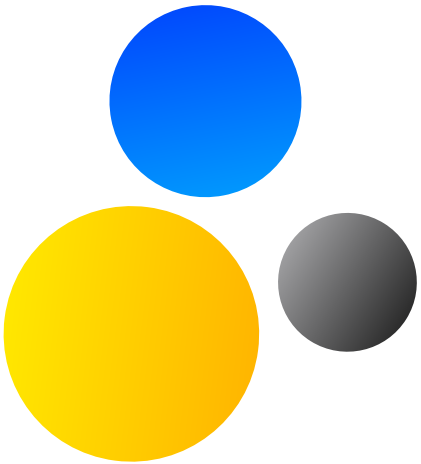
Freie Universität Berlin



Designing and Implementing Cross-IDE Compatibility for Saros

Master's Thesis at the Freie Universität Berlin

Tobias Bouschen
Freie Universität Berlin

February 12th, 2020

Outline

Outline

Saros

- Plugin for distributed pair programming (DPP)
  - Allows collaborative real-time editing of text files
  - Provides shared workspace awareness information
- Developed at the Freie Universität since 2006
- Available on GitHub: `https://github.com/saros-project/saros`

Outline

My Involvement With the Saros Project

- Worked on an initial alpha release of Saros for IntelliJ as part my bachelor's thesis in 2017 [1]
- Worked on Saros as a student research assistant since 2017
  - Mainly worked on Saros for IntelliJ
  - Worked on initial alpha release
  - Worked on subsequent alpha releases increasing feature set and stability

Outline

Short Saros History

- Initially created in 2006 as part of diploma thesis of R. Djemili [2]
- Initially only designed as an Eclipse plugin (later named *Saros/E*)
- Later on decided to add support for other IDEs
- Moved core logic of Saros to a separate component to simplify the process (*Saros Core*)
- Added Saros implementation for IntelliJ (*Saros/I*)

Outline

Problem Statement

- Many Saros APIs are too closely modeled after the Eclipse API
  - Makes implementation process for new IDEs slow and cumbersome
- Current Saros implementations are not compatible
  - Tight integration into the IDE systems lead to clashes where IDE concepts don’t match

Outline

Thesis Goal

- Analyze current structure of Saros
- Identify roadblocks on the way of supporting IDE cross-compatibility
- Adjust the Saros design (if necessary) to allow for IDE cross-compatibility
- Create a list of steps needed to reach this new design
- Implement the new Saros design following the list of steps

Outline

Milestones

- *Milestone 1* – A basic overview over the Saros architecture and possible roadblocks has been achieved
- *Milestone 2* – The list of steps to take on the way to implementing IDE cross-compatibility has been compiled
- *Milestone 3* – The Saros filesystem interface has been revamped
- *Milestone 4* – Basic IDE cross-compatibility is implemented

Outline

Related Work

- Saros in general
  - *Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierung verteilter Paarprogrammierung* (R. Djemili, 2006) [2]
  - *Industrially Usable Distributed Pair Programming* (J. Schenk, 2018) [9]
- Saros Core extraction
  - *Refaktorisierung des Eclipse-Plugins Saros für die Portierung auf andere IDEs* (A. Lasarzik, 2015) [4]
  - *Entwicklung und Evaluation eines unabhängigen Sitzungsservers für das Saros-Projekt* (D. Washington, 2016) [11]
  - *Verbesserung und Erweiterung der Core-Bestandteile von Saros* (D. Sungaila, 2016) [10]
- Saros filesystem rework
  - *Saros: Refactoring the Filesystem* (O. Hanßen, 2019) [3]

Outline

Preparation

- Read up on Saros Literature [5, 6, 7, 8, 9]
- Skimmed Saros codebase looking for potential issues
- Discussed general Saros design ideas and potential technical issues with other Saros developers

Outline

- Current resource sharing logic is to closely coupled with the IDE project/module model
  - Such coupling is necessary to provide the current functionality of creating a module/project as part of the session start
- Project model of Eclipse and module model of IntelliJ are very different
  - Transforming resources from one model to the other is non-trivial
  - A direct mapping is not always possible, even for "normal" project (e.g. Saros)

Solution approach:

- Introduce a new resource handling based on reference points (described in more detail in the thesis of O. Hanßen [3])
- Drop support for project/module creation as part of session start

Outline

- IntelliJ normalizes all editor contents to use UNIX line ending (’\n’)
- Eclipse uses the file line endings in the editor contents

Solution approach:

- Introduce a normalization component that normalizes all line endings in content passed to the Core
- Adjust selection offset handling to use line-based coordinated

Outline

- IntelliJ does not open editors for files without a known file type association
- Opening binary files could lead to data loss due to automatic content normalization

Solution approach:

- Implement logic determining which file is non-binary
- Open unknown file in plain text editor in IntelliJ if it is determined to be non-binary
- Check whether set of non-binary resources matches on session start

Outline

- Projects/modules may contain resources that should not be shared
  - Automatically generated artifacts
  - User/system-specific configurations
- Current approach uses IDE logic to determine such files
- IDEs might not agree on what should not be shared
- IDE logic might not cover all resources that are not supposed to be shared

Solution approach:

- Host determines which resources are non supposed to be shared
- Offer other ways of determining what not to share
  - A ’.sarosignore’ file
  - Integration with git (i.e. the ’.gitignore’ configuration)

Outline

Steps on the Way to Providing IDE Cross-Compatibility I

Freie Universität Berlin

1. Preparation for the Filesystem Handling Refactoring
   - 1.1 Removal of the Partial Sharing Logic
   - 1.2 Extraction of Core Components That Need to Resolve Project Resources
2. Refactoring the Filesystem Handling to Work With Reference Points *(Roadblock 1)*
3. Cleaning Up Old Project/Module Specific Logic
   - 3.1 Remove remnants of old project/module specific logic
   - 3.2 Adjust UI to match new resource sharing system
4. Implementing a Proper Handling for Binary Files *(Roadblock 3)*
   - 4.1 Adjusting the ’IFile’ Interface
   - 4.2 Provide IDE-specific implementations for the binary check
   - 4.3 Implement an initial check whether the sets of non-binary resources match during session negotiation
   - 4.4 Adjust IntelliJ Handling for Opening Non-Binary Files

5. Implementing a Bridge Component Handling Content Normalization *(Roadblock 2)*
   - 5.1 Create Normalization Bridge in Core
   - 5.2 Adjust Initial Content Sharing Logic to Use File Content for Non-Binary Files
   - 5.3 Adjust Handling of Initial Content During File Creation
   - 5.4 Adjust Handling of Recovery Content for Non-Binary Files

6. Implementing a New System to Determine What Not to Share *(Roadblock 4)*
   - 6.1 Introduce a Framework to Set Which Resources Should Not Be Shared
   - 6.2 Adjust Existing Logic to Offer Excluded/Derived Resource Settings for New Framework
   - 6.3 Introduce a Base Implementation Using a ’.sarosignore’ file
   - 6.4 Remove the Old IDE-Dependent Logic
   - 6.5 *(Optional)* Offer ’.gitignore’ Integration

7. Handling Miscellaneous Issues
   - 7.1 Fix BOM Handling
   - 7.2 Implement Unified Handling of Binary Editors

References I

[1] Tobias Bouschen.
Bestandsaufnahme und Arbeit an einer Alpha-Version des Saros-Plugins für die IntelliJ-Plattform.
Master's thesis, Freie Universität Berlin, Inst. für Informatik, 2017.

[2] Raid Djemili.
Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierung verteilter Paarprogrammierung.
Diploma thesis, Freie Universität Berlin, Inst. für Informatik, 2006.

[3] Oliver Hanßen.
Saros: Refactoring the Filesystem.
Master's thesis, Freie Universität Berlin, Inst. für Informatik, 2019.

[4] Arndt Lasarzik.
Refaktorisierung des Eclipse-Plugins Saros für die Portierung auf andere IDEs.
Bachelor's thesis, Freie Universität Berlin, Inst. für Informatik, 2015.

References II

[5] Lutz Prechelt.
Some non-usage data for a distributed editor: The saros outreach.
In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '11, page 48, New York, NY, USA, 2011. Association for Computing Machinery.

[6] Lutz Prechelt and Karl Beecher.
Four generic issues for tools-as-plugins illustrated by the distributed editor saros.
In *Proceedings of the 1st Workshop on Developing Tools as Plug-Ins*, TOPI '11, page 9–11, New York, NY, USA, 2011. Association for Computing Machinery.

[7] Edna Rosen, Stephan Salinger, and Christopher Oezbek.
Project kick-off with distributed pair programming.
In *PPIG*, 2010.

References III

[8] Stephan Salinger, Christopher Oezbek, Karl Beecher, and Julia Schenk.
Saros: An eclipse plug-in for distributed party programming.
In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '10, page 48–55, New York, NY, USA, 2010. Association for Computing Machinery.

[9] Julia Schenk.
Industrially Usable Distributed Pair Programming.
PhD thesis, Freie Universität Berlin, Inst. für Informatik, 2018.

[10] David Sungaila.
Verbesserung und Erweiterung der Core-Bestandteile von Saros.
Bachelor's thesis, Freie Universität Berlin, Inst. für Informatik, 2016.

[11] Denis Washington.
Entwicklung und Evaluation eines unabhängigen Sitzungsservers für das Saros-Projekt.
Bachelor's thesis, Freie Universität Berlin, Inst. für Informatik, 2016.

Freie Universität Berlin

Thank you for listening!

For more information, you can have a look at my thesis page:
`http://www.inf.fu-berlin.de/w/SE/ThesisSarosCrossIDE`