Effective Course Projects for Teaching Distributed-Application Development

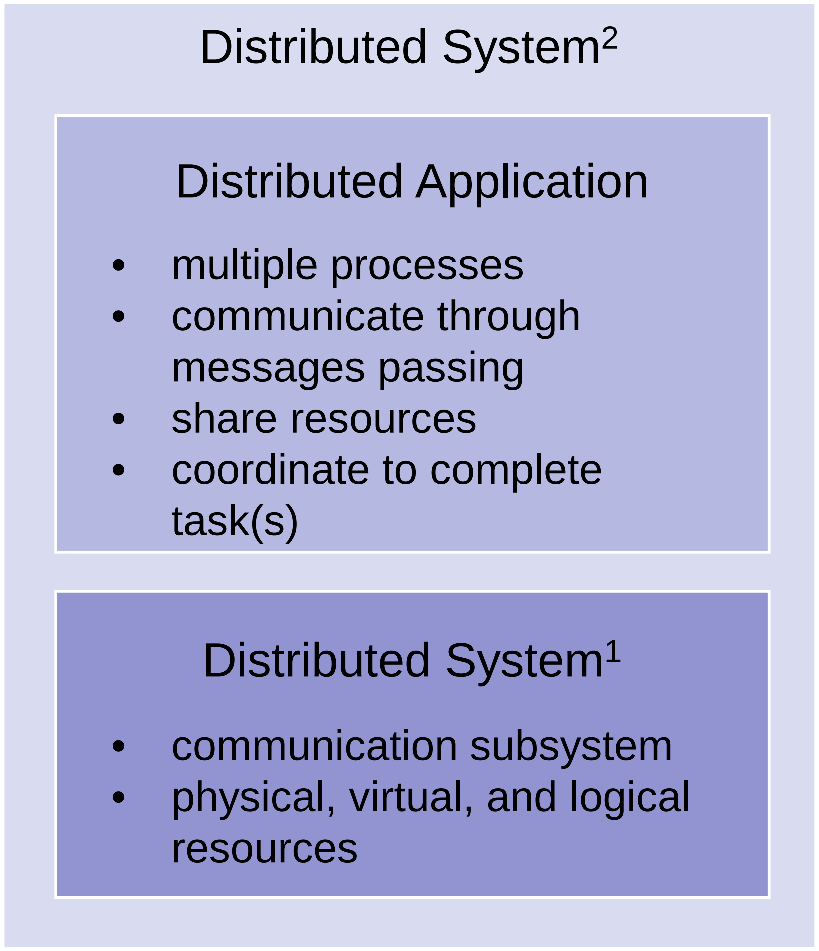
Stephen W. Clyde

Utah State University

Distributed Systems and Applications

Distributed Systems and Applications

- Distributed Application: An end-user system consisting of software components running on multiple host machines that share resources and coordinate their actions to complete a task (or tasks) through message passing
- Distributed System:
  1. A distributed environment in which a distributed application runs
  2. Also, the distributed application and the distributed environment together



Distributed System[2]

Distributed Application

- multiple processes
- communicate through messages passing
- share resources
- coordinate to complete task(s)

Distributed System[1]

- communication subsystem
- physical, virtual, and logical resources

The Need

Students graduating in

Software Engineering, Computer Science, or other related disciplines

need to know how to use, build, test, deploy, maintain, and operate distributed systems and applications

Knowledge, Skills, Abilities

Some Suggested Knowledge

- Underlying Theory of Distribution
- Common system models and architectures
- Desirable characteristics for distributed applications (e.g. extensibility, scalability, maintainability, etc.)
- Design principles
- Best practices for implementation
- Testing theory and principles
- Requirements capture and analysis (including, Business model, who are the actors, their use cases, operational environment)
- Data engineering
- Data science

Some Required Skills

- Network communications
- Inter-process concurrency
- Intra-process concurrency (e.g. Threading)
- Proper handling of partial failures
- Managing multiple concurrent communication channels
- Task synchronization
- Efficient communication protocol design
- Testing and debugging techniques
- Modeling skills (Conceptual)
- Integration (including integration testing)
- Prototyping
- Technology research (and evaluation)

Some Required Abilities

- Evaluating design alternatives
- Making appropriate design choices that balancing requirements, cost, and schedule
- Ability to achieving the following to an appropriate level in a variety of circumstances
  - Reliability, Security, Scalability, Extensibility, Maintainability
  - and other desirable characteristics
- Return-of-investment
- The ability to read, understand, and evolve specification
- Teamwork
- Continuous Improvement
- Realization of abstracts into implementations

A Few Thoughts on Teaching

- Help students
  - Gain knowledge
  - Develop new skills
  - Strengthen abilities
- Encourage students to
  - Discovery ideas on their own
  - Take initiative and be innovative
  - Learn how to learn

Purpose of this Tutorial

- Explore ideas related
  - Designing course projects so they are engaging and cover as many of the knowledge areas as feasible,
  - Coaching students as their develop new skills and to help them successful complete the assignments
  - Evaluating the student performance in constructive ways that helps them improve their ability to solve real problems

Tutorial’s Learning Objectives

- Gain a better understanding of the knowledge, skills, and abilities that students need to be effective distributed-application developers.
- Gain a better understanding of how distributed-application development concepts can be taught in conjunction with good software engineering principles and practices.
- Gaining new ideas about how to make a course project more engaging.
- Gaining new insights into how to better coach students to successful completion of a substantial project.
- Gain new insights into how to evaluate student performance constructively.

Programming Assignments

- What makes a good programming assignment?
  - Relevant to student body and contain course
  - Customized to the right level
    - Leaving open the opportunity to develop skills and improve abilities
  - Real-world problem (from industry)
  - Resume building potential
  - Non-functional requirements

Programming Assignments

- What is not necessary for a good programming assignment?

Example of a Programming Assignment

- Context:
  - OO Software Development Course
  - Seniors and 1st-year graduates
  - 1st programming assignment
  - Current principles
    - Become familiar with abstraction and modularity
    - Become familiar with *Localization of Design Decisions*, part of modularity and based on David Parnas' work on decomposition of modules
  - Current skill
    - The strategy pattern

Example of a Programming Assignment

- Assignment Description
  - Estimated time
  - Learning Objectives
  - Overview
  - Instructions and Requirements
  - Provided codes or materials (if any)
  - Notes and Hints
  - Review and Submission Instructions
  - Grading Criteria

https://www.dropbox.com/s/5v59vele524ht2p/Sample-Assignment.pdf?dl=0

Exercise

- Using the Tello drone, design a programming assignment for the following:
  - Distributed Systems Design Course
  - Seniors and 1st-year graduates
  - 1st programming assignment
  - Current knowledge areas
    - Request-reply communication patterns
    - Intra-process concurrency
  - Current skill
    - Implementing UDP Communications

https://www.dropbox.com/s/zvy7z68dxlpq09k/Tello-User-Guide.pdf?dl=0

Coaching

Coaching

- What can an instructor do to coach or mentor students during a programming assignment?

Coaching

- What kinds of ”help” from an instructor will lessen the students’ opportunities to develop their own skills?

Evaluation (Grading)

Evaluation

- What can an instructor doing when evaluating a student’s performance to help them improve their abilities?

Evaluation

- What should an instructor not do during evaluation?

Summary