Design Defects and Restructuring

Engr. Abdul-Rahman Mahmood

 abdulrahman@nu.edu.pk

 alphapeeler.sf.net/pubkeys/pkey.htm

 pk.linkedin.com/in/armahmood

 www.twitter.com/alphapeeler

 www.facebook.com/alphapeeler

 abdulmahmood-sss  alphasecure

 armahmood786

 http://alphapeeler.sf.net/me

 alphapeeler#9321

 *reddit.com/user/alphapeeler*

 www.flickr.com/alphapeeler

 http://alphapeeler.tumblr.com

 armahmood786@jabber.org

 alphapeeler@aim.com

 mahmood_cubix  48660186

 alphapeeler@icloud.com

 pinterest.com/alphapeeler

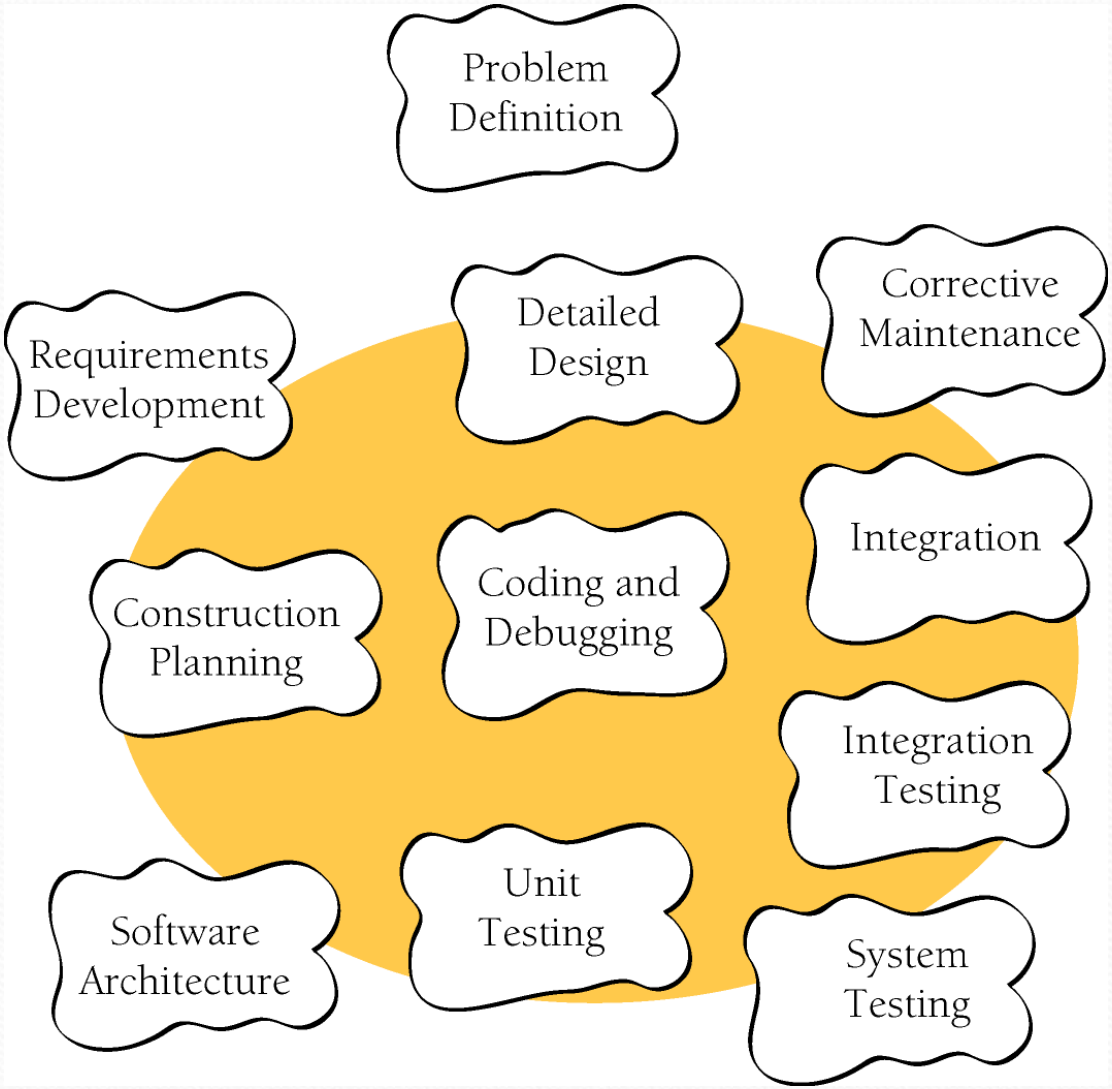
 www.youtube.com/user/AlphaPeeler

Introduction to Metaphors

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



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 the 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 isn't 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.

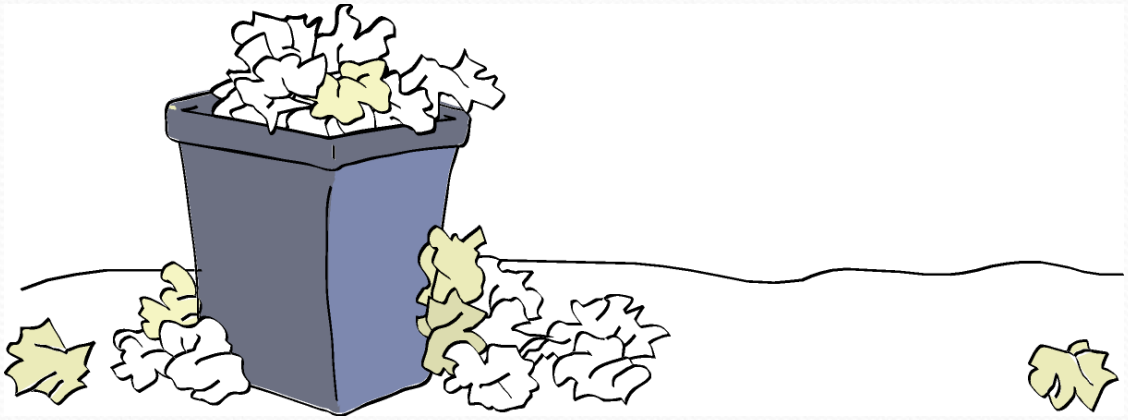
algorithm	heuristic
Take Highway 167 south to Puyallup. Take the South Hill Mall exit and drive 4.5 miles up the hill. Turn right at the light by the grocery store, and then take the first left.	Find the last letter we mailed you. Drive to the town in the return address. When you get to town, ask someone 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

Common Software Metaphors

- David Gries : software is a science (1981), Donald Knuth : art (1998). Watts Humphrey : process (1989). P. J. Plauger & Kent Beck : like driving a car. Alistair Cockburn : game (2002). Eric Raymond : bazaar (2000). Andy Hunt & Dave Thomas : gardening. Paul Heckel : it's like filming Snow White and the Seven Dwarfs (1994). Fred Brooks : farming, hunting werewolves, or drowning with dinosaurs in a tar pit (1995). Which are the best 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.

letter-writing metaphor

- Unfortunately, the letter-writing metaphor has been perpetuated 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.

Software Oyster Farming: System Accretion

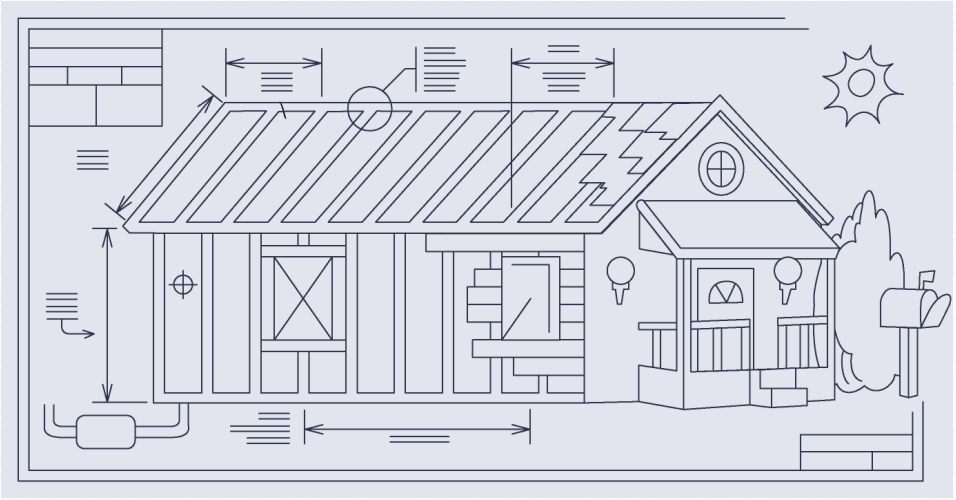
- Accretion : growth or increase in size by a gradual external addition or inclusion.
- "incremental," "iterative," "adaptive," and "evolutionary.“
- As a metaphor, the strength of the incremental metaphor is that it doesn't overpromise. It's harder than the farming metaphor to extend inappropriately. 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.

Software Construction: Building Software

- 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. make 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.

Software Construction: Building Software

- 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.