***MODELS*** WATERFALL MODL-Downside: no iteration(something done is done for good), 1-way flow of data. Output of 1 phase is input for the next. Upside: good for small projects where problem is well understood. V MODEL- Validation (answr to are we building correct product?) and Verification (answr to are we building product correctly?). Waterfall with PROTOTYPING MODL- must be small enough to build quickly. 10% LOCs compared to fin products LOC. ReqAnalys. most important phase. small prototype allows devs. to tell users how they view their problems. throw away proto and evolutionary prototypes. reduces risk and uncertainty. V MODEL-variation of waterfall. uses acceptance testing to validate requirements. go back to L side of model when R side things fail. easy and very good model that shows explicit iteration. gives place to back to when problems. V shares problem w/ waterfall that final product is not seen by customers till end of process. TRANSFORMATIONAL MODEL- everything formal after system requirements and formal specification. **user feedback is important and early feedback is very important in all**. SPIRAL MODEL- model is presented in spiral in which each iteration is represented by a circuit around 4 major activities (plan, determine goals, evaluate risks, develop/test). risk analysis differentiates this model bc no others include risk. AGILE MODEL- focus on human interactions and flexibility/quickness. focus on customer collaboration and responding to change rather than create plan and follow it. most glaring criticism of Agile: Agile does away with formallacy and rigid contracts.

***KEY PHRASES*** USER FEEDBACK: we want feedback early and often. project is ILL-UNDERSTOOD (to devs at beginning). SIZE of the problem is huge, and COSTS are HUGE. 3essential stages: ReqAnal; Design; Test.

***USE CASE***: is a case of uses. A case of use of the system we will build. a case of use of the system. ex- banking existed before online banking. ex- physical library. to computerize a person getting a book we need to keep use cases borrow/return/latefine the same. the total number of use cases is very small(und.20) and they define the system. in ReqAnal we produce a list of use cases then Design is centered around providing functionality of use cases, then Testing is ensuring each use case is properly delivered. UsCase driven process. UsCas wll have multiple scenarios(standard/success or failure/non-stand. @ least). succes and failure scenarios should be clearly defined to detailed level as possible in ReqAnal & Design parts.

OOP-Class Identification🡪Use nouns from text description of project to guide us in choices for classes. use nouns and noun phrases in ReqAnal to find out classes. do NOT use units of time or quantities for classes. NO abstract nouns such as “laws/rules” for classes. written code description🡪find nouns🡪get classes.

***UML***-graphical lang. that helps devs. and users communicate. stick figures = actors ex. a 2nd computer system. Association: two classes in a model need to communicate with each other. Aggregation and Composition are subsets of association. In both, object of one class “owns” object of another class. Aggregation implies relationship where the child can exist independently of the parent. kill the parent and child classes still get to stay in Aggreg. EX: Class(parent) and Student(child). Delete Class and Students still exists. Composition implies a relationship where the child cannot exist independent of the parent. kill the big thing and little things die in Compost. EX: House(parent) and Room(child). Rooms do not exist separate of houses. It’s Composition if one parent class object owns another child class object and that child class object cannot meaningfully exist without the parent the parent class object. It’s Aggregation if child class can exist without parent class. These dependencies translate to code in that one class uses another by parameter or return type.

***QUIZ*1**. y is soft dev. process hard?🡪 problem is ill-understood, large, and worked on by more than 1 person. SIZE itself accounts for problems. time and cost 2 main contraints.|**2**. limitations of waterfall meth.?🡪No customer interaction/feedback, solid for small projets w well understood prob.|**3.** pros and cons of V modl?🡪 1 to 1 correspondence of how to fix things/\hard to make heavy changes|**4**. what is use case? wat are UsCases in soft.dev. projects?🡪 “a case of uses” a single case of use. uses of UsCases: ReqAnal to figure out actions; development phase we want to be use case driven |**5**. similarities/differences in aggregation vs composition in UML?🡪 logical component vs physical component. |6. soft. engineering in 1 sentence?🡪 large scale industrial production of software where initial requirement are ill-understood and worked on by many people. |7. y soft. engineering not soft. science?🡪 final product is tangible product not theory, emphasis on product. Science has no room for ad-hocism, bc of scale and many people we do. |||| ***ProjectSpecs*** 1. Problem Statement and Prob. Parameters. Prob params. MUST HAVE. Size most important. others: v, Wi, W. min/max range. 2. Problem Type- 0-1 or F(x)al knapsack? 3. precision? 4. single or multi user? 5. web based or not? 6. time to develop? 7. cost to dev. 8. response time 9. software/hardware platform

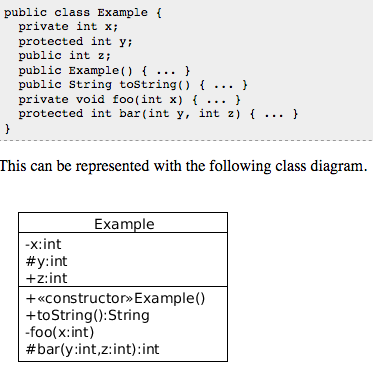
**suppose a is an object of A we could have statement a.m1(….) 🡪 this is an example of message passing. a is the object that receives the message. “who is sending the call to a?”

Figure 5.1: 1 class called Book and it has a bunch of attributes and methods. **STATE** of the object is the sum of all attributes or the values of each individual attribute. change of attributes causes a change of state.

***Objects*** communicate with one another by sending messages. A message is a method call from a message-sending object to a message-receiving object. An object responds to a message by executing one of its methods. Additional information, known as arguments, may accompany a method call. A message is valid if the receiver has a method that corresponds to the method named in the message and the appropriate arguments, if any, are supplied with the message. Only valid messages are executed by the receiver.