction's product, the source code, is often the only accurate description of the software.
- Construction is only activity that's guaranteed to be done.
Importance of Metaphors

- Kekulé’s dream of snake ~ molecular structure for benzene.
- Kinetic theory ~ "billiard-ball" model.
- Sound travelling in air ~ light as wave travelling in “ether” medium
- A good metaphor is simple, relates well to other relevant metaphors, and explains much of the experimental evidence and other observed phenomena.
- The Aristotelian thought that stone was really doing was falling with difficulty. ~ Galileo saw a pendulum. He thought that the stone was repeating the same motion again and again, almost perfectly. Galileo discovered laws the Aristotelians could not discover because their model led them to look at different phenomena.
Importance of Metaphors

- In 1973 Bachman pointed out that the ancients of data processing wanted to view all data as a sequential stream of cards flowing through a computer (the computer-centered view). The change was to focus on a pool of data on which the computer happened to act (a database-oriented view).

- The history of science is a series of switches from the "wrong" metaphor to the "right" one.
How to Use Software Metaphors

- A software metaphor is more like a searchlight than a road map. It doesn't tell you where to find the answer; it tells you how to look for it. A metaphor serves more as a heuristic than it does as an algorithm.

- An algorithm is a set of well-defined instructions

- A heuristic is a technique that helps you look for an answer.

- Person using metaphors to illuminate software-development will be perceived as someone who has a better understanding of programming & produces better code faster than people who don't use them.

<table>
<thead>
<tr>
<th>algorithm</th>
<th>heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take Highway 167 south to Puyallup. Take the South Hill Mall exit and drive 4.5 miles up the hill. Turn right at the light by the grocery store, and then take the first left.</td>
<td>Find the last letter we mailed you. Drive to the town in the return address. When you get to town, ask someone where our house is. Everyone knows us—someone will be glad to help you. If you can't find anyone, call us from a public phone</td>
</tr>
</tbody>
</table>
Common Software Metaphors


- Software Penmanship: Writing Code
  It doesn't require any formal planning, and you figure out what you want to say as you go. The writing metaphor implies a software-development process that's too simple and rigid to be healthy.
Unfortunately, the letter-writing metaphor has been spread by one of the most popular software books on the planet, Fred Brooks's The Mythical Man-Month (Brooks 1995).

The letter-writing metaphor suggests that the software process relies on expensive trial and error rather than careful planning and design.

Fred Brooks: Plan to throw one away; you will, anyhow. (1975)

Craig Zerouni: If you plan to throw one away, you will throw away two.
Software Farming: Growing a System

- creating software as something like planting seeds and growing crops. You design a piece, code a piece, test a piece, and add it to the system a little bit at a time. By taking small steps, you minimize the trouble you can get into at any one time.

- In this case, the incremental technique is valuable, but the farming metaphor is terrible.

- The weakness in the software-farming metaphor is its suggestion that you don't have any direct control over how the software develops. You plant the code seeds in the spring. Farmer's Almanac and the Great Pumpkin willing, you'll have a bumper crop of code in the fall.
Accretion: growth or increase in size by a gradual external addition or inclusion.

"incremental," "iterative," "adaptive," and "evolutionary."

As a metaphor, the strength of the incremental metaphor is that it doesn't overpromise. It's harder than the farming metaphor to extend inappropriately. The image of an oyster forming a pearl is a good way to visualize incremental development, or accretion.

Fred Brooks, wrote his landmark book The Mythical Man-Month so radically changed his own practice as incremental development (1995).

Tom Gilb made the same point in his book, Principles of Software Engineering Management (1988), which introduced Evolutionary Delivery and laid the groundwork for today's Agile programming approach.
The image of "building" software is more useful than that of "writing" or "growing" software. It's compatible with the idea of software accretion and provides more detailed guidance. Building software implies various stages of planning, preparation, and execution that vary in kind and degree depending on what's being built. When you explore the metaphor, you find many other parallels.

If you forget to provide for a door, as shown in Fig, it's not a big problem; you can fix it. All you've wasted is part of an afternoon. This loose approach is appropriate for small software projects too. If you use the wrong design for 1000 lines of code, you can refactor or start over completely without losing much.
More complicated structures require more careful planning

Greater complexity and size imply greater consequences in both activities. In building a house, materials are somewhat expensive, but the main expense is labor. Ripping out a wall and moving it six inches is expensive not because you waste a lot of nails but because you have to pay the people for the extra time it takes to move the wall. You have to make the design as good as possible, as suggested by Fig, so that you don't waste time fixing mistakes that could have been avoided. In building a software product, materials are even less expensive, but labor costs just as much. Changing a report format is just as expensive as moving a wall in a house because the main cost component in both cases is people's time.
Software Construction: Building Software

- It generally doesn't make sense to code things you can buy ready-made. House building: prefabricated cabinets, counters, windows, doors etc. Software building: use prebuilt libraries of container classes, scientific functions, user interface classes, and database-manipulation classes.

- Careful planning doesn't necessarily mean exhaustive planning or over-planning. You can plan out the structural supports and decide later whether to put in hardwood floors or carpeting, what color to paint the walls, what roofing material to use, and so on. A well-planned project improves your ability to change your mind later about details. It makes sense to code things you can buy ready-made.

- Customization: building a fancy house, you might have your cabinets custom-made. Dishwasher, refrigerator, and freezer built-in to look like the rest of your cabinets. Windows custom-made in unusual shapes and sizes. Customization in software: own scientific functions for better speed or accuracy, own container classes, user interface classes, and database classes to give your system a seamless, perfectly consistent look and feel.
You'd use still different approaches for building a school, a skyscraper, or a three-bedroom home.

Costly Change Requests: To move a wall six inches costs more if the wall is load-bearing than if it's merely a partition between rooms.

Capers Jones reports that a software system with one million lines of code requires an average of 69 kinds of documentation (1998). The RS doc for such a system would be about 4000–5000 pages long, and the design doc can easily be two or three times as extensive as the requirements.
Applying Software Techniques: The Intellectual Toolbox

- A good craftsman knows the right tool for the job and knows how to use it correctly.
- The more you learn about programming, the more you fill your mental toolbox with analytical tools and the knowledge of when to use them and how to use them correctly.
- In software, consultants sometimes tell you to buy into certain software-development methods to the exclusion of other methods: Microsoft
- If you buy into any single methodology 100 percent, you'll see the whole world in terms of that methodology.
- The toolbox metaphor helps to keep all the methods, techniques, and tips in perspective—ready for use when appropriate.
Combining Metaphors

- Metaphors are not mutually exclusive.
- Use whatever metaphor or combination of metaphors stimulates your own thinking or communicates well with others on your team.
- Example: You can use both the accretion and the construction metaphors.
- Example: You can use writing if you want to, and you can combine writing with driving, hunting for werewolves, or drowning in a tar pit with dinosaurs.