Metrics for Measuring Quality of Real-time Design Patterns

Saoussen Rekhis*, Hela Marouane*, Rafik Bouaziz†, Claude Duvallet* and Bruno Sadeg*
*University of Le Havre
Le Havre, France
Email: {Hela.Marouane, Claude.Duvallet, Bruno.Sadeg}@litislab.fr
†University of Sfax
Sfax, Tunisia
Email: {Saoussen.Rekhis, Raf.Bouaziz}@fsegs.rnu.tn

***Abstract*—In recent years, the influence of domain specific design patterns on software quality has attracted increasing attention in the area of software engineering. Indeed, such patterns facilitate the development process of systems, leading to efficient solutions for a particular domain. Since the usage of such patterns has been recommended, there is a need to evaluate their efficiency in a domain, i.e., they answer the question if the provided model encapsulates really the concepts tied to a particular domain. It is also important to determine the amount of pattern elements reuse in order to verify that the patterns cover the majority of domain concepts. The amount of reuse metrics determine how much pattern elements are reused in a designed system, whereas reusability metrics are intended to measure the degree to come up with the specificities of a particular domain in the patterns. Our proposal aims to adapt some existing reuse metrics and to define new metrics for reusability assessment. The usage of these metrics is illustrated through a case study in real-time domain.**

***Keywords*—Reusability metrics; Amount of reuse metrics; Design pattern.**

I. INTRODUCTION

Reusability and software reuse are two major aspects in object oriented software. Reusability is the possibility that an artifact can be reused, i.e., the fitness of an artifact to be reusable. Software reuse is the use of existing software components to build new systems, rather than designing and implementing from scratch. It provides significant improvements in software productivity and quality during the life cycle of a system. Software reuse is supported by different approaches including frameworks, product lines, patterns and program libraries. Our research focuses on the reuse of design patterns that are applied in a specific domain (e.g., real-time domain). These patterns offer flexible architectures with clear boundaries, in terms of well-defined and highly encapsulated parts that are in alignment with the natural constraints of the domain [1]. They present a successful mechanism to capture and promote best practices in the software design. These reasons motivated several researchers on the definition and the application of domain specific design patterns [1][2][3]. However, **these researchers do not provide** a quantitative evaluation of effectiveness of applying these patterns.

In this paper, we have seen necessary (i) to adapt two existing metrics namely, Class Template Factor (CTF) and **Function Template Factor (FTF) [4] to measure, respectively,** the amount of pattern classes reuse and operations reuse in a given application and (ii) to add a new metric to compute the amount of attributes reuse since each class in a pattern has two essential parts, corresponding to its attributes and its operations. Moreover, we are interested in defining new metrics for assessing reusability of domain specific design patterns. The aim of these metrics is to determine whether the patterns represent the concepts that suit a specific domain; they compute the degree to meet the concepts related to this domain. These metrics are applied in a validation step of domain specific design patterns. After their definition, these patterns are used to model different systems in the considered domain. For each pattern reuse, we compute the amount of reuse metrics and reusability metrics in order to evaluate the quality of patterns. Indeed, it is essential to show that the application designers need only to add some system specific elements since the majority of application elements are reused from the patterns. Without computing the amount of reuse metrics, we are not sure that the patterns cover the majority of domain concepts. Thereafter, we consider an example of a real-time application (the freeway traffic control application) designed without and with using a domain specific pattern (sensor pattern [5]), as the base for the illustration of the defined metrics. We also interpret how the values measured of these metrics may contribute and be used effectively to evaluate the quality of the sensor pattern.

The remainder of this paper is organized as follows. Section II presents related work. The definition of metrics for the measure of the amount of domain specific design patterns reuse and the reusability assessment is described in Section III. The explanation of these metrics is presented in Section IV. This latter gives a case study to illustrate these metrics. The evaluation of applying the sensor pattern is described in Section V. Finally, Section VI concludes the paper and outlines our future work.

II. RELATED WORK

The use of reusable components (e.g., design patterns) provides a key element in improving the way software is developed and supported over its life cycle. Design and software reuse reduce development efforts and increase the quality of developed systems. In this context, a critical issue is to identify and qualify reusable components. For these reasons, several works on reuse and reusability metrics have been proposed.

A. Amount of reuse metrics

Frakes et *al.* postulated in [6] that "amount of reuse metrics are used to assessing and also monitoring the reuse improvement effort by tracking of the percentages of reuse for

 ISBN: 978-1-61208-304-9

life cycle objects". These metrics aim to determine how much reuse is present within a given system. The common form of these metrics is defined as the ratio between the amount of the life cycle object reused and the total size of the life cycle object [6]. However, there are many ways to implement this metric. Each way provides different aspects of the reuse ranging from how much code is reused to how often it is reused. For example, the amount of code reuse is defined as the ratio between the number of reused lines of code in a system and the total lines of code in a system.

Frakes *et al.* have shown in [7] various implementations of reuse level (RL) and reuse frequency (RF) metrics which have been proposed in [6] for measuring amount of reuse. RL and RF metrics are measured relative to different granularity of items (e.g., line of codes, functions, files and projects) of source software (i.e., C, java and C++).

Zaigham *et al.* [8] analyze the existing amount of reuse metrics on the basis of their industrial applicability. These metrics are applied to different software projects written in C++ to provide a complete understanding of the level of correlation that exists between them and other software metrics such as cyclomatic complexity, volume and lines of code.

Aggarwal *et al.* have proposed in [4] two metrics for measuring amount of reuse in object oriented software using generic programming in the form of templates. The first metric, called CTF, is defined as a ratio between the number of classes using class templates and the total number of classes in a source code. The second metric, called FTF, is defined as a ratio between the number of functions using function templates and the total number of functions.

These works are focused on different reuse metrics, aiming to measure the amount of reuse of software components and to determine the portion of the new or modified code and the portion of the reused code. These metrics only deal with source code which is typically available at the later stages of the software life cycle, failing to address the importance of the software *artifacts* produced during earlier stages such as analysis and design. So, we see that it is necessary to define metrics for the assessment of the level of design structures reuse in application models. In fact, analysis and design are crucial phases in software development, because they heavily influence the cost of the implementation and maintenance phases. Thus, we intend hereafter to adapt existing reuse metrics defined in [4] for measuring the amount of patterns reuse in applications designed with UML. Moreover, we will add another metric to compute the amount of attributes reuse.

B. Reusability assessment

Reusability metrics indicate the possibility that a component is reusable and enable to identify a good quality of a component for reuse, but, they don't provide a measurement of how many components are reused.

Different studies are based on the definition of reusability metrics.

Bhatia *et al.* [9] have proposed an approach to measure the reusability of a class diagram based on Depth of Inheritance Tree (DIT) of a class, Number of Children (NOC) and Coupling Between Object classes (CBO) metrics [10]. This approach consists to define a formula for reusability based on the principle that DIT and NOC have positive effect on reusability, whereas CBO has negative impact on reusability of a class. The authors consider that reusability of a class diagram is equal to the maximum reusability of a class in the diagram.

Gill *et al.* [11] have proposed new metrics which can be computed from inheritance hierarchies: Breadth of Inheritance Tree (BIT), Method Reuse Per Inheritance Relation (MRPIR), Attribute Reuse Per Inheritance Relation (ARPIR), Generality of Class (GC) and Reuse Probability (RP). BIT metric is compared to two existing metrics (DIT [10] and NOC [10]) to indicate that this metric measures the breadth of the whole inheritance tree, not to compute the number of immediate sub classes of a class. MRPIR and ARPIR metrics are compared respectively to Method Inheritance Factor (MIF) [12] and Attribute inheritance Factor (AIF) [12] to highlight that these two proposed metrics give clearer picture of reuse due to inheritance. In fact, MRPIR metric (respectively ARPIR metric) computes average number of reused methods (respectively attributes) in inheritance hierarchy and not in all classes. GC metric considers the generality of the class as feature of reusability whereas DIT does not consider characteristics of the class.

Subedha *et al.* [13] have used reuse utility percent and reuse frequency metrics as the assessment attributes for reusability of the software component in context level. These metrics determine which components have high reuse potential from a set of standard components in an existing environment.

The previous metrics estimate the probability of reusability of a component and evaluate its design quality (e.g., when CBO increases, reusability decreases and it becomes harder to modify the software system). These metrics indicate that whether or not the components are reusable in the future. **But, they do not answer an essential question:** Do the reusable components represent the specificities of a particular domain? In order to fill this lack, we propose in this paper other metrics for reusability assessment of domain specific design patterns. The aim of these metrics is to show if these patterns are well-defined and they take into account the concepts of the considered domain.

III. METRICS DEFINITION

In this section, we adapt some existing metrics related to the amount of reuse for class diagrams. We also define new metrics that determine the reusability of patterns, i.e., the probability of their reuse.

A. Amount of reuse metrics

We have to adapt the pair of metrics CTF and FTF [4] to compute how much classes and operations to reuse are present within a given application. Moreover, we consider as important to add a new metric, called Attribute Reuse Level, to measure reuse level of attributes in each class of a system. In fact, attributes are essential elements that represent the properties of a class.

The values of these metrics range from 0 to 1. When reuse of patterns elements increases, the reuse level value approaches to 1. A reuse level of 0 indicates no reuse of pattern elements.

 ISBN: 978-1-61208-304-9

1) Metric 1: Class Reuse Level (CRL): This metric is defined as the ratio between the number of reused pattern classes (*RPC*) and the total number of classes in the designed system as shown in (1).

Let us consider a model, with *n* classes C_1, C_2, ..., C_n.

$$CRL = \frac{\sum_{i=1}^{n} RPC(C_i)}{n} \quad (1)$$

where,

$$RPC(C_i) = \begin{cases} 1 & \text{if } the\ class\ is\ reused\ from\ a\ pattern, \\ 0 & \text{otherwise.} \end{cases}$$

2) Metric 2: Attribute Reuse Level (ARL): This metric is defined as the ratio between the number of reused attributes (*RAT*) of pattern classes and the total number of attributes in the designed system as shown in (2).

Let us consider a model having *n* classes C_1, C_2, ..., C_n and m_i attributes a_1, a_2, ..., a_{m_i} for each class C_i.

$$ARL = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m_i} RAT(a_{ij})}{\sum_{i=1}^{n} m_i} \quad (2)$$

where,

$$RAT(a_{ij}) = \begin{cases} 1 & \text{if } the\ attribute\ is\ reused \\ & \quad from\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

3) Metric 3: Operation Reuse Level (ORL): This metric is defined as the ratio between the number of reused operations (*ROP*) of pattern classes and the total number of operations in **the designed system, as shown in (3)**.

Let us consider a model having *n* classes C_1, C_2, ..., C_n and k_i operations op_1, op_2, ..., op_{k_i} for each class C_i.

$$ORL = \frac{\sum_{i=1}^{n}\sum_{q=1}^{k_i} ROP(op_{iq})}{\sum_{i=1}^{n} k_i} \quad (3)$$

where,

$$ROP(op_{iq}) = \begin{cases} 1 & \text{if } the\ operation\ is\ reused \\ & \quad from\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

B. Reusability Metrics

We propose new metrics which indicate the possibility that a pattern can be reused in new systems. Moreover, these metrics indicate whether these patterns allow designing the specificities tied to this domain or not. They are calculated from two releases of each application. Release 1 is designed without using any pattern. Release 2 is designed using design patterns. Measurement values of these metrics are always normalized to a number between 0 and 1. When metric values approach to 1, this means that the majority of the pattern elements (i.e., classes, attributes and operations) are recognized in the systems which are designed without using patterns. Thus, the patterns support the requirements related to a particular domain. Otherwise, the value 0 indicates that no pattern elements are identified in the systems designed without using patterns.

1) Metric 1: Class Reusability (CR): The metric CR is defined as the ratio between the number of identified pattern classes (*IPC*) in a model designed without using patterns and the number of reused pattern classes (*RPC*) in this model when designed using patterns as shown in (4).

Let us consider a model with *n* classes C_1, C_2, ..., C_n.

$$CR = \frac{\sum_{i=1}^{n} IPC(C_i)}{\sum_{i=1}^{n} RPC(C_i)} \quad (4)$$

where,

$$IPC(C_i) = \begin{cases} 1 & \text{if } the\ class\ is\ identified\ as \\ & \quad a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

$$RPC(C_i) = \begin{cases} 1 & \text{if } the\ class\ is\ reused\ from\ a\ pattern, \\ 0 & \text{otherwise.} \end{cases}$$

2) Metric 2: Attribute Reusability (AR): The metric AR is defined as the ratio between the number of identified attributes (*IAT*) of pattern classes in a model designed without using patterns and the number of reused attributes (*RAT*) of pattern classes in this model when designed using patterns as shown (5).

Let us consider a model with *n* classes C_1, C_2, ..., C_n and m_i attributes a_1, a_2, ..., a_{m_i} for each class C_i.

$$AR = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m_i} IAT(a_{ij})}{\sum_{i=1}^{n}\sum_{j=1}^{m_i} RAT(a_{ij})} \quad (5)$$

where,

$$IAT(a_{ij}) = \begin{cases} 1 & \text{if } the\ attribute\ is\ identified\ as \\ & \quad an\ attribute\ of\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

 ISBN: 978-1-61208-304-9

$$RAT(a_{ij}) = \begin{cases} 1 & \text{if } the\ attribute\ is\ reused\ from \\ & \ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

3) Metric 3: Operation Reusability (OR): The metric OR is defined as the ratio between the number of identified operations (*IOP*) of pattern classes in a model designed without using patterns and the number of reused operations (*ROP*) of pattern classes in this model when designed using patterns as shown in (6).

Let us consider a model with *n* classes C_1, C_2, ..., C_n and k_i operations op_1, op_2, ..., op_{k_i} for each class C_i.

$$OR = \frac{\sum_{i=1}^{n}\sum_{q=1}^{k_i} IOP(op_{iq})}{\sum_{i=1}^{n}\sum_{q=1}^{k_i} ROP(op_{iq})} \quad (6)$$

where,

$$IOP(op_{iq}) = \begin{cases} 1 & \text{if } the\ operation\ is\ identified\ as \\ & \ an\ operation\ of\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

$$ROP(op_{iq}) = \begin{cases} 1 & \text{if } the\ operation\ is\ reused\ from \\ & \ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

IV. CASE STUDY

In this section, we present a case study as an example to explain the application of reusability and reuse metrics. The measurement of these metrics was carried out in a pattern specific to the real-time domain [5] **(Figure 1)**. We consider this domain as the base of the illustration of the defined metrics since the design of real-time systems is considered to be a complex process, as all components and real-time constraints have to be considered during the design phase. In fact, real-time applications must be able to meet real-time constraints, i.e., they have to guarantee that each action meets its deadline and that data are used during their validity interval. Thus, it is necessary to give a great importance to real-time applications design.

The real-time domain consists of three functionalities: (1) acquisition of data from environment, (2) data analysis and control and (3) sending orders and commands to actuators. For each functionality, we have defined a design pattern that captures RT domain knowledge and design expertise. In this paper, we present only the sensor pattern [5], which focuses on the modeling of data acquisition functionality of real-time domain. This pattern is applied to model the freeway traffic control application.

A. Application description

The COMPASS [3] is a freeway traffic management system intended to improve safety and to provide a better level of service to motorists. According to this system, the current traffic state is obtained from sensors installed in the freeway: inductance loop detectors and supervision cameras. In fact, inductance loop detectors are embedded in the pavement. Their shape may vary depending on the system requirements. Inductance loop detectors, which are active sensors, measure speeds and lengths of vehicles, number of vehicles and occupancy rates of road segments. These acquired measures are updated and transmitted periodically to the Central Computer System to monitor traffic and to identify traffic incidents. Whereas the Closed Circuit Television (CCTV) supervision cameras constitute passive sensors that transmit periodically the images to the Traffic Operation Centre (TOC). These images are used to confirm the reception of data through the inductance loop detectors and to provide information on local conditions. CCTV cameras are normally mounted on the top of 15 meters poles at approximately 1 km apart along the freeway. These cameras are characterized by a resolution 126 x 185 pixels. Each measure, taken from the environment of this system, has a value, a timestamp and a validity interval to verify the temporal consistency of the collected traffic data. In addition, the minimum and maximum thresholds of each taken measure must be defined in order to determine the abnormal values for which COMPASS system may detect an incident.

The data acquisition subsystem of this application is designed without and with using sensor pattern to calculate the reusability metrics defined in Subsection III-B. Whereas the amount of reuse metrics **(Subsection III-A)** are computed based on this application model reusing the pattern. Figures 2 and 3 show respectively the model without and with the use of the pattern. The model without using pattern is designed by three professors who have an experience in UML.

B. Metrics illustration

Table I shows all metrics calculated from the models of freeway traffic management application already presented (c.f. Figures 2 and 3).

TABLE I
REUSE AND REUSABILITY METRICS CALCULATIONS.

	Metrics	Value
Amount of reuse metrics	Class Reuse Level (CRL)	$\frac{5}{9} = 0.55$
	Attribute Reuse Level (ARL)	$\frac{10}{14} = 0.71$
	Operation Reuse Level (ORL)	$\frac{6}{9} = 0.67$
Reusability metrics	Class Reusability (CR)	$\frac{5}{5} = 1$
	Attribute Reusability (AR)	$\frac{7}{10} = 0.7$
	Operation Reusability (OR)	$\frac{4}{6} = 0.67$

According to the model presented in Figure 2, we identify the following classes as elements of sensor pattern: (i) *Sensor*, *InductanceLoop* and *Camera* classes: **they play, respectively,** the role of *Sensor*, *Active_Sensor* and *Passive_Sensor* classes, (ii) *RoadSegment* and *Vehicle* classes: they correspond to the *ObservedElement* class and (iii) *trafficData* class: it matches the *Measure* class. As the *RoadSegment* and *Vehicle* classes play the same role (i.e., *ObservedElement* class), they are

 ISBN: 978-1-61208-304-9

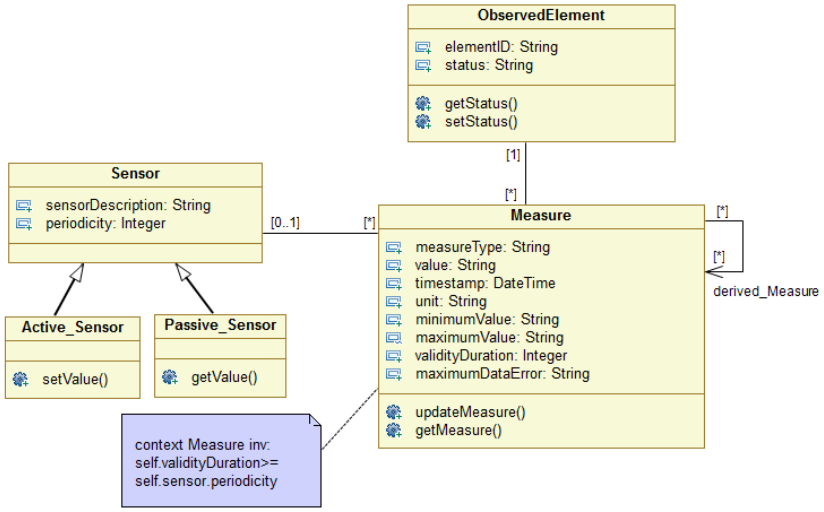

Fig. 1. Sensor pattern [5].

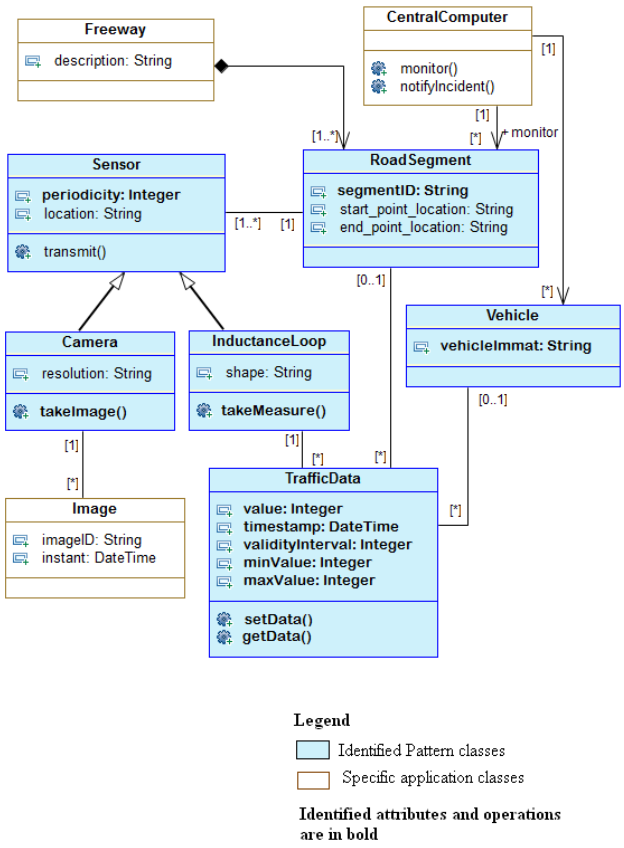

Fig. 2. Data acquisition subsystem of COMPASS without using sensor pattern.

calculated as only one identified pattern element. In other words, the number of identified (*RoadSegment*,*Vehicle*) is equal to 1 (not to 2). This avoids any conflicts in the evaluation of conceptual models with various correct solutions. Indeed, if the elements have the same role in a system, the designer has two solutions: he may use or not the inheritance relationships. Thus, we must calculate the number of identified elements considering the classes which play the same role as one element to obtain the same measurement.

Sensor class has one attribute corresponding to the attribute of the Sensor pattern class: it is *periodicity*. The *takeImage()* operation of *Camera* class matches the *getValue()* operation of *Passive_Sensor* pattern class. Whereas, the *takeMeasure()* operation of *InductanceLoop* class correspond to the *setValue()* operation of *Active_Sensor* pattern class. *RoadSegment* and *Vehicle* **classes have, respectively,** *segmentId* and *vehicleImmat* attributes that correspond to *elementId* attribute of *ObservedElement* pattern class. All attributes and operations

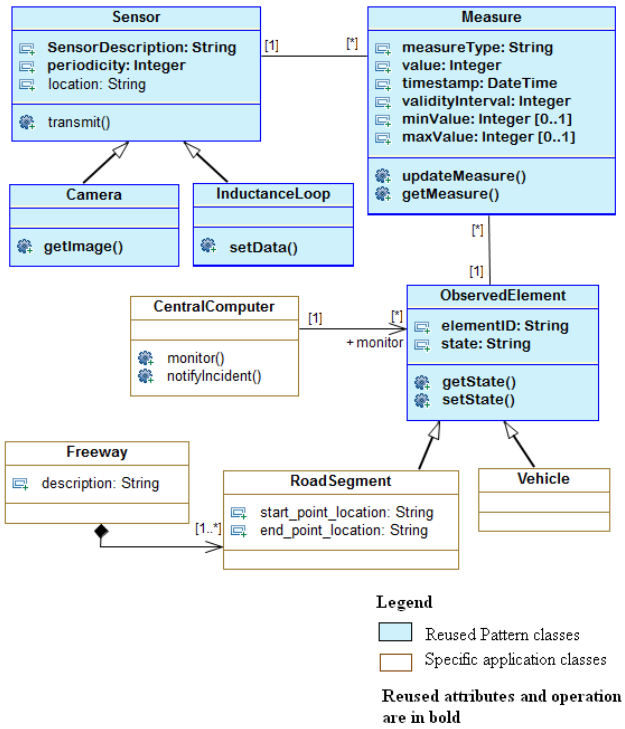

Fig. 3. Data acquisition subsystem of COMPASS with using sensor pattern.

of *TrafficData* class are elements of *Measure* pattern class. The identification of model elements (classes, attributes and operations) as pattern participants is based on a semantic comparison between classes names using a domain dictionary. This dictionary holds for each term (i.e., a class name, an attribute name and operation name) the possible synonyms, antonyms, hypernyms. The construction of this dictionary requires the intervention of pattern designers to determine the linguistic relations for each introduced pair of terms. The designer specifies, for example, that the class name observedElement is the hypernym of vehicle class name.

The reusability metric values presented in Table I approach to 1, it indicates that the majority of pattern participants are recognized in the application model. If we obtain the same results in several case studies, this means that the patterns cover the domain concepts.

As shown in Figure 3, the image is considered as a measure taken by a camera sensor. It has a value (the taken photo), a timestamp and a validity duration. But, it does not have minimum and maximum values. Thus, *minVal* and *maxVal* attributes have the multiplicity [0..1]. The other attributes have the default multiplicity [1].

We have reused the classes *Sensor*, *Active_Sensor*, *Passive_Sensor*, *ObservedElement*, and *Measure*. *RoadSegment* and *Vehicle* classes constitute specific application elements which specialize *ObservedElement* class. This class reuses all features of *ObservedElement* pattern class. We have also reused all attributes and operations of *Measure* class except *Maximum Data Error* attribute. From *Sensor* class, we have instantiated *description* and *periodicity* attributes. In addition, the reused operations of *Active_Sensor* and *Passive_Sensor* classes correspond respectively to *setData()* and *getImage()* operations of *InductanceLoop* and *Camera* classes.

The reuse metric values presented in Table I mean that the majority of application elements (classes, attributes and operations) are reused from the pattern and a limited number of application specific elements are added. This result is approved in the next Section by applying the sensor pattern in several real-time applications.

V. SENSOR DESIGN PATTERN EVALUATION

We present in Table II the values of reuse metrics and reusability metrics obtained for the sensor pattern which is used for modeling ten different real-time applications that we reference A1, A2, ... , A10. On one hand, the values obtained for reuse metrics show that more than half of the classes, the attributes and the operations of real-time applications corresponding to the sensor pattern are instantiated from this pattern. For example, the values of reuse metrics obtained in Table II for the application A1 show that 83% of classes (CRL = 0,83), 62% of attributes (ARL = 0,62) and 87% operations (ORL = 0,87) belonging to the model fragment relative to the sensor pattern are instantiated from this pattern. There are even cases (applications A7, A8 and A9) where all applications classes are instances of pattern classes (CRL = 1). Thus, we deduce a good level of reuse of the sensor pattern elements in the modeling of real-time applications.

On the other hand, the values obtained for reusability metrics calculated for the sensor pattern indicate that the degree of reusability of classes and attributes is better than the reusability of operations. Indeed, we identified all the classes of the sensor pattern (CR = 1) in seven cases of real-time applications modeled without reusing this pattern. In addition, we have identified the majority of the attributes reused from the

 ISBN: 978-1-61208-304-9

TABLE II
RESULTS FOR REUSE METRICS AND REUSABILITY METRICS CALCULATED FOR SENSOR PATTERN.

		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Averge
Amount of reuse metrics	CRL	0,83	0,57	0,7	0,71	0,8	0,83	1	1	1	0,71	0,81
	ARL	0,62	0,59	0,72	0,6	0,83	0,75	0,75	0,76	0,87	0,73	0,72
	ORL	0,87	0,71	0,83	0,75	0,83	0,7	0,83	0,83	0,72	1	0,80
Reusability metrics	CR	1	1	1	1	1	0,6	1	1	0,8	0,8	0,92
	AR	0,8	0,76	0,66	0,73	0,7	0,76	0,77	0,8	0,64	0,81	0,74
	OR	0,71	0,60	0,5	0,66	0,4	0,42	0,6	0,4	0,37	0,5	0,51

pattern classes. For example, the values of reusability metrics obtained in Table II for application A1 show that all reused classes of the sensor pattern are identified (CR = 1), 80% of the attributes are identified (AR = 0,8) and 71% of operations are identified (OR = 0,71). This means that the reuse of this pattern is interesting in the real-time domain because it adequately represents the concepts of data acquisition functionality.

VI. CONCLUSION AND FUTURE WORK

The main objective of this work is to define two categories of metrics that are important for reuse design. The first one aims to assess the reuse level of pattern participants. When the measurement of amount of reuse metrics increases, it means that the pattern elements are simply reused in the model with a minimal possibility for modification. The second category focuses on predicting the reusability of domain specific design patterns. This kind of metrics checks the presence of pattern elements in a system designed without the usage of patterns. When the measurement of reusability metrics increases, it means that the patterns well represent the domain concepts. Reuse and reusability metrics are then illustrated using a case study and they are calculated for ten applications to evaluate the quality of the sensor pattern. The values of reuse metrics show a high degree of the pattern elements reuse (i.e., more than 70% of classes, attributes and operations of the considered applications are modeled by reusing the sensor pattern in the majority of cases). Similarly, the values obtained for reusability metrics show that the attributes, the operations and especially the classes of the sensor pattern are identified in the applications models designed without reuse of this pattern. Thus, we can conclude that it has a good ability to be reused for modeling real-time applications.

In future work, we will check and evaluate the effectiveness of applying other domain specific design patterns (controller and actuators patterns) for real-time systems based on the measurement of these metrics taken for different case studies.

REFERENCES

[1] D. Port, "Derivation of domain specific design patterns," USC Center for software engineering, 1998.

[2] H. Marouane, A. Makni, R. Bouaziz, C. Duvallet, and B. Sadeg, "A real-time design pattern for advanced driver assistance systems," in 17^{th} European conference on Pattern Languages of Programs (EuroPLoP), 2012, pp. C6:1-C6:11.

[3] S. Rekhis, N. Bouassida, C. Duvallet, R. Bouaziz, and B. Sadeg, "A process to derive domain-specific patterns: Application to the real-time domain," in Proceedings of 14^{th} International Conference on Advances in Databases and Information Systems (ADBIS), 2010, pp. 475-489.

[4] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Software reuse metrics for object-oriented systems," in Proceedings of the third ACIS International Conference on Software Engineering Research, Management and Applications (SERA'05). IEEE computer society,2005, pp. 48-54.

[5] S. Rekhis, N. Bouassida, C. Duvallet, R. Bouaziz, and B. Sadeg, "Modeling real-time applications with reusable design patterns," International Journal of Advanced Science and Technology (IJAST), vol. 22, 2010, pp. 71-86.

[6] W. Frakes and C. Terry, "Software reuse: metrics and models," ACM Comput. Surv., vol. 28, no. 2, Jun. 1996, pp. 415-435.

[7] W. B. Frakes, R. Anguswamy, and S. Sarpotdar, "Reuse ratio metrics RL and RF," in 11^{th} International Conference on Software Reuse, Falls Church, VA, USA, 2009.

[8] M. Zaigham and R. Tauseef, "Correlation between amount-of-reuse metrics and other software measures with respect to programming code in c++," Software Quality Control, vol. 11, no. 4, **Nov. 2003**, pp. 301-312.

[9] K. B. Pradeep and M. Rajbeer, "An approach to measure software reusability of OO design," in Proceedings of $2^n d$ National Conference on Challenges & Opportunities in Information Technology (COIT-2008) RIMT-IET, Mandi GobindgarhA, 2008.

[10] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," IEEE Transaction on Software Engineering, vol. 20, no. 6, Jun. 1994, pp. 476-493.

[11] S. G. Nasib and S. Sunil, "Inheritance hierarchy based reuse & reusability metrics in oosd," International Journal on Computer Science and Engineering (IJCSE), vol. 3, no. 6, 2011, pp. 2300-2309.

[12] R. Harrison, S. Counsell, and R. Nithi, "An evaluation of the MOOD set of object oriented software metrics," IEEE Transaction on Software Engineering, vol. 24, no. 6, **1998**, pp. 491-496.

[13] V. Subedha and S. Sridhar, "Design of a conceptual reference framework for reusable software components based on context level," International Journal of Computer Science Issues (IJCSI), vol. 9, no. 3, 2012, pp. 26-31.

 ISBN: 978-1-61208-304-9