# The Role of the Conceptual Invariants Regarding the Prevention of the Software Artefacts' Obsolescence

*Răzvan Bocu*
Department of Mathematics and Computer Science, Transilvania University of Braşov,
Eroilor 29, Braşov 500036, Romania
razvan.bocu@unitbv.ro

*Dorin Bocu*
Department of Mathematics and Computer Science, Transilvania University of Braşov,
Eroilor 29, Braşov 500036, Romania
d.bocu@unitbv.ro

**Abstract:**
The paper presents a series of considerations regarding the role of the conceptual invariants concerning the prevention of the artefacts' obsolescence, with an emphasis on the software engineering. The concept of artefact has the meaning that is defined in (Bocu & Bocu, 2016). The emphasizing of the invariants' role has the goal to allow the understanding, with the respective approximation, of this invariants' role considering the continuous qualitative progress of the human artefacts, generally speaking, but also in connection to the software systems engineering.
**Keywords:** Artefact, Structure, Behaviour.

## 1. Introduction

The literature in the field of software engineering highlights the importance of the concept of structure during the modeling process of the systemic artefacts. There are numerous papers that highlight the importance of the interface considering the capability of an artefact to cooperate. Let us consider that for a given artefact the structure (S) and the interface (I) are invariants, which are featured by certain stability (ST). In this context, let us denote by $ST_S$ the stability of S, and by $ST_I$ the stability of I. In the present paper, we intend to explain the role that the stability of these invariants plays regarding the obsolescence of the software systems. The following aspects are considered.

1. The manner by which $ST_S$ and $ST_I$, as artefacts' features, relate to the need to change the requirements of the environment towards the artefact.
2. What is the nature of an artefact's behavioural dependence to its invariants that belong to the categories S and I?

## 2. Structure, Behaviour and Change

Before approaching the two aspects by synthesizing and extending the contribution from (Bocu & Bocu, 2016), let us study the Figure 1. The *Figure 1* is tributary to the UML vocabulary, and it uses concepts like actor, diagram, and interaction. It illustrates the fact that RCA, as invariant that belongs to category *S*, and IAA, as invariant that belongs to category *I*, are components of the artefact that simultaneously realize:

- The timing of the artefact's internal degradation rhythm (RCA);
- The intelligent connection of the artefact to the changes that affect its behaviour as the time runs out (IAA).

Thus, the structure of an artefact is the trick to which the software systems developer resorts in order to systematize and optimize the relations between the artefact's components, but also to avail of the necessary respite to design new methods for the control of an artefact's structural and behavioural insufficiencies, regardless of the perspective that is considered when observing these insufficiencies. It can be stated that every structure is the synthetic expression of the immanent

intentions of an artefact to restore itself. The necessity for an artefact to restructure, without considering the founding intelligence, has two components:

- The internal dynamics of the artefact. In time, this may highlight insufficiencies, which put together mean obsolescence;
- The dynamics of the artefact's relations with the environment inside which it operates. This may also highlight insufficiencies that contribute to the obsolescence of the artefact and its operating environment.

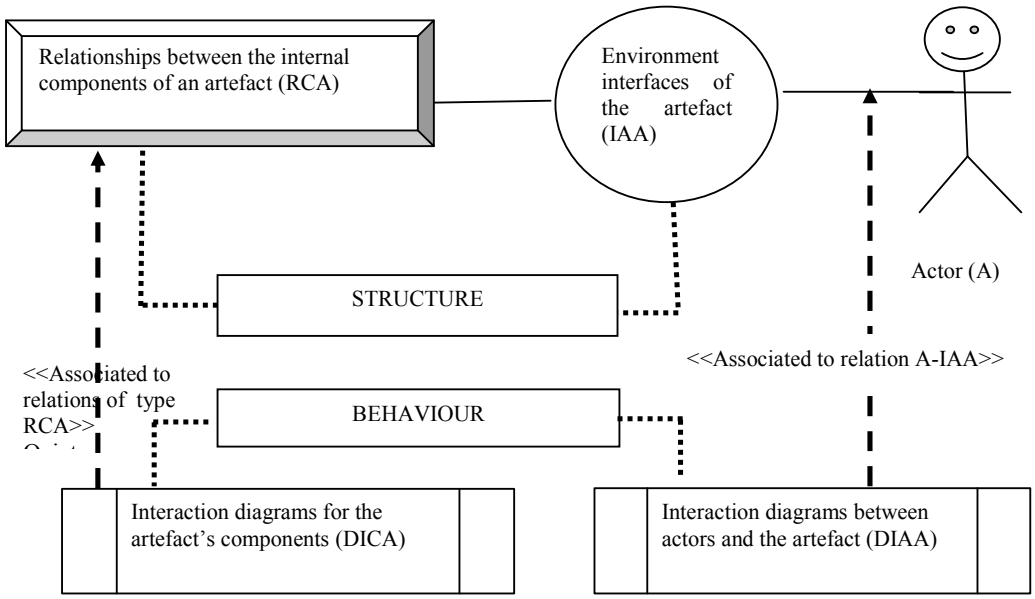


*Figure 1. Structure and behaviour during the specification of an artefact*

The reorganization of the artefacts maintains their internal dynamics to an admissible level of obsolescence, in order to preserve an evolutive trend in the world of the artefacts over a long time.

The reorganization of the artefacts' interfaces keeps the dynamics of their relations with the environment at a level of obsolescence, which is acceptable for maintaining, over a long time, an evolutive trend concerning the artefact's relations with the environment.

In order to conclude, the invariants of the type *S* and *I* may help us to specify the artefacts that are analyzed, from a historical perspective, as an adaptation model to different kinds of changes, which determine the rethinking of the requirements towards the artefacts.

Figure 2 synthetically presents the approximative evolutive model of an artefact in agreement with the numerous exigencies of the operating environment. The essential benefits of an artefact's evolution may be the following:

- a better granularity;
- an added degree of scalability;
- increased adaptability;
- excellent interoperability with other artefacts;
- very good reliability;
- etc.

**The Role of the Changes Regarding the Evolution of the Artefacts' Structure and Behaviour**

In (Bocu & Bocu, 2016) it is suggested that there are possible four essential types concerning the modification of an artefact's utility:

1. The modification of the artefact's environmental interface;
2. The modification of the interfaces that are intended for the customization of the access to the primary structural resources of the artefact;
3. The changes that are brought to the primary structure of the artefact;
4. The modification of the artefact's architecture.

These types of changes appear in the descending order of their occurring frequency. Thus, it is reasonable to expect that most of the iterations, which are due to some new requirements, consider changes at the level of the artefact's environmental interface. At the same time, the modification of the artefact's architecture is a potentially costly initiative that should generate a cautious attitude or, on the contrary, radical changes, as the architectural changes are justifiably considered.

Considering that we may be interested in explaining the changes of an artefact both as a necessity and as a component of a causal chain, questions like the following are plausible:

- Is the artefact affected by chaos?
- Would the artefact benefit from the re-specification of its objectives?
- Does the artefact require additional efficiency?
- Does the artefact have problems with its active interaction towards the environmental context?

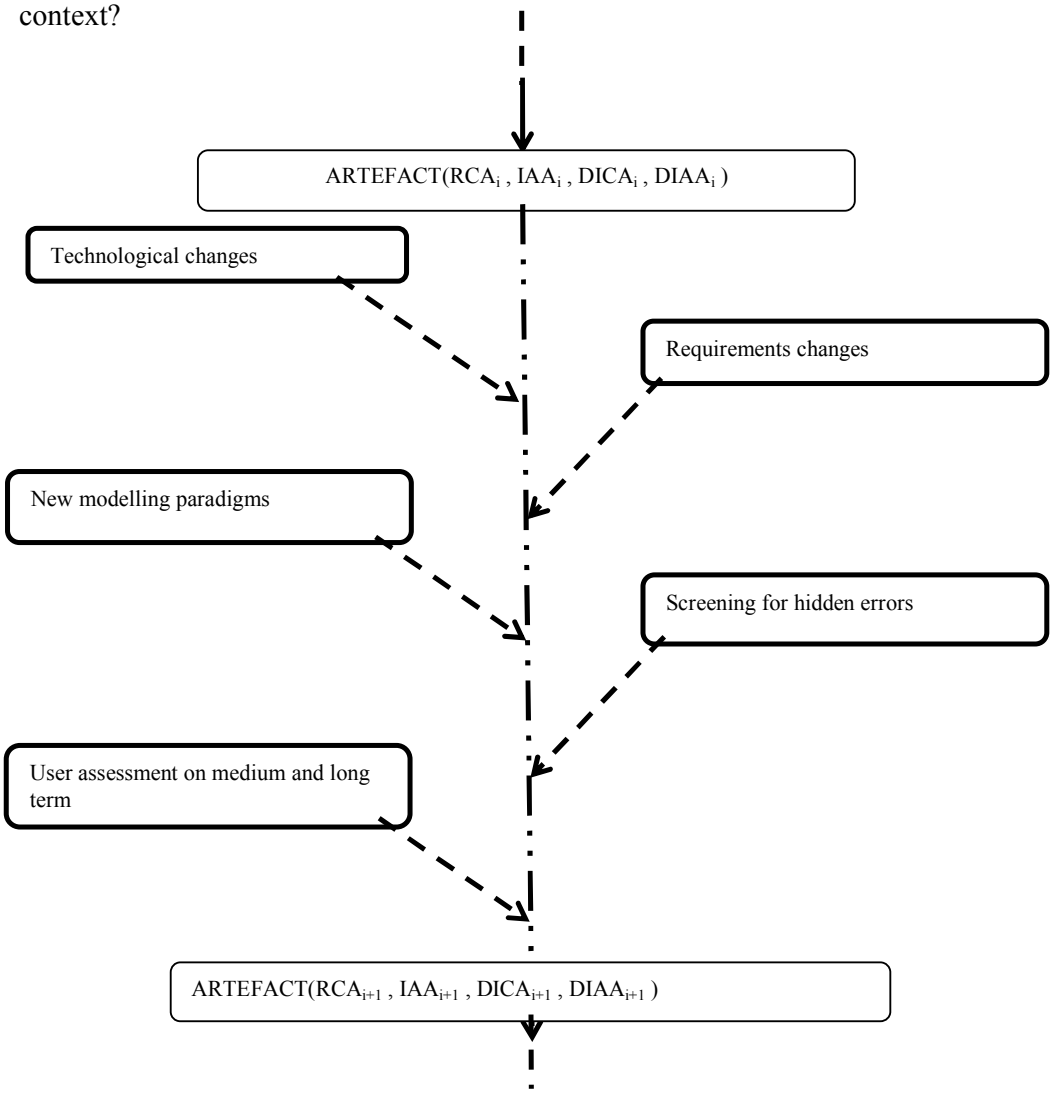


*Figure 1. The adaptation of the artefacts to changes*

Given the fact that each of the already presented scenarios infers the existence of an intelligent observer that identifies and defines the need for change, we may appreciate the change as the modality through which an artefact adapts to new requirements in the operating environment. Some of the frequent and practical causes that generate change are presented in *Figure 2*.

Considering a broader perspective, it becomes plausible the hypothesis according to which the artefact's operating environment may adapt to potential changes of the artefact.
This paper does not intend to explore this perspective.
Thus, we may state that the change is key to the artefacts' survival. Considering the perspective of survival, the change is the expression of an artefact's behaviour.

As soon as the behaviour of an artefact is analyzed, two fundamental types of changes may be observed:

- Changes of the artefacts' states as a result of their interaction with the environment. These interactions' description is dependent on the current structure of the artefacts;
- Changes to the artefacts' current structure.

The first type of changes defines the artefact's behaviour, and it is founded on a given structural invariance. This type of behaviour is captured by various types of UML interaction diagrams, as it is illustrated in Figure 1.

The second type of changes defines the artefact's behaviour, which is associated to a certain reorganization demarche.

Thus, the developers of the IT systems call for changes in order to structure and re-structure the artefacts, with the goal to maintain their highest possible quality standards.

It is important to mention the IT systems developers' capacity to specify artefacts with a structure that is, at the same time:

- Simple (easy to understand and use);
- Adaptive (to new requirements, functional or non-functional);
- Scalable.

The quality of a reasonable software system is founded on a certain modulaization strategy. The artefacts of a software system, in general, have to be structured abiding to the rules of a quality modularization.

**Considering the perspective of the modularization, the artefacts have to be gradually consistent.**

This gradual consistence of the artefacts is a direct consequence of the gradual coupling, which pertains to an artefact's components. More precisely, we can speak about the:

- Consistence and coupling that are associated to an architectural level artefact's components;
- Consistence and coupling that are associated to an artefact's components. This is an artefact at the level of a shared resources package;
- Consistence and coupling that are associated to an objectual level artefact's components;
- Consistence and coupling that are associated to an operational level artefact's components;

We may appreciate that, from an informational perspective, both the universe and the world of the software artefacts are edifices that are based on a rationality, which is essentially defined by the number of the modularization levels that structure their components. This strictly respects the gradualness of their consistency and coupling.

This entire paper has the goal to raise the details at the standards of high level and large stretch facts, in order to allow for a better understanding of the software systems modelling to be achieved.

The performant information systems favour finding unsuspected meanings in a certain detail, and also details full of meaning inside each metaphor or reasoning of the human thinking.

As a consequence, the full realization of the human being from a spiritual perspective is conditioned by his ability to see bridges between the concrete and immediate truths, and the world full of promises of the abstractions.

In the first paragraph, the following question has been asked: „What is the nature of the artefact's behavioural dependence on its structural invariants?"

The temptation to provide a dogmatic answer to this question is strong. This is caused by the preconceptions that specialists from different fields exhibit regarding the concept of structure. Thus, it seems relatively possible, honorable and comfortable to assert that the behavioural performances of an artefact are derivable from its structure.

The challenge is to interpret such an assertion. If the structural stability of the human artefacts was high, it would not be exaggerated to assert that only a remarkable structure (SR) can sustain a special behaviour (CD).

In a symbolic manner, the nature is the creator that skillfully applies this principle. Each natural creation is a remarkable example concerning the usage of the principle $SR \overset{sustains}{\Longrightarrow} CD$. Considering that we have mentioned the nature, it is immediate to ask a question such as the following: „What did the nature consider when it specified the structure of the animal called rabbit?" It is not the moment to explain why the nature invented an animal like the rabbit. Nevertheless, it is correct to remark the fact that the specification of the structure of the animal called rabbit was a demarche governed by a holistic thinking.

In paper (Bocu & Bocu, 2016), it has already been stated that the structure of an artefact can be grasped through a demarche that clearly defines the role of the artefact in the environment. Thus, the assertion $SR \overset{sustains}{\Longrightarrow} CD$ has the following correct interpretation for the creator of artefacts:

**Step 1:** The requirements of the environment towards the artefact are specified (this demarche mixes the structural decisions with the behavioural decisions; the latter ones prevail).

**Step 2**: The structure of the artefact is specified (this demarche mixes the structural decisions with the behavioural decisions; the former ones prevail).

**Step 3:** The structural and behavioural specifications are implemented, as they have been elaborated during the first two steps.

**Step 4:** The artefact is tested.

**Step 5:** Any of the first three steps may be resumed any time that this is necessary.

The strong link between the structure and behaviour is obvious. This link is specified, implemented and assessed in an iterative and incremental manner.

Thus, the idea that a remarkable structure is an absolute precondition for an excellent behaviour, which may seem tempting in the first instance, is not feasible in the real world of the software systems developers.

The borders between structure and behaviour are not as firm as those with algorithmic and dogmatic perspectives on the development process would like. This is the reason why the CASE

development tools offer a generous support regarding the capture of the systems' structural aspects, while the capture of their behaviour is still a problem, which is not entirely addressed yet.
For the moment, it can be stated that during the realization of an artefact, there are iterative (not accidental) returns towards its structure and behaviour. At the same time, it has to be systematically checked whether the process is certainly going closer to the desired solution.

## 3. Conclusions

According to *Figure 2*, but also considering the aspects that are highlighted in the first three sections of this paper, the obsolescence of the software artefacts is unavoidable. Thus, the people that are involved into the realization and maintenance of the software artefacts only have the alternative to address the obsolescence through appropriate methods. Tackling the artefacts' obsolescence is generally possible through proper changes that are effected on the artefacts themselves. The modification of an artefact creates in the short term discomfort for other artefacts. The amplification of this discomfort beyond a certain tolerance threshold is perceived as obsolescence.

Thus, it is useful to found the modelling of the software artefacts on conceptual invariants, together with clear and pragmatic usage principles for these conceptual invariants, in order to have the guarantee of a reasonable longevity of the artefacts. In practice, the invariants of a modelling language represent the basic tools to counteract the obsolescence of the software artefacts. It can be stated that the methodic structuring of the artefacts and their related modelling demarches, represent the most powerful method to combat the obsolescence of the software artefacts. Considering this approach, the artefacts' obsolescence is ameliorated. This brings benefits during the lifecyle phases that involve the controlled accumulation of the obsolescence, but also during the phases that suppose the reorganization of the framework that helps to address the obsolescence. The paper (Czibula & Czibula, 2016) suggestively illustrates the complexity of a demarche that involves the reorganization of an artefact at the moment *t*, if the reason for this is represented by the obsolescence of the conceptual apparatus that has been used to model the artefact at the moment *t-1*. The paper (Czibula & Czibula, 2016) and the real world situations also suggest the importance of the invariants oriented modelling for the realization of the support tools, which are used to reorganize the framework that is used to address the software artefacts' obsolescence.

Consequently, it is obvious that the invariants oriented modelling of the software artefacts is a substantially superior approach, when compared to the existing practices in the software systems industry, which are excessively tributary to the improvisation, random development workflow, and also the logic of the successive approximations.

## References

Bocu, D. & Bocu, R. (2016). *The Fundamentals Regarding the Usage of the Concept of Interface for the Modeling of the Software Artefact**s***, BRAIN. Broad Research in Artificial Intelligence and Neuroscience, Volume 7, Issue 1, ISSN 2067-3957 (online), ISSN 2068 - 0473 (print).

Czibula, I., G. & Czibula, G. (2016). *On Converting Software Systems to Object Oriented Arhitectures*, BRAIN. Broad Research in Artificial Intelligence and Neuroscience, Volume 1, ISSN 2067-3957 (online), ISSN 2068 - 0473 (print).

Blacha, M. (2010). *Patterns of Data Modeling*, CRC Press.

Dargan, P., A., (2005). *Open Systems and Standards for Software Product Development*, ARTECH HOUSE, INC.

Tidwell, J. (2005). *Designing Interface**s***, O'Reilly.

Rozanski, N. & Woods, E. (2011). *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives,* Addison-Wesley.

Chen, L., (2015). *Continuous Delivery: Huge Benefits, but Challenges Too*, IEEE Software, Volume 32, Issue 2.

Haav, H.-M., Ojamaa, A., Grigorenko, P., & Kotkas, V. (2015). *Ontology-Based Integration of Software Artefacts for DSL Development*, Springer International Publishing.

Afonso, L.M., de G. Cerqueira, R.F., & de Souza, C.S. (2012). *Evaluating application programming interfaces as communication artefacts*, Psychology of Programming Interest Group.

Pete, I. & Balasubraniam, D. (2015). *Handling the differential evolution of software artefacts: A framework for consistency management*, IEEE SANER Conference.