# Forecasting of Object-Oriented Faults

S Gaudan, G Motet

# Forecasting of Object-Oriented Faults

S. Gaudan[1,2], G. Motet[2]

1: Thales Avionics, 105 avenue du Général Eisenhower, 31036 Toulouse cedex 1, France
2: Université de Toulouse; INSA; LATTIS (LAboratoire Toulousain de Technologie et d'Ingénierie des Systèmes); 135 avenue de Rangueil, F-31077 Toulouse, France

## Abstract

Object-Oriented Technologies, such as Java, provide efficient features to develop software applications. In particular, they allow the development costs to be reduced. However, issues highlighted by the "Object-Oriented Technology in Aviation" group have to be handled to guarantee a high level of safety, in order to use such technologies in avionics software. In particular, the risk of Object-Oriented Technologies design faults has to be reduced to an acceptable level. These risk reduction actions must be preceded by forecasting the actual risk level of the developed application. The paper aims at presenting a method used to forecast the risk of the presence of these Object-Oriented Technologies faults in a given program. The approach proposed is based on Bayesian networks. Its principles are introduced. It is illustrated on an example of faults: the Accidental Overriding (AO). We highlight that our approach takes into account the complex relationships existing between the various object-oriented features: inheritance, distribution of the attributes and methods, etc. To conclude, we show how the obtained data can be analysed to specify design guidelines allowing an acceptable risk level to be reached.

## Keywords

Object-oriented languages, dependable software, risk management, fault assessment, guidelines.

## 1 Introduction

Avionics systems suppliers are considering Object-oriented technologies as they provide numerous advantages. The most interesting one is certainly the development cost reduction. However, as for any new technologies, its use in critical systems requires its control to be justified. The OOTiA group (Object-Oriented Technology in Aviation) highlighted that the object-oriented technologies cause two kinds of issues [8]: they introduce new types of faults at design time and new types of failures in operation. These last issues are coming from the implementation of the virtual machine often associated with these technologies. For instance, they are illustrated by the unpredictable activation of the garbage collector. We did not handle this kind of problem. We focus on the introduction of design faults which are specific to these technologies. The OOTiA handbook enumerates numerous types of characteristic design faults. For each of them, it proposes guidelines to prevent these faults. However, the origins of these guidelines are not justified and their actual impacts are not assessed. For instance, the OOTiA handbook signals that the inheritance mechanism is a hazardous feature whose use is acceptable if the inheritance depth is less than 6. However, numerous other factors must be taken into account including, for example, the number of methods declared in classes [5]. Moreover, each factor cannot be handled separately. For instance, more than 6 inheritances do not lead to faults if the inherited classes contain few methods or attributes. Our goal is to allow a justified fault risk control to be obtained to guarantee that an expected fault risk level is reached when new proposed guidelines are applied. To achieve this goal, we used a risk management process. This process is based on a well-known standard [1]. At first, the origins of the faults were studied. This step provides the various factors impacting the risk of a given fault type [4]. Then their impact on the risk evaluation must be assessed. At first, we proposed to estimate the value of each factor. This result is obtained considering an object-oriented program or model as a nested structure. The faults come from the difficulty in knowing the available pieces of information in a given class due to the inheritance phenomena [6]. The understanding factors of the fault risk are estimated by a method based on the Shannon theory [3]. However, multiple factors jointly impact the presence of faults whose effects have to be combined to obtain the global estimation of the risk. The control of these various influences also allows prevention guidelines to be proposed which impact various factors and not only one.

In section 2, we introduce a type of fault, the Accidental Overriding (AO), and its identification model presenting its sources. Section 3, summarizes the assessment method of the factors. In section 4, we introduce the global assessment model based on the Bayesian networks. Section 5, shows how this model can be analysed to define guidelines fitted to the program characteristics, that is, acting on the various factors.

# 2 Fault type identification

## 2.1 Accidental Overriding example

Accidental Overriding occurs when a designer introduces a method whose name and parameters describe a signature corresponding to an inherited method, whereas he/she would like to define a new method. It is concretised by a conflict of signatures between two methods accessible in a given class. This conflict leads to the accidental redefinition of the second method by the first one.

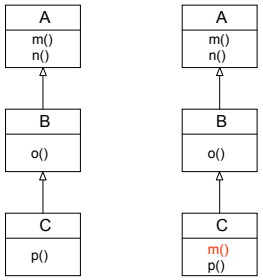



Figure 1: Accidental Overriding

The left part of the figure 1 provides an example of a program. It presents an *hazard* of accidental overriding. Indeed, in this inheritance hierarchy, if a programmer wishes for defining a method `m()` in class `C`, he/she is exposed to the risk of an accidental overriding. The method `A:m()` constitues a piece of information hazard prone. The *hazardous situation* comes from the definition of a new method in a class of an inheritance hierarchy as presented on the right part of Figure 1.

This modification is hazardous due to the presence of the other existing methods. However, it does not always lead to a fault but just to a risk of fault as

- the added method could be introduced to define a new method; in this case the fault is actual;
- the added method could be introduced to override the existing method; in this case there is no fault;

## 2.2 Identification models

In [4], we proposed a *conceptual identification model* introducing the risk features and their relationships: hazard, environment, hazardous situation, event and consequence. This model allows a fault to be setted as a risk problem. This conceptual identification model was illustrated modelling the "Accidental Overriding" issue.

We also proposed an *analytic identification model* based on the fault-tree method. From the inheritance mechanism defined in the Java language specification, we deduce the causes of the "Accidental overriding". The figure 2 presents this model. It highlights the various Object-Oriented features at the origin of the risk of fault.

# 3 Estimation concern

The *analytic identification model* is used to estimate the risk of the presence of this fault in a program.

This estimation needs two steps.

- At first, the various factors of the risk, that is the leaves of the tree, have to be estimated by means of specific metrics.
- Then, using the relationships expressed by the *analytic identification model*, the global risk has to be deduced.

Concerning the first step, numerous software metrics were proposed to estimated the various factors. However, we identified a lack concerning the consideration of the designer understanding. Indeed, the inheritance mechanism leads to the implicit availability of numerous methods and attributes inherited. To handle this issue we defined specific metrics based on the entropy theory [3].

Then, the various values of the factors have to be integrated to obtain the global value of the risk. It is an essential and difficult step symbolized by figure 3.

We established a risk estimation method based on the Bayesian Networks. The principles of the Bayesian Networks are introduced in the following section. They are illustrated on the considered example. The following section shows how this method can be used to specify guidelines allowing a safety level to be reached.

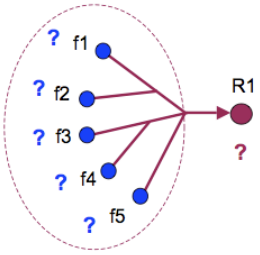



Figure 3: From factors estimation to risk estimation

# 4 Bayesian Networks

Bayesian Networks are frequently used in risk estimation. In this section we present how the Bayesian Networks can be used to estimate the risk of faults and illustrate it on the "Accidental Overriding" (AO) issue.

A Bayesian Network is defined by [7]:

1. a set of identified factors,
2. a set of possible values of each identified factor,
3. a causality structure correlating the factors (network), and
4. a valuation of the influences existing between the factors, that is, the influence of the possible values of each attribute, on the correlated attributes (directly connected in the network).

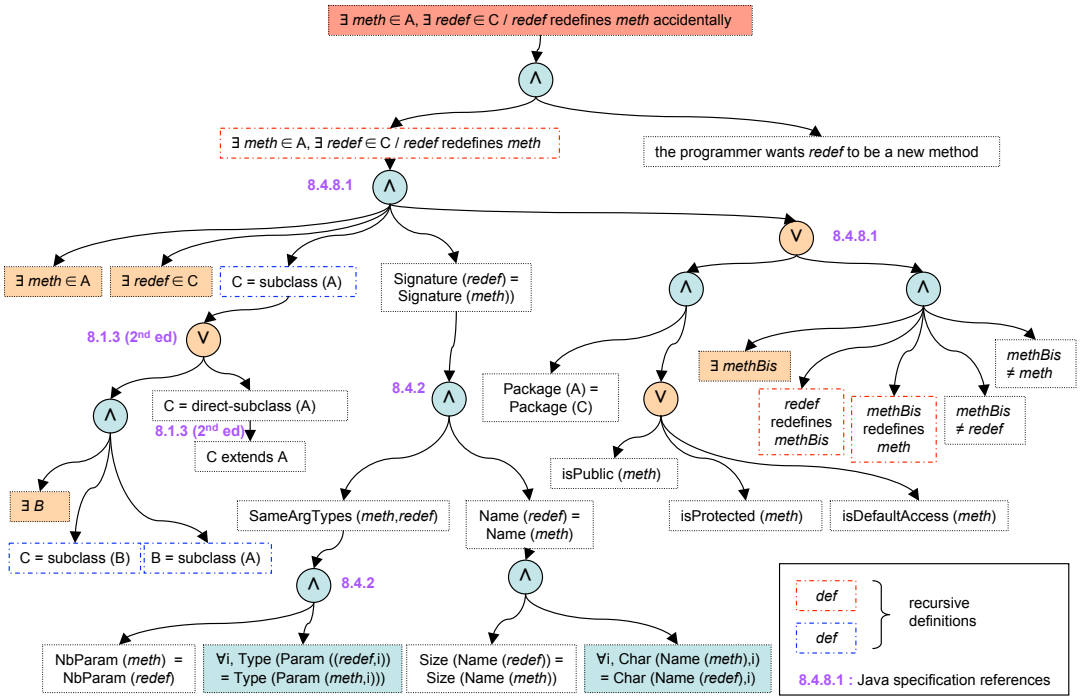



Figure 2: Analytic model to identify the risk of an Accidental Overriding

A "Goal" factor can be specified, representing a particular event whose occurrence has to be estimated from the network factors.

The following subsections present how the Bayesian networks can be built. This introduction is illustrated on the AO issue.

## 4.1 Factors of a Bayesian Network

To define the model, the factors influencing the studied risk have to be identified. This set of factors is not obligatory completed. However the most it is exhaustive, the most the estimation will be precise.

Certain factors affect several distinguished risks of faults. This will allow coupling between risks of various faults to be detected and the correlations between the factors of a Bayesian Network to be established.

We deduce the factors of a Bayesian Network from the identification of the risks of faults. For the estimation of the risk of AO faults, the identified factors are illustrated on figure 4. The proposed illustrations were produced using the tool BayesiaLab [2]. The node representing the "goal" or "target" is identified by concentric circles.

## 4.2 Domains of values

The following step consists in defining the domain of value of each factor. It contains a discrete set of values which often identify ranges when the possible values are continue or plentiful. The definition of these ranges is based on expertise or experience feedback (statistics on the programs, impact of the coding standards, etc.).

The domains of values must be defined for each factor. For instance, two values estimate the goal AO: Occur or

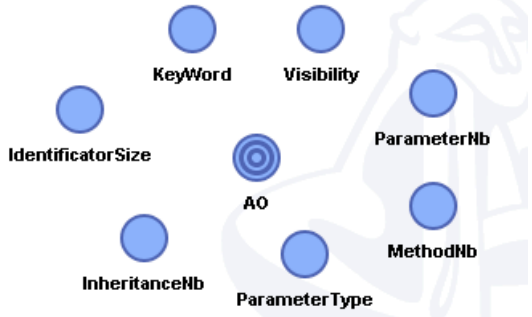



Figure 4: Risk factors of the Accidental Overriding

Absent for the accidental overriding.

For the studied example of AO, we defined the following domains:

- The domain of the factor AO is Occur or Absent (fault occurence or absence of a fault).
- The domain of the factor Identifier-Size is defined by 4 values (the numerous possible values were discritized by intervals): [1-4], [5-10], [11-25], [26+].
- The domain of the factor Number of methods (MethodNb) is defined by 4 values (discretization intervals): [0-15], [16-30], [31-55], [56+].
- The domain of the factor Number of inheritances (InheritanceNb) is défined by 4 values (discretization intervals): [0], [1-2], [3-5], [6+].
- etc.

The considered values can be precise or modified taking into account the applied coding rules. For instance, if a coding rule defines a maximum number of methods in a class, the considered estimation ranges can be

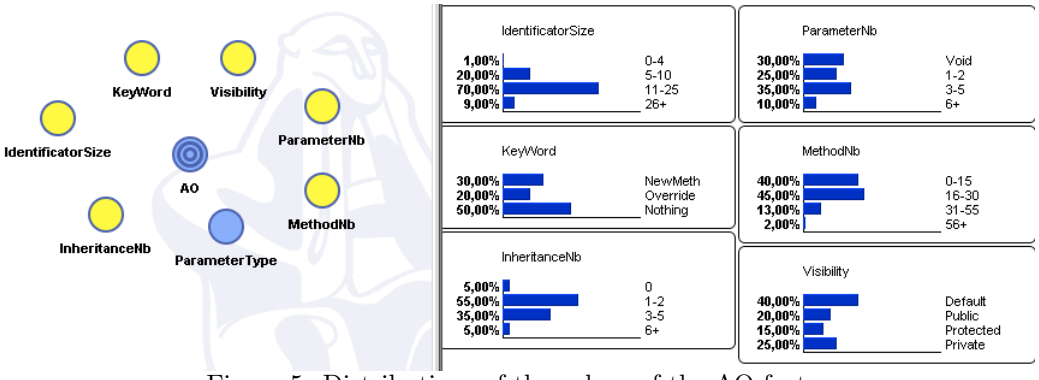



Figure 5: Distributions of the values of the AO factors

adapted. The provided example is just a proposal. In a general way, the definition of the ranges to discretize the values must be justified (experience feedbacks, expertise, best practices such as coding rules, etc.).

## 4.3 Domain and distribution of the values of the factors

The stochastic distribution of the possible values of the factors must be defined. As previously, these distributions have to be justified by experience feedbacks, best practices (such as coding rules), statistics, etc.

For instance, in a firm which requires that the identifiers should have more than 4 characters (coding rule) and taking into account the fact that most of the identifier lenghts are in the range [11-25] characters, we propose the following stochastic distribution for the factor "Identifier-Size" :

- Proba(Identifier-Size = [0-4]) = 0.01,
- Proba(Identifier-Size = [5-10]) = 0.2,
- Proba(Identifier-Size = [11-25]) = 0.7,
- Proba(Identifier-Size = [25+]) = 0.09.

In the same way, we propose stochastic distributions for each factor as illustrated on figure 5. These distributions are defined by expertize. They can be improved taking statistics on program into account as described in section 5.2.

Let us remark that the BayesiaLab tool used to produce the following figures define the stochastic distributions using percentages.

## 4.4 Causalities structure between factors

All the factors of the AO risk influence, directly or undirectly, the Accidental Overriding occurrence probability. This influence is described using arrows in the Bayesian Networks. They represent the causality between factors as illustrated by figure 6.

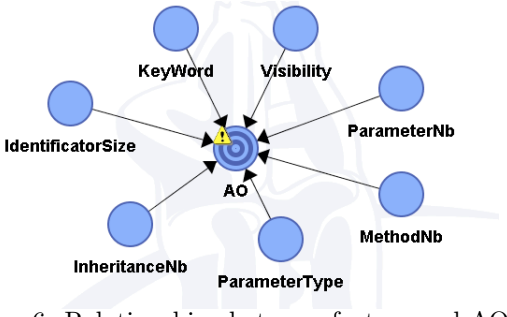
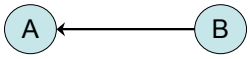



Figure 6: Relationships between factors and AO

Then, for each influence represented by an arrow, the known conditional probabilities have to be specified: proba $(A_i|B_j)$, i.e. proba $(A{=}A_i)$ knowing $B{=}B_j$, for




each values $A_i$ and $B_j$ in their value domain.

The definition of these relationships is generally based on experience feedback, statistics, etc. The conditional probabilities can take one or several values of factors as hypothesis.

- 1 hypothesis:
  proba(AO = Occur | Identifier-Size = [26+]): probability that a method cause an accidental overriding knowing that its identifier size is bigger than 25 characters,
- 2 hypothesis:
  proba(AO = Occur | Identifier-Size = [26+] $\wedge$ InheritanceNb = 0),
- 3 hypothesis:
  proba(AO = Occur | Identifier-Size = [26+] $\wedge$ InheritanceNb = [1-2] $\wedge$ MethodNb = [17-30]).

In the following, we only take 2 risk factors of AO present in the Bayesian Network into account to simplify the figures: Identifier-Size and Keyword. At first, let us consider that the influence of the factor Keyword on the AO risk is known. This knowledge is illustrated by the array provided on figure 7.

This figure shows that if a keyword exists, its knowledge determines in a deterministic way, the occurence or the absence of an Accidental Overriding (deterministic stochastic distribution: probabilities representing

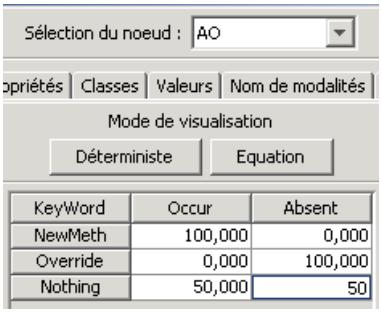



Figure 7: AO | Keyword

the certitude). When a keyword exists or when an annotation is used to express the designer intention to define a new method, each method redefinition is automatically accidental as the designer intention expressed by the keyword is to create a new method. On the contrary, when no keyword is present, not any knowledge allows us to specify the estimation of the AO values.

For instance, let us consider that our knowledge concerns the size of the identifier of a method and the potential keyword. Figure 8 proposes a definition of the influences between the factors of the AO risk.

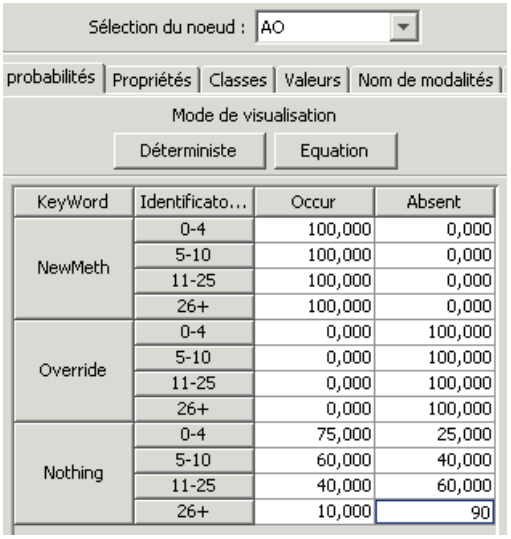



Figure 8: AO | Key-word $\wedge$ Ident-Size

This example shows that the conditional properties have to be defined for each possible combination of the considered factors (here, $3 \times 4 = 12$ distinguished). The deterministic knowledge defined on figure 7 is preserved when a known keyword exists. Any additional piece of information gained does not modify the risk estimation which is null (0%) if the keyword is "Override", and which is maximum (100%) if the keyword is "NewMeth". On the other hand, without further keyword, knowledge on the size of the identifier of the method precises the estimation of the AO risk associated with this method.

Indeed, when a method possesses an identifier having less than 5 characters (high risk of name conflict) and a keyword "Override", the AO risk is null as illustrated on figure 9. The green bars representing probabilities characterize factors whose values are fixed.

In spite of an identifier whose size is very small, the AO risk is null for a method preceeded by the keyword "Override" as it expressed the intent of the redefinition. This determines with certainty the absence of unvolontary redefinition.

In the next figures, the nodes colored in yellow (left part) represent the factors on which we observe the distributions of values (included in the right part). The blue nodes are not observed. The observation of the factor "goal" is distinguished by a red background frame (factor on the right bottom part of figure 9). Finally, the green nodes represent factors whose value was settled. They represent the knowledge available on an element of the code or hypothesis which are made.

In our example, the knowledge of the identifier size has no impact on the estimation of the risk of Accidental Overriding when the factor "Keyword" is "Override" or "NewMeth". Figure 10 shows that the estimation of AO is not modified by the absence of knowledge on the factor "Keyword" (distribution by default of the possible values), relatively to the knowledge illustrated on figure 9 in which the identifier size is fixed between 0 and 4.

On the contrary, we observe that when there is no knowledge on the factor "Keyword" of a method, the estimation of the AO risk is precised by the knowledge of the small size of the identifier. Figure 11 illustrates this remark. Indeed, when no information exists on the keyword, the only knowledge on the identifier size (0-4) does not allow a conclusive estimation to be obtained (67.5% Occur, 32.5% Absence).

We infer that the various values observable for each factors are more or less significant for risk estimation.

All the knowledge about the values of factors and about their influences must be used to express the arrays defining the associated conditional probabilities.

When the knowledge is formalised in the whole in the model, the Bayesian network design is completed. Then this network can be handled by various proposed analysis.

The quality of the established network depends on the quality of the information available to define the network. In particular, this concerns the nodes connection knowledge. Indeed, the more the network is complete and the more the factors are correlated, the more the analysis deduced will be meaningfull.

# 5 Bayesian networks applied to OOTiA faults

## 5.1 Bayesian networks applied to Object-Oriented Technologies

The networks for estimating each risk associated with the methods or the attributes were separately defined. It

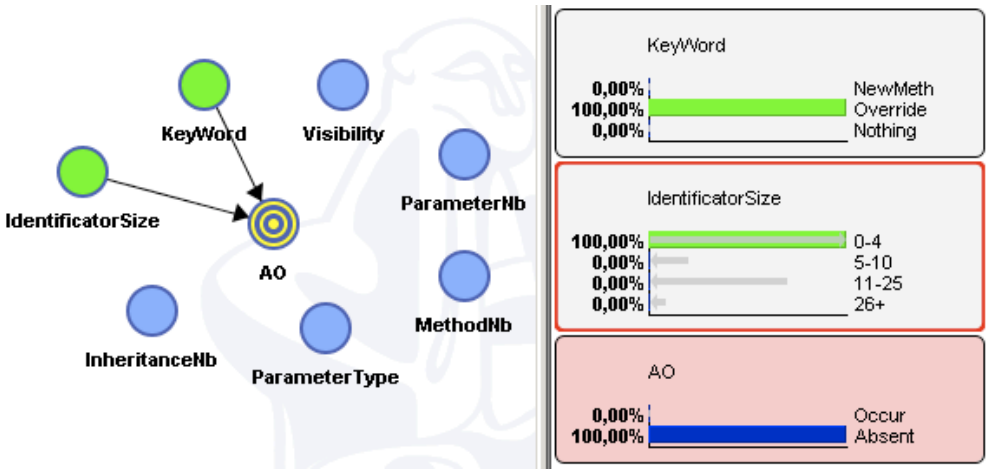



Figure 9: RI | Keyword = Override ∧ Ident-Size = 0-4

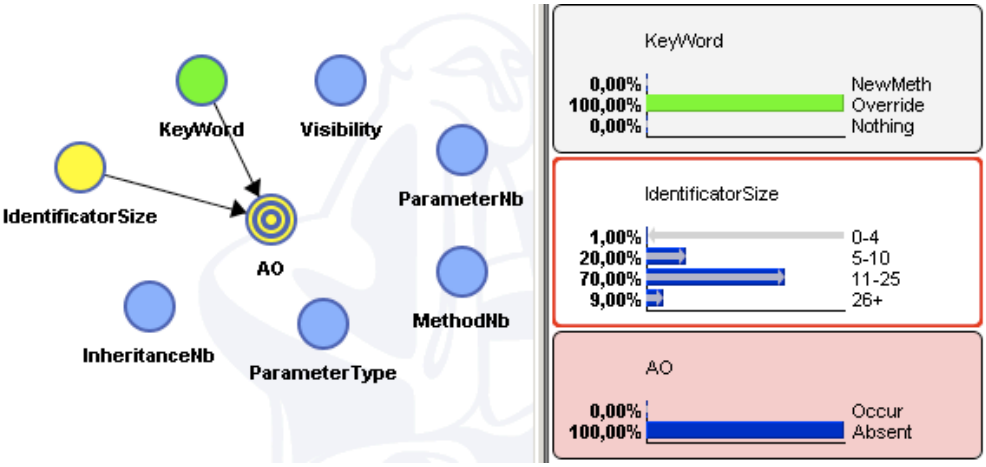



Figure 10: RI | Keyword = Override

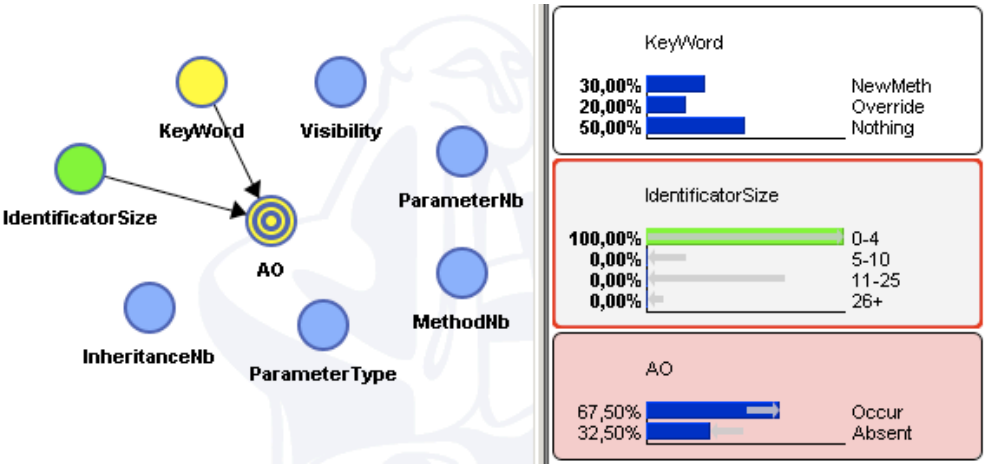



Figure 11: RI | Taille-Ident = 0-4

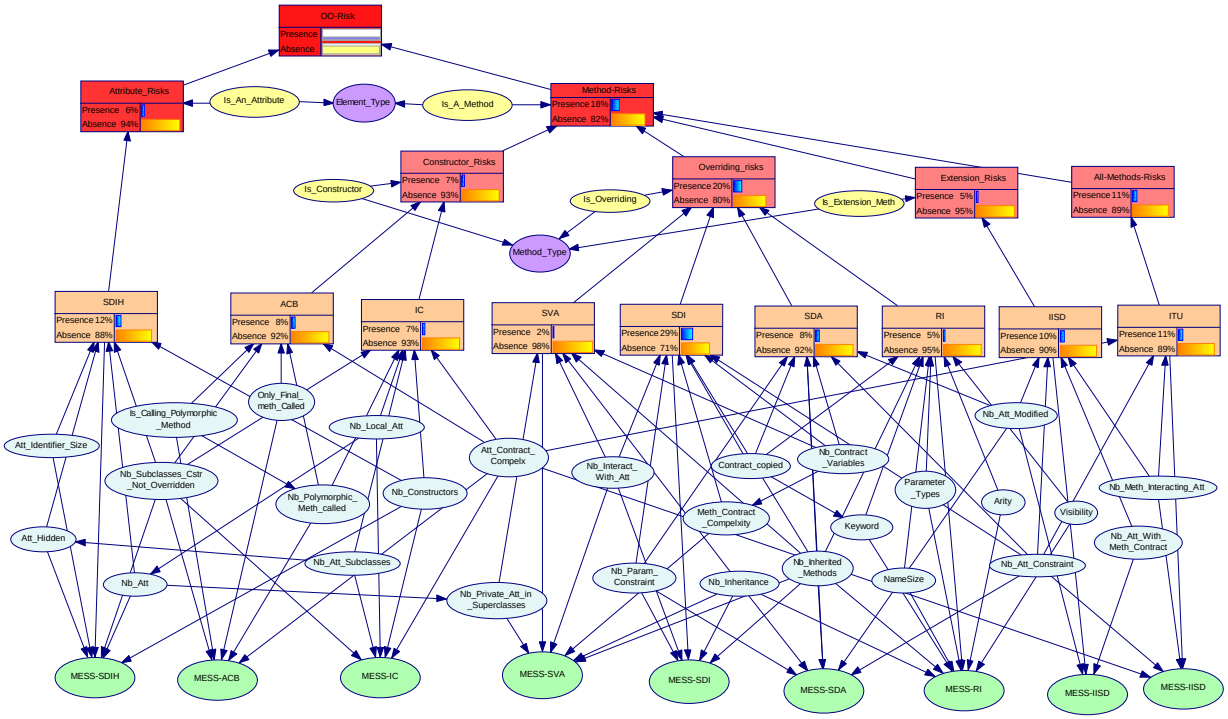



Figure 12: Bayesian network to estimate OOT fault risks

is therefore possible to define a global Bayesian network to estimate the risks of the faults due to an element of the object-oriented code (method, attribut).

We introduce a factor "Element_Type" whose value "Method" or "Attribute" determines the estimation efforts in accordance with the element type.

Figure 12 proposes a Bayesian network to estimate the risks associated with the elements of an object-oriented code for the types of faults cited in the OOTiA document.

Of course, due to its definition mode, it is quite easy to add to the network the estimation of any other useful type of fault.

For instance, we notice on figure 12 that the risks SDIH[1] and IC[2], which concern elements of different types (SDIH concerns the attributs and IC the constructors), have a common factor (the number of classes in which an accessible constructor is not redefined). This means that a correlation exists between the estimations of these two types of faults.

The factors corresponding to the various metrics using information theory [3] (green nodes) present a high cost as their calculation requires a complex static analysis of the code. On the contrary, the yellow nodes represent the type of the analyzed element. This information is easily obtained as the method visibility or the number of local attributes. However, the factors requiring a run in the inheritance chain lead to a non trivial static analysis of the code. Therefore, their cost is more important.

[1]State Definition Inconsistency due to a state variable Hiding
[2]Incomplete Construction

## 5.2 Definition of guidelines

When the Bayesian network associated with the fault risk estimation is defined, various exploitations are available, providing several interesting results. It is possible to use the Bayesian network to process

- generic estimations (independent on the programs), and to process
- particular estimations (on the element of a specific program).

The various kinds of exploitations of the networks are enumerated here after and will be detailed later on.

- Specific estimation of a risk on a given element of a given code (attribut, method);
- Generic estimations:
  - definition of coding rules to reach a specified risk level,
  - definition of coding rules to reach a specified risk level, considering a set of fixed factors;
- Inference: definition of static analysis of code optimized on the more meaningfull factor to obtain a precise estimation of a fault risk;
- Learning techniques and graphical analysis.

### 5.2.1 Specific estimation of a risk of a given element (attribute, method)

At first, we present the most simple type of analysis: the use of the Bayesian network to estimate the risk of a specific fault type for a given element of a code.

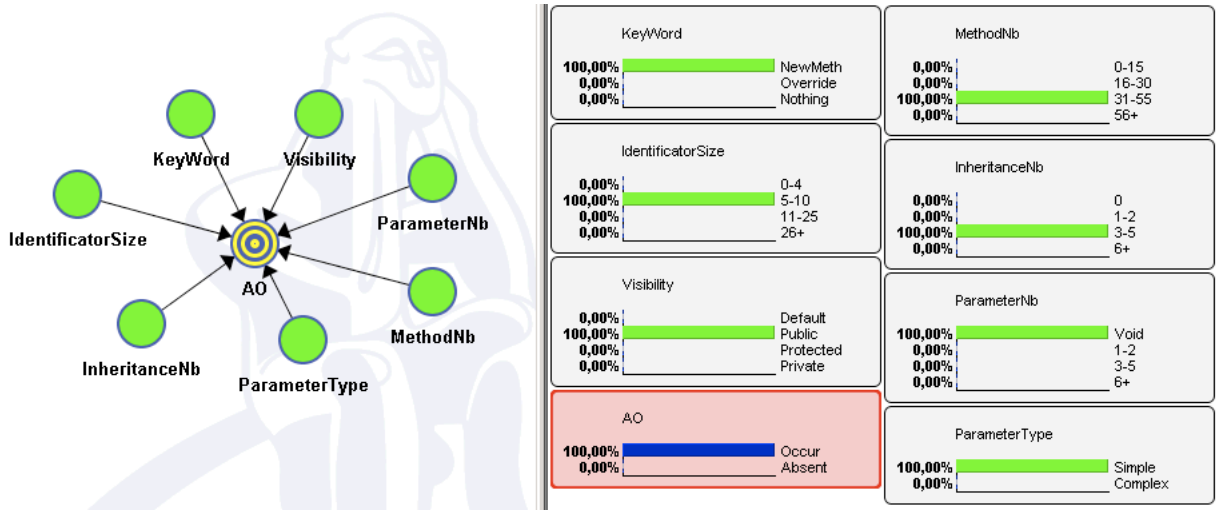



Figure 13: Factors of risk of AO fixed for a specific estimation

To do it, information about the element whose risk has to be estimated must be gathered. Then these pieces of information must be introduced in the Bayesian network fixing the values of the associated factors.

Thus, we deduce the risk occurence likelihood of the studied fault on the considered element. Sometimes, the values associated with some of the factors are not available and so are not considered in the estimation calculus (treatment of the uncertainty). Each unfixed factor also influences the global estimation. Each possible value of a factor has an impact taken into account proportionately to its likelihood a priori. For that purpose, the factor values distributions defined in the Bayesian Network are used.

Figure 13 provides an example of the risk estimation on an Accidental Overriding. It assumes that a method is labelled by the keyword "NewMeth", has an identifier size between 5 et 10 characters, possesses a public visibility and has no parameters (factor ParameterType is estimated as Simple), redefines a method inherited from 3, 4 or 5 inheritance levels, and is in a class which provides from 31 to 55 defined methods.

The estimation of AO obtained in such a configuration is a deterministic estimation of the occurence of an accidental overriding (100% Occur of AO) due to the presence of the keyword "NewMeth" (100% NewMeth).

### 5.2.2 Definition of coding rules from a given risk level

One of the main goal of our work is to propose a set of coding rules for object-oriented technologies that provide guarantees that a given risk level is not overtaken. This means that the actual risk is lower than the acceptable risk threshold. For example, assume that the maximum acceptable probability of the occurence of an AO fault is $10^{-4}$ for each method. Each risk whose estimation value is higher than $10^{-4}$ is considered as unacceptable.

Then, to reach an acceptable level, it is necessary to define coding rules on the language features impacting the pieces of information used to estimate the risk.

To achieve this objective, a maximum value for the factor "Occur" of the goal AO is fixed. Figure 14 shows how this value is specified in the red frame which define the likelihood associated with the various values of AO (Occur and Absence).

Thanks to the property of reversibility of the Bayesian networks, a combination of likelihoods of the factors values can be deduced allowing the fixed risk threshold to be achieved.

Figure 15 presents on our example, the impact of the fixed value for the factor "goal" on the factors. For instance, it shows that the likelihood of presence of the keyword "NewMeth" and that the likelihood associated with a small size of identifiers strongly decreased. Therefore, a coding rule excluding identifiers whose size is less than 10 characters for a redefining method should maintain the risk estimation at an acceptable level. To check its efficiency, we fix the forbidden values to a null likelihood. Then we observe the change of the likelihoods of AO values depending on the various possible configurations of the other factors.

In figure 14, the probability of the presence of a keyword "NewMeth" is 30%. This value is modified when the threshold of presence of AO is fixed at 0.01%. Indeed, we observe on figure 15 that the probability of the keyword "NewMeth" became close to null (0,01%). This kind of the Bayesian network analysis suggests to exclude this type of values.

For such an exploitation of the Bayesian networks, an acceptable threshold has to be defined. Then, processing the network, the constraints on the various factors of the program are deduced.

The risk acceptability threshold is specified depending on the considered criticality level (DAL A, B, etc.). According to the types of applications, the risk acceptability threshold evolves.

The dependency between DAL and design constraints

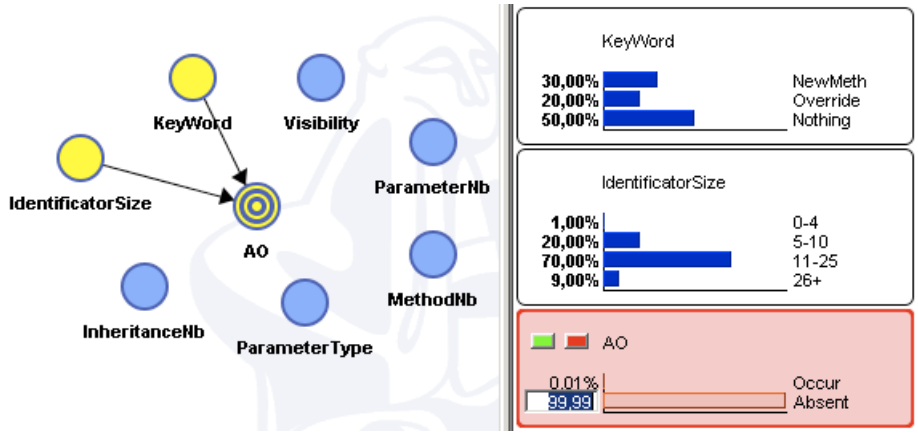



Figure 14: Definition of an acceptable risk level

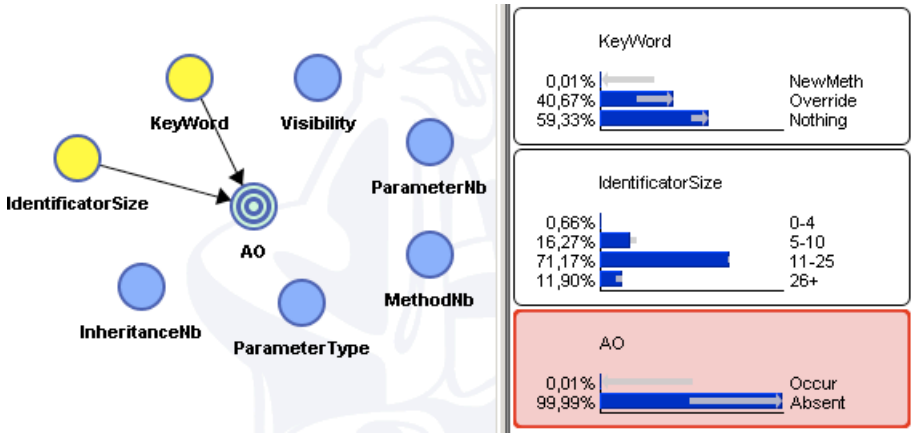



Figure 15: Assignement of the distributions of the values of the factors

is already considered for the process. For instance, for testing, a structural coverage of type MC/DC (Modified Condition / Decision Coverage) is required for level-A applications. For level-B applications, only the C/DC (Condition / Decision Coverage) is required. Thank to our proposals, the criticality level also impacts the way of the technology is used.

The Bayesian networks allow to deduce coding rules specific for each risk acceptability threshold, that is, criticality level, fixing a value to the factor "goal".

However, the definition of a given risk acceptability threshold for each criticality level must be proposed or validated by the certification authorities as they previously specified the required verification techniques for each criticality level.

### 5.2.3 Definition of coding rules relatively to a risk threshold and a set of fixed factors

At first, we justify the interest of the Bayesian networks considering an example of coding rule proposed by OOTiA [8]. This coding rule excludes the use of inheritance chain whose lenght is greater than 5 levels. This requirement does not depend on the criticality level. For this coding rule, the factor "InheritanceNb" is constrained independently to the other risk factors. Moreover the constraint is restrictive.

In such a case, we want to propose more adaptable constraints concerning the inheritance depth. In return, it is necessary to propose additional constraints on other factors to preserve the risk acceptability. So the designer has more freedom degrees to take into account the design constraints as well as the safety requirements.

The Bayesian networks applied to fault risks help in defining such coding rules. Fixing a risk acceptability threshold (depending on the considered criticality level) and other risk factors (for instance, more than 5 levels of inheritance), the Bayesian networks deduce the consequences on the other factors of the risk of fault. Thus, one or several constraints can be relaxed which are compensated by stonger restrictions on other factors. So, the same safety level is obtained with adjustable rules than with generic and restrictive coding rules.

Consider our simple example used to estimate the risk of Accidental Overriding taking 2 factors into account. At first, we fix the acceptability threshold of the presence of this type of fault at 0.01%. This means that an occurence probability of such a fault lower than $10^{-4}$ is acceptable for applications whose DAL is C (for example). Then we observe the consequences on the other factors (one factor in this simplified example). Two distinguished infuences are illustrated on figure 16.

The first proposes to measure the impact of the presence of a method whose identifier is very short in order to reach the acceptable level. It shows that the favoured value of the factor keyword is "Override". On the con-

trary, in the second example provided on figure 16, the important size of the identifier largely compensates the absence of keyword.

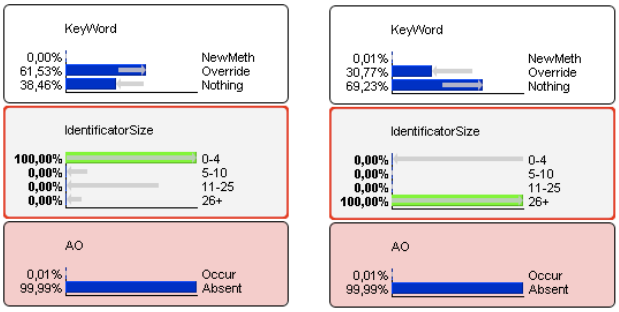



Figure 16: Taille-Identifier influences

The considered example takes only 2 factors into account. So the deduced conclusions could be obtained intuitively. However, they are justified by values obtained by calculus. Moreover, in real situation, that is, when all the factors are taken into account such as at figure 12, the intuition is inefficient.

Realistic Bayesian networks possess a large number of factors. A value of one attribute can be fixed to relax conventional constraints. Then, the Bayesian networks running provides combinations of other factors allowing the acceptable level of risk to be achieved. The values of several factors can be fixed. Then the coding rules are such as "If the inheritance chain depth is greater than 5, the presence of keywords is obligatory if methods are redefined, and the identifier size must be greater than 10".

Consider again the Accidental Overriding. It is important to propose other coding rules than the obligation to use keywords since numerous object-oriented languages do not offer these features.

# 6 Conclusion

To conclude, let us first underline that our approach proposed to define design constraints is generic. Thus it should be used to define good practices to manage other types of software risks (memory risks, time execution risks, etc.), or to manage system risk, at the sytsem level.

Then, we want to mention one more time that the major contribution by applying our risk management methodology is to provide means to mesure the guarantees on the safety that are provided, when using the defined guidelines.

Finally, as the design constraints represent a significant part of the developpers environment, and thus influence significantly the development costs, they should be defined by a strategic analysis, taking into account their efficientness on safety, but also their cost.

# References

[1] ISO/IEC Guide 73. Risk Management, Vocabulary, Guidelines for use in standards. International Organization for Standardization, 2002.

[2] BAYESIA. Bayesialab. BAYESIA S.A., Laval, France, www.bayesia.com.

[3] S. Gaudan, G. Motet, and G. Auriol. A new structural complexity metrics applied to object-oriented design reliability assessment. In *proceedings of the 18th IEEE International Symposium on Software Reliability Engineering (ISSRE 2007), Industrial Track*, Tröllhattan, Sweden, November 2007. IEEE.

[4] S. Gaudan, G. Motet, E. Jenn, and S. Leriche. Identification model of the object-oriented technology's risks for an avionics certification. In *proceedings of the 3rd European Congress Embedded Real-Time Software (ERTS06)*, ISBN 2-91-2328-27-6, Toulouse, France, 2006. SEE.

[5] W. Li. Another metric suite for object-oriented programming. *Journal of Systems and Software*, 44(2):155–162, 1998. ISSN 0164-1212, Elsevier Science Inc., New york, NY, USA.

[6] G. Motet and S. Gaudan. Assessment of the risks for using object-oriented technologies in critical software. In *proceedings of the 6th Workshop on Critical Software Tokyo*, Japan, 2006. JAXA publisher.

[7] P. Naïm, P-H. Wuillemin, P. Leray, O. Pourret, and A. Becker. *Réseaux bayésiens*. Eyrolles, 2nd edition, 2004. ISBN 2-212-11137-1.

[8] OOTiA. Handbook for Object-Oriented Technology in Aviation, Octobre 2004. http://shemesh.larc.nasa.gov/foot/.