

Can We Benefit from Metrics in KBS Development?

Stefan Kramer and Hermann Kaindl

Siemens AG Österreich

Geusaugasse 17

A-1030 Vienna, Austria

{stefan.kramer, hermann.kaindl}@siemens.at

Stefan Schlee

Softlab Austria

Moeringgasse 20

A-1150 Vienna, Austria

sts@softlab.co.at

Abstract

A major issue with knowledge-based systems (KBS) is their maintenance. For dealing with this and other issues in software engineering for conventional software, there is a long tradition of using metrics. More recently, metrics have become popular also for object-oriented development. For the development of a KBS, however, no metrics are available yet.

This paper presents the core of the first metrics suite for KBS development. We focus on the fundamental concepts of coupling and cohesion, as well as on the impact of modifications. All these concepts are very important for ease of maintenance. The application of our metrics to a real-world KBS based on rules and frames helped pinpointing several maintenance problems. Therefore, we propose to use our and other yet to be developed metrics in order to improve KBS development.

1 Introduction

The maintenance of knowledge-based systems (KBS) is one of the major issues in their practical use. According to the experience with conventional software development, many maintenance problems stem from bad designs and implementations. So, improving these is an effective approach to address the maintenance problem.

Both in "traditional" and in object-oriented (O-O) software engineering, *metrics* are used (among other things) to determine current and to predict future qualities of a product relative to expected standards. In particular, it is possible to pinpoint those parts of a product with potential defects. In this way, it is easier to improve the quality of the product and therefore to improve its maintainability.

In order to address the maintenance problem of KBS, we devised the first metrics suite for KBS development.¹

¹The research reported has been carried out in the ViVa (Verification, Improvement and Validation of Knowledge-Based Systems) project, partially funded by the ESPRIT Programme of the Commission of the European Communities

In this paper, we present three metrics of this suite dealing with frames and rules and their relationship in a knowledge base. These metrics represent quantitative models of the fundamental concepts *coupling* and *cohesion*, as well as the impact of modifications.

We have applied these metrics to a real-world KBS based on frames and rules. The numerical values computed there helped locating those frames and rules which would most likely pose problems during the maintenance of the KBS. So, the application of our metrics helped pinpointing maintenance problems.

This paper is organized as follows. First, we present some of the metrics that we devised for frames and rules. Then we show some empirical evidence with these metrics from real-world experience and from using a published knowledge base. After that, we shortly discuss previous work on the use of metrics. Finally, we discuss our approach and the use of metrics in KBS development more generally.

2 Metrics for Frames and Rules

We propose to use metrics for knowledge bases consisting of frames and rules in order to pinpoint potential problems with maintenance. Another possible use of metrics reported in the literature is the evaluation and comparison of alternatives. In both cases, metrics are useful as a means to make certain properties related to quality objective and quantifiable.

In the metrics literature there is a general distinction between *process metrics* (measuring certain aspects of the development process), and *product metrics* (measuring aspects of items produced during software development). In this paper we focus on product metrics related to knowledge bases consisting of frames and rules. In case of procedural attachment in the form of method slots, it would have been possible to simply transfer well-known metrics from the object-oriented paradigm. (For a comprehensive discussion of O-O met-

as project number 6125. The overall goal of the ViVa project is to provide a methodology and a supporting tool-set for V & V of knowledge-based systems in industry. More details can be found at <http://www.cri.dk/ViVa> — the *ViVaWeb* — where also our metrics suite can be tried out.

rics, see [Henderson-Sellers, 1996].) However, for knowledge bases consisting of frames and rules that are not attached to frames, a new set of metrics had to be derived.²

2.1 Procedure for the Derivation and Validation of Metrics

For the derivation and validation of the metrics, we adopted a procedure that is agreed upon in the software engineering literature (e.g. see [Hitz and Montazeri, 1996]). This procedure consists of three steps:

1. Identify the attributes of interest. In this context, attributes are properties of the item in question. For the attributes of interest, quantitative models are to be defined.
2. Derive the metrics for these attributes. The derivation starts with an intuitive account of the so-called *empirical relation system*, i.e., a kind of qualitative model of the attribute of interest. Then formalize the relations established in the empirical relation system.
3. Validate the metrics by determining the *scale type*, by careful application of some of *Weyuker's axioms* [Weyuker, 1988] and by checking the effects of so-called *atomic modifications* [Zuse, 1990].

We followed this procedure, and our description parallels its steps.

2.2 Identification of the Attributes of Interest

Regarding the selection of attributes of interest, we decided to focus on the concepts of coupling and cohesion, which are generally accepted to be relevant both in object-oriented and “traditional” software engineering. Coupling and cohesion can be explained in terms of elements (like modules, classes or frames) that are linked in some way (for instance by function calls, common usage of global variables, or message connections). The degree of connectivity *within* such an element is called cohesion, and the degree of connectivity *among* these elements is called coupling. In general, low coupling and high cohesion are indicators of good modularization and minimal interfaces, and thus contribute to maintainable software. In our context, the elements are frames, and they are linked by those rules where they are referenced together. Regarding cohesion, we are interested in the degree to which the *slots* belong together. This is different from the object-oriented version of cohesion, where the cohesion of *methods* is usually determined by their use of instance variables.

Besides coupling and cohesion, we are interested in measuring the impact that a change of a frame may have on the rule base. This additional attribute supplements coupling and cohesion, and is also related to maintenance.

²For a discussion of the concepts underlying object-oriented systems, frame systems and description logics see Appendix A.

Frame A	
Slot v	
Slot w	
Slot x	
Slot y	
Slot z	

Frame B	
Slot x	
Slot y	

Frame C	
Slot x	
Slot y	

Frame D	
Slot z	

Rule r_1 : if $(A.w = 2)$ and $(A.x = 2)$ and $(B.x = 0)$ then $(C.y = 0)$

Rule r_2 : if $(A.v = 2)$ and $(A.z = 1)$ then $(C.y = 1)$

Rule r_3 : if $(A.w = 2)$ and $(A.x = 0)$ and $(A.y = 1)$ then $(B.y = 0)$

Rule r_4 : if $(B.x = 0)$ and $(D.z = 0)$ then $(C.x = 1)$

Figure 1: Frames and rules of a hypothetical mini-KBS.

2.3 Derivation of the Metrics

The derivation will be illustrated by a running example, a hypothetical mini-KBS as shown in Fig. 1. It consists of four rules. The rules refer to frames (A, B, C, D) and their slots (A.v, A.w, ...).

First we want to derive a quantitative model for coupling. We focus on a single frame, and the degree of coupling between this frame and the other frames. The degree of coupling of a frame can be thought of as the “connectivity” with respect to other frames. The larger the number of other frames that are linked to a frame, the higher the coupling of the frame. As stated above, frames are linked by rules, but more precisely they are linked if and only if there is at least one rule where both frames are referred to. So, the degree of coupling of a frame may simply be defined as the number of other frames that are referred to in the same rules as the frame under consideration.

To formalize this notion of coupling of a frame, we define two auxiliary functions. Let R be the set of rules in the knowledge base, and F be the set of frames in the taxonomy. Then we define *Rules* as

$$Rules : F \rightarrow \mathcal{P}(R)$$

This function determines all rules that refer to a particular frame. For our running example, $Rules(A) = \{r_1, r_2, r_3\}$. Dually, we define *Frames* as

$$Frames : R \rightarrow \mathcal{P}(F)$$

This function simply returns the set of frames referred to by a given rule. For instance, $Frames(r_1) = \{A, B, C\}$.

Now we can define the metric for coupling, $DCpF$ (*Degree of Coupling of Frame*) as:

$$DCpF(f) = \left| \bigcup_{r_i \in Rules(f)} Frames(r_i) \right| - 1$$

In our running example, $DCpF(A) = 2$, since $\bigcup_{r_i \in \{r_1, r_2, r_3\}} Frames(r_i) = \{A, B, C\}$.

Another attribute of interest related to maintenance is

the impact that a modification of some frame has on the rule base.³ In case of such a modification, the affected rules have to be checked if they are still valid. Given that we have to check those rules where the particular frame is referred to, the impact on the rule base may simply be measured by the number of rules using the frame. So we define the metric *IMF* (*Impact of Modification of Frame*) as follows:

$$IMF(f) = |Rules(f)|$$

So, for our running example, $IMF(A) = 3$.

Next, we want to derive a quantitative model for cohesion. Our goal is to define a function measuring the degree to which the slots of a frame belong together (with respect to the rules). Intuitively, two slots belong together if they are connected by the rules, i.e. if they are used in the same rules. In other words, there is a connection between two slots if and only if they are both used by at least one rule. It is convenient to think of this in terms of graphs: for each frame we define a graph, where the nodes represent the slots, and an arc exists if the corresponding slots are both used in at least one of the rules. An obvious way to define the lack of cohesion is to count the number of "disconnected subgraphs", in graph-theoretical terminology the number of components. We define the metric for cohesion, *LChF* (*Lack of Cohesion of Frame*) as follows:

$$LChF(f) = \text{Number of components of graph } G,$$

where the nodes of G represent the slots of f , and an arc exists in G if the corresponding slots are both used in at least one of the rules.

In our example, there are two components of the corresponding graph of frame A , one containing $\{w, x, y\}$, and one containing $\{v, z\}$ (see Fig. 2). Thus, $LChF(A) = 2$.

2.4 Validation of the Metrics

The soundness of these metrics can be checked by determining the scale type, by applying those of Weyuker's axioms that make sense in our context, and by checking the effects of atomic modifications. These "acid tests" are based on measurement theory and are established in the theory of metrics in the software engineering literature.

The *scale type* of all three metrics is absolute, since they are all counting some entities. This means that a full range of descriptive statistics is applicable, for instance, means and standard deviations may be calculated.

Weyuker's axioms state desirable properties of software complexity measures. For various reasons (explained in [Henderson-Sellers, 1996; Chidamber and Kemerer, 1994]) only a subset of Weyuker's axioms makes sense in our context. These properties can be paraphrased as follows:

³This attribute is different from coupling, since it relates elements of *different* sorts (frames and rules), while coupling just relates elements of the *same* sort (namely frames).



Figure 2: An example graph for computing $LChF(A)$.

- Property 1: not all possible frames have the same values for the metric.
- Property 3: for every frame there might exist another frame of the same complexity.
- Property 4: design decisions should be reflected in the values calculated by the metrics.
- Property 5: complexity should increase when adding something.

Properties 1, 3 and 4 are trivially satisfied. For property 5, we define the concatenation of frames as the merging of the slots into a new frame. The proof for property 5 is simple, but we cannot include it in this paper due to lack of space. In brief, all of these properties are satisfied by the metrics.

Finally, we check the effects of *atomic modifications* [Zuse, 1990]. Atomic modifications are the smallest possible changes in the model underlying the calculation of the metric. If the changes in the values of a metric caused by these modifications do not translate back to acceptable statements within the empirical relation system, the metric has to be rejected. In our context, the atomic modifications are additions or deletions of rules or frames. Since our metrics are all basically counting some entities, there is a direct correspondence between the modifications and the values of the metrics. Due to the straightforward derivation of the metrics, the increase in the metrics directly translates back to correct qualitative statements about the KBS. For instance, adding a rule that refers to a frame A increases *IMF* by 1, and the statement in the empirical relation system is that the impact of changing A increases as well.

We briefly sketched three ways of validating the metrics for frames and rules. Summing up, our metrics turned out to be valid according to all three "acid tests".

3 Empirical Results

In this section we present the results obtained for two knowledge bases. First, we present some findings from a real-world project, where the knowledge base is implemented in Kappa [IntelliCorp, 1993], a hybrid frame system with traditional frames and rules. In the second application, we put the generality of the approach to the test, and calculated values for a knowledge base represented in CLASSIC [Brachman *et al.*, 1991], a knowledge representation system based on description logic.

3.1 Results for a "Traditional" Frame System

In the following, we describe the application of the metrics to a real-world KBS implemented using a "traditional" frame system. Our role was to control the quality of

Frame	<i>DCpF</i>	<i>IMF</i>	<i>LChF</i>
<i>DataBoxes</i>	3 (60%)	5 (8%)	1
<i>TMEntry</i>	1 (20%)	2 (3%)	1
<i>StorageDisks</i>	3 (60%)	14 (21%)	1
<i>Spindles</i>	2 (40%)	28 (42%)	1
<i>Diagnosers</i>	5 (100%)	66 (100%)	1
<i>AmfModels</i>	4 (80%)	40 (60%)	1

Table 1: Values obtained by the metrics (*DCpF*...Degree of Coupling of Frame, *IMF*...Impact of Modification of Frame and *LChF*...Lack of Cohesion of Frame) for 6 frames of a real-world knowledge base.

the KBS in question. So we were not involved in the development of the KBS, but in quality assurance. Table 1 summarizes the data obtained by the metrics. To improve the comprehensibility of the results, we also include percentages.

Applying the metric for coupling (*DCpF*) shows that the frame *Diagnosers* couples *all* frames (5 frames, 100%). In addition, the value for *IMF* indicates that a change in this frame affects the *whole* rule base (66 rules, 100%). The second finding is that the frame *AmfModels* couples 4 out of 5 frames (80%), and that 60% of the rules (40 rules) would be affected by some change of this frame. These findings hinted at the possibility that both frames *AmfModels* and *Diagnosers* might have been pragmatically created for implementation considerations rather than designed as semantically meaningful units. So we asked the development team, and they confirmed our assumptions. In effect, the metrics helped us uncover a flaw that might have led to serious maintenance problems.

Checking the cohesion of the frames, we did not find indications for problems using the corresponding metric, since *LChF* = 1 for all frames. However, the applications of our other metrics helped pinpointing serious problems with maintenance. Our experience also suggests that metrics are useful as a means to determine certain properties of KBS in an objective and effective way, and that they help to focus the attention on potential problems.

3.2 Results for a CLASSIC Knowledge Base

We applied our metrics to the “winery” knowledge base that is distributed as a demo of the CLASSIC knowledge representation system. In order to apply our metrics to CLASSIC knowledge bases, it is first necessary to conclude which concept is the right-hand side concept of a rule. In Table 2, we summarize a few results for the “winery” knowledge base. We included values for its most fundamental concepts, *wine* and *meal-course*, and for a subconcept of *meal-course*: *fruit-course*. In addition, we performed a thought experiment, and calculated values for a slightly modified knowledge base: in CLASSIC, the antecedent of a rule has to refer to a defined concept. So if the antecedent of the rule is not yet

Concept	<i>DCpF</i>	<i>IMF</i>	<i>LChF</i>
<i>wine</i> w/o s.	23 (17%)	148 (100%)	7
<i>wine</i> w.s.	1 (20%)	148 (100%)	1
<i>meal-c.</i> w/o s.	0 (0%)	0 (0%)	2
<i>meal-c.</i> w.s.	1 (20%)	57 (39%)	1
<i>fruit-c.</i> w/o s.	1 (0.7%)	2 (1.4%)	2

Table 2: Values obtained by the metrics with and without substitution of “antecedent concepts” for the CLASSIC demo knowledge base “winery”.

defined as a concept, we have to define it to formulate the rule. We were interested in the values for the metrics if we could simply substitute the definitions of these “antecedent” concepts, and refer to them directly in the rules.

The results for the original winery knowledge base show that for the concept *wine* the degree of coupling (*DCpF*) is high (23 frames, 17%), and that the impact of modifications (*IMF*) is very high (148 rules, 100%). This is due to the fact that many rules (91 rules, 61%) are used to define contingent properties of wines (properties that do not belong to the concept itself), and that the rest of the rules (57 rules, 39%) make statements about the relationship between specific meal courses and sorts of wine. Note that the concept *meal-course* itself is never referred to in the rules (0 rules, 0%), but only specific meal courses like *fruit-course*.

Looking at the modified knowledge base, we find that the degree of coupling of the concept *wine* (*DCpF*) is low (1 concept, namely *meal-course*), but the value for *IMF* is still very high (148 rules, 100%). Moreover, *meal-course* itself is referred to in the rules, but it couples only the concept *wine* (so *DCpF* = 1). Although coupling is low, the impact of modifications of this concept (*IMF*) is high (57 rules, 39%).

For the original knowledge base (without substitution of the antecedent definition), the cohesion of *wine* and *meal-course* is low (the lack of cohesion *LChF* is high). The reason for this is that all the rules use only one attribute/role of these concepts. Interestingly, this is not the case if we substitute the antecedent definitions. The metric reflects that the substitution yields rules where attributes/roles of the same concept are used together.

Although the metrics have not been devised for description logics, they helped us make a few interesting observations.

4 Related Work

Coupling and cohesion are well-known and established concepts from traditional software engineering. Although the definition of the *concepts* of coupling and cohesion is clear and established in software engineering, there are various versions of metrics defined for these concepts (e.g., in [Heyliger, 1994]).

Chidamber and Kemerer [1994] were the first to define a metrics suite for object-oriented design and programming. Their metrics suite includes measures for cou-

pling and cohesion. Coupling among objects is defined in terms of incoming and outgoing messages (method invocations) of an object. We defined coupling in analogy to this formulation in the object-oriented framework, where the rules take the role of the methods. Additionally, we defined a metric measuring the impact of modifications of a frame (*IMF*), that is specific for knowledge bases consisting of frames and rules.

Since Chidamber and Kemerer's metric for cohesion suffers several measurement-theoretic anomalies, we took a different approach. We adopted both the view of cohesion as proposed in [Li and Henry, 1993] and the graph-theoretic approach by [Hitz and Montazeri, 1996]. In a graph-theoretic view of Li and Henry's definition, the nodes represent the methods of a class, and they are connected if there is at least one instance variable that is used by both methods corresponding to the nodes. While in this view the measure for cohesion by Li and Henry also counts the number of components of a graph, in our approach the graph is defined in a dual way, i.e., the nodes represent slots, and the arcs are defined by "common" rules.

Besides coupling and cohesion, there are various metrics in the object-oriented literature measuring properties of taxonomies. (Some of them, like *DIT*, the depth of inheritance tree, are problematic from a measurement-theoretic view.) We also developed a few metrics for taxonomies, but in this paper we focus on coupling and cohesion.

In conventional and object-oriented software engineering there is a rapidly increasing number of publications about metrics. To the best of our knowledge, however, metrics have not yet been described in the published literature about KBS development.

5 Discussion

Now let us discuss our approach more generally. The question may arise, for instance, what such numbers signify. In some sense, this question relates to the utility of metrics in general. While this issue is surely beyond the scope of this paper, we claim to have developed our metrics suite according to the state of the art. In addition, we have validated them both according to acknowledged theoretical standards and through practical experience.

It must be understood that these like any other comparable metrics are just special quantitative models of "quality". So, it is very important for successful applications to beware of misunderstandings and misuse. Still, wherever some of these numbers indicate problems, it is worth a closer look.

More specifically, the concepts of coupling and cohesion and therefore also the corresponding metrics are related to the principle of minimal interfaces. While this principle is generally agreed to in the context of software engineering, it is not normally dealt with in frame systems and rule bases. A major effect of adhering to this principle is that we can achieve greater ease of maintenance. Since this should be a goal for the development

based on frames and rules as well, and since also our third metric pursues this goal, we think that the metrics proposed in this paper should be useful for improving the development of KBS based on frames and rules.

Let us shortly point to an interesting difference of our approach compared to the corresponding approach for O-O software development (for the general correspondence between objects and frames see the Appendix A). While we measure how rules use the slots of frames, the usual O-O approach measures how the methods *within* an object use the instance variables of the same object. So, our approach measures more global dependencies between frames and rules that are *outside* of these frames. Normally, global dependencies are more critical for maintenance than local ones. So, we conjecture that our approach may be even more helpful in the context of KBS based on frames and rules than the approach for O-O software development.

Finally, our metrics approach relates to verification and validation of KBS consisting of rules and frames. In contrast to mainstream work in this area dealing with static KBS properties, we focus on the *relation between* rules and frames.

6 Conclusion

In this paper, we presented three metrics from our metrics suite for KBS development. They represent the core of this suite and measure the fundamental concepts of coupling and cohesion, as well as the impact of modifications. In essence, these metrics help pinpointing problems for the maintenance of knowledge bases consisting of frames and rules.

To the best of our knowledge, this is the first published account of a metrics suite for KBS development. Some of our metrics have been developed in analogy to existing metrics in software engineering, but we also developed completely new metrics: the metric measuring the impact of modifications of a frame has no analogue, and our measure for cohesion is dual to the one proposed by [Li and Henry, 1993].

The results from applying these metrics to a real-world KBS ("traditional" frames and rules) as well as to a demo for CLASSIC (description logic and rules) suggest the usefulness of these metrics. So, we think that this approach has the potential to improve the development of a KBS in the sense of enabling quantitative and objective judgements of quality properties relevant for the developer. Therefore, we propose to use our and other yet to be developed metrics in order to improve KBS development.

Acknowledgments

We would like to thank Martin Hitz for useful discussions and comments on an earlier draft of this paper. Leslie Benjamin helped to improve the English. The implementation of our metrics in the *ViVaWeb* is due to Gert Schmeltz Pedersen.

References

- [Brachman *et al.*, 1991] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: when and how to use a KL-ONE-like language. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann, San Mateo, CA, 1991.
- [Chidamber and Kemerer, 1994] S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.
- [Fikes and Kehler, 1985] R. Fikes and T. Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–920, 1985.
- [Henderson-Sellers, 1996] B. Henderson-Sellers. *Object-Oriented Metrics: Measures of Complexity*. Prentice Hall, Upper Saddle River, NJ, 1996.
- [Heyliger, 1994] G.E. Heyliger. Coupling. In J.J. Marciniak, editor, *Encyclopedia of Software Engineering*, pages 220–228. John Wiley & Sons, Inc., 1994.
- [Hitz and Montazeri, 1996] M. Hitz and B. Montazeri. Chidamber & Kemerer’s metrics suite: A measurement theory perspective. *IEEE Transactions on Software Engineering*, 22(4), 1996.
- [IntelliCorp, 1993] IntelliCorp. Kappa user’s guide. Manual, IntelliCorp, Inc., Mountain View, CA, September 1993. Publication Number: K3.0-UG-2.
- [Li and Henry, 1993] W. Li and S. Henry. Maintenance metrics for the object-oriented paradigm. In *Proc. First International Software Metrics Symp.*, pages 52–60, Los Alamitos, CA, May 1993. IEEE Comp. Soc. Press.
- [Minsky, 1975] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975.
- [Weyuker, 1988] E.J. Weyuker. Evaluating software complexity measures. *IEEE Transactions on Software Engineering*, 14(9):1357–1365, 1988.
- [Winston, 1977] P. H. Winston. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 1977.
- [Winston, 1984] P. H. Winston. *LISP*. Addison-Wesley, Reading, MA, 1984.
- [Zuse, 1990] H. Zuse. *Software Complexity: Measures and Methods*. Walter de Gruyter, Berlin, 1990.

A Relation Between O-O Languages and Frame Systems

In order to make this paper self-contained, we summarize the relation between object-oriented languages and frame systems, including systems based on description logics like CLASSIC [Brachman *et al.*, 1991].

Object-Oriented Languages	Traditional Frame Systems	Description Logics
(object) classes	(class) frames	concepts
(object) instances	(instance) frames	individuals
class / instance var.	slots	roles

Table 3: Correspondence of key notions.

Objects and traditional frames have much in common. However, there are also some fundamental differences.

Although Winston heavily uses the notion of a frame due to [Minsky, 1975] in his AI textbook [Winston, 1977], his definition from the viewpoint of implementing frames in LISP simply says that a frame is a generalized property list [Winston, 1984, p. 311]. Traditional frame languages like, e.g., FRL, KEE [Fikes and Kehler, 1985] or Kappa [IntelliCorp, 1993], differ in that they provide frames in a way similar to records in Pascal. The general theme, however, is that a frame is a data structure that combines data stored in slots. While these definitions do not explicitly mention operations (in contrast to most of those for objects), these are typically representable via *procedural attachment*, just being stored in slots like data. But these definitions do not mention at all that data are hidden from the outside.

The principal element of systems based on description logics like CLASSIC [Brachman *et al.*, 1991] is the *structural conceptual object*, or *concept*. The components of a concept are its subsuming⁴ concepts (its super-concepts), and its local internal structure. This structure is expressed in terms of *roles*, which describe potential relationships between instances of the concept and those of other concepts (i.e., its properties, parts, etc.), and *structural descriptions*. In CLASSIC it is possible to model *concepts*, *roles* and *individuals*.

Table A summarizes the correspondences of these key notions in object-oriented languages and frame systems. (Object) classes and (class) frames roughly correspond to concepts (described in expressions according to a description logic). (Object) instances and (instance) frames correspond to individuals. Class / instance variables and slots have roles as their counterpart.

While it is possible to put these notions into correspondence, there are essential differences particularly between concepts in description logics and traditional frames as well as objects: firstly, they differ in the way subsumption is (formally) defined. Secondly, because the formal and well-defined descriptions of the former systems are based on a logic. Thirdly, the general intent of the former systems is to model the semantics and conceptual structure of a space of concepts, rather than to act as a data structure in a computer implementation. Still, the correspondence may be exploited for the purpose of this paper — to apply similar metrics.

⁴ *Subsumption*: given two CLASSIC descriptions, an important question to consider is whether one *subsumes* the other—that is, whether an instance of one is always an instance of the other.