Ronda: A fine grained collaborative development environment

Fernando Olivero, Michele Lanza, and Marco D'ambros

REVEAL @ Faculty of Informatics - University Of Lugano, Switzerland

Abstract. Programmers seldom work in isolation: Software development is a social human activity, which therefore requires collaboration among the involved programmers. We argue that the main vehicles for programming—the integrated development environments (IDEs)—were designed without collaboration in mind. IDEs focus on a single viewpoint of the system, hence team members are aware of system changes only after the code is committed to the versioning system, which delays discussions that would otherwise prevent conflicts.

We propose a novel IDE, named Ronda, devised from the ground up, to fully embrace the collaborative nature of programming. Such an IDE allows a team of developers to take part in development sessions, both individually and in a group, within the same environment, promoting awareness and coordination, by tracking, broadcasting and visualizing fine-grained changes to the system.

1 Introduction

The art of crafting programs to solve problems is rarely accomplished by a single human working in solitude. Early psychological theories of programming [13] acknowledged that the software development process is a social human task, and practitioners have observed that therein lies the main cause of project failures [6]. Nonetheless, the main vehicle for programming—the integrated development environment (IDE)—remains as it was conceived in the 1970s, focused on a single point of view of the system.

There have been attempts to provide better collaboration support in existing IDEs, such as Palantir [8] and Syde [3]. However, in these cases collaboration support is an afterthought stapled on top of existing environments that struggle to overcome their single-developer nature.

We present *Ronda*, an extension to *Gaucho* (a 2D IDE based on a canvas metaphor [7]). Ronda offers first-class support for collaborative development sessions. *Ronda* is a change-centric environment: It promotes awareness of fine-grained changes to the system under development. We describe the infrastructure we built to support team collaboration within the scope of sessions, and we illustrate the features and usage of *Ronda*.

2 The Vision: Object-Focused Collaboration

In this section we motivate the need for *Ronda*, a novel integrated development environment (IDE), devised from the ground up to support collaborative sessions within the environment, and built around an alternate user interface metaphor.

2.1 Shared Development Sessions

The interaction amongst people assembled in groups can be categorized into focused and unfocused gatherings. In the former, several participants get together for a clearly stated purpose, while in the latter each of them might have different goals during the rendezvous [2].

Most of the collaborative development environments developed to date, provide some form of unfocused gathering by means of a shared editable view of the system, or by visualizing the modifications made by other developers. For example, in the 1990s researchers at Sun Microsystems devised Kansas, a 2D space for real-time collaboration, including a shared large flat space which hosted directly-manipulable representations of the objects [11], see Figure 1. In Kansas, any change to the system is immediately displayed to every developer, but the system lacks the concept of a session to guide developers into which modifications must be performed.

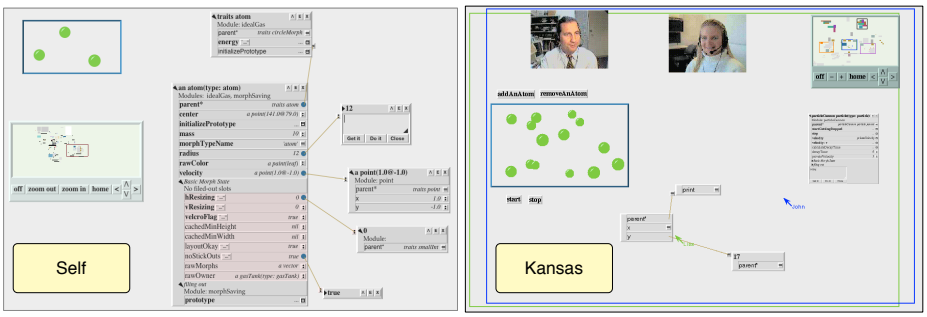

Fig. 1. Self and Kansas, collaboration within an object-focused IDE

More recent examples are Syde and Jazz. Syde is a set of plugins that augment the Eclipse IDE with awareness of fine-grained changes to the system [3], see Figure 2. Jazz [1], is a collaboration platform that can integrate with the IDE to enable task tracking capabilities and source control.

We argue that such environments are missing a fundamental piece of the puzzle: A first-class presence of shared development sessions, with clearly defined boundaries, objectives, and outcome. In *Ronda*, development sessions are first class objects, which provide a context for accomplishing tasks using the IDE.

[1] `https://jazz.net/`

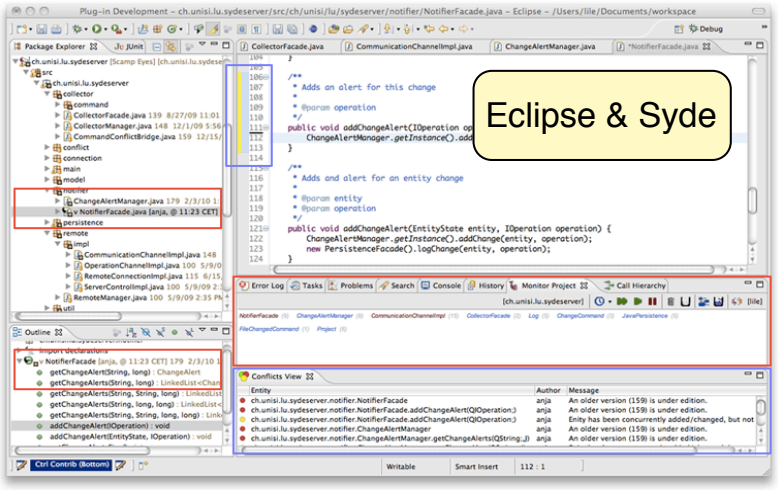

Fig. 2. Syde: collaboration support within the Eclipse IDE

2.2 The Object-focused metaphor

We designed *Ronda* around the object-focused metaphor, a term coined by the creators of the Self programming language [10], depicted in Figure 1. The interfaces built around this metaphor minimize the presence of tools to give prominence to high-level views of objects, that provide means to fully manipulate them via direct manipulation.

We make use of an alternate user interface metaphor, to avoid the many problems that traditional IDEs struggle to overcome, mostly related to the allocation of real estate resources [1]. To escape from the bento box philosophy that confines the IDE within a single window with sub-panes [1]. The bento-box model forces Syde's plugins to compete for a portion of real estate with the traditional tools of Eclipse, see Figure 2. A 2D open-space IDE in the vein of Self, on the other hand, easily accommodates collaborative aspects due to its libertarian usage of screen space, and a more concrete representation of the objects in the interface.

The use of direct manipulation enables a more focused display of visual cues to denote changes to the system, given the continuous representation of the objects of interest characteristically of such interfaces [4]. For instance, in Syde, which is built on top of a traditional IDE, the notification of changes is detached from the actual portion of the source code describing the changed entities, resulting in a so called "ping pong" interface [5].

We want to provide a framework that enables the creation, announcement, development, and tracking of sessions, enabling team members to engage in *focused gatherings* within an object-focused IDE. This is the main principle of *Ronda*, presented next.

3 Ronda: An Object-Focused Collaborative Environment

Ronda is an object-oriented development environment that enables a group of developers to remotely collaborate to accomplish tasks within the scope of sessions. *Ronda* is built on top of an object-focused IDE named *Gaucho* [7]. *Gaucho* minimizes the presence of tools in favor of shapes, directly manipulable views of objects, that populate a 2D surface named the Pampas. Figure 3 portrays a Pampas including shapes that represent classes, tests, methods and changes.

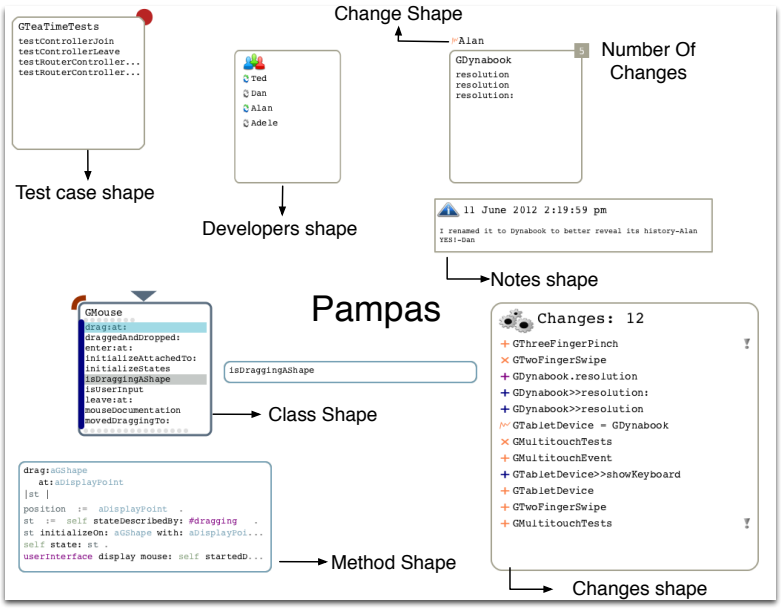

Fig. 3. A *Ronda* session: the Pampas including several Shapes

Awareness of Fine-Grained Changes. *Ronda* is a change-centric development environment which includes several shapes which provide the means to fully manipulate the represented objects. For example, developers can create, rename, remove, and add methods and variables to classes by solely interacting with a *Class Shape*. We track fine-grained changes within the IDE, to provide real-time awareness support, hence every (minor) change to the system is immediately broadcasted to the other participants to attain a level of awareness, which is simply not possible in single-person mainstream IDEs, where—as previous research pointed out [12]—often developers engage in a blind race to commit first and avoid the merge of conflicting changes. However, we believe the use of the object-focused metaphor, as opposed traditional bento-box interfaces, provides better support for visualizing those changes and revealing conflicts, mostly

because of the stronger presence of objects within the interface (*i.e.,* high-level views of objects vs enriched source code editors). For instance, Figure 3 depicts a class shape that was renamed by another participant. The shape presents visual cues for quickly understanding the nature of the change and who produced it.

Avoiding Conflicts. Even though developers in the same team seldom work on the same objects at the same time, conflicts may occur because they are working to solve the same task. In *Ronda*, we avoid conflicts by broadcasting any shape edition which might lead to a system change, thus developers have a consistent view of the shapes—the objects—currently under manipulation: A session tracks both changes and editions. Editions are any manipulation which might result in a system change. For instance, opening a class rename shape or a method add shape, and receiving input from the developer.

Communication. Developers can attach notes to ask, inform or hint about the system. These notes enable both *in-place* documentation and conversations within the IDE.

Change Authoring and Trust Levels. In *Ronda*, we distinguish between trusted and untrusted developers. The former produce trusted changes, whereas modifications of the latter result in untrusted changes, which are visualized differently, and might be discarded by more knowledgeable trusted developers. The trust levels are granted by the owner when creating the session, by enumerating the trusted developers and specifying wether untrusted developers can join.

Figure 4 portrays two pampas, from different developers, with several past changes and ongoing editions.

Fig. 4. Generation and awareness of changes in the interface

Fig. 5. Ronda: The Initial Display

Shared Development Sessions. Figure 5 depicts two different initial displays presented when *Ronda* developers open the environment, and are presented with the available sessions they can join. The sessions have a named *task* that describes the purpose of the gathering, an *owner* who is responsible for closing and committing its outcome, a list of *developers* who can participate, and a list of those who are logged in. When a developer joins a session, *Ronda* synchronizes to an updated state of the ongoing session, by downloading and installing a snapshot containing the system under development and all the changes performed so far, and then opens the session in the interface.

4 The Infrastructure

In *Ronda*, we make use of a simplified version of TeaTime [9], a decentralized distributed framework that relies on replication of computation instead of data. TeaTime revolves around the concept of an Island, which is a secure container of objects. An Island is an abstract concept, with no inherent location. Islands are *projected* onto numerous concrete replicas, located in hosts of the network.

Consistency amongst replicas is maintained by broadcasting any message that alters the state of the Island, via controllers connected to the same router, following a two-phase commit protocol. TeaTime messages originate in a host, then travel from the controller to the router, and finally get dispatched to all the connected controllers, including the original one.

The state-changing messages are generated in response to events performed by the developer, when manipulating the objects of the Island via their graphical counterpart within the user interface. The messages sent by the router are ordered by a sequence number and a timestamp, to preserve the order of execution of all received messages in each replica. Thus, the replicas deterministically evolve over time, because each replica is an exact copy of the Island *i.e.,* they hold the same objects, and send the same ordered stream of messages.

4.1 Customizing Tea Time for *Ronda*

In *Ronda*, a TeaTime Island includes the shared development sessions, replicated in the IDE of all collaborating developers. A development session includes the system under construction, a list of trusted developers, and all the past changes.

Figure 6 depicts the core architecture in *Ronda*, consisting of one or more developers running a *Ronda* IDE (Alan, Ted, Dan and Adele), composed by a controller connected to the Island's router via a TCP Socket, a replica of the ongoing Session, and an augmented *Gaucho* IDE.

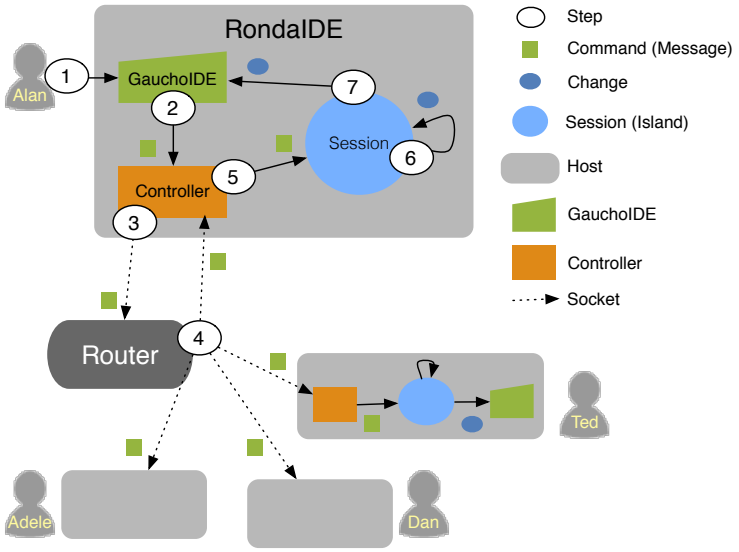

Fig. 6. The infrastructure for collaboration in Ronda

When developers manipulate shapes in the Gaucho IDE ①, a Tea Time message in the form of a UI command is generated ②, that either represents a fine grained system change or a UI element edition, like a class rename or a class shape name edition. The command is sent to the controller and forwarded to the island's router ③. The router broadcasts the command to all the connected controllers ④. Afterwards, upon reception, the command is executed producing the same result in every replica ⑤, which results in a change ⑥ that alters the ongoing session, and is presented in the user interface of the IDE ⑦.

4.2 Implementation Details

Ronda is written in Smalltalk, a highly dynamic and fully reflective language. It is implemented on top of Pharo[2], a modern open source IDE for Smalltalk.

[2] `http://pharo-project.org`

Ronda augments the *Gaucho* IDE, to implement the infrastructure described throughout this section. *Ronda* is free, open source, and released under the MIT license. It can be downloaded from the *Gaucho* web site located at `gaucho.inf.usi.ch`. A screencast, demonstrating the main features of *Ronda*, is available at `http://vimeo.com/17443946`.

5 Conclusions

We have presented *Ronda*, a novel IDE designed to support collaboration by means of shared development sessions, and a change-centric environment which tracks and visualizes fine-grained changes to the system under construction.

Acknowledgements. Olivero is supported by the Swiss Science foundation through the project "GSync" (SNF Project No. 129496).

References

1. DeLine, R., Rowan, K.: Code canvas: zooming towards better development environments. In: Proceedings of ICSE 2010 (32nd ACM/IEEE International Conference on Software Engineering) - Volume 2. pp. 207–210. ACM (2010)
2. Goffman, E., Wootton, A.J.: Exploring the interaction order. Polity Press (1988)
3. Hattori, L., Lanza, M.: Syde: A tool for collaborative software development. In: Proceedings of ICSE 2010 (32nd ACM/IEEE International Conference on Software Engineering). pp. 235–238 (2010)
4. Hutchins, E., Hollan, J., Norman, D.: Direct manipulation interfaces. Human-Computer Interaction 1, 311–338 (1985)
5. Lieberman, H., Fry, C.: Bridging the gulf between code and behavior in programming. In: CHI. pp. 480–486. ACM/Addison-Wesley (1995)
6. Marco, T.D.: Peopleware - Productive Projects and Teams. Dorset House (1999)
7. Olivero, F., Lanza, M., D'Ambros, M., Robbes, R.: Enabling program comprehension through a visual object-focused development environment. In: Proceedings of VL/HCC '11 (IEEE Symposium on Visual Languages and Human-Centric Computing). pp. 127–134 (2011)
8. Sarma, A.: Palantir: enhancing configuration management systems with workspace awareness to detect and resolve emerging conflicts. Ph.D. thesis, CalState University (2008)
9. Smith, D.A., Kay, A., Raab, A., Reed, D.P.: Croquet - a collaboration system architecture. IEEE Computer Society (2003)
10. Smith, R.B., Maloney, J., Ungar., D.: The self-4.0 user interface: Manifesting a system-wide vision of concreteness, uniformity, and flexibility. In: OOPSLA '95 Conference Proceedings. pp. 47–60 (1995)
11. Smith, R.B., Wolczko, M., Ungar, D.: From kansas to oz: collaborative debugging when a shared world breaks. Commun. ACM 40 (1997)
12. de Souza, C.R.B., Redmiles, D., Dourish, P.: Breaking the code, moving between private and public work in collaborative software development. In: Proceedings of GROUP 2003 (International ACM SIGGROUP Conference on Supporting Group Work). pp. 105–114. ACM Press (2003)
13. Weinberg, G.: The Psychology of Computer Programming. Dorset House, silver anniversary edn. (1998)