

A decorative graphic of white, wavy, overlapping lines that create a sense of motion and depth, spanning across the middle of the slide.

# Efficient Data Processes

ORSA as an Example

Sara Busato

Sept 29, 2023

# INTRODUCTION

## Something about myself



systemorph 

### Sara Busato

born in Venice, Italy

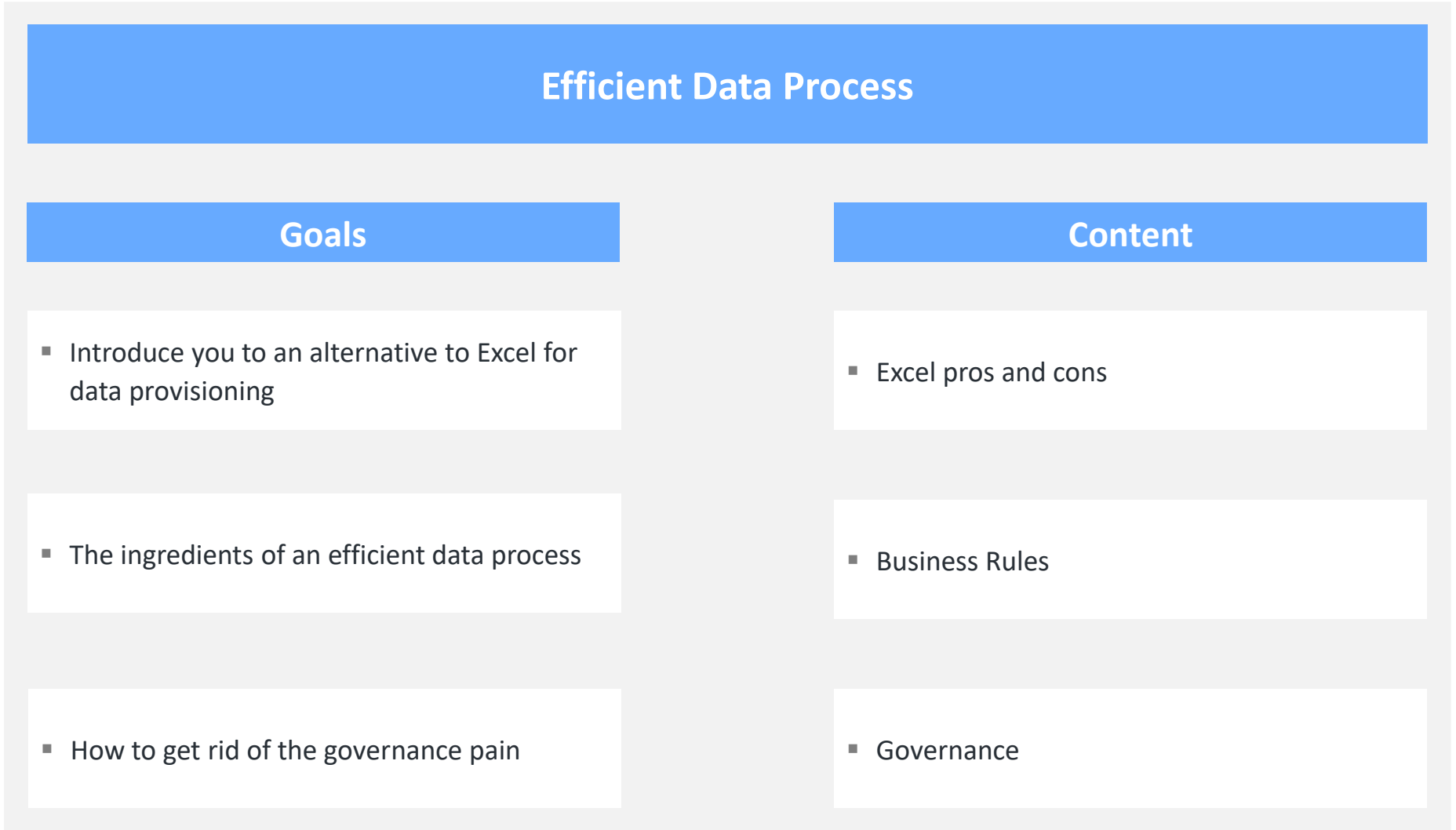
living in Berlin, Germany since 2022

- B.Sc. & M.Sc. in Mathematics, University of Udine, Italy in 2017
- Senior Analytics Engineer, Systemorph, Zurich since 2017
- Working with insurance and reinsurance clients to gather specifications and implement requirements using the Systemorph platform.
- Managed projects and PoCs on IFRS17, Reserving, ORSA, Group MCEV Reporting System...

# INTRODUCTION



## Goals and content of the presentation



# EXCEL THE «GO-TO TOOL»



## Why do we love Excel?



User friendly: intuitive, ease of use and learning



Widely used in the market



Costs: commonly available in most organizations



It can perform basic and advanced calculations and give visual representation

**Actuaries are in control until...**



# WHEN EXCEL IS NOT ANYMORE THE «GO-TO TOOL»

Why Excel is not always the right answer?



...until it gets out of hand...

Special cases in formulas are difficult to track

Changes are difficult to spread

Very often involves a lot of manual work

Business Rules

Difficult to document

It doesn't provide any governance

Governance

---

# BUSINESS RULES

# BUSINESS RULES: AUTOMATING YOUR REPORTING PROCESS



## Definition



### Data Model

- Data is structured
- Data is stored in a proper data storage
- It provides a querying language

### Business Domain Language

- Implementation is as readable as the specifications
- Implementations needs to be flexible to accommodate exceptions

### Documentation

- Possibility to document transparently the modelling choices

**The computational notebook format allows you to bring all these points together in the same artifact**

# BUSINESS RULES APPLIED ON A REAL-WORLD BUSINESS PROCESS



## Recap on ORSA



## Own Risk and Solvency Assessment

ORSA covers all the processes and procedures undertaken by insurers to:

- identify, evaluate, monitor and manage risks during the planning period, as well as those for reporting
- determine **capital adequacy (i.e., comparing capital requirements and capital available) during the planning period**

To this end, they perform a forward-looking self-assessment of their risk situation and capital requirements on at least an annual basis



Too often to perform it manually

This self-assessment enables the insurer to develop a comprehensive overview of its risk and capital situation including the interrelationships between risks and capital



Bring together data from many different sources

Source: <https://www.finma.ch/en/supervision/insurers/cross-sectoral-tools/selbstbeurteilung-der-risikosituation-und-des-kapitalbedarfs-orsa/>



# BUSINESS RULES: DATA MODEL



What are the basic rules to define a good data model



```

+ CODE + TEXT
EF
Outline
1. References
1.1. Libraries
1.2. Usings
1.3. Notebooks
2. Business Process
2.1. Business Processes
2.2. States
2.3. Roles
2.4. Attribute
2.5. Data Packs
3. Data Infrastructure
3.1. Interfaces
3.2. Abstract Classes
4. Dimensions
4.1. Line Of Business
4.2. Economic Basis
4.3. Legal Entity
4.4. Business Segment
4.5. Amount Type
4.6. Report Type
4.7. Estimate Type
4.8. Scenario
4.9. Shock
4.10. Data Node

5.4 Scenario Parameter
ORSA's forward-looking perspective is expressed through various scenarios covering the entire planning period.
The Scenario Parameters define the shock that is applied to the planning data to simulate the insurers' individual risk situation.

1 [BusinessProcess(BusinessProcesses.ScenarioUpdate)]
2 public record ScenarioParameter : KeyedRecord
3 {
4     [Required]
5     [NotVisible]
6     [IdentityProperty]
7     [Dimension(typeof(int),nameof(AccidentYear))]
8     [Range(1990, 2100, ErrorMessage = "Value for {0} must be between {1} and {2}.")]
9     public int AccidentYear { get; init; }
10
11     [Required]
12     [NotVisible]
13     [IdentityProperty]
14     [Dimension(typeof(Scenario))]
15     public string Scenario { get; init; }
16
17     [Required]
18     [NotVisible]
19     [IdentityProperty]
20     [Dimension(typeof(AmountType))]
21     public string AmountType { get; init; }
22
23     [Required]
24     [NotVisible]
25     [IdentityProperty]
26     [Dimension(typeof(BusinessSegment))]
27     public string BusinessSegment { get; init; }
28
29     public double Value { get; init; }
30

```

1

It is good practice to keep the data domain, i.e., the data model, separate from any functionality

2

All information on the data model are defined in one place

3

Highly structured data model is required to ensure data quality

@@ScenarioParameter

BusinessSegment	AccidentYear	Scenario	AmountType	Value
H	2024	ORSA1	CI	0.019
M	2024	ORSA1	CI	0.01
H	2025	ORSA1	CI	0.019
M	2025	ORSA1	CI	0.013
H	2026	ORSA1	CI	0.019
M	2026	ORSA1	CI	0.015

# BUSINESS RULES: BUSINESS DOMAIN LANGUAGE



How a good business domain language can support the implementation of formulas



1

The business domain language allows implementation to be clear and readable

2

Exceptions to the default rule are not hidden, but transparent

In order to create different scenarios, we need to shock the actuarial planning data by an adjustment factor that is calculated starting from the imported Scenario Parameters. In particular, this factor obeys to the following logic:

$$\text{Adjustment Factor} = \begin{cases} \text{ORSA Delta Premium} + 1, & \text{if Scenario} \neq \text{Base Case} \\ 1, & \text{otherwise} \end{cases}$$

where ORSA Delta Premium is the weighted Market Premium scenario parameter.

#### 4.1 Adjustment Factor

```
1 public interface IAdjustmentFactor : IScope<CashflowIdentity, ReportStorage>, IDataCube<IAdjustmentFactor>
2 {
3     static ApplicabilityBuilder App(ApplicabilityBuilder builder) => builder.ForScope<IAdjustmentFactor>(s => s
4         .WithApplicability<IAdjustmentFactorScenario>(x => x.Identity.Scenario != BaseCase));
5
6     double Value => 1;
7 }
8
9 public interface IAdjustmentFactorScenario : IAdjustmentFactor
10 {
11     double IAdjustmentFactor.Value => GetScope<IOrsaDeltaPremium>((Identity)).Value + 1;
12 }
```

Exception check

Default implementation

Exception implementation

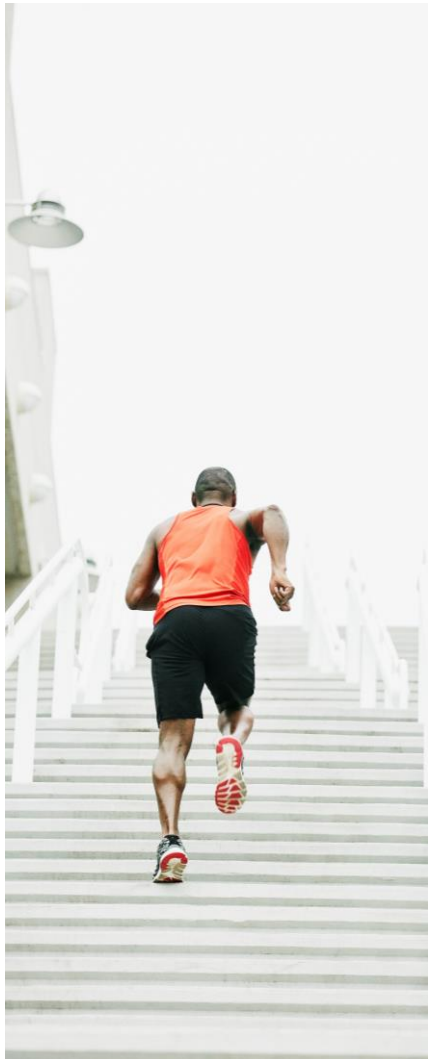
#### 4.2 Adjusted Nominal

```
1 public interface IAdjustedNominal : IScope<CashflowIdentity, ReportStorage>, IDataCube<IAdjustedNominal>
2 {
3     double Value => GetScope<INominalBaseAPE>(Identity).Value * GetScope<IAdjustmentFactor>(Identity).Value;
4 }
```

# BUSINESS RULES: DOCUMENTATION



Documentation as an integral part of the solution



```
1 [BusinessProcess(BusinessProcesses.ScenarioUpdate)]
2 public record ScenarioParameter : KeyedRecord
3 {
4     [Required]
5     [NotVisible]
6     [IdentityProperty]
7     [Dimension(typeof(int),nameof(AccidentYear))]
8     [Range(1900, 2100, ErrorMessage = "Value for {0} must be between {1} and {2}.")]
9     public int AccidentYear { get; init; }
10
11     [Required]
12     [NotVisible]
13     [IdentityProperty]
14     [Dimension(typeof(Scenario))]
15     public string Scenario { get; init; }
16
17     [Required]
18     [NotVisible]
19     [IdentityProperty]
20     [Dimension(typeof(AmountType))]
21     public string AmountType { get; init; }
22
23     [Required]
24     [NotVisible]
25     [IdentityProperty]
26     [Dimension(typeof(BusinessSegment))]
27     public string BusinessSegment { get; init; }
28
29     public double Value { get; init; }
30 }
```

**1** Possibility to document all the modelling choices

**2** Specifications are integrated with the code

In order to create different scenarios, we need to shock the actuarial planning data by an adjustment factor that is calculated starting from the imported Scenario Parameters. In particular, this factor obeys to the following logic:

$$\text{Adjustment Factor} = \begin{cases} \text{ORSA Delta Premium} + 1, & \text{if Scenario} \neq \text{Base Case} \\ 1, & \text{otherwise} \end{cases}$$

where ORSA Delta Premium is the weighted Market Premium scenario parameter.

**4.1 Adjustment Factor**

```
1 public interface IAdjustmentFactor : IScope<CashflowIdentity, ReportStorage>, IDataCube<IAdjustmentFactor>
2 {
3     static ApplicabilityBuilder App(ApplicabilityBuilder builder) => builder.ForScope<IAdjustmentFactor>(s => s
4         .WithApplicability<IAdjustmentFactor>(x => x.Identity.Scenario != BaseCase));
5
6     double Value => 1;
7 }
8
9 public interface IAdjustmentFactorScenario : IAdjustmentFactor
10 {
11     double IAdjustmentFactor.Value => GetScope<IOrsaDeltaPremium>((Identity)).Value + 1;
12 }
```

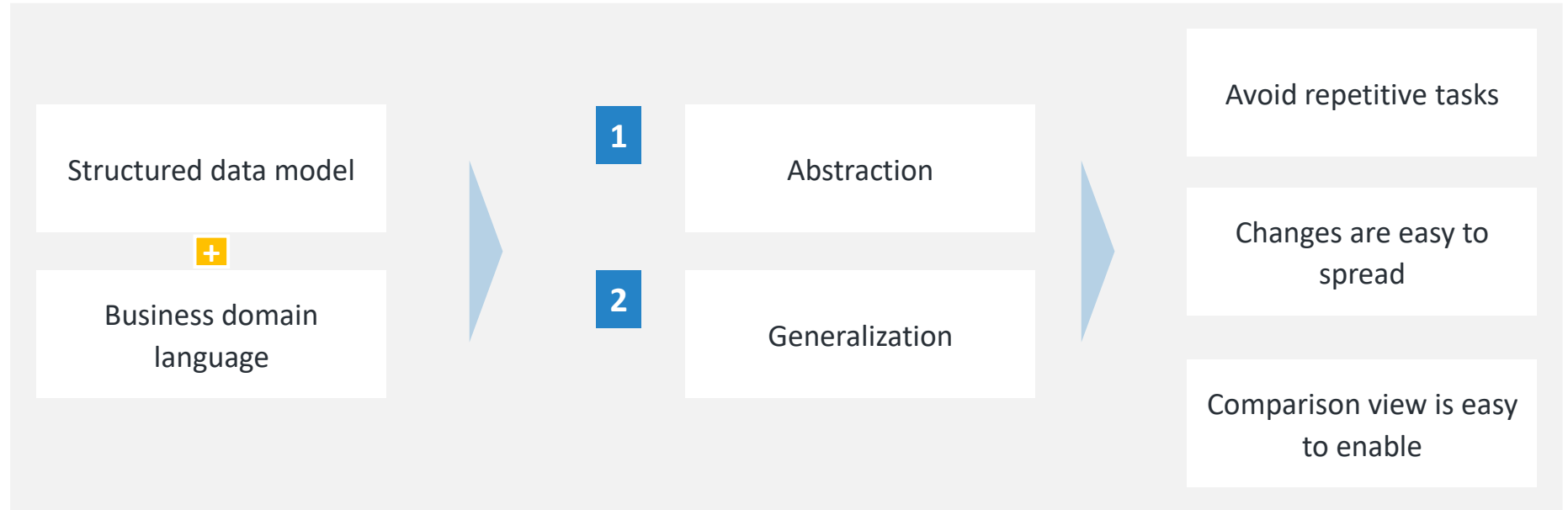
**4.2 Adjusted Nominal**

```
1 public interface IAdjustedNominal : IScope<CashflowIdentity, ReportStorage>, IDataCube<IAdjustedNominal>
2 {
3     double Value => GetScope<INominalBaseAPE>(Identity).Value * GetScope<IAdjustmentFactor>(Identity).Value;
4 }
```

# BUSINESS RULES: RECAP



## Two concrete examples



### Add new Scenario

1. Define the new Scenario
2. Import the corresponding parameters
3. Import eventual transactional data

### Enable comparison between 2 years

1. Evaluate calculations for the two years
2. Visualize the two years next to each other by simply slicing the data cube by year

---

# GOVERNANCE

# GOVERNANCE MADE WITH EASE



## Definition and Goal



Once data is properly structured, we need to ensure proper Governance around it.



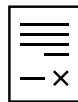
Clear Ownership



Clear Access Rights



Clear Business Processes



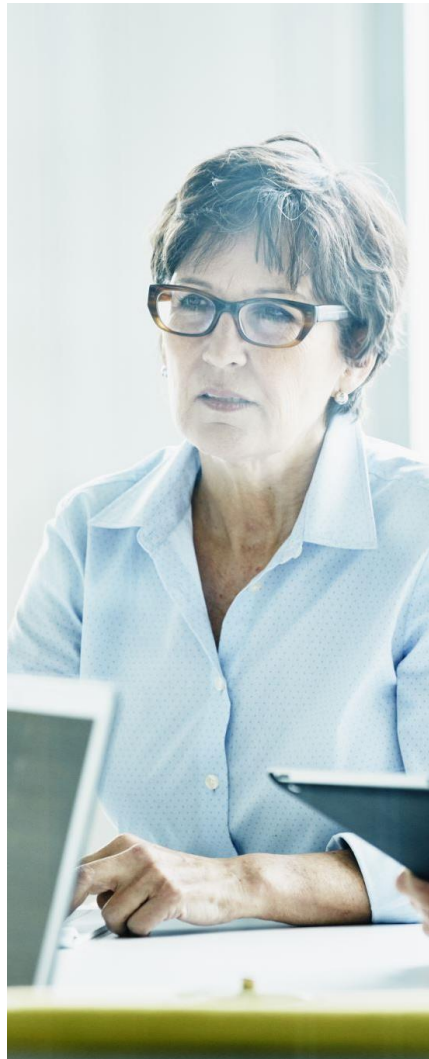
Clear Process Documentation

All of this in an easy to use and maintain state

# GOVERNANCE: BUSINESS PROCESSES

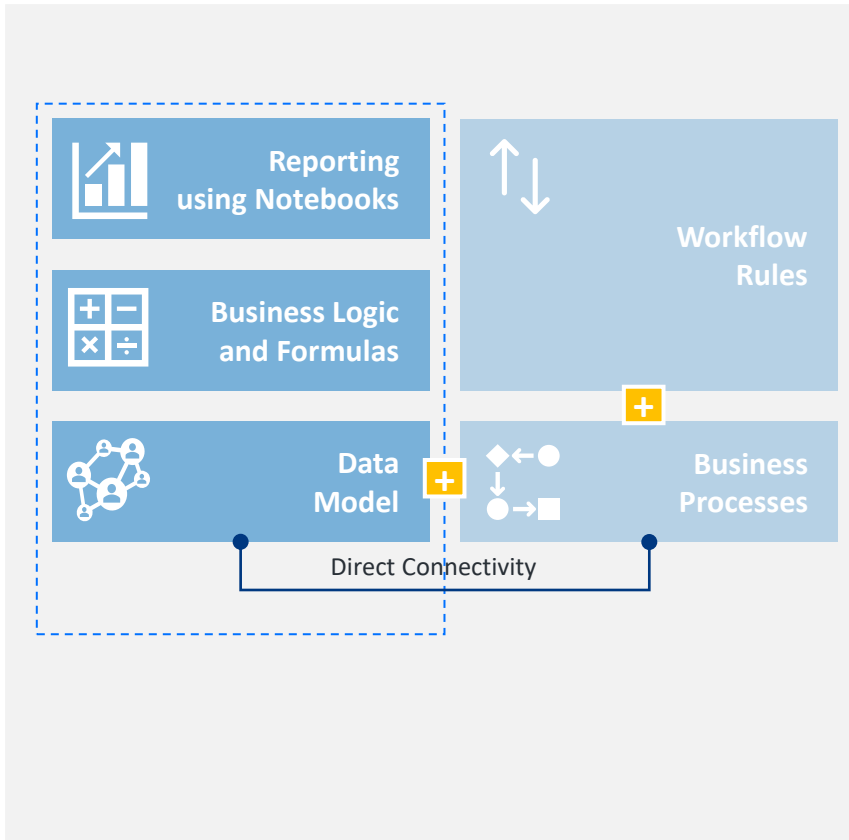


## Connecting Business Processes to the Data Model



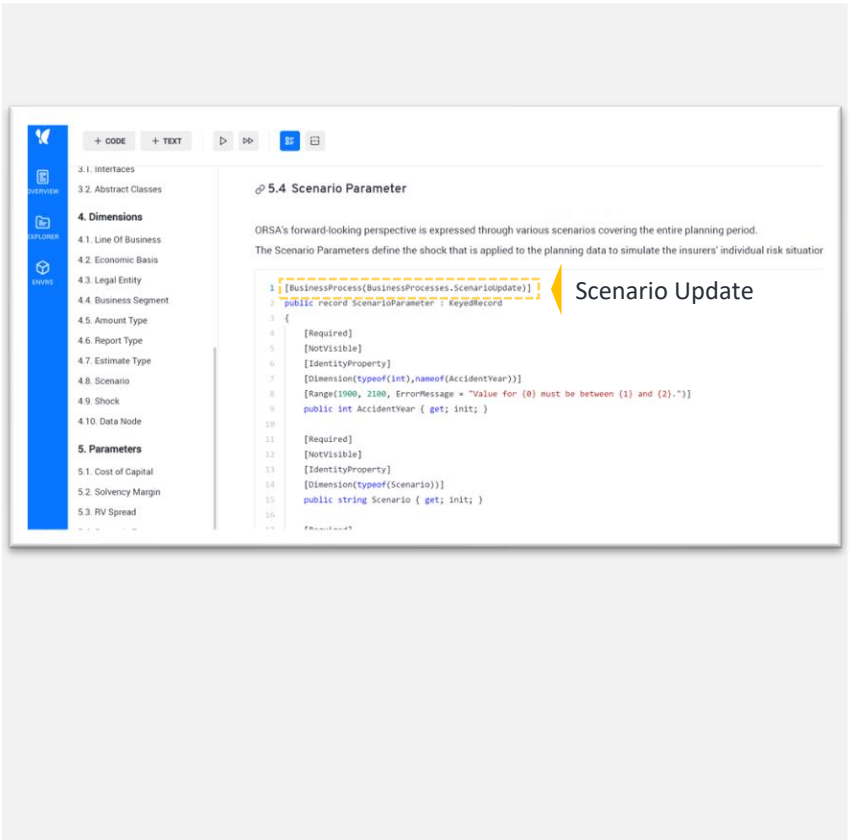
Sept 29, 2023

## Connecting Business Processes to the Data Model



Efficient data processes - ORSA as an example

## Goals of Adding the Business Processes



15

# GOVERNANCE: BUSINESS PROCESSES



Building blocks necessary to define the workflow rules



A Business Process is a Contract that defines who can do what and by when

## Key Capability

## Description

### STATES

States define the status of the business process. For example, data submitted, data reviewed, data signed off, completed.

### ROLES

Users are grouped by role. A set of permissions is assigned to each role.

### TRANSITIONS

Transition is the process of promoting a business process from one state to another. During transition, validations are performed.

### VALIDATIONS

Validations are automatic rules that...

- support users in their workflow (promotion validations);
- improve data quality (business validations);
- ensure access control (access rights validations, import validations).

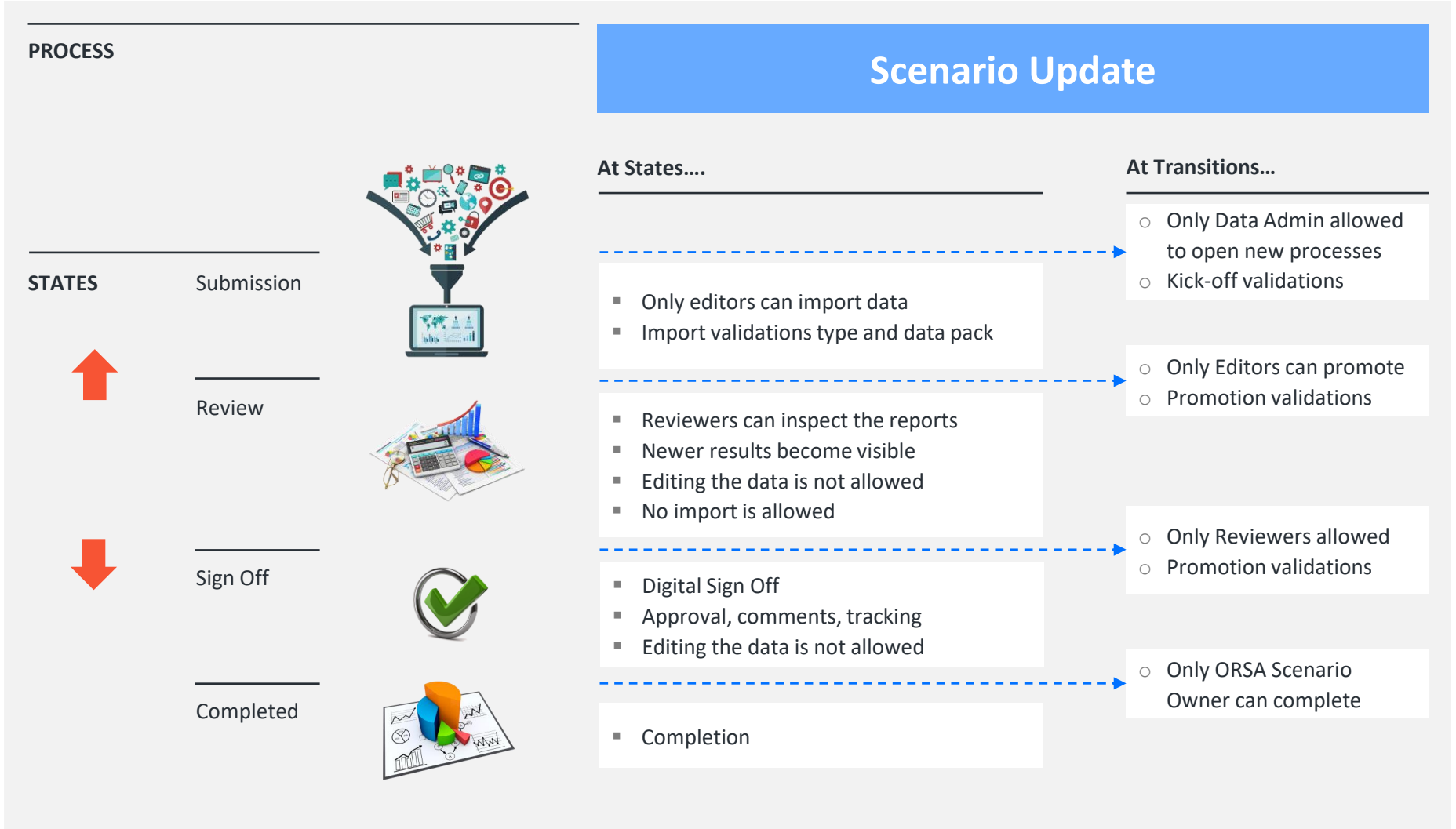
Most validations are applied when promoting from one state to another.



# GOVERNANCE: BUSINESS PROCESSES



The application to the ORSA Process shows the pragmatic and straight forward approach to digitalize Business Processes



# GOVERNANCE: BUSINESS PROCESSES



Define Business Processes with a language that is easy to read and to maintain



```
1 BusinessProcess.Create<ScenarioProcessGranularity>(BusinessProcesses.ScenarioUpdate, "Scenario Update")
2
3     .WithStates( states => states
4     .WithState(States.Submission, state => state
5         .WithTransitionTo(States.Review, transition => transition.WithValidation(c => UserCanKickOff(c))
6         .WithValidation(c => UserCanPromote(c)).WithDisplayName("Complete Submission")))
7     .WithState(States.Review, state => state
8         .WithTransitionTo(States.Sign, transition => transition.WithValidation(c => UserCanPromote(c)).WithDisplayName("Complete Review"))
9         .WithTransitionTo(States.Submission, transition => transition.WithValidation(c => UserCanPromote(c)).WithDisplayName("Back to Submission")))
10    .WithState(States.Sign, state => state
11        .WithTransitionTo(States.Review, transition => transition.WithValidation(c => UserCanPromote(c)).WithDisplayName("Back to Review"))
12        .WithTransitionTo(States.Completed, transition => transition.WithValidation(c => UserCanPromote(c)).WithDisplayName("Sign off")))
13    .WithState(States.Completed, state => state.ProcessEnd())
14    .WithDataSource(Workspace);
```

This code is:

- easy to read
- changes are easy to implement
- it is maintainable by a business team

# GOVERNANCE: APPLICATION EXAMPLE



Business Rules and Business Processes flow into an application for the end users



**BUSINESS PROCESS OVERVIEW**  
**MCEV 2023 Q1** Kick-off: 01/04/2023 Deadline: 21/04/2023

**OPEN PROCESSES** ^

- MCEV 2023 Q1
- SST CAPITAL 2022 Q4
- PRE-CLOSE 2022 Q4
- MCEV 2022 Q4

**CLOSED PROCESSES** v

**KICK-OFF**

**REPORTS**

**IMPORT**

**EXPORT**

**BUSINESS PROCESSES**

**EMEA**

- Switzerland
  - SUB 3/12 REV 4/12 SGN 3/12 CPL 2/12 PROMOTE ↻ ⋮
- Europe
  - Switzerland
    - SUB 0/12 REV 1/12 SGN 1/12 CPL 1/12 PROMOTE ↻ ⋮
  - Germany
    - Europe
      - Switzerland
        - SUB 3/12 REV 2/12 SGN 1/12 CPL 1/12 PROMOTE ↻ ⋮
      - Germany
        - Switzerland
          - SUB 2/12 REV 1/12 SGN 0/12 CPL 1/12 PROMOTE ↻ ⋮
        - Italy
          - Germany
            - Switzerland
              - SUB 1/12 REV 1/12 SGN 1/12 CPL 1/12 PROMOTE ↻ ⋮
            - United Kingdom
              - Italy
                - Switzerland
                  - REV SST AFR PROMOTE ↻ ⋮
                - CPL RBC PROMOTE ↻ ⋮
              - SGN AFR PROMOTE ^ ⋮

**Sign off**

**Back to Review**

**COLLAPSE**


---

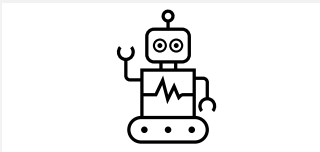
# CONCLUSION

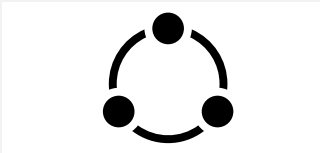
# TAKE AWAYS



- 1 Highly skilled people are freed up from non-value activities


- 2 Moving to a digitalized solution enables end-to-end automation


- 3 Introducing a governance layer ensures data quality and compliance through a customized review process



---

## YOUR CONTACT



Systemorph AG  
Bellerivestrasse 5  
8008 Zürich



Sara Busato  
Senior Analytical Engineer



[sbusato@systemorph.com](mailto:sbusato@systemorph.com)



[www.systemorph.com](http://www.systemorph.com)