# A Transformation Approach: Using DFD to Map as Different Views in System Modeling

Houda Kaffela
RIADI Laboratory-ENSI, University of Manouba, Manouba, Tunisia.

Tel.: +21697485902; email: houdakaffela@hotmail.fr
Manuscript submitted November 28, 2023; revised February 19, 2024; accepted April 30, 2024; published July 22, 2024
doi: 10.17706/jsw.19.3.79-85

**Abstract:** This paper describes an approach to transform Data Flow Diagrams (DFD) into intention-oriented model expressed in MAP. This transformation is performed by establishing the correspondence between DFD and map constructs, by applying the mapping rules for the transformation of DFD into map and finally producing an intention-oriented model. We will focus on the relation between DFD and map to find a way to translate data flow models represented by DFD into map.

**Keywords:** Data Flow Diagram (DFD), Intention-oriented requirements, map, mapping

## 1. Introduction

To ameliorate the system requirements analysis and design processes, many approaches have been proposed [1], each of which has its own purposes. Among those approaches, which can be used to designing systems, are process-oriented (e.g., Data Flow Diagram (DFD) [2]), data-oriented and object-oriented. Other approaches for requirements engineering have been emerged to help the system designers in the specification of the requirements. Some of the most notable approaches of this kind are scenario-based [3], goal/scenario-oriented [4] and intention/strategy-oriented (e.g., map [5]). In this paper, we concentrate on two types of those approaches process-oriented and intention-oriented.

Many process models exist to help organization in the specification of their processes. One of these models is the Data Flow Diagram, which can be used to represent the movement of data between processes. Similarly, there are several intentional models that offer an effective way to understand the organizational and users' requirements. These models have gained importance in recent years. Furthermore, they play a key role within requirements engineering. We quote among them map which is mainly intended to describe these requirements in terms of intentions and strategies between them.

But the process of transformation from one model to another is an important yet challenging task. It is difficult to draw a line between these models. To overcome this problem, this paper aims to combine process-oriented approach with the intention/strategy-oriented one. In our point of view, such a combination of both approaches would bring many benefits to the DFD designers and map designers. For example, the amount ambiguities in the design system can be reduced.

This approach combines two different views of the system (i) a functional view usually focused on the functionality of the system. This view is described using the DFD and (ii) an intentional view concentrates on the requirements of the system. This view is expressed using map model.

In this work our goals are to: (a) propose a set of rules for transforming DFD into map and (b) show how the data flow models expressed by DFD can be transformed into map by applying these rules.

Following we give, in Section 2, the background needed for the understanding of this approach with an overview of DFD and map. Section 3 discusses the process of transforming the DFD into map models. In Section 4, an example bank account management system is presented in which this approach is illustrated. The paper ends with some conclusions and comments about our approach.

## 2. Background

Many works have already presented the mapping between DFD and other type of modeling languages and formalisms [6–9]. Here we only mention the work that we feel is most relevant to the present work.

This work is presented in [10], which aims to couple an existing and well-known requirement engineering approach (map) with DFD for system design. This coupling is achieved through the definition of a set of transformation rules that establish a correspondence between map and DFD. The result of this coupling is an approach that supports the analysis and design of systems.

Despites similarities in the notation, the authors conclude that there are some differences between DFD and map because they represent two different views of a system. map adopts an intentional view focusing on requirements specification, while DFD adopts a functional view focusing on functions specification. An overview of the approach presented in [9] is given in the Fig. 1.

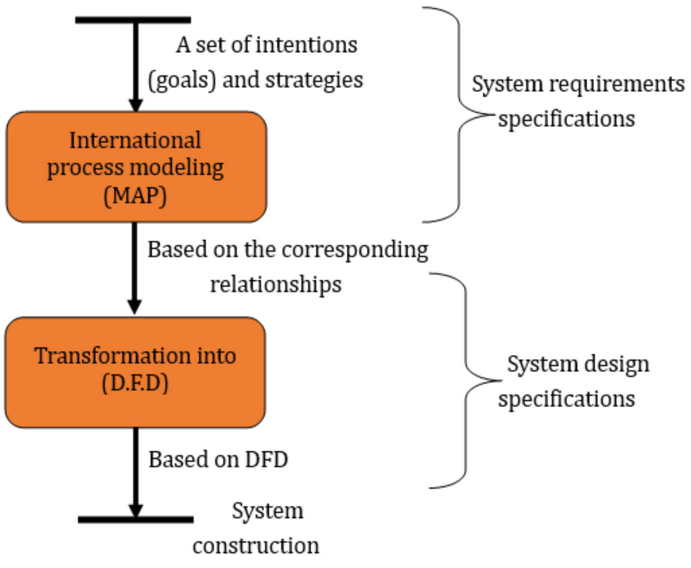
A set of intentions
(goals) and strategies
International process modeling (MAP)
System requirements specifications
Based on the corresponding relationships
Transformation into (D.F.D)
System design specifications
Based on DFD
System construction

Fig. 1. Top-down approach: From map to DFD.

Our approach follows the similar objective but, into the opposite direction, going from a DFD to a map. A reverse transformation is applied to the DFD model to obtain the map model.

Next, we provide a short DFD and map overview. The main purpose of this overview is to introduce the basic concepts of both modeling formalisms.

### 2.1. Review Stage

Data Flow Diagram (DFD) [2] is one of the most popular structured analysis tools. It can seem like a Structure chart, State machine or some kind of structured approach tools. DFD is used for modeling, analyzing, and describing any system at different levels of abstraction. DFD is graphically represented as a directed graph where the nodes of graph are processes, and its arcs are labeled with data. A node may be an external entity, a process, or a data store. An arc represents the data flow. The DFD is refined until each process performs a simple function, which is easy to implement. There are several notations of the DFD. In the following, four types modeling concepts are explained.

*External entities:* are those things that are identified as needing to interact with the system under consideration. Processes: are defined as actions performed by the system which have input and output. A process can be decomposed to several levels of detail.

*Data flow:* a data flow represents an input or the output of data in the system. It is used to connect one process to another.

*Data stores:* are defined as repertories where data may be stored.

DFD was chosen to represent functional view of the system because it is the most important diagram used in structured approach and widely accepted in the software engineering domain. A practical example of DFD usage will be shown in the Section 5.

### 2.2. Map

Map [5] is a well-known intention/strategy-oriented requirement engineering approaches allows eliciting and describing requirements of a system in a set of directional graphs (namely maps) composed of intentions and strategies. The intentions represent the nodes of the graph and strategies are arcs linking those nodes. In Fig. 1, the strategy Sik represents the flow from the intention Ii to the intention Ik. Each map has two specific intentions (Start and Stop) that represent the starting and ending points of the map. A map consists of two or more sections. A section is a triplet composed of source intention, target intention and strategy. For example, in Fig. 2 the triplet <Ii, Ik, Sik> represents a section. Refer to Ref. [5] for a more thorough description of the map. As an example, considers the map of Fig. 2.

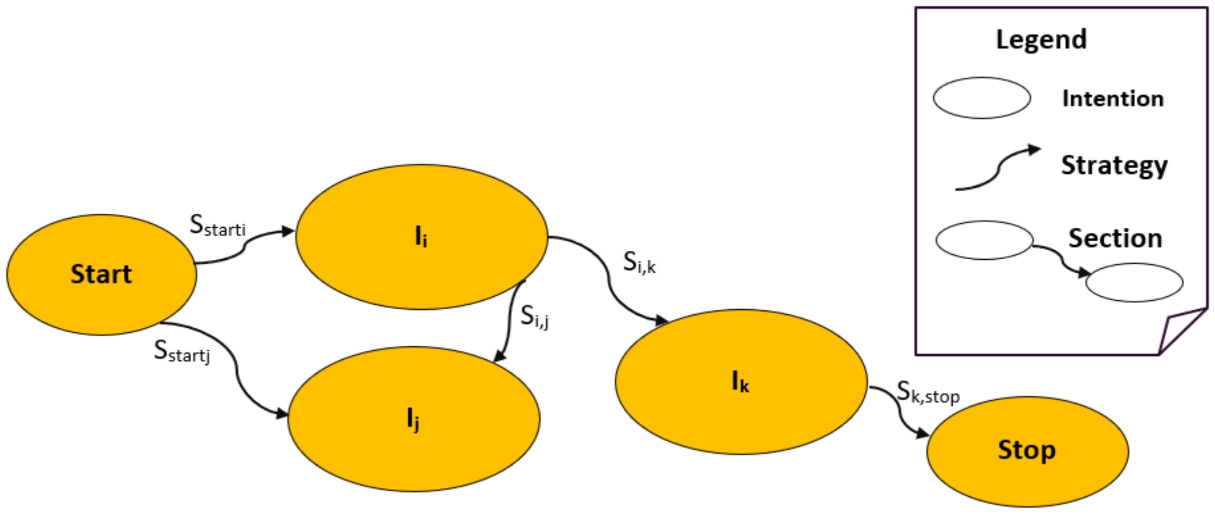

Fig. 2. An example of the map.

Map can be compared to the existing requirements engineering approaches like KAOS [11] and i* [7], but map has certain advantages over them. One of these advantages is that it considers the strategies which can be used to attain these intentions.

## 3. Bottom-up Approach: From DFD to Map

This section aims to define our approach for mapping between the main concepts of the DFD and map models. For this, a set of transformation rules to obtain a map model from a DFD model is described below:

- **1st rule:** Each DFD will be represented as a map.
- **2nd rule:** in DFD a process is refined by one or more sub-process like it is in map that a section may contain sub-section. Both have the same meaning and can be mapped directly to each other.
- **3rd rule:** following the rule 2, hierarchies of DFD can be translated into hierarchies of map. When the process is decomposed into more than one level, it will be transformed into a sub-section.
- **4th rule:** in DFD it is possible to have multiple data inputs and outputs for a process. Thus, the data flow to or from a process can be mapped into a strategy in the map model.
- **5th rule:** Sequential data flow in DFD they become a sequence of sections in the map model.
- **6th rule:** Data store is created to store data for the later use. However, there is no map construct to express the stores.
- **7th rule:** in DFD external entities are components sits outside the system but interact with system. External entities are not modeled in the map model, but they can be inferred by searching parameter associated to the intentions.

In the next section we will discuss the how data flow diagrams can be transformed into map.

## 4. An Example: Account Management in a Bank

Our example refers to the account management process in a bank. A customer presents a request to open a new account to the bank. Once the client is registered, he can withdraw funds from his account or deposit funds into his account. If the customer is performing transaction to withdraw or deposit funds, it needs to provide an account number and the amount to be withdrawn or to deposit to the clerk in charge of his account. The customer's information (such as name and address) is stored in Customer details data store. The information about the accounts and amount of withdrawal is retrieved from Bank accounts data store.

In the following, we are representing the level 1 data flow diagram of the account management system which is a refinement of the context data flow diagram (Fig. 3). This DFD consists of three processes Open new account, Close account and Deposit funds and Withdraw funds.

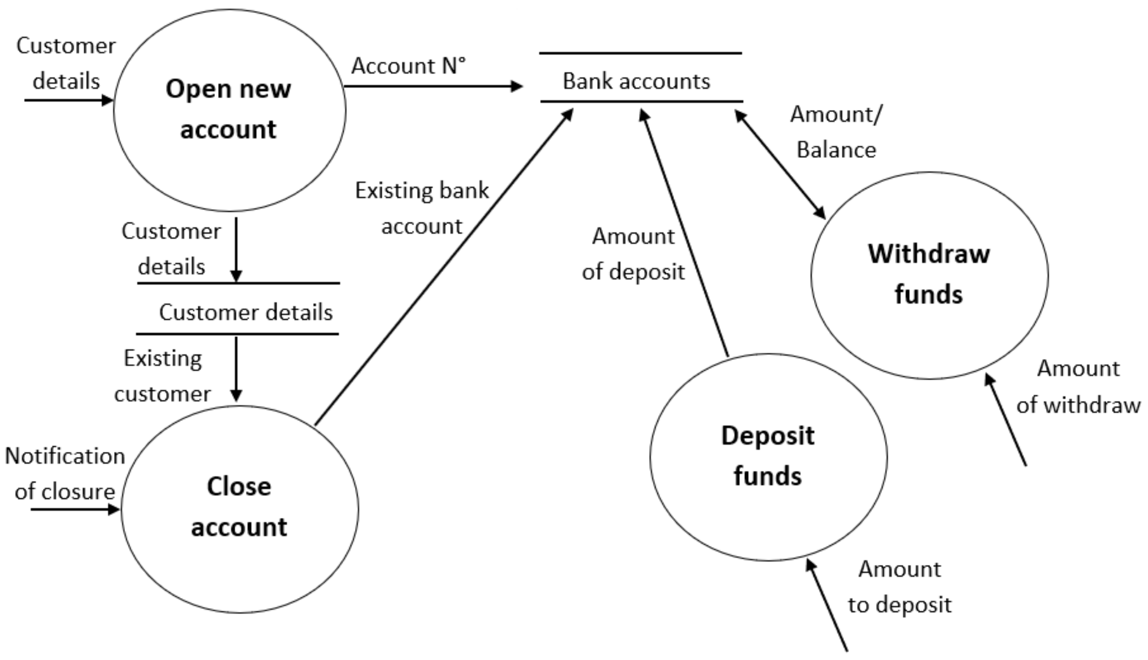

Fig. 3. The level 1 DFD-Account Management System.

## 4.1. DFD of Account Management Process

In the above level 1 DFD, there are two processes that can further be refined into level 2 DFD: Withdraw funds process and Deposits funds process. Fig. 4 shows a refined view of Withdraw funds process. In this process, the bank checks the status of balance in the customer's account and then it sends this information to Update balance process. Finally, a statement is provided to the customer.

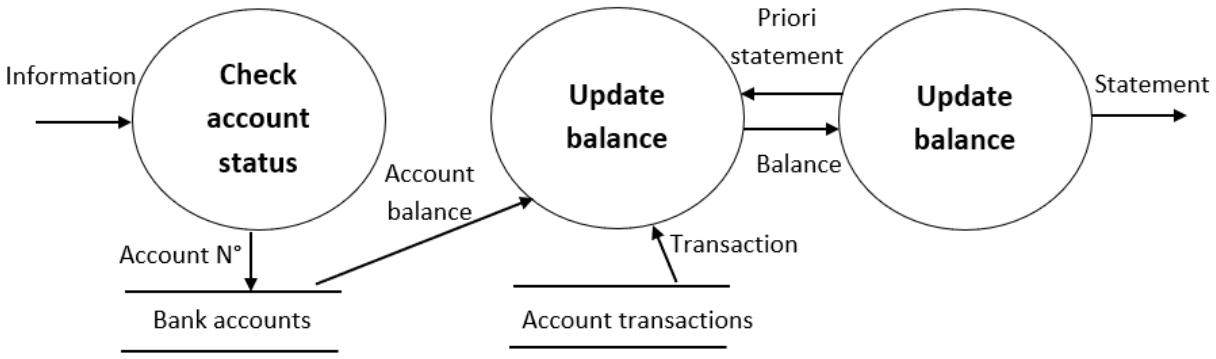

Fig. 4. DFD refining withdraw funds process.

## 4.2. Map of Account Management Process

By applying the mapping rules, we will show how the map models can be obtained from DFD models. Fig. 5 presents the map model obtained after performing the proposed mapping rules.

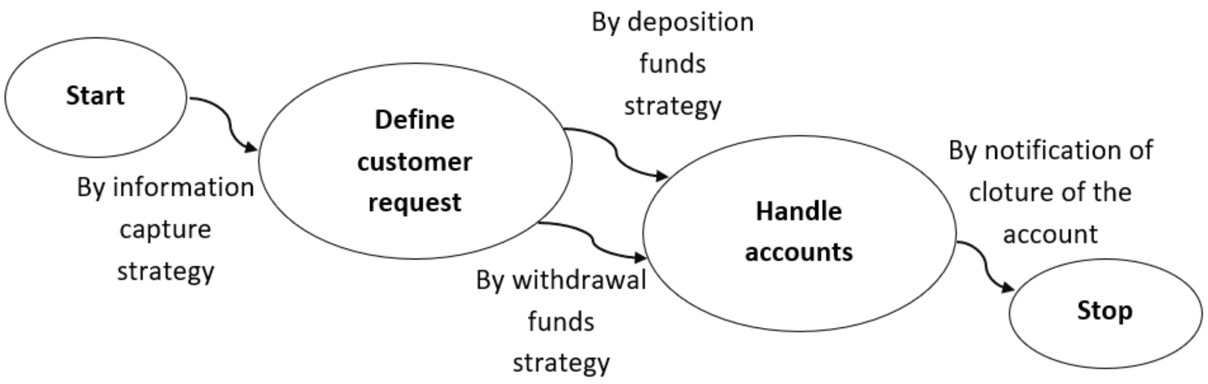
Start
By information capture strategy
Define customer request
By deposition funds strategy
By withdrawal funds strategy
Handle accounts
By notification of cloture of the account
Stop

Fig. 5. Map model from DFD model.

Table 1 illustrates the mapping of the account management process into sections.

Table 1. The Mapping of Account Management Process

| Process | Section |
|---|---|
| Open new account | <Start, Define customer request, By information capture strategy> |
| Deposit funds | <Define customer request, Handle accounts, By deposit funds strategy> |
| Withdraw funds | <Define customer request, Handle accounts, By withdraw funds strategy> |
| Close account | <Handle accounts, Stop, By the notification of cloture of the account> |

Fig. 6 gives the map model of the refined process Withdraw funds.

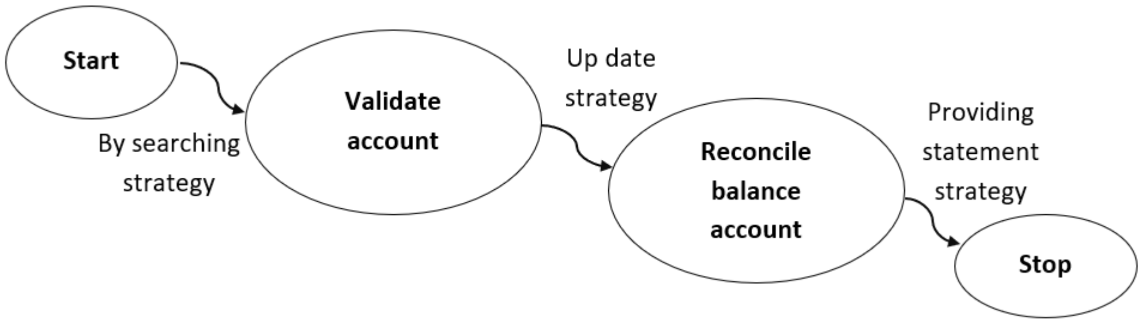

Fig. 6. The map of the refined process.

Table 2 illustrates the mapping of the refined process Withdraw funds into sections.

Table 2. The Mapping of Withdraw Funds Process

| **Process** | **Section** |
|---|---|
| Check account status | <Start, Validate account, By searching strategy> |
| Update balance | <Validate account, Reconcile account balance, Update strategy> |
| Provide statement | <Reconcile account balance, Stop, Providing statement strategy> |

## 5. Conclusion

The main goal of the paper was to suggest an initial version of an approach to obtain map models from DFD models. The proposed approach aims to ensure a good connection between these models.

The mapping process proposed in this paper can be summarized as follow: First, we have analyzed the elements in DFD and the elements in map that can be mapped from DFD. Based on the corresponding relationships between DFD and map, the mapping rules from DFD to map are proposed. Finally, an example is used to illustrate our approach.

While defining mapping between DFD and map constructs, we observed that there are some constructs in DFD that do not have an equivalent in map. This is the case, for example, of the data stores in DFD. We discovered in this mapping a lack of full correspondence between the models. Therefore, this approach needs to be extended to deal with the "non-mapping" situations. This is subject for future work.

## Conflict of Interest

The author declares no conflict of interest.

## References

[1] Olle, T. W., *et al*. (1991). *Information Systems Methodologies: A Framework for Understanding*. Addison Wesley Pub. Co.

[2] DeMarco, T. (1979). *Structured Analysis and System Specification*. Yourdon Press Computing Series, Prentice Hall, Englewood Cliffs, NJ.

[3] Jacobson, I. (1995). The use case construct in object-oriented software Engineering. In John M. Carroll (Ed.), *Scenario-Based Design: Envisioning Work and Technology in System Development*, John Wiley, 309–336.

[4] Rolland, C., Prakash, N., & Benjamen, A. (1999). A multi-model view of process modelling, *Requirements Engineering Journal, 4(4)*, 169–187.

[5] Rolland, C., Souveyet, C., & Acour, C. B. (1998). Guiding goal modeling using scenarios. *IEEE Transactions on Software Engineering, 24(12)*.

[6] Tan, H. B. K., Yang, Y., & Blan, L. (2006). Systematic transformation of functional analysis model in object oriented design and implementation. *IEEE Transaction on Software Engineering, 32(2),* 111–135.

[7] Dardenne, A., Lamsweerde, A., & Fickas, S. (1993). Goal-directed requirements acquisition. *Science of Computer Programming, 20*, 3–50.

[8] Yu, E. (1997). *Towards Modelling and Reasoning Support for Early-Phase*. (pp. 226–235).

[9] Tran, T. N., Khan, K. M., & Lan, Y. C. (2004). A framework for transforming artifacts from Data Flow Diagrams to UML. *Proceedings of the 2004 IASTED International Conference on Software Engineering*.

[10] Truscan, D., Fernandes, J.M., & Lilius, J. (2004). Tool support for DFD-UML based transformation. *Proceedings of the IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04)*.

[11] Prakash, N., & Rolland, C. (2006). Systems design for requirements expressed as a map. *Proceedings of IRMA 06*.