

Chapter 11

AKM in Open Source Communities

Ioannis Stamelos and George Kakarontzas

Abstract Previous chapters in this book have dealt with Architecture Knowledge Management in traditional Closed Source Software (CSS) projects. This chapter will attempt to examine the ways that knowledge is shared among participants in Free Libre Open Source Software (FLOSS¹) projects and how architectural knowledge is managed w.r.t. CSS. FLOSS projects are organized and developed in a fundamentally different way than CSS projects. FLOSS projects simply do not develop code as CSS projects do. As a consequence, their knowledge management mechanisms are also based on different concepts and tools.

11.1 Introduction

One should not expect to find in FLOSS the same knowledge management approaches and tools that are in use or considered in CSS. With respect to the architectural knowledge views (Chap. 2) the *dynamism-centered* view requires formalization of architectural knowledge and is probably distant from the “*spirit*” of OSS which is centered on the creation of source code. Also the *requirements-centric* view requires a cohesive team that co-evolves the requirements and the architecture in an iterative fashion, and therefore is more appropriate for closed development environments in which the collaboration is more direct. The *decision-centric* view seems to be very attractive for FLOSS projects, since the explicit documentation of architectural decisions’ rationale will enable distant developers to better capture the essential characteristics of FLOSS projects; however to the best of our knowledge

Ioannis Stamelos (✉)

Aristotle University of Thessaloniki, Greece, e-mail: stamelos@csd.auth.gr

George Kakarontzas, Department of Computer Science and Telecommunications, TEI of Larissa, Greece, and Aristotle University of Thessaloniki, Greece, e-mail: gkakaron@teilar.gr

¹ We use the term FLOSS in order to accommodate all three terms: Free, Libre, Open, that are in use to denote open source software

it is not currently used (at least not explicitly with tools specific to this approach) in FLOSS. For several reasons (that we will explain later in this chapter) it seems that the predominant approach to knowledge management is the *pattern-centric* approach.

This chapter attempts to provide answers to such questions, as:

- How do FLOSS processes differ from CSS processes? How are FLOSS projects organized and managed? Which are the incentives and motivations behind participating in a FLOSS project? How is architecture defined, implemented and assessed in FLOSS? Which is the current understanding about the level of quality of FLOSS architectures w.r.t. CSS architectures?
- How is quality pursued and achieved in FLOSS in general? How are decisions made for shaping FLOSS architectures? Is there anything like FLOSS architecture documentation?
- How is knowledge managed in FLOSS in general, who are the knowledge creation and maintenance mechanisms and tools in FLOSS, is FLOSS knowledge personalized, codified or both? How is domain AK matched with application AK? Who are the carriers of AK in FLOSS? How do all these apply to AKM in particular? What is the role of major software companies that support FLOSS on a regular basis?
- What should be expected in the future regarding AKM in FLOSS? Which are the research directions? What are the implications for software architecture education?

Initially, the chapter discusses briefly the fundamental differences between CSS and FLOSS processes providing the context for the analysis that follows. Then it proceeds by briefly discussing software architecture in FLOSS and moves into the core issue, i.e. how software architectural knowledge is generated, captured and managed by FLOSS communities. Sources for the chapter material are academic papers and books, and web resources, such as FLOSS project discussion lists, blogs, wikis, web pages and white papers, combined with authors' personal studies and experience with/on FLOSS, or FLOSS like projects.

11.2 FLOSS Projects in General

FLOSS projects often provide excellent examples of self-organized, successful projects, producing highly effective systems [119]. They are based on open, self organized communities of volunteers, who manage to develop, support and maintain software effectively. This unique kind of virtual communities provides an excellent environment for learning how to communicate with, cooperate with and ultimately learn from other members of the community. Knowledge generation and sharing [302, 301, 303] is implicit in the everyday operations of FLOSS communities.

One interesting variation of FLOSS is the so-called hybrid FLOSS projects. Such projects are initiated and supported by companies that are interested in developing