The CTB Protocycle

In addition to the CTB practice model, we approach situations with a specific methodology unique to CTB for rapid applications development. The classical development life cycle used by many **other** organizations has grown from host application development over the years. It is often characterized in five serial System Development Life Cycle [SDLC] steps:

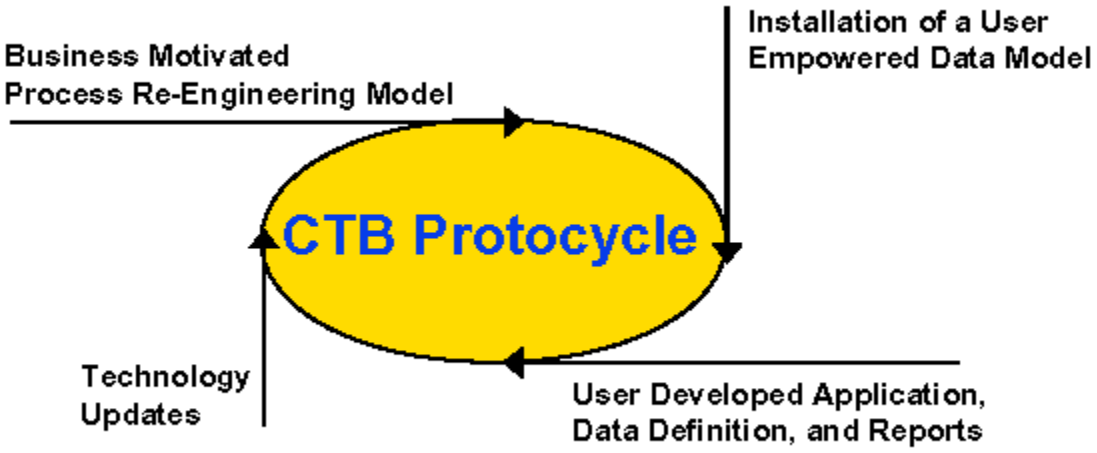
- Analysis
- Design
- Construction
- Testing and Documentation
- Implementation and support

This classical model and methodology assumes two parties are involved throughout the process. One is the user the second is the developer. An important aspect of this classical process is that the functionality requirements need to be well defined to the users' best ability for his application needs today. Apart from the users ability to express an understanding of these requirements, they can usually be expected to change over time.

For the classical development model to be effective, two things must be true. First, the end user must be able to spend the time necessary to accurately articulate his system needs to an analyst or consultant developer. The developer is more focused on construction than a working understanding of the complex business functions. In specialized and complex business situations, gaining this understanding is not a trivial task even when the user has a good understanding of his requirements. In many cases, the user does not have a comprehensive understanding. A second assumption for this development model is that the application needs, which are being coded, are static during the period, which the application is being analyzed, designed, constructed, and rolled out. Often this turns out not to be true because of external business changes or a changed understanding by the user. More often in current business environments, these two assumptions are increasingly untrue.

In addition, cost pressures and the need for faster development have reduced the time and resources available for the development of most systems. These realities often force impossible expectations and schedules on a moving target of specifications and scope. The net result is that the classical 5-step development life cycle has not been able to meet many of the development requirements of today.

What has evolved at CTB is a new development model we call the "CTB Protocycle." This CTB methodology and associated processes directly address the two difficulties of the classical development life cycle. The word CTB Protocycle is a combination of the word prototype with the classical development life cycle. It promotes a new development process and includes end-users in the evaluation and input process. This is, by design, not a terminating process like the classical model. It is rather a continuum, which cycles on a regular basis multiple times a year. That is to say that the CTB Protocycle is a repetitive process of continually redeveloping and reapplying to the system the phases that make up its development process. Its success is in part due to its ability to implement rapid change non-disruptive into classical host based systems.



The CTB Protocycle can be defined in four high level stages:

The **first stage** is the System concept. In this stage, end users and business owners develop process and application re-engineering models that are business focused and non-technical. The medium for exploring and presenting these ideas is to mock-up the application in a stand-alone demo. The mock-up is focused on the user's client application and is structured to present the process concepts in a non-functional mode. The tools for this process are usually graphical in nature and in most cases the user can develop the demo in a "what if" mode with only marginal help from technical resources. The business models that are produced purposely define payback and return for the corporation. This stage of the process is fully end user driven. The technical resources involved need not be steeped in a close understanding of the business. When approved by management for development, the mock-up will be applied to a client-server systems environment that is flexible and adaptive by design. This flexibility permits the creation of an infrastructure, which can provide the information requirements for the application. This information's flexibility gives the end user full opportunity of receiving quickly defined business models at a high level. Once the information requirements for the application are available, these models are refined into prototypes in later stages.

The **second stage** of the Protocycle process is the set up of an end user enabled system structure. This involves creating a flexible information and system architecture that allows access to the information required and ensuring that the computing platform provides a mission critical reliable infrastructure. The information architecture and systems infrastructure needs to be built on a single relational data model. The model must be made up from a complete set of the information required for the application and access to the information by the end user must be reliable. Some of this information will be found residing on legacy host systems and some on newly constructed local databases. The information model and systems need to provide a stable, reliable, and administered client server environment for access to both local and remote repositories. It must be fully compatible with off-the-shelf desktop, APIs and development tools. These tools are what the end users will use for prototype development within the structure, framework, and controls of the information model. Often this involves providing from within the model "canned routines" and triggers to assist the developers in the more complex aspects of the information. The information model and the process of designing it will be developed in more detail in the next section.

A design benefit for this architecture and infrastructure is the ability to develop and adjust application uses of the new data model without disrupting the existing legacy uses of host data. This allows for the graceful introduction of new client-server applications. In addition data will migrate down to the local Client repository on an on-going basis only for the data needed by the new application.

In addition to providing reliable access to the data in the information model for desktop development tools, the client-server platform must be administered. Permitting users the ability to actively participate in development does not mean that users should control the infrastructure. Most importantly, version control and change control management must be applied as applications or upgrades are developed. This process involves two structural components to be effective. First, the development platform must be separate from the production platform. A secure firewall must allow developers the freedom to experiment and make mistakes while at the same time preventing defects from creeping into the production environment. Physically segregating the developers from production systems and exercising a controlled roll out process for introducing changes into the environment most easily achieve this. A controlled roll out process must be established which allows administration of application code at a version level with full rollback capabilities. In this way, beta tested code can be introduced with a select a pilot group to the production environment. If errors have escaped detection during final test they can be quickly diagnosed in the controlled production environment of the pilot. If the problems are critical in nature, then the users can be rolled back to the previous version while the defect is being investigated.

The **third stage** of the process is the actual development and construction of the application. The end user drives the construction with assistance from internal and external technical resources. Development is performed upon the information model of combined host and local data. The integrity of the data is maintained at the database server, and new information requirements can be accommodated through a process of amending the local structure rather than re-writing the legacy. The task here is straight forward and in full agreement with the users' desires. Rather than teaching the analyst about the application, the process enables the end user with an accurate understanding of the tools and information structure. The state of tool development technology today makes this process more efficient than trying to teach the developer about the business. It is also able to more quickly adapt to dynamic improvements in technology and shifts in the applications business focus. The process of end user development is designed to be an iterative one. That is to say, it is never ending. The information model, upon which the development work is completed, controls the integrity of the data and reliability of the application foundation. The use, integration, and presentation of this information is controlled by the user. Through the iterative process, the end user explores and better understands his own application and re-engineering model. He can then easily adjust to improvements in the approach at a more granular level as information is discovered and the model is better understood.

The **fourth stage** of the Protocycle process is the enhancement and redefinition of the architecture and system just described. The enhancement and redefinition process involve upgrades and improvements to either the information model, the tools available in the original systems design, or the client-server infrastructure. These design changes need to negotiate the global upgrade of affected systems to the new platform, and is often an over- the-weekend "big bang" rather than gradual migration. In the current client-server world of today, these occur 3-5 times a year. This final stage forms the basis for the continuum of the process. As enhancements and improvements in technology are added to the system in a controlled manner, the process starts over again with a new series of business opportunities. The technologies that are available today permit the integration of new pieces in a relational model usually without having to disrupt or replace previous structures. This is piloted and investigated in the development database in a straightforward fashion often using CASE tools that are integrated with the application development environments, which end users use on their desktops. It also involves the addition of new product tools to the off-the-shelf development environments that provide access to the information model.

The improvements provided by the process involve quicker development along with the use of end users on an ongoing basis to enhance and refine application models. The fact that this new process, the system it operates within, the data it accesses, and its presentation, that is driven by the end user, makes the application more reactive to business and technology changes. The overall key here is to ensure that the model effectively acts as a catalyst in integrating the professionals in the MIS group with their end users. The combined skills and resources of these groups provide a real competitive advantage in the area of rapid development.