Modeling of an Enterprise and Information System: Process Specification Based on the Flow of Things

Sabah Al-Fedaghi
Computer Engineering Department
Kuwait University
P.O. Box 5969 Safat 13060 Kuwait
+96599939594
sabah.alfedaghi@ku.edu.kw

Yousef Alduwaisan
IT Department
Ministry of Defense
Kuwait
+96599585874
ykalduwaisan@mod.gov.kw

ABSTRACT

Current modeling of *enterprise and information systems* is based on diverse methods such as function-orientation, data-orientation, process-orientation, object-orientation, and object/process-orientation. Other emerging modeling methods include the ontology capture method and Business Process Modeling Notation (BPMN). These approaches have been criticized for lack of execution environment into which events that cause change could be incorporated. Such a topic is important in the study of dynamic behavior of systems for both analysis and control. This paper further develops a new approach oriented toward *things that flow*. Specifically, the paper concentrates on the study of *process specification* in analysis and design of enterprise/system architectures in order to most effectively facilitate their control. Here we produce a single, integrated diagrammatic representation that uniformly incorporates structural and behavioral aspects into an underlying conceptual description. The viability of the model is demonstrated by applying it to a case study of services provided by an existing organizational unit.

CCS Concepts

- Software and its engineering~Model-driven software engineering
- Software and its engineering~System modeling languages

Keywords

Process specification; conceptual modeling; Data Flow Models; BPMN; UML

1. INTRODUCTION

The process of *modeling an enterprise and information system* provides a framework for solving problems in the analysis, design, and development of the system. *Model*, in this context and in terms we define later, refers to a representation of reality that involves *Things* and their flows and “operations” on them within *spheres* that demarcate zones of work activities. Things, in this view include, for example, people, data, information, software, processes, equipment, products, time, events, and raw materials. This model conceptualization is developed in this paper.

Currently, under this topic of modeling of enterprise and

information systems, methods are said to be oriented toward a central notion such as function-orientation, data-orientation, process-orientation, object-orientation, and object/process-orientation. These methodologies are accompanied by specification tools as Data Flow Diagram (DFD), IDEF, Entity-Relationship Diagram (ERD), UML and OPM diagrams, ontology capture method, and Business Process Modeling Notation (BPMN). Similar to these approaches, this paper advances a diagrammatic apparatus, the Flowthing Machines (FM) model, that is *oriented toward things that flow in flow machines*, for the purpose of analysis and design of an enterprise and information system at the physical and data levels.

We claim that this approach sheds new light on *process specification* in modeling of enterprise and information systems, including how to approach process structure and behavior. The model also stresses conceptual description (rather than technical notions) that can be used to facilitate communication among stakeholders in the modeled system.

In this paper, *process specification* refers to a diagrammatic description of the bottom level of a system, or its “flow machines.” It defines what must be done in order to, in Yourdon’s [16] words, “transform inputs into outputs”; however, this transformation of input into output reflects the classical input-process-output (IPO) concept of a process. Alternatively, in this paper, *process specification* refers to what must be done in order to control the flow of things through stages of *create*, *process* (in a more restricted sense), *release*, *transfer* (input/output), and *receive*.

A variety of tools are available for producing a process specification. In Yourdon’s [16] words, any method should satisfy the following requirements:

- The process specification must be expressed in a form that can be verified by the user and the systems analyst.
- The process specification must be expressed in a form that can be effectively communicated to the various audiences involved. [16]

1.1 Some motivation for FM

Most of the variously oriented approaches mentioned above have been heavily scrutinized and critiqued. Take for example BPMN, described as representing “high-level graphical representation of business processes easily understood by business analysts, and especially useful in communicating business requirements” [11]; however, according to Dijkman et al. [8],

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICGDA '18, April 20–22, 2018, Prague, Czech Republic.

©2018 Copyright is held by the owner/author(s).

ACM ISBN 978-1-4503-6445-4/18/04.

<https://doi.org/10.1145/3220228.3220246>

The mix of constructs found in BPMN makes it possible to obtain models with a range of semantic errors... [and] the static analysis of BPMN models is hindered by ambiguities in the standard specification and the complexity of the language. The fact that BPMN integrates constructs from graph-oriented process definition languages with features for concurrent execution of multiple instances of a subprocess and exception handling, makes it challenging to provide a formal semantics of BPMN. Even more challenging is to define a semantics that can be used to analyze BPMN models.

Certainly, it is not possible to produce a detailed scrutiny of these approaches, but in order to give some justification for FM development; it seems acceptable to examine one problem such as that of *process variability modeling*. According to La Rosa et al. [12],

It is common for organizations to maintain multiple variants of a given business process. Conventional business process modeling languages do not explicitly support the representation of such families of process variants. Existing approaches are characterized by the fact that they extend a conventional process modeling language with constructs that make it able to capture customizable process models. At present, the configuration of multi-tenant systems is manual and resource-intensive due to the large number of configuration points typically offered by such systems.

While research is ongoing into customizable process models [9], [15], new approaches such as FM may contribute to solving this problem through its notion of event machines in which static descriptions and behavior specifications are superimposed on each other, as we discuss later.

1.2 Additional motivation

The modeling of an enterprise and information system has become a challenging task for complex systems. One problem is that people find it difficult to conceptualize some of the basic notions in the field, such as *process*, *behavior*, and *events*. This problem can be relieved by “the cooperation of stakeholders from IT management, software engineering and systems operation by providing appropriate model-based” [7]. The FM model views these notions in term of *flows* that circulate through diverse fields, such as, for example, supply chain flows, money flows, and data flows in communication models. The basic construct in FM is a *thing*, in the context of a diagrammatic (abstract) *machine* or *pattern*. The machine represents a *process* in the general sense, a sequence of steps. The nontechnical nature of FM makes it suitable for communication of requirements, design, and implementation to stakeholders lacking a technical background.

For the sake of a self-contained paper, section 2 briefly describes the basics of FM, which has been utilized in many applications [1–6]. Section 3 provides an example of the static and dynamic aspects of the model. Sections 4 and 5 contrast FM with DFD and ER models. After explaining the features of the FM model and making these contrasts, we demonstrate its feasibility in a real environment by modeling procedures of an actual IT department of a government ministry.

2. FLOWTHING MACHINE (FM)

This paper promotes a flow-based orientation in modeling by use of FM, and this section briefly reviews the foundation of this approach; the example given here is a new contribution.

The FM model is a diagrammatic language that begins with flowthings (hereafter, *things*), defined as what can be created, released, transferred, received, and processed in passage through stages in a *flow machine* (Figure 1). This flow begins when a thing is created by the machine or imported from another machine.

Figure 1. Flow machine

The notion of flow here entails transition or realization of change as well as movement and positioning. *Creation* is the appearance or introduction of a thing into the system. The rest of the flow is succession from one stage to the next until final Release and Transfer to another machine or outside the system. Such flows are specified in drawings analogous to traffic flows shown on a city map. Here, as discussed later, dynamic behavior can be controlled in terms of events that describe the behavior of the system. Continuing the analogy, city streets are streams of flow of cars, pedestrians, and more. For example, a one-lane street may appear on a map to allow driving in both directions, but in reality, at the level of events, the road allows traffic to flow in only one direction during AM hours and in the other direction during PM hours.

Flow is often thought of as physical movement, but in FM, it can be much more than that. It is a notion that captures conceptual movement in thought, sensation, being, and doing. The modeler builds a conceptual construct and also conceptual “movement” that we call flow. Thus, a physical house “flows” from one sphere to another when a transition in ownership occurs, e.g., from an individual to a bank, and a car being assembled in a factory is affected by various flows of robots and workers simultaneously when it is processed, e.g., one is inserts glass while another attaches tires. Flows might be fast or slow, parallel or sequential, physical or digital or mental, but involve only certain set stages, shown in Figure 1 and described as follows:

Arrive: A thing reaches a new machine.

Accept: A thing is approved to enter a machine. If arriving things are always accepted, Arrive and Accept can be combined as a *Receive* stage.

Process (change): The thing goes through some kind of transformation that changes it without creating a new thing.

Release: A thing is marked as ready to be transferred outside the machine. Note that things can be released from a given system without being transferred, as in the case of goods ready for shipment, or emails in queue when a channel is down.

Transfer: A thing is transported from/to outside the machine.

Create: A new thing appears in a machine.

Flow machines use the notions of spheres and subspheres. These are constructs (mental products) of machines and submachines. Multiple machines can exist in a sphere if needed. A sphere can be, for example, a person, an organ, an entity (e.g., a company, a customer), a location (a laboratory, a waiting room), a communication medium (a channel, a wire). A machine is a

subsphere that embodies the flow; it itself has no subspheres. This notion of sphere is taken from cognitive linguistics, where an idea is treated in terms of complex units associated with other entities or other forms of association.

FM also utilizes the notion of triggering. *Triggering* is the activation of a flow, denoted in FM diagrams by a dashed arrow. Triggering occurs in a dependency relationship among flows and parts of flows. A flow is said to be triggered if it is activated by another flow, or by another point in the flow. Triggering can also be used to initiate events such as starting up a machine. Multiple machines can interact by triggering events related to other machines in those machines' spheres and stages.

3. Example

This section illustrates general features of the FM diagram through an example from the literature. Shen et al. [14] describe a typical loan approval service system involving a *Customer* sending a loan request to a financial organization. The entire process consists of two services: "Risk Assessment" and "Loan Approval." In Risk Assessment, the service is provided with information needed to evaluate customer's reputation and loan amount. Only when the risk assessment satisfies the requirements specified for Loan Approval can the application be approved; otherwise, the loan request is rejected.

According to Shen et al. [14], much effort has been expended to model services based on business processes as graph-based workflows, e.g., Petri nets. The authors present the diagram shown in Figure 3, with the process depicted in a Control Flow Graph (CFG).

Figure 3. CFG depicting steps in Loan Approval process (redrawn, partial from [14])

Figure 4 shows a preliminary FM diagram for this loan approval case, with a complete representation shown in Figure 5. In FM, we can identify all the spheres and subspheres of this case, along with their flows, in the figure. First, the customer makes (creates) a request for an application form (circle 1), which flows to the financial organization (2), which sends an application to the customer (3). The customer fills out the application (4) and it flows back to the organization (5), where it is analyzed for two subspheres (components): Amount (6) and Profile (7). The amount is checked against a permitted value (8). The amount and profile flow to Risk assessment (9 and 10) where they are processed (11). If Amount = profile (the way it is expressed in

Shen et al.'s [14] problem), then this produces an "OK" (12) (according to our understanding of the problem). Meanwhile, if Amount < permitted (according to a stored value), then this triggers (14) creating information that Risk is low (15).

Figure 4. Flow and spheres in the FM representation

Figure 5. FM representation of the Loan Approval case

The satisfied conditions of low Risk (15) and approved Risk assessment (12) together (16 and 17) trigger the creation of a Decision (18) that triggers creation (19) and transfer of a Reply to the customer (20).

Note that the stages of the various flow machines help in identifying the spheres and subspheres shown in Figure 4. Flow machines are spheres that do not include subspheres. All the “higher-level” spheres are specified in terms of machines. In the diagram, the boxes enclosing some machines are omitted for simplification since their flows are known, e.g., application stages in the Financial Organization sphere are not enclosed in a box. Note that Figure 4 can be produced from Figure 5 and vice versa. The representation can be further simplified as shown in Figure 6.

Figure 6. Further simplification of the FM representation

4. Process specification levels

This section contrasts the FM approach with Data Flow Diagrams (DFDs), especially in the aspect of phases of development that involve, in DFD, refinement through levels.

4.1 Context-level specification

DFD is widely used in the analysis and design of systems. It is “a modeling tool that allows us to picture a system as a network of *functional processes*, connected to one another by ‘pipelines’ and ‘holding tanks’ of data” [16]. DFDs illustrate “functions” and how data are exchanged in a system [13]. In the example DFD of a food ordering system, the steps in decomposition begin with a context diagram (Figure 7).

Figure 7. Context diagram of a food ordering system (redrawn from [13])

Figure 8 shows a preliminary FM representation corresponding to this DFD context diagram. The customer creates an order (1) that flows to the services part of the restaurant (2), which sends it to the kitchen (3). This triggers (4) making food (5) that flows to the services unit (6) to be delivered to the customer (7). Delivering

the food to the customer triggers (8) creation of a report (9) that flows to the manager (10). Note that the colored boxes indicate flow machines. Figure 9 shows these machines and their spheres. Figure 10 shows the FM representation after the stages of various machines have been added.

4.2 Level-0 specification

The so-called DFD Level-0 diagram as given by Li and Chen [13] provides more detail and is developed as shown in Figure 11. In the figure the management report is broken down into more detailed processes.

Figure 8. “Context” FM diagram of food ordering system

Figure 9. Machines of different spheres in the “context” FM diagram

Figure 10. FM diagram of food ordering system with

Figure 11. Level-0 diagram of food ordering system (redrawn, partial from [15])

Figure 11 shows a level-0 DFD for the food ordering system. In its corresponding FM diagram (Figure 12), continuing from the point at which the food company delivers the food to the customer, data are generated (1) about *Sold food* that include, we assume, the name of the sold item (2) and the amount paid (3). The data are fed (4) into the information system, which formats it (5) and inserts it (6) into the *Sold goods* file (7). At a certain point of time (e.g., end of day), the Sold goods file is processed (8) to create the *Daily sold goods amounts* in the management report (9). Additionally, the whole Sold goods file is sent (10) to a procedure that updates the *Inventory* file.

This procedure also receives the inventory file (11), processes the two files (12), and sends the new inventory file back to the inventory system (13). The inventory system processes (14) the new file and triggers the creation (15) of *Daily inventory depletion* amounts in the management report.

Note how the physical flows are separated from data processing (*Information System* sphere). The information system description is almost an abstraction of a programming language:

Read transaction data (circle 4)

Format data (5)

Insert data in sold goods file (6–7)

Calculate Daily goods sold amounts (8–9)

Update inventory file (10–13)

Calculate Daily inventory depletion amounts (14–15)

5. Specification of dynamic behavior

In such systems as SysML, after the main use cases are specified, a top-level system behavior model is developed and analyzed using different types of diagrams such as sequence and state diagrams. In FM, modeling of behavior takes place in a phase that occurs after the static description (such as Figure 12) is created and involves modeling the “events space” in which an event happens.

An *event* is specified by its *spatial area* or subgraph, its *time*, the event’s own *stages*, and possibly by other descriptors such as intensity or extent (strength).

Figure 12. FM diagram with details of level-0

Figure 13 shows the (non-basic) event *A customer sends an order to the services of the food company*. Similarly, in Figure 14 we identify 13 events previously shown in Figure 12. For simplification, the times of event stages are not shown.

- Event 1 (E1): *A customer sends an order to the food Co.*
- Event 2 (E2): *The order reaches the kitchen and the food is cooked*
- Event 3 (E3): *The food is delivered to the customer*
- Event 4 (E4): *Data are created*
- Event 5 (E5): *Data are received by the information system*
- Event 6 (E6): *Data are formatted*
- Event 7 (E7): *Data are inserted into a file*
- Event 8 (E8): *Sold goods value is put in the report*
- Event 9 (E9): *Sold goods file is sent to update the inventory file*
- Event 10 (E10): *The inventory file is updated*
- Event 11 (E11): *Daily inventory depletion is put in the report*
- Event 12 (E12): *The report is created*
- Event 13 (E13): *The report is received by the manager*

Figure 13. The event *A customer sends a order to the Food Co.*

6. Specification at Stored Data Layout

At this point in a discussion of process specification, it is important to demonstrate FM features for describing the data layout as modeled in an entity relationship (E-R) diagram. This diagram is a network model that describes the *stored data layout* of a system at a high level of abstraction. It highlights relationships between data stores shown on the Data Flow Diagram. It complements the DFD, which models the *functions* performed by a system, and the state-transition diagram, which models the time-dependent *behavior* of a system [16].

Why should we be interested in a data model of a system? Higher level executive users in an organization are often more concerned with the data: What data do we need to run our business? How are the data related to other data? Who owns the data? Who is allowed to access the data? [16]

From the FM point of view, the interesting aspect is binding the description of the *stored data layout* of a system at a high level of abstraction to a physical model of activities. Within the organization, the database administration group (DBA) is usually located within the IT department (while the data administration group is not necessarily so located), and its job is to ensure that computerized databases are organized, managed, and controlled effectively. The entity-relationship diagram is an effective modeling tool for communicating between the two groups [16].

A typical ER diagram is given by Yourdon [16] and is shown in Figure 15.

Figure 14. Events of the FM Diagram with details of level-0

Figure 15. Typical ER diagram (redrawn, partial from [16])

Figure 16 shows the corresponding FM representation. First, the customer sends an order (1) for a book accompanied by his/her ID (2 - assumption). These flow to the sales representative (3). The arrival of the order request (assumedly to the sales representative) triggers (4) registering it in the information system (e.g., giving it a time stamp).

The arriving order request is also processed (5) to trigger (6) release of the book (7) to be given to the customer (8). Release of the book also triggers (8) feeding its serial number into the information system (9). We also assume that the order time stamp, customer ID, and sales rep ID are fed (9, 10, and 11) into the information system. Such assumption is utilized to illustrate the relationship among Customer, Sales representative, and Book. (This does not restrict the model since any type of data can be used instead of these).

At this point, we enter the sphere of *data structure*, where, we assume, a “record” is constructed from the order time stamp, customer ID, sales rep ID, and book serial number (12). The record flows (13) to be inserted into the sales file (14).

Figure 17 shows the FM representation in terms of the relationship among Customer, Sales representative, and Book in reality and in the context of data spheres.

Figure 16. FM representation corresponding to a typical ER diagram

Figure 17. The involved relationships in reality and in the information system spheres

7. CASE STUDY

This section reports the results of applying FM as a process modeling tool to currently existing processes in the IT department of a Kuwait government ministry (where the second author is employed). Note that each process modeled in this section represents a cyber-physical system that integrates computation and physical processes, where users interact utilizing both their digital identities and physical interaction.

Every employee in the IT department is eligible to apply for various *services* and transactions within the department, including annual vacation, sick leave, overtime compensation, and personal data updating. Figure 18 shows two processes; the first occurs when an employee asks for specific software, and second, when certain software is a mandatory release to all employees' computers.

Figure 18. FM representation of: requesting specific software, and a mandatory release to all employees' computers.

The figure includes two main spheres and two external ones. One is the department itself (circle 1), which includes the employees (2) and their requests until they reach the IT manager (4) who is responsible for accepting or rejecting requests; the second main sphere is the system (3) in which the administrator (14) resides and controls which software is to be released to which employee. The external spheres include the previously mentioned IT manager, and the Software firm (5) which releases software and updates into the system.

The Employee (2) creates a request for a specific software program that flows (7) to the system to be processed, and a form (8) is delivered to his computer (9), which he then fills out, prints (10), and hands to the Head of Department (11) who can either accept (12) or reject (13) the request. Accepted requests are sent to the IT manager (4) then flow to the system administrator (14), who accesses a console to issue the specified software from the uncommon software database (15) back to the employee's computer for installation (17).

The second process occurs when an update or certain software is released onto all computers. First, the updating system (18) checks for updates by comparing a list of all existing software against the software at the Software Firm; if a new version exists, it is released from the firm's database into the updating system, which determines whether the software is uncommon (15) or common (16). The software is then processed at the common software database to cycle through all employees (19), identify their computers, and release the software for them to install.

Finally, in the case of new uncommon software, the system notifies (20) the employees of the new software, and their computers receive a notification (21) of it. They can then issue a request (6) for that software.

8. CONCLUSION

According to Hay [10], three aspects of an organization that are the subject of most modeling techniques are:

- The *objects* that it manipulates: the things of significance about which the organization must hold and maintain information.
- The *functions* or *activities* performed by the business.
- The *events* in the world that cause those functions to be performed.

In FM, a flow machine that *Creates, processes, releasees, transfers*, and *receives* things seems to be an appropriate tool for these techniques. The paper demonstrates FM as a unifying approach to modeling systems and demonstrates its capabilities by modeling portions of an actual system. A weakness of FM is lack of a computer-based design support system in this phase of development because of deficiencies in the current research environment (e.g., no PhD program); thus, a goal of this paper is to attract researchers to the potentials of this approach.

9. REFERENCES

- [1] Al-Fedaghi, S. 2017. Business process modeling: blueprinting. *Int. J. Comput. Sci. Inform. Security*, 15, 3, 286–291.
- [2] Al-Fedaghi, S. 2016. Designing home circulation: application to smart homes. *Int. J. Smart Home*, 10, 12.
- [3] Al-Fedaghi, S. 2013. Flow-based conceptual representation of problems. *J. Theor. Appl. Inform. Technol.*, 58, 2, 347–356.
- [4] Al-Fedaghi, S. 2008. Modeling communication: one more piece falling into place. The 26th ACM International Conference on Design of Communication (SIGDOC 2008), Lisboa, Portugal, Sept. 22–24, 2008.
- [5] Al-Fedaghi, S. 2007. How sensitive is your personal information? The 22nd ACM Symposium on Applied Computing (ACM SAC 2007), Seoul, Korea, March 11–15, 2007.
- [6] Al-Fedaghi, S., and Alahmad, H. 2018. Integrated modeling methodologies and languages, ACM 12th International Conference on Ubiquitous Information Management and Communication, January 5–7, 2018, Langkawi, Malaysia.
- [7] Breu, R., Agreiter, B., Farwick, M., Felderer, M., Hafner, M., and Innerhofer-Oberper, F. 2010. Living models: ten principles for change-driven software engineering. *Int. J. Softw. Inform.*
- [8] Dijkman, R. M., Dumas, M., and Ouyang, C. (n.d.) *Formal Semantics and Automated Analysis of BPMN Process Models*. Technical Report Preprint 5969. Queensland University of Technology.
- [9] Fehling, C., Leymann, F., Schumm, D., Konrad, R., Mietzner, R., and Pauly, M. 2011. Flexible process-based applications in hybrid clouds. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD)*. IEEE, 81–88.
- [10] Hay, D. C. 1997. Object Oriented Data Modeling: Entity Life Histories. Essential Strategies, Inc.. <http://www.essentialstrategies.com/publications/objects/elh.htm>
- [11] Ko, R.K.L., Lee, S.S.G., and Lee, E. W., 2009. Business process management (BPM) standards: a survey. *Bus. Process Manage. J.*, 15, 5.
- [12] La Rosa, M., Van der Aalst, W.M.P., Dumas, M., and Milani, F. P. 2017. Business process variability modeling: a survey. *ACM Comput. Surv.*, 50, 1.
- [13] Li, Q., and Chen, Y. 2009. Data Flow Diagram. In *Modeling and Analysis of Enterprise and Information Systems*, Springer, Berlin, 85–97.
- [14] Shen, J., Beydoun, G., Low, G., and Wang, L. 2014. Aligning ontology-based development with service oriented systems. *Future Generation Comput. Syst.*, 32, 263–273.
- [15] Van der Aalst, W. M. P. 2011. Business process configuration in the cloud: How to support and analyze multi-tenant processes? In *Proceedings of the 9th IEEE European Conference on Web Services (ECOWS)*. IEEE, 3–10.
- [16] Yourdon, E. 2006. *Just Enough Structured Analysis*. Yourdon Press, New York, pp. 147–166.