**CS2810 Team Project**

**Project Owner** – maintains backlog, priority

**Scrum Team** – Actual technical

**Scrum Master** – Everyone follows the rules + practices of scrum

**Product Backlog Refinement**

* Held before first sprint
* Product owner presents team with vision
* Turn it into user stories
* EVERYTHING could possible put into product
* Owner = gives a business value in each item
* TEAM = amt of effort each item cost
  + how much each sprint = velocity

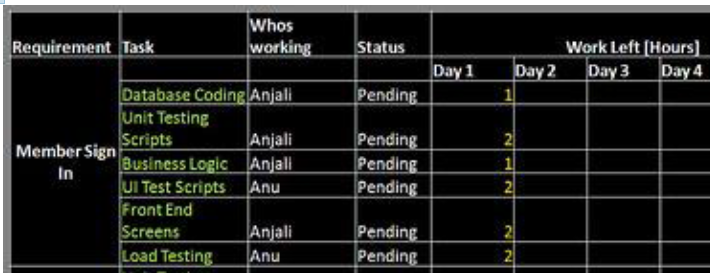
Sprint Planning Meeting

2 parts

Part 1

* Priority setting
* Agreement of item meaning
* Agreement of tests

Part 2

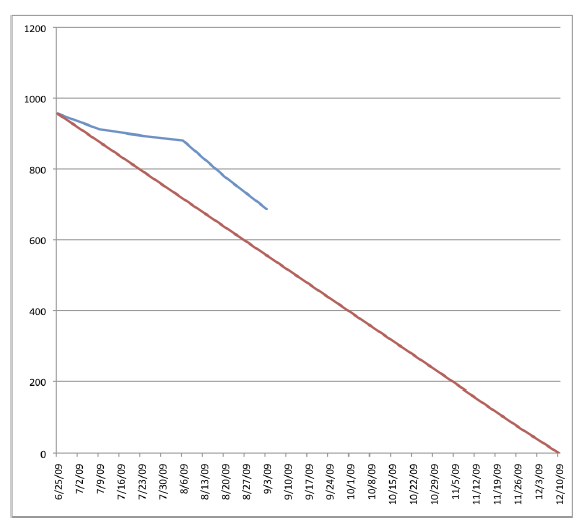
* Time each team mem can commit
* Choose product backlog to complete
* Break product backlog into tasks + estimate effort
* Allocates tasks to team members
* Tracking list:
* 

Scrum Meeting

* What was done
* Planning to do till next meeting
* Problems/impediments
* Short, punctual, nothing else
* Top priority = remove problems

Burndown Chart

* Remaining effort remaining



General Points:

* Communication
* Ideas for features and changes come up
* Fail Fast

Sprint Review

* 5 min presentation to owner
* If no pass of acceptance test = go back to backlog

Sprint Retrospective

* Discussion on what is good/bad

Rules

* Time-boxing
* Focus
* Honesty

Wrong

* Behind schedule
* Cant complete sprint backlog
* End a sprint early

Scrum Management

* Trello
* Board for each sprint
  + Lists = store user stories
  + Card = User story
* Use checklist

Boards

* Product backlog
  + Cards as user stories
* Sprint N

**Professional Issues**

Computer Scientist =

* Technical competence
* Judgement
* Professionalism

Ethical Issues FORCC

* Focus
  + Definite goals (don’t change)
  + Next step (don’t worry)
  + Use whatever (meet goals)
* Openness
  + Git/trello
  + Cannot Do = tell your team
* Respect
  + Some may need help
  + Someone cannot complete task
  + Scrum master = not boss
* Courage
  + Choose task, and accomplish task
  + Experiment and try
  + Git will keep old version
  + Fail fast
* Commitment
  + Committed, no boss

Dependability

Reliability = link to dependability

Attributes

* Availability (Readiness)
* Reliability (continuation of correct service)
* Safety
* Integrity
* Maintainability

Threats

* Fault
* Error
* Failure

Means

* Fault tolerance
* Fault prevention
* Fault removal
* Fault forecasting

Fault -> Error -> failure (not necees)

80-20 rule

Software Quality = defects

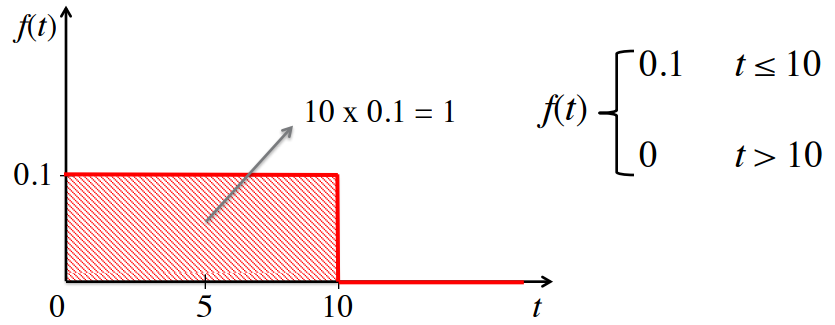
* No. of defects/size
* Time to fix each

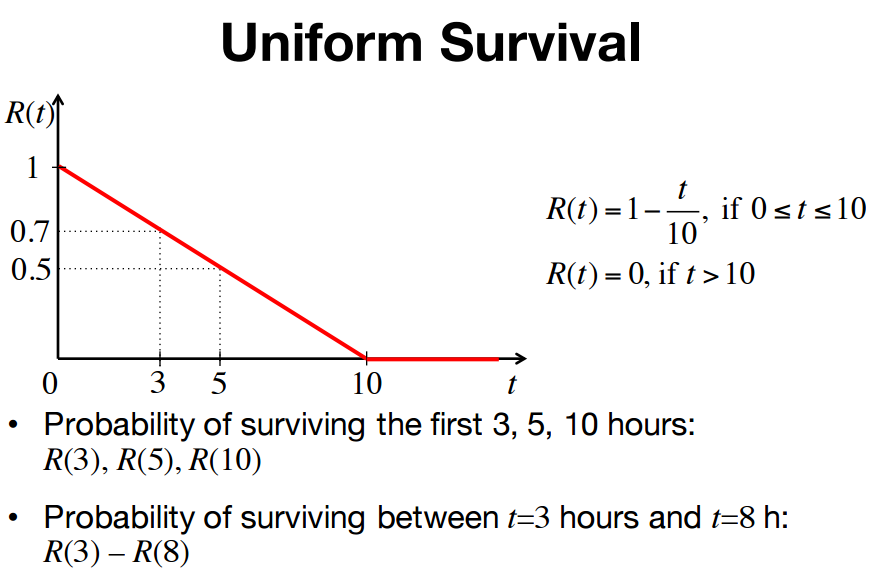
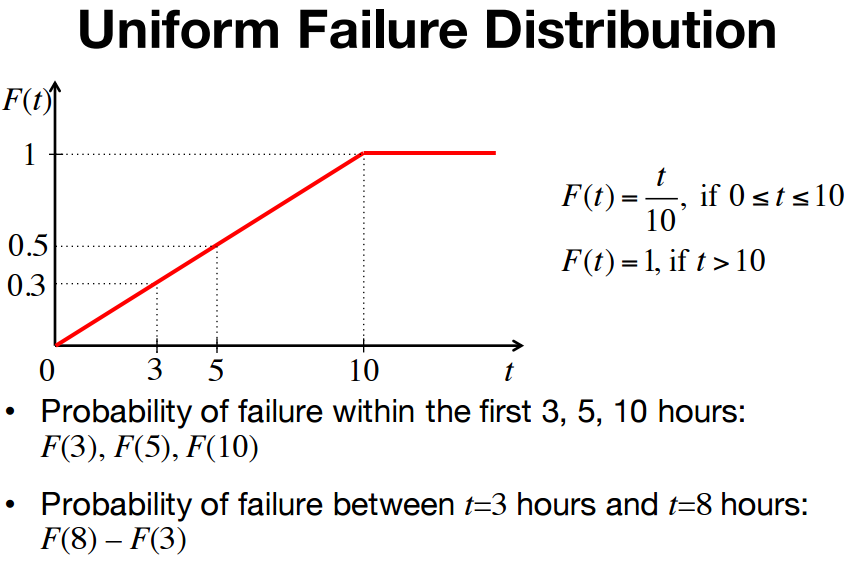
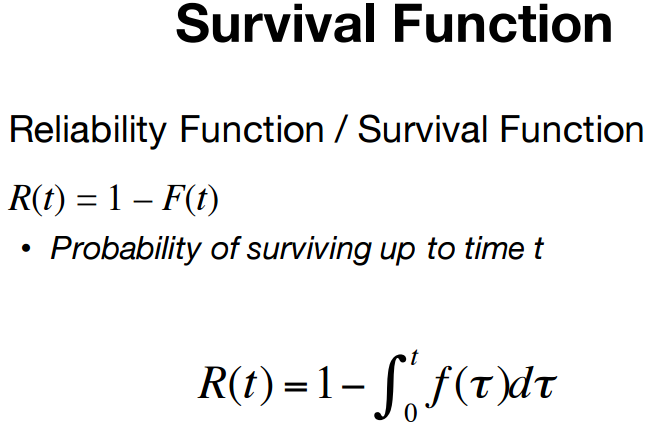
Software reliability = quantitative study of operational reliability of software

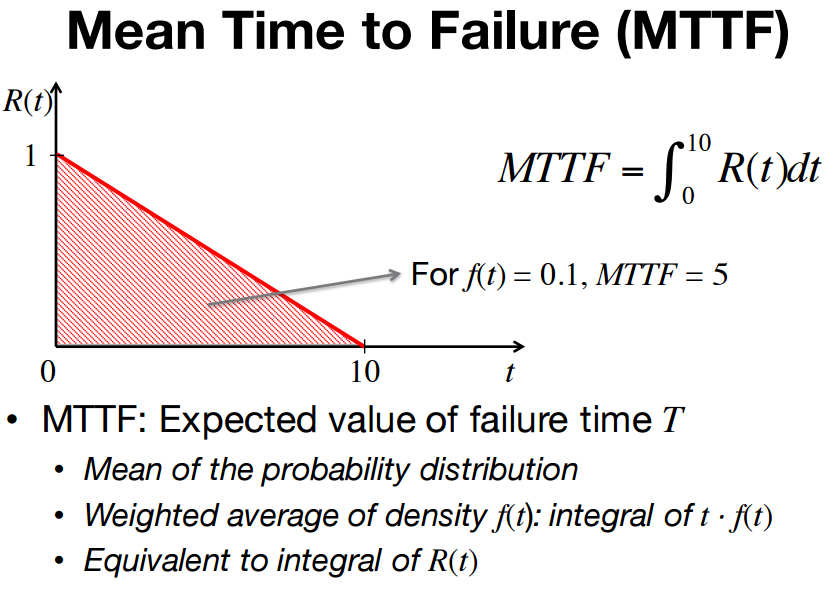
Predicting Failure time

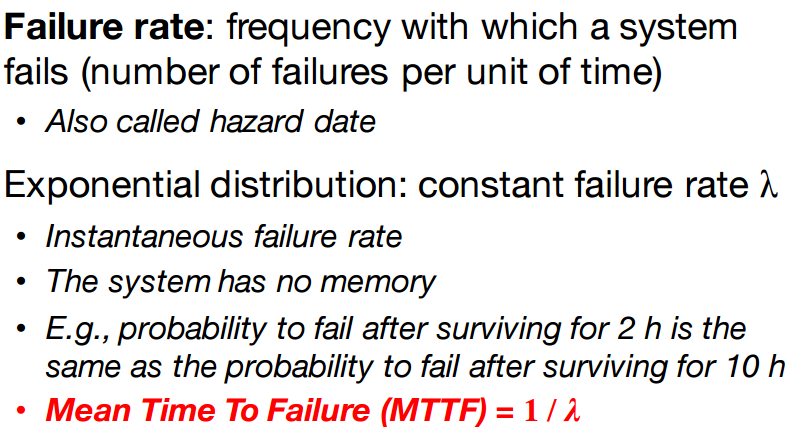
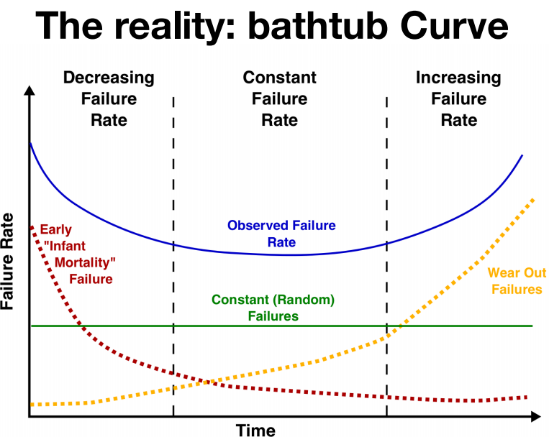
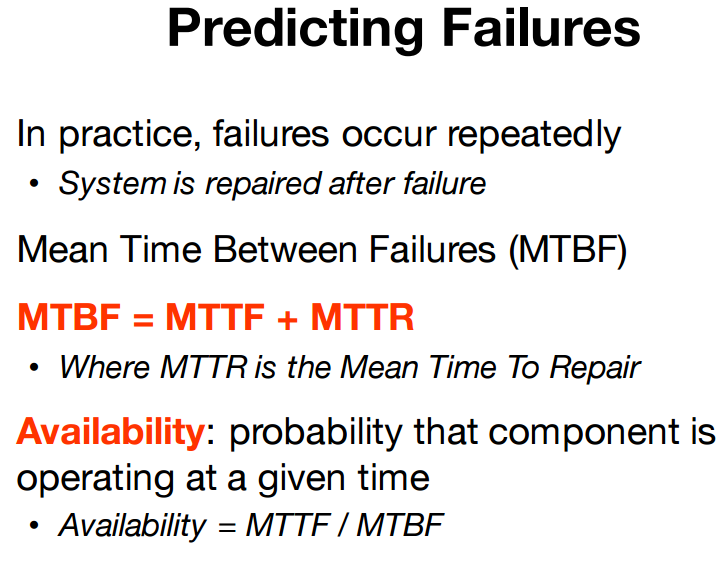
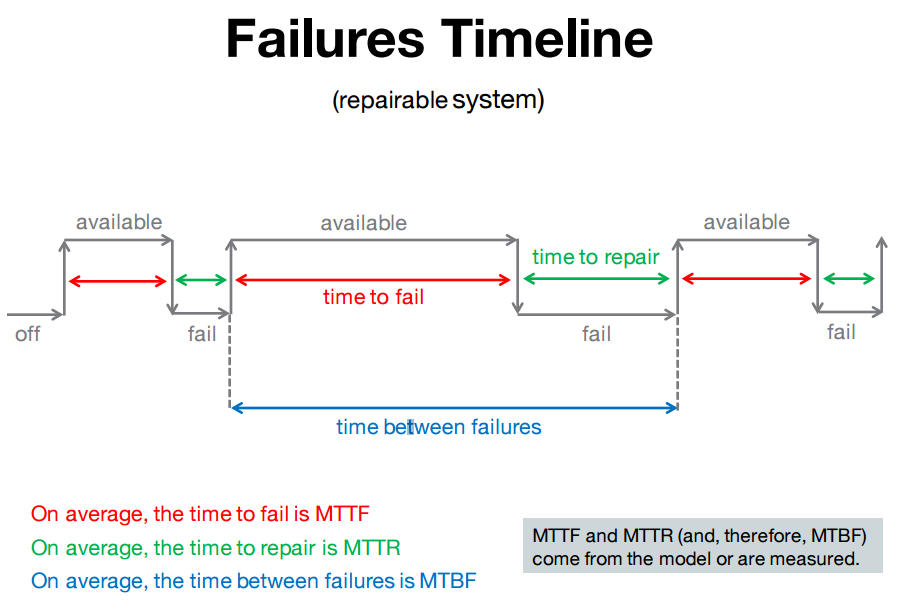
* Failure occurrence are probabilistic in nature
* MTTF

Probability Density Function

* Graph of failure over time
* 
* 

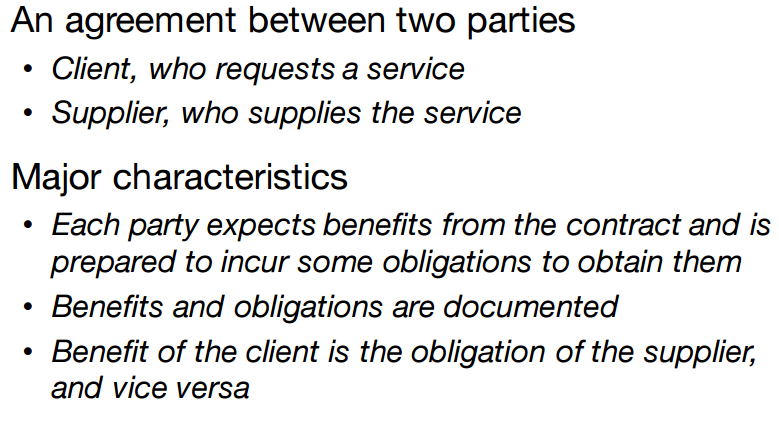
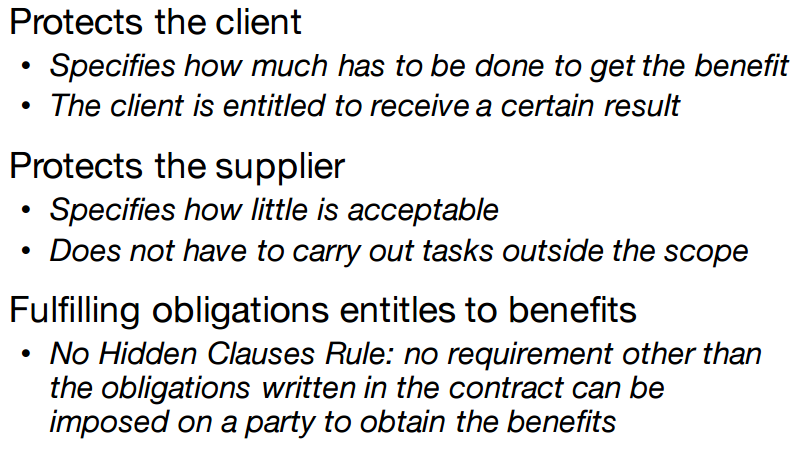
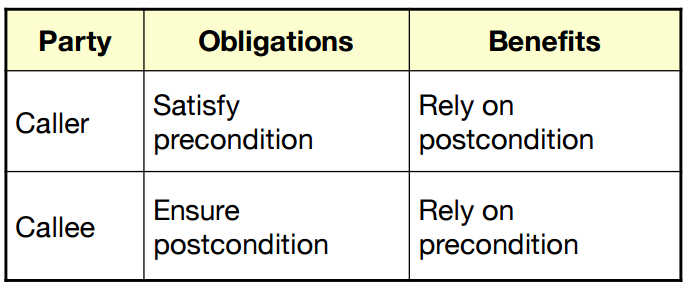
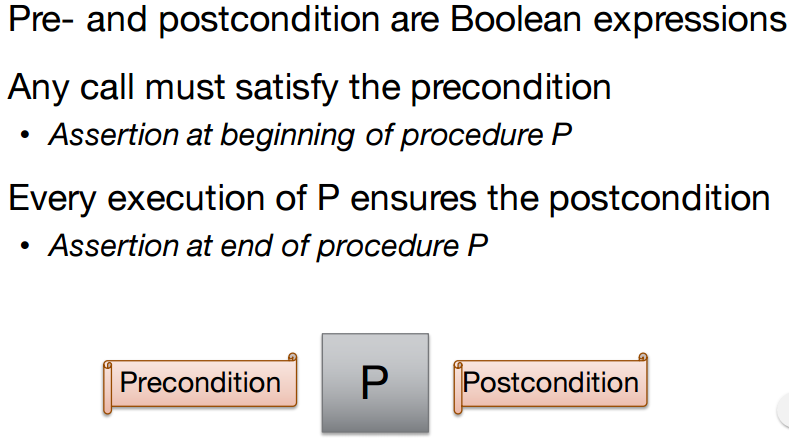




Robust Code

* Stable, no random crash
* Handles corner cases
* Reacts gracefully to unforeseen events/inputs

**Contract**   

**FPA**

FP = UFC \* TCF

* Can use to estimate 2 person-day per FP

No 1 formula

International function point user group (IFPUG)

* Maintains counting practices manual

Data Element Type (DET)

File Type Reference (FTR)

Record Element Type (RET)

Counting EIs:  
For each EI:

* no . of DETs
* no. of types of files

EO: intent to present info to user through logic

Counting EOs:

* Count involved DETs and FTRs

EQ: present info to user through retrieval of data

ILF: user-identifiable info

Counting ILFs: Count the number of DETs  
(sub groups) called Record Element Types

EIF: maintained outside the boundary of app

Functional Vs non-Func Req

FPA analyse costs, time, value of tools, check KPI

Other approaches: Use cases, User Story