Journal of Computing and Information Technology - CIT 21, 2013, 3, 161–170
doi:10.2498/cit.1002192

Process and Data Logic Integration: Logical Links between UML Use Case Narratives and ER Diagrams

Boris Jukić[1], Nenad Jukić[2] and Svetlozar Nestorov[3]

[1] School of Business, Clarkson University, Potsdam, NY, USA
[2] Quinlan School of Business, Loyola University Chicago, USA
[3] Computational Institute, University of Chicago, USA

We propose a methodology for providing clear and consistent integration of the process and data logic in the analysis stage of information systems' development lifecycle. While our proposed approach is applicable across a variety of data and process modeling schemas, in this paper we discuss it in the context of UML use cases for process modeling and ER diagrams for data modeling. We illustrate our approach through an example of modeling an execution of a retail transaction. In our example we integrate a step-by-step process model and the corresponding data model at the attribute level detail. We discuss the potential benefits of this approach by illustrating how this methodology, by providing a critical link between process and data models, can result in better conceptual testing early in the analysis process, ensuring better semantic quality of both process and data models.

Keywords: information systems development lifecycle, process model, data model, UML, ER diagrams, DFD, CRUD

1. Introduction and Background

There are many different ways of capturing both information system's process and data logic at the conceptual level. Traditional business and systems process modeling methodologies and tools range from standard or customized flow charts to more specific and detailed modeling methods. Overview of many traditional methodologies for process charting and modeling tools is available in [13]. In the last few decades the UML approach has been exceptionally prominent. The promise of Universal Modeling Language (UML) is to provide a holistic view of all the aspects of an information systems (IS) application analysis and design. UML provides a key foundation for Object Management Groups (OMG) Model-Driven Architecture®, which unifies every step of development and integration from business modeling, through architectural and application modeling, to development, deployment, maintenance, and evolution [2]. Diagrams and schemas of varying levels of complexity enable designers and developers to focus on different important aspects of information systems analysis and design at different stages of a project. In the requirements gathering and early system analysis stages, the typical UML construct that is invoked is a so-called use case. Each use case represents a particular functionality of a system under consideration that a user (actor) can interact with or receive services from. The most common initial conceptualization of a system being analyzed is that of a so called use case diagram, which provides a functional depiction of a system as a collection of use cases. Each individual use case can then be captured via textual description known as a use case narrative and/or diagrammatically, by using a UML activity diagram. Use case narratives are most widely used methods of process flow capturing in requirements validation stage, while activity diagrams are popular in the analysis phase [5]. One of the main strengths of UML use case narratives is their emphasis on clearly outlining business processes as step-by-step flows of interactions between the system and its various constituents, known as actors in UML. Therefore, the process view of a system is very clearly and consistently represented. As stated in [1], when using use case narratives, it is important to ensure that the required system functionality is adequately captured early on, which reduces the effort to accommodate unforeseen functionality changes in later stages of system development.

UML is based on object-oriented analysis and design approach. Therefore, the data view of a system is represented by various types of class diagrams (static structure class diagrams, collaboration diagrams and sequence diagrams are the most commonly used ones). In particular, UML's *static structure class diagrams* are structurally similar to and consistent with standard data modeling technique of Entity Relationship (ER) modeling [4]. Traditional ER modeling is the most widely accepted way of data requirement description at the conceptual stage. The example we present will use ER diagrams, but all the conclusions and insights are fully applicable to the situation where a UML class diagram may be used.

Research on effectiveness of various process and data models has been mostly focused on comparing benefits of different models within these two distinct groups rather than the effectiveness and benefits of the interconnectivity and mutual support of process and data models. Within the realm of data models, attention has been focused on differences and comparison of Object Oriented Relational Database (OORDB) class models that are modeled in UML vs. traditional ER models, which are often associated with a so-called structured approach to system analysis and design. A typical example of work in this area is [11]. While the investigation of comparative advantages and applicability of traditional data models and object-oriented model is important and deserves the amount of attention it has been given, the issue of mutual consistency and integration of process and data models, while equally important, has not been addressed with equal enthusiasm.

The one traditional conceptual tool for illustrating the relationship of processes and data stores is a so-called Data Flow Diagram (DFD). DFDs have typically been well covered and discussed in traditional systems analysis and design texts such as [7] or [10]. Another conceptual tool for illustrating interactions between processes and data stores is a Create, Read, Update, Delete (CRUD) matrix. CRUD matrices have been a popular approach to modeling data and process interactions enterprise wide [9].

Both DFD's and CRUD matrices in their standard form exhibit the same deficiency that our approach is designed to address. Namely, they consider each individual process facilitated by an information system under consideration as a monolith, without breaking it down into individual steps (unlike activity diagrams or use case narratives). This does not provide the opportunity to identify concrete steps /actions that interact with the system's data. In addition, DFDs and CRUD matrices typically do not consider data depositories at attribute (or property, if object-oriented) level of detail.

In this paper we expand on a framework first proposed in [6] for identifying exact data interactions necessitated by every step of each process of a system under consideration. This framework is compatible with existing diagrams for depicting data and process view of the systems. It can be implemented to provide a crucial structural connection between those two types of diagrams. In this paper we will illustrate our approach by providing a structural connection between a use case narrative and an ER diagram. Firstly, we will add a column to a standard use case narrative template that captures associated CRUD operations on all associated entities of an ER diagram. Secondly, the structural connection between the process model (use case narrative) and the data model (ER diagram) will be summarized in a separate document: an extended (attribute-level) CRUD matrix. As stated above, CRUD matrix is a conceptual modeling that has typically been used in requirements gathering and analysis as well as design stages of information systems. However, the traditional use of a CRUD matrix is at high level of granularity, listing processes and/or users as column headers, and whole data depositories and/or loosely defined entities as row headers. As such, the traditional use of CRUD matrices has been limited to providing a high level summary of user, process and data interactions. Our approach extends their use to detailed conceptual testing and mutual process and data logic consistency assurance. In our example, we will illustrate how an extended, attribute-level CRUD matrix can be used to verify that the data model contains the necessary entities and attributes that satisfy the informational needs of every step of a process being captured by the use case narrative. Our proposed approach, with its lower dedication level between data and processes, is closest to an approach described in [3], with two key differences. Our emphasis is on the conceptual stages of system development, before design models and code are generated. Secondly, we emphasize the recursive nature of development of data and process models, not taking the data model as set in place ahead of the dynamic, process model development.

2. Process and Data Logic Integration

As stated in [8], the basic goals of use case modeling should include, in addition to the system requirements specification and creation of guidelines of developers, facilitation of the system testing process at every stage of the development. In this section, we will show how our proposed methodology, in the context of an example, establishes explicit, logical and conceptually testable links between the basic units of a use case narrative and an ER diagram.

2.1. External vs. Internal System View

Before we describe our approach in more detail, it would be appropriate to address the question of whether it is appropriate to extend the purpose of use cases (or other process models with similar goals of user requirement gathering) beyond describing only the interaction between an external user and the system being built. As stated in [8] (reflecting the academic professional community consensus), the use cases should provide only a black box view for using the system, and so called "white box" interactions, describing internal behaviors of the system, separately from a conceptual process model. Given that recommendation, one may express misgivings towards our stated goal of providing explicit links between a "black box" UML use case narrative and a system's data model. As we will show in the discussion below, while our approach will not change the use case narrative's structure or content, it will indeed provide an additional connecting element between it and the data model. However, we strongly believe that the conceptual data model, defining which entities and corresponding attributes should be included in the data model, is very closely tied to external user requirements and, at the conceptual level, is very much dependent on the understanding of organizational rules, and therefore needs to be jointly developed by wide range of constituents, most of whom are indeed the external users.

2.2. Process and Data Logic Integration Example: Attribute Level CRUD Matrix

In our example, we will identify CRUD operations implied in every step of a use case narrative. This same technique can be applied in a similar fashion on any other process models that clearly identify process steps such as activity diagram or flow charts. The basic idea is to recognize that every interaction between the system and its constituent actors has potential impact on the state of data in the system, and to clearly and unambiguously identify those impacts at a proper level of detail.

The following use case narrative example describes steps of order confirmation of item(s) collected in a web shopping cart. This is one of the standard process examples widely used in practice and this particular step sequence is a slight modification of a use case narrative used

Case Name: Place Order **Actor:** Customer **Preconditions:** Customer's identity has been validated by the system Customer has invoked "Browse Catalog" and has opened a shopping cart.
Basic Flow of events:
1. Customer adds additional item to shopping cart
2. Customer selects "Review Order"
3. System calculates and displays total cost, tax and shipping charges
4. Customer selects "Place Order"
5. Customer clicks "Submit Order"
6. System displays Customer's receipt
Post Conditions: Customer leaves the screen or the Web site

Figure 1. Basic flow of the use case narrative.

	Customer	Order	OrderLineItem	Product
Review Order	cRud	cRud	cRud	cRud
Submit Order	cRud	cRUd	cRud	cRUd

Figure 2a. Corresponding CRUD Matrix.

in [12]. We chose this particular sequence since it was published with a corresponding high-level CRUD matrix which can be used to outline the differences between it and our, more detailed, attribute-level approach.

Figure 1 shows the basic flow of the use case narrative. It consists of six steps describing interaction between the actor (customer in this case) and the system in the course of placing an order for the items placed in the shopping cart online. Figure 2a shows the corresponding CRUD matrix as shown in [12], and Figure 2b shows the implied ER diagram. The capitalized operations in the table depicted by Figure 2a are executed on corresponding entities (column headers) during the particular process step (row headers). The first row of the table shows that, in the "Review Order" step, a Read operation is executed on all four entities (Customer, Order, OrderLineItem and Product). The second row shows that, in the "Submit Order" step, an Update operation is executed on Order and Product entities, while Read operation is performed on all four entities in order to show the corresponding information for all of them in the next step: generation and display of the receipt.

As seen from Figure 2b, only a few entities and CRUD interactions were identified. We will contrast this output with more detailed models created as a result of our approach. The next two figures (Figure 3 and 4) show the use case with CRUD interactions considered at every step. The rows in Figure 3 show the same use case narrative basic flow of the events as

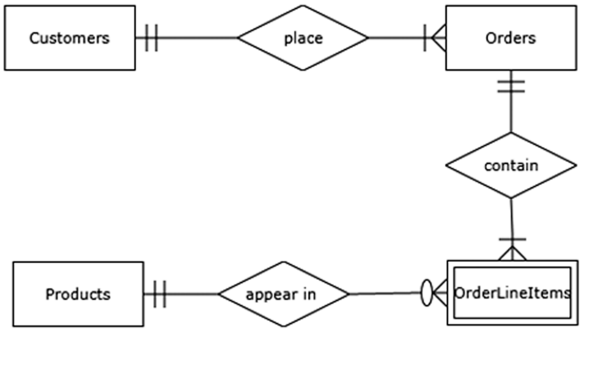

Figure 2b. Implied ER Model.

Figure 1, but with an added column that contains CRUD operations necessitated by every event in the basic flow.

It is important to emphasize that, in our approach, Figures 3 and 4 are developed in an interactive fashion, considering and re-evaluating data needs of every step in the process, as well as their effect on the state of every entity in the data model.

A complete description of this evaluation process, leading to the list of associated CRUD operations in Figure 3, as well as to the detailed version of the ER diagram in Figure 4, is included in Section 3. Here, we illustrate the process by focusing on the evaluation of step 3 "calculating and displaying total cost, tax and shipping charges" in greater depth, where the specific information needs of this step are revealed. The information about product size and weight is needed for shipping charge calculation, as well as customer address and warehouse location information for all candidate warehouses. Similarly, Customer Address is needed for tax calculation, if we assume that taxes are location-based. Once the calculation is completed, the actual line item shipping cost and shipping charge will need to be updated as well as the total order charge. Finally, in order to display all the relevant information to the customer, all attributes of the Order as well as necessary Product and Line Item attributes will need to be retrieved (read) from the system.

By adopting this approach in the early conceptual modeling stage, the picture that emerges reveals a much more complete data model containing all necessary data elements and all possible CRUD interactions. Figure 4 shows the detailed ER model with all needed data entities and their attributes needed to complete the process outlined in the narrative. It is important to re-emphasize that Figures 3 and 4 are developed in a recursive fashion, whereby consideration of data needs of every step in the process model in Figure 3 results in further re-evaluation and modification of the data model in Figure 4. Conversely, every data model in Figure 4 needs to have a justification in a process

Case Name: Place Order
Actor: Customer
Preconditions:
Customer's identity has been validated by the system.
Customer has invoked "Browse Catalog" and has opened a shopping cart
	`cRud Customer(CustomerID)`
	
	`Crud Orders(OrderID,OrderStatus)`

Basic Flow of events:	**Associated CRUD operations**

1. Customer adds additional item to shopping cart	`cRud Products(ProductPrice)`
	`cRud Customer(CustomerOtherAttributes)`
	`Crud OrderLineItems(LineItemCharge,LineItemQuantity,OrderLineItemNumber)`

2. Customer selects "review order"	`cRud Orders(OrderID)`
	`cRud Products(all attributes)`
	`cRud OrderLineItems(LineItemCharge,OrderLineItemNumber)`

3. System calculates and displays total cost, tax and shipping charges	`cRud OrderLineItem(LineItemQuantity)`
	`cRud Products(ProductWeight)`
	`cRud Warehouses(WarehouseAddress)`
	`cRud Customers(CustomerAddress)`
	`crUd OrderLineItems(LineItemShippingCharge)`
	
	`cRud OrderLineItems(LineItemCharge,LineItemShippingCharge)`
	`crUd Orders(LineItemTotalCharge,ShippingTotalCharge,PreTaxOrderTotal)`
	`cRud Customers(CustomerAddress)`
	`crUd Orders(OrderTaxCharge,CompletedTotalOrderCharge)`
	
	`cRud Products(ProductName)`
	`cRud OrderLineItems(LineItemQuantity,LineItemCharge,LineItemShippingCharge)`
	`cRud Orders(LineItemTotalCharge,ShippingTotalCharge,PreTaxOrderTotal,`
	`OrderTaxCharge,CompletedTotalOrderCharge)`

4. Customer selects "Place order"	

5. Customer clicks "Submit Order"	`crUd Orders(OrderDateTime,OrderStatus)`
	`Crud Shipments(all attributes)`
	`cRud OrderLineItems(LineItemQuantity)`
	`crUd InventoryLevels(all attributes)`

6. System displays Customer's receipt	`cRud Customer(all attributes)`
	`cRud Orders(all attributes)`
	`cRud OrderLineItems(all attributes)`
	`cRud Products (all attributes)`
	`cRud Shipments (all attributes)`
	`cRud Warehouses (all attributes)`

Post Conditions: Customer leaves the screen or the Web site	

Figure 3. Modified Use Case Narrative with Associated CRUD Operations.

model. The process is not linear, i.e. development of the data model is not preceded by the completed development of the process model. Rather, both models result from the simultaneous process and data analysis through a repeated data needs audit and conceptual testing of process and data consistency.

Figure 5 summarizes all the CRUD interaction shown in Figure 3 in an extended, attribute and step-level CRUD matrix pointing out all CRUD interactions for every attribute of affected entities. The reduced (single letter) notation is used in order to save space. This matrix illustrates the potential of this approach for conceptual testing. The audit of every entity is possible on a column-by-column basis, as well as by detailed analysis of every process step on a row-by-row basis. For example, the process steps that need two separate CRUD descriptions (such as step 3) may be considered as three separate steps that can be split up in subsequent analysis. Another example would be identifying an entity or entity attribute that is not used by any process step and deciding that it is not needed.

A larger scale implementation of this approach on every process of a system would ensure complete mutual consistency between all the processes and data depositories. An audit procedure can be designed, for example, to ensure that for every entity in the system's complete data model, at least one process exists that creates instances of that entity. If such process

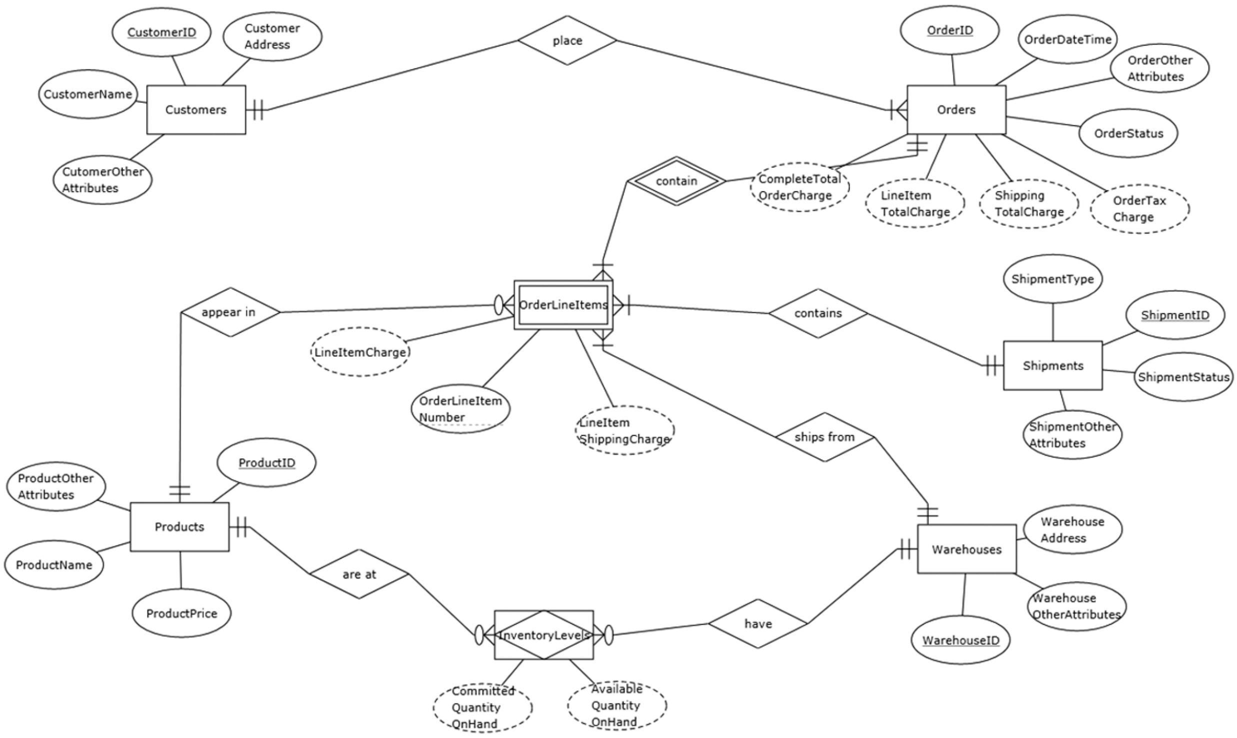

Figure 4. Detailed ER Diagram.

	Customers				Orders				Order Line Item				Shipments				Products				Inventory Levels		Ware-houses	
	Customer ID	Customer Name	Customer Address	Customer Other Attributes	OrderID	Order Date Time	Order status	Order Charges – (all charges merged)	LineItemNumber	LineItemCharge	LineItemQuantity	LineItemShippingShipping Charge	ShipmentID	ShipmentStatus	ShipmentType	ShipmentOtherAttributes	ProductID	ProductName	ProductsPrice	ProductWeight	CommitedQuantityOnHand	AvailableQuantity OnHand	WarehouseID	WarehouseAddress
1. Customer adds additional item to shopping cart				R					C	C	C								R					
2. Customer selects “review order”					R				R	R							R	R	R	R				
3. System calculates and displays total cost, tax and shipping charges			R R					R U R	R	R R	R	U R R					R	R		R				R
4. Customer selects “Place order”																								
5. Customer clicks “Submit Order”						U	U					R	C	C	C	C					U	U		
6. System creates and displays Customer’s receipt	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R			R	R

Figure 5. Process – Data Interaction CRUD Matrix on Attribute Level.

is not detected, the process model is not complete. Other needed data operations can be accounted for system-wide data model in similar fashion. Also, if a process step is detected that implies an information interaction or multiple interactions that cannot be accommodated by the current system-wide data model, the data model is not complete. Conversely, unnecessary data elements not needed for facilitation of any of the processes this system is being designed to facilitate may be detected, leading to reduction and simplification of the system wide-data model.

3. Development of the Detailed ER Diagram and the Associated CRUD Matrix from the Use Case Basic Flow Narrative

3.1. Model Assumptions

Before we provide a detailed description of all implied data interactions of every event in the basic flow use case narrative, let us clarify some of the working assumptions.

The use case narrative used in this example depicts only the basic flow of the events, representing the most typical sequence of events. Use cases may and very often do contain many alternative flows, exceptional event flows, recursive loops etc. These flows will of course have their own implied interactions with elements of the data model, and can be analyzed in the same fashion as the basic flow. The other simplifying assumption at this conceptual stage is that all the information needs of every event in this narrative are handled by persistent (permanently stored) attributes of. Later in the design stage, a decision can be made as to which, if any, of the calculated values really need to be stored as attribute values of entities or if some or all of them can be temporarily stored as system variables only to be used during the lifetime of the process for which they are needed.

3.2. Step-by-step Event Flow

Let us outline every event of this basic flow in terms of its needed interactions with the elements of the data model, starting with the preconditions that specify the state our system has to be in before the flow of events can resume. The first precondition assumes that the customer is known to the system at this point, having its credentials validated prior to the creation of the order. This translates to the informational requirement that the value of the appropriate *CustomerID* attribute of the *Customer* entity is known (read) by the system. Another precondition assumes that the shopping cart has been created, and in return a provisional instance of *Orders* entity has been created with a unique value set for the *OrderID attribute*, the value of *OrderStatus* attribute set to a default "incomplete" value. We will assume that the values of other attributes of this entity are set to null at this initial stage of order completion. This is all summarized by the following expressions:

```
cRud Customer(CustomerID)
Crud Orders(OrderID,OrderStatus)
```

The first event: <u>Customer adds additional item to the shopping cart</u>, represents the act of customer choosing a desired product in the quantity of one or more and adding it to its existing order. This event results in the creation of a new instance of *OrderLineItems* entity with the *OrderLineItemNumber* attribute set to an appropriate value (for example, representing its position in the order list). The value of the *LineItemQuantity* attribute is set to the quantity specified by the customer and *LineItemCharge* derived attibute is set to the value of line *LineItemQuantity* attribute multiplied by the *ProductPrice* attribute of the chosen instance of the *Products* entity minus any applicable discounts that may depend on the date (available from the system clock) and/or one or more attributes of the current instance of the *Customers* entity (such as customer status or a demographic profile, abstracted here as *CustomerOtherAttributes*). This can be summarized as:

```
cRud Products(ProductPrice)
cRud Customer(CustomerOtherAttributes)
Crud OrderLineItems(LineItemCharge,
  LineItemQuantity,OrderLineItemNumber)
```

The second event: <u>Customer selects "Review Order"</u> represents the step in which customer decides to review his/her order prior to its final completion. This event requires certain information to be displayed to the customer interface. Therefore, this information needs to be retrieved (i.e. read) by the system. We will assume that the information displayed to the customer consist of the value of the *OrderID* attribute of the current *Order* entity instance, and for each item chosen for the order, all the corresponding values of the corresponding *Products* entity instance attributes (including the original product price) for each line item, plus the quantity chosen and the price actually charged for each line item, which are the values of the *LineItemQuantity* and *LineItemCharge* attributes of the *OrderLineItem* entity instance, as well as the item's order line item number, which is the value of the attribute *OrderLineItem* in the same entity instance. This can be summarized as:

```
cRud Orders(OrderID)
cRud Products(all attributes)
cRud OrderLineItems(LineItemCharge,
  LineItemQuantity,OrderLineItemNumber)
```

The third event: System calculates and displays total cost, tax and shipping charges represents a series of steps. In the first step, the shipping charge and tax are calculated for every line item in the order. We will assume that, in order for each item's shipping charge to be calculated, in addition to the value of the *LineItemQuantity* attribute for each instance of the *OrderLineItem* entity, the value of the attribute *ProductWeight* is needed from the corresponding *Product* entity instance. Additionally, we will assume that calculation of shipping charges requires values of *WarehouseAddress* attributes for all feasible instances of *Warehouses* entity as well as the value of *CustomerAddress* for the current instance of the *Customer* entity. For example, an algorithm for calculating shipping charges may include a search to find the nearest warehouse to the customer that has all the products listed in the order. Once the shipping charges are calculated, they are used to update the value of the *LineItemShippingCharge* attribute of every instance of the *OrderLineItems* entity that corresponds to the current order. The data operations for this step can be summarized as:

```
cRud OrderLineItem(LineItemQuantity)
cRud Products(ProductWeight)
cRud Warehouses(WarehouseAddress)
cRud Customers(CustomerAddress)
crUd OrderLineItems
  (LineItemShippingCharge)
```

In the second step of this event, total cost of the order is calculated. In order to accomplish that, the values of the *LineItemCharge* and *LineItemShippingCharge* of all corresponding instances of the *OrderLineItem* entity are retrieved (read) in order to be added up and to update the values of the attributes *LineItemTotalCharge, ShippingTotalCharge* and *PreTaxOrderTotal* attributes of the *Orders* entity instance. Then the value of the *OrderTaxCharge* attribute is calculated and updated and this value is added up to the value of the *PreTaxOrderTotal* in order to calculate and update the value of the *CompletedTotalOrderCharge* attribute. We will assume that, in this example, taxes are location-based and that tax calculation requires the value of the *CustomerAddress* attribute of the current customer. This can be summarized as:

```
cRud OrderLineItems(LineItemCharge,
  LineItemShippingCharge)
crUd Orders(LineItemTotalCharge,
  ShippingTotalCharge,PreTaxOrderTotal)
cRud Customers(CustomerAddress)
crUd Orders(OrderTaxCharge,
  CompletedTotalOrderCharge)
```

The final step of this event displays all the calculated charges, by retrieving quantity and all charge attributes for each corresponding instance of the *OrderLineItem* entity (also showing the value of the *ProductName* attribute for each corresponding instance of the *Product* entity), as well as all charge attributes for the *Order* entity. This can be summarized as follows:

```
cRud Products(ProductName)
cRud OrderLineItems(LineItemQuantity,
  LineItemCharge,LineItemShippingCharge)
cRud Orders(LineItemTotalCharge,
  ShippingTotalCharge,PreTaxOrderTotal,
  OrderTaxCharge,
  CompletedTotalOrderCharge)
```

Given that this event has several clearly identifiable steps with specific data interactions, it may be decided in subsequent analysis that it is beneficial to split it into several consecutive events.

The fourth event Customer selects "Place order" is strictly navigational, representing customer's selection of the menu item that will allow him/her to place the order in the next step. It has no interaction with any of the data elements, and it may be merged with the subsequent step in the next revision of this use case narrative.

The fifth event Customer clicks "Submit Order" represents the customer's commitment to complete the current order, which triggers the shipment generation. This event results in updating of *OrderDateTime* attribute value of the *Orders* entity instance to the current state of system clock, as well as update of the *OrderStatus* attribute value of the same entity from the default "incomplete" to "complete". Also, a new instance of the Shipments entity is created, with all the necessary attributes. This event also results in the updating of values of the attributes *CommittedQuantityOnHand* and *AvailableQuantityOnHand* for all corresponding instances of the *InventoryLevel* entity. For every product corresponding to the line item shipped from a certain warehouse, the committed quantity on hand has to increase by the product quantity being ordered and shipped, and available quantity on hand has to be reduced by that same quantity. In order to perform this calculation, the value of the *LineItemQuantity* attribute of every corresponding instance of the *OrderLineItem* entity will needed to be retrieved. This can all be summarized as:

```
crUd Orders(OrderDateTime,OrderStatus)
Crud Shipments(all attributes)
cRud OrderLineItems(LineItemQuantity)
crUd InventoryLevels(all attributes)
```

The sixth and last event of this basic flow represents the final display of all relevant order and shipment information to the customer, signifying the system's acknowledgement that the transaction is successfully completed and its fulfillment is underway. We will make a simplifying assumption that all attributes of all relevant entities will be displayed to the customer. Later, in the design stage, the list of attributes that are of actual interest to the customer for confirmation purposes may be reduced. Note that, even at this early conceptual stage, we acknowledge that customer need not be aware of how his/her order fulfillment is affecting inventory levels, and that entity is excluded from the display. This can be summarized as follows.

```
cRud Customer(all attributes)
cRud Orders(all attributes)
cRud OrderLineItems(all attributes)
cRud Products (all attributes)
cRud Shipments (all attributes)
cRud Warehouses (all attributes)
```

This detailed description outlines the rationale and the thought process that will occur as a result of inclusion of the data interaction description in the use case narratives. As a consequence, much more detailed and accurate ER model will emerge, even at this early conceptual modeling stage.

4. Conclusion and Directions for Future Research

Benefits of this proposed framework are manifold, but the overarching applicability is in enabling systematic conceptual testing of both process and data models, providing more opportunity to refine requirements and evaluate the data needs of each individual process step in early stages of the systems development lifecycle. It also provides a clear logical link between the conceptual process and data models before more detailed design models are to be considered. Our approach promises to improve data model's semantic quality by having the potential to reveal all the needed entities and attributes and eliminate unnecessary ones. Conversely, it also has potential to improve process model's quality by detecting and improving poorly defined process steps.

We plan to use this proposed framework as a basis for several further research initiatives. One extension will focus on developing and testing a tool that will use above outlined principles to enable visual use case narrative creation with visual links to ER model, and in final phase visual programming interface for code and DB development. Another direction will be to investigate the ability of this approach by facilitating a more efficient and immediate Extraction, Transformation and Loading (ETL) process by identifying data capture opportunities in operational systems that can be implemented directly to the Data Warehouse Dimensional Model. We expect that using our approach to identify data capture opportunities early in the conceptual stage of Data Warehouse Design will provide an excellent opportunity to assess and improve its implementation details and economic feasibility. Finally, another research direction will focus on development of a formal testing procedure and methodology for semantic completeness of both process and data models, by exploring the possibilities for the formal quantitative treatment and analysis as facilitated by our proposed framework.

References

[1] N. BOLLOJU, S. SUN, Benefits of Supplementing Use Case Narratives with Activity Diagrams – An Exploratory Study. *The Journal of Systems and Software*, **85** (2006), 2182–2191.

[2] G. BOOCH, J. RUMBAUGH, I. JACOBSON, *The Unified Modeling Language User Guide*. (2nd Edition), Addison-Wesley, 2005.

[3] D. BRANDON, CRUD matrices for detailed object-oriented design. *Journal of Computing Science in Colleges*, **18**(2) (December 2002), 306–322.

[4] T. CARTE, J. JASPERSON, M. CORNELIUS, Integrating ERD and UML Concepts when Teaching Data Modeling. *Journal of Information Systems Education*, **17**(1) (Spring 2006), 55–63.

[5] B. DOBING, J. PARSONS, How UML is Used. *Communications of the ACM*, **49**(5) (2006), 109–113.

[6] B. JUKIĆ, N. JUKIĆ, S. NESTOROV, Framework for Integrating Process and Data Logic: Connecting UML Use Cases and ER Diagrams. *Proceedings of the 35th International Conference on Information Technology Interfaces, ITI 2013*, (June 2013) Cavtat, Croatia.

[7] K. KENDALL, J. KENDALL, *Systems Analysis and Design*. Prentice Hall, (5th Edition), 2001.

[8] L. MATTINGLY, H. RAO, Writing effective use cases and introducing collaboration cases. *Journal of Object-oriented Programming*, (Oct 1998), 77–84.

[9] A. POLITANO, Salvaging Information Engineering Techniques in the Data Warehouse Environment. *Computer Technology Review*, **21**(2) (Feb 2001), 53–56.

[10] J. W. SATZINGER, R. JACKSON, S. BURD, *Systems Analysis and Design in a Changing World*. (3rd Edition), Course Technology, 2003.

[11] S. SIRCAR, S. NERUR, R. MAHAPATRA, Revolution or evolution? A comparison of object-oriented and structured systems development methods. *MIS Quarterly*, **25**(4) (Dec 2001), 457–471.

[12] M. WANG, Using UML for Object-relational Database Systems Development: A Framework. *Issues in Information Systems*, **9**(2) (2008), 538–543.

[13] B. YU, BING, D. WRIGHT, Software Tools Supporting Business Process Analysis and Modeling. *Business Process Management Journal*, **3**(2) 133–150.

Received: July, 2013
Revised: October, 2013
Accepted: October, 2013

Contact addresses:

Boris Jukić
School of Business
Clarkson University
372 Bertrand H. Snell Hall
Potsdam
NY 13699-5790
USA
e-mail: bjukic@clarkson.edu

Nenad Jukić
Quinlan School of Business
Loyola University Chicago
1 E Pearson
Chicago
IL 60661
USA
e-mail: njukic@luc.edu

Svetlozar Nestorov
Computational Institute
University of Chicago
Ryerson 275A
Chicago
IL 60637
USA
e-mail: evtimov@cs.uchicago.edu

BORIS JUKIĆ is a professor of information systems and the Associate Dean of Graduate Programs at the School of Business at Clarkson University in Postdam, New York. Dr. Jukić has been teaching undergraduate, graduate, and executive education classes in the Clarkson University School of Business since 2004. Between 1998 and 2004, Dr. Jukić was an assistant and associate professor of information systems at the School of Management at George Mason University in Fairfax, Virginia. He received his undergraduate degree in computer science and electrical engineering from the School of Computing and Electrical Engineering at the University of Zagreb in Croatia, his MBA from Grand Valley State University in Allendale, Michigan and his Ph.D. in management science and information systems from the University of Texas in Austin, Texas, where he was also a research assistant and instructor. Dr. Jukić conducts research in various information technology–related areas, including e-commerce, networks infrastructure, business process modeling, database modeling and management, data warehousing and IT strategy. His work has been published in numerous management information systems and management science academic journals, conference publications, and books.

NENAD JUKIĆ is a professor of information systems and the director of the graduate certificate program in business intelligence and data warehousing at the Quinlan School of Business at Loyola University Chicago. Dr. Jukić has been teaching undergraduate, graduate, and executive education classes in the Information Systems and Operations Management Department at the Quinlan School of Business since 1999. Between 2005 and 2007, he was also a visiting professor of information systems at the Beijing International MBA Program at the China Center for Economic Research at Peking University, Beijing, China. Between 1997 and 1999, he taught at the School of Computing and Information Systems at Grand Valley State University in Allendale, Michigan. Dr. Jukić received his undergraduate degree in computer science and electrical engineering from the School of Computing and Electrical Engineering at the University of Zagreb in Croatia, and received his M.S. and Ph.D. in computer science from the University of Alabama in Tuscaloosa, Alabama. He conducts research in various information technology–related areas, including database modeling and management, data warehousing and business intelligence. His work has been published in numerous management information systems and computer science academic journals, conference publications, and books. In addition to his academic work, Dr. Jukić also worked on projects for corporations and organizations, varying from startups to Fortune 500 companies to U.S. government and military agencies.

SVETLOZAR EVTIMOV NESTOROV is a senior research associate at the Computation Institute at the University of Chicago. Previously, he was an assistant professor in computer science at the University of Chicago, where he taught databases and computer systems to undergraduate and graduate students. While on leave, he co-founded Mobissimo, a venture-backed travel search engine that was chosen as one of the 50 coolest Web sites by Time magazine in 2004. Dr. Nestorov received his undergraduate degrees in computer science and mathematics from Stanford University and his M.S. in computer science from Stanford University. He received his Ph.D. in computer science from Stanford University with a dissertation titled "Data Mining Techniques for Structured and Semistructured Data." His advisor was Professor Jeffrey Ullman. Svetlozar leads the design and development of the data warehouse project at the Nielsen Data Center at the Kilts Center for Marketing, which is part of the Chicago Booth School of Business. His research interests also include data mining, high-performance computing, and Web technologies.