# **Software Engineering Project Report**

**Report #2: SYSTEM DESIGN -- Iteration 1(b) -- due dates for individual parts of this report are** [**given here**](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/syllabus.html#DELIVERABLES)**:**

This report shall be submitted in three steps:

1. Part 1 (Section 1 *Interaction Diagrams*)

2. Part 2 (Section 2 *Class Diagram* and Section 3 *System Architecture*)

3. Entire Report #2 (all sections included)

**Note**: Keep this report for future reference because it will be part of the final report ([Report #3](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report3.html)). Thus, as you progress through the semester, take every opportunity to revise, update and/or correct it.

While preparing this report, you may wish to check the [**grading checklist**](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report2-grading.html) and avoid some typical mistakes in your report.

## **1. Report Format**

When submitting a part or the entire report, include the following sections and sub-sections. Each section/subsection should be clearly delineated with a proper heading.

* Cover Page and Individual Contributions Breakdown, as specified [**here**](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/projects.html#REPORTS).
* The contributions breakdown must contain the *responsibility matrix* and *responsibility allocation chart* as described in [Report #1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html#EFFORT)
* Table of Contents
* Make sure that the page numbers listed here are correct

**Part 1:**

1. Interaction Diagrams
   * Do [interaction diagrams](http://en.wikipedia.org/wiki/Unified_Modeling_Language#Interaction_diagrams) for the use cases you elaborated (“fully dressed”) in [Report #1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html). You should do at least [sequence diagrams](http://en.wikipedia.org/wiki/Sequence_diagram), but you may do some other UML interaction diagrams, as well.
   * **Describe** what design principles you employ in the process of assigning responsibilities to objects. This can be done either as comment “bubbles” in the diagram, or in the caption of the diagram.
   * Read the [UML textbook](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/index.html#TEXTBOOK) about interaction diagrams and Section 2.5.1 in the [**course lecture notes**](http://www.ece.rutgers.edu/~marsic/books/SE/), in order to learn about good design principles
2. Here are examples of [software tools for UML diagramming](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/projects.html#UML-TOOLS) available for free download

* Project Management (described in Section 7 below) and References (described in Section 8 below)

**Part 2:**

1. Class Diagram and Interface Specification
   1. Class Diagram
   2. Show all classes and their associations. Only indicate visibilities of attributes and operations; full details about the types and signatures should be provided in the next item.
   3. If you cannot fit the class diagram on one page, or it looks too cluttered, create one “overview” class diagram showing all classes and their relationships, but for each class show only a single compartment with the class name (leave out attributes and operations).
   4. Then on subsequent pages show partial class diagrams, with three compartments and all the attributes and operations of a class. Make sure to indicate in diagrams and describe in text how partial diagrams fit into the overall class diagram.
   5. Data Types and Operation Signatures
   6. Independently of the class diagram, write down class specification in UML notation. For every class, specify data types of all attributes and operation signatures.
   7. **Define** the meaning of each class, operation, and attribute in plain language.
   8. Traceability Matrix
   9. Show how your classes evolved from your domain concepts. Provide explanations for modified names or multiple classes that evolved from a single concept—providing only a matrix with checkmarks is *not* enough!
   10. Note that if your system has many classes, the matrix may become difficult to read. Alternatively, you may list, line-by-line, all your domain concepts and explain which classes were derived from each concept and why.
2. System Architecture and System Design
   1. Architectural Styles
   2. Describe the [architectural styles](http://en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_.2F_Patterns) used in your design (Google search [software architectural styles example](http://www.google.com/search?hl=en&q=software+architectural+styles+example&aq=0&oq=software+Architectural+Styles))
   3. Identifying Subsystems
   4. Draw and describe **UML package diagram** of subsystems in your system (See the [UML textbook](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/index.html#TEXTBOOK))
   5. Mapping Subsystems to Hardware
   6. Does you system need to run on multiple computers? For example, you may have a client (web browser) and a server (web server) subsystems, running on different machines.
   7. If NO, skip to the next item;
   8. If YES, what subsystems run on which machine?
   9. Persistent Data Storage
   10. Does your system need to save data that need to outlive a single execution of the system?
   11. If NO, skip to the next item;
   12. If YES, identify the persistent objects and select the storage management strategy, e.g., flat files, relational database, etc.
   13. Attach the description of the file format and/or **database schema** (format of database tables, printed by the command description)
   14. Network Protocol
   15. If your system runs on a single machine, this question probably does NOT apply; skip to the next item;
   16. Otherwise, which communication protocol do you use, e.g., plain Java sockets, Java RMI, Java JDBC, HTTP, etc. Explain why you made your specific choice.
   17. If you are using plain sockets, attach the description of the communication protocol (types of messages, message format, etc.).
   18. Global Control Flow
       * *Execution orderness*: Is your system *procedure-driven* and executes in a “linear” fashion, where every user every time has to go through the same steps, or is it an *event-driven* system that waits in a loop for events, and every user can generate the actions in a different order?
       * *Time dependency*: Is there any timers in your system?
       * Is your system of event-response type, with no concern for real time, or is it a real-time system? If it is real-time, is it periodic, and what are the time constraints for each period?
       * *Concurrency*: Does your system use multiple threads?
       * If NO, skip to the next item;
       * If YES, identify the objects that have separate threads of control and describe any synchronization between the threads?
   19. Hardware Requirements
   20. What system does resources your system depend upon? Examples are, screen display, disk storage, communication network, or you may be accessing some special sensor/instrument.
   21. Describe exact requirements for these resources in order for your system to run. For example, you need color display, with minimum resolution of 640 × 480 pixels; minimum of 2 Gbytes hard disk space; minimum network bandwidth 56 Kbps.

* Project Management (described in Section 7 below) and References (described in Section 8 below)

**Part 3:**

1. Algorithms and Data Structures (if applicable)
   1. Algorithms
   2. Describe the algorithms that implement mathematical models from your Report #1. Does your system use any other complex algorithms? For example, when computing a motion trajectory for an animate figure in a game, you may use some numerical or computer-graphics algorithms. Or, when assessing stock market movements, you may be using statistical algorithms.
   3. If NO, skip to the next item;
   4. If YES, describe your algorithms. For example, for the animate figure example above, will the path coordinates be precomputed and stored in a look-up table or will they be computed using a spline interpolation algorithm.
   5. It is a good idea to use *activity diagrams* to describe the algorithm design.
   6. Data Structures
   7. Does your system use any complex data structures, such as arrays, linked lists, hash tables, or trees?
   8. If NO, skip to the next item;
   9. If YES, what criteria you used in deciding what data structure to use, e.g., performance vs. flexibility?
2. User Interface Design and Implementation
   1. Describe whether and how you modified and *implemented* the initial screen mock-ups developed for [Report #1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html). Comment only on significant changes in your user interface, those that reduce (or increase) the [user effort](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1-appA.html). Changes of colors or styles are less important and should be omitted from your report.
   2. The textbook does not deal much with the GUI design. Excellent guidelines for GUI design can be found here:
   3. Sun Microsystems, Inc. *Java Look and Feel Design Guidelines*. Mountain View, CA, 1999. Available at: <http://java.sun.com/products/jlf/ed2/book/>
   4. “Ease-of-use” is generally considered a key characteristic of user interface. “Ease-of-use” should not be confused with a flashy interface, with lots of colors, picture, graphics, etc. On the contrary, you should avoid flashy user interfaces. “Ease-of-use” means that interface is intuitive, easy to understand and operate, without having to ask many questions or read voluminous documentation. A minimal user interface that is well organized should be sufficient. You already considered the user effort as part of [Report #1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html), and here you should strive to minimize the user effort, thus maximizing the “ease-of-use”.
3. Design of Tests
4. Note that for this report you are just *designing* your tests; you will *program and run* those tests as part of work for your first demo, [see the list here](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/demo1.html#GRADING).
   1. List and describe the test cases that will be programmed and used for unit testing of your software.
   2. Discuss the test coverage of your tests.
   3. Describe your *Integration Testing* strategy and plans on how you will conduct it.
5. Describe also your plans for testing any algorithms, non-functional requirements, or user interface requirements that you might have stated in your Report #1.
6. Project Management and Plan of Work
   1. Merging the Contributions from Individual Team Members
   2. Compiling the final copy of the report from everyone’s work, ensuring consistency, uniform formatting and appearance.
   3. Describe what issues were encountered and how they were tackled.
   4. Project Coordination and Progress Report
   5. What use cases have been implemented?
   6. What is already functional, what is currently being tackled?
   7. List and describe other relevant [project management](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/projects.html#TEAMS) activities.
   8. Plan of Work
   9. List the projected milestones and dates by which you plan to accomplish them. Preferably, you should use [Gantt charts](http://www.ganttchart.com/) for planning and scheduling your project.
   10. Breakdown of Responsibilities
       * List the names of modules and classes that each team member is currently responsible for developing, coding, and testing
       * Who will coordinate the integration?
       * Who will perform and integration testing? (The assumption is that the unit testing will be done for each unit by the student who developed that unit.)
7. References
8. The list of references should contain exact *references and URLs* of any material that is used in the project and does not come from the textbook.

**NOTE:** No matter what programming language you use to implement your software (Java, PHP, C#, Flash or something else), *you must first provide* ***generic****, language-independent design diagrams (interaction diagrams) in UML*.

Second, if you are implementing your software in a non-object-oriented language, you should additionally provide *implementation-specific* sequence diagrams for your software. In this case, please explain, step-by-step, **how your implementation diagrams correspond to your generic UML design**. This means that the classes and methods from the generic design must be traceable to the functions of your non-object-oriented implementation. See Chapter 2 in the textbook and Miles & Hamilton: *Learning UML 2.0* for the how to create [interaction diagrams](http://en.wikipedia.org/wiki/Unified_Modeling_Language#Interaction_diagrams).

Source code should *not* be submitted with this report, but rather should be included with your [first demo materials](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/demo1.html#GRADING).

**NOTE:** *Comment your diagrams!* Describe all design decisions and other things that are not obvious from the diagrams. Any useful information is welcome. There is no limit on the number of pages for the report. Having good comments and explanations greatly helps in reading and evaluating the project and will positively contribute to your grade.

*Drawing Interaction Diagrams*. Theoretically, a single interaction diagram should be drawn for each system function (check your system sequence diagram - SSD). Because some system functions are trivial, I suggest the following practical technique. Start drawing one interaction diagram *per use case*. If, at some point, the diagram overflows to the next page, *split your interaction diagram* and draw one diagram *per system function*.

*Applying Design Principles*. It is very critical to pay attention to the proper use of the design principles described in Section 2.5 of [class lecture notes](http://www.ece.rutgers.edu/~marsic/books/SE/). (Quick refresher: design principles include: *Expert Doer*, *High Cohesion*, *Low Coupling*.) Proper *responsibility assignment*, separation of *application logic* from *presentation* and *storage* tiers is of the utmost importance and will be given a great weight in grading the reports. *It is important to document the alternative solutions that were considered in the design process, identify all the tradeoffs encountered and explain why the alternatives were abandoned.*

You should also check other design principles that were not covered in the class, such as [SOLID](http://en.wikipedia.org/wiki/SOLID_%28object-oriented_design%29) (also see [here](http://www.techrepublic.com/blog/programming-and-development/solid-design-principles-improve-object-oriented-development/5931)) and [GRASP](http://en.wikipedia.org/wiki/GRASP_%28object-oriented_design%29).

The tests in Section 6 are preliminary designs that may be changed and refined when you will do actual testing for your [first demo](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/demo1.html).

Note that you are *not* expected to write any program code for unit testing for this report. Test code writing and running the tests is expected for the first demo, [see item #2 in this list](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/demo1.html#GRADING).

For Report #2, you should just describe which classes/functions should be tested and how you would go about doing it. You need to derive the state diagrams and to draw the test case tables.

Also discuss the coverage of your unit tests. Recall that you are doing State-based testing, so your coverage should be analyzed in terms of states and transitions. A minimum acceptable strategy for responsible testing of a state diagram comprises testing these:

* Cover all identified states at least once (each state is part of at least one test case)
* Cover all valid transitions at least once
* Trigger all invalid transitions at least once

Design your tests for the *generic* UML design of your software.

If you will implement your software in a non-object-oriented language, do *not* report on test design for your implementation. Implementation tests will be reported as part of your [first demo contributions](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/demo1.html#GRADING).

Search for [testing PHP](http://www.google.com/search?hl=en&q=unit+testing+php) or [testing Flash](http://www.google.com/search?hl=en&q=unit+testing+flash), etc…

As you progress with presenting your UML diagrams, make it clear in the text what tools/platforms/languages you are using or will be using to implement your system. State where to find more information about the software tools/platforms/languages that you are using, and list the relevant links in the references (Section 8). Do not leave us wondering if even a serious detective work would be able to find out what exactly you used or are planning to use for your project.

### **1.1. Traceability of Classes to Domain Concepts**

The greatest problem is that Report #2 usually has is that it may appear to be created almost independently of your Report #1, without any reference to your Report #1. A key of good software engineering is to maintain logical progression among the artifacts (i.e., *traceability*): from requirements, through conceptual modeling, class deign, to code implementation.

This most often reflects in how your interaction diagrams evolved from your Report #1, i.e., from your system sequence diagrams and your domain model. Often, there is no clear connection! It is impossible to trace how classes/objects in your Report #2 evolved from the domain model in your Report #1. The diagrams in your Report #2 may seem to have been invented independently of all analysis that you did for report #1!! That renders your report #1 useless and that is a bad software engineering.

To do it properly, your Report #2 must do the following.

1. For each interaction diagram, it must be explicitly stated from which system sequence diagram it evolved and which system method it details.
2. In each interaction diagram, at least some of the concepts from your domain model must be used. Even if, while doing Report #2 (after Report #1 was submitted), you discovered that most of those concepts need to be modified or new concepts introduced, your Report #2 must explicitly state which new objects (in your interaction diagrams) evolved from the old concepts (shown in the domain model of Report #1, which you may revise and enclose when submitting Report #2). If you needed to introduce new objects (and there is nothing in your domain model in Report #1 that corresponds to these new objects), that is fine, but you must explain why these new objects are needed and explain in at least one sentence why they were not discovered during the domain analysis phase in Report #1.

It is essential that it is possible to trace the classes/objects in your Report #2 back to domain concepts from Report #1. You must make this tracing explicit, instead of expecting the grader to labor through your reports and try to figure out which interaction diagram originated from which system sequence diagram, and which class originated from abstract concept in your domain model. The progression must be evolutionary, rather than sudden leaps where your new design has no clear connection to the analysis performed in report #1. Again, a lack of continuity from Report #1 to Report #2 renders Report #1 useless and means that doing it was simply waste of time.

A common excuse heard from students is that their software ended up being implemented using PHP, Flash, or some other non-object-oriented language, so there is no clear correspondence between classes and PHP scripts. This is not a valid point. Your implementation must be based on your conceptual analysis (otherwise your conceptual analysis is useless). Your implementation is just one way to implement your conceptual model, using the programming language of your choice. If you did good job in your conceptual analysis, then it should be possible for another developer to take your conceptual model and implement it in a different programming language. Therefore, you must be able to trace the origin of your PHP scripts (or whatever other program units) back to the concepts from your conceptual model. You must make this tracing explicit, instead of expecting the instructor to labor through your reports and try to figure out which PHP script originated from which abstract concept in your domain model.

### **1.2. Changes to Report #1**

As you gain better understanding of your problem or invent a better solution, you should make all necessary changes to your documentation. That is the point of [iterative approach](http://en.wikipedia.org/wiki/Iterative_and_incremental_development): we develop a system through repeated cycles (iterations) and take advantage of what was learned during development of earlier parts or versions of the system. Therefore, you are encouraged to make revisions and improvements to your past documentation ([Report 1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html)) as you learn more.

*Updating the Materials From Report* #1. Use case diagrams, domain model, and even the requirements from ([Report #1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html)) may need to be updated as you progress with your project development. This is normal on software projects and it is important that you update your first report to accurately reflect your current project status. If you made significant changes and it may be difficult to understand your Report #2 without seeing your revised first report, then you should provide us with your revised report, as well. It would be helpful if you provide a brief summary of changes somewhere in your updated report version, so we can better understand the evolution of your project.

We will **not** grade your revised report #1 nor provide feedback on it, but we will consult it when grading Report #2. We will grade your revised Report #1 when it will be submitted as part of [Report #3](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report3.html).

## **2. Report Preparation**

The guidelines remain the same as for [Report #1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html#PREP).

Here are some options for [software tools for UML diagramming](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/projects.html#UML-TOOLS). You may search the Web and find some other tool as well. Any tool that supports the UML symbols is acceptable.

## **3. Report Grading**

See also the [grading policy](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/projects.html#GRADING) for the assigning the overall team grade vs. grades for the individual members.

This report will be graded only after the entire report is submitted.

Individual parts of the report will *not* be graded immediately after the submission.

However, **50% of points** will be deducted from the parts that were not submitted on time.

If time permits, we *may* provide feedback on the individual parts before the entire report is due.

First, all reports will be graded ***independently*** of each other, as follows:

(NOTE: Check also the [**actual grading checklist**](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report2-grading.html).)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Aspect | Interaction Diagrams | | | Classes + Specs | | Sys Arch & Design | | | | | Alg's  & data  struct | User Interface | | Testing  design | Project management | | | Ref's |
| UML  diagrams | Prose  descr.  of diag | Alt.sol'n  descr.  (#) | Class  diag &  descr. | Signa  tures | Sty  les | Pckg  diag | Map  hard  ware | Data  base | other  ($) | App  ear  ance | Prose  descr.  (+) | Doc.  merge | Proj  coord/  progress | Plan  of  wrk |
| Points | 10 | 10 | 10 | 5 | 5 | 5 | 2 | 2 | 3 | \* 3 | \* 4 | 6 | 5 | 12 | 11 | 5 | 2 | (-5) |
| Total  points | 100 | | | | | | | | | | | | | | | | | |
| \* Point numbers preceded with an asterisk will be graded only if applicable. Otherwise, these  points will be assigned to Section 5 of this report: *User Interface Design*.  (#) This refers to the narrative describing the *alternative design choices* that you considered and their interaction diagrams.  State explicitly what design choices were made, what design principles were applied, and provide the rationale for your choices.  ($) Other design aspects, if applicable, given in the list above, such as network protocol, global  control flow, and hardware requirements.  (+) This refers to the narrative describing your user interface design.  State explicitly what design choices were made and provide the rationale for your choices.  Up to 5 points will be deducted for missing or incomplete references | | | | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Second, all reports will be ***cross-compared*** and rank-ordered. See [Report #1](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html#CROSS_COMPARE) for explanation of this second step.

## **4. Report Submission**

You may include your revised Report #1 with your Report #2, if you feel that you made significant changes and having your updated Report #1 can help us better understand your Report #2.

However, *we will* ***not*** *grade the updated Report #1 and will* ***not*** *provide any feedback on the revisions*. We will grade revised Report #1 when it will be submitted as part of [Report #3](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report3.html).

When submitting **Part 1**, the document must include the following sections:

* Cover Page and Individual Contributions Breakdown
* Table of Contents
* Section 1: *Interaction Diagrams*
* Project Management
* References

When submitting **Part 2**, the document must include the following sections:

* Cover Page and Individual Contributions Breakdown
* Table of Contents
* Section 1: *Interaction Diagrams* — revised as needed
* Section 2: *Class Diagram*
* Section 3: *System Architecture*
* Project Management
* References

The **full report** must contain **all** sections specified in the [Report Format](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report2.html#FORMAT) (previously submitted parts of this report may be revised as needed).

Each team should email their report (**PDF document only!!!**) both to the instructor *and* the [TA](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/index.html#TA) on or before the due date.

The report should also be posted for download on your project website (for free web hosting, see [here](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/projects.html#RESOURCES)).

Submission deadline: **5:00 p.m. on the** [**due date**](http://www.ece.rutgers.edu/~marsic/Teaching/SE1/syllabus.html#DELIVERABLES).