Instituto Tecnológico de Tijuana

Ingeniería en Sistemas Computacionales

Investigación I:

Pair Coding

Materia: Datos Masivos

Unidad: Unidad I

Facilitador:

José Christian Romero Sánchez

Alumno: Hernández Negrete Juan Carlos

Fecha:

Tijuana Baja California a 09 de Octubre del 2020.

Pair Coding

Definition

Pair programming consists of two programmers sharing a single workstation (one screen, keyboard and mouse among the pair). The programmer at the keyboard is usually called the “driver”, the other, also actively involved in the programming task but focusing more on overall direction is the “navigator”; it is expected that the programmers swap roles every few minutes or so.

Also Known As

More simply “pairing”; the phrases “paired programming” and “programming in pairs” are also used, less frequently.

Common Pitfalls

- Both programmers must be actively engaging with the task throughout a paired session, otherwise no benefit can be expected
- A simplistic but often raised objection is that pairing “doubles costs”; that is a misconception based on equating programming with typing – however, one should be aware that this is the worst-case outcome of poorly applied pairing
- At least the driver, and possibly both programmers, are expected to keep up a running commentary; pair programming is also “programming out loud” – if the driver is silent, the navigator should intervene
- Pair programming cannot be fruitfully forced upon people, especially if relationship issues, including the most mundane (such as personal hygiene), are getting in the way; solve these first!

Origins

The names of various celebrities have been invoked in an attempt to imbue pair programming with an aura of necessity if not sanctity; anecdotes of John Von Neummann, Fred Brooks, Jerry Weinberg, Richard Gabriel or Edsger Dijkstra using the practice are fascinating but sometimes hard to substantiate. However, the following timeline of verifiable sources does suggest that pair programming, in its modern form, has been around since well before the Agile movement:

1. 1992: “Dynamic Duo” is the term coined by Larry Constantine, reporting on a visit to Whitesmiths Inc., a compiler vendor started by P.J. Plauger, one of the implementors of C: “At each terminal were two programmers! Of course, only one programmer was actually cutting code at each keyboard, but the others were peering over their shoulders.” Whitesmiths existed from 1978 to 1988.

2. 1993: “The benefits of collaboration for student programmers” by Wilson et al. is one early empirical study indicating benefits of pairing for programming tasks specifically. Posterior studies are more abundant and driven by the desire to “validate” pair programming after it had already gained popularity through Extreme Programming.

3. 1995: the pattern “Developing in Pairs” is given a brief description, in Alexandrian pattern form, in Jim Coplien’s chapter “A Generative Development-Process Pattern Language” from the first patterns book, “Pattern Languages of Program Design”.

4. 1998: in “Chrysler goes to Extremes”, the earliest article about Extreme Programming, pair programming is presented as one of the core practices of the C3 team; it is later described formally as one of XP’s original “twelve practices”

5. 2000: (or earlier) – the roles of Driver and Navigator are introduced to help explain pair programming; the earliest known reference is a mailing list posting; note however that the reality of these roles has been disputed, for instance Sallyann Bryant’s article “Pair programming and the mysterious role of the navigator“

6. 2002: “Pair Programming Illuminated“, by Laurie Williams and Robert Kessler, is the first book devoted exclusively to the practice and discusses its theory, practice and the various studies up to that date

7. 2015: James Coplien publishes Two Heads are Better Than One which provides an overview of the history of Pair Programming that traces its origins back to the mid 1980’s if not before.

Skill Levels

As suggested above one of the major issues preventing effective pairing is passivity. When used simultaneously with test-driven development, one variant called “ping-pong programming” encourages more frequent switching of roles: one programmer writes a failing unit test, then passes the keyboard to the other who writes the corresponding code, then goes on to a new test. This variant can be used purely for pedagogic purposes, or by already experienced programmers as a playful variant.

- *Beginner*:
    - Able to participate as navigator, in particular to intervene appropriately
    - Able to participate as driver, in particular to explain code while writing it
    - Intermediate
    - Can tell the right moment to give up the keyboard and switch roles
    - Can tell the right moment to “steal” the keyboard and switch roles
- *Advanced:*
    - Able to “drop in” when another pair has been working on a task and pick up the navigator role smoothly

Signs Of Use

- The room's furniture and workstations are set up so as to encourage pairing (in teams new or hostile to pairing, obvious mistakes are tolerated, such as desks with too little room for two chairs)

- The room's noise level is controlled: the muted conversations from several simultanous pairs create a background hum but do not rise to the level where they would disturb anyone's work

- If, on entering the room, you spot any programmer wearing an audio headset, take that as a "negative" sign – not only is pairing probably not practiced in the team but the conditions for successful adoptions are likely not met

Expected Benefits

- Increased code quality: "programming out loud" leads to clearer articulation of the complexities and hidden details in coding tasks, reducing the risk of error or going down blind alleys
- Better diffusion of knowledge among the team, in particular when a developer unfamiliar with a component is pairing with one who knows it much better
- Better transfer of skills, as junior developers pick up micro-techniques or broader skills from more experienced team members
- Large reduction in coordination efforts, since there are N/2 pairs to coordinate instead of N individual developers
- Improved resiliency of a pair to interruptions, compared to an individual developer: when one member of the pair must attend to an external prompt, the other can remains focused on the task and can assist in regaining focus afterwards

Referencias

- Desconocido. (2019). Pair Programming: Does It Really Work?. 2020, de Agile Alliance Sitio web: https://www.agilealliance.org/glossary/pairing/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'pair*20programming))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

- Javier Garzas. (25 de Junio del 2012). ¿Beneficios del pair programming? ¿Dos programadores en un solo ordenador es perder medio equipo?. 2020, de 233Academy.com Sitio web: https://www.javiergarzas.com/2012/06/beneficios-pair-programming.html