Computers in Human Behavior 57 (2016) 352–376

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

Process-mining enabled feedback: "Tell me *what* I did wrong" vs. "tell me *how* to do it right"

Gayane Sedrakyan*, Jochen De Weerdt, Monique Snoeck

K.U.Leuven, Faculty of Business and Economics, Department of Decision Sciences and Information Management, Research Center for Management Informatics (LIRIS), Naamsestraat 69, B-3000, Leuven, Belgium

ARTICLE INFO

Article history:
Received 25 June 2015
Received in revised form
16 December 2015
Accepted 17 December 2015
Available online 31 December 2015

Keywords:
Teaching/learning modeling
Domain modeling
Conceptual modeling
Process-oriented feedback
Modeling patterns
Information systems education
Process mining
Learning analytics

ABSTRACT

Fast advancement of technology has led to an increased interest for using information technology to provide feedback based on ***learning behavior*** observations. This work outlines a novel approach for analyzing behavioral learner data through the application of ***process mining*** techniques specifically targeting a ***complex problem solving process***. We realize this in the context of one particular learning case, namely, domain modeling. This work extends our previous research on process-mining analysis of domain modeling behavior of novices by elaborating with new insights from a replication study enhanced with an extra observation on how novices verify/validate models. The findings include a set of typical ***modeling and validation patterns*** that can be used to improve teaching guidance for domain modeling courses. From a scientific viewpoint, the results contribute to improving our knowledge on the cognitive aspects of problem-solving behavior of novices in the area of domain modeling, specifically regarding ***process-oriented feedback*** as opposed to traditional ***outcome feedback*** (is a solution correct? Why (not)?) usually applied in this type of courses. Ultimately, the outcomes of the work can be inspirational outside of the area of domain modeling as learning event data is becoming readily available through virtual learning environments and other information systems.

1. Introduction

In the modern ICT-driven world the quality of information systems is critical but even more importantly, the value of information systems is largely dependent on their alignment with business strategy and operations. Therefore, in enterprise information systems engineering, enterprise modeling (also sometimes coined as business modeling) is an important step next to (or even prior to) developing an application and technological architecture (TOGAF, 2015).

Typically, an enterprise model addresses different dimensions of the enterprise such as the "what", "how", "who", "when", "where" and "why" (Bernaert, Poels, Snoeck, & De Backer, 2014; Zachman, 1987). These aspects are captured through different kinds of models such as goal models (capturing the "why"), organizational charts (capturing the "who"), conceptual data models capturing business objects and their relations ("what"), and business processes models (capturing the "how", related to "who" and "what"). Each of these models captures a particular "view" of the enterprise covering one or more aspects and abstracting away other aspects. While working with different views is a powerful mechanism to master complexity, obviously, those views are not totally independent, but need to present an integrated view of an enterprise and should therefore be consistent with each other. Enterprise modeling is therefore a complex problem solving process, and teaching it requires a careful scaffolding of learning tasks, addressing the modeling of individual aspects first, but ultimately addressing the capability of developing an integrated model consisting of several mutually consistent views.

While different aspects of the enterprise can be described through natural language, it is common practice to capture the different views through the creation of formal (graphical) models because models enable quality control at a level impossible to reach with requirements formulated in natural language (Sikora, Bastian, & Pohl, 2011). Models are represented as diagrams, using specific modeling languages, such as BPMN for business process diagrams, and UML for class diagrams. A particular feature of UML is that this language is based on the principles of object orientation, hence

* Corresponding author.
E-mail addresses: Gayane.Sedrakyan@kuleuven.be (G. Sedrakyan), Jochen.DeWeerdt@kuleuven.be (J. De Weerdt), Monique.Snoeck@kuleuven.be (M. Snoeck).

http://dx.doi.org/10.1016/j.chb.2015.12.040
0747-5632/

allowing capturing part of the "how" aspects related to behavior of business objects. When using UML for describing business objects, their relationships and their behavior, one therefore speaks about "domain modeling". As an example, a sales company can be described through a collection of business processes drawn in BPMN, capturing the workflows associated to registering new orders, issuing invoices, shipping goods to customers, handling complaints and so on. An object-oriented domain model will be described through several sub-views. The UML class diagram (the ***structural view***) will capture business objects (such as customer, order, invoice, product), and their associations (e.g. each order belongs to exactly one customer; a customer can have zero to many outstanding orders). UML statecharts capture the potential states a business objects can be in (a customer may be blacklisted or not; an order may be invoiced, paid, shipped, …; a product may be available, out-of-stock, …) and which actions are allowed in particular states or not (e.g. adding products to an already paid order is not allowed), and how actions may cause transition to a next state (e.g. "pay" causes the transition to the state "paid"). Finally, an interaction model will capture the required coordination across business objects. For example, adding a product to an order requires checking the state of an order (is the order still modifiable ?), the state of the product (is the product available ?) and potentially even the state of the customer (is the customer not blacklisted ?). If adding the product is permitted, then both the order, and the stock level of the product need to be updated. The statecharts and interaction model together form the ***behavioral view***.

Teaching modeling skills to novice business analysts is a challenging task considering that system analysis is by nature an inexact skill and the tacit knowledge the experts gain over time is difficult to transfer to novices. As stated by (Schenk, Vitalari, & Davis, 1998), "in their early careers junior system analysts produce incomplete, inaccurate, ambiguous, and/or incorrect information requirements". Translating these requirements into correct models consisting of different partial views that need to be consistent with each other, adds yet another layer of difficulty.

The adoption of UML as a modeling language has become very prominent since the introduction of the Model Driven Architecture (MDA) framework and the Model Driven Engineering (MDE) approach to software development (Bézivin, 2006). MDA and MDE recognize that models are the foundation of software system development by focusing on automated code generation from models and hence shifting the focus of software quality assurance from system implementation (software testing) towards system modeling (model verification and validation).

Despite the dominance of UML there is a certain degree of difficulty in understanding a system represented by means of UML diagrams (Bavota et al., 2011; Otero & Dolado, 2004; Siau & Cao, 2001; Siau, Erickson, & Lee, 2005). Previous research has indicated several reasons: among which (1) the level of structural complexity of UML exceeding the limits of human working memory (cognitive load) in terms of the ability for effective information processing (Cruz-Lemus, Genero, & Piattini, 2008; Cruz-Lemus, Maes, Genero, Poels, & Piattini, 2010; Erickson & Siau, 2007; Wilmont, Hengeveld, Barendsen, & Hoppenbrouwers, 2013); and (2) lack of comprehension methodologies (Erickson & Siau, 2007) and, in particular, its impreciseness about the combination of interactive, structural and behavioral aspects together in a single model (Gustas, 2010). Furthermore, (3) it is not easy to find relevant subsets suitable for a modeling goal. The (4) "noisiness" of UML with variety of concepts can result in models that use the misused language concepts in a way not intended for the modeling domain (Buckl, Matthes, & Schweda, 2010). Finally, (5) there is a lack of validation guidance and tool support for model testing (Shanks, Tansley, & Weber, 2003).

In a teaching context, model comprehension difficulties are additionally associated with the insufficient level of experience, i.e. domain knowledge, of novices and as a result their limited cognitive resources to identify relevant triggers for model verification (Bradley, Paul, & Seeman, 2006; Damassa & Sitko, 2010; Schenk et al., 1998). According to complexity analysis by Siau and Cao, 2001 UML class diagram ranks the highest in complexity among the structural diagrams followed by statecharts among the dynamic diagrams (Carbone & Santucci, 2002; Cruz-Lemus, Genero, Manso, Morasca, & Piattini, 2009; Cruz-Lemus, Genero, Morasca, & Piattini, 2007; Genero, Miranda, & Piattini, 2003) because of their high cognitive and structural complexity (Cruz-Lemus et al., 2008, 2010).

The work presented in this paper relies on the teaching of enterprise modeling, according to the method MERODE,[1] an Enterprise Information Systems engineering methodology that has been developed by the Management Informatics research group at the faculty of Business and Economics, KU Leuven (Snoeck, 2014). In MERODE an enterprise model consists of a collection of business process models (making use of BPMN as modeling language) and an object oriented domain model (making use of the UML). To alleviate some of the above-mentioned problems related to UML, MERODE uses simplified versions of the UML class diagram and statecharts. In a Delphi[2] study by Erickson and Siau, 2007 identifying the kernel of "essential" UML (i.e. diagrams that are highly used) class diagram and statecharts are found to have the highest usability ranks by practitioners and educators from software industry and academic field with the relative importance rate of 100%. Furthermore these are also among the top used diagrams present in the context of educational material such as books, tools, courses and tutorials (with percentages of 100% (class diagram) and over 96% (statecharts) (Reggio, Leotta, Ricca, & Clerissi, 2013) also being in the subset of diagrams that provide a support for conceptual modeling goals (Embley & Thalheim, 2012). Furthermore, to maintain the high abstraction level required for enterprise modeling, the interaction model has been replaced by a CRUD-table, a technique borrowed from Information Engineering (Martin, 1990). As a result, a MERODE domain model uses three simplified sub-views: an existence dependency graph (EDG) describing business objects and their associations through a simplified UML class diagram, 2. finite state machines (FSMs), a simplified form of UML statecharts, to capture the individual behavior of business objects, and 3. a CRUD-table to capture business object interactions. For the given example, the different models would look like in Fig. 1. The method is supported by a tool JMermaid.[3] While this tool offers some built-in features for basic consistency checks (Snoeck, Michiels, & Dedene, 2003), the modeler needs to actively cross-validate the different views to ensure an integrated perspective on the system-to-be.

While MERODE contributes to modeling quality through simplification and intelligent tool support, this doesn't guarantee the quality of the outcome of the modeling process: for a same modeling task, large variations in quality of the obtained models can still be observed. Recently new research domain emerged that investigates the process of process modeling aiming at understanding of how humans model and if and how modeling styles can affect the quality of the modeling process outcome. Those studies are however limited to the process of business process modeling. So far, no research was found that observes a modeling process involving conceptual data modeling, object oriented domain

[1] Attempt to reach a reliable consensus by incorporating opposing views from experts in specialized areas.

[2] http://merode.econ.kuleuven.ac.be/mermaid.aspx.

[3] Disco is a commercial tool developed by Fluxicon: http://fluxicon.com/disco/.