Capstone Design in CSE

Feasibility Study

Current Stage on Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering

→ **Week 2 (Homework in Week 1)**

3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

Analysis and Design

- System Analysis
  - Planning and Scheduling
- Software Design

Planning

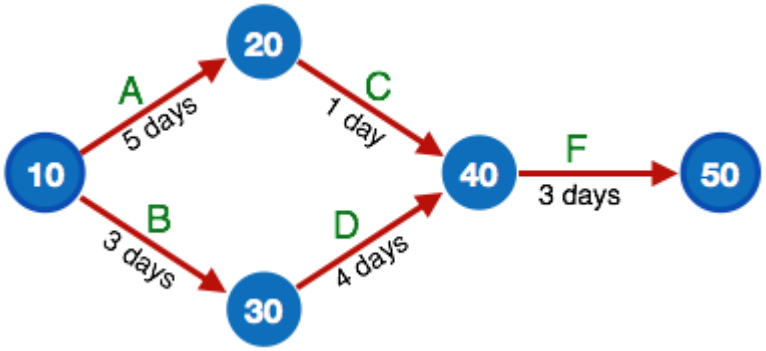
- Project planning
  - Scope management
    - Define the scope.
    - Divide the project into various smaller parts.
  - Project estimation
    - SW size estimation (e.g., # of components, lines of code, and # of function points)
    - Effort estimation (e.g., Man-hour)
    - Time estimation
    - (Cost estimation)

Scheduling

- Project Scheduling
  - roadmap of all activities to be done with specified order and within time slot allotted to each activity
  - Define milestones (expected results for important activities)
  - look for tasks that lie in critical path in the schedule (task interdependency)
- Steps required
  - Break down the project tasks into smaller, manageable form.
  - Find out various tasks and correlate them.
    - You may use a PERT chart
  - Estimate time frame required for each task
  - Divide time into work-units
  - Assign adequate number of work-units for each task
  - Calculate total time required for the project from start to finish.
    - You may use a Gantt chart. (schedule on a weekly basis)
  - Assign tasks for each person.
  - Define criteria to evaluate the progress (for each week and for each team member).

Scheduling

- Useful tool 1: PERT chart
  - PERT (Program Evaluation & Review Technique) chart is a tool that depicts project as network diagram.
  - graphically representing main events of project in both parallel and consecutive way.
  - Events, which occur one after another, show dependency of the later event over the previous one.
  - Events are shown as numbered nodes. They are connected by labeled arrows depicting sequence of tasks in the project.

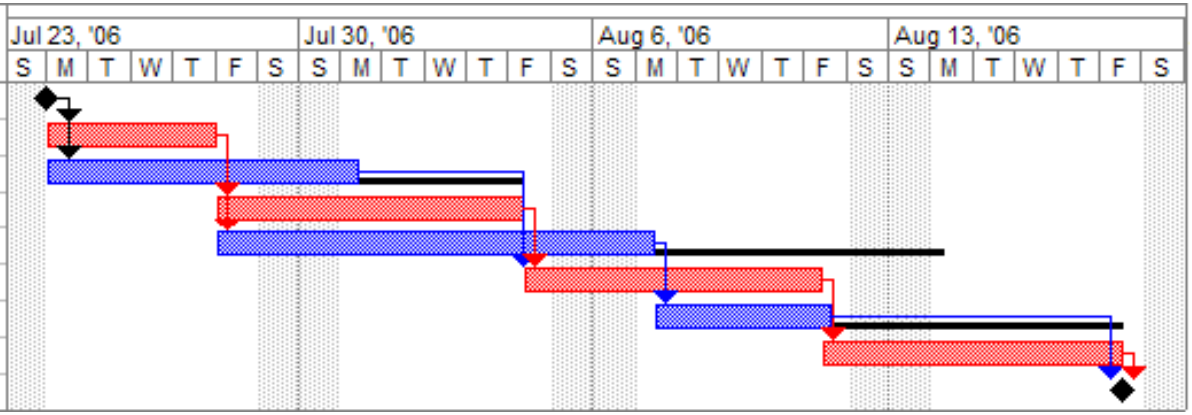


Scheduling

- Useful tool 2: Gantt chart
  - type of bar chart, devised by Henry Gantt in the 1910s, that illustrates a project schedule
  - It represents project schedule with respect to time periods. It is a horizontal bar chart with bars representing activities and time scheduled for the project activities.

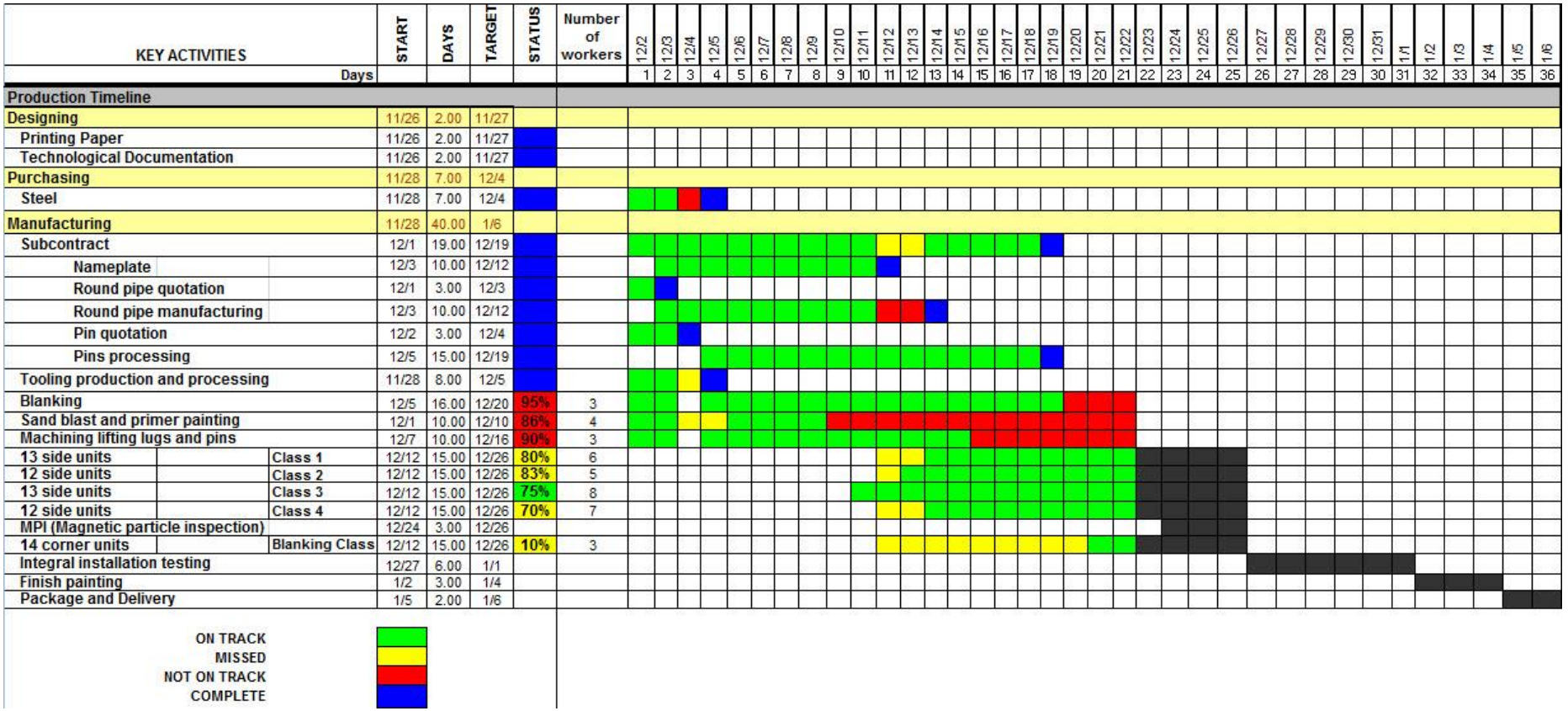
Example 1

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



Example 2

KEY ACTIVITIES			START	DAYS	TARGET	STATUS	Number of workers
Days							
Production Timeline							
Designing			11/26	2.00	11/27		
Printing Paper			11/26	2.00	11/27		
Technological Documentation			11/26	2.00	11/27		
Purchasing			11/28	7.00	12/4		
Steel			11/28	7.00	12/4		
Manufacturing			11/28	40.00	1/6		
Subcontract			12/1	19.00	12/19		
Nameplate			12/3	10.00	12/12		
Round pipe quotation			12/1	3.00	12/3		
Round pipe manufacturing			12/3	10.00	12/12		
Pin quotation			12/2	3.00	12/4		
Pins processing			12/5	15.00	12/19		
Tooling production and processing			11/28	8.00	12/5		
Blanking			12/5	16.00	12/20	95%	3
Sand blast and primer painting			12/1	10.00	12/10	86%	4
Machining lifting lugs and pins			12/7	10.00	12/16	90%	3
13 side units		Class 1	12/12	15.00	12/26	80%	6
12 side units		Class 2	12/12	15.00	12/26	83%	5
13 side units		Class 3	12/12	15.00	12/26	75%	8
12 side units		Class 4	12/12	15.00	12/26	70%	7
MPI (Magnetic particle inspection)			12/24	3.00	12/26		
14 corner units		Blanking Class	12/12	15.00	12/26	10%	3
Integral installation testing			12/27	6.00	1/1		
Finish painting			1/2	3.00	1/4		
Package and Delivery			1/5	2.00	1/6		

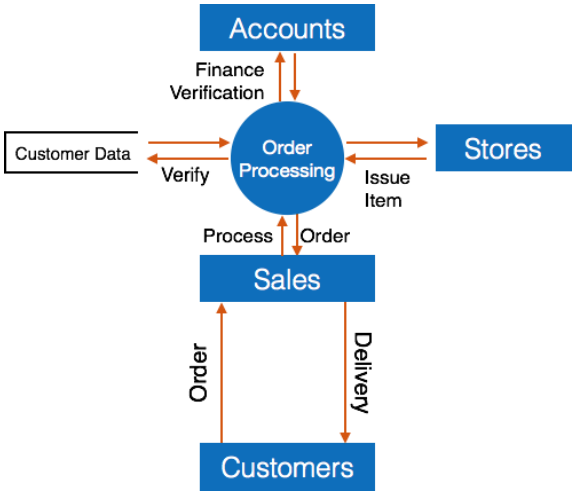


Software Design

- OOP design principles
  - Abstraction
    - Naming and explaining the functionality of SW parts
    - abstract data types
  - Encapsulation
    - Different components should not reveal the internal details
    - tract interface
  - Modularity

Design Tools

- SW analysis and design tools
  - Data Flow Diagram



source: http://www.tutorialspoint.com/software_engineering/software_analysis_design_tools.htm

- UML (Unified Modeling Language) diagram
  - Expresses the organization of a program, especially in OOP SW design.

UI Design

- UI requirements
  - User centric approach
  - easy to operate / easy navigation / simple interface
  - Provide default settings / Provide help information
  - Consistent UI elements / group based view settings
  - quick in response
  - effectively handling operational errors
  - Strategical use of color and texture

4주차 과제 (1)

- 팀별 주제 확정
  - 2주차의 개인별 발표 및, 3, 4주차 팀별 발표 내용과 의견들을 종합하여 팀별 주제 선정
- 제안서 작성
  - Deadline: **9/30 월요일 오후 3시**
  - Technical report의 가이드라인을 준수할 것 (IEEE 프로젝트 계획서 양식 및 **주어진 제안서 양식** 참고)
    - 간단 명료한 단어, 짧은 문장 사용
    - 최대한 많은 그림, 다이어그램, 표를 활용
    - 전체 문서, 장, 절, 문단, 문장에서 중요한 개념이 앞에, 설명이 뒤에 오게(두괄식) 작성: 서두에 요약 있으면 효과적임
  - 반드시 다음 사항을 포함할 것
    - Literature survey
    - 목표의 명확한 정의 (기능, 성능, 상세 구조, 정량 목표)
    - 최종 결과물에 대한 스케치: UI prototype, Flowchart for operations, Data Flow Diagram 등
    - Implementation platform (HW, OS, …)
    - Required components (구성 요소)
    - Available open source libraries or APIs
      - 설치 후 정상동작하는 지 확인 (예) 해당 library 설치 후 hello world 라도 찍어볼 것
    - Available data sources
    - 기술적 사항 이외의 고려사항
      - 사례: 감시 카메라 응용에서 개인 정보 활용에 대한 법적 이슈, 윤리적 문제, 카메라 및 전력, 네트워크 설치 허용 여부 등

4주차 과제 (2)

- 팀별 진행상황 발표자료 upload
  - **9월 30일(월) 오후 3시까지**
  - 제안서 내용을 설명할 것
  - 발표는 팀당 15분

향후 진행 계획

- 발표자료 upload 후 격주 발표
  - 배경 설명은 최대한 간단히 하고, 2주간의 진행 사항을 자세히 설명
  - Gantt chart 기반으로 설명할 것
    - 2주 별, 팀원 별 목표 결과물의 달성 여부 명시
    - 목표 달성 미비 시 문제점 확인
    - 다음 계획 설명
  - 발표는 팀당 15분 + 질의응답 5분