Spring 2022 Software Architectures (SWE 443)

Syllabus

Professor: Mike Reep, Ph.D.
E-mail: mreep@gmu.edu
Class Hours: Wednesday, 4:30-7:10 pm
Class Location: Planetary Hall 206
Prerequisites: CS 321, CS 421, SWE 321, or SWE 421
Office Hours: After class by appointment or advance notice

Text

The text is available via GMU library in addition to the bookstore or Packt directly. However, eBook versions are often available from Packt on sale (sometime as low as for $5) as a PDF version to avoid the one-page at a time view from the library access.

- *Software Architect's Handbook*, Joseph Ingeno, Packt Publishing, Aug 2018 (Required)

Catalog Description

Teaches how to design, understand, and evaluate software systems at an architectural level of abstraction.

By end of course, students will be able to:

- recognize major architectural styles in existing software systems
- describe a system's architecture accurately
- generate architectural alternatives to address a problem and choose from among them
- design a medium-size software system that satisfies a specification of requirements
- use existing tools to expedite software design
- evaluate the suitability of a given architecture in meeting a set of system requirements.

Outcomes

These additional learning outcomes provide a focus on current industry activity and the growing use of cloud infrastructure.

Students will be able to:

- explain the role and function of software architecture and the software architect in modern team environments and development methodologies
- use domain driven design to model core business concepts
- identify and write appropriate software quality attributes and requirements

- document software architectures using correctly formed and appropriate UML diagrams
- use agile approaches to complete software development projects
- design and develop code for implementing software architecture patterns
- articulate and implement the core software development principles and practices
- incorporate security considerations into software architectures
- describe the impact and adjustments in developing systems using a cloud environment
- explain approaches for modernizing legacy systems

Class Format

The class is being conducted using a hybrid approach - asynchronous for covering new material and in-person for a review of key points plus in-class group exercises. The goal is to leverage the in-person time for gaining a better understanding of the material, the application of the material, and facilitating group project coordination.

Each week recorded videos and supplemental material(s) are provided on Blackboard for the next set of learning objectives by Friday. Reading assignments in the textbook and in the Blackboard weekly folder along with watching posted videos or links must be completed before the weekly class time period. The videos after the first week will have an embedded quiz question to encourage viewing and evaluate comprehension. These quizzes are included in the overall quiz grade for the class grade.

Exercises are a key component of most class sessions and incorporate material from the reading assignment. These will generally be small groups exercises focusing on the group project although some may be individual or full class exercises. The exercises are announced in class and points are only earned in class. Each class is worth 2 points for the participation grade and watching the required videos before class plus an overall assessment of participation throughout the semester.

Discussion Board

The Blackboard Discussion Board is used to maintain communications between classroom sessions, post a current topics assignment (see below) and allow students the opportunity to interact with each other on the group project. An "Ask the Professor" is provided for any questions or topics that may be of interest to the entire class. These types of inquiries are not accepted by email and must be posted on the Discussion Board for all to see. (Personal or sensitive topics are still handled via email.)

All electronic postings must be professional, respectful, positive and courteous. The Core Rules of "Netiquette" provide guidelines on how to carefully craft your communications in the online classroom to avoid misinterpretation.

Quizzes

We will have weekly quizzes starting the second week. Quizzes are available on Blackboard from Noon until 4:00 pm on Wednesday before class with a 20 minute timeline. (Quizzes are closed-book, closed notes, and no other assistance allowed.) The quizzes will focus on the assigned readings for that class with a emphasis on the weekly learning objectives plus an extra question or two on the technical material. However, material from previous classes may also be included. The quizzes consist of a combination of multiple-choice, true/false, and fill-in-the-blank to evaluate understanding of the terms and concepts. There are no-retakes or make-up quizzes but the lowest four scores are dropped.

Current Topics in Software Architecture

As part of the participation grade, each student is required to read a recent (last 2 years) paper or article on a topic of interest related to software architecture. You will then post a summary of the paper and an associated URL onto the associated Blackboard discussion board. The article is accompanied with an explanation of why the article was of interest and key points for the other class members to take away. The summary and explanation must be at least 200 words although more is encouraged. A rubric will be posted with the discussion thread to provide the basis for grading. You will not be able to read other students postings until you post your own.

Each student is required to read **at least two** of the posted articles and reply to the discussion thread with their own insights, ideas, commentary or questions for class consideration. These reviews are due as listed in the schedule.

Group Project

A group project is a key component of moving from theory to practice. Each group goes through the stages of software architecture from initial problem statement on to requirements, design and implementation. Software Architects must present their work products to be successful so presentations are done at the end of the semester by the entire team. Selected groups will present their material for peer-review and input as part of the in-class discussions. Each group member is expected to participate in all aspects of the effort including coding and presenting. Interim deliverables are scheduled to validate continual progress is being made.

Mid-Term Exam

The mid-term exam will be available on-line for remote access during the assigned class period. The mid-term will focus on the course learning outcomes listed in the syllabus and the weekly learning outcomes posted in Blackboard. A study guide will be provided prior to the previous class sessions to facilitate preparation. The exam is closed book, notes, phone, tablet or any other type of assistance.

Final Exam

The final exam will focus on the material and learning objectives since the mid-term to the extent possible. (The material builds upon the previous topics so it is not possible to completely eliminate any overlap.) A laptop will also be needed to complete the on-line exam during the assigned time period. A study guide will be provided prior to the end of the scheduled class sessions to facilitate preparation. The exam is closed book, notes, phone, tablet or any other type of assistance.

Grading

- Participation: 10%
- Quizzes: 15%
- Group Project: 30%
- Mid-Term: 20%
- Final Exam: 25%

Submission Deadlines

All assignments listed on the schedule are due by 4:00 pm that day unless otherwise noted on the schedule or assignment listing. Late submissions are subject to a 10% penalty for missing the deadline and not accepted after 24 hours without permission prior to the due date. Assignments will be submitted in Blackboard either through Discussion Board forum postings or via the Assignment feature. You are expected to verify your own Blackboard responses by returning to the appropriate place in Blackboard after the work has been posted.

E-mail

I will occasionally send important announcements to your Mason email account. If I am running late for the class or have some other issue that will impact the class, I will make that announcement through Blackboard. Emails sent to the me should start the subject line with "SWE 443"and then include a topic. Questions about the technical material, class policies, discussions or other topics of interest to the entire class must be posted on the associated Blackboard discussion board or *Ask the Professor* discussion board and not sent by email.

My goal is to answer emails and board postings within 48 hours. However, please note that in general I am not able to receive or respond to emails and postings during the business day. If I will be away from email for more than one day, I will post an announcement in the Blackboard course folder. In accordance with GMU policy, all email communication will be sent only to your Mason email account.

Before sending an email, please check the following (available on the Blackboard course menu) unless the email is of a personal nature:

1. Syllabus
2. *Ask the Professor* discussion board
3. On-demand Blackboard videos on how to use Blackboard features, and Technical Requirements.

Feel free to respond to other students in the *Ask the Professor* forum if you know the answer.

Schedule

Every attempt is made to adhere to the original schedule. Changes are made to facilitate learning and provide opportunities to thoroughly address topics within the class. Changes are announced via Blackboard and the revised schedule is posted on the site.

Technology

You will need a reliable computer, functional camera and microphone, and internet access to view course materials in Blackboard, take the quizzes and exams, complete the coding for the group project, and record assignments for the group project which captures the screen and voice.

Social Media

I accept LinkedIn requests from current and former students – please be sure to include the class and year in the request. In general, I do not accept other social media requests on my personal accounts from school or work.

Office of Disabilities

If you need academic accommodations, please see me and contact the Disability Resource Center (DRC) at 993-2474. All academic accommodations must be arranged through the DRC and you must inform me, in writing, at the beginning of the semester. All academic accommodations must be arranged through that office. Please note that accommodations MUST BE MADE BEFORE assignments or exams are due. I cannot adjust your grade after the fact.

Religious Holidays

If you need accommodations for a religious holiday, it is your responsibility to let me know the dates of major religious holidays on which you will be absent or unavailable due to religious observances within the first two weeks of the semester. The university calendar is available at https://ulife.gmu.edu/religious-holiday-calendar/ for your reference.

Safe Return to Campus Statement

All students are required to follow the university's public health and safety precautions and procedures outlined on the university Safe Return to Campus webpage (https://www2.gmu.edu/safe-return-campus). Similarly, all students must also complete the Mason COVID Health Check prior to class. The COVID Health Check system uses a color code system and students will receive either a Green, Yellow, Red, or Blue email response. Only students who receive a "green" notification are permitted to attend class. If you suspect that you are sick or have been directed to self-isolate, please quarantine or get testing. Faculty are allowed to ask you to show them that you have received a Green email and are thereby permitted to be in class.

Students are required to follow Mason's current policy about facemask-wearing. All students are required to wear a facemask in all indoor settings, including classrooms. An appropriate facemask must cover your nose and mouth at all times in our classroom. If this policy changes, you will be informed; however, students who prefer to wear masks will always be welcome in the classroom.

Honor Code Statement

As with all GMU courses, SWE 443 is governed by the GMU Honor Code. In this course, all quizzes and exams carry with them an implicit statement that it is the sole work of the author, unless joint work is explicitly authorized. When joint work is authorized, all contributing students must be listed on the submission and must not include students who did nto participate. Any deviation from this is considered an Honor Code violation, and as a minimum, will result in failure of the submission and as a maximum, failure of the class.

Weekly Schedule

Unless otherwise stated, all assignments are due at **4 pm before the class** in which they are assigned.

To help you manage your schedule and time to complete the individual assignments in this course along with the group project, the class schedule is provided below. If you have a question or concern or encounter a problem about an assignment, please contact me immediately so we can discuss and work out a resolution.

For reading assignments:

- SAH: Software Architect's Handbook (Chapter titles follows the chapter number with SA for "Software Architecture")
- Supplemental: Additional materials beyond the information in SAH will be provided in the weekly folder (including links to external sources or videos).

Class	Date	Reading	Topic	Notes
1	1/26	SAH: Ch. 1 (The Meaning of SA) SAH: Ch. 2 (SA in an Organization) – stop at "Project Management", read section on "Software Risk Management"	Overview: Syllabus, Schedule, Project Meaning of Software Architecture (SA) SA in Organization Architecture in Agile (supplemental) Team Formations (supplemental) Technical: HTML	- Welcome! - Post an Introduction to yourself in Piazza

2	2/2	SAH: Ch. 3 (Understanding the Domain) - stop at “Requirements Engineering” SAH: Ch. 12 (Documenting and Reviewing SAs) – stop at “Reviewing software architectures”	Understanding the Domain - Domain Driven Design Documenting Software Architectures - Architecture Views - UML - C4 (supplemental) Technical: Github and Maven	- Groups Formed with scenario - Quizzes Start
3	2/9	SAH: Ch. 3 (Understanding the Domain) - “Requirements Engineering” to end SAH: Ch. 4 (Software Quality Attributes)	Understanding the Domain - Requirements - User Stories (supplemental) Software Quality Attributes Technical: Spring Framework and Spring Boot	- Current Topics Post Due
4	2/16	SAH: Ch. 5 (Designing SAs) - excluding “Architecture Development Method” SAH: Ch. 7 (SA Patterns) - Skip “Event Driven Architecture” and stop at “Command Query Responsibility Separation”	Designing Software Architectures Software Architecture Patterns I: - Layered - Model-Views Technical:Thymeleaf	- Sprint 1 Review
5	2/23	SAH: Ch. 6 (Software Development Principles and Practices) – stop at “Helping Your Team Succeed”	Software Development Principles/Practices Technical: JPA	- Current Topics Review 1 Due
6	3/2	SAH: Ch. 8 (Architecting Modern Applications) – Stop at “Serverless Architecture)	Architecting Modern Applications - Monolithic - Microservices - Microservices Patterns (supplemental) Technical: Microservices Code	- Sprint 2 Review - Design Method - Component Diagrams - Class Diagrams
7	3/9	SAH: Ch. 9 (Cross-Cutting Concerns) SAH: Ch. 12 (Documenting and Reviewing SAs) - “Reviewing software architectures”	Cross-Cutting Concerns AOP and Spring (supplemental) **Reviewing Software Architectures (in class exercise)** Mid-Term Review	- Prepare for Architecture Review - Current Topics Review 2 Due - Posting of Review on Other Group (After class)
	3/16	Spring Break		
8	3/23	Mid-Term		- Sprint 3 Review with Revised Component and Remaining UML
9	4/6	SAH: Ch. 7 (SA Patterns) - “Event Driven Architecture” and “Command Query Responsibility Separation”	Software Architecture Patterns II: - Event Driven - CQRS	
10	4/13	SAH: Ch. 7 (SA Patterns) – “Service oriented Architecture”	Software Architecture Patterns III - SOA - Pipe and Filter pattern Others	- Sprint 4 Review

11	4/20	SAH: Ch. 8 (Architecting Modern Applications) – "Server-Less Architecture" and Cloud native Applications" SAH: Ch. 10 (Performance Considerations) - stop at "Server Side Caching"	Architecting Modern Applications Server-Less and Cloud Native Performance	
12	4/28	SAH: Ch. 11 (Security Considerations) SAH: Ch. 13 (DevOps and SA)	Security DevOps	- Sprint 5 Review
13	5/4	SAH: Ch. 14 (Architecting Legacy Applications) SAH: Ch. 16 (Evolutionary Architecture)	Architecting Legacy Applications Evolutionary Architecture Final Review Material	
14	5/11		Presentations	- Sprint 6 Ends - Final Submission - Presentation Slide Deck
	5/18		Final Exam	

Project Description

Overview

The project objective is to build a software system according to an architectural design by following each step in a defined architectural process. You and your team will implement architectural patterns currently used in industry and document the architecture system using UML diagrams. Once your architecture has been reviewed by another team following a structured format, you will code these patterns into working software.

The primary purpose is to follow architectural processes, use architectural concepts and implement architecture patterns. (We are not trying to build fancy systems with sophisticated algorithms and beautiful user interfaces. Learning and implementing the processes and patterns are key - not the system itself.)

Software development is a team activity so this project is performed in a group. The overall success of a project relies on each team member and, at the same time, each team member is judged (graded) based on their participation, effort and outcome. The evaluation is done by the team management (myself) along with input from the team members.

Agile is one of the most common approaches to developing software. We will use the basics of agile within the constraints of a university project. We will use two week sprints to iteratively and incrementally develop the project with a review for feedback at the end of each sprint. The team will prepare the project documentation in a wiki to facilitate team communicates and update the documentation in an agile manner (incrementally and iteratively). Each team will discuss their efforts with the successes and opportunities for improvement documented and addressed in the next iteration.

Learning Outcomes

The following learning outcomes from the course syllabus are applicable to this project.
The student will be able to:

- describe a system's architecture accurately
- generate architectural alternatives to address a problem and choose from among them
- design a medium-size software system that satisfies a specification of requirements
- use existing tools to expedite software design
- evaluate the suitability of a given architecture in meeting a set of system requirements.
- use domain driven design to model core business concepts
- identify and write appropriate software quality attributes and requirements
- document software architectures using correctly formed and appropriate UML diagrams
- use agile approaches to complete software development projects
- design and develop code for implementing software architecture patterns
- articulate and implement the core software development principles and practices

Steps

The following components are in the project:

- Prepare a scenario and model to establish the business functionality for the software system
- Develop functional and architectural requirements along with software quality attributes as the basis of the system
- Select an Architecture Design Method and follow the steps in the method with appropriate documentation
- Document the software architecture with selected development principles and architecture patterns
- Explain your architecture to another team in accordance with a selected review approach
- Code components of the architecture to demonstrate successful implementation
- Record a demonstration of your part of the project with an explanation of how the architectural requirements, patterns and components were addressed,
- Prepare a presentation summarizing the project and give the presentation to class

Scenario

Each team needs to come up with a scenario for a sample organization and business system. Your scenario starts with information on your organization including name, business area, and target functionality for the system.

The scenario must have at least three separate functional areas that work together into one system. (The number is adjusted based on the number of team members – one per member.) Each functional area will need to have at least one screen with basic capabilities. Each functional area must also include processes for data storage, data retrieval for display, and updates to the data store from user input.

Requirements

Each team will develop the functional requirements for each epic that explains how the system will operate. You will then develop a domain model to set the context for your organization along with epic user stories and then use cases.

The team must also come up with 12 quality attributes with at least one requirement in each of the areas covered listed in the text. For each attribute, an explanation is required on how the architecture implements or addresses the attribute in the final submission.

Architectural Design Method and Architectural Review Methods

Each team will pick one of the design methods presented in the book and then follow the outlined steps. The submission must show evidence that each step was performed with consideration given for completeness. The team may also pick a design method not in text with prior approval. The same process will be conducted to perform an architectural review in class.

Software Architecture

Each of the functional areas will implement a model-view pattern for the user interface and use the microservice pattern for the underlying service(s). Other architectural patterns may be used with prior permission. The architecture must clearly show the implementation of both patterns.

Each functional area must call and use data from the API of another service developed by a different student. The appropriate UML documents must be provided for the system and each selected pattern. (Note: the goal is not to show the whole system but the interactions between the functional areas.) The architectural description must also describe the development principles implemented in each functional area.

Project Coding Preparation Assignments

Throughout the first 1/2 of the semester, technical assignments and reading material is assigned to prepare for the project coding using the approaches outlined for the project. The topics include a foundation in Spring and Spring Boot, GitHub, Maven, basic UI, database use and microservices structures. The end result is a simple, sample project with a user interface, database repository and separate service component.

Development

Each functional area must be substantially coded by a different team member. Teamwork is encouraged in overcoming development issues, ensuring the various parts work together and that there is a common look and feel to the application. However, each person's code must be their own work. The use of Spring Boot for the microservices component is required unless another approach is approved in advance. In order to appropriately implement a microservices architecture, two separate Java projects will be required. The final submission must map the code modules to the UML diagram describing the component.

Agile Approach

In the spirit of agile, the team will develop the system incrementally with a review at the end of each two-week sprint. (There are six sprints in the project with three weeks allocated for the sprint over Spring break.) The team will conduct a short sprint planning session at the beginning of Sprint 4 to determine what will be delivered at the end of each sprint. Overall, the team will develop the system incrementally to build upon the previous work for the final delivery. So the team will have the opportunity to refine and improve project artifacts as the project progresses. At the end of each sprint, the team will conduct a retrospective to review what went well, what did not go so well, and how to improve with the results documented in the wiki.

Wiki

In modern development efforts, the artifacts developed by the team are publicly available and updated real-time on a common portal. (As opposed to a Word document stored on a protected server.) The team will follow this model in the development of the class project and as the basis for reviewing each other's work.

Final Submission

The team will prepare a document to explain their overall project and summarize the group aspects of the effort. An introduction to the overall scenario, the problem being solved by the system, lessons learned section, and system-wide UML diagrams should be divided among the team. The whole paper should read as a team effort. Each team member will also record a video for their final submission to demonstrate their functional area is operational. The video will explain the user interface as compared to the system architecture, development principles and the underlying code. Access to a source code repository must be provided for a code review (Zip files with code are not accepted.)

Presentation

Each team will have time allocated to present their project to the class. The first period is allocated to an overview and appropriate diagrams that show the overall system. Each person must then present their own work by describing the functional area and patterns implemented in

a 5-minute period. The presentation will end with a summary of lessons learned and best practices identified during the effort.

Extra Credit

Each team or student can propose additional tasks to be undertaken as part of the project to enhance learning, exploration of advanced topics and provide "resume enhancers". The points assessed to the extra credit will vary based on the complexity and effort involved. Potential areas for extra credit are implementing an Continuous Integration pipeline for automated build process using Jenkins, implementing automated testing, coding unit tests and demonstrating substantial code coverage and deploying the solution to a cloud environment.

Deliverables

The following items will be graded for the project with rubrics provided for each item.

Week	Summary	Points
4	Sprint Review 1 including scenario, domain model, epics, requirements and software quality attributes	10
6	Sprint 2 Review including design method, use cases, component diagrams and class diagrams	10
7	Review other groups models	5
8	Sprint 3 review with microservices details and additional UML diagrams plus updates to previous architecture and UML diagrams from feedback	15
8	Project Coding Preparation Assignments	5
10	Sprint 4 review (based on team sprint planning)	10
12	Sprint 5 review (based on team sprint planning)	10
14	Final Submission at end of Sprint 5	30
14	Presentations	5

Assignments are due by 4 pm of the class day. Each assignment will have one submission for the group and one submission for each team member.