# CS 320

Introduction to Software Engineering
Spring 2017

February 13, 2017

# Logistics

**Reading assignment**

- Purposes, Concepts, Misfits, and a Redesign of Git (Moodle)
- In-class discussion on 02/22

**Project-related assignment**

- Discussion of requirements analysis techniques on 02/15
- Initial software requirements specification document by 02/24

# Today

**More on version control systems**

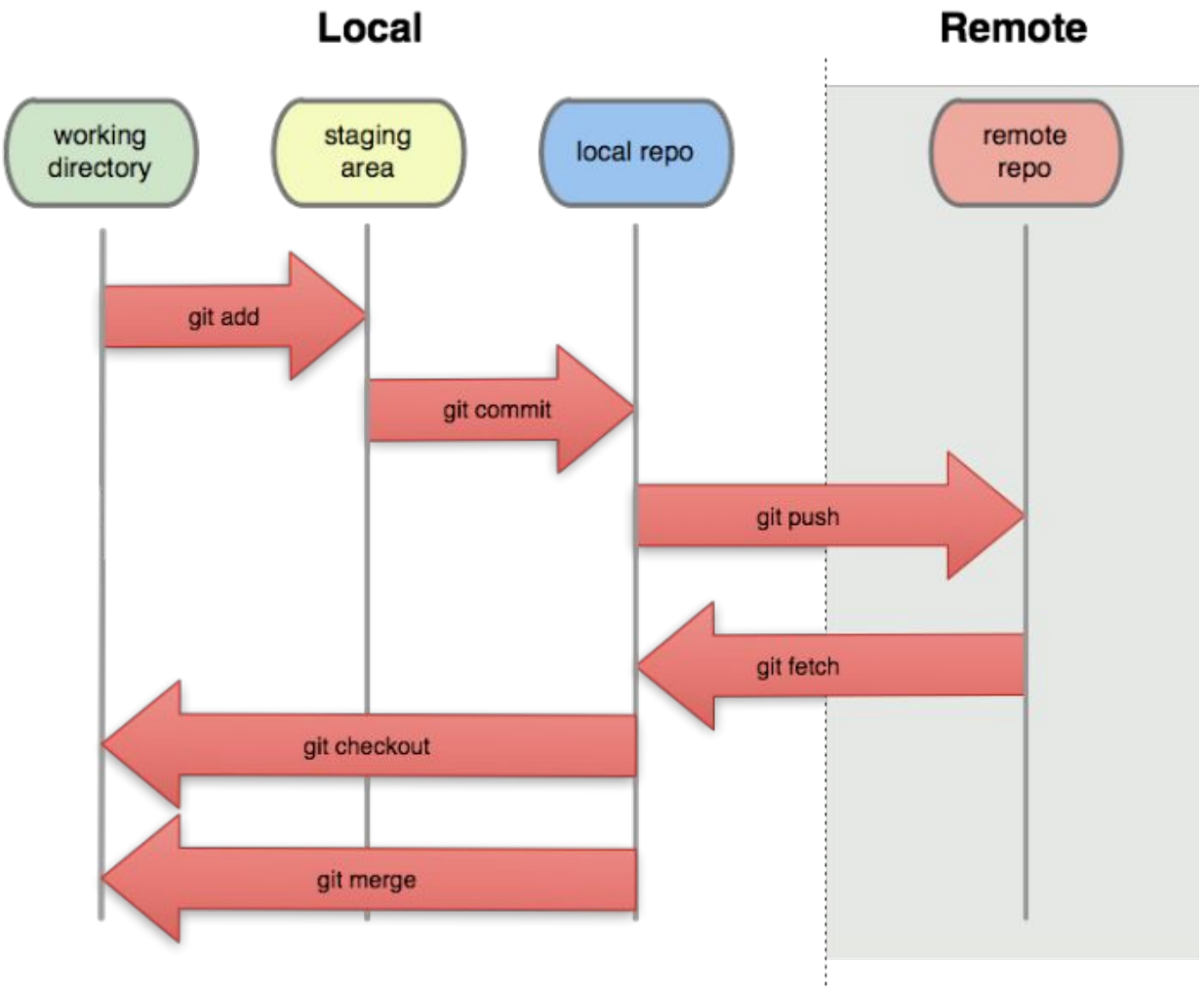
- In-class exercise post-mortem & best practices
- Git concepts and terminology

**Analysis phase: analysis and requirements**

# In-class exercise post-mortem and best practices

Live demonstration, using the basic-stats example.

# Git: concepts and terminology



# Analysis phase: analysis and requirements

# Importance of analysis and requirements

“A complete understanding of software requirements is essential to the success of a software development effort. No matter how well designed or well coded, a poorly analyzed and specified program will disappoint the user and bring grief to the developer.”

*Software Engineering A Practitioner's Approach*, Roger S. Pressman.

# The analysis phase

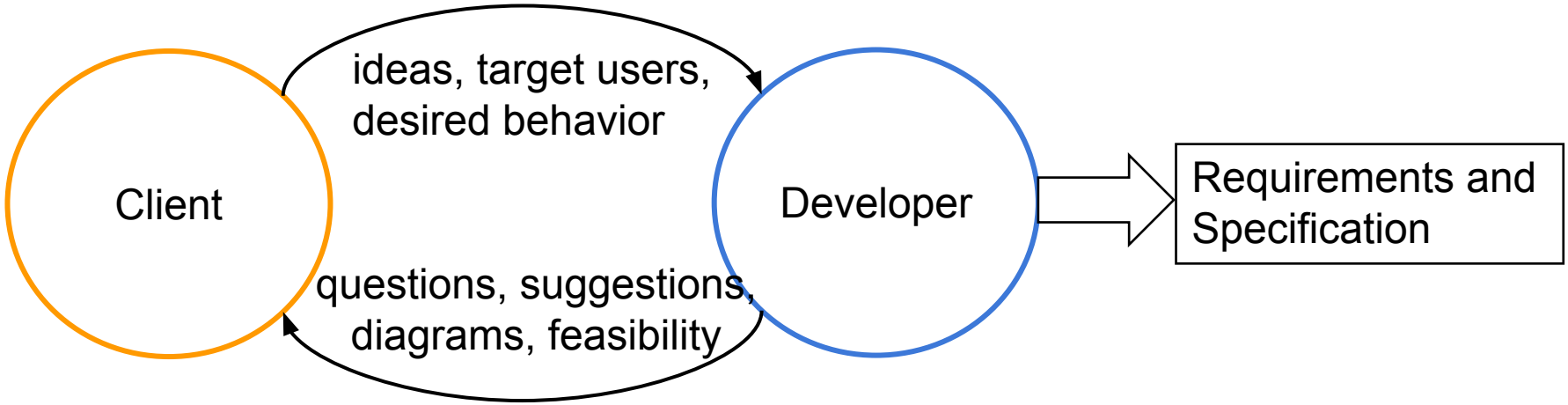
- **Purpose**: obtain a **clear understanding** of the **client's needs** (clear means unambiguous in this context).
- **Software engineer** is an **active participant**. Client may not understand the potential/feasibility of a system.
- Analysis may uncover unknown potential/pitfalls.
- The market potential of a system is determined by its expected behavior but market analysis is not a software engineering task. One pitfall is to agree on building a system that the client "has to have" but is infeasible.

# The analysis phase

- **Discover** and **formally define** the important aspects of a proposed software system.
- It is the process of formally specifying exactly **what** is to be built based on the client's ideas and needs.

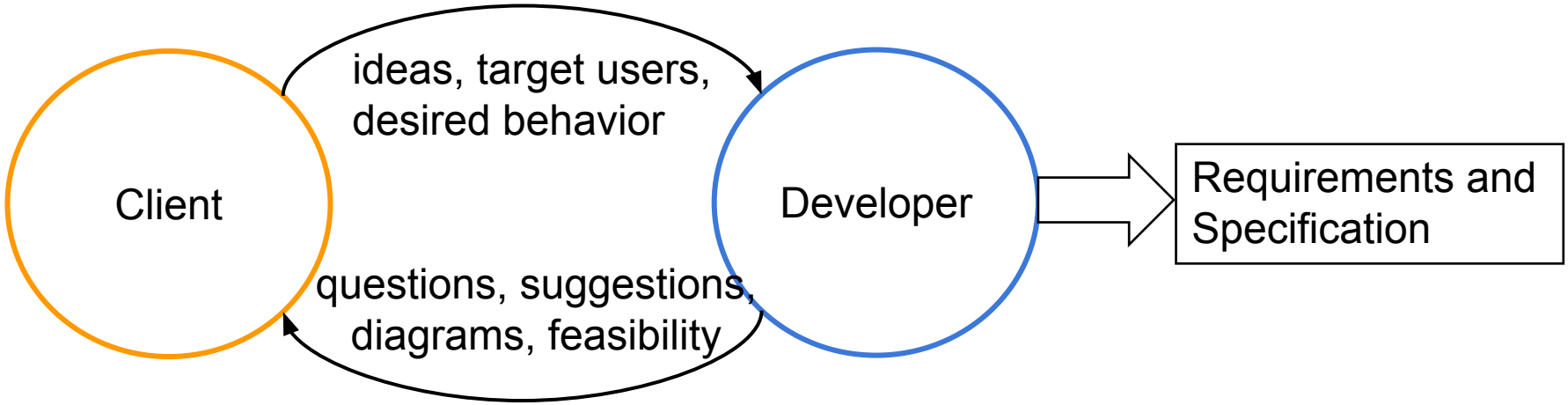
# The analysis phase

- **Discover** and **formally define** the important aspects of a proposed software system.
- It is the process of formally specifying exactly **what** is to be built based on the client's ideas and needs.



# The analysis phase

- **Discover** and **formally define** the important aspects of a proposed software system.
- It is the process of formally specifying exactly **what** is to be built based on the client's ideas and needs.




What questions should we get answered in this phase?

# Analysis questions

The high-level questions analysis answers:

1. What is the purpose of the system?
2. Who uses the system (can be human and non-human)?
3. What functionality/benefits must it provide (what's the desired behavior)?
4. Is it feasible to build the system?

The **answers** to these questions are **formally specified** in a **requirements and specification** document.

# Purpose

- Why does the system exist? What need does it serve?
- Decompose large systems and analyze the subsystems.

# Purpose

- Why does the system exist? What need does it serve?
- Decompose large systems and analyze the subsystems.
- Example: healthcare.gov:

  “The exchange facilitates the sale of private health insurance plans to residents of the United States and offers subsidies to those who earn less than four times the federal poverty line. The website also assists those persons who are eligible to sign up for Medicaid, and has a separate marketplace for small businesses.”, *Wikipedia.*

  Large, hierarchical system with multiple purposes. Each subsystem and the overall system should be analyzed.

# Users

- What **entities** interact with the system?
  An entity can be a **human** or a **machine** (e.g., manager, administrator, customer, or an accounting system).
- What is the context in which the system operates?
- What is the interface between the system and an entity?
  For humans typically a **GUI**, for machines typically an **API**.
- What data is required and will be passed into and out of the system?

# Functionality

- What is the expected behavior the system?
- Discover the core set of functions the system must perform.
- Generally, categorize all functions into must have, should have, could have, won’t have.
- Identify what data each function processes, and map functions to input-output specifications.

# Feasibility

- Is the system possible to build with the expected functionality?
- A feasibility study answers the question and assesses the risk of whether or not the project's goals can be met.
- Four main considerations:
  - Technical: is the project technically possible?
  - Financial: is there sufficient funds for the project?
  - Organizational: will the system be compatible with existing practices?
  - Ethical: is the impact of the system socially acceptable?
- May have to revise scope/functionality to make it feasible.

This course is not concerned with feasibility.

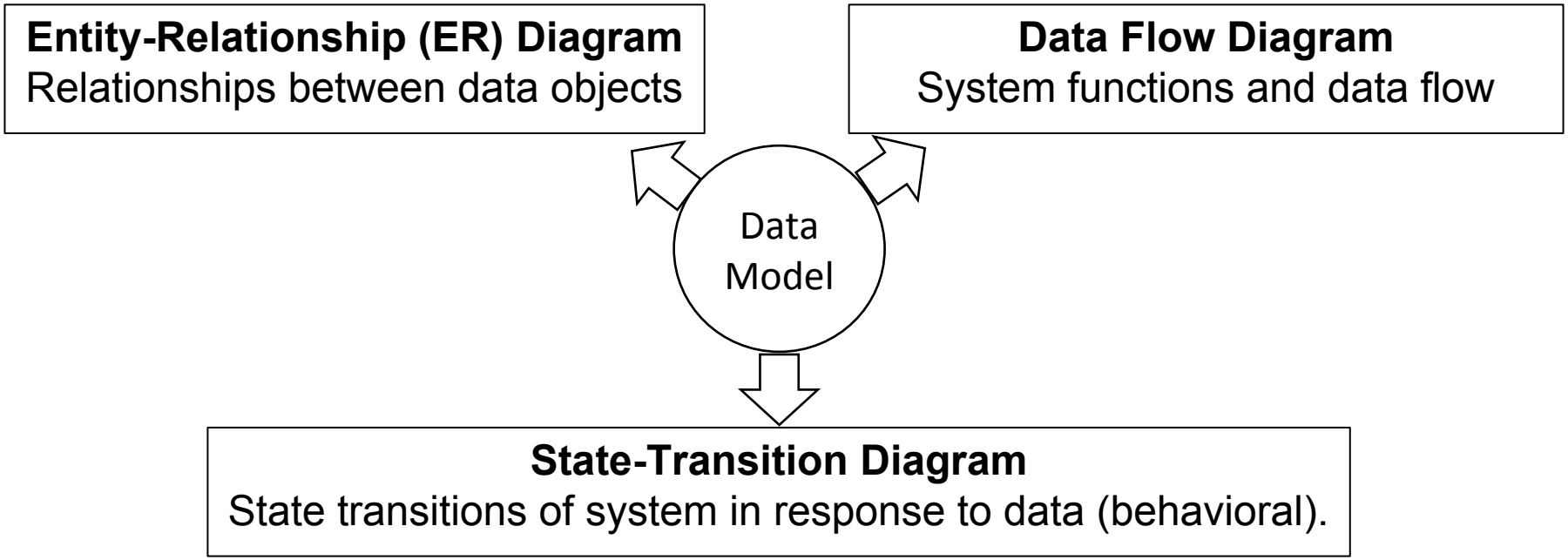
# Analysis techniques

Common analysis techniques:

- Meetings, questionnaires, interviews with users.
- Prototyping.
- User stories, use cases.
- Modeling (Diagramming): Data Flow, State-Transition, Entity-Relationship, and many others.

# Data modelling and diagramming

- Software can be thought of as a system that processes data.
- At the heart of any system, therefore, is a **data model**.
- These three **diagramming techniques provide views** of the **data model** based on the way a system processes data:



# Summary: analysis phase

- The goal of the analysis phase is to formally specify the needs and otherwise informal ideas of a customer.
- Analysis is a collaboration between client and developer.
- There are many analysis techniques that help discover and quantify the behavior and data of a system.
- Analysis can be done on existing systems. It can be, and has been done on systems without any computers at all.
- Analysis leads to a set of **formally defined requirements** that **all stakeholders** can **understand**.