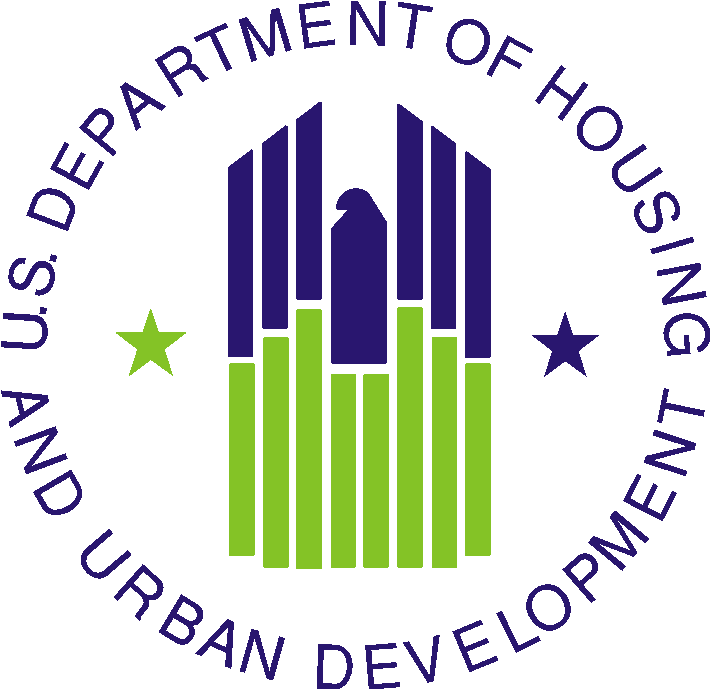
The Department of Housing and Urban Development



Service Layered  
Architecture Profile  
(SLAP)

Version 1.0

**<Date>**

**<System Name>**

**Solution Information**

|  |  |
| --- | --- |
|  | Information |
| **Solution Name** |  |
| **Solution Acronym** |  |
| **Project Cost Accounting System (PCAS) Identifier** |  |
| **Document Owner** |  |
| **Primary Segment Sponsor** |  |
| **Version/Release Number** |  |

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version No.** | **Date** | **Author** | **Revision Description** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Note: The latest version of this document supersedes all previous versions.*

Table of Contents

[Document History 2](#_Toc178157564)

[Introduction 5](#_Toc178157565)

[Architecture Guidance Disclaimer 5](#_Toc178157566)

[Background 6](#_Toc178157567)

[**Investment Selection Information** 6](#_Toc178157568)

[**HUD Business Architecture and Example Review Process Requirements** 6](#_Toc178157569)

[**Enterprise Architecture Guidance** 6](#_Toc178157570)

[Document Conventions 10](#_Toc178157574)

[Architecture Recommendation Summary 10](#_Toc178157575)

[Case Management Capability 10](#_Toc178157576)

[Case Management Market Analysis 11](#_Toc178157577)

[Architectural Recommendations for Implementing the XYZ Application 11](#_Toc178157578)

[Recommendation 1: Cloud First - XYZ SHALL Be Developed as a Custom-Built Open-Source Solution Deployed To Azure Cloud 13](#_Toc178157579)

[The HUD Open-Source Application Technology Profile 13](#_Toc178157580)

[Open-Source Reference Implementation Architecture 14](#_Toc178157581)

[Recommendation 2: OCIO will provide Development/Test as a Service (DTaaS) 16](#_Toc178157582)

[Recommendation 3: XYZ SHALL Be Implemented in Accordance With HUD Layered Service Architecture 17](#_Toc178157583)

[XYZ Specification Architecture 17](#_Toc178157584)

[Service Layer Examples 19](#_Toc178157585)

[Recommendation 4: XYZ SHALL Employ Design Best Practices 23](#_Toc178157586)

[Model/View/Controller design pattern 23](#_Toc178157587)

[Intended use of JMS and MuleSoft ESB 23](#_Toc178157588)

[Rules Engine 23](#_Toc178157589)

[Transaction Management 24](#_Toc178157590)

[System Monitoring 24](#_Toc178157591)

[Development/Test as a Service (DTaaS) 24](#_Toc178157593)

[Test Automation 25](#_Toc178157594)

[Recommendation 5: XYZ SHALL Reuse Existing Utility Services 26](#_Toc178157595)

[Recommendation 6: Integrate with the Following HUD Systems 27](#_Toc178157596)

[Recommendation 7: XYZ Shall Use HUD ESB 28](#_Toc178157597)

[Recommendation 8: XYZ SHALL Use HUD EDW for Data Warehouse Needs 29](#_Toc178157598)

[Recommendation 9: XYZ Shall Align To HUD CARS Data Model 30](#_Toc178157599)

[Recommendation 10: XYZ SHALL use ICAM solution to implement application security 31](#_Toc178157600)

[References 32](#_Toc178157601)

[Appendix A: HUD Layered Service Architecture Guidance 33](#_Toc178157602)

[Appendix B: HUD Enterprise Nonfunctional Requirements 36](#_Toc178157603)

[Appendix C: XYZ Specification and Deployment Views 37](#_Toc178157604)

[Appendix D: XYZ Logical Data Model 46](#_Toc178157605)

[Appendix E: Agile Development 50](#_Toc178157606)

[Appendix F: Acronym and Definitions 54](#_Toc178157607)

# Introduction

## Architecture Guidance Disclaimer

## Background

### **Investment Selection Information**

### **HUD Business Architecture and Example Review Process Requirements**

### **Enterprise Architecture Guidance**

| Name / Title | Description | POC or Universal Resource Locator (URL) Address |
| --- | --- | --- |
| XYZ Business Process Model and Requirements | Draft business process model developed by EA (Oct 2015) and supporting User Stories drafted in (Nov 2015). | [EnterpriseArchitecture@Hud.Gov](mailto:EnterpriseArchitecture@Hud.Gov) |
| Project Portfolio Management (PPM) | The Department of Housing and Urban Development's (HUD's) Project Planning and Management (PPM) Life Cycle V2.0 provides practical approaches to optimize innovation, minimize schedule and budget risk, and better plan and execute projects. | <http://portal.hud.gov/hudportal/HUD?src=/program_offices/cio/ppm/PPMV20Artifacts> |
| Security | For information regarding application security requirements please contact the HUD Chief Information Security Officer. | HUD Chief Information Security Officer |
| Enterprise Technical Architecture Version (ETA) 1.1 | The HUD ETA documents outline layered and tiered architecture principles and standards for developing modern applications. This document provides an overview of the HUD ETA, the layers of that architecture, the security required to safeguard it, and the infrastructure required to operate it. This document is pertinent to application and system stakeholders. | [EnterpriseArchitecture@Hud.Gov](mailto:EnterpriseArchitecture@Hud.Gov) |
| Common Application Relational Schema (CARS) | The Common Application Relational Schema (CARS) logical data model is the Enterprise Data Model for HUD. The purpose of the CARS model is to define the data needed by HUD to fulfill the organization’s mission using a methodology to allow the model to be used by new development projects to establish the foundation of the data model and data base. | [EnterpriseArchitecture@Hud.Gov](mailto:EnterpriseArchitecture@Hud.Gov) |
| HUD Cascading Style Sheet | A Cascading Style Sheet template that may be used for development of websites and web-based applications. | <http://portal.hud.gov/hudportal/documents/huddoc?id=hudapril.css> |

Exhibit 1 – HUD Architecture Guidance Resources

## Scope

## Audience

## Purpose

## Document Conventions

# Architecture Recommendation Summary

## System Capability

## System Market Analysis

## Architectural Recommendations for Implementing the XYZ Application

|  | Recommendation | Rationale | Notes |
| --- | --- | --- | --- |
| 1 | **XYZ SHALL Be Developed As A Custom-Built Open Source Solution Deployed To The Azure Cloud** | Open Source software provides a low-cost, low-risk alternative to proprietary software. HUD seeks to establish a standardized, managed DevOps environment to speed delivery and reduce the cost of custom-built solutions.  This option provides maximum flexibility to HUD in terms of maintenance & operation of the system as business rules and needs change over the years. | HUD OCIO has already made the investment to establish the open-source architecture. |
| 2 | **OCIO will provide Development/Test as a Service (DTaaS)** | The XYZ project will have direct access to and the support of the Enterprise Architectural Design Services team and the HUD CI/CD team to facilitate the use of the provisioned environment. This will expedite delivery of a quality XYZ solution.  Expedite delivery by enabling solution team to start work on Day 1. | Improved system quality due to the automated continuous testing capabilities inherent in the HUD CI/CD environment.  Reduced total cost of ownership as result of improved system quality. |
| 3 | **XYZ SHALL Be Implemented In Accordance With The HUD Layered Service Architecture** | Establishing interfaces that align with HUD’s Enterprise Capability Model and HUD’s Layered Service Architecture will provide investment and technology flexibility in the future, simplify HUD’s application portfolio, and provide assets that can be extended for enterprise reuse and consolidation. | Positions capabilities developed for XYZ to easily identified and reused in support of subsequent projects. |
| 4 | **XYZ SHALL Employ Design Best Practices** | Employing consistent design patterns simplifies downstream integration and provides higher degrees of information assurance.  The ADS team will provide sample executable code and an application framework that demonstrates the anticipated use of these design patterns for XYZ. | Expected to reduce out-year maintenance and support costs |
| 5 | **XYZ SHALL Employ Utility Services** | As part of custom code development, expose utility services securely as REST based APIs so that other applications do not have to rewrite the business logic. They can simply invoke the RESTful services and perform the desired operation. | Expected to reduce out-year maintenance and support costs |
| 6 | **XYZ SHALL Integrate With The Following known HUD Systems. List will be updated as final scope and business requirements are delivered to OCIO.** | CHUMS, LEAP, AARTS, SFHEDW  Please see **Error! Reference source not found.**.  The XYZ solution will provide reusable integration patterns for accessing these systems. | These interfaces will be factored in to the cost of any XYZ solution. |
| 7 | **XYZ SHALL Use HUD ESB** | HUD has selected MuleSoft’s Enterprise Service Bus (ESB) and established a policy that all applications will utilize the ESB for systems integration. | Will reduce the impact of change between systems integrated using the ESB. |
| 8 | **XYZ SHALL Use HUD EDW For All Data Warehouse Needs** | The data warehouse will provide reliable and consistent data for the enterprise. This will eliminate inconsistencies between isolated views of data that can come from reporting on individual subsystems. | The Enterprise Architecture Enterprise Data Warehouse team will implement any reporting and data warehousing requirements for the XYZ solution. |
| 9 | **XYZ SHALL Align To HUD CARS Data Model** | The Common Application Relational Schema (CARS) logical data model is the enterprise data model for HUD. Alignment to the CARS model and its methodology and design approach will ensure data quality, adhere to consistent business rules, and institutionalize HUD data standards. | The Enterprise Architecture Enterprise Data team will develop CARS data model which will be used by XYZ solution. |
| 10 | **XYZ SHALL use ICAM solution to implement application security** | ICAM solution (TBD) will be used for HUD partners and HUD PIV personnel to authenticate and authorize into the applications. | The Enterprise Architecture team recommends integrating with existing ICAM Solution (infrastructure) but build out ICAM functionality to standards so that there is easy migration to another platform/solution down the road. |

Exhibit 2 - Recommendations for XYZ Applications

The following sections will address each recommendation in more detail.

# Recommendation 1: Cloud First - XYZ SHALL Be Developed as a Custom-Built Open-Source Solution Deployed to Azure Cloud

## The HUD Open-Source Application Technology Profile

|  |  |
| --- | --- |
| Technology | Product Name |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Exhibit 3 - Application Technology Profile Overview

## Open-Source Reference Implementation Architecture

# Recommendation 2: OCIO will provide Development/Test as a Service (DTaaS)

|  |  |
| --- | --- |
| Technology | Product Name |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Exhibit 5 - Development Environment Technology Profile

HUD EA CI/CD Team will provide support for the Development Test as a Services (DTaaS) environment.

# Recommendation 3: XYZ SHALL Be Implemented in Accordance With HUD Layered Service Architecture

## XYZ Specification Architecture

Exhibit 6 – Overview of Example Review System Specification Architecture

## Service Layer Examples

**Process Service Layer Example**

Exhibit 7 – Example Process Layer Service Internal Design

**Exhibit 8 – Description of Example Process** **Layer**

|  |  |
| --- | --- |
| Component Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Exhibit 8 – Description of Example Process Layer Service Internal Design

**Core Data Service Layer Example**

Exhibit 9 – Example Core Data Service Internal Design

*Exhibit 10 –Core Data Service Internal Components* provides a description of each component in Exhibit 9.

|  |  |
| --- | --- |
| Component Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |

Exhibit 10 –Core Data Service Internal Components

**Underlying Service Layer Example**

Exhibit 11 – Example Underlying Service Internal Design

**Exhibit 12 – Underlying Service Internal Components** provides a description of each component in *Exhibit 9 – Example Core Data Service*. This is a typical example of how the Underlying Services Layer component would be packaged. The underlying services layer components are typically called from the Capability or the Core Data services layer. The underlying services are packaged within a Mule ESB component that encapsulates the internal details of accessing and transforming data from external systems.

|  |  |
| --- | --- |
| Component Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Exhibit 12 – Underlying Service Internal Components

# Recommendation 4: XYZ SHALL Employ Design Best Practices

## Model/View/Controller design pattern

## Intended use of JMS and MuleSoft ESB

## Rules Engine

## Transaction Management

## System Monitoring

## Logging

## Development/Test as a Service (DTaaS)

## Test Automation

# Recommendation 5: XYZ SHALL Reuse Existing Utility Services

# Recommendation 6: Integrate with the Following HUD Systems

Exhibit 13 – XYZ Candidate System Interfaces

|  |  |
| --- | --- |
|  | Candidate System Interfaces |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 

# Recommendation 7: XYZ Shall Use HUD ESB

# Recommendation 8: XYZ SHALL Use HUD EDW for Data Warehouse Needs

# Recommendation 9: XYZ Shall Align To HUD CARS Data Model

# Recommendation 10: XYZ SHALL use ICAM solution to implement application security

# References

1. Department of Housing and Urban Development, DME Activity Summary: Access To Credit With XYZ - *Appendix A – Initiative: Access To Credit*; Project: Example Review System (XYZ)/Quality Assessment Methodology; Project Evidence Demonstrating Compliance to Statutory Condition 1. (FY2014)
2. HUD Example Review System Business Process Reengineering Results: XYZ Target Business Process.
3. HUD Example Review Process Requirements
4. HUD ETA Documents
5. Example Review System (XYZ) - To-Be Process Flows 2015-11-06.doc
6. XYZ -demo-userstories\_v1.xls
7. HUD XYZ Demo Functionality.ppt

# Appendix A: HUD Layered Service Architecture Guidance

In a Service Layered Architecture, we classify services according to the type of capability they provide. By focusing each service on a particular type of capability, it enables greater modularity and separation of concerns, enabling them to be more easily shared or composed into new services. For example, core business services (entity) provide operations focused on managing the information about a key business resource - like customers, orders, or products. These are independent of the processes that use that information. Process services provide operations that enable solution assemblers to interact with the different steps in the process. In turn, the process services then consume the appropriate core business services.

In this way, Solution assemblers can change a process service, add new operations and such, or add new processes, without impacting the core business services (data services). If we mix these capabilities together, we run the risk for example that the entity information is only provided via a specific process. We see this often and it's just too coarse grained and not a very agile approach. Business processes typically change more frequently than the information the business needs to manage. Many different processes require information about the same entity types and entities. To ensure our architecture is responsive to these change patterns, Core business services often have broader scope of use than a single business process.

**HUD-EAS recommends the following Service Layers for XYZ:**

| ****Architectural Layer**** | ****A Service in this layer**** | ****Guidance**** |
| --- | --- | --- |
| Process Service Layer | is designed to provide functions that support one specific business process or sub-process. This service SHOULD be independent from any particular user interface design. | MUST be associated with one Business Process or Subprocess, and no other Business Modeling Element. |
| Capability Service Layer | is designed to support a particular business capability. This service SHOULD be independent from any particular business process. | MUST be associated with one Business Capability and no other Business Modeling Element. |
| Core Business Service Layer (data services) | is responsible for maintaining records about the instances of particular set of business types. This service SHOULD be independent from any particular business process or business capability. | MUST be associated with one or more Business Types, and no other Business Modeling Element. |
| Utility Service Layer | provides common or specialized operations that MAY be consumed by any other business service. | MAY be associated with a Process, Business Capability or Business Type or Business Event in cases where appropriate if helpful, but need not be. |
| Underlying Service Layer | SHOULD normally be hidden from solution developers, since invocation is difficult or error-prone in some way. For example, the operations are obscure, they are not in the terminology used by your business, some operations are inappropriate, the operations are highly generic and could get used inconsistently. This service can be “hidden” from solution developers by “wrapping” it within higher level services, and making a rule that solution software and process services do not directly request operations of a service in this layer. | MUST not be associated with any business modelling element, but … may be associated with one or more Applications, which provide some of the implementation of this service, through the “provides implementation” association. |

**XYZ Architecture Layer Rules**

| **Layer: Main Role** | **Operations are:** | **Dependencies allowed:** | **More rules:** | **Data Storage:** |
| --- | --- | --- | --- | --- |
| Solution Logic (Presentation):  provides an effective end-user experience | Non-existent - this layer does not contain services | May call Process, Core Business & Utility services directly | Supplies user interface, validation, user messages, session management | Not normally, except for temporary session data |
| Process Services (Business):  Orchestrate other services; apply business process rules. | UI-independent, but designed for a specific business process | May directly call Core Business, Underlying & Utility services | May directly call Core Business, Underlying & Utility services | Only where not stored by Core Business Services. Stored data likely more transient than for Core Services |
| Business Capability Services(Business):  Process independent and Channel (UI) independent business services | UI- and business process- independent, so can be used in different contexts | May directly call other Core Business, Underlying and Utility Services | Cyclic dependencies not normally permitted, except for “call-back”. May not call Process Services | Enforce Business rules |
| Core Business Services (Business Entity):  Apply enterprise-wide business rules | UI- and business process- independent, so can be used in different contexts | May directly call other Core Business, Underlying and Utility Services | Cyclic dependencies not normally permitted, except for “call-back”. May not call Process Services | Maintain the principal data and enforce data integrity rules |
| Utility Services Highly reusable services or those employed to standardize technical infrastructure or to hide complex business calculations | UI-, business process- and often domain- independent | May call other Utility Services directly. Some may use Underlying Service. | Cyclic dependencies not normally permitted. | Often required — for directories, look up tables and audit trails for example. |
| Underlying Services (Integration)  difficult for solutions developers to consume correctly | Highly generic or implementation-leaking, so its interface not ideal for exposing to solution developer | May call Utility Services, but normally would not | May not call Core Business or Process Services | Often maintain significant data stores, and core services call underlying services |

# Appendix B: HUD Enterprise Nonfunctional Requirements

See Excel Spreadsheet Attachment named for this appendix.

# Appendix C: XYZ Specification and Deployment Views

**Exhibit 6** is a Unified Modeling Language (UML) class diagram that depicts the services (classes stereotyped as Service Specification) that participate in the XYZ. The services are arranged in different layers and these layers are described in *Appendix A: HUD Layered Service Architecture Guidance*. The associations between the services are not depicted in this diagram, but are shown in other diagrams presented in this section.

The stereotypes used have two functions: The first function provides additional properties for documentation purposes, and the second function provides the properties needed to generate implementation code from the UML objects. The stereotypes used in the Specification Model are Service Specification and Service Operation Specification. Listed below are the properties of each stereotype:

**Service Specification -** Defines the interface that the service offers.

**Service Operation Specification** – Defines the behavior of an operation offered by the service specification.

The notation for each stereotype attribute denotes the optionality and cardinality of the attribute. [1] means that a value for the attribute must be present. [0..1] denotes an optional attribute. The notation [0..n] reflects an optional attribute that may have up to “n” values. “\*” denotes that there is no logical limit to the number of occurrences of this attribute.

**XYZ Intake Process Step – Specification View**

**Exhibit 6 – Overview of Example Review System Specification Architecture**

Services Specified in the Intake Process Step

|  |  |  |
| --- | --- | --- |
| Service Specification | Layer | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**XYZ Example Selection Process Step – Specification View**

**Exhibit 8 – Description of Example Process** Layer

Services, Classes Stereotyped as Service Specifications in the Example Selection Process

|  |  |  |
| --- | --- | --- |
| Service Specification | Layer | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**XYZ Triage Process Step – Specification Review**

Exhibit 10 –Core Data Service

**Services, Classes Stereotyped as Service Specifications in the Triage Process**

|  |  |  |
| --- | --- | --- |
| Service Specification | Layer | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**XYZ Deployment View**

**XYZ Application Deployment View**

The following table provides a list of all the components represented in the deployment view:

Exhibit 14 – Deployment View Component Descriptions

|  |  |  |
| --- | --- | --- |
| Component | Tier | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Appendix D: XYZ Logical Data Model

Example Review Entity List and Descriptions

| Entity Name | Definition |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Appendix E: Agile Development

HUD OCIO has adopted an agile approach for software development to reduce the risk of project failure, and to assure that the delivered system performs as it is intended. These methods have repeatedly been shown to:

* Improve time-to-mission-value
* Reduce project risk
* Reduce cost
* Improve visibility
* Better adapt to changing needs

Agile approaches use an iterative, incremental process that is characterized by small, frequent releases developed in close collaboration with the customer. These practices provide tight feedback loops and frequent opportunities for course correction. Agile approaches are consistent with “modular development” or “modular IT” as defined in OMB. Modular development of IT capabilities requires that programs deliver functionality in increments, generally of no longer than 6 months each. Agile development provides a best-practices way of conducting such incremental delivery.

The greatest process risk for agile teams is simply following a set of mechanical steps or “ceremonies” rather than adhering to agile values and principles and adopting the agile mindset. Teams that do this are likely to lose the empirical and adaptive benefits of agile, and hence lose the natural controls that are inherent in the agile approach.

Therefore, understanding and practicing the core values and principles of agile are the primary means of obtaining the benefits of agile. The essence of agile methods is the use of an “inspect and adapt” paradigm: agile projects practice continuous improvement of processes, requirements, and design.

Agile Practice (s) – the minimum agile practices required of all agile teams include the following:

1. Frequent Delivery – Projects shall deliver usable product to end users at least quarterly.
2. Time-boxed Iterations – Projects shall adopt a fixed-length iteration (between 2-4 weeks) as their standard cadence for planning, completing, and demonstrating potentially-deployable functionality.
3. User Stories – Project teams shall employ User Stories as the basic unit of planning, executing, and tracking their work, with User Stories being defined as small, independent, and testable units of functionality. Note that User Stories under this definition are not necessarily expressed in classic user story format (“As a <who> I want to <what> in order to <why>”). Indeed, many formats are possible for requirements (for example, samples of reports for BI projects) as long as they are small, independent, and testable.
4. Product Owner – Each project shall have a distinct individual representing the business involved with the team, with the authority to make timely decisions regarding user story development, prioritization, and acceptance.
5. Release Planning – Each project shall conduct a release planning exercise culminating in a release planning review (RPR) and a standard set of deliverables prior to commencement of work on a release.
6. Iteration Reviews – Project teams shall conduct a facilitated review at the end of the each iteration to demonstrate and/or test functionality and to solicit feedback from the project’s stakeholders.
7. Retrospectives – Project teams shall hold a facilitated meeting at the end of each iteration to reflect on the team’s performance and identify opportunities for improvement.
8. Continuous Testing – Project teams shall test User Stories up to and including in a staging environment during the iteration in which those stories are developed.

The practices discussed in this appendix represent the foundational practices that all projects must adopt to achieve a minimum level of agility within HUD. However, project teams are not limited by this and are encouraged to adopt other practices that advance the goal of continuous deployment.

Principles: The principles are designed to guide how HUD implements agile processes and represents the foundational principles that all projects must adopt to achieve a minimum level of agility within HUD

Lean Thinking

Lean emphasizes seeing the whole (including the end-to-end view of the delivery pipeline), and is not constrained by organizational boundaries and promotes delivering as fast as possible. Lean also emphasizes the “amplification of learning” through techniques, such as retrospection and mentoring, in order to increase the effectiveness of people, and therefore the speed at which work can be done.

Lean supports the concept of amplified learning, through mentoring and focused efforts to have people learn by doing with appropriate supervision. Agile also promotes the concept of a sustainable work pace, which promotes quality and excellence because people have the energy to pay attention to doing things right instead of rushing things through. This means that work must be planned in a way that does not over-promise, and that is based on actual measurement of work capacity instead of aggressive externally imposed deadlines. Rushed work leads to re-work, and does not save time in the long run.

Transparency

Agile requires that people work across business functions. For example, software cannot be developed in an agile manner unless architecture, testing, security, and release management all collaborate to define a way to perform all of those functions in parallel, in a just-in-time, collaborative manner, rather than one group handing things off to the next in sequence. OIT is defining a new processes that crosses business functions, but such processes must be adaptable, and as such, they will never be fully defined. The organization must rely on its staff to evolve processes over time. This requires being highly proactive, with an emphasis on reaching across silos to achieve an objective and not feeling constrained by rigid processes.

For this level of collaboration to be successful, management must provide a safe environment in which people are encouraged to reach out, across business functions, and are not punished for going directly to the staff they need to talk to. Those who reach out should be encouraged, not admonished. This is a significant change from the practices of most established hierarchical organizations, but it is a norm for innovative organizations. Proactiveness and collaboration are strong drivers of grass-roots innovation, and innovation is one of the strategic goals of OCIO.

Along with normalizing change, agile stresses transparency through continual delivery of output. By designing project activities so that every activity has a demonstrable result that the user can understand, and that is easily recognizable as tangible progress, one is able to make sure that a project stays on course. This greatly reduces risk because erroneous approaches become evident very quickly when the output of work is demonstrated.

Continuous Improvement

In an ever-changing world, business processes need to evolve continuously. This calls for regular reflection and process improvement. Continuous delivery advocates that the best way to address a recurring process problem is to address it sooner – right at the outset of the process, if possible – and that will focus attention on it and get it solved. Both lean and agile advocate identifying work that does not add value and eliminating wasteful activity. Measurement is important for continuous improvement because measurement often shows where problems really manifest. Measurement also helps to determine what the organization’s true capacity is for handling work. Measurement is therefore a critical enabler for agility.

Mission-Focused and Results Oriented

HUD must be responsive to direction from Congress and other key stakeholders, and be ready to implement policy changes rapidly. The implementation of policy cannot be held up by IT impediments. Continuous delivery emphasizes the creation of a repeatable, reliable pipeline process for building and releasing software, and the automation of almost everything in that process pipeline. Agile methods emphasize the delivery of value frequently. At a tactical level, being mission-focused translates into being results oriented. Agile promotes a focus on tangible outcomes, as opposed to progress against a plan. All business processes and work should be defined and conducted in a manner that emphasizes results, rather than tasks. Completion of a task is not evidence of progress, unless it produces something that can be assessed as tangible progress, and ultimately, that it advances the HUD mission. Continuous delivery advocates the concept that “Done means released”, and this means that a task to create an artifact (i.e., software, a business procedure, a document) should never be considered done until it has actually been put into use or operationalized.

Focused on Execution Excellence

Execution excellence means making time to do something right so that it does not cause problems later. Lean promotes the concept of building integrity into a business process, so that the process is self-correcting and does not require manual intervention. This frees the organization to focus its resources on new capabilities. Quality is extremely important in things that will endure: business processes, software, and any artifact that will be used repeatedly. Continuous delivery is enabled by software development processes that promote getting things right continuously, rather than catching errors later. Quality promotes repeatability. Automation also promotes repeatability, and therefore promotes quality.

Welcome Change throughout the Development Process

The strategic goal of responding quickly to mission needs requires allowing for changes to requirements at any time rather than creating rigid plans. Change is disruptive, and it is prudent to think ahead to minimize change; but responding to change is also essential in being responsive to the changing needs of the business, rather than plying ahead with an obsolete plan.

It is not easy to allow for change. It often means that there is work that must be discarded, or that many decisions must be re-visited. But change also means learning. If software is built and the users realize that it needs to change, then the positive view is that the users now have a clearer idea of what is really needed. Agile proposes that expecting users to know exactly what they need before they see it is not realistic. The strategies promoted by the agile and lean communities are to (a) allow for change, and (b) to manage risk by releasing small increments at a time, thereby containing change to those small increments. Lean also promotes the concept of making decisions at the last responsible moment; that is, to postpone a decision until it can be postponed no longer.

Importance of Security and Reliability

HUD is the steward of our nation’s Housing data that is both critical and sensitive. It also provides services that are used by the public, and the public's expectations for reliability and security are high. These services affect people's lives in a dramatic way, and thus the organization cannot afford to have these services be fragile, risky, unreliable or insecure. As mentioned earlier, lean thinking promotes the concept of building integrity into a process or system, rather than creating a substandard process and cleaning up later. For software development, this means building security into an application from day one. It means testing for security throughout the development cycle, not at the end. It means testing for error cases in the same way: continuously, rather than as an afterthought.

Informed by Architecture

Architecture consists of a set of decisions and models about how things should work. As such, architecture informs decision-making at all levels: at a business level (business architecture), and at a technical level (technical architecture). Architecture can be very ineffective and burdensome if it is allowed to operate in a manner that is very disconnected with operations and projects.

What is needed is effective architecture that is solution oriented and that operates in an issue-focused manner, helping the organization to find the right path as needs are considered and implemented. This type of architecture is deeply immersed in solution efforts, and only spends its time on issues that have an actual impact on the business. Most importantly, it operates collaboratively, working with development teams to help them to consider issues and make decisions and choices about technologies and solution approaches. In other words, it shares its knowledge and helps people, rather than focusing its work around documents.

# Appendix F: Acronym and Definitions

This subsection describes all terms and abbreviations used in support of the development of this document and critical to the comprehension of its content that are not contained in the Office of the Chief Information Officer (OCIO) Terms, Acronym and Definitions List.

Below is a list of Acronyms and Definitions contained in this document:

| Acronym | Definition |
| --- | --- |
| AARTS | Approval/Recertification/Review Tracking System |
| API | Application Program Interface |
| APPS | Active Partners Performance System |
| CARS | Common Application Relational Schema |
| CHUMS | Computerized Homes Underwriting Management System |
| CISO | Chief Information Security Officer |
| COTS | Commercial Off-the-Shelf |
| DTaaS | Development and Test as a Service |
| EA | Enterprise Architecture |
| EAI | Enterprise Application Integration |
| EDW | Enterprise Data Warehouse |
| ESB | Enterprise Service Bus |
| ETA | Enterprise Technical Architecture |
| FHA | Federal Housing Administration |
| HECM | Home Equity Conversion mortgages |
| HOC | Home Ownership Center |
| HUD | The Department of Housing and Urban Development |
| JMS | Java Messaging Service |
| LEAP | Lender Electronic Assessment Portal |
| XYZ | Example Review System |
| OCIO | Office of the Chief Information Officer |
| OIG | Office of Inspector General |
| PII | Personally Identifiable Information |
| POJO | Plain Old Java Object |
| PPM | Project Portfolio Management |
| SFHEDW | Single Family Housing Enterprise Data Warehouse |
| SLAP | Service Layered Architecture Profile |
| SOA | Service Oriented Architecture |
| UML | Unified Modeling Language |
| VA (Examples) | Veteran Affairs |
| VM | Virtual Machine |