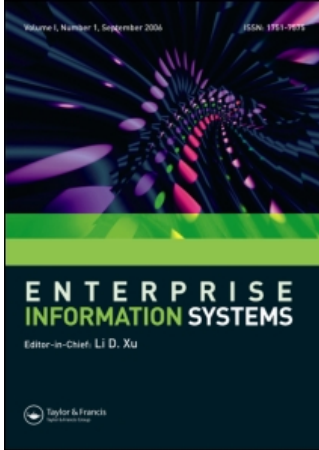


This article was downloaded by:[Barjis, J.]  
On: 24 November 2007  
Access Details: [subscription number 787091454]  
Publisher: Taylor & Francis  
Informa Ltd Registered in England and Wales Registered Number: 1072954  
Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



## Enterprise Information Systems

Publication details, including instructions for authors and subscription information:  
<http://www.informaworld.com/smpp/title~content=t748254467>

### Automatic business process analysis and simulation based on DEMO

J. Barjis<sup>a</sup>

<sup>a</sup> Department of Computer Information Systems, University of Wisconsin at Stevens Point, Stevens Point, WI 54481, USA

Online Publication Date: 01 November 2007

To cite this Article: Barjis, J. (2007) 'Automatic business process analysis and simulation based on DEMO', Enterprise Information Systems, 1:4, 365 - 381

To link to this article: DOI: 10.1080/17517570701646590

URL: <http://dx.doi.org/10.1080/17517570701646590>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.informaworld.com/terms-and-conditions-of-access.pdf>

This article maybe used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

## Automatic business process analysis and simulation based on DEMO

J. BARJIS\*

Department of Computer Information Systems, University of Wisconsin at Stevens Point,  
Stevens Point, WI 54481, USA

*(Received 22 June 2007; in final form 1 September 2007)*

Enterprise process modelling as a core area of the Enterprises Information Systems (EIS) research has long attracted theoreticians of new concepts, designers of artefacts, and practitioners of modelling. As a result a vast number of modelling methods have been generated. However, each method is challenging because they lack certain aspects. First, these methods view processes as a flowchart, undermining the deep (nested) structure of processes. Second, most of these methods are based on informal or semi-formal representation, failing direct model checking and formal analysis without further translations and mapping procedures. Finally, because the majority of these methods originated from a computing perspective, their focal point is control-flow rather than interaction of social actors. In reality, an enterprise is a complex socio-technical system that includes interaction of technical and social components. In this paper, we discuss an innovative method that will address some of these challenges and provide an improved tool for this purpose. We base our method on the field-proven DEMO theory and methodology, where emphasis is made on the interactions between social actors. The proposed method encompasses a set of graphical notations and constructs based on the Petri nets semantics that allows direct model analysis and simulation.

*Keywords:* Enterprise process modelling; Business systems analysis; Process simulation; Requirements elicitation; DEMO methodology; Design science; Modelling technique; Enterprise information systems

### 1. Introduction

A thorough study of enterprises information systems (EIS) requires a broad grasp of the organizational and social context, in which the envisioned EIS will have to operate and enable business processes. An EIS is not machinery alone or a collection of information technology (IT) components, but a complex socio-technical phenomenon, where business process and users are the other two main components.

In the past two decades, an innovative trend emerged advocating a new theory of IS design, that indicates better understanding of business processes, through the study of the interactions (communication) enacted in the people behaviour as they carry out the organization's mission. Thus, an accurate EIS design entails

---

\*Email: JBarjis@uwsp.edu

a significant deal of human communication and interaction study (Winograd 1997). For example, a customer interacts with an insurance company to request a new policy. The insurance company agent makes a commitment to the customer to process the request and present the result to the customer. Once the result is presented to the customer, the customer may decline, accept or further negotiate the results. The acceptance of the result implies obligations such as the premium payment by the customer. This is what is referred to as the interaction-based business process modelling approach developed within the language-action perspective (LAP) community with roots in the speech acts theory. The interested reader is referred to the *Communications of the ACM* Special Issue 'Two decades of the language-action perspective', May 2006, volume 49, issue 5, where contributions of pioneering authors of the LAP community are published.

The emergence of business process modelling has been widely recognized as a dominant component of EIS research. In fact, the growing popularity of EIS during the last 15 years is due to the increasing expansion of business processes across organizational and corporate boundaries, and abreast of the globalization of economy, business processes cross national boundaries (Xu 2007). Business process modelling is the basis of process-centric IT systems implementations, e.g. enterprise resource planning systems (Robinson and Dilts 1999). The increasing interest in enterprise business process modelling as a tool for capturing requirements and graphically documenting the processes of an organization to be supported by the envisioned EIS is widely evident from the mainstream literature (Dietz 2006a). However, this complex socio-technical phenomenon is more complicated than the scope and features of the conventional methods for its study. Therefore conventional methods are not capable of capturing the intricacy is complex socio-technical phenomenon.

Over the past few decades, businesses were misguided by the belief that IT alone will solve all their corporate woes, and consequently businesses overemphasized the role of IT while underestimating the importance of a clear understanding and critical analysis of their business processes (Carr 2003). However, the tide has turned and the prevailing standpoint in recent publications suggests a much greater focus on process-centric and process-driven enterprises, rather than merely investing in complicated expensive EIS. This trend requires for future generations of EIS to be increasingly driven by business process models, which, in turn, makes business process modelling a serious challenge (Møller 2007).

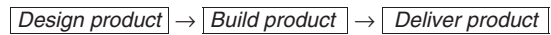
In hoping to contribute towards this challenge, in this paper we discuss and introduce a business process modelling method based on an innovative concept derived from the language-action perspective and based on Petri net formalism.

## 2. Existing issues

Although this is not a discussion paper, an attempt is made to highlight some pressing issues that current enterprise process modelling methods face.

Information systems are developed to support and enable underlying business processes. As business processes in modern enterprises become more complex, overlapping, and cross functional, so do the corresponding information systems. Business process is not merely a flow of activities, but rather a process having deep

and nested structure. In the literature business process models normally represent activities in a flowchart. For example, a flowchart model of product development is illustrated as:



A typical diagram of the product development process would consist of three rectangles connected by arrows. It may also have starting and ending points, and that is all. If the actors are illustrated, it would be only a customer who orders a new product and the contractor who delivers the product. However, the complex situation of real world business may not be that straightforward. For example, if the contractor does not have the capability of certain product design, they will seek a subcontractor for the design of the entire (or part) product. Furthermore, delivering product entails payment settlement. Unless the work is paid, the product cannot be delivered or the order cannot be closed. This reality is even more complicated with all the notions of outsourcing and partnering. The whole process may not be a one way progression, but a complex process of back and forth interactions (negotiations, co-ordination, commitments). This issue is referred to as the deep structure of enterprise business processes (Dietz 2006b). The deep or nested structure of business processes is often undermined in conventional methods and views, or artificially implemented through hierarchical structures. A business process model can further be complicated by the social nature of an enterprise. An enterprise and its business system is a social system (Stamper 1994), where social actors collaborate and interact, including the utilization of artefacts. Stamper argues that information and communication cannot be separated from the context of action (e.g. social actors), and therefore it is dangerous to rely on flowchart-like methods and expect accurate requirements specifications and the capturing of a rich social subtle context. This issue, overlooked by conventional analysts, is referred to as social characteristics (Stamper 1994). Building a complex software system such as EIS requires not one but a number of perspectives and levels of abstraction. For this purpose, the UML methodology offers several different sets of diagrams, recently complemented with more extensions. These diagrams are assumed to collectively provide a multi-perspective and multi-level abstraction of an envisioned software system. However, in the UML case, models may be correct, but difficult to check whether they are constructed correctly; they may be complete, but far too complicated for a novice analyst (Bollojy and Leung 2006); they may be detailed, but difficult to visualize, thus, creating enormous cognitive load for analysts. Although some research has been done in this respect which targets the UML learning difficulties (e.g. Dobing and Parsons 2006), it still remains the main obstacle. This issue is referred to as 'cognitive load' (Moody 2004). Finally, there is the quality dimension that should evolve abreast with modelling techniques. As argued by researchers (Lindland *et al.* 1994), there are certain quality attributes that conceptual modelling should adhere to. They classify them into three quality types: 'syntactic', 'semantic', and 'pragmatic' qualities. Syntactic qualities require rules and grammar that drive modelling and prevent construction errors. In pragmatic qualities, strong emphasis is put on the execution of models, their visualization, simulation and animation. This issue is referred to as 'quality aspects in modelling' (Lindland *et al.* 1994). Many of the conventional methods do not lend themselves to automatic analysis; thus, both their

syntactic and pragmatic qualities are hindered. For example, UML activity diagrams are often translated into Petri net for checking, analysis or simulation (Eichner *et al.* 2005, Eshuis 2006). Similarly, EPC (event-driven process chain) models are translated into Petri net models for analysis (Dehnert and Aalst 2004). Although translation between diagrams or formal representations is common and should not be viewed as a serious challenge, it requires additional rules and procedures that could potentially compromise the model accuracy if not properly followed.

With these issues as the main motivations, this paper is further motivated by the following recently published 'Special Issues':

*Science of Computer Programming*, Special Issue, 'Increasing adequacy and reliability of EIS', March 2007, volume 65, issue 1. This issue is dedicated to the importance of modelling in building accurate and adequate enterprise information systems—verifiable models, not merely static diagrams. This special issue, in confirmation with previous works such as (Lindland *et al.* 1994), recognizes that the quality of the end products significantly depends on the accuracy of conceptual modelling used. Many of the systems fail due to poor conceptual modelling (faulty requirements), it is extremely important for the system's ultimate success to ensure the quality of the modelling methods and tools (Bolloyj and Leung 2006).

*Communications of the ACM*, Special Issue, 'Two decades of the language-action perspective', May 2006, volume 49, issue 5. In the main, this issue is a manifestation of the needs, importance and role of innovative approaches in EIS design. As one of such frameworks, the two decades contributions and results of the language-action perspective in IS design and development are summarized. This issue emphasizes the revolutionary breakthrough of 'human communication and interaction' perspective in system design. This special issue reaffirms the need for adapting innovative approaches, in contrast to conventional ones, if the researchers and developers want to improve the practice of EIS design.

*Journal of Database Management*, Special Issue, 'Systems analysis and design using UML', October–December 2000, volume 11, issue 4. The issue, in part, is dedicated to the difficulties and complexities of UML diagrams. Subsequent studies also confirm that UML, as a method for conceptual modelling and tool for IS design, is difficult to learn, is complex, and there seem to be too many diagrams (Siau and Cao 2001, Siau *et al.* 2005). Furthermore, as another study found out (Dobing and Parsons 2006), not all of the diagrams are used by analysts in IS design.

Taking into consideration the early mentioned issues (drawbacks), and the discussions and future research direction set in the above Special Issues, the overriding concern is to provide EIS analysts and designers with a sound method and concept that delicately balances between the rich informality of social systems and notational artefacts that are amenable to automatic analysis for validation and model checking purposes. As affirmed by others, the underlying model should draw a balance between relationship and action, cognition and affect, message and medium (Te'eni 2001).

### 3. Underlying concepts

In the proposed method, two concepts are consolidated: the language action perspective (LAP) as a theoretical ground theory and Petri nets formalism.

An extensive LAP-related research programme was initiated and conducted at Delft University of Technology, from where the DEMO methodology and its transaction concept originate. The transaction concept will be thoroughly discussed in the following section.

Application of Petri nets in enterprise business processes is not a new idea. Due to its formal semantics and intuitive diagrammatic representation, Petri nets have long attracted researchers for use in business process modelling and design. A host of research works in this area can be found in a collection of papers in Aalst *et al.* (1998). Most of the Petri net models proposed are dominantly process or workflow oriented rather than enterprise business process in the sense of socially interacting and communicating actors. In the framework that we apply Petri net, it is implied that the underlying system is of a social nature, an enterprise where social actors make requests, commitments, negotiations, and bring about new results. Thus, in the proposed method, the emphasis is placed on the social characteristics. In particular, the proposed method is well suited for service-oriented business systems with intensive interactions among participating actors, an organization and customers, and across organizational processes. In contrast to prevailing process-oriented and object-oriented models, the introduced method captures not only process flow, but also takes into account the social character of the modelled enterprise and the nested structure of their activities. This paper further extends the works of Dietz and Barjis (1999) and Dietz (2006b).

This paper aims to make the following contributions:

1. *Executable models of business systems based on the transaction concept.* The contribution is to make the resulting models executable to help system designers with model checking and validation, making changes to the model and study the impacts of the changes. As stressed by Dalal *et al.* (2004), formal analysis in enterprise process modelling is essential for learning the effects of changes, redesign and alternatives in the process logic and parameters on business performance measures.
2. *Compact models of complex systems using the transaction concept.* Often, in modelling, designers are either not interested in all the details, or the system under study is too large to be depicted at detailed level, or the designers may want to spotlight only a part of the system and leaving other parts concealed. For this purpose, compact modelling, where certain activities are compressed into one well defined component, has great advantage. Also, when using diagrams, models may rapidly get too large to manage.

#### 4. The DEMO transaction

In this section we first briefly discuss the DEMO transaction's original diagram and concept, and then we introduce extended notations that are based on Petri net formal semantics. The motivation behind the formal semantics is that the resulting models can be automatically analysed and simulated.

#### 4.1 Original notations

This section is based on the original works conducted in the framework of DEMO methodology by its author Dietz (1994). The results of more than a decade development of this methodology are summarized in Dietz (2006a, b). DEMO is an acronym for ‘design and engineering methodology for organizations’ (see [www.demo.nl](http://www.demo.nl) for more information).

According to the DEMO theory, social actors in organization perform two kinds of acts: production acts (P-acts, for short) and co-ordination acts (C-acts, for short). By engaging in P-acts, the actors bring about new results or facts, e.g. they deliver service or produce goods. Examples of P-acts are: register a student into new course; issue a ticket for a show; make a payment. By engaging in C-acts, the actors enter into communication, negotiation, or commitment towards each other. Examples of C-acts are: making a request for new course; presenting an issued ticket to the customer. The generic pattern in which the two kinds of actions (P-acts and C-acts) occur is called a ‘transaction’, see figure 1. In fact, a transaction is steps of C-act → P-act → C-act that correspondingly result in C-fact (e.g. commitment to register a student) and P-fact (e.g. do register a student).

As it becomes obvious, a transaction is carried out in three phases (Dietz 2006b): the ‘Order’ phase (O-phase, for short), the ‘Execution’ phase (E-phase, for short) and the ‘Result’ phase (R-phase, for short). These three phases involve two actor roles. The actor role that initiates a transaction is called ‘initiator’. The actor role that carries out a production act is called ‘executor’.

In the following section we introduce and discuss the extensions we made to the DEMO transaction. In the terminologies used, we return to earlier terms used for P-acts and C-acts (Dietz 1994). Instead of P-act, we refer to it as ‘action’, and C-act as ‘interaction’.

#### 4.2 Extended notations

Throughout this subsection, we will introduce different properties of the transaction concept along with the notations (artefacts) we have developed for modelling

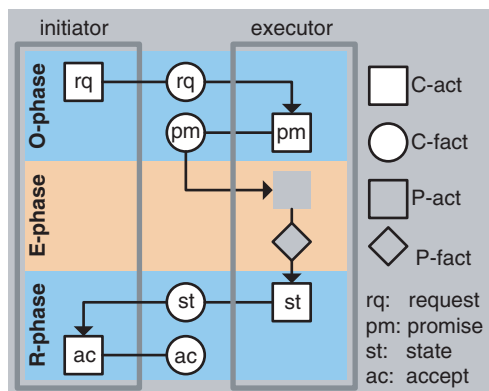


Figure 1. Basic transaction concept (adapted from Dietz 2006b).

enterprise processes. The proposed notations are based on the Petri net formalism, which helps to construct models that can be automatically analysed.

Transactions are patterns of interactions and actions, as illustrated in figure 2(a). An ‘action’ is the core of a business transaction and represents an activity that brings about a new result. An ‘interaction’ is a communicative act involving two actor roles to co-ordinate or negotiate a particular action. Examples of an interaction could be ‘requesting a new insurance policy’, ‘clicking an apply/submit button on an electronic form’, ‘inserting bank card into an ATM to withdraw cash’, or ‘pushing an elevator’s summon button’. Replying to the interacting actors and fulfilling their requests is an action, e.g. ‘issuing a new policy’, ‘processing an e-form’, ‘dispensing bills’, and ‘moving an elevator to the corresponding floor’.

Each business transaction is carried out in three distinct phases, the Order phase, the Execution phase, and the Result phase. These phases are abbreviated as O, E and R correspondingly (see figure 2b). The figure illustrates a business transaction in detailed OER form and compact transaction form (T). Note that the order (O) and result (R) phases are interactions and the execution (E) phase is an action. These three phases are a distinct feature that entails the discussed method as a business process modelling technique versus just process modelling.

In a structured language, a transaction is described according to table 1, where a transaction is portrayed through the activity pattern it represents, its initiator, executor, and the result it delivers (or the new fact it creates). Since real business processes are an arbitrary chain of transactions with the involvement of numerous actors, it is suggested to conveniently denote transactions by the letter T and accordingly number them (T1, T2, T#), and actors by the letter A and number them (A1, A2, A#).

Now, we try to introduce the further notions of the transaction concept along with the Petri net notations we adapted. Figure 3(a) depicts a business transaction using the Petri net notations, where each of the three phases (OER) is represented as a transition (rectangle shape). In a compact notation, these three phases are compressed into a single transition, called Transaction (T). In the figure, the start and the end places are marked by different circles. These notations will

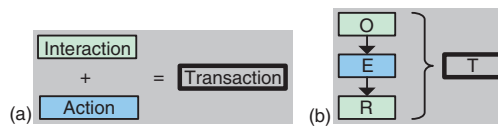


Figure 2. (a) The transaction concept. (b) The OER diagram (detailed and compact notations).

Table 1. Transaction description in a structured language.

Transaction: An atomic process bringing about new result (e.g. placing an order)	
<i>Initiator</i>	Name of the role that initiates the transaction (e.g. customer)
<i>Executor</i>	Name of the role that executes the transaction (e.g. supplier)
<i>Result</i>	The result created as the transaction is carried out (e.g. a new order is created)

show helpful when a complex process consisting of several sub-processes is modelled.

Another notion of the transaction concept is the role of actors involved in a transaction. Each business transaction is carried out by exactly two actor roles, see figure 3(a). The actor that initiates the transaction is called the ‘initiator’ of the transaction, while the actor that executes the transaction is called the ‘executor’ of the transaction. Since the Order (O) and Result (R) phases are interactions between the two actors, their corresponding transitions are positioned between the two actors. The Execution (E) phase is an activity solely carried out by the executor and, therefore, its corresponding transition is positioned within the confines of the executor.

From the information system perspective, a transaction diagram should also represent how the created result (data) is recorded. Since each transaction brings about a new result, the Result phase of a transaction is linked to an oval-shaped element representing the new result created (see figure 3b). For simplicity sake, the depiction of the oval representing a transaction result may be omitted in the models studied later. If a business transaction is a simple one (not nesting further transactions), it is better to compress its three phases into a compact notation, see figure 3(c). In this case, the transaction is placed within the boundary of the executing actor, while the initiation and ending points are placed within the boundary of the initiating actor.

Distinction is made between different types of transaction, simple (causal), composite, and optional transactions. Actors’ interactions may be arbitrarily complex, nested, extensive and multilayered (hierarchical). A complex process typically consists of numerous transactions that are chained together and nested into each other. A ‘simple’ (causal) transaction does not involve (trigger or cause) other transactions during its execution (like in the above figure). It is carried out straightforwardly. In a ‘composite’ transaction, on the other hand, one or more phases will trigger further, nested, transactions. For instance, think if actor A1 contacts actor A2 to reserve a hotel room (we denote this request as Transaction 1, or T1). Actor A2 receives the request and checks the room availability, but in order to fulfil the request, it has to request actor A1 for a payment guarantee (we denote this second request as Transaction 2, or T2). For actor A2 to complete the reservation transaction, first the payment transaction should be completed. This process is represented in figure 4(a) in the form of a nested transaction. Notice that the Execution phase of T1 now has several sub-phases or interactions, where each of the sub-phases is distinguished with a letter of the alphabet attached to the

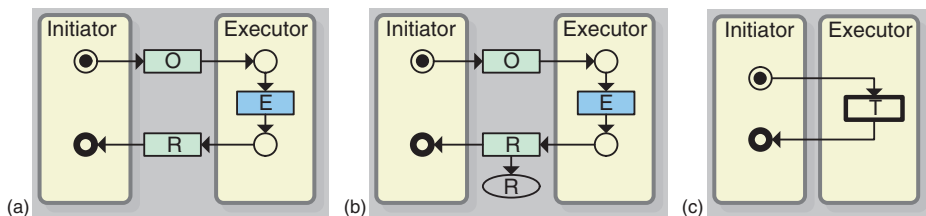


Figure 3. A process diagram of transaction: (a) detailed, (b) with the result and (c) compact.

transaction number (e.g. T1a/E denotes ‘first sub-phase of the Execution phase of Transaction T1’). The process illustrated in the figure starts with the receiving of a reservation request and checking the room availability, then it waits for the payment transaction to get completed, only then the Execution phase gets completed, let say, by conveying a confirmation number to the first actor. A close look at the reservation process reveals that in fact, the payment transaction, T2, is carried out between the hotel and a credit card company. Thus, the process rather involves three actors (actor roles): A1 (customer or guest), A2 (hotel receptionist) and A3 (credit card company). The interaction process between the three actors forms a nested transaction structure, which reveals the deep structure of business process usually ignored or omitted in conventional methods.

One of the limitations in many modelling techniques is coping with complex real-life systems. Usually models of real systems turn too large using diagrammatic representation. In dealing with this issue, we introduce the ‘composite’ (or nesting) notation graphically represented as a multiple (layered) rectangle. For instance, the model illustrated can be reduced to one composite transaction as shown in figure 4(b). This can be applied to any part of a complex process for the sake of compactness or for spotlighting a specific part of the process while concealing the other parts. The notion of nesting structure is especially helpful in inter-organizational process modelling in which a whole process within an organization or business unit can be reduced to a single composite transaction, thus, keeping the model more manageable.

It should be noted that at any point (phase) an actor may quit the process or decline to proceed or a process is terminated due to internal or external circumstances.

In this manner, complex processes with any number of transactions, actors and outcomes can be modelled and illustrated. However, for more complex processes one needs to use often the compact notation of a transaction in order to keep the model better managed and controlled. The compact notation is useful for those transactions that are simple (not nesting further transactions). If a compact notation is used, by a convention, the whole transaction is positioned within the confines of the executing actor. Two instances of such a compact modelling are represented in figure 5. In the first case, the nested transactions are initiated and executed in sequence, and in the second case, they are initiated and executed in parallel.

Another notion is of probability; optional transactions that may take place depending on some conditions. To indicate that a transaction is optional, a small decision symbol of diamond shape is attached to its initiation point as illustrated

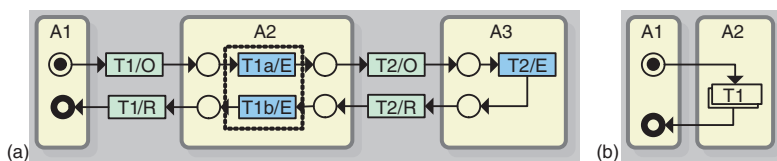


Figure 4. Nested transactions with three actors: (a) detailed and (b) compact.

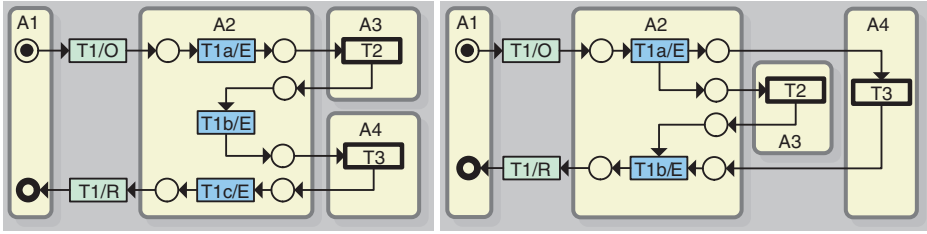


Figure 5. A model with nested transactions: in sequence and in parallel.

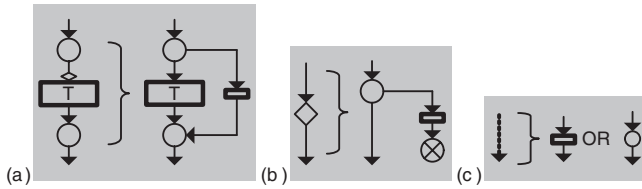


Figure 6. Standard Petri net of (a) an optional transaction and (b) a decision state.

in figure 6(a). In order to transform this optional transaction construct into standard Petri net semantics, a traditional XOR-split that could be modelled by one place leading to two transitions is used. It requires the addition of a skip (dummy) transition as demonstrated in the figure (notice the tiny rectangle with no labels). A dummy transition means that it has virtually zero duration and resource utilization.

Finally, there are situations that a process may halt and result in a termination. For example, if an exception occurs, it may force the process to halt. This situation is modelled through a place identified as ‘decision point’ graphically represented by a decision symbol of diamond shape, see figure 6(b). As it is seen, for the transformation of a decision state into standard Petri net semantics, a traditional XOR-split that could be modelled by one place leading to ‘proceed’ or ‘stop’ is used. Depending on the value of the state, the process either proceeds or terminates as indicated by a place filled with a cross.

One more construct, introduced for practical purpose, is a conditional link (dotted arrow), as depicted in figure 6(c). If a conditional link is needed to connect two places (circles), then it should be represented with an addition of a dummy transition. If it connects two transitions, then it should be added with a dummy place (small circle). This ensures to keep the semantic correctness of the end models.

Through these few simplified constructs and mini-models, we aimed to introduce how the proposed method can capture typical situations in a business process, provide a sound concept based on communication, and ultimately contribute towards more accurate business process modelling and consequently a more adequate EIS design. Now that the basic ideas and constructs of the proposed method are introduced, in the following section we illustrate how this method can be applied to a simple real world business system.

## 5. Case study

What follows is a partial version of a case study demonstrating the proposed method's capability of capturing the system interaction with external and internal entities and objects (actors). This study was conducted when a large regional Pharmacy, assisted by a robotic system, was planning to acquire a new system. This case study was conducted to help understand the Pharmacy's operations and requirements for a new information system that will function within the enterprise and interact with several other applications.

A patient requests prescription refilling by submitting a prescription to the pharmacy technician (pharmacist). For new patient, the technician creates a profile in the pharmacy system's QuickScrip database. After selecting a medicine, the system checks the current medicine for interactions with prescriptions the patient is already taking. The user is alerted if any interactions are found and the process stops. If there are no interactions, the user is asked by the software to transmit a claim to the patient's insurance company, if one is provided. The computer generates a label and sends the information to the 'robot' for automatic filling. The medicine is dispensed into a pre-selected bottle and counted using a laser and gear system which places the medicine into the bottle. A conveyer belt sends the prescription out for a final check by a pharmacist. Once verified, the prescription is bagged and then sent out to the cashier for pick-up by the patient. The entire process normally takes no more than 10–15 minutes. At the pick-up counter, the patient signs for their prescription and pays the cashier. Every time a new prescription is issued, the inventory is also updated. The inventory control sub-process entails a number of activities (transactions).

The inventory controller (software component) conducts complete inventory of the medicines database. If the inventory level is low, it orders new restocking from a supplier (pharmaceutical company). Depending on the amount of order, the supplier sends an invoice to the pharmacy and meanwhile prepares packing and shipping of the medicine. After receiving the payment, the medicine is shipped to the pharmacy.

### 5.1 *The pharmacy transactions*

The process of 'prescription filling' starts when a patient presents a prescription to be filled. Thus, the first transaction (T1) is 'filling prescription'. Actually, this is a super transaction that nests other transactions. This transaction is initiated by a 'patient' and executed by the 'technician'. The result of this transaction is a filled prescription. In this manner we identify all other transactions (see table 2):

Now, based on the transactions identified, we build a detailed business process model of the pharmacy system as shown in figure 7. By disclosing Transaction T1 (splitting it into three phases), all other nested transactions are revealed. This figure shows that once medicine is issued (T1/R), the inventory control process is activated. As the inventory control process is out of the scope, which itself is a

Table 2. Transactions of the pharmacy case.

T1:	Filling prescription
<i>Initiator</i>	Patient
<i>Executor</i>	Technician (pharmacy)
<i>Result</i>	A prescription is filled
T2:	Creating profile
<i>Initiator</i>	Technician
<i>Executor</i>	Patient
<i>Result</i>	A profile is created
T3:	Checking medicine interaction
<i>Initiator</i>	Technician (software agent)
<i>Executor</i>	QuickScrip
<i>Result</i>	An interaction fact is established
T4:	Processing claim
<i>Initiator</i>	Technician
<i>Executor</i>	Insurance company
<i>Result</i>	A claim is processed
T5:	Dispensing medicine
<i>Initiator</i>	Technician
<i>Executor</i>	Robot
<i>Result</i>	Medicine is dispensed
T6:	Paying for the medicine
<i>Initiator</i>	Technician
<i>Executor</i>	Patient
<i>Result</i>	Medicine is paid
T7:	Conducting inventory control
<i>Initiator</i>	Pharmacy (an actor role)
<i>Executor</i>	Inventory controller
<i>Result</i>	Inventory control is carried out
T8:	Ordering new medicine for restocking
<i>Initiator</i>	Inventory controller
<i>Executor</i>	Supplier (pharmaceutical company)
<i>Result</i>	Medicine is shipped
T9:	Paying for the restocking
<i>Initiator</i>	Supplier (pharmaceutical company)
<i>Executor</i>	Inventory controller
<i>Result</i>	Restocking is paid

network of transactions, we just illustrate it as a composite transaction (T#). Within the scope of our model, only Transaction T1 is a composite transaction and, therefore, we decompose it. All other transactions (T2, T3, T4, T5 and T6) are simple transactions and, therefore, are represented in a compact form to keep the model condensed.

The model of figure 7 is read as follows:

**The refilling process starts (T1/O)** with the prescription submission. The technician checks (T1/e1) whether the patient needs a profile (T2) or proceed to select medicine from the system database (T1/e2). The technician requests the system to check the selected medicine for interaction (T3). If any interaction is found, the process stops (notice the circle containing diamond shape); otherwise the technician (T1/e3) simultaneously sends a claim to the insurance company to define the patient's portion of the medicine payment (T4), and a command to the robot to

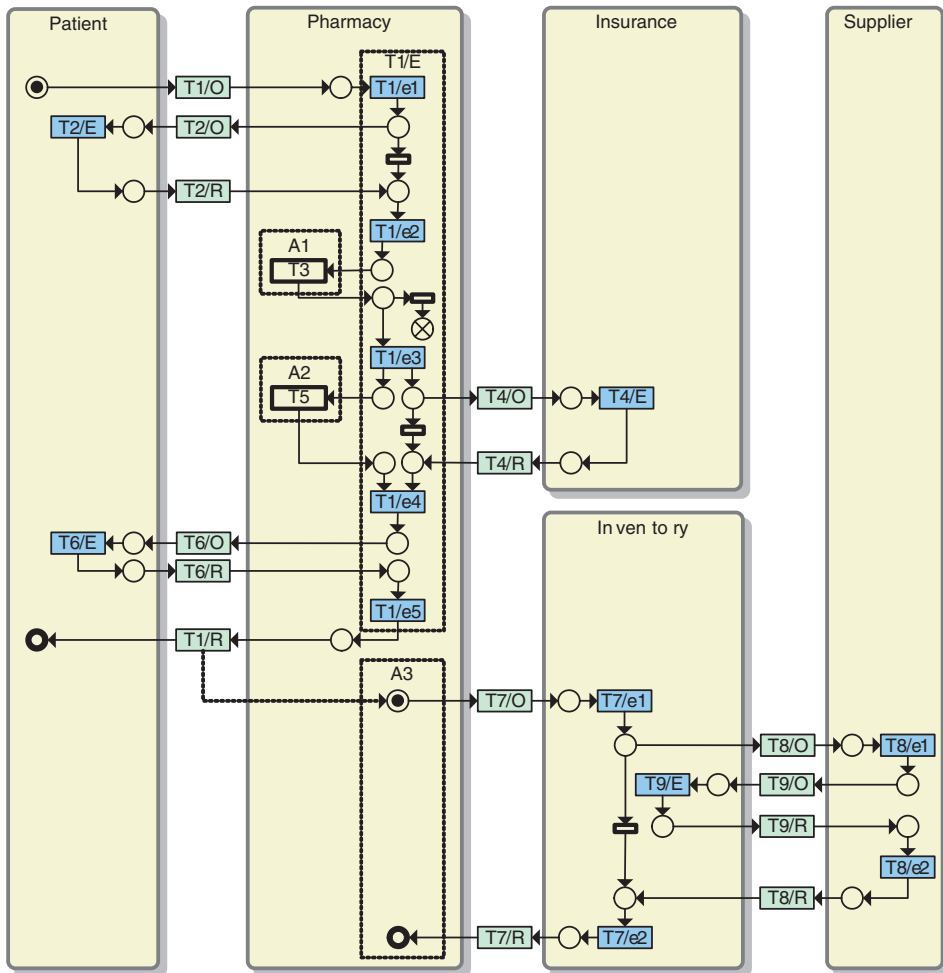


Figure 7. The pharmacy detailed model.

dispense the medicine (T5). When the results of these transactions are received (T1/e4), the patient is requested to make their payment (T6), and then the medicine is issued (handed) to the patient (T1/e5) and the patient accepts the medicine, and **the refilling process ends (T1/R)**.

Notice that the inventory process regularly checks if any restocking is needed. Although this is a separate process, there is a conditional link between medicine issuance (T1/R) and the inventory process (T7/O).

**The inventory process starts (T7/O)** with the issuance of medicine to a patient. The inventory controller checks if new restocking should be ordered (T7/e1). If yes, then a supplier is requested for restocking (T8/O), otherwise the skip transition is executed. If restocking is requested, the supplier initiates the payment process (T9/O) that should be executed by the inventory controller (T9/E). After the payment is received, the medicine is shipped (T2/R), and **the inventory process ends (T7/R)**.

## 6. Evaluation and comparison

### 6.1 Application

A new method and artefacts require evaluation to examine their soundness and rigorousness. Hevner *et al.* (2004) suggest that graphical representation should be simple, intuitive and easy to understand, but at the same time, the accuracy and adequacy of such a representation should not be compromised. Furthermore, Hevner *et al.* (2004) recommend that methods deploying artefacts should be evaluated using 'observational' (e.g. case study) and 'experimental' (e.g. simulation) methods. The observational (case study) approach reveals the applicability potential of a method and its artefacts in a given environment and category that are targeted by the method. For example, case studies on organizations of different sizes, different levels of complexity, and different levels of abstraction, but all within the same sector (e.g. service oriented organizations such as insurance companies, healthcare/hospitals, hotels). The experimental (simulation) approach reveals if models can be checked, analysed and verified. This not only allows models to be checked for consistency, but also eliminates syntactic errors, illustrates dynamic behaviour of models, and leads to formal analysis.

In light of these recommendations, the proposed method has been tested on both observational and experimental bases. A dozen case studies have been conducted using the proposed method, and discussed among researchers and experts. Each of the models has been straightforwardly simulated to check the models accuracy. All the models were communicated to, and discussed with the users. The feedback obtained from modellers, designers, users, and researchers greatly helped to polish and improve the method and eliminate flawed constructs. It is not claimed that this method is now perfect and therefore we urge researchers to use and analyse it. The feedback received, however, confirms that this method has a promising potential for the EIS community. We would appreciate to see more researchers use the method and give it a critical analysis.

### 6.2 Comparison

There are a plethora of methods adapted for business process modelling that one could discuss for comparison with the proposed method. But neither the scope of this paper nor the space available allows us to do so. We confine our comparison to a brief discussion.

As a first comparative remark, contrasting the proposed method UML, EPC and other Flowchart methods, the main distinction that should be made is the social emphasis of the proposed method. In the proposed method, each action is seen in the context of its actor and intended audience; neither of the actors are separated from the actions nor are the actions studied in isolation from the actors. In fact, as the discussed examples illustrated, the model starts around (social) actors and their role in the process. In UML, this issue is addressed through the notion of a swim-lane drawn around objects. However, the use of swim-lane is optional and takes place after the model is constructed, i.e. swim-lanes can be added after the model is built. Furthermore, most of the conventional models are checked and analysed via translation to other formal diagrams using mapping procedures. For instance, UML

activity diagrams are often translated to Petri nets for checking (e.g. see Eichner *et al.* 2005, Eshuis 2006), and several tools are developed to translate UML diagrams to Petri net for further simulation. Another widely accepted method, investigated in Dehnert and Aalst (2004), is event-driven process chain (EPC). The authors propose a five-step guideline to translate EPC models to Petri net models in order to investigate whether the process is correctly described in EPC. The analysis showed that ambiguities of EPC models will result in faulty Petri net executions. In this regard, the superior advantage of our method is its direct adaptation of Petri net formal semantics for the developed notations and constructs. Thus, analysts do not need further translations that could compromise the accuracy and adequacy of modelling, or cause further sophistication through the development of mapping procedures.

As for the modelling notations that compete with Petri net, e.g. BPMN, EPCs, role-activity-diagrams, IDEF, UML, RIVA, etc., the reasons for adapting Petri net are its formal semantics, logics and formalism, and also its widespread use among researchers, practitioners and a variety of academic disciplines. In addition, Petri net is supported by a large number of tools for its analysis. Aalst (1996) identifies three main reasons why Petri net possesses advantageous features: formal semantics despite the comprehensive graphical representation; state-based representation instead of event-based; abundance of analysis techniques. Process modelling techniques ranging from informal techniques (e.g. dataflow diagrams) to formal ones (e.g. process algebra) are event-based, while Petri net approach allows state-based modelling.

## 7. Conclusion

This paper presented a method for enterprise process modelling. The innovation of the proposed method is in its underlying concept of communication as a tool for elicitation and identification of action patterns. The purpose of this method is to conduct enterprise process modelling in a manner that the resulting models can serve as a preliminary blueprint for EIS design. In this regard, the method's special features are:

- *Capturing deep business process structures.* Implemented through composite and nested transactions.
- *Compact presentation:* Implemented via several constructs with utilization of compact notations.
- *Automatic analysis, simulation and animation:* Facilitated by adopting Petri net formalism.
- *Rich representation with least complexity:* The proposed method in a single diagram captures many attributes and aspects that in other methods would require several diagrams (e.g. use case, activity, state chart diagrams).

The graphical language of Petri net makes it easier to communicate the models among analysts and users, while at the same time they are based on formal semantics. Thus, the models are fully graphical for illustration, and formal to enable complete analysis and powerful animation as well as simulation. All models developed and

discussed in this paper were analysed for syntactic correctness and animated via token game to illustrate the system dynamic behaviour.

Compared with typical flowchart models, where each rectangle is labelled with the name of activity, the proposed method may seem more technical and less readable. The flowchart models are easy to follow, but they are challenged even when a slightly complex process is modelled, in which case the chart is broken in pieces. In this regard, though the proposed method seems more technical, it can construct fairly complex models on a single sheet of paper, especially with the deployment of compact and composite notations.

Now that the method is emerging from its development stage and seems a ready tool for practical use, there are a few potential directions this method can be further developed and tested:

First of all, keep using the method in more complex real life enterprises to explore turns and twists of enterprise business processes and examine the proposed method's capability in coping with them.

Develop concise modelling guidelines for practitioners to use the proposed model in a systematic way.

Conduct critical comparative analysis of the proposed method and conventional methods such as UML, EPC, RAD.

Develop tools to support the method (graphical editor, simulator, animator, analyser).

## References

- Aalst, W. van der, Desel, J. and Oberweis, A. (eds), *Business Process Management: Models, Techniques and Empirical Studies*, 1998 (Springer-Verlag: Berlin/Heidelberg/New York).
- Aalst, W.M.P. van der, Three good reasons for using a Petri-net-based workflow management system, in *Proceedings of the International Working Conference on Information and Process Integration in Enterprises*, 1996, pp. 179–201.
- Bider, I., Choosing approach to business process modelling—practical perspective. *J. Concept. Mod.*, 2005, (34) (Publisher InConcept).
- Bolloy, N. and Leung, S.S.K., Assisting novice analysts in developing quality conceptual models with UML. *Comm. ACM*, 2006, **49**(7), 108–112.
- Carr, N.G., IT doesn't matter. *Harv. Business Rev.*, 2003, **81**(5), 41–49.
- Dalal, N.P., Kamath, M., Kolarik, W.J. and Sivaram, E., Toward an integrated framework for modelling enterprise processes. *Comm. ACM*, 2004, **47**(3), 83–87.
- Dehnert, J. and Aalst, W.M.P., van der, Bridging the gap between business models and workflow specifications. *Int. J. Coop. Inform. Syst.*, 2004, **13**(3), 289–332.
- Dietz, J.L.G., Business modelling for business redesign, in *Proceedings of the 27th Hawaii International Conference on System Sciences*, IEEE Computer Society Press, Los Alamitos, 1994, pp. 723–732.
- Dietz J.L.G. and Barjis J., Supporting the DEMO methodology with a business oriented Petri net, in *Proceedings of the 4th CAiSE/IFIP8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD)*, Heidelberg, Germany, 14–15, June 1999.
- Dietz, J.L.G., *Enterprise Ontology—Theory and Methodology*, 2006a (Springer: New York).
- Dietz, J.L.G., The deep structure of business processes. Special Issue: Two decades of the language-action perspective. *Comm. ACM*, 2006b, **49**(5), 59–64.
- Dobing, B. and Parsons, J., How UML is used. *Comm. ACM*, 2006, **49**(5), 109–113, May.
- Eichner, C., Fleischhack, H., Meyer, R., Schrimpf, U. and Stehno, C., Compositional semantics for UML 2.0 sequence diagrams using Petri nets, in *SDL Forum*, 2005, pp. 133–148.

- Eshuis, R., Symbolic model checking of UML activity diagram. *ACM Trans. on Software Eng. & Methodol.*, 2006, **15**(1), 1–38.
- Hevner, A.R., March, S.T., Park, J. and Ram, A., Design science in information systems research. *MIS Quart.*, 2004, **28**(1), 75–105.
- Lindland, O., Sindre, G. and Sølvyberg, A., Understanding quality in conceptual modelling. *IEEE Software*, 1994, **11**(2), 42–49.
- Møller, C., Process innovation laboratory: A new approach to business process innovation based on enterprise information systems. *Ent. Inform. Syst.*, 2007, **1**(1), 113–128.
- Moody, D.L., Cognitive load effects on end user understanding of conceptual models: An experimental analysis. In *Lecture Notes in Computer Science*, Vol. 3255, 2004 (Springer: Berlin/Heidelberg).
- Robinson, A.G. and Dilts, D.M., OR & ERP: A match for the new millennium? *OR/MS Today*, 1999, **26**, 30–35.
- Siau, K. and Cao, Q., Unified modelling language (UML)—a complexity analysis. *J. Database Manage.*, 2001, **12**(1), 26–34.
- Siau, K., Erickson, J. and Lee, L.Y., Theoretical vs. practical complexity: The case of UML. *J. Database Manage.*, 2005, **16**, 40–57.
- Stamper, R.K., Social norms in requirement analysis—an outline of MEASUR. In *Requirements Engineering: Social and Technical Issues*, edited by M. Jirotko and J. Gorguen, 1994 (Academic Press Ltd: London, UK).
- Te'eni, D., A cognitive-affective model of organizational communication for designing IT. *MIS Quart.*, 2001, **25**(2), 251–312.
- Winograd, T., The design of interaction. In *Beyond Calculation. The Next 50 Years of Computing*, edited by P. Denning and B. Metcalfe, 1997 (Springer-Verlag: New York).
- Xu, L., Editorial: Inaugural issue. *Ent. Inform. Syst.*, 2007, **1**(1), 1–2.