Chapter 3
AMPLA:
An Agile Process for Modeling Logical Architectures

Nuno António Santos
https://orcid.org/0000-0002-8247-7253
CCG, Universidade do Minho, Portugal & ZGDV Institute, Portugal

Nuno Ferreira
https://orcid.org/0000-0003-3561-331X
i2S Insurance Knowledge, S.A., Portugal

Ricardo J. Machado
CCG, Universidade do Minho, Portugal & ZGDV Institute, Portugal

ABSTRACT

Software architecture design, when performed in context of agile software development (ASD), sometimes referred to as "agile architecting," promotes the emerging and incremental design of the architectural artifact in a sense of avoiding "big design upfront" (BDUF). This chapter presents the Agile Modeling Process for Logical Architectures (AMPLA) method, an approach for supporting the emergence of a candidate (logical) architecture, rather than BDUF, the architecture in an early phase. The architecture then emerges throughout agile iterations, where AMPLA plays a key contribution for providing traceability between models, from the business need to service specifications, ranging from design stages to deployment, hence covering a software development life cycle (SDLC).

INTRODUCTION

At the time that the 'Agile Manifesto' (Agile Alliance, 2001) was proposed, there was a big shift on focusing in delivering software and less in technical documentation and specifications. Based in one of the values of the Manifesto, '*Working software over comprehensive documentation*', specification of requirements have been reducing to as minimum as possible. Thus, the use of software models was also

DOI: 10.4018/978-1-7998-4165-4.ch003

reduced, both in requirements and in design tasks, where only models that actually help teams develop software are used (Schwaber & Beedle, 2001).

In plan-driven approaches (e.g., Waterfall), tasks related to Requirements Engineering (RE) discipline are traditionally managed in a phase separated in time from design and development. In change-driven approaches, like ASD, RE discipline – also called "*Agile RE*" – activities remain the same but are executed continuously (Grau & Lauenroth, 2014), and takes an iterative discovery approach (Cao & Ramesh, 2008). Additionally, requirements modeling require an agile approach in order to prevent unnecessary efforts in "*You Aren't Gonna Need It*" (YAGNI) features.

In ASD frameworks, the requirements are included in a product backlog, typically in form of use User Stories (IIBA, 2017) as items in the backlog for "reminders of a conversation" about a functionality. However, using only User Stories, without attached requirements specifications or models, may be insufficient to assure a common understanding or, in case of multi-teams, to clearly define inter-systems interaction.

Typically, a first release on a new product encompasses a product's subset able to address priority scenarios, previously identified in order to respond to market needs. In fact, many of these product releases are market-driven, where the release is deployed into the market so it is possible to get feedback from it, *i.e.*, a minimum viable product (MVP). Alongside with these requirements concerns, projects struggle to design candidate architectures for the MVP, endangering development when they conclude that the architecture requires modifications and updates. In an era where software development is more and more agile-oriented, the upfront effort is replaced by the emergence of the design throughout iterative cycles. Such efforts are in opposition to "*Big Design Upfront*" (BDUF). In ASD contexts, BDUF approaches often result in features that are disregarded after some time (YAGNI features).

This chapter introduces an Agile Modeling Process for Logical Architectures (AMPLA), an approach for supporting the emergence of a candidate (logical) architecture, rather than BDUF the architecture in an early phase. AMPLA includes the core known features within the initial phase and designs a logical architecture using a stepwise method, without refining information. The emerging characteristics of AMPLA are supported in four stages, two performed before development cycles or Sprints and two in parallel with ASD cycles: (1) eliciting a small set of high-level requirements; (2) deriving a candidate logical architecture; (3) define subsystems for refinement; and (4) refine requirements and the architecture regarding the subsystem in small cycles or Sprints. As the name implies, AMPLA is guided by the Agile Modeling (AM) (S Ambler, 2002) philosophy. AM is about modeling practices, aiming to deliver small portions of models and collecting feedbacks, within ASD processes. AMPLA is driven to be "lean", because it aims to minimize waste within modeling, since the emerging modeling of the features enables leaving out YAGNI features. AMPLA is based in UML models, namely Use Cases diagrams for requirements, and Components diagrams for the logical architecture design. A logical architecture is a view that primarily supports the functional requirements, taken mainly from the problem domain (Kruchten, 1995). For the architecture derivation, AMPLA uses the *Four-Step-Rule-Set* (4SRS) method (Ferreira, Santos, Machado, Fernandes, & Gasevic, 2014) in order to assure the logical components are aligned with the functional requirements. AMPLA is demonstrated in this chapter within a research project called Unified Hub for Smart Plants (UH4SP).