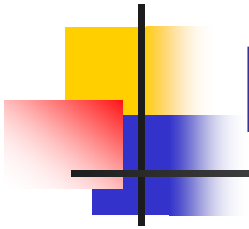
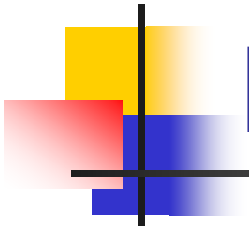


# Putting It All Together

- Covered so far:
  - System reliability
  - Data reliability
  - Timeliness
- Design problem: Design a “safe” robot

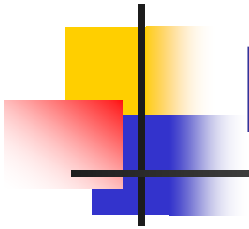


# Requirement Engineering

- “Conceptual integrity is the most important consideration in system design.”

  Frederick P. Brooks, The Mythical Man-Month

- “The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements … No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.”
  - Frederick P. Brooks, The Mythical Man-Month



# Requirement Engineering

- work on requirement engineering decomposes the requirements according to their criticality and complexity. For example:
  - Safety critical (simplest)
  - Mission critical
  - Performance/feature enhancement
  - Optional features and customization
- The dependency relations between levels must be:

# Mission Requirements to Technical Requirements

- "I believe that this nation should commit itself to achieving the goal, before this decade is out, of landing a man on the Moon and returning him safely to the Earth." *John F. Kennedy*
- An outline of high level technical requirements from http://science.howstuffworks.com/first-man-on-moon.htm
  - operate rocket engines and spacecraft in a vacuum and without gravity
  - control huge multi-stage rockets
  - build space suits
  - space walk and moon walk
  - dock and undock spacecraft in orbit
  - get in and out of moon orbit
  - land on the moon and take back off

# Asimov Laws of Robotics

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

# Modern Asimov Laws of Robotics

## When your boss owns the robot

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

# Modern Asimov Laws of Robotics

## When your boss owns the robot

- A robot may not injure a human being (or another robot)
- A robot must protect its own existence, except when it conflicts with First Law
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First or Second Law.

# Modern ~~Asimov~~ Laws of Robotics

## When your boss owns the robot

- A robot may not injure a human being (or another robot)
- A robot must protect its own existence, except when it conflicts with First Law
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First or Second Law.
  - Assume that the robot is on a scientific mission and carries some payload (e.g., a camera) with its own software and start/stop commands.

# A Robotic Mission
## Ancient Mine Exploration

- A farmer was planting a tree in their back yard when the ground caved in uncovering an entry way to an ancient underground maze.
- A research team is assembled. They send a robot into the maze.
- The robot must map the maze while taking pictures of all encountered objects.
- Since this maze is labeled an archeological site, the robot must not run into and damage any objects.
- The maze may include cliffs and traps. The robot must protect itself while in the maze
- Scientists are excited and have software (payload) they want to download to the robot. This payload will execute analytics on data received. It may be buggy, but it needs to be executed in the maze because limited bandwidth prevents the robot from reporting all data outside.
- The robot should complete the mission as quickly as possible.

# Safety versus Performance Requirements

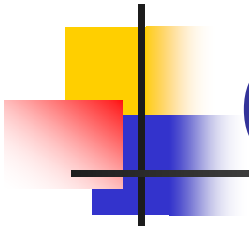
- Critical (safety) requirements
  - …
  - …

What are the hazards? (Environmental, CPU, …)
What are open-loop safe states?

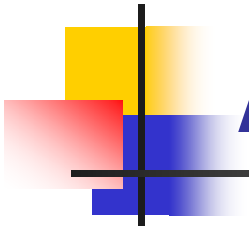
- Mission and performance requirements
  - …
  - …

Requirements on natively supported functions?
Requirements on payload?

# Well-formed Dependencies (Use but do not Depend)



- Consider computing, communication, sensing, timing, …
- How can we protect the safe control component?
  - What are the sensors & actuators needed for by safety control? What if they crash?
  - What are the services that should be exported by safety component?
  - How can the safety component safely USE outputs from mission critical and performance enhancement components?



# Architecture Design

- What are the mission critical requirements?
- What are the safety constraints enforced by safety components and how to work within the safety constraints?
- What are the services from safety components that it can depend upon?
- What is the simple “no frill” method to implement the mission requirements?
- What are the services from mission component that can be USED by safety component?
- What are the services from mission component the performance enhancement component may depend upon?
- Recovery: Should the mission component fails, how can we restart the mission component while the Roomba is being control led by the safety component?