# Liberating Pair Programming Research from the Oppressive Driver/Observer Regime

Stephan Salinger, Franz Zieris, Lutz Prechelt
Freie Universität Berlin
Institut für Informatik
Takustr. 9, 14195 Berlin, Germany
salinger|zieris|prechelt@inf.fu-berlin.de

***Abstract*—The classical definition of pair programming (PP) describes it via two obvious roles: driver (the person currently having the keyboard) and observer (the other, alternatively called navigator). Although prior research has found some assumptions regarding these roles to be false, so far no alternative PP role model took hold. Instead, most PP research tacitly assumes the classical model to be true and thus PP to be no more difficult than solo programming.**

**We perform qualitative research (using Grounded Theory Methodology) to find a more realistic role model, and have uncovered a suprising complexity: There are more than two roles, they are assumed and unassumed gradually, multiple roles can be held by one person at the same time, and some of their facets are subtle.**

**Mastering this complexity requires specific PP skills beyond mere programming and communication skills. By ignoring such skills, previous PP studies (in particular the controlled experiments) have investigated a rather mixed bag of situations, which explains their heterogeneous results.**

**The emerging result is that qualitative research on the PP process will lead to constructive behavioral advice (process patterns) for pair members and to more meaningful designs for quantitative PP research.**

## I. On Pair Programming

The "classical" definition of pair programming (PP) is probably the one by Williams, Kessler, and Cunningham [1] which says *"In pair-programming, two programmers jointly produce one artifact (design, algorithm, code, etc.). The two programmers are like a coherent, intelligent organism working with one mind, responsible for every aspect of this artifact. One partner is the 'driver' and has control of the pencil/mouse/keyboard and is writing the design or code. The other person continuously and actively observes the work of the driver – watching for defects, thinking of alternatives, looking up resources, and considering strategic implications of the work at hand. The roles of driver and observer are deliberately switched between the pair periodically. Both are equal, active participants in the process at all times and wholly share the ownership of the work products whether they be a morning's effort or an entire project."* The Observer role is sometimes called Navigator instead, but with the same idea of its meaning.

PP is claimed to reduce the wall-clock time required, to improve the design quality of the resulting code, to reduce the number of defects that remain in code, to accelerate learning, to improve the information flow within a team, to reduce bottlenecks in the availability of someone knowing a particular piece of code, to be more fun, and to have various other positive properties [2].

PP advocates suggest that PP should be the standard or even exclusive mode of programming. PP skeptics (e.g. from management) usually doubt that it can be wise to invest up to twice the amount of resources if a single programmer could also solve the task.

## II. On Pair Programming Research

There is a substantial body of empirical research on PP; most of the studies are controlled experiments. They find that PP is usually faster than solo programming (in hours, not person hours) and that it tends to reduce the number of defects in the resulting code [3]. Some surveys and interviews show that pair programmers have higher confidence in their work results and tend to have higher work satisfaction ( [1], [4]–[6]).

There is substantial conflict across many results; for instance in a meta-analysis done by Hannay et al. [3] regarding the duration, *"two of [...] 11 studies show a negative effect, while the remaining nine show positive effects. [...] Heterogeneity is significant at a medium level [...]."* Empirically founded explanations for these differences are yet lacking.

Almost all previous PP research tacitly assumes

1) that whenever you stick two people together to do PP, the above classical definition will apply, and
2) that, while general programming skill, general communication skill, and domain knowledge are important for PP success, there is no such thing as a relevant PP skill.

## III. Our Pair Programming Research Approach

In our work, we are interested in how (not just how well) PP actually works: what it is that the pair members really do and how that contributes (or doesn't) to the positive properties listed as claims above. Our long-term goal (which we have been following since 2004) is to describe how to do PP well and what to avoid and to explain this in the form of behavioral patterns and anti-patterns.

We are convinced that the second of the above tacit assumptions is dangerous. We will present evidence that carrying out a PP session is a complex process and therefore requires specific

PP skill. Ignoring this fact will produce misleading research conclusions.

The first of the tacit assumptions is also dubious. For instance, Sallyann Bryant has investigated its implicit claim that driver and observer are mostly working on different abstraction levels: The driver on medium levels (*"the code"*), the observer on high (*"strategic implications"*) and low levels (*"defects"*). She found that this is not the case; rather, driver and observer mostly move through similar levels of abstraction *together* [7]. However, Bryant's work stops at this point; she does not propose an alternative model of PP roles.

Our research has now led us to the topic of roles, so we start filling this gap in the present work. Our results so far are not primarily a new set of roles, but rather a meta model describing the elements and context of roles and role use. The present article focuses on conclusions from these incomplete results regarding the overall approach needed for meaningful PP research.

Methodologically, our raw data comprises recordings of complete PP sessions, consisting of desktop video, webcam video, and audio. We analyze such data by conceptualizing it via the Grounded Theory Methodology (GTM), in particular open coding and axial coding [8], starting at a fairly fine granularity, one code typically covering one or a few seconds of material [9]. The results sketched below arise from the first few PP sessions we have analyzed.

## IV. Pair Programming Roles

When we started our GTM analysis with the goal of identifying PP roles, we encountered a confusing variety of role-related phenomena. To create order, we arranged the stable concepts formed from these phenomena into a PP roles meta model. A simplified version is shown in Figure 1.

Note there are two different perspectives on PP roles:

- The researcher perspective is interested in understanding and characterizing the roles.
- The practitioner perspective is interested in obtaining practical advice on PP behavior such as role use.

The meta model adopts the researcher perspective: it is primarily a tool that helps the analysis process. Our discussion is mostly from the researcher perspective and considers the practitioner perspective only on the side, because the results are only just emerging and not yet mature enough to formulate much practical advice.

This section will proceed to present the (simplified) meta model and explain each of its classes; the roles themselves are instances of one of these classes and only some will be described as examples that serve to explain the meta model concepts and ideas.

### *A. 1st Role, Role, Facet, Action*

In the meta model, a **Role** is described by means of characteristic **Facets**. For example, the Role *watchman* has three Facets:

- *recognizing hazards*: Detecting and mentioning issues that could become problematic, for example asking whether the current set of files is up-to-date with respect to the versioning system or considering possible negative consequences for the ongoing overall release process in case the PP session does not achieve its goal fully.
- *setting priorities*: Insisting that something be done now (or early enough) to avoid negative or ensure positive consequences.
- *shifting*: Triggering a switch from one context of consideration into another, for example from considering the specific PP session to considering the overall development process.

Roughly speaking, the mission of a *watchman* is momentarily making sure nothing bad happens while the partner is occupied with something else.

Unfortunately, a mission as such is difficult to observe when looking at a specific PP session. In contrast, the Facets connect the Role to **Action**s, which are readily observable. There is no fixed rule as to which or how many Facets must be observed to conclude the respective pair member has assumed the Role. Usually, a single Facet is sufficient if it occurs multiple times and multiple Facets are sufficient even if they occur only once, but this depends on the particular Role and situation.

When it comes to the *watchman*, for instance, one might see just one single instance of *recognizing hazards*, but already consider the *watchman* role to be filled, because it is obvious that only prolonged attention can have enabled the pair member to make this particular intervention.

If such attention was observable, it would be a Facet, because it is quite typical and characteristic of *watchman* behavior. However, as attention can be observed only indirectly and this happens only rarely, we have decided that, for our role analysis, the attention concept is insufficiently operationalizable to become a Facet.

Note that, from the practitioner perspective, learning to be attentive is likely key to learning to use the *watchman* role properly. So once we move from role analysis to giving constructive advice, the Roles will have to be reformulated to include relevant unobservable aspects as well.

### *B. 2nd Role, Knowledge, Context, Episode*

As a second example, the Role *task expert* has two Facets:

- Facet-1, *passing on task knowledge explicitly*: Task knowledge is **Knowledge** about artifacts created before the current PP session that is relevant for the task addressed by the session. The Facet describes communication of such knowledge from one pair member to the other.
- Facet-2, *turning task knowledge into proposals*: Making a product-related or process-related proposal that only someone who is in possession of appropriate task knowledge can make.

Roughly speaking, the mission of a *task expert* is to bring into the session a certain important type of task-relevant information.

For *task expert* it is common that an occurrence of only one of the Facets is sufficient for ascertaining the Role. Which of

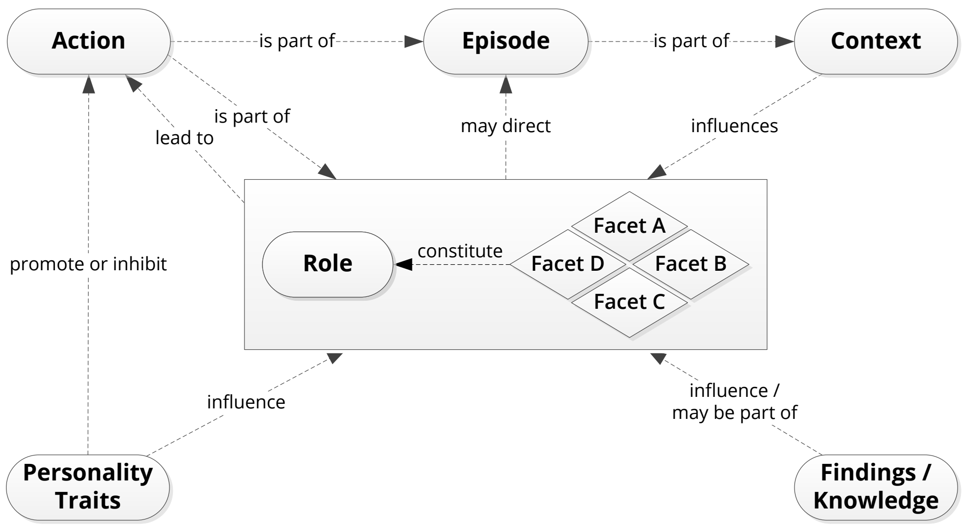

Fig. 1. Simplified meta model of PP-roles-related phenomena. The figure uses an informal notation whose semantics will become sufficiently clear within the text.

the Facets to expect depends on the current **Context** within the session: A PP session can usually be interpreted as a sequence of **Episodes**. If, for instance, the session is currently in a knowledge transfer episode, a task expert will execute his/her role only in the form of Facet-1, not Facet-2.

### *C. 3rd Example, Personality Traits*

A *spokesperson* has the mission to act as representative for a concern, typically towards people outside the pair. Facets of the *spokesperson* Role are:

- *opening a dialog* about the concern.
- *carrying forward the dialog*, often in such a way that expert knowledge is not required.
- *rounding off the dialog*: Not accepting an end of the dialog without a resolution of the concern.

It is typical for the *spokesperson* Role that it is assumed by the more assertive member of the pair. Assertiveness, however, is a **Personality Trait** and should not be confused with a Facet or Role.

### *D. Other Roles*

There are many more Roles beyond the above three, but our analysis of them is far from complete. We have, however, already recognized some regular structures. One of them is the fact that some Roles have counterparts. For instance when a Facet-1-type *task expert* explains task knowledge, the other pair member acts as *mentee*. A *task expert* can have a counterpart, but needs not (the Facet-2 case), while other Roles never have a counterpart (e.g. *watchman*) or always enforce or require one (e.g. *guide* and *robot*).

### *E. Further Observations from Our Role Analysis*

Three facts complicate defining Roles or recognizing Role assumptions:

- A Role assumption may start and end gradually over a stretch of time.
- Having a Role does not mean acting out that Role at every moment.
- A pair member may have more than one Role at the same time.

The full version of the meta model uses several additional classes (for instance Role Filling and Facet Execution) to describe such phenomena.

### *F. Towards a Better Role Catalog*

We found that we could define Facets reasonably well but the notion of a particular Role always remained somewhat arbitrary. For instance, we once had an *updater* Role that occurred only at the beginning of a session and explained the initial status of the artifacts to make clear from where the work will have to start. We have since incorporated this in the much more general *task expert* Role and find this new Role quite convincing and appropriate. In contrast, we also had a more specialized predecessor of *watchman*: The *foreign minister* Role whose mission amounted to only a subset of the *recognizing hazards* Facet of today's *watchman*. Yet, even though we have generalized away the *foreign minister* Role, we are far from convinced that the *watchman* Role will persist in its current form.

It may well be that we will never find a canonical set of Roles that is convincing for everybody, in particular since pair programmers' Role needs will likely be different depending on their work context and tasks. One solution might be to define a somewhat larger set of overlapping Roles from which a pair can pick the ones they deem most appropriate for their purposes.

## V. What Is Emerging?

Above, we have sketched roles assumed by PP participants during a PP session that go far beyond the conventional driver/observer role model in many respects:

- PP participants fulfill more different, conceptually separate functions (the Facets) than only the commonly described and hopelessly overloaded driver and observer (or navigator) roles.
- There are hence also far more than just two roles.
- It is not at all obvious which Facets belong together to form a meaningful role.
- "having the keyboard" and "using the keyboard" are Facets, but they are neither particularly highly relevant nor relevant throughout.
- Roles are switched more frequently and more fluently than meets the eye.

The list of roles and also the understanding of the contents of each role are still incomplete; the understanding of the influence that each role has on the PP process overall is only just emerging.

Nevertheless, this extended view of how the members of the pair complement each other has a number of important consequences:

- The driver/observer model, with its inflexible assignment of responsibilities, may mislead beginning pair programmers into dysfunctional behavior.
- There is currently a lack of guidance how to do PP efficiently.
- This also means that previous PP research may have investigated a wild mix of different PP styles and pair member PP competence levels, which may explain the partially inconsistent results.
- Therefore, qualitative (and qualitative-quantitative) PP research should be used more often, to facilitate better use of PP by painting a more complete picture of what PP actually is and by eventually providing guidance in the form of best practices. For instance, the relative frequency with which different roles (should) occur in a session may correlate with task type. If task type is known, this can allow programmers to focus on important PP activities ("we must not neglect X-role behavior").

## Acknowledgments

This work was partially supported by a DFG grant.

## References

[1] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," *IEEE Software*, vol. 17, no. 4, pp. 19–25, 2000.

[2] L. Williams and R. Kessler, *Pair Programming Illuminated.* Addison-Wesley Professional, 2002.

[3] J. Hannay, T. Dybå, E. Arisholm, and D. Sjøberg, "The effectiveness of pair programming: A meta-analysis," *Information and Software Technology*, vol. 51, no. 7, pp. 1110–1122, 2009.

[4] B. Hanks, C. McDowell, D. Draper, and M. Krnjajic, "Program quality with pair programming in CS1," in *ITiCSE '04: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education.* New York, NY, USA: ACM Press, 2004, pp. 176–180.

[5] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald, "The impact of pair programming on student performance, perception, and persistance," in *ICSE '03: Proc. 25th Int'l Conf. on Software Engineering.* IEEE Computer Society, 2003, pp. 602–607.

[6] J. T. Nosek, "The case for collaborative programming," *Communications of the ACM*, vol. 41, no. 3, pp. 105–108, 1998.

[7] S. Bryant, P. Romero, and B. du Boulay, "Pair programming and the mysterious role of the navigator," *International Journal of Human-Computer Studies*, 2008.

[8] A. L. Strauss and J. M. Corbin, *Basics of Qualitative Research: Grounded Theory Procedures and Techniques.* SAGE, 1990.

[9] S. Salinger and L. Prechelt, "What happens during pair programming?" in *Proceedings of the 20th Annual Workshop of the Psychology of Programming Interest Group (PPIG '08)*, Lancaster, England, September 2008, www.ppig.org. [Online]. Available: http://www.ppig.org/workshops/20th-programme.html