45 points total.

1. (5 points) Name and briefly describe the five perspectives on quality according to Garvin.

- **Transcendental view**: where quality is something we can recognize but not define
- **User view**: where quality is fitness for purpose
- **Manufacturing view**: where quality is conformance to specification
- **Product view**: where quality is tied to inherent products characteristics
- **Value based view**: where quality depends on the amount the customer is willing to pay for it

2. (15 points) Consider an automated library circulation system. Every book has a bar code, and every borrower has a card bearing a bar code. When a borrower wishes to check out a book, the librarian scans the bar codes on the book and the borrower’s card, and enters C at the computer terminal. Similarly, when a book is returned, it is again scanned and the librarian enters R. Librarians can add books (+) to the library collection by a particular author (the librarian enters A= followed by the author’s name), all the books with a specific title (T= followed by the title), or all the books in a particular subject area (S= followed by the subject area). Finally, if a borrower wants a book currently checked out, the librarian can place a hold on the books so that, when it is returned, it will be held for the borrower who requested it (H= followed by the number of the book). Write down a precise specification of the library circulation system.

The sample solution will be expressed with Use cases. However, this format was not required. What was important is that you specify the same thing but using any valid technique

**Purpose**: The library circulation system will automate the process of book check outs, returns and additions in an efficient way

**Scope**: The library circulation system will manage the addition of books to the library collection as well as the check in/out process for them. Also, it will have the ability to put a book on hold when a borrower needs a currently checked out book. The system will interface with the borrower management system (the system in charge on adding, modifying, etc. the borrowers’ information) for accessing data managed by it.

**Use Case UC1**: Check out a book(s)

- **Primary actor**: Librarian
- **Supporting actor**: Barcode reader
- **Preconditions**: Librarian is authenticated with the system
- **Brief description**: … (this can be filled almost directly from the question stem)

**Basic Flow**

1. Borrower arrives to the library checkout with book(s) to borrow
2. Librarian scans book’s barcode
3. System stores the code and presents the title, author and other information
Librarian repeats steps 2 and 3 for every book that the borrower wishes to check out
4. The librarian scans the borrower’s card
5. System displays the borrower’s information with a list of the books that the he or she is about to checkout
6. Librarian presses “C”
7. Systems records the transaction registering each book as checked out by this specific borrower
**System repeats step 7 for every book**
8. System presents receipt
9. Borrower leaves with receipt and books (if any)

**Alternative flows**
2. Invalid identifier
   1. System signals error and rejects entry.
4. System detects that the user is late in his monthly payments … (this is optional, but you get the idea)
7a. System detects that the Book is on hold by this borrower
   1. Remove the hold
   2. Proceed with checkout
7b. System detects that the Book is on hold by another borrower
   1. System signals an error for this specific book

**Use Case UC2: Return a book(s)**
**Primary actor:** Librarian
**Supporting actor:** Barcode reader
**Preconditions:** Librarian is authenticated with the system
**Basic Flow**
1. Borrower arrives to the librarian’s desk with book(s) to return
2. Librarian scans book’s barcode
3. System stores the code and presents the title, author and other information
**Librarian repeats steps 2 and 3 for every book that the borrower wishes to return**
4. Librarian presses “R”
5. Systems records the transaction and records that book is no longer checked out
**System repeats step 5 for every book**
6. Borrower leaves without any book

**Alternative flows**
2. Invalid identifier
   1. System signals error and rejects entry.
7a. System detects that another borrower has put this book on hold
   1. Notify the borrower (or next borrower if you wish to maintain a queue) who put a hold on this book that the book is no longer checked out
   (optional)
7b. System detects that the due date of the book is past
   1. System notifies the librarian
   2. System computes any fines (again, this is optional)
   3. …
Use Case UC3: Put a book on hold
   Primary actor: Librarian
   Secondary Actor: barcode reader
   Preconditions: Librarian is authenticated with the system
   Basic Flow
   1. Borrower arrives to the librarian’s desk with the number of the book that he
      wishes to check out once it’s available (note: you can change this UC so that
      it can hold several books)
   2. Librarian inputs “H=” followed by the book’s code
   3. System presents the book information
   4. Librarian scans the borrower’s card
   5. System places a hold on this book for this particular borrower
   6. Borrower leaves with his card

   Alternative flows
   2a. Book number is not recognized by the system
      1. System signals error and rejects entry.
   2b. System detects that this book is already on hold (if you maintain a queue, this
      can be handled differently)
      1. Notify the librarian
      2. Cancels transaction

Use Case UC4: Add books to the library collection
   Primary actor: Librarian
   Preconditions: Librarian is authenticated with the system
   Basic Flow
   1. This UC begins when the Librarian has a list of books to add to the library
      collection
   2. The Librarian presses “+”
   3. Systems enters into “addition mode”
   4. Librarian does one of the following1
      - Add Books by Author
      - Add Books by Title
      - Add Books by Subject Area2
   5. Systems ends “addition mode”

Use Case UC5: Add Books by Author
   Primary actor: Librarian
   Preconditions: Librarian is authenticated with the system
   Librarian have made the system to enter into insertion mode
   Level: Sub function

---

1 Although here we are handling this UC with the <<includes>> relationship, this kind of UC (“the user can
do one of several kinds of things here”) is a candidate for the generalization relationship (except if we
don’t know the number of choices in advance. In that situation, generalization is still possible, but
<<extends>> is most frequently used for conditional invocation). This is more plausible when we are
diagramming UCs and the corresponding interaction diagrams. However, for the UC textual
documentation, the generalization relationship is seldom used in practice
2 This practice of underlining the name of a UC that we are going to <<includes>> is from the book
“Writing Effective Uses Cases” by Alistair Cockburn. It’s not a rule of the Rational Unified Process (RUP)
Basic Flow
1. Librarian enters “A=” followed by the author whose books the librarian wants to add
2. System prompts the user for the number of books that will be entered
3. Librarian inputs the number of books
4. System asks for the remaining information of the book (it already has the author)
5. Librarian inputs the book code/number, its title, its subject area, etc.
6. System records the information of the newly added book
Steps 4-6 are repeated until the number of books specified by the librarian have been added

Alternative flows
1. System could not locate the Author in the system
   1. System signals error
   2. System asks the librarian for a new Author name
5. Book code is already used in the system
   1. System signals error and rejects entry.
2b. System detects that this book is already on hold (if you maintain a queue, this can be handled differently)
   1. System notifies the librarian
   2. System cancels transaction

Technology and Data Variations List
5. Code is already attached to the book and can be entered using the barcode reader

Note: The other “Add book by …” UCs are handled in similar way

Data Specification

<table>
<thead>
<tr>
<th>Borrowers</th>
<th>Books</th>
<th>CheckedOutBooks</th>
<th>BooksOnHold</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrwrCode (or ID)</td>
<td>BookCode (number)</td>
<td>BrwrCode</td>
<td>BrwrCode</td>
</tr>
<tr>
<td>Name</td>
<td>Title</td>
<td>BookCode</td>
<td>BookCode</td>
</tr>
<tr>
<td>e-mail (optional)</td>
<td>Author</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SubjectArea</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Others (optional)</td>
<td>Subject Areas</td>
<td>Librarians</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Note: Relationships and Cardinality are beyond the scope of the course

It’s a bad idea to store a flag for this in the book record if you are using a relational database, but if you chose to use an object oriented modeling of the entities it could be Ok (depending on your decisions of course). This is just a sample solution.
3. (7 points) Explain the difference between errors, faults, and failures. Give an example of an error that leads to a fault in the requirements; the design; the code. Give an example of a fault in the requirements that leads to a failure; a fault in the design that leads to a failure; a fault in the test data that leads to a failure.

Error is a mistake made by a human being in performing some software activity and the fault is an encoding of the error. A fault can reside in any development or maintenance product. A failure is a departure from the system’s required behavior. The system is not performing as required even if it’s performing as specified, because the specification contains faults.

Error that leads to a fault in the requirements. For example, during Surpass development, one person reads that he must “Detect cycles in formula definitions”. He misunderstands this (the error) and thinks that a cycle is only when a formula in one cell refers to the same cell and doesn’t recognize that a cycle must also be present when a formula in a cell (e.g. A1) can refer to another cell (e.g. C16), and this referred cell (C16) can in turn contain a formula that refers to the first cell (A1). He states this bad interpretation in the requirements specification document (the fault)

Error that leads to a fault in the design. When designing a web based forum for the intranet of a company of more than 100 users, the designer specifies SQL Server as the database of choice. However failing to understand its limits (the error), he specifies just the MSDE version (Microsoft SQL Server Desktop Engine). That version cannot handle a very high number of concurrent users (according to Microsoft, just 5 at full performance), but it comes free with Visual Studio .NET (the tool of choice), so the designer thinks that is a good decision. At the end, the design contains this fault.

An error that leads to a fault in the code: Coding a function that outputs one type (inches) and having another function take that output as input but assumes the input was of another type (centimeters)

A fault in the requirements that leads to a failure. For example, a client want us to develop a spreadsheet program, One of his requirements is that the user should be able to enter text in a cell that can be separated by a new line character, for example:

“This is, possible,
in a cell, using Excel,
Use Alt+Enter after checking Wrap text, in Format\Cells\Alignment”
(note the use of commas)
Also the client wants that the file can be exported in “comma separated format”. Both requirements are implemented “as is”, and when a sheet with a cell that contains multiple commas and new line characters is exported, the resulting file will be inconsistent (e.g. a cell with too many commas and new lines, can appear as one or more entire row(s)). The fault here is having the two requirements that may conflict between them

A fault in the design that leads to a failure. In the previous case about the web based forum, the site performs very well the first weeks when not too many employees use it. However, when the site become popular, it became impossible for MSDE (the fault was to specify this version in the design) to handle all the traffic and the site begin to refuse visitors (the failure)

A fault in the test data that leads to a failure: testing an antivirus program with low-risk viruses or with viruses of just a restricted number of types
4. (2 points) What do you think happens when requirements validation uncovers an error? Who is involved in correcting it?

According to the book you must check traceability between the definition and the specification. If there is an error there (inconsistency, lack of traceability) then it's obviously an error made by the analyst(s) and they have to correct this.

However, requirements validation is more than a simple check of traceability. In order to check if the user's goals are met, a review, that involves not only analysts but also users, is conducted. If an error is uncovered, it's the responsibility of both sides to come up with a solution.

5. (1 point) A manufacturer of a military aircraft lists $2,000,000 as the risk exposure for a jet at a loss probability of 1%. What is the plane's price?

\[
\text{Price} = \frac{2,000,000}{0.01} = 200,000,000
\]

6. (5 points) Name five of the top ten risk items put forth by Boehm and what he suggested to address them.

- Personnel shortfall. Staffing with top talent; job matching; team-building; morale-building; cross-training; pre-scheduling key people.
- Unrealistic schedules and budgets. Detailed, multi-source cost and schedule estimation; design to cost; incremental development; software reuse; requirements scrubbing.
- Developing the wrong functions and properties. Organizational analysis; mission analysis; operational concept formulation; user surveys; prototyping; early users' manuals.
- Developing the wrong user interface. Prototyping; scenarios; task analysis.
- Gold-plating ("I don't know if I'll need this feature or not, but I might as well specify it just in case.") Requirements scrubbing; prototyping; cost-benefit analysis; design to cost.
- Continuing stream of requirements changes. High change threshold; information-hiding; incremental development (defer changes to later increments).
- Shortfalls in externally furnished components. Reference-checking; pre-award audits; award-fee contracts; competitive design or prototyping; team-building.
- Shortfalls in externally performed tasks. Benchmarking; inspections; reference checking; compatibility analysis.
- Real-time performance shortfalls. Simulation; benchmarking; modeling; prototyping; instrumentation; tuning.
- Straining computer-science capabilities. Technical analysis; cost-benefit analysis; prototyping; reference checking.

7. (5 points) For each type of coupling, give an example of two components coupled in that way.

- Content coupling: When a) One component modifies an internal data item in another component or b) when one component branches into the middle of another component. For example, you have a class Vector3D with 3 fields: x, y, z and several methods that manipulate these fields. Later, you declare a class
called Arch which declares a field “baseline” of type Vector3D. One of the methods of Arch called “slant” assigns values directly to the fields x and y of baseline without using methods of Vector3D

- Common coupling: when two modules share the same global data (e.g. a global variable). For example, a company that develops the programs WordPlus and CalcSheet has some data stored in the registry that the two programs read. One of the settings stored is the font used by default. An error made by the programmers allows this setting being modified by any of two without notifying the other. The user starts WordPlus and CalcSheet and changes the default font in the latter from Arial to Helvetica. Then he closes both programs, and other day he starts WordPlus and open one of his documents and see that, since he used the default font, now the document looks different.

- Control coupling: one component/module controls the logic of another, by passing it information on what to do (e.g. passing a what-to-do flag). For example, in a university system, one routine is called InputAll and is in charge of input student names, professor names and course names, depending on a flag passed to it, instead of having three different routines each one performing a specific task

- Stamp coupling (Data-structured coupling) - Stamp coupling is when modules share a composite data structure, each module not knowing which part of the data structure will be used by the other (e.g. passing a student record to a function which calculates the student's GPA).

- Data coupling - Data coupling is when modules share data through, for example, parameters. Each datum is an elementary piece, and these are the only data which are shared (e.g. passing an integer to a function which computes a square root).

8. (5 points) Using design by contract, write a design for a component that implements a hashtable. Clearly specify the obligations (pre conditions) and consistency constraints (invariants).

**Operations**

**Create (int initialCapacity, float loadFactor)**

Pre conditions
Initial capacity is no less than zero and load factor is positive
Post conditions
An empty hashtable is created with the specified initial capacity and the specified load factor

**put (X :Element, Key: String)**
preconditions
Number of elements <= than capacity
Neither X nor Key is null
Postconditions
containsKey(Key)
get(Key) = X
The number of elements is incremented by one
containsKey(\textbf{Key: String})
preconditions
Key is not null

postconditions
True has been returned if there is an entry in the hashtable associated with the key
False has been returned otherwise

get (\textbf{Key: String})
preconditions
Key is not null
Key exists in the table and is associated with a value

postconditions
The value associated with the key has been returned to the caller
Number of elements in the table remain the same

remove(\textbf{Key: String})
preconditions
Key is not null
Key exists in the table and is associated with a value

postconditions
The number of elements has been decremented in one
not containsKey(\textbf{Key})

\textbf{Class Invariant(s)}
Capacity is not incremented unless the number of entries in the hashtable exceeds the
product of the load factor and the current capacity