Unit - III

Technical Design

An **e-Commerce design** can be at the level of a system or component, and generally includes:

- Relevant goals or requirements (functional and non-functional);
- Static structure (e.g., components, interfaces, dependencies);
- Dynamic behavior (how components interacts);
- Data models or external interfaces (external to the system/component described in the document); and
- Deployment considerations (e.g., runtime requirements, third-party components).

Note that all of these descriptions are at an abstract level. The purpose is to give the reader a broad general understanding of the system or component. There may be many levels of design documents (e.g., system- or component-level).

A **technical Design specification** describes the minute detail of either all or specific parts of a design, such as:

- The signature of an interface, including all data types/structures required (input data types, output data types, exceptions);
- Detailed class models including all methods, attributes, dependencies and associations;
- The specific algorithms that a component employs and how they work; and
- Physical data models including attributes and types of each entity/data type.

Technical Design

Technical specifications, at least in the form of a technical design, are *part* of the design documents, along with, for example, requirements lists, functional designs, user stories, graphics design mockups, usability studies, UML diagrams, business process diagrams, data model specifications, etc.

Technical specifications of the type that you write after the fact, to document the finished product, are *not* generally part of the design documents, but they can be included in the set of design documents of a later version (for reference) or another product that relies on them.

Objectives and Scope of the Technical Design

Technical Design has the main **objective** of converting the Functional Design into a model of the system that can be built by the developers.

(i). The Technical Specifications define in very clear terms the system with the developers in mind.
(ii). The Technical Design shows “How” the new system will be built.
(iii). The design view is a detailed view.
(iv). The Technical Design process is iterative. It may result in a revision of the requirements.

The overall Design Process is a composite activity relying on a variety of inputs as shown below:

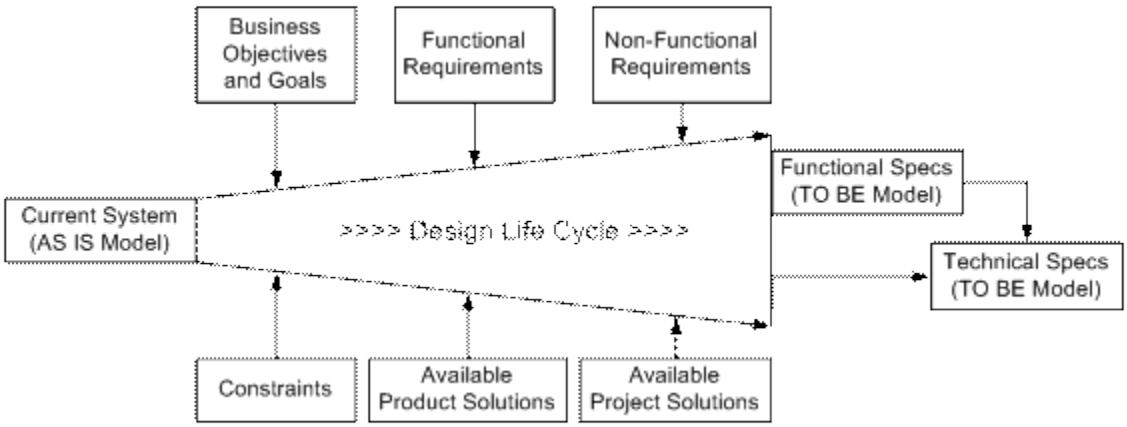


Figure 1: Converting Functional to Technical Specifications

It is very important to differentiate, as mentioned before, between Functional Design and Technical Design. The Functional Design discussed earlier is used to qualify that the new system is what is required by the stakeholders and that it will work they way they wish it to. The Technical Design is the upgraded Functional Design that is submitted to the developers to build the product.

The **scope** of this process covers software development projects implemented by the Ministry or Agency. Since this is a technical design, it would only apply to software that is developed internally.

For externally developed software, it remains the responsibility of the vendor to develop the Technical Specifications using the Functional Specifications submitted by the Ministry or the Agency as part of the bidding documents.

Construction of Technical design for e-commerce projects

The process for completing the design and construction of a building is often divided into notional ‘stages’. This can be helpful in establishing milestones for the submission of progress reports, the preparation of information for approval, client gateways, and for making payments.

However, there is a great deal of ambiguity between the naming of stages and the definition of what individual stages include and so it is important that appointment documents make it clear precisely what activities fall within which stage, and what level of detail is required.

Generally the phrase ‘technical design’ refers to project activities that take place after the detailed design (or 'developed design' or 'definition') has been completed, but before the construction contract is tendered or construction begins.

Increasingly, however, technical design may continue through the preparation of production information and tender documentation and even during construction itself, particularly where aspects of the technical design are undertaken by specialist subcontractors.

The lead designer co-ordinates the preparation of the technical design. As this may involve design not only by the client's core design team but also by specialist subcontractors, it may be appropriate to organize a specialist contractors' start-up meeting at the beginning of the stage. A design responsibility matrix can help allocate design tasks between the project team members, and on complex projects, it may be necessary to appoint a design coordinator responsible for co-ordination and integration of different aspects of the technical design.

There is some skill in establishing the order for undertaking technical design. For instance the ceiling tile grid has to be established so that light fittings, sprinkler heads and smoke detectors can be located Centre of tiles and access provisions to services in ceiling voids can be established. Similarly, mullion positions for cladding systems dictate partition locations between cellular offices. Drainage set to falls has priority over ceiling pipe work, ductwork and electrical trunking the latter being more flexible in its routing. It is argued by some that co-ordination between the different aspects of this technical design is best carried out by the client's design team despite the increasing tendency to transfer responsibility to contractors.

By the end of the stage the architectural, structural and mechanical services design and specifications should describe all the main components of the building and how they fit together, any performance specified work should be defined and there should be sufficient information for applications for statutory approval to be completed. Room data sheets are also likely to have been prepared along with outline technical specifications.

Regular reviews should be carried out during the stage to assess construction sequencing, build ability and the interfaces between different elements of the design, the project programme and risk. The client's design team may be required to review design information prepared by specialists to ensure proper integration into the wider design.

It may be appropriate to arrange visits to the specialist contractors' premises to assess samples or mock-ups and to witness tests. Some samples may require approval by the client.

Once the client is satisfied with the technical design, the lead consultant should freeze the design and specifications and introduce change control procedures and remaining statutory approvals and other approvals should be completed.

DETAILED DESIGN

In e-Commerce, Detailed design follows a process which entails conceptual design; embodiment design and detail design and, when performed professionally, eventually results in a well-designed solution.

Conceptual design is Phase One of detailed design in which drawings are the main output. The drawings produced are often quite simple ideas with little detail, but the aim of the conceptual phase is to commit ideas to paper.

The Embodiment phase of the detailed design process starts with the concept and develops it into a workable system that can be further developed. During this phase, Developers will typically follow a framework of clarity, simplicity and safety in achieving the design goal.

Detailed design is the phase where the design is refined and plans, specifications and estimates are created. Detailed design will include outputs such as 2D and 3D models, P & ID's, cost build up estimates, procurement plans etc. This phase is where the full cost of the project is identified.

Detailed design of the system is the last design activity before implementation begins. The hardest design problems must be addressed by the detailed design or the design is not complete. The detailed design is still an abstraction as compared to source code, but should be detailed enough to ensure that translation to source is a precise mapping instead of a rough interpretation.

Note: As mentioned on the Interface Design page, many of the examples will be scoped to a particular part of the overall system under development. Detailed design artifacts are going to contain a large amount of details which, if included in full, would obscure the point of this page.

The detailed design should represent the system design in a variety of views where each view uses a different modeling technique. By using a variety of views, different parts of the system can be made clearer by different views. Some views are better at elaborating a systems states whereas other views are better at showing how data flows within the system. Other views are better at showing how different system entities relate to each through class taxonomies for systems that are designed using an object-oriented approach.

A template for detailed design would not be of much use since each detailed design is likely to be unique and quite different from other designs. What is helpful in regards to guidance on detailed design are examples. This page provides some detailed design content for the example microwave oven system that is referenced periodically by this website.

The major sections of this web page are the following:

1. Structural Chart
2. Control-Flow Model
3. Class Diagram
4. Collaboration Diagram
5. Sequence Diagram

6. Activity Diagram
7. Pseudocode
8. V-Model Diagram

1. Structural Chart

The intent of a structural chart is to specify the components (boxes) of the system under development along with the interactions (arcs) that take place between those components.

2. Control-Flow Model

A Control-Flow Diagram (CFD) models the flow of execution through data objects. This is a venerable approach to specifying the flow design of a system with a great amount of precision. UML offers a collection of diagrams to convey the same type of system information. These diagrams are activity diagrams, collaboration (object-interaction) diagrams, and sequence diagrams.

Each rhomboid represents a software object and each directed arc represents a single flow of control ordinally segmented according to the call sequences.

3. Class Diagram

A class diagram is a superset of an inheritance model in that it models the class taxonomy; however, it also models relationships between the various class entities like a semantic model.

A class diagram is a type of UML (Unified Modeling Language) diagram. A class diagram describes the various entities that will be used to construct the functioning system. Describing object (an object is an instantiated class much like a process is an instantiated program) attributes and operations allows the developer to begin to logically apportion data and functions in a fashion that is natural to human beings. Participating objects are defined via their classes (or templates). Their capabilities, their roles, and their associations / relationships are specified. By specifying the classes from which objects are created, a developer is essentially adding parts to a kit from which a fully functional system will be produced. This type of model is iterative. It may start with classes that only have names. As the developer's understanding of the problem improves, the developer can begin to add the necessary attributes and operations that the classes will need. In this way, the solution system is systematically developed increment by increment.

4. Collaboration Diagram

A collaboration defines a collaborative effort between cooperating objects. Again, an object is an instance of a class. The collaboration diagram models the interactions that take place between the collaborating objects to affect a desired end result.

5. Sequence Diagram

Sequence diagrams model the sequence of interactions between collaborating objects with the arrow of time pointing downwards. The first interaction or call would be at the top of the diagram just below the object squares. The second interaction would be the next call down and so on. Sequence diagrams are sometimes informally referred to as "lollipop diagrams" since the objects are shown as squares with long stems hanging down thus giving them the appearance of a lollipop. The reason for this odd type of symbol is that the interactions need to be drawn between objects in swim lanes so that the vertical spatiality of the diagram is easy to discern.

6. Activity Diagram

Activity diagrams are very similar to the previously discussed control flow diagrams. Activity diagrams show the flow of execution through the various activities that occur during a given operation. Activity diagrams are very semantically rich design artifacts and are therefore good tools for drilling down into the finest details of a detailed design.

7. Pseudocode

Pseudocode can be anything from sentence fragments to a formally specified language. Pseudocode allows expression of operations in an easily readable from without being constrained by the demands of the compiler. Pseudocode also allows the creater to code at the level of detail currently known. Code that must be compiled might not lend itself to high levels of abstraction and low levels of detail and yet that level of detail might be exactly what the design artifact needs to capture and convey to the user of the artifact.

8. V-Model Diagram

As illustrated in the below V-Model diagram, at this point in the software development process, creation of unit test plans should begin. Unit tests must focus on the implementation details of the various software components of the system under development.

SCOPE OF THE DETAILED DESIGN

Detailed design ensures that the overall design solution satisfies the projects objective. Often the breadth of scope is so vast that no single manager, engineer, operator or scientist has the knowledge to provide the overall detailed design and engineering solution.

If the goal of a project included the need to generate energy, a mechanical design engineers input may be to recommend a piston or turbine. An electrical design engineers input maybe to recommend a generator or solar power and a chemical design engineer's recommendation maybe to include a reaction which would provide exothermic or endothermic energy dependent on the need. Obviously not all types of energy creation would be suitable for every project. In this example, as in many other instances, it is therefore necessary for all elements of the problem to be considered and the most suitable decided upon during the build up of the detailed design and engineering solution.

For successful detailed design, there is often the need for someone to take control of the various stakeholders and manage them. Often a project manager will be appointed to bring all the interested parties together and work towards a common goal which will result in a full detailed design and engineering solution.

There are many frameworks that engineers will adopt in reaching the detailed design and engineering phase of a project. In essence each of them takes the idea or concept that solves a problem from a coarse and rough plan/ design to one that is fine and detailed and solves the problem.

The skill of detailed design is to estimate what it is you think is reasonably required and refine and refine to ensure that the plan you settle with fits the bill.

Differences between Detailed Design, High Level Design and Low Level Design

HLD -- High Level Design (HLD) is the overall system design - covering the system architecture and database design. It describes the relation between various modules and functions of the system. data flow, flow charts and data structures are covered under HLD.

High Level Design gives the overall System Design in terms of Functional Architecture details and Database design. This is very important for the ETL developers to understand the flow of the system with function and database design wise. In this phase the design team, testers and customers are plays a major role. Also it should have projects standards, the functional design documents and the database design document also.

LLD -- Low Level Design (LLD) is like detailing the HLD. It defines the actual logic for each and every component of the system. Class diagrams with all the methods and relation between classes comes under LLD. Programs specs are covered under LLD.

Low - Level Design (LLD) - This document is need to do during the detailed phase, the view of the application developed during the high level design is broken down into separate modules and programs for every program and then documented by program specifications.

DLD -- Detailed Level Design (DLD) -- DLD's are referring to a process known as top-down design. In short, when you think about the problem you are trying to solve, you start at the

highest level and then work yourself into the details. This approach works very well when you have an overall structure you want your application to live within. At the macro level you are considering how many machines will be needed to host your application, which existing services you will need to use, etc. As you dive deeper, you are looking at use cases (or user stories if you prefer that terminology), and error handling (use cases have both normal and error condition paths to worry about). As you go even further into the details, you are looking at your algorithm, state transitions, logical sequence, and how internal parts of the code work together.

High Level Design

High-level design (HLD) explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces. The HLD uses possibly nontechnical to mildly technical terms that should be understandable to the administrators of the system. In contrast, low-level design further exposes the logical detailed design of each of these elements for programmers.

Purpose of HLD

Preliminary design — In the preliminary stages of a software development, the need is to size the project and to identify those parts of the project that might be risky or time consuming.

Design overview — As the project proceeds, the need is to provide an overview of how the various sub-systems and components of the system fit together.

In both cases the high-level design should be a complete view of the entire system, breaking it down into smaller parts that are more easily understood. To minimize the maintenance overhead as construction proceeds and the lower-level design is done, it is best that the high-level design is elaborated only to the degree needed to satisfy these needs.

High Level design of Business transactions Applying High-Level design

When it comes to running a business, there are various ways to earn money. Some customers choose an item to purchase, pay with cash, and are never seen again. Others open credit cards with the company and return frequently. Businesses even have contracts with vendors for products and services provided.

Imagine you own a tire and lube shop. You would love to simply have each customer pay cash. However, the marketing department has explained to you that you need to give customers options and find ways to keep them loyal.

Business Transaction

A **business transaction** occurs when goods, services, or money are passed between one person, business, account, etc. and another such entity. For example, this could be you going stopping to

buy gas for your car. The transaction is you purchasing the gas from the gas station. There are three types of business transactions.

Types of Business Transactions

The type of business transaction you have with a customer goes hand in hand with the type of relationship you have. It could be short-term or long-term. It depends on what you offer the customer. Let's review each type.

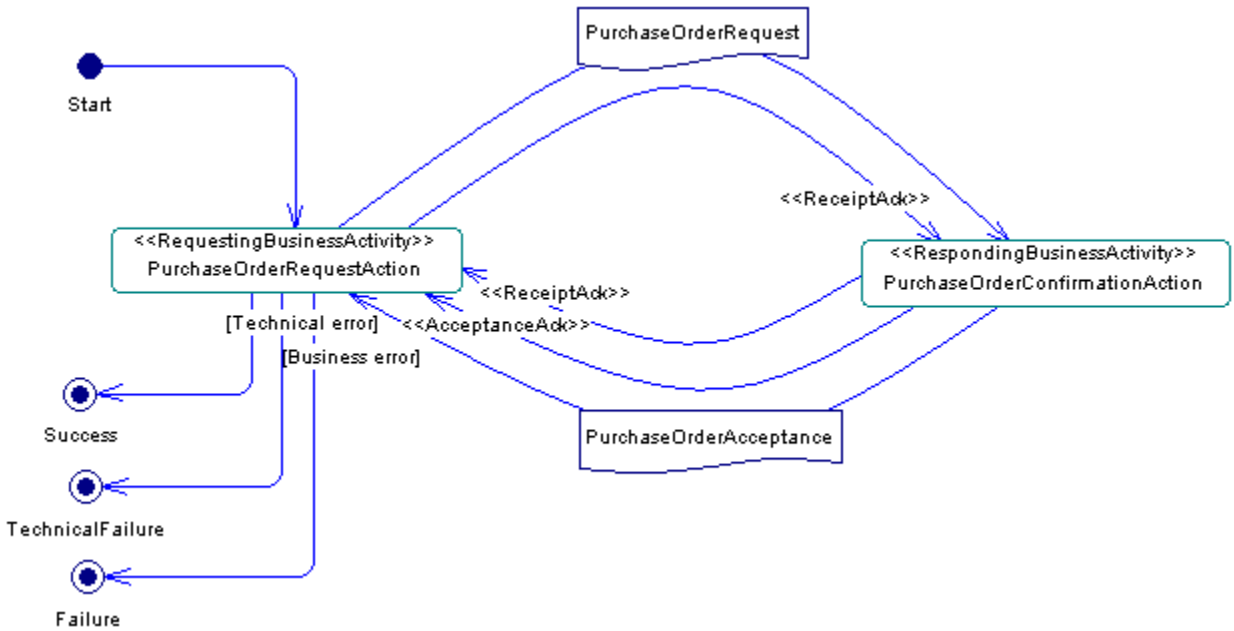
a) **Simple transactions** are typically a singular transaction, which may or may not happen again, between a vendor and customer (e.g. buying cup of coffee, getting a haircut, purchasing a washer and dryer).

b) **Complex transactions** are transactions that require a series of events before completion (i.e. buying with credit, travel through travel agency, buying a home).

c) **Ongoing transactions** typically involve a contract. For example, you and your bank have ongoing business transactions or contracts with vendors.

High Level Design of Business Transaction Applying High Level Design

The Business Transaction process is a composite process with a sub-diagram that High Level designs the Business Signal exchanges between partners. This sub-diagram contains one **Responding Business Activity** (a process with <<RespondingBusinessActivity>> stereotype) and one **Requesting Business Activity** (a process with <<RequestingBusinessActivity>> stereotype):



The **Signals** are designed with flows between Requesting and Responding activities. The <<ReceiptAck>> and <<AcceptanceAck>> flow stereotypes allow the design of a Receipt Acknowledgment Signal and Acceptance Acknowledgment Signal. The Request Document and Response Document are also designed with flows with message formats that design the document format.

A specific ebXML tool Palette allows you to directly create a Business Transaction composite process in the current diagram and initialize the composite process with the Requesting and Responding Activities.

Document Envelope

A **Document Envelope** conveys business information between the two roles in a Business Transaction. One Document Envelope conveys the request from the requesting role to the responding role, and another Document Envelope conveys the response (if any) from the responding role back to the requesting role. This concept is designed with a flow and a message format in the Business Process Model.