SOEN 384 -Midterm Review

Lecture 1 [Intro to Software Project Management (SPM)}:

Project

- 1st time planned effort
- Start and end Date
- Team
- Schedule & budget
- Roles, responsibility and authority assigned

Project Management

- Set of rigid methods

Software Project Management

- Diff dev processes
- Diff types of software (large, real time, imbedded)
- Diff project size

Roles of Software Dev Managers

- Understand business needs + constraints (budget, schedule, scope…)
- Negotiate commitments (scope, budget, schedule)
- Lead in planning
- Communicate goals, plans and status to all stakeholders + execute + identify discrepancies

Measurement:
Process by which numbers or symbols are assigned to things in the real world

Management:
Planning and coordinating the work activities of others so they achieve goals

Why Project Management?

- Project on track? Can it be finished?
- Common problems: Acceptance of impossible constraints
  - Increase in churn, Budget overrun, fully lose control

McCabe Cyclomatic Complexity (Number of linearly dependent paths in code to test)
Monitors logic-based code coverage and produces untested test conditions so that test effectiveness can be incrementally increased

Project Measurement

- *Reasons:* Target setting // Planning // Attention Directing // Performance Evaluation // Improvement tracking // Decision Making

Software Quality Control

- Procedures to ensure that a software product will meet its quality goals (requirements)

- Involves a series of inspections, reviews and tests
- Work products must have defined and measurable specifications

Quality Control Techniques (w/o code execution)

- Software Review
  - Informal review (during a meeting with project personal, managers & customers)
- Inspection
  - Formal (very) peer review, following a process to find defects

What is Quality?

- Quality Design
  - Encompasses requirements, specifications, and system design
- Quality Conformance (implementation)
  - The degree in which everything follows a procedure *(ie. How implementation follows a design)*
- User Satisfaction = compliant product + good quality + delivery w/in budget & schedule

4 Major Activities of Software Project Management

1. Planning & Estimating *(ie. Prepare schedule and budget + identify work activities)*
2. Measuring and Controlling *(ie. Requirements, quality and productivity, product evolution)*
3. Leading and Communicating *(ie motivating, coaching, communicating with management)*
4. Managing Risk *(ie. Identify and confront potential problems)*

Managing vs Leading

- Managing = quantitative aspect => measure, control, plan, estimate
- Leading => communicating, inspiring, coordinating

Success Criteria

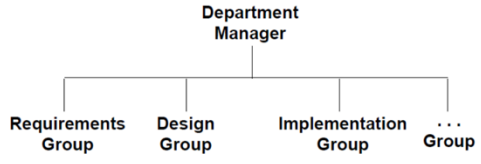
- On time product delivery and within budget, satisfies client, easy to modify

Challenges

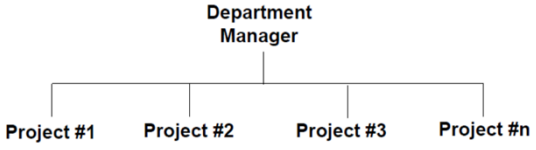
- Scope
- Budget
- Resources
- Technology
- Delivery date

Organizational Structure

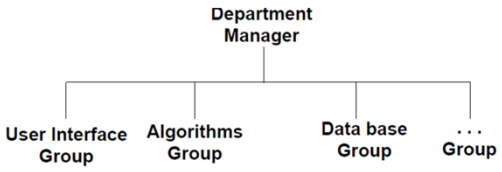
Process-Structured Functional Organization



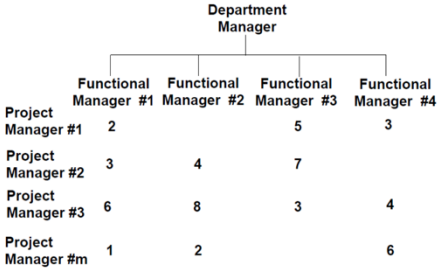
Project-Structured Organization



Product-Structured Functional Organization



Matrix-Structured Organization



Lecture 2 [Establishing the Project Foundations]:

Requirements Management

- Concerned with baseline control of requirements + maintain consistency (effort, budget)

Foundation Elements for Software Projects

- **Product Foundation**
  - Operational Requirements *(user view of system)*
  - System Requirements and Architecture *(hardware, software)*
  - Design Constraints *(predetermined design decisions)*
  - Software Requirements *(internal technical specifications)*
- **Process Foundation**
  - Worklflow Model *(Managerial work activities + product)*
  - Development Model *(Technical work activities + product)*
  - Contractual Agreement *(Btw supplier and acquirer)*
  - Project Plan *(roadmap)*

Scope Management

- Time *(soft boundary ..subject to change)*
- Resources
- Functionality *(functional size of system features)*
- *SUM of (Effort)/(R *T)* ***–Project Scope must be below 100% else OVERSCOPE and reduce it - baseline***
  - *Effort = person hours*
  - *T = available time*
  - *R = # of developers*
  - *SUM of Effort < = R*T*
  - *Budget = SUM Effort * Salary (+overhead)*
  - *Use baseline – Assign Priority for each feature (critical, important, useful)*

Lecture 3 [Plan and Planning 1]:

Requirements Misinterpretations (defects)

- Noise // Silence // Over-Specification //Contradiction // Ambiguity // Forward Reference//Whishful Thinking

- Write requirements from user's point of view

Requirements and Plans

- Requirements => product specification
- Project Plan => process specification
- They are crossed referenced

Lecture 4 [Plan and Planning 2]:

Selecting Development Model

- User-Intensive => Prototyping to clarify operational req and provide info for design interface
- Imbedded System => Need participation of you and technical leader on the System Eng Team
- Staged Delivery => Incremental Build Strategy **(whenever possible)**
- First of its kind => Evolutionary Dev Strategy
- Agile Process => Dev and ongoing enhancements -where req are changing rapidly

Plan-Driven Approach

-Presence of formal contractual agreement
-For large, complex projects internal to an organization

Plan for Supporting Process

- Configuration Management (Work products => version control)
- Verification and Validation (Joint customer, developer reviews)
- Documentation
- Quality Assurance
- Review and Audit
- Problem Resolution
- Subcontractor Management
- Process Improvement

Agile

- Fewer than 10 developers
- Work as one time (product owner, developer, manager)
- Timeboxed (fixed iterations)
- User Stories
- Individual over process, working software, customer collab, responding to change (Manifesto)
- Planning Onion Approach **Changing plans does not mean we change the dates

Plan

- Release Plan
  - Longer planning horizon (months)
  - User Stories
  - Story points to estimate or ideal days
- Iteration Plan (*do not allocate tasks during iteration*)
  - Shorter planning horizon (weeks)
  - Tasks
  - Ideal hours

Lecture 5 [Estimation Techniques (1)]:

Goal Estimation

- Determine parameters which will provide a high level of confidence you will be able to deliver within the bound constraints
    - Product Feature/Quality Attributes/Effort/Schedule/Budget/Technology/resources

Effective Cost-Estimation

Provides a sound baseline foreach project phase and activity – For planning and control

Effective Planning and Control

Provides useful data for cost estimation model study

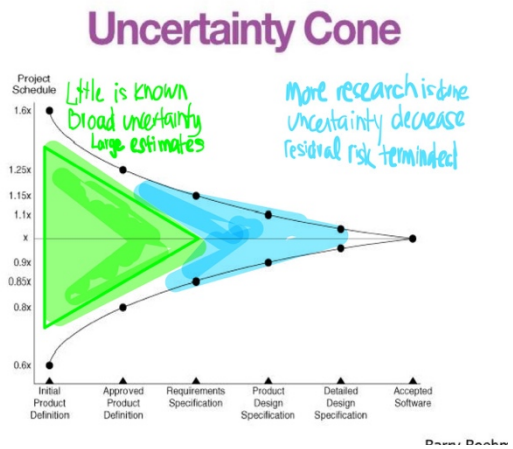
Fundamental Principle Estimation

- **Fundamental Principle #1 Estimation**
    - Projection from the past -Capture historical data
    - Future captured by the requirements for the software to-be + constraints

- **Fundamental Principle #2 Estimation**
    - Estimates based on
    - Assumptions -Statement taken as true w/o being verified
    - Constraints -Externally imposed condition that must be observed

- **Fundamental Principle #3 Estimation**
    - Projects must be re-estimated periodically as it grows and parameters change

Cone of Uncertainty



Types of Estimating (2)

- Experienced-Based
  - Estimation by Analogy
    - Estimate new projects by compare to similar past ones (*decompose them*)

Subsystem	Actual Size of AccSellerator 1.0	Estimated Size of Triad 1.0	Multiplication Factor
Database	10 tables	14 tables	1.4
User interface	14 Web pages	19 Web pages	1.4
Graphs and reports	10 graphs + 8 reports	14 graphs + 16 reports	1.7
Foundation classes	15 classes	15 classes	1.0
Business rules	???	???	1.5

    - *Find factor* and this becomes your size and estimate ratio

  - Estimation by Expert Judgement /Delphi Estimation
    - Estimation is done through multiple iterations
    - 1 Coordinator gives info to many experts on which to base estimates
    - Team members provide their individual effort estimate of task
    - Process terminates when estimates stabilize (3-4 rounds)
    - Create Best and Worst Case estimates
    - ExpectedCase = [BestCase +(4*MostLikelyCase)+WorstCase]/6
    - Decomposition and Recomposition -Law of Large Numbers

- Parametric Models (top-down)
  - Theory and Regression Based
    - Effort = staff-months
    - Time = schedule in months
    - Size = product size
    - a, b, c = constants
    - Small Projects =>Effort = a * Size + c (2-3 people teams)
    - Large Projects => Effort = a * Size^b

Lecture 6 [Measurement & COSMIC]:

***Rule**: Size of software solution is always at least as large as the size of the problem*

COSMIC Measurement Procedure

Functional Size

- Data Movement Types (4): Entry (E), Exit (X), Read (R) and Write (W)
- Each data movement
  - Counted once within the functional process
  - Does not involve another data movement type
- Function Points (CFP) -Min 2 (always need an Exit (X) at the end)

Functional Process
-Elementary component of a set of Functional User Requirements, comprising a unique, cohesive and independently executable set of data movements.

- Independently executable
- Triggered by an event

Lecture 7 [Software Measurement]:

Use of measurement (collect, validate and analyze)

- Understand
- Control
- Improve

Theory: Supposition supported by experience and observations
Proposition: Claims which are assumed to be true
Hypothesis: Proposed explanation for a phenomenon. Must be testable

Levels of Measurements

- Nominal Scale
  - Sorting elements into categories with regards to a certain attribute
  - Example: Place people into different religion folders
- Ordinal Scale
  - Classified in respect to a degree (ranked)
  - Example: Place people in to classes (1st class, 2nd class…)
- Interval Scale
  - Indicates exact difference btw measurements (**NO** * or /)
  - 8 more crashes OK but **NOT** 8 times as more crashes
- Ratio Scale
  - Highest level of measurement – When absolute 0 is located on interval scale it becomes a ration scale (* and / are allowed)

Reliability: Consistency of a number of measurements taken with the same measurement method
Validity: Whether the measurement or metric really measures the thing I had to

Measuring Software

3 Ps

- Product (Actual software system, documentation and deliverable)
  - Measured Aspects: Size, Functionality offered, Cost, Quality Attributes
- Process
  - Lifecycle used? Deliverables produced? How can process help to produce faster?
  - Involves analysis of the way a product is developed
- People
  - Involves analysis of people, how fast they work? VERY controversial

Product Metrics

- Size -> LOC or KLOC
  - Issue: Same system, different languages = different LOC readings
  - Issue: Same system, different developers = different LOC readings
  - Issue: Wait until system is implemented, not good for cost and effort estimation

- Function Points (instead of LOC)
  - Measures functionality offered by the system
  - Can be calculated before a system is developed
  - Language and developer independent
  - Five major components: External inputs (x4), external outputs(x5), logical(x10) and external(x7) internal files and External inquiries(x4)

Lecture 7 [Software Measurement (1)]:

- Physical LOC -Includes everything from the start until the end (incl blank lines and comments)
- Logical LOC -Number of statements in a program

McCabe's Cyclomatic Complexity Metric

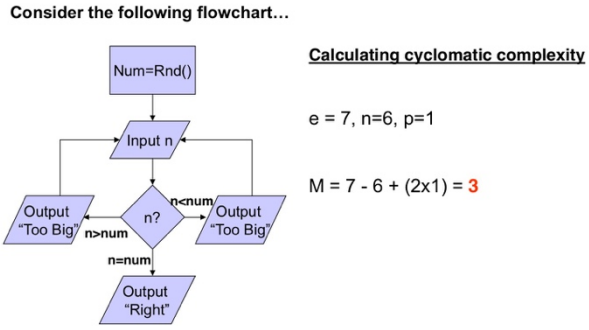
$$M = V(G) = e - n + 2$$

where

V(G) = cyclomatic number of Graph G

e = number of edges

n = number of nodes



The number of linearly independent paths of the following program segment is:

```
dx=0.1;
N=50;
for {i=0; i<=N; i++}
      {z[i]=sin(x)/ (x-2.5); x=x+dx}
```

a) 4
b) 2
c) 5
d) 1
e) None of the above

Any for/while, if statement switch statement = then add 1 Thus, we get (2)

McCabe's Essential Complexity EV(G) – Flow graph reduction -Reduce to one node BEST IS 1

Cyclomatic Complexity

- Code Testability
- If HIGH, lead to impenetrable code, higher risk, difficult to test and understand

Essential Complexity

- Code Structure
- *Measures degree which code module contains unstructured constructs*
- If HIGH, increased maintenance costs with decrease code quality

- Coupling: Relationships among the software components
  - Evaluating the design partitioning of a system
  - Best methods: message and data coupling
- Cohesion: Relationships within software components
  - Measure binding of the elements within the module
  - Best binding: Object and functional -NO side effect

- IDEAL: Low coupling and high cohesion to control software complexity

Lecture 8 [Software Measurement (2)]:

Software Requirements must be:

- Comprehensible
- Detailed
- Prioritized
- Stable
- Testable
- Feasible
- Traced

Requirement Verification Techniques

- Inspection //Traceability//Prototyping//Analysis//Validation Test

Review Types

- Inspection: To find defects, train participants, assign roles, record participants, analysis
- Walkthrough: To communicate, Developer typically presents

Quality Requirements (all finish in "….ility")
Safety// Security reliability// availability//usability//maintainability

Failure occurs when software does not satisfy its requirements. Results of one or more defects. Defects are human errors (omission or commission errors)

Severity Level of Failure: Catastrophic, Critical, Sever, Marginal, Minor

Types of Rework: Evolutionary, Retrospective, Corrective
**Rework of any baseline work product that must be changed because of a mistake should be counted as a defect*

Failure Rate (Rate of defect over time)

$$\lambda = \frac{R(t_1) - R(t_2)}{(t_2 - t_1) \times R(t_1)}$$

where,

t_1 and t_2 are the beginning and ending of a specified interval of time

R(t) is the reliability function, i.e. probability of no failure before time t

Calculate the failure rate of system ***X*** based on a time interval of 60 days of testing. The probability of failure at time day 0 was calculated to be ***0.85*** and the probability of failure on day 60 was calculated to be ***0.2***.

$$\lambda = \frac{R(t_1) - R(t_2)}{(t_2 - t_1) \times R(t_1)}$$

$$\lambda = \frac{0.85 - 0.2}{60 \times 0.85}$$

$$= \frac{0.65}{51}$$

$$= 0.013 \quad \text{Failures per day}$$

Mean Time Between Failure (MTBF)

- Measurement for Reliability
- Useful in safety-critical applications *(ie. air traffic control)*
- Need to use the *failure rate value (above), inversed*

Consider our previous example where we calculated the failure rate (λ) of a system to be 0.013. Calculate the MTBF for that system.

$$MTBF = \frac{1}{\lambda}$$

$$= 76.9 \text{ days}$$

This system is expected to fail every 76.9 days.

Defect Calculations

- percent of requirements defects detected during the requirements phase:
  (RDr x 100) / RDt
- percent of design defects that escape the design phase:
  [1 – (DDd / DDt)] x 100
- percent of total defects detected by users:
  (OSt x 100) / TOTAL

Detect Containment: *Calculated by determining the percent of defects that are detected during the work activity that generates the work product*

- Goal: >75% @ each stage of our work

Maintainability

- Availability is calculated in terms of Mean Time To Failure (MTTF) plus Mean Time To Repair (MTTR)
  MTBF = MTTF + MTTR
- Availability, A:
  A = MTTF / (MTTF + MTTR)
- For example, if MTTF = 8 hrs and MTTR = 2 hrs
  A = 8 / (8 + 2) = 8 / 10 = 0.8
  it is 80% probable that the system will be available at any particular time

- Maintainability is based on Mean Time To Repair
- M = T / MTTR – IF MTTR = 2hrs, then probability it will be accomplished in 1 hour
  - M = 1/2

GQM to Choose Product Measure

- Goals (ie. Reduce defects in software Dev)
- Questions (ie. How many defects are there?)
- Metrics (ie. Total number of defects found in software dev)

Lecture 9 [Classical OO Measurements)]:

- WMC (Weight Methods per Class)
  - Code size
  - Predict effort required to re-write or modify class
- CBO (Coupling Between Object classes)
  - Coupling
  - Strong coupling makes it hard to understand and test classes
- LOCM (Lack of Cohesion of Methods)
  - Cohesion
  - If % is low, simplicity and high reusability
  - If % is high, class candidate for refactoring and could be split in to subclasses (low cohesion)
- DIT (Depth of Inheritance Tree)
  - Inheritance
  - If >6, increases the testing effort
  - If <2, poor exploitation of OO
- NOC (Number Of Children)
  - Inheritance
  - Classes with more children, more complex to modify

Lecture 10 [Classical OO Measurements 1)]:

How to Perform Measurement

- Goal: analyze code quality given
  - Extract -collect code quality measure
  - Evaluate -analyze data
  - Execute -improve code

Code quality trend

For Cyclomatic Complexity: measurement data for the modules, calculate averages per milestone and plot on a trend graph.

Lecture 11 [Project Planning Techniques)]:

Plan Driven Approach

- When formal agreement (btw acquirer & supplier)
- When large, complex projects (internal to an organization)

Scenarios for Developing a Project Plan

- Determine project feasibility
- Estimate and commit schedule, budget and resources
- Determine product's characteristics

Project Planning Tools and Techniques

- Architecture Decomposition View
- WBS
- Work Packages
- Activity Network
- CPM (Critical Path Method)
- PERT (Program Evaluation and Review Technique)
- Gantt Charts
- Resource Loading Histograms??

Project Planning Strategy

- *First Pass*: planning done without constraints
- Initial Planning Activities:
    - Develop Architecture Decomposition View (ADV)
        - Allocate requirements and interfaces for each element
    - Develop work breakdown structure of the elements previously mentioned
    - Develop work packages for the tasks in the WBS
    - Define objectively measurable milestones schedule
    - Determine Critical Path
    - Perform PERT estimate of project duration
    - Identify quantity of resources needed (when and what)
    - Prepare estimate: optimal effort, cost, schedule and resources
    - Negotiate with customer in order to satisfy all constraints

Design Product Structure

- Conway's Law
    - Software structure = build's team structure
- Fairley's Corollary
    - Software architectural hierarchy = structure of the team's work assignment

Some Quality Attributes may apply to:

- 1 individual, several or all elements of the ADV

Work Breakdown Structure (WBS)

- Tree structure or indented list
- Hierarchy partitions so that activities can be separated and assigned to different groups/indivi

- TO CALCULATE EFFORT: SUM (DURATION * #Staff)

Side Note

- ADV – NOUNS to denote things
- WBS -VERB phrases to denote work
- Work activities to develop ADV elements are embedded in WBS
  - Example: ADV -Validator → WBS -Build Validator
- WBS elements
  - Lower level elements = tasks (smallest unit of management)
  - Higher level elements = activities (aggregation of tasks and subordinate activities)
  - Work packages used to document tasks

Tools and Techniques for Schedule Development: CPM

- Critical Path Methods (CPM) → float to determine which activities have the least scheduling flexibility
- Output: Expected schedule of tasks with projected start times, (essentially the box info)

WBS vs Activity Network

- WBS: tree structure "is part of" relation -> activities and tasks (aggregation relation) **Hierarchy
- Schedule Network: "is-preceded-by" relation -> tasks (time-ordering relation) **Sequence

Gantt Charts

- To plan all tasks that must be completed for the project
- Distributes estimated effort across the planned project duration

Lecture 12 [Wrap Up)]:

Defect rate during testing = Defect rate in the field

Defect rate during *testing same or lower than previous* release

- Testing of current release deteriorate? If YES → more tests ELSE →Ok

Defect rate during testing higher than previous release

- Did we *plan for and actually improve testing effectiveness*? If YES →OK ELSE→more testing

People Metrics

**People don't like to be measured

Interesting for managers because:

- Financial purposes
- Project management purposes
- HR problem identification

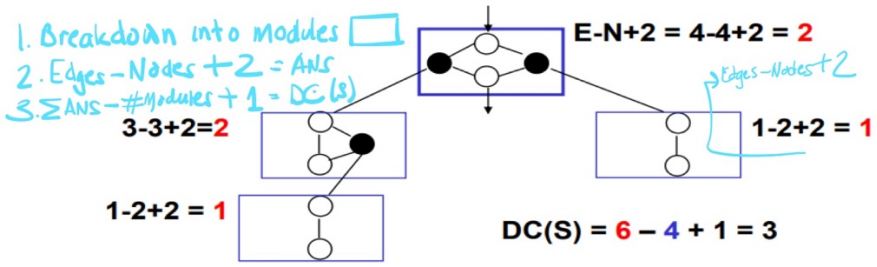
The metrics used:

- Cost per Function Point
- Mean Time to develop Function Point

- Defects produced per hour
- Defects produced per Function Point

FROM CHAPTER: Software Measurement (1)

Software Design Complexity: *Structural complexity of a collection of software modules*



Lecture 13 [Project Planning Techniques]:

SAME AS LECTURE 11

Measuring and Controlling

- Collecting
- Validating
- Analyzing … project status information

Controlling

- Status of the work products
  - Quantity and quality of work products
- Status of the development process
  - Schedule, budget, resources, risk factor

Developing Project Plan

- Determine feasibility of the project
- Estimate and then commit to the schedule, budget and resources
- Determine the characteristics of a product

Lecture 14 [Measuring and Controlling Work Processes]:

Complete a project earlier than the critical path (CP)?

- Doubling resources -> team member training, coordinating task split, integrating contribution -> only productivity will increase not reduce time
- Shortening CP -> lead to a different series of activities becoming part of CP

Areas to Monitor

- **Schedule**
  - Milestones
  - Action Items
- ***Cost***
  - Expenses
  - Staffing
  - Effort

Activity Tracking Indicators

Planned Value (PV): For a given task, **Effort** planned for that task translated into **cost**

Earned Value (EV): % of PV of work actually completed by a given point in time on the project schedule

- EV **higher** than PV (but can be lower than AC)➔ Project ahead of schedule
  - More activities completed than planned at this time
- EV **lower** than PV (but above AC) ➔ Project Behind Schedule

Actual Cost (AC): Total costs on tasks that have actually been completed at a given "…"

- AC **higher** than PV AND EV ➔ Project Over Budget
- AC **lower** than EV ➔ Project Under Budget
  - Spending less money than the earned value of work being completed
  - Maybe understaffed but tasks are being done more efficiently

Performance Indicators

Cost Variance (CV): EV – AC ➔What was actually paid for work performed

Schedule Variance (SV): EV – PV ➔What was indented to be paid for work accomplished

- Positive → Ahead of schedule
- Negative → Behind schedule

Cost Performance Index (CPI): EV/ AC ➔ What should have been paid for work performed
SHOULD **NOT** exceed +- 15%

- Near 1.0 → Within its defined budget
- Less than 1 → Cost of completing work Is higher than planned
- More than 1 → Cost of completing work is less than planned

Schedule Performance Index (SPI): EV/ PV ➔ Rate of progress (efficiency using scheduled resources) SHOULD **NOT** exceed +- 15%

- Near 1.0 → Efficient execution of the project schedule
- Less than 1 → Not Favorable
- More than 1 → Favorable

Status of the project to-date

- The whole division ➔ Derived measure
- Tasks completed and planned ➔ Base measures

$$\frac{\text{Tasks completed}}{\text{Tasks planned}} * 100$$

****Count** number of completed and planned activities

Software Measurement Plan (SMP)

Section 1

- Purpose (ie Importance of SMP to the success of the project)
- Organizational Description (ie scope affected by SMP)
- References

Section 2

- Organizational Measurement Roles and Responsibilities (key positions, roles and responsibilities →Software Project Manager – Data Analysis Team – Development Team)

Section 3

- Measurement Information Needs (AKA Operational Goals)(ie List of business goals and subgoals)
- Measurement Information Models (ie Needs are refined into precise indicators, success criteria and quantifiable measures)

Section 4

- Data Collection and Storage Procedures (ie ensure reliability and accessibility of collected data)
- Schedule (ie work breakdown of main measurement by milestones -Collect, analyze data & report)

Lecture 15 [Managing Project Risk]:

Risk: *Combination of the probability of an event and its negative consequence*
Risk Category: *A class or type of risk (Technical, legal, organizational, safety, economic, schedule, cost)*

- *Technology Risks*
- *People Risks*
- *Organizational Risks*
- *Tool Risks*

- *Requirements Risks*

Risk Management Process

- Risk Assessment:can be executed at beginning of project dev and reassessed at iteration
  - **Risk Identification** ➔ Outcome: collection of risk items
  - **Risk Analysis**
    -Likelihood of occurrence
    - ***Qualitative: Scale*** (Low, Moderate, High)
    - ***Quantitative:*** Probability of occurrence

    -Impact on project, product of each risk item
    - ***Qualitative:*** Scale (Low, Moderate, High)
    - ***Quantitative:*** where data is available

    *Impact vs Likelihood ➔ Risk Exposure = Risk-Probability * Risk Impact*

  - **Risk Prioritization**
    - Where to focus risk mitigation efforts
    - Combine likelihood and impact to get priority (as seen above)
- Risk Control
  - **Risk Planning:** Can be executed at beginning of project dev and reassessed at beginning iteration
    - Type of strategies:
      - ***Acceptance***: Not to change project plan or unable to identify response strategy➔
        - Develop contingency plans
        - Identify risk-trigger points
        - Periodic review of risks and trigger points
        - Use contingency allowance (time, budget, staff)
      - ***Avoidance***: Change project plan to eliminate the risk
        - Use familiar contractors
        - Reduce scope (eliminates high risk)
        - Add resources or time
        - Use familiar approaches
      - ***Risk Transfer:*** Shifting consequences of a risk to a 3rd party
        - Use of insurance and performance bonds
        - Use of warranties and guaranties
        - Use of contracts (liability transfer)
        - Use of fixed-price contract with subcontractors
      - ***Mitigation***: Reduce probability or consequences of an adverse risk
        - Adopt fewer complex processes
        - Plan additional testing
        - Use more reliable / stable vendors
        - Use a prototype in dev process
        - RRL ➔ risk exp RE before – risk exp. RE after / cost of reduction
        - RRL must be greater than 1
    - Contingency Measures

- **Resolution:** Can be performed throughout the project
  -
- **Risk Monitoring:** Can be performed throughout the project
  - ***Response Audits***
    - Examine and document the effectiveness of risk response
    - PM should ensure it is done regularly
    - Clear format and objectives
  - ***Risk Review***
    - Identification of new risks
    - Reassessment of current risks
    - Closing outdated risks
  - ***Additional Risk Response Planning***

Integrating Risk Management into Project Planning: Seven Basic Steps ??? (6)

- Customer Requirements
- WBS
- Task List with estimation linkages and resources
- Risk Items
- Network Diagram
- Gantt Chart

Risk-Base Scheduling

- Use PERT analysis to re-estimate duration of tasks considering the effect of risk
- (Opt + 4 x MostLikely + Pessimistic) /6

Project risk item	Risk category	Suggest a risk response planning technique. Justify your choice.
A. New tool we don't have enough experience with	Tools risk, Technology risk	**Mitigation:** Plan for hiring an expert to train the staff
B. Volatile requirements	Requirements risk	**Mitigation:** Use prototypes in the development process
C. Adopting new information management system which does not meet space requirements	Technology risk	**Avoidance:** Adopt another MIS that meets space requirements

$$\text{Risk reduction leverage } (RRL) = \frac{RE_{before} - RE_{after}}{\text{Cost of risk reduction}}$$

- Project required the use of a special purpose computer for testing the software that your team is developing. You are not sure that the computer will arrive in time to start testing on schedule.
  - Your purchasing department has estimated that there is a **70%** probability that delivery will be delayed by 2 months, which will delay your team's delivery by a comparable amount.
  - The contract that you have with the customer reduces the customer's cost by $200,000 for each month of delivery delay. Therefore, if the computer is delayed by 2 months, your delivery will be delayed 2 months and it will cost you **$400,000** in reduced revenues from your customer.
  - Your purchasing department has suggested that if you offer the computer vendor a fee of **$200,000** for on-time delivery, the probability of a 2-month delay will be **reduced to 20%**
- Should you seriously consider this option?

- **Solution:**
  - RRL = (70% * 400,000 – 20% * 400,000) / 200,000 > 1
- **Conclusion:**
  - RRL > 1 indicates that this is a cost effective risk measure

Risk & Software Cost Estimation

- Sources of risk in software estimates
  - Sizing software (COSMIC)
  - Using Cost and Effort Estimation Models

Examples

- **Risks related to System Definition**
  - Inconsistency in the understanding of a project's size and needs or changing project conception
  - System is pushing the limits of tech
    - Critical techs have **not been proven**
    - Key requirements or functions incl. in program risk
- **Risk related to System Development**
  - CP activities are unknown or unresolvable → **no float/slack**
  - Inadequate monitoring, mitigation and reporting
  - Info about risks, probability of occurrence and consequence → **unavailable**

Estimation Process Risk: Mitigation

- Pros and Cons of various cost-estimation methods

Lecture 16 [Agile Estimation and Planning 1: Estimating Size]:

Goal: Build features, resources and schedule
Planning Process = Quest for value → reduce risks, reduce uncertainty, support reliability of decisions..

A few stats:

- 2/3 project cost overruns
- 64% features are rarely or never used
- Average project exceeds its schedule by 100%

What to watch out for:
Estimates are probabilities, hence commitments cannot be made a probability
Deal with uncertainty? → short iterations

----------Agile----------

The onion planning (Agile Planning):
Day < Iteration < Release < Product < Portfolio < Strategy
Changing the plan does **NOT** mean we change the dates

Agile Methodology Principles

- *One Team*: product owner, customer, developer…
- *Short iterations*: Timebox
- *Frequent deliveries*: Business priorities and User stories

User Story
"As a user, I want to be able to book a flight"

Condition of satisfaction aka user story acceptance tests
"A user who cancels more than 24hours in advance gets a complete refund"
-Drives both release and iteration planning

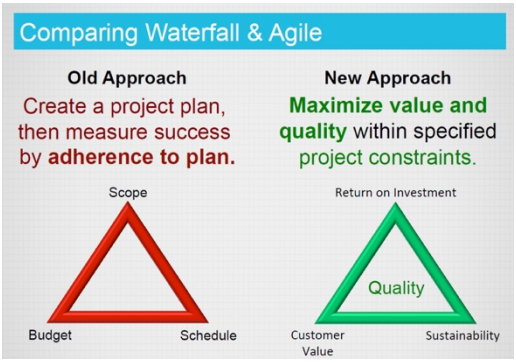
Estimating duration → estimate project size (Ideal time & story points)

Desired Feature < Estimate Size < Estimate Duration < Schedule

- **Ideal time vs elapse time**
  - Ideal time: The time we estimate it will take to complete a story uninterrupted (consider this an estimate of size).
  - Elapsed Time: The **actual amount of time** it will take to complete the story.
- Story points (**0**-10)
  - Most common but ***relative***
  - Combination of size and complexity of work
  - Estimation scales: Beware of law of diminishing returns on time spent of estimations
    - Fibonacci: 1, 2, 3, 5, 8, 13
    - 1, 2, 4, 8 (doubles)
    - T-Shirt: XS, S, M, L, …
  - ***How to estimate size:*** Planning Poker (loosely based on wideband Delphi)
  - **Advantages of Agile estimation (3)**
    - Forces to use relative estimation
    - Units in which we can add together
    - Separates estimation of effort from estimation of duration
  - Size, velocity & duration
    - ***Velocity AKA rate of progress***
      - No partial credit for partial user stories
      - Story points per iteration
    - [Story Points] / [Story Points per Iteration] = [# of iterations = derive duration]
    - Each developer should be able to accomplish 1 task per day
  - Re-Estimate
    - When a story's relative size has changes
  - Splitting User Stories
    - When it doesn't fit into a single iteration
    - How? Based on CRUD, remove cross-cutting concerns -concurrency and security
  - Release Planning
    - Inputs
      - Velocity
      - Project Length
      - Prioritized Product Backlog

1. Who should participate in the planning poker on your project? Everyone?
2. Suppose a project has 90 story points in total. Over the first 4 iterations the team has achieved a velocity of 6, 20, 10, and 17. You are asked when the project will be done. **What do you say?** Total completed: 53 in 4 iterations
   Average ⇒ 53/4 = 13
   So: 90/13 ≈ 7 ??

Lecture 17 [Agile 2: Planning for value, agile scheduling and tracking & communicating]:



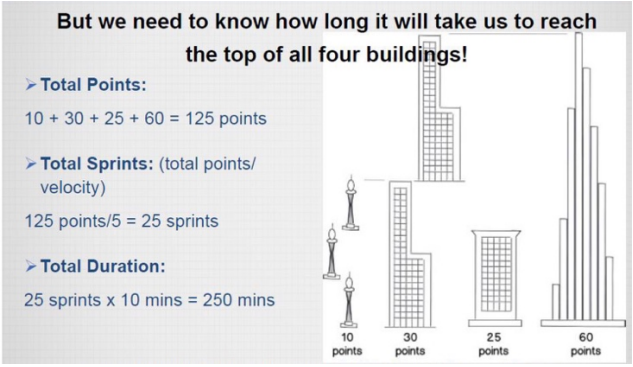
Scrum Terminology

- Epic
  - Large, high level user story
  - Not to be completed in single iteration
- User Story
  - Short, simple description…
- Theme
  - Set of related User Stories

Agile Estimation

- **Relative Estimation**
  - Compare relative effort for completing user story to a previously estimated one
  - Applied at a product backlog level rather than at spring backlog level
- **Estimation Basics**
  - Estimate size not duration
- **Measure of Size**
  - *Traditional*:
    - Lines of Code
    - Function Points
  - *Agile Estimation*:
    - Story Points (relative and cannot decay)
    - Ideal Days (absolute time, 4-5 hours avg)
- **Story Points Estimation Influenced by:**
  - Complexity
  - Functionality
  - Current knowledge
  - Level of effort
    - Analysis
    - Coding
    - User Acceptance Testing (UAT)
    - Documentation (design, user guide, etc)
  - Risk

- Calculate Estimate (use 10 min timebox)



- Sprint Burndown (ideal and current)
  - Best is when: Below ideal !

<u>Definition of Agile Estimating and Planning</u>
Date Driven: Variable Scope and Fixed Time (working backwards to today)
Scope Driven: Fixed Scope and Variable Time (how many sprints to delivery)
Budget Driven: Scope?? and Time?? (until money is gone)
-Sum product backlog, estimate velocity as a range, ***[Size of product Backlog / Velocity Range]***

Best and Worst Case ➔ Forecasting

Forecasting, is basically a prediction or projection about a future event, depending on the past and present performance and trend. **MIGHT**

Planning, is the process of drafting plans for what should be done in future, based on the present performance **plus** expectations. **WILL**

Agile Program Scope
- Customer Satisfaction
- Speed to Market
- System Quality
- Project Success