

Can Agility be Introduced into Requirements Engineering for COTS Component Based Development?

Kendra M. L. Cooper
The University of Texas at Dallas
kcooper@utdallas.edu

Abstract

The Rational Unified Process is a popular software development process framework that can be tailored to meet the needs of different kinds of projects (large, small, component development, component based application development, etc.). This position paper presents an overview of a comprehensive RUP based process for developing COTS based systems and poses a set of questions that need to be considered to introduce agility into the process from a requirements engineering perspective.

1. Introduction

The need to rapidly produce quality software and respond to changes in a flexible and quick manner has driven the definition of new techniques including Agile Methods (AM) and Component Based Software Engineering (CBSE) approaches, which use a variety of software component products (e.g., off-the-shelf (OTS), commercial OTS (COTS), etc.). These approaches share many common goals: increasing the productivity of the development teams, reducing products' time-to-market, reducing the development costs, and improving customer satisfaction.

AMs are a set of software processes that share the same core values defined in Agile Manifesto [1]: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to changes over following a plan. Numerous AMs have been proposed including Scrum [11], Extreme Programming [2], and DSDM [6]. AMs are not prescriptive processes: they don't define detailed procedures for how to create a given type of model; instead they provide advice, or guidance.

The requirements in AMs are recognized as key elements; this can be seen from the highest priority principle in agile development: satisfy the customer. The requirements engineering (RE) activities and

artifacts are defined in some form by the AMs. For example, XP describes the RE activities in terms of User Stories. A User Story is a brief summary of a discussion with the customer about the requirements that is hand-written on an index card; they capture high-level functional and non-functional requirements and are prioritized. Simplified, point form versions of use cases have been proposed as an agile alternative to a fully defined use case model [2]. The point form use cases do not necessarily contain an overview, pre-conditions, post-conditions, and so on. The flows of events can be presented very simply as bullets.

CBSE methods are a set of software processes that support the identification, evaluation, acquisition, integration, testing, and/or maintenance of systems that are developed using components (i.e., reusable software products), rather than from scratch. The focus in this work is on COTS CBSE (CBSE_C) techniques, in which the products are blackboxes (source code is not available). The use of OTS products needs to be considered as a separate, more general problem, as the products are whiteboxes (source code is available). Proposals that emphasize the requirements engineering activities in CBSE_C include Scarlet [9], COTS Aware Requirements Engineering and Software Architecting [5], and the Evolutionary Process for Integrating COTS Based Systems (EPIC) [3].

Introducing agility into CBSE_C techniques has received little attention. An analysis of selected CBSE_C techniques with respect to agile principles is reported in [10]. Points of consistency and conflict are identified. For example, the authors' analysis on the agile principle to "welcome changing requirements" is viewed as being consistent with COTS based software development. Other principles, however, have a mix of consistent and conflicting points. For example, the principle that states the "business people and developers need to work together" brings up practical issues of having the vendors working on site, protecting intellectual property, etc. In AMs, the business people include stakeholders who understand

the requirements for the application under development.

The position presented in this work is that it is important to further investigate how to introduce agility into a CBSE_c. Here, the first step is taken by identifying some of the research questions and issues involved. A specific CBSE_c approach, EPIC, has been selected for this phase of the work to provide a concrete process definition to work with; additional approaches need to be considered. EPIC has been captured as a tailored version of the established Rational Unified Process (RUP).

The remainder of this paper is structured as follows. The RUP and a tailored version of the RUP for CBSE_c, the IBM RUP COTS plug-in, are overviewed in Section 2. Open research questions on defining an agile component oriented RE process are considered in Section 3. Conclusions are in Section 4.

2. Tailoring the Rational Unified Process

2.1 Rational Unified Process

The Rational Unified Process (RUP) is a popular, adaptable process framework that describes how to effectively develop software using proven techniques [8]. RUP is organized along two dimensions. The horizontal dimension represents the dynamic structure; it is described in terms of cycles, phases, iterations, and milestones. The vertical dimension represents the static structure. Here, disciplines (or workflows) such as business modeling, requirements, etc. are described in terms of the roles, activities, artifacts, workflows, and additional process elements (templates, guidelines, concepts, roadmaps, and help on tools). RUP defines activities (how to do the work), roles (who is doing the work), workflows (when the work is done), and the artifacts (what is delivered).

The RE discipline begins in the inception phase, in which the architecturally significant use cases are identified and outlined by the analyst. The work continues with substantial effort in the elaboration phase, in which the use case model is completed to 80%, then tapers off in the construction and transition phases. The requirements are managed throughout.

The RUP is intended to be tailored, to better meet the needs of a specific project or organization. RUP is included into the IBM Rational Method Composer (RMC) product, which allows customizing the process and publishing the resulting process through the internet [7]. A tailored version is called a plug-in. For example, a plug-in for agile projects with the Agile Unified Process has been proposed [1]; numerous

plug-ins have been published by IBM, including the COTS component based development plug-in.

2.2 EPIC and the IBM RUP COTS Plug-in

EPIC is a comprehensive, tailored version of the RUP that meets the challenges of building, fielding, and supporting COTS component-based business solutions [3]. It simultaneously defines and makes trade-offs among four *spheres of influence*: the stakeholders' needs and their business processes (i.e., more traditional RE, analysis), the components currently available in the marketplace, the architecture and design of the system, and programmatics and risks. The trade-offs are considered in each iteration. As iterations progress and decisions are negotiated, the solution space becomes smaller, knowledge of the solution accumulates, and stakeholder buy-in increases. As a result, the components available in the marketplace are re-examined in each iteration to determine if they are suitable (refer to Figure 1).

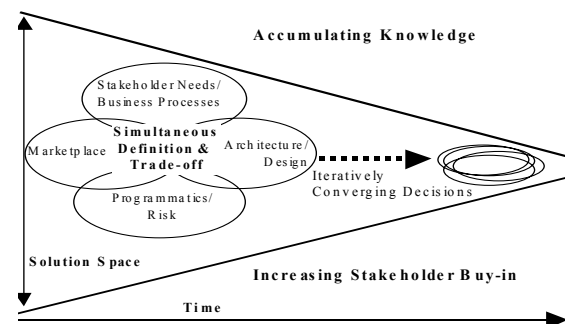


Figure 1. Overview of the EPIC Framework [3].

The milestones, workflows, activities, artifacts, terms, and descriptions from the RUP are re-used and/or modified as needed; new process elements are added. For example, new artifacts to characterize the component marketplace include the market segment information, component dossier (for each component that is considered), and the component screening criteria and rationale. For each new artifact, EPIC provides a guideline that describes the purpose of the information gathered, key questions to answer, the information needed, and techniques to use.

EPIC has been used as the basis for the IBM RUP for COTS Package Delivery plug-in. For example, templates for the COTS inception, elaboration, construction, and transition iterations are defined. Under each of these, the activities are refined into sub-activities and ultimately tasks. Each task has properties that can be instantiated, including the guidelines. The guidelines are left empty in the plug-in to provide

flexibility. The guidelines are targeted in this work as a way to introduce agility into the process.

3. Considering Agility and COTS in RE

The Agile Manifesto and the 12 principles for AMs [2] provide the basis for introducing agility into the IBM RUP COTS plug-in. Many of the principles are highly related to the RE discipline (e.g., satisfy the customer, adapt to changing requirements, deliver working software frequently, business people and developers work together). Of these principles, one in particular stands out as a significant technical challenge in introducing agility into RE for component based systems: *deliver working software frequently*.

AMs propose to deliver working software in short iterations (from a couple of weeks to a couple of months). From a RE perspective, these frequent deliveries are excellent opportunities to validate the requirements realized in the software with the customer. When the delivery is demonstrated, the customer has an opportunity to identify what they do and don't like; problems can be quickly addressed. However, many questions arise when considering the short iterations from a component perspective:

1. Does a short iteration give enough time to wisely identify, evaluate, select, and integrate COTS components to meet the customer's requirements?
 - a. Can a component dossier be simplified and still be useful or replaced with tacit knowledge?
2. How does the relationship with the vendor need to change to make COTS selection agile?
 - a. Can the evaluation workshop (i.e., meeting to evaluate the components) with the vendor be streamlined?
 - b. When, or for how long, is it appropriate to have the vendors working on-site?
 - c. Can the process to address change requests to a COTS component product be more agile?
3. What and how much information should be collected for the four spheres of influence, particularly in the inception and elaboration phases, when the RE effort is significant?
4. Are only some kinds of components suitable in an agile approach? For example, should only the following kinds of components be considered:
 - a. The components meet an international standard (IEEE, ISO, ANSI, etc.); their capabilities are well documented and understood
 - b. The components are relatively small and/or simple to understand and evaluate with respect to the customer's requirements. If the component is small and/or simple, then is it worth buying or should the requirements be realized in house?

- c. The developers have used the component before to meet similar requirements, reducing the effort needed to understand and use it.

5. Are all of the requirements potential candidates for realizing with COTS? AMs focus on realizing the requirements for the current iteration. Is there a way to characterize requirements that are better suited for realization with COTS, for example, those that are less likely to change in future iterations?

4. Conclusions and Future Work

There are numerous questions to investigate just considering the single agile principle to "deliver working software frequently" in the definition of an agile RE process that addresses the management of the component based system under development and the reusable software products. The other agile principle highly related to RE also need to be investigated.

5. References

- [1] Agile Manifesto, available at <http://www.agilemanifesto.org/>
- [2] Ambler, S., *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*, Wiley, March, 2002.
- [3] Albert, C. and Brownsword, L., *Evolutionary Process for Integrating COTS Based Systems (EPIC) Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions*, technical report CMU/SEI-2002-TR-005, November 2002.
- [4] Beck, K., *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.
- [5] Cooper, K., Ramapur, C., and Chung, L., *A COTS-Aware Requirements Engineering and Architecting Approach: Defining System Level Agents, Goals, Requirements and Architecture (Version 4)*, UTDCS-24-05, The Univ. of Texas at Dallas, December 2005.
- [6] DSDM Consortium, www.dsdm.org
- [7] IBM Rational Method Composer, www-128.ibm.com/developerworks/rational/library/nov05/kro11/index.html
- [8] Kruchten, P., *Rational Unified Process, The: An Introduction*, 3rd Edition, Addison-Wesley, 2004
- [9] Maiden, N., Kim, H., Ncube, C. "Rethinking Process Guidance for Selecting Software Components", in Proc. of 1st ICCBSS, LNCS 2255, 2002.
- [10] Navarrete, F., Botella, P., and Franch, X., "How Agile COTS Selection Methods are (and can be)?", in Proc. of the 31st EUROMICRO Conf. on Software Engineering and Advanced Applications, Porto, Portugal, Aug. 30th – Sept. 3rd, 2005, pp. 160-167.
- [11] Schwaber, K. and Beedle, M., *Agile Software Development with Scrum*, Prentice-Hall, 2001.