Pair Modeling

Pankaj Kamthan
Concordia University, Canada

INTRODUCTION

We model software for a variety of reasons: to assess the viability of or plan software systems to be built, to optimize use of (minimize, or ideally, avoid waste of) resources in response to inevitable changes in business, social, or technological environments, or simply to understand existing software systems. Indeed, as indicated by the model-driven approach to software development (Beydeda, Book, & Gruhn, 2005), models are becoming first-class members of organizations and software process environments that embrace them.

The examples of collaboration in implementation of software are well-known (Nosek, 1998; Williams & Kessler, 2003). As models get large and complex, the need for creating them collaboratively in a systematic manner arises, and we propose pair modeling as an approach.

There are two main factors that have motivated the realization of modeling in collaboration. First, modeling has become complex enough for an individual, requiring intimate knowledge of both the domain being modeled *and* the knowledge and skills in using a high-level modeling language. This is particularly non-trivial when new and large domains are being constantly addressed by software. The learning curve involved in the use of today's modeling languages and corresponding tools have only exacerbated the situation. Second, models created in an organizational context usually need social understanding and acceptance. From a semiotic (Stamper, 1992) viewpoint, the social level of model is concerned with the understanding of the meaning of symbols, taking into account an understanding of different stakeholder viewpoints and preferences. Modeling in pair enables the first transitory and viable step towards that.

The rest of the article is organized as follows. We first outline the motivational background necessary for later discussion. This is followed by the framework within which pair modeling occurs, contexts in which it should prove useful, trade-offs for conducting it, and its application to education. Next, challenges and directions for future research are outlined. Finally, concluding remarks are given.

BACKGROUND

Over the last decade, there has been an increasing reliance of activities and deliverables in the later phases in the software life cycle on model artifacts that are created *earlier*. In particular, modeling has begun to play an increasingly central role in adaptive software process environments such as extreme programming (XP) (Beck & Andres, 2005) and the unified process (UP) (Jacobson, Booch, & Rumbaugh, 1999). The increasing dependency on models raises the question of how such models can *cooperatively* be created within the context of a project team.

There are various instances in software engineering that necessitate human cooperation, in particular working in pairs. It has long been recognized that computer programmers alone cannot always produce error-free programs, and can benefit from assistance by others (Weinberg, 1998). A software pattern is an entity of reusable knowledge based on past experience and expertise that provides a proven solution to a recurring problem in a given context. A pattern often needs to go through "shepherding" (essentially a validation process) before it get accepted by the community at-large (Appleton, 1997). The author of the candidate pattern is assisted by a domain expert in refining the submission with the common goal that it will reach the status of a pattern. The trust and collaboration between the author and the shepherd is crucial for success. Collaborative programming (or pair programming) (Nosek, 1998) involves two people writing a program, and is one of the 12 founding practices of XP. In case of data flow diagramming, it has been shown empirically (Powell, Bordoloi, & Ryan, 2007) that learners perform better cooperatively than individually.

Pair Programming and Pair Modeling: Similarities and Differences

Pair modeling is complementary to pair programming. Still, there are evident similarities and differences between pair programming and pair modeling.

The infrastructure requirements and processes in both cases are, in principle, similar. Both aim for improved productivity and better quality of underlying artifacts. However, the affiliation, background knowledge, and skill set of persons involved in each case are different. For example, in pair programming, both persons are programmers from the same organization; however, in pair modeling the problem domain expert may be from one organization (the client) and the modeling expert from another (organization that is hired by the client for producing the software under development). While the pair in pair programming is essentially homogeneous in their knowledge and skills, the pair in pair modeling, by necessity, is heterogeneous in their knowledge and skills.

AN ANALYSIS OF THE PAIR MODELING ENVIRONMENT

We define pair modeling formally as a practice that involves two people such that one person (the primary person or the pilot) works on the model using some input device while the other (the secondary person or the co-pilot) provides support in decision making and provides input and critical feedback on all aspects of the model as it evolves.

The primary role is active but the secondary role is not passive. If the pair is a part of a team with more than two members, the structure of the pair can also change. Like in XP, the pair takes a collective responsibility of the model that is created. The roles are complementary to each other and each role requires a different skill set. Therefore, it is not automatic that the pair is interchangeable (and is thus non-symmetric).

The primary person needs to be an expert in the modeling language deployed while the secondary person must be an expert in the problem or solution domain as the case may be.

In the rest of this section, we discuss the environment within which pair modeling can be effective, its trade-offs, and how it can be applied to software engineering education.

Suitability of Pair Modeling

We contend that pair modeling may neither be feasible, nor a candidate option in certain cases. If an organization is following a software process in which systematic model building does not have a major role to play, such as when it is less than level 3 on the capability maturity model (CMM) (Paulk, Weber, Curtis, & Chrissis, 1995), adopting pair modeling may not be possible. Even if there is willingness, pair modeling does require two persons to be involved in an activity and it is not automatic that an organization may have the resources (people, budget, availability of tools) to commit to it. Also, doubling the number of people involved in the modeling process does mean double the employment cost. However, it does *not* automatically mean double the productivity.

Pair modeling may also be unsuitable in open source software (OSS) process (Raymond, 1999) environments where people tend to be distributed and the probability of physical proximity is low.

Finally, if there is a high probability of instability in the status of the pair being together, pair modeling may not be a recommended practice. For example, employment uncertainties or recurrent medical leaves of absence can bring about such a situation.

Cost of Pair Modeling

Like any other activity, pair modeling has its own associated costs that must be measured against the benefits for it to be socially, technically, and economically feasible.

Pair modeling requires certain infrastructure for its realization. The partners share a desk, an extra "neutral" computer (or at least a computer monitor) that does not belong to any of the partners, and an input device (keyboard, mouse). Since there will be two people sitting in front of a computer, it should be equipped with a reasonably large, preferably wide, screen. The computer must have the necessary capabilities, including modeling tools. As it is quite likely that conversations will pursue, for example, a laboratory dedicated to public use is not a recommended location.

Human Factors in Pair Modeling

Pair modeling is a human activity, and therefore human characteristics pertaining to it need to be taken