Part IV
Wrap-Up

This fourth part introduces the notion of a software design document and presents some ideas for what you should learn next.

Software Design Document

34

The objective of this chapter is to present a template for a software design document that may be used to consolidate design knowledge into a single document.

34.1 Preconditions

The following should be true prior to starting this chapter.

- You understand five program design criteria: separation of concerns, design for reuse, design only what is needed, performance, and security.
- You understand the six characteristics of a good software design: simplicity, coupling, cohesion, information hiding, performance, and security.
- You understand the notion of abstraction as a design process. Abstraction is the ability to generalize a concept by removing details that are not needed to properly convey the design perspective being emphasized.
- You have created and/or modified models that described an object-oriented or structured software design.
- You understand how to apply the Model–View–Controller architectural pattern to modify an existing design into these three components.
- You understand how to apply the Model–View–Controller architectural pattern to create a high-level design that accurately reflects the domain requirements while satisfying the design constraints inherent in MVC.

D. P. Voorhees, *Guide to Efficient Software Design*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-28501-2_34

34.2 Concepts and Context

Documents are produced in software development projects to encapsulate the relevant information into a well-structured and thorough presentation. As described in this book, a software design contains knowledge represented using many types of models and textual descriptions. Including all of this into a single document allows for an easy way to share your design information. The question addressed by this chapter is how should we organize our design knowledge?

34.2.1 What Would a Software Design Artifact Describe?

Pressman describes four types of design information [1].

Architecture: A high-level description of the observable elements making up the system.
Data: Includes logical data models, physical data models, and a data dictionary.
Interfaces: Describes human–computer interactions, external interfaces, and internal interfaces between components.
Components: Includes design descriptions of each component making up the system. Includes both observable components (shown in architecture) and non-observable components.

34.2.2 What Would a Software Design Artifact Look Like?

The structure of a design document is based on these four types of design information. Table 34.1 shows an outline for a software design document.

Each section of the design artifact is briefly described below.

34.2.2.1 Introduction

This section describes the purpose and scope of the design document.

34.2.2.2 Architecture

Create and insert one or more diagrams to show the high-level design of the application. The types of diagrams you use to show your application architecture can include one or more of the following. Please note these diagram types are not listed in order of preference.

- A data-flow diagram (DFD) to show the major components of the design and how these components are related to each other via data flows and data stores.
- An IDEF0 function model to show the major components of the design and how these components are related to each other via input, control, output, and mechanism data flows.

Table 34.1 Design document Structure

1. Introduction
Describe the purpose and scope of the design document.
2. Architecture
Describe the high-level design of the application by inserting one or more software architecture diagrams.
3. Data
This section contains a description of the logical and physical view of the data used by the application.
3.1 Logical Data Model
Insert a logical data model here, and then list the data rules based on the cardinalities shown in the logical data model.
3.2 Physical Data Model
Insert a description of the persistent data store. This may include a physical data model (e.g., relational database design or hierarchy of XML tags).
3.3 Internal Data Structures
Describe the memory-based data structures constructed and used during execution of the application.
3.4 Data Dictionary
Include a table to describe the characteristics of each attribute shown in the data models.
4. Interfaces
This section contains a description of interfaces that must be supported by the application.
4.1 Human-computer Interaction
Describes the user interactions of the application/system.
4.2 Communication with External Entities
Describes the ways in which the application will communicate with different types of external entities.
4.3 Communication between Internal Components
Describes the ways in which the components of this application/system will communicate with each other.
5. Components
This section contains a more detailed description of each component described in the Architecture section.
5.1 Component X
Include text and design models that describe the component. Each component section should describe the behavior and structure.

- For an object-oriented design, a UML package and/or class diagram to show the major components of the design and how the classes in each package are related to classes in other packages.

34.2.2.3 Data

The Data section will contain descriptions for logical data models, physical data models, internal data structures, and a data dictionary.

In section *3.1 Logical Data Model*, create and insert a logical data model (LDM) for the application. This should show the strong entities, weak entities, and the relationships between each data entity. The model should show the attributes associated with each data entity and which attribute(s) represent the primary key.

After the LDM, list the rules shown in the logical data model. Each rule is a single sentence describing a relationship between two logical entities. A few example rules are listed below, based on an LDM that describes contact information. (The text in parentheses is an explanation of what the LDM would look like in order to generate this rule. This text is part of these instructions and not intended to be included in your design artifact.)

- A contact has a name that is not required to be unique. (The LDM shows a contact entity with an id attribute as the primary key and a name as a non-key attribute.)

Table 34.2 Data dictionary information—text file

Attribute	Type	Size	Format/range	Structure

- A contact has zero or more phone numbers. (The LDM shows a relationship between a contact entity and a phone number entity. The cardinality shows that a phone number instance is not required for each contact instance.)
- A contact helps to identify their phone numbers. (The LDM shows a relationship between a strong contact entity and a weak phone number entity.)

In section *3.2 Physical Data Model*, when a persistent data store is being used by the application, create and insert a physical data model to show the physical representation of the data as it will exist in the persistent data store. When a persistent data store is not being used, this section may be removed from your design artifact.

The type of physical data model used is based on the type of persistent data storage to be used by the application. The following is not intended to be an exhaustive list of persistent data storage types.

- For a text file, this section needs to describe the contents and format of the lines of text. When there are different types of text lines in the text file, each type of text line needs to be described.
- For a text-based XML file, this section needs to describe the hierarchy of XML tags that describe the structure of the data.
- For a relational database, this section needs to describe the tables, primary keys, foreign keys, non-key fields, and indexes based on the logical data model.

In section *3.3 Internal Data Structures*, when non-persistent data structures (e.g., linked list, array, tree, graph, document object model) will be used by the application, provide a description for each data structure. Otherwise, this section is not applicable and may be removed from your design artifact.

In section *3.4 Data Dictionary*, list the attributes shown in the logical data model and describe their physical characteristics. Use one of the following tables based on the type of physical data store being used. Tables 34.2, 34.3, and 34.4 show three different formats for a data dictionary, each containing the following columns.

Attribute: the name of the attribute on the logical data model.
Type: the type of data stored in the attribute.
Size: the size of the attribute, based on its Type.
Format/Range: any special format rules or range of values the attribute must adhere to.

Table 34.2 shows what a data dictionary might look like for a text file. The *structure* column identifies the internal data structure(s) used to store values associated with this attribute.

Table 34.3 Data dictionary information—XML file

Attribute	Type	Size	Format/range	XML tag

Table 34.4 Data dictionary information—relational database

Attribute	Type	Size	Format/range	Table

Table 34.3 shows what a data dictionary might look like for an XML file. The *XML Tag* column identifies the XML structure(s) used to store values associated with this attribute.

Table 34.4 shows what a data dictionary might look like for a relational database. The *Table* column identifies the table(s) containing the attribute.

34.2.2.4 Interfaces

This section contains a description for three types of interfaces: the human–computer interaction; the ways the application will communicate with external entities as described in the Architecture section; and how components of this application communicate with each other.

When one of these interface types does not exist for your application you may remove that section from your design artifact.

In section *4.1 Human–computer Interaction*, when the application has an interface with a human user, create and insert a state machine diagram (aka statechart) to explain how the user would navigate through the application. When the HCI design is large or complex, you may want to have more than one statechart or you may want to include descriptive text to help explain the interactions between the application and a user. In addition, a large or complex HCI should also be described using pictures to illustrate its appearance. When doing this, adding text to connect the state machine diagram to the picture(s) would help in understanding the HCI design.

In section *4.2 Communication with External Entities*, when the application must communicate with one or more external entities (as shown in the Architecture section), create and insert a structure or behavior model to show the way(s) in which the application will communicate with each type of external entity. These diagrams would show the application programming interface (API) to be used between your application code and the external entity.

In section *4.3 Communication between Components*, when your architecture/design contains two or more components, this section should describe how these components communicate with each other. This description would likely include both text and diagrams, using one or more behavior and structure diagrams.

34.2.2.5 Components

This section contains a more detailed description of each component described in the Architecture section. Create a separate [Component X] section for each component and rename each section to one of the component names.

In section *5.1 [Component X]*, each component in your architecture/design will have at least two diagrams—one that shows the structure and another that shows the behavior of the component.

34.3 Post-conditions

The following should have been learned when completing this chapter.

- You understand one approach for consolidating design knowledge into a single document/artifact. This approach describes a design at a high-level of abstraction via the Architecture section; data models representing both memory-based and persistent data storage; interface descriptions for the user interface, to external entities, and between components within the design; and details on each component described in the Architecture section.

Exercises

Hands-on Exercises

1. Using an existing design solution, create a design document containing all of the design knowledge you've developed for your solution.
2. Use your development of an application you started in Chaps. 3 or 4 for this exercise. Create a design document containing all of the design knowledge you've developed for this application.
3. Continue hands-on exercise 3 from Chaps. 12 or 13 by developing a design document containing all of the design knowledge you've developed for this problem domain. The list below serves as a reminder of the application domain you may have chosen for this exercise. Refer to the Hands-on Exercises in Chaps. 12 or 13 for details on each domain.

    - Airline reservation and seat assignment
    - Automated teller machine (ATM)
    - Bus transportation system
    - Course-class enrollment
    - Digital library
    - Inventory and distribution control
    - Online retail shopping cart
    - Personal calendar
    - Travel itinerary

Reference

1. Pressman RS (2005) Software engineering: a practitioner's approach, 6th edn. McGraw-Hill, New York