Including Non-functional Issues in Anna/Ada Programs for Automatic Implementation Selection

Xavier Franch

Universitat Politècnica de Catalunya
Department Llenguatges i Sistemes Informàtics
Pau Gargallo 5. E-08028 Barcelona (Catalonia, Spain)
e-mail: franch@lsi.upc.es

Abstract. We present an enrichment of the Anna specification language for Ada aimed at dealing not only with functional specification of packages but also with non-functional information about them. By non-functional information we mean information about efficiency, reliability and, in general, any software attribute measuring somehow the quality of software (perhaps in a subjective manner). We divide this information into three kinds: definition of non-functional properties, statement of non-functional behaviour and statement of non-functional requirements; like Anna annotations, all of this information appears in Ada packages and package bodies and their syntax is close to Ada constructs. Non-functional information may be considered not only as valuable comments, but also as an input for an algorithm capable of selecting the "best" package body for every package definition in a program, the "best" meaning the one that fits the set of non-functional requirements of the package in the program.

1 Introduction

Component programming [Jaz95, Sit+94] is a useful and widely employed way of building complex software systems by means of combining, reusing and producing software components. What a component does is stated by its functional properties. Different implementations must satisfy them, but they will differ in some non-functional aspects, such as execution time or reliability.

Among other possibilities, we are interested in software components as an encapsulation of abstract data types (ADT) [Gut75], described by algebraic specifications and implemented using an imperative or object-oriented programming language. To be more precise, we view software components consisting of: a) the definition of an ADT stating both functional and non-functional characteristics, and b) one or more implementations, each one including a description of its non-functional behaviour.

In this paper, we propose component programming with ADTs using:
• Ada [Ada83] as the programming language. Then, definitions of ADTs are encapsulated in packages while implementations appear inside package bodies\(^1\).

• The Anna specification language [LH85, Luc90] for stating functional properties of ADTs.

• Some new constructs [Fra96, FB96, FB97] for dealing with non-functionality.

We consider three kinds of non-functional information:

• **Non-functional property** (short, *NF-property*): any attribute of software which serves as a means to describe it and possibly to evaluate it; for instance, time efficiency of a procedure or portability of a package body.

• **Non-functional behaviour** of a component implementation (short, *NF-behaviour*): any assignment to the NF-properties which have been declared of interest for the implemented software component.

• **Non-functional requirement** imposed on a software component (short, *NF-requirement*): any constraint referring to a subset of the NF-properties which have been declared of interest for the software component.

In order to make our approach more attractive, we integrate these last new constructs into the Anna notation, providing then an integrated framework where functional and non-functional aspects of software are uniformly considered, as we think they always should be.

The rest of the paper is structured as follows. We review in section 2 the main features of the Anna specification language. Sections 3, 4 and 5 introduce non-functional properties, behaviour and requirements, respectively. Section 6 shows how our packages are managed to produce different files distinguishing the non-functional part from the functional one. Section 7 gives an outline of the automatic selection algorithm. Finally, section 8 provides the conclusions.

### 2 The Anna Specification Language

The Anna specification language (ANNotated Ada) [LH85, Luc90] is a language extension of Ada that includes features supporting functional specification such that:

Anna program = Ada program + formal comments

Formal comments are just comments from the Ada point of view, and so Anna programs are acceptable by Ada compilers with no changes at all. However, these comments obey some syntactic rules and they have a semantic meaning. There are two types of formal comments:

---

\(^1\) Hereafter, we use the words "definition" and "package" interchangeably, and the same with "implementation" and "package body".