Extreme programming: All of the elegance but none of the models?

Sallyann Bryant
IDEAS Laboratory
University of Sussex
S.Bryant@sussex.ac.uk

Keywords: Extreme programming, XP, pair programming, metaphor, external representation

Abstract

Extreme programming (XP) is a software development methodology which is becoming increasingly popular, but about which there remain many unanswered questions. The current research on XP mainly focuses on academic studies or experience reports which do not question 'how' and 'why', but simply 'whether' the various techniques work. This paper suggests the investigation of three areas in order to obtain further insight into how XP might work. These areas are: External representations; metaphor and pair programming. It then suggests a design for an observational study to consider each of these in the hope of furthering our understanding from a cognitive perspective.

Introduction

Beck (2000) coined the phrase 'extreme programming' in the book 'Extreme programming explained' in which he explains its 12 practices. Three of these: external representations, pair programming and system metaphor are areas about which underlying psychology has not yet been considered. In fact, XP studies seem to directly contradict literature on the psychology of programming, which for example has consistently found both metaphors and external representations to be useful in problem understanding (e.g. Suwa & Tversky, 2002, Carrol & Thomas, 1982). The following paper discusses each of these areas, giving a brief explanation of their use and outlining potential issues requiring further exploration. This is followed by the design of a study, based on the ethnographic practice of grounded theory (Glaser & Strauss, 1967), to consider some of the implications of these practices.

External representations

Extreme programming requires a project to be defined as a number of 'stories'. The stories are written on a set of cards that are refined into tasks, and detailed on task cards. These cards typically consist of a single sentence. Beyond the cards, information is communicated verbally via a system metaphor or representationally via the code itself. Thus the only three types of physical representations prescribed are: story cards, task cards and the program code. This is in direct contrast with traditional system design, where a number of representations of the system design are produced and maintained, for example structured diagrams of various types. Extreme Programming asserts that these diagrams are not worth the effort required for their upkeep, as coding is an evolutionary endeavour and requirements usually change long before development is complete. This assumption that between them the cards and code articulate the system at the only levels necessary raises two questions:

1. How do developers define and understand problems and produce solutions using only the code as representation?
2. How do the user and programmers communicate about the requirements?

Verbal metaphor as architecture

XP recommends the use of a system metaphor instead of a formal architecture. This metaphor acts as a shared model, which may be directly taken from the problem domain ('naïve metaphors'

like ‘customer’), or may not (‘true metaphors’ like ‘cookie cutter’). For further examples of potential systems metaphors see Wake (2002).

Rather than being physically expressed, metaphors are used mainly to ‘talk about’ the system. In practical situations, the use of metaphor has been seen to be sparse (e.g. Harrison, 2003; Rumpe and Schroder, 2002; Deias et al, 2002, Lappo, 2002). In fact, despite being considered a ‘last resort’ in XP literature, current research shows that naïve metaphors are most often used. Empirical work suggests experts always have mental models of system design work in progress (Adelson & Soloway, 1988) and the many success stories regarding the use of metaphor in other arenas imply that true metaphor use is highly desirable. A first step towards understanding why the use of true metaphor is problematic would be:

1 Ascertain the extent to which metaphor use is attempted on XP projects

2 Identify where problems seem to occur via studies of commercial projects

Pair programming skills

XP advocates programming in pairs. A programming pair develop a task together, taking it in turns to ‘steer’. Pairs are dynamic and regular changing of pairs assist in maximizing knowledge distribution. Whilst studies have compared pair programming favourably with programming alone in terms of quality of software produced and positive side effects (e.g. Williams, 2002) its generalised use has been questioned.

One potential advantage of the pair programming approach is the availability of an apprenticeship environment (Lave & Wenger, 1991). Another issue concerns whether pair programming is more suited to particular working styles. A third interesting aspect is the collaborative use of a computer system designed for a single person. The collaborative use of a single screen is nothing new – people often work together on word-processing problems, for example (see Twidale, 2000). However, the provision of a collaborative programming tool, potentially for use remotely, could prove instrumental in allowing the benefits of pair programming to be more easily achieved. The questions raised are therefore:

1 What characteristics and behaviours lead to successful pair programming?

2 To what extent does pair programming provide an apprenticeship environment?

3 How suitable is existing technology in facilitating pair programming and what other alternatives might prove more suitable?

4 How feasible is distributed pair programming?

Study methodology

Where studies of XP have taken place, they have tended to be either academically based (e.g Williams et al, 2002) or practitioners experience reports (as often seen in proceedings from XP conferences). As exemplified by Curtis (1986), extrapolating academic findings to industry may not be useful. Whilst practitioners experience reports are helpful in advising how to introduce or practice XP, and highlighting potential problems, they do not provide insight into the mechanisms employed, nor do they tend to take a disciplined approach to study design. Ethnographic studies of extreme programmers ‘in the wild’ could assist in obtaining insight into the cognitive aspects of extreme programming. This paper suggests an observational approach in order to provide a real-world basis for testable hypotheses. As the study concerns hypothesis formation rather than testing, sample size and control of all independent variables is not critical to its success .

Rather than study XP as a whole, three specific phenomenon are of interest. The focus of the study can therefore be closely defined, allowing for the collection of a mixture of qualitative and quantitative data. This study format has been tested via a pilot pair programming observational study within the University of Sussex which suggest that the data gathering techniques are

operationally feasible. A small survey has also taken place. Whilst not sufficiently large to provide statistically significant results, this confirmed that pair programming, representing system architecture and the use of metaphor were areas where confusion or difficulty may occur. Thus indicating that these may indeed prove useful areas of study.

Study design

A pre-assessment will take place in order to ascertain an individual's effectiveness at pair programming. This pre-assessment will consider:

1 Managers assessment (High/Medium/Low rating)

2 Managers assessment according to Dick & Zarnett (2002) categories – communication, comfort in pair working, confidence and ability to compromise.

3 Self assessment of expertise (how experienced a pairer are You - H/M/L?)

4 Self assessment of attitude (do you like pair programming? Is it useful?)

5 IT experience, general programming experience, programming experience in this language/environment

6 Months pairing experience

7 Academic qualifications

During 'the planning game' the production and use of metaphor and external representations will be noted and examined. Any representations used will be recorded. Notes will provide information regarding who they were produced by, at what point, for what purposes they were used, what form they took, how long they persisted and how they were discarded. For any metaphors used, the metaphor itself will be noted, whether it was naïve or true, how and by whom it was produced and how it was articulated. If possible this will be recorded either with sound or video for verbal articulation and/or still photographs. Particular attention will be paid to any problems in the production or use of metaphor and how the metaphor evolves over the project.

During the pair programming study discussed below similar attention will be paid to the use of metaphor and the creation/use of alternative metaphors. Notes will detail whether discarding the metaphor is an active decision and why, when and how this takes place. Where external representations of metaphors are produced they will be captured with notes on when, why, how and by whom they are produced, how long they are used for and for what, and when, how and by whom they are discarded. Follow-up interviews may be required to clarify some aspects.

The pair programming observational study will then produce data in two forms. First, a matrix of potential behaviours (see Fig 1) will be used in order to produce quantitative information. This matrix was produced using the potential pair programming behaviours described by Wake (2002) and then adapted following the pilot study. Each row represents one minute and each entry is numbered in order to retrospectively ascertain the sequence of interactions. Additional behaviours will be noted. The format will be validated by recording the first observation session.

Programmer A: B: Date: Time:

Driver	Explain	Ask	Confirm/ Agree	Review/ Refactor	Test	Suggest/ Counter	Remind	Rest	Solo	Meta-phor	Look up	Ext Repr

Figure 1 – Matrix of pair programming behaviours

A number of programmer pairings will be observed, each for the duration of a single task (approx 1 day). In order to draw a comparison between successful and unsuccessful pairers and assess the

effect of a cross-pairing, the study will attempt to observe at least one pair of each of the following types:

A. Two highly successful pairers according to defined rating
B. Two unsuccessful pairers according to defined rating
C. One successful pairer with one unsuccessful pairer[1]

Hand-written notes, time-stamped to cross-refer to the table, will capture further information about the use of metaphor and external representation. Whilst the majority of information will therefore be in the form of completed tables and notes, where possible audio recordings, photographs and video footage will be used. This may be limited according to the extent to which the company and any individuals involved are able to authorise such recordings. Each evening the author will annotate the notes with any relevant information and note any thoughts, theories or additional observations regarding immerging patterns or hypothesis.

Participation in the study will be on a voluntary basis. A disclaimer will inform subjects of these rights before the study begins. As the subjects of study are working in a pressured environment, interaction and interruption will be minimised by the observer, who will remain sensitive to the needs of both the individuals and the company for which they work. Participants will remain anonymous in all resultant material.

Data analysis

The data will be analysed in a number of ways: The tables will be used to ascertain whether there are significant differences in behaviour between 'successful' and 'less successful' pair programmers. Further analysis will then consider the effect of inter-type pairing on these behaviours. Information regarding the use of metaphor and external representation will be assessed on an overall basis to provide information about their usefulness and application. This will then be considered on a case by case basis to ascertain whether either technique was used more often, for longer periods or more successfully by more or less successful pairers.

Conclusion

Within XP experience reports pair programming is generally considered a successful, useful and enjoyable approach, however, some have reservations about its applicability to all situations. Data from the observational study will give insight into successful pair programming in two ways: First, it may help provide information regarding the suitability of particular individuals to pair programming by identifying successful pairing characteristics. Second, the identification of successful pairing behaviours may assist in identifying training needs to improve pairing. It may also provide evidence regarding whether cross-pairing successful with less successful pairers can provide a helpful learning experience.

Current literature seems to suggest that metaphors in the XP sense are rarely used, and when they are their use is somewhat problematic. Similarly, it assumes that the production of diagrammatical representations - particularly of the systems architecture - are an un-necessary overhead. This is in direct contradiction with psychology of programming studies, which have shown metaphors and external representations to assist problem solving. This study hopes to start to unravel these apparent differences in opinion, along with obtaining real world examples of the uses and abuses of metaphor and external representation use in XP.

[1] Medium successful pairers are being excluded to better assess the two extremes

Acknowledgements

The author would like to thank Professor Benedict du Boulay and Dr Pablo Romero for their invaluable advice and ideas.

References

Adelson B & Soloway E (1988). A model of software design. The Nature of Expertise, Chi MTH, Glaser R, Farr MJ (eds), p185-208. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Beck K (2000). Extreme programming explained: Embrace change. Addison Wesley

Carroll JM, Thomas JC (1982). Metaphor and the cognitive representation of computing systems. IEEE Transactions of systems, man and cybernetics, 12 (2), p107-116.

Curtis, Bill (1986). By the way, did anyone study any real programmers? Empirical studies of programmers, Soloway E & Iyengar S (Eds), p.256-261

Deias R, Mugheddu G, Murru O (2002). Introducing XP in a start-up. Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering

Dick AJ, Zarnett B (2002). Paired programming and personality traits. Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering

Glaser BG & Strauss AL (1967). The discovery of grounded theory: Strategies for qualitative research. Aldine de Gruyter, Hawthorne, New York

Harrison NB (2003). A study of extreme programming in a large company. www.research.avayalabs.com/techreport/ ALR-2003-039-paper.pdf

Lappo P (2002). No pain, no XP: Observations on teaching and mentoring extreme programming to university students. Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering

Lave J & Wenger E (1991). Situated Learning: Legitimate peripheral participation. Cambridge University Press

Rumpe B, Schroder A (2002). Quantitative survey on extreme programming projects. In Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering.

Suwa M & Tversky B (2002). External representations contribute to the dynamic construction of ideas. Diagrammatic representation and inference, Hegarty M, Meyer B, Narayanan N H (Eds), p341-343.

Twidale, M.B. (2000). Interfaces for Supporting Over-The-Shoulder Learning. Proceedings, HICS 2000: The Fifth Annual Conference on Human Interaction with Complex Systems. The Beckman Institute, University of Illinois at Urbana-Champaign.

Wake, W (2002). Extreme programming explored. Addison Wesley, NJ, USA

Williams L, Kessler RR, Cunningham W, Jeffries R (2002). Strengthening the case for pair-programming. IEEE Software 2002.