1993.

Qualitative Reasoning and Software Quality

L. Travé-Massuyès**, A. Missier**, Spyros Xanthakis *

* OPL
Futuropolis
6, rue Maryse-Hislsz
31502 Toulouse Cedex
FRANCE
Tel. (33) 61 80 91 40
Fax. (33) 61 80 12 81

** LAAS-CNRS
7, avenue du Colonel-Roche
31077 Toulouse Cedex
FRANCE
Tel. (33) 61 33 63 02
Fax. (33) 61 33 64 55
Email. louise@laas.fr

Abstract: One of the main problems in qualimetric models is the extreme quantification of the interpretation process. Quality engineer is often overwhelmed by raw numeric data and can hardly foresee the major impacts of his decisions. It seems that decision making in quality engineering mainly consists in comparing relative tendencies rather than estimating exact values. A new field of Artificial Intelligence, called Qualitative Reasoning (based on deep knowledge models and capable of representing qualitative features of the phenomena) can be considered as a novel alternative which enables one to translate quantitative models into new models based on qualitative calculus (qualitative algebras, orders of magnitude operators, etc.). In this paper we expose the different limitations of quantitative qualimetric models and we propose a complementary qualitative approach, followed by an illustrative example.

1. Qualimetric models: main features and limitations.

The quantification of the software development process has always been one of the main concern of industrialists in order to control and increase software productivity and reliability. The seminal works of J. MacCall [14] and B. Boehm [15] appear as two of the most sophisticated and detailed qualimetric models which are nowadays largely adopted by the Q&A community.

The general principle of a qualimetric model is to identify and analyze characteristics concerning the development process (i.e. quality of the specification documents) as well as the software product itself (size and complexity of source code, volume of data processed, etc.). In the MacCall model (implemented in [2]), these characteristics are organized in an inverted tree-like structure. The root of the tree symbolizes the overall software quality and is connected to the most abstract quality concepts (i.e. *maintainability*), called *factors*. Factors depend to a set of *criteria* (i.e. *traceability*) which in their turn depend on *metrics* (i.e. *number of comments* in the source code). Metrics form the leaves of the tree and can be directly estimated manually or automatically (by means of static analysis tools).

One can compare metrics to captors which supply the relevant information concerning the software components under development. The tree-like structure can be compared to a cause-effect diagram (causality graph). The main goal of a qualimetric model is to interpret correctly the various interconnections among metrics, criteria and quality factors. In such a way a rough quantitative information, like the *number of comments*, processed and propagated in a bottom-up way, determines a criterion like *readability* which in turn influences the factor *maintainability*.

Since the initial theoretic foundations, one of the first tasks of the software quality community has been to identify and quantify more precisely the causal dependencies governing software characteristics [1], [3], [4]. This quantification can be based on graph theoretic principles (i.e. the factor *testability* depends on the *cyclomatic number* of the control graph [16]), information theory (i.e. the software physics approach of M. Halstead [17]), and, more often, on statistical evaluations. In the great majority of these numeric models we find a relationship (often obtained by linear regression methods) where an abstract quality concept Q, is expressed with a weighted sum of n (often normalized) measures μ_i :

$$Q = \sum_{i=1,n} w_i \mu_i$$

For instance, in de Millo&Lipton [18] we find an estimation of the effort required for program development (*PD*):

[1] : PD = 2.7a+121b+26c 12d+22e-497

where:

a = estimated number of statements/1000,
b = estimated complexity (1-5)
c = number of external documents
d = number of internal documents
e = size (in words) of the database.

In [2] we find an estimation of a criterion like the *modularity* of final design (*MD*):

[2] : MD = 0.25.(HCMPX + SCMPX + MPM + MLPSM)

where:

HCMPX : Hierarchical complexity of the call graph
SCMPX : Structural Complexity of the call graph
MPM : Mean Number of Paths per Module
MLPSM : Mean Number of Pseudo-Code Lines per Module.

These functions are not always linear. For instance the well-known textual complexity estimation (*C*) is given in [17]:

$$[3]: C = \frac{n_1.N_2.(N_1 + N_2)\ log_2(n_1 + n_2)}{36.n_2}$$

where:

n_1 : unique operators in the program
n_2 : unique operands in the program
N_1 : operator occurrences in the program
N_2 : operand occurrences in the program

We expose these different equations in order to stress the extreme quantification of the qualimetric interpretation process. Even if the different qualimetric formulae have solid mathematical foundations, the quality engineer is very often overwhelmed by raw numeric data (statistics, Kiviat diagrams, etc.) and can hardly foresee the major impacts of its managerial decisions. This observation is confirmed by our experience in software engineering consulting during the last ten years of activity in Europe and mainly in France (Thomson CSF, Airbus-Industrie, British Telecom, Telefonica, CEA, etc.). Metric's sizing and tuning are assured heuristically and are often based on subjective and incomplete information.

As stated in [19] the problem with many formulae is that they contain quantities of quite different dimensions. It is meaningless to add a textual complexity and structural complexity because these complexities do not measure the same concept. The only thing we can say is, for instance, that the testability of the software product depends on these two complexities. With a same reasoning we can know that:

- In formula [1] the program development effort increases when the number of external documents increases,

- In formula [2] the modularity decreases when the structural complexity of the call graph decreases,

- In formula [3] the structural complexity of the call graph increases when the number of unique operators increases but, when the number of unique operands in the program is increasing the overall tendency cannot be known exactly.

By adopting a flat numeric formula we amalgamate numbers and loose the meaning and the salient features. In the other hand, how can we be sure on the weights we adopted inside a formula? What does it mean to have a maintainability coefficient equal to 0.78? Is this number twice better than 0.39?

Finally, all qualimetric models are static. They rarely take into account in which development phase a specific quality metric has been obtained. We understand that the relative importance of a metric is not constant during the software development. The person in charge of the quality assurance must know this information in order to schedule palliative actions. Unfortunately, trying to adopt some corrective actions during the development process (e.g. increase code reviews, or increase the charge of software testing for a module) may have the opposite effects. Software metrics are strongly correlated and have their own dynamics. A qualimetric formula cannot express these dynamic relationships which are sometimes due to intricate feedbacks. It is in fact impossible to avoid, for decision making purposes, the combinatorial explosion if we wish to capture these dynamics by means of a precise numeric qualimetric model. This is true for the majority of engineering fields.

Practically, it seems, by our experience, that decision making on software quality can be compared to a high level expertise where reasoning mainly consists in comparing *relative* tendencies (e.g. if this metric is high then this quality factor will decrease) rather than estimating *exact* values (e.g. the testability ratio equals 0.78).

We do not claim here that classic qualimetric models are useless. We simply consider that they have equally to be coupled with reasoning and explanatory interpretation mechanisms which are closer to human practical reasoning. These considerations naturally lead to consider qualitative reasoning as a novel and complementary alternative of great interest.

The next section comprises a presentation of this new area of Artificial Intelligence. Follow an illustration of the concept of qualitative reasoning applied to software quality evaluation and tracking, as well as to decision making.

2. Qualitative Reasoning

Qualitative Reasoning (QR) is a rather new research area of Artificial Intelligence which aims at providing concepts and methods for reasoning qualitatively on the basis of highly abstracted deep knowledge models, hence reproducing and extending human mental qualitative reasoning schemes [7]. The key idea is to reason from a model which only captures the qualitative features of the phenomena at hand. This is performed by describing the phenomena through an appropriate set of variables (software metrics and quality factors in the case of quality software analysis) and a set of constraints

expressing the relationships between these variables.

A central issue in QR is *prediction* which allows one to estimate the states (generally values and tendencies) of the subset of endogenous variables given the states of the exogenous variables [13]. This is performed by propagating the exogenous variable states through out the constraints constituting the qualitative model of the system. If the model has been devised so as to capture the system dynamics, prediction can be added a temporal dimension since it provides the temporal evolution of all qualimetric variables in the future, starting from an initial global state. The type of algorithm allowing one to do so is referred to as a *qualitative simulation* algorithm.

The prediction algorithms are closely related to the representation formalism used for stating the model, which is itself closely dependent on the domain of interest. For example, it is not surprising that the electronic circuits domain has lead to a formalism based on relative order of magnitude relations [10, 11], which are extensively used by the electronic engineer when analyzing a circuit. On the other hand, several continuous domains fall into a description in terms of algebraic and differential equations, which explain the emphasis put on Qualitative Differential Equations (QDE) models [20] by the QR community.

The representation primitives of the QDE formalism include arithmetic and functional operators as well as a derivation operator. These operators are qualitative in the sense that they permit the representation of weak relationships like monotonic increasing or decreasing dependencies. These primitives appear to be very suitable to represent the type of dependencies existing among software metrics and quality factors. In consequence, the following of the paper focuses on the QDE formalism and presents our own QDE-based prediction algorithm SQUALE [21].

When the available knowledge on which to build a model is incomplete and inaccurate, performing qualitative prediction presents several advantages compared to numerical prediction. First, it allows to represent inaccuracy in an explicit way by considering qualitative values and qualitative relations rather than to require numerically grounded assumptions which may be difficult to work out. By doing so, the qualitative prediction output provides the set of *all possible estimations* (when performing qualitative simulation, these estimations are temporal evolution) which are consistent with the lack of precision of the model, hence summarizing an infinity of numerical predictions. On the other hand, qualitative estimations only outline the significant distinctions which are necessary to reason at a decision making level [6].

3. Qualitative simulation

3.1 Principles of SQUALE

This work is based on the formalism defined by B. Kuipers in his paper *Qualitative Simulation* [20] with new improvements presented in [21, 22]. A qualitative model of a system is given by constraints between physical parameters. Constraints may be mathematical relations *add, mult, d/dt, const,* but also two more qualitative constraints m^+ and m^-, which precise that there exist a monotonic increasing or decreasing relation between x and y. These last constraints allow to model whole classes of real systems, which are not necessarily equivalent to their real solutions. But the set of predicted behaviors provided by QSIM, will contain all the possible real behaviors.

The qualitative value of a variable is given with regard to the quantity space which is a totally ordered set of landmarks. These landmarks are purely symbolic. A qualitative state is actually the pair position and direction of change *(qval, qdir)*.

The algorithm distinguishes two types of transitions:

• *P-transitions* when moving from a date to an interval of time,
• *I-transitions* when moving from an interval of time to a date.

As variables are supposed to be reasonable functions (i.e. piece wise continuously differentiable which is the case of qualimetric models), possibilities of state transitions are limited by the intermediate value theorem. Possible transitions are provided under the form of tables [20].

If, at a given time point, several variables, linked each other by a constraint, are all positioned on landmarks, the set of these landmarks is called a tuple of "corresponding values" for the considered constraint.

The process of simulation determines first, for each variable, the changes of possible qualitative value and then filters these changes, by applying rules deduced from the constraints. If, at the end, the qualitative simulation produces several consistent changes of qualitative states, then the current state has several possible successor states and the simulation produces a branching.

Moreover, SQUALE uses considerations on global interpretations to discard again some inconsistent states.

Basic global filters are:

• *No-change* : discards the successor state if it is identical to the current one.
• *Cycle* : Detects cyclic behaviors.
• *Quiescence and saturation* : if one variable reaches infinity, or has all its derivatives equal to zero, then stop the simulation.

Unfortunately, the basic algorithm presents some fundamental limitations. The main problem we are faced with is the combinatorial explosion of the number of predicted behaviors. Even in some simple cases, the tree of possible behaviors may be unexploitable. This explosion increases with the system complexity. Obtaining new methods of modeling and filtering to discard some inconsistent behaviors is the basis axis of current research in this area.

3.2. Example

The interest of qualitative simulation is to provide a tool for assisting modeling or decision. Indeed, before obtaining a sophisticated qualimetric model, it could be relevant to test some hypothesis about relations between variables, first from a qualitative point of view. This avoid to specify parameters which signification is not obvious. In decision tasks, the aim is to provide an envisionment of all the possible consequences of some operating strategies. As an interactive tool, the qualitative simulator should not replace humans in making fundamental choices about a system. But its efficiency is to propagate automatically, quickly an reliably, all the influence relations between variables, and to provide thus a qualitative view of the consequences of a given particular choice.

Most of qualimetric models may be transcribed using *add*, m^+ and m^- constraints. Dynamic relations, expressed with the derivative constraint d/dt, are rarely encountered. Thus, most of relations are static and hence a qualitative qualimetric model comes generally down to a set of confluences like in [23]. The confluences are relations between the tendencies of the variables which express their mutual influence : for example in our model we assume that *efficiency* is positively inflenced by *coupling* and negatively by *fault tolerance, reusability, language* and *generality*. This assumption will be modeled by a qualitative constraint :

efficiency ≈ *coupling - fault tolerance - reusability - language - generality*

The qualitative values have been taken in the set {0,+,-}. We have implemented a simple model of 14 equations of that type (see

figure). This model do not pretend to be a very good model for qualimetry ; nevertheless, it shows the potential interest of the method. For example, our model predicts that if a higher evoluted *language* is adopted, *reliability* will increase, also will *testability, coherency, storage efficiency,* but *efficacity* an *structural complexity* will decrease. On the other hand, the model do not allow us to conclude what will be the resulting influence on *charge*. Simulation has provide 3 states depending on whether the charge remains constant, increases or decreases.

```
\*********************************************************
\                    QUALIMETRIC MODEL
\*********************************************************

\*********************************************************
\           Definition of the variables : quantity spaces
\*********************************************************

:-
quantites(Temps,time,[0,infp]),

quantites(Charge,load,[infm,zero,infp]),
quantites(Fiabilite,reliability,[infm,zero,infp]),
quantites(Efficacite,efficiency,[infm,zero,infp]),
quantites(Maintenabilite,maintenability,[infm,zero,infp]),
quantites(Testabilite,testability,[infm,zero,infp]),
quantites(Reutilisabilite,reusability,[infm,zero,infp]),

quantites(Coherence,coherency,[infm,zero,infp]),
quantites(Tolerance,'fault tolerancy',[infm,zero,infp]),
quantites(Generalite,generality,[infm,zero,infp]),
quantites(Completude,completness,[infm,zero,infp]),
quantites(Complexite_structurelle,'structural complexity',[infm,zero,
quantites(Complexite_textuelle,'textual complexity',[infm,zero,infp]),
quantites(Commentaires,commentaries,[infm,zero,infp]),
quantites(Efficacite_stockage,'storage efficiency',[infm,zero,infp]),

quantites(Langage,language,[infm,zero,infp]),
quantites(Couplage,coupling,[infm,zero,infp]),
quantites(Modularite,modularity,[infm,zero,infp]),
quantites(Revues,'code revues',[infm,zero,infp]),
quantites(Documentation,documentation,[infm,zero,infp]),

\***************** internal variables ***********************

quantites(P1,p1,[infm,zero,infp]),
quantites(M1,m1,[infm,zero,infp]),

quantites(P2,p2,[infm,zero,infp]),
quantites(M2,m2,[infm,zero,infp]),

quantites(N31,n31,[infm,zero,infp]),
quantites(N32,n32,[infm,zero,infp]),
quantites(N3,n3,[infm,zero,infp]),
quantites(M3,m3,[infm,zero,infp]),

quantites(P4,p4,[infm,zero,infp]),

quantites(P51,p51,[infm,zero,infp]),
quantites(P52,p52,[infm,zero,infp]),

quantites(P6,p6,[infm,zero,infp]),

quantites(P7,p7,[infm,zero,infp]),
quantites(N7,n7,[infm,zero,infp]),
quantites(M7,m7,[infm,zero,infp]),

quantites(M8,m8,[infm,zero,infp]),

quantites(N91,n91,[infm,zero,infp]),
quantites(N9,n9,[infm,zero,infp]),

\*********************************************************
```

```
\           constraints defining the qualimetric model
\*********************************************************

contrainte(add(Tolerance,Documentation,P1)),
contrainte(mmoins(Coherence,M1),[[zero,zero]]),
contrainte(add(P1,M1,Charge)),

contrainte(add(Tolerance,Testabilite,P2)),
contrainte(mmoins(Generalite,M2),[[zero,zero]]),
contrainte(add(P2,M2,Fiabilite)),

contrainte(add(Generalite,Tolerance,N31)),
contrainte(add(Reutilisabilite,N31,N32)),
contrainte(add(Langage,N32,N3)),
contrainte(mmoins(N3,M3),[[zero,zero]]),
contrainte(add(Couplage,M3,Efficacite)),

contrainte(add(Modularite,Coherence,P4)),
contrainte(add(Commentaires,P4,Maintenabilite)),

contrainte(add(Modularite,Coherence,Testabilite)),

contrainte(add(Modularite,Coherence,P51)),
contrainte(add(Generalite,P51,P52)),
contrainte(add(Commentaires,P52,Reutilisabilite)),

\                    *********************

contrainte(add(Revues,Langage,P6)),
contrainte(add(Documentation,P6,Coherence)),

contrainte(add(Revues,Modularite,P7)),
contrainte(add(Complexite_structurelle,Complexite_textuelle,N7)),
contrainte(mmoins(N7,M7),[[zero,zero]]),
contrainte(add(P7,M7,Tolerance)),

contrainte(mmoins(Couplage,M8),[[zero,zero]]),
contrainte(add(Modularite,M8,Generalite)),

contrainte(mplus(Documentation,Completude),[[zero,zero]]),

\                    **********************

contrainte(add(Langage,Modularite,N91)),
contrainte(add(Couplage,N91,N9)),
contrainte(mmoins(N9,Complexite_textuelle),[[zero,zero]]),

contrainte(mmoins(N91,Complexite_structurelle),[[zero,zero]]),

contrainte(mplus(Documentation,Commentaires),[[zero,zero]]),

contrainte(mplus(Langage,Efficacite_stockage),[[zero,zero]]),

\*********************************************************
\          resolution : the known variables are initialised
\                       unknown variables : ?

\   All the tendencies are considered to be steady (std)
\   because it is a static problem. Note however that the
\   qualitative values represent in reality the value of
\   the derivatives, as if the constraint net has been
\   derived one time
\*********************************************************

initialisation(P1,?,std),
initialisation(M1,?,std),

initialisation(P2,?,std),
initialisation(M2,?,std),

initialisation(N31,?,std),
initialisation(N32,?,std),
initialisation(N3,?,std),
initialisation(M3,?,std),

initialisation(P4,?,std),

initialisation(P51,?,std),
initialisation(P52,?,std),

initialisation(P6,?,std),

initialisation(P7,?,std),
initialisation(N7,?,std),
initialisation(M7,?,std),

initialisation(M8,?,std),

initialisation(N91,?,std),
initialisation(N9,?,std),

\                    ******************

initialisation(Temps,0,inc),

initialisation(Charge,?,std),
initialisation(Fiabilite,?,std),
initialisation(Efficacite,?,std),
initialisation(Maintenabilite,?,std),
initialisation(Testabilite,?,std),
initialisation(Reutilisabilite,?,std),

initialisation(Coherence,?,std),
initialisation(Tolerance,?,std),
initialisation(Generalite,?,std),
initialisation(Completude,?,std),
initialisation(Complexite_structurelle,?,std),
initialisation(Complexite_textuelle,?,std),
initialisation(Commentaires,?,std),
initialisation(Efficacite_stockage,?,std),

\          ******** command variables *************

initialisation(Langage,[zero,infp],std),
initialisation(Couplage,zero,std),
initialisation(Modularite,zero,std),
initialisation(Revues,zero,std),
initialisation(Documentation,zero,std).
```

5. Conclusions

A software engineer is able to intelligently predict the tendencies of quality factors of a software piece without performing heavy numerical computations. This paper shows that Qualitative Reasoning techniques, emerged in Artificial Intelligence in the last ten years, can be useful to perform the same type of qualitative analysis. The main advantage of this approach is that it does not require to precisely specify the dependencies among the different factors acting on the quality of the software but it can work from a weak description, which is without doubt much more realistic. Furthermore, working with qualitative values allows one to cover a whole class of numeric instances, which certainly may save much time in the analysis procedure. On the other hand, it makes it particularly easy to update the quality model and greatly benefits of being a generic approach. This paper shows that qualitative simulation allows one to analyze the quality of a piece of software dynamically by predicting the effects of varying some attributes of the software at hand. Tendency analysis and limit checking can be easily performed.

REFERENCES

[1] J. Holdsworth, *The Ami Project: Validating Quantitative Approaches to Software Management*, Eurometrics'92, Brussels, Belgium, 1992.

[2] T. Forse, *Qualimétrie des Systèmes Complexes*, OPL-France & Les Editions de l'Organisation (Eds), France, 1989 (in French).

[3] N. Fenton, *Software Metrics, A Rigorous Approach*, Chapman & Hall, 1991.

[4] C.R. Symons, *Software Sizing and Estimating*, Wiley, Chichester, 1991.

[5] Special Volume on Qualitative Physics, *Artificial Intelligence Journal, Vol. 24*, 1984.

[6] M.G. Singh, L. Travé-Massuyès (Eds), Decision *Support Systems and Qualitative Reasoning*, North-Holland, 1991.

[7] L. Travé-Massuyès, *Qualitative Analysis: Scope*, In: Singh M (ed) Systems and Control Encyclopedia, Pergamon Press, Supplementary Volume 1,1989, 473-481.

[8] L. Travé-Massuyès, *Qualitative Reasoning over time: History and current prospects*, Knowledge Engineering Review, Vol 7, n° 1, 1992.

[9] L. Travé-Massuyès, N. Piera, *Order of magnitude models as qualitative algebras,* 11th IJCAI, Detroit, USA, 1989.

[10] O. Raiman, *Order of magnitude reasoning*, Fifth National Conference on Artificial Intelligence *(AAAI-86)*, Philadelphia, USA, 1986.

[11] P. Dague, *Order of magnitude revisited*, First Qualitative Physics Workshop, Paris, France, 1988.

[12] L. Travé, J.L. Dormoy, *Qualitative Calculus and Applications*, In *IMACS* Transactions on Scientific Computing *88*, vol.2, J.C. Baltzer AG, 1989.

[13] L. Travé-Massuyès, *Qualitative Reasoning for Dynamical Systems Simulation*, In: Singh M.G. (ed)Systems and Control Encyclopedia, Pergamon Press, Supplementary Volume 2,1992, 887-897.

[14] J. MacCall & al., *Concepts and Definitions of Software Quality*, in Factors in software quality, NTIS, 1977, Vol 1.

[15] B. Boehm, *Characteristics of software quality,* TRW series of software technology, 1978, Vol 1.

[16] T. J. McCabe, *A complexity measure*, IEEE transactions on software engineering, 1976, vol SE-2, no4, pp 308-320.

[17] M. H. Halstead, *Elements of software science*, Elsevier 1977.

[18] A. Perlis, & al., *Software metrics: an analysis and evaluation,* MIT Press, 1981.

[19] R. Watts, *Measuring Software Quality, NCC Publications*, 1987.

[20] B. Kuipers. *Qualitative simulation.* Artificial Intelligence, Vol 29, 289-338, 1986.

[21] A.Missier, L. Zimmer, P. Jezequel, L. Travé-Massuyès, *The SQUALE qualitative simulator: potentialities for diagnosis,* International conference on fault diagnosis, Tooldiag, Toulouse, France, 1991.

[22] A. Missier. *Structures mathématiques pour le calcul qualitatif, Contribution à la simulation qualitative*, Phd thesis of Institut National des Sciences Appliquées, Toulouse, France, 1991 (in french).

[23] J. De Kleer and J. S. Brown, *A qualitative physics based on confluences*, Artificial Intelligence Journal, Vol 24, 1984.