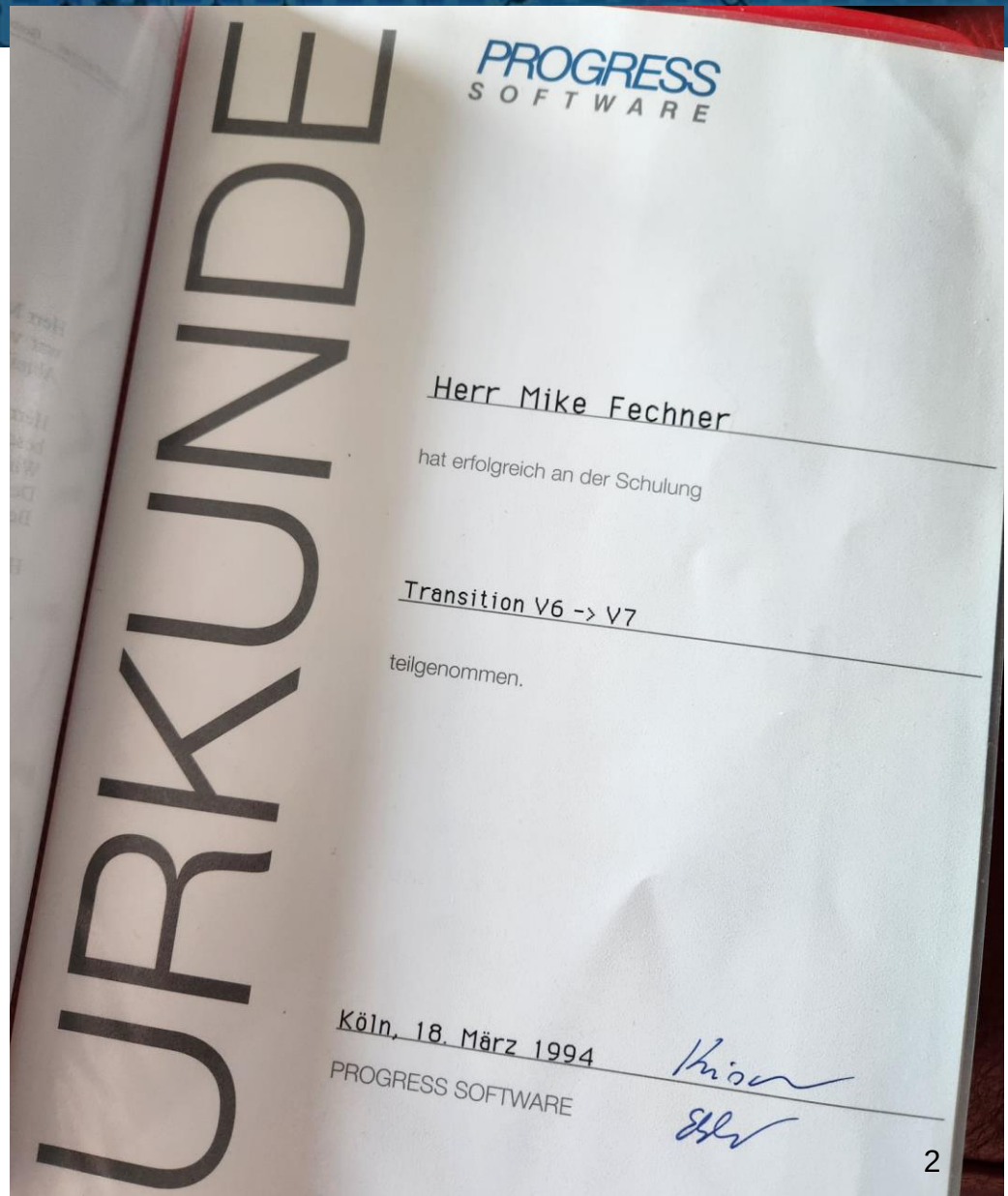
Consultingwerk
software architecture and development

ABL legacy code refactoring Patterns and Strategies

Mike Fechner
mike.fechner@consultingwerk.de

Mike Fechner

- Director, Lead Modernization Architect and Product Manager, Architect of the SmartComponent Library and WinKit
- Specialized on object-oriented design, software architecture, desktop user interfaces and web technologies
- 30 years of Progress experience (V5 … OE12)
- Active member of the OpenEdge community
- Frequent speaker at OpenEdge related conferences around the world



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Consultingwerk Software Services Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and **related technology**
- Located in Cologne, Germany, subsidiaries in UK, USA 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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Services Portfolio, Progress Software

- OpenEdge (ABL, Developer Tools, Database, PASOE, …)
- Telerik DevCraft (.NET, Kendo UI, Angular, …), Telerik Reporting
- OpenEdge UltraControls (Infragistics .NET)
- Telerik Sitefinity CMS (incl. integration with OpenEdge applications)
- Kinvey Plattform, NativeScript
- Corticon BRMS
- WhatsUp Gold infrastructure-, network- and application monitoring
- Kemp Loadmaster
- …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Services Portfolio, related products

- Protop Database Monitoring
- Combit List & Label
- Web frameworks, e.g. Angular
- .NET
- Java
- ElasticSearch, Lucene
- Amazon AWS, Azure
- DevOps, Docker, Jenkins, ANT, Gradle, JIRA, …
- …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Agenda

- **Modernization Process**
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- Proparse
- User interface refactoring

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Modernization drivers

- The obvious: a new user interface
  - Web interface
  - Modern desktop UI
  - Mobile or satellite applications

- Functional requirements
  - Integration with 3rd party applications (in and out)
  - Localization
  - Hard to keep up with new features
  - Redundancy kills agility

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Modernization drivers

- Improved code quality / maintainability
  - Improvements to application longevity
  - Component independency
  - Module independence
  - Method length
  - Test driven development to improve quality and agility
- Get ready for a new generation of software developers
  - Foreseeable retirement of key developers
  - Need to make application attractive to young developers
  - Enable application for distributed development

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Modernization drivers

- Modernization drivers need agreement between all stake-holders
  - development team
  - business
- When time-pressure comes, goals not directly visible to end users may otherwise be sacrificed
  - code-optimization
  - adherence to architectural standards
  - test-driven-design
  - technical documentation

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Know your constants

- I expect that OpenEdge and PASOE will still be around 10 years from now
- I expect that OpenEdge will keep fundamentally backwards compatible with todays source-code
- Majority of application functionality should be moved to PASOE

- I will not even try to foresee the trends in user-interface technology in the next few years

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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
  - Reduce risk, gather first experience

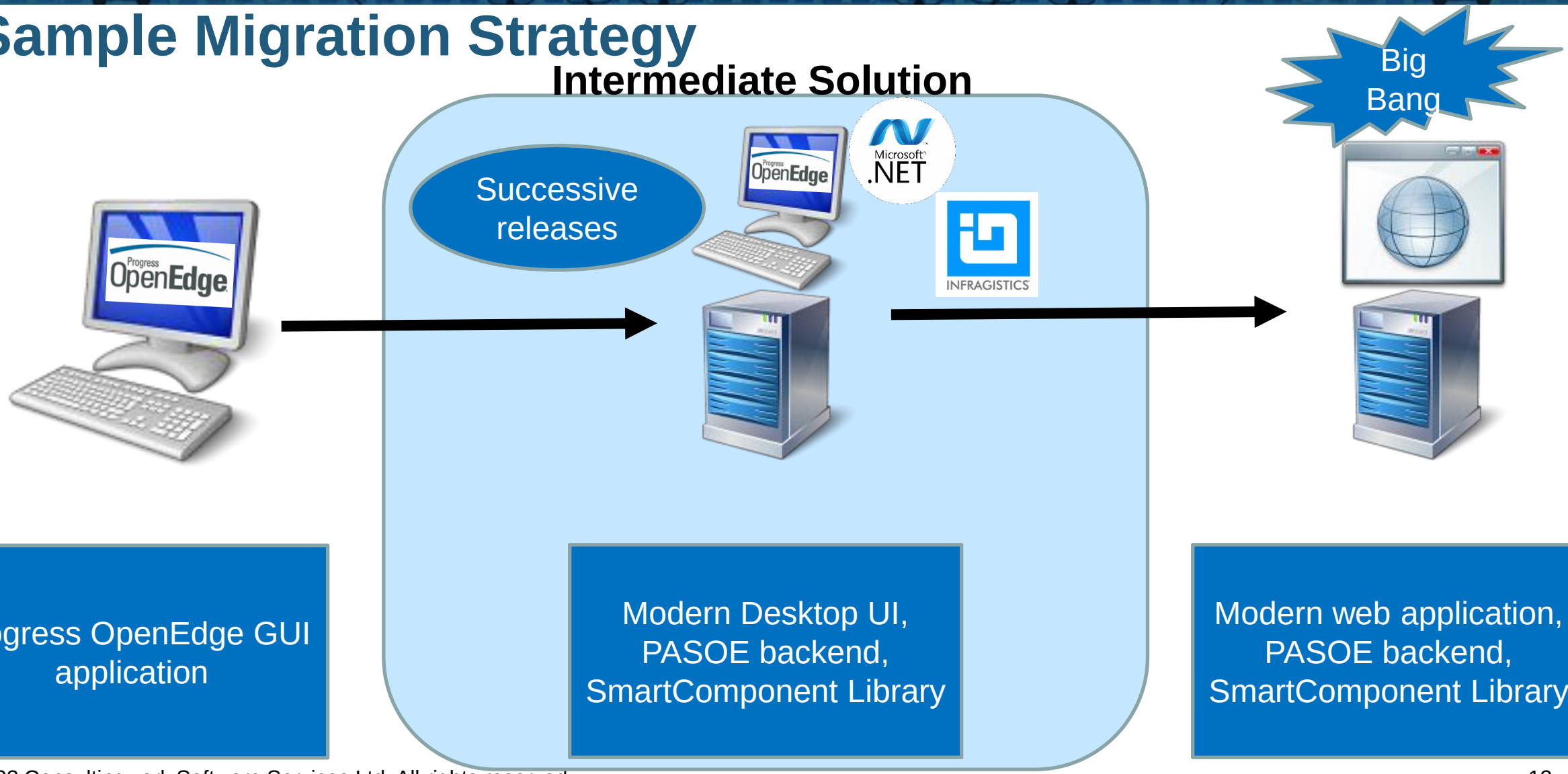
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Modernization Strategies

- Modernizing OpenEdge GUI (or TTY) to N-Tier first
  - Preparing the Application Backend for the Web
- Modernizing the whole application to the Web
  - Driven by demands from? Users, IT organization, marketing?
  - Definition of MVP – minimum viable product
  - How much functionality must be delivered on the web for user acceptance
- Developing one or multiple “satellite” web applications
  - Deliver quick and with reduced risk

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Sample Migration Strategy



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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?

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Development Team Skills

- New development process (agile)
- New tools (Progress Developer Studio, SCM, Unit Tests, DevOps, Docker, Frontend tools) and Frameworks
- New architecture: Distributed
- New development languages
  - OOABL
  - HTML, JavaScript, TypeScript, rapidly changing
  - Desktop technologies
  - Web and Mobile frameworks

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Agenda

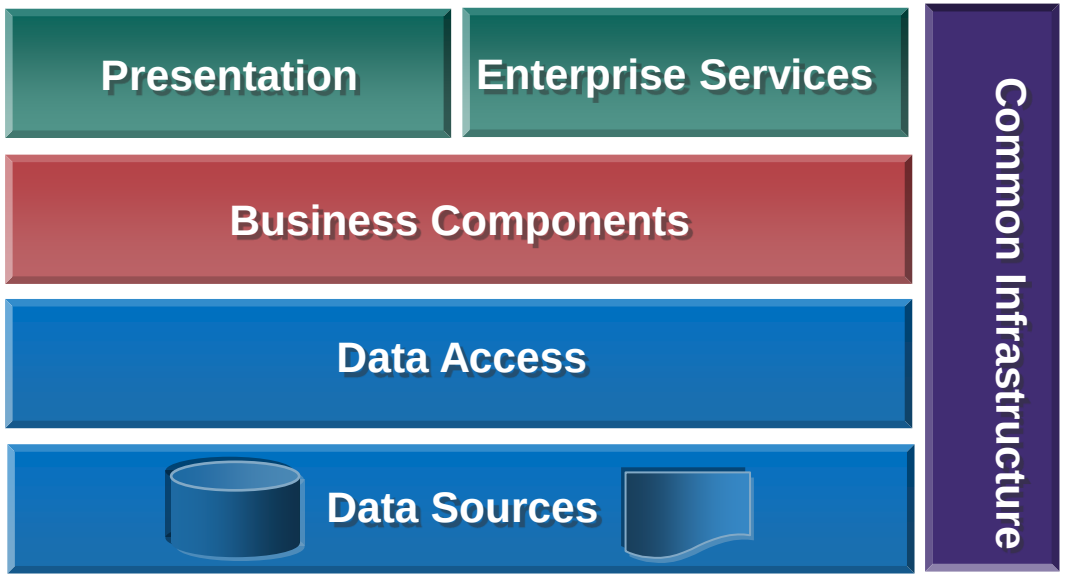
- Modernization Process
- **Application Architecture**
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- Proparse
- User interface refactoring

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

The OpenEdge Reference Architecture (OERA)

The OpenEdge Reference Architecture (OERA) defines the general functional categories of components that comprise an application. It can be used as a high-level blueprint for developing OpenEdge service-oriented business applications.

Each layer of the OERA consists of distinct components, each with specific characteristics, roles and responsibilities. In addition, the OERA provides guidelines as to how each of the architectural components interacts. The following diagram illustrates the component architecture and the relationships between each of the components.



https://community.progress.com/s/question/0D54Q0000819wkqSAA/introduction-to-the-openedge-reference-architecture

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

The OpenEdge Reference Architecture (OERA)

- Focus is on high-level architecture "blueprint"
- OERA is not prescriptive …
  - Choose to use procedural or OOABL code
  - Choose to implement some or all layers
  - Choose to keep existing code
- Service Interface Layer almost entirely ignored
- No guidance given on implementation, other than sample code

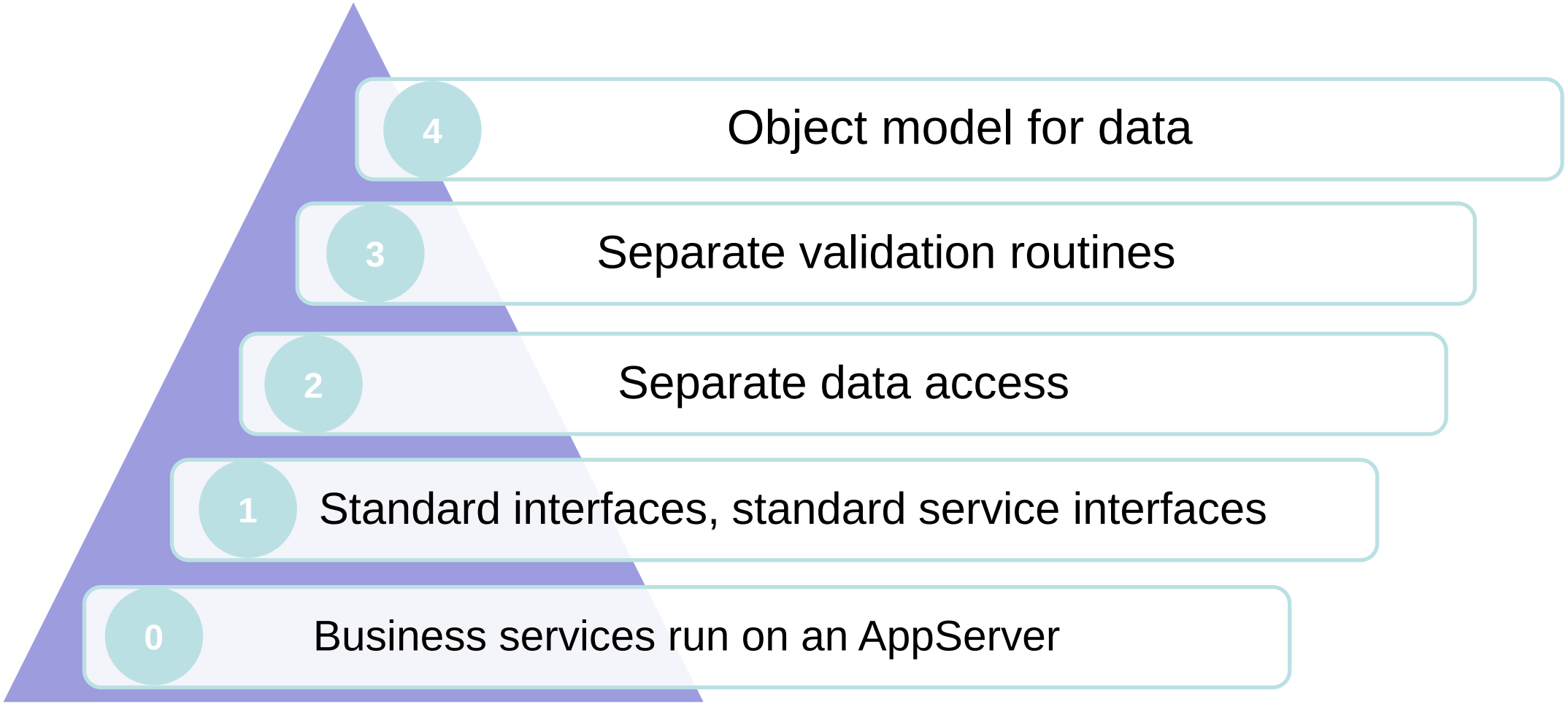
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

The Consultingwerk OERA Maturity Model

- An *opinionated* attempt to give application architects and developers some orientation in how to implement OERA compliant ABL applications
  - E.g. We don't believe a Data Source layer provides value
- Assumes that different developer teams have different requirements and expectations for the architecture and coding style of modernized ABL applications
- Builds upon the OERA and is focused on *implementation*, primarily relating to the Business Components and Data Access layers
- Soon on https://www.consultingwerk.com/news/blog

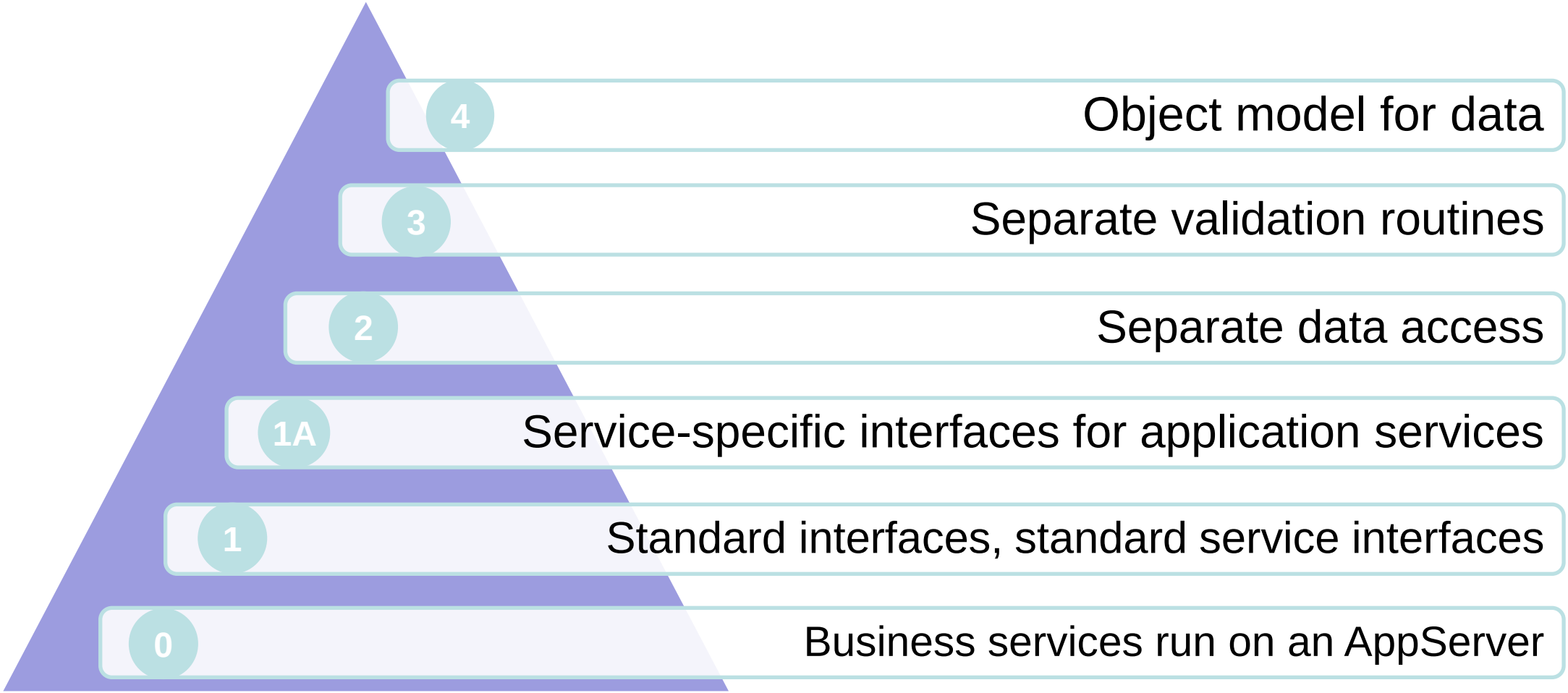
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

The Consultingwerk OERA Maturity Model



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

The Consultingwerk OERA Maturity Model



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Consequences

- ✓ More, smaller programs
  - Programs better follow Single Responsibility Principle / Separation of Concerns
  - Unit testing made easier
- ✓ OOABL ***strongly*** recommended
- ✓ Pattern-based development
  - E.g. fetching data always looks the same, regardless of which business service's data is fetched
- Reuse of existing code typically reduced as a result

- Increasing developer productivity via
  - ✓ Compiler help with OOABL
  - ✓ Standardized development approaches
  - ✓ Reduced merge pain
  - ✓ Smaller programs, smaller impact from changes
  - ✓ Fewer "God programs"

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Reduced reuse of legacy code?

- Implementing full separation of concerns can mean it's much harder to reuse existing blocks of code
  - Reusing large parts of existing code promises faster migration process
  - Existing unit- and system tests can continue to be used when reusing legacy code

- Find the balance between migration speed and risk reduction, and future-proofing and increased maintainability

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Service Interface(s)

Bouncer

Babelfish

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Service Interface(s)

- The Service Interface receives calls from clients or external consumers
- A very important and often under-appreciated component
  - The Service Interface is responsible for Validating the request (including Authentication and Authorization)
  - Ensuring the User-Session is in the correct state
  - Allocating the service (the Application Service, Business Task or Entity)
  - Converting the request data from an external format to internal
  - Converting the response data from internal format to external

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Service Interface(s)

- Business Logic is the most valuable piece of the application
- User interfaces come and go (TTY, ABL GUI, GUI for .NET, Web, Mobile, Chat, …)
- We do not want to rewrite – or even change the Business Logic for every new UI trend
- Multiple parallel used UI technologies should be using the same Business Logic
  - When there are very specific requirements for a single UI (e.g. Wizard style vs. plain data entry, consider using Application Service for this as an aggregate of multiple Business Tasks or Entities)

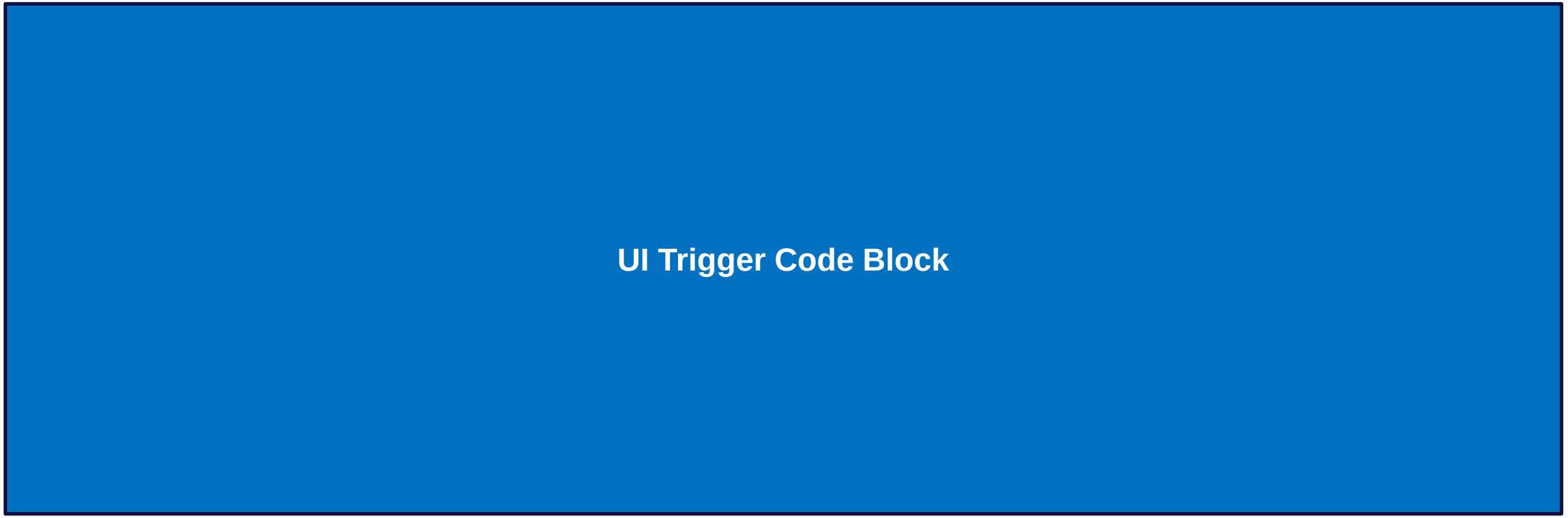
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Top-down code-generalization

- Existing code considered to be closest to an application service
- First step is moving code from UI into an application service
- Simplifies automation during code-refactoring (almost statement by statement replaces)
- Further steps will improve code-reuse and single-concern by extracting code from application service into domain services
- Code de-duplication requires more design and guidelines
  - Into how many pieces do we cut the monolith?

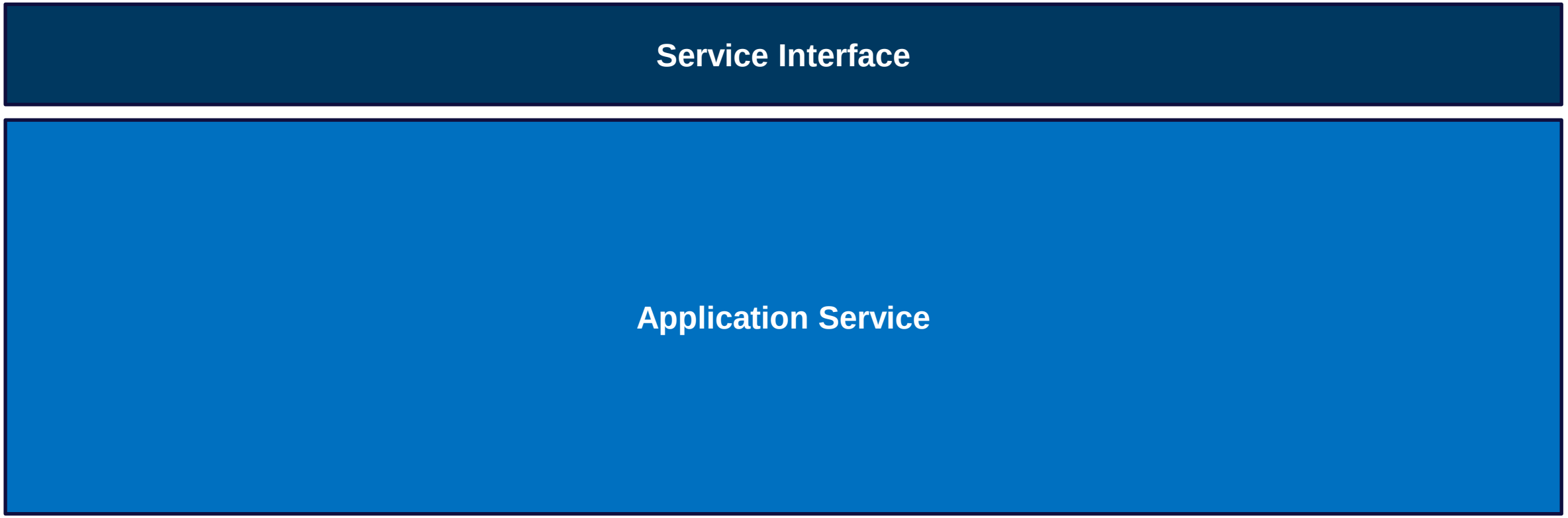
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Top-down code-generalization



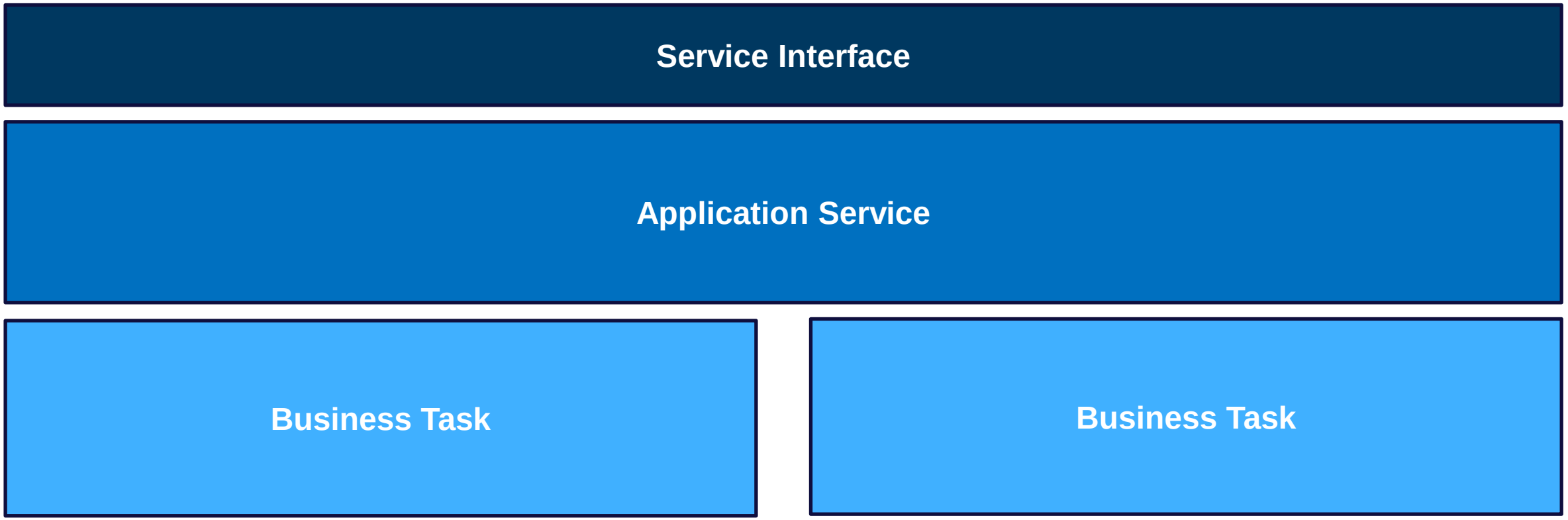
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Top-down code-generalization



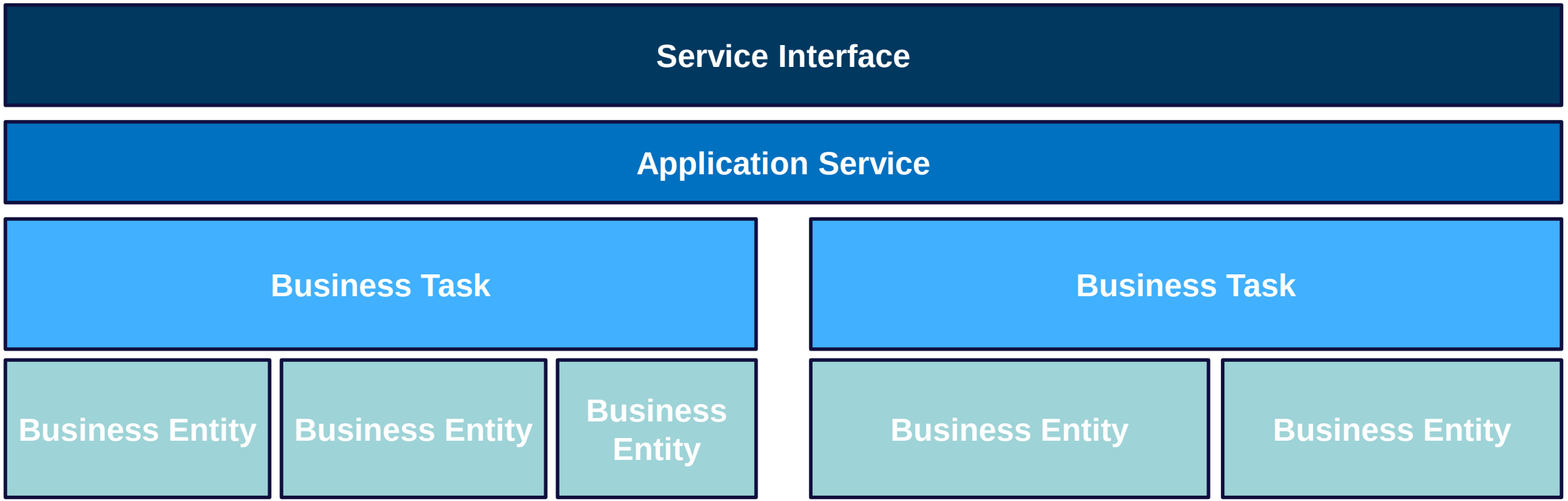
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Top-down code-generalization



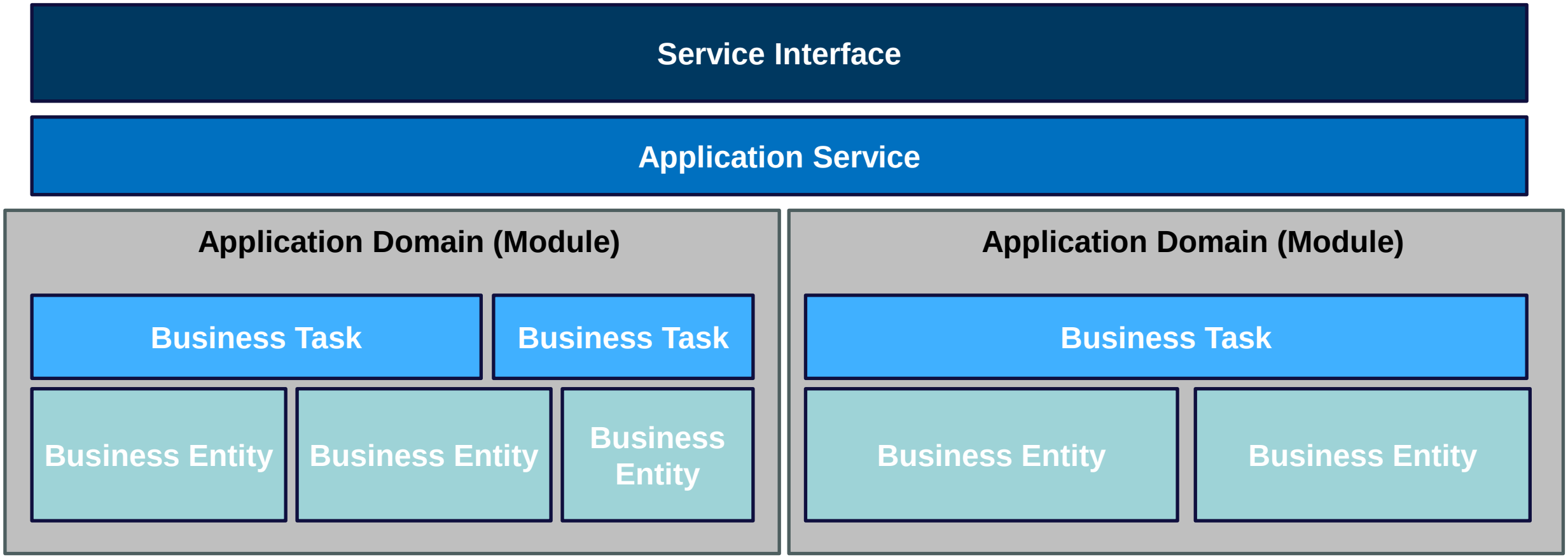
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Top-down code-generalization



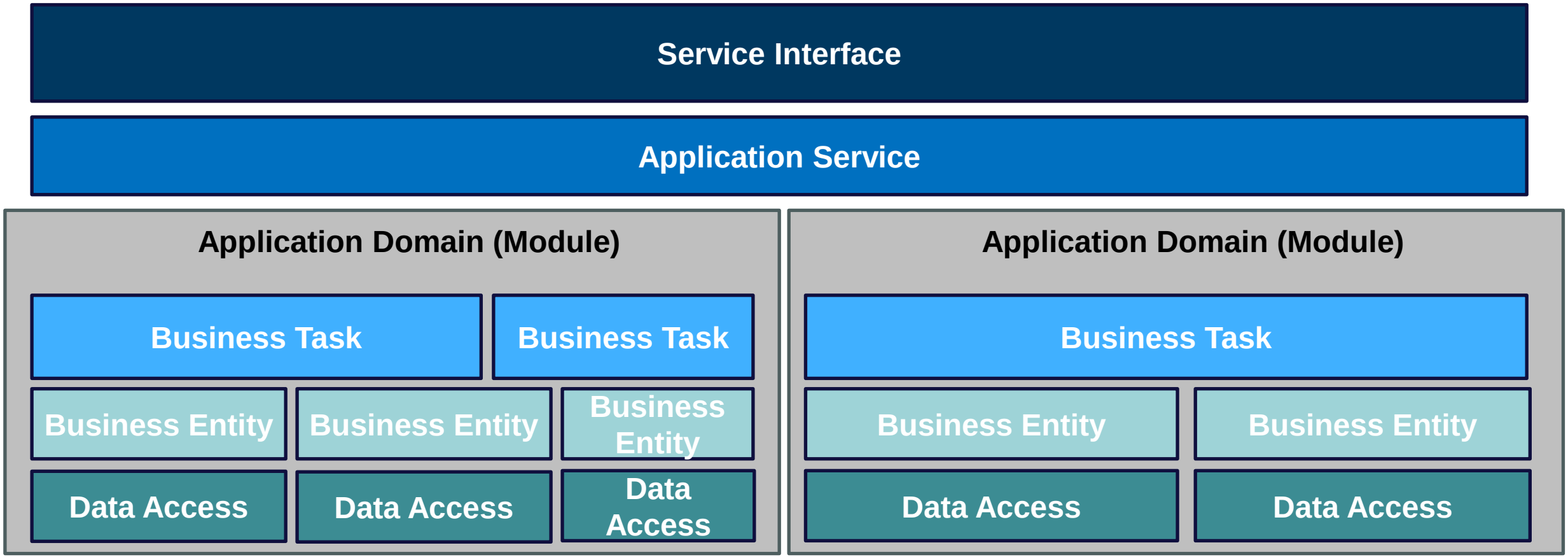
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Top-down code-generalization



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Top-down code-generalization



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Demo

- Review different stages of migrating the „ship order“ trigger code
- Move out of GUI - into AppServer ready code
- Deal with questions using „MessageInteractionService“
- Throw Errors as Exception
- Introduce PreCheck method to reduce impact of undo’ing transactions on pending questions

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

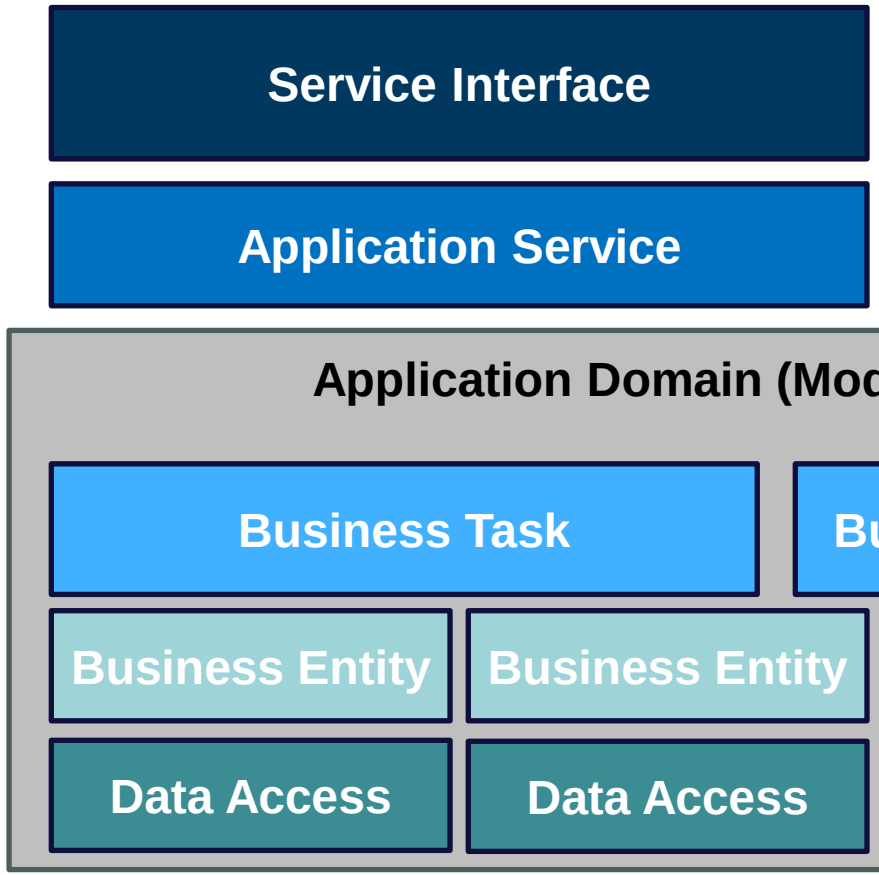
Benefits of top-down code-generalization

- First introduce service-ready component based on existing business logic
- Hide implementation details behind service interface
- Flow of business logic remains largely the same – **this will reduce risk**
- Component interface will allow
  - Use in modern user-interfaces
  - Implementation of unit-tests
- Unit tests will improve confidence when optimizing the code

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Aspects of Top-Down code generalization

- Business Tasks and Business Entities should only deal with “their concern”
- Use factories or service managers – never directly new any application or domain business service object
- Only “allow” calls from top to bottom
- Services within a domain may call each other
- Services across domain boundary should use domain service interface



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Further considerations

- Use parameter objects
- Separate screen-context from request parameters
- Selected warehouse may be screen context (might be a screen setting)
- Screen-context might be modified in UI and backend
- Selected order may be request-context (it’s the subject of ship order)

- Variables defined in the “definitions section” vs. parameters to internal procedures

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Agenda

- Modernization Process
- Application Architecture
- **Dealing with (GLOBAL) SHARED Variables**
- Dealing with messages or prompts
- Proparse
- User interface refactoring

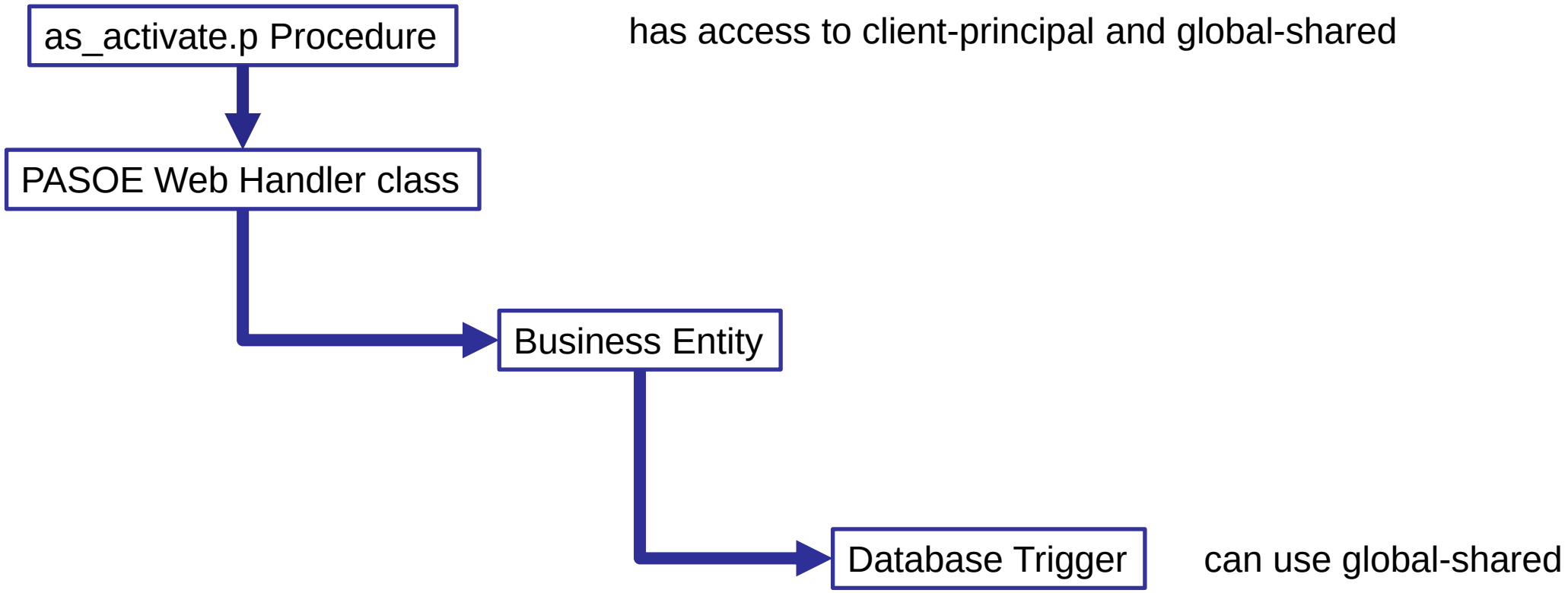
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

GLOBAL SHARED or SHARED variables …

- GLOBAL SHARED variables are less trouble
- SHARED variables should be reconsidered – many of them may be replaced with GLOBAL SHARED, usually a bad legacy
- Class based code (most new code, PASOE Web handlers) has NO access to any GLOBAL SHARED SHARED context

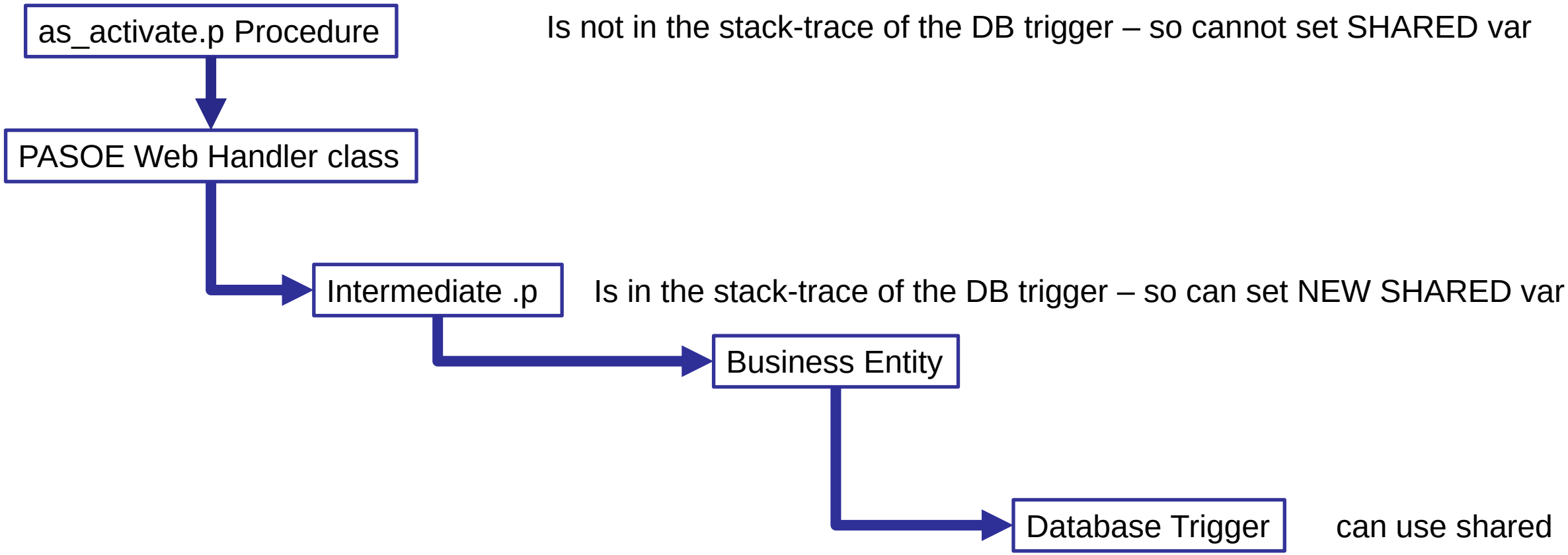
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

DB Trigger relying on a GLOBAL SHARED variable



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

DB Trigger relying on a SHARED variable



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Agenda

- Modernization Process
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- **Dealing with messages or prompts**
- Proparse
- User interface refactoring

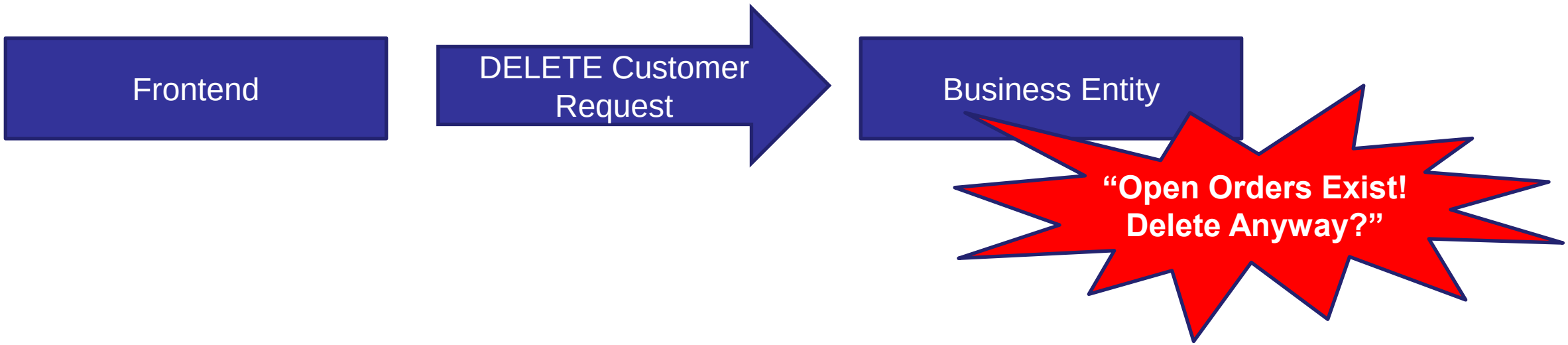
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Input blocking from the Backend

- Progress Application Server does not support Input Blocking on the UI
- Once AppServer is invoked, client waits for response
- Web technologies such as Socket.IO may be used to send messages from Backend to frontend
  - Back not vice-versa, no WAIT-FOR
- **When UI can foresee** that AppServer **may** require additional information when processing request, try adding this to the request
  - However UX should not be ignored. Too many irrelevant options confusion / annoying to users

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Input Blocking, fat client ABL




```
IF CAN-FIND (FIRST Order OF Customer WHERE ....) THEN
    MESSAGE "Open Orders Exist! Delete Anyway?"
        VIEW-AS ALERT-BOX QUESTION BUTTONS YES-NO UPDATE response .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Input Blocking, fat client ABL


```
IF CAN-FIND (FIRST Order OF Customer WHERE ....)
   AND poRequest:DeleteOpenOrders = TRUE THEN …
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Example challenge: Interaction between Back and

- Assumption: Existing Business Logic in large parts suitable as foundation for new application (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)

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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.

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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 */
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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:
    /* Versicherter 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.
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Migration using MessageInteractionService API (SmartComponent Library framework)

- 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 e.g. a GUID identifying their location in code
- Support for multiple iterations (Loops, FOR EACH, …): Each question is also flagged with a records PUK value (GUID, combined key fields)

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

JSON Representation of the question

```
{
  "SerializedType": "Consultingwerk.Framework.MessageInteraction.Question",
  "MessageText": "Der Versicherte ist am 24\/12\/50 verstorben. \n\n
                  Die zugehörigen Wohnadressen werden gesperrt.\n
                  Überprüfen Sie, ob noch Revisionen vorgesehen sind\n
                  und\/oder Hilfsmittel zurückgenommen werden müssen.\n",
  "MessageButtons": "YesNo",
  "MessageReply": "Unanswered",
  "DefaultReply": "ReplyYes",
  "MessageID": "eb09af84b1e2197b:4cb274e8:15608162bb6:-8000",
  "MessageContext": "ac54bf82-56c4-bab2-2514-8e3d5c34775d"
}
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Automation

- Migration of MESSAGE Statements into API calls can be automated using Proparse based tooling

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Agenda

- Modernization Process
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- **Proparse**
- User interface refactoring

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

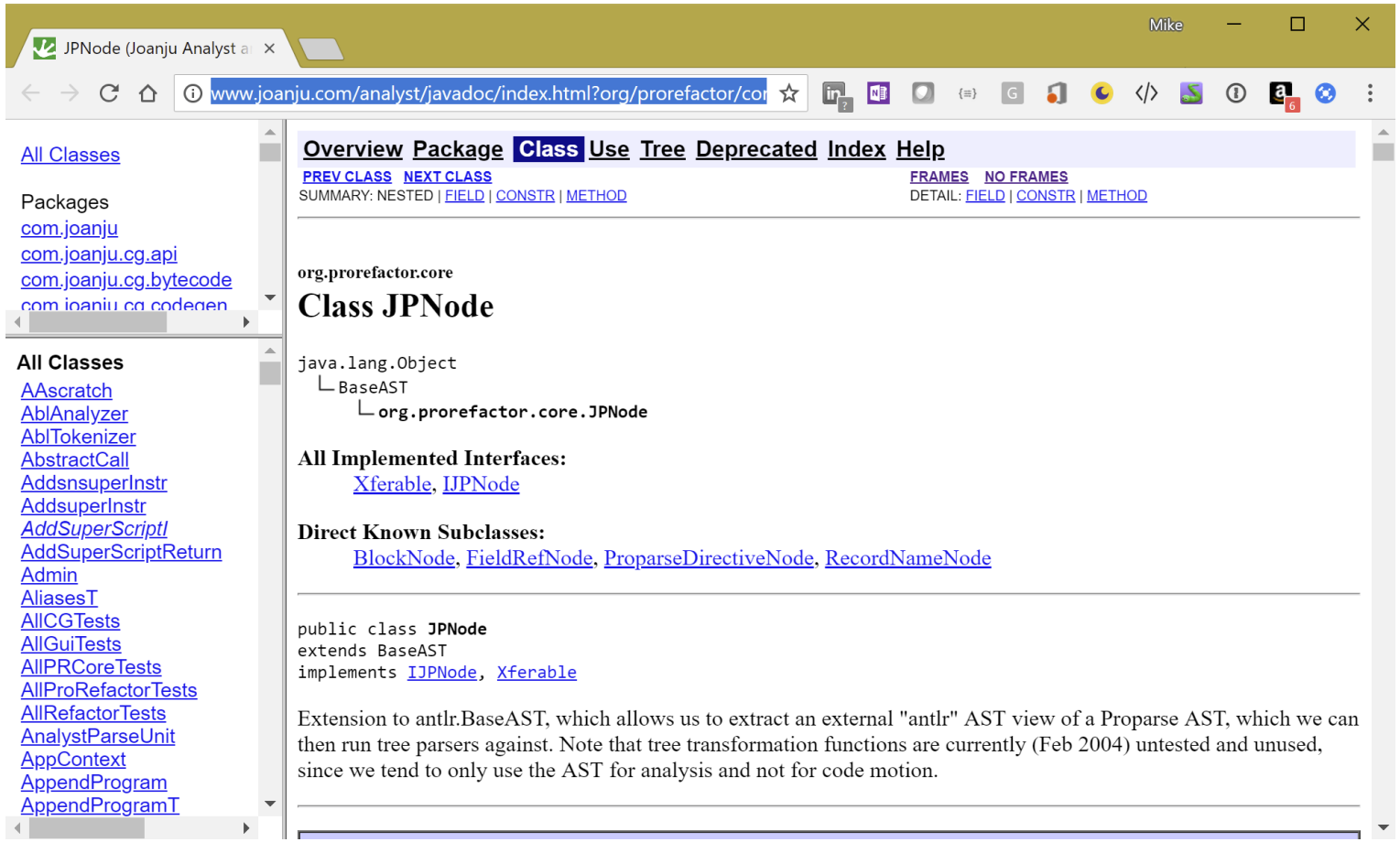
Source code parsing using Proparse

- 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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Proparse

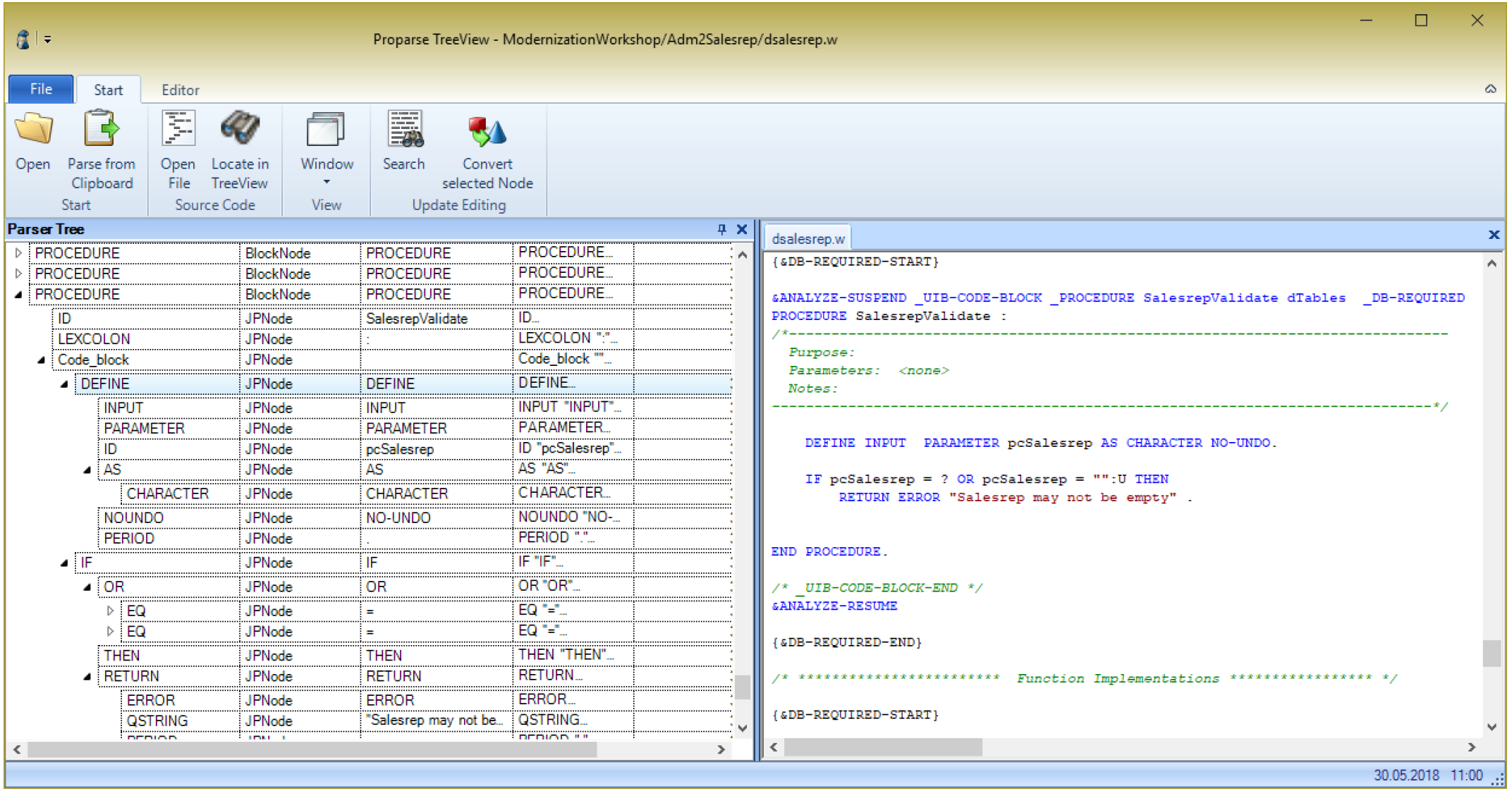
- http://www.joanju.com/analyst/javadoc/index.html?org/prorefactor/core/JPNode.html



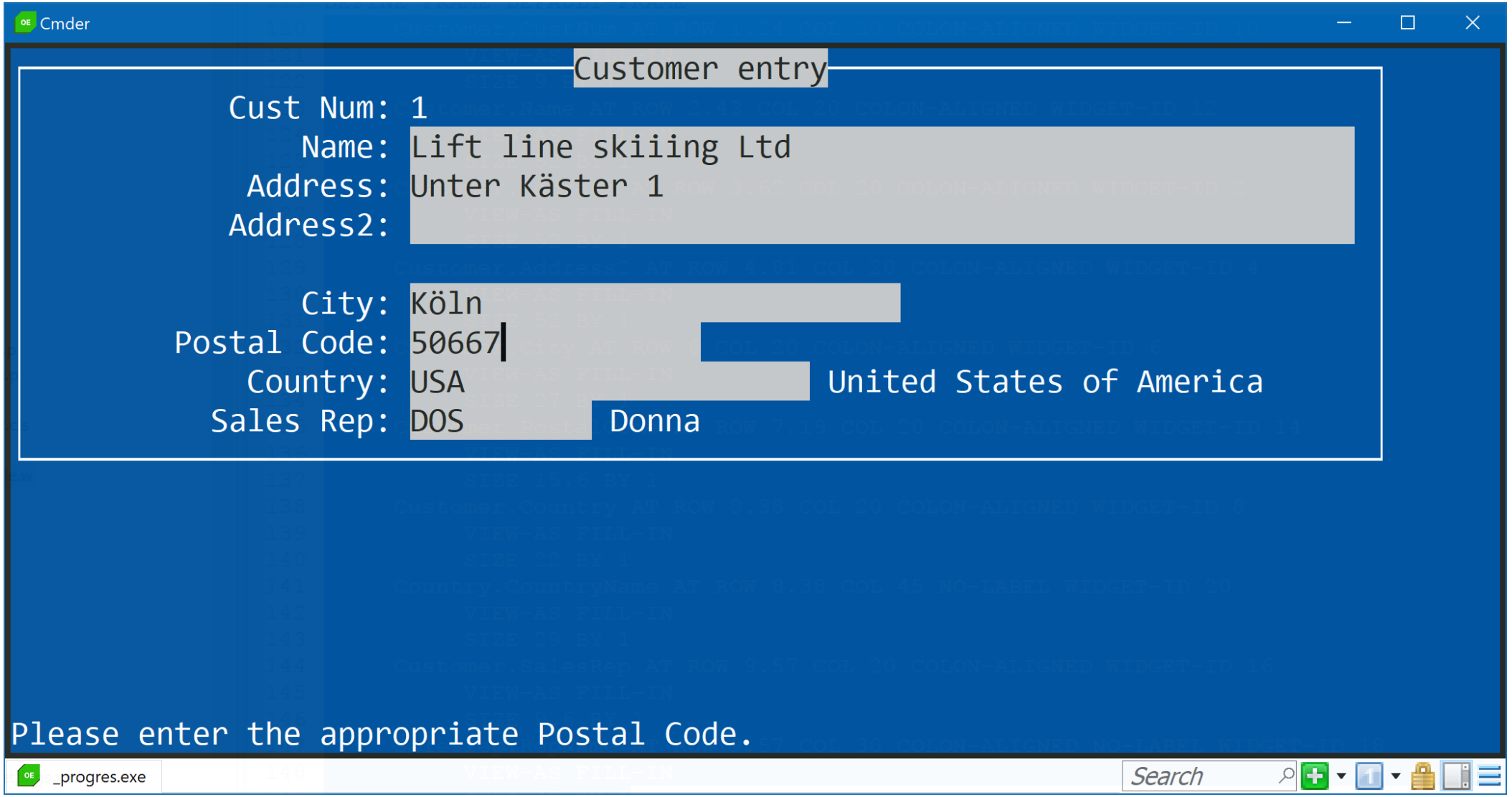
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

```
javafile = NEW java.io.File (pcFilename).

IF (NOT javafile:exists()) THEN
    UNDO, THROW NEW FileNotFoundException (pcFileName,
                                           SUBSTITUTE ("Could not find file: &1."{&TRAN}, pcFileName),
                                           0) .

IF cProparseCodepage > "" THEN DO:
    IF NOT Codepages:IsKnownCodepage (cProparseCodepage) THEN
        UNDO, THROW NEW InvalidValueException (cProparseCodepage, "ProparseCodepage":U) .

    oParseUnit = NEW ParseUnit(javafile, cProparseCodepage).
END.
ELSE
    oParseUnit = NEW ParseUnit(javafile).

pu:treeParser01().

DELETE OBJECT javafile .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.



© 2023 Consultingwerk Software Services Ltd. All rights reserved.



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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 ********** */
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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. &1 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.
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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. &1 is
                                                       "SalesRep":U) .
END.
ELSE
    ASSIGN eCustomer.SalesRep = UPPER (Salesrep.SalesRep)
           eCustomer.RepName = Salesrep.RepName .

FIND Country WHERE Country.Country = eCustomer.Country
    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:
    ASSIGN eCustomer.Country = Country.Country .
    ASSIGN eCustomer.CountryName = Country.CountryName .
END .
```

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Demo

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Agenda

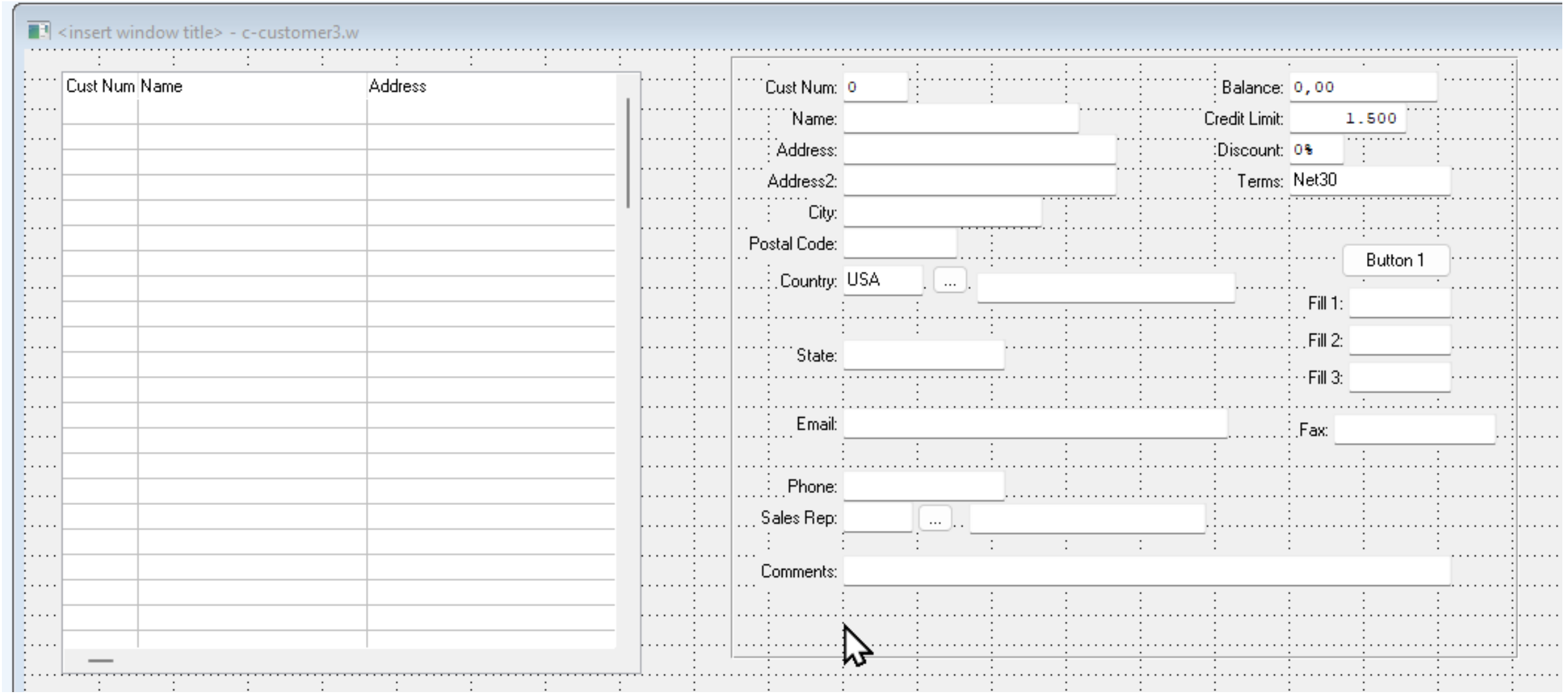
- Modernization Process
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- Proparse
- **User interface refactoring**

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

ABL GUI refactoring

- 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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.



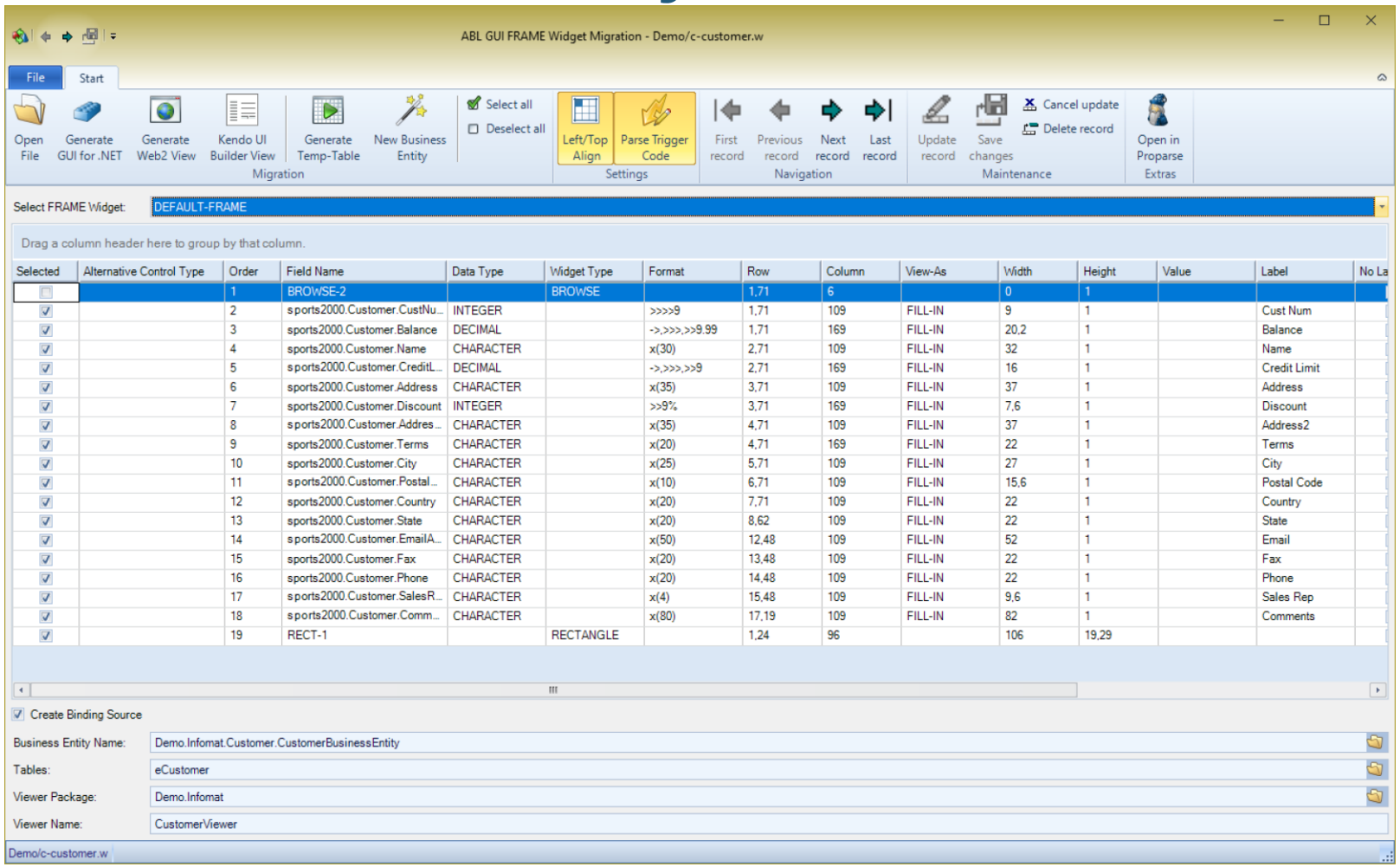
© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Abstract view on screen layout



Selected	Alternative Control Type	Order	Field Name	Data Type	Widget Type	Format	Row	Column	View-As	Width	Height	Value	Label	No La
☐		1	BROWSE-2		BROWSE		1,71	6		0	1			
☑		2	sports2000.Customer.CustNu...	INTEGER		>>>>9	1,71	109	FILL-IN	9	1		Cust Num	
☑		3	sports2000.Customer.Balance	DECIMAL		->,>>>,>>9.99	1,71	169	FILL-IN	20,2	1		Balance	
☑		4	sports2000.Customer.Name	CHARACTER		x(30)	2,71	109	FILL-IN	32	1		Name	
☑		5	sports2000.Customer.CreditL...	DECIMAL		->,>>>,>>9	2,71	169	FILL-IN	16	1		Credit Limit	
☑		6	sports2000.Customer.Address	CHARACTER		x(35)	3,71	109	FILL-IN	37	1		Address	
☑		7	sports2000.Customer.Discount	INTEGER		>>9%	3,71	169	FILL-IN	7,6	1		Discount	
☑		8	sports2000.Customer.Addres...	CHARACTER		x(35)	4,71	109	FILL-IN	37	1		Address2	
☑		9	sports2000.Customer.Terms	CHARACTER		x(20)	4,71	169	FILL-IN	22	1		Terms	
☑		10	sports2000.Customer.City	CHARACTER		x(25)	5,71	109	FILL-IN	27	1		City	
☑		11	sports2000.Customer.Postal...	CHARACTER		x(10)	6,71	109	FILL-IN	15,6	1		Postal Code	
☑		12	sports2000.Customer.Country	CHARACTER		x(20)	7,71	109	FILL-IN	22	1		Country	
☑		13	sports2000.Customer.State	CHARACTER		x(20)	8,62	109	FILL-IN	22	1		State	
☑		14	sports2000.Customer.EmailA...	CHARACTER		x(50)	12,48	109	FILL-IN	52	1		Email	
☑		15	sports2000.Customer.Fax	CHARACTER		x(20)	13,48	109	FILL-IN	22	1		Fax	
☑		16	sports2000.Customer.Phone	CHARACTER		x(20)	14,48	109	FILL-IN	22	1		Phone	
☑		17	sports2000.Customer.SalesR...	CHARACTER		x(4)	15,48	109	FILL-IN	9,6	1		Sales Rep	
☑		18	sports2000.Customer.Comm...	CHARACTER		x(80)	17,19	109	FILL-IN	82	1		Comments	
☑		19	RECT-1		RECTANGLE		1,24	96		106	19,29			

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Abstract view on screen layout

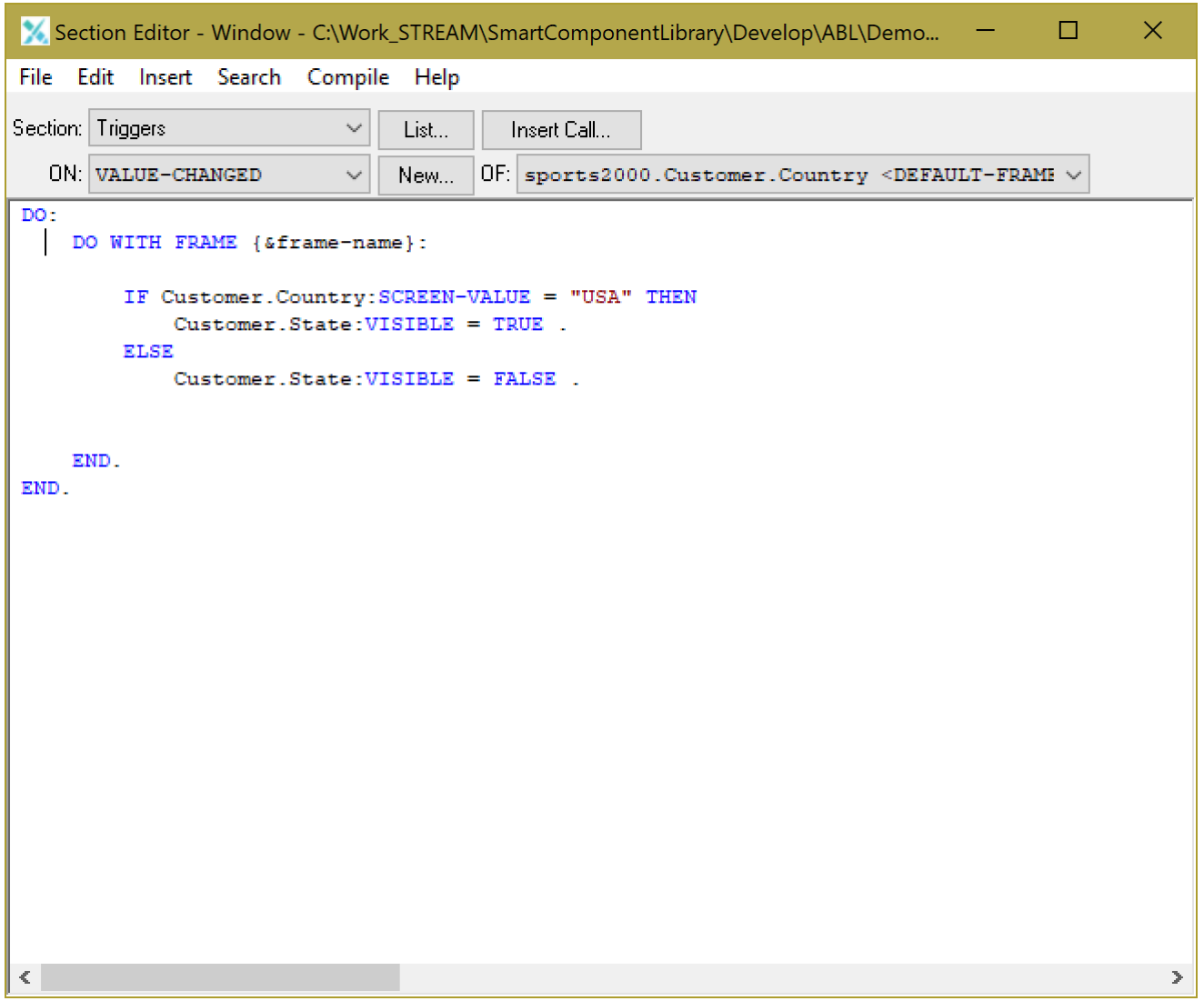
- Allows generation of various UI's
  - GUI for .NET
  - Angular
  - Kendo UI Builder
  - Meta-Data for UI repository database
  - …

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.



© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Migrated Trigger Code

```
METHOD PRIVATE VOID Customer_Country_ValueChanged (sender AS System.Object, e AS System.EventArgs):

    /* 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_Country AS Consultingwerk.Windows.LegacyGuiMig ation.Widgets.IWidgetFacade NO-UNDO .
    DEFINE VARIABLE Customer_State AS Consultingwerk.Windows.LegacyGuiMigra on.Widgets.IWidgetFacade NO-UNDO .

    Customer_Country = Consultingwerk.Windows.LegacyGuiMigration.Widgets.In istics.InfragisticsWidgetFactory:FromControl (THIS-
    Customer_State = Consultingwerk.Windows.LegacyGuiMigration.Widgets.Infra ics.InfragisticsWidgetFactory:FromControl (THIS-OB

    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

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Different types of screens following different templates

- Different types of screen
  - Data entry
  - Enquiry
  - Search screens
  - Report screen
  - Processing screens
  - Factory floor screens
  - …
- Might be migrated using different templates

© 2023 Consultingwerk Software Services Ltd. All rights reserved.

Questions

© 2023 Consultingwerk Software Services Ltd. All rights reserved.


Consultingwerk
software architecture and development