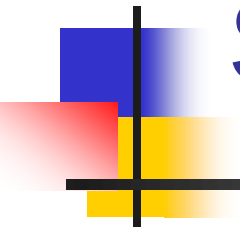
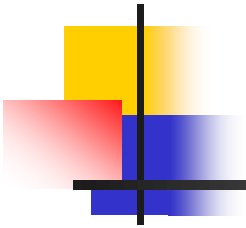


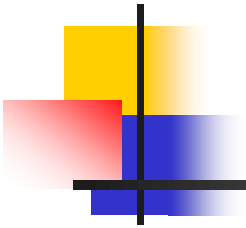
SWE 205 - Introduction to Software Engineering

Lecture 2



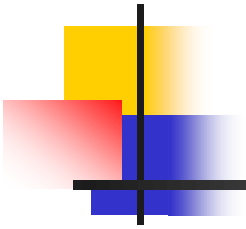
Lecture Objectives

- Legacy Software Systems
- Software Evolution
- Software Engineering Costs
- Software Myths
- Software Engineering Challenges



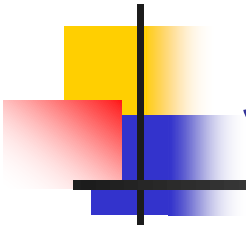
Legacy Software

- Many programs still provide a valuable business benefit, even though they are one or even two decades old.
- Software systems need to be continually updated if they are to remain useful to their customers.
- These programs must be maintained and this creates problems because their design is often not amenable to change.



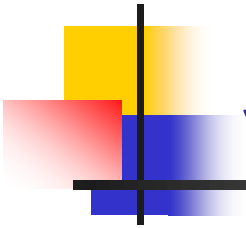
Legacy Software

- Why must it change?
  - Software must be **adapted** to meet the needs of new computing environments or technology.
  - Software must be **enhanced** to implement new business requirements.
  - Software must be **extended to make it interoperable** with other more modern systems or databases.
  - Software must be **re-architected** to make it viable within a network environment.



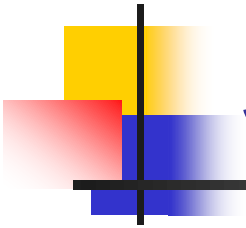
Software Evolution

- Continuing Change
  - A program that is used in a real-world environment changes or become less and less useful in that environment
- Increasing Complexity
  - As an evolving program changes, its structure becomes more complex unless active efforts are made to avoid this phenomenon



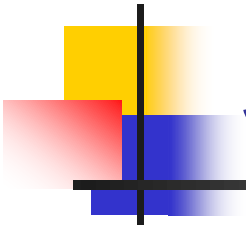
Software Evolution

- Program Evolution
  - Program evolution is a self-regulating process and measurement of system attributes such as size, time between releases, number of reported errors etc. reveals statistically significant trends and in-variances



Software Evolution

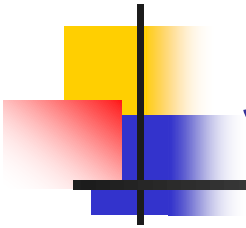
- Conservation of Organizational Stability
  - Over the lifetime of a program, the rate of development of that program is approximately constant and independent of the resources devoted to system development



Software Evolution

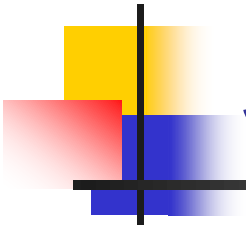
- Conservation of Familiarity
  - Over the lifetime of a system, the incremental system change in each release is approximately constant

Lehman, M et al. 'Metrics and Laws of Software Evolution - The Nineties View', In Proceedings of METRICS 97, 1997



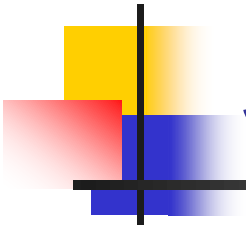
Software Evolution

- Implications
  - Change is inevitable, plan for it
  - Don’t attempt to make very big changes in a single increment
  - Adding staff to a project will have a limited effect on project recovery



Software Engineering Costs

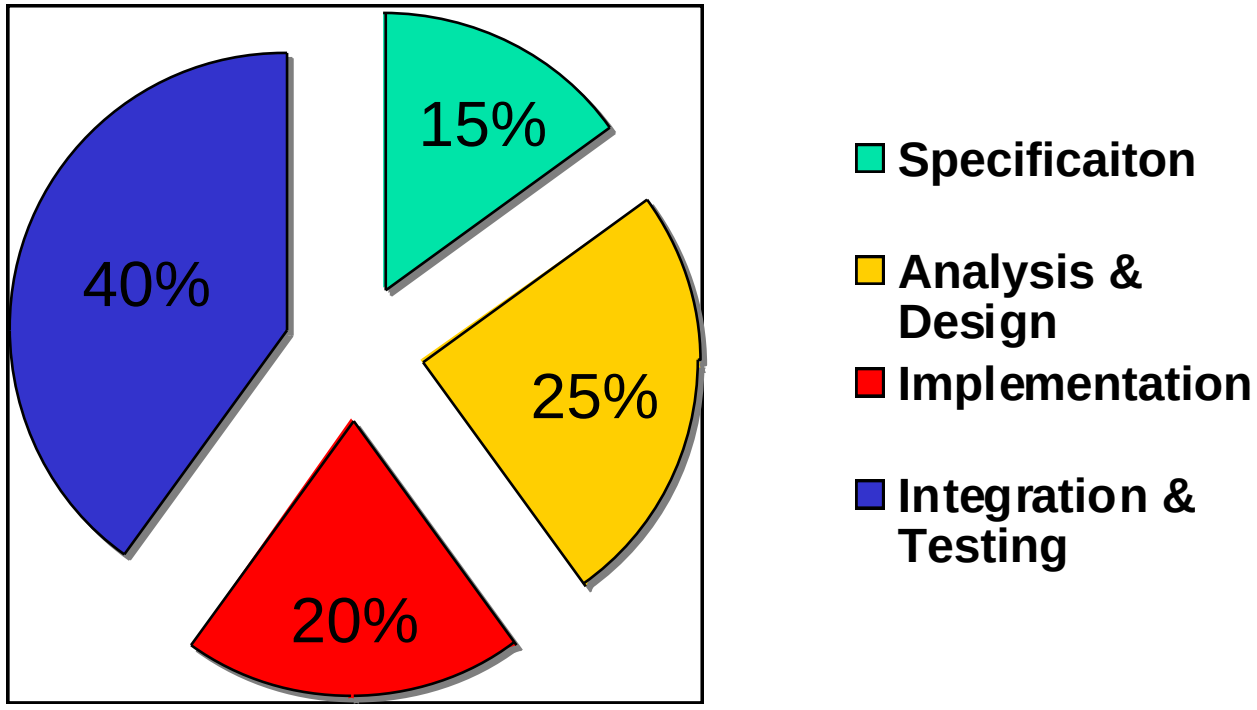
- Distribution of costs in computing projects is changing - software rather than hardware is the largest single cost item.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs
- Software engineering is concerned with cost-effective software development



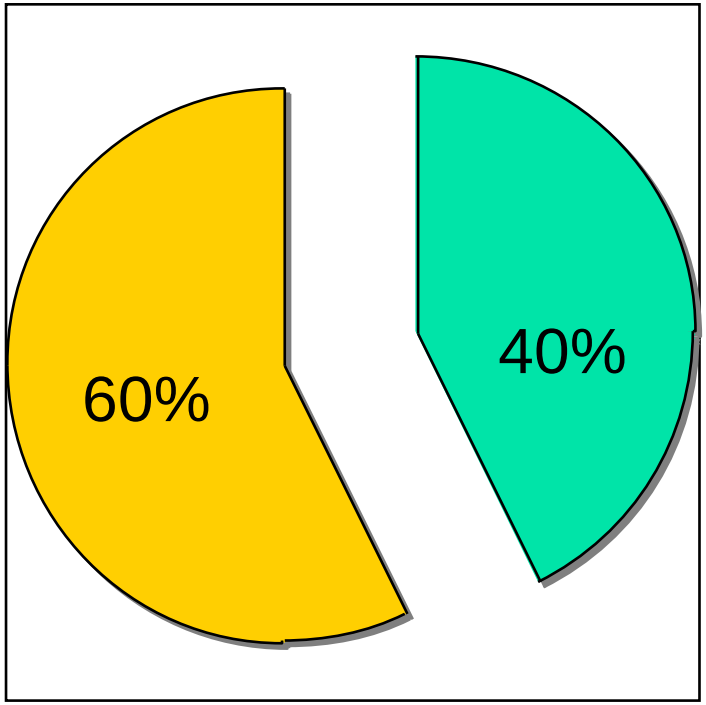
Software Engineering Costs

- Distribution of costs depends on the development model.
- Costs vary depending on
  - the type of system being developed and;
  - the requirements of system attributes such as performance and system reliability.

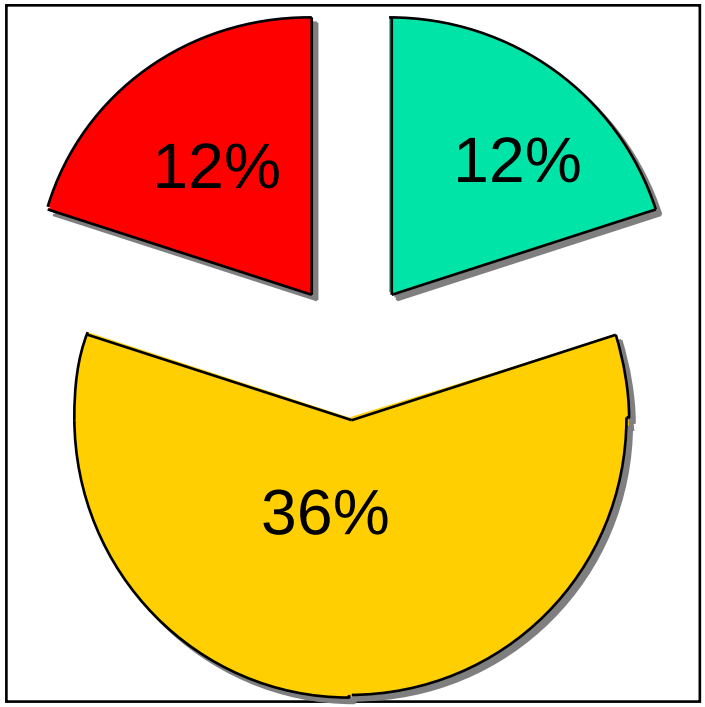
Activity Cost Distribution

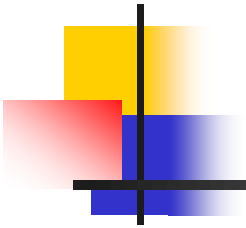


Development - Maintenance Cost Ratios



Maintenance Cost Ratios

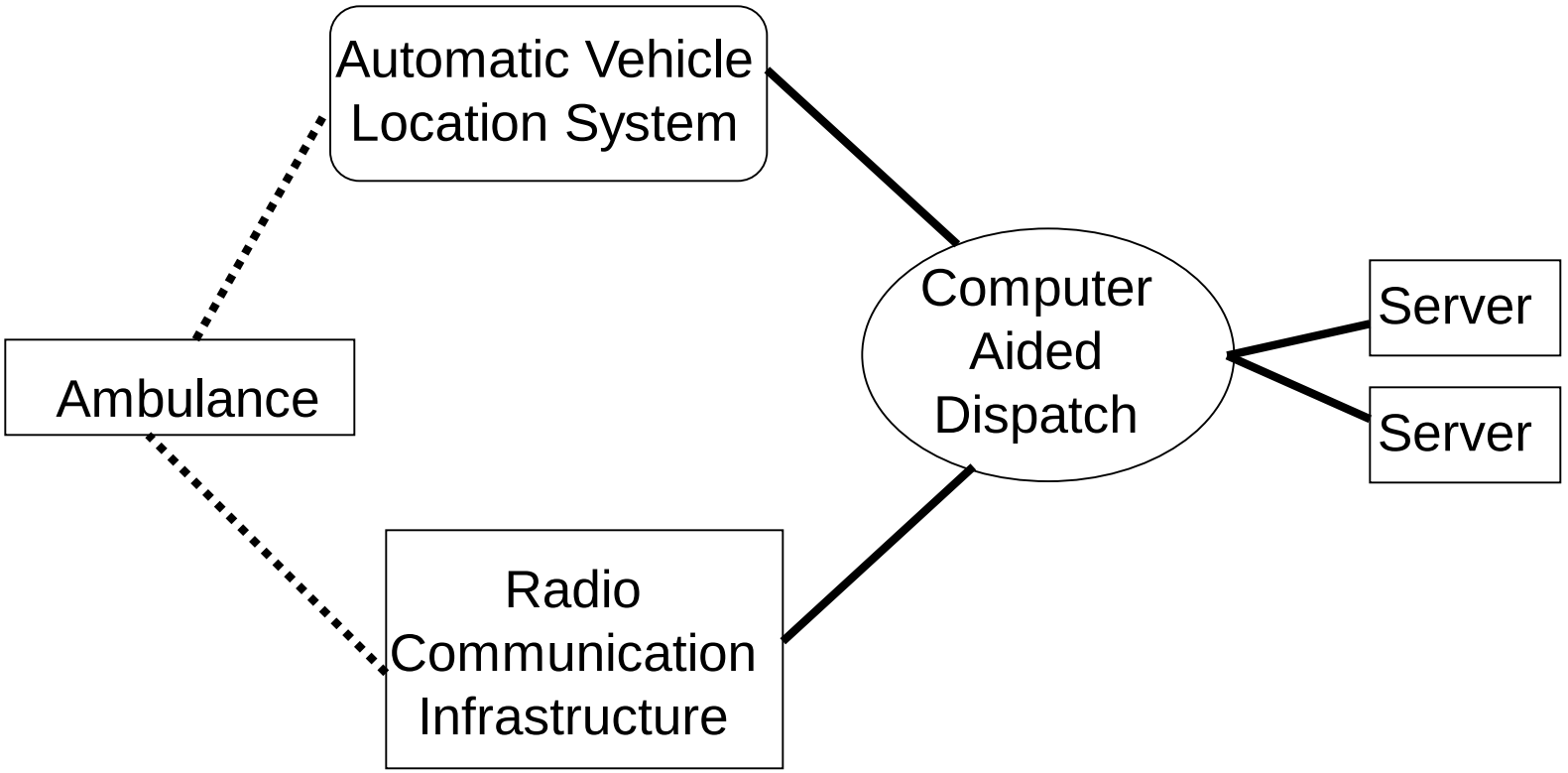




Important Question

- Why do we continue to have difficulty in software development projects?

London Ambulance Service Case Study

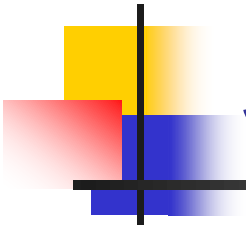


London Ambulance Service Case Study

- System could not keep track of the location and status of ambulances.
- Database entries begin to store incorrect information.
- As a result,
  - A large number of exception messages
  - System starts to slow down

London Ambulance Service Case Study

- Entire system descended into chaos
  - Ambulance arrives for a ‘Stoke’ patient after 11 hours.
- The CAD system was removed and aspects of its function (dispatch decisions) were performed manually.



Software Myths

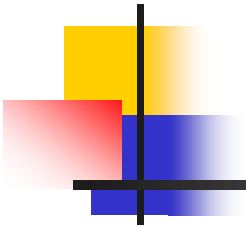
- Affect managers, stakeholders, and practitioners
- Are believable because they often have elements of truth

but…

- Invariably lead to bad decisions,

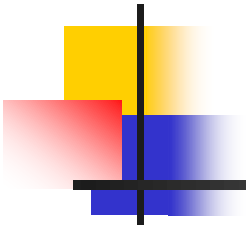
therefore….

- Insist on reality as you navigate your way through software engineering



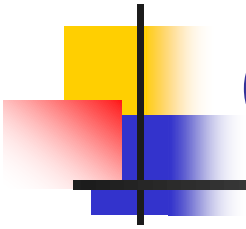
Management Myths

- *‘We already have books full of standards and procedures for building software. That will provide my people with everything they need to know’*
- *‘My people do have state-of-the-art software development tools. After all, we buy them the latest computers’*



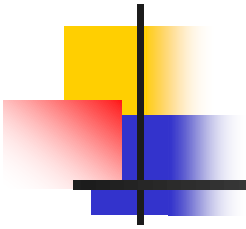
Management Myths

- *‘If we get behind schedule we can add more programmers and catch up’*
- *‘If I decide to outsource the software project to a third party, I can just relax and let that firm build it*



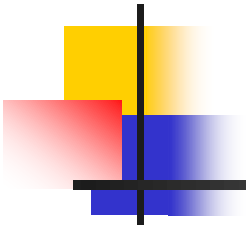
Customer Myths

- ‘A general statement of objectives is sufficient to begin writing software - we can fill in the details later’
- ‘Project requirements continually change but change can be easily accommodated because software is flexible’



Practitioner's Myths

- 'Once we write the program and get it to work our job is done'
- 'Until I get the program running I really have no way of assessing its quality'
- 'The only deliverable for a successful project is the working program'

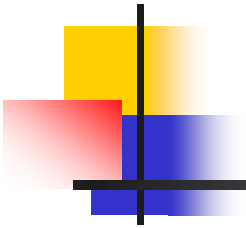


Key Challenges

- Heterogeneity
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- Delivery
  - Developing techniques that lead to faster delivery of software;
- Trust
  - Developing techniques that demonstrate that software can be trusted by its users.

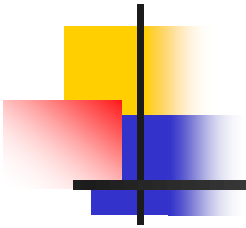
Key Challenges

- An accompanying shift from a concern with *whether* a system will work towards how *well it* will work.
- Components are selected and purchased ‘off the shelf’ (COTS) with development effort being refocused on configuration and interoperability



How a Project Starts?

- Every software project is precipitated by some business need
  - Need to correct a defect in an existing application
  - Need to adapt a legacy system to a changing business environment
  - Need to extend the functions and features of an existing application
  - Need to create a new product or system



Key Points

- Software is a complex engineering product.
- Approaches which work for constructing small programs for personal use do not scale-up to the challenges of real software construction.
- A disciplined engineering process and associated management disciplined is needed.