



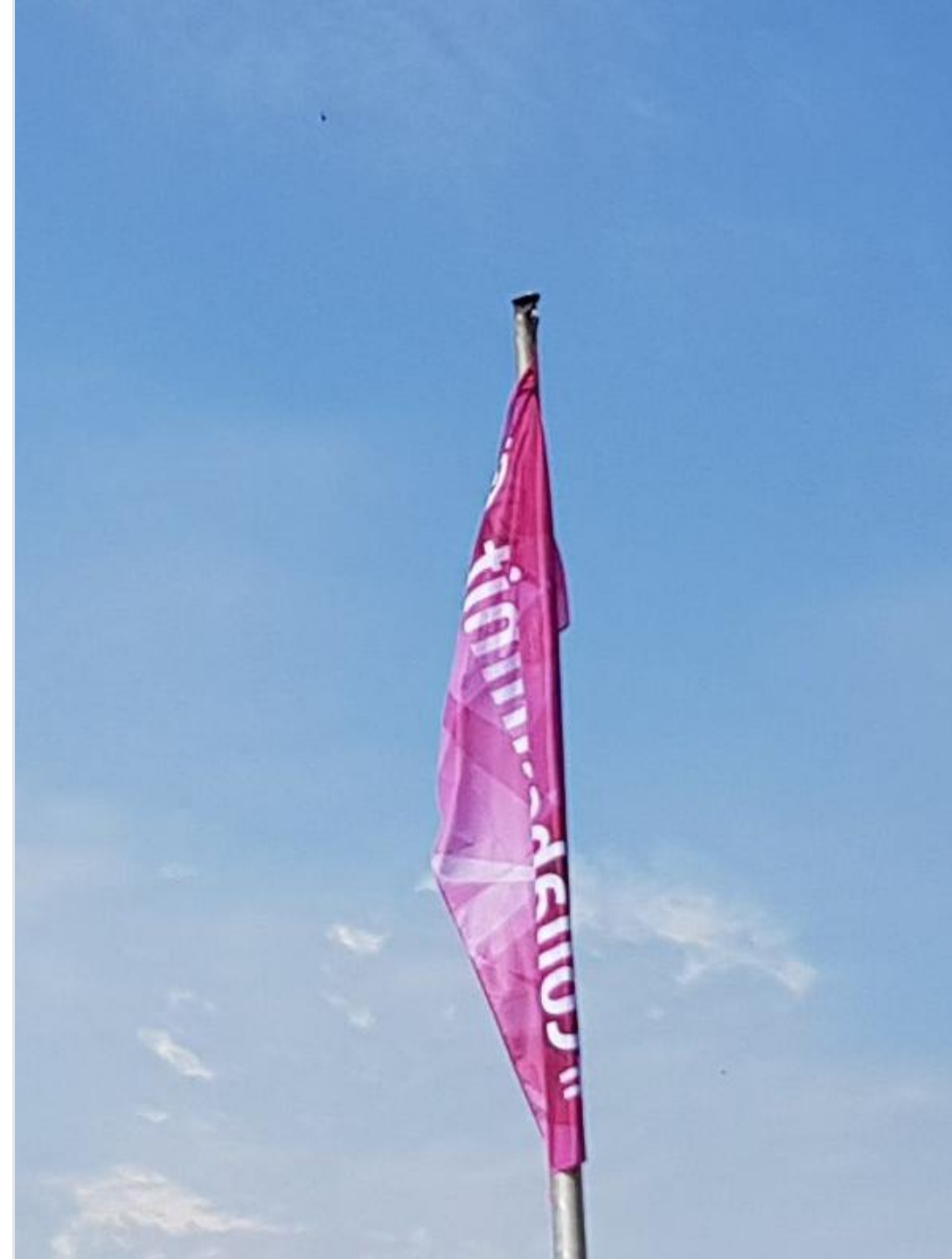
ECS 2025

AN INTRO TO BUSINESS CENTRAL DEVELOPMENT & DELIVERY

TOBIAS FENSTER

MANAGING DIRECTOR @ 4PS GERMANY
CHIEF ENGINEER @ HILTI

AZURE MVP & REGIONAL DIRECTOR
DOCKER CAPTAIN



Premium Partner



Premium Sponsor



Technology Partner



Diamond Sponsor



Platinum Sponsor



Gold Sponsor



Silver Sponsor



Bronze Sponsor



Startup Silver



Startup Bronze



Startup Sponsor





DISCLAIMER

This is meant for people new to or interested in Business Central development and delivery. If you already know how this works, you probably won't learn a lot.



Development in BC (and NAV)



General approach (past and present)

Customer code

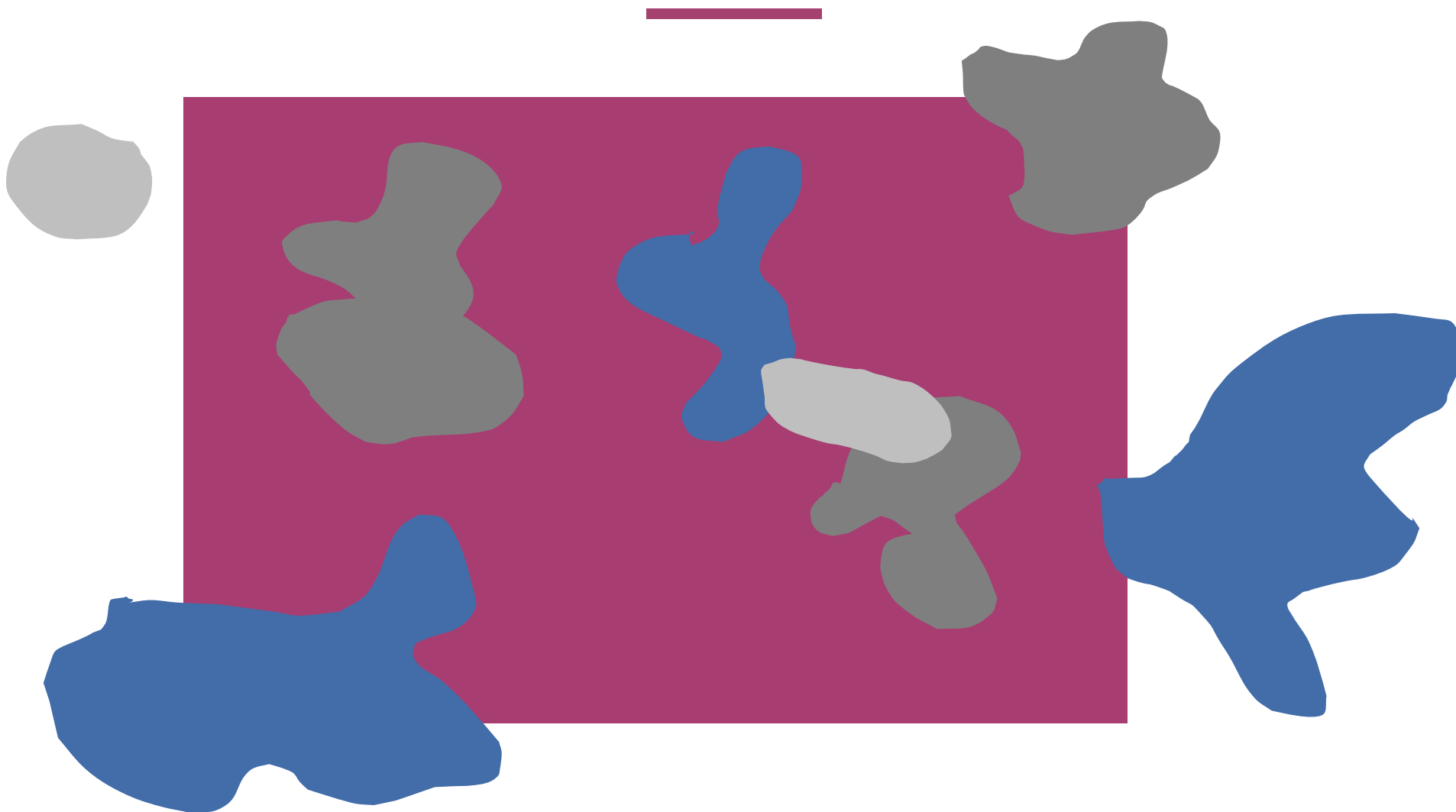
Partner(s) code

Local Microsoft code (DE, UK, ...)

International Microsoft code



Implementation reality (past)





Implementation reality (present)

- You can't change or remove Microsoft code, you can only extend it
 - With a few exceptions
 - And a concept to replace whole “modules” if they are designed for it
- You can create for the whole world (AppSource) or for a customer (“Per Tenant Extension” PTE)
 - But you must say which localizations you support
- Overall a lot cleaner, better to understand and maintain
 - But also more limited



General structure

- Business Central with classic 3-tier architecture: database, middle/application layer, (web) client
- Code is imported into the database, middle layer generates C# code, SQL structures, front end
- Because of generative approach, developer has far-reaching, but not complete influence on all layers



Tooling (past): C/SIDE

The screenshot shows the SAP C/AL Editor window titled "Codeunit 50000 CrInvForCust10000And70000 - C/AL Editor". The editor contains the following code:

```
Documentation()
OnRun()
LOCAL CreateInvoiceDocument(VAR SalesHeader : Record "Sales Header")
WITH SalesHeader DO BEGIN
    INIT;
    "Document Type" := "Document Type"::Invoice;
    "No." := '';
    INSERT(TRUE);
END;

LOCAL TestIfCustomerIsBlocked()
WITH Customer DO BEGIN
    GET('10000');
    TESTFIELD(Blocked, Blocked::" ");
END;

LOCAL SelectCustomer10000(VAR SalesHeader : Record "Sales Header")
WITH SalesHeader DO
    VALIDATE("Sell-to Customer No.", '10000');

LOCAL SaveHeaderToDatabase(VAR SalesHeader : Record "Sales Header")
WITH SalesHeader DO
    MODIFY(TRUE);

LOCAL CreateSalesLine(VAR SalesHeader : Record "Sales Header";VAR SalesLine : Record "Sales Line")
LOCAL TestIfItemIsBlocked()
LOCAL SelectItem70000(VAR SalesLine : Record "Sales Line")
LOCAL SaveLineToDatabase(VAR SalesLine : Record "Sales Line")
LOCAL OpenSalesInvoicePage(VAR SalesHeader : Record "Sales Header")
```



Tooling (present): VS Code + AL extension

```
app > src > Subcontracting > AL PurchaseOrder4PD.PageExt.al > {} PageExtension 11316765 "Purchase Order 4PD"
0 references | Simon Häuser, 7 months ago | 2 authors (Simon Häuser and one other)
1 pageextension 11316765 "Purchase Order 4PD" extends "Purchase Order" Simon Häuser, 2 ye
2 {
3     Simon Häuser, 15 months ago | 2 authors (Simon Häuser and one other)
4     layout
5     {
6         movebefore("Invoice Received Date"; "Subcontracting Invoice")
7         addafter("Subcontracting Invoice")
8         {
9             0 references | Simon Häuser, 15 months ago | 2 authors (Simon Häuser and one other)
10            group(Subcontracting4PD)
11            {
12                ShowCaption = false;
13                Visible = Rec."Subcontracting Invoice";
14            }
15            0 references
16            field(SubcontractNo4PD; Rec."Subcontract No.")
17            {
18                ApplicationArea = All;
19                ToolTip = 'Specifies the value of the Subcontract No. field.';
20                Comment = 'de-DE=Nachunternehmervertrag Nr. | nl-NL=Hiermee wordt de
21                waarde van het veld Onderaannemerscontractnr. opgegeven.';
22                Editable = (Rec.Status <> Rec.Status::Released) and (Rec.Status <>
23                Rec.Status::Closed);
24            }
25            0 references
26            trigger OnAssistEdit()
27            var
28                SCContractMgt: Codeunit "SC Contract Mgt. 4PD";
29            begin
30                SCContractMgt.ShowSubcontractAndCreateOptionally(Rec);
31                CurrPage.Update(false);
32            end;
```



Development language (C/)AL

- 4th generation programming language ("4GL") with a much more generative approach than 3GL (C#, Java, Pascal, ...)
 - Developer writes business code, engine generates backend (C#) / frontend (HTML, JavaScript etc) code
- Optimised for the development of application logic
- Very quick to learn, relatively easy to use
 - But like everywhere else, it takes talent and hard work to be above average...
- Pascal-like syntax
- Past C/AL, present AL



Objects in AL

- Objects in Business Central != Object-orientation (OOP) in other languages
- Code and structures are stored in different object types:
 - Tables (correspond to tables on SQL Server)
 - Pages (UI for displaying data, among other things)
 - Codeunits ("containers" for business logic, roughly comparable to DLLs)
 - Others for different purposes (reports, XML ports, profiles, ...)



DEMO: Development



Delivery in BC (and NAV)



General approach (past)

- Very manual process
 - Develop in your local database (not file based, no real version control!)
 - Extract the changed objects into a file (".fob")
 - Connect to the target environment
 - Import the .fob file
- Could be automated in pipelines, but very cumbersome and with ugly workarounds



General approach (present)

- File-based development, resulting in modular .app packages
- Build in build pipelines
 - AL-Go by Microsoft works in GitHub
 - ALOps works on Azure DevOps
 - COSMO Alpaca works on Azure DevOps and GitHub (and provides development environments)
- Release publicly available solutions to AppSource
- Release customer-individual solutions directly to customer environments
 - Also possible directly from VS Code, but you shouldn't...



General approach (present)

- Test automation as part of the code
 - Typically run in pipelines as well
- Relatively new feature: page scripting
 - Non-technical people record steps they take in the client
 - Results in YAML files describing the steps
 - Can be exported and imported into other environments
 - Can run in pipelines as well



DEMO: Delivery

THANK YOU,
YOU ARE AWESOME ❤️

PLEASE RATE THIS SESSION
IN THE MOBILE APP.

Find me on tobiasfenster.io

