605.601 Foundations of Software Engineering

Fall 2020

Module 09: Estimation

Dr. Tushar K. Hazra

tkhazra@gmail.com

(443)540-2230

Course Module 09: Estimation

Topics for Discussion

- Software Project Planning
- Estimation for Software Projects
- Estimation Models
- Estimation for OO Projects
- Estimation for Agile Projects

Course Module 09: Estimation

Software Project Planning

- The overall goal of project planning is to establish a pragmatic strategy for controlling, tracking, and monitoring a complex technical project.

- Why? So the end result gets done on time, with quality!

Project Planning

1. Establish project scope
2. Determine feasibility
3. Analyze risks
4. Define required resources
   - .., Determine required human resources
   - .., Define reusable software resources
   - .., Identify environmental resources

605.601 Foundations of Software Engineering
Course Module 09: Estimation

Project Planning (continued)

5. Estimate cost and effort

- .., Decompose the problem
- .., Develop two or more estimates using size, function points, process tasks or use cases
- .., Reconcile the estimates

Project Planning

6. Develop a project schedule

- .., Establish a meaningful task set
- .., Define a task network
- .., Use scheduling tools to develop a timeline chart
- .., Define schedule tracking mechanisms

Estimation

- Estimation of resources, cost, and schedule for a software engineering effort requires. . .
  - .., experience,
  - .., access to good historical information (metrics), and
  - .., the courage to commit to quantitative predictions when qualitative information is all that exists
- Estimation carries inherent risk and this risk leads to uncertainty

What is Scope?

- Software scope describes. . .
  - .., the functions and features that are to be delivered to end users,
  - .., the data that are input and output,
  - .., the “content” that is presented to users as a consequence of using the software, and
  - .., the performance, constraints, interfaces, and reliability that
  - bound the system.
- Scope is defined using one of two techniques
  - .., A narrative description of software scope is developed after communication with all stakeholders
  - .., A set of use cases is developed by end users

605.601 Foundations of Software Engineering
Course Module 09: Estimation

Project Estimation

- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent in the process

Image source: Pressman, R. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, 7th edition

Course Module 09: Estimation

Estimation Techniques

- Past (similar) project experience
- Conventional estimation techniques
  - .., task breakdown and effort estimates
  - .., size (e.g., function points) estimates
- Empirical models
- Automated tools

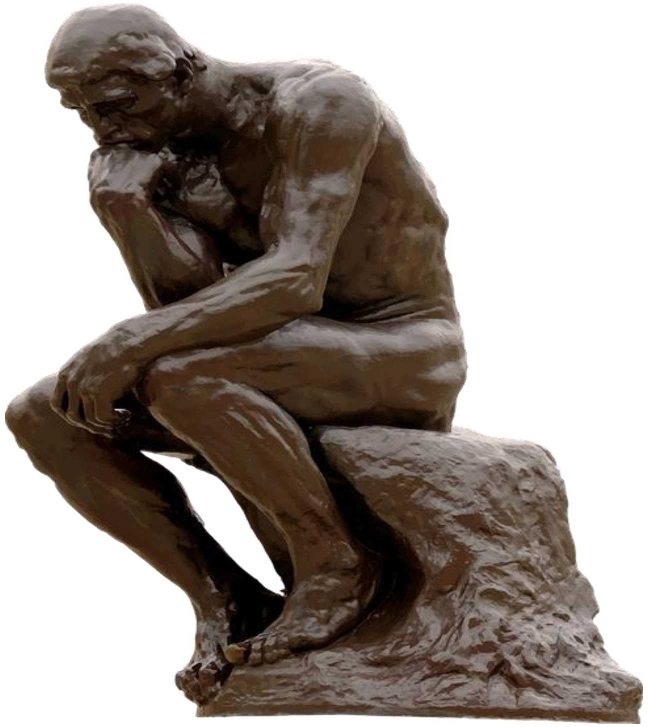


Image source: https://pixabay.com/en/art-auguste-rodin-bronze-famous-1301872/

Course Module 09: Estimation

Estimation Accuracy

- Predicated on. . .
  - The degree to which the planner has properly estimated the size of the product to be built
  - The ability to translate the size estimate into human effort, calendar time, and dollars (a function of the availability of reliable software metrics from past projects)
  - The degree to which the project plan reflects the abilities of the software team
  - The stability of product requirements and the environment that supports the software engineering effort

Course Module 09: Estimation

- Functional Decomposition

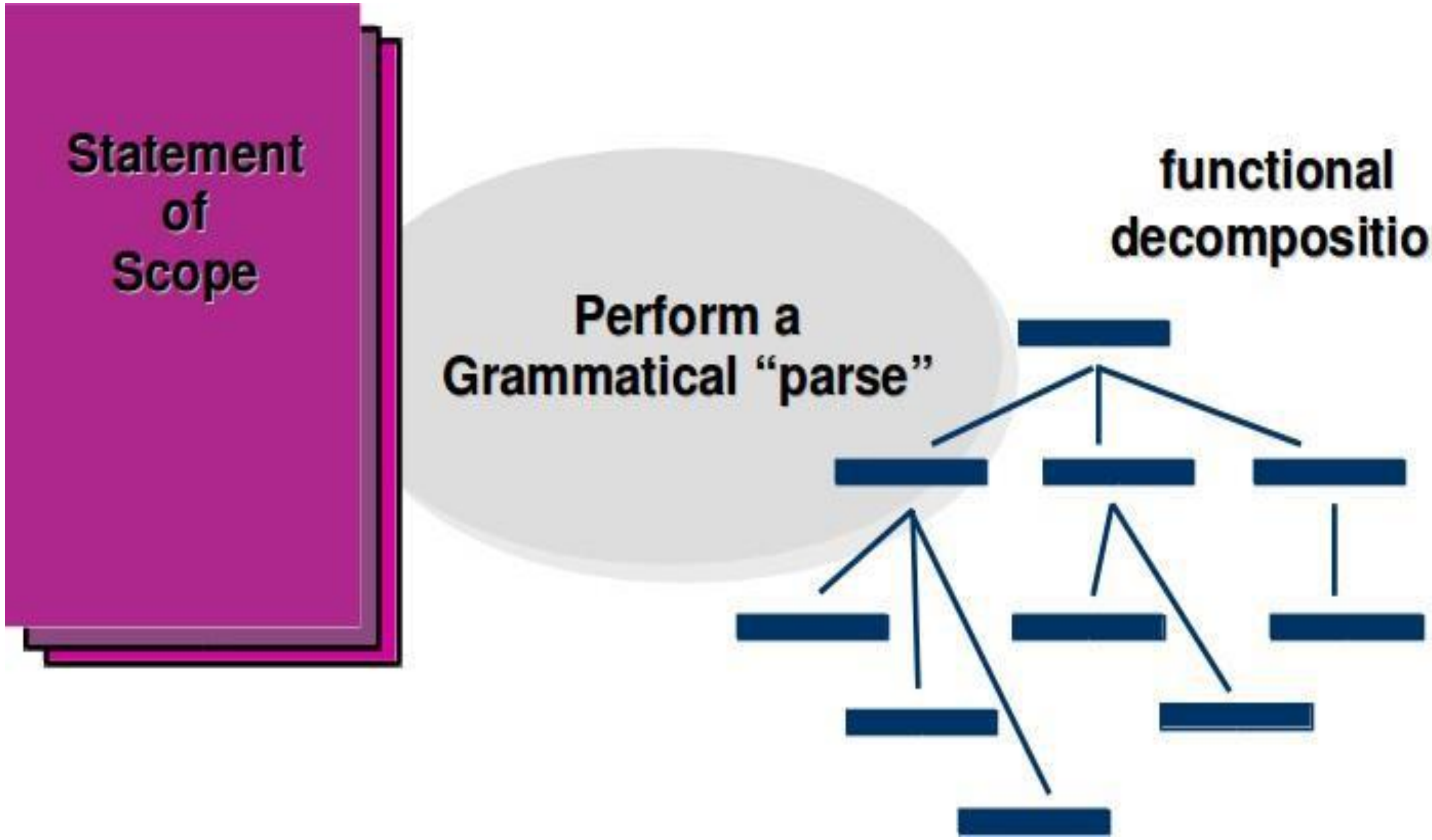


Image source: Pressman, R. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, 7th edition

Course Module 09: Estimation

- Conventional Methods
  - Compute lines of code (LOC) / function points using estimates of information domain values
  - Use historical data to build estimates for the project

- A function point is a unit of measurement derived from the functional user requirements of a software system.

Empirical Estimation Models

- General form:
  - $\text{effort} = \text{tuning coefficient} \cdot \text{size}^{\text{exponent}}$
- where
  - effort is usually derived as person-months of effort required,
  - the tuning coefficient is either a constant or a value derived from the complexity of the project,
  - size is usually LOC but may also be function points, and
  - the exponent is empirically derived

COCOMO-II

- COCOMO II is a hierarchy of estimation models that address the following areas:
- Application composition model Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount
- Early design stage model Used once requirements have been stabilized and basic software architecture has been established
- Post-architecture-stage model Used during the construction of the software

Estimation for OO Projects I

1. Develop estimates using effort decomposition, function point analysis, and any other method that is applicable for conventional applications.
2. Using object-oriented requirements modeling, develop use cases and determine a count.
3. From the analysis model, determine the number of key classes (i.e., analysis classes).
4. Categorize the type of interface for the application and develop a multiplier for support classes:

Estimation for OO Projects II

5. Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.
6. Multiply the total number of classes (key + support) by the average number of work-units per class. Lorenz and Kidd suggest 15 to 20 person-days per class.
7. Cross check the class-based estimate by multiplying the average number of work-units per use case

Estimation for Agile Projects

- Each user scenario (a miniature use case) is considered separately for estimation purposes.
- The scenario is decomposed into the set of software engineering tasks that will be required to develop it.
- Each task is estimated separately. Note: estimation can be based on historical data, an empirical model, or "experience."
  - .., Alternatively, the 'volume' of the scenario can be estimated in LOC, function points, or some other volume-oriented measure (e.g., use case count).

Estimation for Agile Projects (continued)

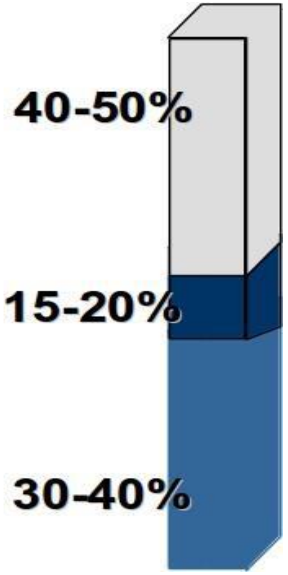
- Estimates for each task are summed to create an estimate for the scenario.
  - .., Alternatively, the volume estimate for the scenario is translated into effort using historical data.
- The effort estimates for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.

- Why Are Projects Late?
  - An unrealistic deadline established by someone outside the software development group
  - Changing customer requirements that are not reflected in schedule changes
  - An honest underestimate of the amount of effort and/or the number of resources that will be required to do the job
  - Predictable and unpredictable risks that were not considered when the project commenced

- Why Are Projects Late? (continued)
  - Technical difficulties that could not have been foreseen in advance
  - Human difficulties that could not have been foreseen in advance
  - Miscommunication among project staff that results in delays
  - A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem

605.601 Foundations of Software Engineering
Course Module 09: Estimation

- Effort Allocation

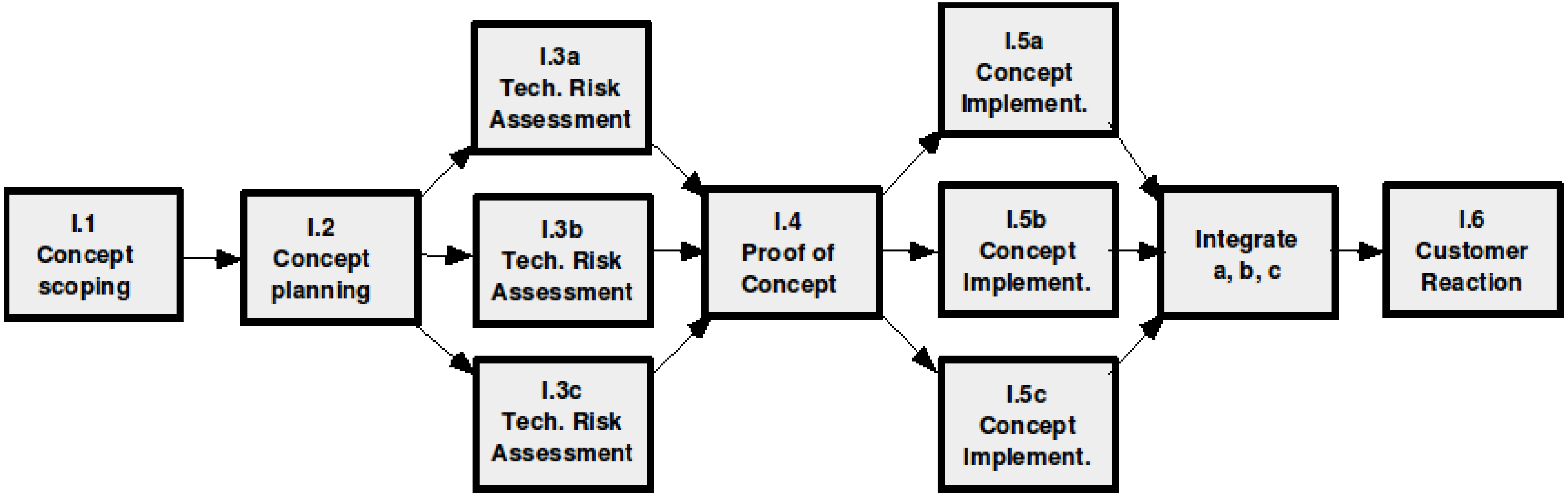


- Front-end activities
  - customer communication
  - analysis
  - design
  - review and modification
- Construction activities
  - coding or code generation
- Testing and deployment
  - Unit and integration
  - White box and black box
  - Regression

Image source: Pressman, R. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, 7th edition

- Defining Task Sets
  - Determine type of project
  - Assess the degree of rigor required
  - Identify adaptation criteria
  - Select appropriate software engineering tasks

- Define a Task Network



Three I.3 tasks are applied in parallel to 3 different concept functions

Three I.3 tasks are applied in parallel to 3 different concept functions

Image source: Pressman, R. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, 7th edition

Progress on an OO Project I

- Technical milestone: OO analysis completed
  - .., All classes and the class hierarchy have been defined and reviewed.
  - .., Class attributes and operations associated with a class have been defined and reviewed.
  - .., Class relationships have been established and reviewed.
  - .., A behavioral model has been created and reviewed.
  - .., Reusable classes have been noted.

Progress on an OO Project I (continued)

- Technical milestone: OO design completed
  - .., The set of subsystems has been defined and reviewed.
  - .., Classes are allocated to subsystems and reviewed.
  - .., Task allocation has been established and reviewed.
  - .., Responsibilities and collaborations have been identified.
  - .., Attributes and operations have been designed and reviewed.
  - .., The communication model has been created and reviewed.

Progress on an OO Project II

- Technical milestone: OO programming completed
    - .., Each new class has been implemented in code from the design model.
    - .., Extracted classes (from a reuse library) have been implemented.
    - .., Prototype or increment has been built.

Progress on an OO Project II (continued)

- Technical milestone: OO testing
  - .., The correctness and completeness of OO analysis and design models has been reviewed.
  - .., A class-responsibility-collaboration network (Chapter 6) has been developed and reviewed.
  - .., Test cases are designed and class-level tests (Chapter 19) have been conducted for each class.
  - .., Test cases are designed and cluster testing (Chapter 19) is completed and the classes are integrated.
  - .., System level tests have been completed.