CSSE 374 – Software Architecture and Design I

Scoring Rubric for Milestone 5

We will focus on four areas in grading this assignment and they will be examined in light of the evaluation table at the end of this Rubric.

1. The teams should update their Domain Model, System Sequence Diagrams, Operation Contracts, Interaction Diagrams, Logical Architecture, and Design Class Diagrams from Milestone 4 based on the feedback provided on the paper. There is an implicit expectation that they will continue to refine their analysis and design models to reflect emerging design decisions.
    a. Domain Model (DM) – teams should provide a DM that outlines the primary conceptual classes with their key attributes. There should be no operations in the class. Associations and dependencies should be also reflected in the model.
    b. System Sequence Diagrams (SSD) – the teams should provide SSDs describing the behaviors and events between the junior project system and key actors in the application domain. For some projects this will be more challenging than others. To grade this effectively, you should look at how these diagrams capture key operations with relevant parameters. The event arrows should go to and from the :System with the appropriate arrow heads and line types (solid arrow heads and lines for synchronous events and stick arrow heads and dashed lines for asynchronous events). Note that there can be more than one actor working with a :System in the middle so long as the actors are outside the system. The System Diagrams will describe these operations with classes within the :System as part of #4 below.
    c. Operation Contracts (OC) – the teams detail key operations more formally in their analysis model (Chapter 11) using OC. There should be a title, short description, cross-references (for Use Cases and SSDs), Preconditions, and Post-Conditions. Post-Conditions must be stated in the past tense. See Chapter 11 and slides on OCs for details on post-conditions. In particular, the post-conditions should create/delete/update class instantiations and/or associations from the domain model.
    d. Logical Architecture – Using the analysis model elements (DM, SSDs, and OCs), the team should formulate the allocation of classes to packages based on guidelines from Chapters 12 & 13 of the text (allocate the packages to appropriate layers and partitions). Note that the primary focus is on the Domain Layer, but other layers like the UI and Technical Services layers should be present. The team should indicate key dependencies between packages and/or classes in packages, and describe why they are there (either through note tags embedded in the model and/or in a textual description that follows the diagram). The textual description presents rationale and assumptions for the elements in the model (e.g., incorporated appointment in schedule package since a schedule consists of appointments).
    e. Interaction Diagrams (ID) – using relevant system operations the team should develop Sequence Diagrams (SD) and/or Communications Diagrams (CD) as appropriate, that model the key behaviors for their system as it is implemented at the end of the term. These diagrams should show the detailed messages and objects involved in implementing the operations (Chapter 15). Again, each diagram should have some textual description or embedded notes.
    f. Design Class Diagram (DCD) – the team should produce a set of DCD for their system as it is implemented at the end of the term following the guidelines in the book (Chapter 16) and discussed in class. Note that this means progressing from the Domain Model classes into more detailed design classes that contain attributes, operations, and have relationships between classes for dependencies, various associations, aggregations/compositions, generalizations, and

the like. While aggregations/compositions and generalizations need not have labels, most of the others should have some labels indicating the association or dependencies.

2. The team should identify as many of the 9 GRASP principles as is possible in their design (Low Coupling, High Cohesion, Information Expert, Creator, Controller, Polymorphism, Indirection, Pure Fabrication, and Protected Variations) and describe how they are used to arrive at their design. Similarly, the Gang of Four (GoF) patterns covered before Chapter 35 must be addressed (those afterwards are offered as Extra Credit. If they have made the design changes based on the tradeoffs presented they get full credit. If they just describe what they would do, they get 80% credit. They should reference their DCDs and interaction diagrams. The objective of this task is to compare their design to alternatives and reason for a selection that improves their solution.
3. Iteration 3 (a working version of the system, though some features and advanced use cases might be omitted) – the teams must build upon their work from Milestone 4 by refining the classes for their domain layer and implementing their user interface. They should follow the guidelines from Ch. 20 to transition their designs into code. The acceptance test plan from CSSE 371 should be updated as necessary to reflect the system as it will be delivered. If there are requirements that are yet to be implemented, these should also be noted.
4. The teams will demonstrate their software for this third iteration by Wednesday of finals week (at a time determined by team and instructor). The teams will also walk through their code with their instructor demonstrating how their code corresponds to their design documents. While it will not be feasible to evaluate all code elements in this way, the key elements should be examined.

As always, they should provide accompanying text and/or embedded notes indicating what they did in their modeling where it is not clear from the diagram alone. The models and information should be communicated in a way that a reasonably knowledgeable software engineer could understand. Hence, presentation or polish is important – not necessarily pretty, but complete, unambiguous, and comprehendible. Further, the information between the models should be relatively conflict free.

Be mindful that in 10th week, I will be sending a message to **your client** about your performance working with them and their impression of your product thus far. You should have shown the system to your client along with your design for their buy-in.

Excellent work (A) would include a large segment of the things listed above. Major points are taken for one of the key task items missing or largely incomplete. The final project is graded from 0 to 100, with:
90-100 points earned for an A (superior or excellent work),
80-89 points earned for a B (very good work),
70-79 points earned for a C (reasonable work),
60-69 points earned for a D (poor work), and
0-59 points earned for an F (unacceptable or very poor work).

Use the tables on the following pages for grading the overall document after leaving comments in document for recommended improvements.

Completeness Checklist for Milestone 5

- ☐ Domain model
- ☐ System sequence diagrams
- ☐ Operation contracts
- ☐ Logical architecture
- ☐ Interaction diagrams
- ☐ Design class diagram(s)
- ☐ Analysis of the GRASP Principles
- ☐ Analysis of the use of GoF Patterns
- ☐ Acceptance test plan
- ☐ Code
- ☐ Functional Demonstration
- ☐ Demonstration of correspondence between design document and code
- ☐ System shown to client along with design document to get their buy-in

Scoring Rubric for Milestone 5

Criteria (weight)	5 Exemplary	3 Satisfactory	1 Needs Improvement	Weighted Score
Professionalism (×2)	Document is neatly drawn and formatted. (Apart from any problems with the notation) it could be shared with a stakeholder without changes. Document is free of errors in spelling, grammar and punctuation.	Document is somewhat sloppy, but could be shared with a "real-world" stakeholder after some revisions. Document has a small number of errors in spelling, grammar, or punctuation.	Document is largely unprofessional. It would have to be largely reworked before sharing the document with a savvy stakeholder. Document has many errors in spelling, grammar, and punctuation.	
Cohesiveness (×2)	The parts of the document reinforce each other. Each piece is consistent with the others and the document as a whole tells a story.	The parts of the document mostly reinforce each other. Each piece is generally consistent with the others with just a few minor differences.	The parts of the document are disjointed. They are largely inconsistent, to the point that it is unclear whether they describe the same system.	
Clarity of Diagrams (×2)	Diagrams are well labeled and at an appropriate level of abstraction so that stakeholders familiar with the problem domain could readily understand them.	Diagrams are mostly well labeled, with no more than 15% cryptic labels. Diagrams are generally at an appropriate level of abstraction, though a stakeholder familiar with the problem domain might need some guidance to understand them.	Labels are often cryptic or abstraction is used to the point that the actual analysis and design implications would be obscured to all but an expert in both the notation and the domain.	
Conciseness of Diagrams (×1)	Diagrams appropriately use the abstraction features of the notation to minimize useless redundancy	Diagrams may include some unhelpful redundancy, but the general representations are still readily comprehensible	Diagrams are highly redundant to the point that they are difficult to comprehend.	
Effectiveness of Analysis (×2)	Analysis artifacts identify all important domain concepts and clearly define the system interface. They demonstrate a deep understanding of the problem domain.	Analysis artifacts identify many important domain concepts and define the system interface. They demonstrate a reasonable understanding of the problem domain.	Analysis artifacts identify only a few of the domain concepts or only cursorily define the system interface. They betray a superficial understanding of the problem domain.	
Effectiveness of Design Models (×3)	Design conveys all important elements, constructs, and behaviors. It demonstrates a deep understanding of the solution to the problem.	Design conveys many key elements, constructs, and behaviors. Some situations might be treated in an unusual manner, but such treatment is documented.	Design minimally conveys key elements, constructs, and behaviors. It shows a superficial understanding of the problem and its solution.	
Correctness of Solution (×3)	The design is viable within assumptions and rationale presented. Key tradeoffs are successfully analyzed and defended.	The design is largely viable within assumptions and rationale presented. Key tradeoffs are presented, but may not be fully or clearly analyzed.	The viability of the design is questionable. Some assumptions and rationale lacking. Key tradeoffs are missing or may be poorly analyzed.	
Elegance of Solution (×2)	Design effectively applies GRASP principles and GoF design patterns to reduce coupling, increase cohesion, and lower the representation gap.	Design often applies GRASP principles and GoF design patterns to reduce coupling, increase cohesion, or lower the representation gap	Design does not seem to apply GRASP principles and GoF design patterns. It is ad hoc and does not demonstrate commonly accepted design practices.	

Criteria (weight)	5 Exemplary	3 Satisfactory	1 Needs Improvement	Weighted Score
Discussion of Patterns – GRASP and GoF (×2)	Document discusses the application of design patterns such that design decisions are clearly communicated and supported.	Document discusses the application of design patterns, demonstrating a basic understand of the patterns, but not consistently showing how those patterns informed the design decisions made.	Document discusses design patterns in a cursory manner or not at all.	
Correct Use of Notation (×2)	All notation used in the diagrams is appropriate to the diagram type and is used correctly.	All notation used in the diagrams is appropriate to the diagram type. At most two sorts of errors are made in the application of each diagram type.	Diagrams use notation inappropriate to the diagram type or contain a large variety of errors in the application of the notation.	
Software Demonstration (×4)	Software is free of obvious defects. Demonstration told a story. The important features of the system were covered in a compelling way that made clear how the problem was solved from the user's perspective.	Software shows no more than 4 obvious defects. Demonstration provided concise, but thorough review of the system that made clear how the problem was solved from the user's perspective.	Software shows 4 or more obvious defects. Demonstration was either incomplete or was just a litany of features.	
Software Style (×1)	Code is clear and well documented with consistent and appropriate naming and formatting. No "magic numbers" are used.	Code is mostly clear and well documented. The majority of identifiers are well named and the formatting is mostly consistent. No "magic numbers" are used.	Code is often unclear or undocumented. Obscure or terse identifiers are the norm. Formatting may be inconsistent. "Magic numbers" may be used.	
Correspondence of code and design (×4)	Code for the system is consistent with design diagrams, both structurally (as documented by Design Class Diagrams) and behaviorally (as documented by Interaction Diagrams).	Code for the system is mostly consistent with design diagrams apart from a few minor discrepancies.	Code for the system is inconsistent with the design diagrams.	
			Subtotal Score (Sum of above):	
			÷ 1.5 = Subtotal %:	
			× (% of Assignment Completed):	
			= Total Score:	