Decisions for Information or Information for Decisions? Optimizing information gathering in Decision-Intensive Processes

S. Voorberg[a], R. Eshuis[a], W. van Jaarsveld[a], G.J. van Houtum[a]

[a]*Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, The Netherlands*

Abstract

Decision-intensive business processes are performed by decision makers who gather different pieces of information to reach the process objective: a final decision of high quality, for instance, the final price of a quote or the diagnosis of a failure of a hightech machine, as a result of an information-gathering process with minimum costs and efforts. Gathering all possible pieces of information results in high quality decisions, but also yields high costs and efforts. Therefore, decision makers require decision support to determine which information to gather to make the best final decision. This paper introduces an approach that supports a decision maker in the continuous trade-off between the effective acquisition of more information and cost-efficient decision making. The approach uses a well-defined modeling notation for decision-intensive processes, CMMN, and links it to a standard optimization technique, Markov Decision Processes. The approach calculates an optimal information-gathering solution, such that the expected result of the main decision minus the process cost for collecting information is optimized. The approach uses the solution to configure a run-time recommendation tool for the decision maker. The approach is flexible and allows that a decision maker ignores the advice; it then continues to offer recommendations in the subsequent states. We show the feasibility and effectiveness of our approach on a real-world quote process.

Keywords: Decision Intensive Process, Markov Decision Process, Information Acquisition, Decision Support

1. Introduction

Motivation. Nowadays many business processes rely on knowledge workers that gather data from different sources to make informed decisions. Examples are handling complex insurance claims, developing new medicines, or diagnosing failures of high-tech machines. Such business processes require substantial flexibility [1], as new information may lead to new insights, hence new decisions. Business processes that are dependant upon knowledge workers who perform interconnected decision-making tasks to drive critical organizational objectives are called Decision Intensive Processes (DIPs) [1, 2].

The purpose of a DIP is to assist knowledge workers, i.e. decision makers, in making efficient and effective final decisions [1, 2]. Effectiveness refers to weighing the alternatives and making the 'correct' final decision, efficiency to using capital optimally for gathering information. These two factors are dependent, requiring a trade-off to be made. A good decision should require a minimum investment of money and effort and an efficient decision should still have a high quality. Information is the linking pin between these two factors. Collecting information improves the knowledge position of a decision maker and increases the effectiveness of the final decision in a DIP. However, the gathering of information costs money and slows down the process, thus impedes efficiency. Hence, the goal of making an efficient final decision can be translated into the goal of effectively collecting information.

A simple example of a DIP is the process of diagnosing a failure of a hightech machine. To determine which part of the machine is failing, an engineer can collect information on software failures, do remote checks, do test runs, etc. Each piece of gathered information improves the knowledge position of the engineer and therefore decreases the uncertainty of diagnosing a wrong cause of failure. However, the time that a machine is not running is costly to the organization. Hence, an engineer needs to balance, while diagnosing, the costs and gains of gathering additional information in determining the cause of failure.

Problem statement and related work. For many DIPs an abundance of data is available to support human decision making. A decision maker can find themselves entangled in a web of possible actions and decisions and feel the pressure to make cost-effective decisions of good quality. In environments where the amount of information is overwhelming, this causes problems to oversee the situation [3]. Still, intelligence of human decision makers, i.e., tacit knowledge, is indispensable in many situations. Hence, there is a need to improve the support for data-driven human decision making [4, 5].

Next to the work on DIPs [2, 1, 6], there is a stream of related work on knowledge-intensive processes [7, 8, 9], in which decision-making plays a less prominent role, or decision-aware processes [10, 11, 12], which do not focus on decision making by knowledge workers. Some of these works [8, 13] stress the need for incurring data that appears during the process. However, the use of only data is not enough to optimize the actions that are taken in a DIP. The process model can raise additional constraints on actions that have to be incurred in the optimization. In the described streams of literature, this combination of information and process optimization has been recognized [6, 11, 14]. However, no paper has given a clear approach on how to combine the information and the process model into one optimization problem. The main modeling language for decision making in business processes is the DMN notation [15], which declares decision rules organized in decision tables. However, decision rules specify patterns based on information without continuously optimizing the expected benefit of decisions or checking process constraints that affect those decision rules. Hence, there is the need for an approach where human decision makers can control a DIP while receiving recommendations for decisions that optimize for both the process model and the information model.

Aim and contributions. This paper defines an approach that guides decision makers in making decisions in DIPs in the most efficient and effective way. The approach optimizes the cost of acquiring additional information versus the benefit of a final decision with improved quality and thus more profit. There are two sequential phases: a design-time phase and a deployment phase (Figure 1). In the design-time phase the decision maker is assisted in finding an optimal information-gathering solution from process models and probabilistic data. A user-defined DIP model (specified in CMMN [16]), consisting of information gathering tasks and decisions, is translated into a Markov Decision Process (MDP [17]) with constraints. An MDP is well suited since the discrete event structure of an MDP is comparable to that of a DIP. Furthermore, we introduce an information structure: a function that computes the reward for each final decision given the available information. To maximize the expected profit according to the information structure, for every state of the DIP the optimal action (gathering new information or taking a final

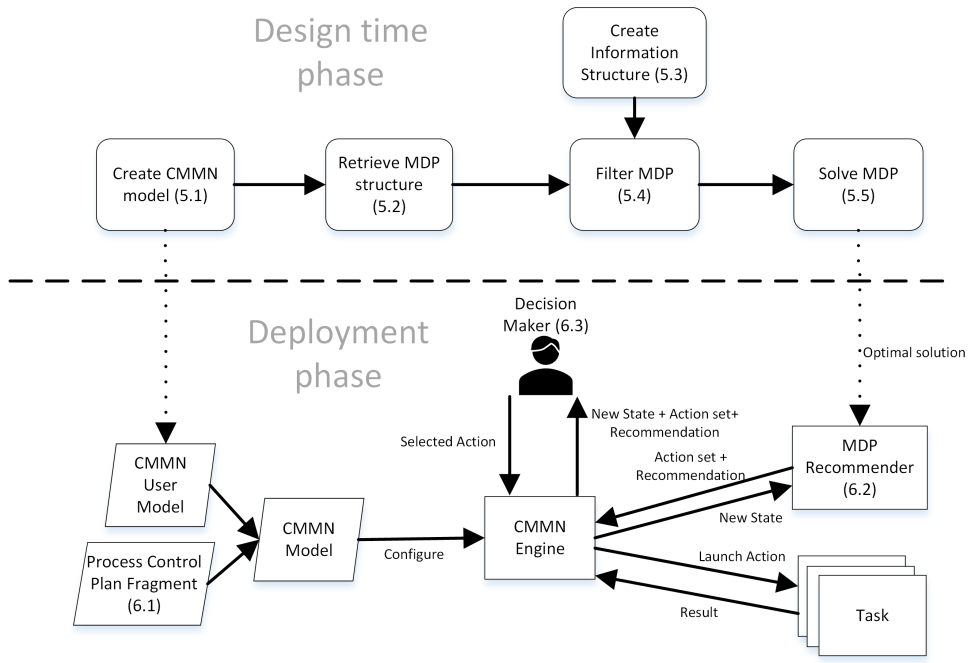


Figure 1: Overview of the approach; each number refers to a section

decision) is computed by the MDP. In the deployment phase, the CMMN user model and a general process control plan fragment combine into a CMMN model that can be deployed with a CMMN engine; the deployed instance is linked with the MDP information-gathering solution in order to provide in every state a recommendation to the decision maker who controls the DIP.

As host notation we use CMMN [16], since it is an OMG standard that is well suited for expressing DIPs [18]. Especially for DIPs, CMMN allows the flexibility of manually chosen tasks, opposed to BPMN. Although some form of flexibility can be introduced in BPMN by using ad-hoc sub-processes, we will use CMMN which was specifically designed for processes in which users can decide to perform or skip an enabled task (manual activation). We extend CMMN with flexible decision support in a way that goes beyond the current state of the art. Using an optimization method that takes into account both process and information we surpass the sole declaration of decision rules in DMN. The flexibility occurs when the decision maker ignores the support and takes a different action. Other than DMN, the MDP has an optimal solution also for any new situation that appears, and the approach supports such ignoring actions. The approach of this paper applies to any process modeling language in which decision makers have to decide per case which information-gathering tasks to perform.

This paper makes three main contributions, as we explain in Section 8. First, we define an approach that optimizes a given DIP. Second, the approach considers the data and process structure when optimizing a DIP. Third, we introduce a flexible recommendation tool that guides a human decision maker, while allowing recommendations to be ignored.

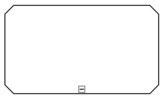
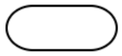
Organization. In Section 2, we introduce CMMN as host notation for DIPs and MDP as optimization method. Subsequently, we introduce a motivating example and its CMMN case in Section 3. In Section 4, we discuss important assumptions and constraints in our approach that allow to use it in a consistent way. Section 5 introduces the design time steps of the approach and Section 6 introduces the deployment phase of the approach. Section 7 shows the use of the approach on the motivating example. Finally, Section 8 and 9 discuss the related work and future directions, respectively.

2. Preliminaries

2.1. Case Modeling Management Notation

The OMG standard Case Management Model and Notation (CMMN) [16] offers a representation for semi-structured business processes in which cases are handled in a flexible way. CMMN mainly focuses on defining, in a declarative way, the flow of a case in a case plan model. The structure of the accompanying case file is assumed to be modeled by a different language, for example using a UML class diagram or an ER model. A CMMN case plan model, consists of *stages*, in which composite work is performed, *tasks*, in which atomic pieces of work are performed, and *milestones*, which are intermediate or final business objectives. These constructs are related via business rules, called sentries. Changes in the case file cause sentries to be triggered and allow tasks or actions to get enabled and started. Manual activation stages/tasks can be performed or skipped if they are enabled. CMMN can be used in combination with BPMN, for modeling structured routine business processes, and DMN, for modeling decisions embedded in processes. Table 1 shows the graphical notation for CMMN.

Table 1: CMMN constructs

Stage		Sentry	◇
		Manual activation stage/task	▷
Task		Required stage/task	!
		Case task	🗀
		Process task	⊃
Milestone		Decision task	▤

In the case of a DIP, the decision maker is constantly balancing between making progress by collecting more information or making a final decision based on the collected information. As we will show in this paper, CMMN is well suited to express this. Especially, CMMN allows stages and tasks to be manually started or skipped (triangle symbol). However, CMMN does not provide guidance to decision makers when to perform these actions. Therefore, we introduce MDPs to consider the future effect of decisions on both the case file and the case plan.

2.2. Markov Decision Process

The approach that we define relies on Markov Decision Processes (MDPs) as mathematical optimization model [17]. A Markov Decision Process is a technique that optimizes sequential decision making under uncertainty. MDPs optimize the expected reward over a series of future actions. Thus, MDPs can be used to foresee what the effects will be of taking actions now and in the future. Using backward induction (sometimes also called dynamic programming) as solution method for finite horizon problems[17, p. 92], one can compute the optimal action for every state the process can be in. We specifically discuss this method for our approach in Section 5.5. Every *action* taken by the agent, for instance a decision maker, brings the process in a new *state* and returns a certain *reward* (see Figure 2). The subsequent state after taking an action is defined by the transition and more specifically by the *transition probability.* Hence, the next state is uncertain and depends on a probability distribution.

2.3. CMMN and MDP

In a CMMN case, the *state* is defined by both the status of the case file and the case plan. One takes an *action* by deciding which subsequent task to perform, either informa-

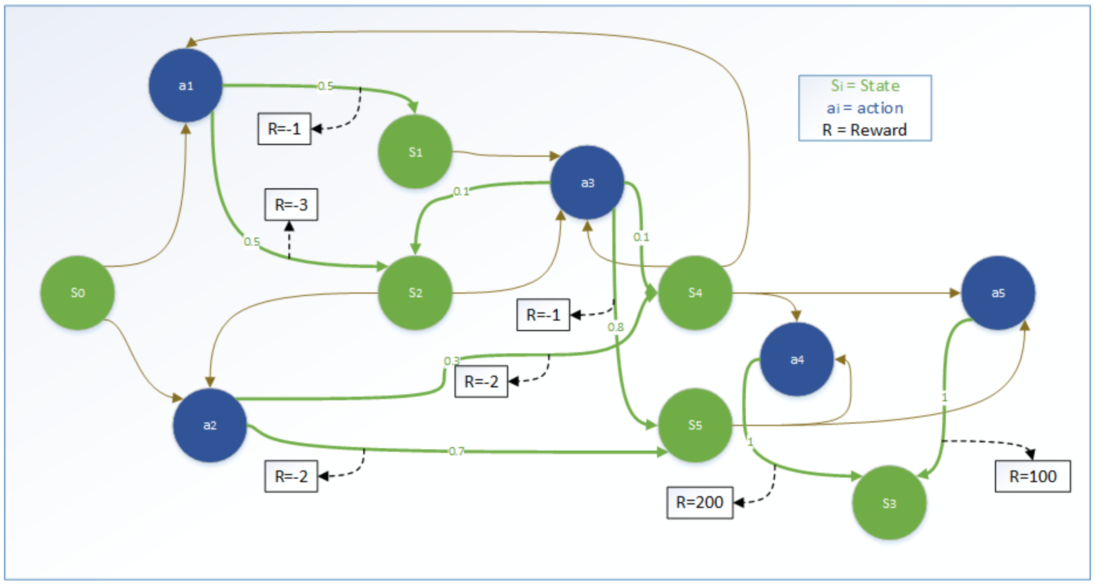


Figure 2: Markov Decision Process

tion acquiring or making a final decision. Every action causes a change of process(case file) or information(case plan) status. The uncertainty appears in the unknown outcome of information acquiring tasks based on *probability*. By putting cost (negative reward) on performing more tasks and returning a reward based on how good the final decision is that was made we include the decision variables in the MDP and force the MDP to find the best action such that the expected profit is maximized. Section 5.4 gives a detailed description of how we translate a CMMN case into an MDP model.

3. Motivating example

This section introduces a real-world case scenario that describes the need for decision support in information gathering. We use this example case in Section 7 to show the feasibility of our approach and its benefits.

Koffer services is a company that performs maintenance on aircraft components. Airliners close service contracts with Koffer for a specific component over multiple years. The market of aircraft component maintenance is competitive and requires sharp offers from Koffer to bring in contracts. To know the profit margins that are available for a specific component, information gathering is crucial. Koffer engineers need support on which information is most important and still valuable enough to collect such that they

Table 2: Information acquiring tasks and attributes

Task	Data Attribute	Reference
Obtain net repair cost	Net expected repair cost	A
Obtain Expected number of repairs	Expected number of repairs per year	B
Check customer credibility	Customer credibility	C
Check market prices	Current mean market price	D
In-house capability check	In-house capability	E
Determine PMA/DER	PMA/DER possibilities	F
Determine over and aboves	Over and above options	G
Fix contract length	Contract period	H
Obtain transport cost	Transportation cost	I

win attractive contracts and avoid unattractive contracts.

The process of these service contracts works as follows. Koffer receives a Request for Quotation (RFQ) from a potential client, who may also send this RFQ to competitors of Koffer. The RFQ refers to an airplane component, for which the client requests a proposed contract that explains Koffer's terms for the service contract, in this scenario a specific repair price for the serviced component. After performing an internal DIP in which extra information can be gathered, Koffer sends the terms of their contract. Koffer either earns the contract or the client chooses the offer from a competitor. Koffer is looking for an approach to support the decision makers by informing them about the value of gathering more information and the value of current options for the conditions of the contract.

For this paper, we focus on the internal DIP. At the moment that the RFQ arrives, Koffer has no exact information concerning the properties of the component that should be quoted. Koffer can gather information on this component and the terms and conditions of this RFQ through the data attributes listed in Table 2. Note that Parts Manufacturer Approval (PMA) and Designated Engineering Representative (DER) are methods that suppress the cost of a component by using cheaper alternatives. Over and above parts are excluded from the contract and can be charged separately.

Koffer aims to optimize the process of gathering information through these attributes and hence to increase their certainty about how much profit they can make on this component. This increased certainty improves the effectiveness of their quote price. At the same time, Koffer loses efficiency because of valuable time that is lost while gathering information. Other, possibly more profitable, RFQs might not be handled as long as this

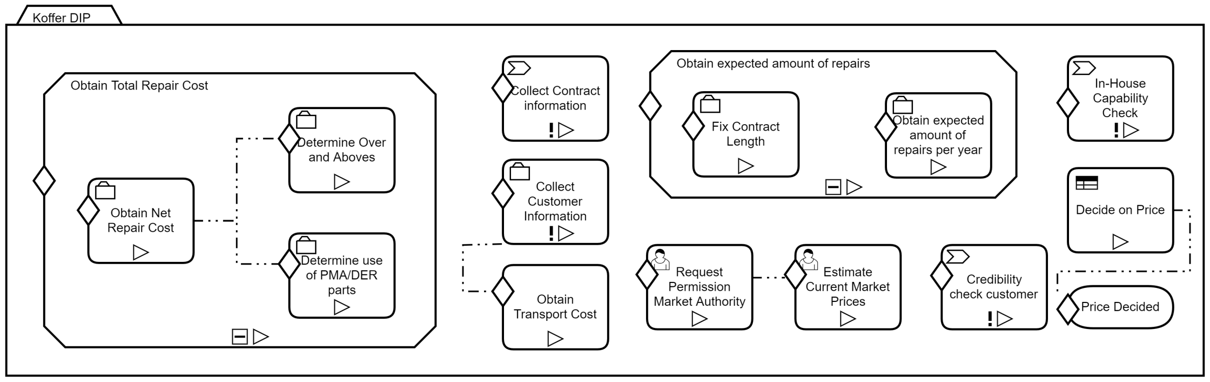

Koffer DIP
Obtain Total Repair Cost
Obtain Net Repair Cost
Determine Over and Aboves
Determine use of PMA/DER parts
Collect Contract information
Collect Customer Information
Obtain Transport Cost
Obtain expected amount of repairs
Fix Contract Length
Obtain expected amount of repairs per year
Request Permission Market Authority
Estimate Current Market Prices
Credibility check customer
In-House Capability Check
Decide on Price
Price Decided

Figure 3: CMMN plan for Koffer

RFQ is still the main focus. Hence, they need a DIP to optimally make the trade-off between time and information. Figure 3 shows the CMMN plan for the internal DIP. This figure contains more tasks than the information-acquiring tasks of Table 2, since the entire DIP includes other tasks as well.

4. Optimizable DIPs

In this paper we focus on DIPs with a specific structure that allows linking them to Markov Decision Processes (MDPs) for optimization. Hence, we call them *optimizable DIPs*. In Section 4.1, we set out what is the specific constraint for optimizable DIPs. In Section 4.2 we discuss how CMMN is used to model these *optimizable DIPs*.

4.1. Stable information

In a DIP, information is gathered to reach the end goal in an optimal way. In this paper we use data attributes to model the information needs of a DIP. In the class of optimizable DIPs, we assume acquired information to be accurate and *stable*, i.e., values acquired for data attributes are assumed to be correct and are not changed once acquired. The assumption that data attributes are stable allows us to use them to decrease uncertainty on the received reward and on which final decision is optimal. Attribute values are retrieved by performing tasks. If a DIP only has stable data attributes, tasks are monotonic: if a task writes an attribute it is performed at most once during execution.

Before executing the DIP, the value of a data attribute X can be estimated using a probability distribution, $P(X = x) \in [0, 1]$, where x is a value for X. The probability distribution could be derived for example by analyzing historical data or by interviewing domain experts. For each execution of a DIP with n data attributes, the set with data attributes is denoted by $D = \{X_1, \ldots, X_n\}$. For each data attribute $X_i \in D$ we assume a domain $dom(X_i)$ of possible values for X_i. The space of data attribute values is denoted by $\mathcal{D} = dom(X_1) \times \ldots \times dom(X_n)$. Using a probability distribution for each data attribute allows us to know the set of possible values and also the relative appearance of every possible value. The probability distributions enable to estimate the different outcomes (variability) of a specific decision. This is what an MDP uses to optimize decisions. Also, one can compare different future scenarios where more information becomes available, which would improve the certainty of decisions by decreasing the variability of possible outcomes. Hence, knowing these distributions enables to determine if data attributes can provide extra information for a final decision.

4.2. Plan Constraints in CMMN

Although the main trade-off for an optimizable DIP lies in the data attribute value gathering (information perspective), we are still considering this problem from the perspective of a business process. As such, many different constraints might prevent a decision maker from taking the most direct path in collecting certain attribute values. For example, there might be subtasks that must be performed first before one can continue with other tasks. These constraints are modeled in the CMMN process model, for instance required tasks, precedence constraints between tasks, and stages that contain tasks. Furthermore, we allow for tasks in the CMMN model that are part of the procedure but actually do not affect the decision making part of the DIP. Our approach, in particular the MDP, deals with all these additional constraints. This means that we do not only optimize from an information-perspective (case file), but also from a process-perspective (case plan).

While we allow for plan constraints, we also make some simplifying assumptions in this paper. These assumptions are necessary to filter out a tractable MDP model (cf. Section 5.4). First of all, our algorithm does not support milestones. Therefore, any milestones that are currently modeled in a CMMN model will be excluded, except for the milestone that models the achievement of the main goal of the whole case. In section 9, we discuss how this assumption can be relaxed. Furthermore, we assume that tasks are instantaneous, so upon invocation they return immediately an outcome (data attribute value). Non-instantaneous tasks lead to an explosion of the state space of an MDP.

5. Dynamic decision support for optimizable DIPs: Design time

In this section, we define the design-time steps of our approach to optimize the process of information gathering in Decision Intensive Processes. The upper part of Figure 1 shows a flowchart of this part of the approach, consisting of five steps. These steps translate the DIP into an optimizable MDP model and solve it. Section 6 explains how we use the solved MDP to configure a recommender that offers decision support during deployment.

5.1. Create CMMN model

The first step is modeling a DIP with the CMMN notation. A CMMN process model is called a case plan model, which consists of a top-level stage that contains stages, tasks and milestones. CMMN calls these elements plan items, and distinguishes between the definition and the use of plan items. However, we assume that each definition is used only once in a case plan model, hence we equate plan item definitions and plan items. Note that this assumption follows directly from the stable information assumption in Section 4.1, causing tasks to be monotone. Therefore, any defined plan item is only used once.

We allow the use of all possible CMMN task types including human tasks, case tasks and process tasks. For example, in Figure 3, tasks such as Define Over and Aboves or Collect Customer Information are case tasks, while Estimate Current Market Prices is a human task. However, since only the output data of a task is needed to make decisions, the specific CMMN task type can be ignored.

CMMN is suited for dynamic data where stages or tasks can be visited multiple times to update data items. However, because of the stable information assumption (Section 4), the stages and tasks in our DIP are not reactivated after their completion. Tasks for which the decorator (!) is used are *required*: these model elements need to be finished before the parent stage can be completed.

This paper focuses especially on tasks and stages that need to be activated manually by the user, i.e., decision maker. The approach helps the user to decide, while enacting the CMMN process model, whether and when to activate such a task or stage. Note that a manually activated task or stage can have an entry criterion, that needs to evaluate to true before the manual activation decision can be made. The approach allows that for some task or stage automatic activation is used. However, to keep the interpretation of the model clear, we have not implemented this in our example case. We only consider entry criteria having sentries with **on** statements. To keep the model representation interpretable, sentries having **if** statements are not considered. However, MDPs do allow for the use of conditions on the data through **if** statements.

Data in a CMMN case is stored in a case file, which contains case file items. Therefore, the *case file* will denote the set of data attributes that can be acquired. To simplify the presentation, we allow a task to be connected to at most one case file item. However, this constraint can be relaxed by taking the joint probability distribution of the case file items for that specific task. To see an example of a CMMN model that is according to the optimizable DIP description, we refer to Figure 3.

5.2. Retrieve MDP Structure

To be able to continue our approach, we require that all tasks are assigned to one of four classes: three classes of tasks and a class of stages. Furthermore, we require the presence of a single milestone that terminates the process. Figure 4 is a meta model in UML notation that shows how these different classes are associated with each other and relate to the CMMN classes. The class diagram also contains the class of case file items with their known probability distribution.

Tasks. Consider the complete set of tasks T also introduced in Figure 4. For every task $t \in T$ we assign an associated cost c_t, representing the cost and effort op performing the task, and a boolean variable that keeps track of whether the task has been performed.

Each DIP works towards a *final decision.* We model this final decision with a decision making task. We assume a DIP has one such decision making task, that is not followed by other tasks. However, in Section 9, we come back to this assumption and give suggestions on how the approach could be extended with multiple (sub)decisions in one DIP.

Definition 5.1 (Decision making tasks/Milestones). *A decision making task uses acquired information from the case file to choose a decision f from a set of decision alternatives $\mathcal{F}$. The decision finishes the DIP by achieving its final milestone.*

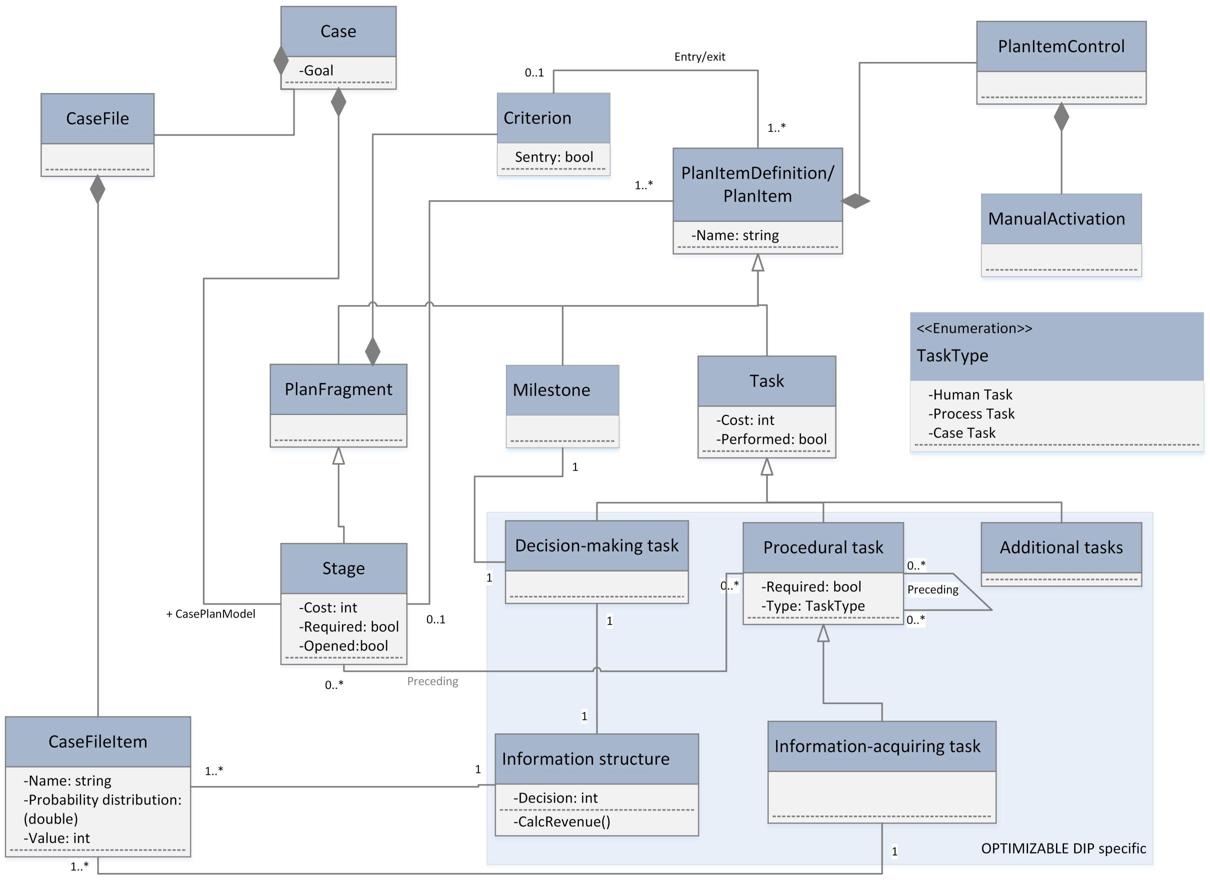


Figure 4: CMMN meta model [19, 16] extended with support for DIPs (in gray)

We define the decision making task $d \in T$. In accordance with the UML class diagram in Fig. 4, a decision making task d is associated with an information structure (Section 5.3) and a single milestone and inherits two attributes from the task class. In the CMMN models, we interpret decision (DMN) tasks to be decision making tasks. Moreover, a decision making task is always followed by a milestone. Normally an underlying DMN model would define the decision tree for a decision task, but in our approach we define an MDP that governs the decision through a look up table that is introduced in Section 6.

Definition 5.2 (Procedural tasks). *A procedural task is a task that is performed to progress towards the final decision of the DIP. It has a direct effect by being either required(1) or preceding(2) or part of a stage(3) or information acquiring(4).*

We define the set of procedural tasks $P \subset T$. In the example we see the task of Requesting Permission Market Authority preceeds task Estimate Current Market Price. Hence, the permission is a preceding task(2) in this example, which makes it a procedural task. We define procedural tasks that collect one of the attribute values of unknown attributes as information acquiring tasks, $I \subseteq P$.

Definition 5.3 (Information Acquiring tasks). *An information acquiring task is a procedural task that updates one or more attributes in the case file when it finishes.*

An example of an information acquiring task is the In-House Capability check in Figure 3. Every information acquiring task has a known association with one case file item and these tasks also inherit the process-related class attributes for procedural tasks.

Besides decision making and procedural tasks, a DIP can have additional tasks. Although they may have a certain cost associated, they do not directly affect the decision in the process and thus are negligible for the MDP and the approach. Therefore, any additional tasks are discarded. Discarding those tasks for the decision-making model does not mean those tasks are removed from the business process.

Definition 5.4 (Additional tasks). *An additional task is a task that is in the CMMN model and is neither a decision making nor a procedural task.*

Stages. Stages $v \in V$ create the hierarchy inside a plan model. We use stages to group certain clusters of tasks. The stage is an identifier of a cluster of tasks. Similar to a task if a stage is in the *preceding* set of another stage or task, it prevents the task/stage inside the cluster from starting if it has not been opened yet. Due to their structure, for the MDP model we can interpret stages as a special class of procedural tasks.

Labeling tasks. Based on the UML class diagram and using the constraints for the optimizable DIP, we can recognize and retrieve all the above defined classes of tasks and stages automatically. The decision making task is attached to the only milestone in the plan. Furthermore, all tasks that do not fulfill any constraints to other tasks and are not associated with an attribute are discarded as additional tasks. The remaining tasks are procedural. Any of these procedural tasks that have an association with an attribute are classified as an information acquiring task. Similar to procedural and additional tasks, stages can be filtered naturally from the starting CMMN model and do not require additional verification by the plan builder. Note that the Koffer case according to Figure 3 does not contain any additional tasks.

The introduced task types resemble activity types that have been proposed for making decisions in processes [14]: Decision making tasks resemble decision activities, information acquiring tasks resemble administrative activities and additional tasks resemble operational activities. However, procedural tasks with procedural constraints (required, precedence) have no counterpart in [14].

5.3. Information Structure

A DIP is performed to achieve a specific functional goal that relates to a decision to obtain the best outcome. This *final decision* is the crucial step that affects the outcome of the whole process. For example, in the Koffer case in Section 3, the decision on the quoted price is the *final decision*. While each DIP has a specific functional goal, i.e., an outcome, a final decision should be both efficient and effective, which are non-functional goals. Since both non-functional goals are in conflict, this requires a trade-off between them. Note that the literature on DIPs or related Knowledge-Intensive Processes (Section 8) considers only functional goals and not the trade-off between non-functional and functional goals to reach an optimal decision.

To optimize for decision efficiency, one could set constraints on, for example, the total effort spent on the process. However, these predefined constraints could affect the search for optimality in a negative way. For example, if a constraint permits to only collect a single piece of information, one cannot react in a flexible way to the situation. Hence, it is preferable to be able to compare efficiency and effectiveness in a direct way. Therefore, we propose to define a cost factor for the time spent on different tasks. This way, we can also distinguish better between tasks that are cheap (simple task) versus expensive (difficult task). Together with a reward function based on the process outcome, it is then possible to make this direct comparison. We next introduce mathematical notation for specifying how to optimize the process. We require a function that connects the effect of acquired data on final decisions and the cost of collecting that data to a reward. More precisely, the input of this function includes a final decision f from a set of alternatives $\mathcal{F}$, which must be chosen to terminate the process, and all the data attribute values $\mathcal{D}$, with a known value. We call this function the information structure. In earlier work [20], we already introduced a first definition for this information structure. We extend that definition here, such that it also optimizes over the decision variable that directly determines the outcome of the process.

Definition 5.5 (Information Structure). *Given a set of data attributes $D = \{X_1, \ldots, X_n\}$ with possible values $\mathcal{D} = dom(X_1) \times \ldots \times dom(X_n)$ and a set of final decision alternatives $\mathcal{F}$, the **information structure** is a function $Y : \mathcal{D} \times \mathcal{F} \to \mathbb{R}$, which gives the reward given all information and the chosen alternative. We denote by $Y(X_1, \ldots, X_n, f)$ a function that gives the reward, where the data attributes are random variables.*

The method CalcRevenue() in Figure 4 is the actual information structure Y that based on the attribute values and the set of decision alternatives calculates the optimal decision for the decision making task. An information structure can be derived through a combination of data analysis and domain knowledge. If there is a set of historical cases of this process, where preferably all data attributes, the decision and the outcomes are known, one could use techniques such as regression to estimate the effect of parameters on the outcome, while taking into account the decision options, as we discuss in [20].

We give a small example to show what the information acquisition can do with the outcome of the process. Suppose we have two data attributes, X_1, X_2, for which the values are unknown: $P(X_1 = 0) = P(X_1 = 3) = 0.5$, $P(X_2 = 1) = P(X_2 = 4) = 0.5$. The two decision alternatives that can be taken are 0 (decline) or 1 (accept): $\mathcal{F} = \{0, 1\}$. The information structure is attribute 2 minus attribute 1 multiplied by the final decision: $Y(X_1, X_2, f) = (X_2 - X_1)f$. The acquisition cost is 0.5 per attribute: $c_1, c_2 = 0.5$. Suppose the decision maker first wants to explore the acquisition of attribute 2. If $X_2 = 4$, we know the optimal final decision, independent of the value of X_1: $f = 1$. Then, knowing that $P(X_1 = 0) = P(X_1 = 3) = 0.5$, there is an expected profit of $\mathbf{E}[Y] = (0.5 \cdot (4 - 3) + 0.5 \cdot (4 - 0)) \cdot 1 = 2.5$. However, when $X_2 = 1$, the optimal decision is: $f = 0$, because $\mathbf{E}[Y] = (0.5 \cdot (1 - 3) + 0.5 \cdot (1 - 0)) \cdot f = -0.5 \cdot f$. Hence, the optimal profit $\mathbf{E}[Y] = 0$ for $f = 0$. Similarly, we can derive that without any information the optimal decision is $f = 1$, with an expected profit of 0.5. To interpret the value of attribute 2, we compare the profits when knowing and not knowing the value of X_2. For attribute 2: we have $(0.5 \cdot 2.5 + 0.5 \cdot 0) - 0.5 = 0.75$ as profit increase, which exceeds the costs c_2. So, it does pay off to acquire the information for attribute 2 right now.

Thus, next to efficiency, the effectiveness of a decision can also be compared using the introduced information structure. In conclusion, executing a DIP in the most efficient and effective way can be viewed as a problem of profit maximization. Collecting less information can decrease the cost of the process. However, the shortage of information creates uncertainty about the outcome of the final decision. By collecting more information this uncertainty can be decreased. The information structure is the solution to consider both factors by making a trade-off between extra cost versus improved revenue.

5.4. Filter MDP model

Given the information structure and the assignment of tasks to the specific classes, one can now filter out an MDP model. In this paper, we focus on the approach and not on the specific mathematical details of filtering out an MDP.

For the CMMN model, the MDP state $\mathbf{s}$ of the DIP is a vector that contains for each variable its value. A variable is either a case file item (data attribute) or a status attribute of a task or stage (case plan item). During the execution of the DIP each stable data attribute X_i, $1 \leq i \leq n$ begins with an unknown value, expressed with symbol $\perp$. The actual value x_i for X_i can be retrieved. Together the data attribute variable i has a state $s_i \in dom(X_i) \cup \{\perp\}$. A vector with data attribute values, both known and unknown, is interpreted as an information state, denoted with $s = (s_1, \ldots, s_n)$. Furthermore, the total space of possible states is denoted by $\mathcal{S} = (dom(X_1) \cup \{\perp\}) \times \ldots \times (dom(X_n) \cup \{\perp\})$. For each procedural task or stage $y \in P \cup V$, we use a binary status attribute σ_y that has value 1 if and only if the task or stage y has finished: $\sigma = (\sigma_a, \sigma_b, \ldots, \sigma_{|P \cup V|})$. We call σ the plan state. The total state of the process is $\mathbf{s} = (\sigma, s)$.

For the information acquiring tasks to have the performed variable set to true, we require the associated case file items to have a known value. Based on this, the decision maker can choose from two different sets of actions. First, the decision maker can choose to perform a procedural task and possibly acquire an uncertain attribute, i.e., $\mathbf{a} = p \in P$, which changes the state because the task performed attribute is set to true. Notice that although we do not include the exact value of the acquired attributes information in the plan state, this value does have a crucial effect on the outcome of the process as part of the information state s. Second, the decision maker can decide to make a final decision, $\mathbf{a} = f \in \mathcal{F}$. In that case we end up in a terminating state because the DIP is finished. Notice that the opening of stages and required or preceding tasks puts some extra constraints on the actions allowed.

Let $preceding_{\mathbf{a}}$ be the set of procedural tasks or stages that must have been performed before task or stage $\mathbf{a}$ is allowed. In Figure 3 this is denoted by a dotted line that connects the two tasks. Finally, let $required_{\mathbf{b}}$ be the boolean variable that denotes if task $\mathbf{b}$ must be performed before the process can be finished. Then, for the decision maker to take action $\mathbf{a}$, we require the following conditions to hold:

$$\forall \mathbf{b} \in preceding_{\mathbf{a}} : \sigma_{\mathbf{b}} = 1 \qquad \text{(Preceding tasks or stages)}$$

$$\text{if } \mathbf{a} \in \mathcal{F} : \forall \mathbf{b} \in P \cup V, \text{if } required_{\mathbf{b}} = 1, \text{ then } \sigma_{\mathbf{b}} = 1 \qquad \text{(Required tasks or stages)}$$

As long as the decision maker takes information acquiring actions, we incur a cost for the specific task that is performed. The moment the decision maker decides to make a final decision, we retrieve an expected reward based on the currently known attribute values and the decision taken. Hence,

$$R(s, \mathbf{a}) = \begin{cases} -c_{\mathbf{a}} & \text{if } \mathbf{a} \in P \\ \mathbf{E}[Y(s, \mathbf{a})] & \text{if } \mathbf{a} \in \mathcal{F} \end{cases}$$

Note that we can take the expectation over Y as it has been defined also for the set of random variables D. For a state s where some information is already revealed, the expected reward of taking final action a can be expressed as a conditional expectation as follows $\mathbf{E}[Y(s, \mathbf{a})] = \mathbf{E}[Y(X_1, \dots, X_n, \mathbf{a}) | \forall i : x_i = s_i \text{ if } s_i \neq \perp]$. This takes into account the known attribute values, while using the distribution for the remaining attributes.

Transitions happen with a certain probability that can be derived from the probability distributions that we establish when classifying the information-gathering tasks. We define two transitions: state transitions (case plan) and information transitions (case file). The total transition of an action $\mathbf{a}$ in the DIP constitutes of the product of both transitions. Looking back at the defined classes of tasks, only in information acquiring tasks we have an information transition. In the other classes the information transition probability will always be 1.

For state transitions we have the following:

$$P^p_{\mathbf{a}}(\sigma, \sigma') = \begin{cases} 1 & \text{if } \mathbf{a} \in P \cup V \text{ and } \sigma_{\mathbf{a}} = 0 \text{ and } \sigma'_{\mathbf{a}} = 1 \text{ and } \forall \mathbf{p} \in P \setminus \{\mathbf{a}\} : \sigma_{\mathbf{p}} = \sigma'_{\mathbf{p}} \qquad \text{(Procedural task)} \\ 1 & \text{if } \mathbf{a} \in \mathcal{F} \text{ and } \forall \mathbf{b} \in P \cup V : \sigma_{\mathbf{b}} = \sigma'_{\mathbf{b}} \qquad \text{(Decision-making task)} \\ 0 & \text{otherwise} \end{cases}$$

For information transitions we define the following, where $\leftrightarrow$ means an information-acquiring task and case file item are associated:

$$P^c_{\mathbf{a}}(s, s') = \begin{cases} P(X_i = x_i) & \text{if } \mathbf{a} \in I : \text{ if } i \leftrightarrow \mathbf{a} : s_i = \perp \text{ and } s'_i = x_i \text{ and for } j \not\leftrightarrow \mathbf{a} : s_j = s'_j \\ 1 & \text{if } \mathbf{a} \in \mathcal{F} \cup V \cup P \setminus I \text{ and } 1 \leq i \leq n : s_i = s'_i \\ 0 & \text{otherwise} \end{cases}$$

Hence, the total transition probability is then:

$$P_{\mathbf{a}}(\mathbf{s}, \mathbf{s'}) = P^p_{\mathbf{a}}(\sigma, \sigma') \cdot P^c_{\mathbf{a}}(s, s')$$

By collecting extra information, it is possible to reduce the uncertainty on the final decision outcome. Hence, there is a higher certainty that one actually will earn what is expected. However, to arrive at that point, efficiency is lost while acquiring additional information. The trade-off between collecting more information and maximizing the expected reward based on the final decision is the key parameter that this MDP tries to maximize by scaling those rewards to their probability of happening. One needs to know the probability distributions of the individual attributes because they allow to compare the change in profit between the attributes based on their probabilities.

5.5. Solve MDP

To solve a finite horizon MDP, we can use the technique of backward induction. However, this is only the case in scenarios where the state space still is tractable in computer memory. Backward induction starts from all the possible scenarios that the process could end up in and their reward. By thinking backwards what would have been the best action to maximize reward if still one variable would be uncertain, we set one step back in time. This becomes a recursive procedure all the way back until all attributes are uncertain. During this backward induction, every possible state is visited. This also means that for every possible state, we have determined what is the optimal action to take. To know in detail how backward induction works for a MDP based on an optimizable DIP, we refer to our earlier work [20], which builds on existing MDP solution techniques [17]. Note that the state space of an MDP grows exponentially. This means that we can solve problems up to the size of the example case. Hence, scalability is an important future extension, which we discuss in the conclusion. An MDP always optimizes for the expected profit. However, by retrieving information one also decreases the uncertainty of the outcome. Because in this paper we dot not put any conditions on the information structure, this function can become very complicated with many dependencies between attribute values. Therefore, we cannot give any further indication on the variance reduction for different information-acquiring decisions.

6. Dynamic decision support for optimizable DIPs: Deployment

The outcome of our approach for the design-time phase is a solved MDP that offers an advice for every possible state in which the DIP can be. We now define how to use these MDP advices during deployment. The lower part of Figure 1 shows how a decision maker controls a CMMN engine while interacting with the MDP recommender. For this purpose, we introduce a plan fragment to control the execution of the user-defined CMMN process model. A plan fragment is a reusable part of a case plan model and often contains multiple plan items. Inserting the plan fragment into the CMMN user model results in an integrated CMMN model. The CMMN engine is configured with this integrated CMMN model to let the decision maker control the DIP and launch tasks. After every task completion, the result is incurred in the new state that is sent to the MDP recommender to obtain a new action set and recommendation. Next, both the new state and the action set and recommendation are sent to the decision maker, who selects one of the actions.

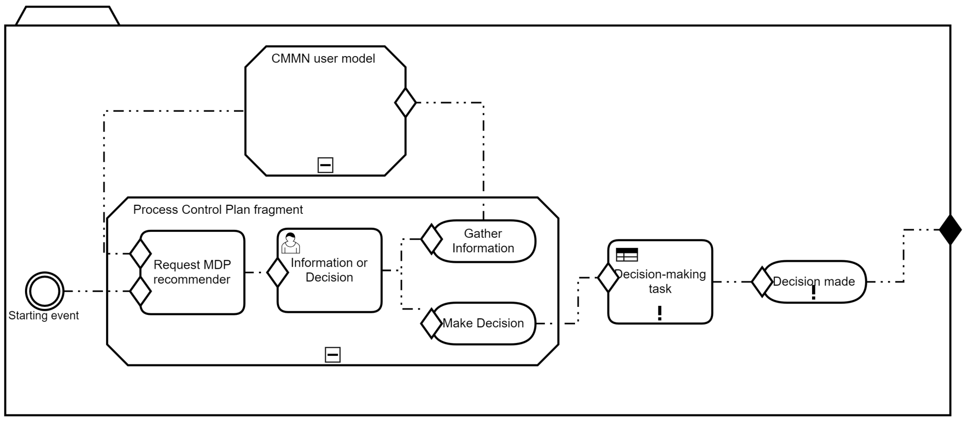


Figure 5: Process Control Plan Fragment for user CMMN model

6.1. Process Control Plan fragment

Figure 5 shows the process control plan fragment that fits the constraints of an optimizable DIP and its context, the CMMN user model. The figure shows the *final decision* point as defined by Decision-making task; in the example in Figure 3 this would be task Decide on Price. Furthermore, when deciding to gather more information the corresponding task from the CMMN user model is performed (cf. Figure 3). The actual need for a plan fragment and the extra tasks is because of the effectiveness versus efficiency trade-off. This specific trade-off only appears after the DIP has been modeled according to our approach. Hence, when performing the DIP the decision maker needs a recurring process phase where he actually makes the trade-off supported by the MDP recommender.

The plan fragment consists of a task Request MDP recommender which results in a recommendation and subsequently the task Information or Decision where the decision maker can consult the MDP recommendation and decide on the next step. The two milestones capture the decision maker's decision and control the next phase of the integrated plan. By integrating the plan fragment with the user model, an integrated CMMN model is obtained that configures a CMMN engine supporting the DIP.

6.2. Decision making support

The outcome of solving the MDP can be seen as a multi-dimensional matrix that functions as a look-up table for decision makers. During the Information or Decision task, the decision maker makes a decision, given the current state and MDP support, i.e., an optimal action and the corresponding expected profit for that action. Also other possible actions and their expected profit are shown to the decision maker. Section 7.3 will give an example of how the information could be shown to the decision maker.

The MDP support consists of three parts. Those parts are components of the recommendation handed to the decision maker in the Information or Decision task in Figure 5. First, the decision maker is given a list of allowed actions according to the current state of the CMMN user model. Second, the MDP advises the optimal next action, either a procedural task, e.g., which task from the CMMN user model in Figure 5, or a final decision. Note that for specific scenarios, we first might need to perform a 'non-acquiring' procedural task before we can actually perform the information acquiring task that gives us the required information. In that case the MDP will first advice the non-acquiring task. Third, the MDP shows the expected profit for all possible actions. This comparison was discussed in section 5.3. As long as additional information still offers more value in expectation, the recommendation will be to continue information gathering.

6.3. Decision maker

As shown in Figure 1, the decision maker is the responsible process controller, i.e., the outgoing arcs of the Information or Decision task are based on a decision from a decision maker who is assisted by the MDP recommender. The inserted plan fragment then allows the CMMN engine to control the process being enacted. When the task is completed, the CMMN engine calls the recommender for a new state-specific advice, even when the previous advice was not followed by the decision maker. This new advice is then shown to the decision maker. This loop continues until the decision maker takes a final decision, after which the CMMN engine will terminate the process. However, tacit knowledge of decision makers cannot be captured in a mathematical model. Our approach therefore allows a decision maker to ignore recommendations and acquire information that is not most profitable according to the MDP. Still, the MDP offers advice in any resulting state.

7. Evaluation/Case study

Based on the example DIP case introduced in Section 3, we next show a proof of concept for our approach and validate its feasibility. Furthermore, the outcome of the

MDP for the example case is compared to the current decision method within Koffer. On `https://is.ieis.tue.nl/staff/heshuis/OptimizingInformationGathering`, a second case is made publicly available, where we focus on optimizing failure diagnosing and dispatching of maintenance engineers.

7.1. Validation case

The Koffer case that was introduced in Section 3 is based on a real-world case, but the company is confidential. The validity of this model was discussed with actual decision makers for this specific kind of processes at the company. Also the data set-up, though simplified to four possible values per attribute to maintain tractability, has been retrieved from this company. Finally, the decision tree that we will use as benchmark was developed in cooperation with the same decision makers.

7.2. Design-time approach

Create CMMN model. For the first stage of our approach, we require a CMMN model of the optimizable DIP according to the constraints defined in Section 5. For the example DIP we already introduced the CMMN model in Section 3, which does satisfy the required optimizable DIP set-up according to the constraints introduced in Section 4.

Retrieve MDP structure. The next step that is necessary to create an MDP is to assign all necessary tasks in the CMMN model to one of three subsets. How to assign tasks to these sets follows directly from the CMMN model in Figure 3. Table 3 shows the stages and procedural tasks and their associations with other tasks including also the information acquiring tasks and the associated case file items. Table 4 defines the case file containing all case file items. There are no additional tasks.

To retrieve the MDP structure of the example DIP we need an information structure for the decision-making task. We define the following information structure, based on the case file items introduced in Table 4 and the Koffer case,

$$
\begin{aligned}
Y(s,f) &= E \cdot \overbrace{B \cdot H}^{\text{Expected number of repairs}} \cdot \overbrace{P(\mathcal{D} \geq f)}^{\text{Quote accepted}} \cdot \overbrace{[f - A(1-0.2G) - F - I]}^{\text{Expected Revenue}} \quad \forall f \in \mathbb{R} \\
Y(s,\text{No bid}) &= 0
\end{aligned}
$$

	Stages	Cost	Preceding	Required	
1.	Koffer DIP	0	()	0	
α.	Obtain total repair cost	20	()	0	
β.	Obtain expected amount of repairs	0	()	1	
	Procedural tasks	Cost	Preceding	Required	
2.	Request permission market authority	20	()	0	
3.	Collect contract information	10	()	1	
4.	Collect customer information	30	()	1	
	Information acquiring tasks	Cost	Preceding	Required	Case File Item
5.	Obtain net repair cost	10	(α)	0	A
6.	Obtain yearly amount of repairs	8	(β)	0	B
7.	Check customer credibility	2	()	1	C
8.	Estimate current market prices	100	(2)	0	D
9.	In-house capability check	3	()	1	E
10.	Determine PMA/DER	20	(α, 5)	0	F
11.	Determine over and aboves	15	(α, 5)	0	G
12.	Fix contract length	4	(β)	0	H
13.	Obtain transport cost	12	(4)	0	I

Table 3: Procedural tasks for Koffer case

	Case File items	Distribution values	Distribution
A.	Net expected repair cost	{1500,1700,1900,2100}	Uniform
B.	Expected number of repairs per year	{40,60,80,100}	Binomial(0.2)
C.	Customer creditworthiness	{0,1}	Bernoulli(0.3)
D.	Current mean market price	{1700,1900,2100,2300}	$[\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}]$
E.	In-house capability	{0,1}	Bernoulli(0.6)
F.	PMA/DER possibilities	{100, 200}	Bernoulli(0.3)
G.	Over and above options	{0,1}	Bernoulli(0.5)
H.	Contract period	{1,2,3,4}	$[\frac{1}{6}, \frac{2}{6}, \frac{2}{6}, \frac{1}{6}]$
I.	Transportation cost	{50,100,150,200}	Uniform

Table 4: Case File items for Koffer case

The random variable $\mathcal{D}$ is modeled by a normally distributed random variable with $\mu = D$ and $\sigma = 200$. Furthermore, we have decision alternatives, $\mathcal{F} = \{\text{No bid}, 1800, 2000, 2200\}$.

Filter MDP model. The next step in our approach is to build up the MDP model from the information structure and the tasks assigned to different classes. Following the approach in Section 5.4, we filter the following plan state which is a tuple of tasks 2 to 13 and stages α and β. In this state, the σ parameter is the 'completed' attribute for tasks and stages: $\sigma = [\sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_\alpha, \sigma_\beta]$. Note that besides this case plan state, there is also an information state that contains the acquired values of the case file items, where x is the data attribute value: $s = [x_A, x_B, x_C, x_D, x_E, x_F, x_G, x_H, x_I]$. The action space consists of two sets. Either we collect more information by performing one of the twelve procedural tasks or we open a stage, or we take a final decision:

$$\mathcal{A} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, \alpha, \beta\} \cup \{\text{No bid}, 1800, 2000, 2200\}$$

Table 3 lists precedence and required constraints for the information gathering actions to become enabled in a specific state.

Next, the reward for a given state and action is either the extra cost for gathering information, given in the tables, or the expected profit based on the information that is available in the current state **s** and final decision $f \in \mathcal{F}$:

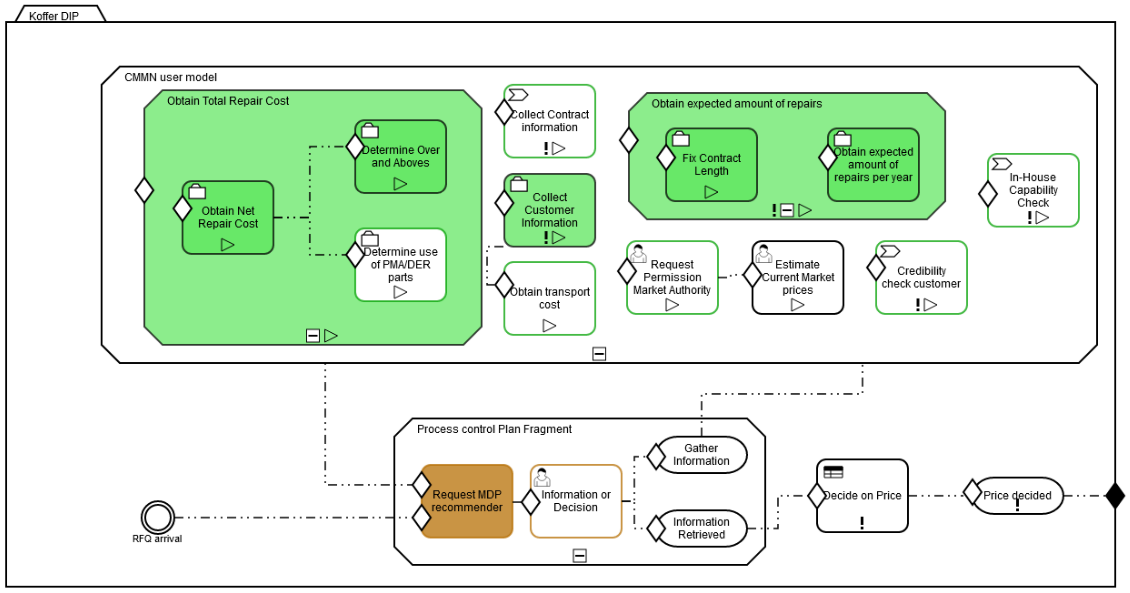
$$R(s, \mathbf{a}) = \begin{cases} -c_{\mathbf{a}} & \text{if } \mathbf{a} \in P \cup V \\ \mathbf{E}[Y(s, f)] & \text{if } \mathbf{a} = f \end{cases}$$

Finally, the transitions can be derived exactly the way they were introduced in section 5.4 using the probability distributions in Table 4. As described in Section 5.5, the filtered MDP can be solved using backward induction techniques. We used Python to program an algorithm that optimizes MDPs that are built from tables such as Table 3 and 4.

7.3. Deployment

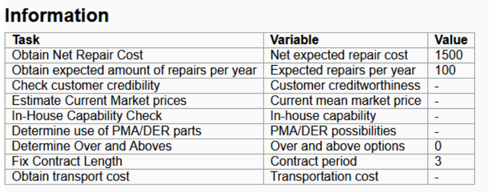
After all the design-time steps have been taken as explained above, the process to be enacted can be deployed. Having the MDP solution, the decision plan fragment is combined with the user CMMN plan and enacted by the CMMN engine to support the decision maker in performing the DIP. Figure 6(a) shows the current state of the case, where green filled tasks have been performed and green lined tasks are enabled. Because of constraints for this specific state, the final decision is not yet allowed and the decision maker can choose between six different tasks to continue the DIP. In expectation the optimal choice would be to perform In-house capability check. However, the five other tasks are also enabled in the current situation and their expected profit is also given. In this specific state most of the important information is already collected and therefore the differences in profit between the different options are marginal.

The recommender shows the allowed options and the expected profit for each decision in the *Possible Tasks* table in Figure 6(c). Moreover, because the MDP calculates the



Koffer DIP
CMMN user model
Obtain Total Repair Cost
Obtain Net Repair Cost
Determine Over and Aboves
Determine use of PMA/DER parts
Collect Contract information
Collect Customer Information
Obtain transport cost
Obtain expected amount of repairs
Fix Contract Length
Obtain expected amount of repairs per year
In-House Capability Check
Request Permission Market Authority
Estimate Current Market prices
Credibility check customer
Process control Plan Fragment
Request MDP recommender
Information or Decision
Gather Information
Information Retrieved
Decide on Price
Price decided
RFQ arrival

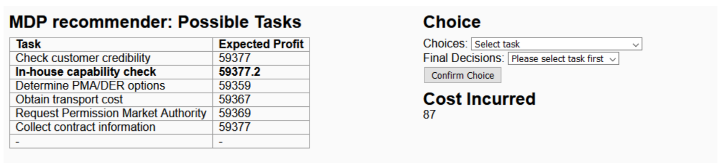
(a) Current state; tasks filled green and brown have been performed; tasks with green and brown lining are enabled



Information

Task	Variable	Value
Obtain Net Repair Cost	Net expected repair cost	1500
Obtain expected amount of repairs per year	Expected repairs per year	100
Check customer credibility	Customer creditworthiness	-
Estimate Current Market prices	Current mean market price	-
In-House Capability Check	In-house capability	-
Determine use of PMA/DER parts	PMA/DER possibilities	-
Determine Over and Aboves	Over and above options	0
Fix Contract Length	Contract period	3
Obtain transport cost	Transportation cost	-

(b) Case File



MDP recommender: Possible Tasks

Task	Expected Profit
Check customer credibility	59377
In-house capability check	**59377.2**
Determine PMA/DER options	59359
Obtain transport cost	59367
Request Permission Market Authority	59369
Collect contract information	59377
-	-

Choice

Choices: Select task

Final Decisions: Please select task first

Cost Incurred

87

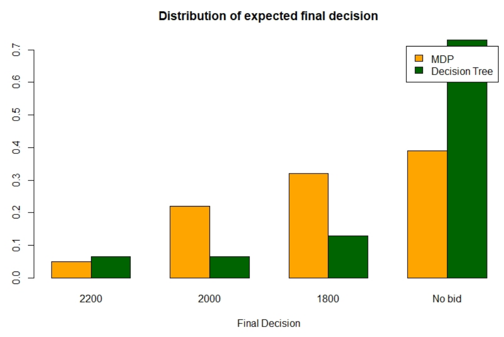
(c) Recommendations

Figure 6: Recommender view

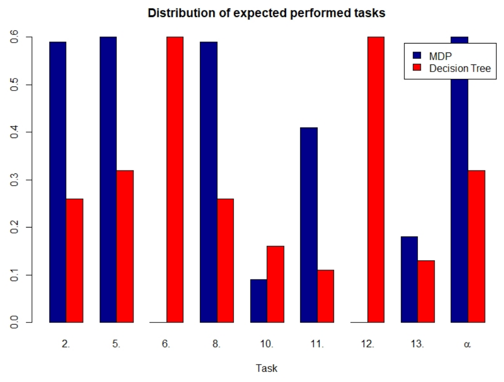
optimal future profit, the decision maker can estimate the effects of current decisions. Using the decision plan fragment that is inserted in the user plan, shown in Figure 6 with the brown color, the CMMN engine can enact on the decisions that are taken. In accordance with Figure 1, the CMMN engine returns the new state of the process and calls for the MDP recommender in the new state. A demonstrator for this specific case can be accessed via `https://is.ieis.tue.nl/staff/heshuis/OptimizingInformationGathering`.

7.4. Numerical analysis

We compare the outcome of the approach with a decision tree that is based on the current decision process and is created in cooperation with sales managers in the company that inspired the example. The decision tree is shown in Figure 8 in appendix A. The level of complexity is already high for a human decision maker using the decision tree. However, the number of nodes in this tree (14) is marginal compared to



(a) Decisions



(b) Tasks

Figure 7: Distributions

Table 5: Comparison MDP vs Decision Tree

Information	*MDP*	*Decision tree*
Expected profit	15867.6	8226.0
Number of final states	136	1254
Average retrieval cost	143.5	100.06
Expected revenue	16011.0	8125.94

the number of states of the MDP (4050000). In Table 5 we introduce some statistics based on the case model that was introduced in Section 3. Figure 7(a) shows the distribution of final decisions over all possible outcomes of the case file items. Figure 7(b) shows the proportion of cases in which a task is executed. Note that we removed the tasks that are *required* (3,4,7,9,β) as they are always executed to finish the process.

The main comparison can be made based on the expected profit of both methods. The MDP is able to double the expected profit compared to this human decision tree. If we look at the final decision distribution in Figure 7(a), clearly too many process instances are discarded in this decision tree. Also the distribution over the information sources that are acquired is, apart from the structurally enforced part, rather different. Furthermore, the decision tree also has a larger number of final states, meaning states where no more information is acquired. Together with the average retrieval cost, this implies that the decision tree is not complex enough to capture the dependencies between different attributes as defined in the information structure.

In Table 6, we show the results for nine randomly selected instances of the Koffer case process. Hence, we can show how both algorithms work for those instances. There are five instances where both algorithms conclude it is not profitable to bid for the maintenance

Instance	*Final Information State [A,..,I]*	*Profit MDP*	*Profit Decision tree*
1.	[2100, 40, 1, 1900, 1, 100, 0, 1, 150]	1656	-122
2.	[1500, 100, 1, 2100, 0, 100, 1, 4, 100]	69794	65469
3.	[2100, 80, 1, 2100, 1, 200, 0, 1, 50]	-45	-45
4.	[2100, 40, 0, 2300, 0, 200, 0, 3, 150]	16385	14692
5.	[1900, 100, 0, 2300, 0, 200, 0, 4, 200]	-45	-45
6.	[1900, 40, 0, 2300, 0, 200, 0, 3, 150]	-45	-45
7.	[1500, 40, 1, 1900, 1, 100, 0, 1, 150]	-45	-45
8.	[1700, 80, 1, 1700, 1, 100, 0, 2, 200]	27416	27419
9.	[1900, 60, 0, 1900, 0, 100, 1, 1, 150]	-45	-45

Table 6: Comparison of 9 instances

contract. Therefore, they make a loss on the pieces of information that are always required to acquire. However, looking at instance 8, the human-built decision tree actually makes a better final decision than the MDP or makes that decision with less acquired information. This instance is a good example why the introduced approach does not only use the MDP algorithm but always requires the final opinion of the decision maker, where they can use tacit knowledge to judge the instance. However, including the instances 2 and 4, where the MDP is clearly outperforming the decision tree, we can see that on average for those 9 instances the MDP does improve upon the decision tree.

For this specific case the statistics have proven the value of using our approach during deployment with a doubling of the expected profit. Together with the steps that have been followed during the design time approach in Section 7.2, the approach has been shown to be feasible and applicable in real-life process cases. Furthermore, by using a real-world case and by validating the approach with decision makers, we show that this approach is improving upon the current approach that is based on decision trees. Given the necessary constraints, we consider this approach valuable for many more DIPs in different areas of expertise, such as financial services or governmental tender procedures.

8. Related Work

Process modeling languages like BPMN and UML Activity Diagrams offer basic support for modeling decision points. To enable reuse of such decision logic, the modeling language DMN was developed to declare recurring decision points in DMN decision tables that can be referenced in tasks of business processes [15]. DMN focuses on what information is needed to make decisions. The last few years, however, several authors have recognized that decisions often are an integrated part of the business process that actually even controls the flow [12, 10, 14]. But there is no guidance for decision makers about which information to gather in order to perform DIPs both efficiently and effectively.

Hasic et al. [14] introduce five necessary *principles* to connect a process model and a decision model: The first principle states that the decision logic should be externalized as a service to the process. In this paper, we use a separate MDP tool that implements the decision logic, which could be realized as a service. A second principle is the need to model all data objects. Here, we use case file items, modeled as data attributes. The third principle of always modeling all possible decision outputs is followed in our approach. The fourth principle asks for ordering decision activities. By using a declarative language such as CMMN we allow the flexibility of deciding on the activity order and at the same time include any process constraints that define an order. The final principle involves making subdecisions. This specific extension to our current approach is discussed in Section 9.

DIPs have been modeled using declarative process modeling. Vaculin et al. [1] propose to model DIPs with declarative business artifacts. They develop design pattern support for the execution of DIPs. Guidance is modeled in the form of expert-defined constraints. Mertens et al. [12] develop DeciClare, a declarative process modeling language targeted for DIPs. The process perspective in DeciClare is modeled with Declare, the decision perspective with DMN. No guidance support is provided. The approach defined in this paper can be used in combination with DeciClare, replacing DMN-related parts with the decision fragments defined in Section 5 and 6.

In general, the approach can be used in combination with other declarative process modeling languages, notably DCR graphs [21, 22] and Declare [23]. For instance, in the context of Declare, recommendation support has been developed, based on past process executions and their costs [24]. The approach in this paper is much more advanced, since it considers optimized decision making based on actual output data of tasks.

Next, there are approaches to guide the execution of information-intensive business processes, which produce information products [25, 26]. The structure of the information products is specified in product data models, which are tree structures that declaratively define how a data item is assembled from other data items. Different strategies exist to optimize the assembly of the informational products. Our approach focuses on optimizing decision making within the execution of DIPs. Therefore, we use information structures, which focus on the value of data items for decision making rather than their structure.

In earlier work [20, 6] we discussed the value of using MDPs to support the execution of DIPs that are modeled in an artifact-centric process language. There we presented a detailed translation of Guard-Stage-Milestone process models into MDPs [20]. The present approach builds upon this translation by defining an approach that shows how DIPs expressed in CMMN can be configured such that decision makers are guided during execution of a process to optimize decision making.

AI planning has been used to support the execution of knowledge-intensive processes [13, 27, 28]. A plan is a sequence of tasks such that a goal state is reached. Performing a task results in a new state. Tasks can be modeled at different levels of granularity. If the user deviates from the plan and ends up in a state that is not in the plan, a new plan is generated. While these approaches support the flexible ad-hoc execution of processes, they do not support decision making, especially making the trade-off between gathering more information and making final decisions.

For expert systems, several optimization approaches for information acquisition strategies have been proposed [29], for instance using dynamic programming [30]. However, none of these consider the relation with process models, such as CMMN and DIPs consisting of information acquiring tasks, procedural tasks, additional tasks, and decision-making tasks. Finally, we use the notion of an information structure which links the benefits and costs of gathering data for final decisions.

In sum, the main contribution of this paper is a flexible approach, based on optimization, for guiding decision makers in performing DIPs both efficiently and effectively.

9. Discussion

Hierarchy in DIPs. One of the important modeling assumptions for the CMMN user model is the use of a single milestone, which concludes the entire process. However, an important reason why milestones are part of the CMMN language is the opportunity of setting subgoals during the process. Related, CMMN offers the possibility to create hierarchy in a DIP. By using substages one can define different layers of tasks and decisions to meet subgoals. This structuring can actually be of great value to DIPs with many tasks and sources of information. By introducing more milestones and stages, one could create a hierarchical tree of smaller size decision problems. This decreases the state space of the DIPs dramatically. For example, if we cut a 10 attribute process all with a domain of four values into two smaller size processes, we can go from as much as 1048576 states to 2048 states. The main reason why we did not include this in our approach is that one needs more information structures and a higher level decision making that oversees which subgoals have the highest priority. This requires a 'two-level' approach that would obfuscate the key ideas and principles underlying our solution.

Probabilities. The existence of an MDP solution fully relies on the availability of probabilities and the assumptions that those probabilities represent the real world. Although frequencies of historical cases can give a good estimate, we know that in real-life things are always different. This is why we introduce the MDP as an advisor to the decision maker. Tacit knowledge is necessary to recognize specific historical cases and see dependencies between those and the current process. Still, we show in Section 7.4 that in expectation the MDP significantly improves the profit. This does not rule out that for specific cases the decision maker can use additional information not available to the model to create an even higher outcome by diverging from the 'average' advice. However, as back bone for the complete set of cases, this MDP model is shown to make a real difference.

10. Conclusion

This paper introduced a new approach for decision support for information gathering in Decision Intensive Processes. The approach translates a CMMN process model into an MDP. After calculating the optimal solution for the MDP model, a decision maker is given advice during deployment what information to gather in order to make a final decision that is both efficient and effective. The approach therefore defines a CMMN plan fragment for letting a decision maker use MDP advice to control the process execution. Using the approach, decision makers performing DIPs are supported in deciding what information to gather to optimize the final decisions. Although these information-gathering decisions could be automated, we value the opinion of the decision maker and view its role as essential to decide which information to gather for a specific case.

An exemplary case based on a real-world industrial scenario shows the feasibility of the approach. For this real-world scenario, we compared a basic implementation of the approach using an MDP optimizer with a decision tree that was constructed by a decision maker from industry. We showed that the algorithm can double the expected profit compared to the decision tree. We also showed that there are still instances where the decision maker can make better decisions, e.g., based on information that is not available to the model, which suggests that a human decision maker should be in the lead.

A main topic for further research is improving the scalability of the approach for problems where the state space explodes. We plan to use techniques such as deep reinforcement learning or segmenting the problem into smaller problems by using intermediate goals that are represented in the CMMN model as milestones. Deep reinforcement learning uses neural networks to learn the policy on a subset of the complete state space. Further research can also be done in the direction of how to inform the decision maker. By putting constraints on the information structure, one might be able to give additional information about marginal contribution or variance reduction per attribute.

Acknowledgements. This work was supported by the "Netherlands Organisation for Scientific Research" (NWO). Project: NWO Big data - Real Time ICT for Logistics. Number: 628.009.012

References

[1] R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam, P. Sukaviriya, Declarative business artifact centric modeling of decision and knowledge intensive business processes, in: Proc. EDOC 2011, 2011, pp. 151–160 (2011).

[2] D. Bromberg, Bpm for knowledge workers: The structural foundations of decision intensive processes (dips), BPTrends (January 2007).

[3] P. Bossaerts, C. Murawski, Computational complexity and human decision-making, Trends in Cognitive Sciences 21 (12) (2017) 917 – 929 (2017).

[4] J.-C. Pomerol, Artificial intelligence and human decision making, European Journal of Operational Research 99 (1) (1997) 3 – 25 (1997).

[5] F. E. Horita, J. P. de Albuquerque, V. Marchezini, E. M. Mendiondo, Bridging the gap between decision-making and emerging big data sources: An application of a model-based framework to disaster management in Brazil, Decision Support Systems 97 (2017) 12 – 22 (2017).

[6] R. Eshuis, Modeling decision-intensive processes with declarative business artifacts, in: Proc. ASSRI 2018, 2018, pp. 3–12 (2018).

[7] C. D. Ciccio, A. Marrella, A. Russo, Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches, J. Data Semantics 4 (1) (2015) 29–57 (2015).

[8] S. K. Venero, J. C. dos Reis, L. Montecchi, C. M. F. Rubira, Towards a metamodel for supporting decisions in knowledge-intensive processes, in: Proc. SAC 2019, 2019, pp. 75–84 (2019).

[9] R. Eshuis, M. Firat, Modeling uncertainty in declarative artifact-centric process models, in: Proc. BPM 2018 Workshops, 2018, pp. 281–293 (2018).

[10] A. Yousfi, A. K. Dey, R. Saidi, J. Hong, Introducing decision-aware business processes, Computers in Industry 70 (2015) 13–22 (2015).

[11] A. Yousfi, K. Batoulis, M. Weske, Achieving business process improvement via ubiquitous decision-aware business processes, ACM Trans. Internet Techn. 19 (1) (2019) 14:1–14:19 (2019).

[12] S. Mertens, F. Gailly, G. Poels, Towards a decision-aware declarative process modeling language for knowledge-intensive processes, Expert Systems with Applications 87 (2017) 316 – 334 (2017).

[13] S. K. Venero, B. R. Schmerl, L. Montecchi, J. C. dos Reis, C. M. F. Rubira, Automated planning for supporting knowledge-intensive processes, in: Proc. BPMDS 2020, Springer, 2020, pp. 101–116 (2020).

[14] F. Hasic, J. D. Smedt, J. Vanthienen, Augmenting processes with decision intelligence: Principles for integrated modelling, Decision Support Systems 107 (2018) 1–12 (2018).

[15] Object Management Group, Decision model and notation. version 1.3 (2020).

[16] Object Management Group, Case management model and notation. version 1.1 (2015).

[17] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley Series in Probability and Statistics, Wiley, 1994 (1994).

[18] M. A. Marin, M. Hauder, F. Matthes, Case management: An evaluation of existing approaches for knowledge-intensive processes, in: Proc. BPM 2015 Workshops, 2015, pp. 5–16 (2015).

[19] A. Grudzińska-Kuna, Supporting knowledge workers: case management model and notation (cmmn), Information Systems in Management 2 (1) (2013) 3–11 (2013).

[20] S. Voorberg, R. Eshuis, W. van Jaarsveld, G. van Houtum, Decision support for declarative artifact-centric process models, in: Proc. BPM Forum 2019, 2019, pp. 36–52 (2019).

[21] T. T. Hildebrandt, R. R. Mukkamala, T. Slaats, Designing a cross-organizational case management system using dynamic condition response graphs, in: Proc. EDOC 2011, 2011, pp. 161–170 (2011).

[22] S. Debois, T. T. Hildebrandt, T. Slaats, Hierarchical declarative modelling with refinement and sub-processes, in: Proc. BPM 2014, 2014, pp. 18–33 (2014).

[23] M. Pesic, H. Schonenberg, W. M. P. van der Aalst, DECLARE: full support for loosely-structured processes, in: Proc. EDOC 2007, 2007, pp. 287–300 (2007).

[24] H. Schonenberg, B. Weber, B. F. van Dongen, W. M. P. van der Aalst, Supporting flexible processes through recommendations based on history, in: Proc. BPM 2008, 2008, pp. 51–66 (2008).

[25] I. T. P. Vanderfeesten, H. A. Reijers, W. M. P. van der Aalst, Product-based workflow support, Inf. Syst. 36 (2) (2011) 517–535 (2011).

[26] H. van der Aa, H. Leopold, K. Batoulis, M. Weske, H. A. Reijers, Integrated process and decision modeling for data-driven processes, in: Proc. BPM 2015 Workshops, 2015, pp. 405–417 (2015).

[27] A. Marrella, M. Mecella, S. Sardiña, Intelligent process adaptation in the smartpm system, ACM Trans. Intell. Syst. Technol. 8 (2) (2017) 25:1–25:43 (2017).

[28] I. Sid, M. Reichert, A. R. Ghomari, Enabling flexible task compositions, orders and granularities for knowledge-intensive business processes, Enterp. Inf. Syst. 13 (3) (2019) 376–423 (2019).

[29] V. S. Mookerjee, M. V. Mannino, Sequential decision models for expert system optimization, IEEE Trans. Knowl. Data Eng. 9 (5) (1997) 675–687 (1997).

[30] B. L. Dos Santos, V. S. Mookerjee, Minimizing information acquisition costs, Decision Support Systems 9 (2) (1993) 161–181 (1993).

A. Decision tree

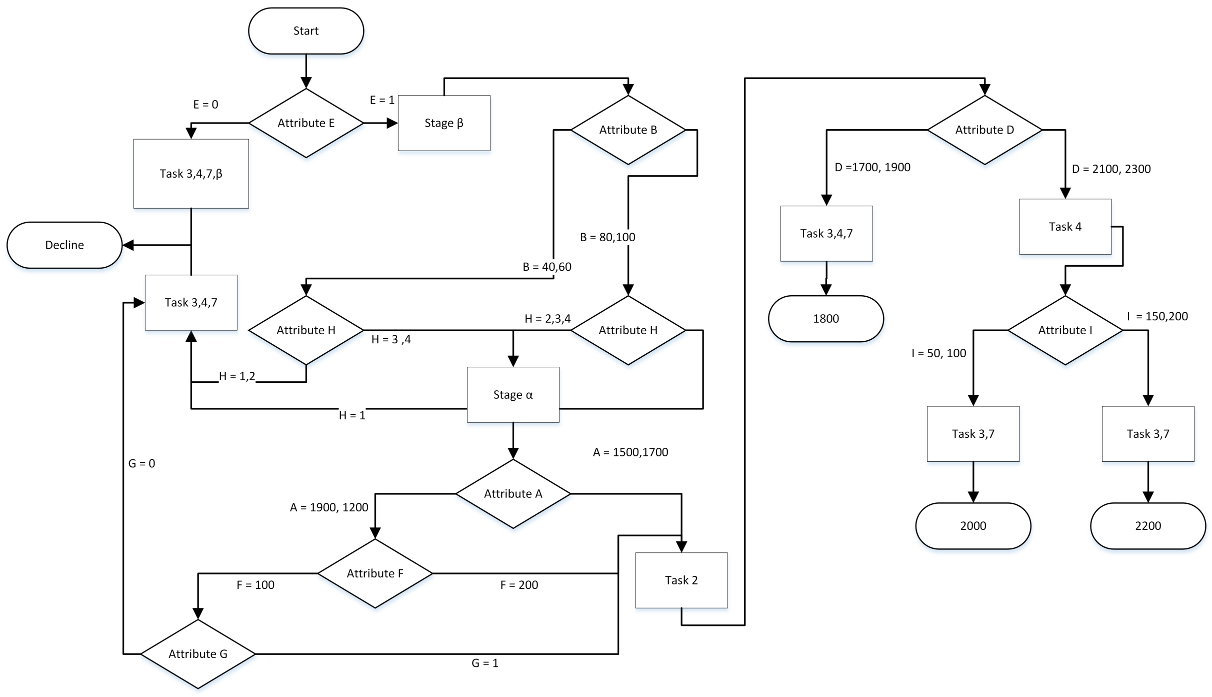


Figure 8: Decision tree