

Written and researched by:
Dr. Kirk Mousley
and Karl Mousley

Software Development Life Cycle

EDC Today is an independent publication on current information and issues in Electronic Clinical Systems (ECS) strategies and technologies for the Biotechnology and Pharmaceutical (Biopharma) industry. Each month we examine topics related to ECS theory, technology, practice, or implementation.

One of the reasons that EDC has been slow in being adopted is that companies have used a pilot project approach, that is, provisionally trying EDC on a small study. Piloting in this manner means that the pilot is not integrated with other applications or systems that already exist in-house. EDC Management believes that the failures to adopt EDC are often due to this lack of integration. Sometimes the lack of system integration is exacerbated by the failure of the project personnel to completely modify their work process to take full advantage of the gains an EDC solution offers.

In this issue, we discuss the Software Development Life Cycle as it applies to system integration, and attempt to explain why a “quick EDC pilot” might fail for reasons other than the actual vendor-supplied EDC solution. While a full-blown software development effort might not be needed, a complete analysis of existing systems and their features along with existing work processes is needed to take full advantage of any EDC solution.

I. Introduction

In the world of clinical trials software, there are several software components that make up a computerized system. First, there is the operating system software, usually a Microsoft Windows or Unix-derivative, installed on a server (i.e., computer). Then there is the clinical trials system software, usually purchased from a vendor and installed on the server. No clinical trials system software can be used right “out of the box” to conduct a clinical trial. Some of the customizations and configurations that are done to prepare the computer system for use with a clinical trial include:

- Operating system extensions (e.g., web server) and third party software such as Oracle RDMS, Brio or Crystal Reports for web.

In the next issue of
EDC Today:

System Migration

About EDC Management:

EDC Management is the leader in Clinical and Data Management and Electronic Data Capture (EDC) consulting services for the biopharmaceutical industry. EDC Management publishes well-researched and timely information about Electronic Data Capture technologies and processes through EDC Today® and EDC In Depth. We do not sell or endorse any specific EDC software application or vendor. Improve process today; position for tomorrow.

EDC Management

P.O. Box 23
Conshohocken, PA 19428
484-530-0300 (voice)
610-567-0357 (fax)
info@edcmanagement.com
www.edcmanagement.com



- Customized installation configurations, which might include the selection and set up of a Relational Database Management System.
- Vendor provided software customizations and/or add-ons.
- User developed customizations, such as custom progress reports.
- Trial specific software including eCRF scripting, edit-checks, and possibly specialized functionalities such as randomization.
- Legacy system integrations and migrations.
- The development, acquisition and loading of data such as Codelists and Dictionaries.

Obviously, with or without EDC, a clinical data management system (CDMS) is a large and complex one. To deal with the size and complexity of larger systems, the expert suggested approach is to follow Information Systems (IS) through a birth to death “software development life cycle” (SDLC).¹

II. The Roots of the Software Development Life Cycle

The development life cycle for software is a time-honored process that seeks to maximize the chances for a project’s successful conclusion. It also strives to maximize efficiency and to reduce effort. The development life cycle has been around for nearly as long as the electronic computer, which came into being just before the start of the Second World War.

“The Atanasoff-Berry Computer was the world's first electronic digital computer. It was built by John Vincent Atanasoff and Clifford Berry at Iowa State University during 1937-42. It incorporated several major innovations in computing including the use of binary arithmetic, regenerative memory, parallel processing, and separation of memory and computing functions.”²

Even after sixty years of attempting to improve development process-specific procedures, nearly “one third of all [software] projects fail. The top 5 causes of failure: Lack of user input, lack of executive support, incomplete requirements and specifications, changing requirements, technology ineptitude.”³

Software projects continue to have a high failure rate even after a number of years and much study involving how to minimize failure. Dozens of methods and approaches for better software development have been developed over the years. Whatever the selected methodology, it will likely have four common threads - analysis, design, programming, and implementation. Programming here may be loosely interpreted to include packaged software and even outsourcing.

Prior to the late 1980s system development was a much more formal process. The introduction of prototyping allowed a more flexible approach toward life cycle development. Most professionals familiar with prototyping would recommend it most effectively for extracting user requirements.⁴ The main problems with prototyping are that the user doesn’t attempt to mock up a design – often they let the developer make the first (most) effort, and that some attributes are not very visible, for instance, data storage definitions (date, number, character types and sizes).

(continued on page 3)



In any event, the goals of the development life cycle are multitudinous. The first and foremost goal is failure prevention. It also serves to facilitate and enhance communication between user and software developer.

III. Software Development Life Cycle in Short

A formal SDLC consists of a number of discreet steps. For the purpose of trying out an EDC solution in a pilot, not all of these steps are necessarily required, but certainly all of the steps should be reviewed for relevancy. The steps involved in the SDLC are: requirements elicitation, analysis, system design, object design, implementation, test, installation and checkout, operation and maintenance, and retirement. A brief discussion of these steps follows.

Requirements Elicitation

Every project begins with an attempt to determine exactly just what is needed. The users of the proposed software, key managers, and developers should meet to define the purpose of the proposed system and describe what that system will do in terms of the business and functional requirements. This description should cover the basic inputs (e.g., data entry) and outputs (e.g., results) of the system. The more complete the description is in elaborating the desired functionality of the system, the more likely the resulting software will meet the needs of the users and management.

Analysis

The next step is for developers to take the description of the business requirements (i.e., functional requirements) and restate them as an object model (i.e., a high level design) that describes the system. This model might be represented as a block diagram that labels inputs, outputs, data transformation, and information storage. The key to this SDLC step is to develop a picture of the distinct objects that make up the system. The object model should be correct, complete, consistent, unambiguous, realistic, and verifiable.

System Design

The developers then take the object model (i.e., analytical results) and produce a more detailed design of the system. Each object of the analysis might be deconstructed into subcomponents that can be realized by individual developers or development teams. There should be a clear description of strategies, subsystem decomposition, and deployment diagram for the system.

Object Design

Using the design document, the developers can then specify the individual custom objects that make up the software system. These objects might include program modules, interfaces, and even off-the-shelf components. The object design will also describe how these components will have to be modified to bridge the gap between the business analysis description and the system as it is defined by the system design. Object design produces a document or set of documents that are detailed at an object level and annotated with constraints and exact descriptions for each element, and are intended to allow the programmer or programmers to create a portion of the system.

(continued on page 4)



Implementation

After all the design work is complete, the developers translate the object models into source code (i.e., program modules). The developers also make sure that any off-the-shelf components are correctly installed and that any data sources are correctly configured. The end result of this SDLC step is a complete set of source code files, along with installation and configuration scripts where appropriate.

Test

After the system is implemented, it should be completely tested by both the developers and the expected users. The developers will test that all code functions according to specification, and the users will test the system functionality, but even more important the users will test to make sure the specifications are in fact appropriate for their needs. At this point, users or their management may realize that they need to update the specification due to unforeseen issues. If this happens, then the SDLC restarts at the first step.

Installation and Checkout

When the system has been tested and found to function according to specifications, it can then be installed into a production environment. Installation will use the scripts that have been developed and tested as part of the implementation step. The result of this SDLC step is that a complete, tested set of software modules is placed in production, and that all change control procedures have been followed. The functionalities of all of the software modules are checked in the production environment. Copies of the new software as well as any software being replaced are archived.

Operation and Maintenance

The users of the software will use the system in the production environment following their operating procedures for as long as the system continues to serve their needs. From time to time, users may require changes to the system or additional functionality. In this case, documented change control procedures are followed. Each step of the software development life cycle is completed for each change or addition starting with requirements elicitation. After the modified software is specified, designed, implemented, and tested, it can be installed into production.

Retirement

At some point in time, the system will no longer meet the needs of the business. This may be because the software components used are no longer supported by their vendors, or perhaps that the business requirements have changed, possibly because of changes in the business. When this happens, the software will be retired. The software may be uninstalled and/or archived. Change control procedures will detail how this retirement is to be performed.

(continued on page 5)



IV. Integration with Legacy Software

Implementing a complete EDC solution with sufficient integration with legacy systems to ensure its successful use in and of itself will likely be a large and complex project. Attempting a “quick pilot” of an EDC solution with the expectation of the pilot being indicative of long term EDC success is a dubious proposition. Using SDLC to guide the assessment of just what business requirements the EDC application is intended to address as well as detailing exactly what is being attempted will help bring to light not only the goals of the EDC pilot but will help quantify the measurements needed to determine the pilot’s degree of success. More specifically, using some of the steps found in the SDLC should:

- 1) Enumerate the business requirements for the overall CDMS application
- 2) Enumerate the business requirements for the EDC application
- 3) Enumerate the business requirements for which the EDC application will replace existing legacy systems
- 4) Enumerate the interfaces between the EDC application and any remaining legacy systems
- 5) Establish a basis for determining how much effort will be involved to integrate the new EDC application with any remaining legacy systems
- 6) Establish a list of business requirements for which the level of fulfillment by the EDC application can be gauged
- 7) Indicate areas where processes will need to be modified in order to take complete advantage of the EDC application (i.e., point out what new SOPs will be needed, as well as which ones will need to be modified)

Obviously, it may not be desirable to execute the complete SDLC approach when undertaking an EDC pilot as it will entail a fair amount of effort, but at minimum, the first two steps, requirements elicitation and analysis, should be done. Some of the remaining steps might be performed on a partial or “outline” basis in order to develop an idea of the scope of the development effort that will be necessary, if the EDC pilot should result in a go ahead for a complete implementation and installation.

V. Failures Due to Non-integration

If the EDC application is not integrated with existing legacy applications, a judgment-call assessment will be necessary when determining the success of the EDC pilot. A determination of the impact that the lack of integration with legacy systems had on the pilot might avoid misconceptions over where the obstacles to smooth EDC operation lie, as well as pinpoint the places where complete integration will or will not be desirable should EDC be adopted.

Another assessment (or judgment) will be necessary when determining whether or not to undertake the effort required to integrate the EDC application, should the pilot demonstrate the desirability to adopt EDC. If the SDLC process has been started, this return on investment assessment should be simpler to make.

(continued on page 6)



VI. Failures Due to Non-updated Human Processes

Likewise, if user processes are not modified to take advantage of the EDC application, a subjective judgment will be necessary to decide if the EDC pilot is a “success”. In many ways, this subjective decision will not be as easy to make as any regarding non-integration. The fact is that some process change will be needed just to use the EDC application. Any assessment will have to be based on comparing the general procedures followed by the personnel involved in the pilot to what might be more optimal procedures for using EDC.

Assessments of this type often break down since the pilot users do not know what the optimal procedures might be. Oftentimes, while running the EDC pilot, those involved are learning how to use the product while evaluating the product. If that is the case, then perhaps an evaluation of the success or failure of the pilot should not be attempted until the learning process is complete, and a better idea is formed as to what the new and revised operating procedures should be. Combining a learning experience with an evaluation experience is almost certain to provide a negative assessment of the product’s capabilities.

It is our opinion that biopharmas should do their homework and learn as much about an EDC application and how to properly use it *before* attempting a pilot. If such preparatory work is done, then the pilot can be a better assessment tool.

VII. Integration Needs and Process Change

The software development life cycle should focus on the business requirements for the overall CDMS, the EDC application, and should recognize the integration needs and required process changes. In establishing the requirements for an EDC pilot, it is important to analyze how the EDC application will integrate with existing software. It may not be necessary as part of the pilot to implement integration, but a gap analysis should be performed to the extent practical. This gap analysis should highlight both the inefficiencies incurred by the EDC pilot due to missing integration and also what effort is needed to design and develop the necessary integration to address those inefficiencies.

Likewise, when documenting the requirements and specifying how users are to interact with the EDC system, a gap analysis of existing processes and the processes the users must follow when using the new system must be made. This gap analysis will point out what new or modified procedures (that may be temporary in nature) will be needed during the EDC pilot. Failure to implement the correct procedures will lead to great difficulties in evaluating how much benefit EDC is providing and what it actually costs. This difficulty will be further exacerbated by the impact of the learning curve on productivity, and the varying degrees in which the users will be able learn the ins-and-outs of the EDC application in the short time the pilot is being run.

(continued on page 7)



VII. Conclusion

SDLC has been an important process in the development of software and computer systems. The main focus of SDLC is to prevent the failure of system building projects. The initial steps consist of requirements gathering and analysis.

The evaluation of an EDC application is something that should be taken seriously, and much thought should go into any attempt to run a pilot that is intended to determine if a particular EDC solution meets business requirements. Pilots cannot be executed in the spare time of current clinical trial professionals and be expected to show meaningful results. Using SDLC when implementing an EDC pilot is one way of putting the necessary effort into the project.

The partial use of SDLC with the emphasis on requirements and analysis for integrating existing legacy systems is preferred over just “running” a pilot. At minimum, describing (i.e., documenting) integration requirements can highlight areas of concern when evaluating the effectiveness of the EDC application.

In addition, using the SDLC requirements enumeration step and talking to users of the EDC application can produce a list of SOPs that need to be created and of existing ones that need to be rewritten. The optimal process should be ascertained before the pilot, and to the greatest extent possible, realized for use during the pilot.

A focus on the requirements and analysis SDLC phases that includes in-depth training on the EDC application can help result in a realistic assessment of the pilot. The cost of the training should be viewed as incidental since learning about EDC in general can help clinical trial participants appreciate the complexities of the data gathering and cleaning process regardless of methodology. At the very minimum, such training can also lead to new ideas on how to implement process improvements to existing processes built around a company’s current CDMS.

References:

¹ Laudon K. and Laudon, J. *Management Information Systems*. Prentice Hall. 1998.

² <http://www.cs.iastate.edu/jva/jva-archive.shtml>

³ Rauch, T., Kahler, S., and Flanagan, G. (Oct 1996) “Usability Techniques - What Can You Do?” *SIGCHI Bulletin*, 28(4), pp 63-64.

⁴ Plyler, R. W., & Kim, Y. (1993). *Methodology myths: Four tenets for systems developers*. *Information Systems Management*, 10(2), pp 39-44.



Who's behind the research?

Our lead researcher, Kirk Mousley, PhD received BS and MS degrees in Electrical Engineering from MIT and a PhD in Computer Science from Lehigh University. He has been the President of Mousley Consulting, Inc. since its founding in 1993 and has directed the company's efforts in the areas of clinical database design, data editing/cleaning, document management, and submissions.

Karl Mousley received his BS in Mechanical Engineering from Rose-Hulman Institute of Technology and a MS in Computer Science from Villanova University. He has been a senior member of the technical staff at Mousley Consulting, Inc. since 1993. Among his significant accomplishments are the investigation, evaluation, and implementation of new computer technologies for clinical data management systems and developing strategic plans for integrating these technologies into current systems. He has extensive experience preparing Standard Operating Procedures (SOPs).



EDC Today and EDC In Depth

EDC Management publishes well-researched, timely information about EDC technologies and processes.

EDC Today is a free electronic technical bulletin.

Each month we examine topic areas related to Electronic Clinical Systems (ECS) theory, technology, practice, or implementation.

Each *EDC In Depth* research report comes with an executive summary and may be purchased individually for \$395 or as a group of related reports for \$975. Available via downloadable electronic version or paper version sent via mail.

To subscribe to ***EDC Today*** or purchase a specific ***EDC In Depth*** research report:

Order online at
www.edcmanagement.com

Email us at
info@edcmanagement.com

Call us at
1-484-530-0300