Guest Editorial

STANDARDS AND SOFTWARE REUSE

Giancarlo Succi

> **reuse** *v* [Tn] use (smt) again: *reuse an old envelope.*
> **standard** *n* **1** thing used as a test or measure for weights, lengths, quality, purity, etc: *the standard of height required for recruits to the police force . . . People were very poor then, by today's standards,* i.e., compared with people today. **2** . . .
>
> *Oxford Advanced Learner's Dictionary*

The goal of software *reuse* is to reduce the cost of producing software systems and to increase the quality of the resulting products through the reuse of *standard* components.

The goal of *standardizing* software components is to make them easier to understand and handle, so that increasingly more people will use and *reuse* them.

There is a deep interaction between these two terms that are so easy and obvious to understand and express. However, we do not yet have a deep understanding of their inter-relationships.

What does it mean to define a set of standard reusable software components? Should we consider the reuse of their functionalities or the reuse of parts of their codes? Should the structure of the code be standardized to achieve higher reusability? Are corporate standards relevant to software reuse? How much are we willing to pay to have standard reusable components? Can we reuse standard processes? Do we want to standardize across a domain or inside the border of a domain; that is, are we pursuing horizontal or vertical reuse?

I do not know the answers to most of these questions, and I probably could not answer any single one with full confidence. And this issue of SV does not provide any answers either. Our goal is to start a multidimensional discussion—in the sense that all the dimensions of software reuse, from software process to domain analysis are discussed and all the key players, from academia to industry, from consultants to practitioners, are involved.

Significant theoretical insights on software reuse and on the key role of software reuse in the software development process have been gained. Still, the very practical question: "does it pay off?" has not been either fully answered or even approximated. The same applies to standards. Our hope is that by joining these two topics and taking a broad and concrete approach we can take a significant step toward the solution of these existential issues.

The structure of this issue reflects this multidimensionality. We begin with a survey paper on defining a framework for software reuse standards by **Baldo**, **Moore**, and **Rine**. This survey shows the areas where the work should focus, namely on principles, domain analysis, reuse of lifecycle processes, and reuse capability assessment. Each of these topics is analyzed from the perspectives of both researchers and practitioners.

Kovács outlines the pursuit of a standard reuse program in his firm, CIM-EXP. **Succi** and **Baruchelli** propose a method for evaluating the costs of production of standard reusable components. **Favaro** describes his methodology for identifying standard domain components. **Fenaroli** and **Valerio** discuss the goals that Thera S.p.A. has selected in adopting a domain analysis process. **Succi**, **Benedicenti**, **Predonzani**, and **Vernazza** propose a technique for software process reuse. **Doublait** shows the limits of current reuse efforts using his company, Sodalia S.p.A., as a testbed.

I would like to thank all the authors for their excellent work. A special thanks goes to CIM-EXP and to Thera S.p.A., which supplied valuable information on their current reuse practices. **SV**