MainTrend Ltd.

E-mail: info@maintrend.net
Site: www.maintrend.net

PowerBuilder to .Net (PB2N)

Thin Client Migration

(*Based on Sybase DataWindow .NET™ technology*)

White Paper

June 2012

Contents

MainTrend Ltd.

Executive Information

This document describes MainTrend's Solution for PowerBuilder to .NET Conversion. The purpose is to provide general information and understanding of the conversion process and a typical, general methodology that can be applied to PowerBuilder application's migration projects.

For many years PowerBuilder was one of the most serious and robust RAD tools for Client/Server and n-tier development. Software firms invested in PowerBuilder frameworks. Billions of PowerBuilder code lines have been written by thousands of trained developers. Reliable high-performance PowerBuilder applications are deployed worldwide. Enterprises invested in a PowerBuilder infrastructure assuming that they were investing in a platform that would remain vibrant for many years. However the forces of the market thought otherwise. PowerBuilder is now a niche player in the application development tool market. The number of new installations is diminishing. PowerBuilder professional personnel are becoming scarce.

But there are many enterprises with deployed active PowerBuilder applications. These enterprises must maintain their current systems. Every system in production has to deal with dynamic changes and enhancements. Enterprises are at a point where they have to make a strategic choice: to continue building applications in PowerBuilder where there is no future or to move to another platform. Remaining with PowerBuilder causes no upheavals but insures the widening of the legacy gap. Moving to another platform is probably strategically correct but there are factors to be considered. One major consideration is an enterprise's investment in legacy applications and the significant cost of a new application development. Another consideration is how to ensure a rapid and smooth migration path from one infrastructure to another without disturbing critical systems, specifically how to integrate the new with the old.

Nowadays more and more organizations are urged to move their PowerBuilder applications to a modern thin-client environment. There are 3 options, which can be considered: face lifting, manual rewriting from scratch and automated conversion.

Face lifting uses the existing PowerBuilder application as a back-end or as a source for the web-deployment, and therefore does not really move it to a modern environment. All the maintenance problems remain the same, with addition of continuous "face lifting deployment" project with each new version of the application.

Rewriting PowerBuilder applications from scratch is a very costly, time consuming process and may lead to the long learning curve for the end-users. Another significant point is that legacy applications and their maintenance changes are not always well documented. Even when an original system analysis document exists, the risk of business knowledge loss is very high.

Automated conversion as a replacement methodology saves all the investments in business knowledge and opens the way to maintenance and further development in a new modern environment.

Migration Methodology

Moving to another platform requires a number of factors to be considered. One major consideration is an enterprise's investment in its legacy applications and significant cost of the new application development. Another consideration is how to ensure a rapid and smooth migration path from one infrastructure to another without disturbing critical systems. The more valuable the encapsulating an organization's business logic software is, the more complex is the process of its evolution. Maintaining legacy software business logic is the important part of an organization's knowledge storage. Smooth and reliable transformation requires preserving all of the needed business logic.

The time to start migration is determined by the moment the new requirements cannot be implemented with the old technology. The main goal is not only to just migrate into a new environment, but also to have an opportunity to implement new technologies. The right migration strategy also means changing the software ideology to meet new requirements. Therefore, just emulating the old legacy environment with the new one is not enough.

More and more organizations are looking to move their PowerBuilder applications away from the thick client, proprietary environment provided via PowerBuilder, into the advanced thin client environment provided by .NET technologies. In order to accomplish this, some organizations attempted to manually rewrite their PowerBuilder applications to .NET. Some other organizations used various conversion tools, currently available on the market.

Each approach has its "pro's" and "contra's".

Rewriting from scratch as a replacement methodology can produce very effective resulting code, which exactly matches the target environment (if the development team is sufficiently qualified, if not, it is better not to touch the application at all). On the other hand, it is a very costly and time-consuming process. It also may lead to the long learning curve for the end users and for the maintenance team. And the significant point is that legacy applications are not always well documented. Even when an organization has the original system analysis document, not all the maintenance changes may have been documented, so the risk of business knowledge loss is very high.

Automatic conversion as a replacement methodology is much faster, much cheaper, and also ensures that all the business knowledge is preserved. Also the resulting application is similar to the original one, and the learning curve for the end users and for the maintenance team is not as long as with a "brand new" application. On the other hand, the resulting code is generated automatically. It is good, but not necessarily optimal for the target environment and for the specific application.

PowerBuilder as a migration source is the ideal case for **the mixed approach**. A central element in PowerBuilder environment is DataWindow, a patented Sybase technology – very powerful object, concentrating almost all the data processing of a PowerBuilder application and almost all the visual representation of the application's data. The PowerBuilder DataWindow object is one of the main reasons for the success that PowerBuilder has achieved as a software development tool. Usually in a PowerBuilder application DataWindow objects consume about a half of the application building efforts. So we decided to use Sybase DataWindow .NET technology for PowerBuilder DataWindow objects in the automatic part of the conversion, and to make all the

other PowerBuilder objects' conversion as flexible as possible to allow manual adjustment of the automatically generated business logic objects' methods.

Such migration concept unites advantages of both approaches:

- The original application is automatically reversed engineered.
- All the PowerBuilder DataWindow definitions are automatically transformed to DataWindow .NET definitions.
- All the other PowerBuilder objects are converted to corresponding .NET classes (or sets of classes).
- All the .NET classes are generated in match with the original inheritance tree.
- All the attributes and method signatures of the original classes are automatically converted to the corresponding attributes and signatures.
- The bodies of all the methods in the generated .NET classes, in addition to the automatically converted code, contain the original PowerScript code included as a comment. This allows manual adjustment for such scripts and getting the effective resulting code, which exactly matches the target environment. Also such approach preserves the business knowledge from being dropped or omitted and makes the maintenance with the original team significantly easier.

PowerBuilder to .NET (PB2N) is a software engine toolset and sets of methodologies that take an application coded in PowerBuilder and generate .NET application source code with reasonable need for a programmer intervention. .NET as a target environment allows maximum platform flexibility. .NET has proven itself to be a superior technology as a modern business application tool for reliable applications. Moving to .NET is a strategic decision for an enterprise, a decision to make another step into the future. An enterprise that decides to migrate to this modern development environment will use PowerBuilder to .NET (PB2N) as its conversion tool.

Target Environment

The conversion target is a full thin client environment based on the modern .NET methodologies.

The resulting application requires no client installation, except of a browser. Currently supported browsers are Internet Explorer 6.0 and higher versions. The same browser software version has to be installed on all the client workstations. All the browser instances on the client workstations should have the same settings (security, JavaScript etc.).

Application changes, which are made to the DataWindow definition files, are available immediately. Business logic changes, encapsulated within application server components, are available immediately after installation. The database is accessed from the application server components.

This architecture provides centralized application management and does not require any additional software to be installed on the end-user workstations.

Migration Process and Solution Architecture

Generally the migration process has the following steps:

1.   Detailed analysis, choosing of the target server-side platform.
2.   Verification of the source application.
3.   Export of all the PowerBuilder objects of the migrated application.
4.   Reverse engineering of the original application.
5.   Code generation.
6.   Manual adjustments of the application classes' method's code.
7.   Preparing test environment.
8.   Database migration (if a new database platform is chosen).
9.   Unit testing, including required database connection for the data access layer objects.
10. External programs migration (if any).
11. Code integration and thin platform adaptation. Application restructuring according to the thin client conversion model.
12. Database integration and required changes according to the chosen database platform.
13. General graphical design, fine tuning of the resulting GUI in line with customer standards
14. Application integration testing and required fixing.
15. User acceptance.
16. Building the production environment and putting the new application to production.
17. Trainings for the customer's developers.

The conversion itself is covered by steps 3 through 6 and uses PowerBuilder Conversion Suite. PowerBuilder Conversion Suite consists of these parts:

▪   PowerBuilder Converter itself, which translates each PowerBuilder application into a set of .NET modules and DataWindow definitions.
▪   Conversion and development methodologies, including best practices for the most effective manual completion and enhancement of the resulting applications.

We use PowerBuilder "library export" functionality to obtain all the sources of PowerBuilder objects for the specific PowerBuilder library set. After the reverse engineering stage, the code generation stage comes. All the DataWindow definitions are automatically used with the new DataWindow .NET technologies. All the other PowerBuilder objects are parsed and converted to corresponding .NET classes. The bodies of all the methods in the generated .NET classes contain automatically generated code and the original PowerScript code as a comment. These methods are adjusted in the manual completion stage. Also the manual part includes GUI tuning according to the customer's requirements and database access tuning.

PB2N toolset uses sophisticated concept of multi-level embedded parsers and template-based code generators. This concept allows use of different parsers for different parts of script and objects' types, and improves the conversion performance.

Visually the conversion process can be depicted in this manner:

```
┌─────────────────────────────────────────────────┐
│                Source Application                 │
│   ┌──────────────────┐    ┌──────────────────┐   │
│   │      PBL         │    │    External      │   │
│   │                  │    │    Resources     │   │
│   └──────────────────┘    └──────────────────┘   │
└─────────────────────────────────────────────────┘
                        │
                        ▼
        ┌─────────────────────────────────────┐
        │ XML Application Report and export files │
        └─────────────────────────────────────┘
                        │
                        ▼
        ┌─────────────────────────────────────┐
        │      Automatic Conversion Part       │
        └─────────────────────────────────────┘
                        │
                        ▼
┌─────────────────────────────────────────────────┐
│         Intermediate Application Skeleton         │
│                                                   │
│  ┌──────────────────┐      ┌──────────────────┐   │
│  │ DataWindow       │      │    External      │   │
│  │ definitions      │      │    Resources     │   │
│  └──────────────────┘      └──────────────────┘   │
│        ┌──────────────────────────────────┐       │
│        │  .NET Packages with .NET classes  │       │
│        └──────────────────────────────────┘       │
└─────────────────────────────────────────────────┘
                        │
                        ▼
        ┌─────────────────────────────────────┐
        │       Manual Conversion Part         │
        └─────────────────────────────────────┘
                        │
                        ▼
┌─────────────────────────────────────────────────┐
│                Target Application                 │
│  ┌──────────────────┐      ┌──────────────────┐   │
│  │ DataWindow       │      │    External      │   │
│  │ definitions      │      │    Resources     │   │
│  └──────────────────┘      └──────────────────┘   │
│        ┌──────────────────────────────────┐       │
│        │          .NET Packages            │       │
│        └──────────────────────────────────┘       │
└─────────────────────────────────────────────────┘
```
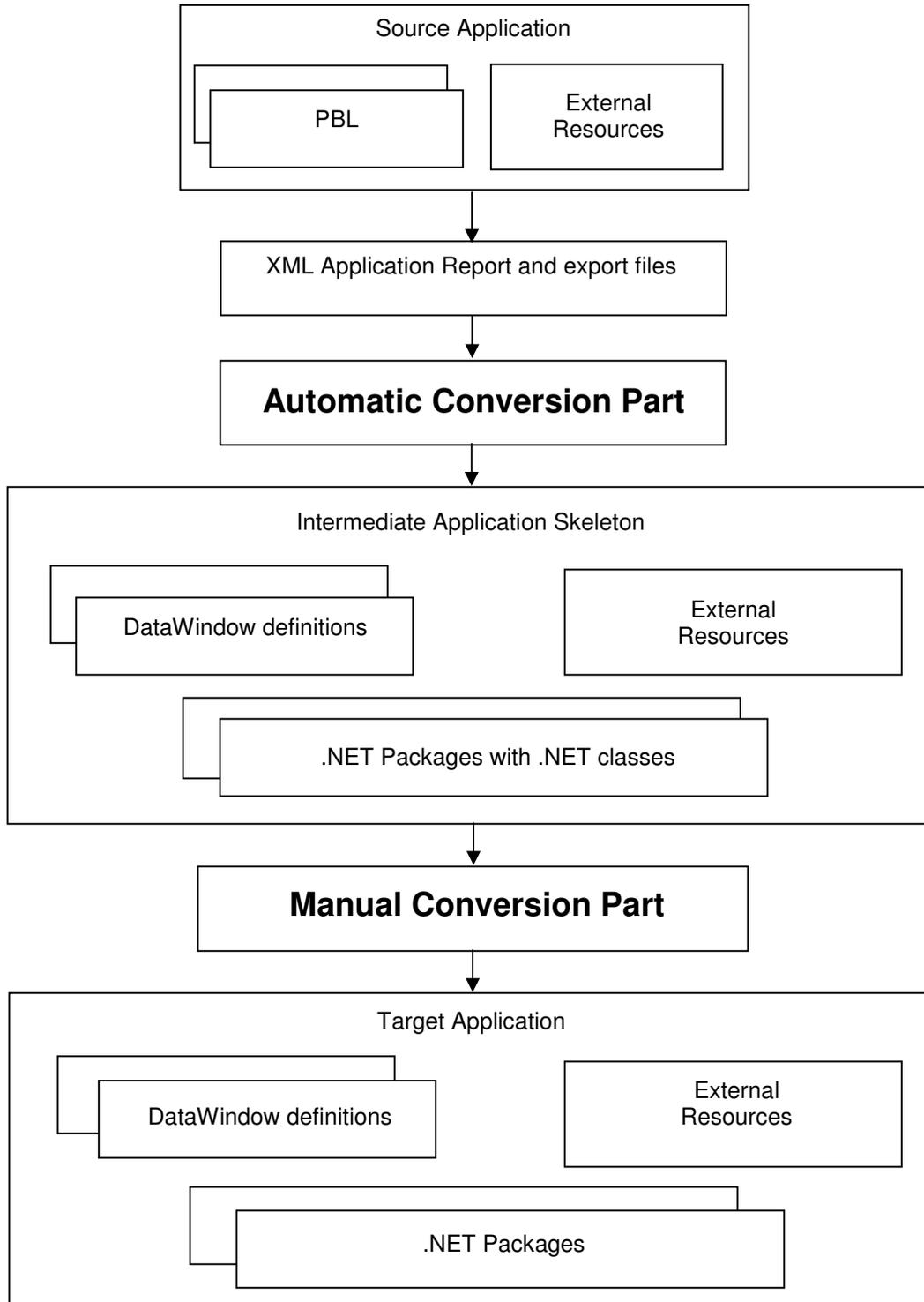
Logically the converted application provides all the needed for the thin client environment functionality:
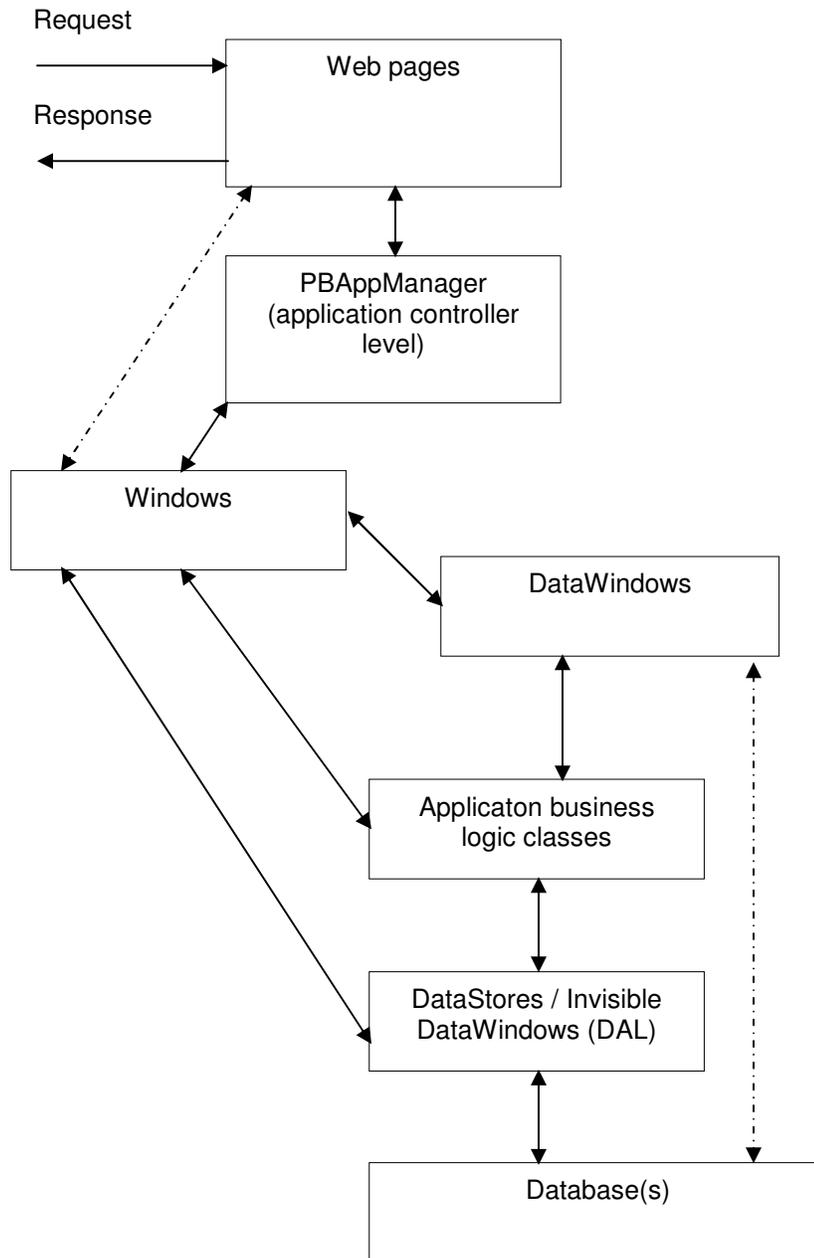
The conversion services, pricing model and estimation

We offer an automated conversion service that can lift PowerBuilder applications to the .NET technologies while preserving business logic base and user interface investments.

Most of the work can be done remotely, and manual part can be divided between MainTrend and a customer or a third party conversion partner, to be as much as possible convenient to all the parties.

There can be different models of the conversion projects – from full conversion to different "reverse engineering" stages, automatically generated script can be used by customer's developers as a tool for business logic capturing, etc. Business model can also vary – from fixed price projects to hourly based consulting. All the mixed type models are also available.

MainTrend Ltd.

Summary

This White Paper describes the MainTrend's path for successful PowerBuilder migration into new advanced .NET environment.

This migration methodology provides:

- Rapid, accurate and valid way for organizations to migrate PowerBuilder applications to .NET.
- Low assimilation costs for the new application.
- The same productivity and more, in a new modern environment.
- No limitations for the target server-side platform: any .NET-supporting platform will fit
- Much lower cost and faster solution for the legacy applications' migration path.
- Flexible open-platform development environment.
- Cross-application code reuse enabled environment and reduced maintenance costs.
- Seamless integration with modern technologies and new approaches.

Reducing the conversion process significantly, PB2N frees organizations up from the translation process so they can continue to do what they do best — remain dedicated to running their core business functions instead of dedicating themselves to one usually lengthy migration project.

By working with the MainTrend's team of specialists, organizations receive the added benefit of applying our experience and advanced expertise to their unique and often complex requirements.