Proceedings of the
IASTED International Conference

IASTED

Applied Simulation and Modelling

June 25 - 28, 2002
Crete, Greece

Editor: L. Ubertini

A Publication of The International Association of Science and Technology for Development - IASTED

ISBN: 0-88986-334-2
ISSN: 1021-8181

ACTA Press

Anaheim • Calgary • Zurich

Proceedings of the
IASTED International Conference

Applied Simulation and Modelling

June 25 - 28, 2002
Crete, Greece

Editor: L. Ubertini

A Publication of The International Association of Science and Technology for Development - IASTED

ISBN: 0-88986-334-2
ISSN: 1021-8181

ACTA Press

Anaheim • Calgary • Zurich

Proceedings of the IASTED International Conference on **Applied Simulation and Modelling**, held June 25-28, 2002, Crete, Greece.

Sponsors

The International Association of Science and Technology for Development (IASTED)

- Technical Committee on Modelling and Simulation

World Modelling and Simulation Forum

Editor

L. Ubertini
IRPI - National Research Council, Italy

International Program Committee

A.R. Abdullah - University Kebangsaan Malaysia, Malaysia
G.K. Adam - University of Thessaly, Greece
K. Adamson - University of Ulster, UK
M. Adelantado - ONERA-CT/DPRS/SAE, France
S. Aliabadi - Clark Atlanta University, USA
Y. Attikiouzel - The University of Western Australia, Australia
F.J. Azuaje - Trinity College Dublin, Ireland
U. Biader Ceipidor - Univ. La Sapienza, Italy
P.I. Brooker - University of Adelaide, Australia
S.R. Das - University of Ottawa, Canada
F. De Felice - University of Cassino, Italy
M. Druzovec - University of Maribor, Slovenia
M. Durali - Sharif Univ. of Technology, Iran
R. Ege - Florida International University, USA
P. Giribone - University of Genoa, Italy
J.S. Glower - North Dakota State Univ., USA
P. Gorce - University of South Paris, France
G.A. Gravvanis - Univ. of the Aegean, Greece
Y. Hamamatsu - Ibaraki University, Japan
P. Hernandez - University Institute of Microelectronic Applications (ULPGC), Spain
R. Hornero - University of Valladolid, Spain
A. Jirachiefpattana - Huachchiew Chalermprakiet University, Thailand
S. Kopacsi - Hungarian Academy of Sciences, Hungary
K. Kurapati - Philips Research Laboratories, USA
A. Lisounkin - Fraunhofer Institute for Production Systems & Design Technology, Germany
M. Lotfalian - University of Evansville, USA
E. Mesbahi - University of Newcastle, UK
E. Moreno - Euripides Foundation of Marilia, Brazil
J. Navarro-Moreno - University of Jaen, Spain
T. Niculiu - Polytechnic Univ. of Bucharest, Romania
Y. Okuyama - Tottori University, Japan
A. Orsoni - DIP University of Genoa, Italy
S.K. Pal - University of Sheffield, UK
X. Pan - University of Chicago, USA
R.J. Paul - Brunel University, UK
N. Peric - University of Zagreb, Croatia
J. Pirron - Fraunhofer Institute of Manufacturing, Engineering, & Automation, Germany
M. Purins - University of Otago, New Zealand
J. Rady Almeida Jr. - Univ. of Sao Paulo, Brazil
A. Rahrooh - University of Central Florida, USA
Y.B. Reddy - Grambling State University, USA
A. Rosa - Technical University of Lisbon, Portugal
J.M.G. Sa da Costa - Technical University of Lisbon, Portugal
Z. Siddique - University of Oklahoma, USA
M. Sirola - Helsinki University of Technology, Finland
A. Swierniak - Silesian Technical University, Poland
W. Tang - National Hsin-Chu Teachers College (NHCTC), Taiwan
A. Tentov - University of St. Cyril and Methodius, Macedonia
M. B. Trabia - University of Nevada - Las Vegas, USA
D.-Y. Tsai - Niigata University, Japan
S. Umeda - Musashi University, Japan
G.A. Wainer - Carleton University, Canada
R. Wamkeue - University of Quebec at Abitibi-Temiscamingue, Canada
A.B. Wolpert - Roosevelt University, USA
D. Xu - Shantou University, PRC
B. Yang - Dong Hua University, PRC
X. Yu - City University of Hong Kong, PRC
A. Zizzari - Otto-von-Guericke University - Magdeburg, Germany

Additional Reviewers

J. Abedi - USA
V.G. Agelidis - UK
M.Al-Nimr - Jordan
P.G. Bao - PRC
S.P. Banks - UK
J. Barros-Guadalupe - Spain
L. Bengtsson - Sweden
A. Bestavros - USA
A. Branca - Italy
M. Brunet - France
D. Campbell - Australia
V. Casamassima - Italy
C. Chatzinikolaou - Greece
C. Corradini - Italy
J.D.S. da Silva - Brazil
A. El Hajj - Lebanon
L.J. Everett - USA
R. Faglia - Italy
E. Ferreira - Portugal
I. Filip - Romania
L. Freeman - UK
M.M. Freire - Portugal
J.L. González Sánchez - Spain
M.A. Hasan - USA
H. Hu - UK
A. Isapour - Iran
H. Jaddu - Japan
A. Jouan - Canada
J. Kamruzzaman - Australia
H. Kawabe - Japan
T. Keikko - Finland
P.-S. Kim - Korea
R. Kowalczyk - Australia
Z. Kremens - USA
J. Krope - Slovenia
R.W.H. Lau - PRC
A. Lazarevska - Macedonia
E. Levner - Israel
O.F. Lin - Canada
J. Nga-kwok Liu - PRC
P. Massacci - Italy
R. Mitchell - UK
J.M. Molina - Spain
E. Monteiro - Portugal
A. Movaghar - Iran
L.B. Orozco-Barbosa - Canada
D. Pacholczyk - France
R.B. Pajarola - USA
S.-won Park - USA
S. Peng - Japan
H.G. Petersen - Denmark
J.E.J. Plasmans - Belgium
J.H.C. Pretorius - South Africa
A. Prodan - Romania
G. Raghavan - USA
D. Robinson - USA
C. Romualdo-Torres - Mexico
M.Rouff - France
S.H. Rubin - USA
[redacted]
S.R. Salem - USA
A.A. Salman - Kuwait
E. Santini - Italy
E.K. Stefanakos - USA
P. Stewart - UK
M. Tahernezhadi - USA
K.K. Tan - Singapore
P. Tusaliu - Romania
A. Varela - Venezuela
G. Varga - Hungary
J. Vittek - Slovak Republic
J.-horng Wen - Taiwan
D.C.H. Yang - USA
E.E. Yaz - USA
N. Yorukeren - Turkey
J.S. Yuan - USA
J. Zeaiter - Australia
Z. Zhang - UK

Special Session Organizers

G. Bella - Tor Vergata Univ. of Rome, Italy

C. Corradini - Perugia Univ [redacted]

ACTA Press
P.O. Box 2481
Anaheim, CA 92814-2481
USA

ACTA Press
#80, 4500 - 16 [redacted]
Calgary, AB T3 [redacted]
Canada

For each IASTED conference, the following review process is used to ensure the highest level [redacted] two separate reviewers on the International Program Committee / Additional Reviewers list. [redacted] sent to a third reviewer.

Abstracted by INSPEC

Publication Code: 363

TABLE OF CONTENTS

Biomechanics

Environmental Modelling and Simulation

Simulation Tools and Techniques

Applied Simulation in Energy Sector

Expert and Fuzzy Systems

Robotics and Manufacturing

Identification and Optimization

Control and Applications

SPECIAL SESSION 1: Applied Simulation and Modelling in Medicine

SPECIAL SESSION 2: Hydraulic and Hydrologic Modelling

Proceedings of the IASTED International Conference
APPLIED SIMULATION AND MODELLING
June 25-28, 2002, Crete, Greece

INTEGRATED METHODOLOGY ALLOWING DESIGN OF ICT APPLICATIONS BASED ON BUSINESS PROCESS INVESTIGATION

Boris Shishkov, Jan L.G. Dietz
Faculty ITS, Delft University of Technology
Mekelweg 4, 2628 CD, Delft
The Netherlands

Abstract

A considerable number of software artefacts that are supposed to support contemporary business processes and are developed at great cost fail to satisfy users. In many cases, the reason for this is the improper reflection of business requirements into software design, which results in lower levels of effectiveness of developed ICT applications. Aiming at aligning business process investigation and software design, we propose a methodology that combines the two. We claim that it is useful for contemporary design of applications, especially when it is essential that they support (effectively) some business processes. The methodology provides an original way of combining several investigation tools (DEMO, Use cases, UML diagrams, Simulation tools), with the particular purpose of basing application development on business process modeling. We illustrate the introduced methodology using a case example.

Key Words

Modeling; Business processes; Software design; UML; DEMO.

1. Introduction

Since the Information and Communication Technology (ICT) has entered almost each sphere of our life, the majority of contemporary business processes depend to a large extent on advanced technological support [1], realized via ICT (software) applications. However, in many cases the applications being developed do not realize support to corresponding business processes as effectively as wanted [2]. This is the case even though there are good tools for both application design /e.g. UML [3]/ and business process modeling /e.g. DEMO [4]/. Actually, we observe two opposite phenomena [5]. On one hand, we observe software being developed without prior (consistent) investigation of the (business) processes to be supported by it. This means that the business requirements are poorly determined and the software design model does not have its roots in a business process model. Therefore, the developed software would support the business processes inadequately to their needs; and although its quality might be high from a software point of view, the effectiveness of the support it realizes to the target business processes would remain low. On the other hand, although (in many cases) sound business process modeling is conducted prior to the design of software, the business process model is only partially used, since it is not straightforwardly transformable into a relevant input for the software design. This does not allow for full employment of the software and ICT possibilities in solving particular business problem(s). Thus, the two outlined tasks need to be aligned in a better way. The business process modeling and the development of ICT applications for the support of business processes should be considered as one integrated task.

In this paper, we aim at contributing to the knowledge on consistently basing the design of software on a business process model. A methodology is introduced that suggests a set of interrelated tasks to be realized in bridging software design and business process modeling, elaborating also on appropriate tools to be applied in this regard. The methodology is considered in Section 3, following a brief analysis of the current situation regarding the mentioned problem (Section 2). In Section 4, a case example, related to a brokerage system for e-business, is studied in order to illustrate the methodology.

2. Necessity to Better Align System Design and Business Process Study

According to surveys considered in [2], 31.1% of the projects, aiming at developing software to support business processes, are cancelled before they ever get completed. 52.7% of projects go over time and/or over budget, at an average cost 89% of their original estimates. As already stated, one of the reasons for these unsatisfactory results is the mismatch between the business requirements and the actual functionality of the delivered ICT application.

Different researchers address issues related to the outlined problem. Dehnert and Rittgen present a formal representation for describing business processes [6]. This is a promising step and could be especially useful if further related to software design. Olivera, Filho and Lucena have also contributed in this direction, by investigating the design of software on the basis of business requirements analysis [7]. Their suggested approach is a step ahead even though it does not still offer a straightforward mapping of a business process model into a software design model. Hikita and Matsumoto have studied how the appearance of additional requirements could be reflected in the system's construction [8], which is also a promising result achieved so far (although not completely solving the problem). Krutchen suggests (based on the existing use case concepts [3]) a "Business use case" – considered useful in bridging business process modeling and software design [9]. But it is still a question how to consistently identify such use cases. Therefore, it might be concluded that further knowledge is still required in the direction of consistently basing application design on business process modeling. The methodology, introduced in the following section, is claimed to be a contribution in this direction.

3. Software Design Based on Business Process Modeling

The suggested methodology focuses on the modeling and elaboration of the functionality of a software system, consistently basing this on business process modeling. The further design and implementation steps are not covered since they do not relate directly to the addressed problem. The methodology provides an original way of combining several modeling tools (DEMO [4], Use cases, UML diagrams, Simulation tools). The methodology steps and corresponding tools are represented below:

- DELIMITATION OF THE DOMAIN.
- BUSINESS PROCESS MODELING: Investigation of the business processes to be supported, aiming at grasping the relationship between the system to be modeled and these business processes.
- SYSTEM MODELING: Modeling the overall functionality of the system under development, basing this on the realized business process investigation.
- ELABORATION: Partial representation of the system as a subsystem, and further granularity of the modeled subsystem(s) with respect to structure and realized activities.
- VALIDATION: Validation of the developed models, concerning the overall system as well as the subsystems.

The suggested investigation tools to be applied for realizing the outlined methodology steps are: DEMO – for realizing the business process modeling; UML (Use Case Diagram) - for system modeling; Use Cases, Activity Diagram – for elaboration; ARENA Simulation Tool – for validating the developed model(s).

Distinguishing between the activities related to business process modeling and those related to the design and elaboration of a software model, we could identify two major phases in the methodology outline: Business process modeling and System modeling & elaboration, which are supposed to bridge the design of software to prior investigation of the target business system. They should be extended with a third phase – Validation of the designed model(s). These phases and their corresponding investigation tools are discussed further on in this section.

3.1 Business Process Modeling

The defined goal, in conducting business process modeling as an input for further software development activities, is to provide a consistent abstract business process model for the software design, a clear model that captures the features which remain unchanged from realization. Such a model could be a sound basis for improving the delimitation, identification and the specification of the modeled software system [10].

Another demand is that the developed business process model should be straightforwardly mappable into the design of software – there must be a direct mechanism for reflecting such a model into the model of the software system under development. As already stated, the system modeling phase of the methodology is based on Unified Modeling Language – UML [3]. The reasons for this choice will be discussed further on. At this stage, it is important to be aware of this fundamental issue since this narrows the demand considered in the current paragraph, making it needed for the designed business process model to be straightforwardly reflectable into a UML model.

Based on an analysis of some of the most consistent business process investigation tools, e.g. DEMO, Semiotic tools [2], Petri net [11], and taking into account the demands outlined above, it has been concluded that DEMO fits best within the particular tasks of the suggested methodology. DEMO is claimed to be the most appropriate tool in this respect because of its completeness and capability of capturing the essence of business processes. Taking into account that the introduced methodology aims at aligning software design and business process modeling, it is considered crucial (as already stated) that the essence of the target business processes is grasped, in order to build a consistent business process model which is fully abstracted from all realization issues. Hence, if the developed software model stems from such a business process model, the software designer would have the right (re)design freedom [4]. Next to that, it was studied that deriving a UML Use case diagram on the basis of DEMO Coordination structure model (to be introduced further on) is straightforward and consistent. It is rooted in the theories behind these tools (and also outlined further on). However, due to the limited scope of this paper this issue is not further elaborated. It is considered also in [5]. The case example (Section 4) illustrates the mapping between a DEMO model and a Use case diagram. Below, DEMO is briefly introduced as a fundamental business process modeling tool concerning the suggested methodology.

Dynamic Essential Modeling of Organizations - DEMO is a methodology for understanding, analyzing, (re)designing and (re)engineering business processes. Its underlying theory about organizations is rooted in the Language/Action Perspective [12], Organizational Semiotics [2] and Philosophical Ontology [13]. DEMO reveals the "construction" and "operation" of an organization, contrary to the current function and behavior-oriented approaches. It is characterized by three major features: 1) a white-box architecture of actors, production and coordination, 2) the extraction of the essence of business processes from their realization, 3) the transaction pattern.

Actors, production, coordination. Like every other system (e.g. an alarm clock or a racing car), the functional behavior of an organization is brought about by the collective working of the constructional components. The construction and the working of a system are most near to what a system really is, to its ontological description [13]. An organization is defined as a (discrete dynamic) system in the category of social systems. This means that the elements are social individuals or actors, each of them having a particular authority to perform production acts (P-acts) and a corresponding responsibility to do that in an appropriate and accountable way. The structure of an organization consists of coordination acts (C-acts), i.e. the actors enter into and comply with commitments regarding the performance of P-acts. The generic white-box organizational model (Figure 1) consists of: the actors, the P-world, and the C-world [4].

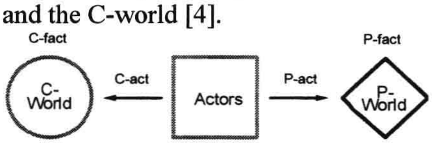

Figure 1: The white-box model of an organization

By performing P-acts, the organization does what it is supposed to do according to its function. C-acts serve to coordinate and control the performance of P-acts.

Essence and realization. In DEMO, three perspectives on an organization are distinguished: essential (the organization viewed as a system of authorized and responsible actors that create new original facts), informational (the organization viewed as a system of information processors that remember facts and derive new facts from existing ones), documental (the organization viewed as a system of formal operators that collect, transport, store, copy and destroy representations of facts) [10].

Take for example the process of withdrawing money from a bank account using an ATM machine. Think of observing this process through essential, informational or documental "glasses" as a metaphor. Looking through documental "glasses" we see someone inserting a card into a machine, pushing buttons on a keyboard and finally getting out the card and other pieces of paper. Nothing with respect to the information on it or the purpose for which they are used, is seen. Looking at the same process through informational "glasses", we see someone providing information to an ATM system: a PIN code and specification of an amount of money. Also, the machine provides information if withdrawal is possible to the customer. We see that the machine outputs money and receipts. Looking through essential "glasses" shows responsible actors, their actions and interactions. A customer requests a bank to withdraw money from an account. The bank decides to do this and states that the money is withdrawn. Further on, the customer accepts it.

The transaction pattern. P-acts and C-acts appear to be performed in particular sequences that can be viewed as paths through a generic pattern called the (business) *transaction* [4]. A transaction is a finite sequence of C-acts between two actor roles, the customer and the producer. It takes place in three phases: the order phase (O-phase), the execution phase (E-phase), and the result phase (R-phase). O-phase is a conversation that starts with a request by the customer and that, if successful, ends with a promise by the producer. E-phase basically consists of the performance of the P-act by the producer. R-phase starts with the statement by the producer that the requested act is performed and ends, if successful, with the accept by the customer. The whole pattern of a transaction is represented by one symbol in the so-called Coordination Structure Diagram (CSD). Fig. 2 exhibits CSD for the money withdrawal example. The two boxes represent the two actor roles involved: A0(A1) is the customer(producer). The small black box indicates that A1 is the producer of T1 (and consequently A0 is the customer). The successful result of a transaction T1 is the P-fact "withdrawal W is performed" where W is constituted by the account, the amount and the time.

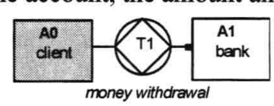

Figure 2: CSD of the money withdrawal example

3.2 System Modeling and Elaboration

Having an abstract business process model beforehand helps to scope the application to be modeled. It is necessary to precisely model the system functionality, as already stated. Further on, it is essential to properly elaborate on the constructed model, identifying the process(es) to be studied in depth.

It was said also that regarding these issues, the methodology is based on UML. This choice is motivated not only by the completeness of UML [3,9] but also by the fact that it turns out to be de facto the standard language for modeling software systems [1], widely accepted by both researchers and practitioners.

In particular, two UML diagrams play an important role in the suggested methodology, namely the Use case diagram and the Activity diagram (AD). Also, Use cases (UC) are considered as modeling constructs that represent functionality of a system. Introducing these modeling tools is left beyond the scope of the current paper since they are well-known to the public from numerous literature and other sources [3,9,14,15]. Attention is paid only to UC, and in particular – to the considered UC concepts of Cockburn [15], which add elicitation value to the "classical" UML concepts [14].

The UC concepts of Cockburn are interesting because his latest work [15] is among the most ambitious recent material in the domain, providing useful and promising ideas. His view is illustrated in Figure 3. As seen from the figure, Cockburn (aiming at looking inside UC) takes into consideration not only the primary actor (or user [3]) but also the stakeholders involved. For this reason, his model seems more complete than the "classical" model of Jacobson who introduced UC [14]. At the same time, the graphical representation of Cockburn's model is unsuitable for presenting a multitude of UC.

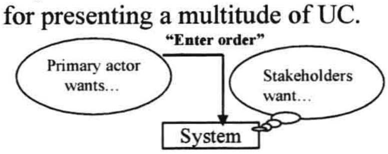

Figure 3: A UC as seen by Cockburn

However, it should be noted that the latest developments in Jacobson's UC concept are put in the perspective of UML. For this reason, it is emphasized not on the complete representation of a UC, but on features allowing developers to show relationships which cover many UCs and actors. This is the main function of the UC diagram [3]. Thus, Jacobson extracts the gist – actors, pieces of functionality and their relationships. Jacobson and Cockburn form their UC perspectives from different angles. This shows that UC needs further exploration in order to provide options for both complete insight and flexible multi-UC representation. The ideas behind the introduced methodology are also in this direction, suggesting application of UC in combination with AD, for a more extended insight.

Regarding the roles of the modeling tools discussed in this subsection, as it was already stated, the UC diagram plays the crucial role of modeling the system functionality grasping the requirements and actors involved. Based on this, looking inside any UC(s) (following the concepts of Cockburn) is essential for having a complete insight regarding important elements of the system. The main output provided by the UC investigation should be: structured information about stakeholders, activities, triggers, etc. Having a complete view over the realized activities, regarding a UC, it is straightforward to model the UC dynamics using AD. Modeling the dynamical aspects of systems is important because it provides information about the sets of activities and their interrelationships.

3.3 Validation

Once we have an AD related to the UC under study, it is straightforward to build a simulation model [16], if necessary. Models represented with the means of AD could be easily simulated, using different tools. Proceeding with computer simulation (if further elicitation is necessary), using e.g. Arena simulation tool [16], would be useful for visualizing processes, providing a dynamic perspective to the information available from modeling tools, providing understandable view over branching, choice and other complex structures. All this could be used to validate the developed AD models. However, discussing this is left beyond the scope of the paper. Usage of AD as a pre-simulation tool is thoroughly investigated in [16].

IN CONCLUSION: DEMO, UC diagram and AD are crucial for the proposed methodology. DEMO provides an abstract business process model as an input for the system design; the UC diagram plays the role of interface towards it (as studied in [5] and also demonstrated in Section 4, the UC diagram plays the role of interface in combining DEMO and UML within the methodology because DEMO provides a consistent guidance in identifying UCs). The diagram allows grasping the system as a whole and subsequently AD considers the dynamics in any parts of it. UCs bridge these two elements of the methodology.

4. The BSEB Case

The considered case concerns e-business (EB) - one of the domains, crucial for the contemporary business development [5]. EB is defined [1] as *business conducted using to a large extent the possibilities of ICT, including Internet. EB encompasses such diverse activities as: identifying relevant partners, negotiating with them, and conducting business transactions.* The case example is related, in particular, to a Brokerage System for EB (BSEB). Such systems are directed towards solving a fundamental problem in EB – the lack of effective enough partner and/or goods searching mechanisms. These issues determine the required functionality of a BSEB (analogous functionality is explored in [17]).

Since the potential users of such a system are spread across multiple locations, BSEB should represent a distributed ICT application, effectively supporting its users across distributed computing environments. Match-making should be realized as follows: There are: 1) different sellers aiming at succeeding to sell their goods as quickly as possible; 2) different buyers aiming at purchasing specific goods they are interested in. BSEB is supposed 1) to let seller *i* find the buyer being interested in the goods offered by him; 2) to let buyer *j* find the seller offering the goods he is interested in. Sellers(S) & Buyers(B) could, for example, pay on a subscriptional basis for the realized service. Anyway, behind this not so complex general functionality, there are many issues, to be taken into account when developing such a distributed application: how to store, operate and maintain the data; how the application should provide its services to users, how some non-standard situations should be approached, etc. Therefore, it is essential to pay special attention to the modeling of such an application. The methodology introduced in Section 3 allows 1) grasping BSEB (and its functionality) as a whole and based on business process modeling; 2) elaborating precisely any part(s) of the modeled application, part(s) having importance for its operation. Below the usage of the suggested methodology for modeling such an application is illustrated.

First. After delimitation of the domain, the business processes to be supported by BSEB are explored with DEMO. Two essential business transactions (transaction types) are identified - listed in Table 1 together with their corresponding resulting fact types. The focus is only on transactions on the essential level, in order to keep the business model abstract enough so that it should remain unchanged during (eventual) re-design of its realization.

Table 1: Business Transactions List

transaction type	result fact type
T1 match-making	F1 *match <M> is made*
T2 payment	F2 *the fee for period <P> by <S/B> is paid*

Based on the transactions and result facts, the system(s) to be investigated should be selected, relevant DEMO actor(s) - identified, and their roles – determined (as customer and producer). Once this is done, all interaction relationships are determined. All this is depicted in Fig. 4, representing the Coordination Structure Model - CSM (it is incomplete, since the goal is only to illustrate the usage of DEMO within the suggested methodology).

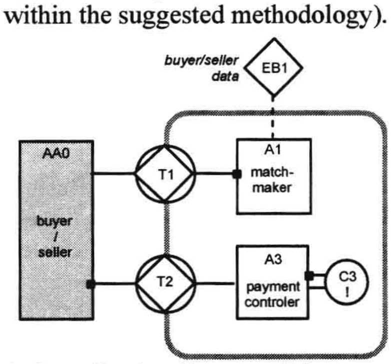

Figure 4: Coordination Structure Diagram of BSEB

The system under study (BSEB) is considered as well as the Seller and Buyer (as actors). Regarding the system under study, it is represented on the figure in more detail: actors A1 and A3 (white boxes) whereas the Seller and the Buyer are taken together in the aggregate actor AA0 (grey box) since they basically play the same role towards the actors A1 and A3. The transaction types are represented by a symbol combining a disk and a diamond symbol. The small disk C3 represents a so-called conversation for initiation. It models the periodic activation of A3 to issue payment requests. The system boundary is represented by a grey round angle. There is a so-called external bank (EB1) which contains the company data provided by the sellers and buyers. The dotted line between EB1 and A1 means that actor A1 is allowed to inspect the contents of EB1. In other words, actor A1 is allowed to know the information provided by the sellers and buyers. The reason for this allowance is that A1 needs to know the provided information. How A1 gets access and also how sellers and buyers add and remove data is not shown. These matters are considered to belong to the informational and documental perspective and thus are not represented in the (essential) CSM.

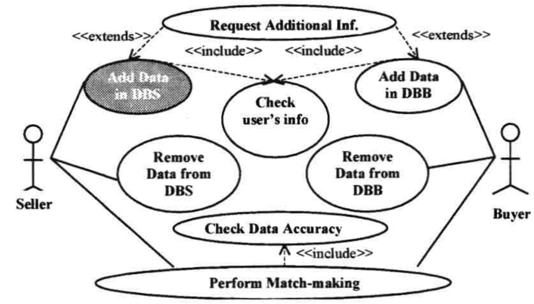

Figure 5: UC Diagram of BSEB

Second. Construction of UC diagram, based on the DEMO CSM. The diagram (Fig. 5) shows UCs and actors typical for such a BSEB. Since the purpose is just illustrative, only some of the UC and actors typical for such a system are considered. In addition to the essential business transactions (focused by DEMO), UC diagram considers also the actions which represent information providing (but are not essential business transactions), e.g. adding data to the database used by BSEB. These actions are additionally identified in building the UC diagram and are of importance for the particular application design.

Regarding the diagram, the abbreviation "DB" stands for the database, used by BSEB. For convenience, DB is virtually divided into DBS and DBB (containing data of offered and searched goods respectively). The diagram shows 2 actors: *Seller* & *Buyer*. Concerning Seller (Buyer) – he takes the decision, has the responsibility, has the goal to add data in DBS (DBB), and/or remove data from DBS (DBB), and have his data matched up with relevant data from DBB (DBS). There are 8 UC: "Add Data in DBS" (it is highlighted since it will undergo the further methodology steps), "Check User's Inf.", etc. There are 3 <<include>> relationships ("Perform Match-making" requires "Check Data Accuracy"; "Add Data in DBS" & "Add Data in DBB" require "Check User's Inf.") and two <<extends>> relationships (in some cases, before adding their data to DBS/DBP, the system might request from Seller/Buyer additional inf., so the basic UCs are "Add Data in DBS" and "Add Data in DBB", and they are extended with "Request Additional Inf.").

Third. Further investigation of any UC(s) of interest, using the concepts of Cockburn [15]. We have selected, for illustrative purpose, the highlighted UC "Add Data in DBS" and the investigation is applied to it:
UC: Add Data in DBS. **Primary Actor**: Seller. **Goal in Context**: Seller's information is added in DBS. **Scope**: System (the UC describes a person's interaction with a computer system). **Level**: Summary (the UC is executed over months or years, and is thus long-running, showing the context in which user goals operate). **Stakeholders and Interests**: Seller – wants his data correctly added in DBS; owner of BSEB – wants to be compensated for running BSEB; public – wants to be sure that the data in DBS and DBB is correct. **Precondition**: none. **Minimal guarantee**: Seller is in a position to provide correct data and pay for the service. **Trigger**: Seller decides to add

data in DBS **Main success scenario**: 1. **Seller**: decides to add data in DBS (and initiates contact with BSEB). 2. **BSEB**: provides initial information & requires ID data & credit card nr. 3. **Seller**: provides ID data & cr. card nr. 4. **BSEB**: initiates cr. card authorization & lets Seller log on. 5. **Seller**: enters BSEB & submits a form. 6. **BSEB**: checks the data provided & asks for Seller's confirmation. 7. **Seller**: confirms his will the data to be saved. 8. **BSEB**: saves the data & charges Seller's cr. card with the fee for the selected period. 9. **Seller**: logs out. Scenario's END reached. **Extensions** (only those related to activity 6 are depicted): **6a**. The data from the form submitted by Seller is incomplete. => BSEB asks Seller to submit again the form & provide complete inf, showing what is incomplete in the submitted form. Go: 5. **6b**. The data from the form submitted by Seller is irrelevant with respect to BSEB's scope => BSEB informs Seller that the data is inadequate to DBS & cancels the authorization procedure. Go: END.

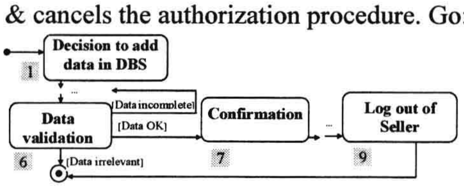

Figure 6: AD model for the UC: "Add data in DBS"

Fourth. Construction of AD model for the chosen UC. As seen from the main success scenario, there are 9 core activities (plus extensions) in the UC "Add Data in DBS". Some of them are shown on Figure 6 as an overall AD.

Finally, from the AD model it is straightforward to proceed with computer simulation. As stated in Section 3, this is left beyond the scope of this paper. The usage of an AD model as a pre-simulation model, prior to realizing simulation (e.g. using Arena), is studied in [16].

5. Conclusion

The paper's goal, as stated in the introduction, is to contribute to the knowledge on software design by introducing a methodology that allows design of ICT applications based on business process modeling. The proposed methodology (founded on an original combination of different tools) is the main contributions of this paper. It was demonstrated how the methodology could solve problems in a business area (e-business) which has significant importance for the contemporary business development. It was shown that by modeling with DEMO the essence of the business processes to be supported by an ICT application (e.g. BSEB), developers could grasp the relationship between the system to be modeled and these business processes. This is an important input for the further system modeling activities. It was shown also that the UC diagram is a helpful starting point for system modeling, providing elicitation in relation to identification of processes and requirements specification. It is of particular benefit that the diagram consists of a number of UC, which allows analysts to choose any desired UC(s) for further study. After carrying out a complete analysis of a chosen UC (following the model of Cockburn, where action steps are described within a scenario plus extensions), it is straightforward to build an AD model which helps to represent and visualize the action steps within a UC in sound graphical notations.

The suggested methodology is expected to be helpful for the development of software for the support of (business) processes in different domains.

6. References

[1] BETADE project, *Building blocks for Effective Telematics Application Development and Evaluation* (TU Delft Press, 2002).

[2] K. Liu, *Semiotics in information systems engineering* (Cambridge University Press, 2000).

[3] OMG, *UML, version 1.3* (Object Management Group, 1999; www.omg.org).

[4] J.L.G. Dietz, Understanding and Modelling Business Processes with DEMO, *Proc. ER*, Paris, FR, 1999.

[5] B. Shishkov, *Business engineering building blocks* (BETADE Project, 2001; www.betade.tudelft.nl).

[6] J. Dehnert, P. Rittgen, Relaxed Soundness of Business Processes, *Proc. 13th Int. Conf. on Advanced Inf. Systems Engineering*, Interlaken, CH, 2001.

[7] T.C. Olivera, I.M. Filho, C.J.P. Lucena, Using XML and Frameworks to Develop Information Systems, Proc. *3rd Int. Conf. on Enterprise Inf. Systems*, Setubal, Portugal, 2001.

[8] T. Hikita, M.J. Matsumoto, Business Process Modeling Based on the Ontology and First-order Logic, *Proc. 3rd Int. Conf. on Enterprise Inf. Systems*, Setubal, Portugal, 2001.

[9] P. Krutchen, *The rational unified process* (Reading, MA: Addison-Wesley Longman, 1999).

[10] J.L.G. Dietz, Business Modelling for Business Redesign, *Proc. 27th IEEE Hawaii Int. Conference on System Sciences*, Los Alamitos, CA, 1994.

[11] W.M.P. Aalst, Finding Errors in the Design of a Workflow Process: A Petri-net-based Approach, *Proc. 19th Int. Conf. on Applications and Theory of Petri Nets, WFM*, Lisbon, Portugal, 1998.

[12] F. Flores, J.J. Ludlow, *Doing and speaking in the office* (G. Fick and H. Sprague, Decision Support Systems: Issues & Challenges. NY, Perg. Press, 1980).

[13] M.A. Bunge, *Treatise on basic philosophy* (vol.4, D. Reidel Publishing Company, Dordrecht, NL, 1979).

[14] I. Jacobson, M. Christenson, P. Jonsson, G. Overgaard, *Object-oriented software engineering: a use case driven approach* (Addison-Wesley, 1992).

[15] A. Cockburn, *Writing effective use cases* (Addison-Wesley, 2001).

[16] J. Barjis, B. Shishkov, UML based Business Systems Modeling and Simulation, *Proc. 4th Int. EUROSIM Congress*, Delft, NL, 2001.

[17] B. Shishkov, J. Barjis, Analysis of a System of Telecenters as a Distributed Network, *Proc. 14th Int. Conf. on Systems for Automation of Engineering and Research*, Varna - St. Konstantin Resort, BG, 2000.

Proceedings of the IASTED International Conference
APPLIED SIMULATION AND MODELLING
June 25-28, 2002, Crete, Greece

Modelling and Verification of Business Processes*

Costin Bădică
Department of Computer Science
King's College
London, WC2R 2LS, UK
badica@dcs.kcl.ac.uk

Chris Fox
Department of Computer Science
University of Essex
Colchester, CO4 3SQ, UK
foxcj@essex.ac.uk

ABSTRACT

This paper introduces a notation for business process modelling and shows how it can be used for static verification of business processes under the assumption of single instance executions. Our approach is based on formalizing business process models as flownomial expressions [1]) and evaluating them over boolean relations. Its main advantage is simplicity: it is based on evaluating algebraic expressions to boolean relations. But it is also more restrictive then other approaches because, basically, it can only indicate those input patterns that provided to a process can cause it to enter an infinite loop without escape or a resource starvation. Nevertheless, this is useful within a tool to check processes, in order to point out problems as early as possible, before running any dynamic simulation.

KEY WORDS

Business Process Modelling, Verification, Flownomial Calculus

1 Introduction

An increased interest in applying information technology to the field of business processes has been manifested during the last decade, both in research and commercial communities, as a response to slogans like business process reengineering or total quality management. This is proven by the large number of papers published on the subject, the large number of emerging standards and proposals for representing different aspects of business processes and the large number of software tools that support tasks like business process modelling, design, analysis or simulation.

Informally, by a *business process* we mean a process that is carried out in an organization in order to achieve the organization business objectives. Because organizations are very complex artifacts. it has been claimed that carefully developed models are necessary for describing, analyzing and/or enacting the underlying business processes ([2]). A business process model is usually expressed by means of a graphical notation, that must be able to capture all the relevant information from the model and additionally must facilitate the analysis of the modelled process by static verification and/or dynamic simulation.

Analysis of business process models is very important in the context of *business process re-engineering* (BPR hereafter), because the task of BPR is to evaluate the current processes with the goal of radically revising them, in order to accommodate their improvement to new organizational needs or goals.

The INSPIRE project (IST-10387-1999) aims to develop an integrated tool-set to support a systematic and more human-oriented approach to BPR. A central aspect in INSPIRE was the development of a notation for representing business process models that is both easy to use and understand by the project partners (consultants and developers; note that consultants are not usually IT experts) and also sufficiently rigorous to allow static verification and dynamic simulation – essential tasks within a BPR context ([3]).

The notation of business process models employed in INSPIRE combines features of IDEF0 ([4]) and IDEF3 ([5]). We started with an IDEF0-based notation and then enhanced it with facilities for representing the dynamics of a business process, based on the process schematics employed in IDEF3 ([5]).

This paper introduces the INSPIRE notation and shows how it can be used for static verification of business processes under the assumption of single instance executions. Our technique is based on formalizing business process models as flownomial expressions [1]) and evaluating them over boolean relations. Other approaches proposed in the literature for verifying business processes are: model checking ([6]), process algebras [7]), knowledge-based systems ([8]) or Petri nets ([9]). One advantage of our approach is simplicity: it is based on evaluating algebraic expressions to boolean relations. Our approach is also more restrictive then the ones above. Basically, it can only indicate those input patterns that provided to a process can cause it to enter an infinite loop without escape or a resource starvation. Nevertheless, this is useful in INSPIRE to check processes and to point out problems as early as possible, before running any dynamic simulation.

The paper is structured as follows: section 2 gives an informal introduction of the notation and the rationale behind it; section 3 shows how we can describe our models as flownomial expressions; section 4 shows how we can verify a business process by evaluating flownomial expres-

*The work described here was supported by funding from the European Union as part of the Framework V INSPIRE project (IST-10387-1999)

sions; section 5 concludes the paper.

2 A Notation For Business Processes

2.1 The Basic Black Box Model

IDEF0 is a technique used to produce a function model of a new or existing system or subject area. The result of applying the IDEF0 technique is an IDEF0 model of the system. An important assumption stated in the IDEF0 standard ([4]) is: only that which is explicitly stated is necessarily implied. As a corollary, what is not (explicitly) prohibited is (implicitly) allowed. This shows that starting from an IDEF0-based notation and extending it in a consistent way is a correct approach. In our case, the extension is based on IDEF3 ([5]).

The modelling elements of IDEF0 are (i) *boxes* and (ii) *arrows*. *Boxes* represent functions defined as activities, processes or transformations and *arrows* represent data or objects related to functions. A *box* describes what happens in a designated function. A *box* has a set of inputs $\{i_1, \ldots, i_m\}$ and a set of outputs $\{o_1, \ldots, o_n\}$. We consider controls as a special kind of inputs. Mechanisms are not considered in our approach, although the notation employed by the INSPIRE tool supports them.

The crucial thing is, how we are to interpret these boxes? The IDEF0 standard ([4]) is very vague with respect to this. Basically it says that in order to produce any subset of the outputs $o_1, \ldots, o_n$ any subset of the entries $i_1, \ldots, i_m$ may be required. In order to understand this statement, we must first recognize that one intuitive interpretation in mathematical terms of an IDEF0 box is a function taking m inputs and producing n outputs. This function actually describes how we can compute the outputs on the basis on the given inputs. However, the "how" component is not explicitly modelled. Eventually, it is just suggested by the verb phrase that names the box.

Let us now return to the basic interpretation as stated in the IDEF0 standard ([4]). If our boxes are always modelling functions, it means that all the outputs will be produced if all the inputs are present. This turns out to be a very restrictive assumption, especially if the IDEF0 method is used in the early stage of requirements capture and specification. Obviously, we would not like to be very restrictive during this stage. Moreover, we would like to be able to model situations when even not all the inputs are present, at least some outputs will be produced. If we abstract away from the actual domains of values for the inputs and outputs and from how the outputs are actually produced, we finally arrive at an interpretation of boxes as boolean relations taking m inputs and producing n outputs. So formally, a box should be interpreted as an unspecified relation $rel \subseteq \{0,1\}^m \times \{0,1\}^n$.

So, our boxes are in fact black-boxes describing the dependency of subsets of outputs on subsets of inputs. A dependency is a pair of boolean strings that just says that it is possible for the underlying activity modelled by the box

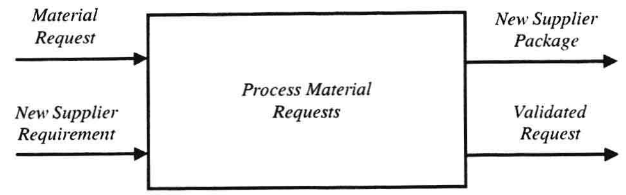

Figure 1. A business process for processing material requests

to produce the specified outputs when the specified inputs are available. For example, the pair (011,100) says that it is possible to produce only output 1 if only inputs 2 and 3 are present.

As for the arrows, they should be interpreted as flows transporting data or object items from producers to consumers. However, special attention must be paid to branching arrows (forks and joins).

According to [4], a fork is an arrow from source to use that is divided into two or more arrows. Because the IDEF0 standard is too vague here, we make the following assumption: a fork indicates the fact that the item placed onto the arrow from source to use will be made available to all of its destinations. In this case all the arrows will have the same label. If the fork is not intended to model this situation, than it must be replaced with a single input/multiple outputs activity.

Also, [4] states that a join is a point where two or more arrows from source to use are merged into a single arrow. Merging is different from splitting because it is difficult to imagine that it could happen outside an activity. That is why we have chosen to model joins as "dummy" multiple inputs/single output activities.

For example, in a manufacturing company we find a business process for material acquisition. It takes material requests and produces purchase orders and payment authorizations. It contains a subprocess for handling the material requests that takes material requests and produces validated requests. The company has a list of available suppliers, but is must be prepared to find and handle new potential suppliers. So, there is an additional input to handle new supplier requirements and an additional output to produce new supplier packages. The process is represented at an abstract level in figure 1. The IDEF0 technique allows the presentation of a model at different levels of detail. The process in figure 1 is detailed in figure 2. Figures 1 and 2 are also called *IDEF0 diagrams* ([4]).

2.2 Adding Glass-Box Views

Sometimes it is useful to be able to constrain the dependency of the outputs from the inputs. One way of doing this is by attaching to each black-box a glass-box view based on IDEF3. IDEF3 is a technique used to produce a dynamic

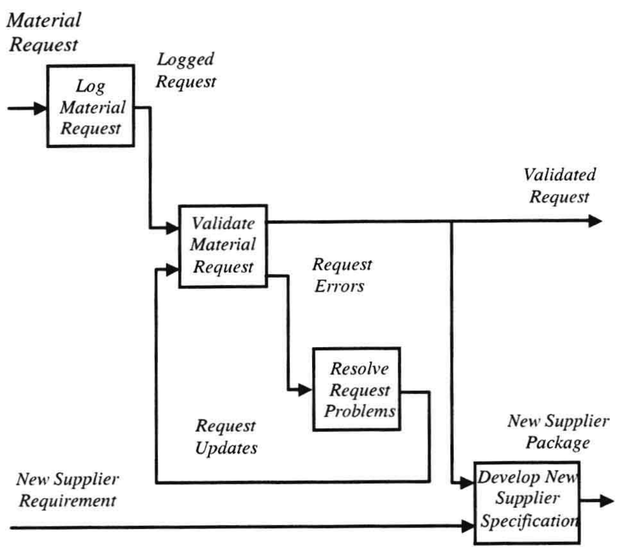

Figure 2. A more detailed presentation of the process for processing material requests

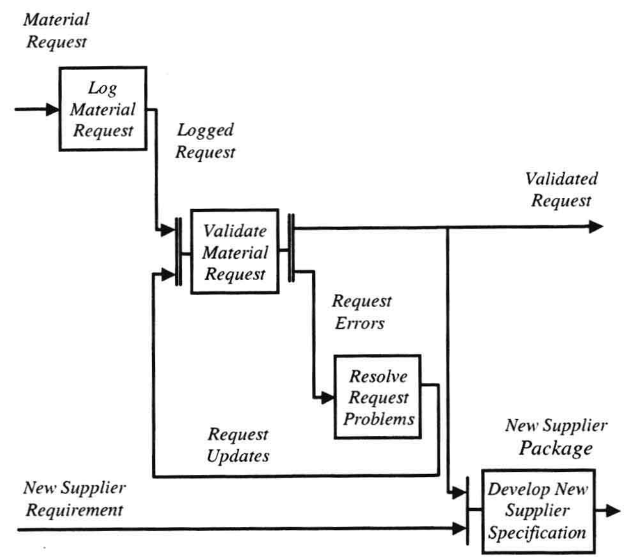

Figure 3. The result of adding glass-box views to the processing material requests process

model of the system. IDEF3 can produce two views of the system: a process-centered view and an object-centered view. Here we are considering only the process-centered view.

The main building blocks of the process-centered view of IDEF3 are (i) *units of behaviors*, (ii) *links* and (iii) *junctions*. *Units of behavior* represent types of happenings (events, acts, etc.), *links* represent temporal relations between units of behavior, and *junctions* are used to specify the logic of process branching. Within the INSPIRE we are using the term *connector* instead of *junction*, so we shall use this term hereafter.

A glass-box view contains a unit of behavior, a tree of input connectors and a tree of output connectors. There are one-to-one mappings between the inputs of a black-box and the leaves of its input tree and between the outputs of a black-box and the leaves of its output tree. The unit of behavior models the "instantiation" of the activity. The input tree models the logic of selecting the inputs participating in the activity, and the output tree models the logic of generating the outputs produced by the activity.

Let us consider the business process in figure 2. We can associate glass-box views to activities *Validate Material Request* and *Resolve Request Problems*. *Validate Material Request* may take a *Logged Request* or *Request Updates* and may produce either a *Validated Request* or *Request Errors*. *Develop New Supplier Specification* takes both a *Validated Request* and a *New Supplier Requirement* to be able to produce a *New Supplier Package*. This is modelled in figure 3.

2.3 Some Comments

The "boxes as mathematical functions" interpretation is not new. It is in fact the basic assumption employed by the formally founded description technique of business processes reported in [10]. Also the concept of "glass-box view" presented in this paper is very similar to the one used in [11]. The main goal in [10] is to document single runs of exemplary system behavior. This is quite similar to the hypothesis we use in our verification technique, namely to focus on single instance executions. Other similarities between our notation and the one described in [10] and [11] are: the use of black boxes to define process signatures and the use of glass boxes to constrain the set of behaviors of a black box.

There are however three important differences between our work and the one reported in [10] and [11]. First, we abstract away from the actual domains of the inputs and outputs and from how the outputs are produced based on the provided inputs. We are interested only what outputs are produced when some inputs are provided. Second, a basic assumption in [10] and [11] is that the process nets are acyclic, to avoid circular dependencies. In our notation we cope also with circular process nets. The feedback operator of flownomial expressions ([1]) is used to provide a semantics for such constructs. Third, the semantics of the notation from [10] is based on functions and their composition, while our approach is based on boolean relations.