CSSE 372 Software Project Management:
First Exam Review

Shawn Bohner
Office: Moench Room F212
Phone: (812) 877-8685
Email: bohner@rose-hulman.edu

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

More on (not moron!) Mythical Man Month 20 years later

By Fred Brooks

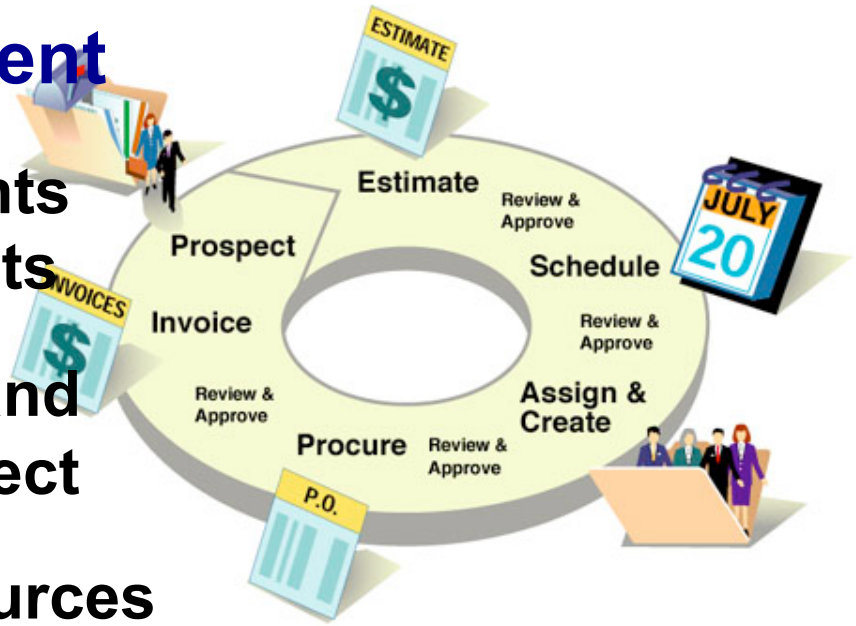
- How does Harlan Mills' making "programming a public process" improve software?
- What did Prof. Brooks mean by "People are everything?"
- How does Prof. Brooks feel about David Parnas' information hiding?
- What did you learn about software projects?

Perspective on Exam 1:

- Exam will be 45 minutes
  - Some multiple choice/short answer (40%)
  - Some essay/exercises/problem sets (60%)
  - Close book/notes/computer…
- What is fair game?
  - Class slides and daily quizzes
  - Book/Readings/Case Studies
- Study Strategy
  - Daily quizzes → Slides → Books/Readings/Cases
- Can have a help sheet
  - 1 page/1 side, no less than 6 point font

Objectives: Fundamental Elements

Identify fundamental elements of Software Project Management



- Determine stakeholders/clients and their project commitments
- Outline the purpose, goals, and objectives of a software project
- Define software project resources
- Outline the concept of project scope
- Outline some software project failures from literature
- Examine causes of software project failures

Anatomy of a Project

- A Customer/Client
- Goal-Oriented Plan
- Practices and Processes
- Project Team
- Resources
- Commitment

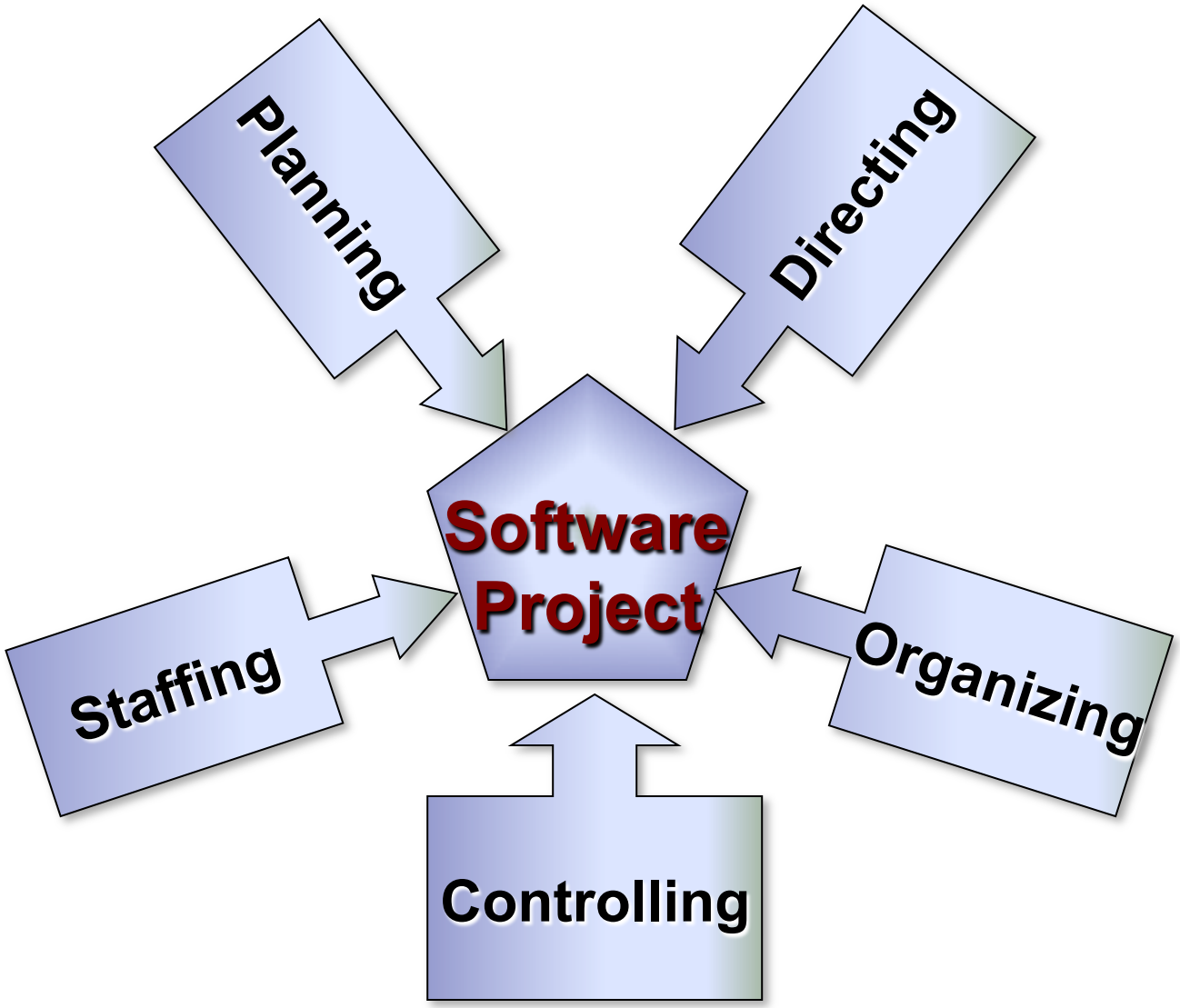
Customers and Clients

- Relationship
- Engagement
- Value proposition
- Buy in

- A client is a stakeholder with the authority to make commitments and decisions regarding the product to be delivered

- Stakeholders may have an interest, influence, or investment, but typically convey buy in

Classic Management Activities



Software Related Failures

Programming errors

Passive failures

Active failures

Byzantine failures

Epic Software Failures

- European Space Agency's Ariane 5 Explosion
- Hospital Radiation Incident
- London Ambulance Service
- Denver Airport Baggage Handler
- East Coast Blackout
  (cascading power plant failures)
- AT&T Switch Failure -$Billion bug
- FAA's Advanced Automation System
- NASA Mars Lander, Pathfinder, and Spirit…

Why Software Projects Fail?

1. Unrealistic or unarticulated project goals
2. Inaccurate estimates of needed resources
3. Badly defined system requirements
4. Poor reporting of the project's status
5. Unmanaged risks
6. Poor communication: clients, developers, & users
7. Use of immature technology
8. Inability to handle the project's complexity
9. Sloppy development practices
10. Poor project management
11. Stakeholder politics
12. Commercial pressures

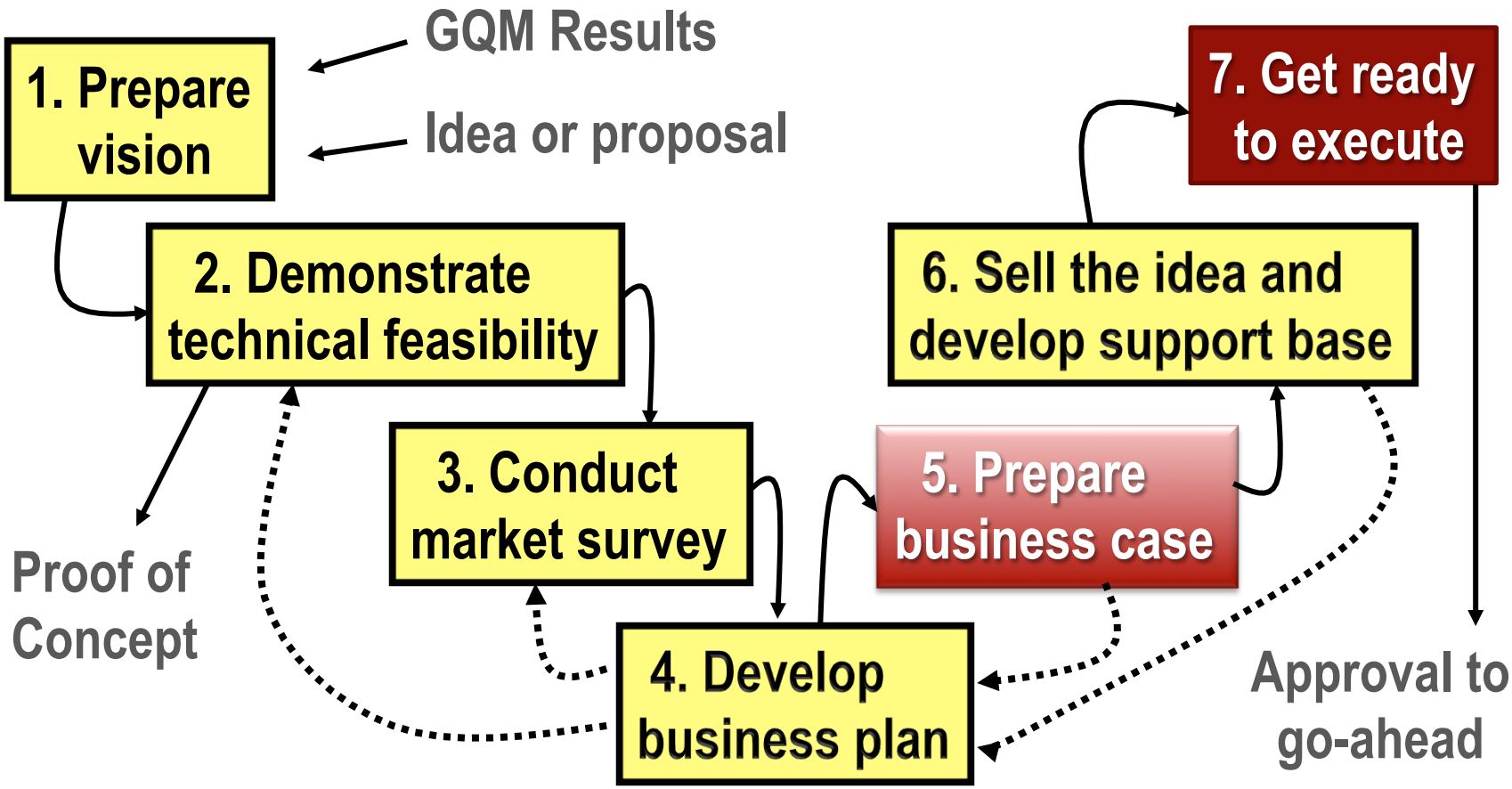
Source: Robert Charette Article

Learning Outcomes: Plan (verb)

Create a plan for an intermediate size software project & manage to the plan as project evolves.

- **Describe a Business Case for a software product**
- **Outline major parts of Software Project Plan**

Business Planning Process



What is a Business Case?

- A business case outlines the overall costs and risks and compares them to the benefits that justify the initial and on-going commitment of time, resources, and funding for a software project.

Financial

Costs, benefits and impact on business performance measures

Strategic

New capabilities and improved competitive position

ROI

Return on Investment

Technical

Benefits to organization's infrastructure and support for technology strategy

Operational

Process and engineering improvements

(Tangible and Intangible)

What's in a Business Case?

- Problem or situation addressed by the proposed project
- Features and scope of the proposed initiative
- Options considered and the rationale for choosing the solution proposed
- Proposed Project's conformity with existing policies, etc.
- An initial implementation plan (very high-level)
- Expected costs and budget
- Anticipated benefits and outcomes
- Potential risks

Let's See what is in a Business Case

- An informal Business Case (Smaller Project)
  - Introduction and Purpose
  - Window of Opportunity
  - Approaches
  - Benefits
  - Costs
  - Risks/Exit Strategy
- A formal Business Case (Larger/Critical Project)
  - Much more detail
  - See Business Case Template…

The 4 P's of a Software Project

- People
- Product
- Process
- Project

Basic Software Planning Steps

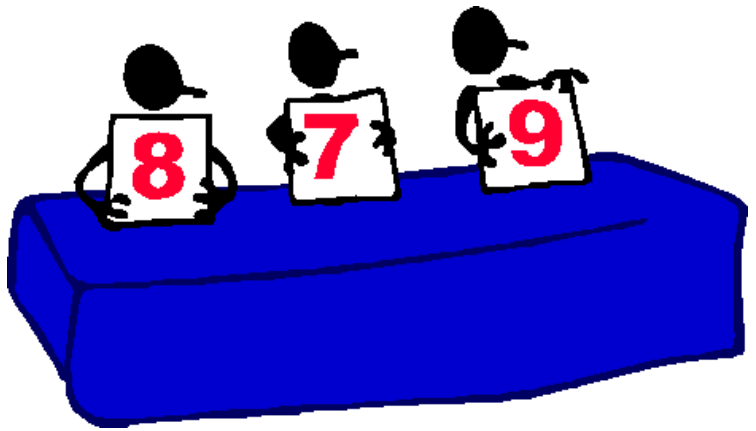
- Scoping
- Estimation
- Risk
- Schedule
- Control strategy

Scope Related Risks and Assumptions

- Technological
- Environmental
- Interpersonal
- Cultural
- Causal Relationships

Importance of Planning

- Planning Reduces Uncertainty
- Planning Increases Understanding
- Planning Improves Efficiency



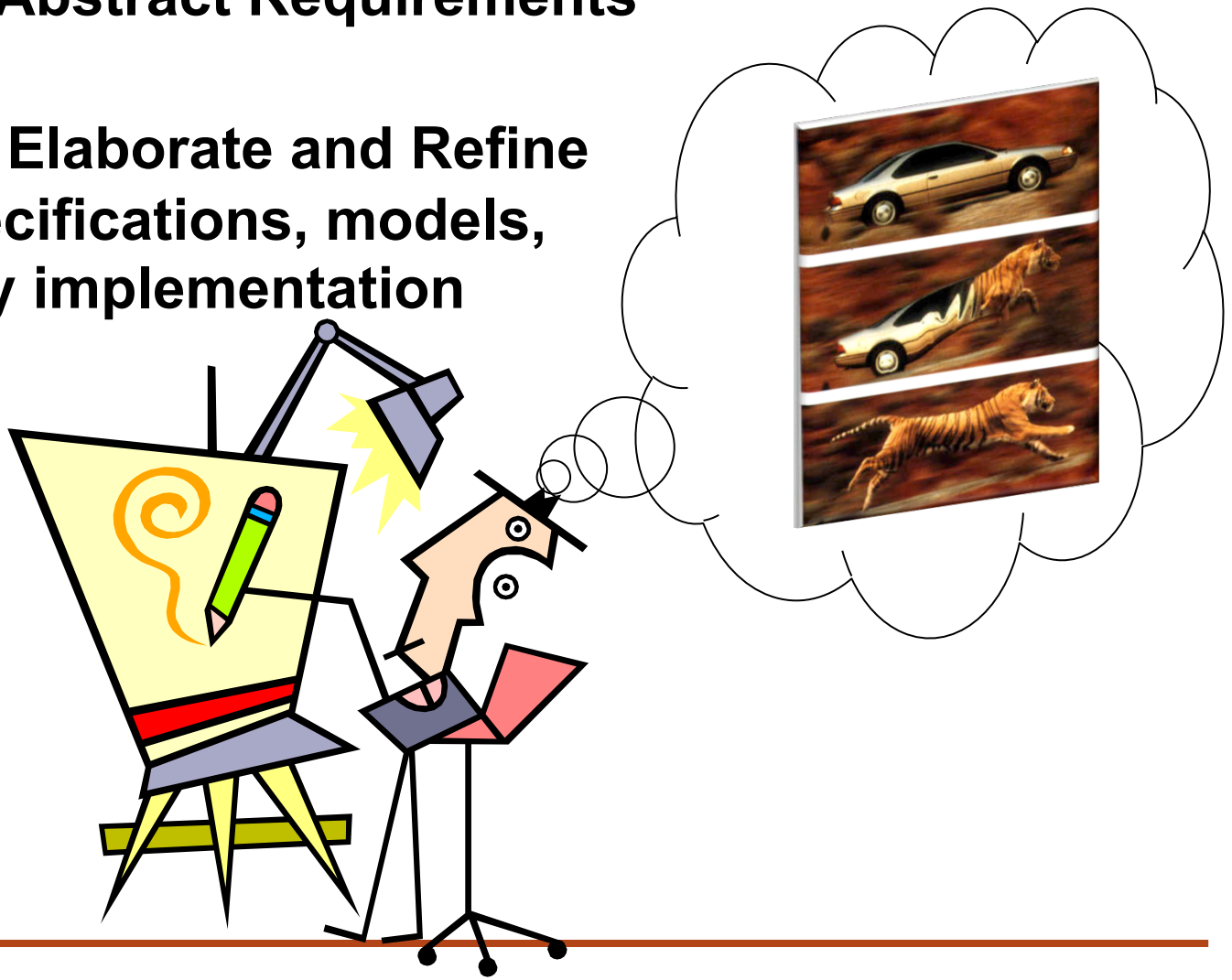
Learning Outcomes: Plan (verb)

Create a plan for an intermediate size software project & manage to the plan as project evolves.

- Define elements of software life cycle process
- Relate software artifacts to software process activities
- Outline key software process models and applications

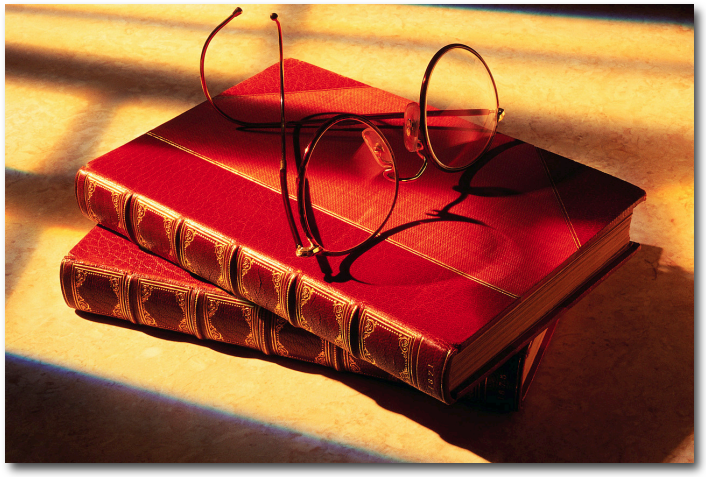
Elaboration and Refinement…

- Starting with Abstract Requirements
- Successively Elaborate and Refine them into specifications, models, and ultimately implementation

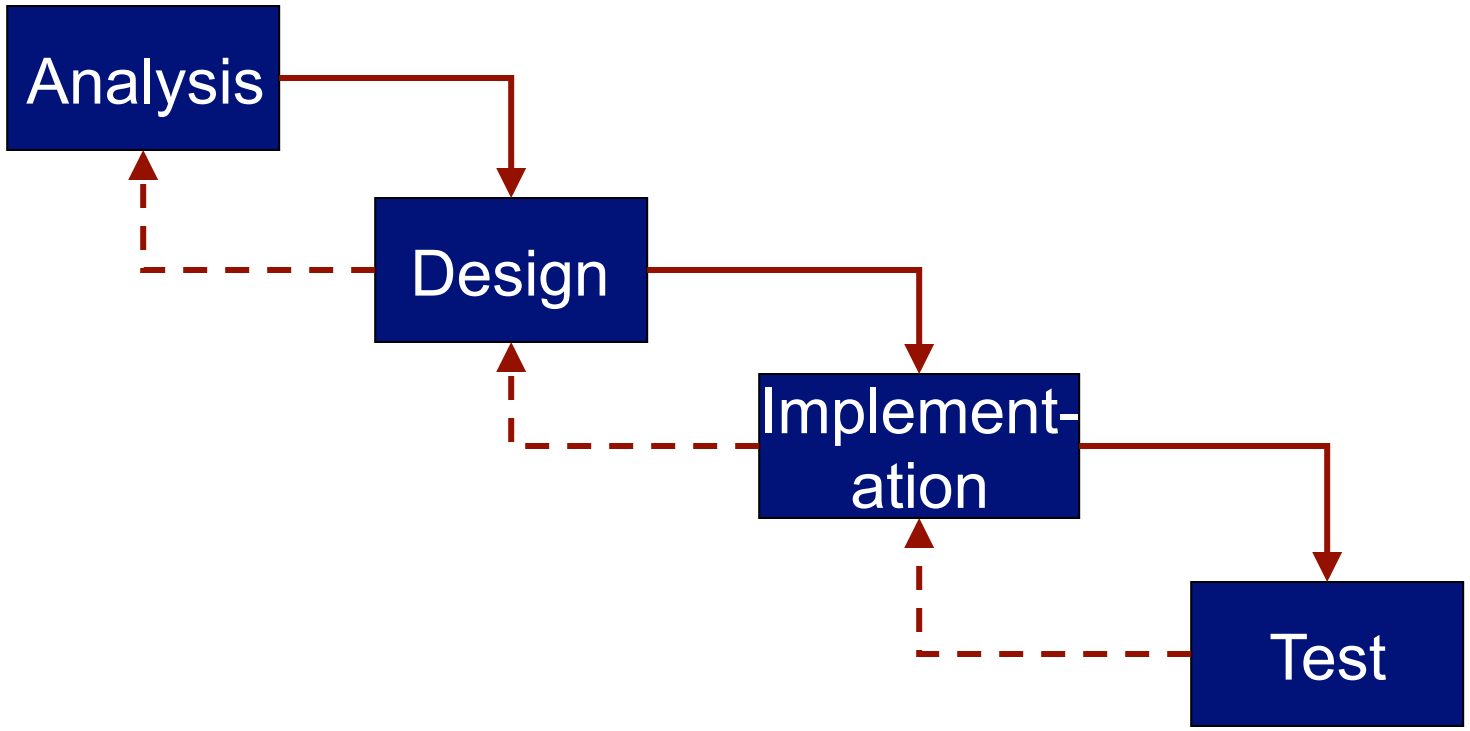


What is a Software Process?

- A **Software Process** organizes the life cycle activities related to the creation, delivery, and maintenance/evolution of software systems
- **Software Life Cycle Activities** are organized within a software process to provide a systematic approach for producing software products
- A **software process model** is an abstract description of the activities to produce a software system from a particular perspective



The Waterfall Model

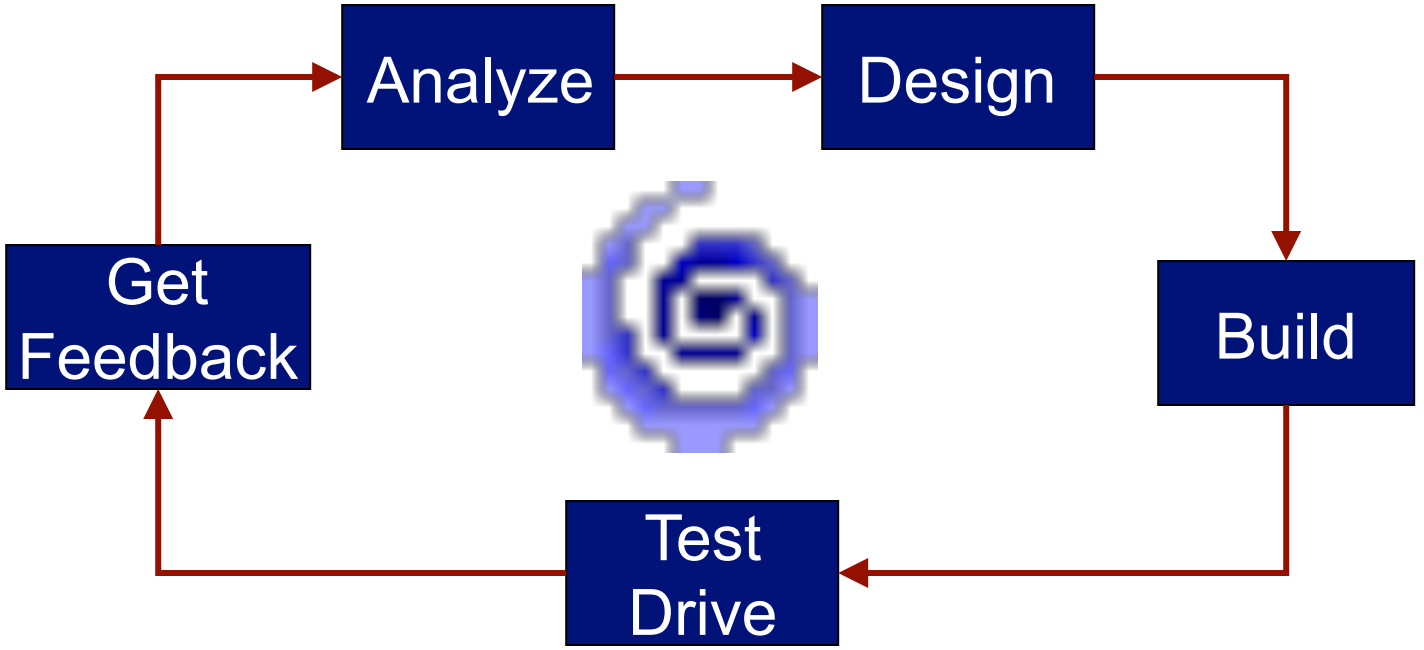


- Appropriate when the requirements are well-understood and changes will be governed/limited
- Mostly used for large systems engineering projects where a system is developed at several sites

Cross-Life-Cycle Activities

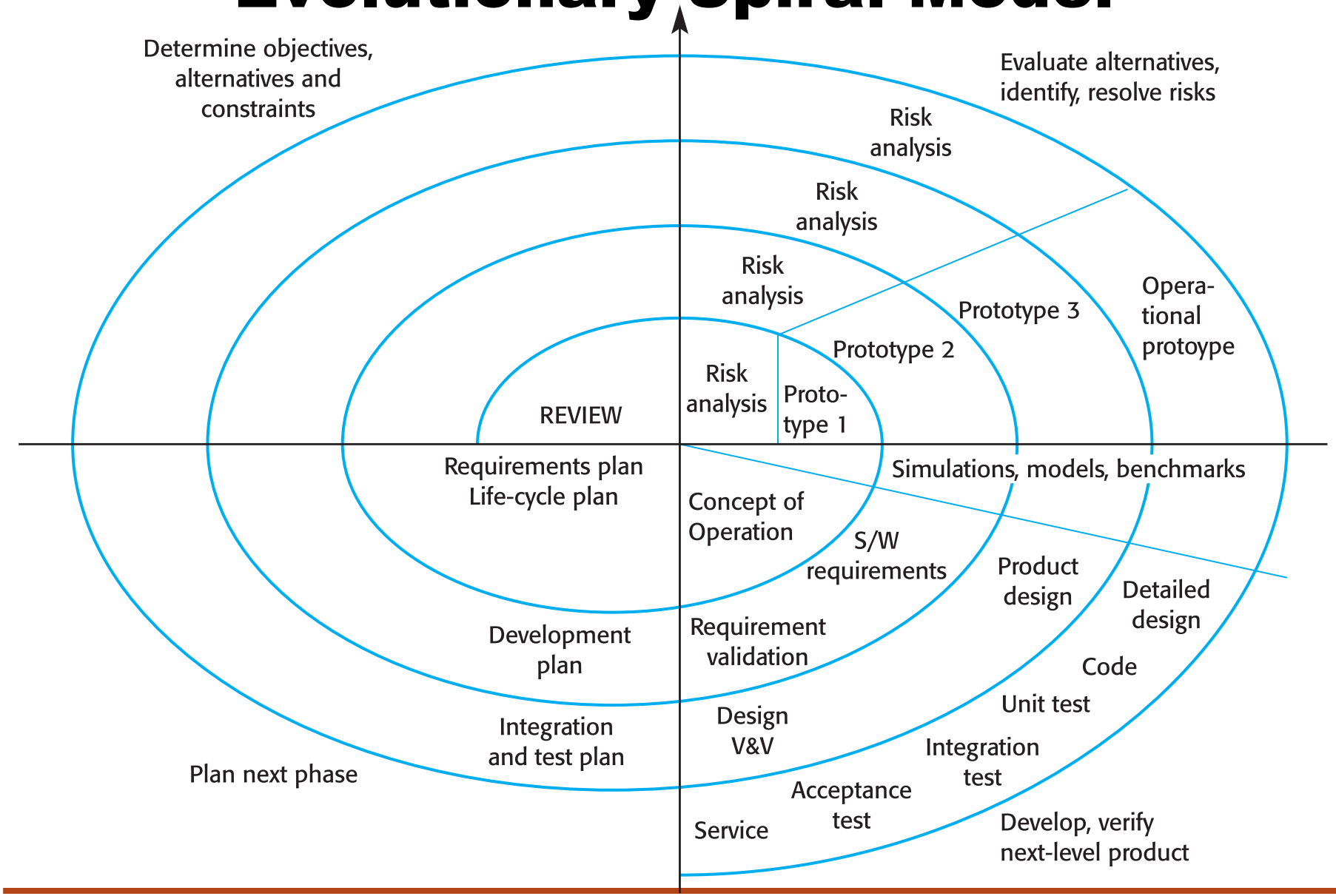
- Cross-Life-Cycle activities are used to gate or control the life cycle activities for better flow and throughput
  - Project management
  - Risk management
  - Configuration management
  - Quality assurance
  - Measurement

Iterative Software Process Models

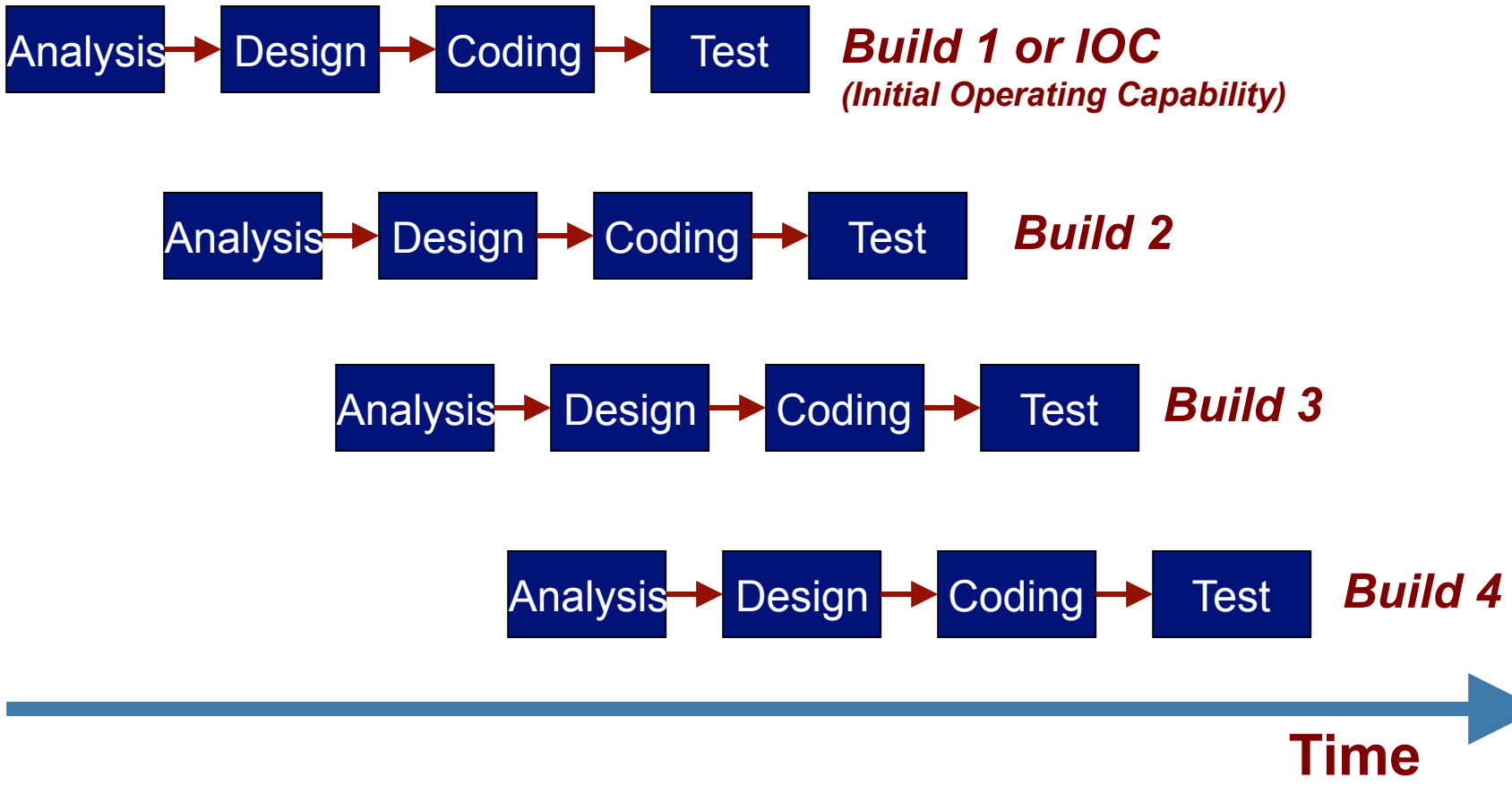


- AKA “Build a little, test a little” or “Learn as you go”
- Used when requirements evolve or are clarified during the course of a project

Evolutionary Spiral Model



Incremental Process Models



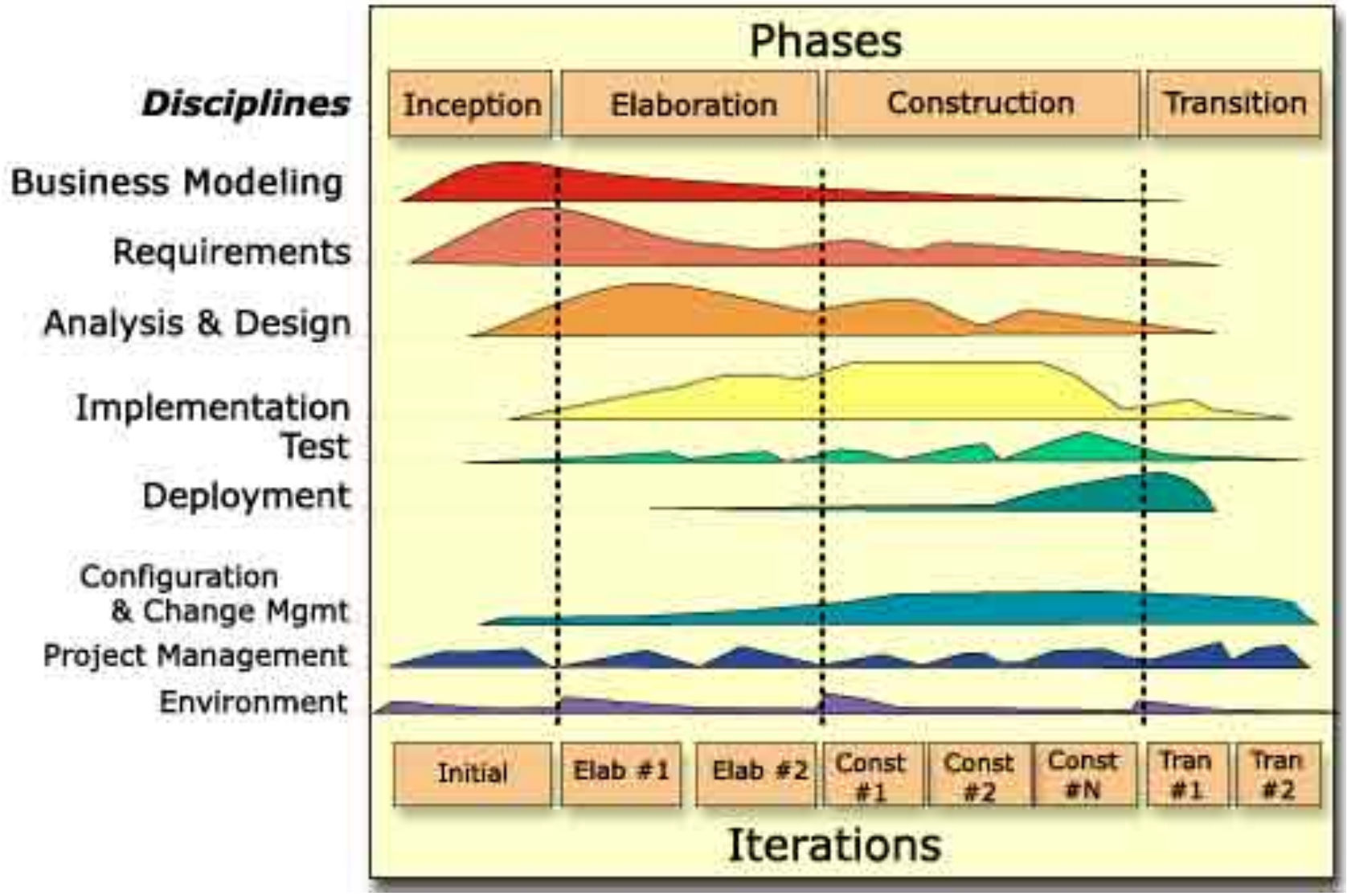
Systematic divide and conquer strategy for completing projects using concurrent activities

Choosing a Software Process

- No uniformly applicable process which can be standardized within an organization

- Consider project characteristics
  - Team's time together, skills mix, experience, etc.
  - Previous experience on this type of system
  - Rigor and quality constraints
  - Timeline and Deadness of deadlines…
  - Type and size of product developed
- Remember to tailor the process to your needs

Unified Process (UP)



Learning Outcomes: Plan (verb)

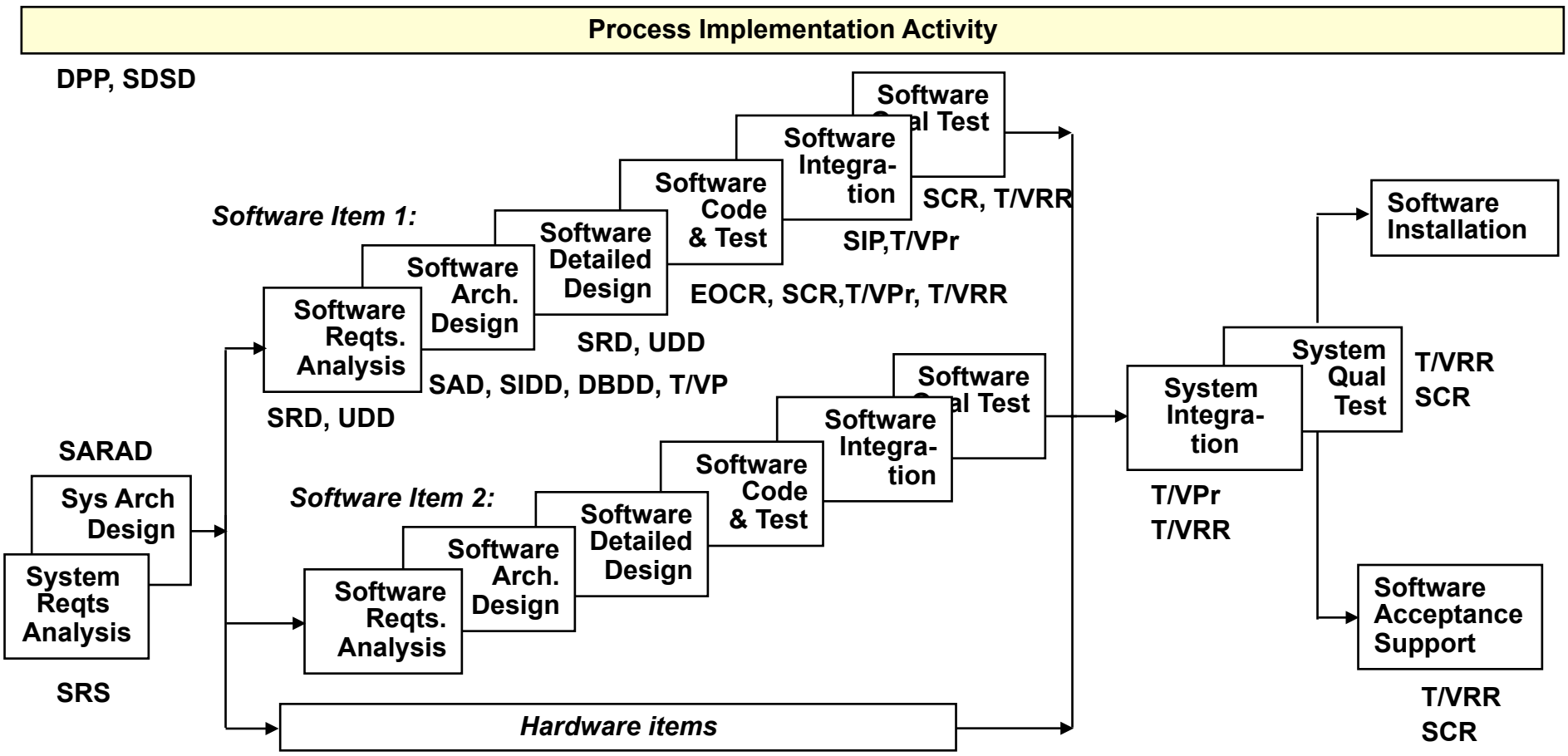
Create a plan for an intermediate size software project & manage to the plan as project evolves.

- **Experience a software project through simulation or game.**
- **Examine software life cycle standards.**
- **Discuss Cleanroom Software Engineering**

Software Process Standards

- The wonderful thing about standards is that there are so many to choose from…
- Standards save time and money
- Reasonable starting point
  - Remember to tailor it for your situation…
- Examples:
  - ISO 12207 Software Life Cycle Standard
  - MIL-STD-498/IEEE 1498 – Software Development Life Cycle
  - IEEE Software Related standards
    - http://standards.ieee.org/reading/ieee/std_public/description/se/

Sample ISO 12207 Development Process



Tailoring ISO 12207

- Tailoring considerations:
  - Life cycle activity: prototyping, maintenance
  - Software characteristics: COTS, reuse, embedded firmware
  - Organizational policies, languages, hardware reserve, culture
  - Acquisition strategy: contract type, contractor involvement
  - Life cycle strategy: waterfall, evolutionary, spiral, etc.

- The Tailoring Process (12207.0 Annex A)
  1. Identify project environment - strategy, activity, requirements
  2. Solicit inputs - from users, support team, potential bidders
  3. Select processes, activities, documentation, responsibilities
  4. Document tailoring decisions and rationale

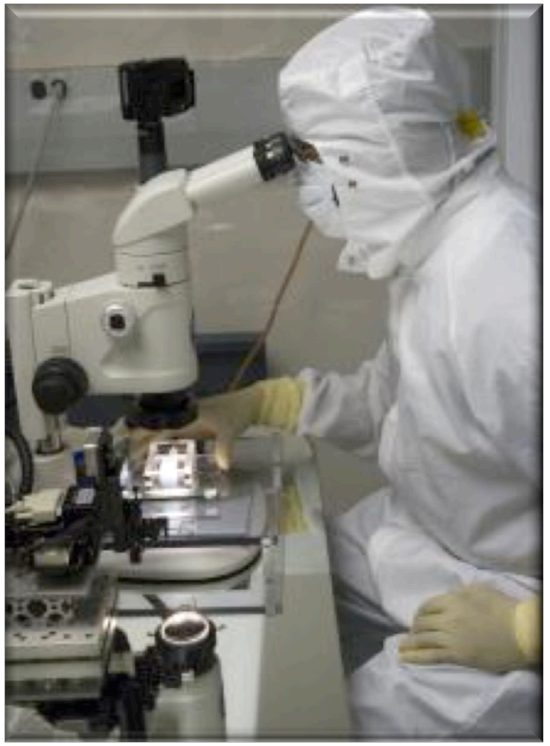
Cleanroom is Shift in Practice

- From
  - ☐ Individual craftsmanship
  - ☐ Sequential development
  - ☐ Individual unit testing
  - ☐ Informal coverage testing
  - ☐ Unknown reliability
  - ☐ Informal design
- To
  - ☑ Peer reviewed engineering
  - ☑ Incremental development
  - ☑ Team correctness verification
  - ☑ Statistical usage testing
  - ☑ Measured reliability
  - ☑ Disciplined engineering specification and design

Focuses on defect avoidance rather than defect removal

Cleanroom Process Teams

- **Specification team**
  - Develops and maintains the system specification
- **Development team**
  - Develops and verifies software
  - Software is not compiled or executes during verification
- **Certification team**
  - Develops set of statistical tests to exercise software after development
  - Reliability growth models used to assess reliability

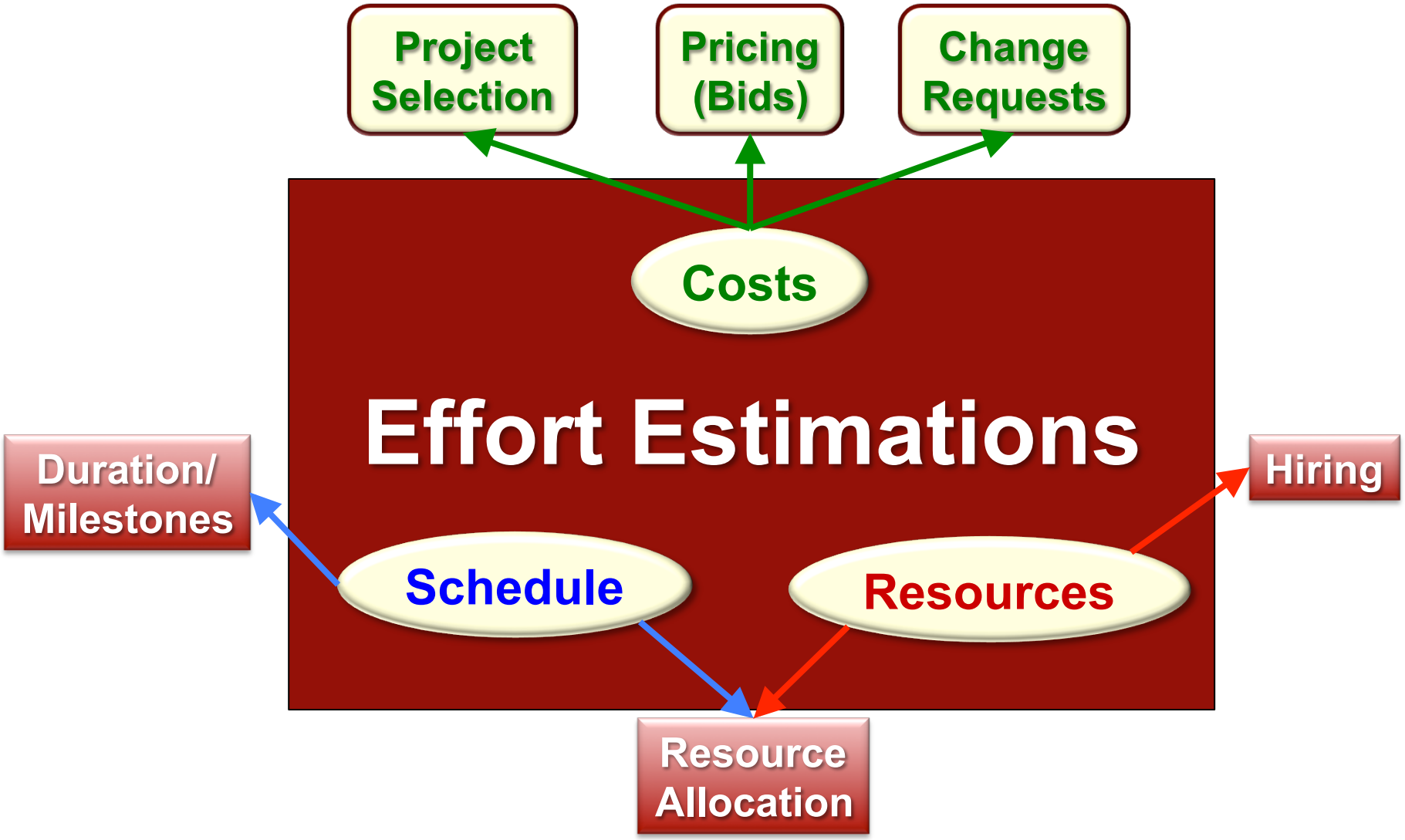


Learning Outcomes: Estimation

Estimate software project effort, cost, and schedule for an intermediate size project.

- Compare contemporary estimation approaches
- Determine appropriate estimation approach for an estimate

Why do Software Project Estimates?



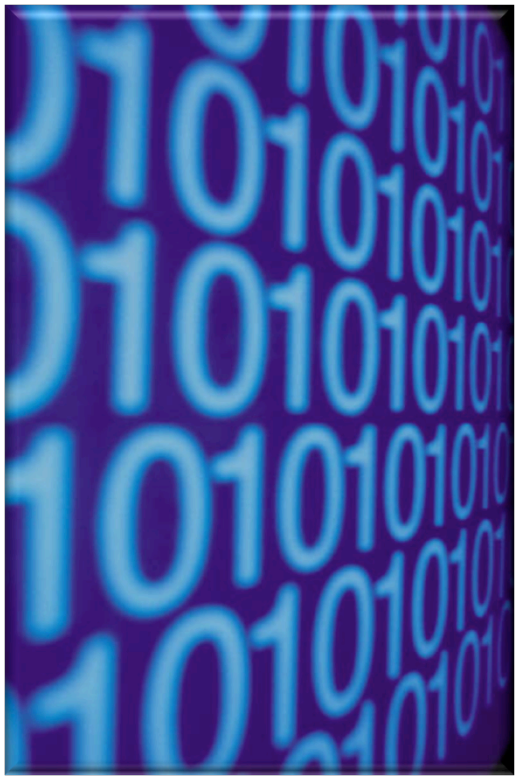
Software Cost Estimation Methods

- Estimation by Analogy
- Delphi – Expert Judgment
- Price-to-Win – Time/Cost Box
- Bottom-Up – Grassroots
- Parametric

Best approach is use of multiple methods, comparing and iterating results to clarify differences

Parametric Estimating 1/2

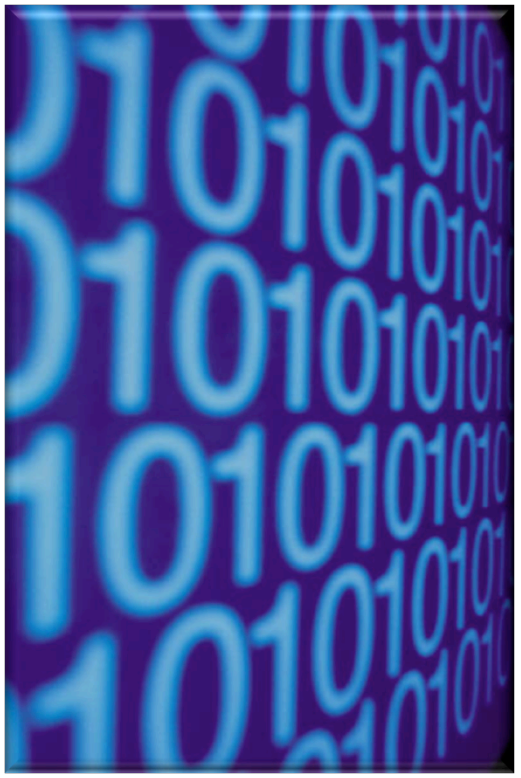
- Relates the cost of a system to one or more parameters of the system such as physical or performance characteristics



- Uses
  - Cost estimates and trade-offs for systems in early development
  - Quick reaction estimates
  - Independent check on other estimates

Parametric Estimating 2/2

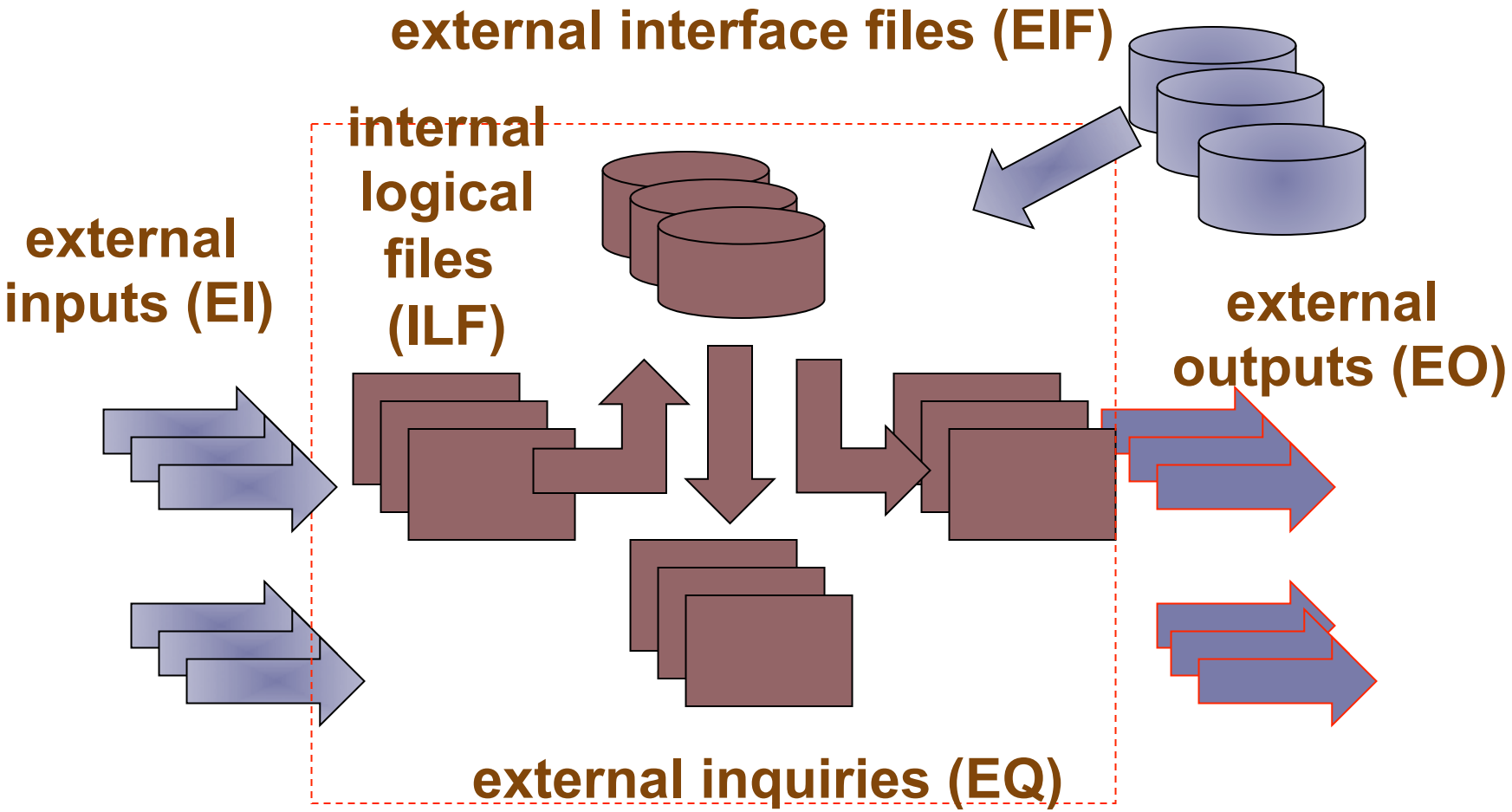
- **Advantages**
  - Sensitive to significant design changes
  - Quantifies effects of cost drivers
  - Based on "real world" experience of many systems
  - Gives quick, reproducible results



- **Disadvantages**
  - Inputs are subjective
  - Results not as precise as bottom-up
  - Requires skilled analyst to develop

Estimating Size with Function Points

(Albrecht & Gaffney)



Function Point Example (continued)

type	simple	average	complex	Fn Types	FP
ILF	7	9	15	1	1x9
EIF	5	8	10	4	4x8
EI	3	4	6	0	0x4
EO	4	6	8	1	1x8
EQ	3	4	6	0	0x4
				Total	49

Problem with Over/Under-Estimating

- Parkinson's Law: **Work expands to fill the time available**
  - An over-estimate is likely to cause project to take longer than it would otherwise
- Murphy's Law: **Anything that can go wrong will go wrong**
  - Under-estimation may lead to quality problem
- Brooks' Law: **Putting more people on a late job makes it later**

Learning Outcomes: Estimation

Estimate software project effort, cost, and schedule for an intermediate size project.

- Outline the COCOMO-II parametric model as an example
- Examine the key factors and adjustments
- Walk through getting started…

Accommodating Progression

3-level model that allows increasingly detailed estimates to be prepared as development progresses

- Early prototyping level
  - Estimates based on “object points” and a simple formula is used for effort estimation
- Early design level
  - Estimates based on “function points” that are then translated to lines of source code (LOC)
- Post-architecture level
  - Estimates based on LOC

Early Design Level

- Estimates made after requirements confirmed

$$PM = A \times Size^{B} \times M + \Pi EM_i$$

where:

- A = 2.5 in initial calibration
- Size in KLOC
- B varies from 1.1 to 1.24 depending on novelty of project, development flexibility, risk management approaches, and process maturity
- EM = (ASLOC × (AT/100)) / ATPROD
- M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED

Post-Architecture Level

- Uses same formula as early design estimates
- Estimate of size adjusted to account for:
  - Requirements volatility
  - Rework required to support change
  - Extent of possible reuse

$$ESLOC = ASLOC \times (AA + SU + 0.4DM + 0.3CM + 0.3IM)/100$$

- ESLOC is equivalent number of lines of new code
- ASLOC is the number of lines of reusable code which must be modified
- DM is the % of design modified
- CM is the % of the code that is modified
- …

Paper: "Software Estimation: An Overview" by Richard Stutzke

- What is main thrust or message of the paper?
- What are some of the reasons that estimation can be challenging?
- Why is accuracy of the estimates converging as a project progresses? If they were not converging, what would it mean?
- How has estimation changed over the years?
- What skill do you need to become a good software project estimator?

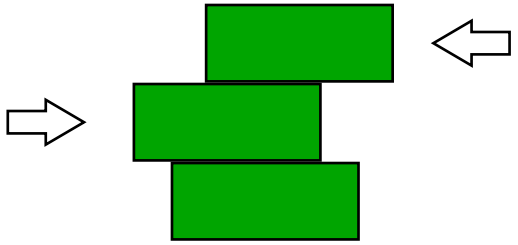
Learning Outcomes: Risks

Identify, analyze, and manage software project risks

- Define risky software situations
- Identify common software risks
- Analyze software risks

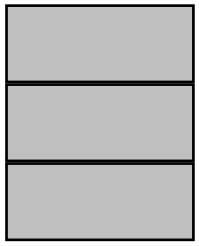
Software Projects are all about Risk

- Risk translates to:
  - Lack of information
  - Lack of time, and/or
  - Lack of control

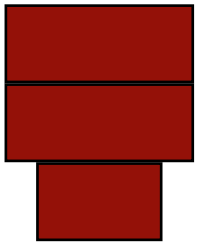


Must Effectively Balance
Risk & Opportunity

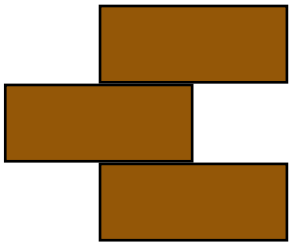
- Risk = Impact X Exposure



Banker's Risk



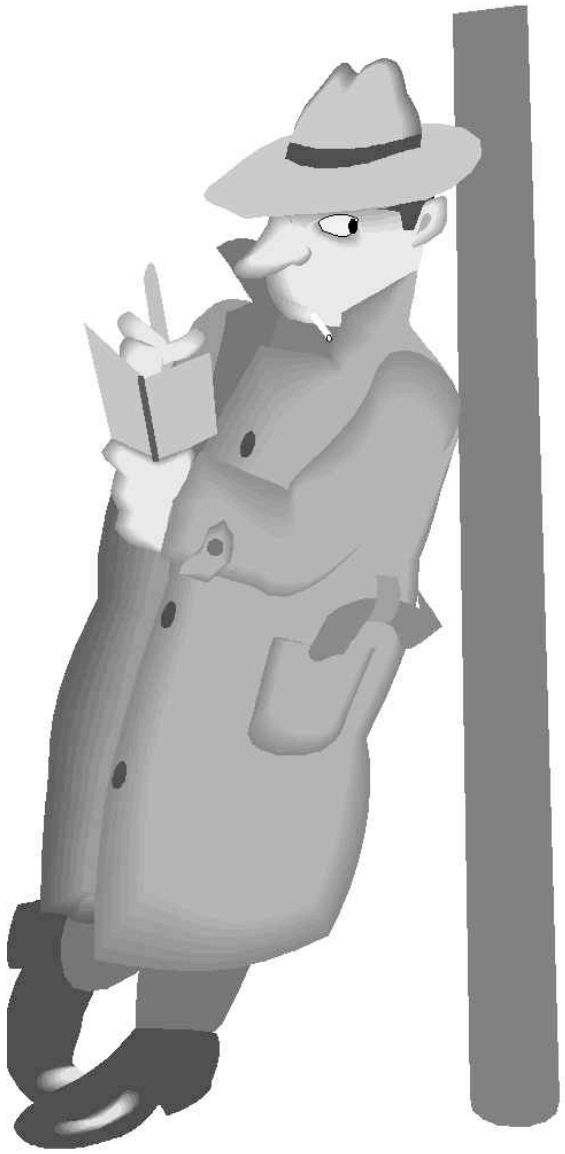
Investor's Risk



Gambler's Risk

Risk Identification

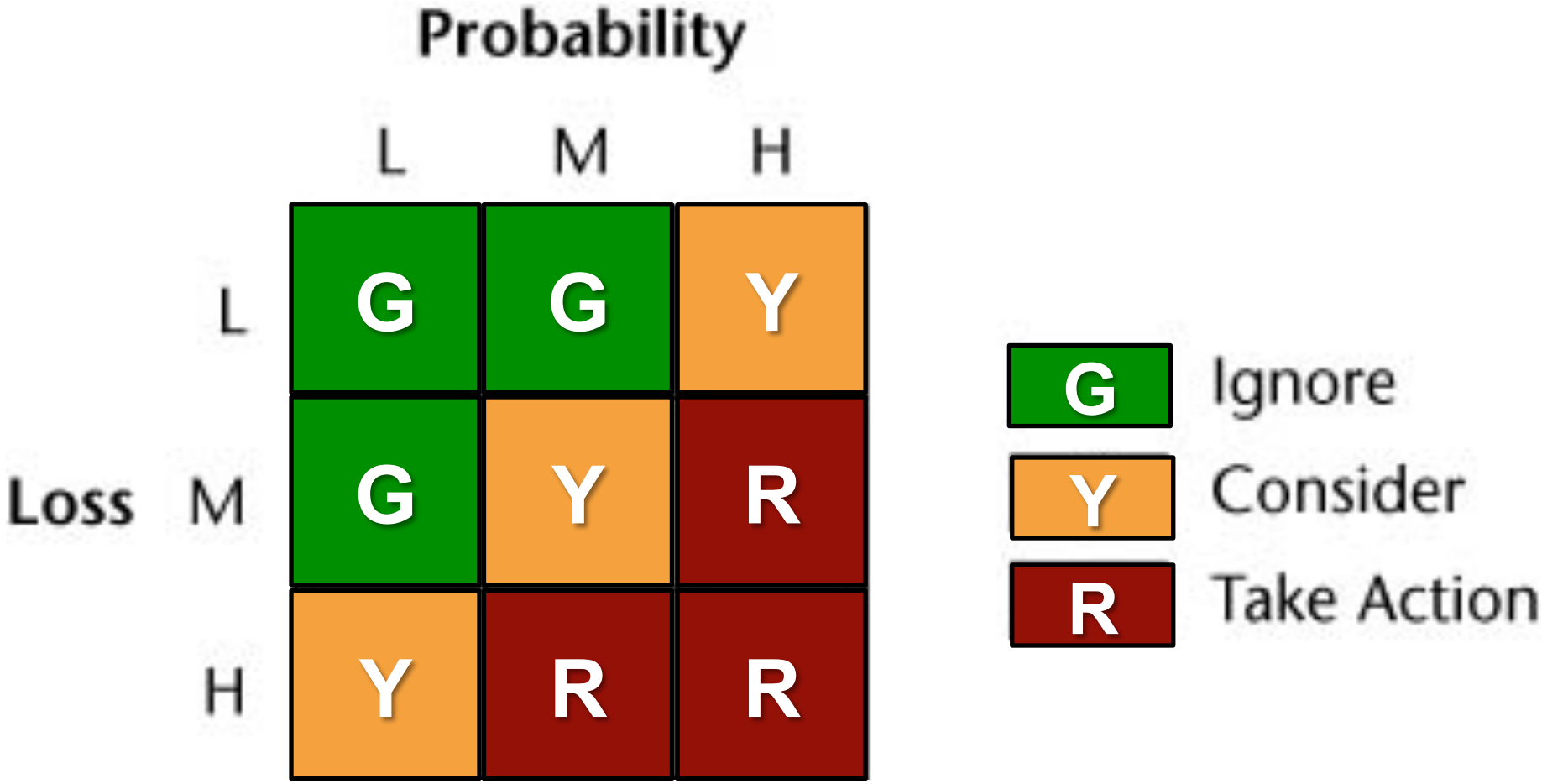
- *Product size*
- *Business impact*
- *Customer characteristics*
- *Process definition*
- *Development environment*
- *Technology to be built*
- *Staff size and experience*



Common Project Risk Components

- ***Performance Risk***—the degree of uncertainty that the product will meet its requirements and be fit for its intended use
- ***Cost Risk***—the degree of uncertainty that the project budget will be maintained
- ***Support Risk***—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance
- ***Schedule Risk***—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time

Qualitative Estimation



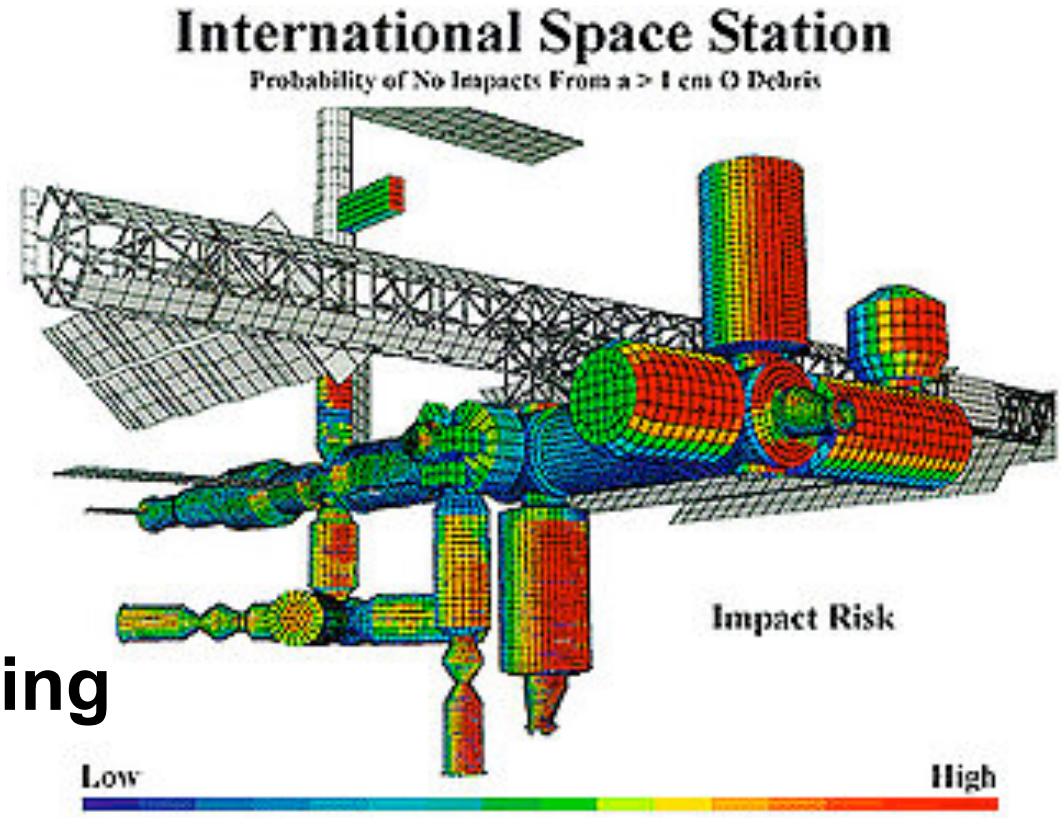
Learning Outcomes: Risks

Identify, analyze, and manage software project risks

- Planning to avoid software risks
- Monitoring identified software risks
- Controlling risky software situations
- Manage software risk for better value

What are Basic Responses to Risk?

- Accept
- Avoid
- Transfer
- Mitigate
- Contingency planning



Basic Risk Management

Building the Risk Table

- Estimate exposure
- Estimate the on project success on a scale of 1 to 5, where
  - 1 = low impact …
  - 5 = catastrophic impact
- Sort the table by exposure and impact

Risk	Exposure	Impact	Rationale
			Risk Mitigation Monitoring & Management

Recording Risk Information

Project Embedded software for XYZ system
Risk type schedule risk
Priority 4
Risk factor Project completion will depend on tests which require hardware component under development. Hardware component delivery may be delayed
Probability 60 %
Impact Project completion will be delayed for each day that hardware is unavailable for use in software testing
Monitoring approach
Scheduled milestone reviews with hardware group
Contingency plan
Modification of testing strategy to accommodate delay using software simulation
Estimated resources 6 additional person months beginning 7-1-96

Look over Homework 4 for specifics here…

Monitoring/Controlling

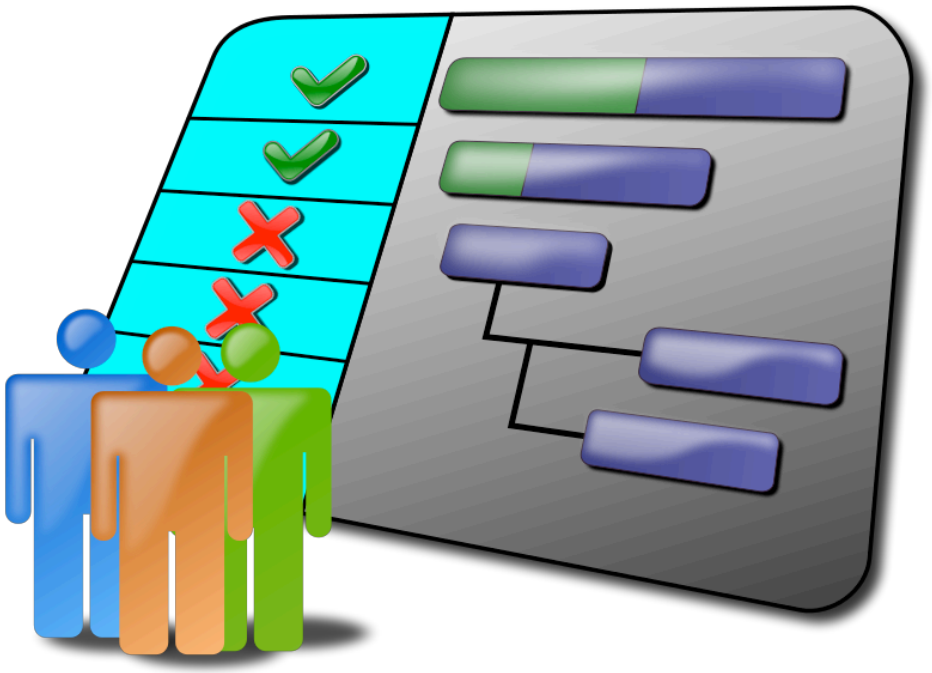
- Risk log
- ID number
- Risk description
- Risk owner
- Action to be taken
- Outcome

- Include Probability/Impact

Learning Outcomes: Schedule

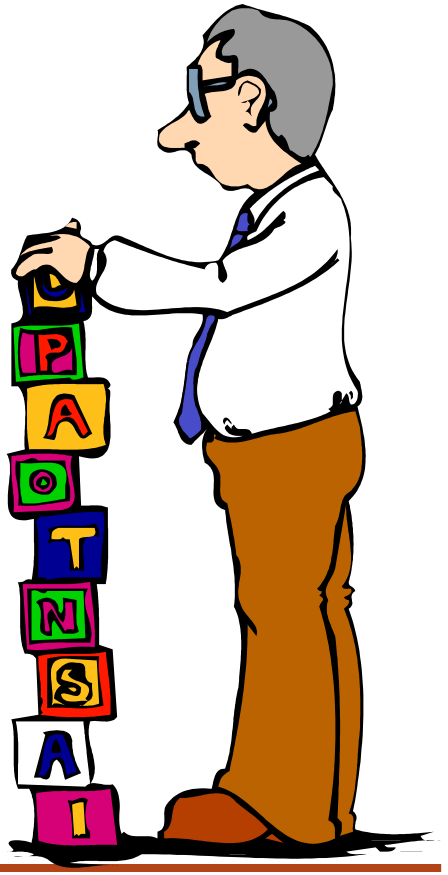
Create and maintain a software project schedule.

- Identify project tasks for planned work
- Develop a Work Breakdown Structure (WBS)

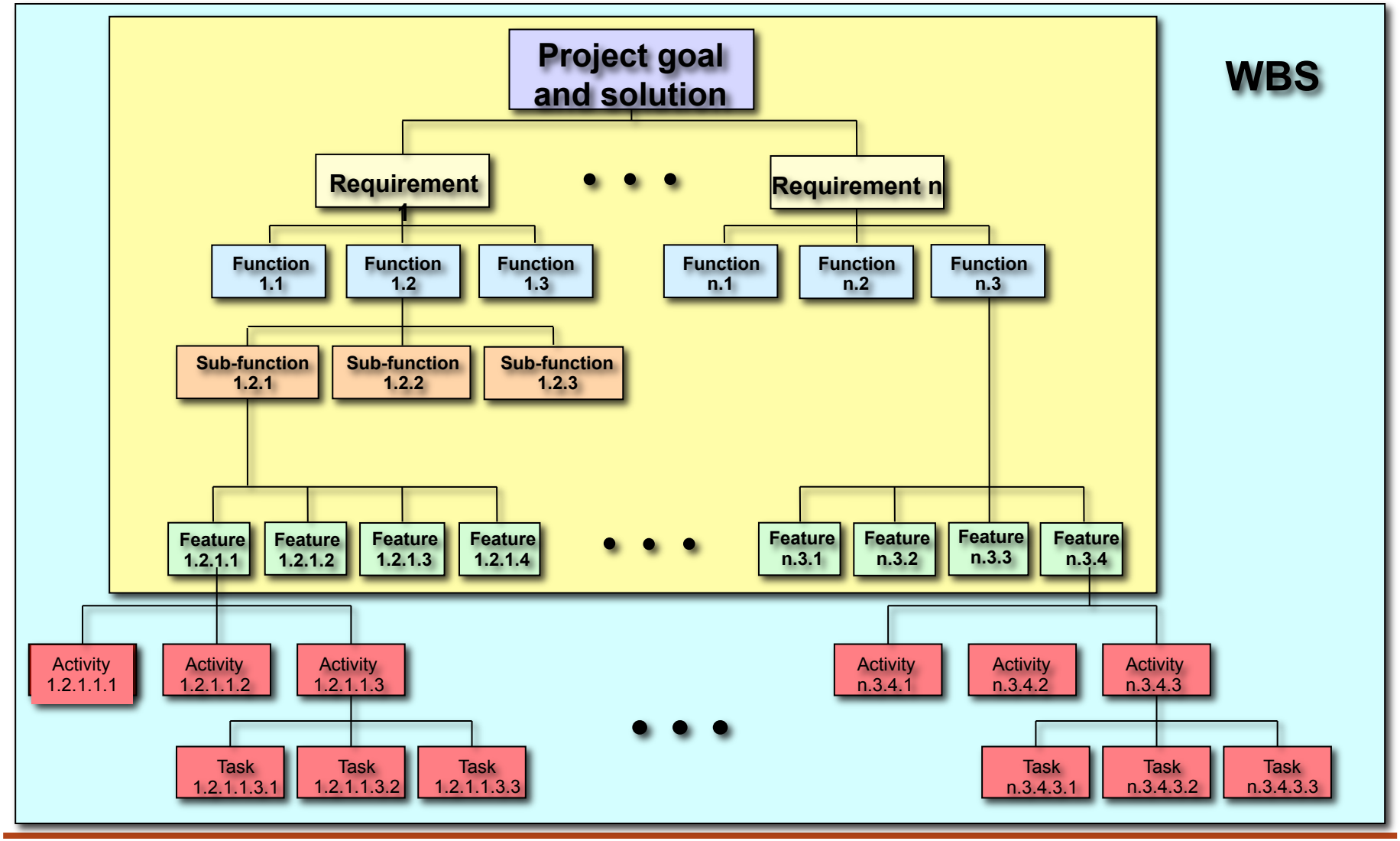


Work Breakdown Structure

The **Work Breakdown Structure** (WBS) is a hierarchical description of all of the work that must be done to meet the needs of the client.



WBS Goals & Requirements



Basic Structures of the WBS

- Deliverables-based structures
  - Physical decomposition
  - Functional decomposition
- Task-based structures
  - Design-Build-Test
- Organizational structures
  - Geographic
  - Departmental
  - Business Function

Homework and Reading Reminders

- Prepare for Tomorrow's Examination
- Complete Homework 4 – Software Risk Tables and Risk Sheets
  - Due by 11:55pm, Tuesday, October 2nd, 2012