Consultingwerk Software Services Ltd.

- Independent IT consulting organization
- Focusing on OpenEdge and related technology
- Located in Cologne, Germany, subsidiaries in UK and Romania
- Customers in Europe, North America, Australia and South Africa
- Vendor of developer tools and consulting services
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration
- Experts in OpenEdge Application Modernization
Mike Fechner

- Director, Lead Modernization Architect and Product Manager of the SmartComponent Library and WinKit
- Specialized on object oriented design, software architecture, desktop user interfaces and web technologies
- 28 years of Progress experience (V5 … OE11)
- Active member of the OpenEdge community
- Frequent speaker at OpenEdge related conferences around the world
- Helps to protect your investment in your OpenEdge based application
- The framework is designed to modernize existing OpenEdge applications and to provide the foundation of new projects
- In the cloud and on premise
- UI flexibility – Desktop, Web & Mobile
- The architecture of the SmartComponent Library simplifies integration with future technologies and the implementation of new business requirements.
Introduction / Application Modernization

- Modern OpenEdge Application Architecture
- ADM2 SDO migration
- TTY Upgrade editing migration
- ABL GUI migration
- OSIV3G Modernization example
Modernization Strategies

- Modernization of the whole application?
  - Going from ABL GUI to GUI for .NET or Web or Mobile
  - What is the “final” UI technology
  - GUI for .NET as an intermediate / integration with legacy GUI while the backend is rearchitected

- Or do we (first) add a few new features?
  - Mobile client for parts of the application
  - REST/REST(ful) interfaces for parts of the application
Quality of the application

- Are parts of the application reusable?
  - With no or little changes
  - Are major functional changes required?
  - Are major changes to the database structure required?

- Can parts of the application serve to describe the requirements
  - Legacy code review as part of the requirements definition
  - Is the existing source code the only (complete) description of the application functionality?
Skills of Development team

- New development process (let’s get agile)
- New tools (Progress Developer Studio, SCM, Unit Tests, Frontend tools)
- New architecture: Distributed
- New development languages
  - OOABL
  - html, JavaScript, TypeScript, rapidly changing
  - Desktop technologies
Modernization Examples in this presentation

- The modernization examples provided in this presentation demonstrate refactoring techniques based on simple examples
- These or similar techniques can be used for other types of ABL legacy applications
- Foundation for source code migration is always
  - understanding of existing code structure/architecture
  - understanding of target architecture
  - a concept
  - tools
  - experience
  - trial and error, or let’s call it a proof-of-concept
Introduction / Application Modernization

- **Modern OpenEdge Application Architecture**
- ADM2 SDO migration
- TTY Upgrade editing migration
- ABL GUI migration
- OSIV3G Modernization example
OERA OpenEdge Reference Architecture

- Architecture blueprint for service-oriented OpenEdge applications
- Initially released with OpenEdge 10.0 (15+ years)
- Primary goals at the time
  - AppServer enabling OpenEdge applications
  - Building non-monolithic OpenEdge applications
  - Supporting client flexibility
  - Providing guidance for use of the ProDataset
  - Providing guidance for use of OOABL (later, around OE10.1+)
OERA today

- Fast forward to 2015 …
- Modernization of OpenEdge applications more relevant than ever; especially since Telerik acquisition and demands for UI flexibility
- OEAA – OpenEdge Application Architecture, redefining the OERA
- OERA back on focus, foundation of the **CCS (common component specification)** project as a vehicle for community and Progress driven architecture-spec efforts
- More detailed specs, rather than just programming samples
- Specs that an application or framework could be certified against
- CCS starting to influence “in-the-box” features
Business Entities

- Business Logic Component in the Business Service Layer
- Manages a set of database tables
  - Customer
  - Order/OrderLine/Item (read-only)
- CRUD actions (create, read, update, delete)
- Custom actions, verbs of the entity (PutCustomerOnCreditHold)
- Primary backend component for the JSDO
  - Kendo UI, Kendo UI Builder
  - NativeScript
The OpenEdge Application Architecture (OEAA)

Can be ABL GUI

That is the JSDO

RESTful, SOAP, ...

Presentation (UI)
- Client Data Object

Enterprise Services (API)

Service Interfaces
- Data Service Catalog
- Service API
- UI Metadata

Business Components
- Task
- Entity
- Workflow
- Synchronization (IT)

Data Access
- Data Server
- ABL
- SQL
- Synchronization (DB)

Common Infrastructure
- Startup Manager
- Context DataManager
- Session Manager
- Authorization
- Service Manager
- Authentication
- Connection Manager
- Message Manager
- Property Manager
- Logging Manager
- Catalog Manager
- Translation Manager
- Analytics Manager

© 2018 Consultingwerk Software Services Ltd. All rights reserved.
- Introduction / Application Modernization
- Modern OpenEdge Application Architecture
- **ADM2 SDO migration**
- TTY Upgrade editing migration
- ABL GUI migration
- OSI4V3G Modernization example
ADM2 SDO migration

- SmartDataObjects (SDO’s) were introduced with Progress Version 9 and the ADM2
- SmartDataObjects have a similar responsibility within an application as a Business Entity
  - Centrally managing all read and update access to a database table
  - Based on temp-tables
  - Providing dedicated hooks for validation and calculated fields
  - Providing standards for change tracking and error reporting
  - Providing a central location for custom code that fits into the scope of the set of database tables
```
RETURNS DECIMAL
{
  /* parameter-definitions */
}

Purpose:
Parameters: <none>
Notes:

RETURN RowObject.MonthQuota1 +
    RowObject.MonthQuota2 +
    RowObject.MonthQuota3 +
    RowObject.MonthQuota4 +
    RowObject.MonthQuota5 +
    RowObject.MonthQuota6 +
    RowObject.MonthQuota7 +
    RowObject.MonthQuota8 +
    RowObject.MonthQuota9 +
    RowObject.MonthQuota10 +
    RowObject.MonthQuota11 +
    RowObject.MonthQuota12 .

END FUNCTION.
```

```
Purpose:
Parameters: <none>
Notes:

DEFINE INPUT PARAMETER pcSalesrep AS CHARACTER NO-UNDO.

IF pcSalesrep = ? OR pcSalesrep = "":U THEN
    RETURN ERROR "Salesrep may not be empty".

END PROCEDURE.
```
Reasons to migrate SDO’s to Business Entities

- Procedural nature
- Unclear separation between frontend and backend
- Complicated API when used from outside the ADM2
- Customization complicated, lots of code, understood only by a few developers
- Single table interface, Proprietary change tracking mechanism based on two temp-tables (a prototype of the ProDataset)
- AppBuilder tooling required for ADM2
- ProDataset better supported with modern tooling and UI
SDO migration

- Well defined source code structure
- Well defined patterns for internal procedures/functions
- Meta data defined in preprocessor directives
- SDO RowObject temp-table can serve as foundation for Business Entities
Source code parsing using Proparse

- ABL syntax parser, abstract view on ABL source code, based on ANTLR
- Eliminates the need for text based source code analysis
  - Resolves issues with line-breaks, abbreviated keywords, mixed order of keywords
- Open source
  - github.com/oehive/proparse
  - github.com/consultingwerk/proparse
  - github.com/riverside-software/proparse
- Actively maintained in various forks, support for 11.7 ABL syntax
Proparse

SDO to Business Entity Migration

- SDO structure imported into SmartComponent Library Business Entity Designer
- Functionality implemented as a plugin to the tool
  - Not relevant for all users of the Business Entity Designer, can be disabled
  - Allows for easy customization in fork of the plugin
- Wizard supports changes to the SDO structure, e.g. adding/removing/renaming fields of the RowObject temp-table; application of new naming standards
Demo

- Use Business Entity Designer plugin to convert SDO into Business Entity
Source Code migration

- Migration of arbitrary source-code influenced by existing coding style
- Migration of SDO source code requires
  - Location of relevant source code
  - Conversion of procedures/functions to methods
  - Modify procedural invocation of sub-routines to class based invocation
  - Change access to RowObject fields to new temp-table name
  - …
Proparse based source-code migration

- Extension to Proparse
  - ABL based API’s to locate relevant code
  - enabling Proparse for in-memory manipulation of source code
- Alternative is to use Proparse for understanding of legacy code and simple OUTPUT TO or LONGCHAR operations to build new source code
- XFEF, COMPILE listing sometimes used as well. But majority of input is present in Proparse
ProparseHelper:Initialize().
ProparseHelper:ExportDatabaseSchema().

oParseUnit = ProparseHelper:ParseFile("ModernizationWorkshop/Adm2Salesrep/dsalesrep.w":U).

oRoot = oParseUnit:getTopNode().
oChild = oRoot:firstChild().

DO WHILE VALID-OBJECT (oChild) ON ERROR UNDO, THROW:

    IF NodeTypes:getTypeName (oChild:getType()) = "FUNCTION":U THEN DO:

        ASSIGN cId = ProparseHelper:GetIdNodeText (oChild).

        IF cId MATCHES "calculate_*":U THEN DO:

            END.
            END.

    FINALLY:

        IF VALID-OBJECT (oChild) THEN
            oChild = oChild:nextSibling().
        END FINALLY.

END.

Continued on next slide
Recursively processes JPNodes

Injected into NodeWalker, rewrites RowObject references in AST

Returns modified function source code
/**
 * Purpose: Processes a JPNode
 * Notes:
 * @param poNode The JPNode to process
 */

METHOD PUBLIC VOID ProcessNode (poNode AS JPNode):

DEFINE VARIABLE cFieldName AS CHARACTER NO-UNDO .
DEFINE VARIABLE oId AS JPNode NO-UNDO .
DEFINE VARIABLE oFieldName AS BufferFieldName NO-UNDO .

IF NOT ProparseHelper:HasChildNodeOfType(poNode, "ID":U) THEN
    RETURN .

    oId = ProparseHelper:FindChildNodeOfType (poNode, "ID":U) .

ASSIGN cFieldName = oId:getText ()
    oFieldName = BufferHelper:ParseFieldName (cFieldName).

IF oFieldName::TableName = cFromBufferName THEN DO:
    ASSIGN oFieldName::DatabaseName = ?
        oFieldName::TableName = cToBufferName .

    oId:setToken (NEW RefactoredToken (oId:getToken(),
        oFieldName:GetExpression ()).

END.

END METHOD.

END CLASS.
Demo

- Migration Routines for
  - Calculated Field source code
  - Validation Procedures
  - Test Business Entity / Calculated Fields in Business Entity Tester
  - Test Update and Validation using source code
  - Define RESTful Endpoint for the Business Entity
Define RESTful endpoints using Annotations

```java
@RestAddress (type="record", address="/MigratedSalesreps/{SalesRep}", tables="eSalesrep", id="SalesRep", fields="eSalesRep.*", canRead="true", canUpdate="true", canDelete="true").

@RestAddress (type="collection", address="/MigratedSalesreps", tables="eSalesrep", id="SalesRep", fields="SalesRep,RepName,Region,AnnualQuota,AverageQuota", canCreate="true").
```
```json
[{
  "id": "BBB",
  "url": "http://localhost:8820/web/Entities/MigratedSalesreps/BBB",
  "SalesRep": "BBB",
  "RepName": "Brawn, Bubba B.",
  "Region": "East",
  "AverageQuota": 2166.3333333333333,
  "AnnualQuota": 25996.0
},
{
  "id": "DKP",
  "url": "http://localhost:8820/web/Entities/MigratedSalesreps/DPK",
  "SalesRep": "DKP",
  "RepName": "Pitt, Dirk K.",
  "Region": "Central",
  "AverageQuota": 1973.5,
  "AnnualQuota": 23682.0
},
{
  "id": "DOS",
  "url": "http://localhost:8820/web/Entities/MigratedSalesreps/DOS",
  "SalesRep": "DOS",
  "RepName": "Donna",
  "Region": "Southern",
  "AverageQuota": 4578.25,
  "AnnualQuota": 54843.0
}]
```
Agenda

- Introduction / Application Modernization
- Modern OpenEdge Application Architecture
- ADM2 SDO migration
- TTY Upgrade editing migration
- ABL GUI migration
- OSIV3G Modernization example
Customer entry

Cust Num: 1
Name: Lift line skiing Ltd
Address: Unter Käster 1
Address2:
City: Köln
Postal Code: 50667
Country: USA
Sales Rep: DOS
Donna

Please enter the appropriate Postal Code.
UPDATE EDITING Blocks

DEFINE VARIABLE w-oldf AS CHARACTER NO-UNDO.

DO TRANSACTION:
   FIND CURRENT Customer EXCLUSIVE-LOCK .

   UPDATE {&ENABLED-FIELDS-IN-QUERY-DEFAULT-FRAME}
       WITH FRAME {&FRAME-NAME}
   blo-edit1:
       EDITING:

       READKEY.

       IF FRAME-FIELD <> "" THEN w-oldf = FRAME-FIELD. | APPLY LASTKEY.

       IF FRAME-FIELD <> w-oldf OR GO-PENDING THEN DO:
           HIDE MESSAGE.

       /* ********** begin validation code ********** */
Single field validation within EDITING Block

IF w-oldf = "Salesrep" OR GO-PENDING THEN DO:

    FIND Salesrep WHERE Salesrep.SalesRep = INPUT Customer.SalesRep
    NO-LOCK NO-ERROR .

    IF NOT AVAILABLE Salesrep THEN DO:
        MESSAGE SUBSTITUTE ("Please enter a valid salesrep code. &l is not a valid salesrep code. ",
                             INPUT Customer.Salesrep) .
        NEXT-PROMPT Customer.Salesrep WITH FRAME {&frame-name}.
        NEXT blo-edit1.

    END.

ELSE

    DISPLAY UPPER (Salesrep.SalesRep) @ Customer.SalesRep
    Salesrep.RepName WITH FRAME {&frame-name} .

END.
UPDATE EDITING Blocks

- Commonly used in TTY and early GUI applications
- Full of validation logic / Lookup functionality (locating foreign key descriptions)
- Tied to UI through “INPUT <fieldname>” references
- MESSAGE Statement used for error messages
- NEXT-PROMPT provides field that should receive input after error
- Record locked during duration of the UPDATE Statement
UPDATE EDITING Blocks

- Iterated for every keystroke or GO-PENDING
- When invoked on GO-PENDING, it’s similar to a commit to a Business Entity
  - Validating all fields at once
  - Processing update when no validation error occurred
  - Returning validation error to user (with instruction of next field)
- Code flow in EDITING Block very similar to typical Business Entity validation
Business Entity Validation based on UPD EDITING

IF eCustomer.CustomerName = "" THEN DO:
    Consultingwerk.Util.DatasetHelper:AddErrorString (BUFFER eCustomer:HANDLE,
    "Please enter customer name.",
    "CustomerName":U) .
END.

FIND Salesrep WHERE Salesrep.SalesRep = eCustomer.SalesRep
NO-LOCK NO-ERROR .

IF NOT AVAILABLE Salesrep THEN DO:
    Consultingwerk.Util.DatasetHelper:AddErrorString (BUFFER eCustomer:HANDLE,
    SUBSTITUTE ("Please enter a valid salesrep code. It is 
    "SalesRep":U) .
END.
ELSE
    ASSIGN eCustomer.SalesRep = UPPER (Salesrep.SalesRep)
    eCustomer.RepName = Salesrep.RepName .
NO-LOCK NO-ERROR .

IF NOT AVAILABLE Country THEN DO:
    Consultingwerk.Util.DatasetHelper:AddErrorString (BUFFER eCustomer:HANDLE,
    "Please enter a valid country name",
    "Country":U) .
END.
ELSE DO:
END .
Business Entity Validation based on UPD EDITING

- IF w-oldf OR GO-ENDING not required; Business Entity typically validates all fields at once
  - Removing at least one level of blocks in the code
- “INPUT <fieldname>” replaced with temp-table field reference
- DISPLAY statements replaces with update of temp-table field
- MESSAGE/NEXT-PROMPT statements replaced with API call to return validation message to the consumer of the Business Entity and control target field
Demo

- Proparse based migration of UPDATE EDITING Blocks into Business Entity Validation block
Agenda

- Introduction / Application Modernization
- Modern OpenEdge Application Architecture
- ADM2 SDO migration
- TTY Upgrade editing migration
- **ABL GUI migration**
- OSIV3G Modernization example
ABL GUI Migration

- Existing GUI (or TTY) screen layout may serve as a starting point for new UI’s
  - Highly dependent on UX of new application
  - Highly dependent on “quality” of layout of new application
<table>
<thead>
<tr>
<th>Cust Num</th>
<th>Name</th>
<th>Address</th>
<th>Credit Limit</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1,500</td>
<td>Net30</td>
</tr>
</tbody>
</table>

**Customer Information**

- **Cust Num**: 0
- **Address**:  
- **Address2**:  
- **City**:  
- **Postal Code**:  
- **Country**: USA
- **State**:  
- **Email**:  
- **Fax**:  
- **Phone**:  
- **Sales Rep**:  
- **Comments**:  

**Financial Details**

- **Balance**: 0.00
- **Discount**: 0%
Screen layout migration

- Screen layout from static code can be refactored based on Proparse
  - FRAME definitions sometimes tricky to understand
  - Multiple FRAME Statements for a single FRAME
  - VIEW-AS phrase from Data Dictionary
  - Default properties of widgets
- Walking the widget tree typically simpler – however this requires changes to application runtime and is not trivial when building general purpose tools
Abstract view on screen layout
Abstract view on screen layout

- Allows generation of various UI’s
  - GUI for .NET
  - Angular
  - Kendo UI Builder
  - Meta-Data for UI repository database
  - …
GUI Trigger Code

- Typically used for validation or control of the UI
- Contains references using widget attributes (:SCREEN-VALUE or :SENSITIVE, etc.) or INPUT <fieldref>
- May contain business logic that should be moved to Business Entity (typically when accessing DB records), LEAVE Triggers typical prospect for validation
DO:  

| DO WITH FRAME {iframe-name}; |
| IF Customer.Country:SCREEN-VALUE = "USA" THEN |
| Customer.State:VISIBLE = TRUE . |
| ELSE |
| Customer.State:VISIBLE = FALSE . |
| END. |
| END. |
Migrated Trigger Code


/* Trigger code from ON VALUE-CHANGED OF sports2000.Customer.Country IN FRAME DEFAULT-FRAME
C:\Work_STREAM\SmartComponentLibrary\Develop\ABL\Demo\c-customer.w - 30.05.2018 13:09:24 */

DEFINE VARIABLE Customer_State AS Consultingwerk.Windows.LegacyGuiMigration.Widgets.IWidgetFacade NO-UNDO.


DO /* WITH FRAME DEFAULT-FRAME */:

    IF Customer_Country:SCREEN-VALUE = "USA" THEN
        Customer_State:VISIBLE = TRUE .
    ELSE
        Customer_State:VISIBLE = FALSE .

END.

END METHOD.

Widget Façade classes allow mapping of widget attributes to control properties
Agenda

- Introduction / Application Modernization
- Modern OpenEdge Application Architecture
- ADM2 SDO migration
- TTY Upgrade editing migration
- ABL GUI migration
- **OSIV3G Modernization example**
OSIV / OSC

- OSIV Service Center: Joint venture of 7 Swiss counties (cantons)
- Maintaining state insurance for occupational disabilities
- Approval of therapies
- Perform Disability and treatment Assessments
- Billing (by doctors, clinics, opticians, occupational disabilities, etc.)
- Document management
- 1300 users
- Very specific domain functionality
- Accepted by the user base, no real competition
Why “refactoring”

- Maintenance effort high
- Training of new users and developers hard
- Aged technology
- Resources / Motivation of developers / Agile methods
OSIV3G: Soft Migration

Harvesting of existing Code

Current OSIV 5.x → OSIV-DB + additional Fields and Tables e.g. GUID’s → Migrated Application Modules → Framework DB's
Example challenge: Interaction between Back and Frontend

- Existing OSIV Business Logic in large parts suitable as foundation for new OSIV3G (functional and structural), especially validation
- Validation may also provide color coding to represent field status etc.
- Validation may have to prompt the user
- Web applications typically:
  - Request (from browser) – Response (from server)
- No Input-Blocking (not possible to wait for user input in Business Logic)
Sample: Yes/No PROMPT in validation

- Demand is to keep the validation flow in major parts „as is“
- Validation may encounter question requiring user input: “Are you sure?” etc.
Sample: Yes/No PROMPT in validation

/* ----------- */
/* Verstorben */
/* ----------- */

if (date(Stamm.Todes_Dat:screen-value) <> ?) then do:
    /* Testen, ob Versicherter gerade eben verstorben ist. */
    if (EDIT_MODE = "UPDATE") then do:
        find Stamm no-lock where recid(Stamm) = MAIN_REC_ID.
        if (Stamm.Todes_Dat = ?) then do:
            /* Versicherter wurde soeben auf verstorben gesetzt. */
            run set_message_param(Stamm.Todes_Dat:screen-value).
            run user_warning("Der Versicherte ist am $1 verstorben. ~n~n" +
                "Die zugehörigen Wohnadressen werden gesperrt.~n" +
                "Überprüfen Sie, ob noch Revisionen vorgesehen sind~n" +
                "und/oder Hilfsmittel zurückgenommen werden müssen.~n", output continue).
            if not continue then return error.
        end.
    end.
end. /* if verstorben */
Sample: Yes/No PROMPT in validation

MSG = {Consultingwerk/get-service.i IMsg}.
SYS = {Consultingwerk/get-service.i ISys}.
MOD_ADD = {Consultingwerk/get-service.i IModAdd}.

if (eStammBefore.Todes_Dat = ?) then do:
   /* Versichelter wurde soeben auf verstorben gesetzt. */
   MSG:set_message_param(string (eStamm.Todes_Dat) /*:screen-value*/).

   continue = MSG:user_warning("Der Versicherte ist am $1 verstorben. ~n~n" +
   "Die zugehörigen Wohnadressen werden gesperrt.~n" +
   "Überprüfen Sie, ob noch Revisionen vorgesehen sind~n" +
   "und/oder Hilfsmittel zurückgenommen werden müssen.~n",
   this-object:GetClass():TypeName,
   "eb09af84b1e2197b:4cb274e8:15608162bb6:-8000",
   string (eStamm.SelfHdl)).

   if not continue then do:
      DatasetHelper:AddErrorString(buffer eStamm:handle, "_CANCEL") .
      return .
   end.

   /*if not continue then return error.*/
end.
Migration using MessagePrompt API (SCL)

- Backend – API maintains list of questions (unanswered and answered)
- Same API Call may ask a new question or return an existing answer
- Supports multiple questions per routine: Questions are flagged with GUID identifying this location in code
- Support for multiple iterations (Loops, FOR EACH, …): Each question is also flagged with a return PUK value
Migration using MessagePrompt API (SCL)

- Questions will be returned to UI in a standard temp-table field
- Current Update-Request will be cancelled (typically before the DB transaction is started)
- UI presents unanswered questions to the user and repeats the same update request
- Repeat this flow if additional questions are required
JSON Representation of the question

```
1
3
4  "MessageText": "Der Versicherte ist am 24/12/50 verstorben. 
5  Die zugehörigen Wohnadressen werden gesperrt. 
6  Überprüfen Sie, ob noch Revisionen vorgesehen sind 
7  und/oder Hilfsmittel zurückgenommen werden müssen. 
8  "MessageButtons": "YesNo",
9  "MessageReply": "Unanswered",
10  "DefaultReply": "ReplyYes",
11  "MessageID": "eb09af84b1e2197b:4cb274e8:15608162bb6:-8000",
12  "MessageContext": "ac54bf82-56c4-bab2-2514-8e3d5c34775d"
```
Questions