Introduction to Knowledge-Intensive Processes and Fragment-Based Case Management

This document provides you a brief introduction to knowledge-intensive processes and how you can model these business processes using the fragment-based Case Management approach. Please read the introduction carefully.

Knowledge-Intensive Processes

Organizations perform *business processes*, which are sequences of value-adding activities aimed at satisfying customers and achieving strategic goals [2]. The individuals who carry out these processes are known as process participants.

Business processes can be categorized based on their flexibility needs. *Structured business processes* need little flexibility and follow predefined sequences of activities without requiring runtime adjustments. In contrast, *knowledge-intensive processes* (KiPs), rely on the process participants' knowledge, wherefore they can neither be fully predicted at design nor at runtime.

Fragment-based Case Management

Fragment-based Case Management (fCM) represents a suitable candidate for modelling KiPs. The following chapter explains the mechanics of fCM and illustrates them based on a sample selling process in a retail store.

"In general, an fCM model consists of four components: (i) process fragments describing the activities in the process and their ordering and data constraints, (ii) a domain model defining the involved data classes and their relations, (iii) object behaviour that specifies the allowed behaviour for all data instances, and (iv) a termination condition stating when the case may be terminated." [3]

Let us start with understanding **process fragments** by recapitulating how you can model a structured process as a BPMN diagram. In BPMN, you state out the whole course of a process. Figure 1 shows a BPMN process diagram of an exemplary selling process at a retail store. As soon as a customer enters the store, a process instance is triggered as the start event "Customer entered the store" is caught. Afterwards, the control flow enables the activity "Welcome customer" wherefore, the process proceeds step by step along the control flow. After the activity "Consult customer", a gateway allows for two different continuations: Either the customer decides for purchasing the product or not. Please note, exclusively one continuation is chosen per process instance! Furthermore, activities depend on and produce data objects. Thereby, a data object has a fixed type as well as a state which can change during the process. For instance, the activity "Sell product" can only be executed if a data object of type

“Consultation protocol” in the state “created”, we write “Consultation protocol[created]” in short, exists, and it produces the data object “Invoice [created]”.

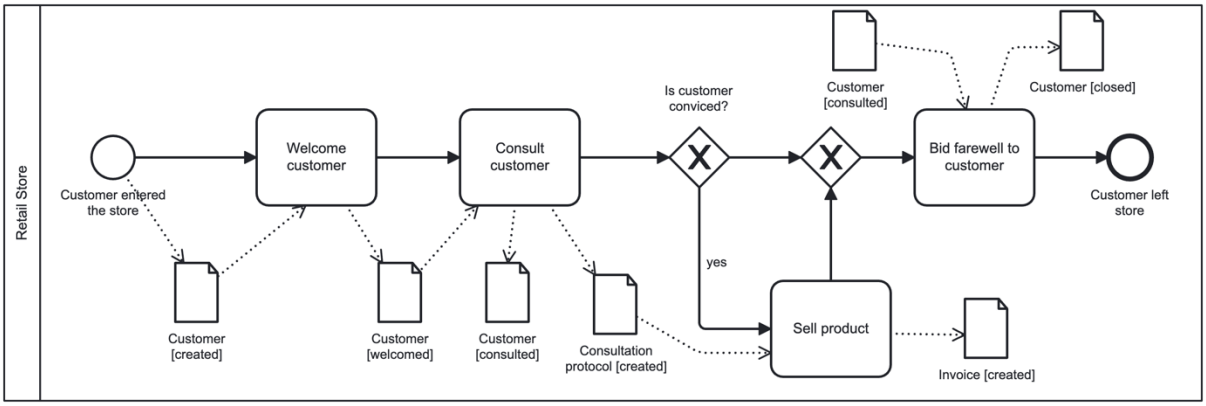

Figure 1: A sample selling process at a retail store, modelled as a BPMN diagram

You can consider process fragments as excerpts or partitions of a BPMN diagram. Figure 2 illustrates how you could split up the sample selling process into four process fragments. Talking about process fragments (PFs) requires considering the following aspects about them:

- Although, BPMN provides many more model elements, like parallel gateways, PFs in fCM rely solely on start events, activities, data objects, XOR gateways and the control flow.
- PFs with a start event at the beginning serve for instantiating a process instance. For instance, F1 catches the event “Customer entered the store” and, thereby, initiates a process instance.
- At runtime, process participants select enabled PFs which they execute. Thereby, they orchestrate the actual course of the process.
- A PF is enabled if the data dependencies of its first activity are fulfilled and if the execution of the PF does not violate any constraint which is represented in the domain model or in the object lifecycle mode. For instance, F3’s first activity “Sell product” requires the existence of the data object “Consultation protocol” in the state “created” to be enabled.
- As soon as the execution of a PF has started, it is executed as BPMN process model.
- The execution of a PF can disable enabled PFs. For instance, after the execution of F1, the data object “Customer[welcomed]” lead to the enablement of F2 and F4. The execution of F4 transfers the state of the “Customer” data object to “closed”. Thus, F2’s data requirement for a “Customer” data object in state “welcomed” is not fulfilled no longer and F2 gets disabled.
- PFs never contain loops, but they can be executed several times.
- PFs can be executed concurrently.

In general, PFs provide flexibility. As the activity “Bid farewell to customer” depends either on the data object “Customer [welcomed]” or on the data object “Customer [consulted]”, the PFs provide now the flexibility to bid farewell to the customer directly after the welcome.

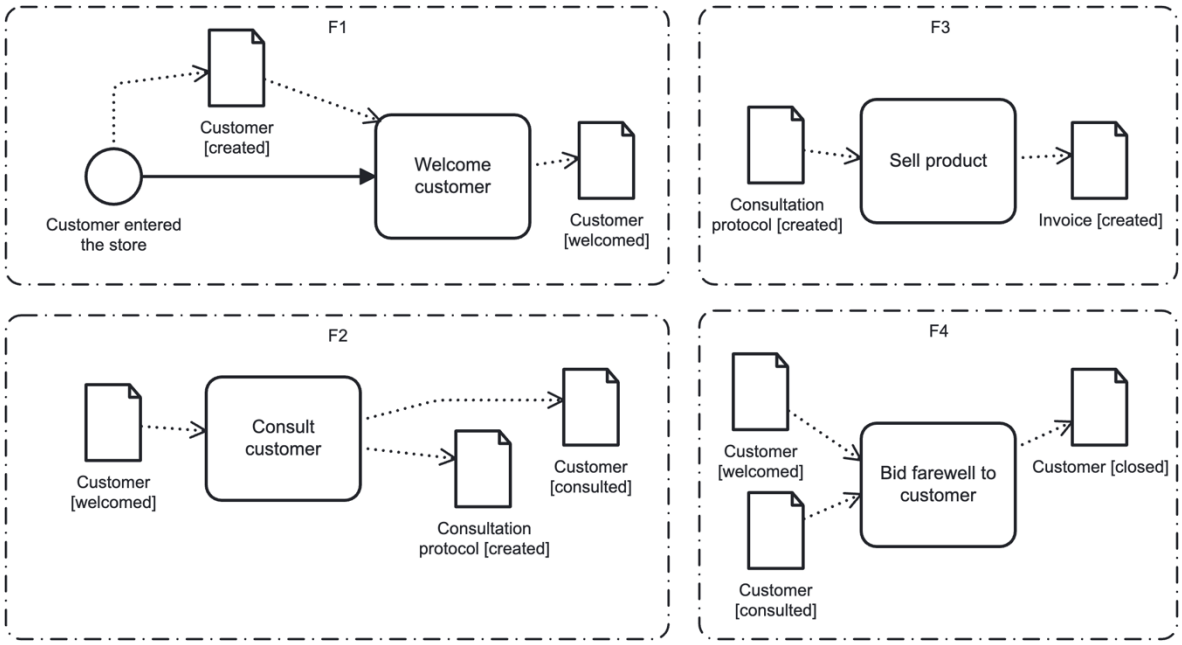

Figure 2: Partitioned selling process into four process fragments

Now, it is time to illustrate the **domain model**. In fact, it is an UML class diagram which contains a data class for each data object type, like “Consultation protocol”, which occurs in a process fragment. The domain model incorporates cardinalities to limit the number of instances of a data class within a process instance. Figure 3 presents an example for the domain model for the sample selling process. As at most one consultation protocol belongs to a customer, you could not execute F2 twice. However, the domain model also allows the absence of the “Consultation protocol” data object, you could also execute F4 directly after F1.

Figure 3: Domain model for the sample selling process

For each class in the domain model, an **object lifecycle diagram** exists and shows the allowed states and state transitions of related data objects. In process fragments, the notation “Customer[consulted]” refers to a data object of type “Customer” in the state “consulted”. Figure 4 shows an example of such an object lifecycle diagram for the “Customer” class. As there is no state transition from the state “closed” to the state “consulted”, it inhibits the consultation of a customer after the seller bid farewell to the customer.

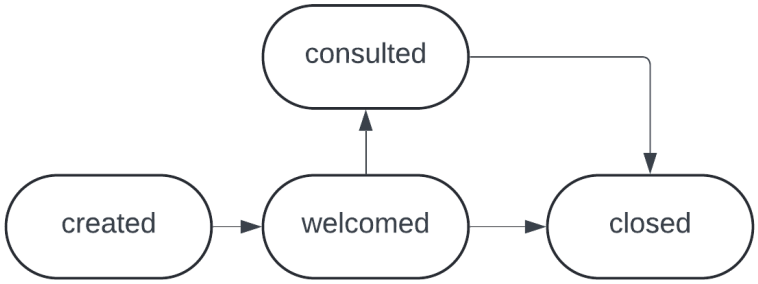

Figure 4: Object lifecycle diagram of the class "Customer" within the selling process

Finally, the **termination condition** is a logical expression based on the states of the data objects. As soon as the termination condition holds, the process participants can terminate the process instance. For instance, the termination condition for the sample selling process could look as follows: Customer[closed].

After reading this brief introduction to the mechanics of fCM, you should continue with answering the questionnaire to assess your fCM skill level.

References

1. Di Ciccio, C. et al.: Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches. J. Data Semant. 4, 1, 29–57 (2015).
2. Schmelzer, H.J., Sesselmann, W.: Geschäftsprozessmanagement in der Praxis: Kunden zufriedenstellen, Produktivität steigern, Wert erhöhen. Hanser, München (2010).
3. Seidel, A. et al.: Extending Case Models to Capture Organizational Aspects and Time. In: Companion Proceedings of the 42nd International Conference on Conceptual Modeling. (2023).