

# Modeling Decision Structures and Dependencies

Feng Wu, Laura Priscilla, Mingji Gao, Filip Caron,  
Willem De Roover, and Jan Vanthienen\*

Department of Decision Sciences & Information Management,  
KU Leuven, Belgium  
jan.vanthienen@kuleuven.be

**Abstract.** Decisions are often not adequately modeled. They are hardcoded in process models or other representations, but not always modeled in a systematic way. Because of this hardcoding or inclusion in other models, organizations often lack the necessary flexibility, maintainability and traceability in their business operations.

The aim of this paper is to propose a decision structuring methodology, named Decision Dependency Design (D3). This methodology addresses the above problems by structuring decisions and indicating underlying dependencies. From the decision structure, different execution mechanisms could be designed.

**Keywords:** Decision structuring, Decision dependencies, Decision modeling.

## 1 Introduction

Managing business decisions separately, and not only hidden in processes and procedures, is important for agility and traceability [1]. Moreover, not explicitly modeling decisions would result in insufficient documentation of the decision, which leads to problems during optimizing and redesigning operations [2].

This paper extends the concept of a decision goal tree, as presented in our earlier research [3] with the following contributions:

- Proposing a methodology that differentiates between different kinds of decision dependencies (information and combination dependencies), each with a distinct notation. The notation aims at distinguishing general decision structure from the more detailed decision logic expressed in business rule notations.
- Designing a mechanism to incorporate the decision dependencies into the execution mechanism using transformations.

The structure of this paper is as follows. Section 2 discusses related work. Section 3 states the overall scope of the Decision Dependency Design (D3) methodology. Sections 4 and 5 introduce the concepts and notation to structure decisions, and describe how to translate the models with a preliminary pattern language. We evaluate and conclude our proposed methodology in section 6.

---

\* Corresponding author.

## 2 Related Work

In [4], Lars Braubach et al. presented a goal-oriented process modeling approach, which extends concepts from goal-driven process design [5]. Yet, concerning the decision making processes, especially those decisions that span across several business processes, the goals elicited for individual process are not necessarily aligned together to represent the characteristics of the final decision.

An alternative method, namely Product Based Workflow Design (PBWD) is presented in [6]. Similarities can be found between PBWD and case handling workflow management systems [7, 8], as they focus on the data elements rather than on the control flow of the process. Our approach is related, but focuses more on the decisions than on the data, and builds upon our earlier research in this area [3].

## 3 Scope of Decision Dependency Design (D3)

### 3.1 D3 Related Business Rules

*A business rule is a statement that defines or constrains some aspect of the business* [9]. While many types and forms of business rules exist, in the methodology we only focus on rules and dependencies for decision making. In the D3 model we distinguish the following dependency types:

- **Input dependencies** elicit all the possible elements to be used as inputs for the decision under consideration. These data elements can be information collected from business activities or the result of other (sub-)decisions. Input dependencies are used to construct the *input dependency model*. The goal of constructing such a model is to visualize the relations between the decision and its constituting elements in a top down approach, the most important decision at the top.
- **Input combination dependencies** determine the set of inputs that is valid in different decision situations. The set (or subset) of inputs is used for the decision situation, but the exact nature of the outcome is not known yet, as it belongs to the business logic. A subset of the inputs might suffice to draw a conclusion about the decision without knowing the other inputs. Input combination dependencies are represented in the *input combination dependency model*. An example of input combination dependencies can be “To decide whether to play tennis or not, we sometimes only consider weather and the schedule of the player, but some other time we only consider weather and time of the day instead”.

The real constellation for deciding the outcome of a decision, based on some inputs, is called the business logic. For a decision, a set of business logics are associated with it, that react on the inputs and assign the output, expressed in the form of rules, tables or trees. Nevertheless, we do not embed detailed business logics in any of the artifacts produced by D3, because this logic can be managed separately so that the agility of the decision model can be enhanced. An example of business logic can be “IF the schedule of the player is busy and the weather is bad, THEN do not play tennis”.

Other types of rules and constraints can exist. Process constraints, e.g., are rules controlling and managing the sequence of tasks appearing in business processes. They are embedded in the business process models produced by D3, but not in the input dependency model or the input combination dependency model. In contrast to the conventional business process design where the decision related rules are hardcoded, such that the ordering of tasks may indicate an input dependency rule or a process constraint, our approach manages them separately. An example of a process constraint can be “Always check weather condition first before checking other conditions”.

Of course input dependencies and input combination dependencies can be derived from the business logic, but they can also exist on their own if the business logic is not fully known yet or often changes. For instance, a business rule stating that “IF the schedule of the player is busy and the weather is bad, THEN do not play tennis.” can be categorized as business logic, since it tells what decision should be made under the given conditions. However, it also contains information about what inputs are needed for making this decision.

The reason for this distinction is that the dependencies might influence the structure of the business process, but business logics normally have no impact on business process execution, so that they can be managed elsewhere. Hardcoding business logics may cause maintenance problems in both process and decision management.

### 3.2 D3 Phases

Four major phases can be distinguished during a D3 project:

- Stage 1: Gathering requirements.
- Stage 2: Eliciting input dependencies and building the input dependency model.
- Stage 3: Eliciting input combination dependencies and building the input combination dependency model.
- Stage 4: Eliciting process constraints and transforming the input combination model to a process execution model.

The D3 model can be used to build one or more process models that comply with the input and input combination dependencies. Changes in dependencies will lead to other process model designs. Nevertheless, we cannot draw a clear line between each phase. Normally the project will be implemented in a cyclical manner. Evaluation of current deliverables will occur in every phase of D3. Business decisions and related business rules are not always well documented and tracked. They hide in process models, code, and texts. However, techniques exist to identify and model decisions from business routines [10].

## 4 D3 Notation

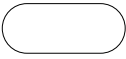

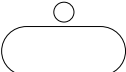
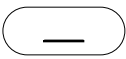

### 4.1 A Running Example: Unemployment Benefits

Suppose a government labor department has the obligation to give unemployment benefits to citizens (see also [2]). Everyday a lot of applicants apply to it each asking for approval. The approval decision is based on one or more of the following criteria: the nationality, the employment period, the employment type, the validation of the ID

card, the physical test result, and also the health condition of the applicant. The health condition itself is determined by physical and psychological tests. Data inputs for each (sub-)decision can be collected from applicants or systems. Some of the data are always required (e.g. nationality), some are not, depending on the situation.

4.2 Input Dependency Model (IDM)

Notation:

	<b>Decision input:</b> Except for the top decision, all other sub-decisions shown in the model serve as inputs for their upper level decision(s).
	<b>Input path:</b> Showing the dependency relationship between input elements and decisions.
	<b>Optional decision input:</b> The result of this decision may be considered as input for upper level decisions under certain situations.
	<b>Reusable sub-decision input:</b> A sub-decision that will be used as repetitive input for multiple decisions. Reusable sub-decision input helps to increase efficiency by avoiding redundant data collection.
	<b>Data input:</b> Data elements which are not output from any sub-decisions.

The IDM reflects the relationship between lower and upper decisions and stays unchanged unless the most fundamental business rules change. The IDM for the unemployment example is presented in figure 1. In this model, there is no information on which input combinations exactly lead to certain decisions, but it already lists required and optional data inputs and sub-decisions. Therefore the model is not subject to minor policy changes and is stable enough to serve as a foundation for the process.

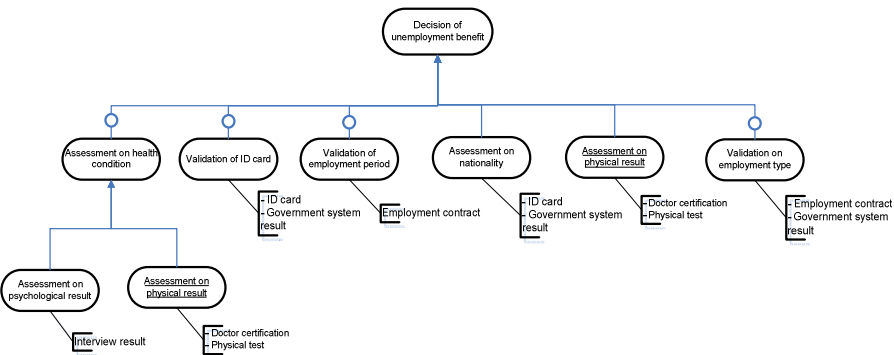
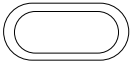
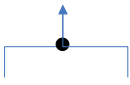



Fig. 1. Input dependency model

4.3 Input Combination Dependency Model (ICDM)

Notation:

Only notations different from the previous model will be presented.

	<b>Indispensable decision input:</b> A decision that will always be required to serve as input(s) for its upper level decision.
	<b>Joint decision input:</b> Showing a combination of inputs which is sufficient in certain situation for making the decision.
	<b>Disjoint decision input:</b> Indicating a combination of only one input which is sufficient in certain situation for making the decision.

Derived from the IDM and input combination dependencies, this model indicates decision paths leading to upper decisions. The ICDM for the unemployment example is presented in figure 2. This model shows all the combination of inputs from which the upper level decision can be made. Such combinations can be expressed as mathematical sets. The following example illustrates three combinations, one of which is <Validation of employment period, Assessment on nationality, Assessment on physical result, Decision of unemployment benefit>. In real life, the combinations can also be demonstrated with different colors for better interpretation. Since input combination rules are less stable because they convey more detailed decision logic than input dependency rules, the ICDM is less stable than the IDM.

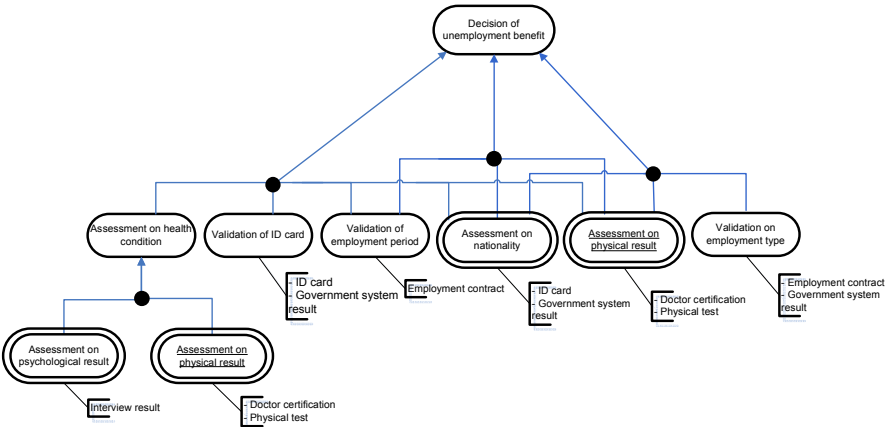


Fig. 2. Input combination dependency model

## 5 Transformation Patterns

A pattern is the description of a proven solution to common problems to achieve efficient and effective problem solving [11]. In this section, two transformation patterns are introduced to give guidance for transforming the decision structure into a process execution model: the disjoint transformation pattern and the joint transformation pattern. More transformation patterns satisfying other criteria will be a topic for future research.

### 5.1 Disjoint Transformation Pattern

This pattern will be applied to all cases that have a disjoint decision input symbol in the ICDM. We propose only one pattern here for transforming the disjoint decision input symbol into a process model: the exclusive gateway. A simple algorithm can be defined for the transformation:

```

If there is a disjoint decision input symbol
  If there is a joint decision symbol below
    Then group all the joint decision nodes as a sub-process containing mutually inclusive tasks
  Else put the disjoint decision nodes as tasks within exclusive gateways;
  
```

For the unemployment example, the process is derived from the ICDM and shown in figure 3. Three decision paths are derived to reach the decision and one of those paths will be chosen during execution.

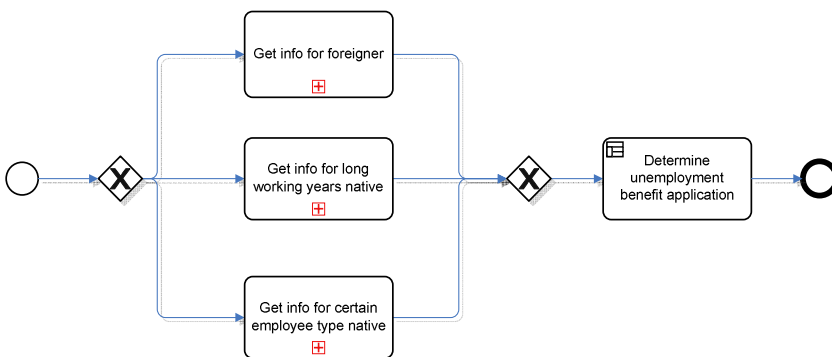


Fig. 3. Exclusive gateway process

### 5.2 Joint Transformation Pattern

This pattern applies to all cases that have a joint decision input symbol in the ICDM. We propose two general types of joint transformation pattern: the parallel pattern and the sequential pattern. Time and cost are the criteria to determine the selection of these patterns, but there can be more criteria for selecting the appropriate business process, such as desired flexibility, data constraints, etc. [12].

## Parallel Pattern

When the criterion is to achieve minimal throughput time, the parallel pattern is best suited to build the process [12]. The algorithm is as follows:

```

If there is a joint decision input symbol
  If the joint decision has a group of lower decisions
    Then put the joint decision node as a sub-process task
  Else put the joint decision node as a task;
  
```

For the activity “Get info for foreigner” in figure 3 e.g., the sub-process to be built is shown in figure 4. This parallel process can be time efficient, because all tasks for the same case can start at the same time. However if the outcome is negative, a lot of unnecessary work may have been performed, so the cost will not always be minimal.

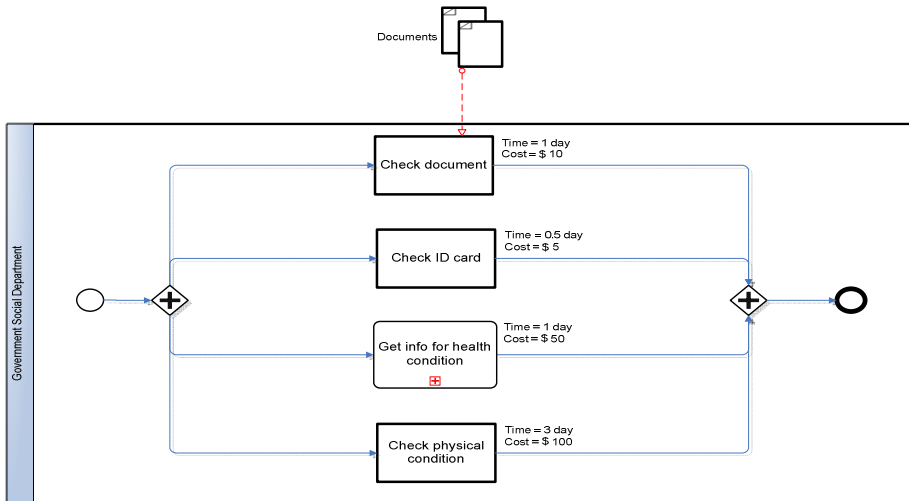


Fig. 4. Parallel pattern

## Sequential Pattern

When the criterion is to minimize execution cost, the sequential pattern is best suited to build the process [12]. To transform an ICDM containing a joint decision input symbol into a sequential business process, a simple algorithm is defined:

```

If there is a joint decision input symbol
  If the joint decision has a group of lower decisions
    Then put the joint decision node as a sub-process task in sequential order
  Else put the joint decision node as a task in sequential order;
  
```

Based on the algorithm above, a sequential process to get information for a foreigner in case of unemployment is presented in figure 5. On average, a sequential process has the advantage of reducing cost because during the execution, a decision process can be terminated if some preconditions are not met, therefore, it saves the cost of carrying out the remaining tasks [12].

The ordering of tasks within the process is still flexible. If there are process constraints applying to certain tasks, the ordering is determined accordingly. Otherwise multiple process models could exist with different orderings, each of which reflects particular business preferences and concerns. Normally it will be good practice to start with the simple (inexpensive) checks.

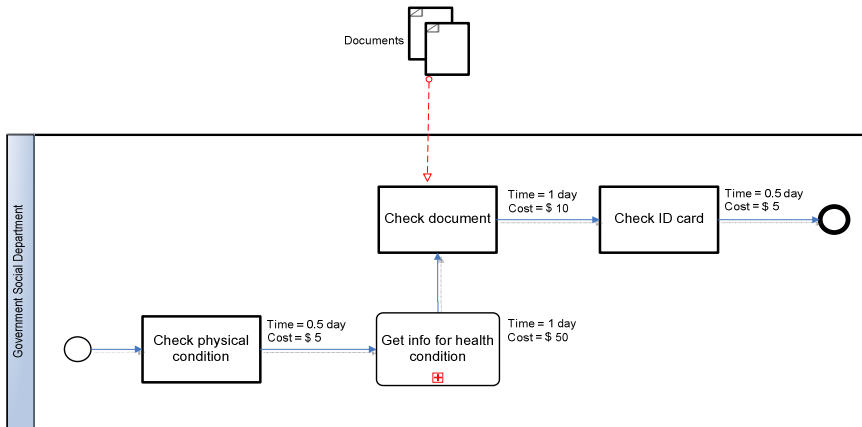


Fig. 5. Sequential pattern

## 6 Discussion and Future Research

Business decision management should have clearly stated goals during the entire process of eliciting, analyzing, defining, tracking, evaluating, and documenting business decisions. Frank Rohde defines five criteria to ensure better decision yield – that is, the impact of decisions on business results [13]: Precision, cost, speed, agility and consistency. The proposed D3 method results in a better separation between business decisions and execution processes, which makes it easier and clearer to achieve the above objectives. Business decisions that have been properly modeled can be better evaluated and improved throughout their lifecycle. Moreover, the transformation patterns allow for added operational agility when the characteristics (e.g. time and cost) for obtaining lower level elements change. Future research must focus on additional transformation patterns that take into account different characteristics (e.g. the organizational aspect where the goal would be to limit the number of handovers) or a combination of characteristics.

As stated in section 5, there can be more patterns for specific business circumstances to guide transforming decision models to business process models. The pattern language can incorporate best practices from industry. Furthermore, using existing systems and standards to support the modeling and implementation of D3 will be a topic for further investigation. Finally, the effectiveness and the complexity of this methodology will be tested in real life situations.



## 7 Conclusion

D3 enables better decision management by separating decisions from processes. It introduces models illustrating the relationships between decisions, and provides ways to derive business processes to facilitate corresponding decision making. This approach will increase the flexibility, traceability and maintainability of the underlying decision making processes, while at the same time, it minimizes the impact from changes caused by modification of specific decision logic.

## References

1. Taylor, J.: Smart (enough) systems: how to deliver competitive advantage by automating the decisions hidden in your business. In: Taylor, J., Raden, N. (eds.) Prentice Hall, Upper Saddle River (2007)
2. Reijers, H.A., Limam, S., van der Aalst, W.M.P.: Product-based workflow design. *J. Manage. Inform. Syst.* 20(1), 229–262 (2003)
3. De Roover, W., Vanthienen, J.: On the Relation between Decision Structures, Tables and Processes. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM-WS 2011. LNCS, vol. 7046, pp. 591–598. Springer, Heidelberg (2011)
4. Braubach, L., Pokahr, A., Jander, K., Lamersdorf, W., Burmeister, B.: Go4Flex: Goal-Oriented Process Modelling. In: Essaaidi, M., Malgeri, M., Badica, C. (eds.) Intelligent Distributed Computing IV. SCI, vol. 315, pp. 77–87. Springer, Heidelberg (2010)
5. Jacobs, S., Holten, R.: Goal driven business modelling: supporting decision making within information systems development. In: Proceedings of Conference on Organizational Computing Systems 1995, pp. 96–105. ACM, Milpitas (1995)
6. van der Aalst, W.M.P.: On the automatic generation of workflow processes based on product structures. *Comput. Ind.* 39(2), 97–111 (1999)
7. Vanderfeesten, I., Reijers, H.A., Aalst, W.M.P.: Case Handling Systems as Product Based Workflow Design Support. In: Enterprise Information Systems, pp. 187–198 (2008)
8. Vanderfeesten, I., Reijers, H.A., van der Aalst, W.M.P.: An evaluation of case handling systems for product based workflow design. In: International Conference on Enterprise Information Systems, ICEIS 2007 (2007)
9. Hay, D.A.H., Anderson, K., Hall, J., Bachman, C., Breal, J., Funk, J., Healy, J., McBride, D., McKee, R., Moriarty, T., et al.: Defining Business Rules What Are They Really? The Business Rules Group (2000)
10. Taylor, J.: Decision management systems: a practical guide to using business rules and predictive analytics, vol. xxviii, p. 284. IBM Press/Pearson plc., Upper Saddle River (2012)
11. Alexander, C.: A pattern language: towns, buildings, construction (1977)
12. Reijers, H.A.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega: The International Journal of Management Science* 33(4), 283 (2005)
13. Rohde, F.: Little decisions add up. *Harvard Business Review* 83(6), 24–+ (2005)