Perils of Legacy Design Description in an Increasing Agile World

Ray Miller, Solution Design Quality and Assurance SME
NTT DATA Federal
Ray.Miller@nttdata.com | Ray.Miller@jhu.edu

Abstract

Producing design documentation is a fact of life for systems and software development. But why is this documentation generated? And how much documentation is enough? If you query project managers, the response is likely to be contractual/regulatory requirements to the former and a verbose 20th century template to the latter. Ask software developers the same questions and you'll likely elicit a litany of tirades not suited for publication. There must be a better reason to allocate significant resources to something that is often a source of aggravation and typically relegated to shelfware.

The United States Citizenship and Immigration Services (USCIS), a Department of Homeland Security component at the forefront of the Agile and DevOps movement in the federal space, has embarked on a concerted effort to address these and other issues impacting post-legacy SDLC environments. The agency is committed to employing Lean Architecture, Lean Software Engineering and Lean Manufacturing techniques across the enterprise to ensure that maximum benefit is derived from its Agile and DevOps investments. This commitment includes a "Three Bears" approach to system and software documentation: not too little, not too much, but just enough.

This paper describes a broad Voice-of-the-Customer effort conducted by USCIS that resulted in an innovative paradigm shift to documenting software design; one grounded in Lean principles and Systems Thinking concepts that exceeded the expectations of most ardent Agile practitioners while satisfying applicable regulations.

Biography

Ray Miller is a Lean practitioner with significant and specialized experience supporting business and technical leadership to ensure that IT products and services are efficiently designed to meet or exceed consumer expectations. Ray's versatile engagement history ranges from boutique firms to a Big-4 consultancy supporting military, federal and state agencies in 1) formulating architecture/engineering quality strategies and measures across the enterprise, 2) facilitating adoption of Lean Manufacturing techniques in DevOps environments, and 3) mentoring management and development teams in Lean Architecture and Lean Software Engineering best-practices.

In addition to quality (CMQ/OE), information security (CISM) and project (PMP) management proficiencies, Ray possesses practitioner-level expertise in solution architecture (TOGAF), software quality engineering (CSQE) and quality auditing (CQA). Ray is a contributing-member of the American Society for Quality (ASQ), the Information Systems Audit and Control Association (ISACA) and the Project Management Institute (PMI).

A featured speaker on emergent design quality and assurance topics and a recognized instructor in software quality engineering best-practices, Ray's previous venues have included the Washington Press Club, DC Metro (WMATA) Headquarters and the International Software Quality Conference. Lastly, Ray is currently pursuing a graduate degree in a recently established Quality Management concentration at Johns Hopkins University.

1. Introduction

Software development methodologies and approaches have evolved significantly over the past decade. This is clearly evident in the rapid adoption of Agile and most recently, Lean techniques within United States Citizenship and Immigration Services (USCIS) and the commercial software development industry as a whole.

However, has essential documentation kept pace with this SDLC revolution? In the project management realm, we've seen the legacy Project Management Plan supplanted by a minimalist Project Oversight Plan to accommodate Lean/Agile methodologies. But what about the ubiquitous System Design Document (SDD)? Within the USCIS Delivery Assurance Transformation Group, the SDD is considered one of four essential artifacts in documenting a *software-based* technology solution:

1. System Design Document
2. User Guide
3. Operations Guide
4. Test Plan/Test Cases

2. SDD Voice-of-the-Customer

Based on the frustration voiced by Agile/Lean project teams during artifact reviews coupled with a corresponding escalation in SDD rework, an appraisal of the current (Legacy) SDD Template was deemed a priority by the USCIS CIO as well as Delivery Assurance Branch (DAB) leadership.

To address senior leadership's concerns, the DAB promptly formed a design team to conduct a broad Voice-of-the-Customer (VoC) effort to ascertain the Legacy SDD Template's *fitness for intended purpose or use* in modern software development environments. The VoC effort concentrated on three areas:

1. SDD Content
2. SDD Structure
3. SDD Platform

It is important to note that SDD process (who, what, when) and tooling (how) were intentionally excluded from this VoC effort as both are the purview of other USCIS branches.

2.1 VoC Findings

The Cause-and-Effect (fishbone) diagram **Figure 1** below depicts the findings of this comprehensive VoC effort:

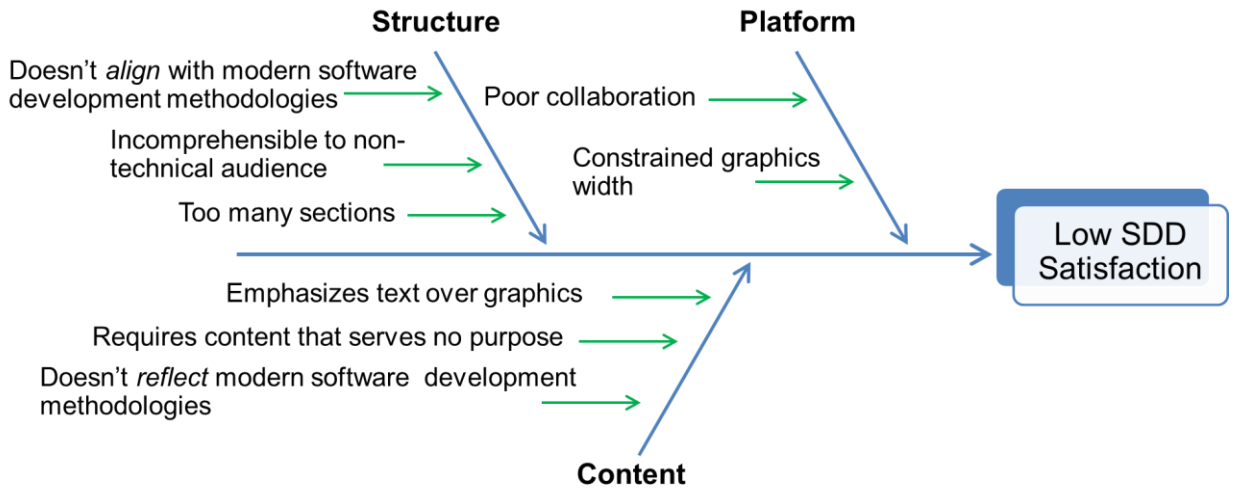


Figure 1: Legacy VoC findings

2.2 Key VoC Takeaways

Based on the in-depth analysis of VoC data collection artifacts, the Design Team surmised four principal takeaways:

2.2.1 The Legacy SDD Template does not embrace Agile values

The Agile Manifesto is an important milestone in software development methodologies. Of the four Agile Manifesto values, "Working software over comprehensive documentation" is most applicable to SDD development. Unfortunately, the Legacy SDD Template runs counter to this value statement as it employs a *kitchen sink* approach; a tailor-down strategy incorrectly assuming that its deeply layered structure of verbose sections subsections will accommodate any design description effort.

2.2.2 The Legacy SDD Template does not apply Lean principles

While the concept of Lean Software Engineering is fairly new to USCIS and commercial software development as a whole, its adoption is rate quickly increasing within mature and disciplined software development teams. Here too, the Legacy SDD Template does not embrace the seven Lean principles with particular emphasis on the first (eliminate waste) and the last (see the whole). Quite the contrary, the content expectation of the Legacy SDD Template is laden with waste and the structure does well to obscure "see the whole."

2.2.3 The Legacy SDD Template is monolithic

There are three generally-accepted methods to content inclusion in a design description artifact:

1. By Reference – Content is incorporated by URL, full path statement, or physical location identifier.
2. By Attachment – Content is incorporated to the rear of the artifact as one or more appendices.
3. By Embed – Content is incorporated into the appropriate main artifact section.

The structure of the Legacy SDD Template strongly encourages attachment or embeds, resulting in a populated artifact that has ranged to thousands of pages in size. Such resulting work product is promptly relegated to shelfware as few modern software developers would consider the artifact as having any practical value. Add to this, the difficulty of continually updating a document of such behemoth proportions and the result is an exercise in futility.

2.2.4 The Legacy SDD Template incorporates hardware into a software description

Modern solution architectures are partitioned into three domains or "layers":

1. Business – A description of the structure and interaction between the business processes and information needs.
2. Application – A *software*-engineered *capability*, described through viewpoints and *independent of any infrastructure*, that supports business process and information needs.
3. Infrastructure – A description of the structure and interconnection of compute and storage resources *at the virtual or physical device level*, provisioned to support the *resource needs* articulated in one or more Application domains.

The scope of any 21st century SDD should be solely the *application* layer. In the case of the Legacy SDD Template, it readily incorporates Infrastructure elements into the application domain. The result is a tightly bound hardware/software solution that fails to take into account the following reality:

> *In an era of cloud computing, modern solution architectures at the application layer must be described as* ***independent of virtual and physical devices****.*

Lastly, enterprise architecture purists may question the absence of the data architecture layer [1] within the above domain model. In DAB solution architecture: 1) Information architecture is incorporated into the business domain, and 2) Data architecture is incorporated into the application domain.

3. Ramifications of Legacy SDD Template Continued Use

The Design Team evaluated the Legacy SDD Template utilizing *fitness for intended purpose or use* as the measurement criteria. In the case of USCIS, the principal purpose and use of an SDD is threefold:

1. A contract document utilized in procurement of software development and maintenance services;
2. An essential onboarding artifact for team members new to the program or project;
3. A documented reflection of the system's current design state at a specific release cycle.

The Legacy SDD Template fails these requisite expectations on all counts; when used to describe a modern software-engineered *capability*, the Legacy SDD Template's predilection to hardware, antiquated structure, and extraneous content all combine to convolute any rational comprehension of system structure and behavior. To paraphrase a USCIS Manager, *to understand the (legacy) SDD, you must first understand the system,* which is analogous to putting the cart before the horse.

4. Advent of the Lean SDD

Based on the aforementioned VoC analysis and intended purpose/scope of a USCIS SDD, the DAB Manager directed the Design Team to *effect sufficient SDD change* which would result in a product that meets *consumer* needs and expectations while satisfying applicable regulations and ancillary stakeholders. Acting on this direction, the Design Team approached the SDD solution as one that would:

1. Adhere to Agile values and Lean principals,
2. Seamlessly integrate into Agile and Lean environments,
3. Remain backward compatible with legacy environments,
4. Require no specialized tooling to populate, and
5. Be *readily comprehendible* by management, practitioners, and ancillary stakeholders.

As a result of the discussion of the aforementioned five guiding principles, a consensus among the Design Team was reached to follow a Lean Engineering approach that would emphasize “eliminate the waste” and “see the whole” principles *in the context of an SDD*. This approach would satisfy the architectural description expectations of the Scaled Agile Framework [2] while remaining in compliance with DHS SDD content expectations.

4.1 Eliminate the Waste

From the “eliminate the waste” perspective, the Design Team resolved to model on *sufficiency criteria*, more commonly known as a “good enough” approach.

To “eliminate the waste,” the Design Team started with the SDD's largest areas of concern and worked until the expected benefit no longer warranted the effort expended (benefit/cost ratio). The four-step process is described in **Figure 2** below:

Figure 2: SDD ‘Eliminate the waste’ process

4.1.1 Properly scope the SDD

The first step was to properly *scope* the SDD as a solution architecture *application* domain artifact void of virtual or physical *device* references. The Design Team accomplished this by 1) *refining* the definition of the application domain, and 2) *unbinding* the application domain from the infrastructure domain while retaining a *loose-coupling*. The textual additions below in bold describe how this was accomplished:

1. Business – A description of the structure and interaction between the business processes and information needs.
2. Application – A software-engineered capability, described through **context, composition, structure, interaction, and information viewpoints**, independent of any infrastructure, that supports business process and information needs.
3. Infrastructure – A description of the structure and interconnection of compute and storage resources at the virtual or physical device level, provisioned to support the **processing node and execution environment** needs articulated in one or more Application domains.

By eliminating the Infrastructure layer from the SDD and enhancing the application and infrastructure domain definitions, the Design Team:

1. Removed extraneous content that provided no value to software development teams, and
2. Facilitated the provisioning of staging and production environments irrespective of platform (physical/virtual/cloud)

4.1.2 Establish the SDD as a control document

The next step was to jettison *redundant* content (Lean practitioners recognize this form of waste as *overprocessing*) from the SDD. **Figure 3** below depicts the SDD’s relationship to external documentation and the description of content these external artifacts provide *by reference* to the Lean SDD:

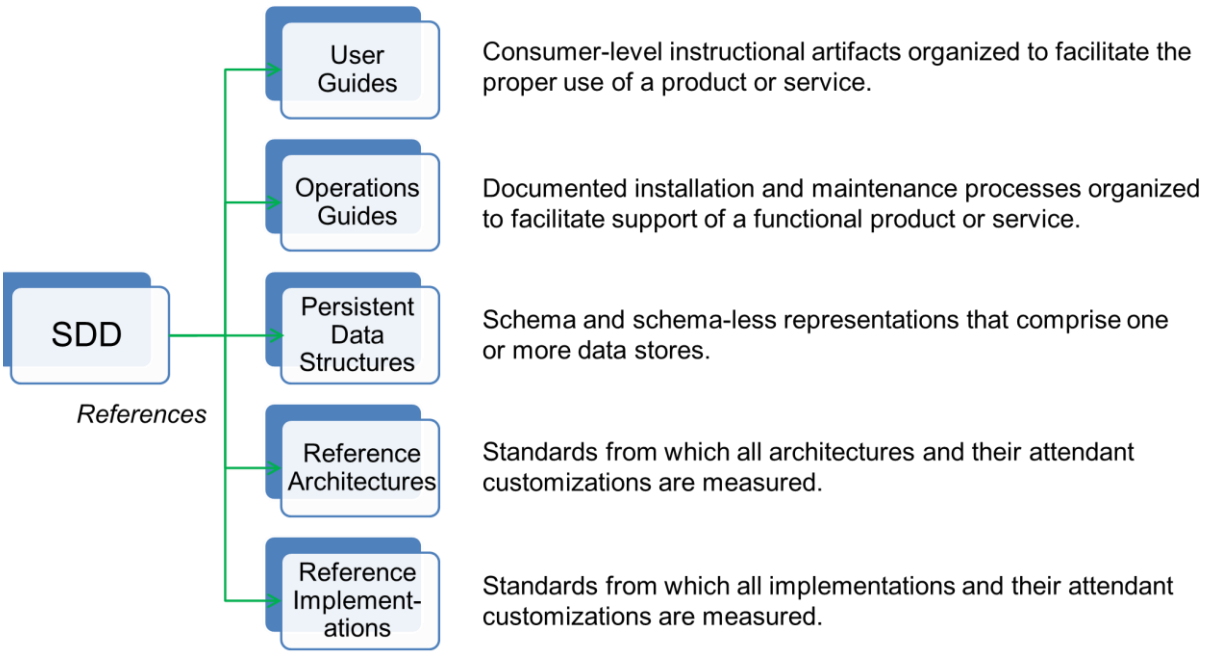


Figure 3: SDD as a control document

By eliminating redundant content from the SDD the Design Team:

1. Removed content that provided no value to the software development team, and
2. Identified the authoritative content sources.

4.1.3 Assemble the SDD as a product suite

The next step was to further refine redundant content elimination. **Figure 4** below depicts and describes the artifacts that comprise the SDD *product suite*:

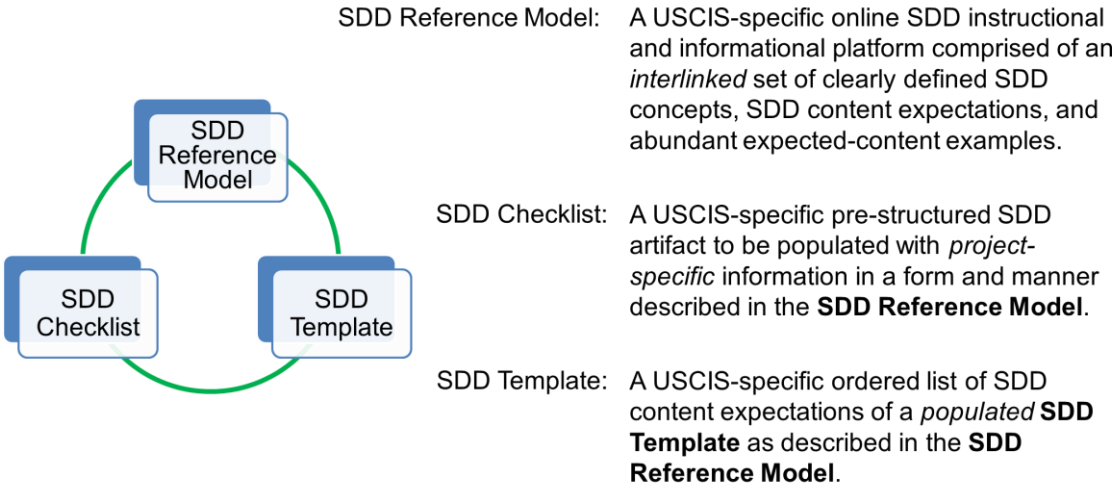


Figure 4: Artifacts comprising the Lean SDD

By relocating instructional content from the SDD Template to an SDD Reference Model, the Design Team was able to:

1. Develop a highly effective instructional platform, free from concern of overwhelming the SDD Template, and

2. Eliminate the mass-deletion of instructional content subsequent to populating the SDD template.

4.1.4 Adopt a tailor-up approach to the SDD

The final step in eliminating waste is to implement a tailor-up approach. In utilizing this method, project teams *build-up* the Lean SDD from a baseline of content, which is *always* required. Thus, content that provides no value is not initially in the SDD Template and added content that provides no value or is redundant is more easily detected.

4.2 See the Whole

From the "see the whole" perspective the Design Team adopted a synergistic approach: *Systems Thinking* and graphics-first.

4.2.1 Systems Thinking

As an essential concept in "see the whole", the two complementary definitions of Systems Thinking [3] adopted by the Design Team were:

1. "A discipline for seeing wholes ... a framework for seeing interrelationships rather than things ... a process of discovery and diagnosis."

2. "The art of simplifying complexity. It is about seeing through chaos, managing interdependency, and understanding choice."

As the Design Team quickly deduced, in embracing a "see the whole" concept, *what constituted a system* within USCIS was problematic:

> *For reasons unknown to the Design Team, USCIS software project teams had adopted the same system boundaries as the System Owner, that is,* ***the functional/technical boundaries of a system were synonymous with boundaries of system funding****. These artificial borders served to contravene the most fundamental of systems science tenets, effectively concealing true system boundaries which are of paramount importance in "see the whole."*

To rectify this situation, three definitions relating to systems were established which would address *true* functional/technical boundaries without infringing on the system boundary definition adopted by USCIS System Owners:

1. System-of-Interest – The top-level system in the system structure, containing **subsystems** and user groups/classes that **directly** interact with the **principal subsystem**. *Note that the subsystems may be systems in their own right and may exist within their own environment.*

2. Subsystem – A collection of people, processes and technologies organized to accomplish a specific function or set of functions **within** a **system-of-interest**. *Note that while a subsystem may be a formally declared USCIS system, it is still considered a subsystem in the Lean SDD.*

3. Principal Subsystem – A **subsystem** within the **system-of-interest** that is decomposed into constituent parts and aspects. *Note that the boundaries of a principal subsystem are synonymous with the boundaries recognized by USCIS Systems Owners (system funding).*

The importance of these definitions in describing true system depth and breadth will become apparent in section *4.3.1 Define and organize SDD viewpoints*.

4.2.2 Graphics-first

The axiom “A picture is worth a thousand words” is applicable when describing a solution architecture *application* domain. Research has proven time-and-time-again that complex ideas can be conveyed with a single still image, making it possible to absorb large amounts of data quickly. To exploit this fundamental human trait, the Design Team adopted a graphics-first approach where *for each Design View*:

1. A *single sentence* states what the view intends to convey,
2. Followed by the primary view graphic,
3. Followed by *textual elaboration* of the view graphic.

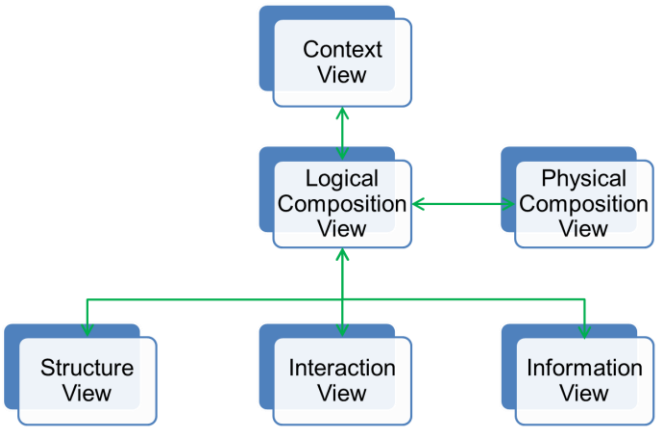
4.3 ‘See the Whole’ Process

In “see the whole”, the Design Team started with the SDD's largest areas of concern and worked until the expected benefit no longer warranted the effort expended (benefit/cost ratio). The four-step process adopted by the Design Team is described in **Figure 5** below:

Figure 5: SDD ‘See the whole’ process

4.3.1 Define and organize SDD viewpoints

Modern design descriptions utilize a viewpoint/view [4] paradigm. While this model can lead to extremes (DoDAF [5] and IEEE-1016 [6]), the Design Team's adherence to Agile values and Lean principles ruled out anything nonessential. Working with project teams, the Design Team settled on the following six views identified in **Error! Reference source not found.** below that would sufficiently describe any solution architecture *application* domain within USCIS:



View	Description
Context	A *top-level* “black box” perspective and scope of the design subject on the **system-of-interest**.
Logical Composition	A high-level “white box” perspective of the **principal subsystem** on the **system-of-interest**.
Physical Composition	The *provisioning* of the **principal subsystem** within the **system-of-interest**.
Structure	**Principal subsystem** static structures and transient data aspects.
Interaction	**Principal subsystem** human-machine and machine-machine behavior and interplay aspects.
Information	**Principal subsystem** persistent data aspects.

Figure 6: Application Domain view hierarchy

4.3.2 Distinguish SDD architecture from SDD engineering

Modern design descriptions differentiate architecture from engineering; a particularly important distinction for those at USCIS implementing Scrum and Scrum-ban [7] flavors of Agile. **Error! Reference source not found.** below depicts the delineation and description of architectural and engineering views:

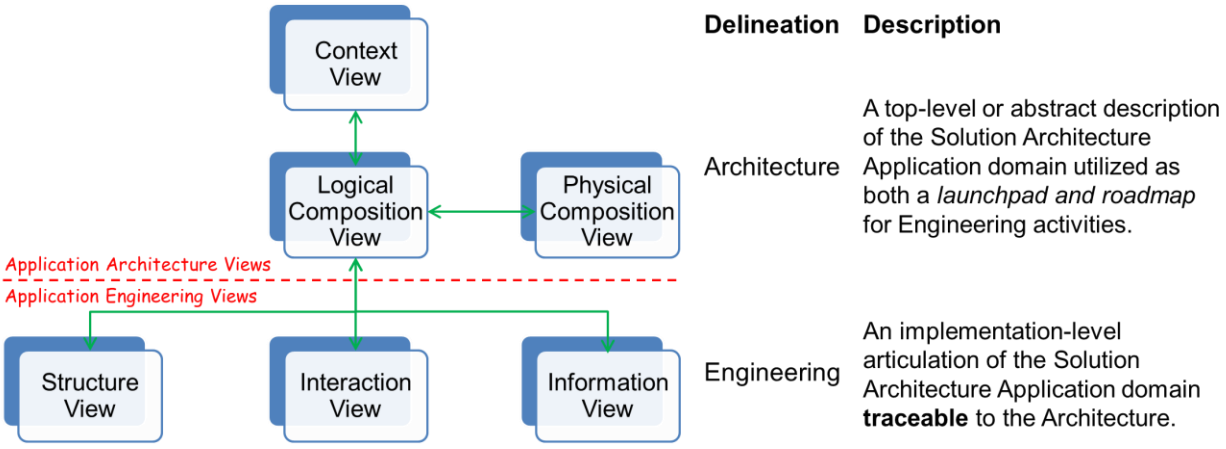


Figure 7: Application Domain view delineation

This delineation between architecture and engineering views bodes well with the Scaled Agile Framework as:

1. Architecture is an iteration (sprint) 0 initiated activity within each increment (release cycle), and
2. Engineering is an iteration 1..n activity within each increment.

4.3.3 Establish SDD symbology

While the Design Team has satisfied three of the five SDD solution guiding principles:

1. Adhere to Agile values and Lean principals;
2. Seamlessly integrate into Agile and Lean environment;
3. Remain backward compatible with legacy environments.

Two guiding principles had yet to be addressed:

4. Require no special tooling to develop the SDD;
5. Be *readily comprehendible* by management, practitioners, and ancillary stakeholders.

These principles were easily satisfied through the adoption of block diagramming [8] techniques. Utilizing a *box-and-line* notation allows stakeholders not proficient in architecture domain symbology to easily articulate and comprehend a design description. The addition of requisite block-*stereotyping* facilitates a consistent understanding of each element across every view. Finally, block diagramming requires no specialized drawing software; all example diagrams contained in the Lean SDD Reference Model views were developed utilizing Microsoft PowerPoint.

4.3.4 Format and publish SDD artifacts

Finally, being *readily comprehendible* in “see the whole” can be a difficult proposition when the form-factor is restricted to 8½ x 11 inches and the platform is a Microsoft Word document. Here too, the Design

Team elected to abandon this 20th century relic and instead format and deliver Lean SDD artifacts utilizing the web page capabilities of USCIS ECN (SharePoint) platform. The advantages of this platform change were fourfold:

1. Project teams can review the Lean SDD Reference Model, template, and checklist *at any time utilizing any device* connected to the USCIS Intranet.

2. Project teams can copy and paste the XHTML-validated Lean SDD template and checklist from the SDD SharePoint site to the project team's SharePoint site *without content or formatting loss*.

3. Project teams can develop diagrams and populate the Lean SDD template *unfettered by the 8½ x 11 inch form-factor*.

4. Project teams can archive an approved Lean SDD from their respective SharePoint sites to the appropriate Information Technology Document Library (ITDL) utilizing the MHTML [9] file format supported by Internet Explorer, Chrome, Firefox, and Opera web browsers.

5. Benefits of Utilizing the Lean SDD

By adhering to Agile values and Lean principles, the Design Team was able to successfully address the principle purpose and use of a USCIS SDD:

1. A contract document utilized in procurement of software development and maintenance services;

2. An essential onboarding artifact for team members new to the program or project;

3. A documented reflection of the system's current design state at a specific release cycle.

The Lean SDD fulfills these requisite expectations on all counts; when used to describe a modern *software*-engineered capability, the Lean SDD's application-domain scoping, systems-thinking approach, graphics-first content, and exponential reduction in volume all combine to advance the comprehension of system structure and behavior at a rate previously thought unattainable. When deployed on a modern content management system, the result is a collaboratively-developed SDD that will seamlessly integrate into any Agile, Lean, or legacy environment.

6. Lean SDD Implementation

The Design Team believes that *documentation is a part of software development, not a separate activity*. This runs counter to how the project teams currently generate design documentation. As change is always a difficult proposition, especially in demanding environments with high expectations, the Design Team collaborated with multiple project teams of varying size to determine the best course of action. The Design Team and project teams jointly concluded that a wholesale transition to the Lean SDD was not a practical solution due to project team workloads and tight delivery schedules. Instead, "evolution, not revolution" would rule the day.

The Design Team, in concert with the project teams, ascertained that this approach would entail a two-step process:

6.1 Identify Minimal Lean SDD Views

Upon review of the three principal intents of an SDD previously articulated in section three *Ramifications of Legacy SDD Template Continued Use*, and section five *Benefits of Utilizing the Lean SDD*, the Design Team focused on intent number two: "An essential onboarding artifact for team members new to the program or project." The Design Team, in collaboration with the project teams, determined that the three views identified in **Figure 8** below are considered imperative for new team member *initial* understanding of application structure and behavior:

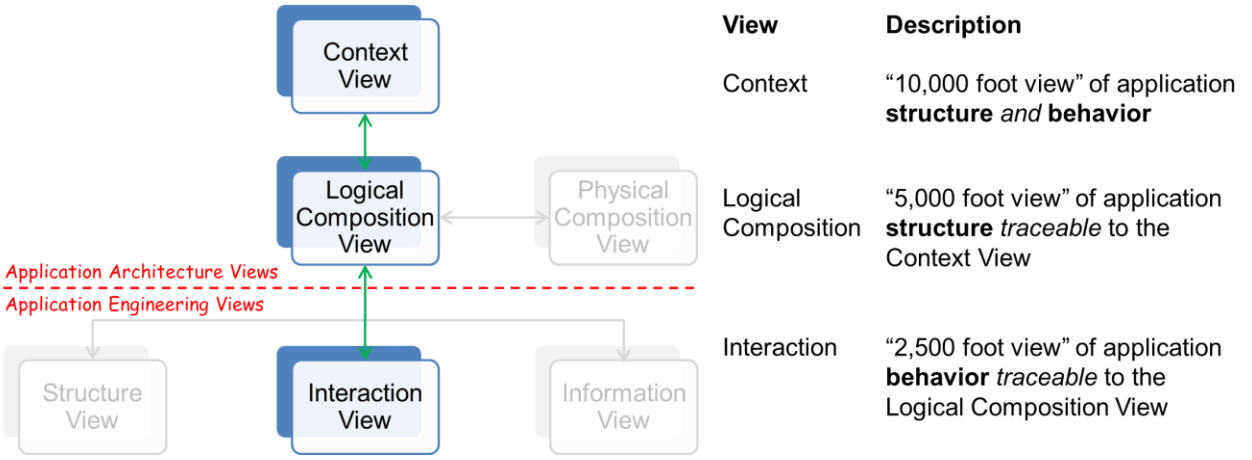


Figure 8: Application Domain imperative views

The three remaining views were *initially* omitted for the following reasons:

1. Some information provided by the Physical Composition View can be abstracted from Infrastructure Architecture diagrams.

2. Some information provided by the Structure View can be abstracted from existing UML Class and Package diagrams.

3. Some information provided by the Information View can be abstracted from existing Entity Relationship Diagrams (ERDs).

6.2 Phase-in Lean SDD Adoption

Active USCIS systems are extensive in number and range from the very small to the extremely large. Once again, the Design Team collaborated with the project teams to integrate Lean SDD development *without impeding software development*. As USCIS release increment velocity typically ranges from two to eight weeks for all projects, two adoption approaches would be utilized based on project size and complexity:

1. Small or medium size projects of normal complexity – Context, Logical Composition, and Interaction Views are to be completed *prior to the release of an increment*. The Physical Composition, Structure and Information Views would be completed prior to the subsequent release increment.

2. Large or extremely complex projects – One view is to be completed and incorporated into the Legacy SDD (supplanting any duplicate content) *per release increment*. At the end of six increments, the Lean SDD will completely supplant the Legacy SDD.

The above approaches allow the project teams to incorporate the Lean SDD into the software development cycle without disrupting iteration velocity and subsequent increment releases.

7. Conclusion

When the Design Team embarked on this SDD *fitness-for-use* endeavor, it did so with a collective open mind. The Design Team correctly reasoned that any attempt to modernize the Legacy SDD Template would be tantamount to *putting fenders on a carriage*. A new approach was clearly needed, one that would *complement* 21st century software development methodology and techniques.

The new approach adopted by the Design Team, *one grounded in Lean principles and Systems Thinking concepts*, ably achieves the scope and purpose expectations of a modern SDD while adhering to the five guiding principles established earlier in this paper.

USCIS project team feedback of the Lean SDD product suite (reference model, template, and checklist) is overwhelmingly positive; to paraphrase agency adopters, *the (lean) SDD conveys system structure, function, and intent in a manner that is immediately understandable to those with little or no system knowledge*, which was the DAB's mission from the very beginning.

References

[1] The Open Group, "TOGAF 9.1 > Part II: Architecture Development Method (ADM) > Phase C: Information Systems Architectures - Data Architecture," [Online]. Available: http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap10.html. [Accessed 19 July 2015].

[2] Scaled Agile, Inc., "Scaled Agile Framework (SAFe)," [Online]. Available: http://scaledagileframework.com/. [Accessed 19 July 2015].

[3] Systems Engineering Body of Knowledge (SEBoK), "What is Systems Thinking?," [Online]. Available: http://sebokwiki.org/wiki/What_is_Systems_Thinking%3F. [Accessed 19 July 2015].

[4] The Open Group, "TOGAF 9.1 > Part IV: Architecture Content Framework > Architectural Artifacts > Views and Viewpoints," [Online]. Available: http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap35.html#tag_35_04. [Accessed 19 July 2015].

[5] U.S. Department of Defense, "DoDAF 2.0 Architectural Models-Views and Descriptions," [Online]. Available: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_2-0-Arch_Models-Views_Descriptions.docx. [Accessed 19 July 2015].

[6] IEEE Standards Association, "1016-2009 - IEEE Standard for Information Technology--Systems Design--Software Design Descriptions," [Online]. Available: http://standards.ieee.org/findstds/standard/1016-2009.html. [Accessed 19 July 2015].

[7] C. Ladas, "Scrum-ban," [Online]. Available: http://leansoftwareengineering.com/ksse/scrum-ban/. [Accessed 19 July 2015].

[8] A. Kossiakoff, W. N. Sweet, S. Seymour and S. M. Biemer, Systems Engineering Principles and Practice, 2nd Edition, John Wiley, 2011.

[9] FileInfo.com, ".MHT File Extension," [Online]. Available: http://fileinfo.com/extension/mht. [Accessed 19 July 2015].