The Case Management Modeling and Notation (CMMN) version 1.0

Abstract

This is a short tutorial of the Case Management Modeling and Notation (CMMN) version 1.0. It is targeted to readers with knowledge of basic process or workflow modeling. It covers the complete CMMN notation. A simple *complaints process* is used to demonstrate the notation.

Short introduction to Case Management

Case management is a type of business process technology that does not uses control flow to describe the process. The case (case file, or case folder) is the main concept, and it contains all the data and information about the process. Case management is about empowering workers by providing them with access to all the information concerning the case and giving them discretion and control on how a case evolves. Case management it is not about the process, it is about the workers.

In a traditional workflow or process system the designer encode the business goal to be accomplished in the model. Thereafter the system is responsible for the business goal and it uses the workers to achieve that goal. In a case management system, in the other hand, the workers are responsible for the business goal and they use the system as a tool to accomplish that goal. That it is why case management relies more in the worker's judgment than in control flow.

For this tutorial we will use the Case Management Modeling and Notation(CMMN) version 1.0 to model a fictitious *complaints process*. The fictitious *complaints process* is a common process in customer service departments. We will assume this company sells very specialized and expensive products. Due to the nature of the products the company also provide services (like installation and configuration) for those products. The company customer service department has a group of highly skilled workers that deal with complaints. The goal of modeling the *complaints process* is to standardize it to provide guidelines and support to the customer service workers in charge of that process. In addition, The case management *complaints process* will allow the company to improve customer service and better track service level agreements (SLA) in the complaints process.

CMMN

The Case Management Modeling and Notation(CMMN) version 1.0 was created by the Object Management Group (OMG) and published in 2014. It is a complementary notation to the Business Process Management and Notation (BPMN) which focus on control flow to describe business processes. We will say that CMMN is declarative in which you describe 'what' is allow and disallow in the process; versus BPMN that is imperative in which you describe 'how' to do the process. BPMN, CMMN, and the Decision Model and Notation (DMN) are the three OMG business modeling notations.

The case is the main concept in CMMN, and it is similar to a process. A case contains a case file (i.e. case data container) and it is described by a case plan (i.e. a model or diagram). We start by introducing the case plan for our *complaints process*.

📑 Case Plan

In CMMN, a model (i.e. diagram) may have multiple cases, and each case is described by a `case plan`. For our *complaints process* we will use a single case, so we will have a single `case plan`. The complete description of the case being modeled is described inside its `case plan`. However, because of the nature of case management not all the work that happens in a case is modeled. In particular the interaction of case workers with the case and case data may not be modeled, or be just partially modeled. In most situations, we don't model how the data gets into the case. Data can be added, removed, modified by the case workers at any time during the processing of a case without the need of modeling it.

We start our model by adding a `case plan` and we will name that case *complaint*.

During modeling, we design one or more `case plans`. Eventually, the `case plan` will be executed and it will be called a case instance. So, a `case plan` is similar to a class (i.e. type), and a case instance is an object of that class.
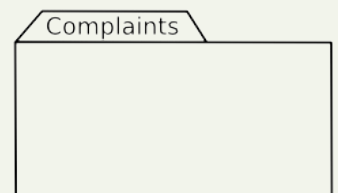
Although the correct term is `case plan`, and the word "case" is overloading (sometimes it refers to a case instance, or the data in the case, or to the process, etc.), for this tutorial we will use `case` to mean `case plan`.

Roles and Case plan items

Roles

In CMMN, roles are defined at the `case (case plan)` level. However, CMMN does not provides a graphical notation for roles, and again not everything a role does is modeled. In particular, a role may be allowed to do case planning (which we will discuss later) that may not be explicitly modeled. A role may also add, create, modify, or remove data and documents from the case file which may not be explicitly modeled either.

In the *complaints process*, we will have a supervisor, a product specialist, an investigator, *etc.* In most situations, we see a case owner role, which in our *complaints process* will be the case worker who is responsible for managing the complaint. This role decides which tasks must be performed or not in the case instance for this particular customer.

Case plan items

An executing model (a case instance) is implementing a plan, and so, its elements are called `case plan items`, because they are part of that plan. The `case plan items` are `tasks`, `stages`, `milestones`, and `event listeners` that we will cover as part of this tutorial.

Milestones

Milestones represent accomplishments during the execution of the case instance. Due to the large variations between case instances, **milestones** are important in understanding the progress of a particular case instance. For our *complaints process*, we will include few **milestones**,

:: *Received* to indicate each time we receive customer information.   (Received)

:: *Exceed SLA* for those situations in which the case exceed the predefined SLA.   (Exceed SLA)

:: *Completed* to indicate when a product specialist produces a product report.   (Completed)

:: *Fraud* to indicate when a fraud investigation has started.   (Fraud)

A `task` represents the execution of actual work. There are four types of `tasks`, namely `non-blocking human task`, `blocking human task`, `case task`, and `process task`. The type of `task` is indicated by an annotator in the upper left corner of the `task` icon. A `task` may have a manual activation decorator (as we will discuss later), which means that a case worker must decide if the task should be executed or not.

In our *complaints process*, we will add a `non-blocking human task` to hand over an assignment to a *product specialist.* We will use that `task` when executing a *product complaints*, which is one of the types of complaints in our *complaints process*. We will discuss the *service complaint* later.

It is the responsibility of the `product specialist` to provide a *report*, so we don't need to wait until he completes the `task`. `Non-blocking human tasks` are handed out to a case worker (therefore the little hand icon) and as soon as it is claimed by a case worker, it will be considered complete.

We will add a `case task` to revert a customer payment. This `task` will create another case to deal with reverting the payment.

We will add a `task` to inform the customer of the outcome of the case. This will be a `blocking human task`. `Blocking human tasks` are executed by a case worker and they must be explicitly completed by the worker.

Later we will also add a `process task`. A `process task` is modeled and executed using imperative process technology like BPMN. This allow organizations to reuse processes.

In our fictitious company, there are at least two type of complaints, one for products and one for services. Therefore, our *complaints process* will need to deal with the two type of complaints, namely *product complaints* and *service complaints*. We have two options to model the differences. We could create two different cases, one for each type of complaint, or alternatively we can use `stages`. `Stages` are containers similar to sub-processes in other workflow or process notations. They are used to manage the complexity of the model by decomposing it into manageable sets.

For our *complaints process*, we will be sharing most of the data for both types of complaints. In addition, some situations may involve both service and product complaints. Therefore, `stages` make more sense than creating completely separate cases. So, we will add two `stages` one for product complaints and another for service complaints.
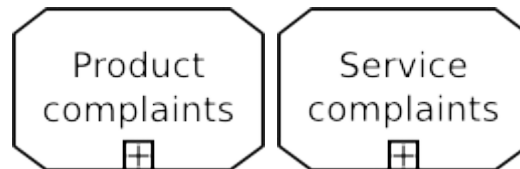
📋 Case file

In CMMN, each case instance contains a single `case file` (also called a case folder, or just the case), and case workers have access to all the data in that `case file`. Case workers can add, remove, modify data in the `case file` even if they are not executing any `task` in the case, as long as they have sufficient privileges. The data in the `case file` is called `case file items`.

Case file items

All data and data structures are called `case file items`. All the `case file items` are stored in the `case file`. `Case file items` are used to represent all kind of data, including a data value in a database, a row in a database, a document, a spreadsheet, a picture, a video, a voice recording, *etc.* In addition to basic data, `case file items` can also represent containers, including, a directory, a folder, a set, a stack, a list, *etc.*

As with most case management applications, we will not model all the data required for the *complaints process*, but we will model the following data,

:: The *report*. As mentioned before, in some situations the *product specialist* will produce a *report* that will satisfy the *completed* `milestone`. In out fictitious company, the *report* is a document that follows a particular template.

report

:: The *input* folder may contain emails, documents, pictures, recording of customer calls, *etc.* All the data submitted by the customer will be collected in a folder called *input*.

input

:: The *resolution* document is produced as part of the case and indicates the outcome of the complaints investigation.

resolution

:: Finally, a customer can cancel the complaint by calling or sending a notification. That will be represented by the *cancel* `case file item`.

Cancel

Criteria

So far, we have created a set of artifacts in the model (i.e. `case plan`, `milestones`, `tasks`, `stages`, and `case file items`), but we have not modeled how they are related. CMMN is declarative, and it reacts to changes on the environment and case worker decisions. `Entry criteria` and `exit criteria` will allows us to model changes in the case environment.

📋 Criteria

Criterion allow us to describe when a `task`, `stage`, or `milestone` should be available for execution (`entry criteria`), or when a `case` (`case plan`), `stage`, or `task` should terminate abnormally (`exit criteria`). Criteria has the following two optional parts,

- One or more trigger events (called `onParts`). These are events that will satisfy the evaluation of the `entry criteria` or `exit criteria`. Events that emanates from other CMMN elements can be visualized by an optional `connector` (a dotted line). However, the visualization is optional and does not describes the type of event.
- A boolean expression (called `ifPart`). This expression must evaluate to true for the `entry criteria` or `exit criteria` to be satisfied.

We can think of the criteria forming a sentence as follows, **([on** <Event 1>[, **on** <Event 2>[, . . .]] **]) AND ([if** <Boolean condition> **])**

Where square brackets ([ ]) indicate optional parts of the sentence, and angled brackets (< >) are place holders to be replaced. Looking at the sentence, we can see why the event is called the `onPart`, and the boolean condition is called the `ifPart`. Note that both the `onPart` and the `ifPart` are optional in the sentence, but for it to make sense at least one of them must be present.
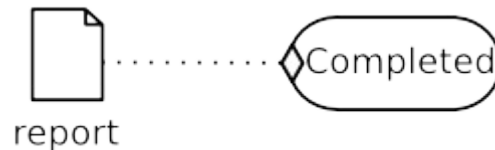
An **entry criteria** (◊) describes the condition that must be satisfy for the **stage**, **task**, or **milestone** to be available for execution. **Stage**, **task**, or **milestones** without **entry criteria** will be available for execution as soon as they are created.

In the *complaints process*, both **stages** *product complaints* and *service complaints* need an **entry criteria**, because they can only execute if the complaint is of their type. In most cases, only one of the two **stages** will execute, although in few situations the complaints may involve both **stages**.
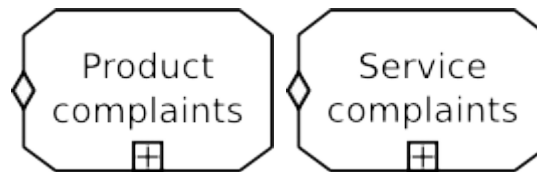
The **entry criteria** can be placed anywhere in the border of the **stage**, **task**, or **milestone**. For our example, we said that a *product specialist* may creates a *report* and that creation generates an event that satisfy the *completed* **milestone**. We visualize that as follows,

Note that without looking inside the **entry criteria** of the *completed* **milestone**, we cannot tell if it is satisfied by the *report* being created, updated, or even deleted. In addition, we don't know if there is a condition (**ifPart**). We only know there is an event (**onPart**) from *report* that is used to trigger the *completed* **milestone**.

We know the *report* `case file item` and the *complete* `milestone` are part of processing a *product complaints*. We also know the *product specialist* `non-blocking task` is also part of processing a *product complaints*. So, we can expand the *product complaints* `stage` to show that, as follows, Notice that *product specialist* `task` does not have an `entry criteria`, and so, as soon as *product complaints* start executing the *product specialist* `task` also start executing. Eventually, the case worker may create a *report* and that creation event satisfy the *completed* `milestone`.
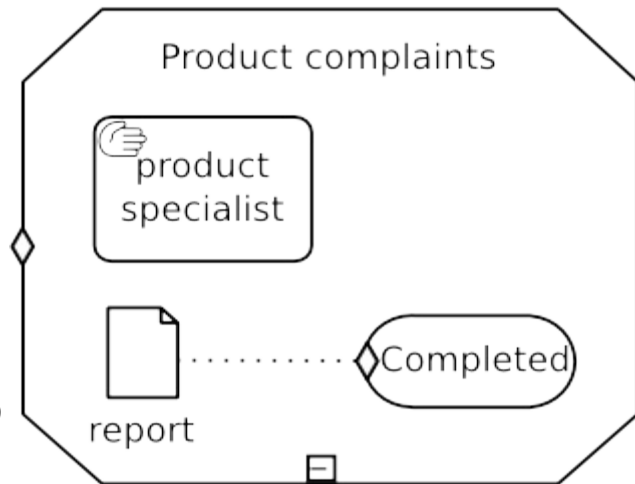
In the *complaints process*, the *send letter* `task` depends on two events, one from the *received* `milestone`, and one from the *resolution* `case file item`. We require *input* from the customer, and we have to complete the *resolution* of the case to send the letter to the customer. That means the `entry criteria` is waiting for two events (two `onParts`), and they forms an AND condition. Both events must happen for the `entry criteria` to be satisfied.
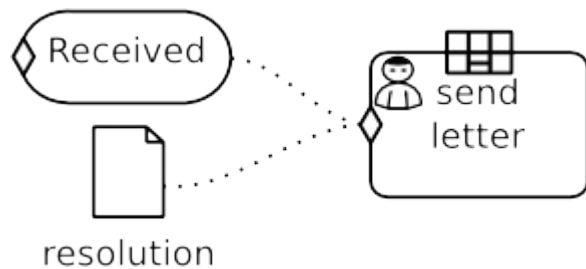
Later we will see an example of an OR conditions (hint, use multiple entry conditions).

Exit criteria

An **exit criteria** (♦) is similar to an **entry criteria**, but it is used to stop working on the **stage**, **task**, or **case** (**case plan**) when it is satisfied.

In the *complaints process*, we will add an **exit criteria** for the case. In the situation the customer calls and cancel the complaint, so we need to stop working on the case. We model this scenario by having a *cancel* **case file item**, which could be a voice recording of a customer call, a letter from the customer, or just a flag in the case data.
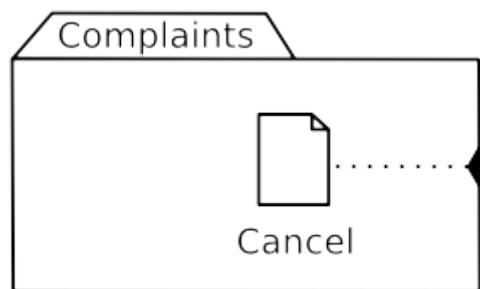
Connecting two criteria

> It is valid for an **onPart** to reference another criteria, instead of an event. When the other criteria is satisfied the **onPart** will also be satisfied. This is visualized by a connector (. . . . . .) between the two criteria. Connecting an **exit criteria** to an **entry criteria** in this way is the only flow control available in CMMN.

Events listeners are similar to events in other workflow or BPM notations. For the *complaints process*, we have a service level agreement (SLA) that we will model using a **timer event listener** and a **milestone**, as follows.
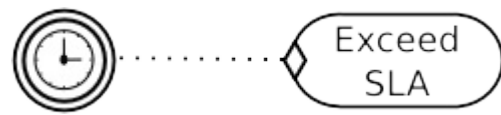
We will also have a **human event listener** to provide the supervisor with a way to trigger the *revert payment* **task**. There are multiple ways that we can use to give the supervisor the ability to trigger the *revert payment* **task**, but for illustration purposes we will use a **human event listener**.
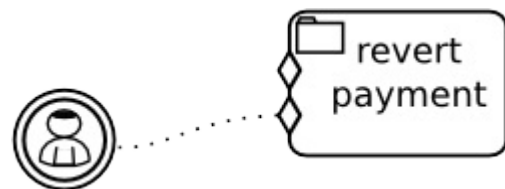
Note that *revert payment* **task** has two **entry criterion**, one is triggered by the **human event listener**, and the other is triggered by another **entry criteria**. When a **task**, **stage**, or **milestone** has more than one **entry criteria** they form an OR condition. Meaning any of the **entry criterion** that is satisfied will start the **task**, **stage**, or trigger the **milestone**.

Plan items standard events

As we described before, the **entry criteria** and **exit criteria** have an event (`onPart`) and a condition (`ifPart`). An executing model (a case instance) is implementing a plan, and so, **tasks**, **stages**, **milestones** and **event listeners** are called **case plan items**, because they are part of that plan. Every **case plan item** generates events that can be used in the `onPart` of an **entry criteria** or **exit criteria**.

The following table list the standard events for the different **case plan items**. There is no need to memorize this table of events, because modeling tools should have the list available when defining an **entry criteria** or **exit criteria**. The table contains a column indicating the events that result from a case worker action. As you can see a case worker (in a role with enough privileges) has a lot of discretion to control and modify the behavior of the **case plan items**.

cases	Tasks & Stages	Events & Milestones	Case Worker	Description
create	create	create		it was created
	start			it started execution
	enable / reenable		yes	it was enabled (it was disabled before)
	manualStart		yes	it was manually executed
	disable		yes	it was disabled (element was not executing)
suspend	suspend	suspend	yes	it was suspended (it was executing before)
	resume	resume	yes	it resumed execution (it was suspended before)
		occur		milestone or event did occur
	parentSuspend			suspended because parent stage or case suspended
	parentResume			resumed execution because parent stage or case resumed
reactivate	reactivate		yes	reactivated after a failure

| reactivate	reactivate		yes	occurred
complete	complete			normal completion
terminate	terminate	terminate	yes	manual termination by case worker or exit criteria
fault	fault			it entered a fault condition
close			yes	case is closed by case worker

Case file item standard events

Case file items (data) also generate events that can be used in an **entry criteria** or **exit criteria**. The following table list the standard events for the case file items. Again, there is no need to memorize this table of events, because modeling tools should have the list available when defining an entry or exit criteria. The table contains a column indicating the events that result from a case worker action. As you can see a case worker (in a role with enough privileges) has the ability to add, delete, and modify the data in the case.

Case file item	Case worker	Description
All type of data (for example, a row in a database, a document, a picture, a video, etc.) generate the following events		
create	Yes	Item was created
replace	Yes	Content of the item has been replaced
update	Yes	Item has been updated
delete	Yes	Item has been deleted
addReference	Yes	A new reference to the item has been added
removeReference	Yes	a reference to the item has been removed
Containers (for example, a directory, a folder, a set, a stack, a list, a database table, etc.) Generate (in addition to the previous events) the following extra events		
addChild	Yes	a new child has been added to the container
removeChild	Yes	a child has been removed from the container

We have defined most of the *complaints process* model. In particular, we have defined most of what will be in the **case (case plan)** when a case instance is created. Putting everything together, this is what we have modeled so far,
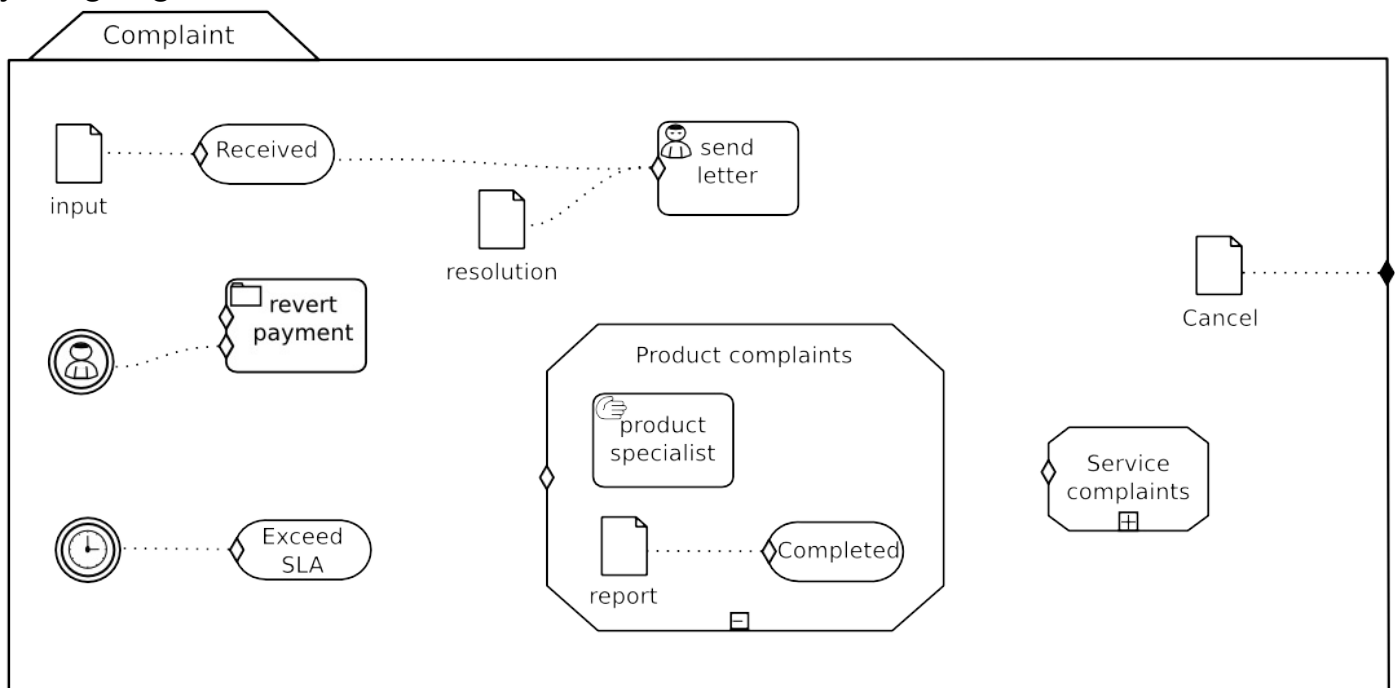

This model shows everything that will be in the execution plan, when a case instance is created. But, CMMN provides a way for case workers to add more elements to the plan. Remember that case workers can disable **case plan items**, but they can also add new item to the plan. CMMN has the concept of **discretionary items**, which are modeled but are not included in the execution plan of a case instance. The only way a **discretionary item** will be added to the plan of a case instance is when a case worker adds it to the plan. Adding **discretionary items** to the **case** is called planning. **Discretionary items** has the same shape than planned items, but using a dashed line, instead of the continuous line used by **case plan items**.

In some uncommon situations the case workers in our fictitious company may suspect they may be dealing with a fraudulent complaint. But,because this situation is not common there is no need to include it in the plan of every case instance. However, we want to give case workers the ability to add a *Fraud investigation* `stage` to the `case`, so we will add a `discretionary stage` as follows,

Note that *Fraud investigation* is a `discretionary stage`, and so it is drawn with a dashed line. Also notice that it contains two non-discretionary items (*Fraud* `milestone` and *Investigation* `task`). Both, the *Fraud* `milestone` and the *Investigation* `task` will start executing as soon as *Fraud investigation* executes, because they are non discretionary and they don't have any entry criteria.
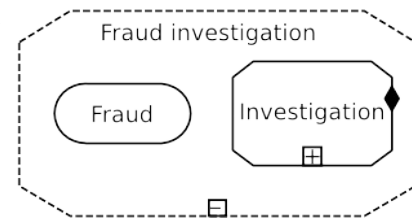
📑 Planned versus discretionary

The CMMN specification describes the distinction between the execution plan and planning, with the following diagram,

At design time, the user models both `plan items` and `discretionary items`. At execution time, a CMMN compliant engine only has in the execution plan the `plan items`. However, case workers can add to the plan the `discretionary items` they consider necessary for the particular case instance.
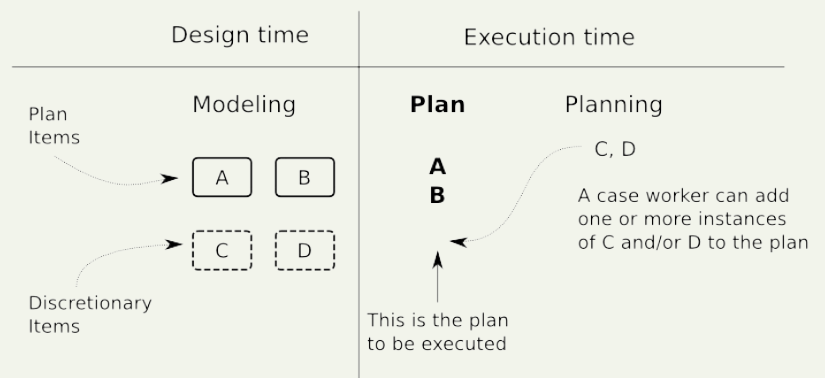
Planning table

`Planning tables` are used to indicate that planning is allowed in a `case` (`case plan`), `stage`, or `human task`. For planning to be allowed, there must be `discretionary stages`, `discretionary tasks`, or `plan fragments` in the scope of the `case`, `stage`, or `human task`. To indicate that a case worker can do planning a `planning table` is used. The `planning table` can be expanded or collapsed.

⊞ expanded ⊞collapsed In our *complaints process*, there are situations in which the case worker sending the letter to the customer may feel it is appropriate to provide the customer with a discount. We don't want to encourage the discount (and so we don't put it in the plan), but we want to give the case worker the ability to add a *process discount discretionary task* to the plan. So, we add a *process discount discretionary task*, and we place that `discretionary task` in the scope of the *send letter* `task` by adding a `planning table`, as follows
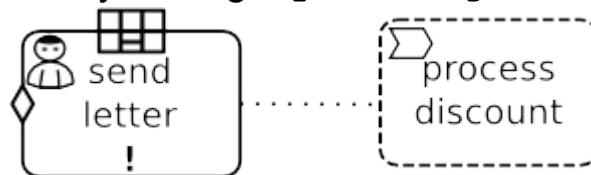
Notice that we draw an expanded `planning table` in the border of the *send letter task*. In addition, because the `planning table` is expanded, we show the *process discount* `discretionary task` connected to the send letter task by a `connector`. Note the `connector` in this situation is different than a `connector` in an `entry criteria` or `exit criteria`. In this situation, the dotted line is indicating that a case worker executing the *send letter* `task` can add a *process discount* `discretionary task` to the plan, and the *process discount* `discretionary task` will start executing as soon as it is added to the case instance plan.

We will also add a *check safety* `discretionary task` to our *product complaints* `stage`. This `discretionary task` will be used in rare situations when a case worker suspect of product safety issues. Therefore, we need to add a `planning table` to this `stage`. We also added other **discretionary items** for the *product specialist* to use, but because space considerations, we will not show them. These are indicated by the collapsed `planning table` in the *product specialist* `task`. So, our *product complaints* `stage` now looks as follows,
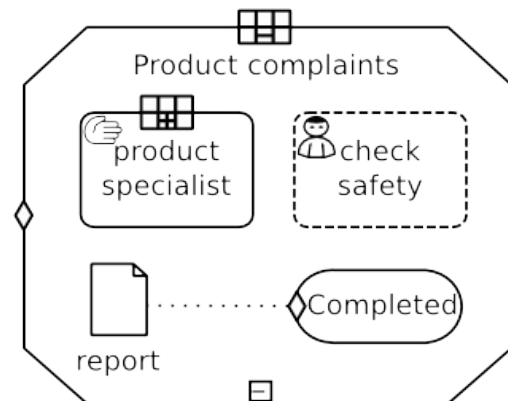
📋 Plan fragment

There are situations in which you want to give the case workers the ability to add a set of `discretionary items` as a single planning action. `Plan fragments` provide a way to do that. A `plan fragment` is just a grouping mechanism for `discretionary items`. In our example, *Audit* is a `plan fragment` that can be started by a manager with the ability to add `discretionary items` to the case plan.

Note that *Audit* is collapsed (⊞), and so, we don't see the `discretionary items` it contains. Both, *Audit* and *Fraud investigation* will be at the case level scope, so we need to add a `planning table` to the *complaints* `case` also.
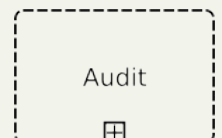
`Case plan items` and `discretionary items` can be annotated to indicate certain characteristics of the item. There are four decorators,

The `Auto complete` decorator (■) indicates that the `stage` or `case` (`case plan`) will complete when all the required `case plan items` are completed. If the decorator is not present, the `stage` or `case` requires manual completion by a case worker after all the required `case plan items` have completed.You will use this decorator when there is no `discretionary items` and you want the system to complete the `case` or `stage` as soon as all the `case plan items` have completed. However, in most situations you will want a case worker deciding if a `stage` or `case` should be completed. Because there may be `discretionary items` or other reasons to avoid prematurely closing the `case` or `stage`.
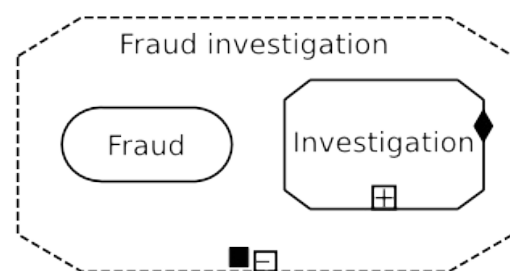
We will place an `auto complete` decorator in the *Fraud investigation* `discretionary stage`, because we want it to complete as soon as both the *Fraud* `milestone` and the *Investigation* `stage` complete.

The `Manual activation` decorator (▷) indicates that the `stage` or `task` must be manually initiated after the `entry criteria` has been satisfied. If the decorator is not present, the `stage` or `task` will automatically start executing when one of the `entry criteria` is satisfied. It is important to provide case workers veto power over the case management system. There are situations, in which an `entry criteria` is satisfied, but you want to give case workers the ability to decide if the `task` or `stage` really needs to be executed and when to start the execution. In our *complaints process*, we have a *revert payment* `task` that will be ready for execution if one of its two entry criterion is satisfy. However, we want to be sure that a case worker verify if *revert payment* should be executed.
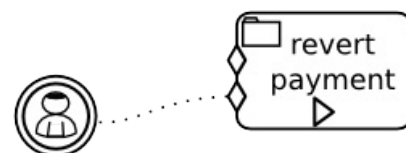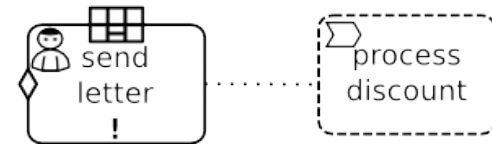
Required decorator (`!`) indicates that a **stage**, **task**, or **milestone** must be executed for the scope (**stage** or **case**) to complete. Note that every **case plan item** that start executing must also complete for the enclosing **case** or **stage** to complete. Therefore, as soon as a **case plan item** start execution it becomes required. **Discretionary items**, by definition are not part of the plan, so they cannot have a **required** decorator. However, when a **discretionary item** is added to the plan by a case worker and start execution it must complete execution for the enclosing **case** or **stage** to complete. In our *complaints process*, we will mark the *send letter* **task** as **required**, because we want to be sure we notify the customer of the outcome of the complaints.

Repetition decorator (`#`) indicates the **stage**, **task**, or **milestone** can be repeated multiple times. Only **stages**, **tasks**, or **milestones** with at least one **entry criteria** can have the **repetition** decorator. In the *complaints process* we will mark the *Received* **milestone** with the **repetition** decorator, because the customer may send multiple different documents or pieces of information that will be stored in the *input* **case file item**, and we want the *Received* **milestone** to occur each time we receive something from the customer.
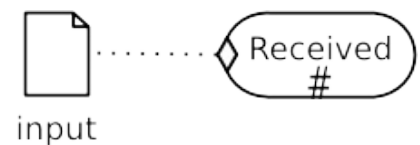

📑 Connectors

We have already saw the two situations in which **connectors** (. . . . . .) are used. First, they are used to visualize the event (`onPart`) of an **entry criteria** or **exit criteria**. In this situation the **connector** is optional. Second, **connectors** are used to visualize the **discretionary items** associated with a **human task**'s **planning table**, in which case the **connector** connects a **human task** with a **planning table** to the **discretionary items** in the **planning table**.

The complete *complaints* case model, including `discretionary items` and decorators, looks as follows,
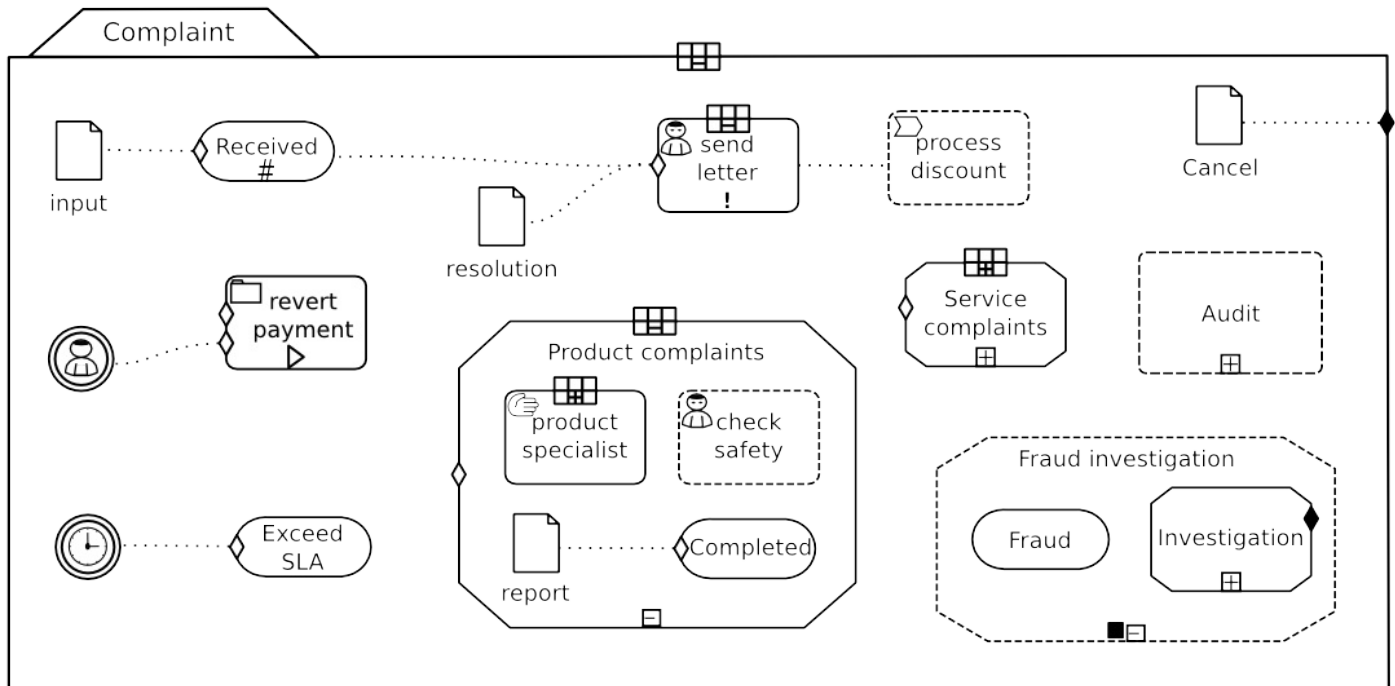
There are few things we left out of the model, namely *Service complaints* and *Audit* are collapsed, so we did not show their content; *Product complaints* `stage` and *product specialist* `task` contain collapsed `planning tables`, so we don't show the `discretionary items` they contain.

Note that the full CMMN model for the complaint process may look disconcerting to a person used to workflow or process models, because not everything is modeled, and because most of what is modeled can be disabled at execution time by case workers.

Case Worker Actions

As we have described before, CMMN gives case workers a lot of control over the execution of a case instance. Although case worker privileges are controlled using roles, it is important to distinguish two types of case workers.

- Case workers executing `tasks` in the case. These workers may have limited privileges and will have similar characteristics that workers in other process or workflow technologies.
- Case workers controlling the case. These are sometimes referred as knowledge workers. In the *complaints process* we have a case worker dealing with the customer and in charge of the case. Some of the activities that case workers controlling the case are able to do include,

Case planning. Adding `discretionary items` to the case plan.

Manual activation. Deciding when a `task` or `stage` should be executed (by manually starting it), or deciding that it should not be executed for a case instance (by disabling it).

Auto complete. Deciding when a `stage` or `case` without `auto complete` should be manually completed.

Suspend and resume. Deciding when to suspend or resume execution of the `case`, `task`, `stage`, `event listener`, or `milestone`.

Ignoring fault conditions. Deciding to continue a `case`, a `task`, or a `stage` that has an error condition.

Adding or modifying data on the case. Adding, creating, replacing, deleting, and modifying data (`case file items`) in the case.

Closing the case. Deciding when to close a case, so that nothing else can be done in that case.

Summary

Case management looks at a process from the perspective of the case workers, with the goal of enabling them to efficiently collaborate between them to achieve a business goal. CMMN achieves that by allowing execution time planning of cases, having the concept of manually activated `tasks` and `stages`, case worker activated events, and reacting to creation, update, and delete of case data via `entry criteria` and `exit criteria`. Case planning itself can be modeled in advance, by providing `tasks` and `stages` that allow planning with a `planning table`. However, any worker in a role that allows planning can do planning at any moment during the case instance execution. Planning is based on the concept of `discretionary items` that are modeled to be used at the discretion of the case workers. Manually activated `tasks` and `stages` are those for which the `entry criteria` has been met, but they are only executed, if a case worker decides to do so.

CMMN formalize the concept of a `case file` (or case folder) that contains all the `case file items` (case data). In most instances, case data is represented by documents, because workers commonly interact by using documents like spreadsheets, presentations, word processor documents, voice recordings, videos, pictures, *etc.* The ability to add, modify, or remove data from a case file at any time during the process is a key feature of a case management system.

Case workers have always access to the `case file` and all the case data. They can be authorized to access the `case file` and its data even when they don't have work assigned. The interaction of case workers with case data may trigger additional `tasks` or activities in the case.

References

OMG. Case Management Model and Notation, version 1.0. Technical Report May, OMG, 5 2014. Document formal/2014-05-05.

Table of Contents