Object-Centric Behavioral Constraint Models: A Hybrid Model for Behavioral and Data Perspectives

Guangming Li
Eindhoven University of Technology
Eindhoven, The Netherlands
g.li.3@tue.nl

Renata Medeiros de Carvalho
Eindhoven University of Technology
Eindhoven, The Netherlands
r.carvalho@tue.nl

Wil M.P. van der Aalst
RWTH Aachen University
Aachen, Germany
wvdaalst@pads.rwth-aachen.de

ABSTRACT
In order to maintain a competitive edge, enterprises are driven to improve efficiency by modeling their business processes. Existing process modeling languages often only describe the lifecycles of individual process instances in isolation. Although process models (e.g., BPMN and Data-aware Petri nets) may include data elements, explicit connections to real data models (e.g., a UML class model) are rarely made. Therefore, the Object-Centric Behavioral Constraint (OCBC) modeling language was proposed to describe the behavioral and data perspectives, and the interplay between them in a single, hybrid diagram. In this paper, we describe OCBC models and introduce the extended interactions between the data and behavioral perspectives on the attribute level. We implement the approach in a plugin and evaluate it by a comparison with other models.

CCS CONCEPTS
• Applied computing → Enterprise computing; Business process management; Business process modeling.

KEYWORDS
Business process modeling, Object-Centric, Databases, Class models, Object models, Declarative constraints

1 INTRODUCTION
Information systems are widely used by enterprises and organizations to support their business process executions. Currently, the information systems one encounters in most organizations are object-centric, i.e., transactions related to some type of objects (e.g., customer orders) are stored in the same database table (e.g., the “order” table) on data perspective, and events are recorded implicitly (e.g., in redo logs) and separately without a common case id. Some examples of these systems are Customer Relationship Management (CRM) and/or Enterprise Resource Planning (ERP) which provide business functions such as procurement, production, sales, delivery, finance, etc. In this paper, we abstract business processes on such systems as three perspectives: control-flow (i.e., the behavioral perspective), data schema (i.e., the data perspective) and communications (indicating how activities in control-flow impact the corresponding tables in databases) in between, as shown in Figure 1.

Figure 1: A business process on object-centric information systems can be modeled as three perspectives: control-flow, data schema and communications.

Existing process modeling languages (BPMN diagrams [11], Petri nets [27], EPCs) assume a case notion in business processes, which is used to correlate events for a specific process instance. They work well on process-centric systems such as WFM/BPM systems, but have problems to describe object-centric systems which do not assume a case notion. Besides, they focus on the behavioral perspective, although some models have data elements. For instance, in BPMN diagrams and Data-aware Petri nets (DPNs) [13], activities can write/read data elements (variables and objects), which describe the data perspective and the interactions. However, the more powerful notations used in data modeling language (e.g., a UML class model [8]) are rarely employed. Artifact-centric approaches [7, 12, 23] are state-of-the-art to solve these problems. However, they still force one to pick an instance notion for each artifact, although a case notion for the whole process is not required. Moreover, they can indeed model the data perspective but the control-flow cannot be related to an overall data model (i.e., there is no explicit data model or it is separated from the control-flow) and interactions between different entities are not visible (because artifacts are distributed over multiple diagrams) [16].

The proposed Object-Centric Behavioral Constraint (OCBC) modeling language [2, 17] combines ideas from declarative, constraint-based languages like Declare [3], and from data/object modeling techniques (ER, UML, or ORM), resulting in OCBC models as shown...
in Figure 2. This paper extends the OCBC modeling language, allowing the definition of attributes for entities (i.e., classes and activities). The communications perspective is strengthened with the annotation of an interaction type (i.e., add, update, delete or read) and the possibility to describe interactions on the attribute level. Besides, a model editor is available to support OCBC models to design business processes.

The remainder is organized as follows. Section 2 briefly introduces OCBC models based on a running example. Section 3 employs class models to describe the data schema. Activity models are defined in Section 4 to describe the restrictions between activities through declarative constraints. Section 5 illustrates OCBC models to combine the above two models. A plugin for designing OCBC models is introduced in Section 6 and Section 8 concludes the paper.

## 2 RUNNING EXAMPLE

Figure 2 shows an OCBC model example, which describes the business process of the "order" module in the real information system Dolibarr. The model consists of a class model (three classes order, order line and product), and two class relationships rt1 and rt2 at the bottom, an activity model (four activities display order, create order draft, add order line and delete order line), and three constraints c1, c2 and c3 at the top, and six interaction relationships (i.e., aoc1,...,aoc6) in between. More precisely, the class model describes the schema of a database with three related tables, e.g., the class "order" corresponds to an "order" table and rt1 indicates the "order line" table refers to the "order" table. Four activities and the constraints indicate the control-flow of the business process. For instance, c1 means that only after executing the "create order draft" activity, the "display order" activity can be executed (cf. Section 4).

Whenever an activity is executed, an event (an instance of the activity) occurs. It may add (Φ), update (Φ), delete (Φ) or read (Φ) records in databases. The forms of interactions (indicated by the different "circle" symbols) show how events impact databases. For instance, aoc2 means one "create order draft" event adds one record in the "order" table. Note that activities and classes have attributes and interactions can happen on the attribute level by defining a mapping between attributes (cf. Section 5).

Note that, in [2, 17] OCBC models include cardinality constraints on class relationships and interaction relationships. In this paper, we focus on the attribute level of OCBC models, and hide the cardinality constraints for simplicity.

## 3 MODELING THE DATA PERSPECTIVE

Databases serve as the backbone of object-centric information systems by storing all transactions happened in the business processes supported by these systems. Therefore, the structure of databases should be consistent with the corresponding business processes. As mentioned, object-centric information systems store transactions of the same type in the same database table, e.g., all the orders are stored in the "order" table, as shown in Figure 3. In this paper, we use classes to represent tables and class relationships to represent the dependency relationships between tables. Classes have attributes as tables have columns, and class relationships connect attributes of classes like that dependency relationships connect foreign keys and primary keys of tables. By integrating the elements mentioned above, we define a class model to describe the structure of databases, referring to the notations in [26].

**Definition 3.1 (Class Model).** Let \( \mathcal{U}_{OC} \) be the universe of object classes, \( \mathcal{U}_{RT} \) be the universe of classes relationship types, \( \mathcal{U}_{Attr} \) be the universe of attribute names and \( \mathcal{U}_{Val} \) be the universe of attribute values. A class model is a tuple \( \text{ClaM} = (OC, Attr, RT, key, addi, val, rel) \) such that

- \( OC \subseteq \mathcal{U}_{OC} \) is a set of object classes,
- \( Attr \subseteq \mathcal{U}_{Attr} \) is a set of attribute names,
- \( RT \subseteq \mathcal{U}_{RT} \) is a set of class relationship types,
- \( key \in OC \rightarrow \mathcal{P}(Attr) \setminus \{\emptyset\} \) maps each class onto a set of key attribute names, \(^5\)
- \( addi \in OC \rightarrow \mathcal{P}(Attr) \) maps each class onto a set of additional attribute names (\( key(oC) \setminus addi(oC) = \emptyset \) for any class

---

1Dolibarr ERP/CRM is an open source (webpage-based) software package (www.dolibarr.org).

2\( \mathcal{P}(X) \) is the power set of \( X \), i.e., \( Y \in \mathcal{P}(X) \) if \( Y \subseteq X \).
Figure 3: A database with three tables.

Figure 4: A class model with attributes

Figure 5: An object model with attributes.

4 MODELING THE BEHAVIORAL PERSPECTIVE

The behavioral perspective of a business process indicates the constraints between activities. There exist temporal restrictions on activities, e.g., "display order" activity can happen only after "create order draft" activity happens. In this section, we abstract the restrictions as behavioral constraints between activities. For instance, the restriction mentioned above can be described as a constraint between activity "create order draft" and activity "display order".

In object-centric information systems, the behavioral perspective is quite flexible. It should be generic enough to enable users to deal with the flexible and dynamic business environments. Therefore, we choose declarative constraints inspired by Declare [3] to model restrictions. Compared with procedural languages like Petri nets, declarative languages are "open" languages, i.e., they allow for anything which is not explicitly forbidden by constraints, and they are more suitable for modeling flexible business processes. A constraint has a reference activity, a target activity and a corresponding constraint type which specifies the restriction.

**Definition 4.1 (Constraint Types).** $\mathcal{U}_{CT} = \{ X \subseteq \mathbb{N} \times \mathbb{N} \mid X \neq \emptyset \}$ defines the universe of all possible constraint types. Any element of...
An activity model is a collection of activities and constraints. Figure 6 shows an activity model \( \text{ActM} = (A, \text{Attr'}, C, \pi_{\text{ref}}, \pi_{\text{tar}}, \text{type}, \text{addi}', \text{val}') \) with four activities, i.e., \( A = \{ \text{create order draft}, \text{display order}, \text{add order line}, \text{delete order line} \} \), eight attributes, i.e., \( \text{Attr'} = \{ \text{id}, \text{create date}, ..., \text{price} \} \), and three constraints, i.e., \( C = \{ c1, c2, c3 \} \). Consider the constraint \( c1 \) as example. \( \pi_{\text{ref}}(c1) = \text{display order}, \pi_{\text{tar}}(c1) = \text{create order draft}, \text{type}(c1) = \{(\text{before}, \text{after}) \in \mathbb{N} \times \mathbb{N} | \text{before} = 1 \land \text{after} \geq 0 \} \) (unary-precedence).

The reference (target) activity of a constraint defines the corresponding set of reference (target) events. For each reference event, the number of its related target events should meet the requirements of the constraint. In traditional process modeling notations, a constraint is defined for one process instance (case) in isolation. This means that the related target events for a reference event are all target events corresponding to the same case. As discussed before, the case notion is often too rigid. There may be multiple case notions at the same time, causing one-to-many or many-to-many relations. In OCBC models, events are correlated based on the data perspective using the approaches in [2, 15], which is out of the scope of this paper.

Definition 4.3 (Events). Let \( \text{ActM} = (A, \text{Attr'}, C, \pi_{\text{ref}}, \pi_{\text{tar}}, \text{type}, \text{addi}', \text{val}') \) be an activity model and \( M^{\text{ActM}} = \{ \text{map} \in \text{Attr'} \rightarrow \mathcal{U}_{\text{id} | \text{map} : \text{attr} \in \text{val}'(\text{attr})} \} \) be the set of mappings of \( \text{ActM} \). \( E^{\text{ActM}} = \{ (a, \text{map}) \in A \times M^{\text{ActM}} | \text{dom}(\text{map}) = \text{addi}'(a) \} \) is the set of all events of \( \text{ActM} \). Each event is related to an activity and instantiates the attributes of the activity. An event example is \( e = (\text{create order draft}, \text{map}) \) where \( \text{create order draft} \) is an activity, and \( \text{map}(\text{creation date}) = \text{t1}, \text{map}(\text{id}) = \text{01} \) and \( \text{map}(<\text{customer id}>) = c1 \) (i.e., the first event in Figure 8).

5 OBJECT-CENTRIC BEHAVIORAL CONSTRAINT MODELS

Up to now, we have introduced class models, i.e., classes and class relationships, and activity models, i.e., activities and constraints. In this section, we combine them through interaction models to cover the three perspectives in Figure 1 in one single diagram.

Definition 5.1 (Object-Centric Behavioral Constraint Model). An object-centric behavioral constraint model is a tuple \( \text{OCBCM} = (\text{ClAM}, \text{ActM}, \text{InterM}) \), where

- \( \text{ClAM} = (\text{OC}, \text{Attr}, \text{RT}, \text{key}, \text{addi}, \text{val}, \text{rel}) \) is a class model,
- \( \text{ActM} = (A, \text{Attr'}, C, \pi_{\text{ref}}, \pi_{\text{tar}}, \text{type}, \text{addi}', \text{val}') \) is an activity model, and
- \( \text{InterM} = (\text{AOC}, \text{form}, \text{linkAttr}) \) is an interaction model between \( \text{ClAM} \) and \( \text{ActM} \) where

\( \text{AOC} \subseteq A \times \text{OC} \) is a set of relationships between activities and classes.
- form ∈ AOC → OP gives the type of a relationship, i.e., how the activity impacts the class, where OP = {⊙, ⊗, ⊘, ⊙},
- linkAttr ∈ AOC × Attr → Attr maps a pair of a relationship and an attribute from its class onto an attribute from its activity.

For convenience, we say that an event e = (a, mapE) corresponds to an object o = (oc, map) by aoc = (a, oc), denoted as e ∼ oc o if ∀attr ∈ key(oc) : mapE(attr) = mapE(attr′) where attr′ = linkAttr((aoc, attr)).

An OCBC model consists of a class model, an activity model and an interaction model. An example is shown in Figure 2. The formalization of effects of events from the activity model on objects of the class model is given in Definition 5.2. Such interactions (effects) are given in terms of AOC relationships in the interaction model.

Definition 5.2 (Effect of an Event). Let InterM = (AOC, form, linkAttr) be an interaction model. Through an AOC relationship aoc = (a, oc) ∈ AOC, an event e = (a, mapE) changes an object from the old state oold = (oc, mapold) to the new state onew = (oc, mapnew), denoted as oold = onew, if and only if

1. form(aoc) = ⊙, then oold ∈ Onull, onew ∈ OCluM and ∀attr ∈ dom(mapnew) : • mapnew(attr) = mapE(attr′) if attr′ ∈ dom(mapE), and • mapnew(attr) = NULL otherwise.4
2. form(aoc) = ⊗, then e ∼ oold and ∀attr ∈ dom(mapnew) : • mapnew(attr) = oper(mapE(attr′), mapold(attr)) if attr′ ∈ dom(mapE), and • mapnew(attr) = mapold(attr) otherwise.
3. form(aoc) = ⊘, then e ∼ oold and onew ∈ Onull, and
4. form(aoc) = ⊙, then e ∼ oold and onew = oold.

where attr′ = linkAttr((aoc, attr)) and oper is a given function based on domain knowledge (explained next).

Definition 5.2 employs some rules to describe how an event impacts objects through one AOC relationship. More precisely, rule 1 indicates that an “add” event (i.e., the type of the AOC relationship is ⊙) creates a new object. Each attribute of the object is initialized as the same value as its corresponding event attribute (indicated by function linkAttr), or “NULL” if it has no corresponding event attribute. Rule 2 means that an event updates its corresponding object (which is identified by the operator ~, cf. Definition 5.1). More precisely, each attribute of the object is updated by oper, or remains unchanged if it has no corresponding event attribute. Note that oper gives a way to compute the new value of an object attribute (i.e., mapnew(attr)) based on the corresponding event attribute value (i.e., mapE(attr′)) and the old value of the object attribute (i.e., mapold(attr)). This enables complex operations (e.g., summations and subtraction) to update object attributes based on domain knowledge. For instance, we can make oper(a, b) = a if we want that the object value is updated as the event value, or oper(a, b) = b - a if we want that the new object value is the result of subtracting the event value from the old object value. Rule 3 denotes that an event removes its corresponding object while, for rule 4, an event reads its corresponding object.

Figure 8 gives an example to show how events change objects in the object model through AOC relationships in Figure 7. For instance, the first event inserts an “order” object o1 through the AOC relationship aoc2, copying all its attributes to the object. The second event inserts an “order line” object o1 through aoc3 and updates the attribute “quantity” of the “product” object phone through aoc6. Similarly, the third event adds an “order line” object o2 and updates the “product” object cap. The fourth event reads the information of the “order” object o1 without changing anything. The last

The domain of linkAttr dom(linkAttr) = {(a, oc), attr | (a, oc) ∈ AOC ∧ attr ∈ key(oc)}.

4 NULL means that the attribute attr is blank after the creation of its corresponding object.
event removes the "order line" object ol1 through aoc4 and updates the "product" object phone through aoc5. Note that an "add order line" event updates the attribute "quantity" of an "product" object based on oper(a, b) = b - a, i.e., subtracting the value of the event attribute "qty" from the old value of the object attribute "quantity". For instance, at the moment t2, "quantity" of the object phone is updated as "86", since its old value is "88" and the value of the event attribute "qty" is "2". In contrast, a "delete order line" event updates the attribute "quantity" based on oper(a, b) = b + a. For instance, at the moment t5, "quantity" of the object phone is updated as "88", since its old value is "86" and the value of the event attribute "qty" is "2". One event can change multiple objects. In the database context, such an event corresponds to a transaction consisting of several SQL statements. A database system must ensure proper execution of transactions despite failures, i.e., either the entire transaction executes, or none of it does.

6 EVALUATION

We have implemented the OCBC modeling language as a plugin "OCBC Model Editor" in ProM nightly builds (installing the OCBC package) to support designing business processes. Figure 9 shows the interface of the plugin and a designed model in panel 5. By dragging a node from panel 2 to panel 5, one can add an activity or a class. When locating the mouse cursor in the center of a node, one can drag out an edge for the node to connect another node. When a node or an edge is selected, it is highlighted in red and

---

5 Access http://www.win.tue.nl/ocbc/ for more information, such as tools, manuals and OCBC model examples.
its attribute information can be viewed or edited in panel 3. It is possible to zoom in/out models by operating panel 4. After designing a model, one can export the model in panel 1.

The idea of evaluating OCBC models is to compare its ability with other models by describing the same business process. The OTC (order-to-cash) business process is a typical and significant scenario in enterprises. It is supported by ERP systems, e.g., Dolibarr, an open source ERP system for small and medium enterprises. The OTC business process covers a range of modules from creating orders to paying bills. Figure 10 employs an informal notation to describe the business process of “order” module (including the behavioral perspective, data perspective and the communications in between) in Dolibarr.

First, we explain the behavioral (i.e., control-flow) perspective including four activities and three constraints. The edges with single arrows indicate the temporal orders between activities while the cardinalities on edges indicate the restriction on the correspondences (e.g., one-to-many) between activities. For instance, the edge 2 means that after a “create order draft” event, one or more “add order line” events can happen. Indicated by 3, each “add order line” event is followed by at most one “delete order line” event, and each “delete order line” event must have precisely one corresponding “add order line” event before.6

Figure 10: An informal model to describe the “order” module in the OTC business process in Dolibarr.

6 An event a having a corresponding event b before does not mean that a happens directly after b, i.e., it is possible that other events happen between b and a.

Figure 9: The OCBC model editor in ProM.
Events on the behavioral perspective may add, update or delete objects on the data perspective. The process in Figure 10 contains three types of objects, i.e., “order”, “order line” and “product”, and ④ and ⑤ indicate the associations between these types. Dotted lines with arrows are employed to show the interactions between the behavioral perspective and data perspective. For instance, ⑥ denotes that a “create order draft” event creates an “order” object and ⑦ means that a “display order” event displays all contents in an order. ⑧ and ⑨ indicate that an “add order line” event adds an order line to the order and meanwhile decreases the quantity of the corresponding product, respectively. In contrast, ⑩ and ⑪ signify that an “add order line” event removes an order line from the order and meanwhile increases the quantity of the corresponding product, respectively.

Figure 2 employs an OCBC model to describe the process explained above. Note that the process is quite flexible and there exist one-to-many relations (i.e., multiple instances), e.g., one “create order draft” event may correspond to multiple “add order line” events. In the OCBC model, we do not assume a case notion, and use the data perspective to correlate events and distinguish multiple instances. Based on this, declarative constraints are used to describe the process more precisely. The interaction types indicate how events on the behavioral perspective impact (i.e., add, update, delete or read) the objects on the data perspective. Besides, Figure 4, Figure 6 and Figure 7 show the attribute level of the process from different perspectives. By doing this, the interactions are presented more clearly. For instance, with the support of interaction types and the attribute level, one can know that an “add order line” event updates the “quantity” value of a “product” object.

Next, we use other modeling techniques to describe the process. Figure 11 presents the process with a DPN diagram. Due to the one-to-many relation (② in Figure 10), “add order line” and “delete order line” events can happen multiple times in the process. Accordingly, the DPN diagram employs several silent transitions (and loops) to allow the occurrence of multiple instances, resulting in an underfitting model (allowing to much behavior). For instance, the constraint (③ in Figure 10) indicating that a “delete order line” event can happen only after its corresponding “add order line” event is not contained in the model. DPN diagrams support modeling the data perspective with variables. For instance, “quantity” is a variable which indicates that “add order line” and “delete order line” events modify the quantity of corresponding product.

Figure 11: A DPN example to describe the “order” module.

Figure 12: A BPMN example to describe the “order” module.

In summary, in the flexible scenario with multiple instances, DPN and BPMN notations suffer an underfitting behavioral perspective. Furthermore, the data perspective and interactions between events and data in both models are not as powerful as those in the OCBC model. In contrast, OCBC models can explicitly represent one-to-many and many-to-many relations, as well as interactions between different process instances in a single, hybrid diagram.

7 RELATED WORK
Process modeling languages play an important role in the design and analysis of business processes. Over time researchers have proposed various languages and models to describe business processes [10, 21, 24].

Most models mainly focus on one aspect of business processes. For instance, Petri nets [27] and Declare models [3] are both focusing on the behavioral perspective to describe the workflows. In contrast, Petri nets are procedural and strict while Declare models are declarative and flexible. The BPMN notation has data symbols to support modeling the data perspective of business processes, e.g., data object, data store, message and data association. In Figure 12, three data objects are created to show the interactions between the behavioral perspective and data perspective. However, the objects are considered in isolation (there are no relations between objects) and the interactions on the attribute level are not shown, e.g., “add order line” events modify the “quantity” attribute of the product object.

In summary, in the flexible scenario with multiple instances, DPN and BPMN notations suffer an underfitting behavioral perspective. Furthermore, the data perspective and interactions between events and data in both models are not as powerful as those in the OCBC model. In contrast, OCBC models can explicitly represent one-to-many and many-to-many relations, as well as interactions between different process instances in a single, hybrid diagram.
and its communication with data perspective by data objects (e.g.,
documents) and data stores (e.g., tables). [25] provides a concrete
model, called RAW-SYS, to model the control-flow, the data
and their interaction, and to verify data-aware processes. Artifact-
centric approaches [7, 12, 23] (including the earlier work on proclots
[1]) aim to capture business processes in terms of so-called business
artifacts, i.e., key entities which drive a company’s operations and
whose lifecycles and interactions define an overall business process.
Artifacts have data and lifecycles attached to them, thus relating
both perspectives.

These process-centric approaches (e.g., DPN and BPMN dia-
grams) can describe the data perspective to some extent. They try
to consider the data perspective by adding data elements, such as
attributes, values, variables and objects, onto the control-flow. How-
ever they do not support explicit data modeling as can be found in
ER models. Artifact-centric approaches need to identify artifacts
beforehand based on domain knowledge, and within an artifact
(proclot, or subprocess), one is forced to pick a single instance
notation.

Besides, the description of the end-to-end behavior needs to
be distributed over multiple diagrams (e.g., one process model per
artifact).

Compared with existing approaches, an OCBC model can de-
scribe the data perspective, the behavioral perspective and the
interplay in between in one single diagram, which provides a pic-
ture of the whole system. Besides, it is powerful on presenting
the interactions through mapping class attributes onto activity
attributes.

8 CONCLUSION AND FUTURE WORK

An OCBC model is a hybrid model consisting of: a class model
describing the structure of databases; an activity model describing
the possible events and restrictions between events; and an inter-
action model describing how events modify the database tables.
This paper strengthens the proposed OCBC models by (i) adding
attribute onto classes and activities, (ii) adding interaction types on
AOC relationships and (iii) formalizing the interactions (i.e., how
events change objects) on the attribute level.

OCBC models outperform existing approaches when describing
flexible business processes on object-centric information systems,
since it is more powerful to model communications and shows
three perspectives of a business process in one single diagram.
Besides, we implemented the language as an editor, which supports
designing models to describe business processes.

Moreover, the OCBC models serve as a starting point for a new
line of research. Based on event logs extracted from data generated
by object-centric information systems [18], automatically discover-
ing OCBC models [17] should be further developed to incorporate
the discovery of attributes and the interactions related. The con-
formance checking techniques [2] will be able to detect different
kinds of deviations taking the attribute level into consideration.
Finally, the behavioral perspective can be extended to incorporate
data-aware constraints (which may be comparable to the guards in
DPN diagrams).

REFERENCES

A Framework for Lightweight Interacting Workflow Processes. International
Balancing Between Flexibility and Support. Computer Science - Research and
Development 23, 2 (2009), 149–177.
Products and Systems Development. Journal of Object Technology 6, 6 (2007),
149–177.
India.
ACM.
Modeling Business Operations and Processes. IEEE Data Engineering Bulletin 32,
Systems Modeling Language. Morgan Kaufmann.
Managing Artifact Interactions with Conditions and Events. In International
Conference on Distributed Event-Based Systems (DEBS 2011). ACM.
Use. Springer-Verlag, Berlin.
Correlation for Process Discovery from Object-Centric Event Data. In 2018 IEEE
https://doi.org/10.1109/ICWS.2018.00033
80–84.
Object-Centric Behavioral Constraint Models. In Business Information Systems:
45–58.
[18] G. Li, E. González López de Murrillas, R.M. de Carvalho, and W.M.P. van der
Aalst. 2018. Extracting Object-Centric Event Logs to Support Process Mining on
Databases. In Information Systems in the Big Data Era, CASE Forum 2018, Jan
Mendling and Haralambos Mouratidis (Eds.). Springer International Publishing,
Cham, 182–199.
ACM symposium on applied computing. ACM, 1454–1461.
tual workflow nets with data: Early detection of errors with the most precision
Information Technology, Griffith University (2000).
Add Data into Business Process Verification: Bridging the Gap between
[26] W.M.P. van der Aalst. 2015. Extracting Event Data from Databases to Unleash