Issue Date: <mm/dd/yyyy>
Revision Date: <mm/dd/yyyy>

Document Purpose

This Practices Guides is a brief document that provides an overview describing the best practices, activities, attributes, and related templates, tools, information, and key terminology of industry-leading project management practices and their accompanying project management templates.

Background

The Department of Health and Human Services (HHS) Enterprise Performance Life Cycle (EPLC) is a framework to enhance Information Technology (IT) governance through rigorous application of sound investment and project management principles, and industry best practices. The EPLC provides the context for the governance process and describes interdependencies between its project management, investment management, and capital planning components. The EPLC framework establishes an environment in which HHS IT investments and projects consistently achieve successful outcomes that align with Department and Operating Division goals and objectives.

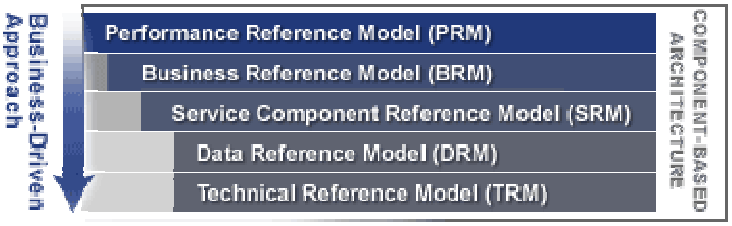
The Design Document describes the technical solution that satisfies the requirements for the Business Product (e.g., system). Either directly or by reference to other documents, the Design Document provides a high-level overview of the entire solution architecture and data design, including external interfaces, as well as lower-level detailed design specifications for internal components of the Business Product that are to be developed.

Practice Overview

Developing systems is usually a complex and challenging endeavor. In general, it is infeasible for a single individual to design and develop a successful system of any complexity without effectively cooperating with many other people. Therefore, it is necessary to apply a disciplined approach to systematically designing, developing, managing, and controlling the development process with the goal of creating high-quality systems that meet the requirements of end-users within a specified project budget and schedule.

According to the Institute of Electrical and Electronics Engineers (IEEE), design is both “the process of defining the architecture, components, interfaces, and other characteristics of a system or component” and “the result of [that] process”. Prelude to system design is analysis which is mainly involved in documenting requirements in a form that can support design and development of the proposed system. This includes business processes as well as the system functionality supporting those processes.

All systems must demonstrate traceability to the Federal Enterprise Architecture (FEA). The FEA is an initiative led by OMB to identify opportunities to simplify processes and unify work across agencies. The FEA is designed using a collection of interrelated “reference models” to facilitate cross-agency analysis and the identification of duplicative investments, gaps, and opportunities. For additional information, please refer to http://www.whitehouse.gov/omb/e-gov/fea/. Its foundation is the Business Reference Model, which is made up of the following five reference models:



- *Performance Reference Model* - is a standardized framework to measure the performance of major IT investments and their contributions to program performance. This model helps produce enhanced performance information to improve strategic and daily decision-making; improves the alignment and better articulates the contribution of inputs to outputs and outcomes; and identifies

performance improvement opportunities that span traditional organizational structures and boundaries.

- *Business Reference Model* - is a function driven framework that describes the Lines of Business and Internal Functions performed by the Federal Government independent of the agencies that perform them. All IT investments (including non-major) are mapped to the BRM to identify collaboration opportunities.
- *Service Component Reference Model* - provides a common framework and vocabulary for characterizing the IT and business components that collectively comprise an IT investment. The SRM will help agencies rapidly assemble IT solutions through the sharing and reuse of business and IT components. A component is a self contained process, service, or IT capability with predetermined functionality that may be exposed through a business or technology interface.
- *Data Reference Model* - describes, at an aggregate level, the data, and information that supports government program and business line operations. This model enables agencies to describe the types of interaction and exchanges that occur between the Federal Government and citizens.
- *Technical Reference Model* - provides a framework to describe the standards, specifications, and technologies supporting the delivery, exchange, and construction of business (or service) components and eGov solutions. The Federal TRM unifies existing Department TRMs and electronic Government guidance by providing a foundation to advance the reuse of technology and component services from a government wide perspective.

Design Approach

Design activities link requirements analysis to implementation of those requirements. There are a number of recognized strategies to assist with the process of system design. However, functional decomposition and object-oriented design approaches are the most commonly recognized. Regardless of which approach is used it should describe the architecture of the system, how it is decomposed, organized, and any interfaces between components.

- *Function-oriented Design* - Decomposes requirements using a top-down approach, first identifying major system functions, then elaborating and refining them to a level appropriate for design
- *Object-oriented Design* - Component-based design methodology that decomposes major system functions into objects rather than procedures
- *Data-structure Centered Design* - Designs a system starting from the data it manipulates rather than from the function it performs. The structures of the input and output data is first identified and then the control structure of the system is developed based on that data
- *Integrated Definition (IDEF)* - A modeling techniques designed to capture the processes and structure of information in an organization through the use of sixteen methods each designed to capture a particular type of information through modeling processes

System design approaches may vary from organization to organization. However, regardless of which design approach is used, a best practice approach to design is comprised of activities resulting in analysis of requirements to produce a description of the structure, organization, and operability of the system being developed. Design efforts decompose and describe system components to a level that allows for their construction. Any successful design approach must have documented guidelines designed to support at least three basic components:

- Identification of classes and objects
- Description and diagramming of relationships between classes and objects (logical blueprints)
- Definition and diagramming of object behaviors by describing functionality of each class

Identification of objects includes data and the procedures necessary to operate on that data. Classes consist of collections, sets, groups, or configurations containing items regarded as having common attributes or traits. Once defined, relationships can be represented in a set of associated models and or flow diagrams. Sometimes referred to as Logical Blueprints or Patterns these models are used to further refine the representation of the system architecture.

A Logical Blueprint identifies the logical node on which each system function resides and diagrams the interfaces between those nodes. This modeling activity should include discovering potential problem areas as well as mechanisms for generating desired system behaviors. Identifying key objects and their responsibilities in different system scenarios is critical to building a successful system. Designers and developers also need to consider how to organize classes and objects. Design documents define this and how to manage any associated complexities and help ensure that system components work together to produce desired results.

Regardless of the method used, system design consists of at least three main activities that fit between the requirements phase and development/testing phase of a project's life cycle. These three activities will be defined further into six phases later in this document.

- *Architectural Design* - Describes the top-level structure and organization of the system and how the system is decomposed and organized into its various components. It defines the internal structure of a system, the way it is constructed, its subsystems, components, and the relationships between them
  - *System Architecture* - Is concerned with how the application interacts with other applications, not necessarily how the application itself works but, that the appropriate data is passed between applications correctly.
  - *Application Architecture* - Is concerned with how the individual components of the system work together, security, best designs interfaces, etc
- *System Design* - Produces a blueprint of the system to be developed by describing specific behaviors of each system component sufficiently to allow for their development
- *Documenting Design* - Design documents that record how design elements will be decomposed, organized, and packaged.

Architecture Design

This section outlines the system and hardware architecture design of the system that is being built. Architecture design describes the top-level structure and organization of the system and how the system is decomposed and organized into its various components. It defines the internal structure of the system, the way it is constructed, its subsystems, components and relationships between them, system architecture, and how the application interacts with other applications. A well documented architecture design may include items such as:

- *Logical View* - A logical view (flow chart) describes the architecturally significant components of the system design such as sub-systems, packages, hierarchies and layers.
- *Hardware Architecture* - Hardware architecture is a description and logical view of the hardware, processes, disciplines, and the rules and relationships among those elements necessary to effectively implement the system being developed. Moreover, a good architecture provides for an easy mapping to user requirements. A robust hardware architecture exhibits an optimal degree of fault-tolerance, backward compatibility, forward compatibility, extensibility, reliability, maintainability, availability, serviceability, usability, and other such things as necessary and/or desirable.
- *Software Architecture* - Software architecture is a description and logical view of the software, processes, disciplines, and the rules and relationships among those elements necessary to effectively build the system being developed. Moreover, a good architecture provides for an easy mapping to user requirements.
- *Security Architecture* - Security architecture is a description and logical view of the system security, processes, disciplines, and the rules and relationships among those elements necessary to effectively secure the system being developed. Moreover, a good architecture provides for an easy mapping to user requirements and should describe all of the technical requirements that affect security such as security audits, cryptography, user data, system identification/authentication, resource utilization, etc.
- *Communication Architecture* - Communication architecture is a description and logical view of the system communication, processes, disciplines, and the rules and relationships among those elements necessary to effectively communicate among elements of the system being developed. Moreover, a good architecture provides for an easy mapping to user requirements and should describe all of the technical requirements that affect interfaces such as protocol management, scheduling, directory services, broadcasts, message types, error and buffer management, security, etc.
- *Performance* - Describes performance requirements and any related processes including a detailed description of specific performance requirements and how they are associated with specific project functionality/deliverables. This may include information such as cycle time, speed per transaction, test requirements, minimum bug counts, speed, reliability, utilization etc.

System Design

This section outlines the specific behaviors of each system component, module, interface, etc to a level that allows for their development. A well documented system design may include references to items such as:

- *Product Requirements* – Reference the high level features or capabilities that the business team has committed to delivering to a customer. Product requirements do not specify how the features or the capabilities will be designed.
  - *Functional Requirements* address what the system does. They define any requirement that outlines a specific way a product function or component must perform.
  - *Non-Functional Requirements* (also referred to as Quality of Service by the International Institute of Business Analysts, Business Analysis Body of Knowledge) address items such as the technical solutions, topics that address the number of people who need to use the product, where the product will be located, the types of transactions processed, and types of technology interactions.
- *Database Design* - Define rules and recommendations to be used as guidelines when designing the system's database. Guidelines should be documented for items such as mapping classes to database structures, handling design inheritances, naming conventions, structure (tables, data fields, dependencies, reports, extracts, etc).
- *Data Conversion* - Data conversion is the conversion of one form of data to another usually for the purpose of application interoperability or capability.
- *Application Program Interface* - An application programming interface (API) is a source code interface to support requests for services made by the computer program. An API specifies details of how two independent computer programs interact.
- *User Interface Design* - User interface design focuses on the user's experience and interaction with the system. The goal of user interface design is to make the user's interaction with the system as simple and intuitive as possible. The design of the user interface is usually performed in conjunction with prototyping and may include components such as workflows, wireframes, prototypes, etc.
- *Performance* - Describes performance requirements and any related processes including a detailed description of specific performance requirements and how they are associated with specific project functionality/deliverables. This may include information such as cycle time, speed per transaction, test requirements, minimum bug counts, speed, reliability, utilization, end-user requirements, etc.

Documenting Design

This section outlines how design elements will be decomposed, organized, and packaged. This would document items included in architectural and system design within a Design Specification document. The Design Specification document is the main design document for the system and describes how major aspects of design will be managed. Either directly or by reference to other documents, the Design Specification document should address the following:

- Architecture Design
  - Logical View
  - Hardware Architecture
  - Software Architecture
  - Security Architecture
  - Communication Architecture
  - Performance
- System Design
  - Requirements
  - Database Design
  - Data Conversion
  - Application Program Interfaces
  - User Interface Design
  - System Performance
  - Section 508 Compliance

Business Analysts, Architects, and Designers are the primary individuals involved in the design process. Business Analysts translate the business requirements into technical requirements for the Architects and Designers. Architect lead and coordinate technical activities such as translating requirements into architecture and system design. Designers are responsible for seeing that individual classes, packages, and sub-systems meet the requirements of the underlying implementation technology. The types of artifacts created by these individuals may include:

- Interface definition and/or design models
- System and architecture definition and/or design models
- Workflows

- Updated business requirements

One design technique that can be used is an iterative process that begins by defining a high-level design and then decomposing that design into more detailed components that complement the implementation requirements of the system. Another technique is Joint Application Design (JAD). This technique brings developers and end-users together to gather requirements as well as design and develop the business product. Designers should follow the design technique that is standard practice at their OPDIV.

Design activities should continue until the system to be developed is defined to a point detailed enough to allow the project team to successfully build the system to meet the project requirements. The actual life cycle of design activities will vary depending on the experience of the project team, the development environment, the complexity of the system being developed, and the detail to which the system requirements have been specified. However, because new systems affect users, and often change the environment that they work in, final system requirements and design may not be fully completed until later in the project's life cycle. As a result, the design process must provide a stable basis for what is often an ongoing, iterative, process.

Design documents record the necessary information required to effectively define system design and architecture to a level that can appropriately guide developers in building the system. These documents provide a level of visibility and verification of the design used and may also include a breakdown of topics related to quality and measurements and how such measures are defined and will be used. A variety of design techniques exist to help ensure the quality of a system design. Some of which include:

- Analysis and reviews
- Simulations and prototypes
- Metrics and measures

Design documents are incrementally and iteratively produced during the system development life cycle, based on the particular circumstances of the IT project and the system development methodology used for developing the system. These documents are initially created during the Planning Phase of the project and updated as necessary throughout the design process. The preliminary versions of these documents are reviewed during the EPLC Stage Gate Preliminary Design Review; the final versions are input to the EPLC project Detailed Design Review. Their intended audience is the project manager, project team, and development team. Some portions of the design documents, such as graphical user interface (GUI) design, may be shared with the client/user and other stakeholder whose input/approval into the GUI is required.

Design activities can be divided into roughly six phases which may include:

- *Requirements Analysis* - Determines the user requirements. This happens prior to the actual design. Collects and analyses user requirements through activities such as focus group, user trials, interviews, observations, etc. This phase defines for the project team an understanding of the intended system functionality, any user concerns and any assumptions/constraints
- *Conceptual Design* - Models the underlying business. Typically, the conceptual design is divided into three models:
  - *Data* - Identifies and defines data entities and defines any relationships between them
  - *Business Functions* - Identifies and defines component business functions
  - *Communications* - Maps interactions between business functions and data
- *Logical Design* - Initiates the process of showing the interrelationships of the solution's components
- *Physical Design* - Examines how to physically implement the logical design
- *Construction* - Converts the logical design into actual system code
- *Evaluation* - Validates the effectiveness of the system that was developed. This is usually accomplished via testing, observation, and user feedback

In most cases, the project team will also need to document strategies for dealing with items such as:

- Specifying the mapping from design to implementation at both the package and class levels
- Documenting and representing reusable components, component systems, libraries, commercial off-the-shelf (COTS) products, etc
- Documenting transaction management and how it will be accomplished including the interaction of transaction management, fault management, and system recovery

- Documenting standards for programming structure and algorithm guidelines, hardware and software interfaces, operations, messages, detecting, handling, and reporting faults, etc.

Practice Best Practices

- **Iterative Process** - Design is often an iterative process involving repetitive interaction with system users to refine design to a point adequate to build a system that satisfies requirements.
- **Clearly Documented** - Design details should be clearly documented in the form of system design documents that may include items such as interface definitions, workflow, models, use cases, etc. Design documentation should not only include individual component designs but should also document any other system-wide concerns
- **Review and Approve** - Review the design and design documents for precision, completeness, and usability. Design should be reviewed and approved by all appropriate stakeholders
- **Traceability** - System design should be directly traceable to satisfying requirements defined during system analysis
- **Reviews** - Regular reviews of design documents and design efforts, and their traceability is a good project management practice
- **Interfaces** - Systems and devices have varying interfaces. Understand any required system interfaces prior to detailed design. This may include hardware, software, network, devices, etc
- **Think Small** - Often systems are forced to work within the confines of devices that may have small screens, small processors, small memory, etc. Design the system accordingly
- **Simple** - Keep the user interface and system navigation simple and consistent
- **Security** - Be aware of system security requirements and how they may affect design
- **Tradeoffs** - In most instances there are a number of ways to accomplish the same action. Be aware of the tradeoffs of selecting one approach as opposed to another and how the affects of that decision may propagate throughout the system
- **Balance** - Balance the goals of the system or requirements against future costs and benefits.
- **Document** - Document rationale behind design decisions and tradeoffs
- **K.I.S.S.** - The key purpose of design is to build a system simple and easy for users to maintain and operate, not necessarily to experiment with the latest high-tech fad. Focus on the needs of the user and keep the system simple
- **Assumptions/Constraints** - Record and address all assumptions, constraints, issues, etc
- Strategy - In some cases it may be necessary to define a unique design strategy for a specific software development project or functionality
- **Design Usability** - The system design must be accessible to and understandable by all users. Include applicable 508 compliance standards.

Practice Activities

- **Guidelines** - Define and document design process guidelines such as recording of assumptions, constraints, issues, etc; use of prototyping, experimentation, etc; design approach, logical blue prints/patterns, documentation, templates, and notation
- **Business Process Specification** - Define the intended future state of business processes
- **Functional Specification** - Define the functional part(s) of the system being developed, external relationships, interaction with the user and other system elements
- **Logical Specification** - Define the internal logic of the system explaining how it operates
- **Conceptual Design** - Models the underlying business.
- **Logical Design** - Initiates the process of prototyping the user interface
- **Physical Design** - Examines how to physically implement the logical design
- **Construction** - Converts the logical design into actual system code
- **Evaluation** - Validates the effectiveness of the system that was developed. This is usually accomplished via testing, observation, and user feedback
- **Verification** - Verify each step within the design process to ensure traceability to requirements that delivers the expected system functionality