

# Prompt Engineering

Peter Heinemann / 2141060

`peter.heinemann@hs-augsburg.de`

## Abstract

ChatGPT erfreut sich großer Beliebtheit in den unterschiedlichsten Bereichen. Es wird zum Zusammenfassen von Texten, bei der Softwareentwicklung und zum Beantworten verschiedenster Fragen genutzt. Doch die Ausgabe ist nicht vorherzusehen und häufig enthält diese Fehler. Mit Prompt Engineering soll versucht werden, darauf Einfluss zu nehmen. Diese Studienarbeit beschäftigt sich mit den Fragen, was sich hinter diesem Begriff verbirgt und wie groß der Einfluss tatsächlich auf die Ausgabe ist. Dafür werden bereits bekannte Methoden und Anwendungsfälle anhand von Beispielen aus dem Bereich der Prozessmodellierung mit BPMN getestet. Diese Beispiele geben einen ersten Einblick darauf, wie stark ein LLM auf Änderungen am Prompt reagiert. Der Prompt nimmt unter anderem Einfluss auf die Richtung der Interaktion, sodass ChatGPT die Fragen stellt und kann das Modell zum "Nachdenken" zwingen.

## 1 Einleitung

Prompt Engineering ist ein noch sehr neuer Begriff aus dem Bereich des Natural Language Processing. Prominenz hat dieser Begriff vor allem mit der Veröffentlichung von ChatGPT gewonnen. Heutzutage verbindet man damit unter anderem einen Beruf, da viele Firmen nach einen Prompt Engineer suchen. Mit dieser Studienarbeit möchte ich anhand eines kleinen Projektes durch eigene Beispiele herausfinden, wie Prompts durch ein systematisches Herangehen verbessert werden können. Dafür habe ich mich auf zwei Quellen beschränkt, aus denen ich Anwendungsfälle und Methoden entnehme, um diese im Kontext von Business Process Modelling and Notation (BPMN) auszuprobieren. Die Modellierung von Prozessen mit BPMN weist Parallelen zu der reinen Softwareentwicklung auf. Da Large Language Models (LLM) sich bereits als taugliche Werkzeuge herausgestellt haben, die die Softwareentwicklung vereinfachen ([White et al., 2023](#)),

stellt sich die Frage, ob das auch für die Prozessentwicklung gilt. Diese Arbeit wird die Frage nicht vollständig klären, sondern ist als ein erster Schritt gedacht, sich mit einem LLM auseinanderzusetzen. Deswegen liegt der Fokus darauf, sich mit der Vorgehensweise von Prompt Engineering vertraut zu machen und verschiedene Methoden auszuprobieren. Zu den Methoden bzw. Anwendungsfälle gehören das Extrahieren von Informationen, das Generieren eines BPMN Prozesses und das Beeinflussen der Ausgabe bezüglich Korrektheit und Struktur. Das Ziel ist es nicht praxistaugliche Anwendungsfälle zu erarbeiten. Des Weiteren habe ich mich entschieden, kein Finetuning durchzuführen und in den meisten Fällen auch mit One-Shot Prompting ([Ng and Fulford](#)) zu arbeiten. Dadurch erhoffe ich mir ein besseres Gefühl zu bekommen, wie gut ChatGPT "out of the box" mit BPMN arbeiten kann. Ich verwende ChatGPT sowie die OpenAI API aufgrund deren Beliebtheit und des einfachen Zugangs. Die Beispiele befinden sich in einem Jupyter Notebook, dass dieser Arbeit beiliegt.

## 2 BPMN

BPMN ist eine Spezifikation zur Beschreibung von Geschäftsprozessen. Der Standard wird von der Object Management Group (OMG) gepflegt, einer Organisation, die vor allem für die Unified Modelling Language bekannt ist ([OMG, 2011](#)). Ein Geschäftsprozess wird als eine Reihe von Aktivitäten definiert, die einen Kundennutzen stiftenden Output besitzen ([Lackes et al., 2023](#)). Das Ziel von BPMN Diagrammen ist jedoch nicht nur die Darstellung von Geschäftsprozessen in einer für Menschen verständlichen Art und Weise, sondern auch das Ausführen der Prozesse durch Software ([OMG, 2011](#)).

Um dies zu erreichen, wird ein BPMN Diagramm als eine XML-Datei abgespeichert, die durch sogenannte Workflow Engines ausgeführt werden kann. Im Idealfall führt das zu einer kompletten Automa-

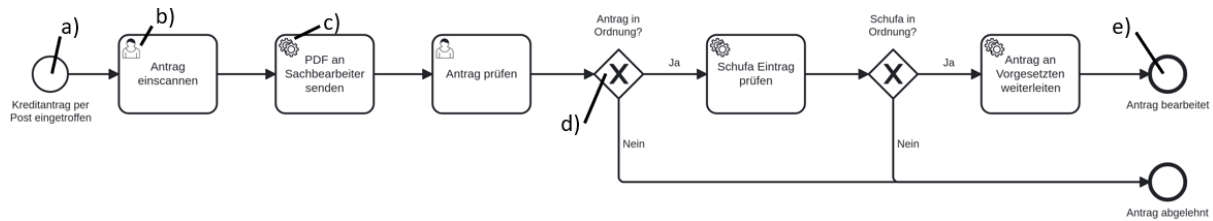


Abbildung 1: BPMN Diagramm über einen Kreditantragsprozess. a) Startevent b) User Task c) Service Task d) Exklusives Gateway e) Endevent

tisierung der Prozesse. Jedoch sind in den meisten Fällen weiterhin menschliche Aktivitäten notwendig. (Freund and Rücker, 2019)

Da der Fokus dieser Arbeit auf Prompt Engineering liegt, werden nur einfache BPMN Diagramme verwendet. Abbildung 1 zeigt ein mögliches BPMN Diagramm für einen Kreditantrag. In diesem Diagramm sind die am häufigsten verwendeten Elemente enthalten. (OMG, 2011)

1. **Events** oder **Ereignisse** stellen Vorgänge dar, die bereits eingetreten sind und einen Einfluss auf unseren Prozess haben. Das Start- und Endevent definieren den Anfang und das Ende eines Prozesses. Neben diesen Ereignissen kennt BPMN noch weitere, die auch während des Prozesses auftreten können.
2. **Tasks** bzw. **Aufgaben** werden durch ein Rechteck mit abgerundeten Ecken dargestellt. Sie beschreiben eine Aktivität, die ausgeführt werden soll. Es gibt viele verschiedene Aufgabentypen, die durch ein Symbol an der oberen linken Ecke gekennzeichnet werden. Die in Abbildung 1 verwendeten Typen geben in erster Linie an, durch wen die Aufgabe bearbeitet wird. Ein User Task wird durch einen Menschen und ein Service Task durch eine Maschine/Software erledigt.
3. Das **Gateway** stellt in diesem Beispiel eine Entscheidung dar. Je nach Entscheidung wird ein unterschiedlicher Zweig durchlaufen. Auch hier gibt es noch weitere unterschiedliche Typen an Gateways.
4. Der **Sequenzfluss** wird durch die Pfeile dargestellt und beschreibt die "zeitlich-logische Reihenfolge in der die Flusselemente zueinander stehen" (Freund and Rücker, 2019).

### 3 Prompt Engineering

Der Begriff des Prompt Engineering kam vor allem nach der Veröffentlichung von ChatGPT vermehrt auf. Mittlerweile ist der Prompt Engineer ein Beruf, der sich mit LLMs beschäftigt. Das Ziel ist es dabei, LLMs in die eigene Organisation oder Produkte einzubinden. Dafür muss das Modell häufig an spezielle Anwendungsbereiche angepasst werden. (Maeda and Bolaños, 2023)

Dies kann mit Finetuning, aber auch mit Anpassen der Prompts erreicht werden. In den folgenden Unterkapiteln werden verschiedenen Methoden und Anwendungsbereiche des Prompt Engineerings vorgestellt.

#### 3.1 Prompt

Der Prompt (engl. auffordern) ist die Aufforderung eines Benutzers an das LLM eine bestimmte Aufgabe zu erledigen. In vielen Fällen, wie auch in dem Beispiel aus Abbildung 3 besteht die Aufgabe darin eine Frage zu beantworten. An diesem kleinen Beispiel sieht man bereits, dass ChatGPT häufig nicht nur die Antwort zu unserer eigentlichen Frage, sondern noch weitere Zusatzinformationen liefert. Nehmen wir nun an, dass unsere Anwendung nur das Ergebnis in Kilogramm benötigt. Mit der kleinen Anpassung aus dem Beispiel in Abbildung 4 des originalen Prompt kommen wir unseren gewünschten Output etwas näher. Abbildung 4 zeigt bereits ein kleines Beispiel für Prompt Engineering.

#### 3.2 Vorgehensweise

Die Ausgabe von ChatGPT kann nicht zu 100 Prozent vorhergesehen werden. Zum Beispiel wurde mit dem Prompt aus Abbildung 3 nur eine Zahl als Antwort erwartet. ChatGPT hat allerdings noch weitere Informationen ausgegeben, nach denen nicht gefragt wurde. Selbst mit dem Prompt aus Abbildung 4 wird weiterhin ein Satz als Ausgabe generiert. Somit müsste der Prompt weiter angepasst

werden. Daran ist zu erkennen, dass eine iterative Vorgehensweise nötig ist, um an den letztendlich für den jeweiligen Anwendungsfall funktionierenden Prompt zu kommen.

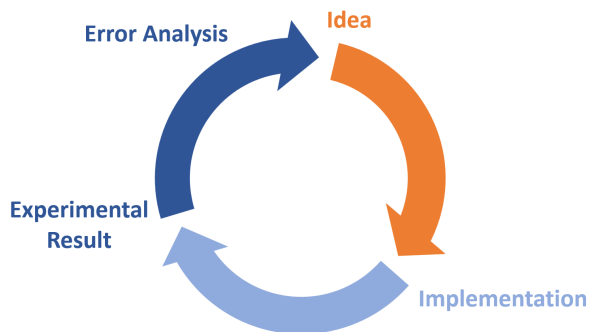


Abbildung 2: Vorgehensmodell bei der Promptentwicklung (Ng and Fulford)

Abbildung 2 stellt das iterative Vorgehen grafisch dar. Dabei steht die Idee für einen Prompt am Anfang eines Kreislaufes. Dieser Prompt wird an ChatGPT übergeben (Implementation) und erhält eine Ausgabe (Experimental Result). Ist das Ergebnis nicht zufriedenstellend, muss der Prompt angepasst werden bzw. die Idee muss überarbeitet werden. Dies wird so lange wiederholt, bis die Ausgabe dem gewünschten Ergebnis entspricht. (Ng and Fulford)

### 3.3 Vorbereitung

Um verschiedene Anwendungsfälle zu testen habe ich drei BPMN Diagramme erstellt. Alle drei Diagramme sind einfach gehalten und nutzen lediglich die in Abschnitt 2 erwähnten BPMN Elemente. Die Diagramme sind dem Anhang beigelegt und entsprechend markiert. Das erste Diagramm (Anhang 1) beschreibt einen Prozess, dessen Ziel die Besetzung einer freien Stelle ist. Im zweiten Diagramm (Anhang 2) geht es um die Auswahl einer Mahlzeit. Dabei besteht die Auswahl zwischen zwei verschiedenen Gerichten, die durch ein exklusives Gateway modelliert wurde. Der dritte Prozess (Anhang 3) beschäftigt sich mit der Erstellung einer Rechnung. Der Unterschied zu den vorherigen Diagrammen liegt in einer Schleife, die es erlaubt eine Erinnerung zur Zahlung mehrmals zu versenden, wenn die Rechnung nicht bezahlt wurde. Die drei BPMN Prozesse sind der Arbeit beigelegt und können z.B. mit dem Visual Studio Code Plugin "Miranum Modeler"<sup>1</sup> oder im Browser<sup>2</sup> angezeigt werden.

<sup>1</sup><https://marketplace.visualstudio.com/items?itemName=miranum.gmbh.vs-code-bpmn-modeler>

<sup>2</sup><https://demo.bpmn.io/>

### 3.4 Prinzipien

Unabhängig vom Anwendungsfall gelten zwei Prinzipien, die beim Prompt Engineering helfen können (Ng and Fulford):

1. Klare und spezifische Befehle
2. Dem Modell Zeit zum "Nachdenken" geben

Je mehr Informationen ein Prompt über die zu erledigende Aufgabe enthält, desto wahrscheinlicher ist es, dass die gewünschte Ausgabe erfolgt. Mit klaren Befehlen sind daher keine kurzen Befehle gemeint, sondern eine klar strukturierte Eingabe. (Ng and Fulford)

Anhang 5 zeigt einen möglichen Prompt, der diesem Prinzip folgt. Das Ziel war hierbei den BPMN Prozess als eine Liste zu erhalten. Wobei jeder Listeneintrag folgende Struktur haben soll:

- Die erste Nummer gibt die Position des BPMN Elements im Diagramm an.
- Danach wird die Art des BPMN Elements angegeben (z.B. Startevent, User Task, etc.)
- Auf das Element folgt das Label des Elements, dass von zwei Hochkommata eingeschlossen wird
- Zum Schluss soll mit einem Pfeil (->) und einer Nummer das nächste Element im Diagramm angegeben werden.

Für diese Aufgabe benötigt ChatGPT das XML, dass den BPMN Prozess beschreibt. Um dem LLM klarzumachen, welcher Teil des Prompt das XML ist, werden Trennzeichen genutzt. Des Weiteren wird eine Vorlage mit den Anweisungen aus oben genannter Liste mitgegeben, um die gewünschte Ausgabe zu erhalten.

Das zweite Prinzip fordert dem Modell Zeit zum "Nachdenken" zu geben. Gemeint ist damit, dem Modell Anweisungen zu geben, die nacheinander abgearbeitet werden sollen, bevor eine Ausgabe erfolgt (Ng and Fulford). Anhang 6 zeigt, wie dieses Vorgehen dabei hilft, die richtige Antwort zu erhalten. In dem Beispiel soll aus dem gegebenen BPMN Diagramm erkannt werden, ob ein Zyklus vorhanden ist oder nicht. Die Antwort soll entweder "Ja" oder "Nein" lauten. Das Ergebnis aus den ersten beiden Prompts aus Anhang 6 lautet in beiden Fällen "Nein", obwohl beim zweiten Prompt das Diagramm einen Zyklus aufweist. Für beide

User: How much mass does our sun have?  
ChatGPT: The mass of the Sun is approximately  $1.989 \times 10^{30}$  kilograms (about  $4.387 \times 10^{30}$  pounds). This value represents about 99.86% of the total mass of the entire solar system. The Sun's immense mass provides the gravitational force necessary to hold the planets and other objects in orbit around it.

Abbildung 3: Beispiel für eine Interaktion mit ChatGPT.

User: How much mass does our sun have?  
Just give me the result in kilograms.  
ChatGPT: The mass of the Sun is approximately  $1.989 \times 10^{30}$  kilograms.

Abbildung 4: Die gleiche Frage aus Abbildung 3 etwas spezifiziert.

Antworten braucht ChatGPT nur wenige Sekunden. Wird der Zusatz der einsilbigen Antwort weggelassen (Anhang 6, prompt\_3) benötigt das Modell weitaus länger, liefert aber die richtige Antwort. Dieses Ergebnis besteht jedoch aus einem längeren Text. Um nun dennoch eine klare "Ja"- oder "Nein"-Antwort zu erhalten, wird im vierten Prompt die Schritte angegeben, die ausgeführt sein müssen, bevor das Modell eine definitive Antwort geben soll. Zwar enthält die Ausgabe nicht nur die einsilbige Antwort, aber dadurch konnte die Ausgabe strukturiert werden ohne, dass die Korrektheit der Antwort beeinflusst wurde.

### 3.5 Informationen extrahieren / zusammenfassen

Eine beliebte Aufgabe für ChatGPT ist es, aus langen Texten die relevanten Aussagen zu extrahieren (Ng and Fulford). Um diesen Anwendungsfall auf BPMN Prozesse zu übertragen, habe ich mir verschiedene Aufgaben ausgedacht. Die Ergebnisse sind unter Anhang 6 gesammelt.

Die erste Aufgabe ist es, einen Prozess in einem Satz zu beschreiben. BPMN Prozesse können sehr schnell, sehr komplex und unübersichtlich werden. Teilweise wird ein Prozess über mehrere Teilprozesse verteilt. Somit könnte es für Prozessentwickler hilfreich sein, sich eine kurze Beschreibung der Prozesse generieren zu lassen. Das Ergebnis ist eine gute Beschreibung der drei Prozesse.

Mit den Prompts aus Anhang 5 sollte eine detailliertere Beschreibung erfolgen. Das Prinzip wurde bereits in Kapitel 3.4 beschrieben. Hierbei war das

Ergebnis unterschiedlich. Während die generierte Liste für den ersten Prozess korrekt erstellt wurde, gab es bei den beiden anderen Prozessen Probleme. Zum einen wurden die Sequenzflüsse nicht wie gewünscht weggelassen und zum anderen wurde der dritte Prozess auch häufiger falsch beschrieben. Das Problem mit den Sequenzflüssen konnte mit einem Few-Shot Prompt (Anhang 5, prompt\_4) gelöst werden.

Mit der zweiten Aufgabe (Anhang 6) sollten die verwendeten BPMN Element aufgelistet und zusammen mit der Anzahl ihres Vorkommens ausgegeben werden. Dies kann ein erster Hinweis auf die Komplexität des Prozesses geben, ohne diesen sich selbst anschauen zu müssen. Die von ChatGPT ausgegebene Liste enthält leider auch nicht relevante Informationen. Dazu gehören Listeneinträge, die keine BPMN Elemente darstellen sowie das bekannte Problem mit dem nicht entfernen von Sequenzflüssen.

Die dritte und letzte Aufgabe aus Anhang 6 sollte den dritten Prozess analysieren und eine Frage zum Prozess beantworten. Das Ergebnis wurde bereits in Kapitel 3.4 beschrieben. Hierbei war es besonders interessant zu beobachten, wie die Struktur der Ausgabe die Korrektheit beeinflusst.

### 3.6 Flipped Interaction Pattern

Mit dem Flipped Interaction Pattern wird die Interaktion mit dem LLM umgedreht. Anstelle des Benutzers, stellt nun das Modell die Fragen. Das kann dabei helfen, wenn nicht klar ist, welche Informationen das LLM benötigt, um die gewünschte



Aufgabe zu lösen. Das Pattern führt dazu, dass dem Benutzer so lange Fragen gestellt werden, bis genügend Informationen vorhanden sind, um die eigentliche Aufgabe zu lösen. Anhang 7 zeigt, wie mit diesem Pattern ein BPMN Diagramm generiert werden kann. Zunächst möchte das Modell eine allgemeine Beschreibung des Prozesses, bevor es nach den spezifischen Schritten fragt. Die Angaben, die der Nutzer machen muss, müssen jedoch sehr genau sein und erfordern erweitertes Wissen in BPMN. Es musste auch zweimal nachgebessert werden, bevor ein akzeptables Diagramm erstellt wurde. Das Ergebnis ist in Abbildung 6 zu sehen. Das generierte Diagramm ist für einen Menschen nur schwer zu lesen, jedoch für eine Workflow Engine ausführbar. Im Anhang<sup>3</sup> befindet sich derselbe Prozess allerdings etwas nachgearbeitet, sodass er auch für einen Menschen nachvollziehbar ist. Abbildung 5 zeigt den verwendeten Prompt.

```
From now on, I would like
you to ask me questions to
create a executable BPMN
2.0 Diagram. When you have
enough information to
create the BPMN Diagram
give me the XML for this
diagram. Ask me the
questions one at a time.
```

Abbildung 5: Beispiel zur Anwendung des Flipped Interaction Pattern.

### 3.7 Reflection Pattern

Das Ziel dieses Pattern ist es, einen nachvollziehbaren Output zu generieren. Das Modell soll nicht nur das Ergebnis liefern, sondern auch erklären, wieso es zu diesem Ergebnis gekommen ist. Dies ermöglicht es dem Benutzer die Qualität der Antwort besser einzuschätzen. (White et al., 2023)

Mit diesem Vorgehen wird das Modell zum "Nachdenken" (siehe Kapitel 3.4) gezwungen.

Dieser Prompt eignet sich für spezifische Fragen zu einem Thema, indem der Benutzer selbst wenig Wissen besitzt. Er kann die zusätzliche Erklärung nutzen, um das Ergebnis zu validieren. Im Bereich von BPMN könnte z.B. eine Frage zu der offiziellen BPMN 2.0 Spezifikation gestellt werden. Anhang 4 enthält einen Prompt, der nach den unterschied-

lichen Task-Typen fragt. Hier liefert das Modell eine falsche Antwort, indem auch andere BPMN Elemente aufgelistet werden. In Anhang 8 wird das Reflection Pattern genutzt, in der Hoffnung eine korrekte Antwort zu erhalten. Tatsächlich wird diese auch ausgegeben. Der verwendete Prompt ist in Abbildung 7 beschrieben. Auch auf eine Frage zu Camunda 7, einer Open Source Workflow Engine, die häufig von ChatGPT falsch beantwortet wird, erfolgt eine korrekte Ausgabe.

### 3.8 Persona Pattern

Das Persona Pattern gibt dem LLM eine bestimmte Rolle vor, aus dieser nicht ausgebrochen werden sollte. Es eignet sich daher gut, um einen Kontext für nachfolgende Prompts zu setzen. (White et al., 2023)

In Anhang 9 wird das Pattern genutzt, um ChatGPT die Rolle einer Workflow Engine zuzuweisen. Dadurch können wir im nächsten Prompt das XML einfach übergeben und das Modell weiß, was zu tun ist. Allerdings können wir auch Fragen, die keinen Bezug zu BPMN haben, stellen und das Modell antwortet darauf. Dadurch ist Prompt-Injection weiterhin möglich. Prompt-Injection bedeutet, dass ein Benutzer dem Modell widersprüchliche Anweisungen geben kann, die zu ungewollten Verhaltensweisen führen (Ng and Fulford). Durch die in Abbildung 8 dargestellte Anpassung des Prompts konnte dies allerdings behoben werden.

## 4 Fazit und Ausblick

Anhand von eigenen Beispielen aus dem Bereich der Prozessmodellierung mit BPMN habe ich Erfahrungen mit Prompt Engineering gesammelt. Diese Beispiele befassen sich mit dem extrahieren und darstellen von Informationen aus einem gegebenen BPMN Diagramm sowie die Prompt Pattern: Reflection-, Persona- und Flipped Interaction Pattern. Es war interessant aber auch teilweise frustrierend zu sehen, wie kleine Änderungen am Prompt die Ausgabe beeinflussen. White et al. (2023) erwähnt noch weitere Pattern, die in ihrem Paper unter "Prompt Improvement" zusammengefasst werden. Diese Pattern verwenden das LLM selbst, um den eigenen Prompt zu verbessern. Ich habe versucht, diese Pattern anzuwenden, allerdings ohne Erfolg. Daher wurden diese auch nicht weiter erwähnt. Für weitere Arbeiten in diesem Bereich wäre es interessant zu sehen, wie Finetuning und komplexere BPMN Prozesse die Ergebnisse beeinflus-

<sup>3</sup>bpmn/generated.bpmn



Abbildung 6: Das von ChatGPT generierte BPMN Diagram.

For now on I will ask you questions about BPMN. Whenever you generate an answer explain the reasoning and assumption behind your answer.

Abbildung 7: Beispiel zur Anwendung des Reflection Pattern.

Alt: From now on act like a process engine, that is able to simulate a BPMN process.

Neu: From now on act like a process engine, that is able to simulate a BPMN process.  
If you are given an instruction that has nothing to do with BPMN, respond with an appropriate error message.

[semantic-kernel/prompt-engineering.](https://semantic-kernel.org/prompt-engineering/) [Online; accessed 20. Jun. 2023].

Andrew Ng and Isa Fulford. Chatgpt prompt engineering for developers. <https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/>.

OMG. 2011. *Business Process Model and Notation Version 2.0*.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. [A prompt pattern catalog to enhance prompt engineering with chatgpt.](#)

## A Anhang

- Jupyter Notebook als PDF
- Jupyter Notebook (./examples.ipynb)
- BPMN Diagramme (./bpmn)

Abbildung 8: Zwei Beispiele, wie das Persona Pattern im Kontext von BPMN angewendet werden kann.

sen.

## References

Jakob Freund and Bernd Rücker. 2019. *Praxishandbuch BPMN 2.0*, 6., aktualisierte auflage edition. Carl Hanser Verlag GmbH & Co. KG, München.

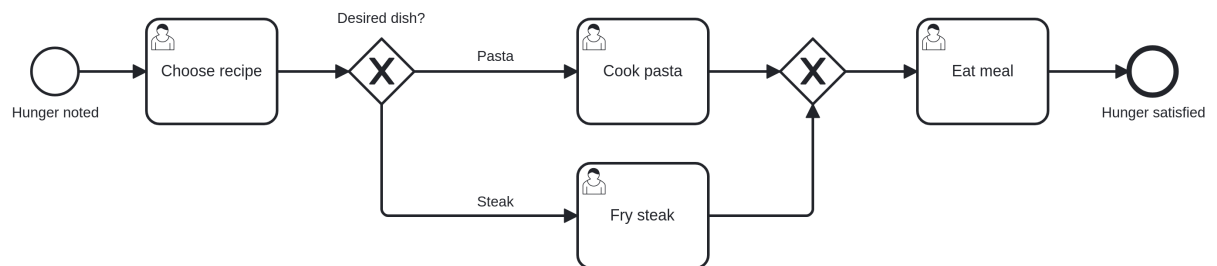
Richard Lackes, Markus Siepermann, and Gerhard Schewe. 2023. Definition: Geschäftsprozess. <https://wirtschaftslexikon.gabler.de/definition/geschaeftsprozess-35399/version-258881>. [Online; accessed 20. Jun. 2023].

John Maeda and Matthew Bolaños. 2023. Prompt engineering overview. <https://learn.microsoft.com/en-us/>

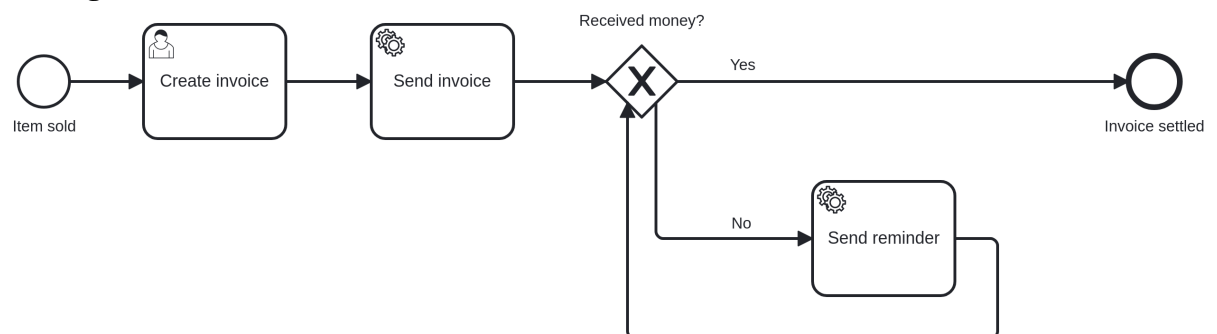
## Anhang 1



## Anhang 2



## Anhang 3



# examples

July 10, 2023

## 1 Prompt Engineering

### 1.1 Setup

Add a .env file with following content:

OPENAI\_API\_KEY=<Your key>

```
[1]: import openai
import os

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file

openai.api_key = os.getenv('OPENAI_API_KEY')
```

```
[2]: def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0, # this is the degree of randomness of the model's output
    )
    return response.choices[0].message["content"]
```

#### 1.1.1 Read BPMN Diagrams

##### Anhang 1 - Application Process



```
[3]: reader = open("bpmn/application_process.bpmn")
application = reader.read()
```

##### Anhang 2 - Cooking Process



```
[4]: reader = open("bpmn/cooking.bpmn")
cooking = reader.read()
```



## Anhang 3 - Invoice Process



```
[5]: reader = open("bpmn/invoice.bpmn")
invoice = reader.read()
```

## 1.2 Anhang 4 - How well does ChatGPT understand BPMN?

### 1.2.1 Question to the BPMN 2.0 Specification

```
[14]: prompt = f"""
List all different task types of the BPMN 2.0 specification.
"""
response = get_completion(prompt)
print(response)
```

1. Task
2. User Task
3. Manual Task
4. Service Task
5. Business Rule Task
6. Script Task
7. Send Task
8. Receive Task
9. Call Activity
10. Sub-Process Task
11. Event Sub-Process Task
12. Transaction Task
13. Ad-Hoc Sub-Process Task
14. Compensation Task
15. Gateway Task
16. Exclusive Gateway
17. Inclusive Gateway
18. Parallel Gateway
19. Complex Gateway
20. Event-Based Gateway.

#### Result:

- Wrong answer Gateways or Sub-Processes are not a Task Type

### 1.2.2 Describe a given BPMN Diagram

```
[12]: prompt = f"""
Your task is to describe a given BPMN Diagram.
```{application}```
"""
response = get_completion(prompt)
```

```
print(response)
```

The BPMN diagram represents a process for filling a job vacancy. The process starts with a "Vacancy emerged" event and then moves to a "Report vacancy" user task. Once the vacancy is reported, the process moves to a "Create job posting" user task. After the job posting is created, the process moves to a "Application received" service task. Once the application is received, the process moves to a "Hold application exam" user task. After the exam is held, the process moves to a "Select applicant" user task. Finally, the process ends with a "Position filled" end event. The diagram includes sequence flows connecting the events and tasks, and BPMN shapes representing the events and tasks.

```
[55]: prompt = f"""
Your task is to describe a given BPMN Diagram.
```{cooking}```
"""
response = get_completion(prompt)
print(response)
```

The given BPMN diagram represents a process for choosing and preparing a meal. Here is a description of the diagram:

The process starts with a "Hunger noted" event, represented by a start event. From this event, a sequence flow leads to the "Choose recipe" user task, represented by a rectangle with the name "Choose recipe". After the user task, another sequence flow leads to an exclusive gateway with the name "Desired dish?". This gateway has two outgoing sequence flows, one named "Pasta" and the other named "Steak".

If the "Pasta" sequence flow is taken, it leads to the "Cook pasta" user task. If the "Steak" sequence flow is taken, it leads to the "Fry steak" user task. Both user tasks are represented by rectangles with their respective names.

After the "Cook pasta" and "Fry steak" user tasks, the process converges at another exclusive gateway, represented by a diamond shape. This gateway has two incoming sequence flows, one from the "Cook pasta" user task and the other from the "Fry steak" user task. From this gateway, a sequence flow leads to the "Eat meal" user task, represented by a rectangle with the name "Eat meal".

Finally, from the "Eat meal" user task, a sequence flow leads to the "Hunger satisfied" end event, represented by a circle with the name "Hunger satisfied".

Overall, this BPMN diagram represents a process where the user notes their hunger, chooses a recipe (either pasta or steak), cooks the chosen dish, eats the meal, and finally, their hunger is satisfied.

```
[56]: prompt = f"""
Your task is to describe a given BPMN Diagram.
```

```

    ...{invoice}...
    """
    response = get_completion(prompt)
    print(response)

```

The given BPMN diagram represents a process flow for handling the sale of an item and the associated invoicing process. Here is a description of the diagram:

1. The process starts with an event called "Item sold" (Start Event). This event signifies the initiation of the process when an item is sold.
2. From the start event, there is a sequence flow that connects to a user task called "Create invoice" (User Task). This task represents the creation of an invoice for the sold item.
3. After the "Create invoice" task, there is another sequence flow that connects to a service task called "Send invoice" (Service Task). This task represents the action of sending the created invoice to the customer.
4. From the "Send invoice" task, there is a sequence flow that connects to an exclusive gateway called "Received money?" (Exclusive Gateway). This gateway represents a decision point based on whether the money for the invoice has been received or not.
5. The "Received money?" gateway has two incoming sequence flows, one from the "Send invoice" task and another from a service task called "Send reminder" (Service Task). These sequence flows represent the different paths that lead to the decision point.
6. From the "Received money?" gateway, there are two outgoing sequence flows. One flow is labeled "No" and connects to the "Send reminder" task, representing the action of sending a reminder if the money has not been received. The other flow is labeled "Yes" and connects to an end event called "Invoice settled" (End Event), representing the successful settlement of the invoice.

Overall, this BPMN diagram represents a process flow for handling the sale of an item, including the creation of an invoice, sending the invoice, and making a decision based on whether the money has been received or not.

### Result:

- Extracted the correct:
  - Process meaning (e.g. "filling a job vacancy")
  - Element Name (e.g. "Vacancy emerged")
  - Task Type (e.g User Task)
  - Sequence Flow
- Include unnecessary information (e.g. the shape of a task or gateway)
- Different output formats over the three prompts

### 1.3 Anhang 5 - Describing the BPMN process

```
[10]: prompt_1 = f"""
You will be provided with an BPMN Diagram as XML delimited by triple backticks.
Whenever I mention "relevant" BPMN elements I mean all BPMN elements described
↳in the official specification excluding the sequence flow.
Describe the process as a list of items.
Every item represents a relevant BPMN element.
The list has the following structure:

1. <ELEMENT> "<NAME>" -> 2.
2. <ELEMENT> "<NAME>" -> 3.
...
N. <ELEMENT> "<NAME>"

Replace <ELEMENT> with the relevant BPMN element (e.g. Startevent, Exclusive
↳Gateway, etc.).
Replace <NAME> with the name attribute in the given XML for the corresponding
↳element.
The first number represents the position of that element.
The number after the arrow (->) represents the next element.

A Gateway can have multiple following elements. If that is the case separate
↳the numbers with a comma.
It is important that the final list do not include any sequence flow.

```{application}```
"""
response_1 = get_completion(prompt_1)
print(response_1)
```

```
1. StartEvent "Vacancy emerged" -> 2
2. UserTask "Report vacancy" -> 3
3. UserTask "Create job posting" -> 4
4. ServiceTask "Application received" -> 5
5. UserTask "Hold application exam" -> 6
6. UserTask "Select applicant" -> 7
7. EndEvent "Position filled"
```

```
[11]: prompt_2 = f"""
You will be provided with an BPMN Diagram as XML delimited by triple backticks.
From now on, whenever I mention "relevant" BPMN elements I mean all BPMN
↳elements described in the official specification excluding the sequence flow.
Describe the process as a list of items.
Every item represents a relevant BPMN element.
The list has the following structure:
```

```

1. <ELEMENT> "<NAME>" -> 2.
2. <ELEMENT> "<NAME>" -> 3.
...
N. <ELEMENT> "<NAME>"

```

Replace <ELEMENT> with the relevant BPMN element (e.g. Startevent, ExclusiveGateway, etc.).

Replace <NAME> with the name attribute in the given XML for the corresponding element. If the attribute name is missing then ignore <NAME>.

The first number represents the position of that element and acts like an id for the element.

The number after the arrow (->) represents the id of the next element.

A Gateway can have multiple following elements. If that is the case separate the numbers with a comma.

It is important that the final list do not include any sequence flow.

```

```{cooking}```
"""

```

```

response = get_completion(prompt_2)
print(response)

```

```

1. StartEvent "Hunger noted" -> 2
2. SequenceFlow -> 3
3. ExclusiveGateway "Desired dish?" -> 4,5
4. SequenceFlow -> 6
5. SequenceFlow -> 7
6. UserTask "Choose recipe" -> 8
7. UserTask "Fry steak" -> 8
8. ExclusiveGateway -> 9
9. SequenceFlow -> 10
10. SequenceFlow -> 11
11. UserTask "Cook pasta" -> 12
12. UserTask "Eat meal" -> 13
13. EndEvent "Hunger satisfied"

```

```

[14]: prompt_3 = f"""
You will be provided with an BPMN Diagram as XML delimited by triple backticks.
From now on, whenever I mention "relevant" BPMN elements I mean all BPMN
elements described in the official specification excluding the sequence flow.
Describe the process as a list of items.
Every item represents a relevant BPMN element.
The list has the following structure:

1. <ELEMENT> "<NAME>" -> 2.
2. <ELEMENT> "<NAME>" -> 3.
...

```

```
N. <ELEMENT> "<NAME>"
```

Replace <ELEMENT> with the relevant BPMN element (e.g. Startevent, Exclusive\_ Gateway, etc.).

Replace <NAME> with the name attribute in the given XML for the corresponding\_ element. If the attribute name is missing then ignore <NAME>.

The first number represents the position of that element and acts like an id\_ for the element.

The number after the arrow (->) represents the id of the next element.

A Gateway can have multiple following elements. If that is the case separate\_ the numbers with a comma.

It is important that the final list do not include any sequence flow.

```
```{invoice}```  
"""
```

```
response = get_completion(prompt_3)  
print(response)
```

1. StartEvent "Item sold" -> 2
2. SequenceFlow -> 3
3. UserTask "Create invoice" -> 4
4. SequenceFlow -> 5
5. ServiceTask "Send invoice" -> 6
6. SequenceFlow -> 7
7. ExclusiveGateway "Received money?" -> 8,9
8. SequenceFlow -> 10
9. SequenceFlow -> 8,9
10. ServiceTask "Send reminder" -> 11
11. SequenceFlow -> 12
12. ExclusiveGateway "Received money?" -> 13,14
13. SequenceFlow -> 15
14. SequenceFlow -> 13,14
15. EndEvent "Invoice settled"

```
[13]: prompt_4 = f"""  
Your task is to answer in a consistent style.  
  
<person>: {prompt_1}  
  
<chatgpt>: {response_1}  
  
<person>: {prompt_2}  
"""  
response = get_completion(prompt_4, "gpt-3.5-turbo-16k")  
print(response)
```

```
<chatgpt>: 1. StartEvent "Hunger noted" -> 2
```



```

2. UserTask "Choose recipe" -> 3
3. ExclusiveGateway "Desired dish?" -> 4,5
4. UserTask "Cook pasta" -> 6
5. UserTask "Fry steak" -> 6
6. ExclusiveGateway -> 7
7. UserTask "Eat meal" -> 8
8. EndEvent "Hunger satisfied"

```

```

[15]: prompt_5 = f"""
Your task is to answer in a consistent style.

<person>: {prompt_1}

<chatgpt>: {response_1}

<person>: {prompt_3}
"""
response = get_completion(prompt_5, "gpt-3.5-turbo-16k")
print(response)

```

```

<chatgpt>: 1. StartEvent "Item sold" -> 2
2. UserTask "Create invoice" -> 3
3. ServiceTask "Send invoice" -> 4
4. ExclusiveGateway "Received money?" -> 5,6
5. ServiceTask "Send reminder" -> 7
6. EndEvent "Invoice settled"
7. EndEvent "Invoice settled"

```

## Result:

- *prompt\_1* leads to the correct output without sequence flows
- *prompt\_2* somehow does not remove the sequence flows sufficiently
  - even after multiple tries with different prompts
  - does not always create a correct list
- *prompt\_3* incorrect process description (11. is never reached)
- *prompt\_4* uses few shot prompting
  - solves the problem with the appearance or sequence flows in the final list
  - needs a different model with more tokens
  - output for third BPMN diagram is wrong

## 1.4 Anhang 6 - Extracting information

```

[9]: processes = [application, cooking, invoice]
for i in range(len(processes)):
    prompt = f"""
    Your task is to generate a short summary of a given BPMN Process. \

    Summarize the process below, delimited by triple

```

```
backticks, in at most 1 sentence.
```

```
Process: ```{processes[i]}```  
"""
```

```
response = get_completion(prompt)  
print(i, response, "\n")
```

0 The given BPMN process involves the steps of reporting a vacancy, creating a job posting, receiving applications, holding an application exam, selecting an applicant, and filling the position.

1 The process involves choosing a recipe based on hunger, cooking the chosen dish, and then eating the meal to satisfy hunger.

2 The given BPMN process involves the steps of creating an invoice, sending the invoice, checking if money is received, and sending a reminder if necessary, with the process ending when the invoice is settled.

```
[11]: prompt = f"""  
You will be provided with an BPMN Diagram as XML delimited by triple backticks.  
Extract all distinct BPMN elements present in the diagram, excluding sequence_  
    ↪flows.  
Also count how often a element is used.  
Display the list in the following format:  
  
* <ELEMENT> <COUNT>  
* <ELEMENT> <COUNT>  
...  
  
Replace <ELEMENT> with the relevant BPMN element (e.g. StartEvent, Exclusive_  
    ↪Gateway, etc.)  
Replace <COUNT> with the number of times the element is contained in the_  
    ↪diagram.  
  
Note that the sequence flows will be excluded from the list, as per my_  
    ↪instructions.  
  
```{application}```  
"""  
response = get_completion(prompt)  
print(response)
```

```
* bpmn:definitions 1  
* bpmn:process 1  
* bpmn:startEvent 1  
* bpmn:sequenceFlow 5  
* bpmn:userTask 4
```

```
* bpmn:serviceTask 1
* bpmn:endEvent 1
```

#### 1.4.1 Search for loop or cycle

```
[16]: prompt_1 = f"""
You will be provided with an BPMN Diagram as XML delimited by triple backticks.
Determine if the diagram contains a loop or cycle in which one or more tasks_
↳can be executed multiple times.
Provide an answer limited to two words "Yes" or "No".

```{cooking}```
"""
response = get_completion(prompt_1)
print(response)
```

No

```
[17]: prompt_2 = f"""
You will be provided with an BPMN Diagram as XML delimited by triple backticks.
Determine if the diagram contains a loop or cycle in which one or more tasks_
↳can be executed multiple times.
Provide an answer limited to two words "Yes" or "No".

```{invoice}```
"""
response = get_completion(prompt_2)
print(response)
```

No

```
[18]: prompt_3 = f"""
You will be provided with an BPMN Diagram as XML delimited by triple backticks.
Determine if the diagram contains a loop or cycle in which one or more tasks_
↳can be executed multiple times.

```{invoice}```
"""
response = get_completion(prompt_3)
print(response)
```

Yes, the BPMN diagram contains a loop or cycle. The loop is formed by the sequence flow between the "Send reminder" task (Activity\_0mgn8rb) and the exclusive gateway (Gateway\_16w5r3m), and the sequence flow between the exclusive gateway and the "Send invoice" task (Activity\_0zlx5ya). This loop allows the "Send reminder" and "Send invoice" tasks to be executed multiple times.

```
[20]: prompt_4 = f"""
You will be provided with an BPMN Diagram as XML delimited by triple backticks.
↪\
Determine if the diagram contains a loop or cycle in which one or more BPMN
↪elements are executed multiple times. \
Don't decide until you followed every possible sequence flow and gateway. \
Perform the following actions: \
1 - Simulate a token. \
    - The token have to follow every possible path step-by-step. \
    - If a Gateway is visited twice simulate each outgoing sequence flow at least
↪twice. \
2 - Create a list with the visited BPMN element of every simulation step. \
3 - Check if the resulting list contains duplicates. \
4 - Provide an answer.

Use the following format: \
Visited elements: \
<list of visited elements> \
Duplicates: \
<duplicates> \
Answer: \
<Yes or No>

```{invoice}```
"""
response = get_completion(prompt_4)
print(response)
```

```
Visited elements: [Event_140mb0o, Flow_1px778j, Activity_0rllx8f, Flow_1yc3h6k,
Activity_Ozlx5ya, Flow_1dcej3e, Gateway_16w5r3m, Flow_09cpp6w, Activity_0mgn8rb,
Flow_10vlhhj, Gateway_16w5r3m, Flow_1gcqri7, Event_Orp17pv]
Duplicates: Gateway_16w5r3m
Answer: Yes
```

## Result

- The output format influence if the answer is correct or incorrect

## 1.5 Anhang 7 - Flipped Interaction

```
[2]: from IPython.display import Markdown, display
display(Markdown("./images/flipped_interaction.md"))
```

You:

From now on, I would like you to ask me questions to create a executable BPMN 2.0 Diagram. When you have enough information to create the BPMN Diagram give me the XML for this diagram. Ask me the questions one at a time.

---

**ChatGPT:**

Sure, I can help you create an executable BPMN 2.0 diagram. Let's start with the first question:

1. What is the process you want to model using BPMN? Please provide a brief description or name for the process.

---

**You:**

1. The process is about the sale of an item and the associated invoicing process.

---

**ChatGPT:**

Great! Now let's move on to the next question:

1. What are the main steps or activities involved in the sales and invoicing process?

---

**You:**

Steps are as follows: 1. Item sold 2. Create invoice 3. Send invoice 4. Wait for payment If we get the money 5. Process ends Else 5. Wait for 1 week 6. Send a reminder

---

**ChatGPT:**

Thank you for providing the steps. Now, let's continue with the next question:

1. Are there any specific roles or participants involved in the process? If so, please provide the names or descriptions of the roles.

---

**You:**

No please create the process without specific roles.

---

**ChatGPT:**

Understood. Moving on to the next question:

1. Are there any specific events that trigger or occur during the sales and invoicing process? Examples of events could be "Item Sold" or "Payment Received." If there are any events, please provide their names or descriptions.

---

**You:**

"Item Sold" is a Start Event. "Wait for payment" is a Timer Event. "Wait 1 week" is another Timer Event. "Payment received" is the End Event.

---

**ChatGPT:**

Got it. Here's the summary of the information you provided: \* Start Event: "Item Sold"\* Activities:

- End Event: "Payment Received" Now, let's proceed to the next question:
- 1. Are there any conditions or gateways that determine the flow of the process? For example, any decision points where the process takes different paths based on certain conditions. If there are any conditions or gateways, please describe them.

---

**You:**

Yes, we need one

---

**ChatGPT:**

Understood. Please provide a description or name for the condition or gateway, and any specific conditions or rules that determine the flow of the process at that point.

---

**You:**

The Gateway has to be after "Wait for Payment". The Gateway has two outgoing sequence flows. One sequence flow is labeled with "Yes" and means that we received the money. The other sequence flow is labeled with "No" and means we have to "Send Reminder".

---

**ChatGPT:**

Thank you for providing the details. Here's the updated summary of the process: \* Start Event: "Item Sold"\* Activities:

- Gateway: After "Wait for Payment" with two outgoing sequence flows
  - Sequence Flow 1: Labeled as "Yes" (Payment received) \* Sequence Flow 2: Labeled as "No" (Send Reminder)
- Activities:
  - 4. Wait 1 Week (Connected to "No" sequence flow)
  - 5. Send Reminder (Connected to "No" sequence flow)
- End Event: "Payment Received" Now, let's proceed to the final question:
  - 1. Are there any additional details or requirements you would like to include in the BPMN diagram? This could be specific events, timers, message flows, or any other relevant information. If there are no additional requirements, please let me know.

---

**You:**

There are no additional requirements.



---

## ChatGPT:

Great! Based on the information you provided, I have created the BPMN diagram for the sales and invoicing process. Here's the corresponding BPMN XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/2010/XMLSchema" xsi:schemaLocation="http://www.w3.org/2001/XMLSchema http://www.w3.org/2001/XMLSchema.xsd">
  <bpmn2:process id="Process_1" isExecutable="true">
    <bpmn2:startEvent id="StartEvent_1">
      <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:task id="Task_1" name="Create Invoice">
      <bpmn2:incoming>SequenceFlow_1</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_2</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:task id="Task_2" name="Send Invoice">
      <bpmn2:incoming>SequenceFlow_2</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_3</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:intermediateCatchEvent id="IntermediateCatchEvent_1">
      <bpmn2:timerEventDefinition/>
      <bpmn2:incoming>SequenceFlow_3</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_4</bpmn2:outgoing>
    </bpmn2:intermediateCatchEvent>
    <bpmn2:exclusiveGateway id="ExclusiveGateway_1">
      <bpmn2:incoming>SequenceFlow_4</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_5</bpmn2:outgoing>
      <bpmn2:outgoing>SequenceFlow_6</bpmn2:outgoing>
    </bpmn2:exclusiveGateway>
    <bpmn2:intermediateCatchEvent id="IntermediateCatchEvent_2">
      <bpmn2:timerEventDefinition/>
      <bpmn2:incoming>SequenceFlow_6</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_7</bpmn2:outgoing>
    </bpmn2:intermediateCatchEvent>
    <bpmn2:task id="Task_3" name="Send Reminder">
      <bpmn2:incoming>SequenceFlow_7</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_8</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:endEvent id="EndEvent_1">
      <bpmn2:incoming>SequenceFlow_5</bpmn2:incoming>
    </bpmn2:endEvent>
    <bpmn2:endEvent id="EndEvent_2">
      <bpmn2:incoming>SequenceFlow_8</bpmn2:incoming>
    </bpmn2:endEvent>
    <bpmn2:sequenceFlow id="SequenceFlow_1" sourceRef="StartEvent_1" targetRef="Task_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_2" sourceRef="Task_1" targetRef="Task_2"/>
    <bpmn2:sequenceFlow id="SequenceFlow_3" sourceRef="Task_2" targetRef="IntermediateCatchEvent_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_4" sourceRef="IntermediateCatchEvent_1" targetRef="ExclusiveGateway_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_5" sourceRef="ExclusiveGateway_1" targetRef="EndEvent_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_6" sourceRef="ExclusiveGateway_1" targetRef="IntermediateCatchEvent_2"/>
    <bpmn2:sequenceFlow id="SequenceFlow_7" sourceRef="IntermediateCatchEvent_2" targetRef="Task_3"/>
    <bpmn2:sequenceFlow id="SequenceFlow_8" sourceRef="Task_3" targetRef="EndEvent_2"/>
  </bpmn2:process>
</bpmn2:definitions>
```

```

<bpmn2:sequenceFlow id="SequenceFlow_4" sourceRef="IntermediateCatchEvent_1" targetRef="Ex
<bpmn2:sequenceFlow id="SequenceFlow_5" sourceRef="ExclusiveGateway_1" targetRef="EndEvent_
  <bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression">#{approved}</bpmn2:conditi
</bpmn2:sequenceFlow>
<bpmn2:sequenceFlow id="SequenceFlow_6" sourceRef="ExclusiveGateway_1" targetRef="Intermed
  <bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression">#{!approved}</bpmn2:conditi
</bpmn2:sequenceFlow>
<bpmn2:sequenceFlow id="SequenceFlow_7" sourceRef="IntermediateCatchEvent_2" targetRef="Ta
<bpmn2:sequenceFlow id="SequenceFlow_8" sourceRef="Task_3" targetRef="EndEvent_2"/>
</bpmn2:process>
<bpmndi:BPMNDiagram id="BPMNDiagram_1">
  <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="Process_1">
    <bpmndi:BPMNShape id="StartEvent_1_di" bpmnElement="StartEvent_1">
      <dc:Bounds x="100" y="100" width="36" height="36"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="Task_1_di" bpmnElement="Task_1">
      <dc:Bounds x="200" y="80" width="80" height="80"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="Task_2_di" bpmnElement="Task_2">
      <dc:Bounds x="350" y="80" width="80" height="80"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="IntermediateCatchEvent_1_di" bpmnElement="IntermediateCatchEvent_1">
      <dc:Bounds x="510" y="80" width="36" height="36"/>
      <bpmndi:BPMNLabel>
        <dc:Bounds x="523" y="123" width="40" height="14"/>
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="ExclusiveGateway_1_di" bpmnElement="ExclusiveGateway_1" isMarkerVi
      <dc:Bounds x="600" y="80" width="50" height="50"/>
      <bpmndi:BPMNLabel>
        <dc:Bounds x="613" y="123" width="24" height="14"/>
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="IntermediateCatchEvent_2_di" bpmnElement="IntermediateCatchEvent_2">
      <dc:Bounds x="680" y="80" width="36" height="36"/>
      <bpmndi:BPMNLabel>
        <dc:Bounds x="682" y="123" width="32" height="14"/>
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="Task_3_di" bpmnElement="Task_3">
      <dc:Bounds x="760" y="80" width="80" height="80"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="EndEvent_1_di" bpmnElement="EndEvent_1">
      <dc:Bounds x="890" y="98" width="36" height="36"/>
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="EndEvent_2_di" bpmnElement="EndEvent_2">
      <dc:Bounds x="890" y="163" width="36" height="36"/>
    </bpmndi:BPMNShape>
  </bpmndi:BPMNPlane>
</bpmndi:BPMNDiagram>

```

```

<bpmndi:BPMNEdge id="SequenceFlow_1_di" bpmnElement="SequenceFlow_1">
  <di:waypoint xsi:type="dc:Point" x="136" y="118"/>
  <di:waypoint xsi:type="dc:Point" x="200" y="118"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="SequenceFlow_2_di" bpmnElement="SequenceFlow_2">
  <di:waypoint xsi:type="dc:Point" x="280" y="118"/>
  <di:waypoint xsi:type="dc:Point" x="350" y="118"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="SequenceFlow_3_di" bpmnElement="SequenceFlow_3">
  <di:waypoint xsi:type="dc:Point" x="430" y="118"/>
  <di:waypoint xsi:type="dc:Point" x="510" y="118"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="SequenceFlow_4_di" bpmnElement="SequenceFlow_4">
  <di:waypoint xsi:type="dc:Point" x="546" y="118"/>
  <di:waypoint xsi:type="dc:Point" x="600" y="105"/>
  <di:waypoint xsi:type="dc:Point" x="600" y="130"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="SequenceFlow_5_di" bpmnElement="SequenceFlow_5">
  <di:waypoint xsi:type="dc:Point" x="650" y="105"/>
  <di:waypoint xsi:type="dc:Point" x="890" y="118"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="SequenceFlow_6_di" bpmnElement="SequenceFlow_6">
  <di:waypoint xsi:type="dc:Point" x="650" y="130"/>
  <di:waypoint xsi:type="dc:Point" x="680" y="118"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="SequenceFlow_7_di" bpmnElement="SequenceFlow_7">
  <di:waypoint xsi:type="dc:Point" x="716" y="118"/>
  <di:waypoint xsi:type="dc:Point" x="760" y="118"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNEdge id="SequenceFlow_8_di" bpmnElement="SequenceFlow_8">
  <di:waypoint xsi:type="dc:Point" x="840" y="118"/>
  <di:waypoint xsi:type="dc:Point" x="890" y="118"/>
</bpmndi:BPMNEdge>
</bpmndi:BPMNPlane>
</bpmndi:BPMNDiagram>
</bpmn2:definitions>

```

Please note that the XML provided above represents the structure and elements of the BPMN diagram based on the information you provided. You can use this XML code to import and execute the BPMN diagram in BPMN-compliant tools or engines.

---

**You:**

Thank you this process already looks very good. But after “Send Reminder” the outgoing sequence flow should go back to the Gateway. This way we have a loop and can execute it multiple times.

---

**ChatGPT:**

Apologies for the oversight. I have updated the BPMN diagram to include the loopback connection from the “Send Reminder” task back to the gateway. Here’s the revised BPMN XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/2010/XMLSchema"
  <bpmn2:process id="Process_1" isExecutable="true">
    <bpmn2:startEvent id="StartEvent_1">
      <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:task id="Task_1" name="Create Invoice">
      <bpmn2:incoming>SequenceFlow_1</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_2</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:task id="Task_2" name="Send Invoice">
      <bpmn2:incoming>SequenceFlow_2</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_3</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:intermediateCatchEvent id="IntermediateCatchEvent_1">
      <bpmn2:timerEventDefinition/>
      <bpmn2:incoming>SequenceFlow_3</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_4</bpmn2:outgoing>
    </bpmn2:intermediateCatchEvent>
    <bpmn2:exclusiveGateway id="ExclusiveGateway_1">
      <bpmn2:incoming>SequenceFlow_4</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_5</bpmn2:outgoing>
      <bpmn2:outgoing>SequenceFlow_6</bpmn2:outgoing>
    </bpmn2:exclusiveGateway>
    <bpmn2:intermediateCatchEvent id="IntermediateCatchEvent_2">
      <bpmn2:timerEventDefinition/>
      <bpmn2:incoming>SequenceFlow_6</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_7</bpmn2:outgoing>
    </bpmn2:intermediateCatchEvent>
    <bpmn2:task id="Task_3" name="Send Reminder">
      <bpmn2:incoming>SequenceFlow_7</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_8</bpmn2:outgoing>
    </bpmn2:task>
    <bpmn2:endEvent id="EndEvent_1">
      <bpmn2:incoming>SequenceFlow_5</bpmn2:incoming>
    </bpmn2:endEvent>
    <bpmn2:sequenceFlow id="SequenceFlow_1" sourceRef="StartEvent_1" targetRef="Task_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_2" sourceRef="Task_1" targetRef="Task_2"/>
    <bpmn2:sequenceFlow id="SequenceFlow_3" sourceRef="Task_2" targetRef="IntermediateCatchEvent_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_4" sourceRef="IntermediateCatchEvent_1" targetRef="ExclusiveGateway_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_5" sourceRef="ExclusiveGateway_1" targetRef="EndEvent_1">
      <bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression">#{approved}</bpmn2:conditionExpression>
    </bpmn2:sequenceFlow>
    <bpmn2:sequenceFlow id="SequenceFlow_6" sourceRef="ExclusiveGateway_1" targetRef="IntermediateCatchEvent_2">
      <bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression">#{!approved}</bpmn2:conditionExpression>
    </bpmn2:sequenceFlow>
    <bpmn2:sequenceFlow id="SequenceFlow_7" sourceRef="IntermediateCatchEvent_2" targetRef="Task_3"/>
    <bpmn2:sequenceFlow id="SequenceFlow_8" sourceRef="Task_3" targetRef="Task_1"/>
  </bpmn2:process>
</bpmn2:definitions>
```

```

    </bpmn2:sequenceFlow>
    <bpmn2:sequenceFlow id="SequenceFlow_7" sourceRef="IntermediateCatchEvent_2" targetRef="Task_3">
    <bpmn2:sequenceFlow id="SequenceFlow_8" sourceRef="Task_3" targetRef="ExclusiveGateway_1"/>
  </bpmn2:process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="Process_1">
      <bpmndi:BPMNShape id="StartEvent_1_di" bpmnElement="StartEvent_1">
        <dc:Bounds x="100" y="100" width="36" height="36"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Task_1_di" bpmnElement="Task_1">
        <dc:Bounds x="200" y="80" width="80" height="80"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Task_2_di" bpmnElement="Task_2">
        <dc:Bounds x="350" y="80" width="80" height="80"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="IntermediateCatchEvent_1_di" bpmnElement="IntermediateCatchEvent_1">
        <dc:Bounds x="510" y="80" width="36" height="36"/>
        <bpmndi:BPMNLabel>
          <dc:Bounds x="523" y="123" width="40" height="14"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_1_di" bpmnElement="ExclusiveGateway_1" isMarkerVisible="true">
        <dc:Bounds x="600" y="80" width="50" height="50"/>
        <bpmndi:BPMNLabel>
          <dc:Bounds x="613" y="123" width="24" height="14"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="IntermediateCatchEvent_2_di" bpmnElement="IntermediateCatchEvent_2">
        <dc:Bounds x="680" y="80" width="36" height="36"/>
        <bpmndi:BPMNLabel>
          <dc:Bounds x="682" y="123" width="32" height="14"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Task_3_di" bpmnElement="Task_3">
        <dc:Bounds x="760" y="80" width="80" height="80"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="EndEvent_1_di" bpmnElement="EndEvent_1">
        <dc:Bounds x="890" y="98" width="36" height="36"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="EndEvent_2_di" bpmnElement="EndEvent_2">
        <dc:Bounds x="890" y="163" width="36" height="36"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_1_di" bpmnElement="SequenceFlow_1">
        <di:waypoint xsi:type="dc:Point" x="136" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="200" y="118"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_2_di" bpmnElement="SequenceFlow_2">
        <di:waypoint xsi:type="dc:Point" x="280" y="118"/>

```

```

        <di:waypoint xsi:type="dc:Point" x="350" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_3_di" bpmnElement="SequenceFlow_3">
        <di:waypoint xsi:type="dc:Point" x="430" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="510" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_4_di" bpmnElement="SequenceFlow_4">
        <di:waypoint xsi:type="dc:Point" x="546" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="105"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="130"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_5_di" bpmnElement="SequenceFlow_5">
        <di:waypoint xsi:type="dc:Point" x="650" y="105"/>
        <di:waypoint xsi:type="dc:Point" x="890" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_6_di" bpmnElement="SequenceFlow_6">
        <di:waypoint xsi:type="dc:Point" x="650" y="130"/>
        <di:waypoint xsi:type="dc:Point" x="680" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_7_di" bpmnElement="SequenceFlow_7">
        <di:waypoint xsi:type="dc:Point" x="716" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="760" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_8_di" bpmnElement="SequenceFlow_8">
        <di:waypoint xsi:type="dc:Point" x="840" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="890" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_9_di" bpmnElement="SequenceFlow_9">
        <di:waypoint xsi:type="dc:Point" x="790" y="160"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="160"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="105"/>
    </bpmndi:BPMNEdge>
</bpmndi:BPMNPlane>
</bpmndi:BPMNDiagram>
</bpmn2:definitions>

```

The updated BPMN diagram includes the loopback connection from the “Send Reminder” task back to the exclusive gateway. This allows the process to be executed multiple times in case the payment is still not received.

---

**You:**

This is great! Now specify the types of the tasks. Instead of the Abstract Task use specific types (e.g. User Task, Service Task, etc.). Decide for yourself which type fits best.

---

**ChatGPT:**



Certainly! I will update the task types based on their functionality. Here's the revised BPMN XML code with task types specified:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/mo
  <bpmn2:process id="Process_1" isExecutable="true">
    <bpmn2:startEvent id="StartEvent_1">
      <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:userTask id="Task_1" name="Create Invoice">
      <bpmn2:incoming>SequenceFlow_1</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_2</bpmn2:outgoing>
    </bpmn2:userTask>
    <bpmn2:userTask id="Task_2" name="Send Invoice">
      <bpmn2:incoming>SequenceFlow_2</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_3</bpmn2:outgoing>
    </bpmn2:userTask>
    <bpmn2:intermediateCatchEvent id="IntermediateCatchEvent_1">
      <bpmn2:timerEventDefinition/>
      <bpmn2:incoming>SequenceFlow_3</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_4</bpmn2:outgoing>
    </bpmn2:intermediateCatchEvent>
    <bpmn2:exclusiveGateway id="ExclusiveGateway_1">
      <bpmn2:incoming>SequenceFlow_4</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_5</bpmn2:outgoing>
      <bpmn2:outgoing>SequenceFlow_6</bpmn2:outgoing>
    </bpmn2:exclusiveGateway>
    <bpmn2:intermediateCatchEvent id="IntermediateCatchEvent_2">
      <bpmn2:timerEventDefinition/>
      <bpmn2:incoming>SequenceFlow_6</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_7</bpmn2:outgoing>
    </bpmn2:intermediateCatchEvent>
    <bpmn2:userTask id="Task_3" name="Send Reminder">
      <bpmn2:incoming>SequenceFlow_7</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_8</bpmn2:outgoing>
    </bpmn2:userTask>
    <bpmn2:endEvent id="EndEvent_1">
      <bpmn2:incoming>SequenceFlow_5</bpmn2:incoming>
    </bpmn2:endEvent>
    <bpmn2:sequenceFlow id="SequenceFlow_1" sourceRef="StartEvent_1" targetRef="Task_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_2" sourceRef="Task_1" targetRef="Task_2"/>
    <bpmn2:sequenceFlow id="SequenceFlow_3" sourceRef="Task_2" targetRef="IntermediateCatchEvent_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_4" sourceRef="IntermediateCatchEvent_1" targetRef="ExclusiveGateway_1"/>
    <bpmn2:sequenceFlow id="SequenceFlow_5" sourceRef="ExclusiveGateway_1" targetRef="EndEvent_1">
      <bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression">#{approved}</bpmn2:conditionExpression>
    </bpmn2:sequenceFlow>
    <bpmn2:sequenceFlow id="SequenceFlow_6" sourceRef="ExclusiveGateway_1" targetRef="IntermediateCatchEvent_2">
      <bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression">#{!approved}</bpmn2:conditionExpression>
    </bpmn2:sequenceFlow>
    <bpmn2:sequenceFlow id="SequenceFlow_7" sourceRef="IntermediateCatchEvent_2" targetRef="Task_3"/>
    <bpmn2:sequenceFlow id="SequenceFlow_8" sourceRef="Task_3" targetRef="EndEvent_1"/>
  </bpmn2:process>
</bpmn2:definitions>
```

```

    </bpmn2:sequenceFlow>
    <bpmn2:sequenceFlow id="SequenceFlow_7" sourceRef="IntermediateCatchEvent_2" targetRef="Task_3">
    <bpmn2:sequenceFlow id="SequenceFlow_8" sourceRef="Task_3" targetRef="ExclusiveGateway_1"/>
  </bpmn2:process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="Process_1">
      <bpmndi:BPMNShape id="StartEvent_1_di" bpmnElement="StartEvent_1">
        <dc:Bounds x="100" y="100" width="36" height="36"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Task_1_di" bpmnElement="Task_1">
        <dc:Bounds x="200" y="80" width="80" height="80"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Task_2_di" bpmnElement="Task_2">
        <dc:Bounds x="350" y="80" width="80" height="80"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="IntermediateCatchEvent_1_di" bpmnElement="IntermediateCatchEvent_1">
        <dc:Bounds x="510" y="80" width="36" height="36"/>
        <bpmndi:BPMNLabel>
          <dc:Bounds x="523" y="123" width="40" height="14"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_1_di" bpmnElement="ExclusiveGateway_1" isMarkerVisible="true">
        <dc:Bounds x="600" y="80" width="50" height="50"/>
        <bpmndi:BPMNLabel>
          <dc:Bounds x="613" y="123" width="24" height="14"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="IntermediateCatchEvent_2_di" bpmnElement="IntermediateCatchEvent_2">
        <dc:Bounds x="680" y="80" width="36" height="36"/>
        <bpmndi:BPMNLabel>
          <dc:Bounds x="682" y="123" width="32" height="14"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Task_3_di" bpmnElement="Task_3">
        <dc:Bounds x="760" y="80" width="80" height="80"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="EndEvent_1_di" bpmnElement="EndEvent_1">
        <dc:Bounds x="890" y="98" width="36" height="36"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_1_di" bpmnElement="SequenceFlow_1">
        <di:waypoint xsi:type="dc:Point" x="136" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="200" y="118"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_2_di" bpmnElement="SequenceFlow_2">
        <di:waypoint xsi:type="dc:Point" x="280" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="350" y="118"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_3_di" bpmnElement="SequenceFlow_3">

```

```

        <di:waypoint xsi:type="dc:Point" x="430" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="510" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_4_di" bpmnElement="SequenceFlow_4">
        <di:waypoint xsi:type="dc:Point" x="546" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="105"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="130"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_5_di" bpmnElement="SequenceFlow_5">
        <di:waypoint xsi:type="dc:Point" x="650" y="105"/>
        <di:waypoint xsi:type="dc:Point" x="890" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_6_di" bpmnElement="SequenceFlow_6">
        <di:waypoint xsi:type="dc:Point" x="650" y="130"/>
        <di:waypoint xsi:type="dc:Point" x="680" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_7_di" bpmnElement="SequenceFlow_7">
        <di:waypoint xsi:type="dc:Point" x="716" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="760" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_8_di" bpmnElement="SequenceFlow_8">
        <di:waypoint xsi:type="dc:Point" x="840" y="118"/>
        <di:waypoint xsi:type="dc:Point" x="890" y="118"/>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_9_di" bpmnElement="SequenceFlow_9">
        <di:waypoint xsi:type="dc:Point" x="790" y="160"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="160"/>
        <di:waypoint xsi:type="dc:Point" x="600" y="105"/>
    </bpmndi:BPMNEdge>
</bpmndi:BPMNPlane>
</bpmndi:BPMNDiagram>
</bpmn2:definitions>

```

The BPMN diagram now includes specific task types for each task. “Create Invoice,” “Send Invoice,” and “Send Reminder” are represented as User Tasks, while the Start Event, Timer Events, Exclusive Gateway, and End Event remain the same.

---

## Result

- Pros:
  - Created the correct process
    - \* Used the correct BPMN Elements
    - \* Correct arrangement of the various elements (sequence flow)
- Cons:
  - The graphical representation needed some rework
  - Needed very specific instructions (e.g. “Gateway with two outgoing sequence flows”, “Send Reminder should go back to the Gateway”)

## 1.6 Anhang 8 - Reflection Pattern

```
[3]: from IPython.display import Markdown, display
display(Markdown("./images/reflection_pattern.md"))
```

**You:**

For now on I will ask you questions about BPMN. whenever you generate an answer explain the reasoning and assumption behind your answer.

---

**ChatGPT:**

Certainly! I'll do my best to answer your questions about BPMN (Business Process Model and Notation) and provide reasoning and assumptions behind each answer. Please go ahead and ask your first question about BPMN.

---

**You:**

What different task types exist?

---

**ChatGPT:**

In BPMN, there are several types of tasks that represent different aspects of work within a business process. Here are the main task types in BPMN:

These task types provide different options for modeling various aspects of work within a business process, allowing for flexibility and clarity in process representation. Assumption: The task types mentioned here are based on the BPMN 2.0 specification, which is the most widely adopted version of BPMN as of my knowledge cutoff in September 2021. There might be additional task types introduced in newer versions of BPMN that were released after my knowledge cutoff.

---

**You:**

Does the Process Engine of Camunda 7 support Adhoc-Subprocess?

---

**ChatGPT:**

As of my knowledge cutoff in September 2021, Camunda 7 did not support the Adhoc Subprocess directly in its process engine. Camunda BPM is an open-source workflow and decision automation platform based on BPMN 2.0, and it provides extensive support for various BPMN elements. However, Adhoc Subprocess was not one of the officially supported BPMN elements in Camunda 7. An Adhoc Subprocess is a special type of subprocess in BPMN that allows for flexible and dynamic modeling of activities within a process. It enables ad-hoc and on-the-fly creation of tasks during the process execution. While Camunda 7 didn't have native support for Adhoc Subprocess, it still offered flexibility and extensibility through its Java and REST API. Developers could implement custom behavior to achieve similar functionality as an Adhoc Subprocess using these capabilities. It's important to note that software products and versions might have been updated or changed

after my last update in September 2021. Therefore, I recommend checking the official Camunda website or documentation for the most current information regarding its features and support for BPMN elements, including Adhoc Subprocess, in the latest version.

---

## Result

- Referencing the official specification there are following task types:
  1. Service Task
  2. Send Task
  3. Receive Task
  4. User Task
  5. Manual Task
  6. Business Rule Task
  7. Script Task (8. Abstract Task) - Task without any type
- [Camunda 7 does not support Adhoc-Subprocesses](#)

## 1.7 Anhang 9 - Persona Pattern

```
[5]: from IPython.display import Markdown, display
display(Markdown("./images/persona_1.md"))
```

**You:**

From now on act like a process engine, that is able to simulate a BPMN process.

---

**ChatGPT:**

Certainly! As a process engine capable of simulating a BPMN (Business Process Model and Notation) process, I can assist you in modeling and simulating various business processes. Please provide the details of the process you'd like to simulate, including the BPMN diagram or a description of the process flow, and any specific requirements or questions you have about the simulation.

---

**You:**

Flow\_09b83pq Flow\_06m65q8 Flow\_0saefyh Flow\_1d7dtjy Flow\_0d241vt Flow\_1kxe0x0  
Flow\_1qgpibb Flow\_02uaq7 Flow\_09b83pq Flow\_06m65q8 Flow\_0saefyh Flow\_0d241vt  
Flow\_1d7dtjy Flow\_1kxe0x0 Flow\_1qgpibb Flow\_02uaq7

---

**ChatGPT:**

Thank you for providing the BPMN XML for your process. Based on the provided diagram, I will simulate the process for you. Here's a step-by-step execution of the process:

Note: In the provided BPMN diagram, the process is marked as "isExecutable=false," which means it cannot be directly executed. However, we can simulate its flow. Please let me know if you have

any specific questions about the simulation or if you would like to explore any particular aspect of the process.

---

**You:**

How much mass does our sun have?

---

**ChatGPT:**

The mass of our Sun is approximately  $1.989 \times 10^{30}$  kilograms (kg). This mass is equivalent to about 333,000 times the mass of Earth. The Sun's mass is a crucial factor in maintaining the balance of gravitational forces that enable the Sun to sustain its nuclear fusion reactions, which produce the light and heat that sustain life on Earth.

---

```
[4]: from IPython.display import Markdown, display
display(Markdown("./images/persona_2.md"))
```

**You:**

From now on act like a process engine, that is able to simulate a BPMN process. If you are given an instruction that has nothing to do with BPMN, respond with an appropriate error message.

---

**ChatGPT:**

Understood! I'll act as a process engine and simulate a BPMN process for you. Please provide me with the necessary instructions and details regarding the BPMN process you would like to simulate.

---

**You:**

Flow\_09b83pq Flow\_06m65q8 Flow\_0saefyh Flow\_1d7dtjy Flow\_0d241vt Flow\_1kxe0x0  
Flow\_1qgpibb Flow\_02uaq7 Flow\_09b83pq Flow\_06m65q8 Flow\_0saefyh Flow\_0d241vt  
Flow\_1d7dtjy Flow\_1kxe0x0 Flow\_1qgpibb Flow\_02uaq7

---

**ChatGPT:**

Thank you for providing the BPMN XML definition. I will now simulate the BPMN process based on the provided information. The BPMN process consists of the following elements: \* Start Event: "Hunger noted" (id: Event\_1huyz9d)\* User Tasks:

- Exclusive Gateways:
  - End Event: "Hunger satisfied" (id: Event\_1x6w9hd)The sequence flows indicate the flow of the process:
1. Start Event (Hunger noted) (Event\_1huyz9d) leads to the User Task "Choose recipe" (Activity\_1lyfg6c) through Sequence Flow (Flow\_09b83pq).



2. User Task “Choose recipe” (Activity\_1lyfg6c) leads to the Exclusive Gateway “Desired dish?” (Gateway\_0hrykqx) through Sequence Flow (Flow\_06m65q8).
3. Exclusive Gateway “Desired dish?” (Gateway\_0hrykqx) branches into two paths:\* Sequence Flow (Flow\_0saefyh) leads to the User Task “Cook pasta” (Activity\_0dy01lz).\* Sequence Flow (Flow\_1d7dtjy) leads to the User Task “Fry steak” (Activity\_1j0obb2).
4. User Tasks “Cook pasta” (Activity\_0dy01lz) and “Fry steak” (Activity\_1j0obb2) both converge to the (Unnamed) Gateway (Gateway\_0liutt1) through Sequence Flows (Flow\_0d241vt) and (Flow\_1kxe0x0), respectively.
5. The (Unnamed) Gateway (Gateway\_0liutt1) leads to the User Task “Eat meal” (Activity\_0yy16t8) through Sequence Flow (Flow\_1qgpibb).
6. The User Task “Eat meal” (Activity\_0yy16t8) leads to the End Event (Hunger satisfied) (Event\_1x6w9hd) through Sequence Flow (Flow\_02ua0q7).

Please let me know if you have any specific instructions or if you would like to simulate the execution of this BPMN process.

---

**You:**

How much mass does our sun have?

---

**ChatGPT:**

I apologize for the confusion, but as a process engine simulating a BPMN process, I don’t have real-time access to external information such as the current mass of celestial bodies. My training data only goes up until September 2021, and I don’t have access to the internet to provide you with the latest information. To obtain the most accurate and up-to-date mass of the sun, I recommend referring to reputable astronomical sources, scientific literature, or conducting an online search using reliable sources.

---

[ ]: