Ser321 Principles of Distributed Software Systems

1. **Course Coordinator**  
   Tim Lindquist

2. **Catalog Description**  
   Principles underlying design and implementation of distributed software components; sockets, protocols, threads, XML, serialization, reflection, security, and events. Prerequisite: Ser222 and MAT243.

3. **Credits and Contact Hours**  
   3 credits, 3 contact hours

4. **Prerequisites or co-requisites by topic areas.**  
   Basic Web page construction in HTML.  
   Data structure design and implementation techniques;

5. **Course Overview:**  
   The course explores the concepts and techniques for designing, constructing and deploying distributed and Web-based applications. It examines the basis techniques of secure multithreaded, and distributed programming, and covers the most common distributed computing paradigms along with the frameworks that support the paradigms. Students will develop distributed applications using multiple languages such as Java, and C++. The course covers serialization, asynchronous behavior, and distributed paradigms such as peer-to-peer, client-server, remote procedure call, and object-based paradigms.

6. **Course Resources**  

7. **Course Student Outcomes (CO) and Mapping to Program Student Outcomes**
   a) Students are able to design and develop distributed applications that apply the concepts of concurrency, asynchronous behavior, threads, and shared data synchronization using software engineering best practices. Program Student Outcome supported: SER2.  
   b) Students have knowledge of fundamental program development tools available on Linux: (1) Command-line interpreter, (2) compiler, (3) debugger, and (4) build-tools to develop, test, and deploy distributed applications. Program Student Outcome supported: SER1.  
   c) Students are able to develop distributed applications consisting of multiple classes, multiple programs (such as client-server), and which utilize pre-existing libraries/frameworks (such as GUI, JSON, or JSON-RPC). Program Student Outcome supported: SER1.  
   d) Students have knowledge of serialization: (1) language built-in, (2) XML, and (3) JSON: Understand the role serialization plays in distributed applications; are able to utilize serialization in developing distributed applications. Program Student Outcome supported: SER1.
e) Students are able to apply common distribution paradigms appropriately for solving problems requiring distribution, such as client-server, service-oriented, and peer-to-peer. Program Student Outcome supported: SER1.

8. Topics Covered in the Course
   a) Prerequisite review and introduction to course objectives.
   b) Introduction to Linux and fundamental software development and software build tools, and deployment.
   c) Serialization: Binary, Text-Based (XML & JSON), and custom.
   d) Concurrency, threading, monitors, and synchronizing shared data.
   e) Runtime environments, reflection, program execution paradigms, and use of pre-existing libraries (such as JSON and GUI).
   f) Distribution, sockets, threaded servers, clients, and thread pooling.
   g) Securing distributed applications: secure communications and securing the runtime environment.
   h) Distribution paradigms including: Object-based, remote procedure call, client-server, peer-to-peer, service-oriented, and cloud. Protocols, such as http, dhcp, tcp/ip, and ftp.
   i) Distributed object management: stateless, singleton and session.

9. Assignments:
   a) Individual programming project to successively develop a distributed application with multi-language clients.
   b) Team-based distributed application security and packaging assignment.

10. Grading:
The projects in the course involve software design and implementation, teaming and communication skills. Semester long individual project and team project contributes to 40% of the grade. Two exams (midterm and final) account for 30% each.

11. Course Offering Information
   f) Desired instructor expertise: background in software development on Linux; experience designing, developing and deploying distributed and web applications on multiple platforms; utilizing multiple high-level programming languages.
   g) No community college articulation information is provided since the course is upper-division. For awarding transfer credit from outside ASU as equivalent to Ser321, it is suggested that the course cover 80% of the same topics, that it be part of an ABET accredited bachelors program in Software Engineering or Computer Science, and that it have similar pre-requisites.