Databorough

A Visual Guide to CA 2E Synon Reengineering

Steve Kilner

Visual Guide to CA 2E Synon Reengineering

While 2E generated code is definitely not a starting point for modernization, the 2E model itself is nearly ideal. Unlike native RPG applications, Synon applications are developed and maintained in an architected model that was ahead of its time for business applications. Aside from enforcing a structured approach to development, the 2E model keeps UI, business logic and IO separate, and organizes them with an early form of object oriented design concepts. The result today is that Synon application logic is completely recoverable and reusable for transforming into a modern architecture and language.

This guide describes how a 2E Synon model can be transformed into a modern application, and includes the following sections:

- Guiding Principles for Reengineering
- Bad practices - What not to do
- Best practices - What to do
- A basic process for reengineering projects
- MVC/OO/REST – Mapping 2E to a targeted architectural pattern
- X2E conversion options and facilities

Guiding Principles for Reengineering

Persistence of Business Constructs

Reengineering, rewriting or replacing a system by any means does not mean replacing business rules and data relationships. R*eplacing a system mostly means recreating these same business rules and data relationships*, in a new, more flexible manner.

Business Logic

By some estimates, a typical 2E application with a million lines of code will contain about 30,000 business rules. These rules have developed over many years and have proven themselves in the operation of the business. In almost all cases, *no one knows what all these rules are. In 2E we know WHERE they are, so it's a matter of recovering and restructuring them for reuse.*

Data Model Relationships

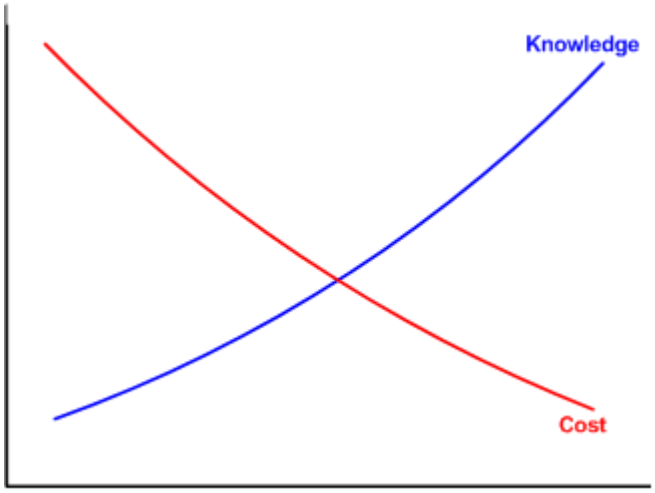
As any 2E developer will know, the application tables contain data elements that relate to each other in a very specific way. The automation of business operations is built upon these very specific relationships, so reusing them is key to the business going forward.

The Foundation of Reengineering

As has been measured in numerous projects[(1)] approximately 35% of the effort of software development, and 45% of defects, originate in the requirements and design phases.

The single most valuable decision that an IT manager can make, when replacing a system, is to *recover and reuse existing knowledge* from the legacy application. Doing so will reduce the cost, time and defects of the new system. By using a tool such as Databorough's X-2E, the business rules that are in action diagrams, and the data model itself, can be readily recovered and manifested in documentation or converted directly to new system code. Other tools use the generated code.

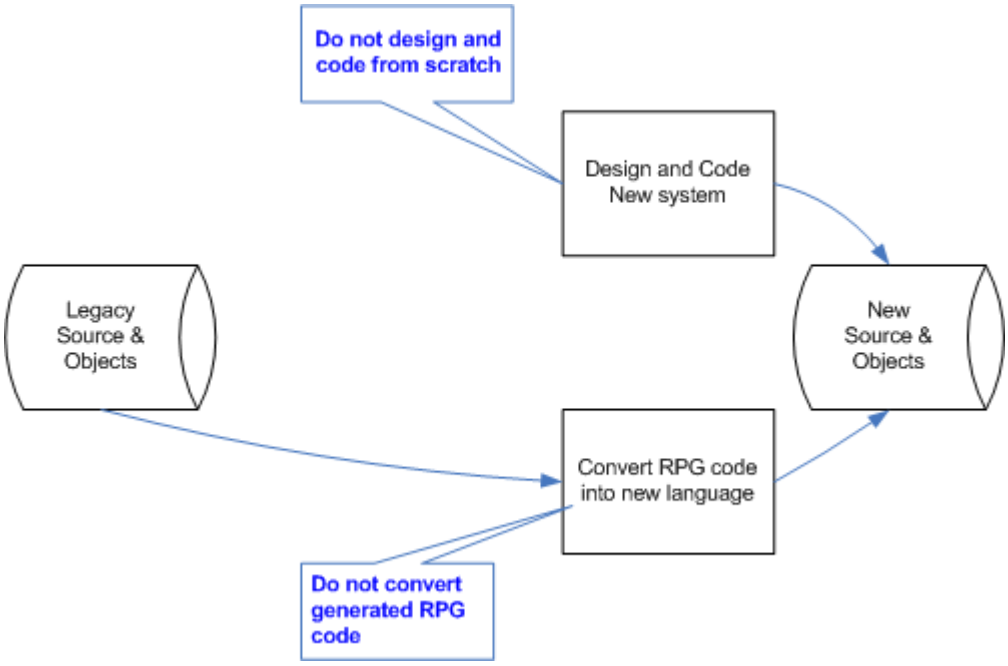
The following simple diagram depicts the fundamental truth of system development.



Stated simply, the more you know, the less it will cost you.

Or conversely, the less you know, the more it will cost you.

Bad Practices of 2E Reengineering – What Not To Do

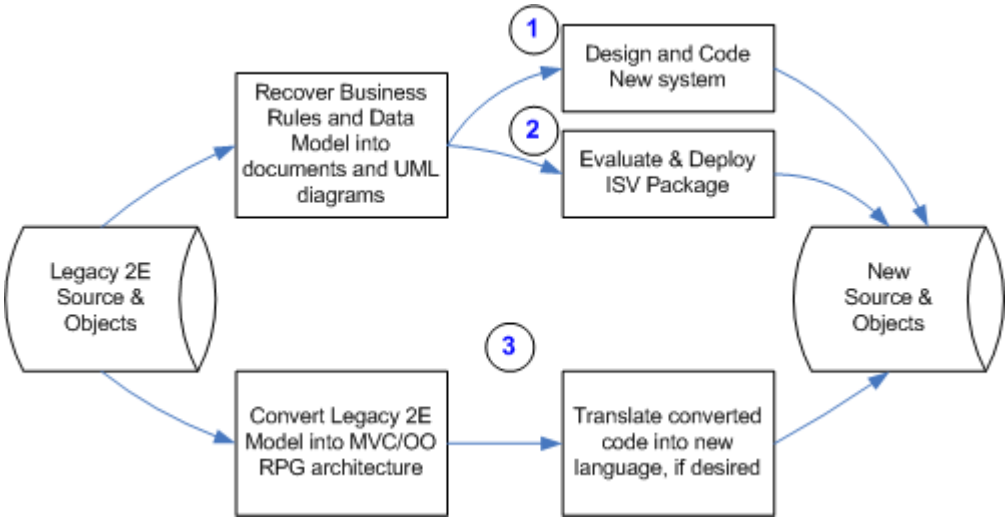


Do Not Design From Scratch – You will add *huge amounts of risk, cost and time* to your project. Your legacy system has vast amounts of proven, recoverable business rules, data rules and application definitions – use them and save money, time and risk. Start from scratch at your own peril.

Do not convert the 2E-generated RPG code – There are a number of reasons why you should not consider converting the 2E-generated RPG code:

- You will be converting approximately 20 times more code
- The new code will not follow a modern architecture design
- No one will recognize the new code and it will be extremely difficult to maintain
- You are likely to have difficulties with scaling and performance
- This is not a viable long term investment

Best Practices of 2E Reengineering – Three Good Options



Options 1 and 2: Recover 2E business rules and data model to 1) rewrite or 2) purchase

Your legacy application may represent as much as millions of dollars of investment.

It's *full of time-tested business rules, data relationships, work flows, application boundaries* and so on. You may well want to improve all those things, but chances are that the vast majority of this design is still good – use it!

The recovered business rules and other information can be output to Word, Excel or XML documents or generated into UML Use Case, Object or Activity Diagrams. Use these outputs to then feed into:

1. the requirements and design processes of a new system, *or*
2. the functionality and gap analyses of packages being considered for purchase

> Rather than converting the 2E generated RPG/COBOL code, which is full of both redundant and otherwise useless 2E code:
>
> *Rebuild from the 2E model itself.*

Option 3: Restructure the model first, and then convert it

There are a number of advantages to this approach (described in more detail below), but overall, you will end up with a cleaner, more modern system. Some of the resulting system characteristics of this approach are:

1. Object oriented components
2. Model-View-Controller, MVC, architecture
3. Stateless, or REST, session management
4. Free formatted coding, such as RPGLE/Free

By first converting the 2E model into RPG /Free in an MVC/OO architecture, the existing team is able to use their existing skills to understand and work with a more modern architecture while maintaining comprehension of the new code.

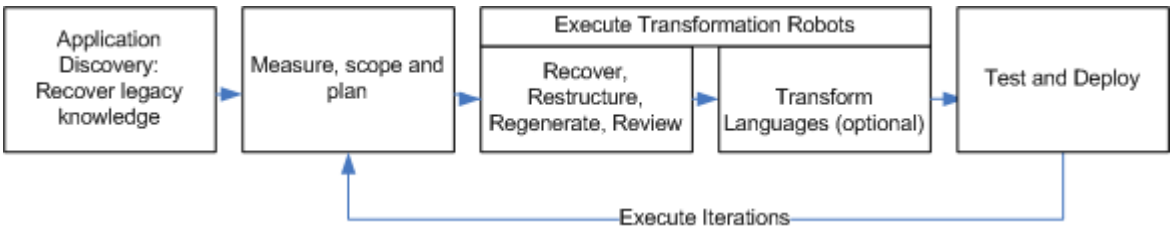
A subsequent, optional step, is to then convert the MVC/RPG code into another language, such as Java/JSF. More details on the overall conversion process and options are described below.

A Basic Process For Reengineering Projects

One of the starting points of reengineering project is to recover existing knowledge from the legacy system. This covers such items as business rules, data model designs, work flow activities, UI designs and so on.

Another key starting point is to construct the beginning elements of a new architecture. For most commercial IT applications this will follow a multi-tiered pattern of a Model-View-Controller (MVC) architecture, constructed with Object Orientated (OO) components, and commonly using REST principles of state management.

With existing knowledge in hand, and decisions made about the architectural principles, planning and execution of the project can begin. Whether the project methodology follows a traditional style, such as waterfall, or something more like Agile, the essential process of this kind of reengineering project is best executed with repeated iterations of planning, designing, coding, testing and deployment (whether to production or staging.) A simple overview diagram:



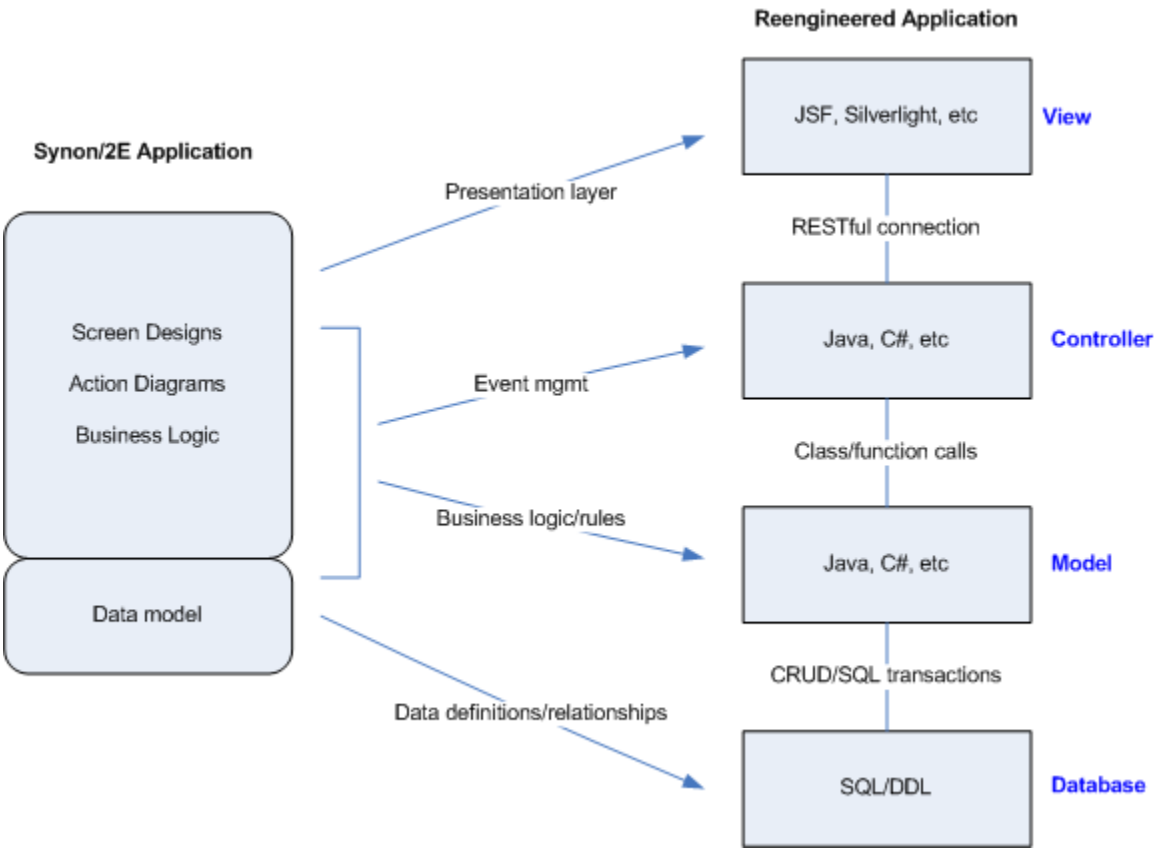
- Application Discovery: recover high level legacy knowledge
- Measure, scope and plan, based on the recovered knowledge
- For each iteration, execute transformation robots
    - Recover detailed business rule, data model and work flow information
    - Restructure old code into new architecture
    - Regenerate code into new structure
    - Review results
- Optionally transform the product into new languages with further use of transformation robots
- Test and deploy the iteration

MVC/OO/REST – Mapping 2E to a targeted architectural pattern

One of the core objectives of any legacy reengineering project is certain to be implementation of a modern architecture. While nothing is final in the world of software engineering, the most widely accepted design principles utilize the concepts of multi-tiered components, developed as objects, in a model-view-controller pattern, operating in a RESTful (stateless) manner. This architecture helps achieve higher level goals of scalability, modularity, maintainability and use of agile development.

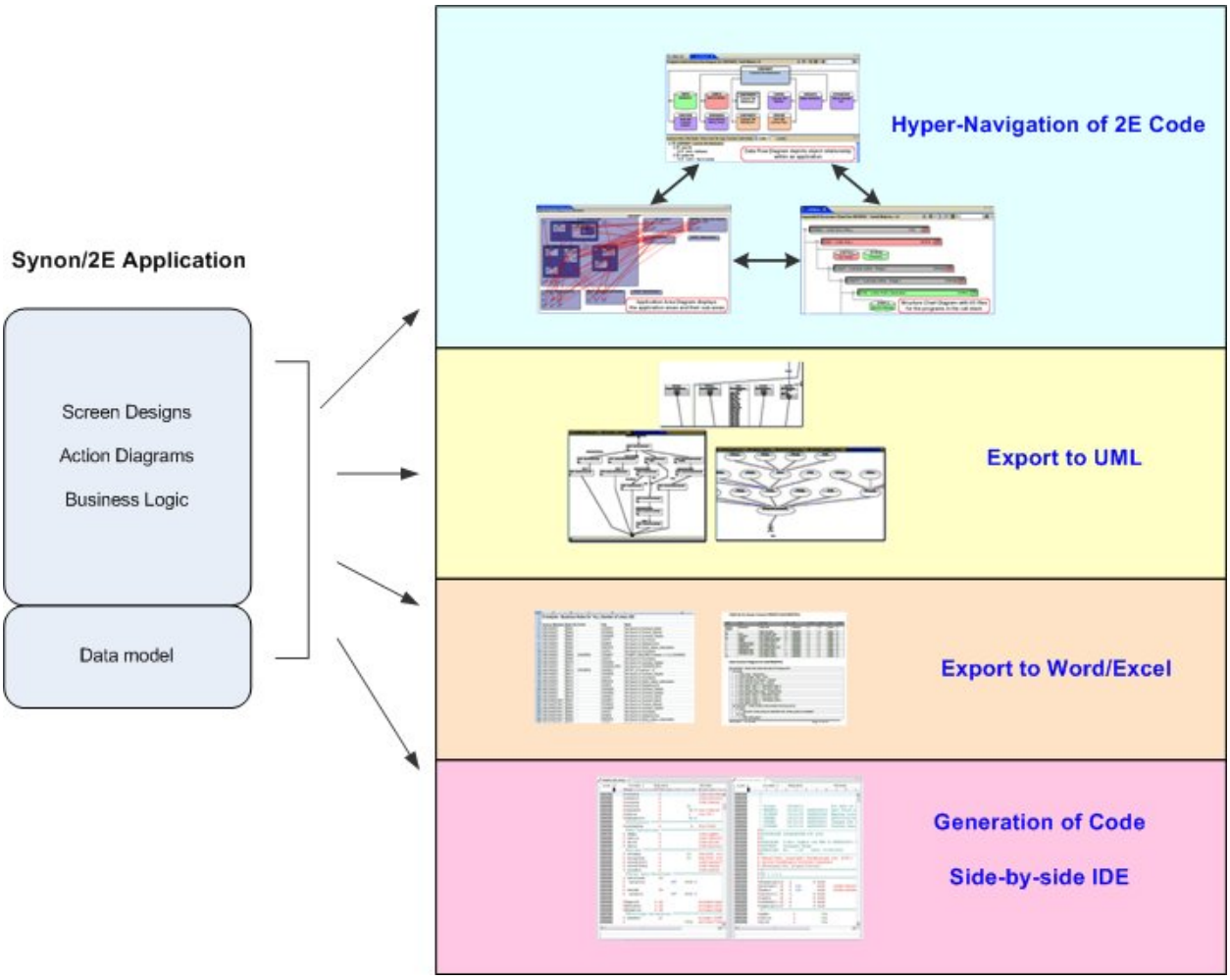
In the RPG world this form of architecture has been rarely used due to the constraints of the language and development tools. The notable exception to this was the in the case of Synon/2E which followed crucial elements of this architecture from the onset. Because these elements are discernable within all applications developed with 2E, it is entirely feasible to automatically reengineer 2E applications into a modern architecture using the languages of choice, such as Java and JSF.

This diagram depicts how 2E elements are mapped and converted into an MVC/OO/REST style architecture. This mapping and conversion can be realized automatically with Databorough's X2E.



X2E conversion options and facilities

Whether you have decided to automate the conversion of your 2E application to another language, or simply decided to recover design information for purposes of feeding a manual rewrite project, X2E provides many options, tools and outputs to expedite your work. The diagram below gives an overview of the X2E facilities.



Hyper-Navigation of 2E Code

The ability to understand the system being replaced is essential to defining the requirements of the replacement system and feeding the design process. This understanding comes in the form of both graphical and textual representations of the database model, action diagrams, business processes and program functions and relationships.

X2E provides a point and click navigation GUI for instant research, analysis and information extraction.

Export to UML

If the existing system is being used to feed the design of a new system then use of UML documents is particularly useful to expedite tasks.

X2E analyzes the 2E model and produces UML Action Diagrams, Use Cases and Object Diagrams.

Export to Word/Excel

All of the screens presented in the GUI can be exported to either Word or Excel to facilitate the processes of analysts and designers.

Generation of Code in Side-by-side IDE

X2E generates code derived from the X2E model in a variety of languages, such as Java, JSF, etc. The generated code can then be compared side by side with the original 2E code for detailed inspection.

Because X2E converts code by first converting the 2E model to RPG in an MVC/OO/REST pattern, this intermediate code can also be viewed and compared to aid in the comprehension of how the conversion took place.

Summary

Databorough's X2E provides an unparalleled, uncompromising solution for migrating legacy Synon/2E applications. For more information and a detailed walkthrough, contact Databorough today.

(1) Statistics are from *Estimating Software Costs* and *Applied Software Measurement*, by Capers Jones