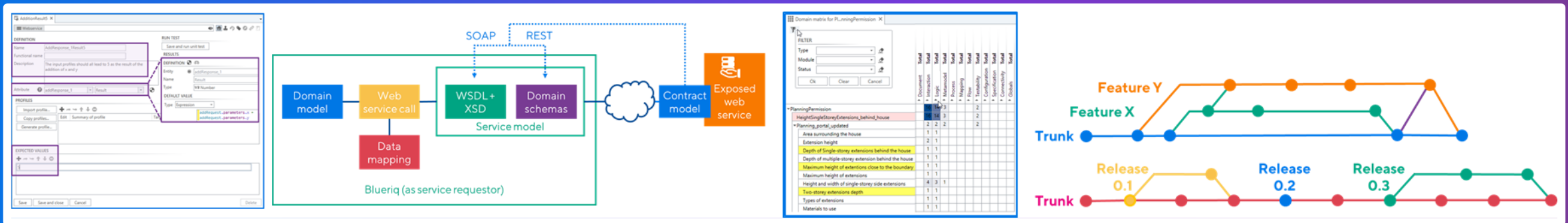


Business Modeling Professional

A next step in Blueriq: setting up a project,
choosing module and library structure, version management,
delivering a well-documented, tested and maintainable solution



blueriq

Goals

Mastering

- Setting up a project
- Translating Requirements to logic
- Complex logic and unit testing

Being able to

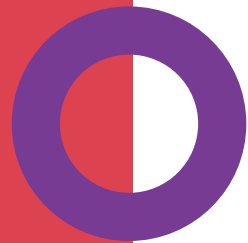
- Document your work
- Implement specifications
- Deliver your application

Understanding

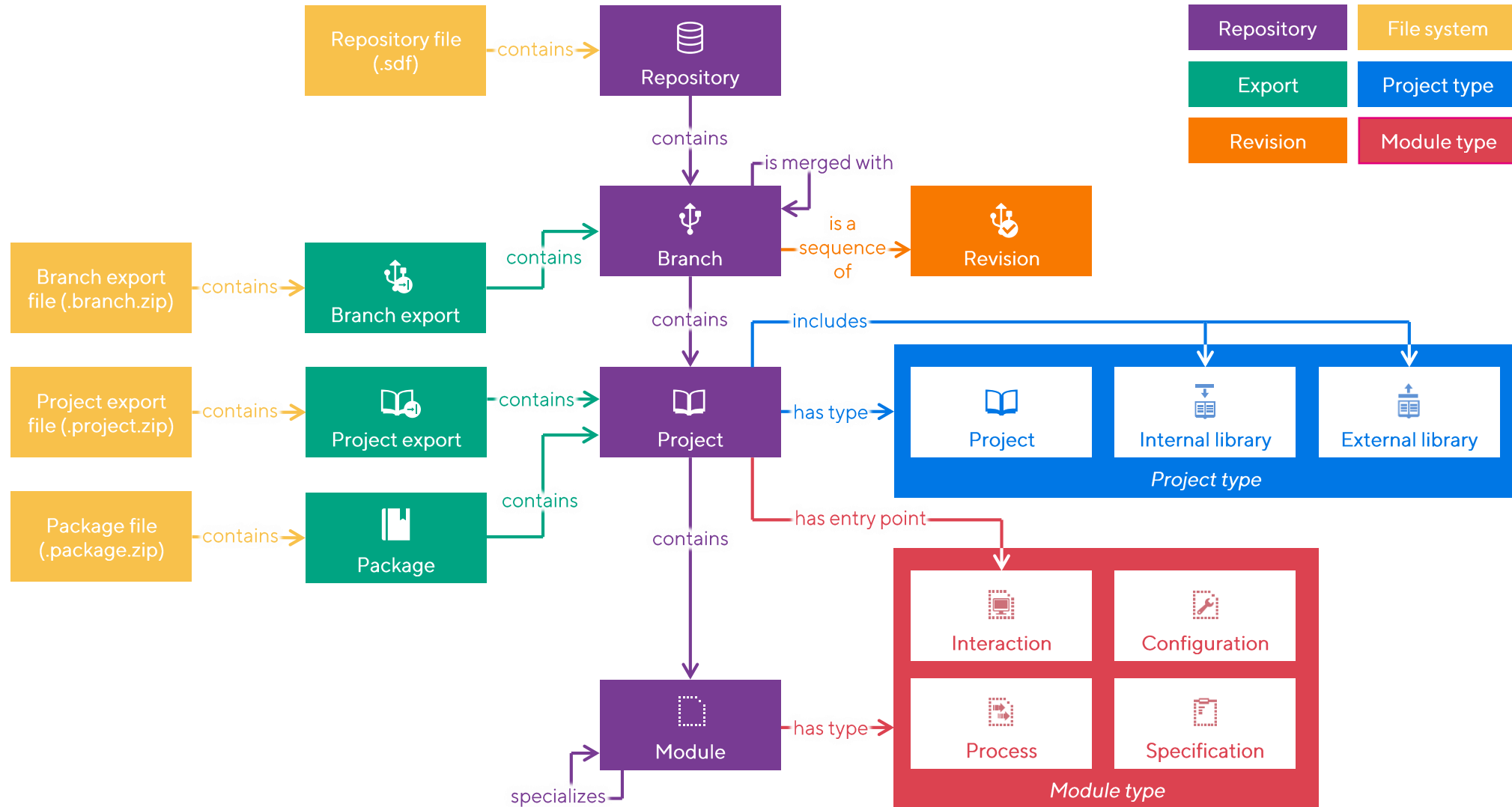
- Webservices

Being introduced to

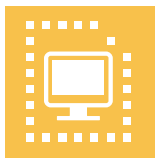
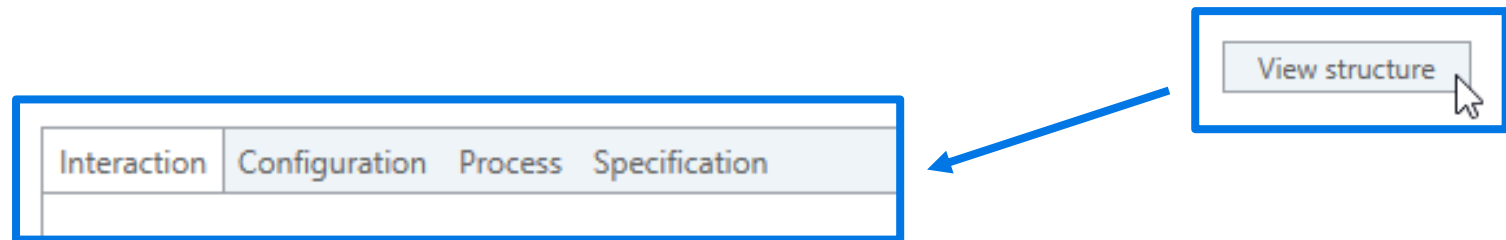
- Variety of Blueriq tools



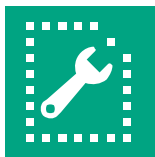
Blueriq application structure



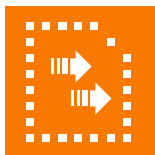
Module types



An **interaction** module is for modeling interaction, documents and services



A **configuration** module is for modeling mappings between modules



A **process** module is for modeling (dynamic) business processes



A **specification** module is for linking specifications to your project

Blueriq application structure



Ship complete **repository.sdf**

- Only when **everything** in the sdf has to be distributed, which is unusual



Export a **package**, containing projects and libraries

- Import in studio, choose what to unpack. Internal libraries now external



Export a **project** as XML for another runtime environment

- Typically to T, A or P, either manually or with the Blueriq Publisher



Export a **branch**, containing 1 version of a complete repository

- Including all its projects and libraries

Branching



...is “the **duplication** of an object under version management, so that **modifications** can be made **in parallel** along multiple branches”



Each repository contains a first branch (no parent): **Trunk**

- A.k.a. MainLine, Master or InitialBranch



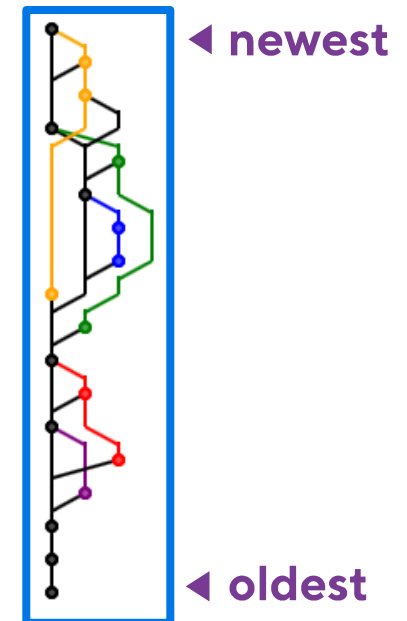
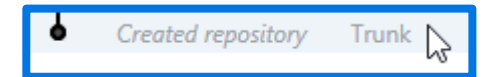
A **new branch** is effectively a **copy** of the entire repository

- Multiple teams can work in parallel, each in their own branch



Two branches can also be **merged** again

- Changes in each branch must be **registered** or committed first



Branching scenarios



No branches

- Unstable trunk
- Only for small teams, one location



Branch for **feature**

- Create a branch per feature
- Work in parallel on separate features



Branch for **release**

- Create branch before release to stabilize
- Merge changes from release to trunk



Branch for **customer**

- Separate branch per customer/product



Branch for **maintenance**

- Create branch for a specific hotfix for a few customers



Branch for **team**

- Create branch for sub-teams

Practical guidelines



Organization

- Choose a workflow
- Plan when to merge
- Use the 2-man rule
- Limit nr of active parallel branches
- Evaluate the strategy & discuss
- Think about project architecture & lifecycle



Strategy

- Choose a strategy / branching scenario
 - Stable/Unstable trunk
 - Fits the organization
 - and the stories / epics
- Before merging to Trunk, pull Trunk into feature branch to solve conflicts
- Commit Trunk afterwards
- Use clear names



Revisions

- Test your model
- Use clear and to the point register messages
- ... that cover the changes
- Register and merge regularly
- ... and with (a) reason
- Use tags (required for the publisher)



Case



[planningportal.gov.uk/
permission/](https://planningportal.gov.uk/permission/)



Planning permissions portal

- Propose and design a solution for local communities

Build a proof of concept

- Interactive personal forms
- Decision management
- Connectivity

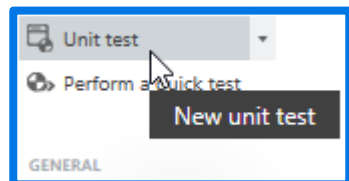
Work in small teams

- You'll be asked to show your progress



Making a Unit test

New unit test



- Select **attribute** to test (and its sourcing logic)
- Give unit test a meaningful **name** and **description**
- Specify **1 expected value** (>1 only if attr is multivalued)
- Add **test cases...**

A screenshot of the 'AdditionResult5' unit test configuration window. The window is divided into several sections:

- DEFINITION**: Contains fields for Name (AddResponse_1Result5), Functional name, and Description (The input profiles should all lead to 5 as the result of the addition of x and y).
- ATTRIBUTE**: A dropdown menu showing 'addResponse_1' and 'Result'.
- PROFILES**: A section with buttons for 'Import profile...', 'Copy profiles...', and 'Generate profile...'. It also includes a table with columns 'Edit', 'Summary of profile', and 'Target'.
- EXPECTED VALUES**: A section with a text input field containing the value '5'.
- RUN TEST**: A button labeled 'Save and run unit test'.
- RESULTS**: A section with a 'DEFINITION' sub-section showing 'Entity' (addResponse_1), 'Name' (Result), and 'Type' (1/2 Number). It also includes a 'DEFAULT VALUE' section with a 'Type' dropdown set to 'Expression' and a text input field containing the expression 'addRequest.parameters.x + addRequest.parameters.y'.

At the bottom of the window are buttons for 'Save', 'Save and close', 'Cancel', and 'Delete'.



Justifications container

Display justification(s) in special container **AQ_Justifications**

Parameter	Direction	Value
attribute-path	Input	Applicant.PersonalPremium
detail-level	Input	ALL
formats	Input	text, TRUE

Show justifications in source tree of **this attribute**

ALL
PRIMARY
SECONDARY

All sub-decisions included
Just the last level (final decision)
Final decision and sub-decisions

ALL
ALL
PRIMARY
SECONDARY

Simple standard format is **text, TRUE**

Additional coverage

Type: Health

Do you smoke? ☒ Yes ☐ No

Premium

Premium before discount	€ 10.00
Discount amount	€ 1.00
Risk surcharge	€ 5.00
Premium	€ 14.00

Justification

Additional health for smokers means a risk surcharge of €5



Linking Specifications

Open specification

Link **all model elements** that are **relevant** for its implementation

Describe the relevancy per element

Check '**Is implemented**' when it works
(spec changes to white in overview)

A specification can also

- be **related** to another
- have **references**, e.g. to legislation

PlanningPermission.A...rrounding_house X

SpecificationPermissions

Area surrounding the house

Technical name PlanningPermission.Area_surrounding_house

Type rule

REQUIREMENT

No more than half the area of land around the original house would be covered by additions or other buildings.

LINKED ELEMENTS

Element	Description	Module	Project
PermissionAreaSurroundingHouse	BR that denies permission	PlanningPermissions	PlanningPermiss
Area_of_land_around_house	calculation of free space area	PlanningPermissions	PlanningPermiss
ExtensionCoverageToFalse	unit test permission denied	PlanningPermissions	PlanningPermiss
ExtensionCoverageToTrue	unit test permission granted	PlanningPermissions	PlanningPermiss
Building.Area		PlanningPermissions	PlanningPermiss
Plot.Area		PlanningPermissions	PlanningPermiss
Area_surrounding_house	justification for refusal	PlanningPermissions	PlanningPermiss

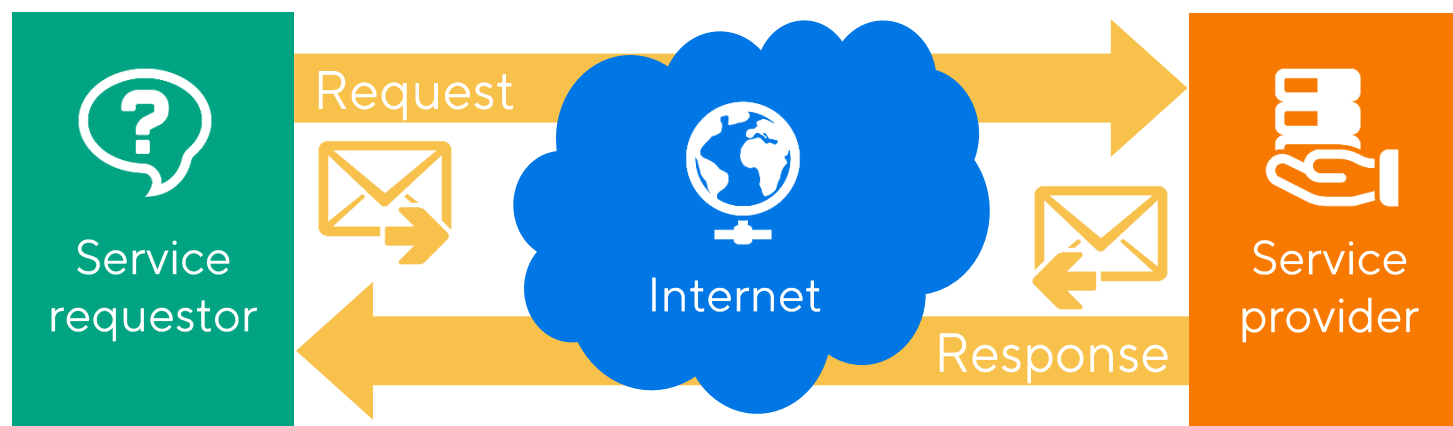
SEE ALSO

REFERENCES

Text	Display text

Web services

A web service is a **way of communication** between systems to **exchange data**



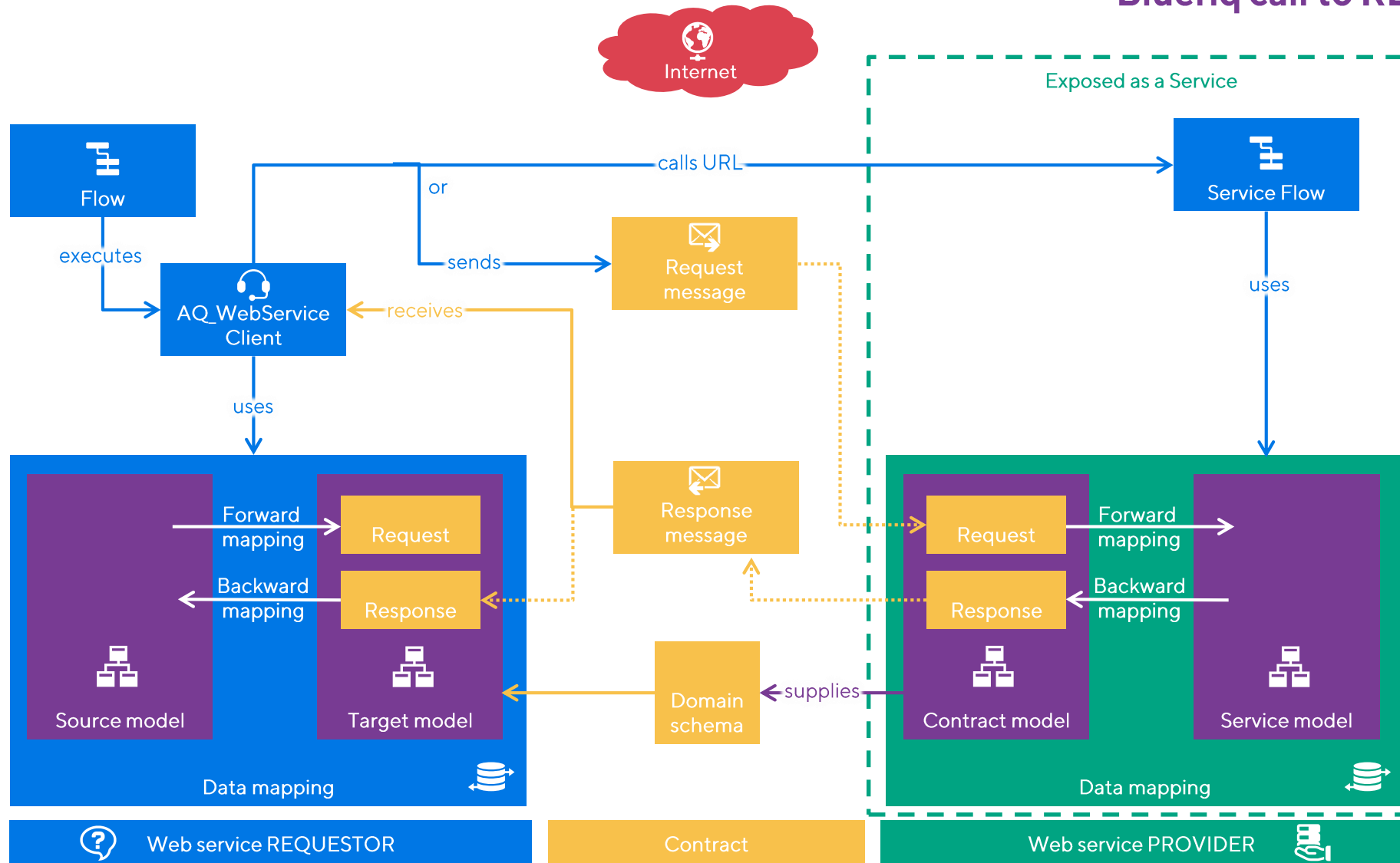
Communication is based on

A **protocol**, like **SOAP** or **REST**

A **contract**, a data structure in a specific format like XML or JSON

Web services in Blueriq

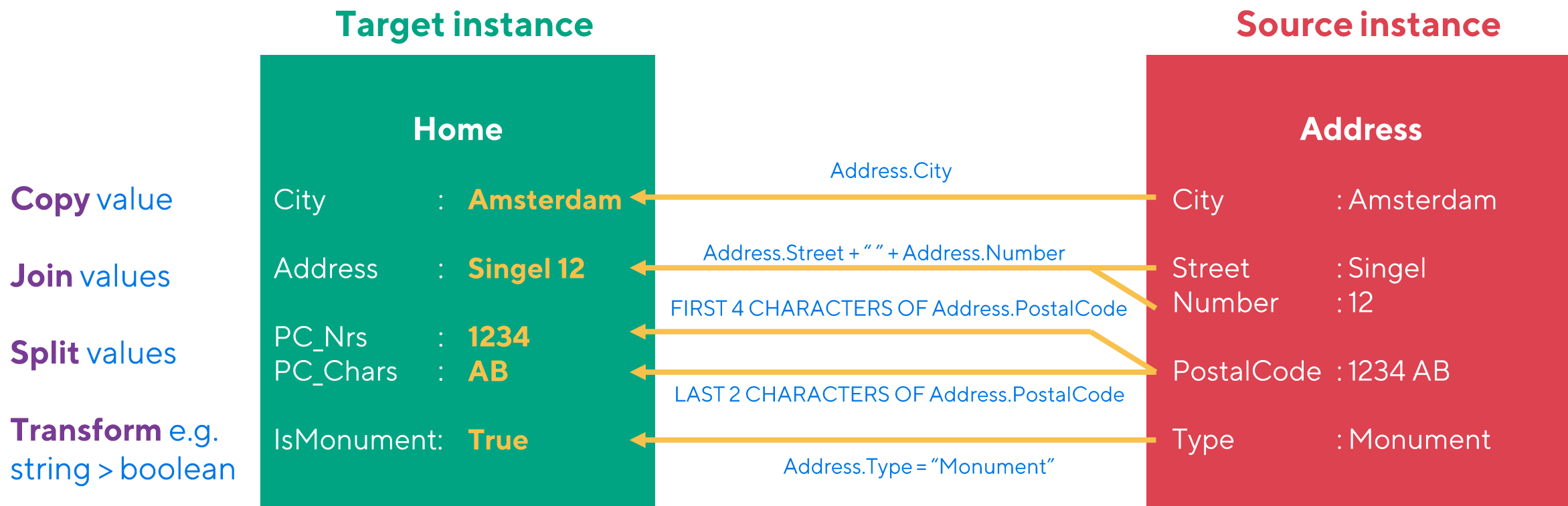
Blueriq call to REST webservice



Example mapping – Target < Source



Simple **one-on-one mapping** (1 target instance gets data from 1 source instance), including **value transformations**

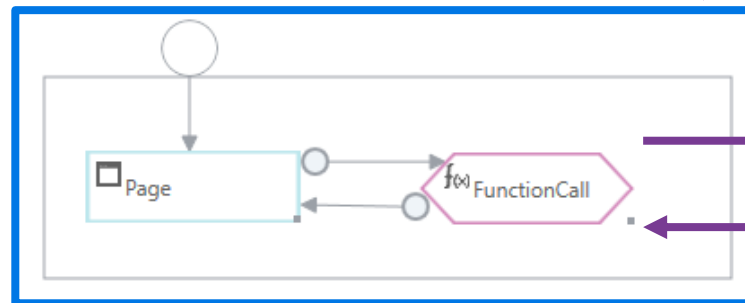


Functions



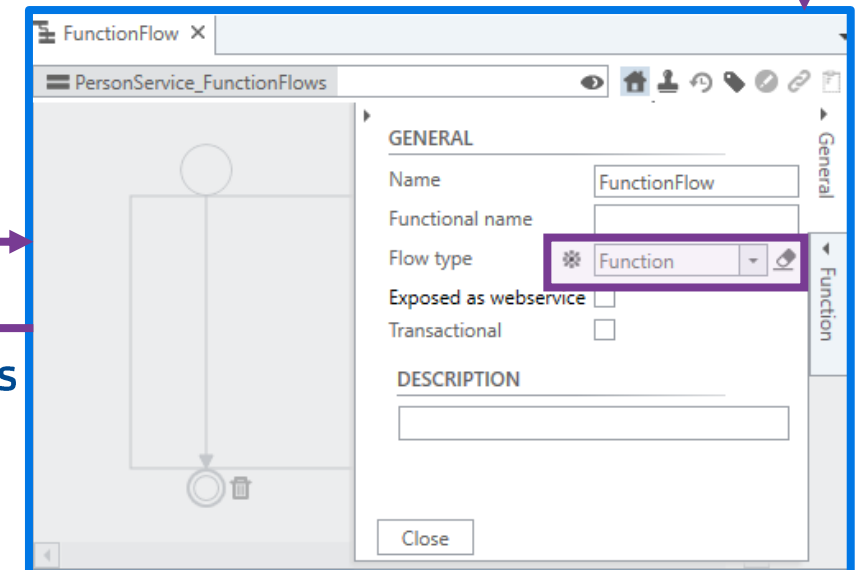
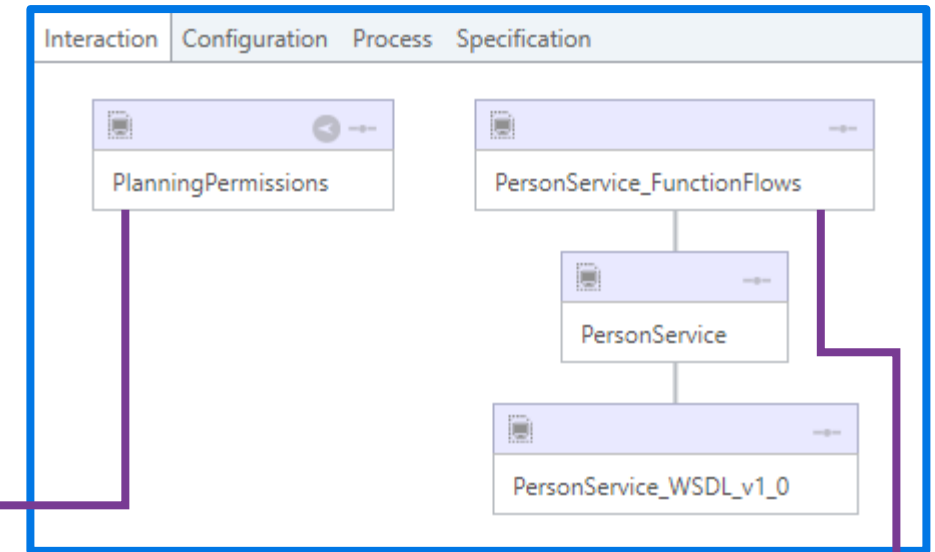
In Blueriq v10.2 **Functions** were introduced

- **Function call** in one module...
- Calls **Function flow** in another,
- **In- and output via parameters**



Input parameters

Output parameters





Delivery

When you **deliver** your model, the receiving party will expect it to work **without problems...**

And if there are **issues** to be fixed, or **changes** are needed, it **should be easy** to find out where and how the model can be adjusted

All delivery suggestions aim to support that goal!

Blueriq Continuous Improvement Toolset



Maintain & Improve

- Unit test coverage
- “Code” quality
- Configure quality parameters

Insight & Refactor

- Insight
- Change – Impact preview
- Change – Execute

Control libraries/projects

- See where reusable elements are used
- Distribute libraries
- Publish projects