Data-centric System Development Life Cycle for Automated Clinical Data Development System

Kevin Lee, MarkLogic, Washington D.C.

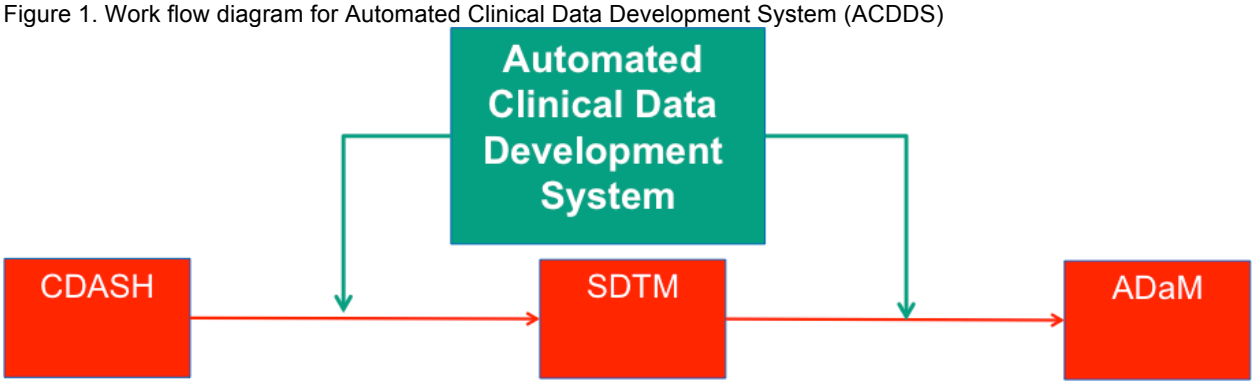
ABSTRACT

Many life science organizations have been building systems to automate clinical data development (e.g., SDTM and ADaM). And such systems are considered as IT product and goes through typical system development life cycle (SDLC); requirements, analysis, design, programming, test and implementation. However, SDLC was initially developed for systems that automate the business process, not the data development. So, the question naturally arises that if life science organizations develop systems to automate data development, should the systems still be developed in process-centric SDLC? or will the current process-centric SDLC satisfy the business need? The presentation will introduce data-centric SDLC. First, the presentation will discuss how some steps of typical process-centric SDLC should be modified and adjusted in data-centric SDLC. For example, test of system requires target data quality assurance. And due to unpredictability of source data, maintenance and system update will be required after implementation. Secondly, the presentation will introduce additional steps and approaches for data-centric system development process such as data profiling and compliance.

EMERGENCE OF AUTOMATED CLINICAL DATA DEVELOPMENT SYSTEM (ACDDS)

The emergence of CDISC standards helps the organization to think about the innovative ways to develop clinical trial data. CDISC standards such as SEND, SDTM and ADaM follows the specific data model and data structures, so if programmers follow CDISC standards to build clinical trial data, the structures of clinical trial data would be consistent across studies, departments, organizations and etc. This opens the door for building the automated clinical data development system, so this system/application could be reused to develop the same data structures again and again.

Figure 1. Work flow diagram for Automated Clinical Data Development System (ACDDS)



INTRODUCTION OF SDLC

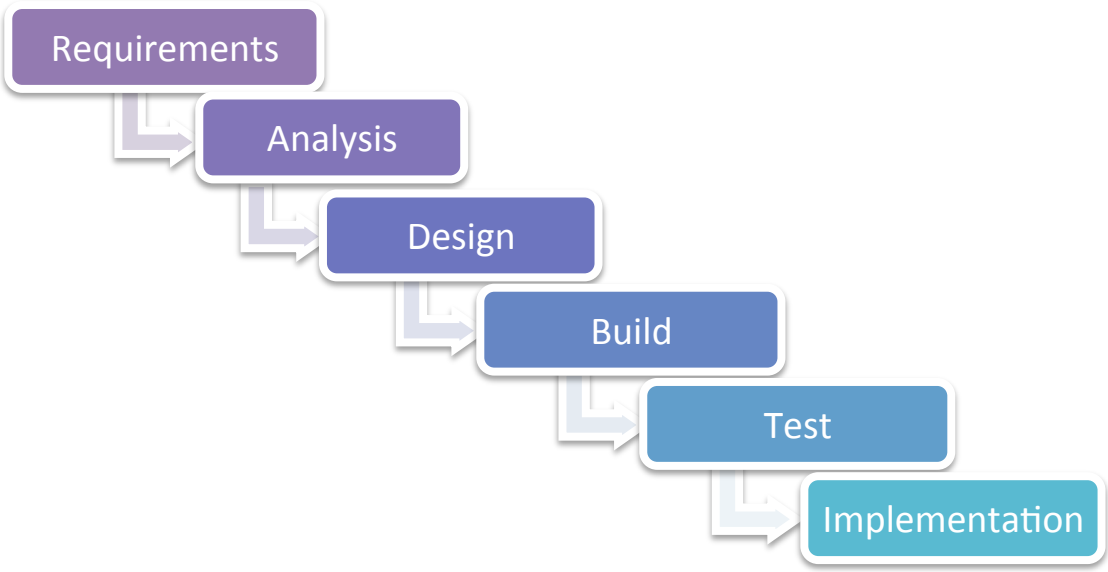
A systems development life cycle (SDLC), also referred to as application development life cycle, is a process of developing an information system such as ACDDS. It is usually composed of a number of clearly defined and distinct work phases which are used by systems developers to plan for, design, build, test, and deliver systems. SDLC usually follow the following steps:

1. Requirements
    a. Users Requirement- This phase will find out what users ultimately need from system.
    b. Functional Requirement- This phase will find out what functions the system needs to do.
2. Analysis- This phase will understand the current state and how it relates to the requirements. It will identify the gap between current state and requirements.
3. Design- This phase will create the future state and design the system according to requirements.
4. Build- The phase will develop the new system according to requirements and design.
5. Test: The phase will consist of several tests to ensure the new system to satisfy the requirements. The following tests that the new system might be tested on.

a. Unit Test - A unit is the smallest entity which can independently exist, e.g. dm.sas. Unit Test verifies that the smallest entity can function correctly when isolated from the rest of the codes/units.
b. Integration Test – Integration Test verifies that units created and tested independently can coexist and communicate among themselves, e.g., ext-demo-from-edc.sas and dm.sas.
c. System Test - Unlike Unit and Integration Test, System Test is usually planned by client's business team. System Test ensures that expectations from system developed are met according to functional requirements. The whole system is tested for its functionality, interdependency and communication. Load and performance testing, stress testing, regression testing, etc., are subsets of system tests.
d. User Acceptance Test (UAT) - UAT is planned by client's business team and usually done by client's users. UAT is performed in a user environment that resembles the production environment, using realistic data. UAT verifies that delivered system meets user's requirement and system is ready for use in real time.

6. Implementation – The phase will put the new system into production and turn it over to users to operate.

Figure 2. Introduction of SDLC Waterfall model phases



ISSUES ON CURRENT SDLC FOR ACDDS

When SDLC was developed and implemented back in 1960s, the focus was on automating business processes that had been performed manually. In those early days, the automation was always to be achieved by building custom-developed software. Therefore, the work phases and steps of SDLC mainly focus on processes and workflows, but in clinical data development, data creation should be the focal point.

IT department has been using SDLC for last 5 decades. Even though other variations on SDLC such as agile and V-model have been introduced and used, IT department won't be able to replace the SDLC in any radical way. IT department are used to a typical, process-centric SDLC. The data development system could be new for many of IT departments, so IT department will try to use a process-centric SDLC for ACDDS. But, ACDDS is different from normal system. The clinical trial programming team need to explain the natures and focus of ACDDS and work with IT department.
There are also another questions on who will lead ACDDS development, IT department or Programing department. IT department will look at automated clinical data development system (ACDDS) as one of system that needs to fit in overall IT architecture and need to integrate and operate properly within. The programming department will simply look at ACDDS as a tool for developing clinical data such as SDTM and ADaM. So, there should be understanding and collaboration between two groups.

ACDDS is built to develop data, not to automate the process. ACDDS will receive source data and develop target data while a typical, process-centric system will develop and contain the data in the system. In order to develop a

complete, first-class ACDDS, the designers and developers need to modify SDLC, which is data-centric SDLC.

THE INTRODUCION OF DATA-CENTRIC SDLC

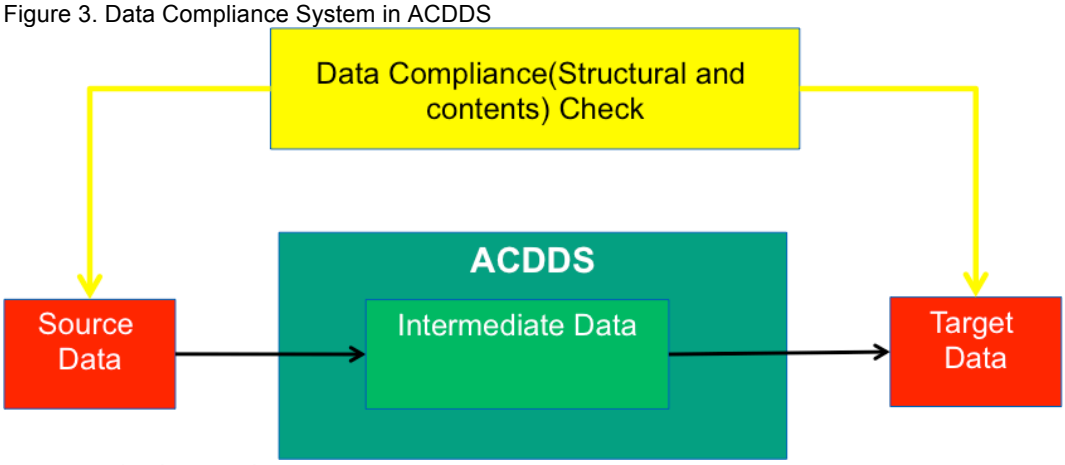
The focus of data-centric SDLC of automated clinical data development system is not to build the system and integrate with other systems. The main focus of data-centric SDLC is to develop the system that will create clinical trial data, which are compliant to CDISC model. There should be some changes on perspective and focus of data-centric SDLC from typical SDLC.

The organizations do not need to give up SDLC, but need to change some perspectives on each phase.

1. Requirement Analysis – A process-centric SDLC focuses on how users want processes to work, but a data-centric SDLC will focus what users want to create from source data and how users want to use them.
    a. Users Requirement - For example, users need to create SDTM DM datasets from demo datasets from EDC (Electronic Data Capture) system.
    b. Functional Requirement - For example, ACDDS will extract demo datasets from EDC system and create SDTM DM datasets without error messages and within 10 seconds.
2. Analysis - This phase will involve understanding source and target data.
3. Design – The design phase of data-centric SDLC needs to include data design as well as system design.
    a. System Design
    b. Data Design
        i. Source data model design – The developers need to develop the data models for source data and terminology of source data.
        ii. Target data model design - The developers need to develop the data models for target data, e.g., SDTM and CT of source data.
        iii. Intermediate data model – The developers need to develop the data models for intermediate data that will be stored inside ACDDS. This model might not be applied to all the clinical trial data development.
    c. Business rules and specification design – The designers, business analysts or developers need to develop business rules, logics and specifications that transform source data to target data. These designs might come as metadata such transformation metadata.
4. Build – If a typical, process centric system is properly built according to requirements, and has been successfully tested, it should function without problem in production However, in ACDDS, there is no real control over the source data. Problems can arise in it at any moment that have nothing to do with ACDDS. So, source data should be monitored constantly. Developers need to build the source data compliance check system before source data are ingested into ACDDS.
5. Test – A data-centric SDLC needs to test not only the typical tests such as Unit Test, Integration Test, System Test and UAT, but also data profiles and compliance test.
    a. Data Profiles – this test will need to check the following
        i. Any data quality issues
        ii. Data contents follow certain terminologies.
    b. Data Compliances – it will test on correct data structures of source and target data.

    Data profiles and compliances tests could be done during SDLC as well as production data development.

Figure 3. Data Compliance System in ACDDS



6. Implementation
7. Post-Development – The developers need to expect that ACDDS will go through post-development due to the following reasons.
    a. New source data – If the sponsors add the new vendor or change EDC system, contents and structures of source data will change. ACDDS should be modified to meet the changes of source data.
    b. Changes in target data model – Due to business needs such as new CDISC data structures and changes on current CDISC data models, ACDDS should be modified to meet the changes of target data.

CONSIDERATION OF ACDDS

ACDDS should be able to handle many kinds of data. Below are data types that ACDDS should take into consideration beside source and target data:

1. Metadata
    a. Source data
    b. Target data
    c. Transformation data
    d. Intermediate data
2. Terminology
    a. Control Terminology
    b. WhoDrug
    c. MedDRA

Since ACDDS will handle data, ACDDS should be built to change. As mentioned in data-centric SDLC, post-development will occur frequently. So, codes and programs in ACDDS should be flexible and easy to change.

The developers also need to consider the situation that ACDDS might not be able to cover all the clinical trial data development. We are trying to build ACDDS to save the time to develop the common, repeatable clinical trial data. If the time and efforts to develop certain data (e.g., extremely complex ADaM datasets) are huge and expensive, those data might not be a good fit for ACDDS.

The data-centric SDLC of ACDDS introduces the data modeling of source and target data in design phase. SDTM have more than 50 data structures. If SDLC follows waterfall model and design all SDTM data structures before build phase, it will take really long time to design all the SDTM data models. So, it is recommended to adopt the agile approaches rather than waterfall model, so developers design one SDTM data model and build programs/codes for the SDTM dataset, then move on to the next SDTM data model.

It is also highly recommended to have governance on data model e.g., CDASH, SDTM and ADaM during SDLC. It is very common to change data models during SDLC, but those changes should be followed and governed so that ACDDS will be also modified according to changes of data models.

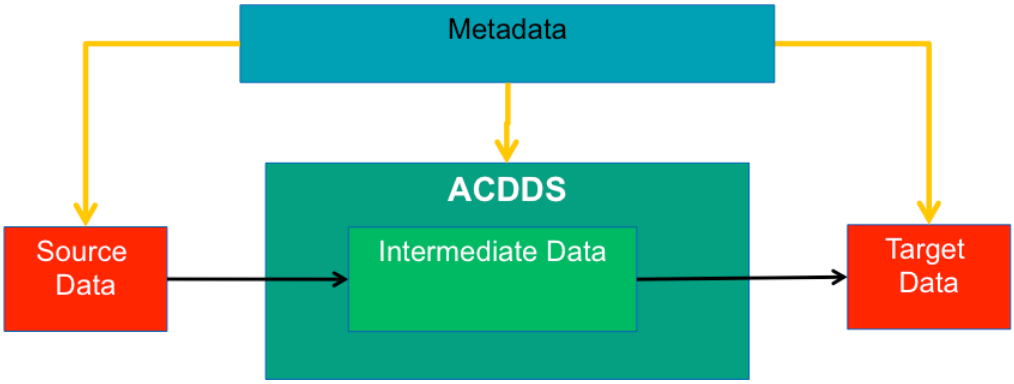
When the developers create codes and programs in build phase, it is recommended that the developers are using

live data rather than test data. The developers are able to see actual issues and concerns with live data and also don't need to spend time to create test data.

METADATA-DRIVEN ACDDS

The clinical trial data models (e.g., SDTM and ADaM) are bound to change and its business rules are also bound to change. Whenever there are any changes on data either in source or target, ACDDS needs to be modified accordingly. However, if ACDDS depends on metadata to develop clinical trial data, the developers might not need to modify codes in ACDDS. Developers or even users can modify metadata according to new changes. ACDDS simply use the new metadata to create data. So, on metadata-driven ACDDS, changes will not happen to codes, but changes are applied only to metadata that ACDDS will consume.

Figure 4. Metadata-driven Automated Clinical Data Development System



CONCLUSION

When organizations build the system that transform source data according to business rules and data specification, they need to focus on a system developing data rather than automating process. A typical, process-centric SDLC might not be a good fit for such ACDDS. ACDDS requires a different, innovative SDLC that can focus on using and developing data, so that system can be designed, developed and tested for automated clinical data development. The paper introduces the data-centric SDLC for that purpose. The data-centric SDLC will be able to help the organizations to design, develop and test the clinical-data-focused automated clinical data development system and put it into the production.

REFERENCES

- System Development Life Cycle in Wikipedia
- V-Model (software development) in Wikipedia

CONTACT INFORMATION

Your comments and questions are valued and welcomed. Please contact the author at

Kevin Lee
Solution architecture and technical sales engineer
MarkLogic
Kevin.lee@marklogic.com