Identifying business functions, cases and registers

Introduction

When designing and developing solutions to solve our customers' problems, discussions might arise with regards to the solution design as well as the design approach. This main characteristic of solution design - and in fact design in general - has two flavors: Bottom-up and top-down. As discussed in Design considerations of the Persistency Management guide, Blueriq allows both a bottom up and a top down design approach. In dynamic case management solutions, a top down approach is the best fit.

In this article, the top-down solution design approach using **Business functions** of the Business function model is discussed. Some questions that arise often are discussed and the given answers should help the designer in making sound and well informed choices, in order to design a solution that fits well in the customer's operations and application landscape.

In this design guide

Intermezzo: implementation method

In order to successfully design and implement solutions with Blueriq, projects follow the Blueriq implementation method very closely. This method is based on years of experience in many projects. In general, the method consists of four iterative phases

- Discover
- Design
- Model
- Use

and two continuous phases

- Continuous control
- Continuous improvement

Throughout the entire implementation of a solution, from discover to use, the customer (and the customer's customer) are at the center.

Each phase has a specific goal, a list of mandatory and optional artifacts and a list of milestones. For more information on the Blueriq implementation method (Dutch only for now), see Blueriq Implementatiemethode

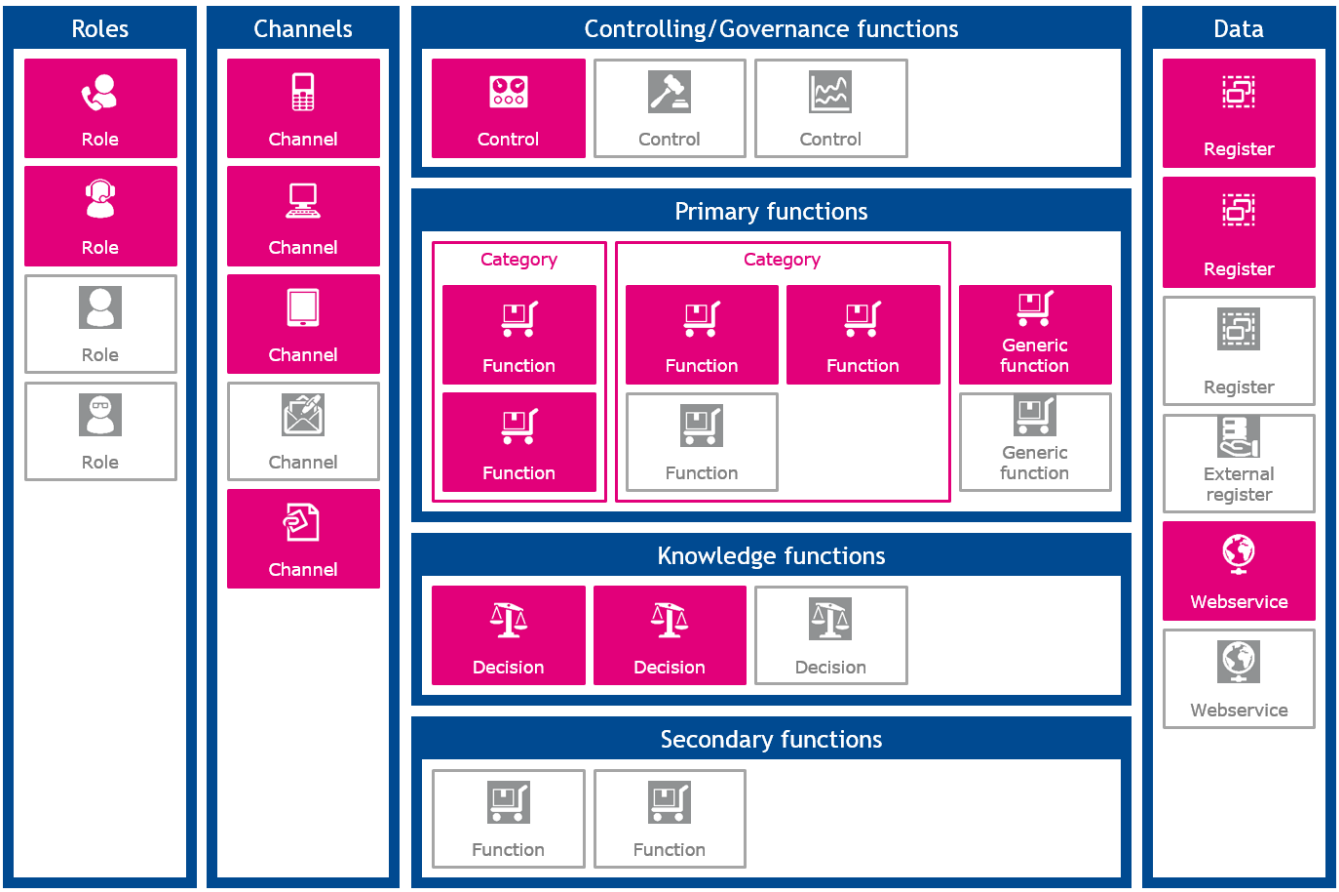
In this article, the focus is on Business functions of the Business function model of the Design phase.

Business function model

Within the design phase of the Blueriq implementation method, one of the most important artifacts is the **Business function model** (in Dutch: bedrijfsfunctiemodel). This model is tailor made for Blueriq's knowledge driven Dynamic Case Management solution implementation.

The Business function model is a description of the operations or functions of a business. It describes *what* an organization does, not how. Since the operations of a business are not as volatile as organizational structures such as departments and divisions, a business function model is a stable reflection of a business' operations and should not change when the organizational structure changes.

Shown below is an abstract version of such a Business function model.



The Business function model allows to scope. In the model above all the business functions are depicted, but not all these functions are in fact in scope of the solution design. Scoping is done using different colors, elements in **pink** are in scope of the solution, elements in **grey** are business functions, but they are not in scope of the design.

The Business function model contains 7 groups:

- **Roles**
  This part of the Business function model contains the roles that play a part in the business' operations, such as the case handler, the citizen or customer, investigators etc.
- **Channels**
  This part consists of the channels and portals on which the business runs its operations, such as a computer, tablet or phone but also the channels that provide incoming and outgoing communication in general, such as mail.
- **Controlling/Governance functions**
  Here all types of controlling functions are depicted. They steer the business and control their policies. Also the links to law and legislation are part of these functions.
- **Primary functions**
  This part of the Business function model contains all primary functions of the business. These functions can also be grouped, for instance by product.
- **Knowledge functions**
  Main decisions can be identified separately from the functions they are used in. They are the main decisions of the business and have their own ownership and life cycle.
- **Secondary functions**
  Next to the primary functions, functions can be distinguished that play an essential role in the business' operations, but are not part of the primary enterprise process.
- **Data**
  The primary and secondary functions section contain data themselves, but there is also a lot of data that is stored outside of these functions. For instance, registers that are part of the solution or registers that exist outside the solution.

Business functions

All business functions together describe the operations of a business. To execute these operations, Dynamic Case Management can be used. Cases and business functions share a commonality: they both have a goal and a result. Because of this fact, many businesses use Dynamic Case Management, Blueriq can be used perfectly for this.

As mentioned before, business functions describe *what* a business' operations are, not *how* these operations are performed. A way to perform an operation is to use one or more cases. This means that the primary and secondary functions can be regarded as cases, that is if the focus of the business solutions is on Dynamic Case Management. When looking at a business function model with a BPMN point-of-view, functions can be regarded as processes. If the focus is tailor made software, business functions can be regarded as separate applications.

It is a common misconception that each and every business function is a case and should be modeled as such. This - as mentioned before - depends primarily on your point of view. But still, even if the point of view is Dynamic Case Management, not all business functions are cases as some business functions can be implemented using multiple cases.

Cases

As discussed in Designing cases using aggregates, the definition of a case is hard to give. Many times, the question arises

> **What exactly is a case?**

Instead of a definition, the following list of characteristics of a case is given:

- Each case is an independent unit, with its own goal and its own result.
- Each case is responsible for its own behavior.
- A case can trigger another case.
- A case can publish its result to another case.
- A case can publish part of its domain to another case.
- A case reasons/infers about its own domain.
- A case can provide navigation to related cases.
- A case does **not** perform transactions in other cases.
- A case has its own life cycle.
- A case has a typical ownership during design time.
- A case has a typical ownership during its life cycle (single or collaboration in a group, can change over time)
- A case reflects a collection of work related to the daily operations of an organization, such as “inspection”, “permit application” or “child supervision” etc.

Now that it is clear what a case is and what it can do, the problem is still to identify the primary and secondary functions of a business and design them as one or more cases, or not. The second question therefore is:

> **What are typical cases?**

Let's review some examples.

- An organization that is in charge of providing benefits to citizens has a business function to handle requests for such benefits. Handing out a benefit will lead to consequences that also need to be processed. The benefit needs to be paid for instance. If the citizen needs to have a medical assessment to determine whether he is eligible for the benefit, this assessment might have to be repeated regularly. The citizen might object to the amount of the benefit and this objection needs to be processed, etc. All aforementioned operations could be regarded as cases. This depends also on the organizational view, some business functions are primary and others secondary. Candidate cases are benefi t, customer, request, objection, payment, assessment, etc.
- An organization that offers pension products to its customers has a primary business function to process the requests for these pensions. Giving out a pension product to a customer will lead to consequences that also need to be handled, such as premiums that need to be collected. Pension funds invest in shares on the stock market, this is another primary business function. Another case could be when a customer divorces, and his capital needs to be split up and one part needs to be allocated to his ex partner.
- An organization that offers insurances to its customers has a business function to process the requests for these insurances. Providing an insurance to a customer will result in business functions of premiums that need to be collected, or claims have to be processed, etc.
- An organization that is in charge of handing out permits to citizens has a business function to process the requests for these permits. Handing out a permit will lead to consequences that also need to be processed, so other business functions are necessary. For instance, the fee for the permit needs to be collected. Or if the permit was handed out under special conditions, it needs to be checked whether these conditions are met. Some permits need to be extended periodically, etc.

Registers

When identifying primary or secondary business functions a third question may arise. This question is:

> **What is a register?**

Cases typically contain processes of which the character is *decision making*. Registers very often do not have processes and if they do, these processes are all about *maintaining data consistency*. Just like was done with the concept case, a list of characteristics of a register is given, in stead of formulating a definition that is prone to discussion.

- A register accommodates or stores administrative data.
- A register stores this data in one single place.
- A register shares administrative data within a business' operations.
- A register is responsible for its own data.
- Only a register is allowed to change its data.

- A register can contain a process, for example validation for the sake of data consistency.
- A register can be external.
- A register always contains at least the current truth about its data.
- A register can contain historic data.
- A register can contain multiple realities.

The fourth question in this article is all about choice:

When should a case (with one or more processes) be used and when a register (with flows)?

The decision whether to model a business function as a case or as a register is straightforward most of the times, but sometimes it is not.

The first few business functions that will be identified when creating a business function model, are likely to be cases. These functions will typically constitute the primary process (requesting a product, processing a claim, etc) and are more decision oriented than data oriented. Usually, when identifying the more secondary functions (for example paying a benefit, repeating a medical assessment), the choice between expressing it as a case or as a register becomes a factor. In order to make this choice, the following two questions can help:

- Is the most important goal of the business function making a decision and communicating about it, or is the most important goal of the function to make sure data is consistent?
- Is this business function repeated periodically as a result of a previously made decision?

If the answer to the first question is that the most important goal is making a decision, identifying the business function as a case is likely to be a good choice. If the answer is making sure data is consistent, a register is a good choice. If the answer to the second question is yes, then a case is likely to be a good choice. Beware that this leaves room for personal or organizational preferences. In a lot of non straightforward situations, there is no right or wrong and the choice for a case or a register is based on discussions with the organization and the opinion of the solution architect.

Consider the next examples: Requesting a benefit and Creating a customer. The first example is very decision oriented and it is therefore highly likely to be expressed as a case. The second is very data oriented and it is therefore highly likely to be expressed as a flow for a register.

Other examples are less straightforward. When a citizen has requested a benefit for example, because he is unable to work due to sickness, the citizen has to have a medical assessment to determine whether he is eligible for this benefit. When the benefit has been granted, the medical assessment has to be repeated regularly to determine if the benefit can be continued. The choice whether the periodical assessment is to be modeled as a case or register is not that straightforward in this example. Both are likely choices, where the importance of the assessment plays a role. If the business merely asks how the citizen is feeling and stores this information, a flow for a register is a good choice. If the medical assessment is deemed to be very important however, expressing this as a related case is better.

When someone moves within the same municipality, this only involves changing his or her address. Such a simple task is likely to be expressed as a flow for a register. If someone moves to another municipality however, the situation is more complex. The amount of his taxes needs to be reassessed, permits need to be withdrawn or reconsidered, etc. It is highly likely that cases are triggered by such a move.

See also

More information on how to design dynamic processes in Blueriq can be found here: Designing dynamic processes.
How aggregates are used to model cases can be found here: Designing cases using aggregates.
Everything about persistency management and aggregates can be found here: Persistency Management guide.
A visualization of most concepts discussed in this article can be found here: Blueriq visuals.