Chapter 7.29
Fuzzy Logic Classifiers and Models in Quantitative Software Engineering

Witold Pedrycz
University of Alberta, Canada

Giancarlo Succi
Free University of Bolzano, Italy

ABSTRACT

The learning abilities and high transparency are the two important and highly desirable features of any model of software quality. The transparency and user-centricity of quantitative models of software engineering are of paramount relevancy as they help us gain a better and more comprehensive insight into the revealed relationships characteristic to software quality and software processes. In this study, we are concerned with logic-driven architectures of logic models based on fuzzy multiplexers (fMUXs). Those constructs exhibit a clear and modular topology whose interpretation gives rise to a collection of straightforward logic expressions. The design of the logic models is based on the genetic optimization and genetic algorithms, in particular. Through the prudent usage of this optimization framework, we address the issues of structural and parametric optimization of the logic models. Experimental studies exploit software data that relates software metrics (measures) to the number of modifications made to software modules.

INTRODUCTION: MODELING SOFTWARE PRODUCTS AND PROCESSES

The important objectives of quantitative software engineering revolve around building models that help express software quality in terms of some software metrics (measures) (Canfora, García, Piattini, Ruiz, & Visaggio, 2005; Cant, Jeffery, & Henderson-Sellers, 1995; Chhabra, Aggarwal, & Singh, 2004; Lanubile & Visaggio, 1997; Lee, 1993; Offutt, Harrold, & Kolte, 1993; Poels &

Dedene, 2000). There have been numerous approaches to such modeling pursuits. In addition to some "standard" linear and non-linear regression models, we can witness other techniques exploiting neural networks, machine learning, neural networks and fuzzy sets (Ebert, 1994, 1996; Mantere & Alander, 2005; Pedrycz, Han, Peters, Ramanna, & Zhai, 2001; Pedrycz & Succi, 2005; Pedrycz, Succi, Musílek, & Bai, 2001; Reformat, Pedrycz, & Pizzi, 2003; Thwin & Quah, 2005).

There are several compelling reasons behind studying different approaches to modeling software processes and software products.

- Typically, the available software data are quite limited yet they may involve a significant number of variables; in this sense, their sparse character requires careful attention.
- As the models produce results to be presented to the user (designers, managers, testers, etc.), it is highly advisable to assure high transparency (user-centricity) of the overall modeling process. In particular, this feature supports an interpretation of results and reveals relationships between software metrics and software quality.
- Software processes are human-oriented and not governed by any laws of physics. This strongly suggests considering modeling practices realized at some abstract levels engaging logic constructs.
- It is highly advisable to develop models in such a way that they could accommodate heterogeneous input information not necessarily being confined to numeric data.
- The models should be endowed with a significant level of flexibility and learning mechanisms to accommodate commonly encountered non-linear relationships and potential variability of the individual projects and software products.

Being aware of these main objectives, we may conclude that ideally the modeling framework should support the development of models in such a way that they combine a high degree of plasticity and learning abilities with an evident transparency and a significant level of interpretability. Interestingly enough, we could state without any exaggeration that the fundamentals of such modeling are inherently rooted in the world of multi-valued or fuzzy logic. The underlying logic nature of the models makes them transparent, and this transparency contributes to a highly interpretative insight into experimental data. The agenda of fuzzy modeling is inherently associated with the transparency of fuzzy models. While this facet of modeling has already started to gain visibility and properly balance the otherwise accuracy-driven fuzzy models, there are still a number of fundamental issues as to the definition of interpretability itself, granularity of models vis-à-vis the characteristics of experimental data, and assessment of the readability of the structure of the model itself (Reformat et al., 2003).

Two-valued logic forms a well-known boundary case of the fuzzy logic. The design of digital systems comes with a diversity of well-established, highly efficient, and scalable architectures and related development algorithms. By acknowledging a point of view that the two-valued logic is just a special case of fuzzy logic, we are then somewhat tempted to generalize or reformulate the already existing architectures and design practices of digital systems and cast them in the framework of fuzzy logic. This point of view is the crux of this approach to the development of fuzzy logic models. We focus on a certain standard technique of implementation of combinational systems by means of multiplexers (the approach which results in an array of multiplexers implementing any Boolean function); see McCluskey (1986) and generalize this concept to the world of fuzzy logic. In essence, we are concerned with the three main phases: (1) building a generic