Abstract. The architecture description of a software product line (SPL) is essential to make it clear how the architecture realizes the feature model and to represent both the domain and application engineering architectural artefacts. However, most architecture description languages (ADLs) for SPL have limited support regarding variability management and they do not express the relationship between features and the architecture, besides the lack of tools for graphical and textual modelling and a non-clear separation between the domain and application engineering activities. In order to overcome these deficiencies, this paper presents LightPL-ACME, an ADL whose main goal is to be a simple, lightweight language for the SPL architecture description, and enable the association between the architectural specification and the artefacts involved in the SPL development process, including the relationship with the feature model and the representation of both domain and application engineering elements.

Keywords: Software product lines architectures, Architecture description languages, ACME, LightPL-ACME.
Abstractions for representing architectures through components, connectors, and configurations. Components represent software functionalities, connectors are communication elements, and configurations describe the relationship between components and connectors.

Although there are some ADLs for describing SPL architectures, most of them have limited support regarding the management of variabilities since they only focus on documenting SPL concepts (similarities and variabilities) and architectural elements rather than the relationship and traceability between the variabilities represented in the feature model and the architecture of an SPL. Moreover, these ADLs do not express the relationship between features and the architecture and suffer from the following limitations: (i) high verbosity that makes the architectural description confusing and difficult to understand; (ii) complexity for instantiating products; (iii) lack of tools for graphical and textual modeling; and (iv) lack of a clear separation between the domain and application engineering activities in the SPL development process.

In this context, this paper presents LightPL-ACME, an ADL that aims to provide a lightweight strategy for describing architectures of SPLs in order to overcome the abovementioned limitations. We have chosen the ACME ADL as basis of the proposed ADL since ACME provides generic structures to cope with a wide range of systems and includes a language based on first-order predicate logic called Armani, which is used to design architectural constraints. The main features of LightPL-ACME are: (i) semantic enrichment of ACME elements originally used to specify the SPL architecture, the so-called base architecture; (ii) elements designed to enable the definition of the referenced architecture, which is a base architecture whose elements refer to the features of the SPL; and (iii) products instantiation, which is based on the architectural description of the SPL and the referenced architecture. In this paper, we illustrate the main elements of the LightPL-ACME ADL with the GingaForAll SPL for Ginga, the middleware adopted by the Brazilian Digital Television System (SBTVD).

This paper is structured as follows. Section 2 describes the basic concepts regarding SPLs and provides an overview about the ACME/Armani ADL. Section 3 presents LightPL-ACME and its application for describing the GingaForAll SPL. Section 4 presents the LightPL-ACME Studio tool. Section 5 presents related work. Finally, Section 6 contains final remarks and future works.

2 Background

2.1 Software Product Lines

Software product lines (SPLs) enable the creation of a family (or product line) of similar products by using a common software infrastructure to mount and configure parts designed to be reused among products and following two main activities, namely domain engineering and application engineering. The construction of a software product, also called instantiation or derivation,