Information and Software Technology 52 (2010) 1188–1203

Contents lists available at ScienceDirect

Information and Software Technology

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

A family of experiments to validate measures for UML activity diagrams of ETL processes in data warehouses

Lilia Muñoz [a,*], Jose-Norberto Mazón [b], Juan Trujillo [b]

[a] Lucentia Research Group, Department of Information Systems, Control, Evaluation and Computing Resources, Technological University of Panama, P.O. Box #0819-07289, Republic of Panama

[b] Lucentia Research Group, Department of Software and Computing Systems, University of Alicante, San Vicente del Raspeig, 03080, Spain

ARTICLE INFO

Article history:
Received 24 January 2010
Received in revised form 24 May 2010
Accepted 9 June 2010
Available online 15 June 2010

Keywords:
ETL processes
Measure validation
Activity diagrams
Empirical software engineering
Data warehouse conceptual modeling
Quality

ABSTRACT

In data warehousing, *Extract, Transform, and Load* (ETL) processes are in charge of extracting the data from the data sources that will be contained in the data warehouse. Their design and maintenance is thus a cornerstone in any data warehouse development project. Due to their relevance, the quality of these processes should be formally assessed early in the development in order to avoid populating the data warehouse with incorrect data. To this end, this paper presents a set of measures with which to evaluate the structural complexity of ETL process models at the conceptual level. This study is, moreover, accompanied by the application of formal frameworks and a family of experiments whose aim is to theoretical and empirically validate the proposed measures, respectively. Our experiments show that the use of these measures can aid designers to predict the effort associated with the maintenance tasks of ETL processes and to make ETL process models more usable. Our work is based on *Unified Modeling Language* (UML) activity diagrams for modeling ETL processes, and on the *Framework for the Modeling and Evaluation of Software Processes* (FMESP) framework for the definition and validation of the measures.

1. Introduction

In the 1990s, Inmon [15] coined the term data warehouse (DW) as being an integrated collection of subject-oriented data in the support of decision making. The components of a DW are usually depicted as a multi-layer architecture in which data from one layer are derived from data of the previous layer as shown in Fig. 1. Importantly, the integration of data sources is achieved through the use of ETL (*Extract, Transform, and Load*) processes (see *Integration Layer* in Fig. 1). Such processes are responsible for the extraction of the data from a variety of heterogeneous data sources, for the transformation of these data (conversion, cleaning, etc.) and their loading into the DW. It is therefore extensively recognized that the appropriate design and maintenance of the ETL processes are key factors in the success of DW projects [21,41,42].

However, in practice, designing and maintaining ETL processes may be extremely complex, prone to failure, and time consuming [39]. Indeed, it has been argued that ETL processes are costly and one of the most important parts of a DW development [15,45]. In [38], it is reported that the cost of ETL and data cleaning tools is estimated to be at least a third of the efforts and budget expenses of a DW.

Bearing these consideration in mind, a first step towards ameliorating the design and maintenance of ETL processes is the adoption of approaches for both the conceptual modeling and logical design of ETL processes for DWs [24,40,44–46,48]. Within these approaches, several design guides have been proposed to address the complexity of ETL processes. However, most of the time, guides are not enough, since they can help designers in their work, but they imply rather subjective decisions [33], and this can lead to difficult-to-maintain ETL processes, which could pose serious DW loading performance problems. In order to overcome this inherent subjectivity of quality, mechanisms are required with which to objectively measure the quality of ETL process models. Furthermore, the quality of ETL processes should be considered early on the development in order to anticipate design and maintenance decisions.

Despite these risks, the vast majority of the works related to DW quality focus on product quality, and ignore the quality assessment of ETL processes. In the most prominent literature related to this topic, very few approaches have been found [1,36,47,48] in which quality measures for ETL processes are presented. Moreover, the proposed measures of those approaches have not been validated in either a formal or an empirical manner.

In order to overcome this scenario, this paper describes and validate a set of measures with which to evaluate the usability and ease of maintenance of the conceptual models of ETL processes, starting from the hypothesis that low usability and low ease of

* Corresponding author. Tel.: +507 775 4563; fax: +507 774 3012.
E-mail addresses: lilia.munoz@utp.ac.pa (L. Muñoz), jnmazon@dlsi.ua.es (J.-N. Mazón), jtrujillo@dlsi.ua.es (J. Trujillo).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.06.003

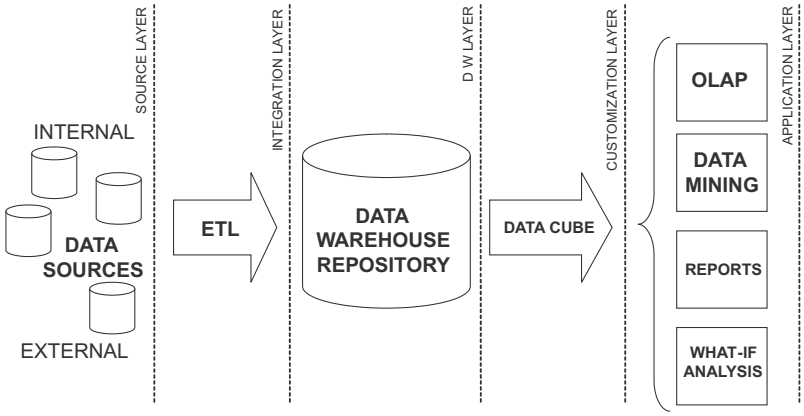

Fig. 1. Multi-layer architecture of data warehouses.

maintenance of ETL conceptual models influence their global quality, and may, therefore, have a critical impact on the development of the DWs (higher costs in both resources and time). The measures were defined by using the FMESP framework (*Framework for the Modeling and Evaluation of Software Processes*) [13]. FMESP is a modeling framework for measuring the software process, in which models are based on *Software Process Engineering Metamodel Specification* (SPEM) [26]. However, owing to its generality and flexibility we have adapted it to the evaluation of ETL process models at the conceptual level. It is worth noting that the defined measures can be applied to any ETL modeling framework, but in order to better exemplify our work we have used our approach for the conceptual modeling of ETL processes presented in [24], which is based on *Unified Modeling Language* (UML) activity diagrams. UML activity diagrams have been proved to provide a suitable representation of ETL processes, e.g. permitting us to represent their dynamic aspects. Moreover, the proposal presented in [24] is framed within a global proposal to undertake the development of the DW with *Model Drive Architecture* (MDA) [27], in which different layers of the DW architecture are considered [22,23,29] in order to develop the DW in a systematic and automated manner.

The remainder of this paper is structured as follows: Section 2 outlines related work. In Section 3, the proposed measures are defined for the conceptual level of ETL process models. A theoretical validation is presented in Section 4. Section 5 explains in detail our empirical validation of the proposed measures. Section 6 presents a global analysis of the results. Finally, conclusions and future works are outlined in Section 7.

2. Related work

DW quality may be influenced by database management system quality, data quality and data model quality (which can be considered at different levels: conceptual, logical and physical) [35]. This section is focused on giving a general overview of the last two aspects.

2.1. Quality measures for data models

With regard to the measures for data models, several proposals appear in literature [4,28,32,34,35]. These are good approaches towards the measurement of DWs; nevertheless, they are incomplete, since they are not part of a quality model that allows designers to use them in an objective and systematic manner. One exception to this is the work presented in [17], in which the authors describe the *Data Warehouse Quality* (DWQ) model. DWQ attempts to assure the quality of the stored data in order to improve the use of the DW by evaluating certain quality dimensions, in particular, the data quality. Nevertheless, the DW architecture is composed of several layers [23] and we consider that the quality of DWs should cover all of them. Unfortunately, none of these proposals evaluates the quality of ETL processes.

2.2. Quality measure of ETL processes

Some works concerning the quality measurement of ETL processes exist. In [47] an ETL setting is modeled by means of a diagrammatic notation and relevant measures over the diagram nodes are introduced. In [48] the same authors define a collection of measures which evaluate the workflow of the ETL processes. On the other hand, regarding data integration for ETL processes, the measures can be placed in the following categories [36]: data types, conformity of data domain, statistical characteristics of data set (maximum value, minimum value, etc.) and reference relations. In [1], a proposal to verify the consistency of the data that are loaded in the DW is presented. The *Shannon entropy* [37] is calculated in partitions that are produced by a set of attributes. This shows that these values can be used as problem indicators in the data extraction process. Nevertheless, a formal or empirical validation for the proposed measures has not been carried out in any of these approaches. The aim of this paper is to fill in these gaps. We therefore present and validate a set of measures with which to assess the ease of maintenance of ETL process conceptual models which, as is argued, has a direct impact on the entire DW quality.

3. Measures for the conceptual modeling of ETL processes

A first step towards obtaining quality ETL processes is the definition of development approaches in which the importance of defining conceptual models is highlighted [44,45]. However, as previously presented, most of the current proposals for designing ETL processes still delegate the quality of conceptual models in the experience of the designer which may not be sufficient to guarantee the quality of ETL processes since human decisions are rather subjective.

As conceptual models of ETL processes are software artifacts, their structural properties (such as structural complexity) have an impact on its cognitive complexity as stated in [7]. Cognitive complexity means the mental burden on those who have to deal with the artifact (e.g. developers, testers, maintainers). High cognitive complexity of an artifact reduces several desirable properties (such as understandability, or modifiability) and negatively affects external quality attributes (such as its usability and maintainability) as is defined in the ISO9126 standard [16].

Therefore, potential relationships that may exist between the structural properties of conceptual models of ETL processes and their quality factors must be investigated. To this aim, we have develop a set of objective measures for quality assessment by means of measuring the structural properties of conceptual models for ETL processes. If we take into account that any software process model (such as a conceptual model of an ETL process) with a high degree of structural complexity is less usable and much more difficult to maintain, hence these measures can be considered a good usability and maintainability indicator. This hypothesis transposes the relationship between structural complexity and both usability and maintainability of software artifacts [7] to the domain of software processes, as depicted in Fig. 2. As we can observe in this figure, our measures evaluate different structural properties of a conceptual model of ETL processes, namely size, complexity, and coupling, according to the theoretical validation results obtained by applying the Briand et al. framework [5] (see Section 4). Accordingly, these structural properties affect usability and maintainability of conceptual models of ETL processes. It is worth noting that usability is defined in the ISO 9126 standard as an attribute that evaluates the necessary effort that must be made in order to use a model, while maintainability refers to attributes that measure the required effort to carry out modifications in a model. In this work, we have specifically focused on studying if the defined measures for conceptual models of ETL processes affect one relevant external quality attribute of the usability: understandability (i.e. easiness with which the model can be understood); and one relevant external quality attribute of the maintainability: modifiability (i.e. easiness with which the model can be modified due to possible errors, specific modification requests or new requirements).

Our approach for defining measures for ETL processes adapts the FMESP [13] for measurement of software process model, since it helps us to determine a set of measures which would be useful in the maintenance of software process models through the evaluation of their structural complexity. Then, the proposed measures can be empirically validated in this sense of [5] where it is stated that a measure is valid if it is proven that it gauges what is proposed to measure and there is an accumulation of convincing evidence that shows its utility.

3.1. UML activity diagrams for conceptual modeling of ETL processes

Measures for conceptual models of ETL processes have been defined within the modeling framework presented in our previous work [24]. This framework is based on *Unified Modeling Language* (UML) activity diagrams [25]. UML activity diagrams have been proven to provide a suitable representation of the ETL processes at the conceptual level, since activity modeling emphasizes the sequence and conditions for coordinating lower-level behaviors (by the commonly called control and object flows), thus permitting to represent at the conceptual level the dynamic aspects and data flow inherent to ETL processes. In turn, these elements allow to represent dynamic aspects and behavior, to arrange the control flow and incorporate restrictions of temporality (e.g., time that a process takes to be executed).

Our development framework establishes the whole activities presented in [20,44], since they are representative operators for the conceptual modeling of ETL processes. Next, we present them together with a brief description:

- *Aggregation*: It aggregates data based on some criteria.
- *Conversion*: It changes data type and format or derives new data.
- *Filter*: It filters and verifies data.
- *Incorrect*: It redirects incorrect data.
- *Join*: It joins two data sources related to each other with some attributes.
- *Loader*: It loads data into the target of an ETL process.
- *Log*: It logs activity of an ETL mechanism.
- *Merge*: It integrates two or more data sources with compatible attributes.
- *Surrogate*: It generates unique surrogate keys.

In order to represent each of these operators by using UML activity diagrams, a customizable and instantiable template based on metaclasses of the UML activity diagrams has been developed (see Fig. 3). An ETL process is modeled though a set of these templates as they model activities through a set of reusable and parameterized elements to emphasize the flow of an ETL process. Each of these templates is defined on the basis of a set of

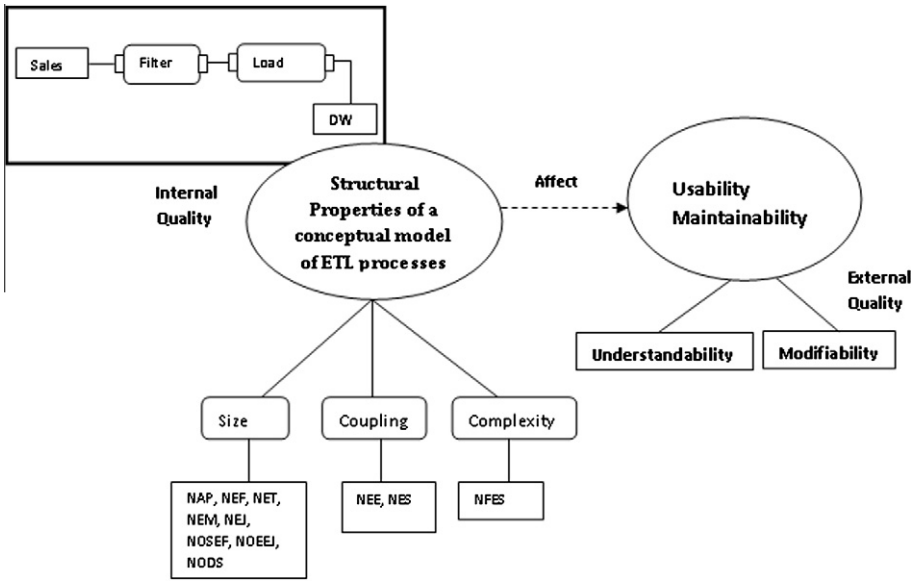

Fig. 2. Relationships between structural complexity and both, usability and maintainability.

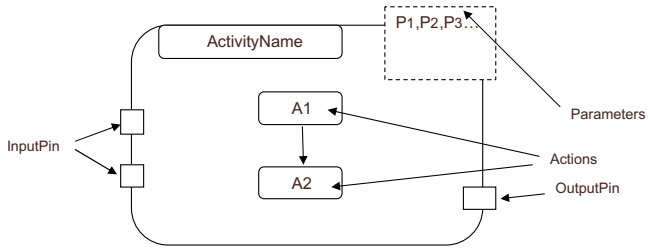

Fig. 3. Template for specifying ETL processes at the conceptual level by using UML activity diagrams.

metaclasses *ParameterableElement*, which are part of the UML activity diagram metamodel. Parameters (*P1*,*P2*,*P3*,...), are used to specify certain indicators of the *ParameterableElement* which may vary from one to another instance according to the required features and attributes. Moreover, in a template we can represent other metaclasses of the UML activity diagram that are useful for modeling ETL processes:

- *Action*: In a UML activity diagram, an action represents a single step within an activity, i.e. it cannot further decomposed within the activity. Therefore, an activity that represents a behavior within an ETL process (e.g. aggregating sales) is composed of individual elements that are actions of an ETL process (e.g. sum the quantity of sold items).
- *InputPin*: In a UML activity diagram, input pins are elements of an action that receive values from other actions through flows. Therefore, they represent input flows in an ETL process (e.g. sales data coming from a table).
- *OutputPin*: Output pins are elements that deliver values to other actions through flows. Therefore, they represent output flows in an ETL process (e.g. summed sales data delivered into a table).
- *Fork*: Forks are elements that splits an input flow into multiple concurrent output flows.
- *Join*: Joins are elements that synchronize multiple input flows in a unique output flow.
- *DataStore*: They are buffers for non-transient information (e.g. a table).

3.2. Measurement of conceptual models for ETL processes

The measures of UML activity diagrams for ETL processes at the conceptual level are defined with the aim of evaluating usability and ease of maintenance of an ETL process model as a whole. The chosen measurement method is, therefore, one which counts the most significant elements of an ETL process model (activities, actions, etc.) along with the significant relationships among the elements (input or output flows). The definition of these measures has been achieved by using the FMESP framework, and it has been based on the metaclasses underlying the UML activity diagrams. It is worth mentioning that some of the measures proposed by FMESP are directly applicable to the conceptual modeling of ETL processes. Nevertheless, due to the expressiveness of UML activity diagrams some important modeling elements are not contemplated by FMESP. It is thus necessary to define new measures at a finer abstraction level, i.e. ETL activities. The definition of these measures is shown in Table 1. As the terms used in this work regarding the measurement of ETL process models are based on the ontology for software measurement defined by Garcia et al. [12], the set of measures identified has been thus grouped into base measures (i.e. an attribute which is functionally independent of other measures and the method for quantifying it) and derived measures (i.e. a measure that is calculated from another base or derived measure):

Table 1
Defined measures for UML activity diagrams of ETL processes.

Measure	Definition
NAP	Number of activities in the ETL process model
NEE	Number of inputs elements in ETL process model
NES	Number of outputs elements in ETL process model
NFES	Number of input and output flows in the ETL process model
NET	Number of time elements in the ETL process model
NEM	Number of merge elements in the ETL process model
NEF	Number of fork elements in the ETL process model
NEJ	Number of join elements in the ETL process model
NOSEF	Number of output objects in a fork element in the ETL process model
NOEEJ	Number of input objects in a join element in the ETL process model
NODS	Number of data store elements in the ETL process model
NTEES	Total number of input and output elements in the ETL process model NTEES = NEE + NES
RFESA	Ratio of input and output flows for each activity in the ETL process model $RFESA = \frac{NFES}{NAP}$
RDInEE	Ratio of dependencies of input elements in the ETL process model $\text{RDInEE} = \frac{NEE}{NTEES}$
RDOutES	Ratio of dependencies of output elements in the ETL process model $RDOutES = \frac{NES}{NTEES}$

- *Base measures*: A total of 11 measures have been defined: NAP, NEE, NES, NFES, NET, NEM, NEF, NEJ, NOSEF, NOEEJ, and NODS. These measures are related to the most significant elements that appear in an ETL process model based on UML activity diagrams as described in the previous section.
- *Derived measures*: Four measures have been defined from the previously defined base measures: NTEES, RFESA, RDInEE, and RDOutES.

In order to illustrate the application of the defined measures, an example of an ETL process model based on UML activity diagrams is presented in Fig. 4. This is composed of a set of ETL related activities in order to load data from online sales of tickets (*Sales* data source) in the DW. *Sales* data source contains the following attributes *IdTicket, IdProduct, Name, Description, Price, Quantity, Discount, and Date*. As the DW stores aggregated daily sales in a fact table named *SalesFact*, the ETL process needs to group sales from the data source by *IdProduct* and *Date*. To this aim, the *Summedsales* activity is used. The flow of this activity is in this way: once the key attributes are verified in the *Verify* action, sales are summed daily by using a *Total* operation: `Total = SUM(Quantity * Price)`. Subsequently, the function *GroupBy* is applied to have the sales grouped by *IdProduct* and *Date*. This information is stored in a table *Table Temp*. The information of the table *Table Temp* is used by the activity *SurrogateSales* to generate an alternate key to establish a relation between the fact table *SalesFact* and the dimension *Time*. A *Log* activity notifies if the key has been correctly generated. Besides, the summarized sales are loaded in the fact table *SalesFact*. This operation is carried out with the activity *SalesLoader*, for the sake of understandability, this activity is not exploded in the example. On the other hand, it is needed to load the list of products into the dimension *ProductsDim*. For this, we use the *ProductLoader* activity that verifies the attributes are extracted from the *Sales* table, and temporarily stored in a table *Table Temp* in which they are updated and subsequently loaded in the *ProductDim* dimension. A temporal condition has been added to this activity, which indicates that the load should be carried out daily.

Table 2 shows the values of the defined measures for the example illustrated in Fig. 4.

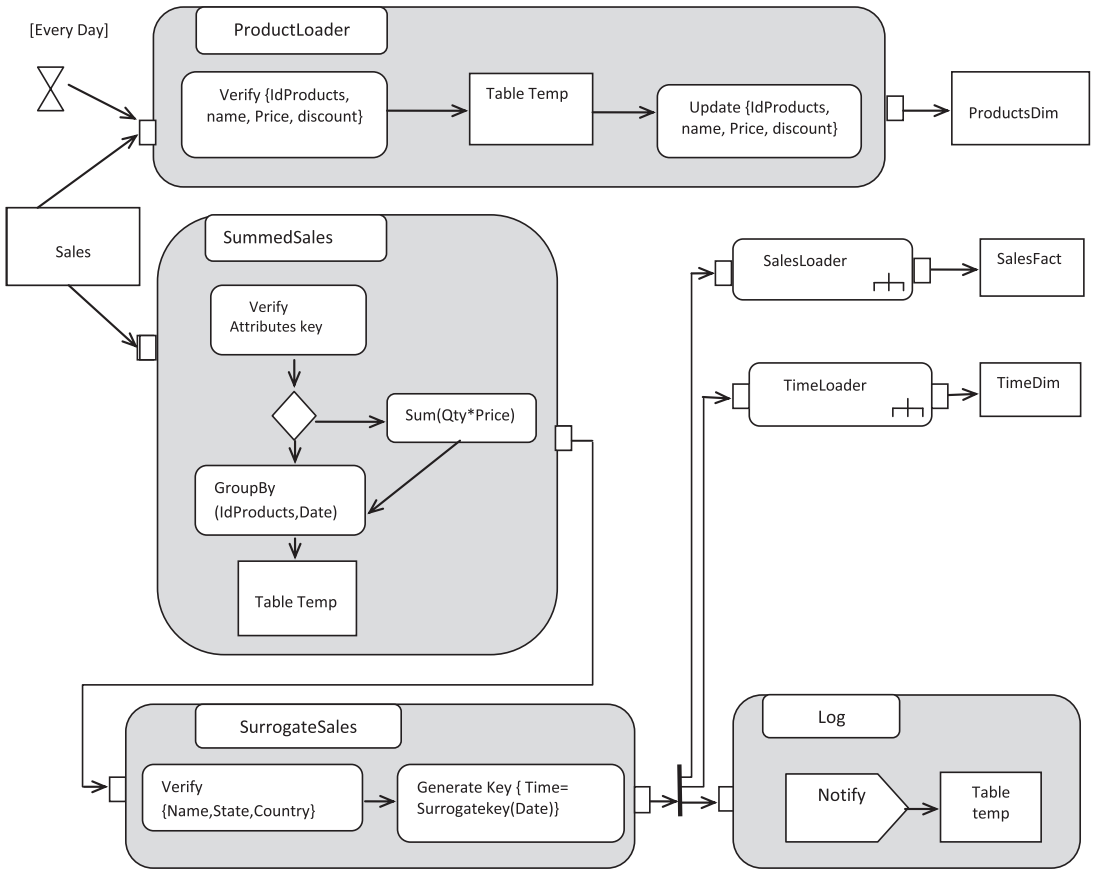

Fig. 4. Example ETL process represented with a UML Activity Diagram.

Table 2
Values of measures for the sample UML activity model of an ETL process.

Measure	Value
NAP	6
NEE	1
NES	3
NFES	11
NET	1
NEM	0
NEF	1
NEJ	0
NOSEF	3
NOEEJ	0
NODS	4
NTEES	4 (1 + 3)
RFESA	1.833 (11/6)
RDInEE	0.25 (1/4)
RDOutES	0.75 (3/4)

4. Theoretical validation

Defined measures must be theoretically validated in order to know when and how they have to be applied. There are two main tendencies in measures formal validation: axiomatic frameworks [6,49] and frameworks based on measurement theory [30,31,50,52]. The goal of the formers is merely definitional, as on these kind of frameworks, a set of formal properties is defined for given software attributes and it is possible to use this set of properties for classifying the proposed measures. On the other hand, in the frameworks based on measurement theory, the information obtained is the scale to which a measure pertains and, based on this information, we can know which statistics and which transformations can be applied to the measure.

In this paper, we have validated our measures by using two different frameworks: (i) the Briand et al. [6] axiomatic framework, and (ii) the DISTANCE framework by Poels and Dedene [30,31] which is based on theory of measurement.

4.1. Formal framework of Briand et al

In order to validate measures, Briand et al. [6] emphasize the need to define the most prominent concepts utilized in the measurement of software products in a non-ambiguous manner. Briand et al. propose a generic framework to rigorously define exactly which mathematical properties characterize these concepts. This framework considers that software products are composed of systems and modules. A *system* S will be represented as a pair $\langle E,R\rangle$, where E represents the set of elements of S, and R is a binary relation on E ($R \subseteq E \times E$) representing the relationships between S's elements. Given a system $S = \langle E,R\rangle$, a system $m = \langle E_m,R_m\rangle$ is a *module* of S if and only if $E_m \subseteq E$, $R_m \subseteq E \times E$, and $R_m \subseteq R$. As an example, E can be defined as the set of code statements and R as the set of control flows from one statement to another. A module m may be a code segment or a subprogram.

It is worth noting that the elements of a module are connected to the elements of the rest of the system by incoming and outgoing relationships. The set $InputR(m)$ are those relationships from elements outside of module m ($m = \langle E_m,R_m\rangle$) to those elements inside of module m. This set is defined as follows:

$$InputR(m) = \{\langle e_1, e_2\rangle \in R | e_2 \in E_m \wedge e_1 \in E - E_m\}$$

On the other hand, the set $OutputR(m)$ are those relationships from elements inside of module m ($m = \langle E_m, R_m \rangle$) to those elements outside of module m. This set is defined as follows:

$$OutputR(m) = \{\langle e_1, e_2 \rangle \in R | e_1 \in E_m \wedge e_2 \in E - E_m\}$$

In this formal framework the measures are classified by using several important measurement concepts whir are related to systems (size, complexity, length) and modules (cohesion and coupling):

- *Size*. In order for a measure to be considered as a size, it must fulfill three properties: it cannot be negative (*non-negativity*), it is expected to be null when a system does not contain any elements (*null value*), and it is expected to be additive when several modules do not have elements in common (*module additivity*).
- *Length*. A measure is considered as a length if: it is not negative (*non-negativity*), it is equal to 0 when there are no elements in the system (*null value*), it is not greater than the length of the original system when a new relationship is introduced between two elements belonging to the same connected component of the graph representing the system (*non-increasing monotonicity for connected components*), it is not smaller than the length of the original system when a new relationship is introduced between two elements belonging to two different connected components[1] (*non-decreasing monotonicity for non-connected components*), and it is not additive for disjoint modules[2] (*disjoint modules*).
- *Complexity*. A measure related to complexity is expected to be non-negative (*non-negativity*) and null when there are no relationships between the elements of a system (*null value*). It should not be sensitive to representation conventions with respect to the direction of arcs representing system relationships (*symmetry*). A relation can be represented in either an "active" (R) or "passive" (R^{-1}) form. The system and the relationships between its elements are not affected by these two equivalent representation conventions, so a complexity measure should be insensitive to this. Also, the complexity of a system S should be at least as much as the sum of the complexities of any collections of its modules, such that there are not modules that share relationships, but they may only share elements (*module monotonicity*). As a consequence of the above properties, system complexity should not decrease when the set of system relationships is increased. Finally, the complexity of a system made of disjoint modules is the sum of the complexities of the single modules (*disjoint module additivity*).
- *Cohesion*. The concept of cohesion has been used with reference to modules or modular systems. It assesses the tightness with which "related" features are "grouped together" in systems or modules. Intuitively, we expect cohesion to be non-negative and, more importantly, to be normalized (*non-negativity and normalization*) so that the measure is independent of the size of the modular system or module. Moreover, if there are no internal relationships in a module or in all the modules in a system, cohesion is expected to be null (*null value*) for that module or for the system, since there is no relationship between the elements and there is no evidence they should be encapsulated together. Additional internal relationships in modules cannot make cohesion decreased, since they are supposed to be additional evidence to encapsulate system elements together (*monotonicity*). When two (or more) modules showing no relationships between them are merged, cohesion cannot be increased because seemingly unrelated elements are encapsulated together (*cohesive modules*).
- *Coupling*. Intuitively, coupling captures the amount of relationship between the elements belonging to different modules of a system. Given a module m, two kinds of coupling can be defined: inbound coupling and outbound coupling. The former captures the amount of relationships from elements outside m to elements inside m; the latter the amount of relationships from elements inside m to elements outside m. Coupling is expected to be non-negative (*non-negativity*), and null when there are no relationships among modules (*null value*). When additional relationships are created across modules, coupling is not expected to decrease since these modules become more interdependent (*monotonicity*). Coupling can only decrease in merging modules since there may exist relationships among them and therefore, inter-module relationships may have disappeared (*merging of modules and disjoint module additivity*).

In order to apply this framework to validate our measures defined for ETL processes, its concepts must be aligned to concepts from UML activity diagrams, in such a way a system is composed of activities and their dependency relationships are input or output flows. A module consists of a subset of activities.

The process of validating our measures by applying the Briand et al. framework is illustrated with the **NFES** measure (number of input and output flows in the ETL process model). The same process has been followed for validating the other measures. Table 3 presents a summary of the validation of the proposed measures for ETL process models: NAP, NEF, NET, NEM, NEJ, NOSEF, NOEEJ and NODS are size measures, NEE and NES are coupling measures and NFES is a complexity measure. Moreover, we can state that the derived measures NTEES, NFESA, RDInEE, RDOutES are valid measures upon being defined from a function of calculation over valid base measures.

Validation of the **NFES** *measure* **NFES** measure (number of input and output flows in the ETL process model) is classified as a complexity measure since it fulfills the following properties:

Table 3
Measures defined for the ETL process model and properties of the Briand et al. framework [6].

Properties	NAP, NEF, NET, NEM, NEJ, NOSEF, NOEEJ, NODS	NEE, NES	NFES
Non-negativity	×	×	×
Null value	×	×	×
Module additivity	×		
Non growing monotonicity for the non plugged components			
Disjoint modules			
Additivity of disjoint modules		×	×
Monotonicity		×	
Merge of modules		×	
Monotonicity of module			×
Symmetry			×
Cohesive modules			
Non negativity and normalization			
	Size	**Coupling**	**Complexity**

[1] This stems from the fact that the new relationship creates a new connected component, where the maximum distance between two elements cannot be less than the maximum distance between any two elements of either original connected component.

[2] Actually, the length of a system containing several disjoint modules is the maximum length among them.

- *Not negativity*. The number of input and output flows of the ETL process model is always greater than or equal to zero, so NFES can never be negative.
- *Null value*. If there are no input or output flows in the activities of the ETL process model then NFES is 0.
- *Symmetry*. The number of input or output in activities does not depend on the convention used to represent these dependencies.
- *Monotonicity of modules*. According to the definition of this property, if m1 and m2 are any two modules of the system that do not have relationships (input and output flow) in common; then, it fulfills that the complexity of the system is not less than the sum of complexity of modules m1 and m2, in other words NFES(S) = NFES(m1) + NFES(m2). An example is shown in Fig. 5 which represents a system consisting of modules m1 and m2 with value of NFES 6 and 4 respectively, and the value of NFES(S) is the sum of NFES(m1) and NFES(m2), i.e. 10.
- *Additivity of disjoint modules*. The number of dependencies among input and output flow and activities in a module obtained by grouping two split modules (without precedence relationships or common elements) is equal to the sum of the number of units of precedence of the two modules separately. The fulfillment of this property is illustrated in Fig. 6, where two split modules m1 and m2 has NFES(m1) = 6 and NFES(m2) = 4. By joining these modules a new *S* module is generated where the NFES value is equal to 10; i.e. the sum of NFES(m1) and NFES(m2).

4.2. DISTANCE framework

The DISTANCE framework [30,31] provides constructive procedures with which to model the interesting attributes of the software and to define the corresponding measures. Software attributes are modeled as distances between the software entities and other entities used as measurement standards or benchmarks. These distances are measured by using mathematical functions. Working with distances is useful because the concepts of similarity and dissimilarity are quite well understood, easily imaginable and they are used in everyday life [43].

The different steps of the DISTANCE framework for constructing measures are as follows (Table 4 summarizes the steps, required inputs and expected outputs of this framework):

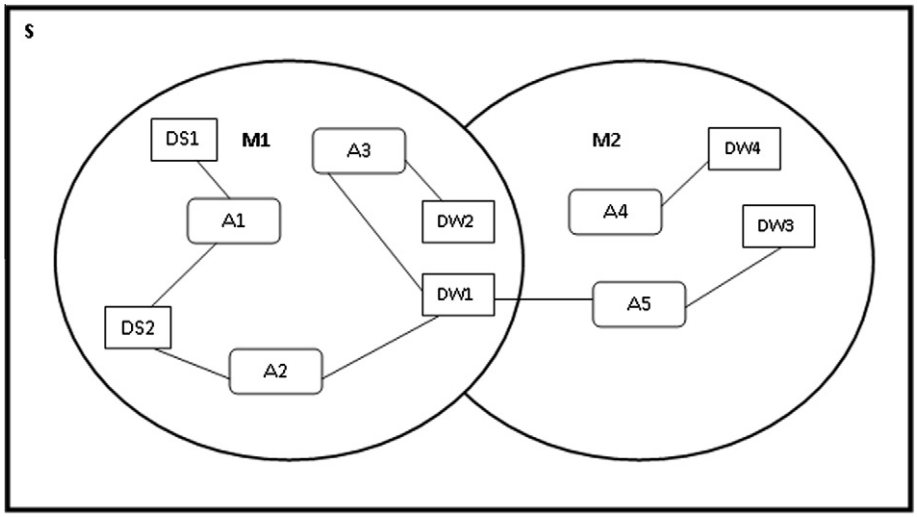

Fig. 5. Sample demonstration of the "monotonicity of modules" property for the measure NFES.

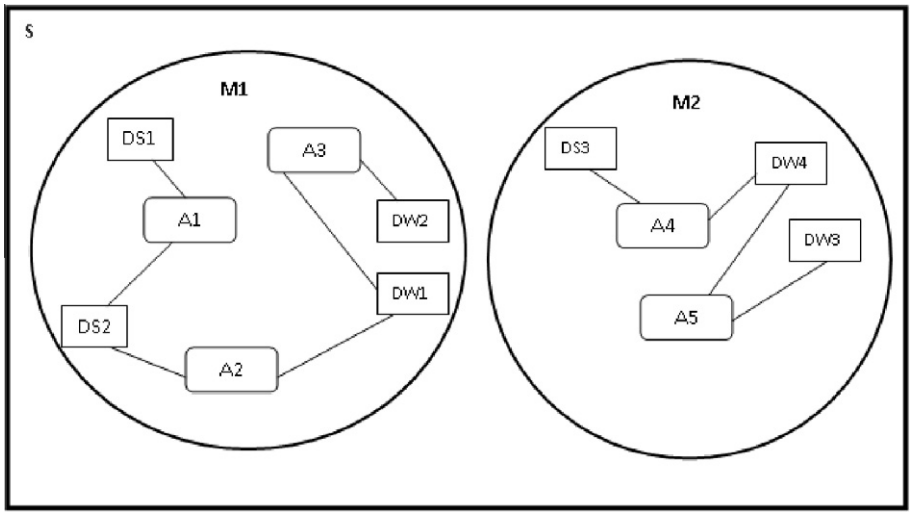

Fig. 6. Sample demonstration of the "additivity of disjoint modules" property for the measure NFES.

Table 4
Steps, required inputs and expected outputs for the DISTANCE framework.

Step	Input	Output
To find an abstraction for the measure	The interest attribute *attr* *A* set of software entities *P*	A set of entities ***M*** (to be used as abstractions of the measure. A function abs: *P* *abs*: $P \rightarrow \boldsymbol{M}$
To model the distances between the abstractions of the measures	***M***	A set of transformation types T_e
To quantify the distances between the abstractions of the measure	***M***, T_e A measure δ: $\boldsymbol{M} \times \boldsymbol{M} \rightarrow \Re$	
To find a reference abstraction	*attr*, *P*, ***M***	A function ref: *ref*: $P \rightarrow \boldsymbol{M}$ (to return a reference abstraction for *attr*)
To define the software measure	*P*, *abs*, δ, *ref*	A function μ: $P \rightarrow \Re$

- *Find a measurement abstraction*. The software entities of interest (e.g. business objects) must be modeled such that the attribute of interest (e.g. functionality) is emphasized. This means that the model should allow observing to what extent a software entity is characterized by the attribute. The result of the first step is a set of software entities **M** that can be used as measurement abstractions or models of the software entities in **P** for the attribute of interest *attr*. We also require a function abs: $\mathbf{P} \rightarrow \mathbf{M}$ that formally describes the rules of the mapping.
- *Model distances between measurement abstractions*. The next step is to model distances between the elements of ***M***. First, distances between the elements of **M** are modeled as sequences of elementary transformations. Such a sequence represents a series of atomic changes applied to an element of **M** to arrive at another element of **M**. The number of atomic changes that are required to transform one element into the other determines the distance between these elements. The formal outcome of this step is a set T_e of elementary transformation types on **M** that must be used to build the sequences.
- *Quantify distances between measurement abstractions*. Next, a measure δ: $\boldsymbol{M} \times \boldsymbol{M} \rightarrow \Re$ is defined to quantify the distances between the elements of ***M***.
- *Find a reference abstraction*. After having determined the measurement abstractions, we now need to determine what the model of a software entity in **P** must look like in case that entity would be characterized by the theoretical lowest value of *attr*. This hypothetical "null" model or reference model can then be used as a reference point or norm for measurement. The result of this step is thus the definition of a function *ref* $\mathbf{P} \rightarrow \mathbf{M}$ that returns for each software entity in *P* a reference abstraction for *attr* in **M**.
- *Define the software measurement*. It is this last step that expresses the basic idea of our approach. The software attribute attr is defined and measured as a specific distance within the metric space **M**. The extent to which *attr* characterizes a software entity $p \in P$ is defined by the distance between the actual model of *p* for *attr* (i.e., *abs*(*p*)) and the reference model for *attr* (i.e., *ref*(*p*)). The larger this distance, the more the actual measurement abstraction differs from the norm that has been set and thus the greater the extent to which *attr* characterizes *p*. Hence, the value of *attr* for *p* is the value returned by the measure δ for the pair $(abs(p), ref(p))$. The formal outcome of the last step is the measure μ: $P \rightarrow \Re$ defined such that $\forall p \in P$: $\mu(p) = \delta(abs(p), ref(p))$.

Next, this process is illustrated by describing each step in the validation of the NAP measure.

Validation of the **NAP** *measure*: The theoretical validation for the NAP measure, following the DISTANCE framework is defined now. In order to illustrate the process, the sample ETL process models of Fig. 7 are used.

- *Finding the measurement abstraction*. In the context of our study, the set of **software entities P** is the Universe of Models of the ETL Process that is prominent in certain aspects of the Universe of Discourse (UdD) and **p** is a model of the ETL process (MP), such that $p \in P$. The interest attribute *attr* is the number of activities, which is an aspect of the structural complexity of a model of the ETL process. UA is the Universe of prominent Activity for UdD. The set of activities of a model in an ETL process, called CA (MP) is a subset of UA. All the set of activities of the ETL process model of UMP are the elements of the power set of UA, denoted by $\wp$(UA). The set of measurement abstractions is consistently considered as $\mathbf{M} = \wp\mathbf{(UA)}$ and the abstraction function is defined as:

$$absNAP : \mathrm{UMP} \rightarrow \wp(\mathrm{UA}) : \mathrm{MP} \rightarrow \mathrm{CA}(\mathrm{MP})$$

This function performs the mapping from a process model in its set of activities. For the example shown in Fig. 7, the set of activities are the following:

  - *absNAP*(MP A) = CA(MP A) = {Aggregation, Filter, Load}
  - *absNAP*(MP B) = CA(MP B) = {Conversion, Surrogate, Load}

- *Modeling distances between measurement abstractions*. Once ***M*** has been defined, the following step consists of defining the distances between its elements. To do this, it is necessary to find a set of transformation types ($\mathbf{T}_e$) for the set of abstractions of the

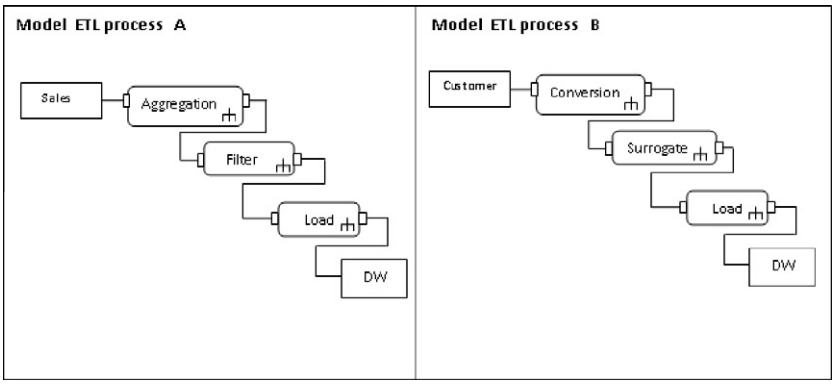

Fig. 7. Example of two models of ETL process defined by using UML activity diagrams.

measurement $\wp$**(UA)**, so that any set of activities can be transformed into any other set of activities by means of an elementary finite sequence of transformations. Since the elements of $\wp$**(UA)** are a set of activities, $\mathbf{T}_e$ only includes two types of elementary transformations: the addition of an activity to the set and the elimination of an activity from the set. Two set of activities $c_1 \in \wp$**(UA)** and $c_2 \in \wp$**(UA)**, c_1 can be transformed into c_2 by first disposing of all the activities of c_1 that are not in c_1 (applying t_{0-NAP}), and then adding all the activities from c_2 to c_1 that were not in the original set c_1 (applying t_{1-NAA}). In the worst of settings, c_1 should transform into c_2 by means of an empty set of activities. Formally,

$$t_{0-NAP} : \wp(\mathrm{UA}) \to \wp(\mathrm{UA}) : c \to c \cup \{a\}, con\ a \in \mathrm{UA}$$
$$t_{1-NAP} : \wp(\mathrm{UA}) \to \wp(\mathrm{UA}) : c \to c - \{a\}, con\ a \in \mathrm{UA}$$

The set of transformations that model the distance between *absNAP* (MP A) and *absNAP* (MP B) the example in Fig. 7 is as follows:

- m0 = {Aggregation, Filter, Load}
- m1 = {Aggregation, Filter, Load, Conversion} = t_{0-NAP}(m0)
- m2 = {Aggregation, Filter, Load, Conversion, Surrogate} = t_{0-NAP}(m1)
- m3 = {Filter, Load, Conversion, Surrogate} = t_{1-NAP}(m2)
- m4 = {Conversion, Surrogate, Load} = t_{1-NAP}(m3)

The length of *T* abs (MP A), abs (MP B) is 4. There are other sequences obtained by varying the order of transformations. Also, they can be used to model the distance between MP A and MP B, but you can see that there is a lack of sequence requiring less than four basic transformations.

- *Quantifying the distances between measurement abstractions*. According to the measurement theory, the notion of distance defined by a structure of segmented additive proximity can be represented for a measure with segment additives. That is, the minimum amount of movement of input data that should be added or removed from a data set movement to be transformed into another set, which is qualified as a measure with additive segments for the structure of definite proximity, qualifies $\wp$(UA). The shortest length of the sequence of elementary transformations taken, for example from c to c', is equal to the cardinality of the symmetrical difference between c and c' This can be used to define a measure when the set of measurement abstractions is a power set:

$$\delta_{NAP} : \wp(\mathrm{UA}) x \wp(\mathrm{UA}) \to (c, c') \to (|c - c'| + |c' - c|)$$

This definition implies that the distance between two sets of activities modeled as the shortest sequence of basic transformations between both sets, are measured by counting the basic transformations of the sequence.

The distance between the sets of activities from the models MP A and MP B is 4, according to the symmetric difference model. Formally:

$$\delta_{NAP}(abs(\text{MP A}), abs(\text{MP B})) =$$
$$|\{Aggregation, Filter, Load\} -$$
$$\{Con version, Surrogate, Load\}|+$$
$$|\{Con version, Surrogate, Load\} -$$
$$\{Aggregation, Filter, Load\}| = |\{Aggregation, Filter\}|+$$
$$|\{Con version, Surrogate\}| = 4$$

- *Finding a reference abstraction*. The condition of this step is to identify the measurement abstraction for the ETL process model with the theoretically lower *functional size*. Since the measurement abstraction is given for the set of activities of ETL process model, the lower theoretical value would be the set without any activity. Therefore, the reference abstraction is given for:

$$ref_{NAP} : \mathrm{UMP} \to \wp(\mathrm{UA}) : \mathrm{MP} \to \emptyset$$

- *Defining the software measurement*. Bearing in mind the proposed example, the number of activities of an ETL process models MP ∈ UMP can be defined as the distance between its set of activities CA (MP) and the of activities ∅ modelled by any shorter sequence of elementary transformations between CA (MP) and ∅. The measure NAP can thus be defined as a function that returns for any MP ∈ UMP the value of the measure δ_{NAP} for the pair of set CA(MP) and ∅:

$$\forall \mathrm{MP} \in \mathrm{UMP} : \delta_{NAP}(\mathrm{CA}(\mathrm{MP}), \emptyset)$$
$$= |CA(MP) - \emptyset| + |\emptyset - CA(MP)| = |CA(MP)|$$

The remainder of the proposed measures can be modeled by means of a set of abstractions (see Table 5). All the measures based on counting elements or relations of the model consequently have a construction process similar to that which it has been followed with the NAP measure.

Since all the measures have been defined by using the process based on a distance measure for their construction, all the measures are defined as distances. This ensures that all the measures are characterized by the scale ratio. Similarly, the derived measures NTEES, RFESA, RDInEE and RDOutES, obtained as ratios of base measures, are characterized by the scale ratio.

Table 5
Function abstraction for the remainder of the measures of the models of ETL processes.

Measure	Function abstraction
NEE	abs_{NEE} : UMP → $\wp$(UEE): MP → CEE(MP) where UEE is the Universe of **Inputs Elements** of an ETL process model in a UdD, CEE(MP) ⊆ UEE is the set of inputs elements of an ETL process model.
NES	abs_{NES} : UMP → $\wp$(UES): MP → CES(MP) where UES is the Universe of **Outputs Elements** of an ETL process model in a UdD, CES(MP) ⊆ UES is the set of outputs elements of an ETL process model.
NFES	abs_{NFES} : UMP → $\wp$(UFES): MP → CFES(MP) where UFES is the Universe of **Input and Output Flows** of an ETL process model in a UdD, CFES is the set of input and output flows of an ETL process model.
NET	abs_{NET} : UMP → $\wp$(UET): MP → CET(MP) where UET is the Universe of **Time Elements** of an ETL process model in a UdD, CET is the set of time elements of an ETL process model.
NEM	abs_{NEM} : UMP → $\wp$(UEM): MP → CEM(MP) where UEM is the Universe of **Merge Elements** of an ETL process model in a UdD, CEM is the set of time elements of an ETL process model.
NEF	abs_{NEF} : UMP → $\wp$(UEF): MP → CEF(MP) where UEF is the Universe of **Fork Elements** of an ETL process model in a UdD, CEF is the set of fork elements of an ETL process model.
NEJ	abs_{NEJ} : UMP → $\wp$(UEJ): MP → CEJ(MP) where UEJ is the Universe of **Join elements** of an ETL process model in a UdD, CEJ is the set of join elements of an ETL process model.
NOSEF	abs_{NOSEF} : UMP → $\wp$(UOSEF): MP → COSEF(MP) where UOSEF is the Universe of **Output Objects in a Fork Element** of an ETL process model in a UdD, COSEF is the set of output objects in a fork element of an ETL process model.
NOEEJ	abs_{NOEEJ} : UMP → $\wp$(UOEEJ): MP → COEEJ(MP) where UOEEJ is the Universe of **Input Objects in a Join Element** of an ETL process model in a UdD, COEEJ is the set of input objects in a join element of an ETL process model.
NODS	abs_{NODS} : UMP → $\wp$(UODS): MP → CODS(MP) where UODS is the Universe of **DataStore Objects** of an ETL process model in a UdD, CODS is the set of DataStore objects of an ETL process model.

5. Empirical validation

In this section, it is described in detail how the previously proposed measures have been empirically tested in order to check their validity by means of defining and conducting a family of experiments. We use two procedures, the first consists of defining a family of experiments as proposed by Ciolkowski et al. [11]. As Basili et al. [3] remark, a family of experiments permits the accumulation of the knowledge needed to extract significant conclusions that can be applied in practice. Subsequently, our second procedure is to define each individual experiment within the family. The definition of each individual experiment is guided by the framework for experimental software engineering of Wohlin et al. [51]. Therefore, while an explanation of our family of experiments has been described in Section 5.1, our individual experiments are explained in Section 5.2.

5.1. Defining a family of experiments

A family of experiments has been planned in order to empirically validate the proposed measures and generalize the results obtained. A family of experiments contains multiple similar empirical studies which pursue the same goal. We have performed a family of experiments based on the experimental method proposed by Ciolkowski et al. [11] Also, we have followed the suggestions provided by Kitchenham et al. [19] and Briand et al. [8] on how to perform controlled experiments. Specifically, a six-step process was employed:

1. *Experiment preparation*. The general goal of the experiments must be stated, i.e. the reason for them. To this aim, the GQM (*Goal-Question-Metric*) template [2] can be used to define the experiment goal. The GQM goal template can be used to articulate the purpose of any study, since it provides a basis to evaluate the appropriateness of a study's specific hypotheses, and dependent and independent variables [3].There are five parameters in a GQM goal template: the object of analysis, its purpose, its focus, its point of view, and its context. As our family of experiments aims to carry out an empirical validation of measures, our goal is to demonstrate the suitability of the previously defined measures. Therefore, our GQM template is as follows:
   - *To analyze* measures related to the structural complexity of ETL processes models in DW,
   - *With the purpose of* evaluating them,
   - *With regard to* their capacity of being used as indicators of the usability of models and the ease of model maintenance,
   - *From the point of* view of ETL designers,
   - *In the context* of students of Software Development and Systems Engineers.

   From this template, variables to be analyzed and hypotheses to be proven have to be defined. Also, the context in which the experiments are carried out (as shown in next step).
   Independent variables are those that can be controlled and changed within each experiment in the family and which make an impact on dependent variables. For our family of experiments, the independent variable is the structural complexity of conceptual models. Furthermore, with the objective of defining measures that could provide useful and objective information with regard to the external quality of the ETL process model, the empirical validation has been focused on two characteristics of external quality which are defined by the ISO 9126, namely usability and ease of maintenance. The former consists of an attribute set that permits us to evaluate the necessary effort that must be made by the user in order to use the model, and the latter refers to the attributes that permit us to measure the effort which is necessary to carry out modifications to the model, whether through error corrections or modification requests. These characteristics will be respectively evaluated by means of two sub-characteristics:
   - *Understandability*: it is a sub-characteristic of the usability. It is the easiness to understand whether the model is suitable and how it can be used for certain tasks or under certain conditions for a particular use.
   - *Modifiability*: it is a sub-characteristic of the maintainability. It is the easiness that permits us to modify aspects of the model, to remove failures or to adapt the model so that it can work in different environments.

   Our family of experiments has been developed in order to discover which measures are useful for evaluating the understandability and modifiability of the ETL processes models. To this aim, these dependent variables are measured by using the subjects' response time in accomplishing the required tasks. Therefore, the following hypotheses are wished to be tested:
   Understandability hypothesis
   - H_{0u}: There is no significant correlation between the measures of structural complexity and understandability time.
   - H_{1u}: There is a significant correlation between the measures of structural complexity and understandability time.

   Modifiability hypothesis
   - H_{0m}: There is no a significant correlation between the measures of structural complexity and modifiability time.
   - H_{1m}: There is a significant correlation between the measures of structural complexity and modifiability time.
2. *Context definition*. In order to ease the generalization of the results, the different groups of subjects must be identified to establish the context of each individual experiment. In our family of experiments, two kinds of subjects are considered:
   - *Professionals*. They are the ideal subjects to generalize the results, and for this reason we have to use this kind of subjects whenever it is possible. In our experiments we relying on Systems Engineers from two companies from Panama: Ingeniería Informática, S.A. (IISA) and Tecnología Inteligente, S.A. (TEINSA)
   - *Students*. They play a very important role in software engineering experimentation, because before performing studies in industrial environments, which requires a lot of resources and time, it is generally useful for researchers to carry out pilot studies with students in academic environments [10]. In addition, students are the next generation of professionals [19] and under some conditions, there is not a great difference between students and professionals. In situations in which the tasks to perform do not require industrial experience, experimentation with students is viable [3,14]. Our individual experiments rely on Undergraduate and Postgraduate students from the Technological University of Panama.
3. *Material*. The objects and material used in the experiments. For example, the document used for collecting data should be presented in terms of its length, complexity, clarity (the number of questions, type, etc.). All characteristics that might have an impact on the results should be mentioned here as formally as possible. As the diffusion of the experimental data is important to the external replication of the experiments [9], all the material from the experiment must be widely available to allow other researchers or practitioners to replicate the study.

   The material prepared for our family of experiments is composed of ten ETL process models. The models are based on UML activity diagrams, including training materials and a survey instrument. Each model has different levels of complexity which were obtained through the variation of the value of the measures, as can be observed in Tables 6 and 7.

Table 6
Values of the base measures for ETL process models.

ETL process model	NAP	NEE	NES	NFES	NET	NEM	NEF	NEJ	NOSEF	NOEEJ	NODS
1	6	1	2	10	1	–	1	–	3	1	3
2	3	1	1	5	1	–	–	–	–	–	2
3	5	3	3	9	1	1	1	2	2	1	2
4	3	2	1	7	1	–	1	3	2	–	3
5	7	2	2	11	–	3	1	1	2	2	4
6	5	3	1	9	3	1	1	–	2	–	4
7	12	6	5	10	–	–	–	–	–	–	4
8	9	4	1	15	2	2	1	1	2	2	5
9	5	2	2	9	3	4	1	–	2	–	4
10	8	1	1	15	–	–	1	1	5	3	2

Table 7
Values of the derived measures for ETL process models.

ETL process model	NTEES	RFESA	RDInEE	RDOutES
1	4	1.6667	0.3333	0.6666
2	4	1.6667	0.5000	0.5000
3	6	1.8000	0.5000	0.5000
4	2	2.3333	0.6667	0.3333
5	6	1.5710	0.5000	0.5000
6	5	1.8000	0.2500	0.2500
7	5	1.6667	0.5000	0.5000
8	5	2.5000	0.8000	0.2000
9	5	1.8000	0.2500	0.5000
10	4	1.8750	0.5000	0.5000

In order to facilitate the replication of our experiments by other researchers or practitioners, all the material are available at (http://www.lucentia.es/index.php/ETLProcessModelling).

4. *Conduct a pilot experiment*. This step is added to the experimental method proposed by Ciolkowski et al. [11] in order to first conduct a pilot study before running the individual experiments, since it allows us to check the experimental materials and process. The subjects that participated in the pilot study were students from the Computer Science PhD Programme, researchers and teacher at the University of Alicante (Spain).
5. *Conduct individual experiments*. According to the general plan of the family of experiments, we have carried out several individual experiments by using the framework proposed in Wohlin et al. [51] as described in Section 5.2.
6. *Family data analysis*. When the resulting data from the individual experiments is collected and analyzed, it is not only important to obtain local conclusions (related to individual experiments), but it is fundamental to extract the overall conclusions obtained from the realization of the family of experiments. This analysis is described in Section 6.

5.2. Defining individual experiments

Each of our individual experiment requires the subjects to perform a set of tasks on the models related to their understandability and modifiability. In these experiments the dependent variables were measured in an objective manner by calculating the time spent by the subjects on performing these tasks. Each experiment has been developed by using the steps described by Wohlin et al. [51]:

- *Definition*. The general goal of each experiment agrees with that proposed when the family of experiments was described: demonstrate the suitability of the defined measures for ETL process model as either usability or maintainability indicators depending on several kind of subjects.
- *Planning*. It describes *how* the experiment is conducted. The planning phase of an experiment can be divided into six steps.
  - *Hypothesis formulation*. We wish to test the hypotheses previously stated when family of experiments was described.
  - *Variables selection*. Individual experiments are related to the variables stated in the family of experiments.
  - *Subjects selection*. The sampling technique for the selection of the subjects is used for convenience (non probability). The subjects are experts in the modeling domain (UML, data bases, etc). However, they had no previous knowledge of the conceptual modeling of ETL processes with UML activity diagrams. A training session took place to provide the subjects with the necessary knowledge to carry out the tasks required in the experiment. Depending on the subjects, within our family of experiments, four individual experiments, as is shown in Fig. 8, have been defined. Basically, the second experiment is a replication of the first and the fourth experiment is a replication of the third.
  - *Experiment design*. In order to obtain significant conclusions of the experiments, statistical analysis methods were applied to the interpretation of the results. We opted to an intra-subject design, since this permits all the subjects to carry out the same experimental tasks. The comparison of performance under various conditions permits us to study the effect of the independent variable. This has the advantage of guaranteeing control of all the variables due to differences among the subjects [7].
  - *Instrumentation*. This is the part of the planning phase that is connected with the instruments that are necessary to carry out and monitor the experiment. It includes the experimental objects, materials, guides and forms of conditioning.

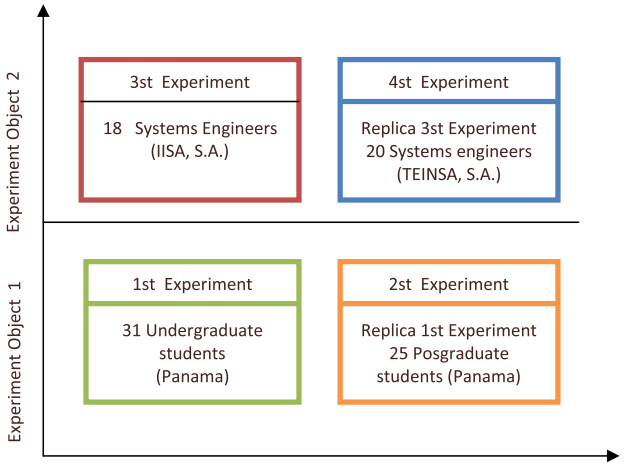

Fig. 8. Overview of the family of experiments.

* *Experimental Objects*: there were ten ETL process models which were developed by using our UML activity diagram approach and, taking into consideration that we needed a cognitively efficient representation, the notation provided explicit mechanisms to support this [18]:
  - *Conceptual integration*: This permits the reader to integrate the information from separate diagrams into a mentally coherent representation of the problem.
  - *Perceptual integration*: The perception signs (contextual orientation and directional information) aid the navigation among the diagrams.

* *Conditioning guides*: An introductory session to ETL process modeling was organized with the objective that the subjects of the experiment would acquire certain dexterity in the management of the experiment. The material also included an answered example, which indicated how the subjects should complete the exercises.

– *Validity evaluation*. It is important to consider the degree of validity of the results of the experiments. That is to say, they should have a suitable validity if they are to generalize the results to the population of interest. Some factors exist that affect the capacity to reach the correct conclusion regarding the relations between the processing and the result of an experiment.
  * *Internal validity*. The internal validity is the degree of confidence in a cause–effect relationship between factors of interest and the observed results. The following issues have been dealt with:
    - *Differences among subjects*. The experiments were intra-subject, so the changeability among subjects was reduced. In the experiments all the subjects had, approximately, the same experience of working with UML and databases.
    - *Differences among models*. All the models were designed with different degree of complexity.
    - *Learning effects*. All the tests in each experiment were put in a different order, to avoid learning effects. The subjects were required to answer in the order in which the tests appeared. However, we could not control whether they really followed the order because they did the tests alone.
    - *Fatigue effects*. On average the experiment lasted for less than one hour, so fatigue was not very relevant. Also, the different order of the tests helped to cancel out these effects.
    - *Subject motivation*. All the students who were involved in the experiments participated voluntarily, in order to help us in our research.

  * *External validity*. External validity is the degree to which the research results can be generalized to the population under study and to other research settings. The greater the external validity, the more the results of an empirical study can be generalized to actual software engineering practice. Three threats to validity have been identified which limit the ability to apply any such generalization:
    - *Materials and tasks*. In the experiment we have used ETL process models which were representative tasks of real cases, but it is necessary to carry out empirical studies using real ETL processes.
    - *Subjects*. To solve the difficulty of obtaining professional subjects, we used, advanced students from a software development degree. But, we were aware that in the context of the family we had to include at least two experiments with professionals to ease the results generalization. However, the tasks to be performed with the experiments of the family, do not require high levels of industrial experience, so, experiments with students can also be appropriate.
    - *Environment*. The experiment was carried out in the university, but the tasks were carried out manually (with paper and pencil). It would be recommendable to use CASE tools in future experiments, thus permitting a more realistic environment to be provided.

• *Operation*. This is the phase in which the experiment is executed, i.e. in which the collection of the data that will be analyzed is carried out. The execution of the experiment is a process that has as input the design of the experiment and as the main output the validated data.
  – *Preparation*. An intensive subject preparation session took place before the experiment began. Nevertheless, the subjects did not know either what aspects we intended to measure or what the formulated hypotheses were. The material that the subjects were provided with was a set of ten ETL process models plus a solved example. Each subject received material composed of ten ETL process models, five of which contained questionnaires about the model and five of which contained modification exercises (see Appendix A). In the case of the questionnaires relating to the understandability of the model the subjects had to answer "yes" or "no" to the five questions listed, and in the case of the questionnaire relating to modifiability they had to carry out four modifications by adding or eliminating activities. The subjects were requested to write the time that they started and completed each of the questionnaires.
  – *Execution*. The experiment execution was controlled. Therefore, no interaction took place between the subjects. All the described material was distributed to the subjects and an explanation of ETL process modeling by using UML activity diagrams was given. The subjects had an hour to resolve the exercises (a pilot experiment had previously been carried out to determine an approximate time for resolving the exercises in which we obtained a time of 40 min).
  – *Data validation*. It is important to consider how valid the results of the experiment are. To do so, once the data had been collected, we ensured that the answer sheets were complete and that the answers and the modifications were understandable. All the questionnaires were correct. The data was then digitalized.

In the next sections, each individual experiment is described in detail.

5.2.1. First experiment

In this subsection we describe the first experiment we have carried out to empirically validate the proposed measures.

The group of subjects in the first experiment was composed of 31 students from the Software Development Degree of the Faculty of Computational Systems Engineering at the Technological University of Panama.

The analysis of the results is presented according to the research questions stated. For the analysis we used the software SPSS version 15, which is an appropriate statistical tool, with the object of promoting the experimental processing.

The statistical analysis was carried out from the summary of the data. This summary was composed of the values of the measures for each of the models and of the averages in the understandability and modifiability times and the subjective appraisal of the models. In the first instance the *Kolmogorov–Smirnov* test was applied in order to corroborate whether the distribution of the data obtained in the experiment was normal. We consequently obtained that the distribution was not normal, and therefore decided to used a statistical non parametrical test as the coefficient of the *Spearman* correlation with a significance level of $\alpha = 0.05$ which indicates the probability of rejecting the null hypothesis when this is certain, and which has a confidence level = 95%. Each of the measures was correlated separately from the understandability and modifiability times and the subjective appraisal of the models.

Table 8 shows the results of the correlation analysis of the first experiment for the understandability and modifiability times and the subjective appraisal of the models. It shows that a correlation exists (rejecting the hypothesis H_{0u}) between the understandability times and the measures NES, NFES, NEM and NTEES. With regard to the time employed by the subjects in the modification of the models, the correlation analysis indicates that a correlation (rejecting the hypothesis H_{0m}) exists between the modifiability times and the measures NES, NFES, NEM and NTEES.

5.2.2. Second experiment (replica of the first experiment)

In order to confirm the results obtained in the first experiment, it was replicated under the same conditions (strict replication) [3]. As the majority of the steps are identical to those of the first experiment we shall only indicate those issues that were different. The second experimental group was composed of 25 students from the Software Engineering Master degree, from the Faculty of Computational Systems Engineering, of the Technological University of Panama. The hypothesis, experimental material and collected data were the same as in the first experiment. As with the second experiment, we applied the *Kolmogorov–Smirnov* test, which revealed that the data collected did not have a normal distribution, and we therefore also used the statistical non parametrical *Spearman* test correlation coefficient with the significance level of $\alpha = 0.05$. The results obtained are shown in Table 8.

According to the results shown in Table 9, a correlation exists (rejecting the hypothesis H_{0u}) between the understandability times and the measures NES, NODS and NTEES. Regarding the modification times of the models, the correlation analysis indicates that a correlation exists between the modification times and the measures NES, NEM and NTEES.

Table 8
Results of the analysis of the Spearman correlation coefficient (first experiment).

Measure	Understandability time	Modifiability time
NAP	−0.022, $p = 0.952$	0.155, $p = 0.670$
NEE	0.205, $p = 0.569$	0.277, $p = 0.438$
NES	0.637*, $p = 0.048$	0.705*, $p = 0.023$
NFES	0.618*, $p = 0.057$	0.634*, $p = 0.049$
NET	0.145, $p = 0.690$	0.034, $p = 0.927$
NEM	0.728*, $p = 0.041$	0.812*, $p = 0.014$
NEF	−0.531, $p = 0.175$	−0.659, $p = 0.016$
NEJ	−0.018, $p = 0.934$	0.143, $p = 0.787$
NOSEF	−0.257, $p = 0.540$	0.353, $p = 0.391$
NOEEJ	0.224, $p = 0.562$	0.080, $p = 0.838$
NODS	0.495, $p = 0.146$	0.457, $p = 0.189$
NTEES	0.718*, $p = 0.019$	0.776*, $p = 0.008$
RFESA	0.508, $p = 0.134$	0.452, $p = 0.190$
RDInEE	−0.070, $p = 0.848$	−0.249, $p = 0.488$
RDOutES	0.215, $p = 0.550$	0.259, $p = 0.471$

Table 9
Results of the analysis of the Spearman correlation coefficient (second experiment).

Measure	Understandability time	Modifiability time
NAP	−0.022, $p = 0.952$	0.155, $p = 0.670$
NEE	0.278, $p = 0.439$	0.195, $p = 0.589$
NES	0.683*, $p = 0.029$	0.650*, $p = 0.042$
NFES	0.461, $p = 0.180$	0.575, $p = 0.082$
NET	0.442, $p = 0.200$	0.138, $p = 0.704$
NEM	0.397, $p = 0.331$	0.714*, $p = 0.047$
NEF	−0.424, $p = 0.296$	−0.558, $p = 0.151$
NEJ	−0.295, $p = 0.570$	0.065, $p = 0.903$
NOSEF	−0.323, $p = 0.436$	−0.225, $p = 0.593$
NOEEJ	−0.214, $p = 0.581$	−0.165, $p = 0.672$
NODS	0.760*, $p = 0.011$	0.408, $p = 0.241$
NTEES	0.793*, $p = 0.006$	0.689*, $p = 0.028$
RFESA	0.420, $p = 0.227$	0.328, $p = 0.354$
RDInEE	−0.014, $p = 0.969$	−0.277, $p = 0.439$
RDOutES	0.152, $p = 0.675$	0.232, $p = 0.519$

5.2.3. Third experiment

The third experiment was developed as part of the family of experiments. This experiment was developed in other context (professionals) to validate the proposed measures.

The group of subjects in the third experiment was composed of eighteen systems engineers from the Ingeniería Informática, S.A. Company, which is a company dedicated to using processes and innovative solutions to construct a path towards the centre of business knowledge. This company has a vast experience in the development and implementation of Business Intelligence tools.

The objects were 10 ETL process models, to which they have made some modifications. The dependent variables were measured by the time the subjects spent on answering the questions in the first section related to the understandability of each model (understandability time) and by the time the subjects spent on carrying out the tasks required in the second section of the experiment (modifiability time). With regard to the first two experiments, there is a variant that consists of some changes in the ETL process model of the experimental material, to confirm whether the non validated measures in the first two experiments could be useful to evaluate the usability and maintainability of the ETL process model.

We applied the *Kolmogorov–Smirnov*. As the data was non normal we used the *Spearman* correlation coefficient, with a level of significance of $\alpha = 0.05$, to correlate each of the measures separately with the understandability time and modifiability time and a subjective validation of understandability and modifiability (see Table 10).

Upon analyzing Table 10, which shows the results of the correlation analysis of the third experiment for the understandability and modifiability times, it can be concluded that for the first experiment a correlation exists (rejecting the hypothesis H_{0u}) between the understandability times and the measures NES, NFES, NEM, NEF, NTEES, because the coefficient of correlation is greater than 0.6320. The measures NAP, NEE, NET, NEJ, NOSEF, NOEEJ, NODS, RFESA, RDInEE and RDOutES do not appear to be correlated with understandability. Regarding the modifiability time, only the measures NEM and NOEEJ have a correlation with modifiability.

5.2.4. Fourth experiment (replica of third experiment)

In order to confirm the results obtained in the third experiment, it was replicated with professionals from an enterprise under the same conditions (strict replication). The fourth experiment was developed in the Tecnología Inteligente, S.A. Company, which is a company dedicated to the Business Intelligence area. The subjects were twenty systems engineers. The experiment is, in turn, specific, it being centered upon a set of measures of structural complexity for ETL process models in DWs.

Table 10
Results of the analysis of the Spearman correlation coefficient (third experiment).

Measure	Understandability time	Modifiability time
NAP	−0.190, $p = 0.598$	0.185, $p = 0.815$
NEE	0.103, $p = 0.776$	−0.175, $p = 0.629$
NES	0.657*, $p = 0.039$	0.487, $p = 0.153$
NFES	0.687*, $p = 0.028$	0.037, $p = 0.919$
NET	0.315, $p = 0.376$	0.454, $p = 0.188$
NEM	0.637*, $p = 0.048$	0.645*, $p = 0.084$
NEF	0.820*, $p = 0.013$	0.618, $p = 0.103$
NEJ	−0.164, $p = 0.756$	−0.272, $p = 0.720$
NOSEF	0.294, $p = 0.479$	0.354, $p = 0.000$
NOEEJ	0.560, $p = 0.117$	0.725*, $p = 0.027$
NODS	0.414, $p = 0.234$	0.244, $p = 0.496$
NTEES	0.749*, $p = 0.013$	0.458, $p = 0.183$
RFESA	0.355, $p = 0.314$	0.043, $p = 0.905$
RDInEE	−0.324, $p = 0.361$	−0.333, $p = 0.348$
RDOutES	0.460, $p = 0.181$	0.441, $p = 0.202$

Table 11
Results of the analysis of the Spearman correlation coefficient (fourth experiment).

Measure	Understandability time	Modifiability time
NAP	0.217, $p = 0.547$	−0.113, $p = 0.775$
NEE	−0.270, $p = 0.450$	0.179, $p = 0.620$
NES	0.734*, $p = 0.016$	0.690*, $p = 0.027$
NFES	0.693*, $p = 0.026$	0.010, $p = 0.979$
NET	0.262, $p = 0.464$	0.662*, $p = 0.037$
NEM	0.893*, $p = 0.003$	0.464, $p = 0.247$
NEF	−0.424, $p = 0.396$	0.065, $p = 0.903$
NEJ	0.237, $p = 0.651$	−0.272, $p = 0.720$
NOSEF	−0.101, $p = 0.811$	−0.204, $p = 0.628$
NOEEJ	−0.455, $p = 0.219$	−0.478, $p = 0.193$
NODS	0.322, $p = 0.363$	0.531, $p = 0.114$
NTEES	0.756*, $p = 0.088$	0.743*, $p = 0.014$
RFESA	0.355, $p = 0.314$	0.146, $p = 0.687$
RDInEE	−0.324, $p = 0.361$	−0.173, $p = 0.634$
RDOutES	0.460, $p = 0.181$	0.367, $p = 0.296$

According to the results obtained for the fourth experiment presented in Table 11, a correlation exists (rejecting the hypothesis H_{0u}), between the understandability times and the measures NES, NFES and NEM. Regarding the time employed by the subjects in the modification of the diagrams, the correlation analysis indicates that only the measures NES, NET and NTEES have a correlation with modifiability.

6. Discussion

Once the individual experiments had been carried out, we performed a global analysis of the results in the context of the family of experiments to determine whether the general goal of the empirical validation had been achieved. A general summary of the results obtained in each individual experiment is provided in Table 12.

Four experiments were performed in which 94 subjects belonging to the following groups participated: undergraduate students and systems engineers. According to the results, the following general conclusions were obtained:

- The measures NES (*number of outputs elements in ETL process model*), NFES (*number of input and output flows in the ETL process model*), NEM (*number of merge elements in the ETL process model*) and NTEES (*total number of input and output elements in the ETL process model*) are valid measures which can be used as ETL process model maintainability indicators. This significant group of measures was correlated in at least three of four the experiments with the dependent variables studied.

Table 12
Summary of experiments with correlated measures.

Measure	Definition	Significant correlations understandability	Significant correlations modificability
NAP	Number of activities in the ETL process model		
NEE	Number of inputs elements in ETL process model		
NES	Number of outputs elements in ETL process model	Exp. # 1, Exp. # 2, Exp. # 3, Exp. # 4	Exp. # 1, Exp. # 2, Exp. # 4
NFES	Number of input and output flows in the ETL process model	Exp. # 1, Exp. # 3, Exp. # 4	Exp. # 1
NET	Number of time elements in the ETL process model		Exp. # 4
NEM	Number of merge elements in the ETL process model	Exp. # 1, Exp. # 3, Exp. # 4	Exp. # 1, Exp. # 2, Exp. # 3
NEF	Number of fork elements in the ETL process model	Exp. # 3	
NEJ	Number of join elements in the ETL process model		
NOSEF	Number of output objects in a fork element in the ETL process model		
NOEEJ	Number of input objects in a join element in the ETL process model		Exp. # 3
NODS	Number of data store elements in the ETL process model	Exp. # 2	
NTEES	Total number of input and output elements in the ETL process model NTEES = NEE + NES	Exp. # 1, Exp. # 2, Exp. # 3, Exp. # 4	Exp. # 1, Exp. # 2, Exp. # 4
RFESA	Ratio of input and output flows for each activity in the ETL process model $RFESA = \frac{NFES}{NAP}$		
RDInEE	Ratio of dependencies of input elements in the ETL process model $RDInEE = \frac{NEE}{NTEES}$		
RDOutES	Ratio of dependencies of output elements in the ETL process model $RDOutES = \frac{NES}{NTEES}$		

- The measures NODS (*number of data store elements in the ETL process model*) and NEF (*number of fork elements in the ETL process model*) were correlated with the understandability time in the second and third experiment respectively. As a result, it would appear that NODS and NEF could also be a useful understandability indicator, but it is necessary to confirm this with new empirical studies focused on these measures.
- The measures NFES (*number of input and output flows in the ETL process model*) and NOEEJ (*number of input objects in a join element in the ETL process model*) were correlated with the modifiability time in the first experiment and third experiment respectively. As a result, it would appear that NFES and NOEEJ could also be useful modifiability indicators, but it is necessary to confirm this with new empirical studies focused on these measures.
- It was expected that some measures were correlated. Specifically, the measure NAP does not appear correlated with neither the time of understandability or maintainability. However, this measure seems to be important in an ETL process model and the reason why it lacks correlation with understandability and maintainability could be that the variation in the values of this

measure in the models used in the material has not been significant enough. This should be considered as future work in the planning of new families of experiments.

- The measures NEE (*number of inputs elements in ETL process model*), NET (*number of time elements in the ETL process model*), NEJ (*number of join elements in the ETL process model*), NOSEF (*number of output objects in a fork element in the ETL process model*), NOEEJ (*number of input objects in a join element in the ETL process model*), RFESA (*ratio of input and output flows for each activity in the ETL process model*), RDInEE (*ratio of dependencies of input elements in the ETL process model*) and RDOutES (*ratio of dependencies of output elements in the ETL process model*) are not correlated with understandability and modifiability. In future studies these measures could also be taken into account to demonstrate whether they really have an influence, or they could be definitively discarded.
- The results also suggest that ETL process models used as material in the experiments are reliable and valid to evaluate the proposed measures. However, additional variables may be added and new experiments should be performed. It is worth to point out that our family of experiments aims to provide a base to determine the impact of measurable variables, such as the understandability and maintainability in the ETL process models.
- Apart from structural complexity of ETL process models, there may be other factors that could threaten the validity of our experiments. Importantly, we were concerned about the layout of the models. Our hypothesis is that we are using well-known standards such as UML activity diagrams, and we believe that layout hardly affects measures or subjects' ability to complete the proposed tasks, since our subjects have in all cases knowledge about this standard. However, our future work will consider experiments which take into account layout properties of UML activity diagrams in a broad sense in order to prove our hypothesis.

Running a family of experiments (including replications) rather than a single experiment provides us with more evidence of external validity and thus allows us to generalize the study results. The same hypotheses were tested and confirmed in four different environments using two different experimental objects. Each replication provided further evidence of the confirmation of the hypotheses. Thus, we can conclude that the general goal of the empirical validation has been achieved.

7. Conclusion and future work

In this paper we have proposed and validated a set of representative measures to evaluate the usability and maintainability of ETL process models. Theoretical validation has been performed by applying two frameworks: Briand et al. [7] and DISTANCE [30,31]. These kinds of validations are important because they are a prerequisite to show the utility of a measure which is the main purpose of an empirical validation. It is clear that both empirical and theoretical validations, as they are defined above, are necessary and complementary.

In order to empirically validate the measures proposed we carried out a family of experiments from which we obtained significant conclusions. These experiments are focused on the relationship between the proposed measures and the understandability and modifiability of ETL process models. In order to analyze this relationship, it was necessary to take into consideration the time the subjects took to carry out the tasks in relation to usability (in particular understandability) and maintainability (in particular modifiability).

As a result of this study, we can conclude that the measures NES, NFES, NEM and NTEES are good maintainability indicators. More maintainable ETL process models can benefit the development of the DW in the following ways:

- Easing the reflection of the changes between the models.
- Reducing costs and effort necessary to change the models.

In the near future, we plan to carry out a case study in enterprise environments. We plan to investigate the generalization of the results obtained in the context of ETL process models. We also plan to motivate other researchers to replicate our study.

Acknowledgments

This work is financed by the following projects: MESOLAP (TIN2010-14860) of the Office of Education and Science of Spain and QUASIMODO (PAC08-0157-0668) of the Office of Education and Science of Castilla-La Mancha, Spain. Lilia Muñoz has a scholarship of the National Secretary of Science, Technology and Innovation (SENACYT) and the Institute for the Formation and Advantage of Human Resources (IFARHU), of the Republic of Panama. We would like to express our gratitude to Jesús Pardillo for his fruitfully discussions about the preliminary versions of this paper.

Appendix A

Questionnaire for ETL process model (understandability).

A.1. *Questionnarie 1*

Write down the starting hour (indicate hh:mm:ss) ———-

1. **Answer the follow question:**

1. ——————– Could the *SurrogateSales* activity be carried out without previously carrying out the *SummedSales* activity?
2. ——————– Would the load of the Total of Online Sales of Tickets be an output product of the *SalesLoader* activity?
3. ——————– Does the ETL process finish once the *ProductLoader, SalesLoader and TimeLoader* activities have been carried out?
4. ——————– Could the process be carried out if the *SummedSales* operation was not carried out?
5. ——————– Should the attributes be grouped in order to complete the *SummedSales* activity?

Write down the ending (indicate hh:mm:ss) ————-
Questionnaire for ETL process model (modifiability).

A.2. *Questionnarie 2*

Assignments
Write down the starting hour (indicate hh:mm:ss) ———

1. **Make the necessary modifications for the following requirements:**

1. A time condition must be included in each load activity.
2. A new activity, *SalesFilter*, must be included after the *SummedSales* activity, which receives as input the information generated from*SummedSales*.
3. A new activity, *SalesDiscard*, must be included to determine which Sales data are not correct. This activity must be after the *SalesLoad* activity and before the *Sales Fact* table.
4. A new activity, *ProductsConversion*, must be executed before the *ProductsLoad* activity.

Write down the ending hour (indicate hh:mm:ss)

References

[1] M. Balta, V. Felea, Using Shannon Entropy in ETL Processes, IEEE Computer Society, 2007, ISBN 0-7695-3078-8. pp. 151–156.
[2] V. Basili, H. Rombach, The TAME project: towards improvement-oriented software environments, IEEE Transactions on Software Engineering (1988) 758–773.
[3] V. Basili, F. Shull, F. Lanubile, Building knowledge through families of experiments, IEEE Transactions on Software Engineering 25 (4) (1999) 456–473.
[4] G. Berenguer, R. Romero, J. Trujillo, M. Serrano, M. Piattini, A set of quality indicators and their corresponding metrics for conceptual models of data warehouses, in: DaWaK, 2005, pp. 95–104.
[5] L. Briand, K. El Emam, S. Morasca, Theoretical and Empirical Validation of Software Product Measures, Technical Report ISERN-95-03, International Software Engineering Research Network, 1995.
[6] L. Briand, S. Morasca, V. Basili, Property-based software engineering measurement, IEEE Transactions on Software Engineering 22 (1) (1996) 68–86.
[7] L. Briand, J. Wuüst, H. Lounis, A comprehensive investigation of quality factors in object-oriented designs: an industrial case study, International Software Engineering Research Network (1998).
[8] L. Briand, S. Arisholm, F. Counsell, F. Houdek, P. Thevenod-Fosse, Empirical studies of object-oriented artifacts, methods, and processes: state of the art and future directions, Empirical Software Engineering 4 (4) (1999) 387–404.
[9] A. Brooks, J. Daly, J. Miller, M. Roper, M. Wood, Replication of Experimental Results in Software Engineering, Technical Report ISERN-96-10, International Software Engineering Research Network, 1996.
[10] J. Carver, L. Jaccheri, S. Morasca, F. Shull, Using Empirical Studies During Software Courses, Experimental Software Engineering Research Network 2001–2003, LNCS 2765, 2003, pp. 81–103.
[11] M. Ciolkowski, F. Shull, S. Biffl, A family of experiments to investigate the influence of context on the effect of inspection techniques, in: Proceedings of the Sixth International Conference on Empirical Assessment in Software Engineering (EASE), Keele, UK, 2002, pp. 48–60.
[12] F. García et al., An Ontology for Software Measurement, Technical Report UCLM DIAB-04-02-2, Computer Science Department, University of Castilla-La Mancha, Spain, 2004.
[13] F. García, M. Piattini, F. Ruiz, G. Canfora, C. Visaggio, FMESP: framework for the modeling and evaluation of software processes, Journal of Systems Architecture 52 (11) (2006) 627–639.
[14] M. Höst, B. Regnell, C. Wholin, Using students as subjects comparative study of students and professionals in lead-time impact assessment, in: Proceedings of the Fourth Conference on Empirical Assessment and Evaluation in Software Engineering (EASE), Keele University, UK, 2000, pp. 201–214.
[15] W. Inmon, Building the Data Warehouse, QED Press/John Wiley, 1992.
[16] ISOIEC 9126-1: Software Engineering Product Quality Part 1: Quality model, 2001.
[17] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis, Fundamentals of Data Warehouses, second ed., Springer-Verlag, 2002.
[18] J. Kim, J. Hahn, H. Hahn, How do we understand a system with (So) many systems research diagrams? Cognitive integration processes in diagrammatic reasoning, Information Systems Research 11 (3) (2000) 284–303.
[19] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Transactions on Software Engineering 28 (8) (2002) 721734.
[20] S. Luján-Mora, P. Vassiliadis, J. Trujillo, Data mapping diagrams for data warehouse design with uml, in: P. Atzeni, W.W. Chu, H. Lu, S. Zhou, T.W. Ling (Eds.), *ER*, Lecture Notes in Computer Science, vol. 3288, Springer, 2004, pp. 191–204.
[21] S. March, A. Hevner, Integrated decision support systems: a data warehousing perspective, Decision Support Systems 43 (3) (2007) 1031–1043.
[22] J-N. Mazón, J. Trujillo, J. Lechtenbörger, Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms, Data & Knowledge Engineering 63 (3) (2007) 725–751.
[23] J-N. Mazón, J. Trujillo, An MDA approach for the development of data warehouses, Decision Support Systems 45 (1) (2008) 41–58.
[24] L. Muñoz, J.-N. Mazón, J. Pardillo, J. Trujillo, Modelling ETL Processes of Data Warehouses with UML Activity Diagrams, LNCS 5333, 2008, pp. 44–53.
[25] Object Management Group. Unified Modeling Language: Superstructure: Version 2.0, formal/05-07-04, 2005.
[26] OMG. Software Process Engineering Metamodel Specification, Adopted Specification, Version 1.0. Object Management Group, Inc., April 2008.
[27] OMG, MDA Guide (draft version 2), 2003. <http://www.omg.org/docs/omg/03-06-01.pdf>.
[28] G. Papastefanatos, P. Vassiliadis, A. Simitsis, Y. Vassiliou. Design Metrics for Data Warehouse Evolution, in: 27th International Conference on Conceptual Modeling (ER'08), Barcelona, Spain, 2008.
[29] J. Pardillo, J.-N. Mazón, J. Trujillo, Model-driven metadata for OLAP cubes from the conceptual modelling of data warehouses, in: 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'08), Turin, Italy, 2008.
[30] G. Poels, G. Dedene, DISTANCE: A Framework for Software Measure Construction, Reserch Report 9937, Dep. of Applied Economics, Katholieke Universiteit Leuven, 1999.
[31] G. Poels, G. Dedene, Distance-based software measurement: necessary and sufficient properties for software measures, Information and Software Technology 42 (1) (2000) 35–46.
[32] N. Prat, S. Cherfi, Multidimensional schemas quality assessment, in: 15th International Conference on Advanced Information Systems Engineering, (CAiSE'03), Klagenfurt, Austria, 2003.
[33] R. Romero, J.-N. Mazón, J. Trujillo, M. Serrano, Quality of Data Warehouses, Encyclopedia of Database Systems (2009) 2230–2235.
[34] M. Serrano, C. Calero, J. Trujillo, S. Luján, M. Piattini, Towards a metrics suite for conceptual models of datawarehouses, Software Audit and Metrics (2004) 105–117.
[35] M. Serrano, C. Calero, J. Trujillo, M. Piattini, Metrics for data warehouse conceptual models understandability, Information & Software Technology 49 (8) (2007) 851–870.
[36] F. Shah, Data Integration Strategies for Reliable Information Delivery, DM Review Magazine, 2005.
[37] C. Shannon, W. Weaver, The Mathematical Theory of Communication, University of Illinois Press, Urbana, 1949.
[38] C. Shilakes, J. Tylman, Enterprise Information Portals.Enterprise Software Team <http://sagemaker.com/company/downloads/eip/indepth.pdf>.
[39] A. Simitsis, P. Vassiliadis, T. Sellis, State space optimization of ETL workflows, IEEE Transactions on Knowledge and Data Engineering 17 (10) (2005) 1404–1419.
[40] A. Simitsis, P. Vassiliadis, A methodology for the conceptual modeling of ETL processes, in: 15th International Conference on Advanced Information Systems Engineering, (CAiSE'03), Klagenfurt, Austria, 2003.
[41] M. Solomon, Ensuring a successful data warehouse initiative, Information Systems Management 22 (1) (2005) 26–36.
[42] K. Strange, ETL Was the Key to this DataWarehouses Success, Technical Report CS-15-3143, Gartner, 2002.
[43] P. Suppes, M. Krantz, R. Luce, A. Tversky, Foundations of Measurement, vol. 2, Academic Press, New York, 1989.
[44] J. Trujillo, S. Luján, A UML based approach for modeling ETL processes in Data Warehouses, in: 22nd International Conference on Conceptual Modeling, (ER'03), Chicago, USA, 2003.
[45] P. Vassiliadis, A. Simitsis, S. Skiadopoulos, Conceptual modeling for ETL processes, in: ACM 5th International Workshop on Data Warehousing and OLAP (DOLAP'02), Virginia, USA, 2002.
[46] P. Vassiliadis, Z. Vagena, S. Skiadopulos, N. Karayannidis, T. Sellis, ARKTOS: towards the modeling, design, control and excution of ETL processes, Information Systems 26 (8) (2001) 537–561.
[47] P. Vassiliadis, A. Simitsis, S. Skiadopoulos, Modeling ETL activities as graphs, in: 4th International Workshop on Design and Management of Data Warehouses, Toronto, Canada, 2002.
[48] P. Vassiliadis, A. Simitsis, M. Terrovitis, S. Skiadopoulos, Blueprints and Measures for ETL Workflows, in: 24th International Conference on Conceptual Modeling (ER'05), Klagenfurt, Austria, 2005.
[49] E. Weyuker, Evaluating software complexity measures, IEEE Transactions on Software Engineering 14 (9) (1988) 1357–1365.
[50] S. Whitmire, Object Oriented Design Measurement, John Wiley & Sons, Inc., 1997.
[51] C. Wohlin, P. Runeson, M. Höst, M. Ohlson, B. Regnell, A. Wesslén, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers., 2000.
[52] H. Zuse, A Framework of Software Measurement, Walter de Gruyter, Berlin, 1998.