Preparing for Large-Scale Development

Morten Knudsen, Soft Design

Morten Knudsen

Soft Design, Consultant

- This presentation will take you through knowledge on starting large Plex projects. This has been gathered in Soft Design through projects, employees, and partners and extended by the introduction of new Websydian products. It also reflects an increased focus on a service-oriented approach to Plex development.
- The presentation will go through a number of relevant decisions and considerations to be made, standards and abstractions to be used, and questions to be asked before starting up real-life Plex projects.

Agenda

- Introduction
- Organization of Plex Development Models
- Specification of Function Parameters
- Scoping and Naming
- Service-Oriented Architecture
- Error Reporting and Sanity Checking
- Questions and Answers

FOR INFORMATION PURPOSES ONLY

Terms of this presentation

This presentation was based on current information and resource allocations as of April 2013 and is subject to change or withdrawal by CA at any time without notice. Notwithstanding anything in this presentation to the contrary, this presentation shall not serve to (i) affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (ii) amend any product documentation or specifications for any CA software product. The development, release and timing of any features or functionality described in this presentation remain at CA's sole discretion. Notwithstanding anything in this presentation to the contrary, upon the general availability of any future CA product release referenced in this presentation, CA will make such release available (i) for sale to new licensees of such product; and (ii) to existing licensees of such product on a when and if-available basis as part of CA maintenance and support, and in the form of a regularly scheduled major product release. Such releases may be made available to current licensees of such product who are current subscribers to CA maintenance and support on a when and if-available basis. In the event of a conflict between the terms of this paragraph and any other information contained in this presentation, the terms of this paragraph shall govern.

Certain information in this presentation may outline CA's general product direction. All information in this presentation is for your informational purposes only and may not be incorporated into any contract. CA assumes no responsibility for the accuracy or completeness of the information. To the extent permitted by applicable law, CA provides this presentation "as is" without warranty of any kind, including without limitation, any implied warranties or merchantability, fitness for a particular purpose, or non-infringement. In no event will CA be liable for any loss or damage, direct or indirect, from the use of this document, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if CA is expressly advised in advance of the possibility of such damages. CA confidential and proprietary. No unauthorized copying or distribution permitted.

Introduction

Participation in Start up of Several Large Projects

- KODA (Danish Composers)
- MSG/Synchronicer
- Danish Film Directors
- SIF Insurance

+40 Man Years

Organize Development Model to Facilitate Key Design Goals

- Uniform, transparent, and predictable
- Facilitation of reuse
- Robustness to changes
- Declarative and high-level rather than procedural code
- Layered design, service-oriented
- Quality (detect and reduce errors)
- Performance

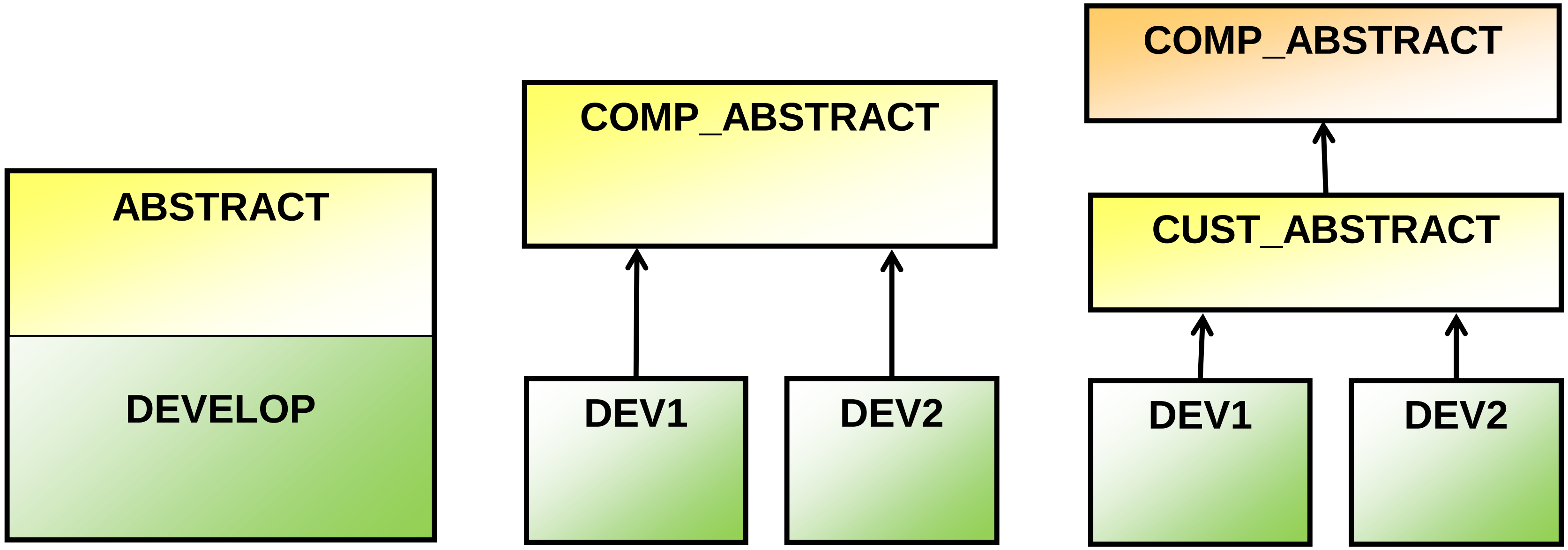
Only Selected Model Issues Covered by Presentation

- Should have been covered
  - Various coding standards
  - System documentation
- Not covered (not Plex model issues)
  - Project management, staffing, and organization
  - How to enforce decided standards
  - User participation
  - Specification
  - Test

Some decisions are hard to redo once been taken and development has started

Organization of Plex Development Models

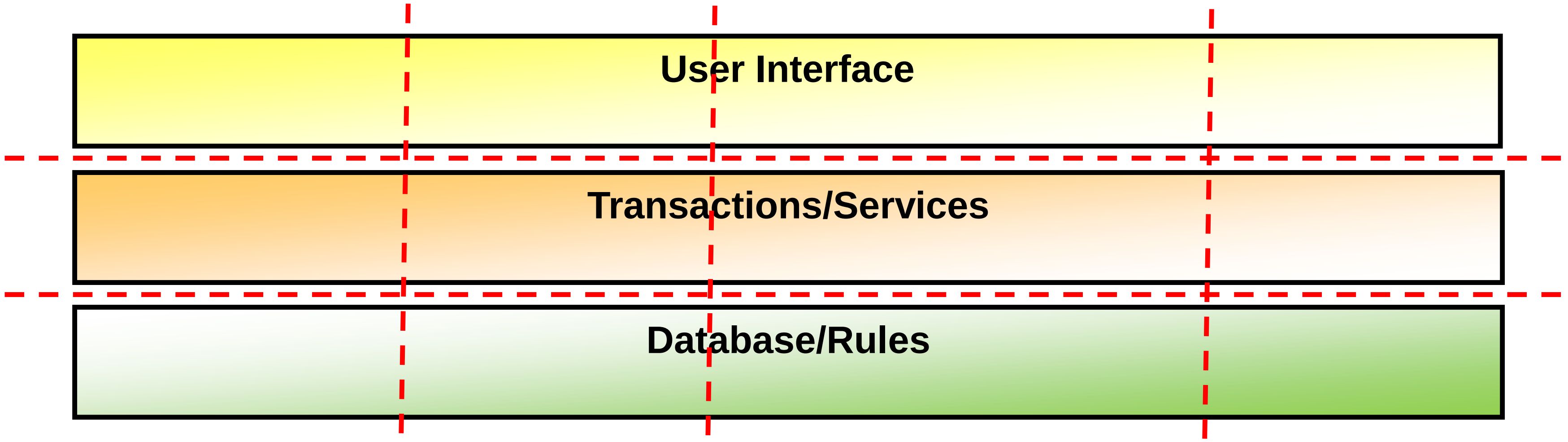
Abstract Patterns and Components in Separate Model(s)?



- Share abstract model between multiple development models
- Reuse abstract definitions across multiple projects/customers
- High ambitions…

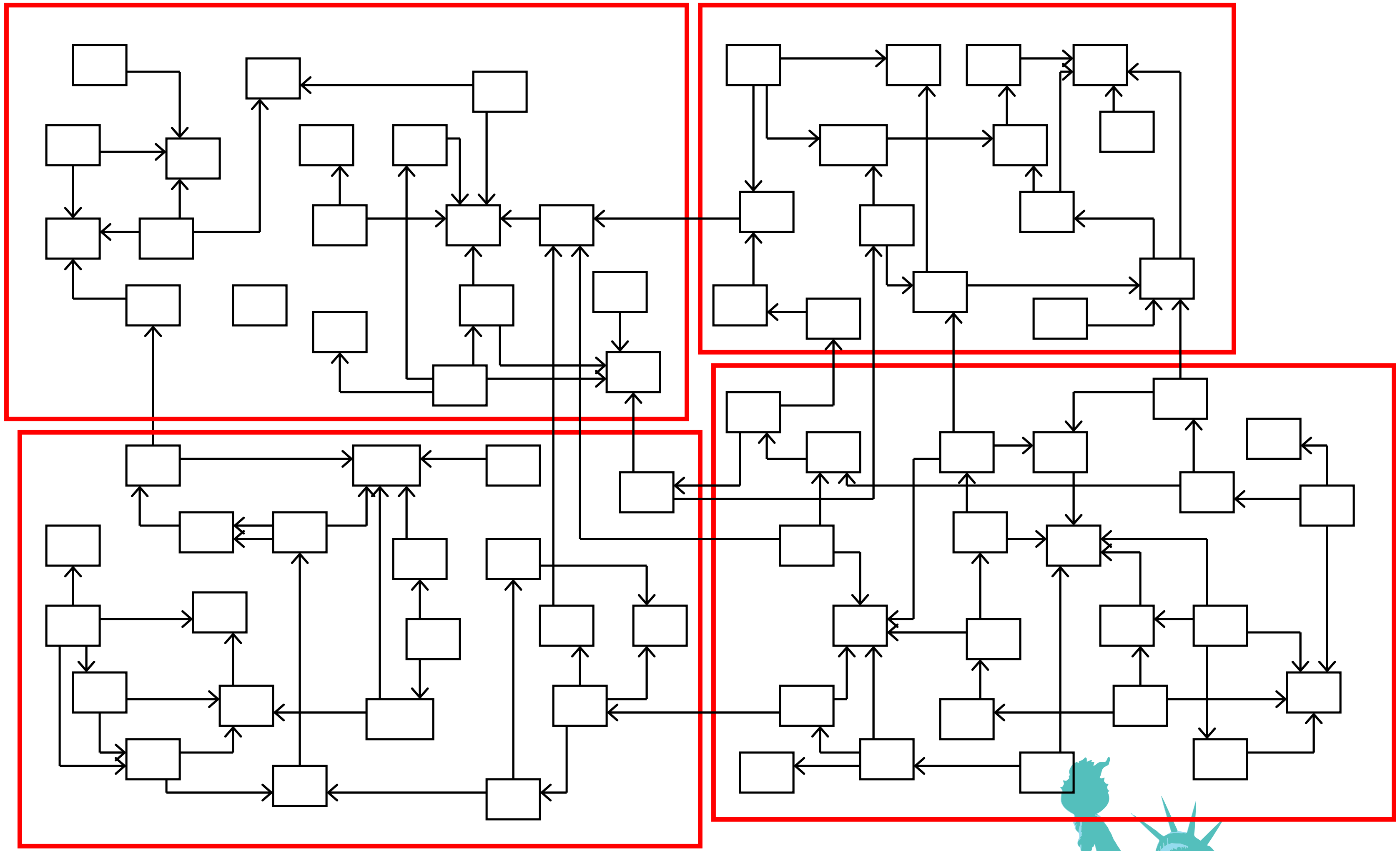
Model splitting is overhead

Horizontal versus Vertical Splitting of Development Models

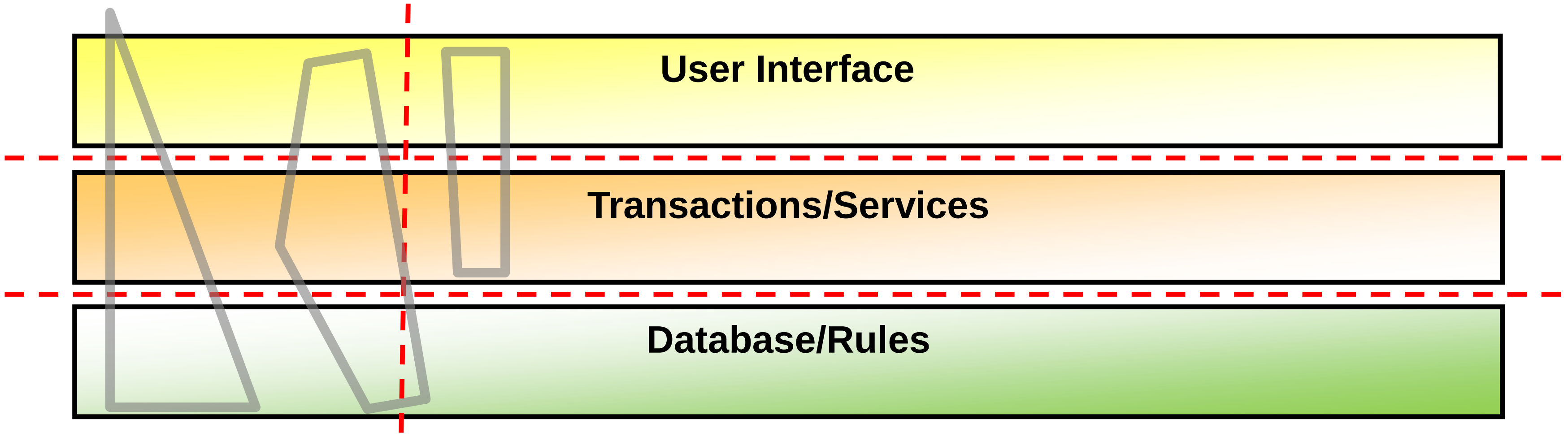


- User Interface
  - Panel/menu functions, Page generators, event handlers
- Transaction/services
  - Server functions for update and retrieval
- Database
  - Entity and field definitions, ‘rule functions’

Plex Model Splitting Based on Inter-Connected Subject Areas in Data Model

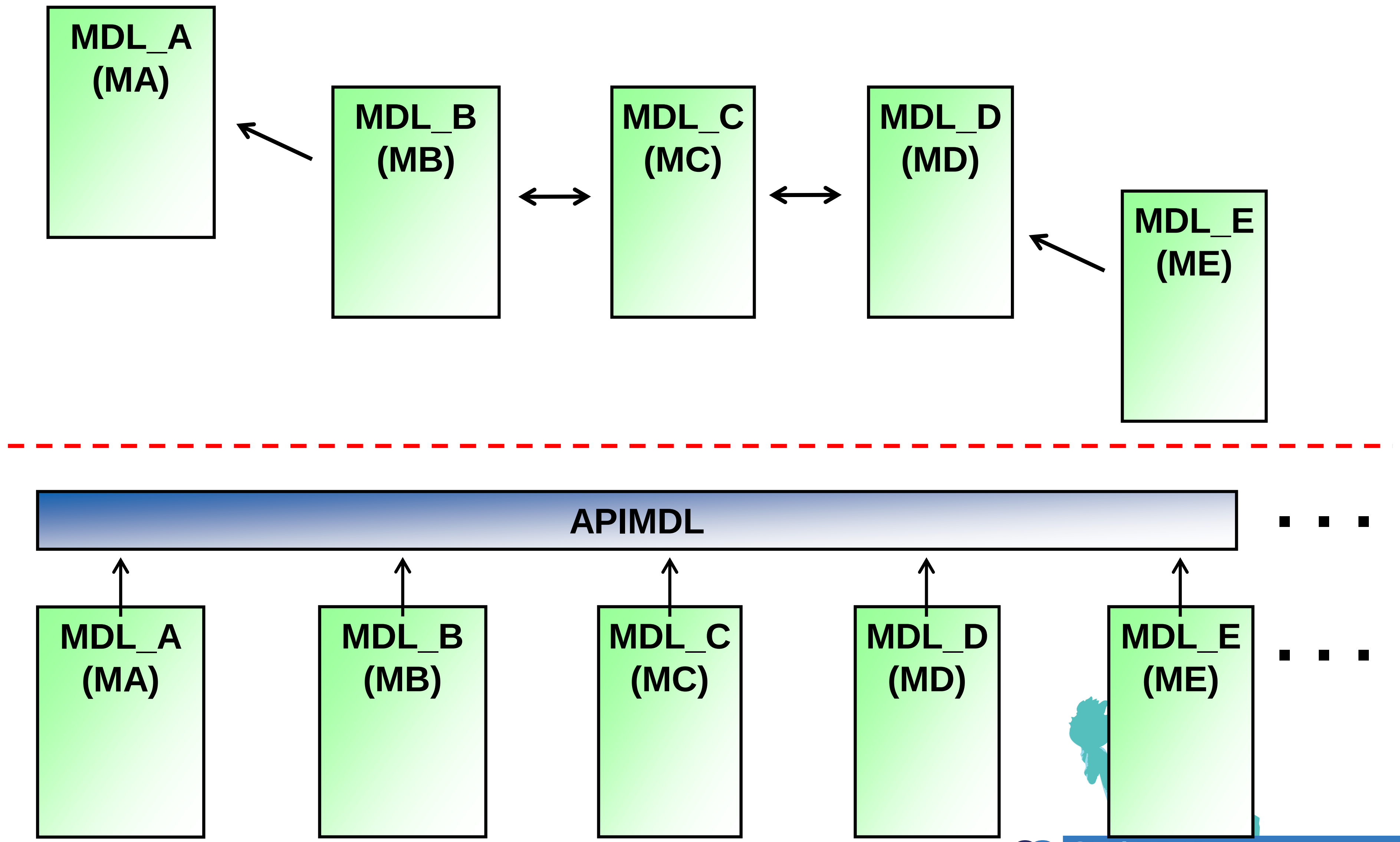


Application Layers Splitting across Development Tasks and Entity Patterns

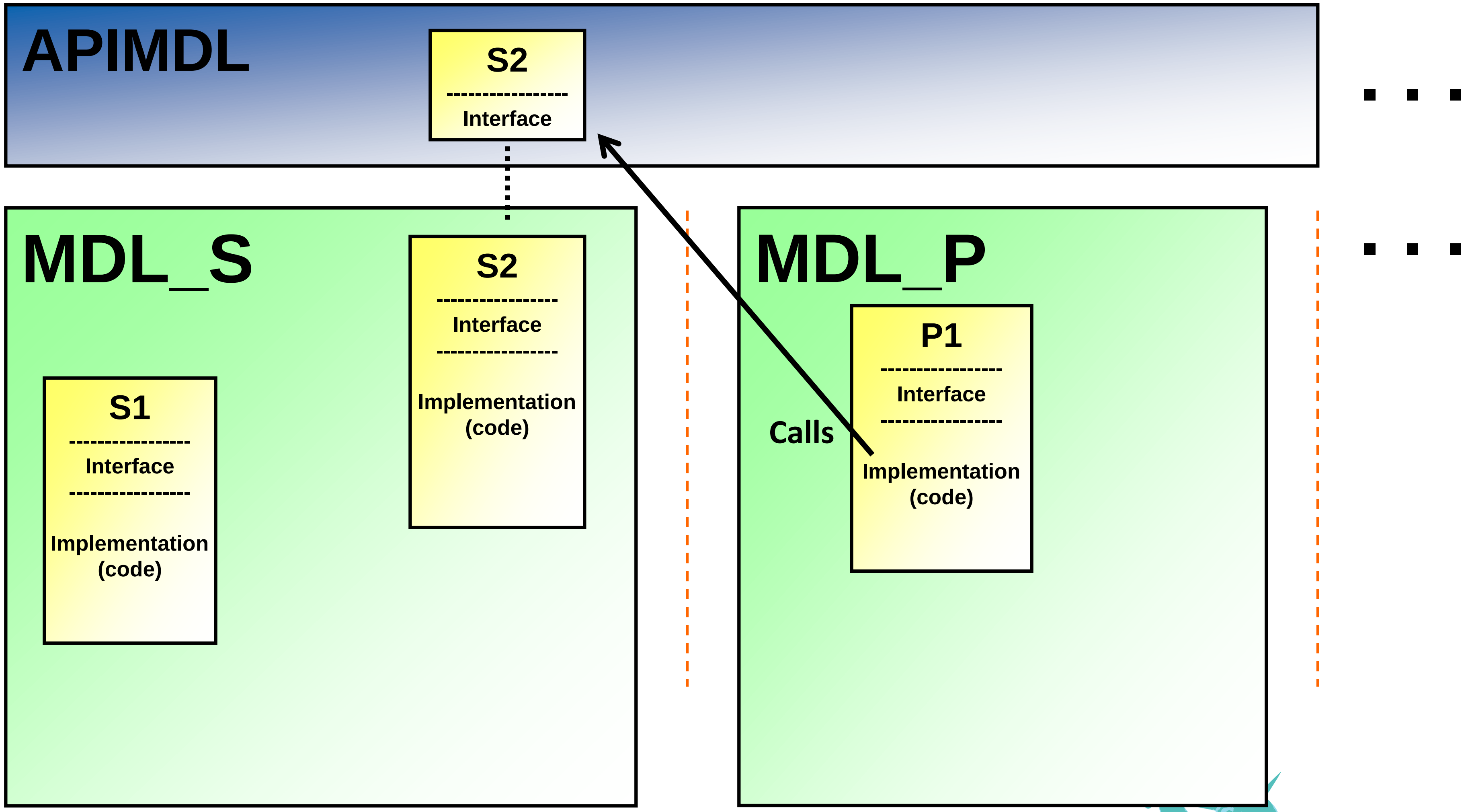


- Typical development tasks
  - Involves components in several layers
  - Cross-model development
- Entity patterns span multiple application layers
  - Less likely to cross data model boundaries
- Even larger overhead if different developers/roles are responsible for each layer

Use API Model to Separate Development Models



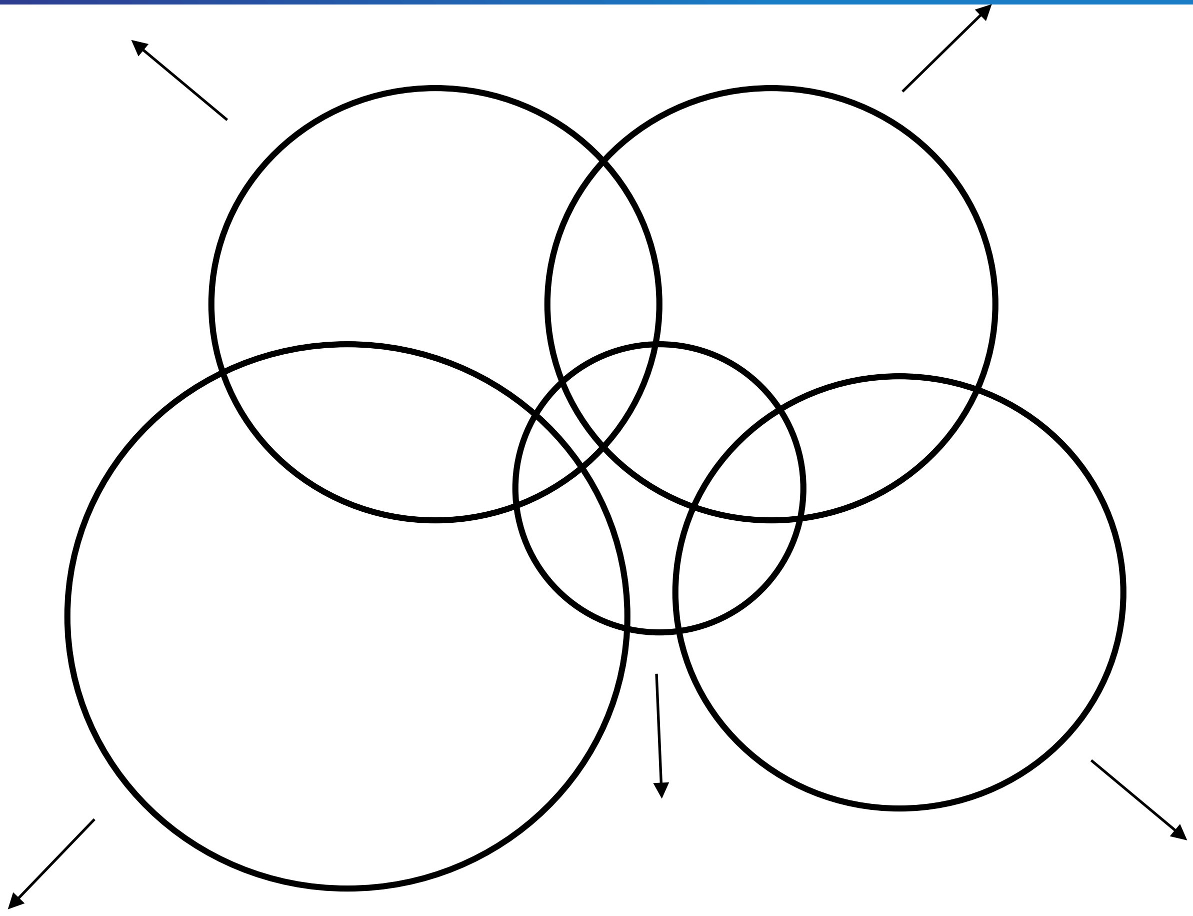
Entity Keys and Interface Specification only in API Model



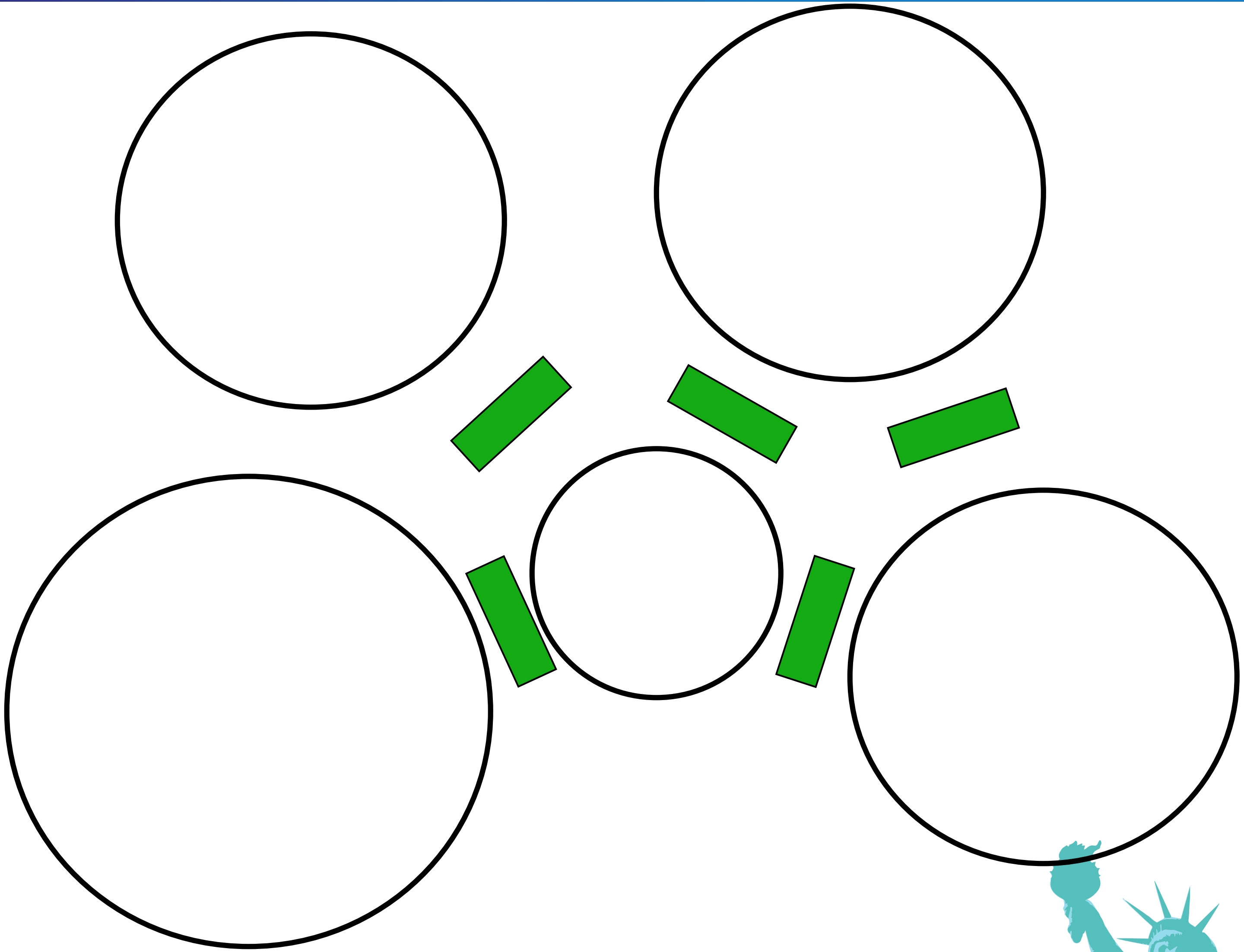
Pro and Cons of API Models

- Pros
  - Smaller models (faster update and extraction)
  - Encapsulation at model level
  - Focus on interface rather than implementation
  - Formal delegation of responsibility
  - Replacement of high-level components
- Cons
  - Redundant specifications need to be made
  - Object “Usage” harder to follow – stopped by the API
  - API functions should be documented…

Service-Oriented Architecture – Separate the Systems



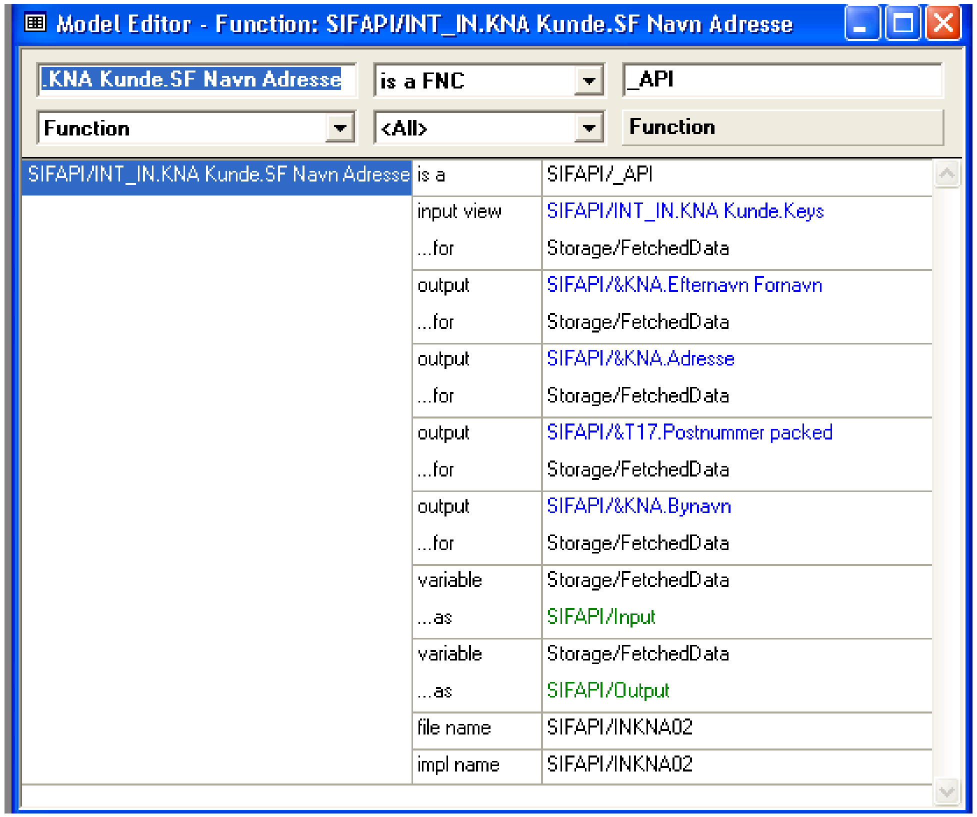
Service-Oriented Architecture – Focus on Interfaces



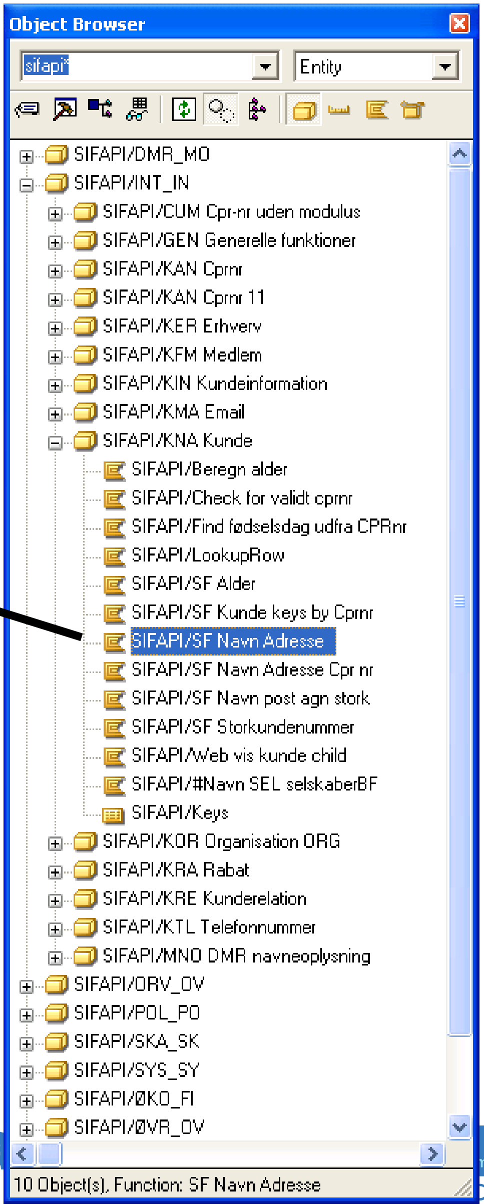
Principle Applied at System Level

API Model as Service Catalogue in Own Lean Model

- API model contains Interface specification only
- Can contain shared data type and domain fields
- Can be handed over to external providers


Model Editor - Function: SIFAPI/INT_IN.KNA Kunde.SF Navn Adresse

SIFAPI/INT_IN.KNA Kunde.SF Navn Adresse	is a	SIFAPI/_API
	input view	SIFAPI/INT_IN.KNA Kunde.Keys
	...for	Storage/FetchedData
	output	SIFAPI/&KNA.Efternavn Fornavn
	...for	Storage/FetchedData
	output	SIFAPI/&KNA.Adresse
	...for	Storage/FetchedData
	output	SIFAPI/&T17.Postnummer packed
	...for	Storage/FetchedData
	output	SIFAPI/&KNA.Bynavn
	...for	Storage/FetchedData
	variable	Storage/FetchedData
	...as	SIFAPI/Input
	variable	Storage/FetchedData
	...as	SIFAPI/Output
	file name	SIFAPI/INKNA02
	impl name	SIFAPI/INKNA02


Object Browser

- SIFAPI/DMR_MO
- SIFAPI/INT_IN
  - SIFAPI/CUM Cpr-nr uden modulus
  - SIFAPI/GEN Generelle funktioner
  - SIFAPI/KAN Cprnr
  - SIFAPI/KAN Cprnr 11
  - SIFAPI/KER Erhverv
  - SIFAPI/KFM Medlem
  - SIFAPI/KIN Kundeinformation
  - SIFAPI/KMA Email
  - SIFAPI/KNA Kunde
    - SIFAPI/Beregn alder
    - SIFAPI/Check for validt cprnr
    - SIFAPI/Find fødselsdag udfra CPRnr
    - SIFAPI/LookupRow
    - SIFAPI/SF Alder
    - SIFAPI/SF Kunde keys by Cprnr
    - SIFAPI/SF Navn Adresse
    - SIFAPI/SF Navn Adresse Cpr nr
    - SIFAPI/SF Navn post agn stork
    - SIFAPI/SF Storkundenummer
    - SIFAPI/Web vis kunde child
    - SIFAPI/#Navn SEL selskaberBF
    - SIFAPI/Keys
  - SIFAPI/KOR Organisation ORG
  - SIFAPI/KRA Rabat
  - SIFAPI/KRE Kunderelation
  - SIFAPI/KTL Telefonnummer
  - SIFAPI/MNO DMR navneoplysning
- SIFAPI/ORV_OV
- SIFAPI/POL_PO
- SIFAPI/SKA_SK
- SIFAPI/SYS_SY
- SIFAPI/ØKO_FI
- SIFAPI/ØVR_OV

10 Object(s), Function: SF Navn Adresse

Use of Levels and Versions

- Possible to correct errors on running version
- No Magic
- Object existence…
- Used to be a performance overhead…
- Levels/versions may be collapsed
- Make first production date version 1.0

Organization of Plex Development Models – Decisions

- Separate models with abstract patterns and components?
- One or more Plex development models?
- Vertical or horizontal model splitting?
- Model ‘encapsulation’ or full extract?
- Use of Plex levels and versions

Organization of Plex development models – early and irreversible decisions

Specification of Function Parameters

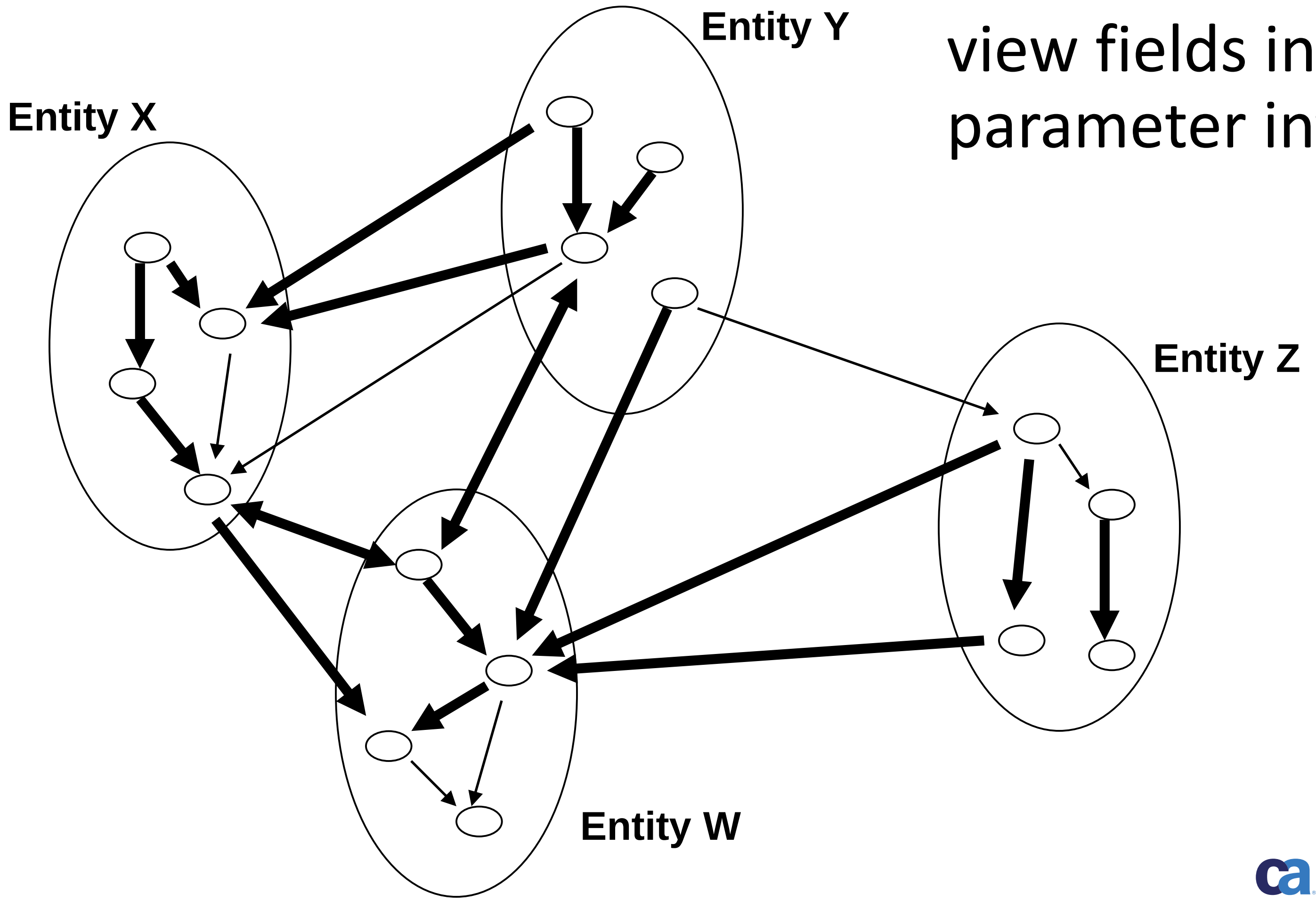
About Interfaces

- “It Is All About the Interface”
- “Make interfaces easy to use the right way and hard to use the wrong way”
- Keep interfaces stable
  - Trivial changes should not change interface definitions unnecessarily

Use of View Parameters in inherited *DataAccess* Functions

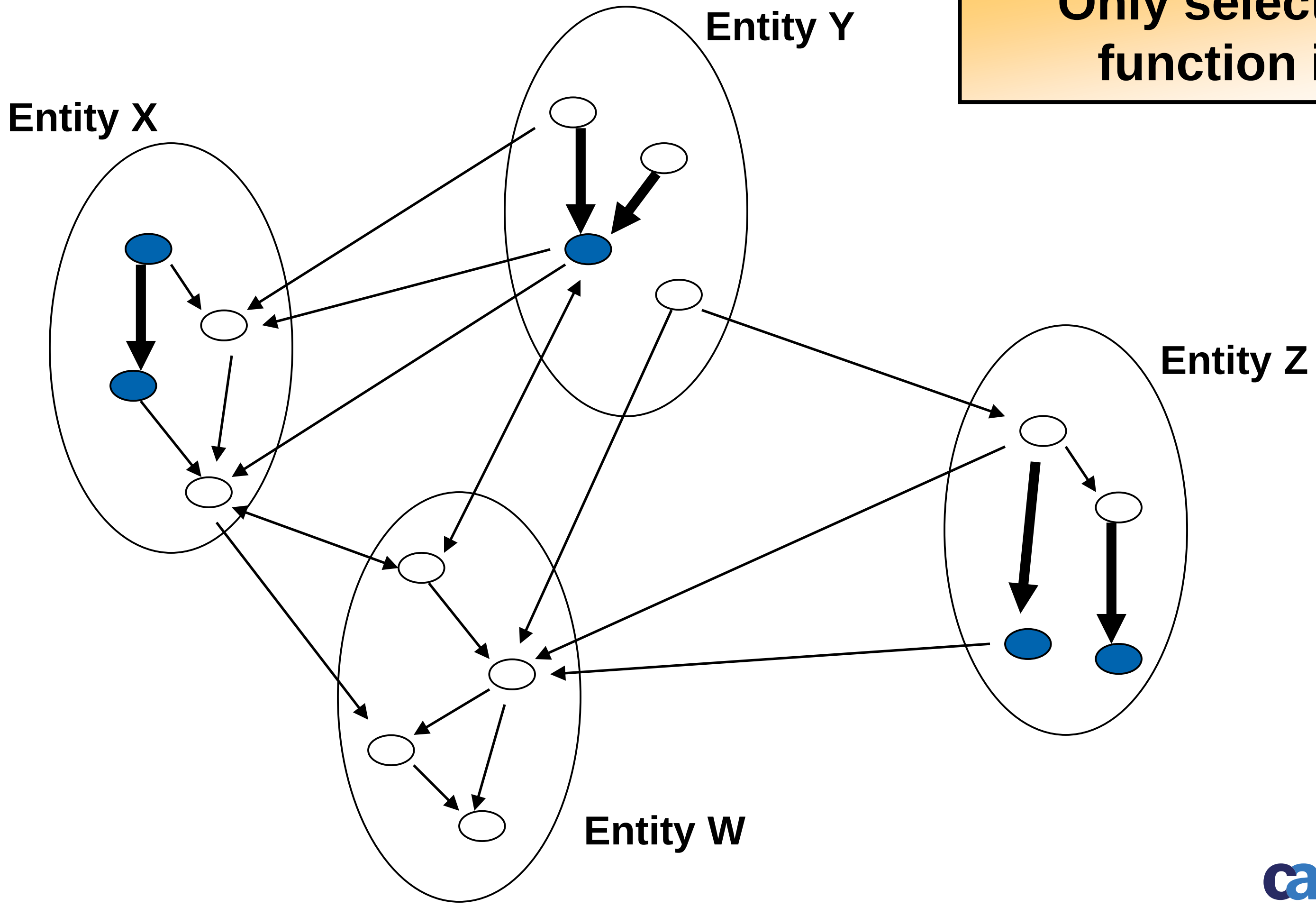
- *Fetch.SingleFetch*
  - *Fetch* view as output in *Output/FetchedData*
- *Fetch.BlockFetch*
  - *Fetch* view (64) as output in *Output/FetchedData*
- *Update.InsertRow*
  - *Update* view as dual input in *Input/InsertData*
- *Update.UpdateRow*
  - *Update* view (non-key) as dual input in *Input/InsertData*

Restricted Usage of All-Field Views as Function Parameters



- Bold arrows indicate calls to functions containing all view fields in their parameter interface

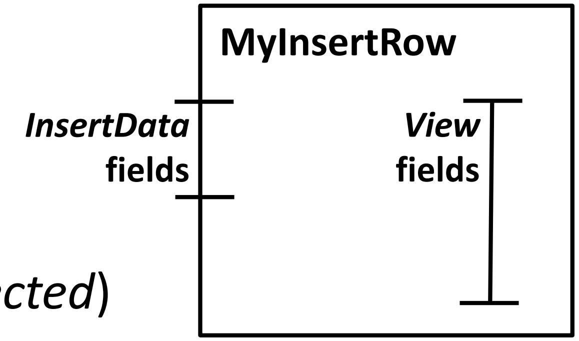
Restricted Usage of All-Field Views as Function Parameters



Only selected fields in function interfaces

Restricting Parameters

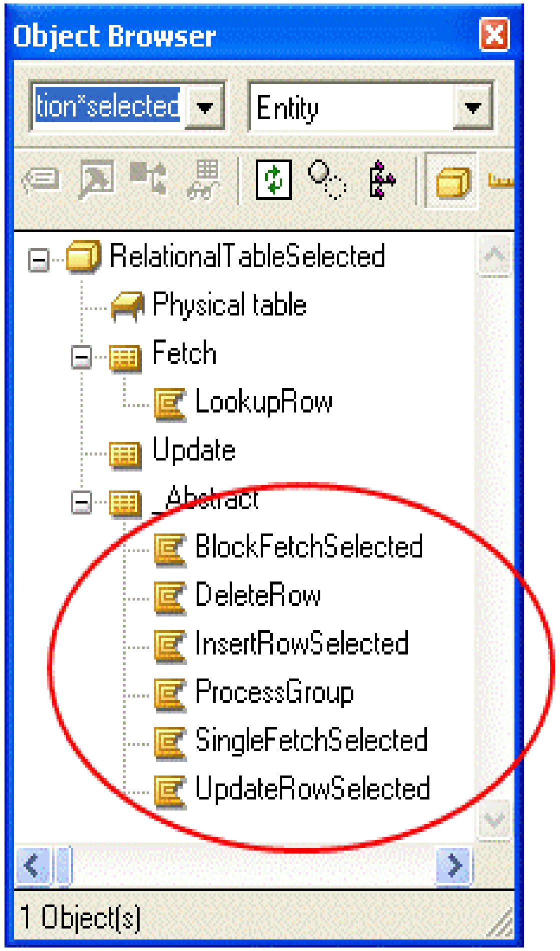
- Specific fields and *Selected* views only in function interfaces
  - Robustness to changes
- Add new fields/relations to entity –> Only generate:
  - Table, Views and Server functions
  - Functions calling explicitly changed functions
- Rules of design
  - Use views as parameter lists (VW *contains Selected*)
  - Omit superfluous parameters
  - Avoid mapping of constants…



Less Parameters -> Looser coupling

Abstract *RelationalTableSelected* Entity

- Define from scratch
  - Traditional naming of *Physical table* and *Update* and *Fetch* views
- Or extend *STORAGE/RelationalTable*?
- Abstract functions with reduced parameter lists
  - Scoped under *_Abstract* view
  - No implementation language specified
- *LookupRow* as only implemented function
  - Inherited call from *InsertRowSelected*

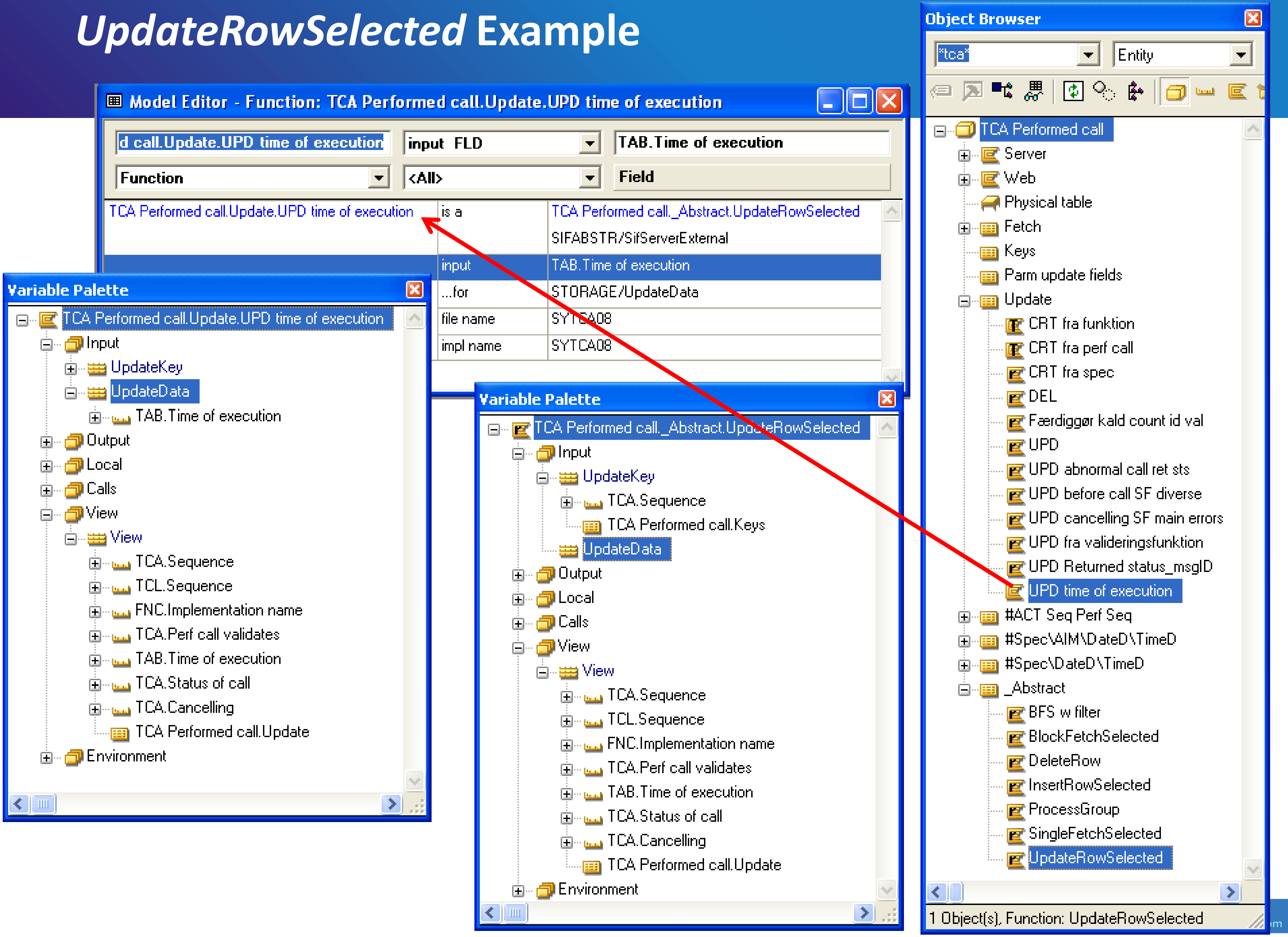


```
RelationalTableSelected
  Physical table
  Fetch
    LookupRow
  Update
  _Abstract
    BlockFetchSelected
    DeleteRow
    InsertRowSelected
    ProcessGroup
    SingleFetchSelected
    UpdateRowSelected
```

1 Object(s)

***RelationalTableSelected* supports parameter restriction rules**

UpdateRowSelected Example



Object Browser

tca | Entity

- TCA Performed call
  - Server
  - Web
  - Physical table
  - Fetch
  - Keys
  - Parm update fields
  - Update
    - CRT fra funktion
    - CRT fra perf call
    - CRT fra spec
    - DEL
    - Færdiggør kald count id val
    - UPD
    - UPD abnormal call ret sts
    - UPD before call SF diverse
    - UPD cancelling SF main errors
    - UPD fra valideringsfunktion
    - UPD Returned status_msgID
    - UPD time of execution
  - #ACT Seq Perf Seq
  - #Spec\AIM\DateD\TimeD
  - #Spec\DateD\TimeD
  - _Abstract
    - BFS w filter
    - BlockFetchSelected
    - DeleteRow
    - InsertRowSelected
    - ProcessGroup
    - SingleFetchSelected
    - UpdateRowSelected

1 Object(s), Function: UpdateRowSelected

Model Editor - Function: TCA Performed call.Update.UPD time of execution

d call.Update.UPD time of execution	input FLD	TAB.Time of execution
Function	<All>	Field

TCA Performed call.Update.UPD time of execution	is a	TCA Performed call._Abstract.UpdateRowSelected
		SIFABSTR/SifServerExternal
	input	TAB.Time of execution
	...for	STORAGE/UpdateData
	file name	SYTCA08
	impl name	SYTCA08

Variable Palette

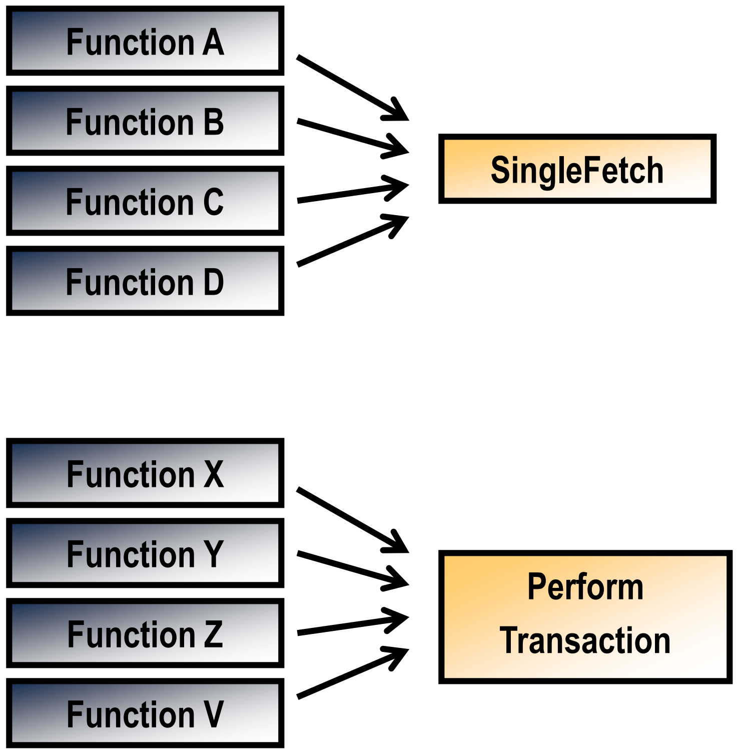
- TCA Performed call.Update.UPD time of execution
  - Input
    - UpdateKey
    - UpdateData
      - TAB.Time of execution
  - Output
  - Local
  - Calls
  - View
    - View
      - TCA.Sequence
      - TCL.Sequence
      - FNC.Implementation name
      - TCA.Perf call validates
      - TAB.Time of execution
      - TCA.Status of call
      - TCA.Cancelling
      - TCA Performed call.Update
  - Environment

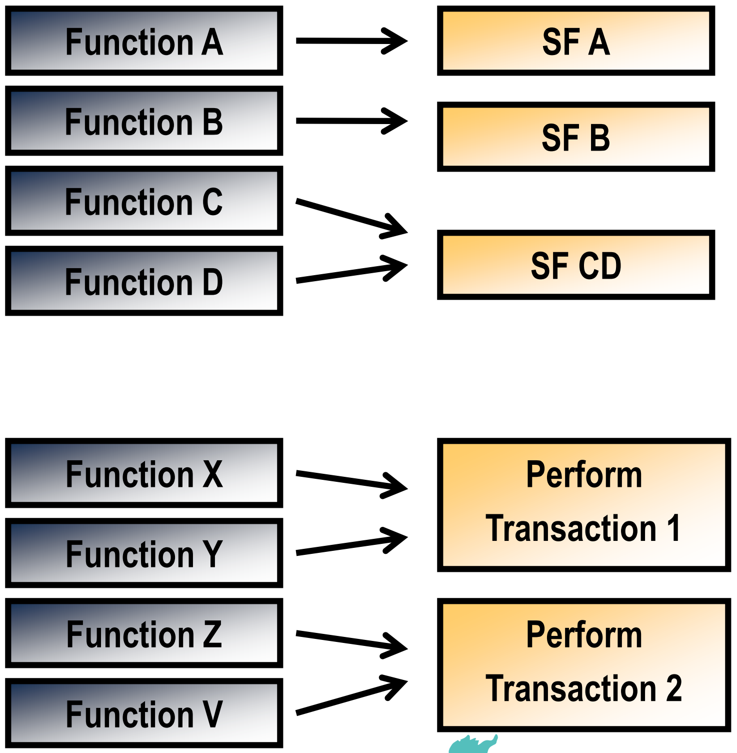
Variable Palette

- TCA Performed call._Abstract.UpdateRowSelected
  - Input
    - UpdateKey
      - TCA.Sequence
      - TCA Performed call.Keys
    - UpdateData
  - Output
  - Local
  - Calls
  - View
    - View
      - TCA.Sequence
      - TCL.Sequence
      - FNC.Implementation name
      - TCA.Perf call validates
      - TAB.Time of execution
      - TCA.Status of call
      - TCA.Cancelling
      - TCA Performed call.Update
  - Environment

General versus Granular Design

- General
- Granular





General versus Granular Design

Pros

- Robustness towards data model changes
- Reduced scope of functions to be generated
  - Developers will not step on each others toes
- Simple design and transparent functionality
- Function are easy to call/use
- Easer tracking of field usage

Cons

- Many function objects in model
- Drown in model function objects – which one to choose?
- Many implemented objects

Specification of Function Parameters – Decisions

- Standards and patterns for parameter restriction?
- How to implement *Selected* entity pattern

Scoping and Naming

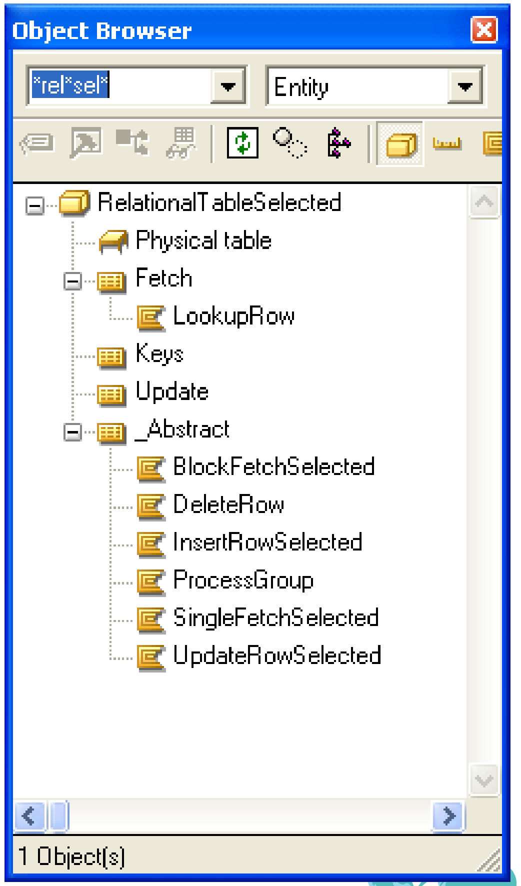
Well-Defined Rules for Scoping

- Use scoping to organize model objects
- Consistent and unambiguous rules for scoping
  - Support navigation in model
  - “Where to find object?” (for reuse)
- Scoping rules often given by Plex abstractions
  - Through inheritance
  - Additional rules may be necessary
- Scoping rules <-> naming standards
- Use of acronyms
- Capitalization of letters

Scoping Levels

- Plex model
- Subject area
- Entities
- Views
- Functions
  - Various levels
- Fields

Given by inheritance from *RelationalTableSelected*



Object Browser

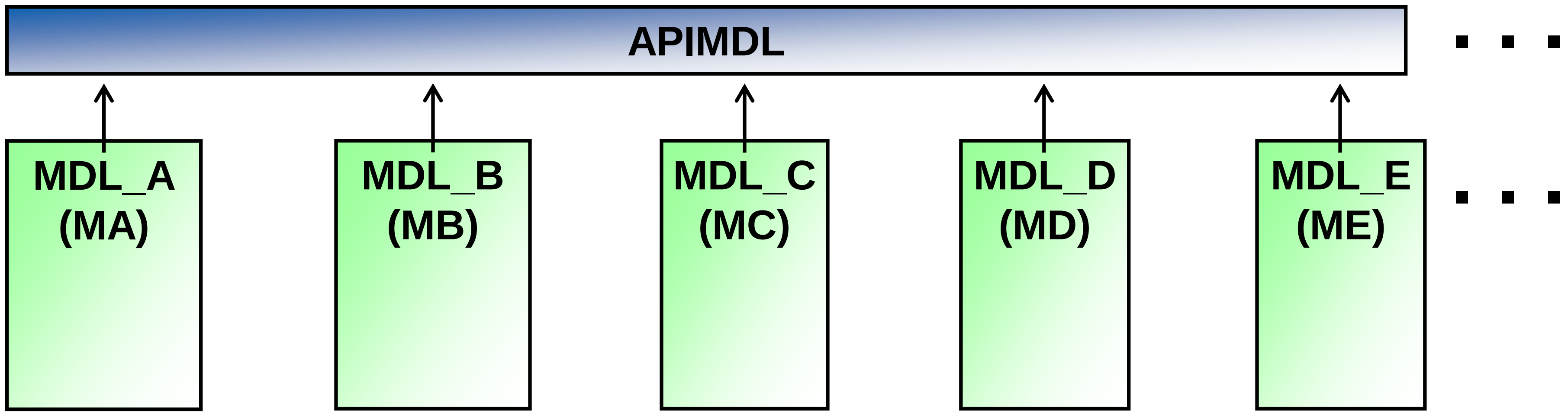
*rel*sel* Entity

- RelationalTableSelected
  - Physical table
  - Fetch
    - LookupRow
  - Keys
  - Update
  - _Abstract
    - BlockFetchSelected
    - DeleteRow
    - InsertRowSelected
    - ProcessGroup
    - SingleFetchSelected
    - UpdateRowSelected

1 Object(s)

Data Model Scoping

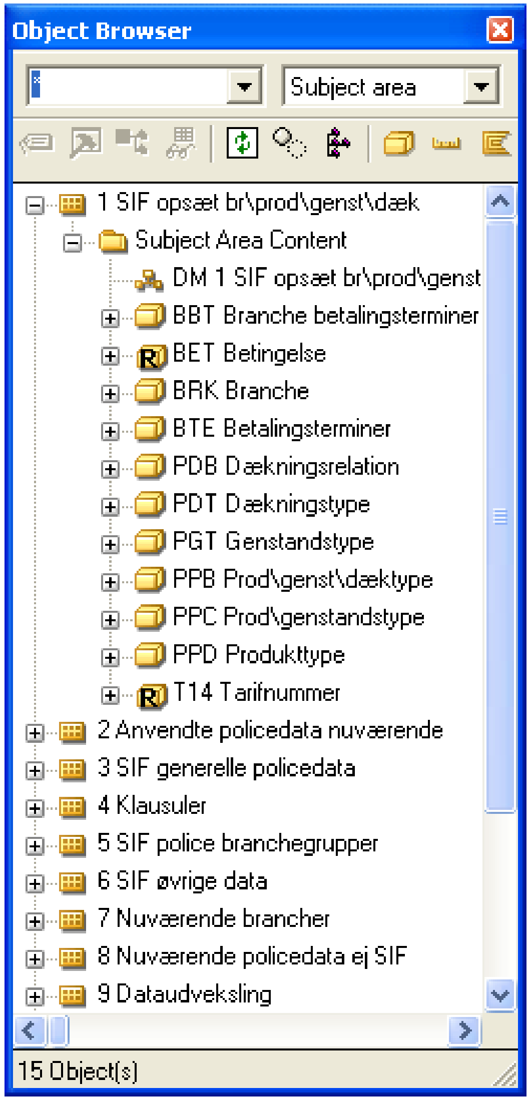
- Model splitting <-> High-level scoping of data model



Splitting development into multiple Plex models is a grouping of the data model!

Group Data Model in Subject Areas

- Subject area – Group of 4-10 entities
- Grouping
  - Plex subject areas
  - Scope by entities
- Entity naming
  - Single instance
  - E.g. “Order”, “Vehicle”

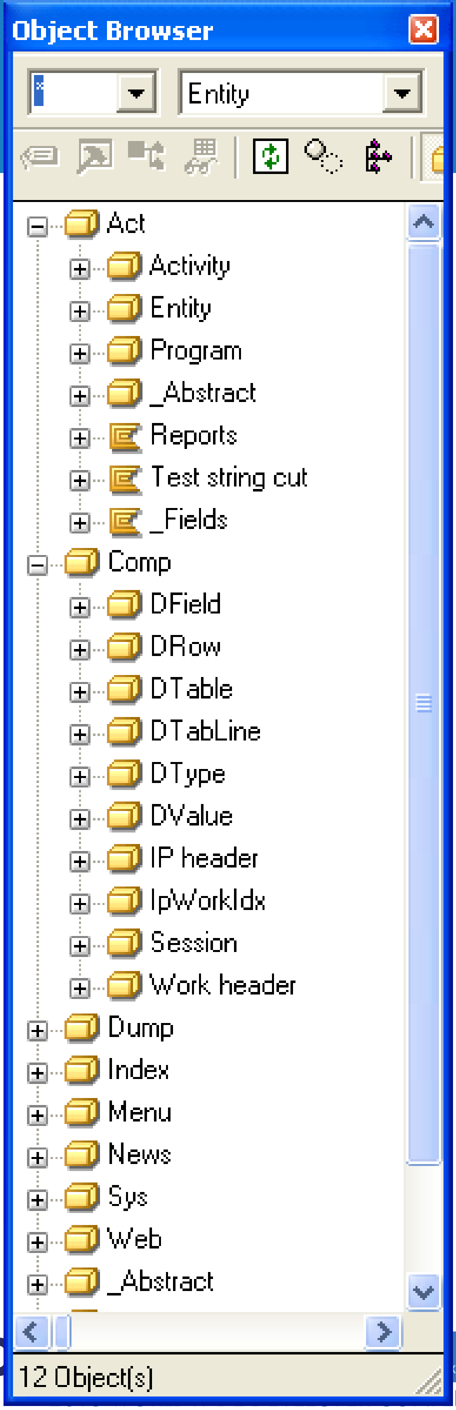


Object Browser

Subject area

```
1 SIF opsæt br\prod\genst\dæk
  Subject Area Content
    DM 1 SIF opsæt br\prod\genst
    BBT Branche betalingsterminer
    BET Betingelse
    BRK Branche
    BTE Betalingsterminer
    PDB Dækningsrelation
    PDT Dækningstype
    PGT Genstandstype
    PPB Prod\genst\dæktype
    PPC Prod\genstandstype
    PPD Produkttype
    T14 Tarifnummer
2 Anvendte policedata nuværende
3 SIF generelle policedata
4 Klausuler
5 SIF police branchegrupper
6 SIF øvrige data
7 Nuværende brancher
8 Nuværende policedata ej SIF
9 Dataudveksling
```

15 Object(s)



Object Browser

Entity

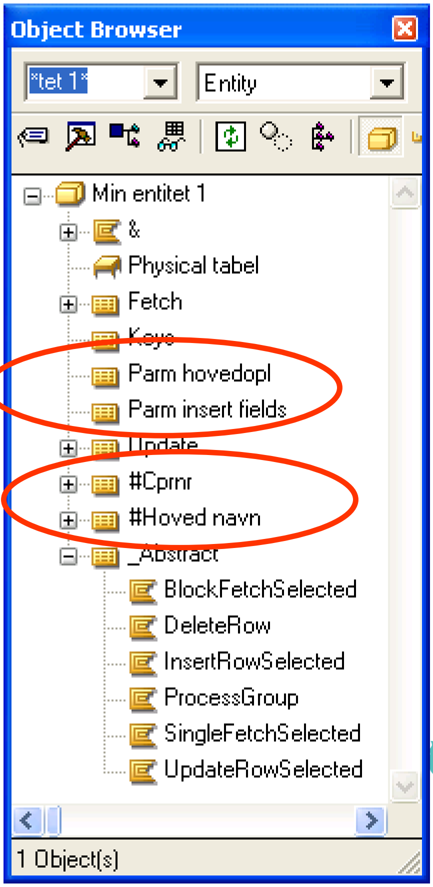
```
Act
  Activity
  Entity
  Program
  _Abstract
  Reports
  Test string cut
  _Fields
Comp
  DField
  DRow
  DTable
  DTabLine
  DType
  DValue
  IP header
  IpWorkIdx
  Session
  Work header
Dump
Index
Menu
News
Sys
Web
_Abstract
```

12 Object(s)

View Naming – Examples

Other variant

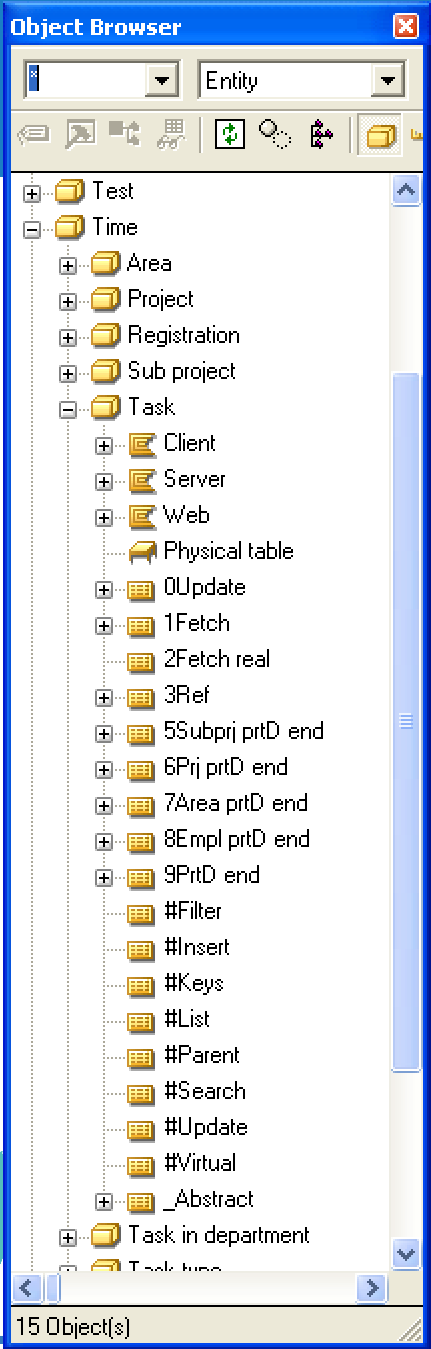
- Implemented/indexed views
  - Name = "#" + sorting
    - *"#Cust nr seq navn"*
    - *"#Post nr cust name"*
- Parameter lists
  - Name = "Parm" + name
    - *"Parm update fields"*
    - *"Parm main info"*


Object Browser

tet 1 — Entity

Min entitet 1
&
Physical tabel
Fetch
Keys
Parm hovedopl
Parm insert fields
Update
#Cprnr
#Hoved navn
_Abstract
BlockFetchSelected
DeleteRow
InsertRowSelected
ProcessGroup
SingleFetchSelected
UpdateRowSelected

1 Object(s)


Object Browser

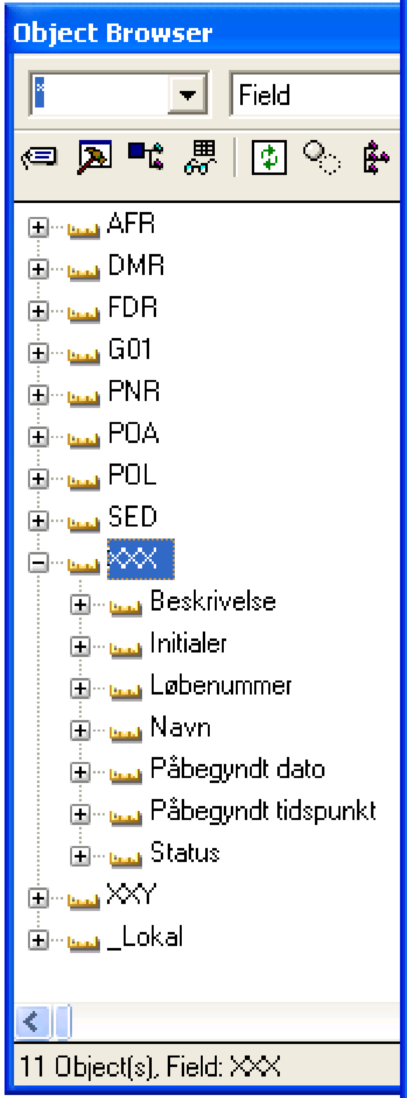
* — Entity

Test
Time
Area
Project
Registration
Sub project
Task
Client
Server
Web
Physical table
0Update
1Fetch
2Fetch real
3Ref
5Subprj prtD end
6Prj prtD end
7Area prtD end
8Empl prtD end
9PrtD end
#Filter
#Insert
#Keys
#List
#Parent
#Search
#Update
#Virtual
_Abstract
Task in department

15 Object(s)

Scope and Naming of Fields Objects – Examples

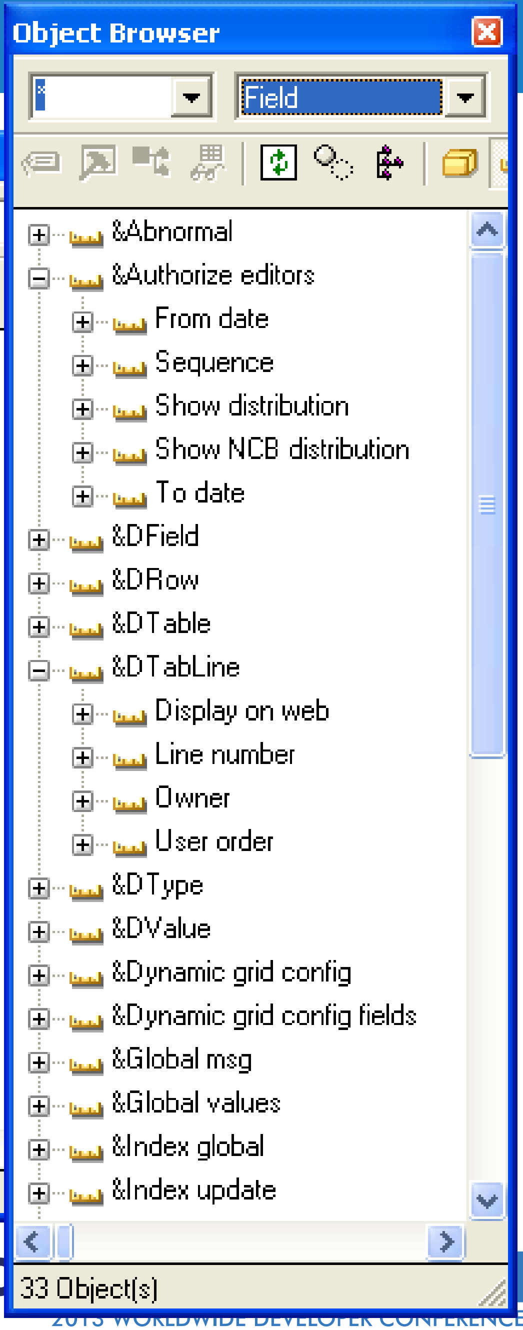
- Database fields may be scoped under 'entity field'
  - Group fields belonging to same entity
  - Underlying fields belonging to one entity only
  - More specific names
    - Default names to appear on panels
  - More characters available for naming
- Scope local fields under *_Local* or *_Work* field


Object Browser

Field

AFR
DMR
FDR
G01
PNR
POA
POL
SED
XXX
  Beskrivelse
  Initialer
  Løbenummer
  Navn
  Påbegyndt dato
  Påbegyndt tidspunkt
  Status
XXY
_Lokal

11 Object(s), Field: XXX


Object Browser

Field

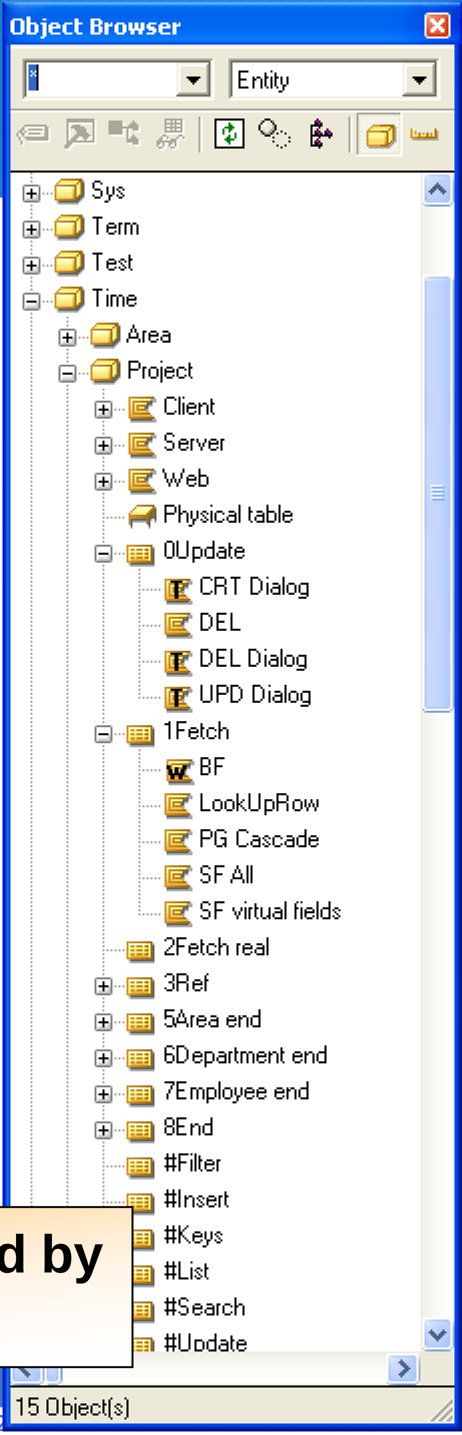
&Abnormal
&Authorize editors
  From date
  Sequence
  Show distribution
  Show NCB distribution
  To date
&DField
&DRow
&DTable
&DTabLine
  Display on web
  Line number
  Owner
  User order
&DType
&DValue
&Dynamic grid config
&Dynamic grid config fields
&Global msg
&Global values
&Index global
&Index update

33 Object(s)

Server/View Functions – Scoping and Naming

- Scope under view being accessed
  - Or under *Server* suite
- Use acronyms for naming, e.g.:

– CRT:	Create
– UPD:	Update
– DEL:	Delete
– SF:	SingleFetch
– CHK:	CheckRow
– BF:	BlockFetch
– BFS:	BlockFetchSet (fetch restricted set)
– PG:	ProcessGroup


Object Browser

Entity

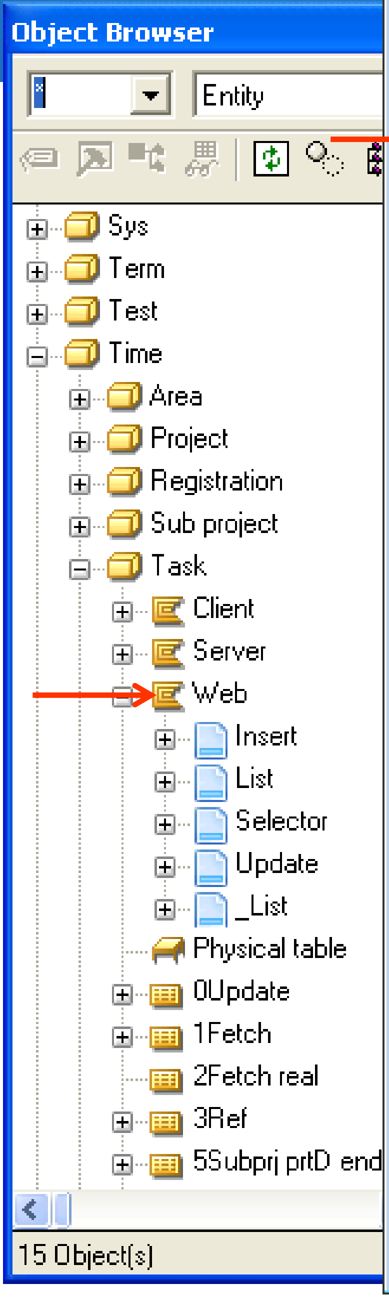
```
Sys
Term
Test
Time
  Area
  Project
    Client
    Server
    Web
    Physical table
    0Update
      CRT Dialog
      DEL
      DEL Dialog
      UPD Dialog
    1Fetch
      BF
      LookUpRow
      PG Cascade
      SF All
      SF virtual fields
    2Fetch real
    3Ref
    5Area end
    6Department end
    7Employee end
    8End
    #Filter
    #Insert
    #Keys
    #List
    #Search
    #Update
```

15 Object(s)

Acronym may stand alone or may be followed by an additional description

Web Functions – Scoping and Naming

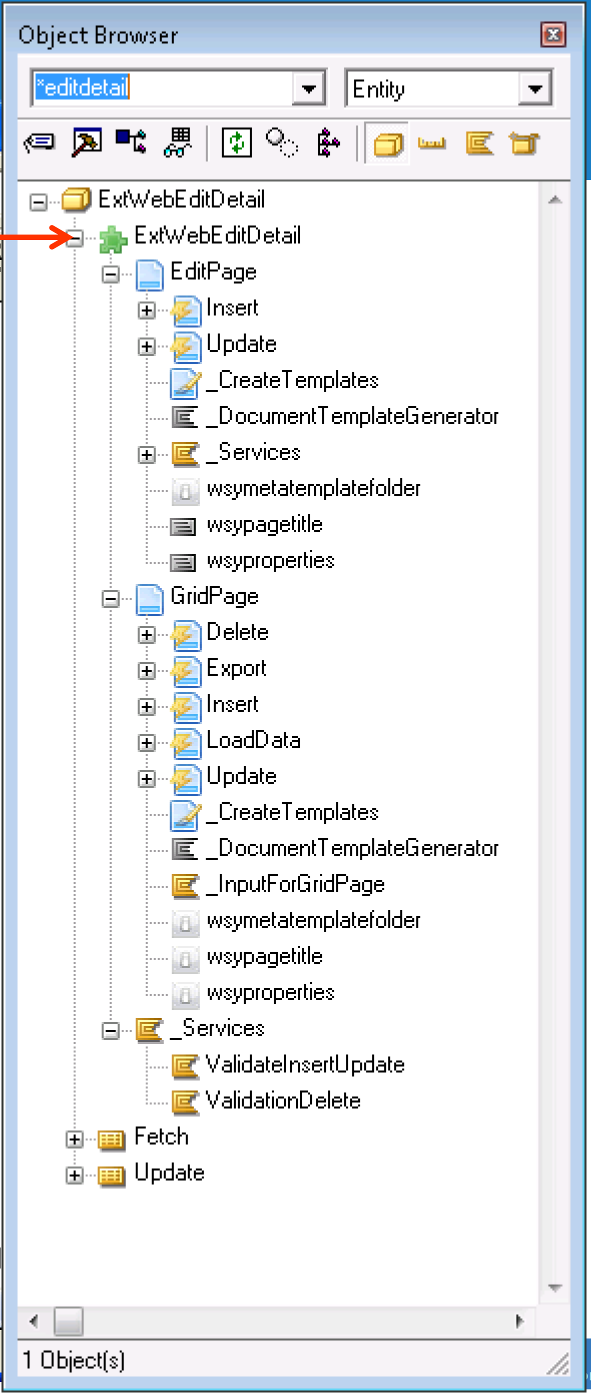
- By use case…
  - Scope Web Pages under Entry (menu point) function
  - Associate Pages to flow
- …or data-oriented?
  - Scope Web Pages under Entity or Entity suite function
  - Associate Pages to data


Object Browser

Entity

- Sys
- Term
- Test
- Time
  - Area
  - Project
  - Registration
  - Sub project
  - Task
    - Client
    - Server
    - Web
      - Insert
      - List
      - Selector
      - Update
      - _List
    - Physical table
    - 0Update
    - 1Fetch
    - 2Fetch real
    - 3Ref
    - 5Subprj prtD end

15 Object(s)


Object Browser

*editdetail | Entity

- ExtWebEditDetail
  - ExtWebEditDetail
    - EditPage
      - Insert
      - Update
      - _CreateTemplates
      - _DocumentTemplateGenerator
      - _Services
      - wsymetatemplatefolder
      - wsypagetitle
      - wsyproperties
    - GridPage
      - Delete
      - Export
      - Insert
      - LoadData
      - Update
      - _CreateTemplates
      - _DocumentTemplateGenerator
      - _InputForGridPage
      - wsymetatemplatefolder
      - wsypagetitle
      - wsyproperties
    - _Services
      - ValidateInsertUpdate
      - ValidationDelete
  - Fetch
  - Update

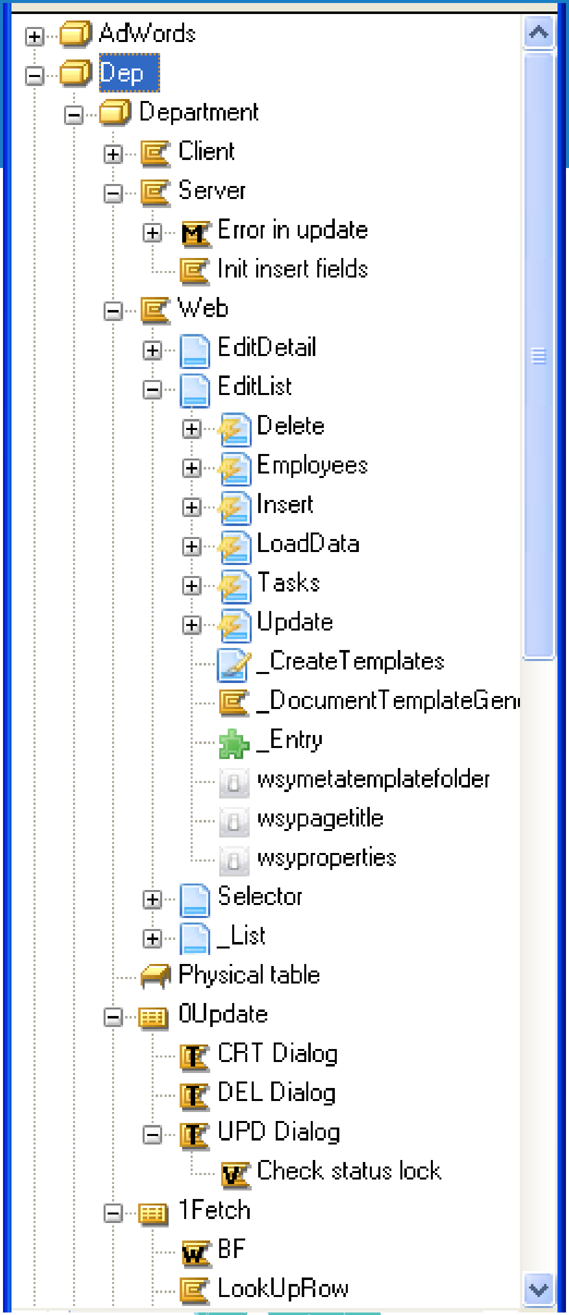
1 Object(s)

Other Model Objects – Naming

- Implementation names
  - Tables
  - Views
  - Page generators?
  - Database fields
- Diagram acronyms, e.g.
  - WF: Web Flow
  - FF: Function Flow
  - DM: Data Model

Using Icons

- Use Icons for core model objects, e.g.
  - Web Pages
  - Events
  - Entry points
  - Transactions
  - Validation functions
  - Message function
  - BlockFetch Wrapper functions
  - Reference entities
  - Objects not used
  - …



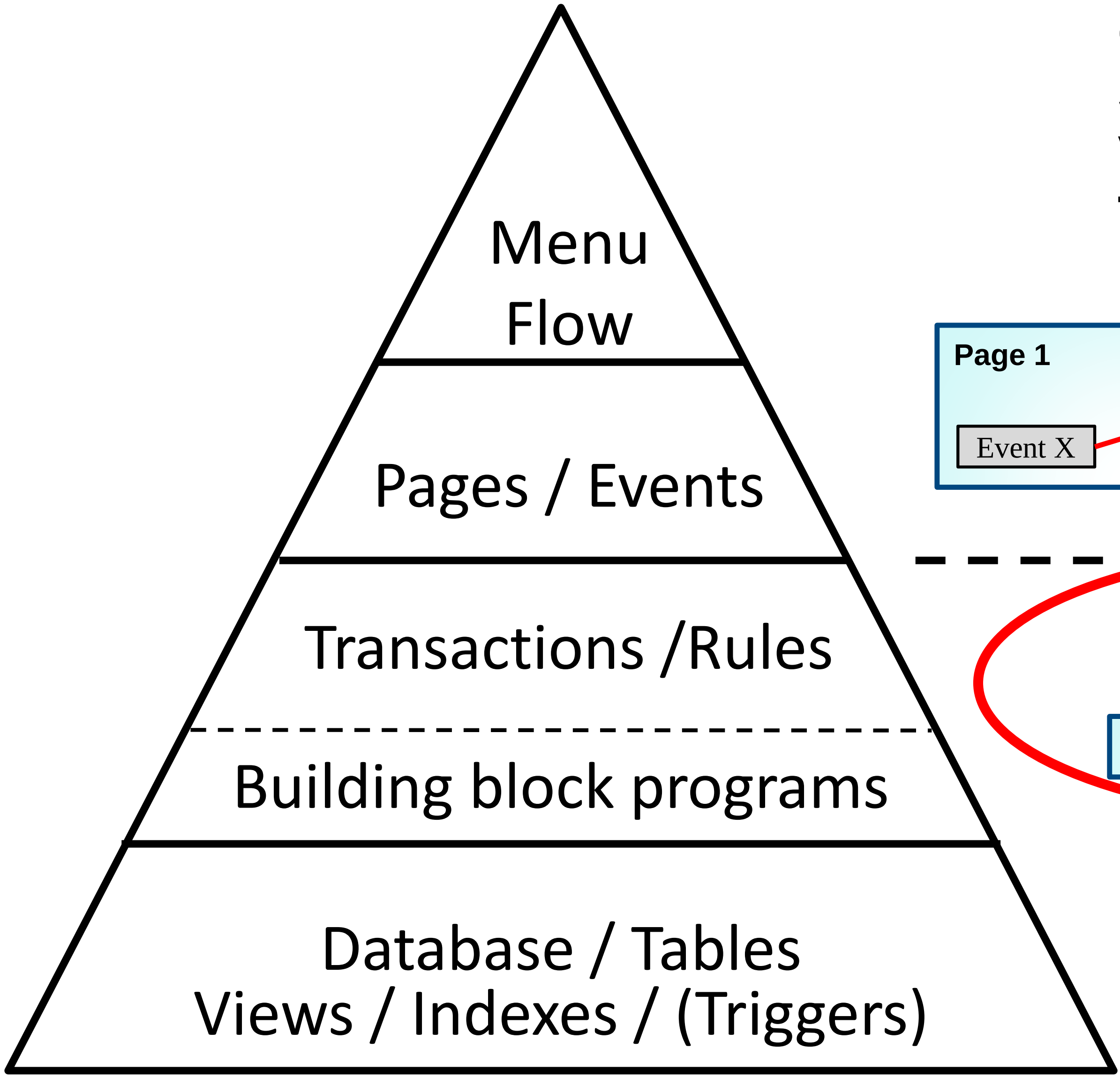
```
AdWords
Dep
  Department
    Client
    Server
      Error in update
      Init insert fields
    Web
      EditDetail
      EditList
        Delete
        Employees
        Insert
        LoadData
        Tasks
        Update
        _CreateTemplates
        _DocumentTemplateGen
        _Entry
        wsymetatemplatefolder
        wsypagetitle
        wsyproperties
      Selector
      _List
    Physical table
    0Update
      CRT Dialog
      DEL Dialog
      UPD Dialog
        Check status lock
    1Fetch
      BF
      LookUpRow
```

Scoping and Naming – Decisions

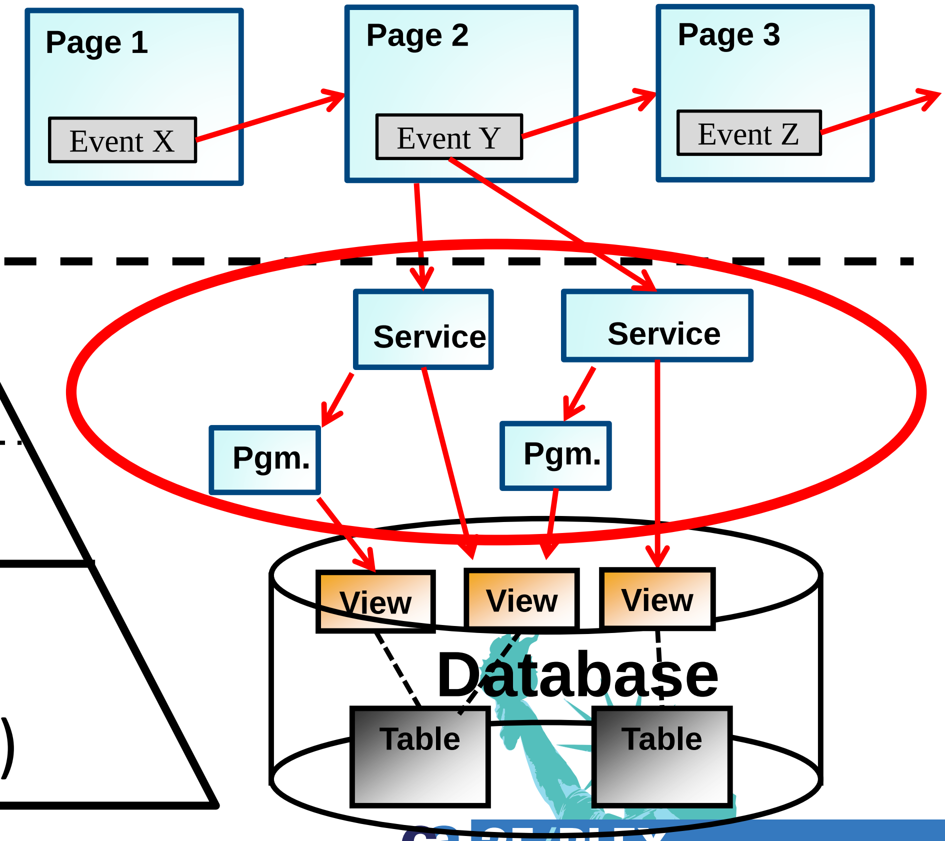
- Divide data model into subject areas – and how?
- Fields scope and naming?
- Standards for function scoping?
- Naming views?
- Use of acronyms?
- Use of icons?
- Explicit specification of implementation names – how, and for what types of objects?

Service-Oriented Architecture

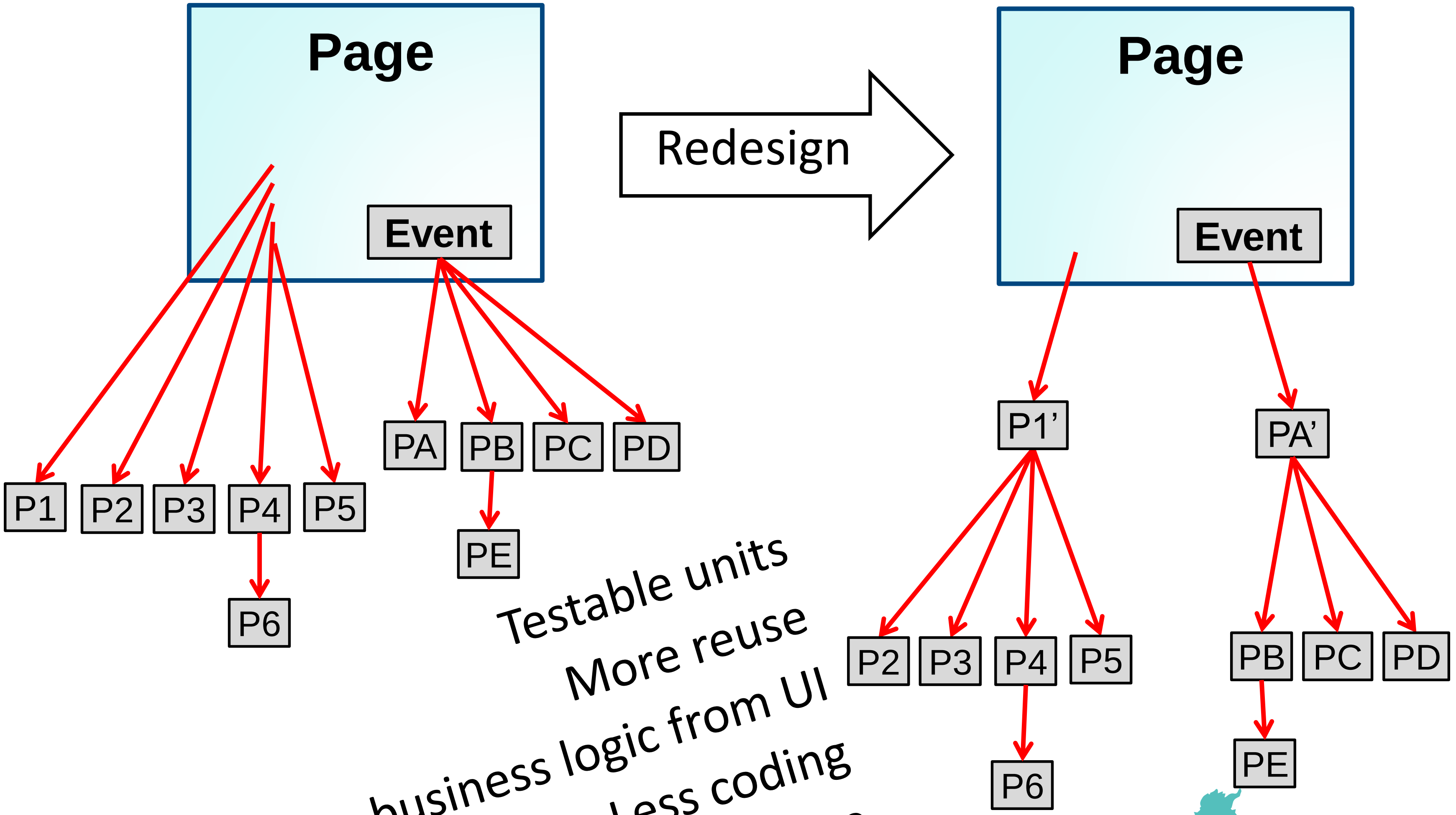
Application Layers (Websydian)



Client functions ->
Server functions ->
Views ->
Tables

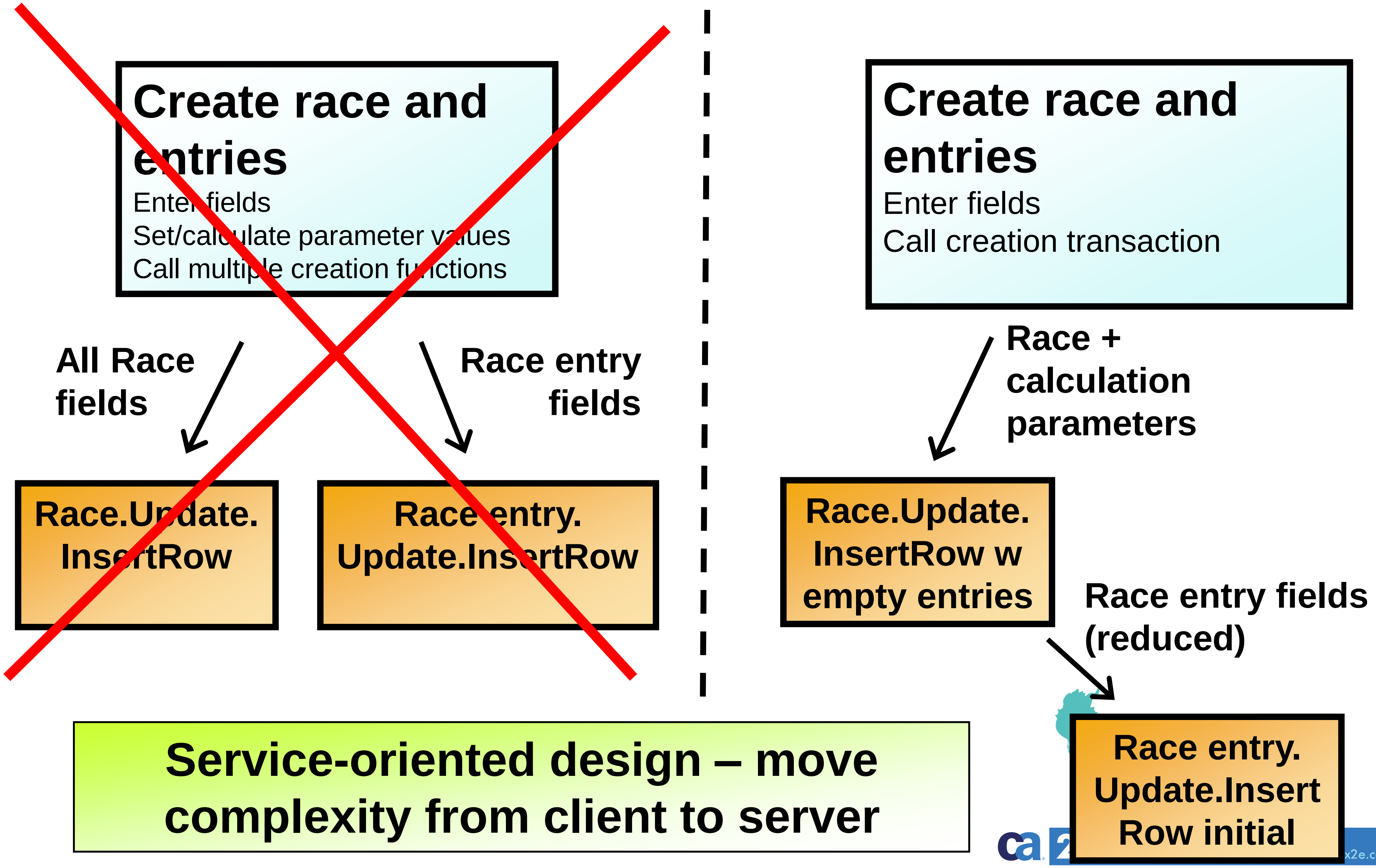


Move Complexity from Client to Server (Websydian)



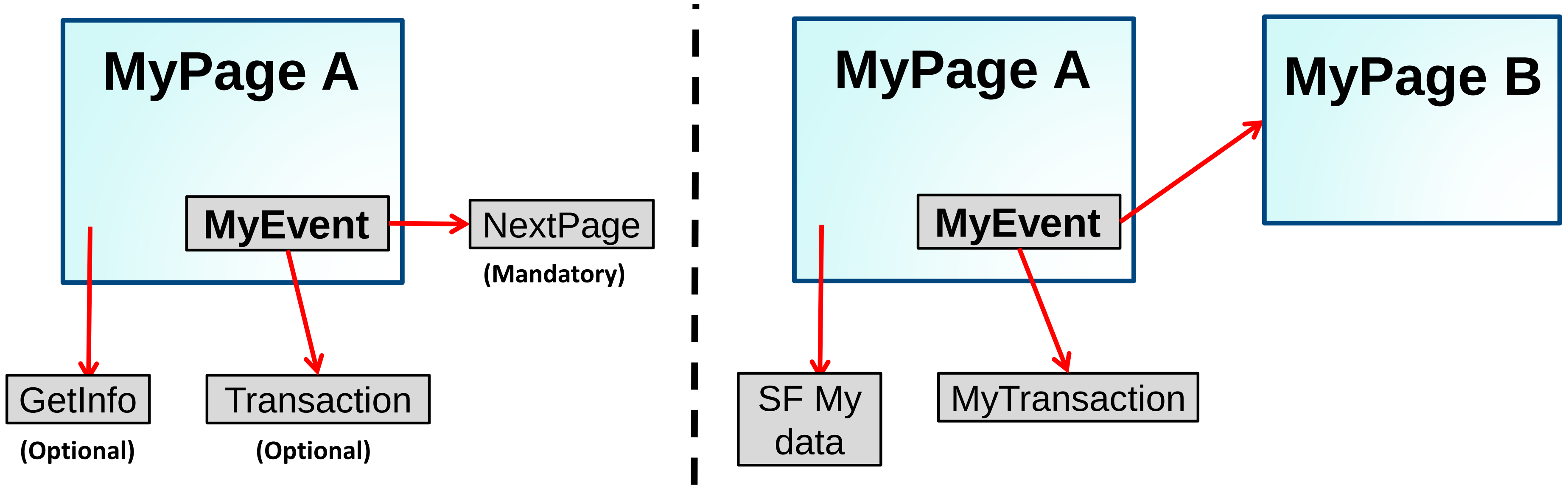
Testable units
More reuse
Separate business logic from UI
Less coding
Better performance

InsertRow – Calculation Field Input


~~Create race and entries~~
~~Enter fields~~
~~Set/calculate parameter values~~
~~Call multiple creation functions~~
~~All Race fields~~
~~Race entry fields~~
~~Race.Update. InsertRow~~
~~Race entry. Update.InsertRow~~
Create race and entries
Enter fields
Call creation transaction
Race + calculation parameters
Race.Update. InsertRow w empty entries
Race entry fields (reduced)
Race entry. Update.Insert Row initial

Service-oriented design – move complexity from client to server

Abstract Patterns Supporting a Service-Oriented Design (Websydian)

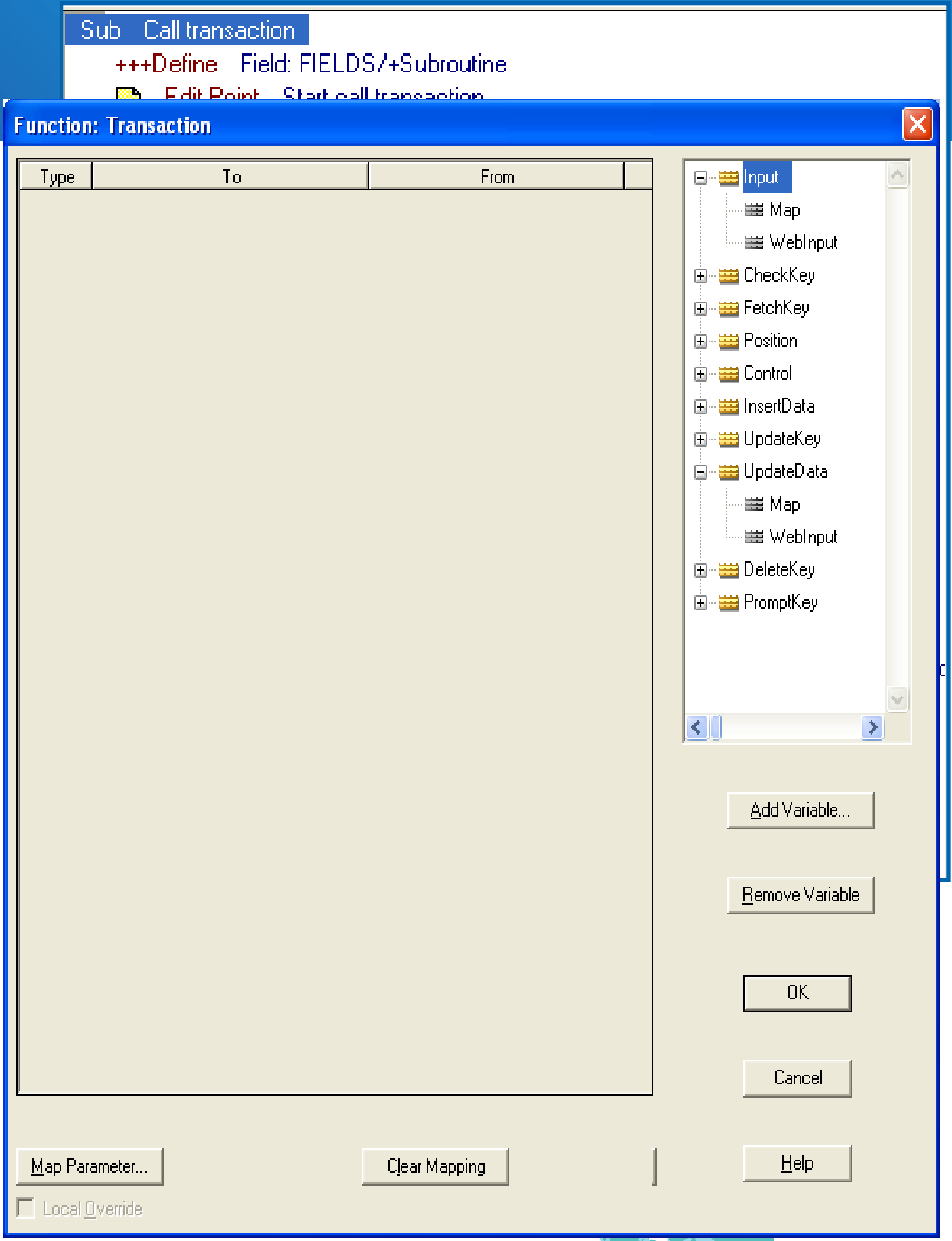


MyPage A	…	…
	includes FNC	MyEvent
	replaces FNC …by FNC	GetInfo SF My data
MyPage A.MyEvent	…	…
	replaces FNC …by FNC	Transaction SF My data
	replaces FNC …by FNC	NextPage MyPage B

Replace inherited calls

Replace Inherited Calls

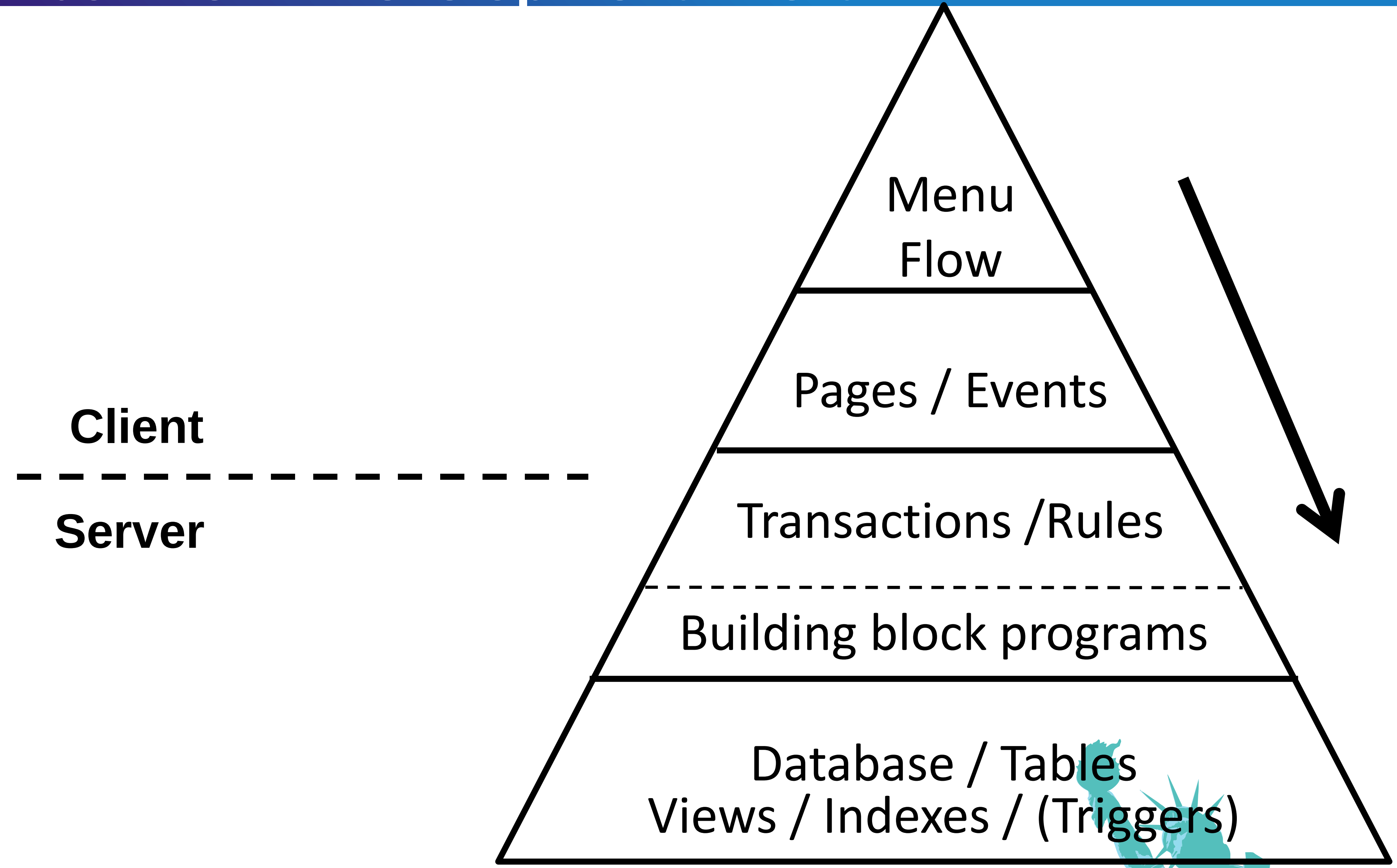
- Code for exception and error handling inherited as well
- *Transaction* function has commonly used variables as input
- Mapping will fall in place at replacement
- Coding effort moved
  - From procedural code
  - To declarative triples
- …and at the same time supporting a better design


Sub Call transaction
+++Define Field: FIELDS/+Subroutine
Edit Point Start call transaction

Function: Transaction

Type	To	From

Validation Specifications and 'Fat' Field Definitions – Push Down Development Effort



‘Fat’ Field Definitions

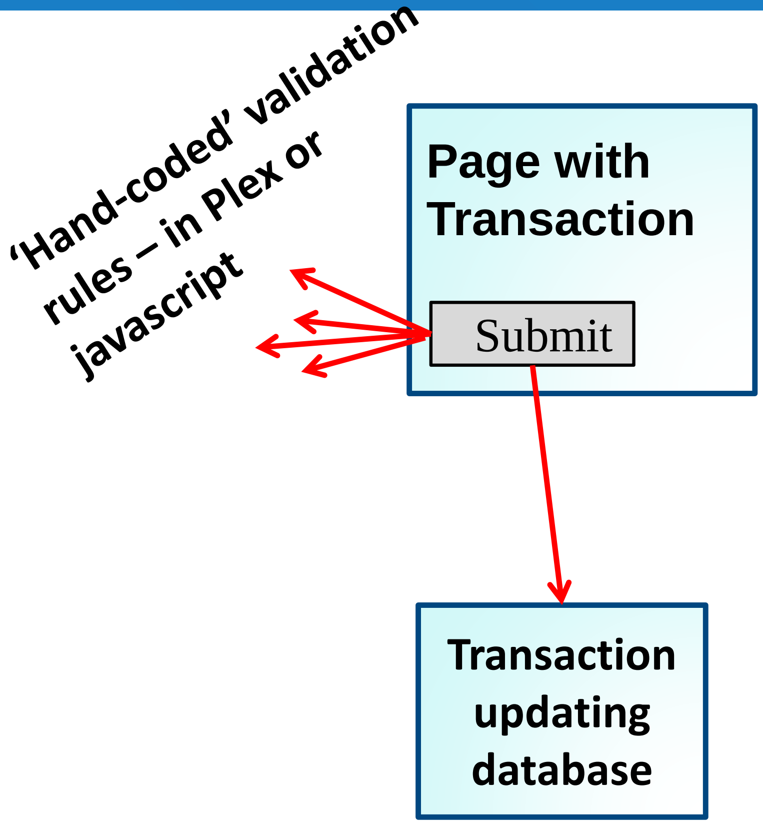
- Presentation
  - Labels, case, edit mode, alignment
- Default values…
- Validation functions ←
- Scoped objects (structure)
  - Values, states, labels, messages, functions etc.
- Other (rarely used)
  - Derived fields, Computed by specifications
- Various…

Rule specifications picked up by inherited meta code

‘Fat’ Field Definitions – Less Work on Client Specification and Development

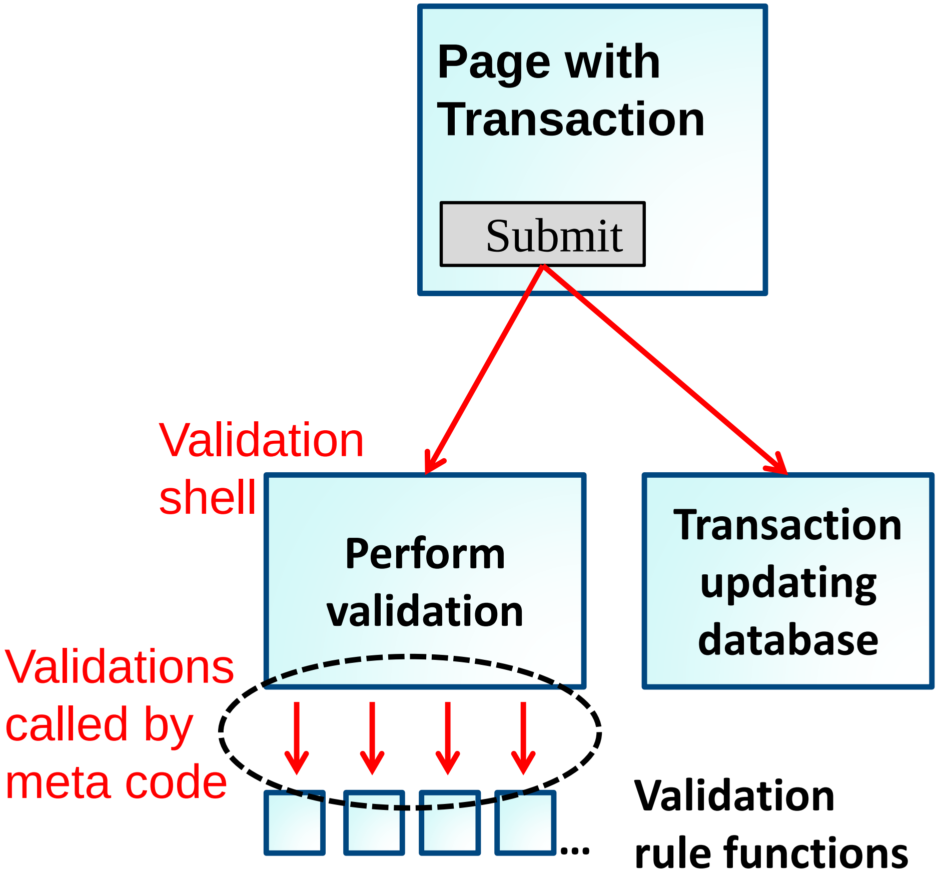
Preconditions of Called Transaction – Capture Rules in Plex

- Mandatory fields
- Validation functions – Simple
  - Entered (status) field value is defined
  - Related record exists (trivial look-up)
  - Other rules based on simple logic
- Validation functions – Complex
  - Depend on other input
  - Depend on database contents


'Hand-coded' validation rules – in Plex or javascript
Page with Transaction
Submit
Transaction updating database

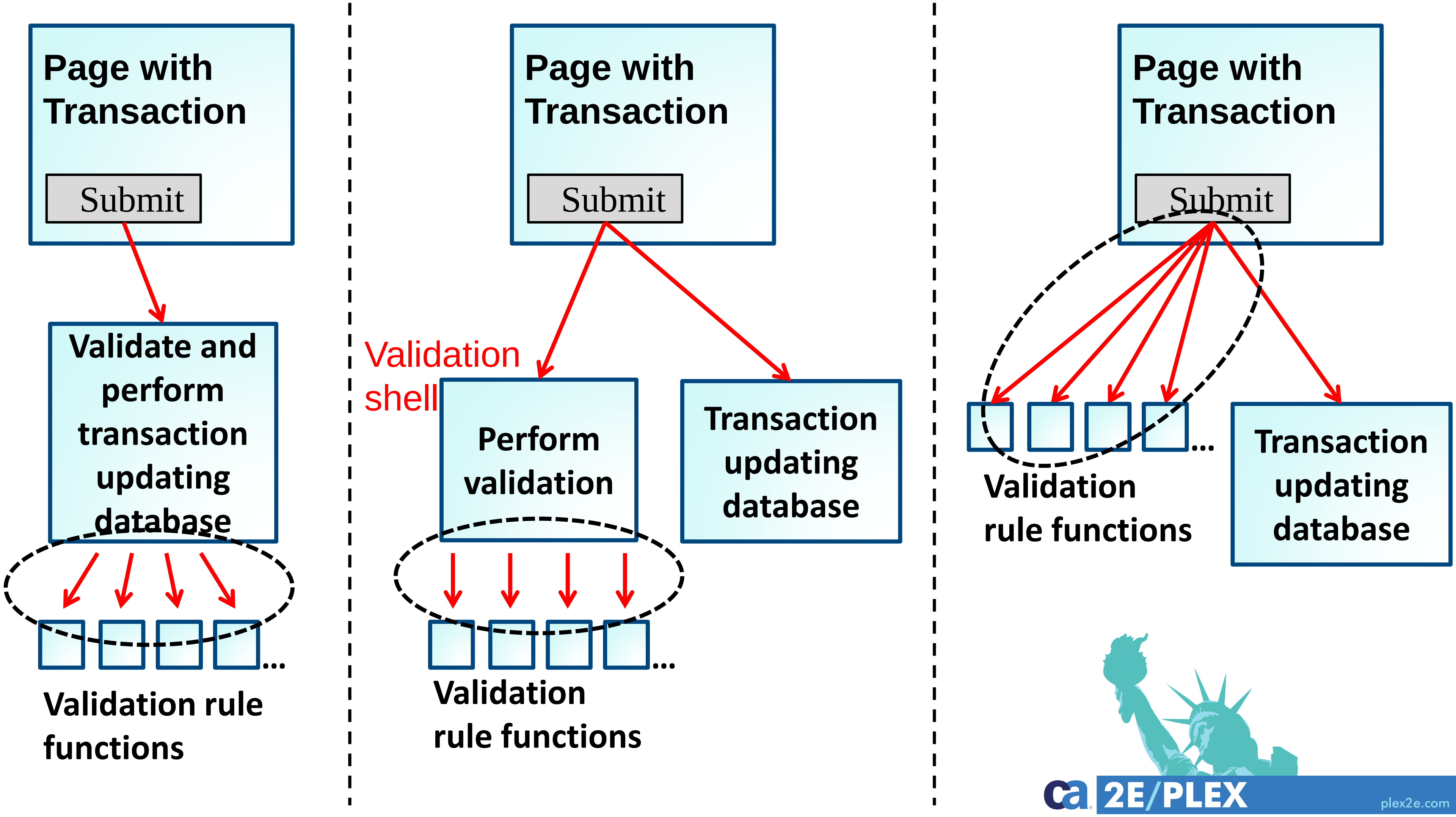
Plex Validation Triples

- **TRP (FLD) optionality SYS**
- **FLD validated by FNC**
- VW validated by FNC
- **TRP (REL) optionality SYS**
- TRP (FLD) validated by FNC
- **TRP validated by FNC**
- **ENT checked by FNC**


Page with Transaction
Submit
Validation shell
Perform validation
Transaction updating database
Validations called by meta code
…
Validation rule functions

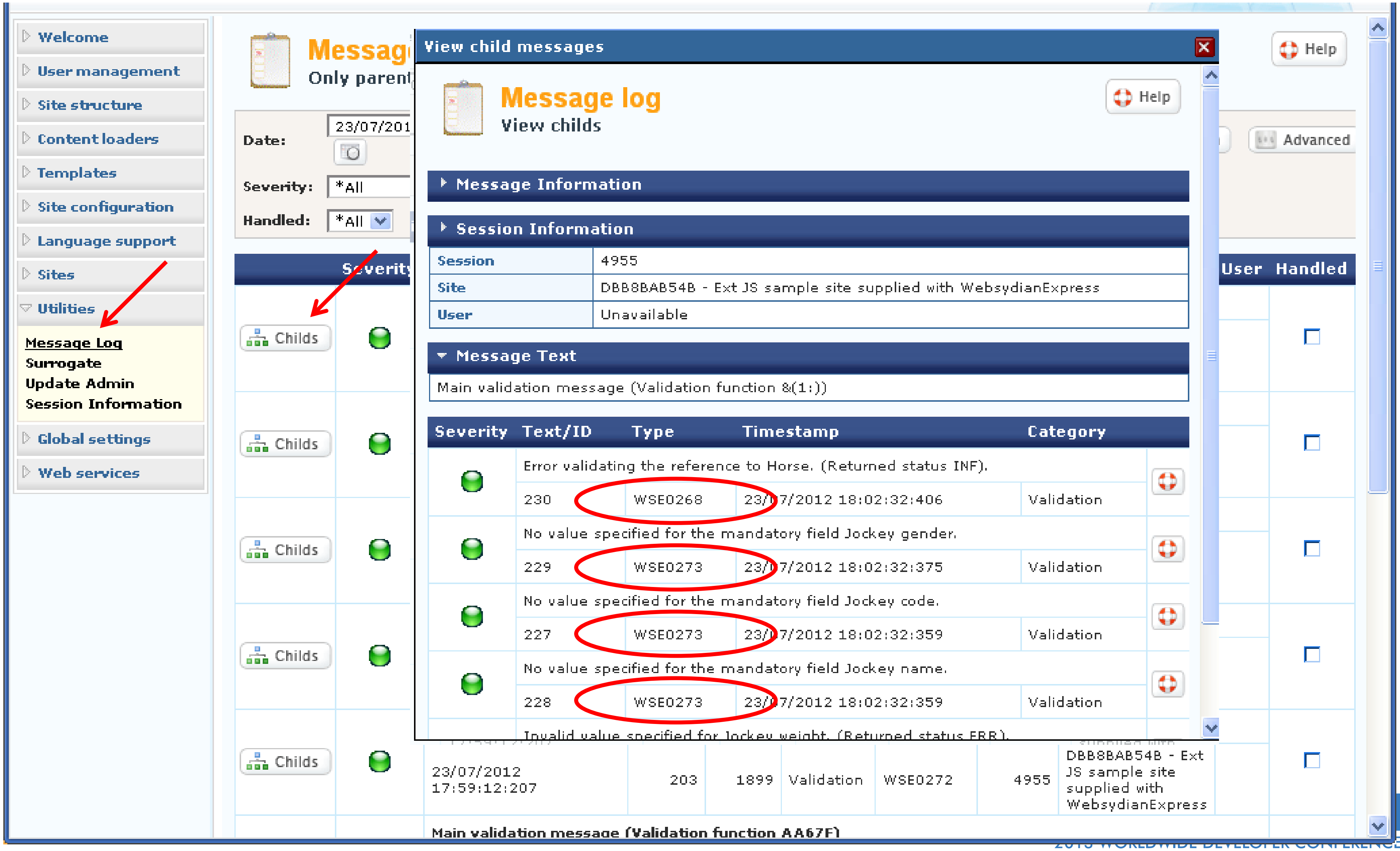
Validation rules specified as part of data model

Different Validation Architectures Based on Same Specifications (Websydian)



Page with Transaction
Submit
Validate and perform transaction updating database
…
Validation rule functions
Page with Transaction
Submit
Validation shell
Perform validation
Transaction updating database
…
Validation rule functions
Page with Transaction
Submit
…
Validation rule functions
Transaction updating database

Websydian Message Log – One Possible Solution for Server-Side Validation



Session	4955
Site	DBB8BAB54B - Ext JS sample site supplied with WebsydianExpress
User	Unavailable

Main validation message (Validation function &(1:))

Severity	Text/ID	Type	Timestamp	Category
●	Error validating the reference to Horse. (Returned status INF).			
	230	WSE0268	23/07/2012 18:02:32:406	Validation
●	No value specified for the mandatory field Jockey gender.			
	229	WSE0273	23/07/2012 18:02:32:375	Validation
●	No value specified for the mandatory field Jockey code.			
	227	WSE0273	23/07/2012 18:02:32:359	Validation
●	No value specified for the mandatory field Jockey name.			
	228	WSE0273	23/07/2012 18:02:32:359	Validation
	Invalid value specified for Jockey weight. (Returned status ERR).			

23/07/2012 17:59:12:207	203	1899	Validation	WSE0272	4955	DBB8BAB54B - Ext JS sample site supplied with WebsydianExpress	

Main validation message (Validation function AA67F)

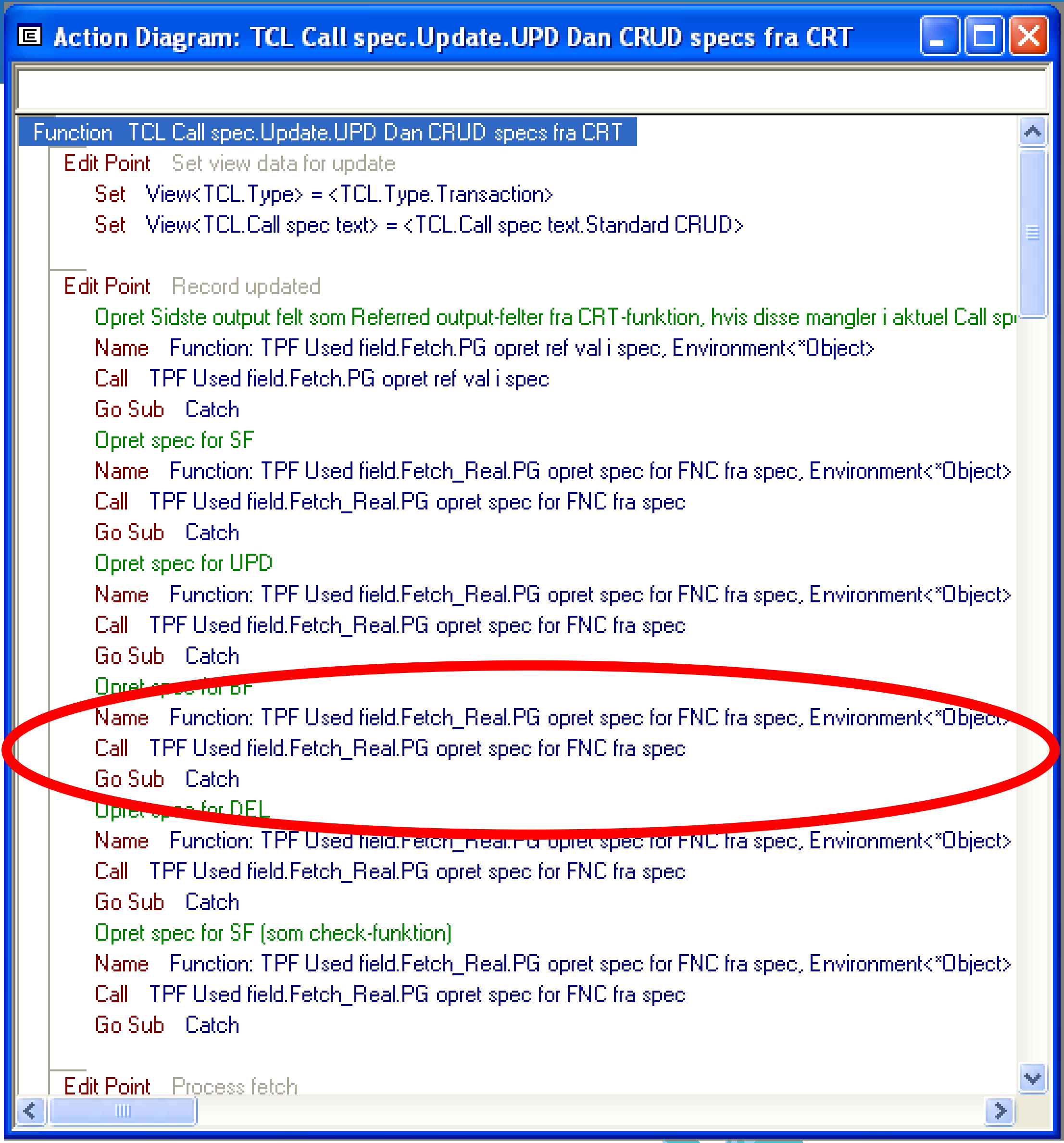
Service-Oriented Architecture – Decisions

- Patterns for support of a service-oriented design?
- Level of use of extended (fat) field specifications?
- Specification of rules in Plex or not – and how to pick up by client?
- Server-side validation or not – and how to implement?
- Validation in ‘mixed’ target environments – Web, Windows/Java UI, 5250

Error Reporting and Sanity Checking

Early Error Detection

- Abnormal **Call status*
- Abnormal/unexpected **Returned status*
- Error states detection
- Write errors to common log



Action Diagram: TCL Call spec.Update.UPD Dan CRUD specs fra CRT

```
Function  TCL Call spec.Update.UPD Dan CRUD specs fra CRT
  Edit Point   Set view data for update
     Set   View<TCL.Type> = <TCL.Type.Transaction>
     Set   View<TCL.Call spec text> = <TCL.Call spec text.Standard CRUD>

  Edit Point   Record updated
     Opret Sidste output felt som Referred output-felter fra CRT-funktion, hvis disse mangler i aktuel Call sp
     Name   Function: TPF Used field.Fetch.PG opret ref val i spec, Environment<*Object>
     Call   TPF Used field.Fetch.PG opret ref val i spec
     Go Sub   Catch
     Opret spec for SF
     Name   Function: TPF Used field.Fetch_Real.PG opret spec for FNC fra spec, Environment<*Object>
     Call   TPF Used field.Fetch_Real.PG opret spec for FNC fra spec
     Go Sub   Catch
     Opret spec for UPD
     Name   Function: TPF Used field.Fetch_Real.PG opret spec for FNC fra spec, Environment<*Object>
     Call   TPF Used field.Fetch_Real.PG opret spec for FNC fra spec
     Go Sub   Catch
     Opret spec for BF
     Name   Function: TPF Used field.Fetch_Real.PG opret spec for FNC fra spec, Environment<*Object>
     Call   TPF Used field.Fetch_Real.PG opret spec for FNC fra spec
     Go Sub   Catch
     Opret spec for DEL
     Name   Function: TPF Used field.Fetch_Real.PG opret spec for FNC fra spec, Environment<*Object>
     Call   TPF Used field.Fetch_Real.PG opret spec for FNC fra spec
     Go Sub   Catch
     Opret spec for SF (som check-funktion)
     Name   Function: TPF Used field.Fetch_Real.PG opret spec for FNC fra spec, Environment<*Object>
     Call   TPF Used field.Fetch_Real.PG opret spec for FNC fra spec
     Go Sub   Catch

  Edit Point   Process fetch
```

Q&A