

Case Study: American Cancer Society

Introduction:

The American Cancer Society had purchased a Tier 1 system that met the majority of their needs, but lacked some functionality that was critical to their specific way of tracking fund-raising activities. They needed a module that would incorporate seamlessly with their new system, but they wanted it fast and needed it built as the business requirements were evolving.

The revolutionary development approach that resulted produced an application that was built in weeks, debugged in minutes and changeable at any time with little effort. The DES methodology that originated at the American Cancer Society is now encapsulated in a revolutionary tool called GenesisOne.

Problem:

The Florida Division of the American Cancer Society (ACS) was faced with a major system implementation that was discovered (too late) to not have the full capabilities they needed. Unfortunately, what was missing was critical to tracking their fund-raising activities, but because of the unfamiliarity with their new system, requirements would be evolving as the application was built.

ACS needed an application module custom built to augment the new system that supplied the missing functionality. However, they couldn't wait the time it usually takes for a "full development project". In addition, the completed application had to require little training since more time than they could afford was already being dedicated to training on the main system.

In addition, they preferred not to support it in-house, so service calls had to be completed quickly and effectively. Ideally, this would only be required when modifications were made to the primary system.

Technical Considerations:

Vision Genesis had to work outside of industry standard processes such as RUP due to time limitations. Because of this, there was no time to dedicate to creating global classes and libraries, nor to develop use cases or a firm set of business requirements.

Since it couldn't be supported in-house, Vision Genesis had to create something that required little time to service when changes were invariably made to the main system. They also had to structure a solution that would make debugging and error-trapping quick to identify and easy to resolve.

With the evolving, sometimes even unknown, process requirements, Vision Genesis had to quickly construct a development process that allowed for modifications, even major ones, to be made as the application was built and even more challenging -- after deployment.

The approach to create a module that would be standard in size and complexity, include all n-layers of the resulting software and able to be isolated and compiled separately. The solution (now the DES foundation for GenesisOne) was in encapsulating the each step of functionality in a structure that was easy to reference and provided its own documentation.

Solution:

Included functionality of a database application with user interface for data entry, a data warehouse with daily ETL processes, reporting and distributed access across 71 locations plus Puerto Rico.

The completed software's structure included all three layers of an n-tier application. Necessity required the first two phases of RUP to be bypassed, so documentation was maintained on the code within the individual forms, and by keeping the forms identified within a table structure and called by reference, forms were free to be resequenced as needed. It also enabled the application to be expanded and reprogrammed numerous times as the requirements were made known.

Since there were no module references, errors were triggered as the form was instantiated, making it very simple and quick to debug and redeploy.

Total development time took approximately two weeks, after which many functional changes were introduced, the most complex of which took no more than one day to complete, including debugging. Total time dedicated to coding was just under two months.

After deployment, upgrades to the main application's database, server changes and other IT changes required modification to the application's functional design. Most of the changes affected ETL and processing rules, and some affected the application flow itself. Regardless of the modification needed, it would take anywhere from 15 minutes to 3 hours to isolate the needed changes, institute it within the application's code and redeploy it. The application has been running nearly maintenance-free for almost 3 years.

The success of the DES methodology is now being implemented at a Fortune 500 company.